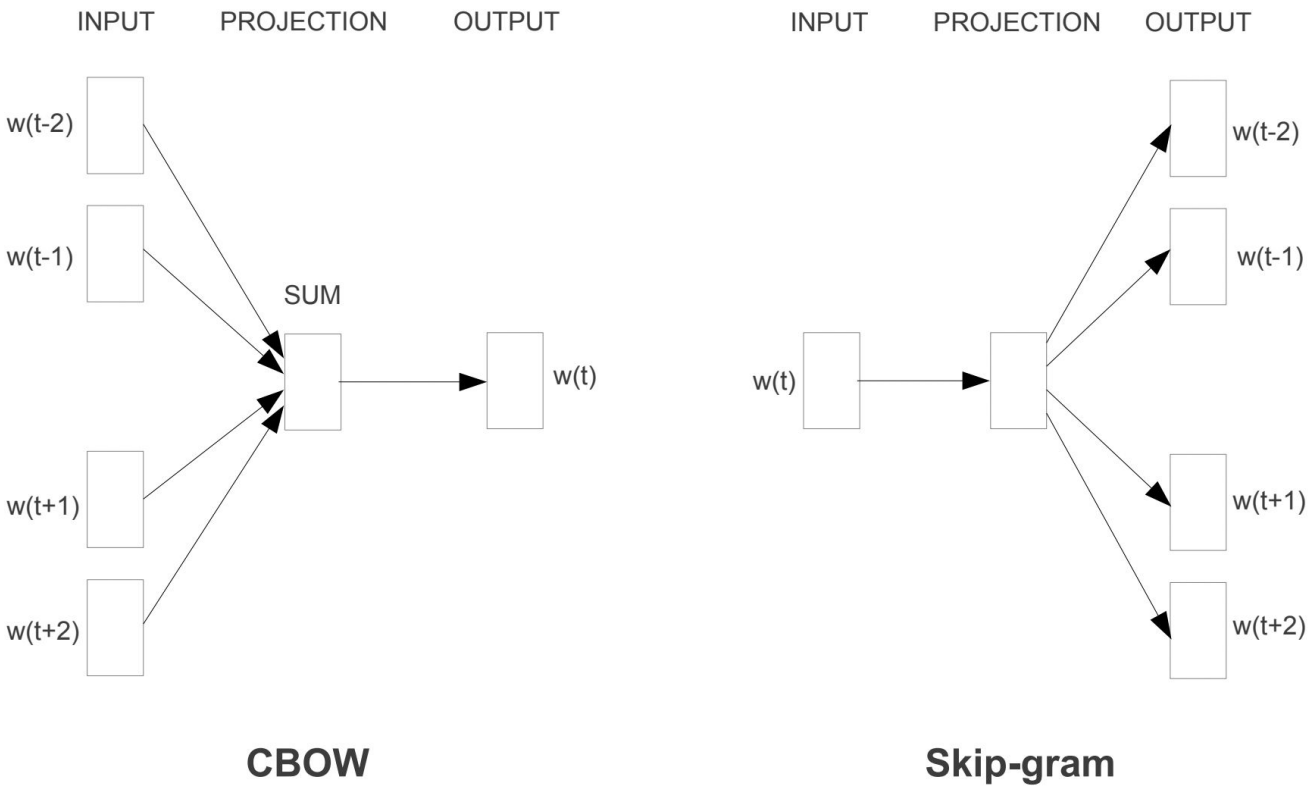
Предобучение языковых моделей

ВШЭ ФКН, Методы предобучения без учителя

Шабалин Александр


ELECTRA
RoBERTa
word2vec
GPT
Glove
ALBERT
DeBERTa
BERT
BART
CoVe
ELMo
MASS
XLM
T5

word2vec, 2013



https://arxiv.org/pdf/1301.3781.pdf

Downstream tasks

- NLI: **N**atural **L**anguage **I**nference (SNLI, MultiNLI, SciTail, RTE, …)
  Given a text and hypothesis need to determine the inference relation between them: *entailment*, *contradiction*, or *neutral*

- Question Answering (SQuAD, RACE, Story Cloze)
  Given a document and a question need to pick the right answer from the set, or to specify the beginning and ending of the answer from the document.

- Semantic Similarity (MRPC, QQP, STS-B)
  Say whether two sentences are semantically equivalent or not

- Classification (CoLA, SST-2, GLUE)

ELMo, 2018

Embeddings from Language Models

- All common methods use only last layer output
- ELMo Aggregates hidden states of LSTM
- Can be applied to an arbitrary LM

$$
\begin{aligned}
R_k &= \{\mathbf{x}_k^{LM}, \overrightarrow{\mathbf{h}}_{k,j}^{LM}, \overleftarrow{\mathbf{h}}_{k,j}^{LM} \mid j = 1, \ldots, L\} \\
&= \{\mathbf{h}_{k,j}^{LM} \mid j = 0, \ldots, L\},
\end{aligned}
$$

$$
\mathbf{ELMo}_k^{task} = E(R_k; \Theta^{task}) = \gamma^{task} \sum_{j=0}^{L} s_j^{task} \mathbf{h}_{k,j}^{LM}
$$

Pre-trains via next token prediction

https://arxiv.org/pdf/1802.05365.pdf

ELMo, 2018

Embeddings from Language Models

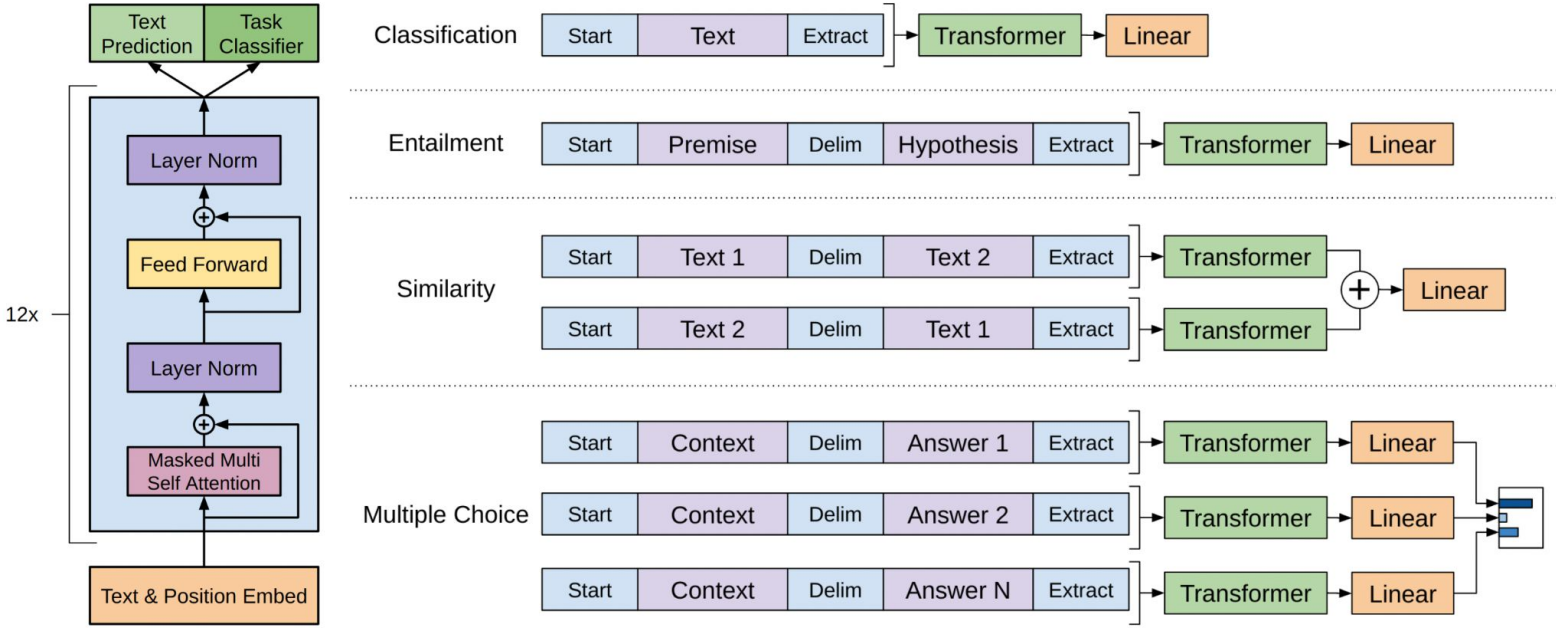
Task	Previous SOTA		Our baseline	ELMo + baseline	Increase (absolute/ relative)
SQuAD	Liu et al. (2017)	84.4	81.1	85.8	4.7 / 24.9%
SNLI	Chen et al. (2017)	88.6	88.0	88.7 ± 0.17	0.7 / 5.8%
SRL	He et al. (2017)	81.7	81.4	84.6	3.2 / 17.2%
Coref	Lee et al. (2017)	67.2	67.2	70.4	3.2 / 9.8%
NER	Peters et al. (2017)	91.93 ± 0.19	90.15	92.22 ± 0.10	2.06 / 21%
SST-5	McCann et al. (2017)	53.7	51.4	54.7 ± 0.5	3.3 / 6.8%

https://arxiv.org/pdf/1802.05365.pdf

GPT, 2018 (OpenAI)

Generative Pre-Training

- Stack of Transformer decoder layers
- Pre-training – next word prediction

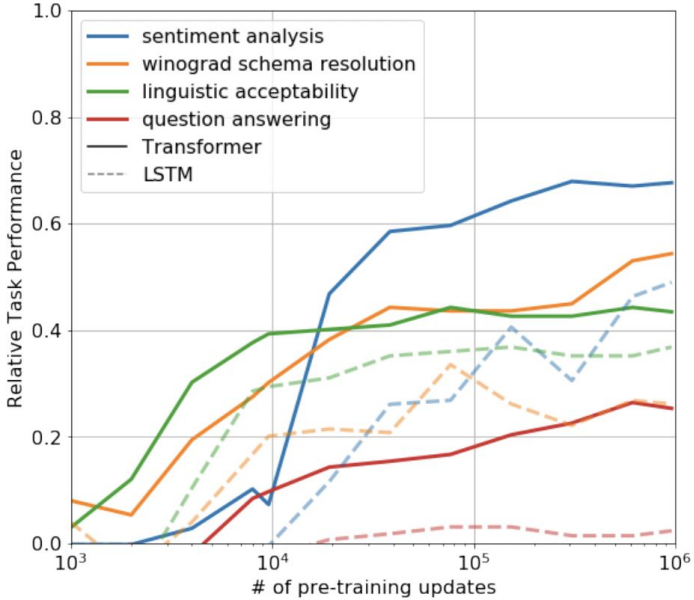


https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf

GPT, 2018 (OpenAI)

Generative Pre-Training

zero-shot



GPT performed better than specifically trained supervised state-of-the-art models in *9 out of 12 tasks* the models were compared on.

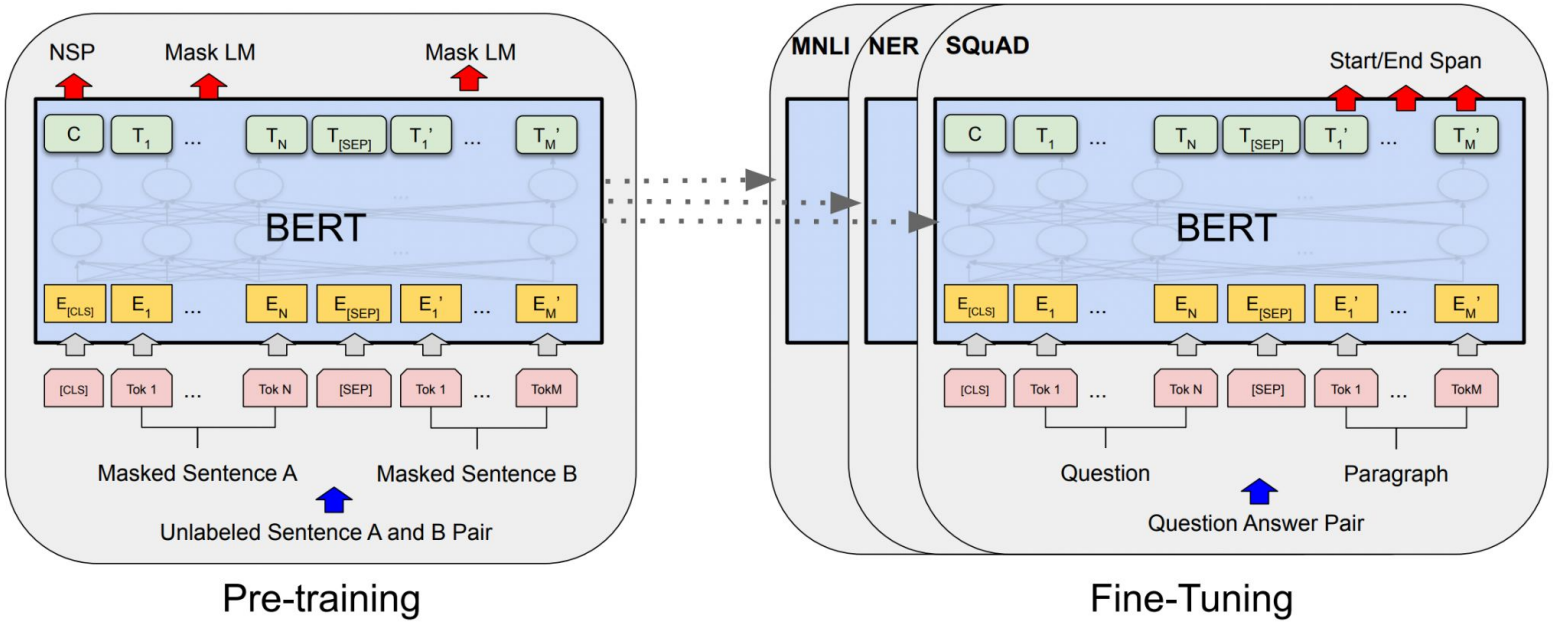
Method	Avg. Score	CoLA (mc)	SST2 (acc)	MRPC (F1)	STSB (pc)	QQP (F1)	MNLI (acc)	QNLI (acc)	RTE (acc)
Transformer w/ aux LM (full)	74.7	45.4	91.3	82.3	82.0	**70.3**	**81.8**	**88.1**	**56.0**
Transformer w/o pre-training	59.9	18.9	84.0	79.4	30.9	65.5	75.7	71.2	53.8
Transformer w/o aux LM	**75.0**	**47.9**	**92.0**	**84.9**	**83.2**	69.8	81.1	86.9	54.4
LSTM w/ aux LM	69.1	30.3	90.5	83.2	71.8	68.1	73.7	81.1	54.6

https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf

BERT, 2018 (Google)

Bidirectional Encoder Representations from Transformers

- **M**asked **L**anguage **M**odel – from 15% of tokens:
  80% masked, 10% replaced, 10% unchanged
- **N**ext **S**entence **P**rediction – needed for downstream tasks (QA, NLI)



https://arxiv.org/pdf/1810.04805.pdf

BERT, 2018 (Google)

Bidirectional Encoder Representations from Transformers

System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	**Average** -
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**92.7**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**82.1**

https://arxiv.org/pdf/1810.04805.pdf

XLNet, 2019 (Google)

BERT's limitations:

- Pretrain-finetune discrepancy because of masking
- Loss of dependency between the masked positions

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

XLNet, 2019 (Google)

BERT's limitations:

- Pretrain-finetune discrepancy because of masking
- Loss of dependency between the masked positions

XLNet estimates the probability of a word token conditioned on *all permutations* of word tokens in a sentence using **Transformer-decoder**

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

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

XLNet, 2019 (Google)

BERT's limitations:

- Pretrain-finetune discrepancy because of masking
- Loss of dependency between the masked positions

XLNet estimates the probability of a word token conditioned on *all permutations* of word tokens in a sentence using **Transformer-decoder**

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

Input: $[\text{New}, \text{York}, \text{is}, \text{a}, \text{city}]$ Permutation: $[\text{is}, \text{a}, \text{city}, \text{New}, \text{York}]$

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

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

What is wrong with this objective?

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

XLNet, 2019 (Google)

BERT's limitations:

- Pretrain-finetune discrepancy because of masking
- Loss of dependency between the masked positions

XLNet estimates the probability of a word token conditioned on *all permutations* of word tokens in a sentence using **Transformer-decoder**

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

Input: $[\text{New}, \text{York}, \text{is}, \text{a}, \text{city}]$ Permutation: $[\text{is}, \text{a}, \text{city}, \text{New}, \text{York}]$

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

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

What is wrong with this objective? It doesn't consider the position of the target token!

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

XLNet, 2019 (Google)

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

$$h_i^{(0)}=e(x_i)$$
$$g_i^{(0)}=w$$

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

XLNet, 2019 (Google)

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

$$h_i^{(0)} = e(x_i)$$
$$g_i^{(0)} = w$$

$$\text{Attention}(Q,K,V)=\text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

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

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

XLNet, 2019 (Google)

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

$$h_i^{(0)}=e(x_i)$$
$$g_i^{(0)}=w$$

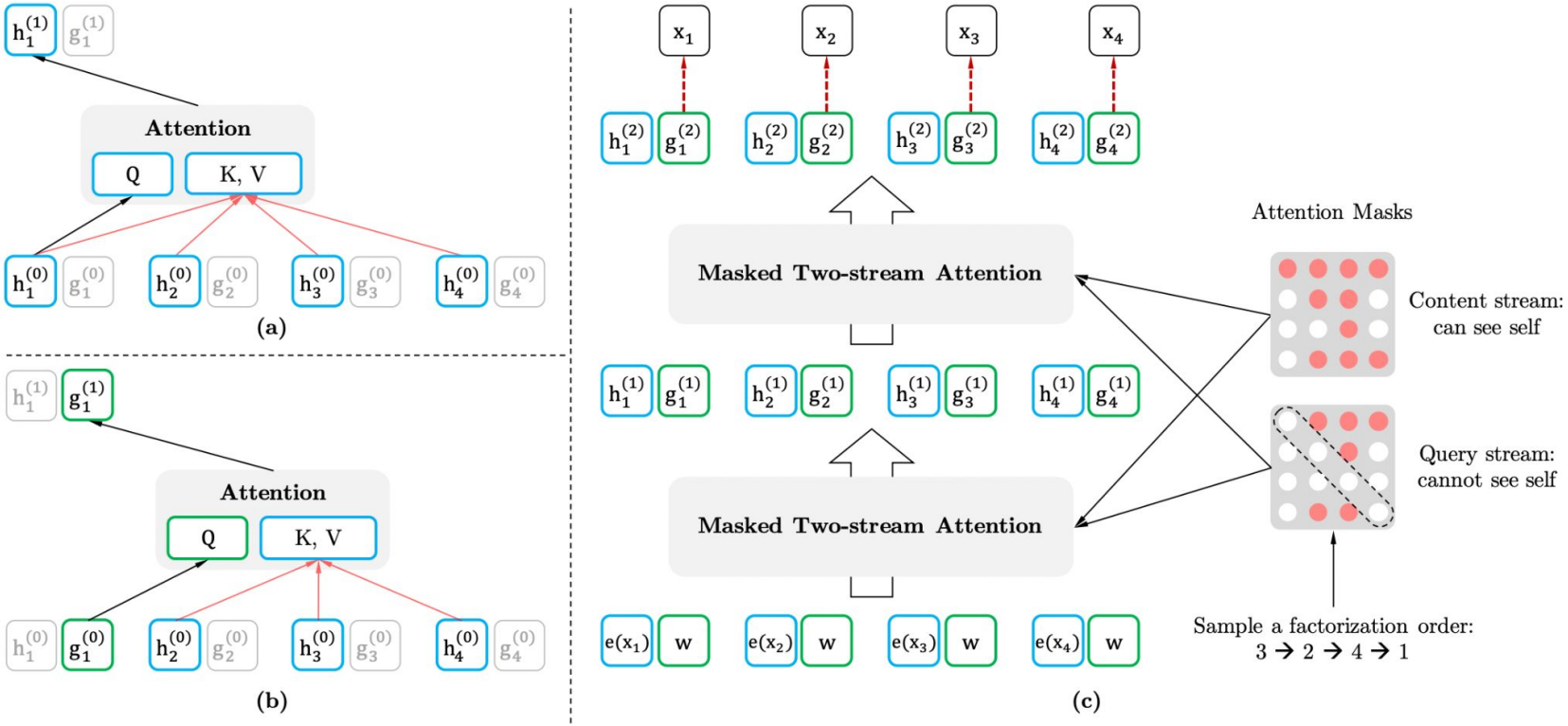
$$\text{Attention}(Q,K,V)=\text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

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

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

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

XLNet, 2019 (Google)



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

XLNet, 2019 (Google)

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**

SQuAD2.0	EM	F1	SQuAD1.1	EM	F1
Dev set results (single model)					
BERT [10]	78.98	81.77	BERT† [10]	84.1	90.9
RoBERTa [21]	86.5	89.4	RoBERTa [21]	88.9	94.6
XLNet	**87.9**	**90.6**	XLNet	**89.7**	**95.1**
Test set results on leaderboard (single model, as of Dec 14, 2019)					
BERT [10]	80.005	83.061	BERT [10]	85.083	91.835
RoBERTa [21]	86.820	89.795	BERT* [10]	87.433	93.294
XLNet	**87.926**	**90.689**	XLNet	**89.898**‡	**95.080**‡

Pre-training data:

- BookCorpus + Wikipedia (13GB)
- Giga5 (16GB)
- Common Crawl (110GB)
- ClueWeb 2012-B (19GB)

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

RoBERTa, 2019 (Facebook)

Robustly Optimized BERT Approach

- Train the model longer
- Use bigger batches
- Collect more data
- Dynamic masking
- Without NSP

Masking	SQuAD 2.0	MNLI-m	SST-2
reference	76.3	84.3	92.8
Our reimplementation:			
static	78.3	84.3	92.5
dynamic	78.7	84.0	92.9

https://arxiv.org/pdf/1907.11692.pdf

RoBERTa, 2019 (Facebook)

Robustly Optimized BERT Approach

- Train the model longer
- Use bigger batches
- Collect more data
- Dynamic masking
- Without NSP

Masking	SQuAD 2.0	MNLI-m	SST-2
reference	76.3	84.3	92.8
Our reimplementation:			
static	78.3	84.3	92.5
dynamic	78.7	84.0	92.9

bsz	steps	lr	ppl	MNLI-m	SST-2
256	1M	1e-4	3.99	84.7	92.7
2K	125K	7e-4	**3.68**	**85.2**	**92.9**
8K	31K	1e-3	3.77	84.6	92.8

https://arxiv.org/pdf/1907.11692.pdf

RoBERTa, 2019 (Facebook)

Robustly Optimized BERT Approach

- Train the model longer
- Use bigger batches
- Collect more data
- Dynamic masking
- Without NSP

Masking	SQuAD 2.0	MNLI-m	SST-2
reference	76.3	84.3	92.8
Our reimplementation:			
static	78.3	84.3	92.5
dynamic	78.7	84.0	92.9

bsz	steps	lr	ppl	MNLI-m	SST-2
256	1M	1e-4	3.99	84.7	92.7
2K	125K	7e-4	**3.68**	**85.2**	**92.9**
8K	31K	1e-3	3.77	84.6	92.8

Model	SQuAD 1.1/2.0	MNLI-m	SST-2	RACE
Our reimplementation (with NSP loss):				
SEGMENT-PAIR	90.4/78.7	84.0	92.9	64.2
SENTENCE-PAIR	88.7/76.2	82.9	92.1	63.0
Our reimplementation (without NSP loss):				
FULL-SENTENCES	90.4/79.1	84.7	92.5	64.8
DOC-SENTENCES	90.6/79.7	84.7	92.7	65.6
$\text{BERT}_{\text{BASE}}$	88.5/76.3	84.3	92.8	64.3
$\text{XLNet}_{\text{BASE}}$ (K = 7)	–/81.3	85.8	92.7	66.1
$\text{XLNet}_{\text{BASE}}$ (K = 6)	–/81.0	85.6	93.4	66.7

https://arxiv.org/pdf/1907.11692.pdf

RoBERTa, 2019 (Facebook)

Robustly Optimized BERT Approach

- BookCorpus + Wikipedia (16GB)
- CC-News (76GB)
- OpenWebText (38GB)
- Stories (31GB)

Model	data	bsz	steps	SQuAD (v1.1/2.0)	MNLI-m	SST-2
RoBERTa						
with BOOKS + WIKI	16GB	8K	100K	93.6/87.3	89.0	95.3
+ additional data (§3.2)	160GB	8K	100K	94.0/87.7	89.3	95.6
+ pretrain longer	160GB	8K	300K	94.4/88.7	90.0	96.1
+ pretrain even longer	160GB	8K	500K	**94.6/89.4**	**90.2**	**96.4**
$\text{BERT}_{\text{LARGE}}$						
with BOOKS + WIKI	13GB	256	1M	90.9/81.8	86.6	93.7
$\text{XLNet}_{\text{LARGE}}$						
with BOOKS + WIKI	13GB	256	1M	94.0/87.8	88.4	94.4
+ additional data	126GB	2K	500K	94.5/88.8	89.8	95.6

https://arxiv.org/pdf/1907.11692.pdf

RoBERTa, 2019 (Facebook)

Robustly Optimized BERT Approach

	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**

https://arxiv.org/pdf/1907.11692.pdf

ALBERT, 2019 (Google)

A Lite BERT

- We can improve quality by scaling a model, but it requires higher computational costs.
- ALBERT propose embedding matrix factorization and parameter sharing to make model lighter.
- Replace NSP loss with SOP loss.

https://arxiv.org/pdf/1909.11942.pdf

ALBERT, 2019 (Google)

A Lite BERT

- We can improve quality by scaling a model, but it requires higher computational costs.
- ALBERT propose embedding matrix factorization and parameter sharing to make model lighter.
- Replace NSP loss with SOP loss.

$$x \in OHE(V)$$

$$emb(x) = x \underbrace{A}_{[V,H]}$$

$$emb_{fact}(x) = x \underbrace{A}_{[V,E]} \underbrace{B}_{[E,H]}$$

https://arxiv.org/pdf/1909.11942.pdf

ALBERT, 2019 (Google)

A Lite BERT

- We can improve quality by scaling a model, but it requires higher computational costs.
- ALBERT propose embedding matrix factorization and parameter sharing to make model lighter.
- Replace NSP loss with SOP loss.

$$x \in OHE(V)$$

$$emb(x) = x \underbrace{A}_{[V,H]}$$

$$emb_{fact}(x) = x \underbrace{A}_{[V,E]} \underbrace{B}_{[E,H]}$$

Model		Parameters	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg
ALBERT base E=768	all-shared	31M	88.6/81.5	79.2/76.6	82.0	90.6	63.3	79.8
	shared-attention	83M	89.9/82.7	80.0/77.2	84.0	91.4	67.7	81.6
	shared-FFN	57M	89.2/82.1	78.2/75.4	81.5	90.8	62.6	79.5
	not-shared	108M	90.4/83.2	80.4/77.6	84.5	92.8	68.2	82.3
ALBERT base E=128	all-shared	12M	89.3/82.3	80.0/77.1	82.0	90.3	64.0	80.1
	shared-attention	64M	89.9/82.8	80.7/77.9	83.4	91.9	67.6	81.7
	shared-FFN	38M	88.9/81.6	78.6/75.6	82.3	91.7	64.4	80.2
	not-shared	89M	89.9/82.8	80.3/77.3	83.2	91.5	67.9	81.6

https://arxiv.org/pdf/1909.11942.pdf

ALBERT, 2019 (Google)

A Lite BERT

- Next Sentence Prediction task is too easy compared to MLM and, hence, useless.
- Idea is to replace if with Sentence Order Prediction (i. e. predict if sentences are swapped).

	Intrinsic Tasks			Downstream Tasks					
SP tasks	MLM	NSP	SOP	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg
None	54.9	52.4	53.3	88.6/81.5	78.1/75.3	81.5	89.9	61.7	79.0
NSP	54.5	90.5	52.0	88.4/81.5	77.2/74.6	81.6	**91.1**	62.3	79.2
SOP	54.0	78.9	86.5	**89.3/82.3**	**80.0/77.1**	**82.0**	90.3	**64.0**	**80.1**

Model		Parameters	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg	Speedup
BERT	base	108M	90.4/83.2	80.4/77.6	84.5	92.8	68.2	82.3	4.7x
	large	334M	92.2/85.5	85.0/82.2	86.6	93.0	73.9	85.2	1.0
ALBERT	base	12M	89.3/82.3	80.0/77.1	81.6	90.3	64.0	80.1	5.6x
	large	18M	90.6/83.9	82.3/79.4	83.5	91.7	68.5	82.4	1.7x
	xlarge	60M	92.5/86.1	86.1/83.1	86.4	92.4	74.8	85.5	0.6x
	xxlarge	235M	**94.1/88.3**	**88.1/85.1**	**88.0**	**95.2**	**82.3**	**88.7**	0.3x

ALBERT, 2019 (Google)

A Lite BERT

	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg
With dropout	94.7/89.2	89.6/86.9	90.0	96.3	85.7	90.4
Without dropout	**94.8/89.5**	**89.9/87.2**	**90.4**	**96.5**	**86.1**	**90.7**

Model	*E*	Parameters	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg
ALBERT base not-shared	64	87M	89.9/82.9	80.1/77.8	82.9	91.5	66.7	81.3
	128	89M	89.9/82.8	80.3/77.3	83.7	91.5	67.9	81.7
	256	93M	90.2/83.2	80.3/77.4	84.1	91.9	67.3	81.8
	768	108M	90.4/83.2	80.4/77.6	84.5	92.8	68.2	82.3
ALBERT base all-shared	64	10M	88.7/81.4	77.5/74.8	80.8	89.4	63.5	79.0
	128	12M	89.3/82.3	80.0/77.1	81.6	90.3	64.0	80.1
	256	16M	88.8/81.5	79.1/76.3	81.5	90.3	63.4	79.6
	768	31M	88.6/81.5	79.2/76.6	82.0	90.6	63.3	79.8

ALBERT, 2019 (Google)

A Lite BERT

Models	MNLI	QNLI	QQP	RTE	SST	MRPC	CoLA	STS	WNLI	Avg
Single-task single models on dev										
BERT-large	86.6	92.3	91.3	70.4	93.2	88.0	60.6	90.0	-	-
XLNet-large	89.8	93.9	91.8	83.8	95.6	89.2	63.6	91.8	-	-
RoBERTa-large	90.2	94.7	**92.2**	86.6	96.4	**90.9**	68.0	92.4	-	-
ALBERT (1M)	90.4	95.2	92.0	88.1	96.8	90.2	68.7	92.7	-	-
ALBERT (1.5M)	**90.8**	**95.3**	**92.2**	**89.2**	**96.9**	**90.9**	**71.4**	**93.0**	-	-
Ensembles on test (from leaderboard as of Sept. 16, 2019)										
ALICE	88.2	95.7	**90.7**	83.5	95.2	92.6	**69.2**	91.1	80.8	87.0
MT-DNN	87.9	96.0	89.9	86.3	96.5	92.7	68.4	91.1	89.0	87.6
XLNet	90.2	98.6	90.3	86.3	96.8	93.0	67.8	91.6	90.4	88.4
RoBERTa	90.8	98.9	90.2	88.2	96.7	92.3	67.8	92.2	89.0	88.5
Adv-RoBERTa	91.1	98.8	90.3	88.7	96.8	93.1	68.0	92.4	89.0	88.8
ALBERT	**91.3**	**99.2**	90.5	**89.2**	**97.1**	**93.4**	69.1	**92.5**	**91.8**	**89.4**

MASS, 2019 (Microsoft)

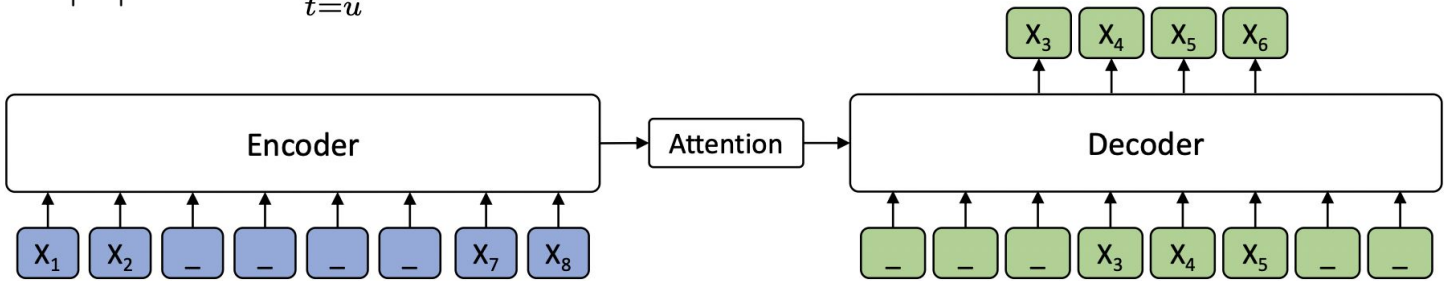
MAsked Sequence to Sequence

- Mask a continuous span of tokens and unmask with vanilla **(encoder-decoder)** Transformer
- Beneficial for sequence to sequence based language generation tasks

$$L(\theta; \mathcal{X}) = \frac{1}{|\mathcal{X}|} \Sigma_{x \in \mathcal{X}} \log P(x^{u:v} | x^{\setminus u:v}; \theta)$$

$$= \frac{1}{|\mathcal{X}|} \Sigma_{x \in \mathcal{X}} \log \prod_{t=u}^{v} P(x_t^{u:v} | x_{<t}^{u:v}, x^{\setminus u:v}; \theta)$$

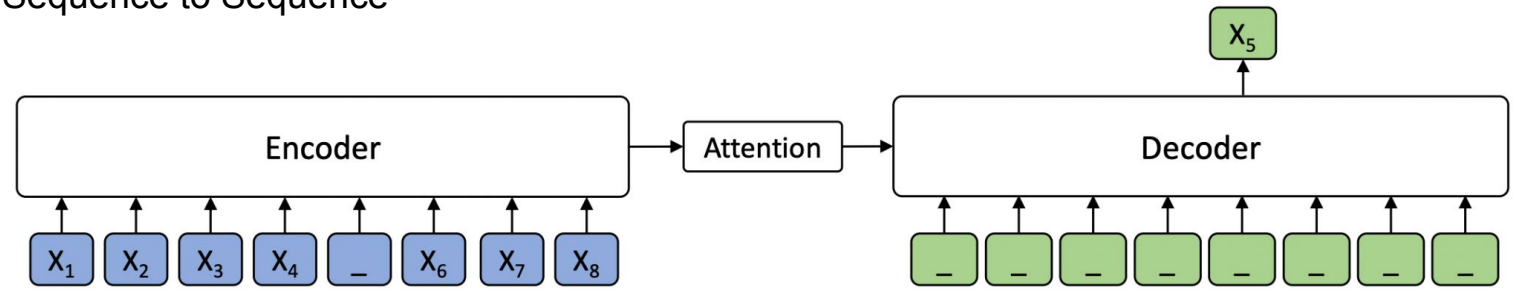
Mask approx. **50%** of a sequence



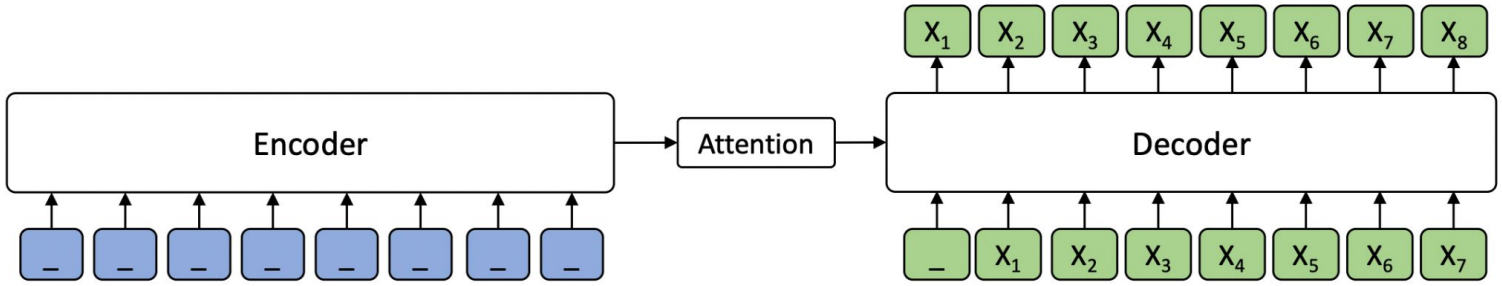
https://arxiv.org/pdf/1905.02450.pdf

MASS, 2019 (Microsoft)

MAsked Sequence to Sequence



(a) Masked language modeling in BERT ($k = 1$)

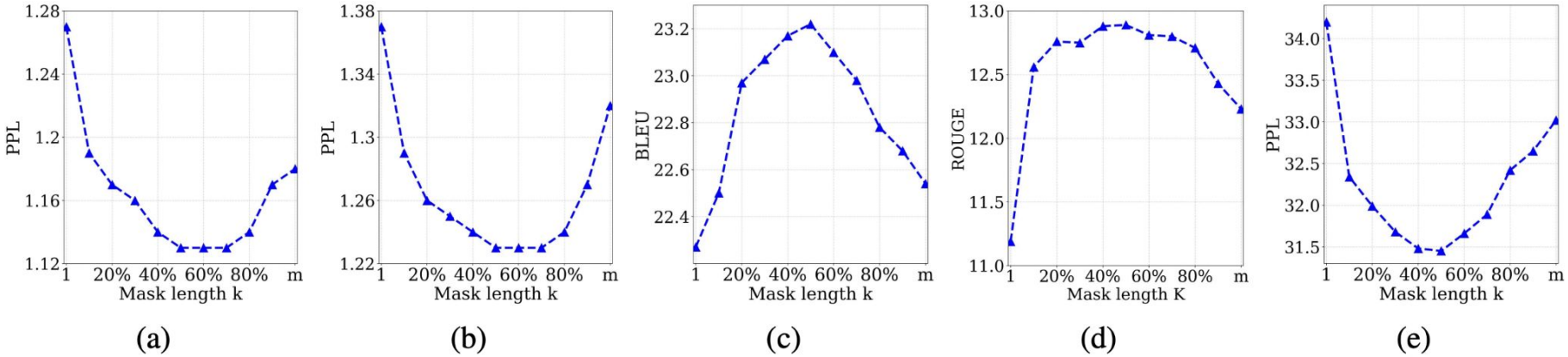


(b) Standard language modeling ($k = m$)

https://arxiv.org/pdf/1905.02450.pdf

MASS, 2019 (Microsoft)

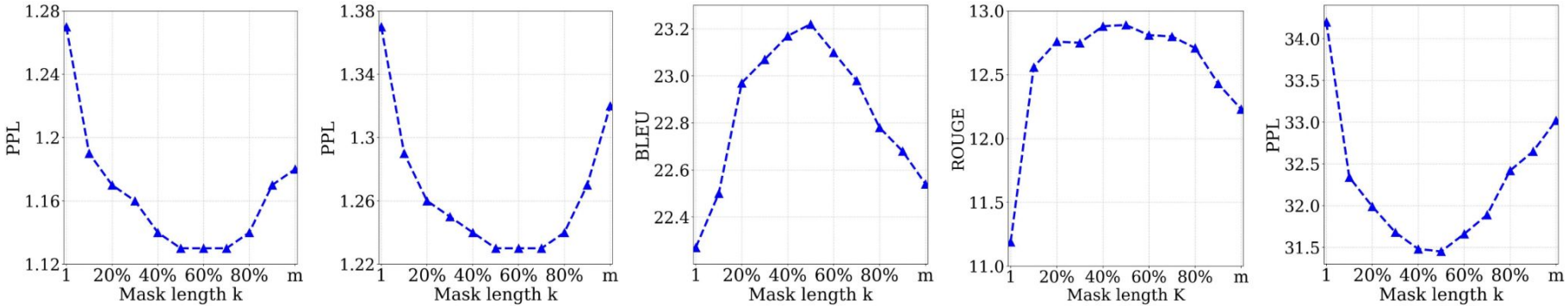
MAsked Sequence to Sequence



https://arxiv.org/pdf/1905.02450.pdf

MASS, 2019 (Microsoft)

MAsked Sequence to Sequence



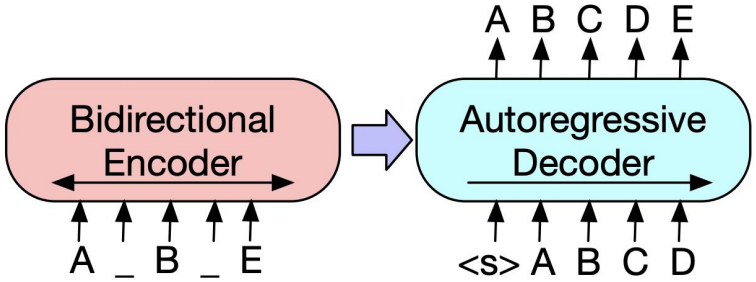
Method	Setting	en - fr	fr - en	en - de	de - en	en - ro	ro - en
Artetxe et al. (2017)	2-layer RNN	15.13	15.56	6.89	10.16	-	-
Lample et al. (2017)	3-layer RNN	15.05	14.31	9.75	13.33	-	-
Yang et al. (2018)	4-layer Transformer	16.97	15.58	10.86	14.62	-	-
Lample et al. (2018)	4-layer Transformer	25.14	24.18	17.16	21.00	21.18	19.44
XLM (Lample & Conneau, 2019)	6-layer Transformer	33.40	33.30	27.00	34.30	33.30	31.80
MASS	6-layer Transformer	**37.50**	**34.90**	**28.30**	**35.20**	**35.20**	**33.10**

https://arxiv.org/pdf/1905.02450.pdf

BART, 2019 (Facebook)

Bidirectional and Auto-Regressive Transformers

- A denoising autoencoder for pretraining seq2seq models.
- Attempts to connect BERT (due to the bidirectional encoder) with GPT (with the left-to-right decoder).

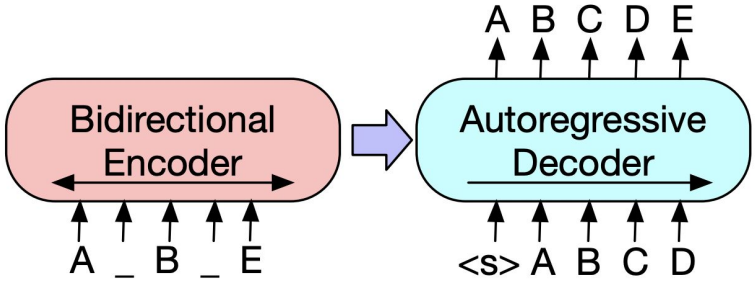


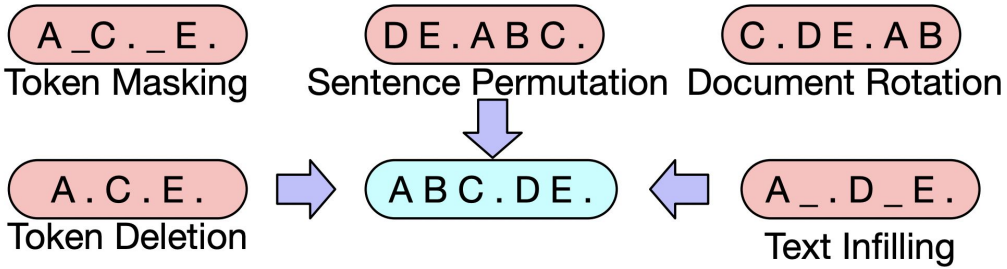
https://arxiv.org/pdf/1910.13461.pdf

BART, 2019 (Facebook)

Bidirectional and Auto-Regressive Transformers

- A denoising autoencoder for pretraining seq2seq models.
- Attempts to connect BERT (due to the bidirectional encoder) with GPT (with the left-to-right decoder).



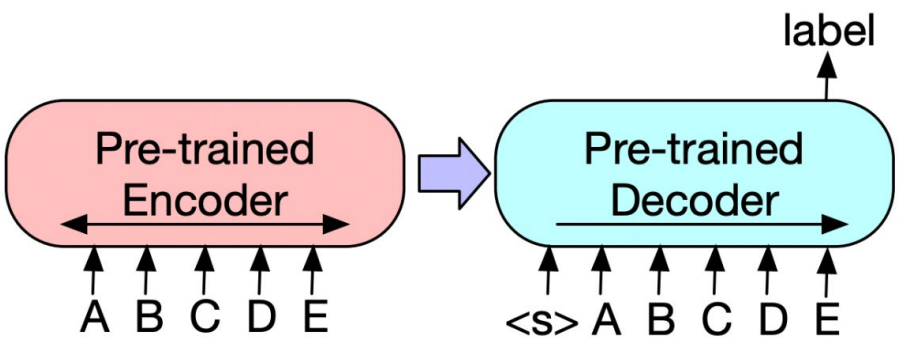


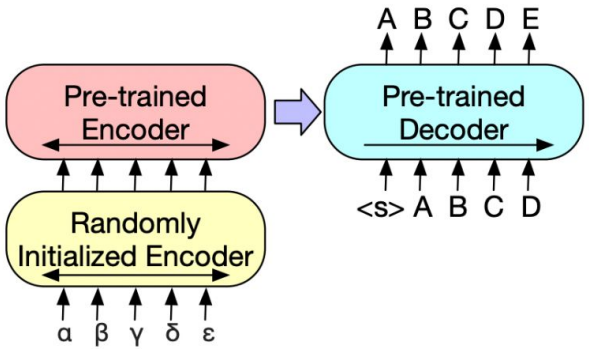
https://arxiv.org/pdf/1910.13461.pdf

BART, 2019 (Facebook)

Bidirectional and Auto-Regressive Transformers

- **Classification:** Top hidden state of the decoder is used as a representation for each word. This representation is used to classify the token.
- **Sequence Generation Tasks:** Standard autoregressive scheme.
- **Machine Translation:** BART's encoder embedding layer is replaced with a randomly initialized encoder. The new encoder can use a separate vocabulary from the original BART model.





https://arxiv.org/pdf/1910.13461.pdf

BART, 2019 (Facebook)

Bidirectional and Auto-Regressive Transformers

Model	SQuAD 1.1 F1	MNLI Acc	ELI5 PPL	XSum PPL	ConvAI2 PPL	CNN/DM PPL
BERT Base (Devlin et al., 2019)	88.5	**84.3**	-	-	-	-
Masked Language Model $(BERT)$	90.0	83.5	24.77	7.87	12.59	7.06
Masked Seq2seq $(MASS)$	87.0	82.1	23.40	6.80	11.43	6.19
Language Model (GPT)	76.7	80.1	**21.40**	7.00	11.51	6.56
Permuted Language Model $(XLNet)$	89.1	83.7	24.03	7.69	12.23	6.96
Multitask Masked Language Model	89.2	82.4	23.73	7.50	12.39	6.74
BART Base						
w/ Token Masking	90.4	84.1	25.05	7.08	11.73	6.10
w/ Token Deletion	90.4	84.1	24.61	6.90	11.46	5.87
w/ Text Infilling	**90.8**	84.0	24.26	**6.61**	**11.05**	5.83
w/ Document Rotation	77.2	75.3	53.69	17.14	19.87	10.59
w/ Sentence Shuffling	85.4	81.5	41.87	10.93	16.67	7.89
w/ Text Infilling + Sentence Shuffling	**90.8**	83.8	24.17	6.62	11.12	**5.41**

https://arxiv.org/pdf/1910.13461.pdf

BART, 2019 (Facebook)

Bidirectional and Auto-Regressive Transformers

	SQuAD 1.1 EM/F1	**SQuAD 2.0** EM/F1	**MNLI** m/mm	**SST** Acc	**QQP** Acc	**QNLI** Acc	**STS-B** Acc	**RTE** Acc	**MRPC** Acc	**CoLA** Mcc
BERT	84.1/90.9	79.0/81.8	86.6/-	93.2	91.3	92.3	90.0	70.4	88.0	60.6
UniLM	-/-	80.5/83.4	87.0/85.9	94.5	-	92.7	-	70.9	-	61.1
XLNet	**89.0**/94.5	86.1/88.8	89.8/-	95.6	91.8	93.9	91.8	83.8	89.2	63.6
RoBERTa	88.9/**94.6**	**86.5/89.4**	**90.2/90.2**	96.4	92.2	94.7	**92.4**	86.6	**90.9**	**68.0**
BART	88.8/**94.6**	86.1/89.2	89.9/90.1	**96.6**	**92.5**	**94.9**	91.2	**87.0**	90.4	62.8

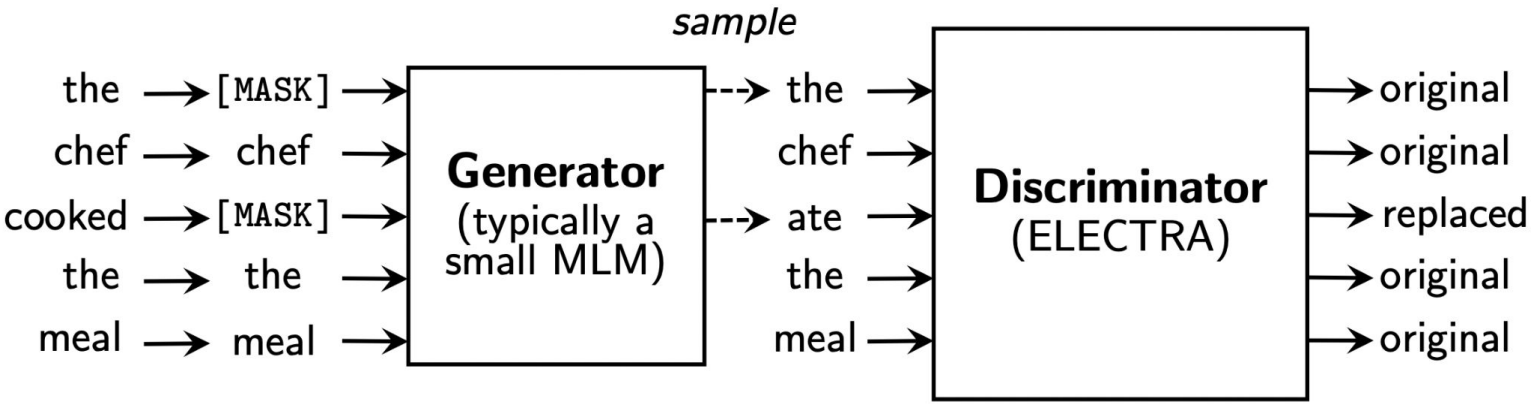
https://arxiv.org/pdf/1910.13461.pdf

ELECTRA, 2020 (Google)

Efficiently Learning an Encoder that Classifies Token Replacements Accurately

- Has two encoder networks: generator **G** (small) and discriminator **D** (ELECTRA)
- Generator learns on MLM objective and predicts masked tokens
- Discriminator learns to distinguish generated tokens from original

$$\mathcal{L}_{\text{MLM}}(\boldsymbol{x}, \theta_G) = \mathbb{E}\left(\sum_{i \in \boldsymbol{m}} -\log p_G(x_i | \boldsymbol{x}^{\text{masked}})\right)$$



https://arxiv.org/pdf/2003.10555.pdf

ELECTRA, 2020 (Google)

Efficiently Learning an Encoder that Classifies Token Replacements Accurately

$$m_i \sim \text{unif}\{1, n\} \text{ for } i = 1 \text{ to } k \qquad \boldsymbol{x}^{\text{masked}} = \text{REPLACE}(\boldsymbol{x}, \boldsymbol{m}, \texttt{[MASK]})$$

$$\hat{x}_i \sim p_G(x_i | \boldsymbol{x}^{\text{masked}}) \text{ for } i \in \boldsymbol{m} \qquad \boldsymbol{x}^{\text{corrupt}} = \text{REPLACE}(\boldsymbol{x}, \boldsymbol{m}, \hat{\boldsymbol{x}})$$

$$\mathcal{L}_{\text{MLM}}(\boldsymbol{x}, \theta_G) = \mathbb{E}\left(\sum_{i \in \boldsymbol{m}} -\log p_G(x_i | \boldsymbol{x}^{\text{masked}})\right)$$

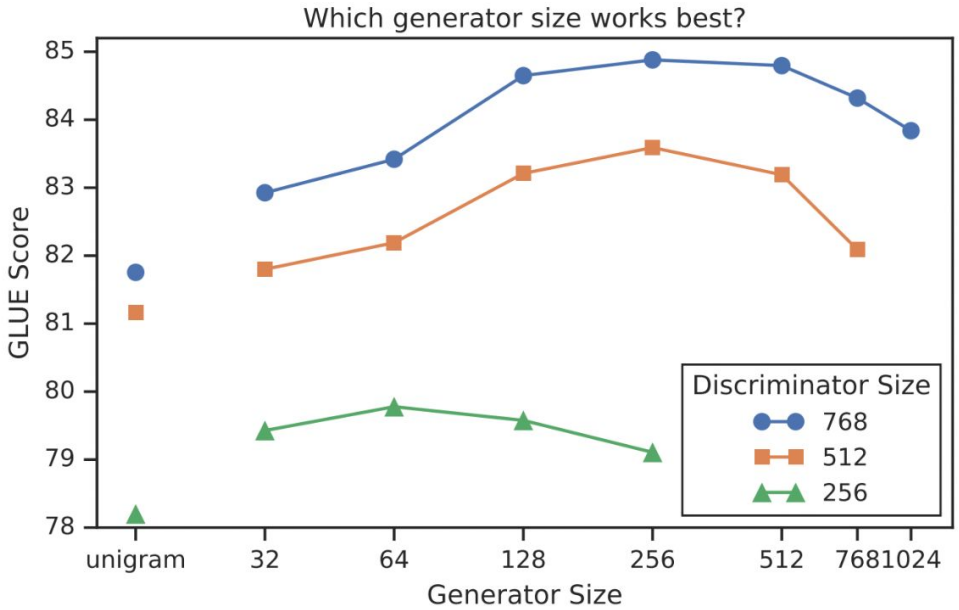
$$\mathcal{L}_{\text{Disc}}(\boldsymbol{x}, \theta_D) = \mathbb{E}\left(\sum_{t=1}^{n} -\mathbb{1}(x_t^{\text{corrupt}} = x_t) \log D(\boldsymbol{x}^{\text{corrupt}}, t) - \mathbb{1}(x_t^{\text{corrupt}} \neq x_t) \log(1 - D(\boldsymbol{x}^{\text{corrupt}}, t))\right)$$

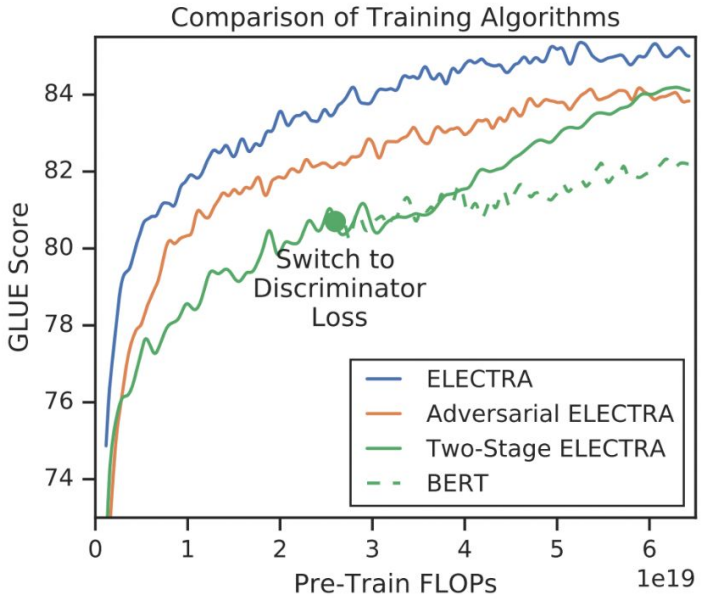
$$\min_{\theta_G, \theta_D} \sum_{\boldsymbol{x} \in \mathcal{X}} \mathcal{L}_{\text{MLM}}(\boldsymbol{x}, \theta_G) + \lambda \mathcal{L}_{\text{Disc}}(\boldsymbol{x}, \theta_D)$$

https://arxiv.org/pdf/2003.10555.pdf

ELECTRA, 2020 (Google)

Efficiently Learning an Encoder that Classifies Token Replacements Accurately





https://arxiv.org/pdf/2003.10555.pdf

ELECTRA, 2020 (Google)

Efficiently Learning an Encoder that Classifies Token Replacements Accurately

Model	Train FLOPs	Params	CoLA	SST	MRPC	STS	QQP	MNLI	QNLI	RTE	Avg.
BERT	1.9e20 (0.27x)	335M	60.6	93.2	88.0	90.0	91.3	86.6	92.3	70.4	84.0
RoBERTa-100K	6.4e20 (0.90x)	356M	66.1	95.6	**91.4**	92.2	92.0	89.3	94.0	82.7	87.9
RoBERTa-500K	3.2e21 (4.5x)	356M	68.0	96.4	90.9	92.1	92.2	90.2	94.7	86.6	88.9
XLNet	3.9e21 (5.4x)	360M	69.0	**97.0**	90.8	92.2	92.3	90.8	94.9	85.9	89.1
BERT (ours)	7.1e20 (1x)	335M	67.0	95.9	89.1	91.2	91.5	89.6	93.5	79.5	87.2
ELECTRA-400K	7.1e20 (1x)	335M	**69.3**	96.0	90.6	92.1	**92.4**	90.5	94.5	86.8	89.0
ELECTRA-1.75M	3.1e21 (4.4x)	335M	69.1	96.9	90.8	**92.6**	**92.4**	**90.9**	**95.0**	**88.0**	**89.5**

https://arxiv.org/pdf/2003.10555.pdf

ELECTRA, 2020 (Google)

Efficiently Learning an Encoder that Classifies Token Replacements Accurately

Model	Train FLOPs	Params	SQuAD 1.1 dev		SQuAD 2.0 dev		SQuAD 2.0 test	
			EM	F1	EM	F1	EM	F1
BERT-Base	6.4e19 (0.09x)	110M	80.8	88.5	–	–	–	–
BERT	1.9e20 (0.27x)	335M	84.1	90.9	79.0	81.8	80.0	83.0
SpanBERT	7.1e20 (1x)	335M	88.8	94.6	85.7	88.7	85.7	88.7
XLNet-Base	6.6e19 (0.09x)	117M	81.3	–	78.5	–	–	–
XLNet	3.9e21 (5.4x)	360M	**89.7**	**95.1**	87.9	**90.6**	87.9	90.7
RoBERTa-100K	6.4e20 (0.90x)	356M	–	94.0	–	87.7	–	–
RoBERTa-500K	3.2e21 (4.5x)	356M	88.9	94.6	86.5	89.4	86.8	89.8
ALBERT	3.1e22 (44x)	235M	89.3	94.8	87.4	90.2	88.1	90.9
BERT (ours)	7.1e20 (1x)	335M	88.0	93.7	84.7	87.5	–	–
ELECTRA-Base	6.4e19 (0.09x)	110M	84.5	90.8	80.5	83.3	–	–
ELECTRA-400K	7.1e20 (1x)	335M	88.7	94.2	86.9	89.6	–	–
ELECTRA-1.75M	3.1e21 (4.4x)	335M	**89.7**	94.9	**88.0**	**90.6**	**88.7**	**91.4**

https://arxiv.org/pdf/2003.10555.pdf

Recap

- **GPT:** Transformer-decoder. Language modeling, learns to predict the next word.
- **BERT:** MLM, NSP. Transformer-encoder. Learns to unmask tokens.
- **RoBERTa:** list of techniques for BERT training.
- **XLNet:** Transformer-decoder build on Transformer(-XL). Autoregressively learns to recover permuted sentence.
- **ALBERT:** Lighter version of BERT with factorized embeddings and shared layers.
- **MASS:** Transformer trained to unmask sequences of tokens.
- **BART:** Vanilla Transformer pre-trained with various corruption functions
- **ELECTRA:** Transformer-encoder trained as discriminator for small encoder-type generator, that was learned to unmask tokens.

Conclusions

- More data is better
- Bigger model is better
- The choice of pre-train task hugely depends on the downstream task