XLNet

Zili Wang & Canjia Li

背景

- one-hot, word2vec, fasttext, lstm
- ELMO:

  通过用**RNN**训练双向语言模型得到单词的Word Embedding
- GPT

  以语言模型作为目标任务，但是采用的是**单向**的语言模型(Transformer)
- Bert的局限性
  - 独立性假设：输入的[MASK]之间是独立的
  - 输入噪音：依赖[MASK]标记识别需要预测的位置。输入输出不一致。

Auto-regressive vs Auto-decoding

- Auto-regressive: 上文无法看到下文

  代表模型：ELMO，GPT

- Auto-decoding：重建出输入

  代表模型：BERT

- **Word2vec该属于哪一种？**

自回归模型（AR, AutoRegressive）： ELMo, GPT

["我1", "今天2", "很3", “开心4”, “< 逗号>5”, “因为6”, “我7”, “中8”, “了9”, “彩票10”]

(["我"] → "今天"), (["我", "今天"]→ "很"), (["我", "今天", "很"] → “开心”), …

自编码模型(AE, AutoEncoder)：BERT

["我1", "今天2", “[mask]”, “开心4”, “< 逗号>5”, “因为6”, “[mask]”, “中8”, “了9”, “彩票10”]

Auto-regressive

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

Auto-decoding

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

Input noise for downstream task

Based on an independence assumption

Context dependency

A natural question to ask is whether there exists a pretraining objective that brings the advantages of both while **avoiding their weaknesses**

Permutation Language Model

Objective: **retains the benefits of AR models** & **allows models to capture bidirectional contexts**

For a sequence x of length T, there are T! different orders to perform a valid autoregressive factorization.

let $\mathcal{Z}_T$ be the set of all possible permutations of the length-T index sequence

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

Permutation Language Model

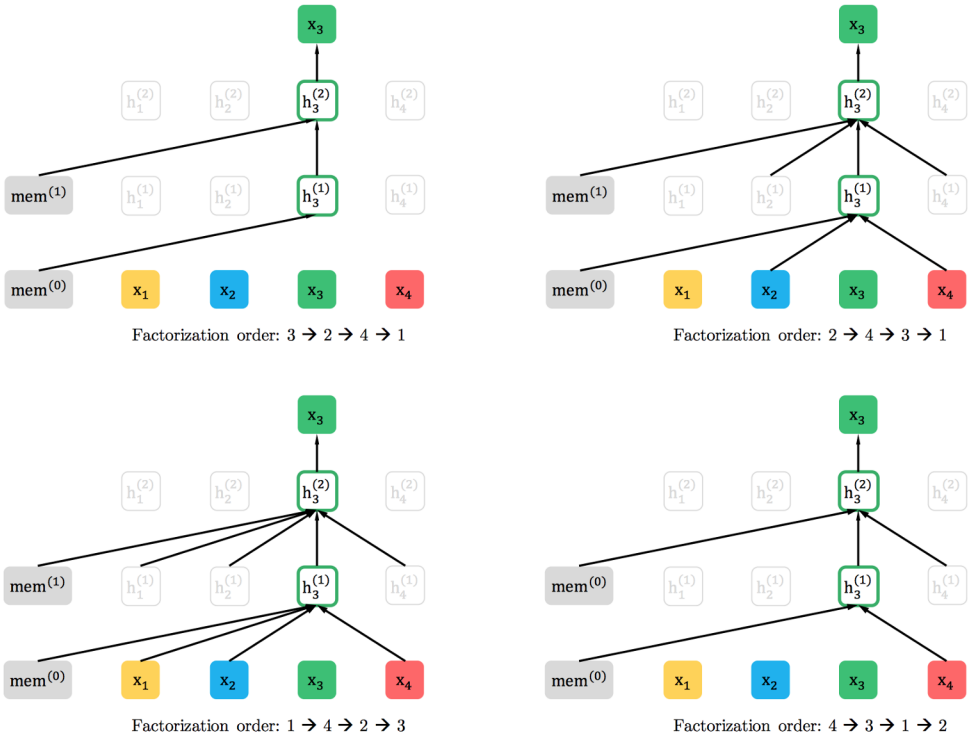
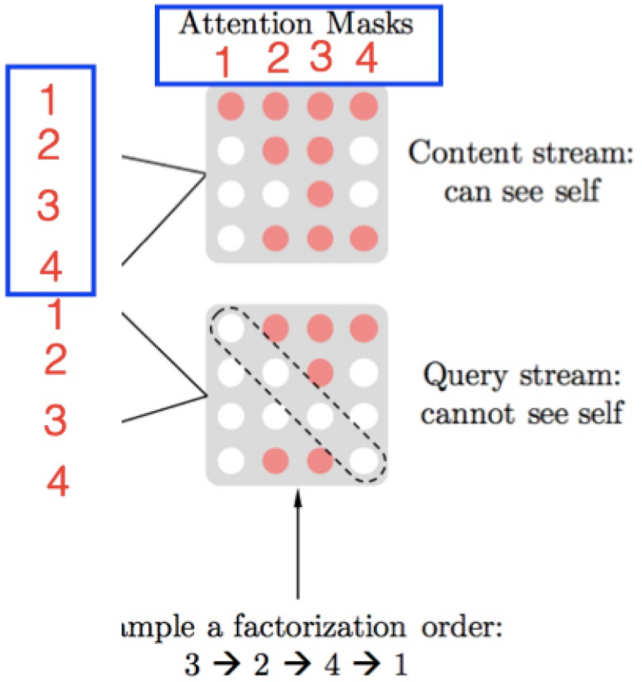


Figure 1: Illustration of the permutation language modeling objective for predicting x_3 given the same input sequence **x** but with different factorization orders.

The example of predicting the token x3 **given the same input sequence x** but **under different factorization orders**

Attention Mask

Attention Mask



要记住：原始的输入顺序是1-2-3-4,
而这个是1-2-3-4随机排列后的结果3-2-4-1.
2能看到上文3，4能看到3和2，1能看到3，2和4，通过掩码实现

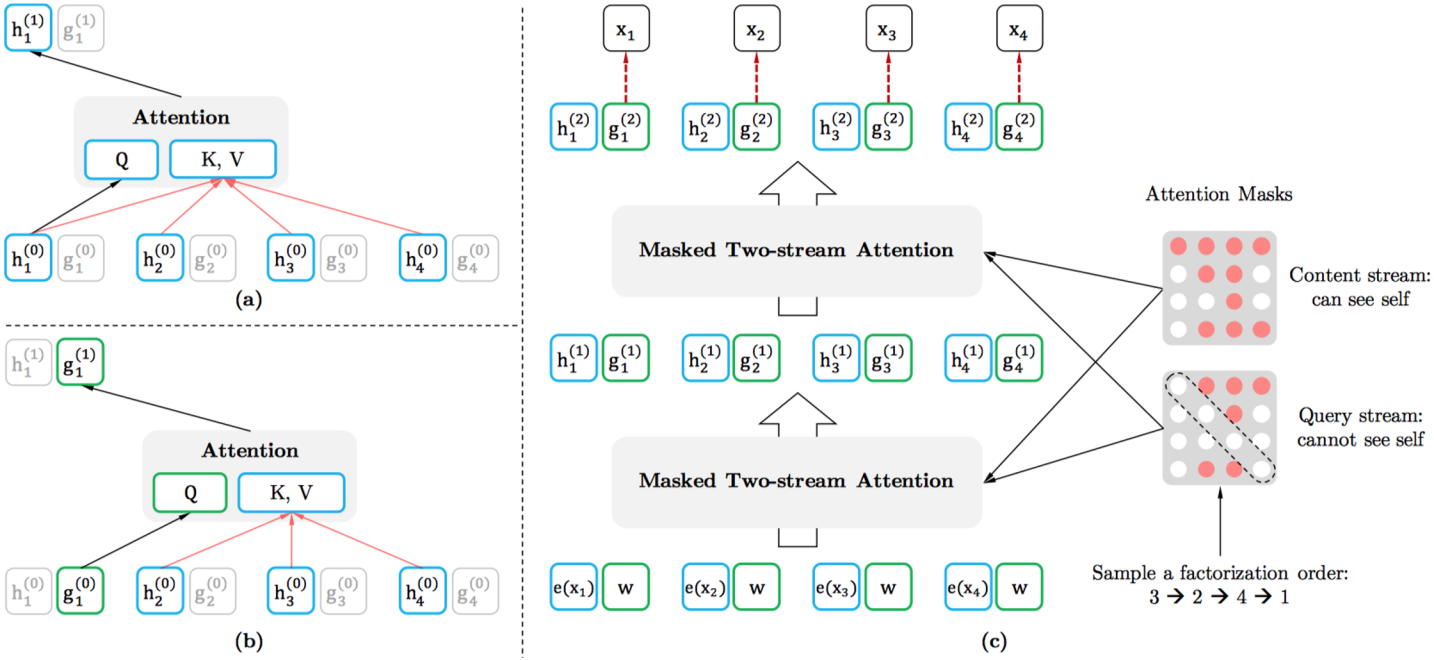
Two-Stream Self-Attention

$$\mathbf{z}_{<t}^{(1)} = \mathbf{z}_{<t}^{(2)} = \mathbf{z}_{<t} \quad \text{but} \quad z_t^{(1)} = i \neq j = z_t^{(2)}.$$

$$\underbrace{p_\theta(X_i = x \mid \mathbf{x}_{\mathbf{z}_{<t}})}_{z_t^{(1)}=i,\, \mathbf{z}_{<t}^{(1)}=\mathbf{z}_{<t}} = \underbrace{p_\theta(X_j = x \mid \mathbf{x}_{\mathbf{z}_{<t}})}_{z_t^{(1)}=j,\, \mathbf{z}_{<t}^{(2)}=\mathbf{z}_{<t}} = \frac{\exp\left(e(x)^\top h(\mathbf{x}_{\mathbf{z}_{<t}})\right)}{\sum_{x'} \exp\left(e(x')^\top h(\mathbf{x}_{\mathbf{z}_{<t}})\right)}.$$

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

Two-Stream Self-Attention



$$g_{z_t}^{(m)} \leftarrow \text{Attention}(\mathrm{Q} = g_{z_t}^{(m-1)}, \mathrm{KV} = \mathbf{h}_{\mathbf{z}_{<t}}^{(m-1)}; \theta), \quad \text{(query stream: use } z_t \text{ but cannot see } x_{z_t})$$

$$h_{z_t}^{(m)} \leftarrow \text{Attention}(\mathrm{Q} = h_{z_t}^{(m-1)}, \mathrm{KV} = \mathbf{h}_{\mathbf{z}_{\leq t}}^{(m-1)}; \theta), \quad \text{(content stream: use both } z_t \text{ and } x_{z_t}).$$

Two-Stream Self-Attention

Initial represetation:

$$\forall t = 1, \ldots, T: \quad h_t = e(x_t) \quad \text{and} \quad g_t = w$$

Cached layer-m content represetation (memory) from previous segment: $\tilde{\mathbf{h}}^{(m)}$

For the Transformer-XL layer $m = 1, \cdots, M$, attention with relative positional encoding and position-wise feed-forward are consecutively employed to update the represetntations:

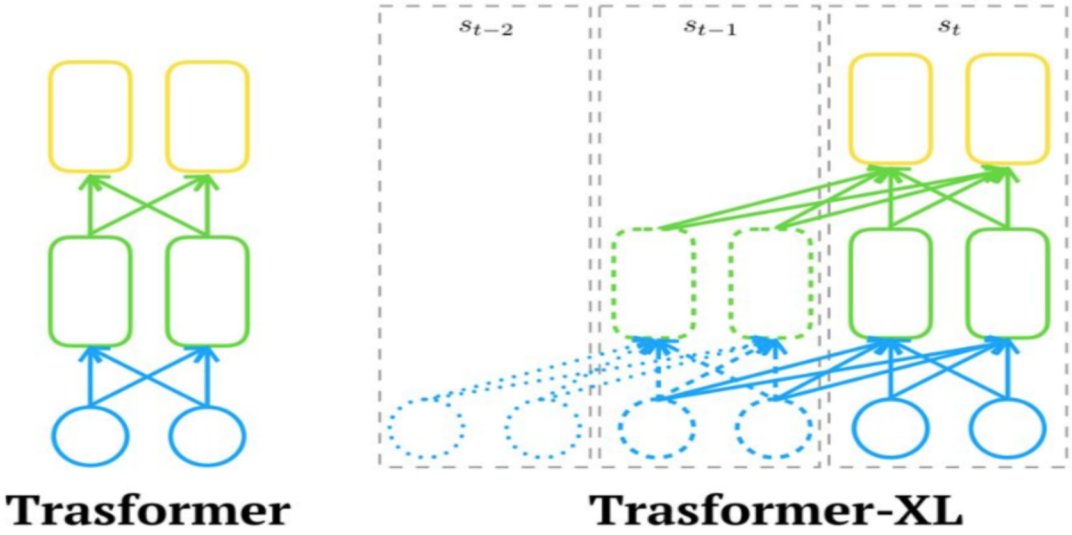
$$\begin{aligned}
\forall t = 1, \ldots, T: \quad \hat{h}_{z_t}^{(m)} &= \text{LayerNorm}\Big(h_{z_t}^{(m-1)} + \text{RelAttn}\Big(h_{z_t}^{(m-1)}, \Big[\tilde{\mathbf{h}}^{(m-1)}, \mathbf{h}_{\mathbf{z}_{\le t}}^{(m-1)}\Big]\Big)\Big) \\
h_{z_t}^{(m)} &= \text{LayerNorm}\Big(\hat{h}_{z_t}^{(m)} + \text{PosFF}\Big(\hat{h}_{z_t}^{(m)}\Big)\Big) \\
\hat{g}_{z_t}^{(m)} &= \text{LayerNorm}\Big(g_{z_t}^{(m-1)} + \text{RelAttn}\Big(g_{z_t}^{(m-1)}, \Big[\tilde{\mathbf{h}}^{(m-1)}, \mathbf{h}_{\mathbf{z}_{\le t}}^{(m-1)}\Big]\Big)\Big) \\
g_{z_t}^{(m)} &= \text{LayerNorm}\Big(\hat{g}_{z_t}^{(m)} + \text{PosFF}\Big(\hat{g}_{z_t}^{(m)}\Big)\Big)
\end{aligned}$$

Target-aware prediction distribution:

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

Transformer-XL

Query 是当前segment的上一层的输出
Key/Value是当前segment当前层的Key/Value拼接上一个segment的self-attention之后的输出



$$h_{z_t}^{(m)} \leftarrow \text{Attention}(\text{Q} = h_{z_t}^{(m-1)}, \text{KV} = \left[\tilde{\mathbf{h}}^{(m-1)}, \mathbf{h}_{\mathbf{z}\leq t}^{(m-1)}\right]; \theta)$$

相对位置编码

Previously, A = E + U

$$\mathbf{A}^{\text{abs}}_{i,j} = \underbrace{\mathbf{E}_{x_i}^\top \mathbf{W}_q^\top \mathbf{W}_k \mathbf{E}_{x_j}}_{(a)} + \underbrace{\mathbf{E}_{x_i}^\top \mathbf{W}_q^\top \mathbf{W}_k \mathbf{U}_j}_{(b)} + \underbrace{\mathbf{U}_i^\top \mathbf{W}_q^\top \mathbf{W}_k \mathbf{E}_{x_j}}_{(c)} + \underbrace{\mathbf{U}_i^\top \mathbf{W}_q^\top \mathbf{W}_k \mathbf{U}_j}_{(d)}.$$

Following the idea of only relying on relative positional information, we propose to reparameterize the four terms as follows

$$\mathbf{A}^{\text{rel}}_{i,j} = \underbrace{\mathbf{E}_{x_i}^\top \mathbf{W}_q^\top \mathbf{W}_{k,E} \mathbf{E}_{x_j}}_{(a)} + \underbrace{\mathbf{E}_{x_i}^\top \mathbf{W}_q^\top \mathbf{W}_{k,R} \mathbf{R}_{i-j}}_{(b)} + \underbrace{u^\top \mathbf{W}_{k,E} \mathbf{E}_{x_j}}_{(c)} + \underbrace{v^\top \mathbf{W}_{k,R} \mathbf{R}_{i-j}}_{(d)}.$$

- The first change we make is to replace all appearances of the absolute positional embedding $\mathbf{U}_j$ for computing key vectors in term (b) and (d) with its relative counterpart $\mathbf{R}_{i-j}$. This essentially reflects the prior that only the relative distance matters for where to attend. Note that $\mathbf{R}$ is a sinusoid encoding matrix (Vaswani et al., 2017) without learnable parameters.
- Secondly, we introduce a trainable parameter $u \in \mathbb{R}^d$ to replace the query $\mathbf{U}_i^\top \mathbf{W}_q^\top$ in term (c). In this case, since the query vector is the same for all query positions, it suggests that the attentive bias towards different words should remain the same regardless of the query position. With a similar reasoning, a trainable parameter $v \in \mathbb{R}^d$ is added to substitute $\mathbf{U}_i^\top \mathbf{W}_q^\top$ in term (d).

Transformer vs Transformer-XL

- Transformer
  - fixed length
  - 长文本被切分成不同的片段之后，片段之间是没有联系的
- Transformer-XL
  - recurrent structure, like RNN
    - Query不变，改进在于Key，Value，通过将上一个片段的表示拼接到当前片段的Key，Value表示。 与Transofrmer encoder-decoder的比较：encoder最后一层的输出是Key，Value表示，Query来自于decoder上一层的输出
  - absolute PE to relative PE（PE: Position Embedding)
    - Motivation: 上个片段跟下个片段是属于同一个文本，如果用相对位置编码，片段之间的位置信息就不一样了
    - 做法：做attention的时候使用相对位置编码。

XLNet的效果

阅读理解

SQuAD1.1	EM	F1	SQuAD2.0	EM	F1
Dev set results without data augmentation					
BERT [10]	84.1	90.9	BERT† [10]	78.98	81.77
XLNet	**88.95**	**94.52**	XLNet	**86.12**	**88.79**
Test set results on leaderboard, with data augmentation (as of June 19, 2019)					
Human [27]	82.30	91.22	BERT+N-Gram+Self-Training [10]	85.15	87.72
ATB	86.94	92.64	SG-Net	85.23	87.93
BERT* [10]	87.43	93.16	BERT+DAE+AoA	85.88	88.62
XLNet	**89.90**	**95.08**	XLNet	**86.35**	**89.13**

Table 2: A single model XLNet outperforms human and the best ensemble by 7.6 EM and 2.5 EM on SQuAD1.1. ∗ means ensembles, † marks our runs with the official code.

文本分类

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).

XLNet的效果

综合任务GLUE

Model	MNLI	QNLI	QQP	RTE	SST-2	MRPC	CoLA	STS-B	WNLI
Single-task single models on dev									
BERT [2]	86.6/-	92.3	91.3	70.4	93.2	88.0	60.6	90.0	-
XLNet	**89.8/-**	**93.9**	**91.8**	**83.8**	**95.6**	**89.2**	**63.6**	**91.8**	-
Single-task single models on test									
BERT [10]	86.7/85.9	91.1	89.3	70.1	94.9	89.3	60.5	87.6	65.1
Multi-task ensembles on test (from leaderboard as of June 19, 2019)									
Snorkel* [29]	87.6/87.2	93.9	89.9	80.9	96.2	91.5	63.8	90.1	65.1
ALICE*	88.2/87.9	95.7	**90.7**	83.5	95.2	92.6	**68.6**	91.1	80.8
MT-DNN* [18]	87.9/87.4	96.0	89.9	**86.3**	96.5	92.7	68.4	91.1	89.0
XLNet*	**90.2/89.7**†	**98.6**†	90.3†	**86.3**	**96.8**†	**93.0**	67.8	**91.6**	**90.4**

Table 4: Results on GLUE. ∗ indicates using ensembles, and † denotes single-task results in a multi-task row. All results are based on a 24-layer architecture with similar model sizes (aka BERT-Large). See the upper-most rows for direct comparison with BERT and the lower-most rows for comparison with state-of-the-art results on the public leaderboard.

信息检索

Model	NDCG@20	ERR@20
DRMM [12]	24.3	13.8
KNRM [8]	26.9	14.9
Conv [8]	28.7	18.1
BERT†	30.53	18.67
XLNet	**31.10**	**20.28**

BERT-FirstP	0.286
BERT-MaxP	0.293
BERT-SumP	0.289

Table 5: Comparison with state-of-the-art results on the test set of ClueWeb09-B, a document ranking task. † indicates our implementations.