XLNet: Generalized Autoregressive Pre-training for Language Understanding

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

Presented by Andrew Or, Ksenia Sokolova

Carnegie Mellon University

Google AI

Outline

XLNet Key Ideas: high-level comparison with BERT

XLNet Backbone: Transformer-XL

Pre-training Objectives: comparison with AR and BERT

XLNet Design: permutation, masks, two-stream attention

Results: XLNet outperforms BERT on 20 tasks

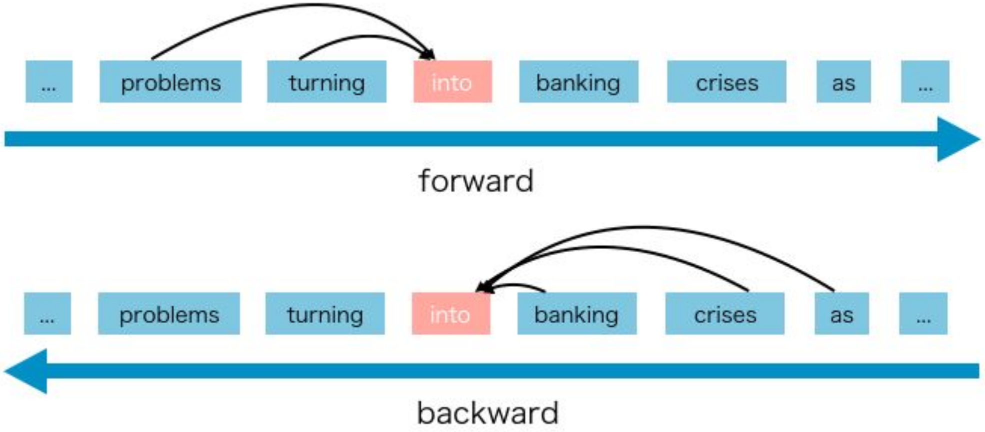
Background

★ Before: autoregressive (ex. ELMo, GPT) and autoencoding (ex. BERT) models are the two most successful pre-training objectives

★ Both approaches have their own limitations

Autoregressive Models

Use context to predict the next word



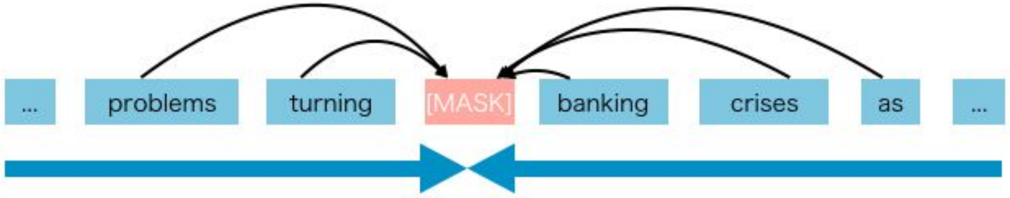
$$p(\mathbf{x}) = \prod_{t=1}^{T} p(x_t \mid \mathbf{x}_{<t})$$

$$p(\mathbf{x}) = \prod_{t=T}^{1} p(x_t \mid \mathbf{x}_{>t})$$

Only considers context in one direction

Autoencoding Models (BERT)

Note: previously a SOTA pretraining approach



❌ **Fine-tuning discrepancy** caused by [MASK] tokens (not in real data)

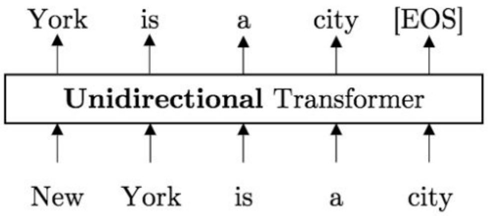
Peter has a [MASK] that does not like [MASK]

Assumes *cat* and *yarn* are independent, which is wrong

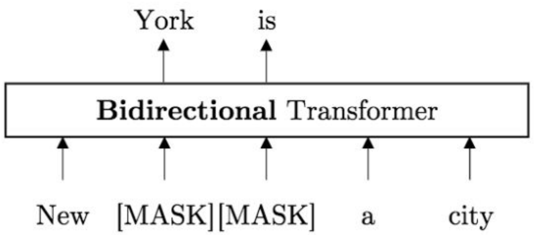
❌ No **joint probability** between masked entries

Two Notable Objectives for Language Pretraining

Auto-regressive Language Modeling



Denoising Auto-encoding (BERT)



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**

XLNet Key Ideas

Autoregressive: use context to predict the next word

Bidirectional context from permutation language modeling

Self-attention mechanisms, uses Transformer-XL backbone

XLNet Key Ideas

Autoregressive: use context to predict the next word

Bidirectional context from permutation language modeling

Peter's ***cat*** *likes yarn*

Peter's cat likes yarn	*yarn Peter's cat likes*	
Peter's cat yarn likes	*yarn Peter's likes cat*	
Peter's likes cat yarn	*yarn cat Peter's likes*	
Peter's likes yarn cat	*yarn cat likes Peter*	…
Peter's yarn cat likes	*yarn likes Peter's cat*	
Peter's yarn likes cat	*yarn likes cat Peter's*	

XLNet Key Ideas

Autoregressive: use context to predict the next word

Bidirectional context from permutation language modeling

Peter's ***cat*** *likes yarn*

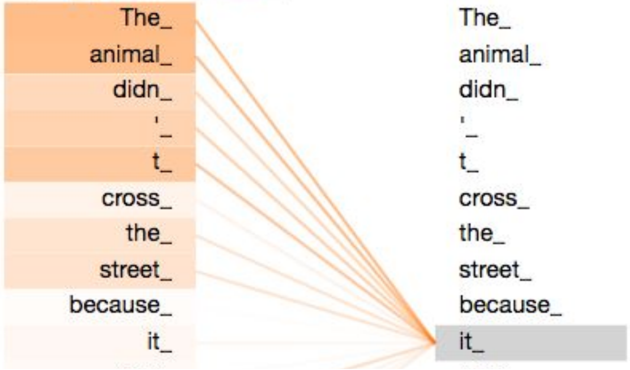
Peter's cat *likes yarn*	*yarn Peter's cat likes*	
Peter's cat yarn likes	*yarn Peter's likes cat*	
Peter's ***likes cat*** *yarn*	*yarn cat Peter's likes*	
Peter's likes yarn cat	*yarn cat likes Peter*	…
Peter's ***yarn cat*** *likes*	*yarn likes Peter's cat*	
Peter's yarn likes cat	*yarn likes cat Peter's*	

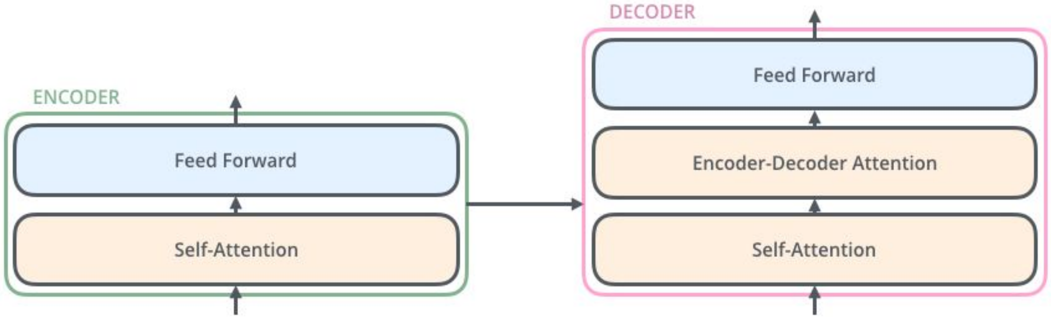
XLNet Key Ideas

Autoregressive: use context to predict the next word

Bidirectional context from permutation language modeling

Self-attention mechanisms, uses Transformer-XL backbone





Transformer-XL

extra long

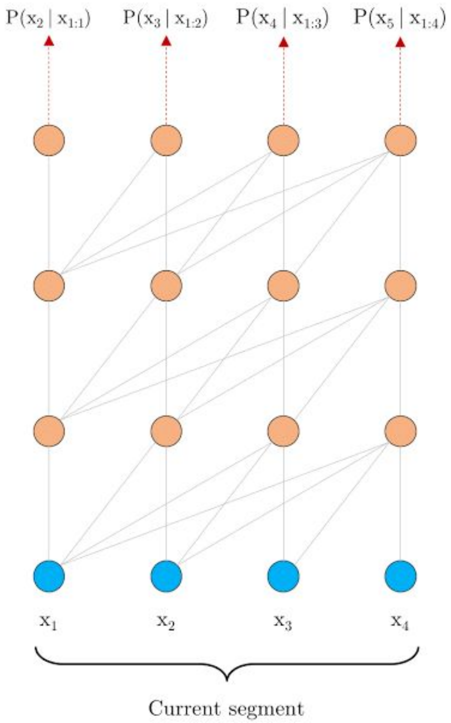
Increases context through segment-level recurrence and a novel positional encoding scheme

Transformer-XL

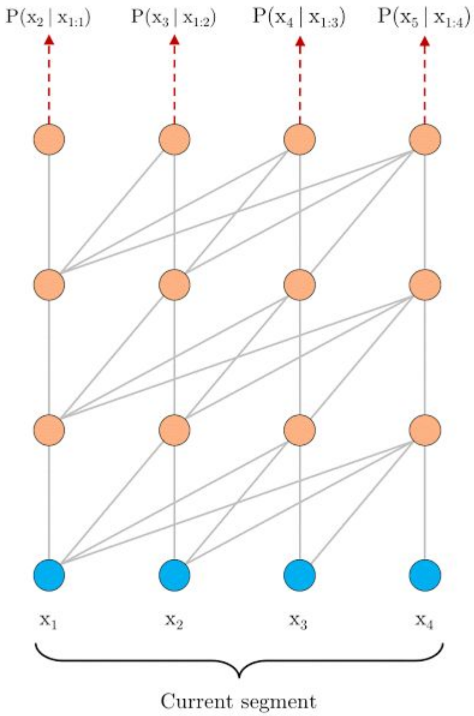
Increases context through **segment-level recurrence** and a novel positional encoding scheme

- **Cache and reuse hidden state** from the previous segment
- Allows **variable-length context**, great for capturing long-term dependencies
- Resolves the problem of context fragmentation

Before (no segment-level recurrence)



After segment-level recurrence

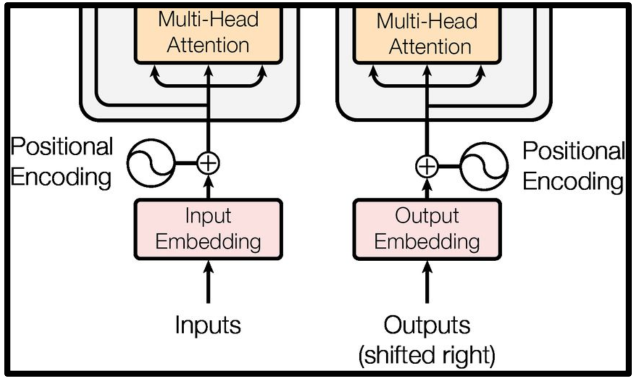


Transformer-XL

Increases context through segment-level recurrence and a novel **positional encoding scheme**

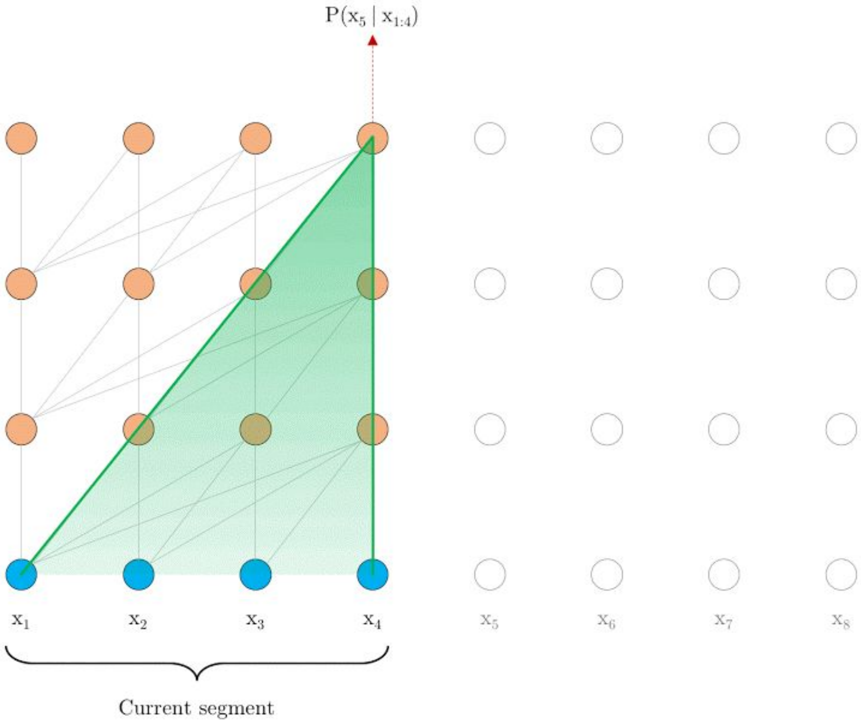
Need a way to keep positional information coherent when we reuse the states

- In the original Transformer: *absolute* position within a segment is used
- Need to encode *relative* position

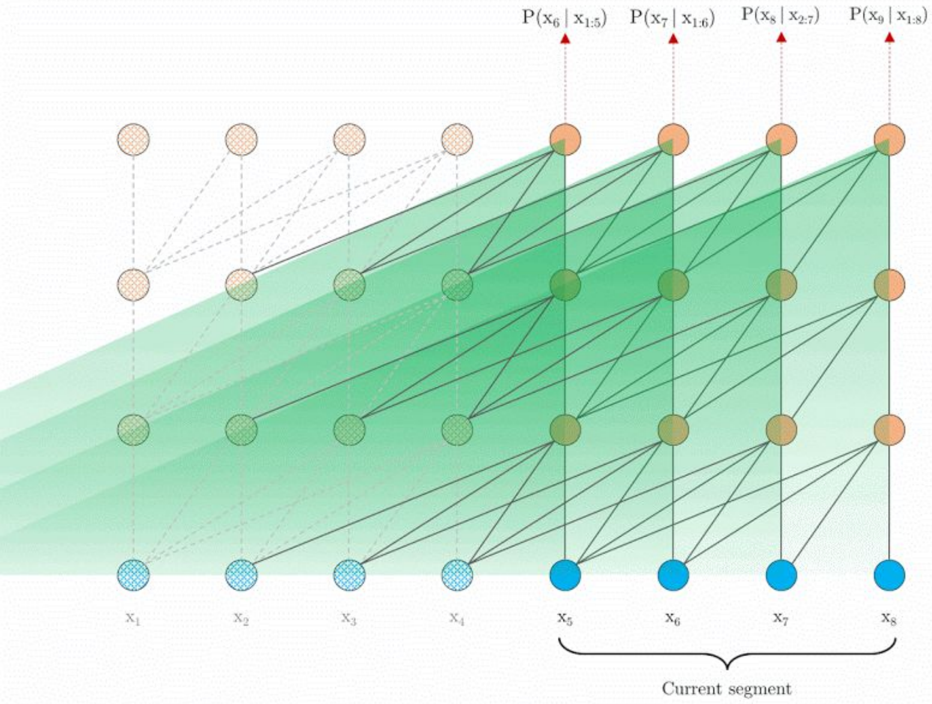


Original Transformer

Before



After



P(x6 | x1:5)
P(x7 | x1:6)
P(x8 | x2:7)
P(x9 | x1:8)
x1
x2
x3
x4
x5
x6
x7
x8
Current segment

Training objectives

Traditional AR models vs BERT vs XLNet

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

Traditional AR models

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

= 1 if masked

BERT

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

Set of all permutations

*XLNet**

XLNet Design

Permutation only on *factorization order*, not the original *sequence order*

Attention masks provide the context for each prediction

Two-stream self-attention allows prediction to be aware of target position

Partial prediction: only predict *1/K* tokens in each permutation

XLNet Design

Permutation only on *factorization order*, not the original *sequence order*

Attention masks provide the context for each prediction

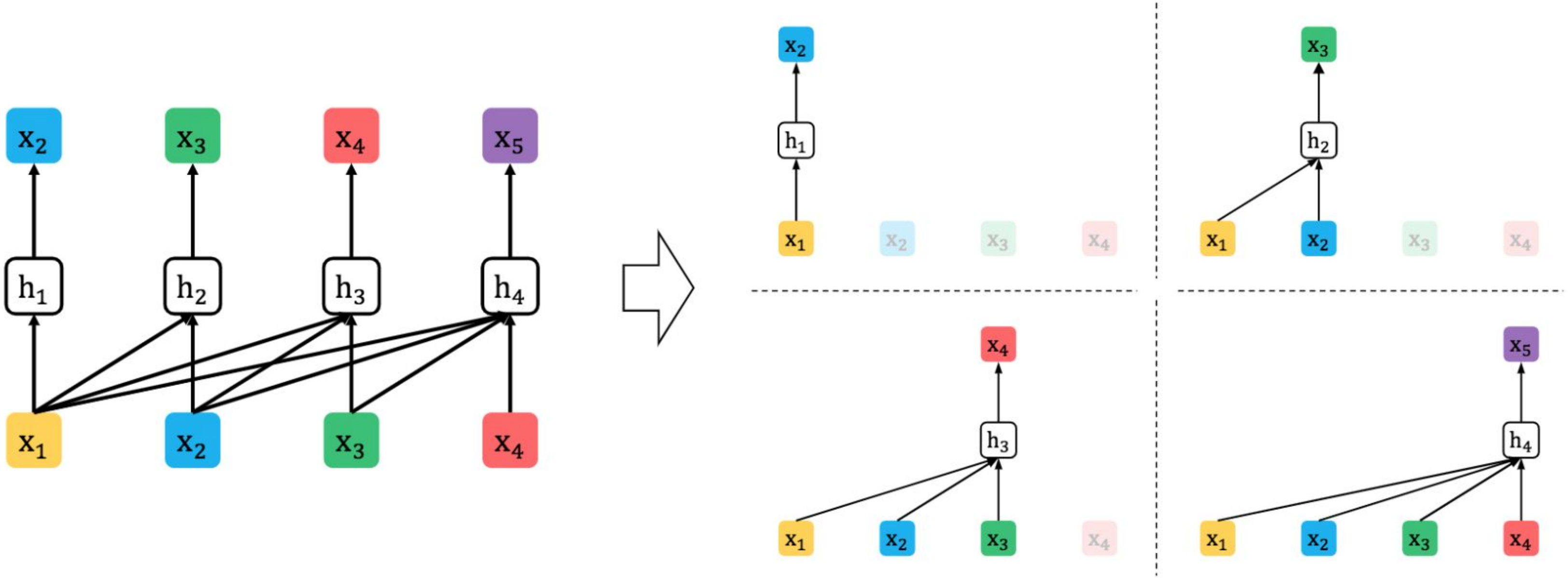
Two-stream self-attention allows prediction to be aware of target position

Partial prediction: only predict *1/K* tokens in each permutation

Context Depends on the **Factorization Order**

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

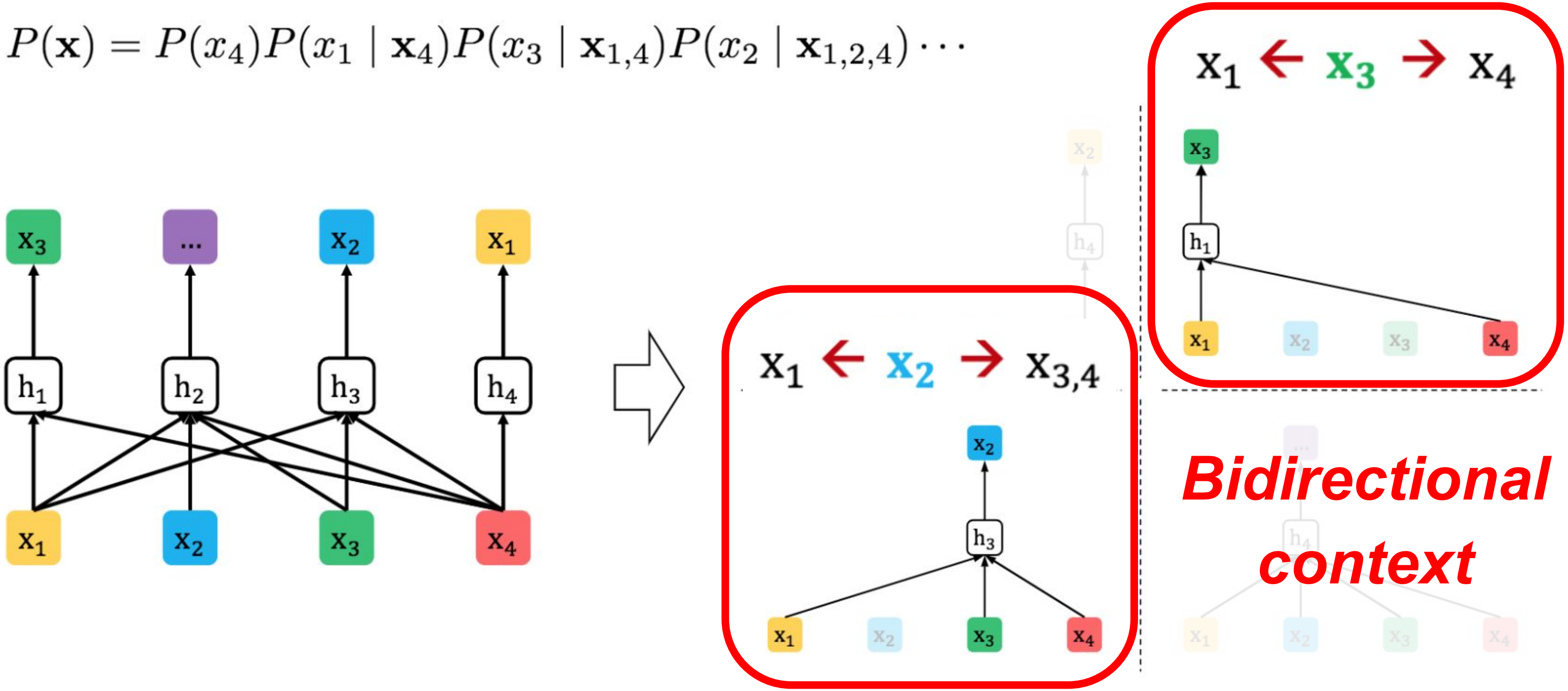
$$P(\mathbf{x}) = P(x_1)P(x_2 \mid \mathbf{x}_1)P(x_3 \mid \mathbf{x}_{1,2})P(x_4 \mid \mathbf{x}_{1,2,3})\cdots$$



Context Depends on the **Factorization Order**

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

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



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

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

XLNet Design

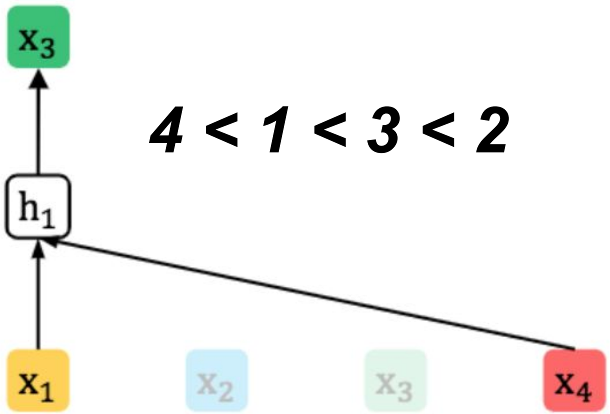
Permutation only on *factorization order*, not the original *sequence order*

Attention masks provide the context for each prediction

Two-stream self-attention allows prediction to be aware of target position

Partial prediction: only predict *1/K* tokens in each permutation

Attention Masks Provide Context



4 < 1 < 3 < 2

$w1 = f(q3 \cdot k1)$

$w2 = f(q3 \cdot k2)$

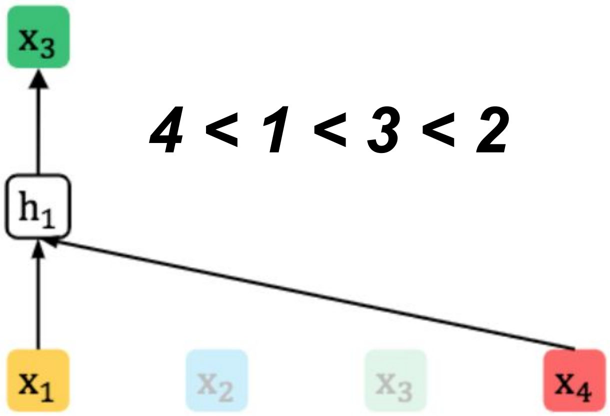
$w3 = f(q3 \cdot k3)$

$w4 = f(q3 \cdot k4)$

f = softmax + scale

Attention output $=$ ***w1*** $* v1 +$ ***w2*** $* v2 +$ ***w1*** $* v3 +$ ***w4*** $* v4$

Attention Masks Provide Context



4 < 1 < 3 < 2

$w1 = f(q3 \cdot k1)$

$w2 = 0$

$w3 = 0$

$w4 = f(q3 \cdot k4)$

f = softmax + scale

Attention output *=* ***w1*** ** v1 +* ***w4*** ** v4*

XLNet Design

Permutation only on *factorization order*, not the original *sequence order*

Attention masks provide the context for each prediction

Two-stream self-attention allows prediction to be aware of target position

Partial prediction: only predict *1/K* tokens in each permutation

Standard AR Parameterization Fails

The apple was eaten

$\boldsymbol{z_1}$ = *apple was* ***the*** *eaten* $\quad \mathrm{p}(\mathbf{x} \mid \mathrm{apple, was})$

$\boldsymbol{z_2}$ = *apple was* ***eaten*** *the* $\quad \mathrm{p}(\mathbf{x} \mid \mathrm{apple, was})$

Predicting ***the*** and ***eaten*** uses the same distribution

Fails to take into account **target position**

Standard AR Parameterization Fails

The apple was eaten

z_1 = *apple was **the** eaten* $\quad p(\mathbf{x} \mid \text{apple, was, } \textbf{[pos=0]})$

z_2 = *apple was **eaten** the* $\quad p(\mathbf{x} \mid \text{apple, was, } \textbf{[pos=3]})$

Predicting ***the*** and ***eaten*** uses the same distribution

Fails to take into account **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

- **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

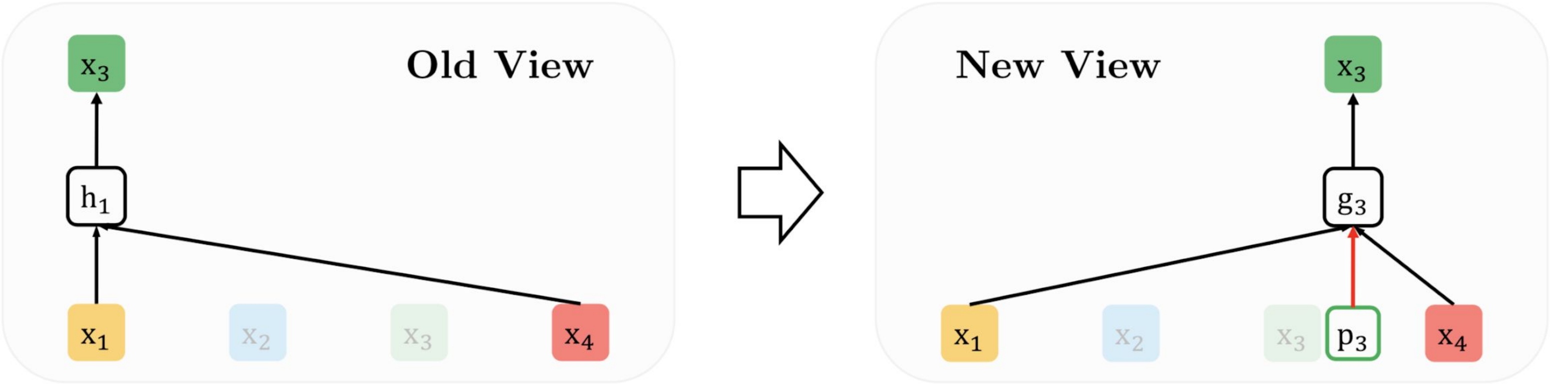
Implement this using **two-stream** architecture

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 $\mathbf{x}_{z_{<t}}$

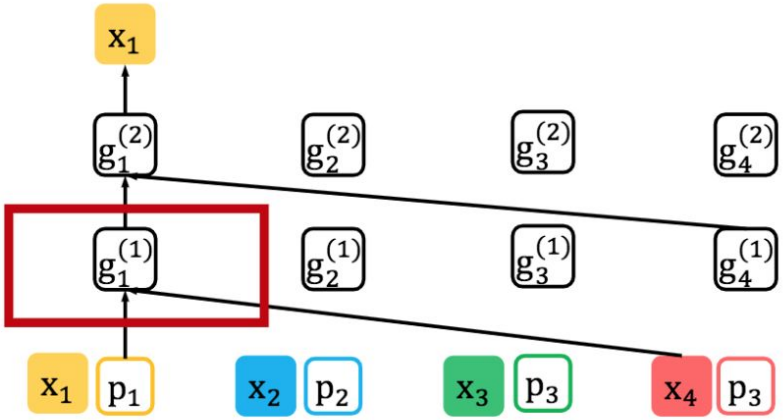
$$g(z_t, \mathbf{x}_{\mathbf{z}_{<t}}) = \mathrm{Attn}_\theta\Big(\underbrace{\mathrm{Q} = \mathrm{Enc}(z_t)}_{\text{Stand at } z_t}, \quad \underbrace{\mathrm{KV} = \mathbf{h}(\mathbf{x}_{\mathbf{z}_{<t}})}_{\text{Gather info. from } \mathbf{x}_{\mathbf{z}_{<t}}} \Big)$$



Contradiction: Predicting Self and Others

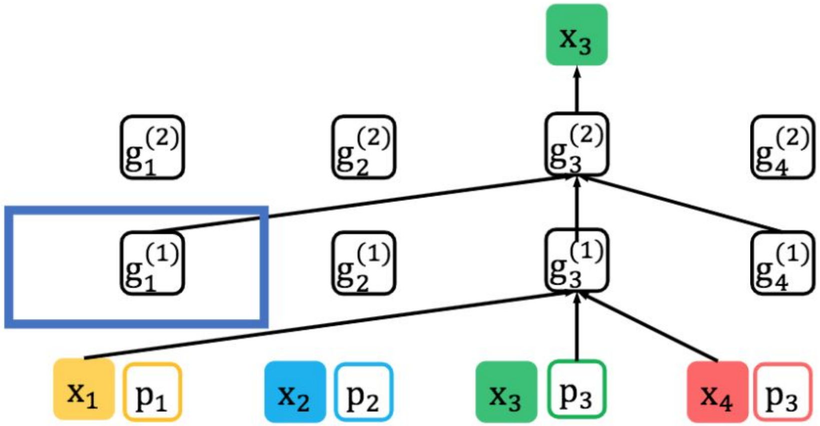
- Factorization order: 4 → 1 → 3 → 2

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



Should not encode $\boldsymbol{x_1}$

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



Should encode $\boldsymbol{x_1}$

Two-Stream Self-Attention

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

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

Query stream encodes target position information (z_t)

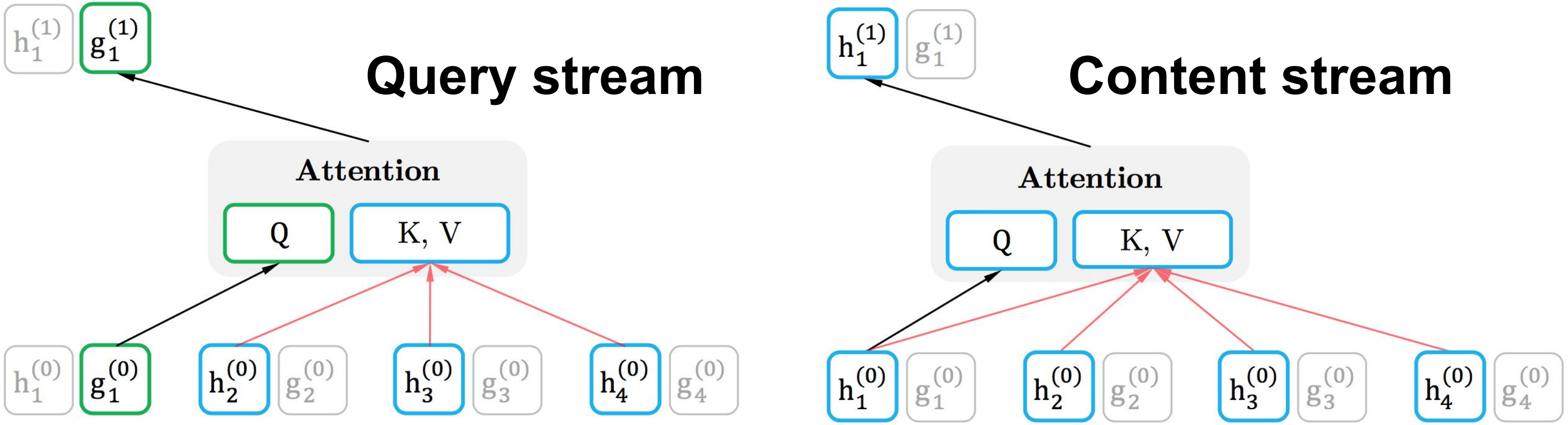
Content stream encodes both context and the target word (x_{z_t})

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

Two-Stream Self-Attention

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

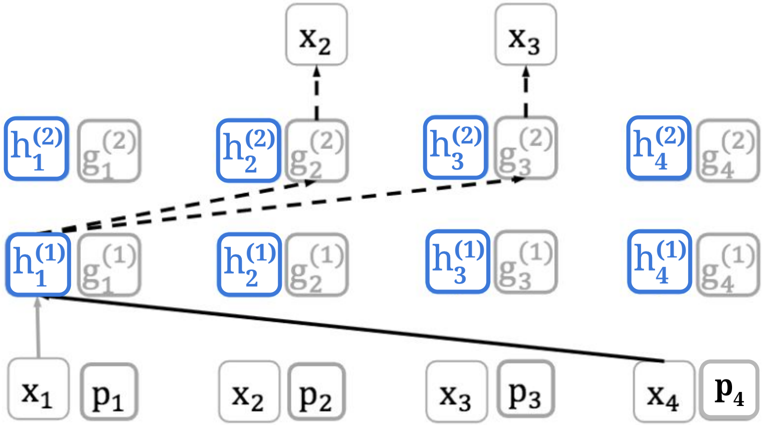
$$h_{z_t}^{(m)} \leftarrow \text{Attention}(\mathrm{Q} = h_{z_t}^{(m-1)}, \mathrm{KV} = \mathbf{h}_{\mathbf{z}_{\leq t}}^{(m-1)}; \theta), \quad \text{(content stream: use both } z_t \text{ and } x_{z_t}\text{).}$$



Two-Stream Attention

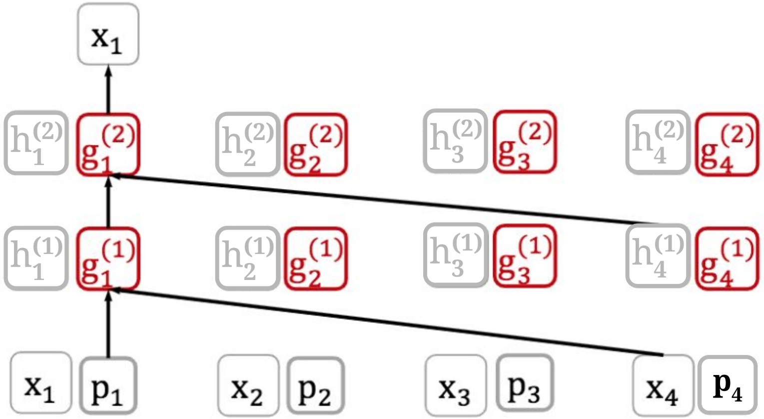
- Factorization order: 4 → 1 → 3 → 2

Encoding. Predicting x_2 and x_3 (others).

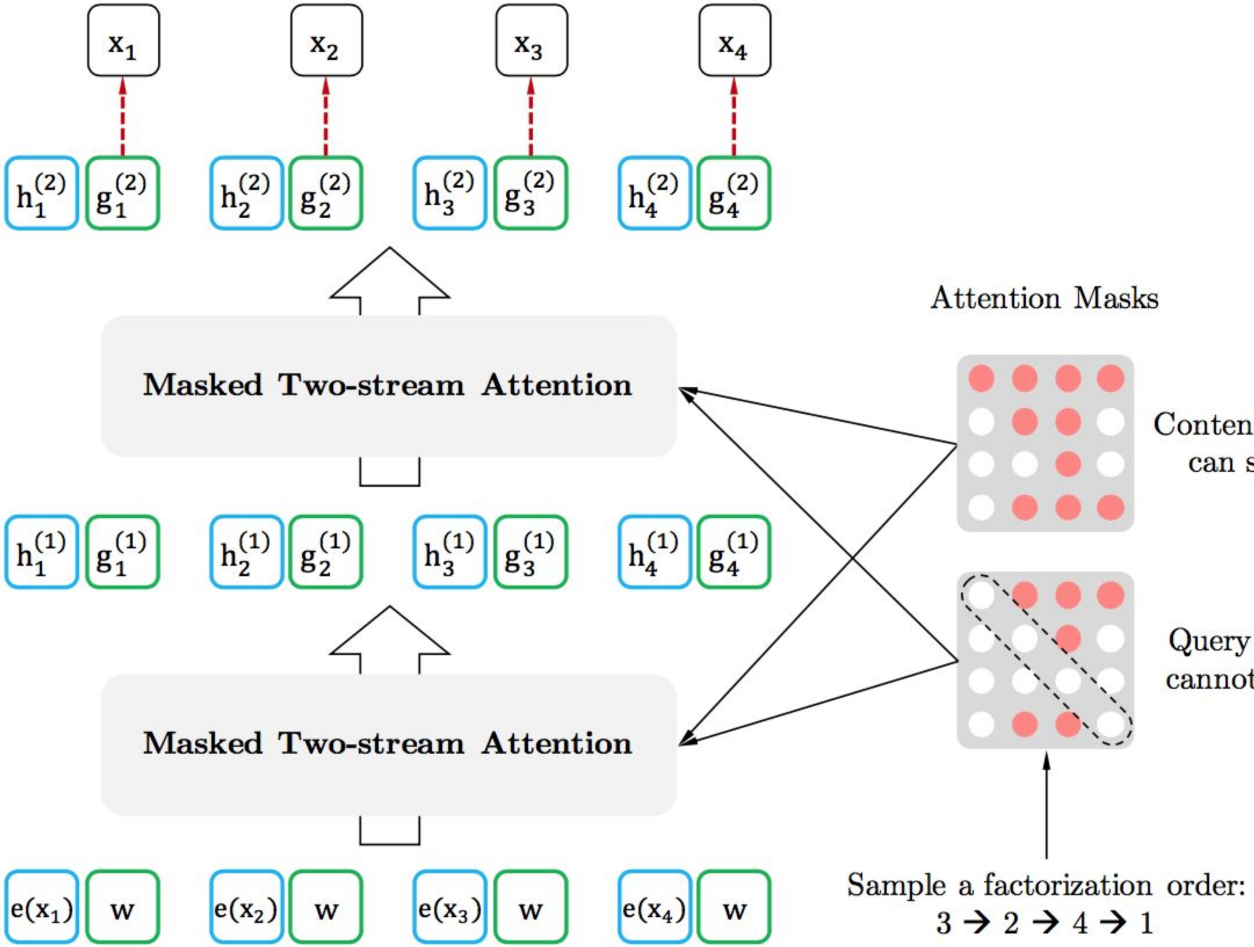


$\boldsymbol{h_1}$ encodes $\boldsymbol{x_1}$

Decoding. Predicting $\boldsymbol{x_1}$ (self).



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



x_1
x_2
x_3
x_4
$h_1^{(2)}$ $g_1^{(2)}$ $h_2^{(2)}$ $g_2^{(2)}$ $h_3^{(2)}$ $g_3^{(2)}$ $h_4^{(2)}$ $g_4^{(2)}$
Attention Masks
Masked Two-stream Attention
Content stream:
can see self
$h_1^{(1)}$ $g_1^{(1)}$ $h_2^{(1)}$ $g_2^{(1)}$ $h_3^{(1)}$ $g_3^{(1)}$ $h_4^{(1)}$ $g_4^{(1)}$
Query stream:
cannot see self
Masked Two-stream Attention
$e(x_1)$ w $e(x_2)$ w $e(x_3)$ w $e(x_4)$ w
Sample a factorization order:
3 → 2 → 4 → 1

XLNet Design

Permutation only on *factorization order*, not the original *sequence order*

Attention masks provide the context for each prediction

Two-stream self-attention allows prediction to be aware of target position

Partial prediction: only predict $1/K$ tokens in each permutation

Partial Prediction

Motivation: reduce optimization difficulty from too little context

Split sequence into ***context words*** and ***target words***, cut off at c

Only predict target words ($1/K$ of original sequence)

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

$$|\mathbf{z}| \,/ (|\mathbf{z}| - c) \approx K = 6$$ (~17% target)

XLNet-Large

Example: Comparison with BERT

Input sentence: New York is a city, masked *New* and *York*

XLNet factorization order: [is, a, city, New, York]

$$\log p(\text{New York} \mid \text{is a city})$$

$$\mathcal{J}_{\text{BERT}} = \log p(\text{New} \mid \text{is a city}) + \log p(\text{York} \mid \text{is a city}),$$

$$\mathcal{J}_{\text{XLNet}} = \log p(\text{New} \mid \text{is a city}) + \log p(\text{York} \mid \text{New}, \text{is a city}).$$

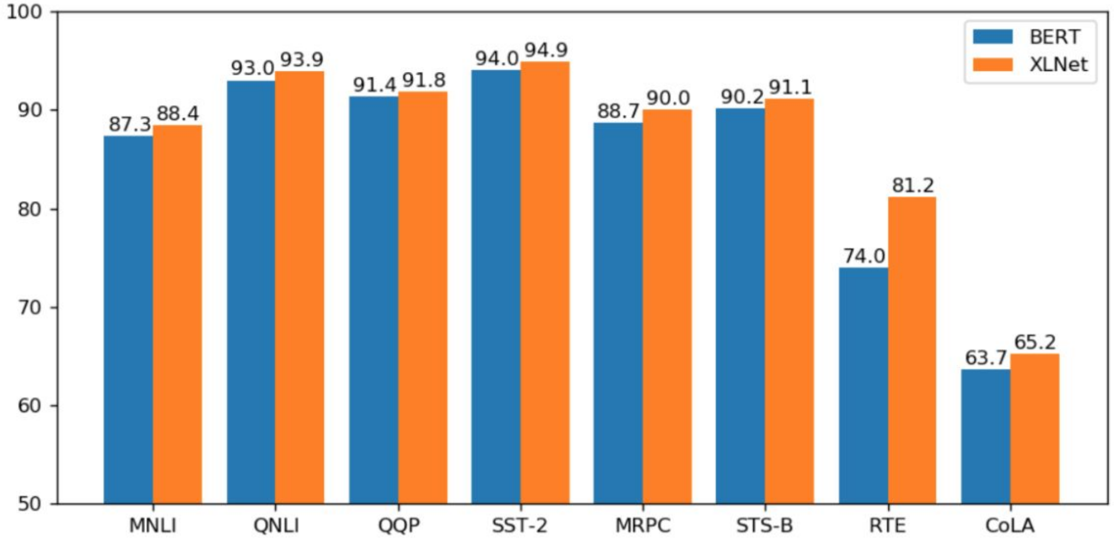
Evaluation

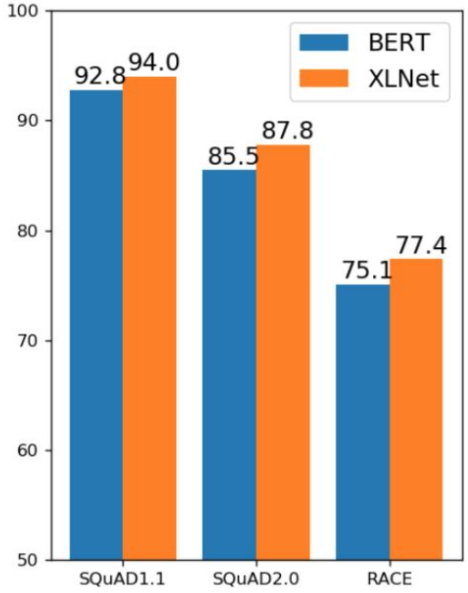
Comparison with BERT

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**

	MNLI	QNLI	QQP	RTE	SST	MRPC	CoLA	STS	WNLI	Avg
Single-task single models on dev										
$\text{BERT}_{\text{LARGE}}$	86.6/-	92.3	91.3	70.4	93.2	88.0	60.6	90.0	-	-
$\text{XLNet}_{\text{LARGE}}$	89.8/-	93.9	91.8	83.8	95.6	89.2	63.6	91.8	-	-
RoBERTa	**90.2/90.2**	**94.7**	**92.2**	**86.6**	**96.4**	**90.9**	**68.0**	**92.4**	**91.3**	-
Ensembles on test (from leaderboard as of July 25, 2019)										
ALICE	88.2/87.9	95.7	**90.7**	83.5	95.2	92.6	**68.6**	91.1	80.8	86.3
MT-DNN	87.9/87.4	96.0	89.9	86.3	96.5	92.7	68.4	91.1	89.0	87.6
XLNet	90.2/89.8	98.6	90.3	86.3	**96.8**	**93.0**	67.8	91.6	**90.4**	88.4
RoBERTa	**90.8/90.2**	**98.9**	90.2	**88.2**	96.7	92.3	67.8	**92.2**	89.0	**88.5**

	MNLI	QNLI	QQP	RTE	SST	MRPC	CoLA	STS	WNLI	Avg
Single-task single models on dev										
$\text{BERT}_{\text{LARGE}}$	86.6/-	92.3	91.3	70.4	93.2	88.0	60.6	90.0	-	-
$\text{XLNet}_{\text{LARGE}}$	89.8/-	93.9	91.8	83.8	95.6	89.2	63.6	91.8	-	-
RoBERTa	**90.2/90.2**	**94.7**	**92.2**	**86.6**	**96.4**	**90.9**	**68.0**	**92.4**	**91.3**	-
Ensembles on test (from leaderboard as of July 25, 2019)										
ALICE	88.2/87.9	95.7	**90.7**	83.5	95.2	92.6	**68.6**	91.1	80.8	86.3
MT-DNN	87.9/87.4	96.0	89.9	86.3	96.5	92.7	68.4	91.1	89.0	87.6
XLNet	90.2/89.8	98.6	90.3	86.3	**96.8**	**93.0**	67.8	91.6	**90.4**	88.4
RoBERTa	**90.8/90.2**	**98.9**	90.2	**88.2**	96.7	92.3	67.8	**92.2**	89.0	**88.5**

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	-
RoBERTa [21]	90.2/90.2	94.7	92.2	**86.6**	96.4	**90.9**	68.0	92.4	-
XLNet	**90.8/90.8**	**94.9**	**92.3**	85.9	**97.0**	90.8	**69.0**	**92.5**	-
Multi-task ensembles on test (from leaderboard as of Oct 28, 2019)									
MT-DNN* [20]	87.9/87.4	96.0	89.9	86.3	96.5	92.7	68.4	91.1	89.0
RoBERTa* [21]	90.8/90.2	98.9	90.2	88.2	96.7	92.3	67.8	92.2	89.0
XLNet*	**90.9/90.9**†	**99.0**†	**90.4**†	**88.5**	**97.1**†	**92.9**	**70.2**	**93.0**	**92.5**

	MNLI	QNLI	QQP	RTE	SST	MRPC	CoLA	STS	WNLI	Avg
Single-task single models on dev										
$\text{BERT}_{\text{LARGE}}$	86.6/-	92.3	91.3	70.4	93.2	88.0	60.6	90.0	-	-
$\text{XLNet}_{\text{LARGE}}$	89.8/-	93.9	91.8	83.8	95.6	89.2	63.6	91.8	-	-
RoBERTa	**90.2/90.2**	**94.7**	**92.2**	**86.6**	**96.4**	**90.9**	**68.0**	**92.4**	**91.3**	-
Ensembles on test (from leaderboard as of July 25, 2019)										
ALICE	88.2/87.9	95.7	**90.7**	83.5	95.2	92.6	**68.6**	91.1	80.8	86.3
MT-DNN	87.9/87.4	96.0	89.9	86.3	96.5	92.7	68.4	91.1	89.0	87.6
XLNet	90.2/89.8	98.6	90.3	86.3	**96.8**	**93.0**	67.8	91.6	**90.4**	88.4
RoBERTa	**90.8/90.2**	**98.9**	90.2	**88.2**	96.7	92.3	67.8	**92.2**	89.0	**88.5**

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	-
RoBERTa [21]	90.2/90.2	94.7	92.2	**86.6**	96.4	**90.9**	68.0	92.4	-
XLNet	**90.8/90.8**	**94.9**	**92.3**	85.9	**97.0**	90.8	**69.0**	**92.5**	-
Multi-task ensembles on test (from leaderboard as of Oct 28, 2019)									
MT-DNN* [20]	87.9/87.4	96.0	89.9	86.3	96.5	92.7	68.4	91.1	89.0
RoBERTa* [21]	90.8/90.2	98.9	90.2	88.2	96.7	92.3	67.8	92.2	89.0
XLNet*	**90.9/90.9**†	**99.0**†	**90.4**†	**88.5**	**97.1**†	**92.9**	**70.2**	**93.0**	**92.5**

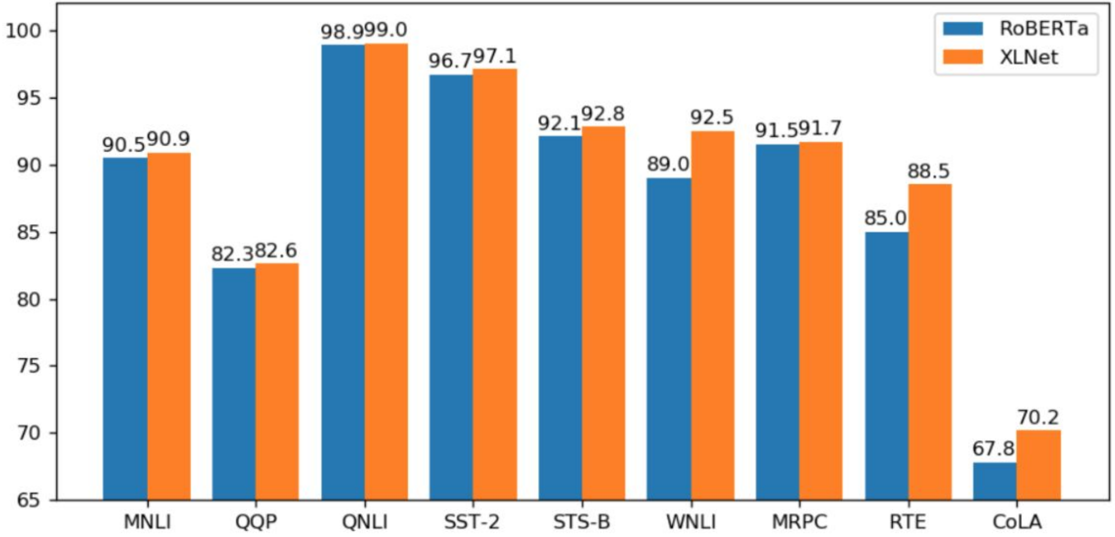
	MNLI	QNLI	QQP	RTE	SST	MRPC	CoLA	STS	WNLI	Avg
Single-task single models on dev										
$\text{BERT}_{\text{LARGE}}$	86.6/-	92.3	91.3	70.4	93.2	88.0	60.6	90.0	-	-
$\text{XLNet}_{\text{LARGE}}$	89.8/-	93.9	91.8	83.8	95.6	89.2	63.6	91.8	-	-
RoBERTa	**90.2/90.2**	**94.7**	**92.2**	**86.6**	**96.4**	**90.9**	**68.0**	**92.4**	**91.3**	-
Ensembles on test (from leaderboard as of July 25, 2019)										
ALICE	88.2/87.9	95.7	**90.7**	83.5	95.2	92.6	**68.6**	91.1	80.8	86.3
MT-DNN	87.9/87.4	96.0	89.9	86.3	96.5	92.7	68.4	91.1	89.0	87.6
XLNet	90.2/89.8	98.6	90.3	86.3	**96.8**	**93.0**	67.8	91.6	**90.4**	88.4
RoBERTa	**90.8/90.2**	**98.9**	90.2	**88.2**	96.7	92.3	67.8	**92.2**	89.0	**88.5**

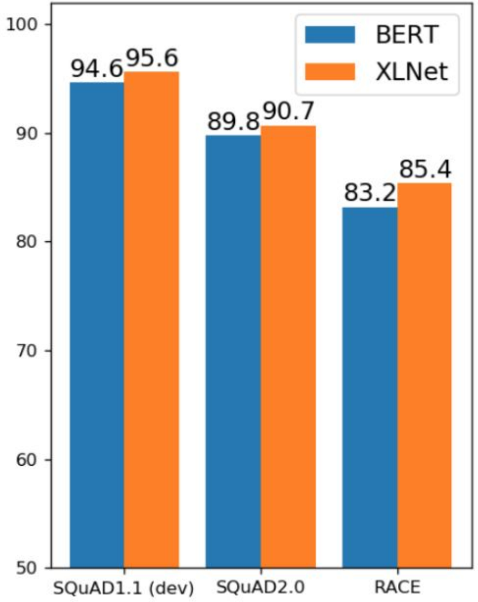
RoBERTa paper

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	-
RoBERTa [21]	90.2/90.2	94.7	92.2	**86.6**	96.4	**90.9**	68.0	92.4	-
XLNet	**90.8/90.8**	**94.9**	**92.3**	85.9	**97.0**	90.8	**69.0**	**92.5**	-
Multi-task ensembles on test (from leaderboard as of Oct 28, 2019)									
MT-DNN* [20]	87.9/87.4	96.0	89.9	86.3	96.5	92.7	68.4	91.1	89.0
RoBERTa* [21]	90.8/90.2	98.9	90.2	88.2	96.7	92.3	67.8	92.2	89.0
XLNet*	**90.9/90.9**†	**99.0**†	**90.4**†	**88.5**	**97.1**†	**92.9**	**70.2**	**93.0**	**92.5**

XLNet paper

XLNet outperforms RoBERTa on all considered tasks





Almost identical training recipes.

Ablation Study

#	Model	RACE	SQuAD2.0 F1	SQuAD2.0 EM	MNLI m/mm	SST-2
1	BERT-Base	64.3	76.30	73.66	84.34/84.65	92.78
2	DAE + Transformer-XL	65.03	79.56	76.80	84.88/84.45	92.60
3	XLNet-Base ($K = 7$)	66.05	**81.33**	**78.46**	**85.84/85.43**	92.66
4	XLNet-Base ($K = 6$)	66.66	80.98	78.18	85.63/85.12	**93.35**
5	- memory	65.55	80.15	77.27	85.32/85.05	92.78
6	- span-based pred	65.95	80.61	77.91	85.49/85.02	93.12
7	- bidirectional data	66.34	80.65	77.87	85.31/84.99	92.66
8	+ next-sent pred	**66.76**	79.83	76.94	85.32/85.09	92.89

Ablation Study

Transformer-XL and permutation LM contribute to the performance

#	Model	RACE	SQuAD2.0 F1	SQuAD2.0 EM	MNLI m/mm	SST-2
1	BERT-Base	64.3	76.30	73.66	84.34/84.65	92.78
2	DAE + Transformer-XL	65.03	79.56	76.80	84.88/84.45	92.60
3	XLNet-Base ($K = 7$)	66.05	**81.33**	**78.46**	**85.84/85.43**	92.66
4	XLNet-Base ($K = 6$)	66.66	80.98	78.18	85.63/85.12	**93.35**
5	- memory	65.55	80.15	77.27	85.32/85.05	92.78
6	- span-based pred	65.95	80.61	77.91	85.49/85.02	93.12
7	- bidirectional data	66.34	80.65	77.87	85.31/84.99	92.66
8	+ next-sent pred	**66.76**	79.83	76.94	85.32/85.09	92.89

Ablation Study

#	Model	RACE	SQuAD2.0 F1	SQuAD2.0 EM	MNLI m/mm	SST-2
1	BERT-Base	64.3	76.30	73.66	84.34/84.65	92.78
2	DAE + Transformer-XL	65.03	79.56	76.80	84.88/84.45	92.60
3	XLNet-Base ($K = 7$)	66.05	**81.33**	**78.46**	**85.84/85.43**	92.66
4	XLNet-Base ($K = 6$)	66.66	80.98	78.18	85.63/85.12	**93.35**
5	- memory	65.55	80.15	77.27	85.32/85.05	92.78
6	- span-based pred	65.95	80.61	77.91	85.49/85.02	93.12
7	- bidirectional data	66.34	[illegible]	[illegible]	[illegible]	[illegible]
8	+ next-sent pred	**66.76**	[illegible]	[illegible]	[illegible]	[illegible]

Memory caching is important. RACE involves longest contexts of the 4

Ablation Study

Bidirectional context is important

#	Model	RACE	SQuAD2.0 F1	SQuAD2.0 EM	MNLI m/mm	SST-2
1	[illegible]	64.3	76.30	73.66	84.34/84.65	92.78
2	DAE + Transformer-XL	65.03	79.56	76.80	84.88/84.45	92.60
3	XLNet-Base ($K = 7$)	66.05	**81.33**	**78.46**	**85.84/85.43**	92.66
4	XLNet-Base ($K = 6$)	66.66	80.98	78.18	85.63/85.12	**93.35**
5	- memory	65.55	80.15	77.27	85.32/85.05	92.78
6	- span-based pred	65.95	80.61	77.91	85.49/85.02	93.12
7	- bidirectional data	66.34	80.65	77.87	85.31/84.99	92.66
8	+ next-sent pred	**66.76**	79.83	76.94	85.32/85.09	92.89

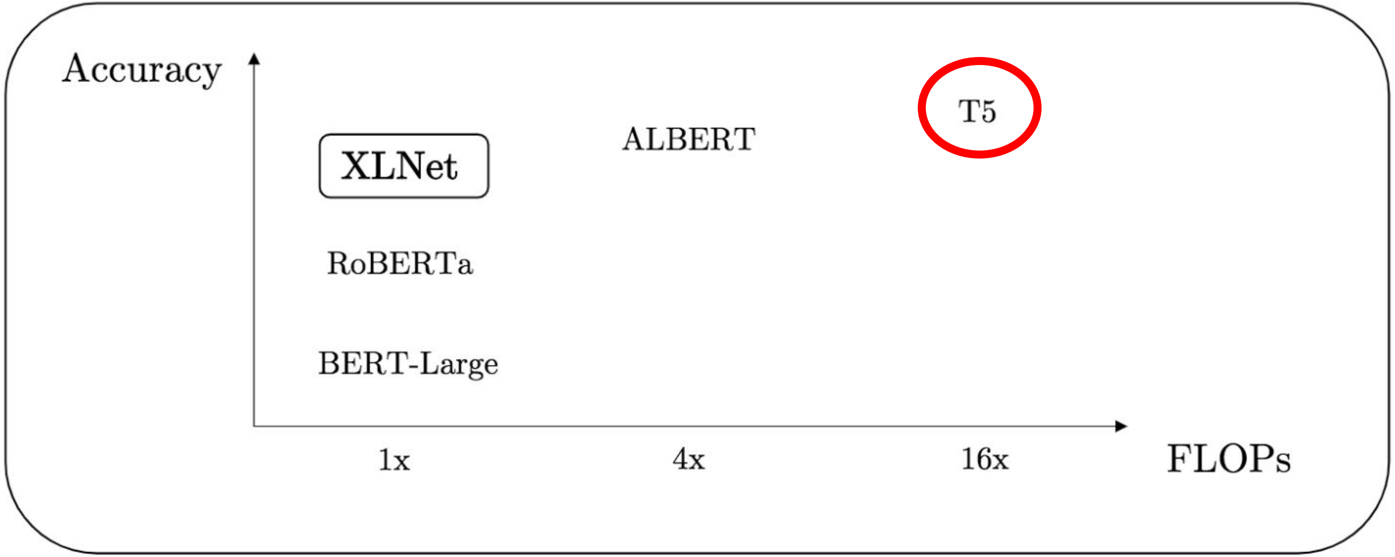
Ablation Study

Next-sentence prediction objective does not necessarily help

#	Model	RACE	SQuAD2.0 F1	SQuAD2.0 EM	MNLI m/mm	SST-2
[illegible]	[illegible]	[illegible]	76.30	73.66	84.34/84.65	92.78
[illegible]	[illegible]	[illegible]	79.56	76.80	84.88/84.45	92.60
3	XLNet-Base ($K = 7$)	66.05	**81.33**	**78.46**	**85.84/85.43**	92.66
4	XLNet-Base ($K = 6$)	66.66	80.98	78.18	85.63/85.12	**93.35**
5	- memory	65.55	80.15	77.27	85.32/85.05	92.78
6	- span-based pred	65.95	80.61	77.91	85.49/85.02	93.12
7	- bidirectional data	66.34	80.65	77.87	85.31/84.99	92.66
8	+ next-sent pred	**66.76**	79.83	76.94	85.32/85.09	92.89

XLNet is

The best pretrained model today
Given standard FLOPs.



Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer

Colin Raffel, Noam Shazeer*, Adam Roberts*, Katherine Lee*, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J. Liu*

Google AI

A systematic study of pre-training objectives, architectures, datasets, transfer approaches and other factors to create the best architecture. Results: T5 models - Base, Small, Large, 3B, 11B

T5

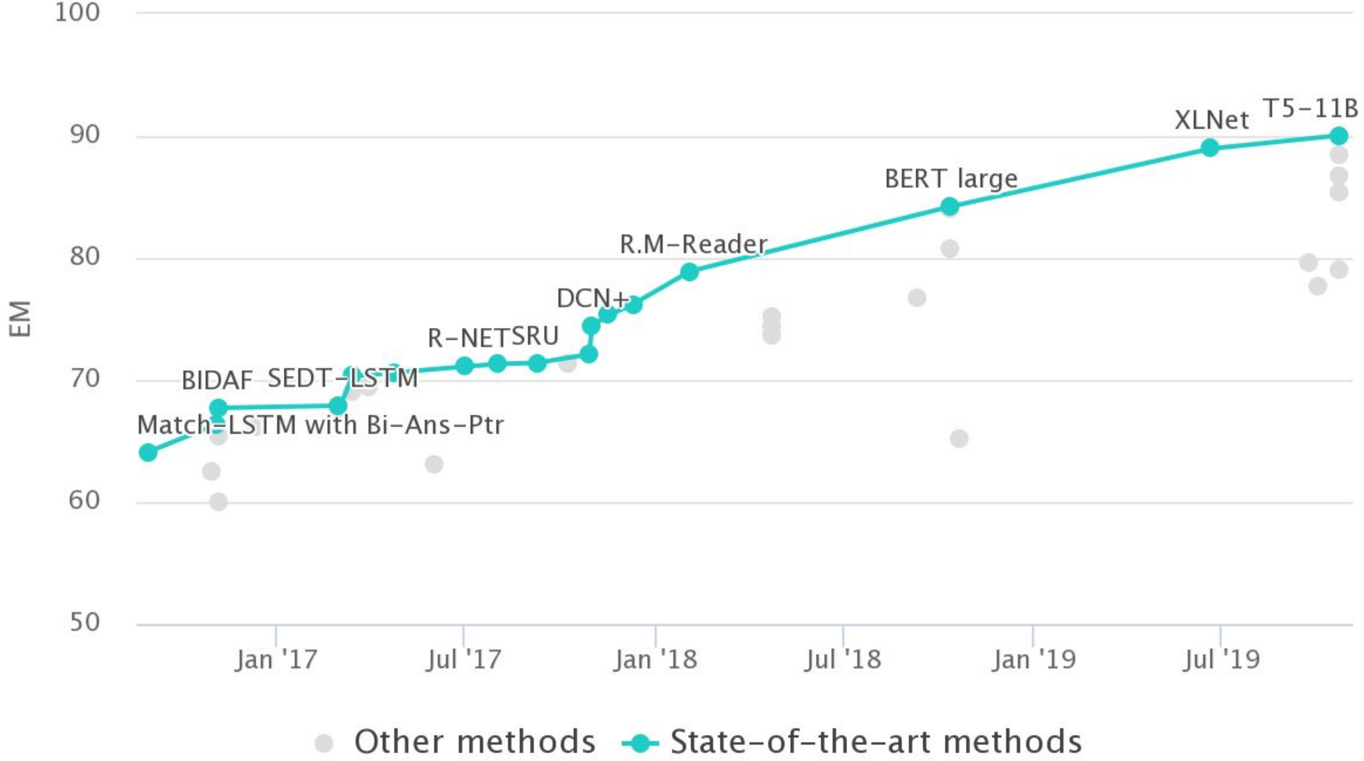
Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer

Colin Raffel, Noam Shazeer*, Adam Roberts*, Katherine Lee*, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J. Liu*

Google AI

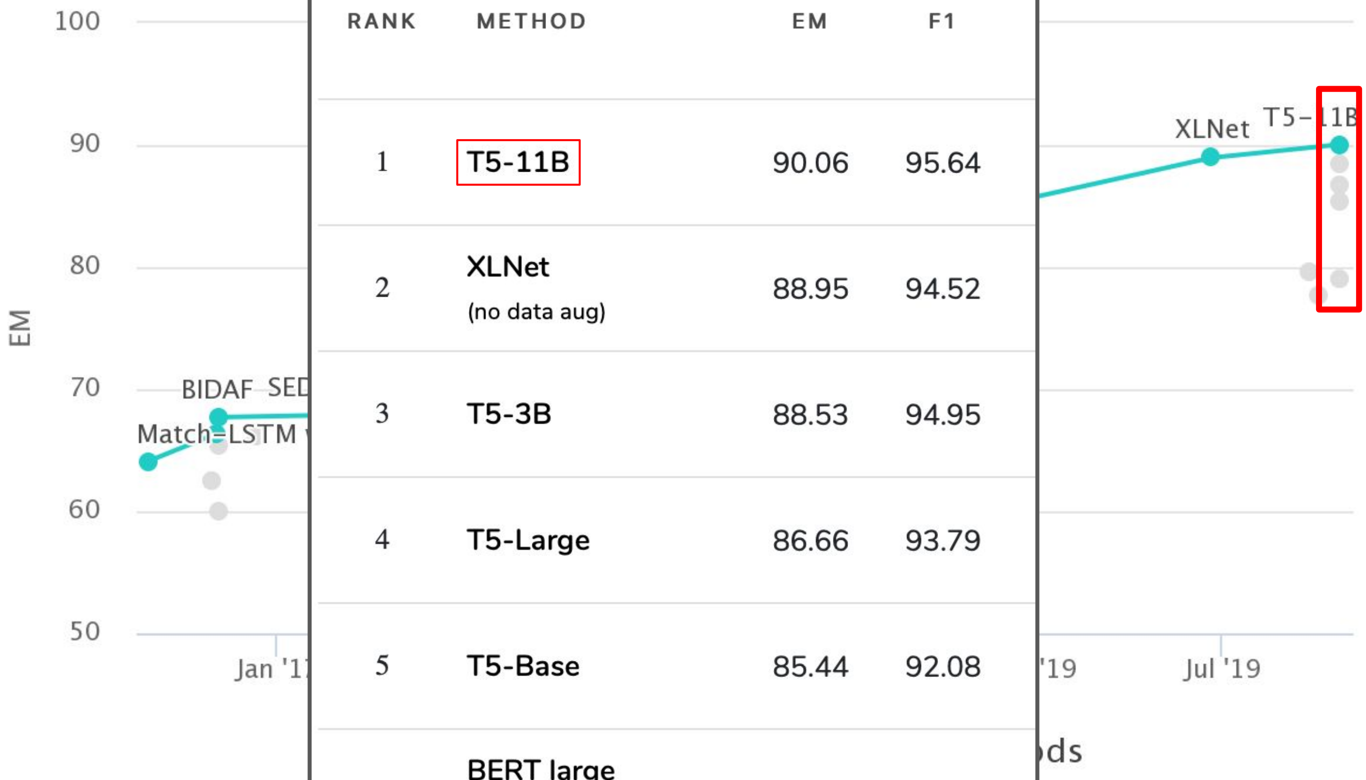
A systematic study of pre-training objectives, architectures, datasets, transfer approaches and other factors to create the best architecture. Results: T5 models - Base, Small, Large, 3B, 11B

Question Answering on SQuAD1.1 dev



From: paperswithcode.com/sota/

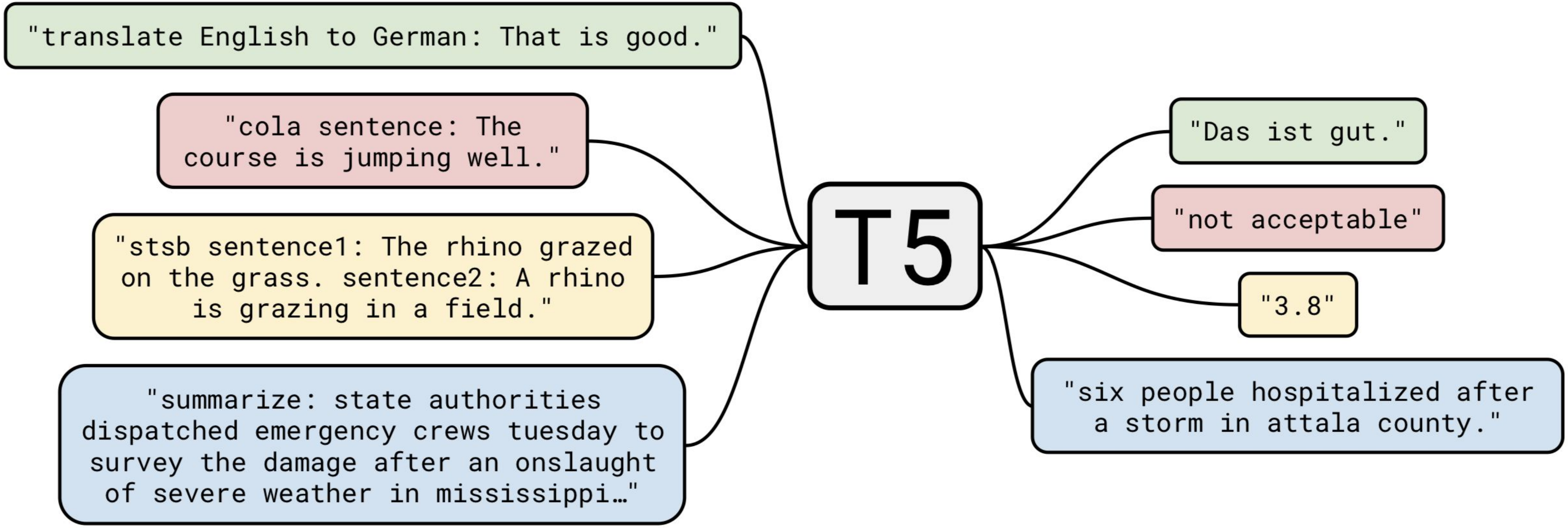
Question Answering on SQuAD1.1 dev



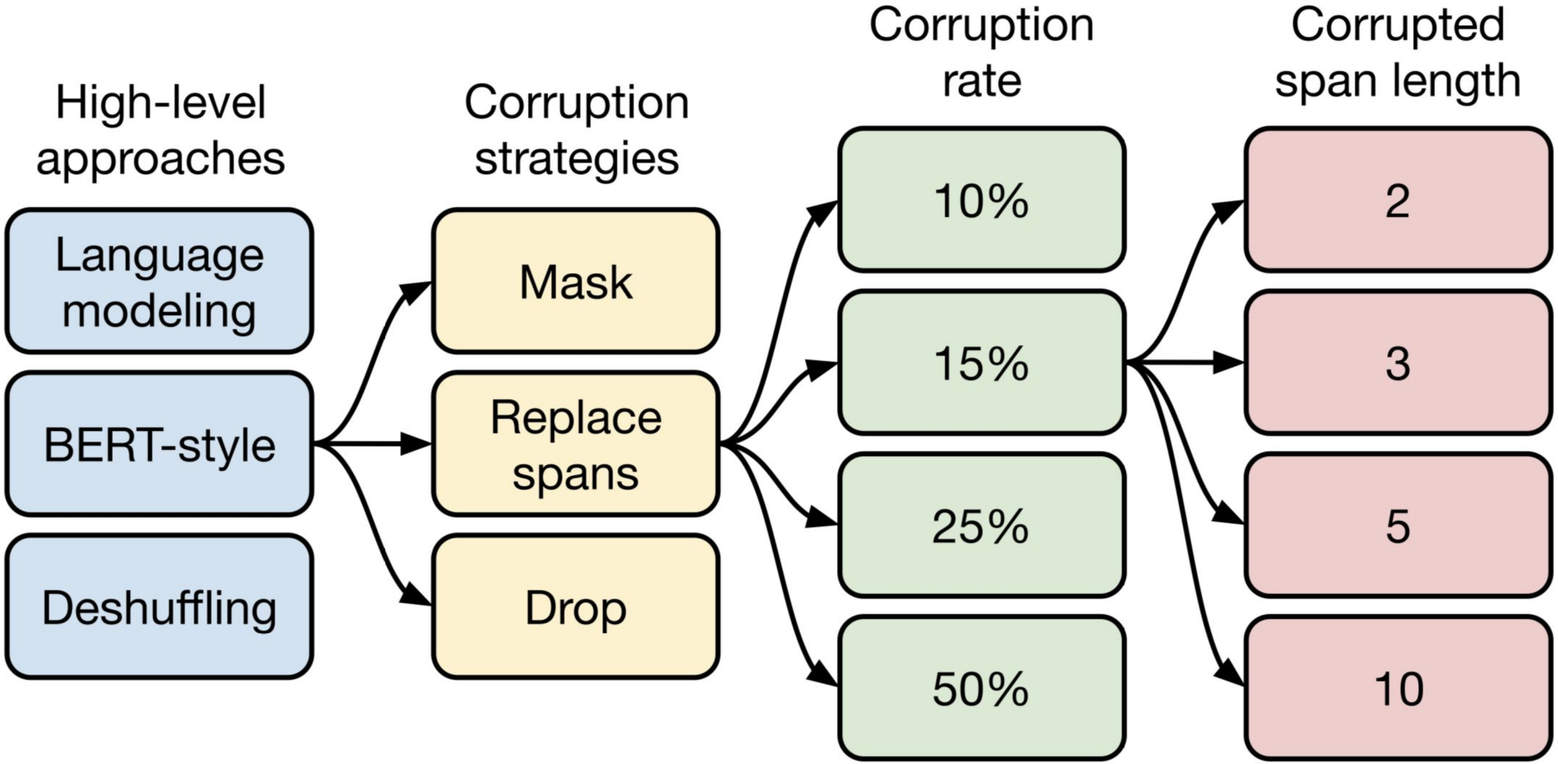
RANK	METHOD	EM	F1
1	T5-11B	90.06	95.64
2	XLNet (no data aug)	88.95	94.52
3	T5-3B	88.53	94.95
4	T5-Large	86.66	93.79
5	T5-Base	85.44	92.08
6	BERT large (+TriviaQA)	84.2	91.1

From: paperswithcode.com/sota/

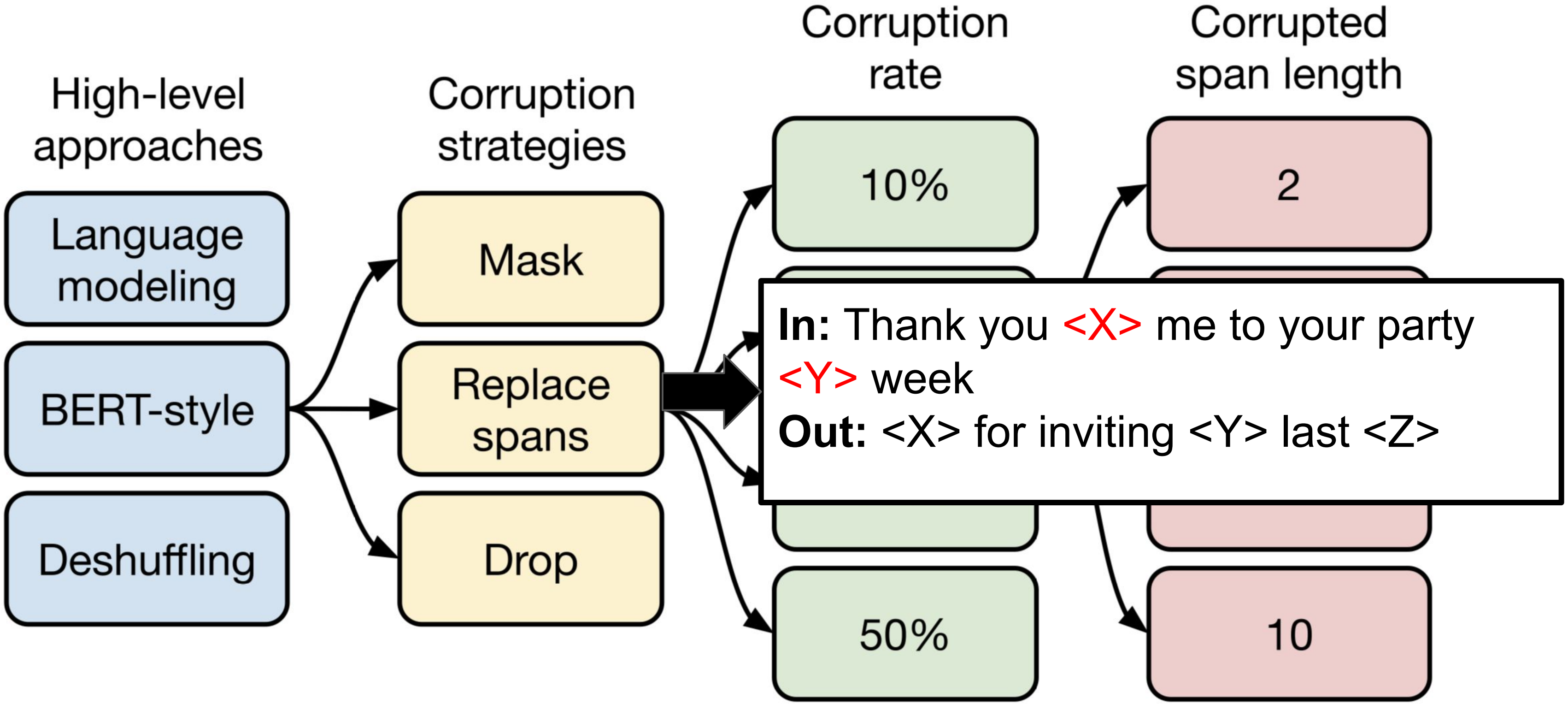
Text-to-text framework



Exploration of unsupervised objectives



Exploration of unsupervised objectives



Examples of the unsupervised objectives

Objective	Inputs	Targets
Prefix language modeling	Thank you for inviting	me to your party last week .
BERT-style	Thank you <M> <M> me to your party apple week .	*(original text)*
Deshuffling	party me for your to . last fun you inviting week Thank	*(original text)*
I.i.d. noise, mask tokens	Thank you <M> <M> me to your party <M> week .	*(original text)*
I.i.d. noise, replace spans	Thank you <X> me to your party <Y> week .	<X> for inviting <Y> last <Z>
I.i.d. noise, drop tokens	Thank you me to your party week .	for inviting last
Random spans	Thank you <X> to <Y> week .	<X> for inviting me <Y> your party last <Z>

Other parameters considered

- Datasets for pre-training
  - Introduce C4: cleaned Common Crawl dataset (745GB)
- Size of the pre-training dataset
- Training strategy
- Scaling: what to do with 4x compute resources

The best model based on experiments: T5

- Encoder-decoder Transformer architecture
- Span-corruption objective, mean span of 3 and corrupt 15%
- Increase number of pre-training steps and batch size
- Use C4 to avoid repetition (large dataset)
- Train 5 different sizes of the models
- Multi-task pre-training
- Fine-tune on individual GLUE/SuperGLUE
- Beam search for tasks with long sequences

Takeaways and insights

Systematic study can lead to an improved model

- Increasing the training time and/or model size improves the baseline
- Objectives that produce short target sequences are more computationally efficient
- Ensembling models that were only fine-tuned separately can give substantial performance and could be a cheaper mean of improving performance
- Pre-training on on-domain data can improve performance
- Updating all parameters during fine-tuning is the most effective but costly

References

Transformer-XL blog: https://ai.googleblog.com/2019/01/transformer-xl-unleashing-potential-of.html

Transformer paper: https://arxiv.org/abs/1706.03762

Transformer-XL paper: https://arxiv.org/abs/1

RoBERTa paper: https://arxiv.org/pdf/1907.11692.pdf901.02860

XLNet paper: https://arxiv.org/pdf/1906.08237.pdf

XLNet NeurIPS slides: https://github.com/zihangdai/xlnet/blob/master/misc/slides.pdf

XLNet blog post: https://towardsdatascience.com/what-is-xlnet-and-why-it-outperforms-bert-8d8fce710335

T5 paper: https://arxiv.org/pdf/1910.10683.pdf