XLNet

Zhilin Yang*, Zihang Dai*, Yiming Yang, Jaime Carbonell,
Ruslan Salakhutdinov, Quoc V. Le
(*: equal contribution)

Carnegie Mellon University
Google AI

Language Pretraining: Related Work

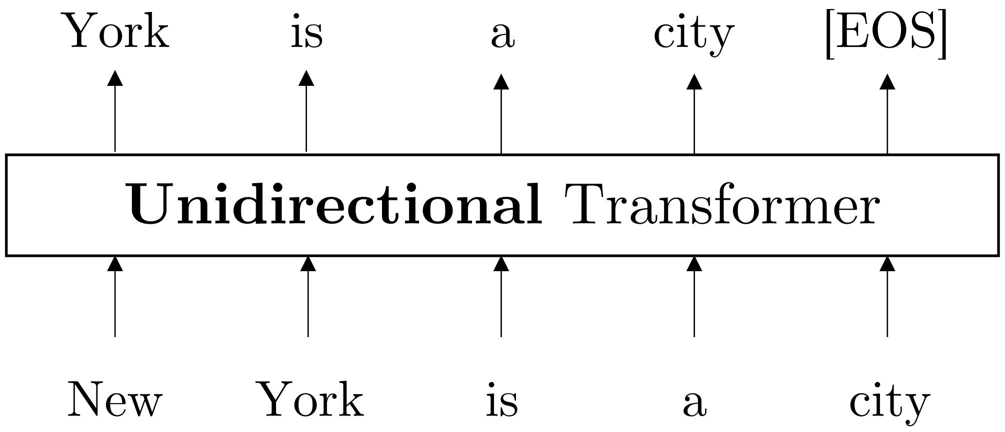
- RBMs (Salakhutdinov et al 2007), Autoencoders (Vincent et al 2008), Jigsaw (Noroozi and Favaro 2016), GANs (Donahue and Simonyan 2019)...

- word2vec (Mikolov et al 2013), GloVe (Pennington et al 2014)

- Semi-supervised sequence learning (Dai and Le 2015), ELMo (Peters et al 2017), CoVe (McCann et al 2017), GPT (Radford et al 2018), BERT (Devlin et al 2018)...

Two Notable Objectives for Language Pretraining

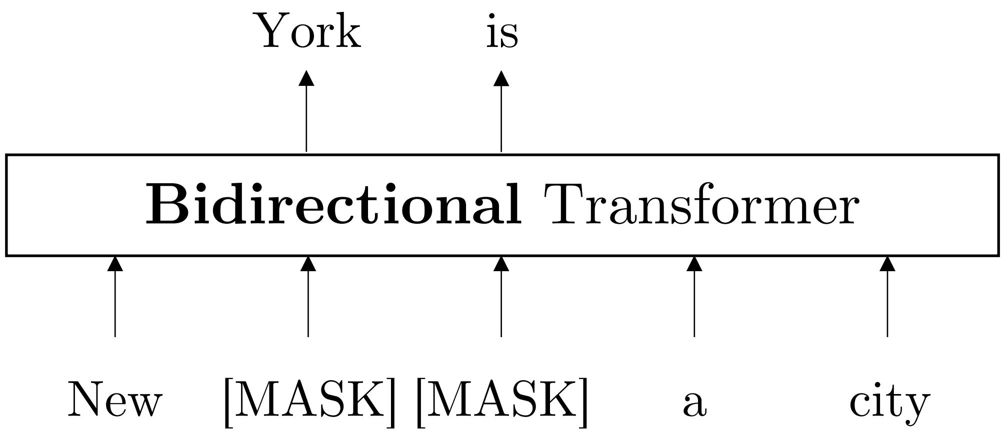
Auto-regressive Language Modeling



$$\log p(\mathbf{x}) = \sum_{t=1}^{T} \log p(x_t \mid \mathbf{x}_{<t})$$

- **Next-token prediction**

Denoising Auto-encoding (BERT)

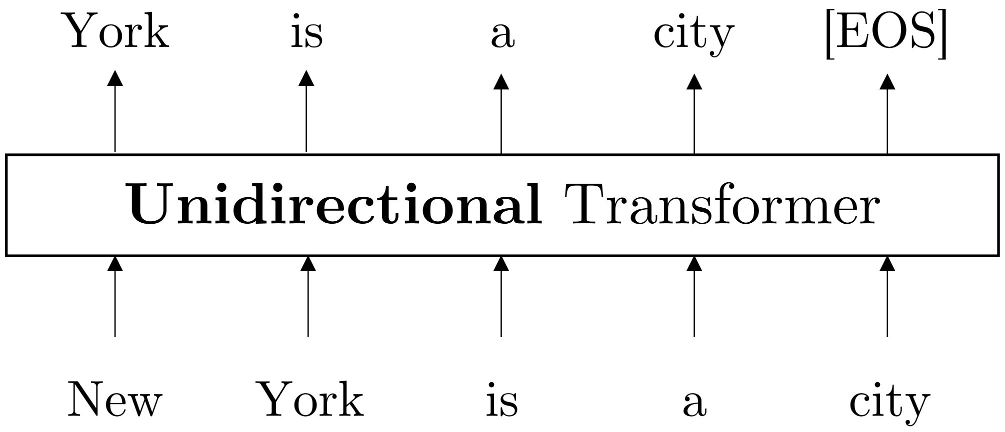


$$\log p(\bar{\mathbf{x}} \mid \hat{\mathbf{x}}) = \sum_{t=1}^{T} \text{mask}_t \log p(x_t \mid \hat{\mathbf{x}})$$

- **Reconstruct masked tokens**

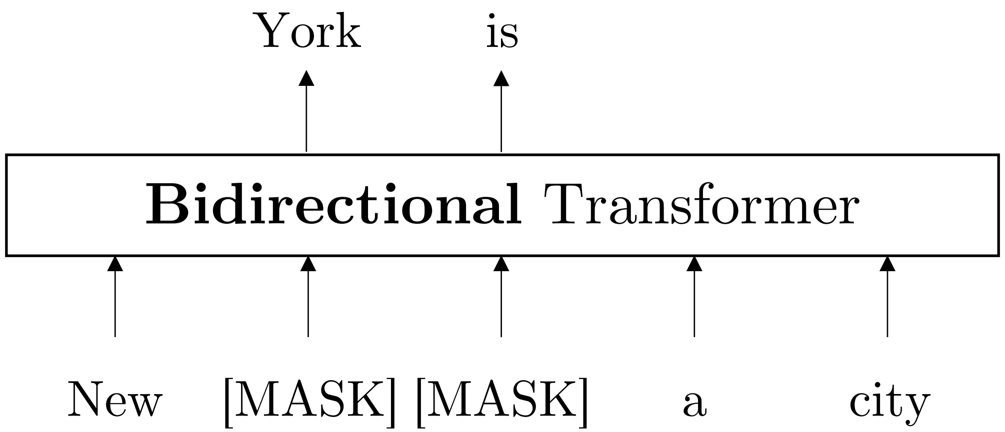
Two Notable Objectives for Language Pretraining

Auto-regressive Language Modeling



No **Bidirectional Context**

Denoising Auto-encoding (BERT)

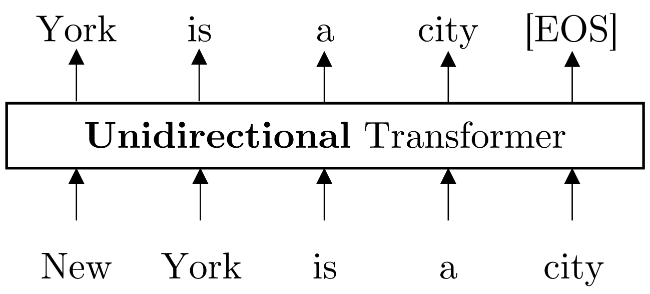


Independent Predictions

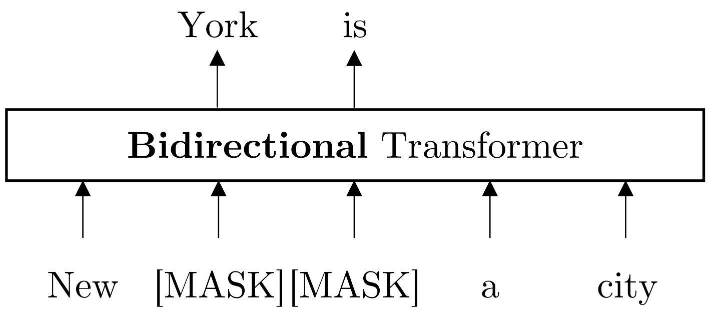
Artificial **Noise**: [MASK]

Two Notable Objectives for Language Pretraining

Auto-regressive Language Modeling



Denoising Auto-encoding (BERT)



Full Auto-regressive **Dependence** ⟺ **Independent** Predictions

Free from artificial **Noise** ⟺ Artificial **Noise**: [MASK]

No **Bidirectional Context** ⟺ Natural **Bidirectional Context**

Desire: Combine the Pros and Remove the Cons

☺ Full Auto-regressive **Dependence**

☺ Free from **Noise**

☺ Natural **Bidirectional Context**

Desire: Combine the Pros and Remove the Cons

XLNet

- An **auto-regressive** model that captures **bidirectional context**

Context Depends on the **Factorization Order**

- **Standard LM**: Left-to-right factorization 1 → 2 → 3 → 4

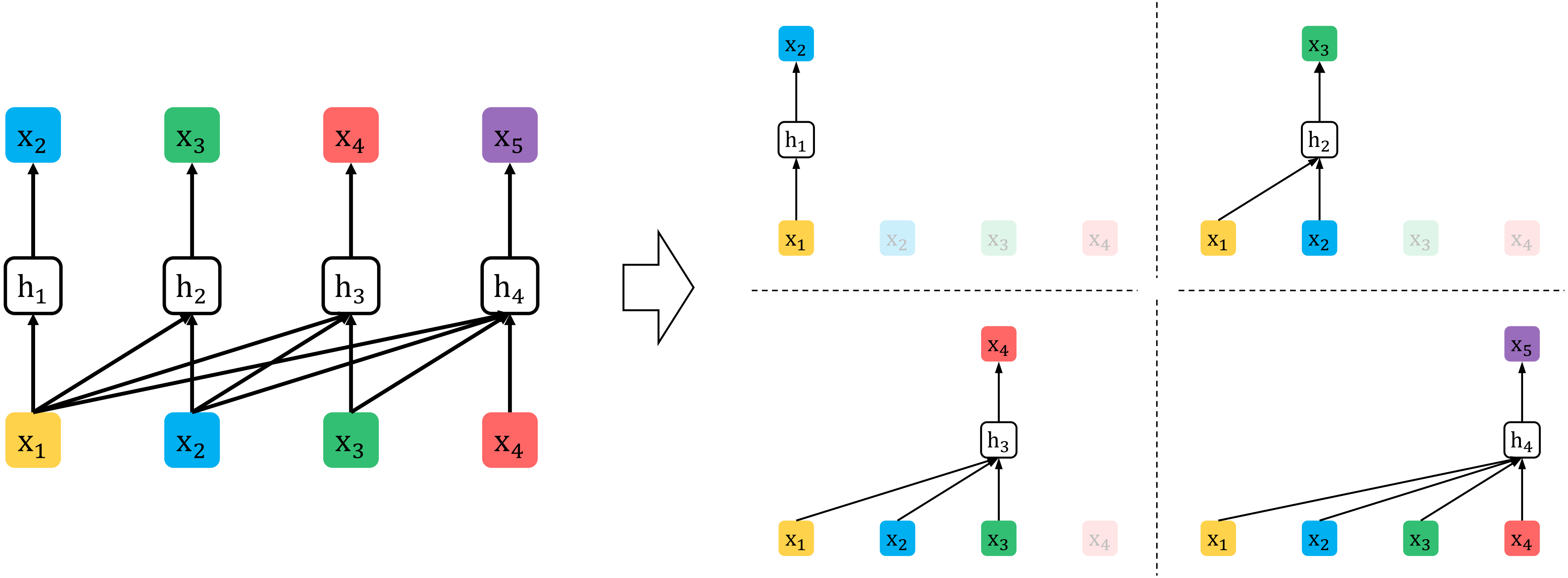
$$P(\mathbf{x}) = P(x_1)P(x_2 \mid \mathbf{x}_1)P(x_3 \mid \mathbf{x}_{1,2})P(x_4 \mid \mathbf{x}_{1,2,3}) \cdots$$

x_1 x_2 x_3 x_4

Context Depends on the **Factorization Order**

- **Standard LM**: Left-to-right factorization 1 → 2 → 3 → 4

$$P(\mathbf{x}) = P(x_1)P(x_2 \mid \mathbf{x}_1)P(x_3 \mid \mathbf{x}_{1,2})P(x_4 \mid \mathbf{x}_{1,2,3})\cdots$$



Context Depends on the **Factorization Order**

- Change the Factorization order to: 4 → 1 → 3 → 2

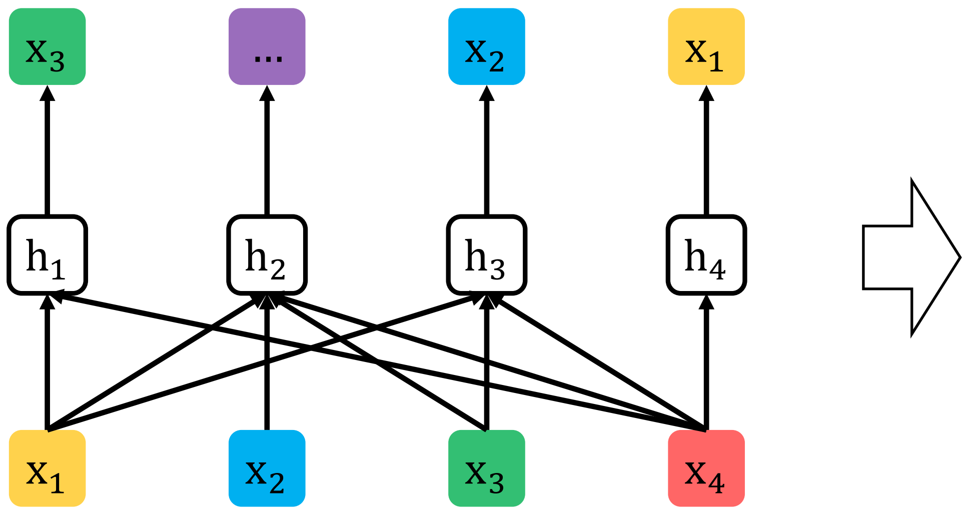
$$P(\mathbf{x}) = P(x_4)P(x_1 \mid \mathbf{x}_4)P(x_3 \mid \mathbf{x}_{1,4})P(x_2 \mid \mathbf{x}_{1,2,4})\cdots$$

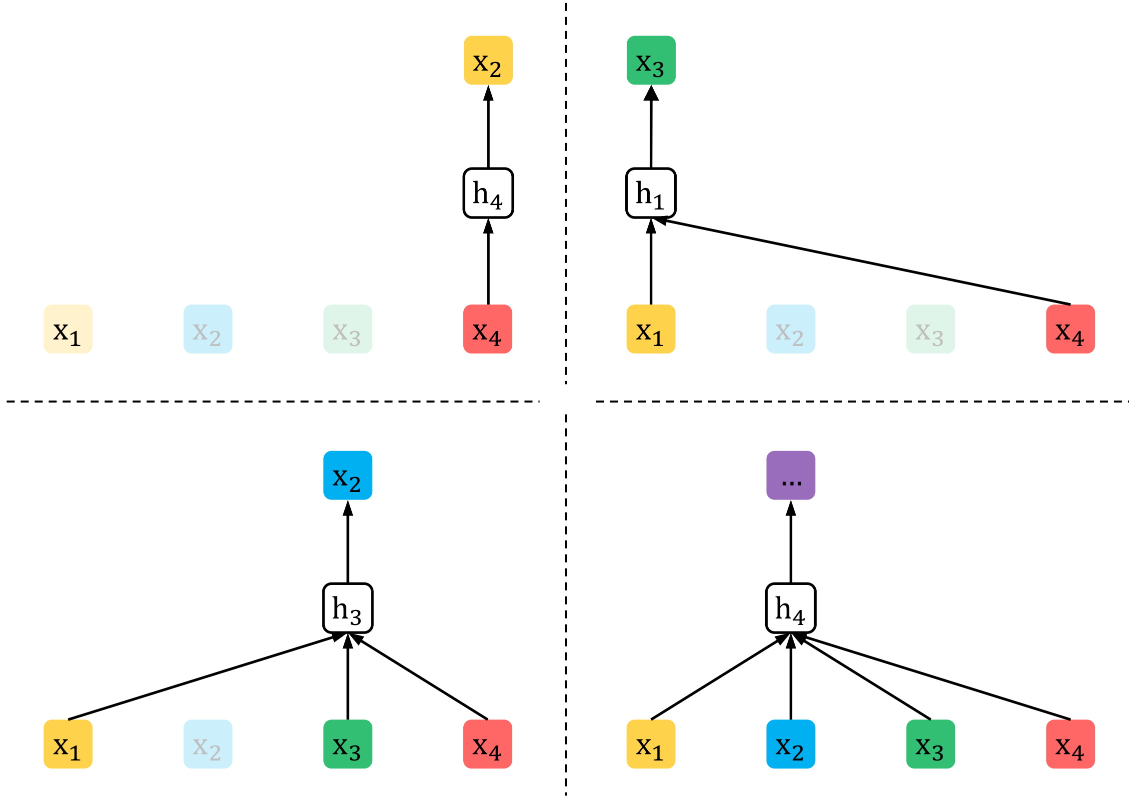
x_1 x_2 x_3 x_4

Context Depends on the **Factorization Order**

- Change the Factorization order to: 4 → 1 → 3 → 2

$$P(\mathbf{x}) = P(x_4)P(x_1 \mid \mathbf{x}_4)P(x_3 \mid \mathbf{x}_{1,4})P(x_2 \mid \mathbf{x}_{1,2,4}) \cdots$$

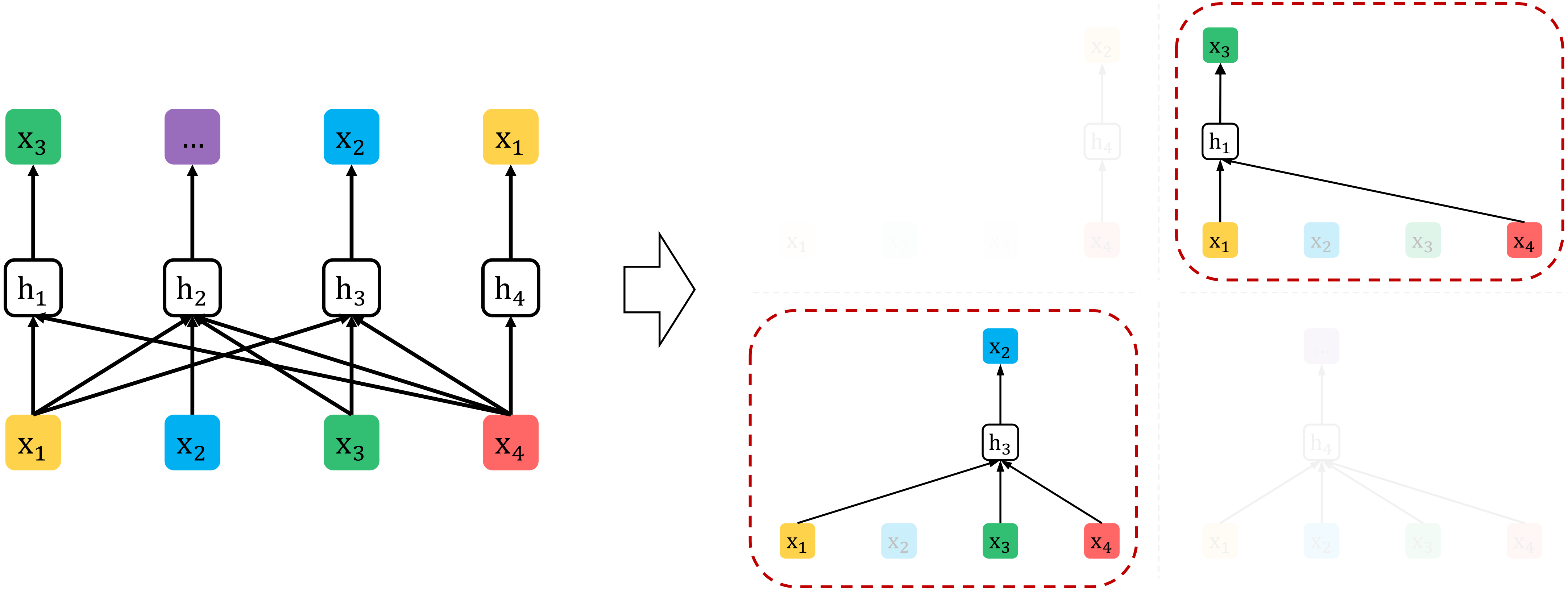




Context Depends on the **Factorization Order**

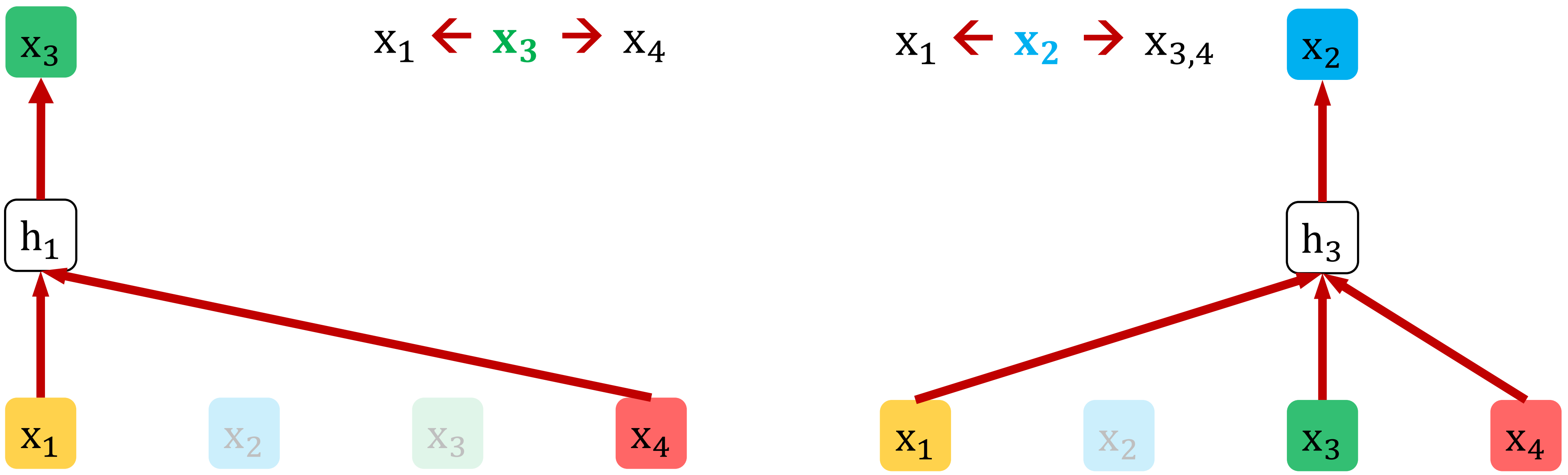
- Change the Factorization order to: 4 → 1 → 3 → 2

$$P(\mathbf{x}) = P(x_4)P(x_1 \mid \mathbf{x}_4)P(x_3 \mid \mathbf{x}_{1,4})P(x_2 \mid \mathbf{x}_{1,2,4})\cdots$$



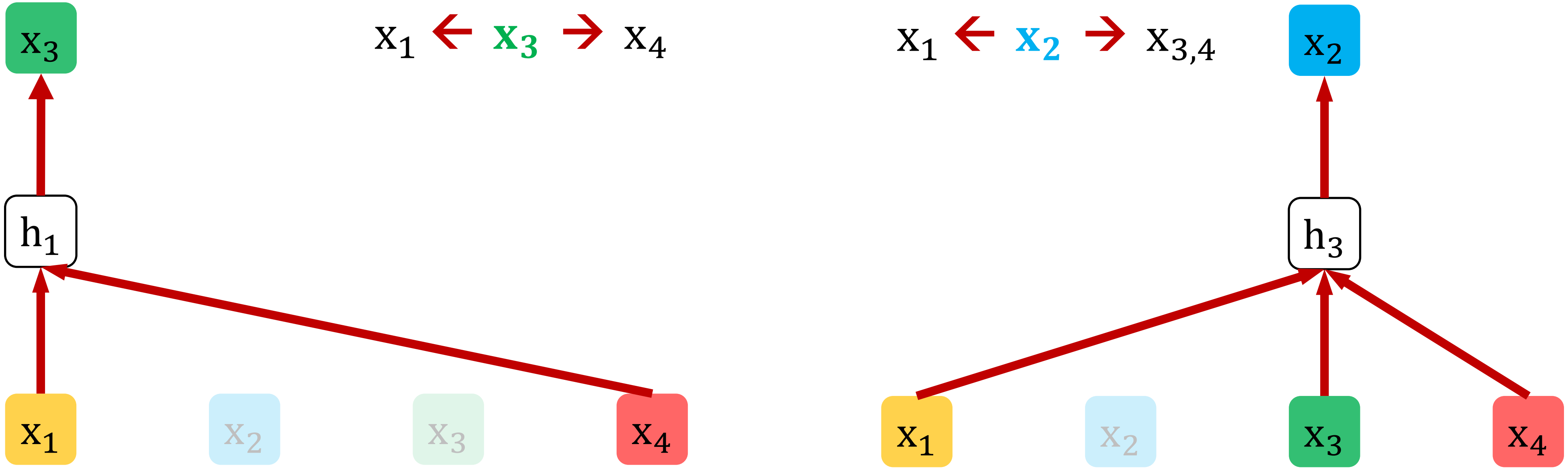
Bidirectional Context via Factorization Order

- Factorization order: 4 → 1 → 3 → 2



Bidirectional Context via Factorization Order

- Factorization order: 4 → 1 → 3 → 2



Bidirectional Context

Permutation Language Modeling

- Given a sequence $\mathbf{x}$ of length T
- Uniformly sample a factorization order $\mathbf{z}$ from all possible permutations
- Maximize the permutated log-likelihood

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

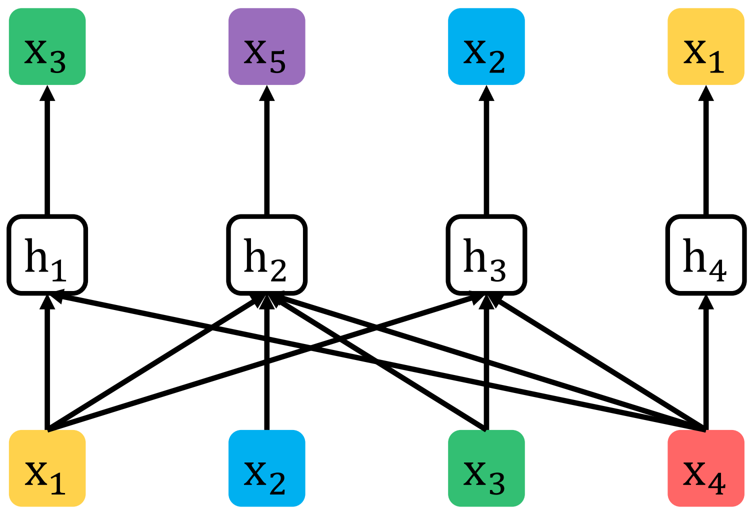
More examples

Factorization order: 4 → 1 → 3 → 2

x_1 x_2 x_3 x_4

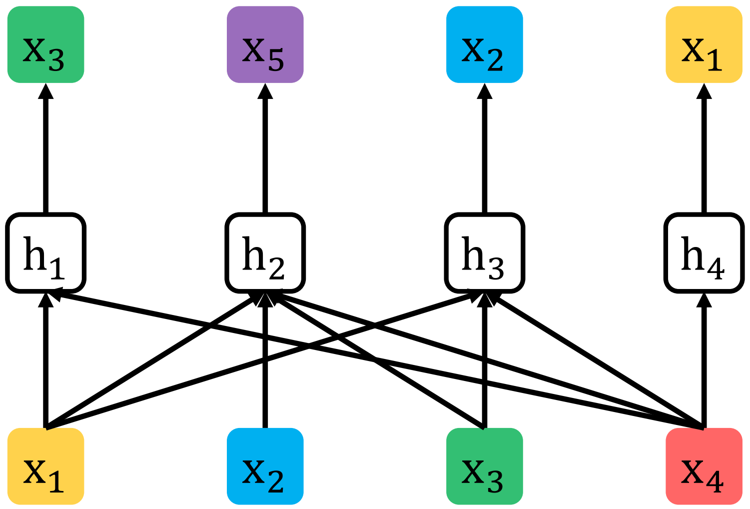
More examples

Factorization order: 4 → 1 → 3 → 2



More examples

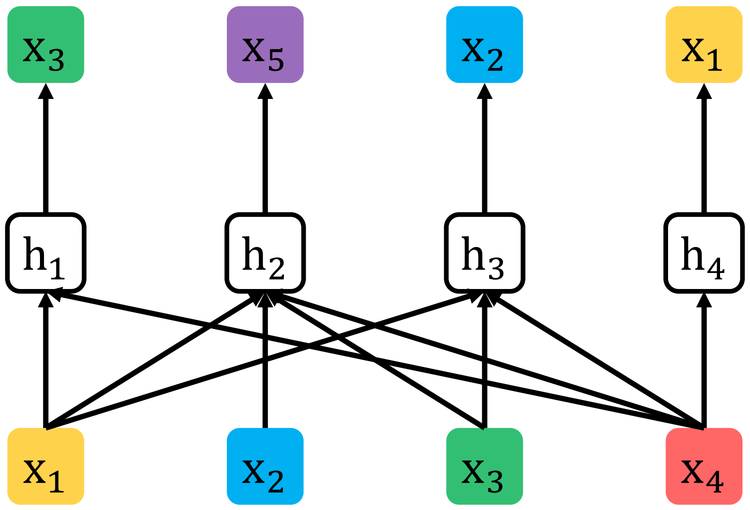
Factorization order: 4 → 1 → 3 → 2



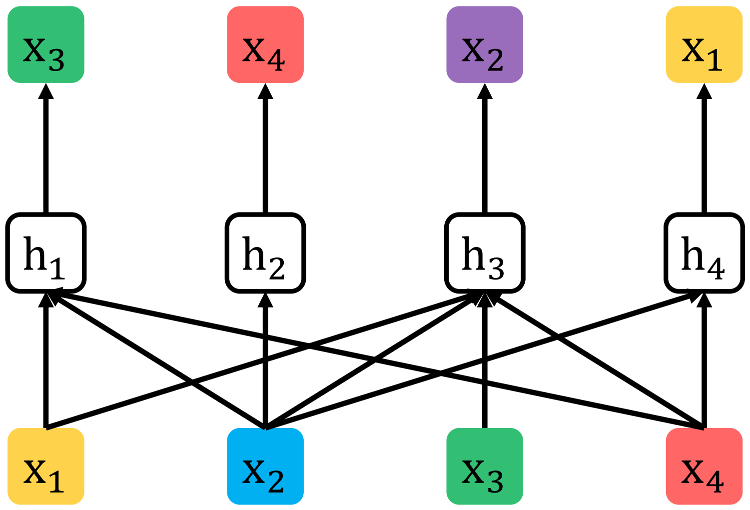
Factorization order: 2 → 4 → 1 → 3

More examples

Factorization order: 4 → 1 → 3 → 2



Factorization order: 2 → 4 → 1 → 3



Target-position-aware Distribution

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

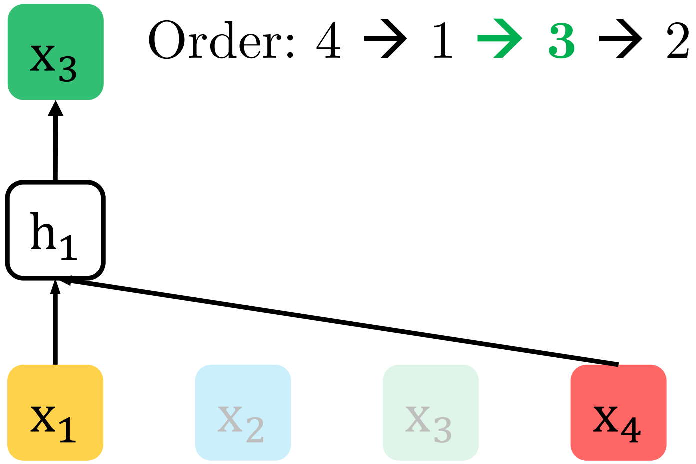
⇨

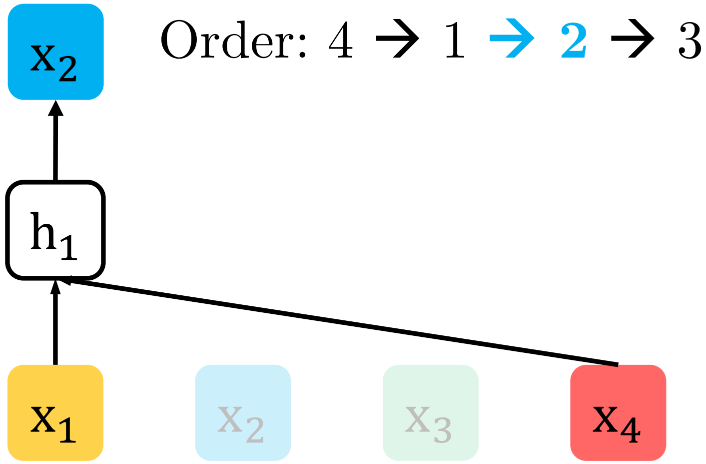
The distribution $P(x_{z_t} \mid \mathbf{x}_{\mathbf{z}_{<t}}, z_t)$ must condition on the **target position** z_t

Target-position-aware Distribution

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

The distribution $P(x_{z_t} \mid \mathbf{x}_{\mathbf{z}_{<t}}, z_t)$ must condition on the **target position** z_t

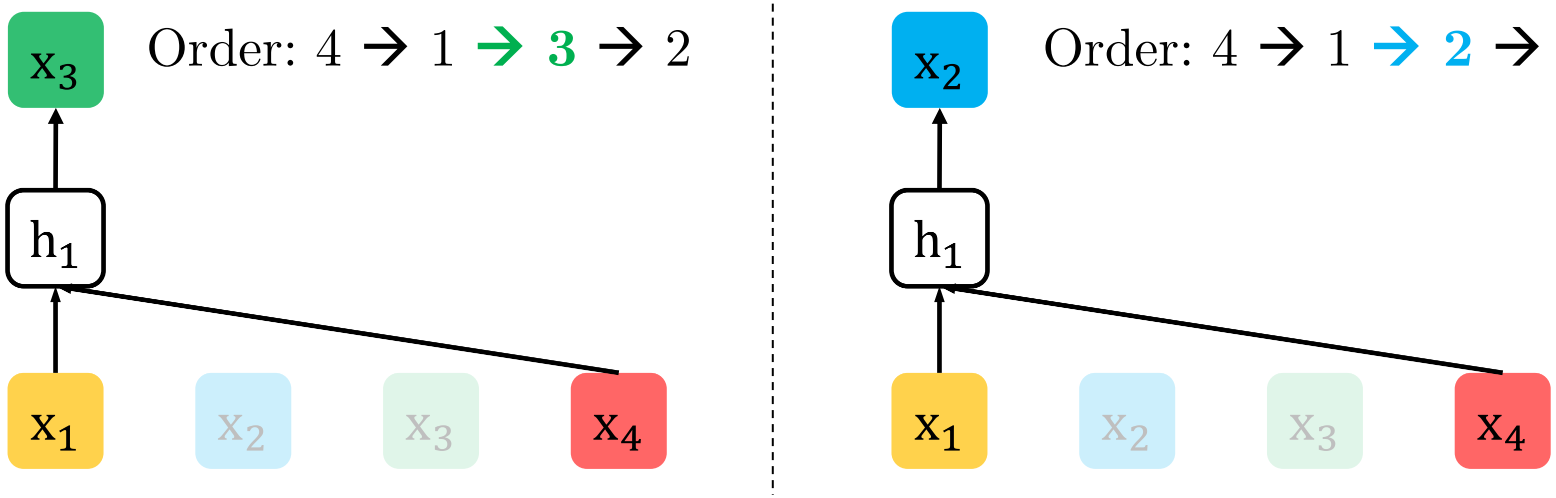




Target-position-aware Distribution

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

The distribution $P(x_{z_t} \mid \mathbf{x}_{\mathbf{z}_{<t}}, z_t)$ must **condition on the target position z_t**



- Predicting **position 3** and **position 2** requires different prediction distributions
- The prediction distribution should **change according to the target position**

Reparameterization

- Standard Softmax does **NOT** work

$$P(x_{z_t} \mid \mathbf{x}_{\mathbf{z}_{<t}}, z_t) = \frac{\exp\left(e(x_{z_t})^\top h(\mathbf{x}_{\mathbf{z}_{<t}})\right)}{\sum_{x'} \exp\left(e(x')^\top h(\mathbf{x}_{\mathbf{z}_{<t}})\right)}$$

No info. of z_t

Reparameterization

- Standard Softmax does **NOT** work

$$P(x_{z_t} \mid \mathbf{x}_{\mathbf{z}_{<t}}, z_t) = \frac{\exp\left(e(x_{z_t})^\top h(\mathbf{x}_{\mathbf{z}_{<t}})\right)}{\sum_{x'} \exp\left(e(x')^\top h(\mathbf{x}_{\mathbf{z}_{<t}})\right)}$$

No info. of z_t

- **Proposed** solution: incorporate z_t into **hidden states**

$$P(x_{z_t} \mid \mathbf{x}_{\mathbf{z}_{<t}}, z_t) = \frac{\exp\left(e(x_{z_t})^\top g(z_t, \mathbf{x}_{\mathbf{z}_{<t}})\right)}{\sum_{x'} \exp\left(e(x')^\top g(z_t, \mathbf{x}_{\mathbf{z}_{<t}})\right)}$$

Deep Net

Reparameterization

- Standard Softmax does **NOT** work

$$P(x_{z_t} \mid \mathbf{x}_{\mathbf{z}_{<t}}, z_t) = \frac{\exp\left(e(x_{z_t})^\top h(\mathbf{x}_{\mathbf{z}_{<t}})\right)}{\sum_{x'} \exp\left(e(x')^\top h(\mathbf{x}_{\mathbf{z}_{<t}})\right)} \longleftarrow \boxed{\text{No info. of } z_t}$$

- **Proposed** solution: incorporate z_t into **hidden states**

$$P(x_{z_t} \mid \mathbf{x}_{\mathbf{z}_{<t}}, z_t) = \frac{\exp\left(e(x_{z_t})^\top g(z_t, \mathbf{x}_{\mathbf{z}_{<t}})\right)}{\sum_{x'} \exp\left(e(x')^\top g(z_t, \mathbf{x}_{\mathbf{z}_{<t}})\right)} \longleftarrow \boxed{\text{Deep Net}}$$

Question: how to implement $g(z_t, \mathbf{x}_{z_{<t}})$?

Target Position Aware Representation: $g(z_t, \mathbf{x}_{z_{<t}})$

Reuse the Idea of Attention ⇨

- Stand at the target position z_t
- Gather information from $\mathbf{x}_{z_{<t}}$

Target Position Aware Representation: $g(z_t, \mathbf{x}_{z_{<t}})$

Reuse the Idea of Attention ⇨

- Stand at the target position z_t
- Gather information from $\mathbf{x}_{z_{<t}}$

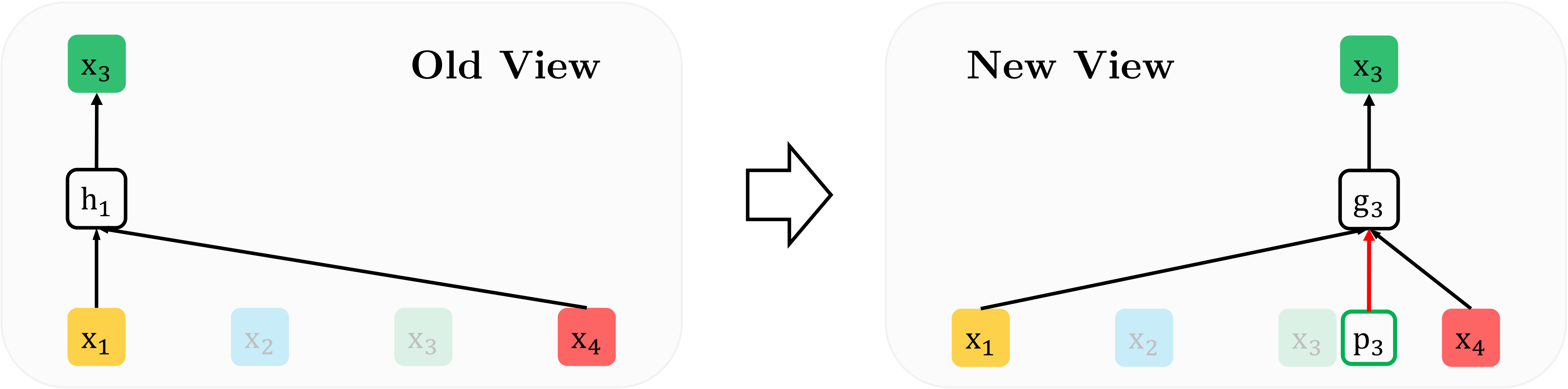
$$g(z_t, \mathbf{x}_{\mathbf{z}_{<t}}) = \mathrm{Attn}_\theta\Big(\underbrace{\mathrm{Q} = \mathrm{Enc}(z_t)}_{\text{Stand at } z_t}, \quad \underbrace{\mathrm{KV} = \mathbf{h}(\mathbf{x}_{\mathbf{z}_{<t}})}_{\text{Gather info. from } \mathbf{x}_{\mathbf{z}_{<t}}}\Big)$$

Target Position Aware Representation: $g(z_t, \mathbf{x}_{z_{<t}})$

Reuse the Idea of Attention ⇨

- Stand at the target position z_t
- Gather information from $\mathbf{x}_{z_{<t}}$

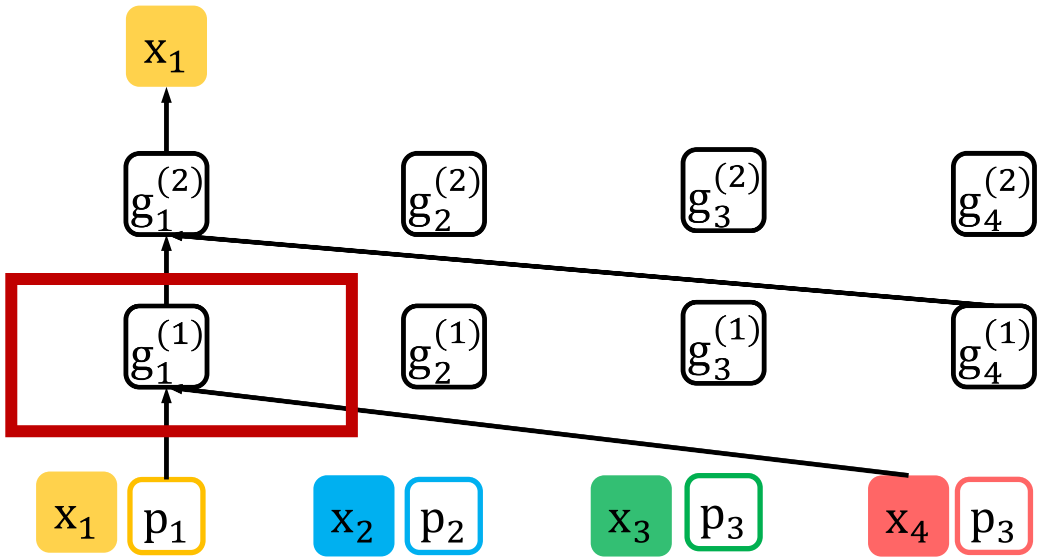
$$g(z_t, \mathbf{x}_{\mathbf{z}_{<t}}) = \mathrm{Attn}_\theta\Big(\underbrace{\mathrm{Q} = \mathrm{Enc}(z_t)}_{\text{Stand at } z_t}, \quad \underbrace{\mathrm{KV} = \mathbf{h}(\mathbf{x}_{\mathbf{z}_{<t}})}_{\text{Gather info. from } \mathbf{x}_{\mathbf{z}_{<t}}}\Big)$$



Contradiction: Predicting Self and Others

- Factorization order: 4 → 1 → 3 → 2

Use $g_1^{(1)}$ to predict $\boldsymbol{x_1}$ (self)



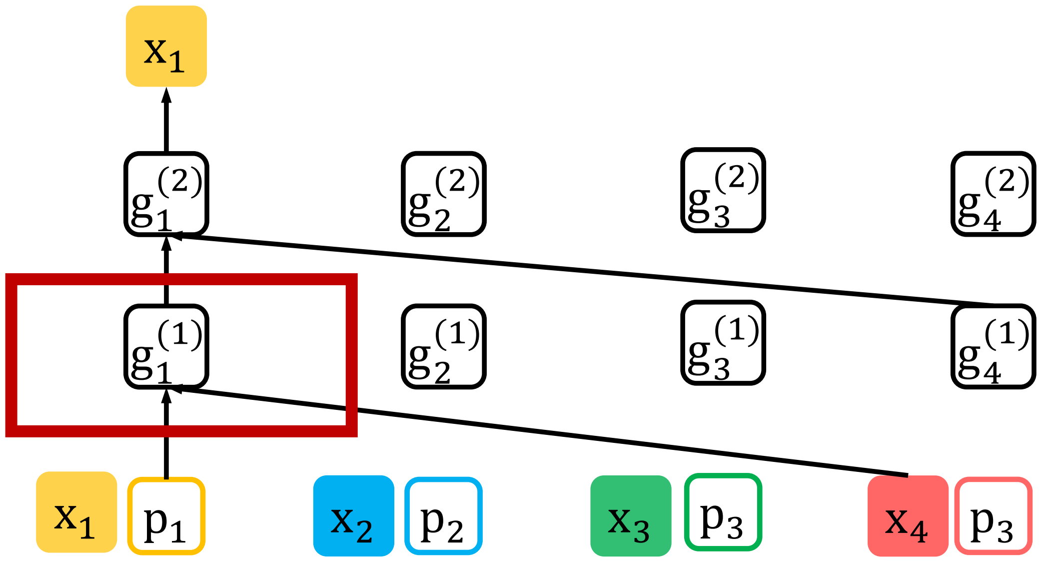
Should not encode $\boldsymbol{x_1}$

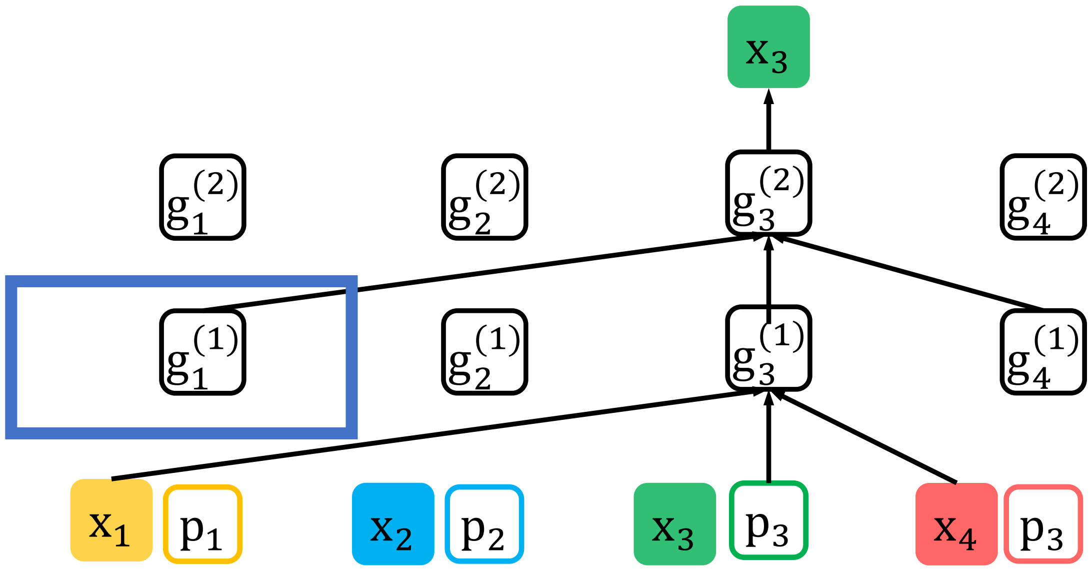
Contradiction: Predicting Self and Others

- Factorization order: 4 → 1 → 3 → 2

Use $g_1^{(1)}$ to predict $\boldsymbol{x_1}$ (self)

Use $g_1^{(1)}$ to predict $\boldsymbol{x_3}$ (other)





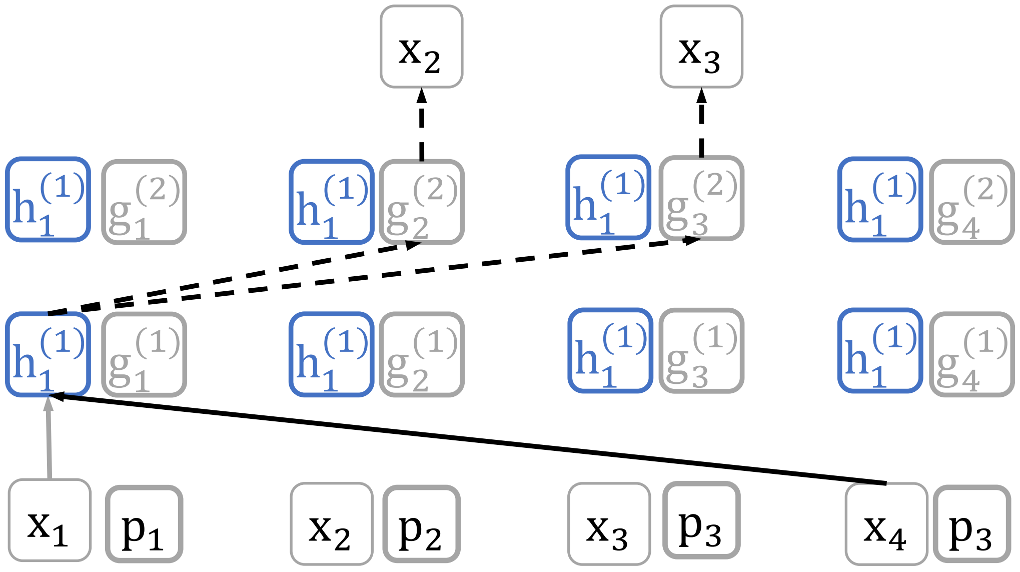
Should not encode $\boldsymbol{x_1}$

Should encode $\boldsymbol{x_1}$

Two-Stream Attention

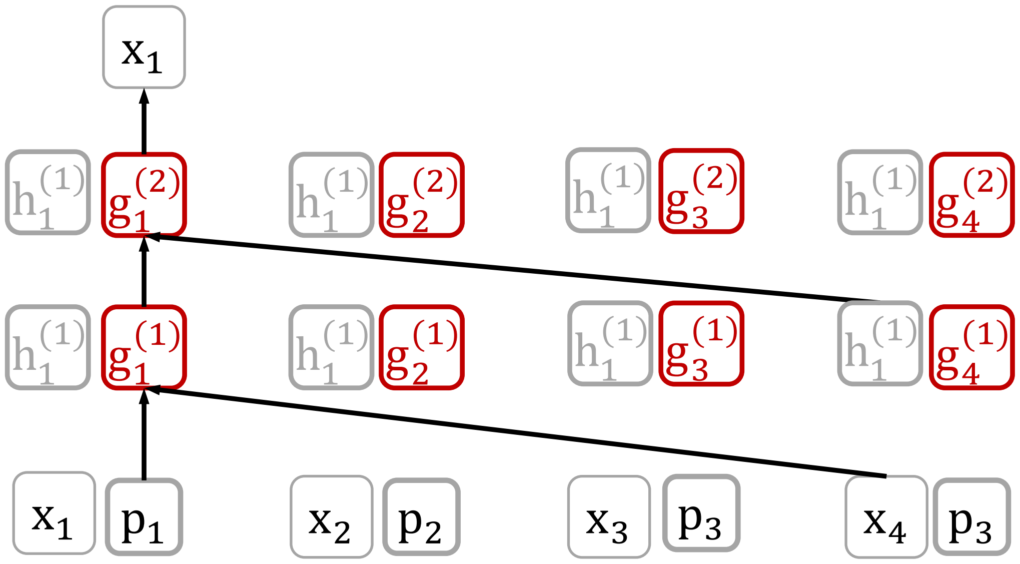
- Factorization order: 4 → 1 → 3 → 2

Encoding. Predicting x_2 and x_3 (others).



h_1 encodes x_1

Decoding. Predicting x_1 (self).



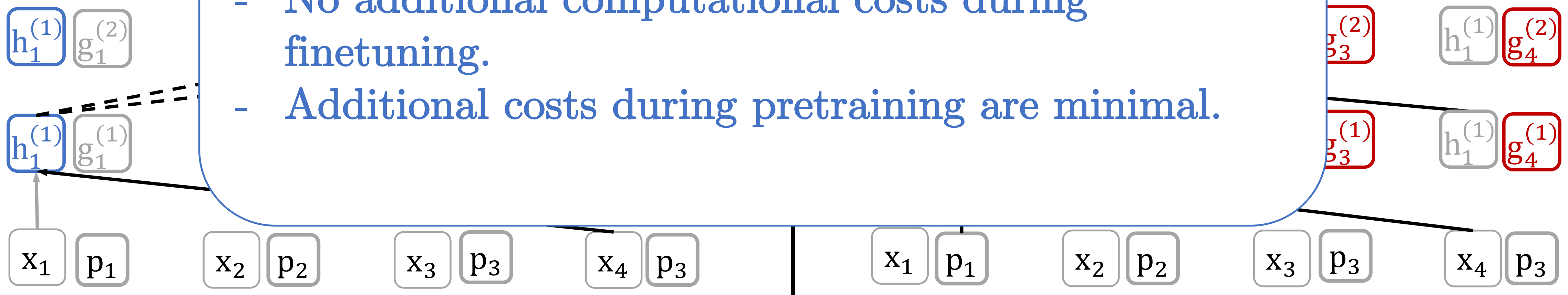
g_1 does not encode x_1

Two-Stream Attention

- Factorization order: 4 → 1 → 3 → 2

Encoding. P ₁ (self).

- Allow Transformers to work with permutation LM.
- No additional parameters.
- No additional computational costs during finetuning.
- Additional costs during pretraining are minimal.



$\boldsymbol{h_1}$ encodes $\boldsymbol{x_1}$

$\boldsymbol{g_1}$ does not encode $\boldsymbol{x_1}$

Summarizing XLNet

Challenges

Solutions

Summarizing XLNet

Challenges

Independence assumption and distribution discrepancy in BERT

Solutions

Summarizing XLNet

Challenges

Independence assumption and distribution discrepancy in BERT

Solutions

Permutation language modeling

Summarizing XLNet

Challenges

Independence assumption and distribution discrepancy in BERT

Standard parameterization is reduced to bag-of-words

Solutions

Permutation language modeling

Summarizing XLNet

Challenges

Independence assumption and distribution discrepancy in BERT

Standard parameterization is reduced to bag-of-words

Solutions

Permutation language modeling

Reparameterization with positions

Summarizing XLNet

Challenges

Independence assumption and distribution discrepancy in BERT

Standard parameterization is reduced to bag-of-words

Contradiction for predicting both self and others

Solutions

Permutation language modeling

Reparameterization with positions

Summarizing XLNet

Challenges

- Independence assumption and distribution discrepancy in BERT
- Standard parameterization is reduced to bag-of-words
- Contradiction for predicting both self and others

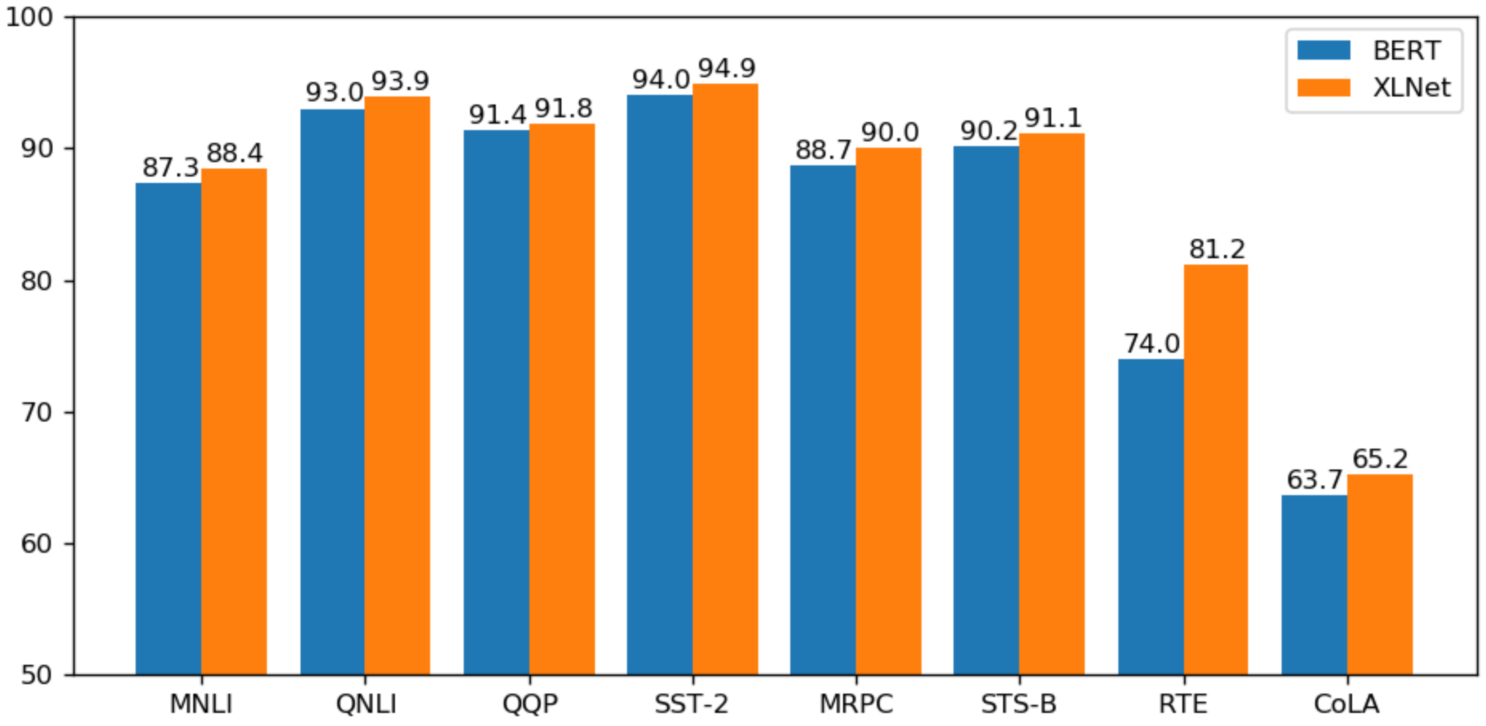
Solutions

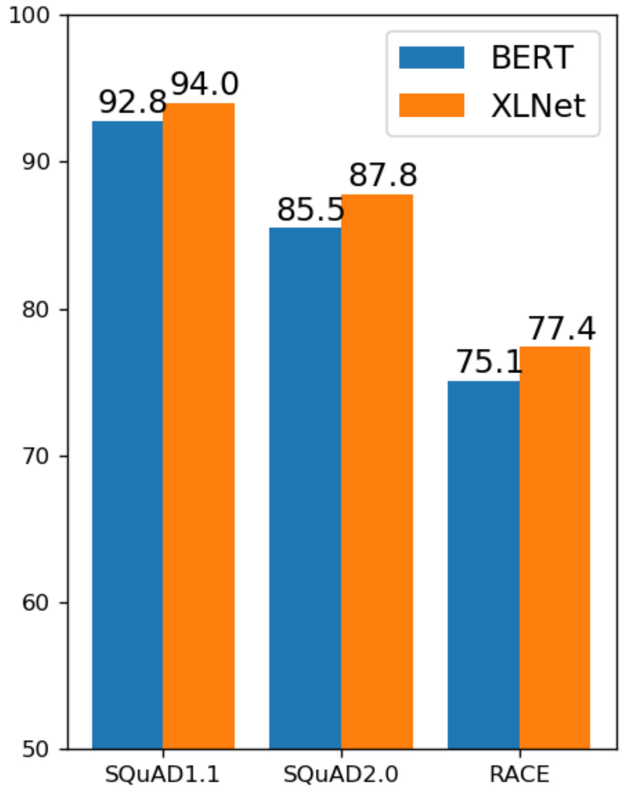
- Permutation language modeling
- Reparameterization with positions
- Two-stream attention

Experiment 1: Comparison with BERT

- **Same** training data **as in BERT**: Wikipedia + BooksCorpus
- **Same** hyperparameters for pretraining **as in BERT**
  - Model size: L=24, H=1024, A=16
  - Batch size: 256
  - Number of steps: 1M
  - ...
- **Same** hyperparameter search space for finetuning **as in BERT**

XLNet outperforms BERT on 20 tasks



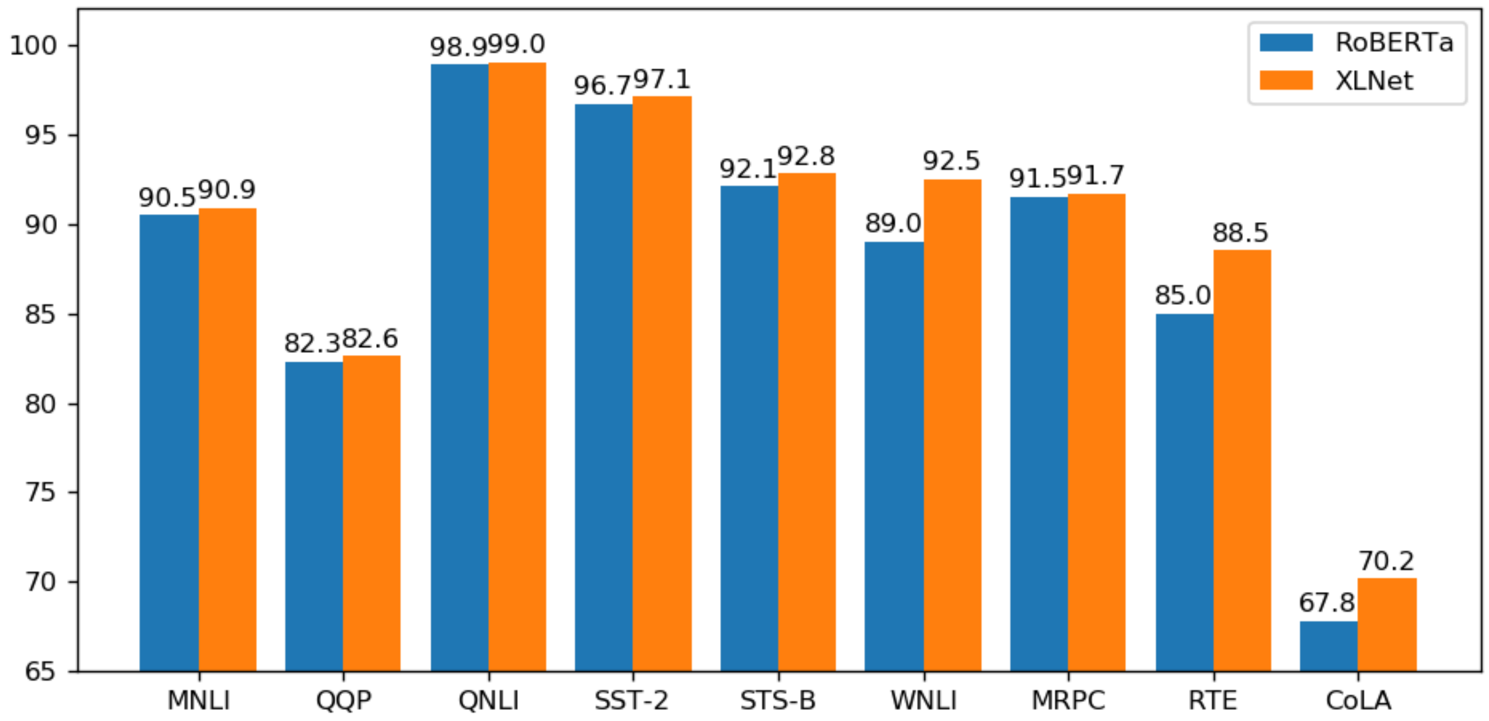


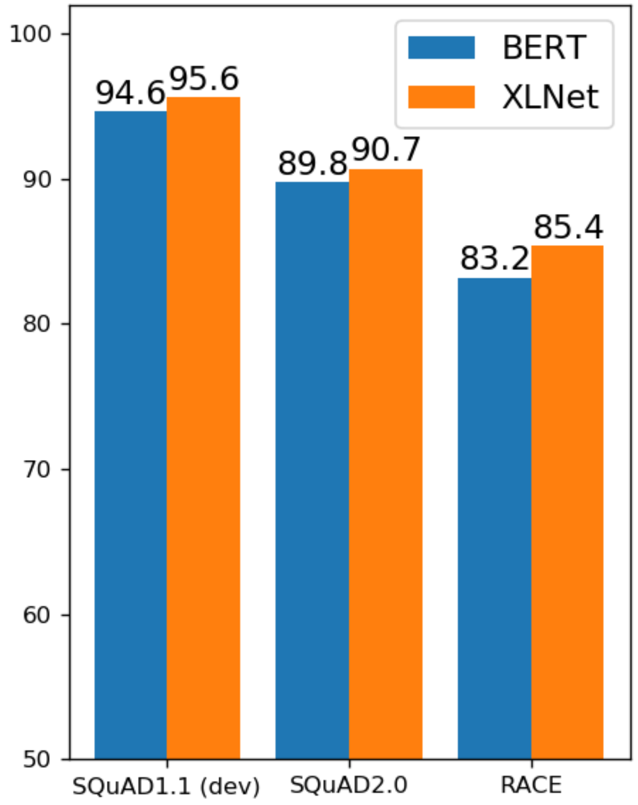
We report the best of 3 BERT variants.
Almost identical training recipes.

Experiment 2: Comparison with RoBERTa

- Less training data for XLNet: 126GB vs 160GB
- **Same** hyperparameters for pretraining **as in RoBERTa**
  - Model size: L=24, H=1024, A=16
  - Batch size: 8192
  - Number of steps: 500K
  - ...
- **Same** hyperparameter search space for finetuning **as in RoBERTa**

XLNet outperforms RoBERTa on all considered tasks



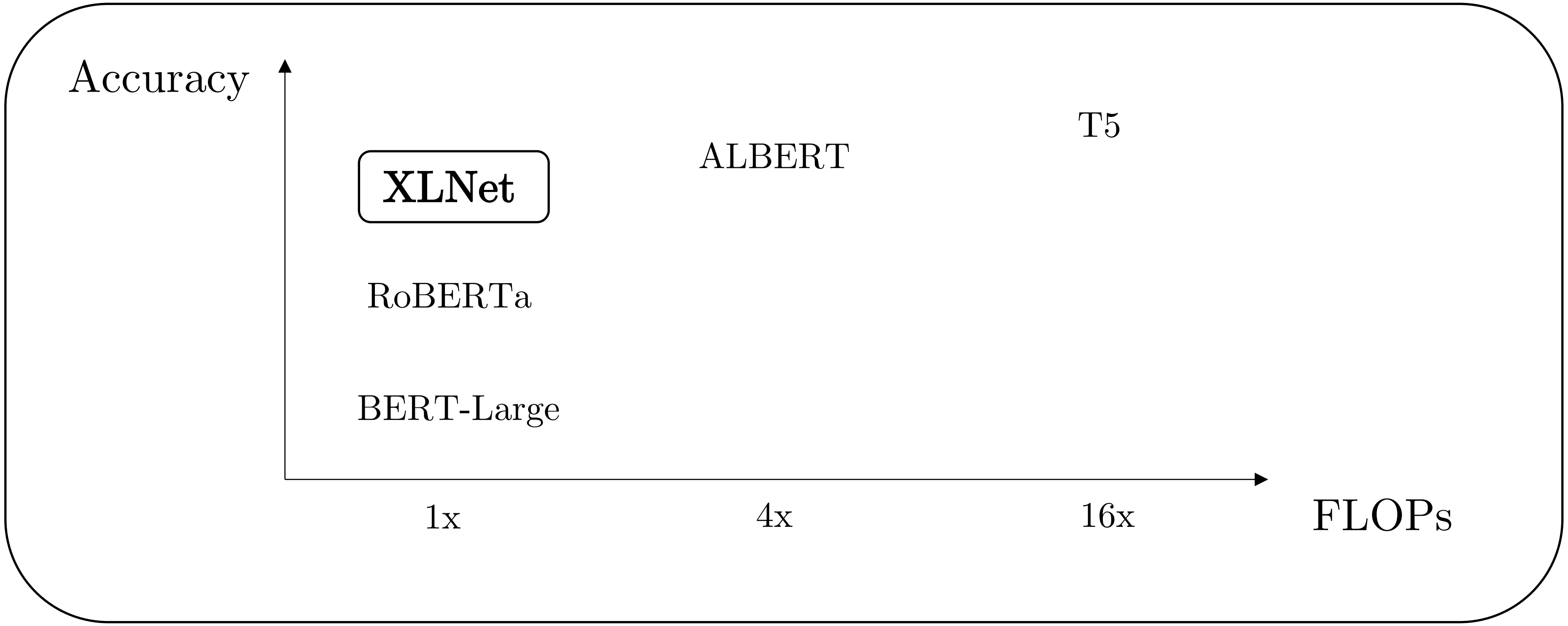


Almost identical training recipes.

XLNet is

The best pretrained model today

Given standard FLOPs.



XLNet-2 Coming Soon!

Optimized data processing

Optimized model implementation

- Only about 10% slower than BERT during pretraining
- Finetuning speed and memory are identical to BERT
- Outperforms BERT (and larger BERT-like models) consistently under all considered settings

To be release at https://github.com/zihangdai/xlnet

Future Work

Large models $\xleftrightarrow{\text{Tradeoff}}$ Efficiency

More data $\xleftrightarrow{\text{Tradeoff}}$ Data quality

Thanks!

Zhilin Yang*, Zihang Dai*, Yiming Yang, Jaime Carbonell,
Ruslan Salakhutdinov, Quoc V. Le
(*: equal contribution)