



Natural Language Processing

ELMo, BERT, GPT, and T5 (BERT and its Family)



News

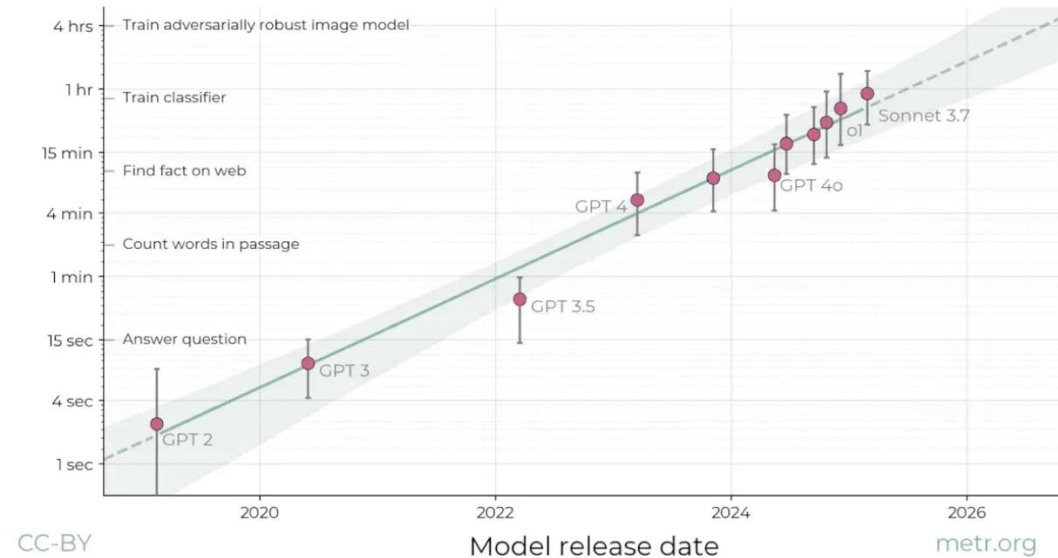


Julian Schrittwieser

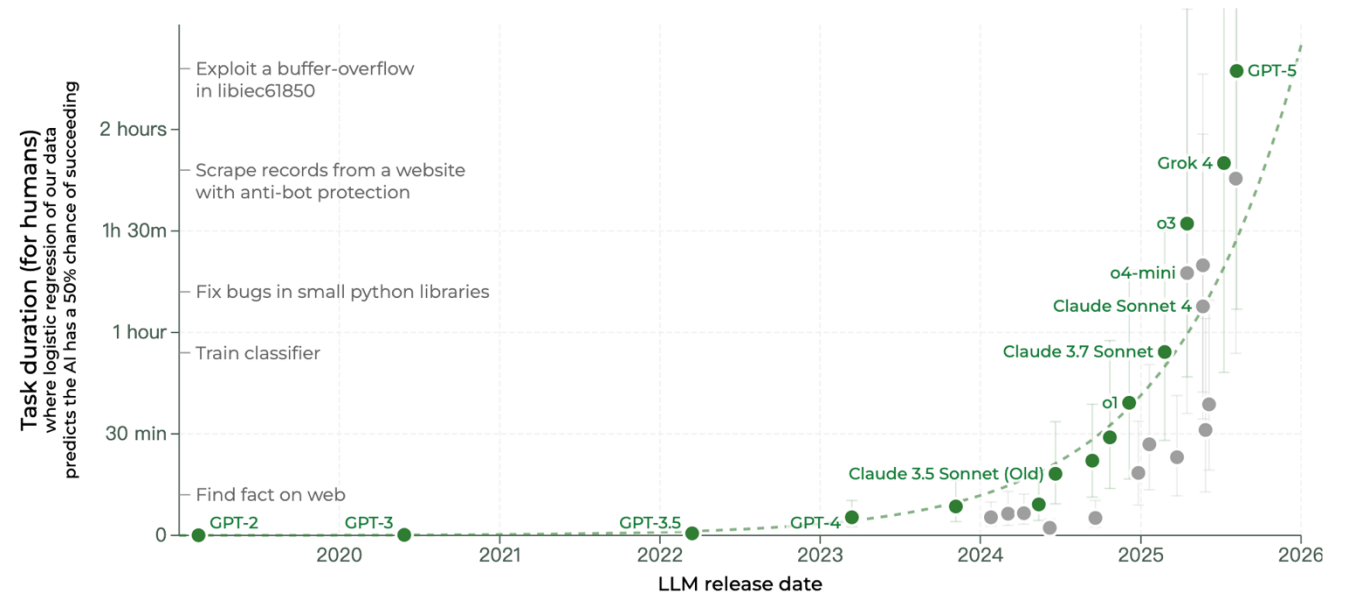
Failing to Understand the Exponential, Again

Sat 27 September 2025

The length of tasks AI can do is doubling every 7 months
Task length (at 50% success rate)

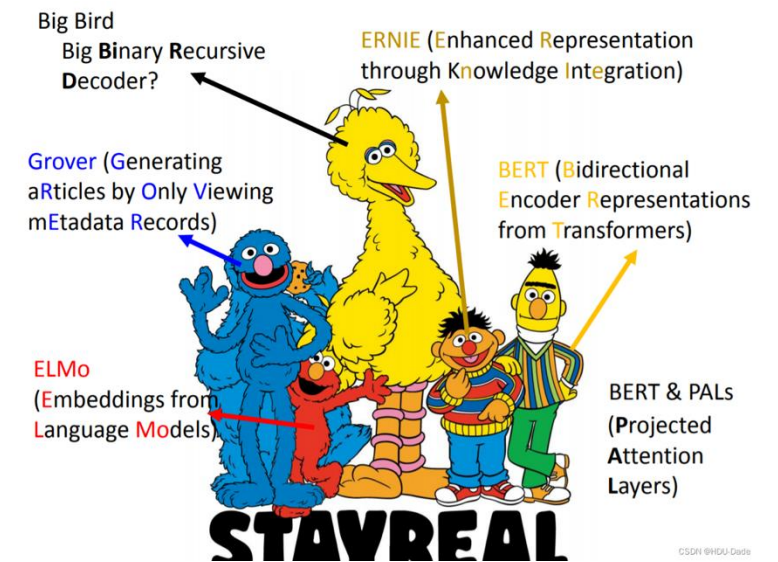


Time-horizon of software engineering tasks different LLMs can complete 50% of the time



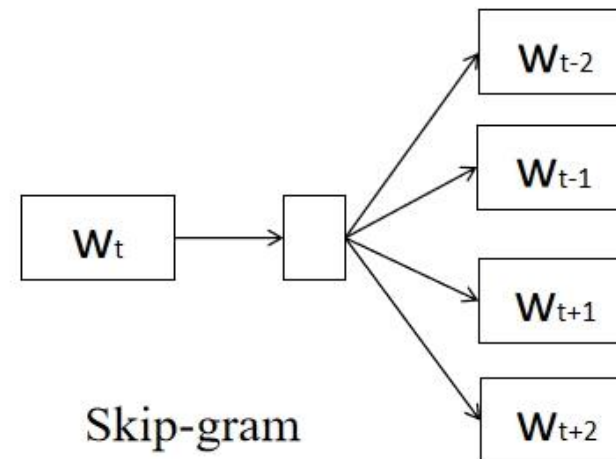
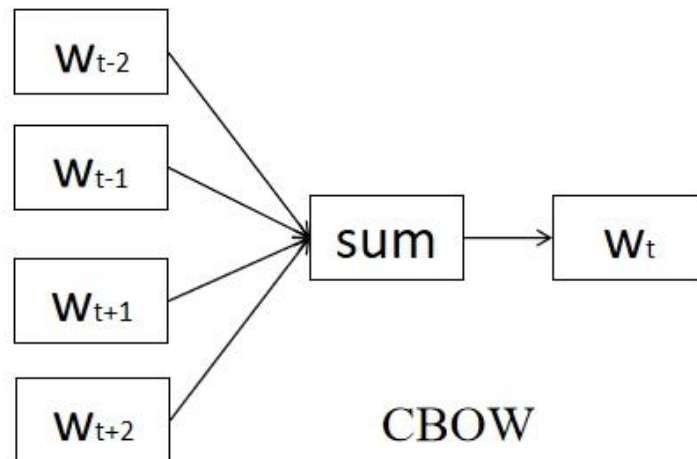
Outline

- Recap: Word Embeddings, RNN
- Pretraining Language Model from Word Embeddings
 - ELMo
- Pretraining by Transformer
 - Encoders (BERT)
 - Encoder-Decoders (T5)
 - Decoders (GPT)
- GPT-3 with In-Context Learning and Large Language Models



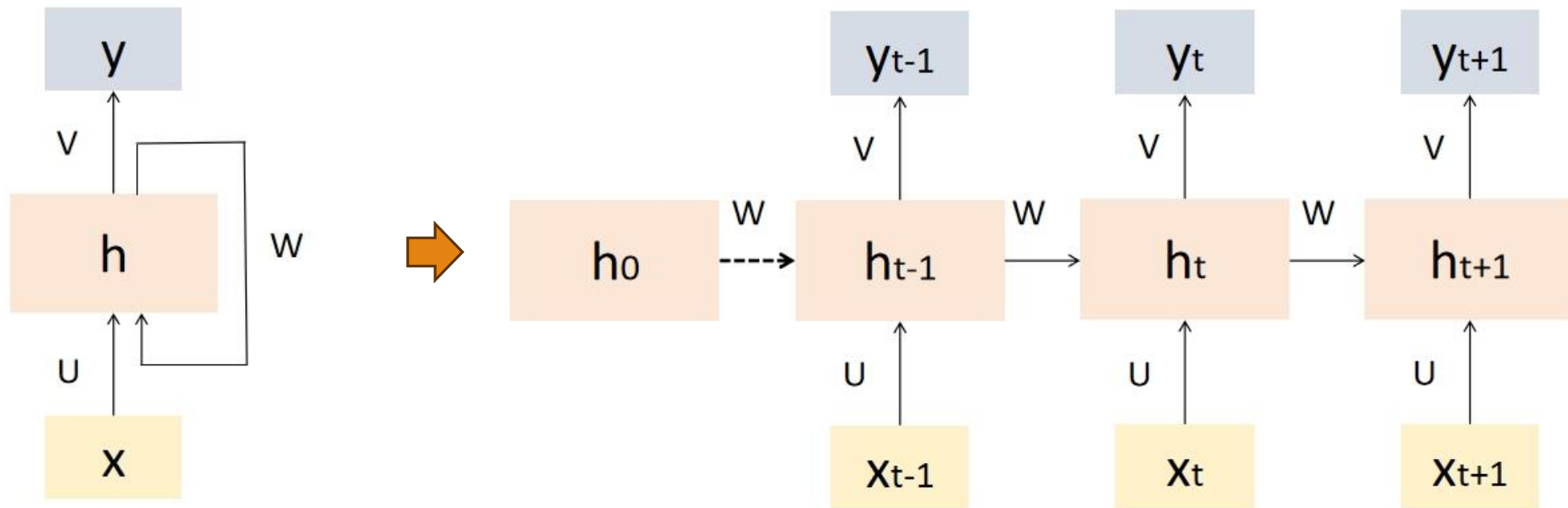
Recap: Word Embeddings (Word2Vec)

- Skip-gram
 - Use words to predict their contexts.
- CBOW
 - Use the context to predict the target word.



Recap: RNN

- Recurrent Neural Networks (RNNs) are a type of artificial neural network designed to handle sequential data by **capturing temporal dependencies**.
 - The same set of weights and biases are used across all time steps



Pretrained Word Embeddings

Consider **I record the record**: the two instances of record mean different things.

-0.17	-0.89	0.06	-0.41	-1.08	-0.74	1.41	-0.94	-0.32	0.12
-------	-------	------	-------	-------	-------	------	-------	-------	------

Word2Vec or GloVe word embeddings of “**record**”

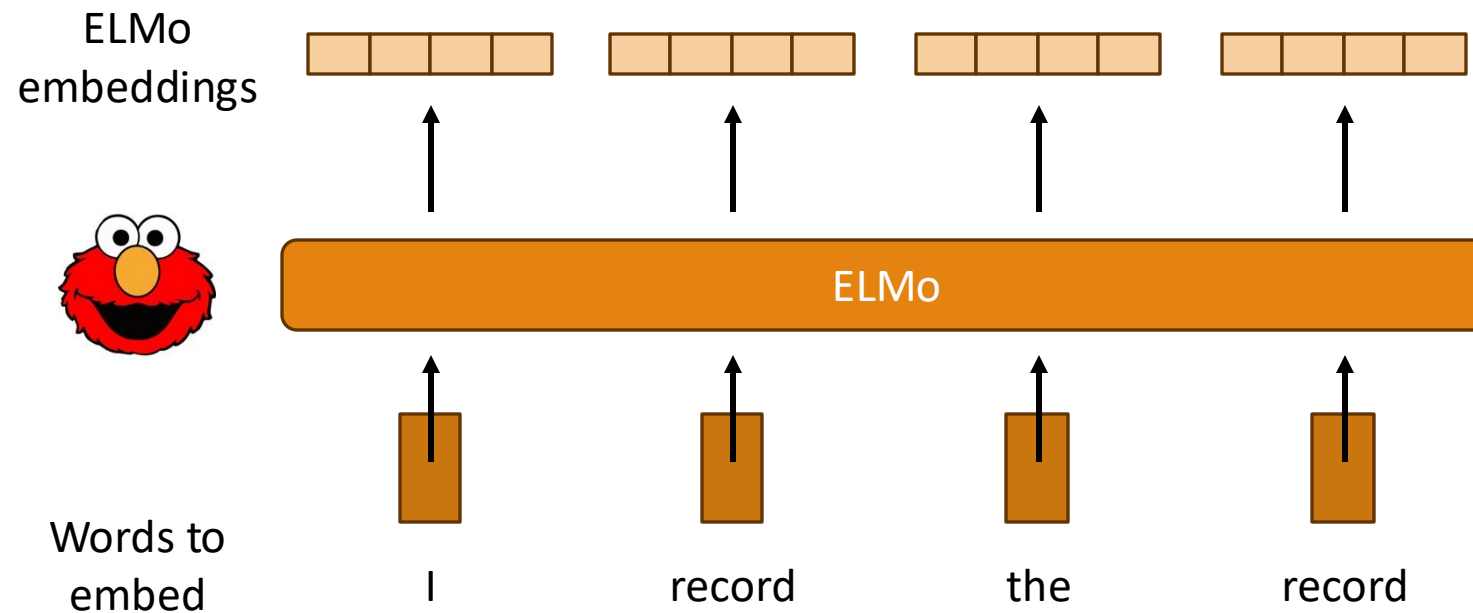


Vector representation

Issue: Both GloVe and Word2Vec represent the word “record” as the same vector regardless of the context.

Context Matters: ELMo (Peters et al., 2018)

Rather than using a fixed representation for each word, **ELMo (Embeddings from Language Models)** considers the entire sentence before assigning each word in it an embedding.



Reconstruct the Sentences (next token prediction)

National Hsing Hua University is in _____. (Hsinchu)

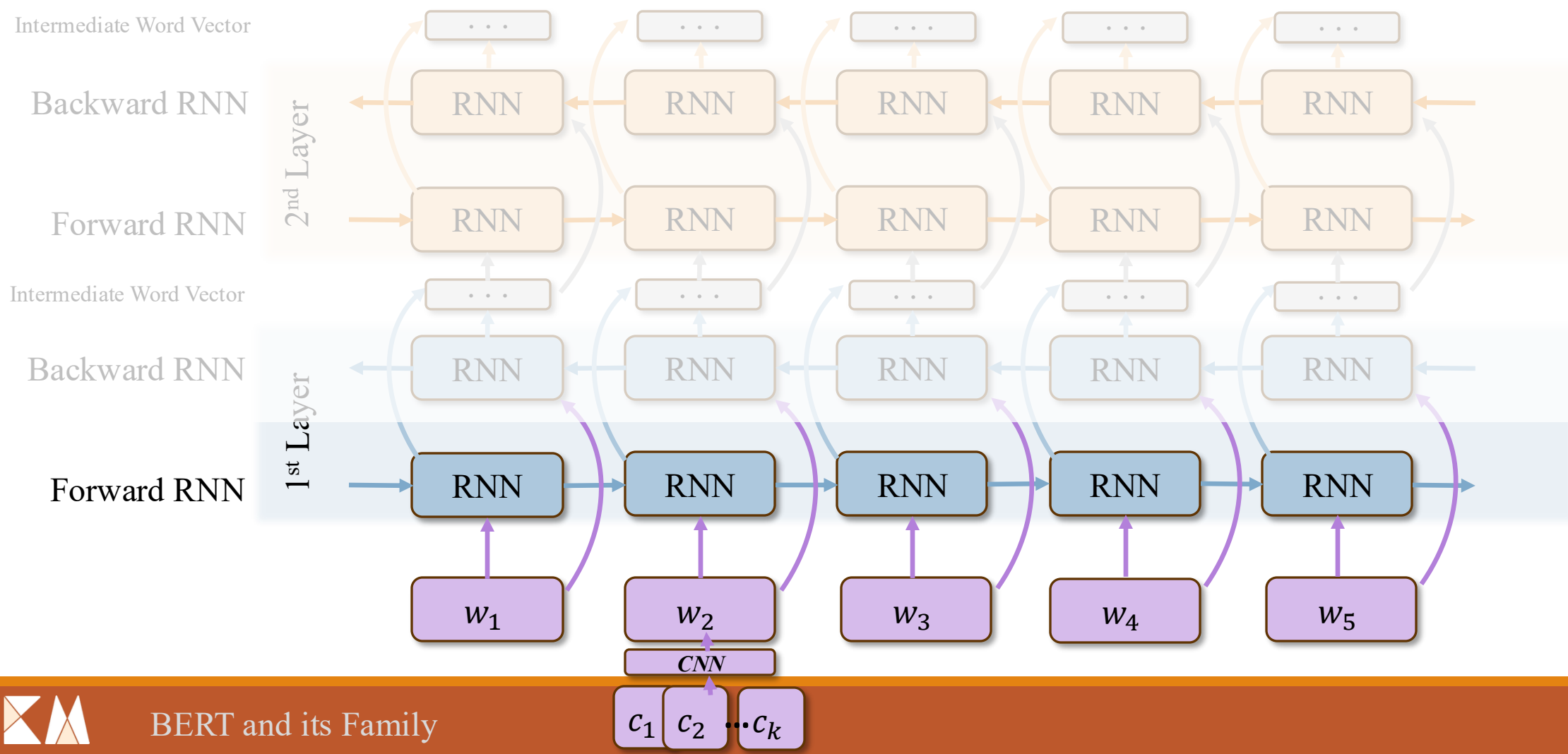
I put ____ bag on the table. (a)

I went to the zoo to see the lions, monkeys, elephants, and _____. ()

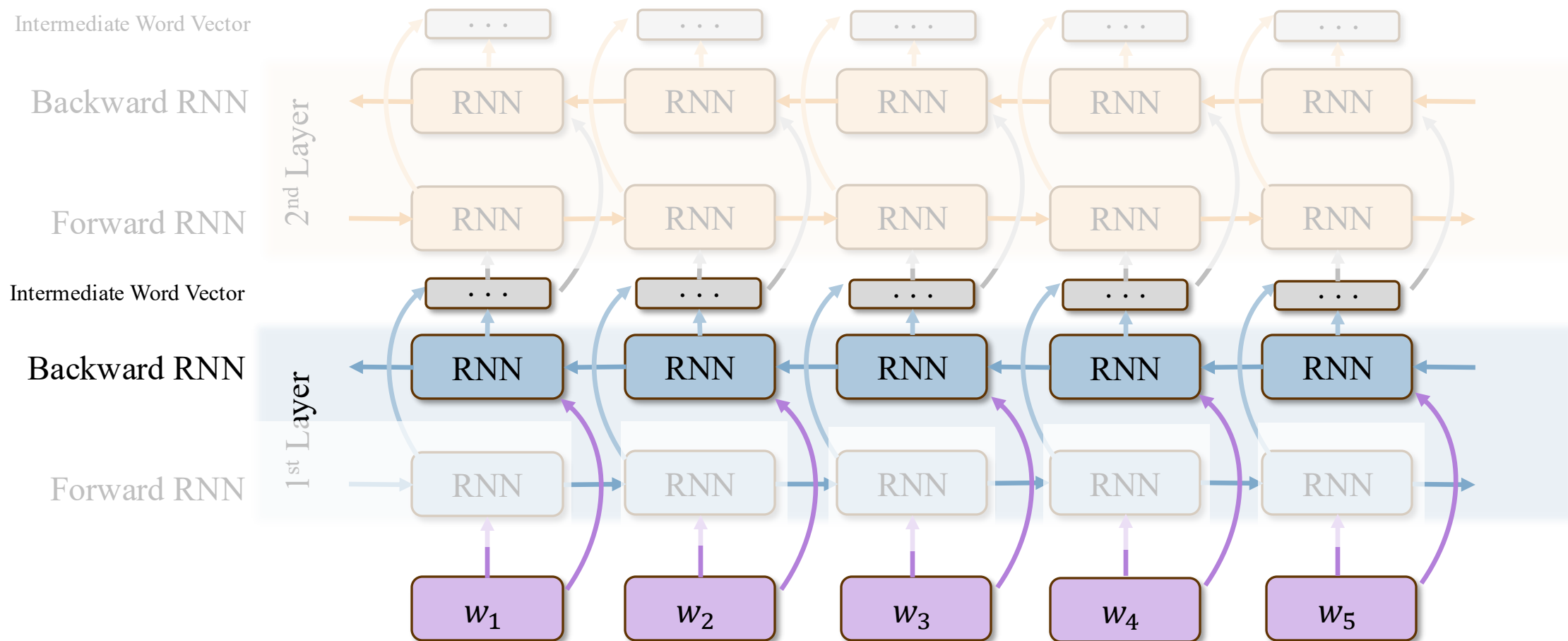
The overall value I got from the two hours of watching it was the total sum of popcorn and the drink. The movie was _____. (bad)

I was thinking about the sequence that goes 1, 1, 2, 3, 5, 8, 13, 21, _____(34)

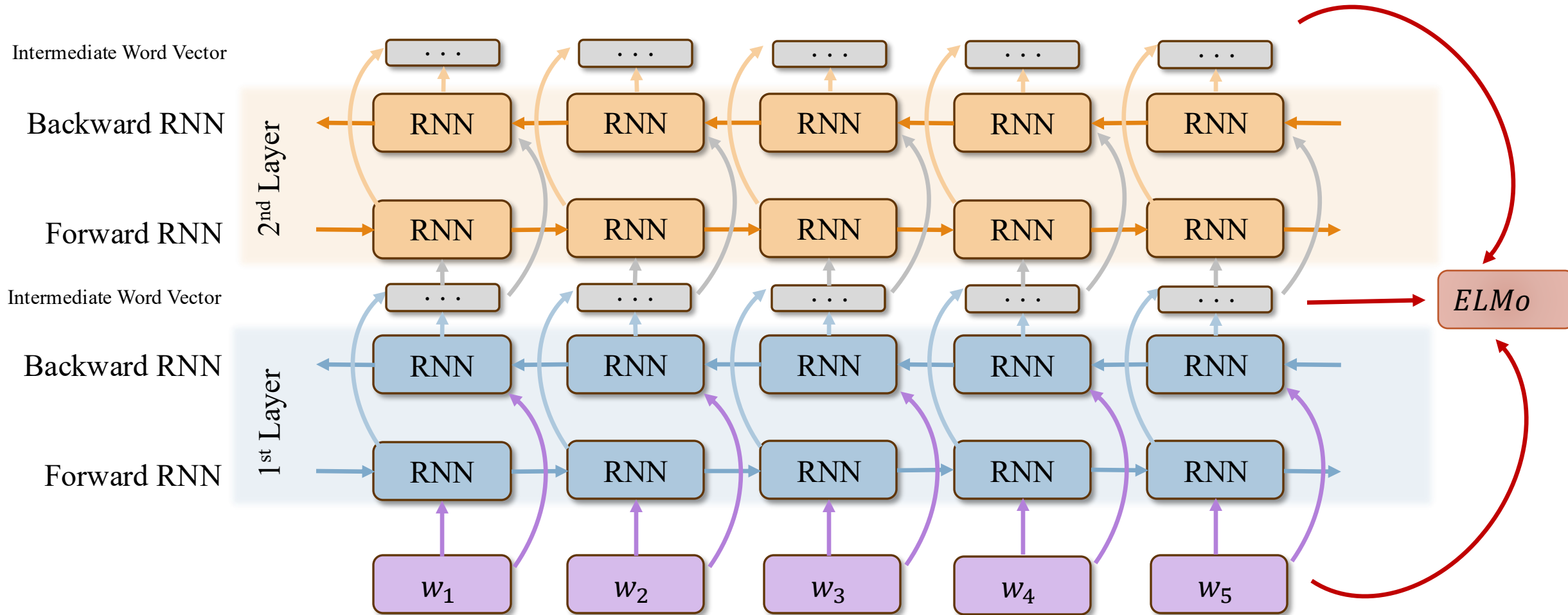
Bidirectional Language Model (biLM)



Bidirectional Language Model (biLM)



Bidirectional Language Model (biLM)



Pretraining through Language Modeling

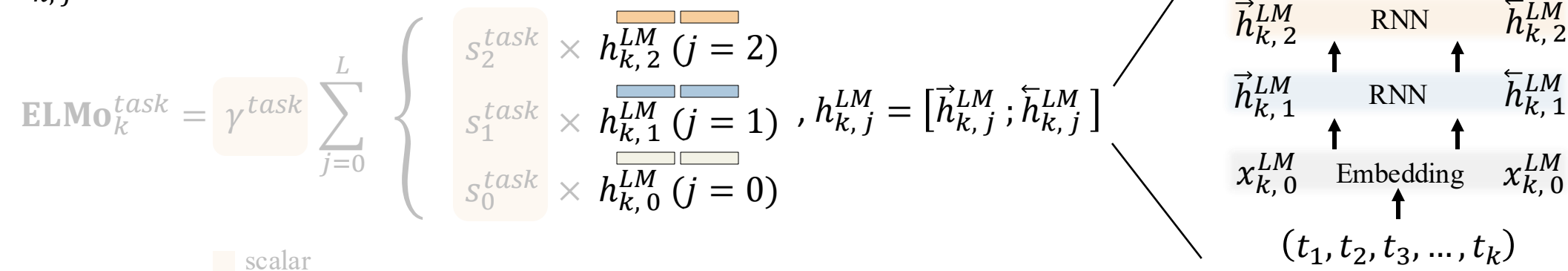
Step 1. Concatenate hidden layers.

Concatenate forward and backward LM of each layer together (for token layer, we concatenate it with itself)

$\vec{h}_{k,0}^{LM}$: Token Layer

$\vec{h}_{k,j}^{LM}$: Forward LM at j layer

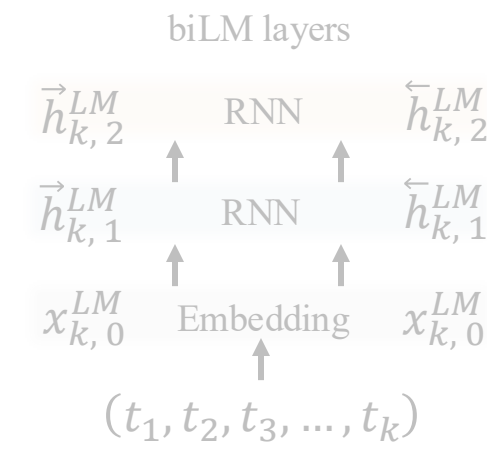
$\overleftarrow{h}_{k,j}^{LM}$: Backward LM at j layer



Pretraining through Language Modeling

Step 2. Multiply each vector by a weight

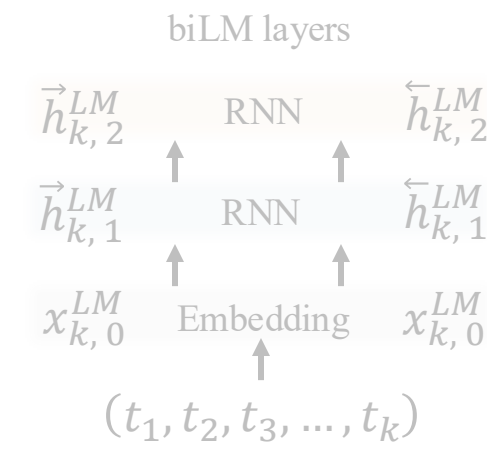
We assign each concatenated layer with softmax-normalized weight, s^{task}

$$\text{ELMo}_k^{task} = \underbrace{\gamma^{task}}_{\text{scalar}} \sum_{j=0}^L \left\{ \begin{array}{l} s_2^{task} \times \overline{h_{k,2}^{LM}} (j=2) \\ s_1^{task} \times \overline{h_{k,1}^{LM}} (j=1) \\ s_0^{task} \times \overline{h_{k,0}^{LM}} (j=0) \end{array} \right. , h_{k,j}^{LM} = [\overrightarrow{h_{k,j}^{LM}} ; \overleftarrow{h_{k,j}^{LM}}]$$


Pretraining through Language Modeling

Step 3. Sum all weighted representations.

Once weighted vectors are added together, we use γ^{task} to allow the task model to scale the entire ELMo vector, which is crucial for optimization purposes.

$$\mathbf{ELMo}_k^{task} = \underbrace{\gamma^{task}}_{\text{scalar}} \sum_{j=0}^L \left\{ \begin{array}{l} s_2^{task} \times h_{k,2}^{LM} (j=2) \\ s_1^{task} \times h_{k,1}^{LM} (j=1) \\ s_0^{task} \times h_{k,0}^{LM} (j=0) \end{array} \right. , h_{k,j}^{LM} = [\vec{h}_{k,j}^{LM}; \overleftarrow{h}_{k,j}^{LM}]$$


The diagram illustrates the biLM layers. At the bottom, the input sequence $(t_1, t_2, t_3, \dots, t_k)$ is processed by an Embedding layer to produce $x_{k,0}^{LM}$. This input is fed into two parallel RNN layers. The forward RNN (left) produces hidden states $\vec{h}_{k,1}^{LM}$ and $\vec{h}_{k,2}^{LM}$. The backward RNN (right) produces hidden states $\overleftarrow{h}_{k,1}^{LM}$ and $\overleftarrow{h}_{k,2}^{LM}$. The hidden states for each time step j are concatenated to form $h_{k,j}^{LM} = [\vec{h}_{k,j}^{LM}; \overleftarrow{h}_{k,j}^{LM}]$.

Pretraining through Language Modeling

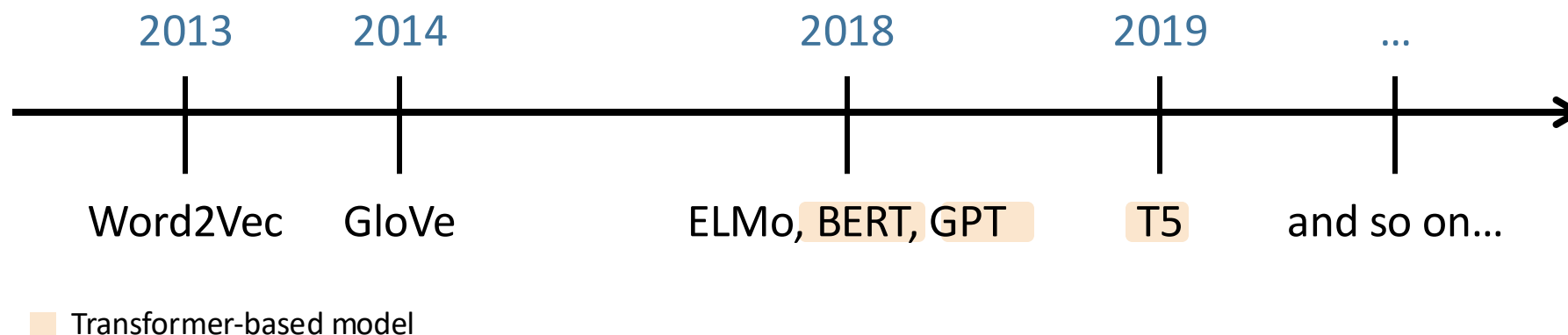
ELMo is a type of deep contextualized word representation that are created through a learning process based on **hidden layers** of a deep bidirectional language model.

ELMo is a task specific combination of the intermediate layer representations in the biLM.

ELMo generates the contextualized embedding by concatenating the hidden states (and initial embedding) together followed by weighted summation.

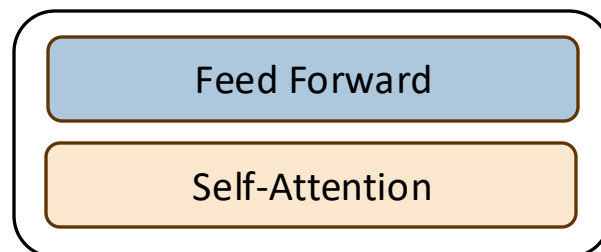
The Transformer: Going beyond RNNs

The release of the **Transformer** along with the results it achieved on tasks like machine translation led researchers to view it as a superior alternative to RNNs, given its improved handling long-term dependencies compared to RNNs.



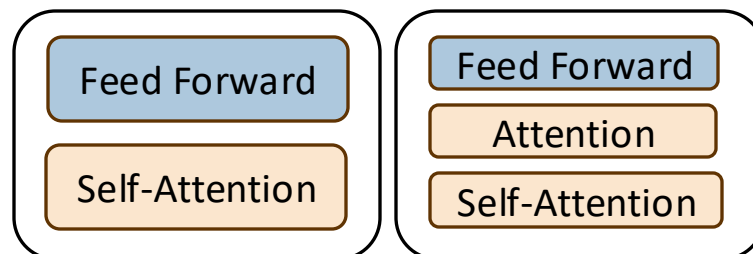
Three Types of Pretraining Ways

Encoders



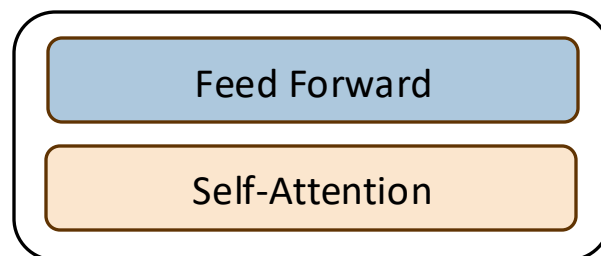
- Bidirectional – capable of looking into the future
- Suitable for downstream task

Encoder-Decoders



- Pros of decoders and encoders?
- Cons of having both during pretraining

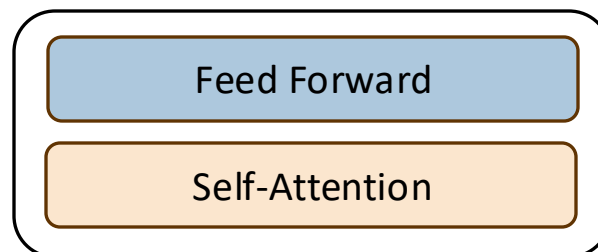
Decoders



- What LMs we have seen so far
- Suitable for generation task
- Not bidirectional

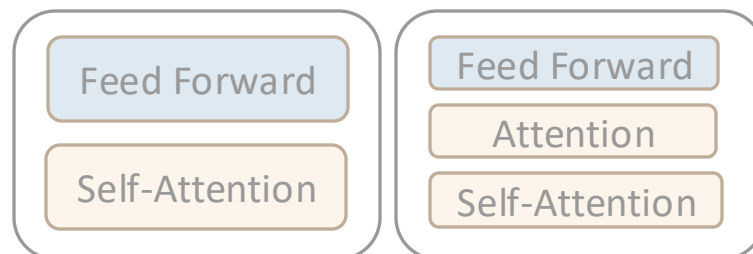
Three Types of Pretraining Ways

Encoders



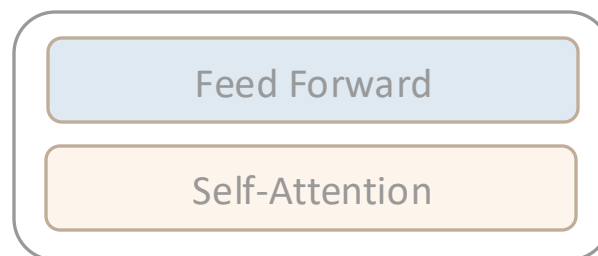
- Bidirectional – capable of looking into the future
- Suitable for downstream task

Encoder-Decoders



- Pros of decoders and encoders?
- Cons of having both during pretraining

Decoders

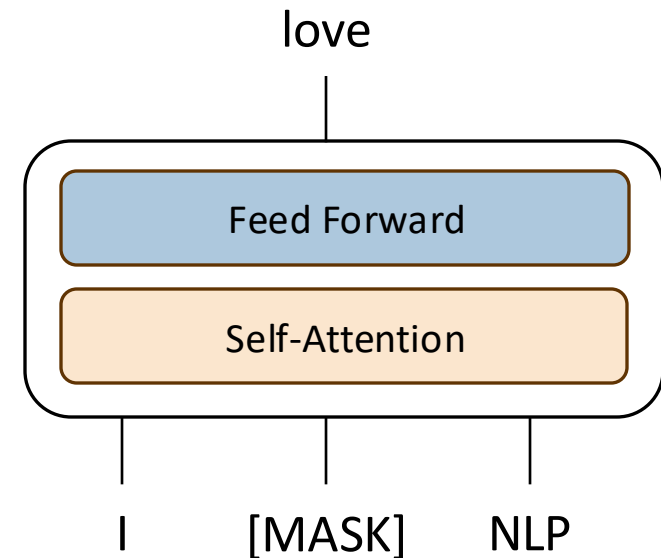


- What LMs we have seen so far
- Suitable for generation task
- Not bidirectional

Pretraining Encoder:

Idea: Encoder works bidirectionally which is more powerful than either a left-to-right model or the shallow concatenation of a left-to-right and a right-to-left model.

- Pretraining task
 - #1 MLM (Masked Language Models):
 - Train the model to reconstruct the sentence by replacing some fraction of words in the input with a special token, [MASK]
 - The model will look at the sentence bidirectionally to try to predict the removed words.
 - #2 NSP (Next Sentence Prediction):
 - In order to train a model that understands sentence relationships, we pre-train for a binarized next sentence prediction task that can be trivially generated from any monolingual corpus



BERT: Bidirectional Encoder Representations from Transformers

(Devlin et al., 2018)

Task 1. “Masked Language Models”

The objective is proposed alongside the weights of a pretrained Transformer, a model they refer to as BERT.

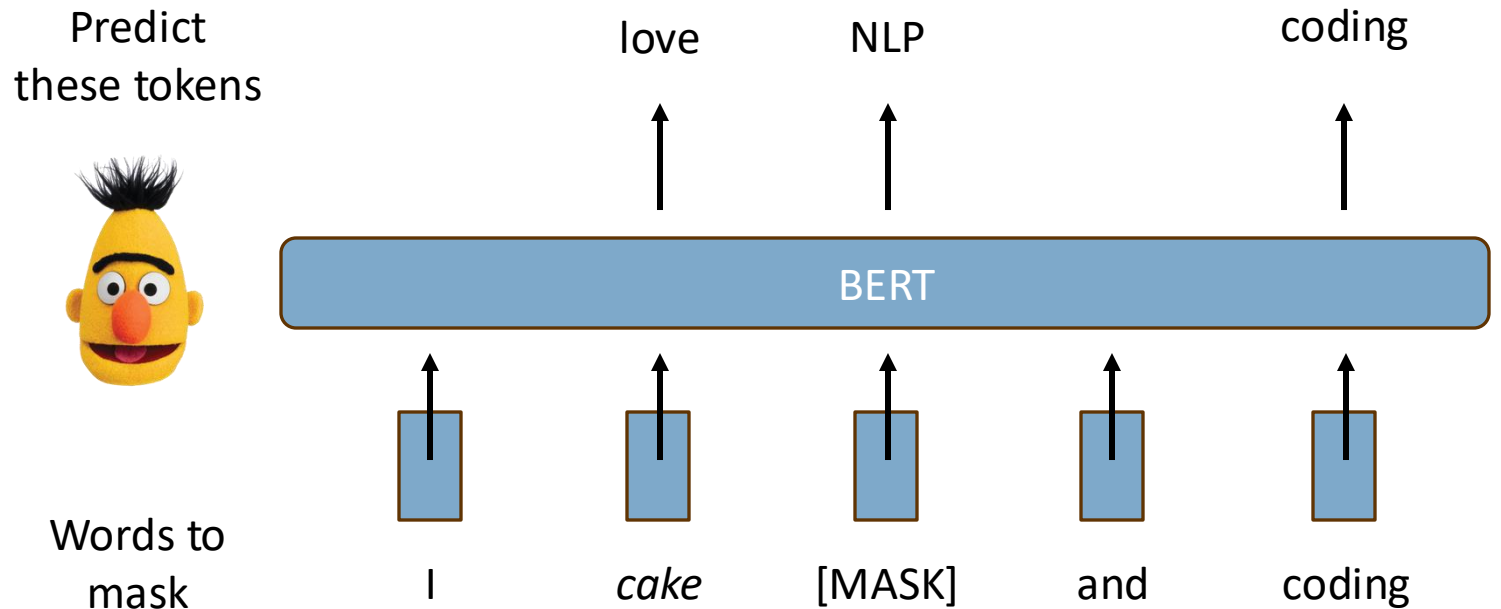
Replace the **15% of the tokens** with

- 1) [MASK] 80% of the time
- 2) Random 10% of the time
- 3) Unchanged 10 % of the time

Predict these tokens

Why?

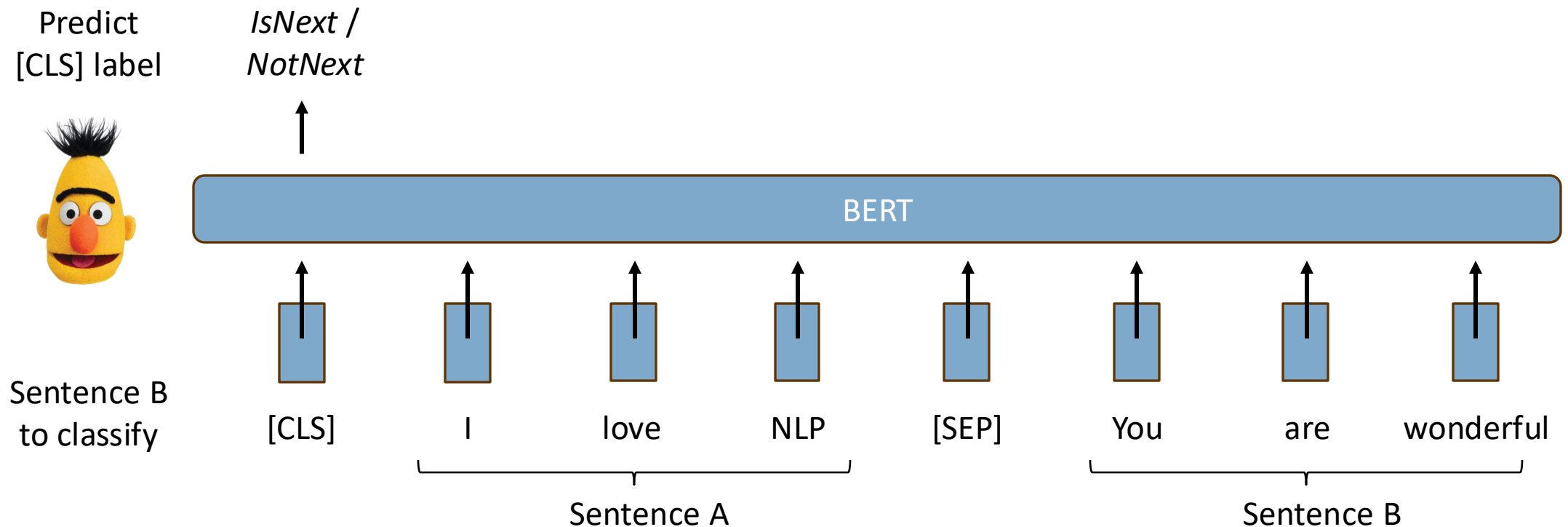
Prevent the model from becoming complacent and failing to construct robust representations for unmasked words. (No masks are seen at fine-tuning time!)



BERT (Devlin et al., 2018)

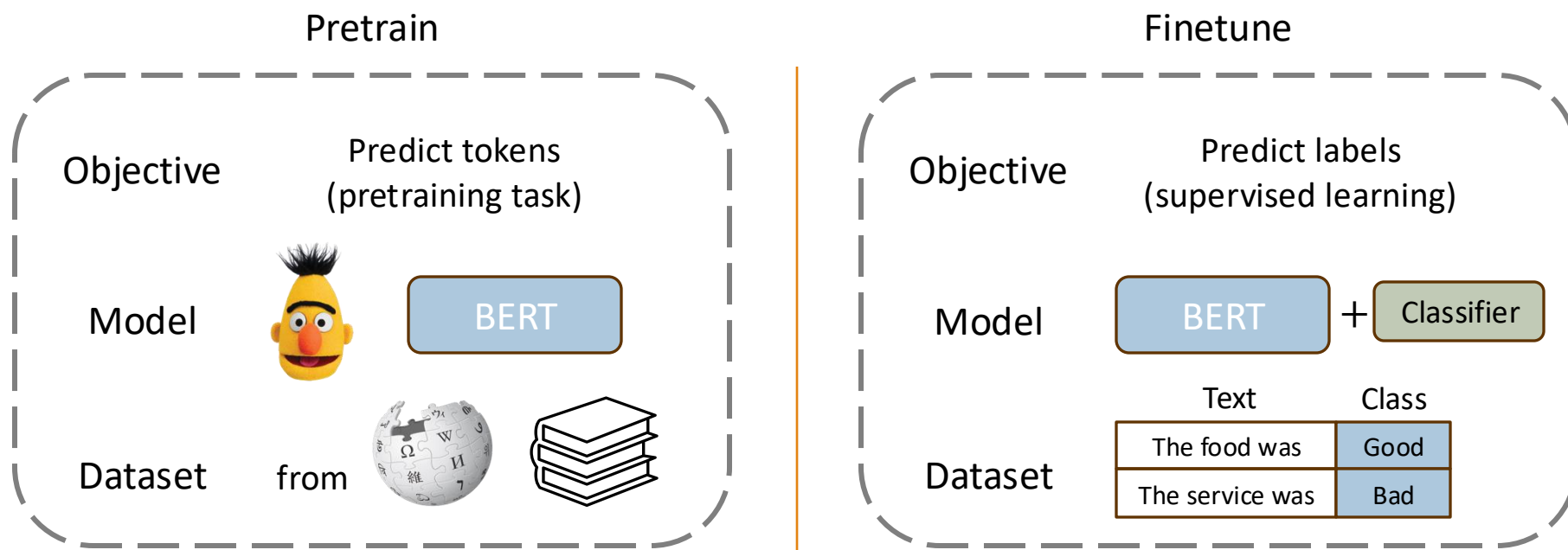
Task 2. “Next Sentence Prediction”

To develop a model that understands sentence relationships, we pre-train for a binarized next sentence prediction task that can be easily generated from any monolingual corpus.



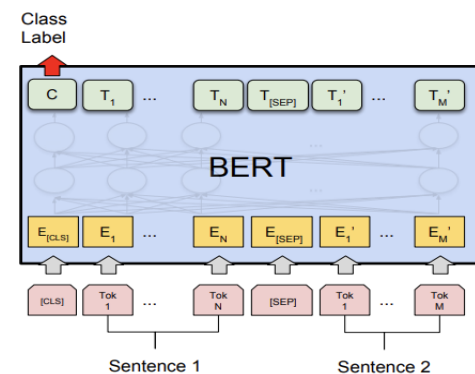
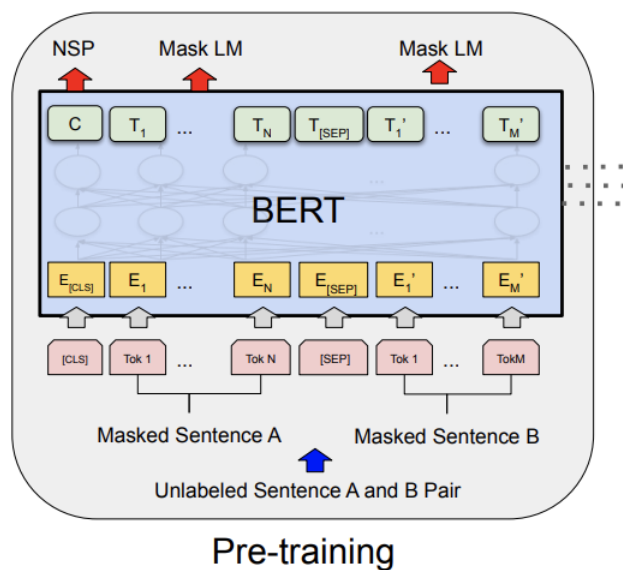
Transfer Learning: Pretrain and Finetune

Modern-day standard approach is to **first pre-train then fine-tune**. During pretraining, a large amount of unlabeled text is used to build a general model of language understanding before fine-tuned on various specific NLP tasks

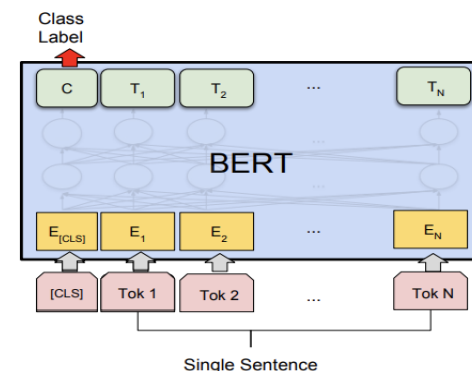


Transfer Learning: Pretrain and Finetune

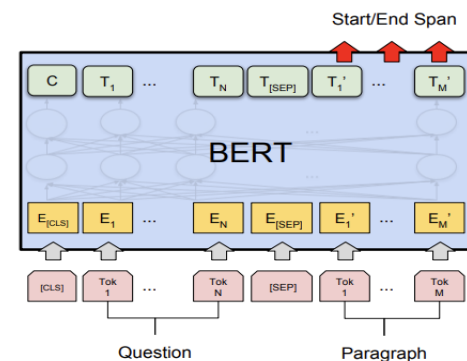
Modern-day standard approach is to **first pre-train** then **fine-tune**. During pretraining, a large amount of unlabeled text is used to build a general model of language understanding before fine-tuned on various specific NLP tasks



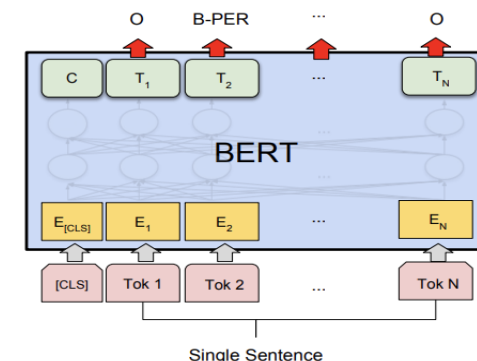
(a) Sentence Pair Classification Tasks:
MNLI, QQP, QNLI, STS-B, MRPC,
RTE, SWAG



(b) Single Sentence Classification Tasks:
SST-2, CoLA



(c) Question Answering Tasks:
SQuAD v1.1



(d) Single Sentence Tagging Tasks:
CoNLL-2003 NER

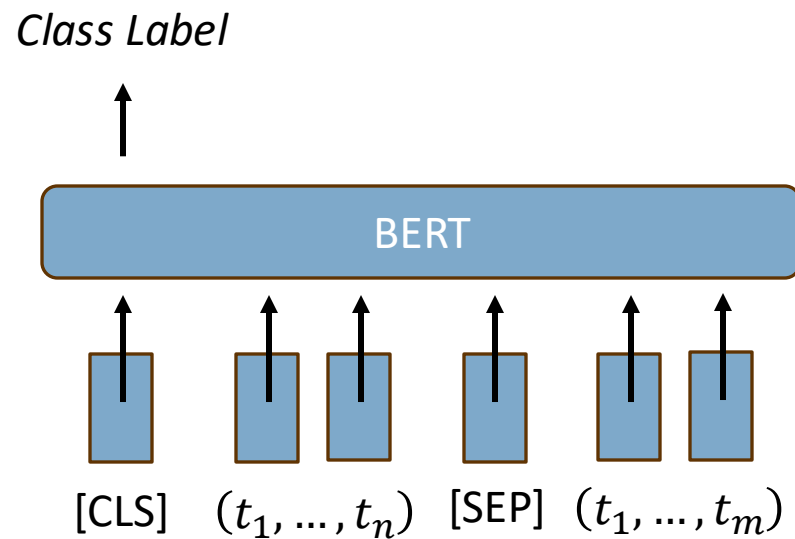
Fine-tuning

Ref: BERT paper

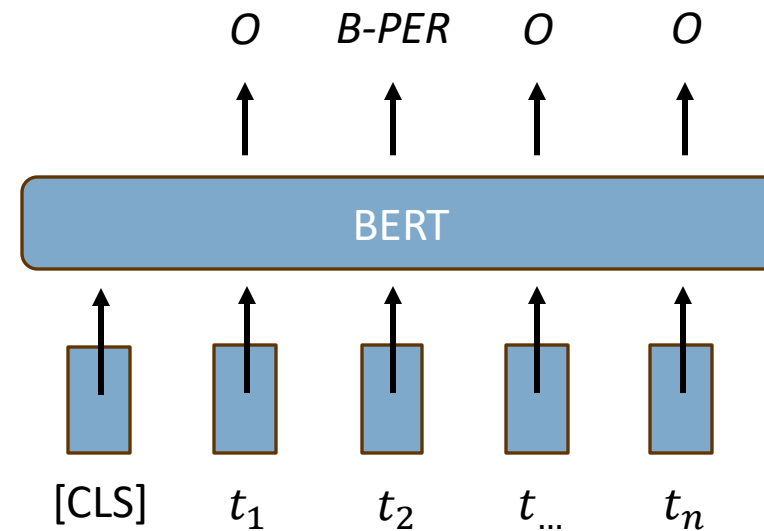


Finetune BERT

Task-specific models are used to minimize the number of parameters need to be trained from scratch. This is done by integrating BERT with one additional output layer.



(A) Sentence Pair
Classification



(B) Single Sentence
Tagging Task

Details about BERT

Google release two models at first:

- BERT-base (110M parameters): 12 layers, 768-dim hidden size, 12 attention heads.
- BERT-large (340M parameters): 24 layers, 1024-dim hidden size, 16 attention heads.

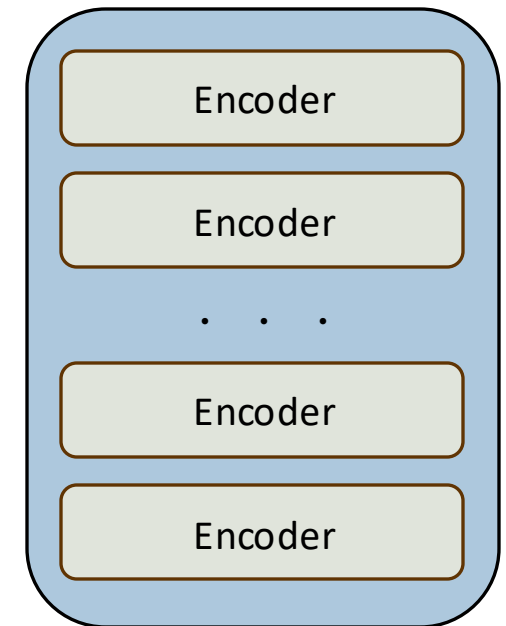
Trained on two datasets:

- BooksCorpus (800M words)
- English Wikipedia (2,500M words)

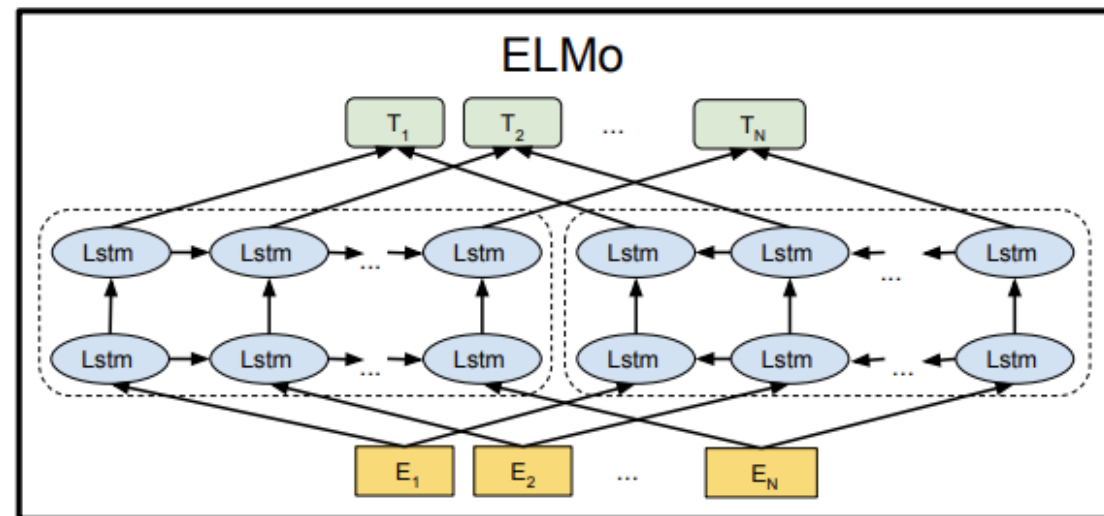
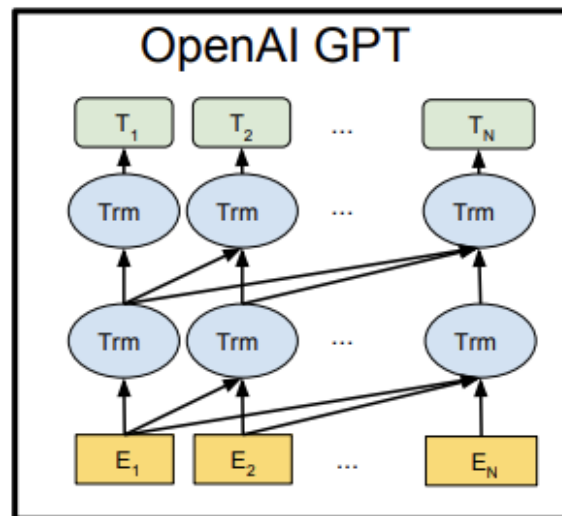
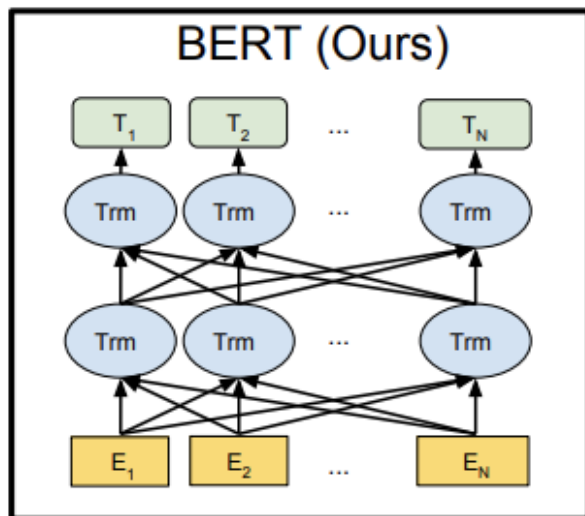
Pretraining is expensive and impractical on a single GPU.

- BERT was pretrained with 64 TPU chips for a total of 4 days.
- (TPUs are special tensor operation acceleration hardware)
- **Finetuning is practical and common on a single GPU**

BERT



Why bidirectional?



Ref: BERT paper

Extensions of BERT

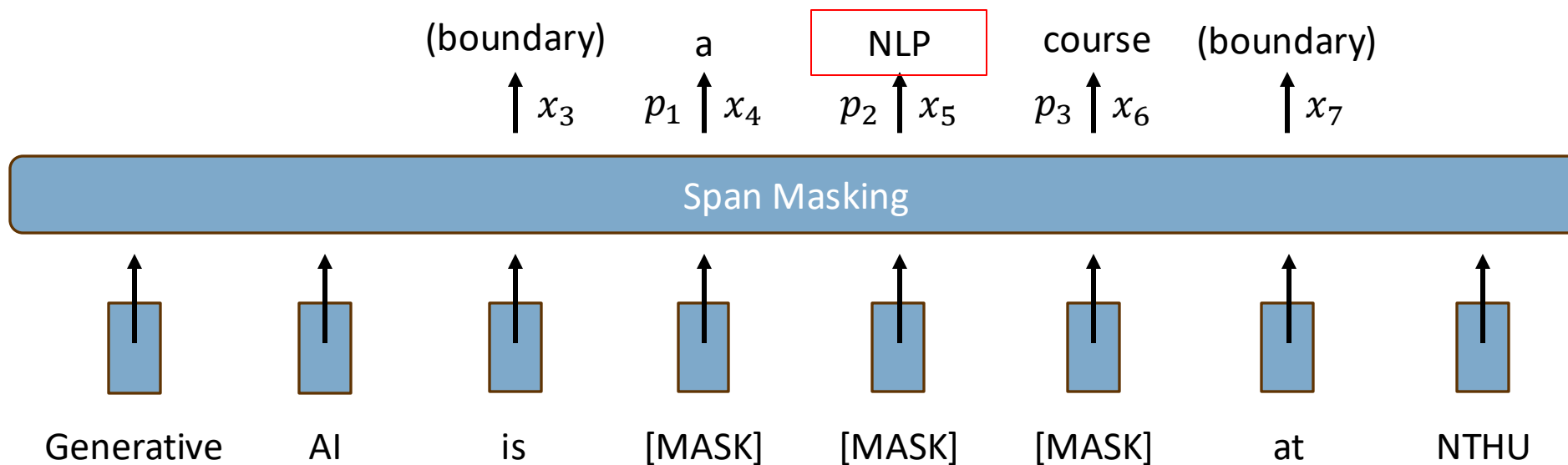
Model	MLM task	NSP task	Release
BERT	Static masking	Sentence relationship	2018/10
RoBERTa	Dynamic masking	Remove NSP task	2019/07
SpanBERT https://arxiv.org/pdf/1907.11692.pdf	Span masking	Remove NSP task	2019/07
Albert	N-gram masking	Sentence order	2019/09
DistilBERT	Compression by knowledge distillation		2019/10
DeBERTa	Decoding-enhanced BERT with Disentangled Attention		2020/06
TinyBERT	TinyBERT: Distilling BERT for Natural Language Understanding		2019/09



Extensions of BERT

Span Masking: Span Boundary Objective (SBO)

$$\mathcal{L}(NLP) = \mathcal{L}_{MLM}(NLP) + \mathcal{L}_{SBO}(NLP) = -\log P(NLP \mid x_5) - \log P(NLP \mid \mathbf{x}_3, \mathbf{x}_7, p_2)$$



RoBERTa: A Robustly Optimized BERT Pretraining Approach

Key Improvements:

- 1.Longer Training Duration:** RoBERTa increases the training time and batch size, utilizing a larger training dataset.
- 2.Removing "Next Sentence Prediction" (NSP) Objective:** Unlike BERT, RoBERTa omits the NSP task, simplifying training.
- 3.Training on Longer Sequences:** Training uses longer text sequences, improving the model's ability to capture context.
- 4.Dynamic Masking:** Uses a dynamic masking strategy, where masking patterns in the training data change every iteration to avoid data overfitting.



RoBERTa Training Approach and Performance

Training Process:

- **Data:** RoBERTa uses a larger dataset of over **160GB** of text, including CC-NEWS, OpenWebText, and other public text sources.
- **Architecture:** Similar to BERT, based on the Transformer architecture with 24 layers, 1024 hidden units, and 16 self-attention heads (for the large model configuration).
- **Optimization:** Improved learning rate and batch size adjustments. Dynamic masking is applied to vary the tokens masked during training, increasing data diversity.

Experimental Results:

- On the GLUE benchmark, RoBERTa achieves state-of-the-art performance on 4 out of 9 tasks.
- In SQuAD (question-answering) and RACE (reading comprehension) datasets, RoBERTa's performance matches or surpasses the best models.
- Demonstrates that BERT's Masked Language Model (MLM) objective is highly competitive when trained with the right strategies.

	MNLI	QNLI	QQP	RTE	SST	MRPC	CoLA	STS	WNLI	Avg
<i>Single-task single models on dev</i>										
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	90.2/90.2	94.7	92.2	86.6	96.4	90.9	68.0	92.4	91.3	-
<i>Ensembles on test (from leaderboard as of July 25, 2019)</i>										
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

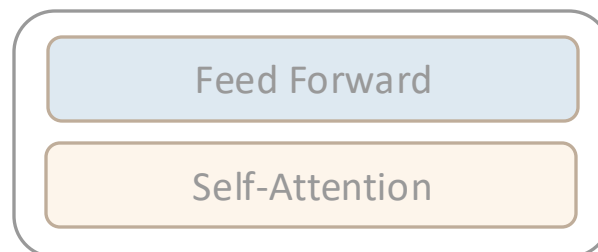
Model	SQuAD 1.1		SQuAD 2.0	
	EM	F1	EM	F1
<i>Single models on dev, w/o data augmentation</i>				
BERT _{LARGE}	84.1	90.9	79.0	81.8
XLNet _{LARGE}	89.0	94.5	86.1	88.8
RoBERTa	88.9	94.6	86.5	89.4
<i>Single models on test (as of July 25, 2019)</i>				
XLNet _{LARGE}			86.3 [†]	89.1 [†]
RoBERTa			86.8	89.8
XLNet + SG-Net Verifier			87.0[†]	89.9[†]

Specific BERTs (2019)

Model	Paper	Link
SciBERT	SciBERT: A Pretrained Language Model for Scientific Text (EMNLP 2019)	https://aclanthology.org/D19-1371.pdf
FinBERT	FinBERT: Financial Sentiment Analysis with Pre-trained Language Models	https://arxiv.org/abs/1908.10063
BioBERT	BioBERT: a pre-trained biomedical language representation model for biomedical text mining	https://arxiv.org/abs/1901.08746
PubmedBERT	Domain-Specific Language Model Pretraining for Biomedical Natural Language Processing	https://arxiv.org/pdf/2007.15779
BlueBERT	Transfer Learning in Biomedical Natural Language Processing: An Evaluation of BERT and ELMo on Ten Benchmarking Datasets	https://arxiv.org/abs/1906.05474
ClinicalBERT	ClinicalBERT: Modeling Clinical Notes and Predicting Hospital Readmission	https://arxiv.org/abs/1904.05342
LegalBERT	LEGAL-BERT: The Muppets straight out of Law School	https://arxiv.org/abs/2010.02559

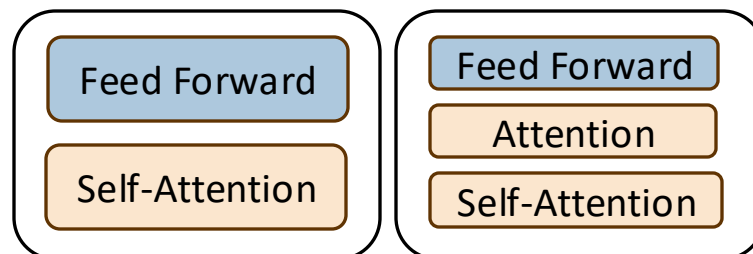
Three Types of Pretraining Ways

Encoders



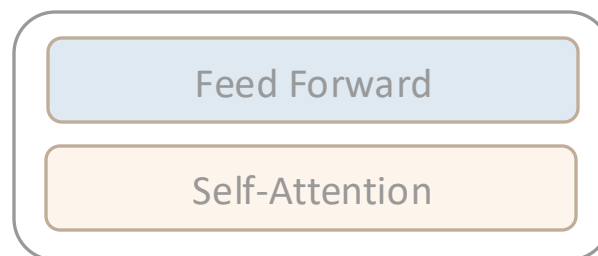
- Bidirectional – capable of looking into the future
- Suitable for downstream task

Encoder-Decoders



- Pros of decoders and encoders?
- Cons of having both during pretraining

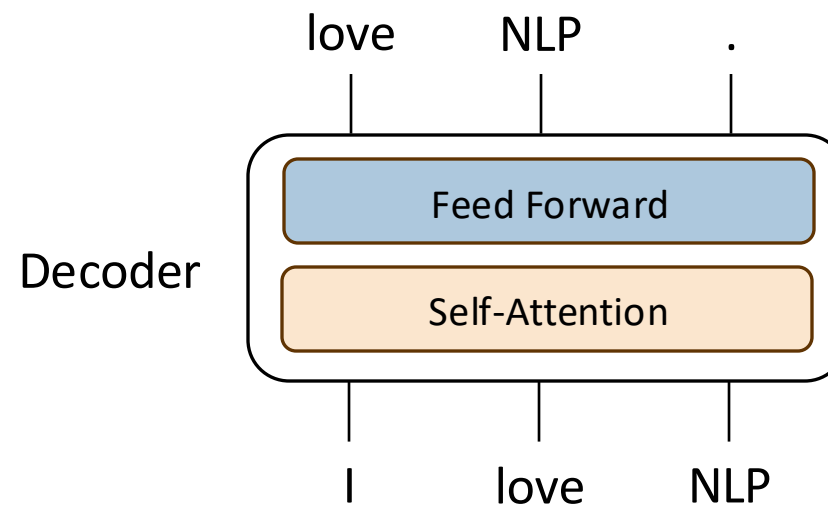
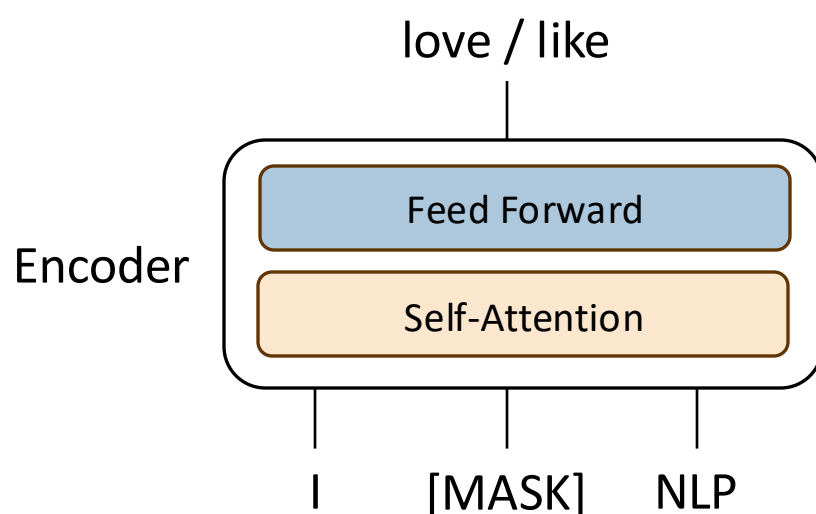
Decoders



- What LMs we have seen so far
- Suitable for generation task
- Not bidirectional

Limitation of Encoders

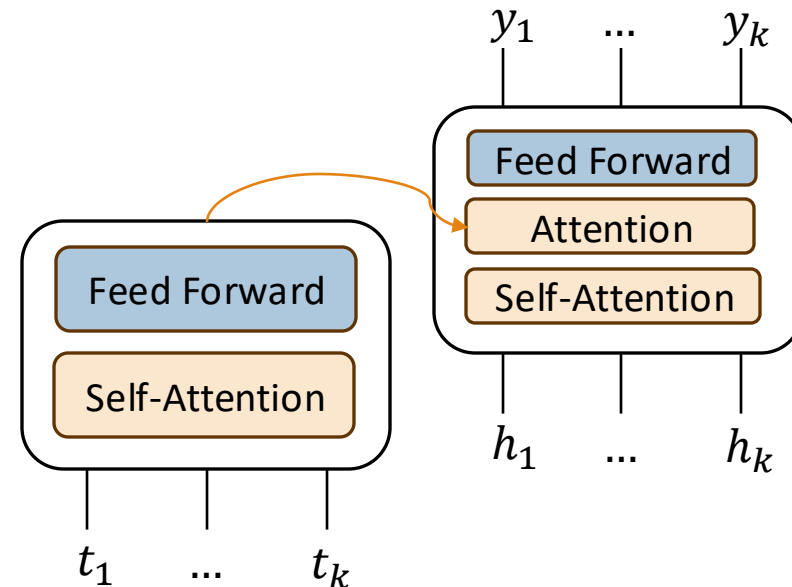
Despite its strong performance on various tasks, it struggles to perform well on **generative tasks** due to the lack of bidirectional context, therefore, when your task involves autoregressive (1-word-at-a-time), consider using a pretrained decoder.



Pretraining Encoder-Decoder

Idea: Capture the context bidirectionally with encoder then use decoder to train the whole model through language modeling.

- Pretraining task
 - **High-level approaches**
 - Language modeling
 - BERT-style
 - Deshuffling
 - **Corruption Strategy**
 - Replace by mask
 - Replace by spans
 - Drop the token



Objective	Inputs	Targets
Prefix language modeling	Thank you for inviting	me to your party last week .
BERT-style Devlin et al. (2018)	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)

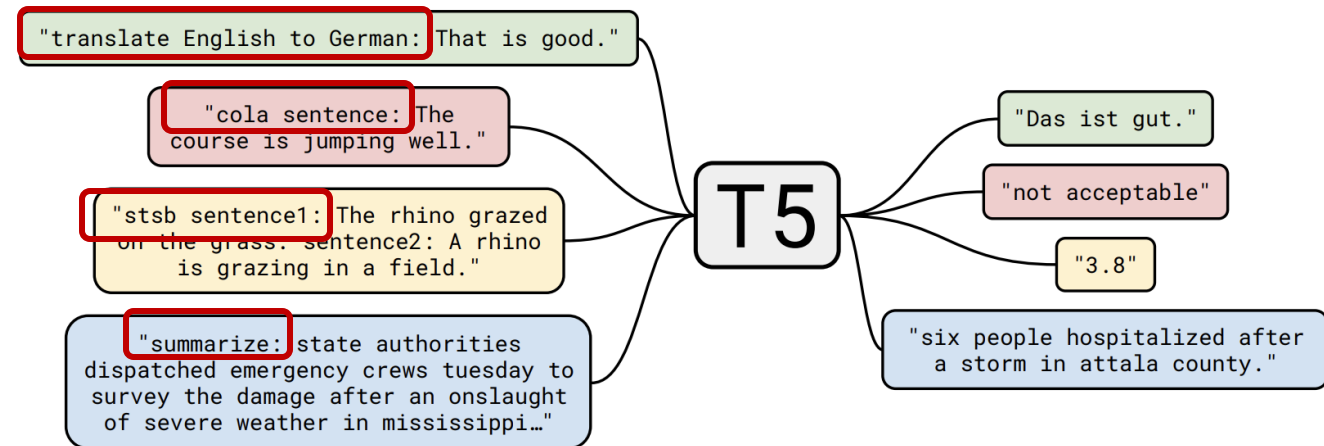
T5 (Text-to-Text Transfer Transformer)

(Raffel et al., Google, 2019)

4 NLP Main Tasks

- Classification
- Content Similarity Measurement
- Sequence Tagging
- Natural Language Generation

Universal
task



T5 (Text-to-Text Transfer Transformer)

(Raffel et al., Google, 2019)

Idea: Replace various spans with different length from the input with unique placeholders; then, reconstruct the removed spans during decoding.

Original text: *“Thank you for inviting me to your party last week.”*

BERT-style masking

Targets Thank you for inviting me to your party last week

T5

Inputs Thank you <M> <M> me to your party *apple* week

Replace spans

Target <X> for inviting <Y> last <Z>

T5

Input Thank you <X> me to your party <Y> week



T5

Later, BERT-style objective, a method originally proposed as a pre-training technique for an encoder-only model trained for classification and span prediction was found to have better performance over the alternative approach.

BERT-style masking

Targets Thank you for inviting me to your party last week

T5

Inputs Thank you <M> <M> me to your party *apple* week

Replace spans

Target <X> for inviting <Y> last <Z>

T5

Input Thank you <X> me to your party <Y> week



T5

“Span” refers to multiple consecutive tokens that have been corrupted. A single unique mask token is then used to replace the entire span, resulting in unlabeled text data being processed into shorter sequences.

BERT-style masking

Targets Thank you for inviting me to your party last week

T5

Inputs Thank you <M> <M> me to your party *apple* week

Replace spans

Target <X> for inviting <Y> last <Z>

T5

Input Thank you <X> me to your party <Y> week

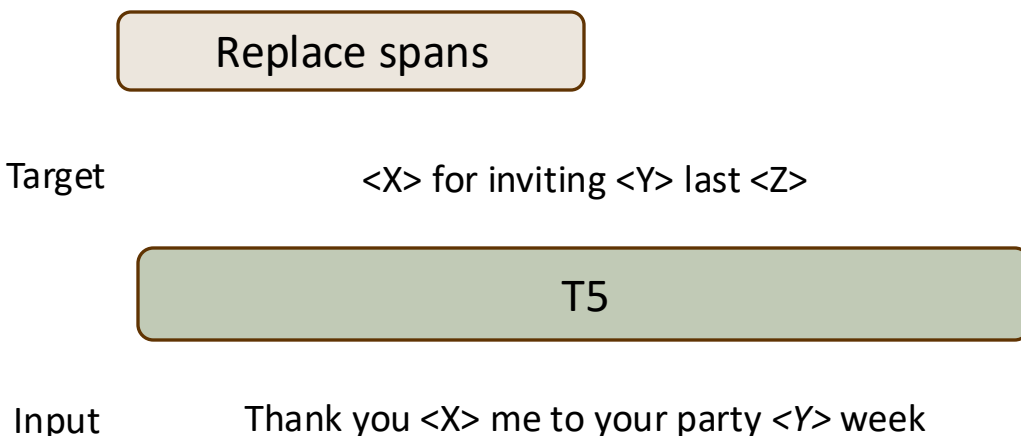
Details about Replace Spans

Our objective of replacing spans can be parameterized by the proportion of tokens to be corrupted and the total number of corrupted. It is also shown that this setting is not sensitive to performance due to slight difference in performance with low corruption rate (below 50%).

- Corruption objective
 - Corruption rate : 15%
 - Corrupted average span length: 3

Both parameters can be used to specified total number of span by using this simple calculation.

$$avg_span_length = \frac{T_{token} \times rate_{corrupt}}{\#_{span}}$$



Details about T5

There are five variants for T5 models:

- T5-small (60M parameters): 6 layers, 512-dim hidden size, 8 attention heads.
- T5-base (220M parameters): 12 layers, 768-dim hidden size, 12 attention heads.
- T5-large (770M parameters): 24 layers, 1024-dim hidden size, 16 attention heads.
- T5-3B: 24 layers, 1024-dim hidden size, 32 attention heads.
- T5-11B: 24 layers, 1024-dim hidden size, 128 attention heads.

Trained on:

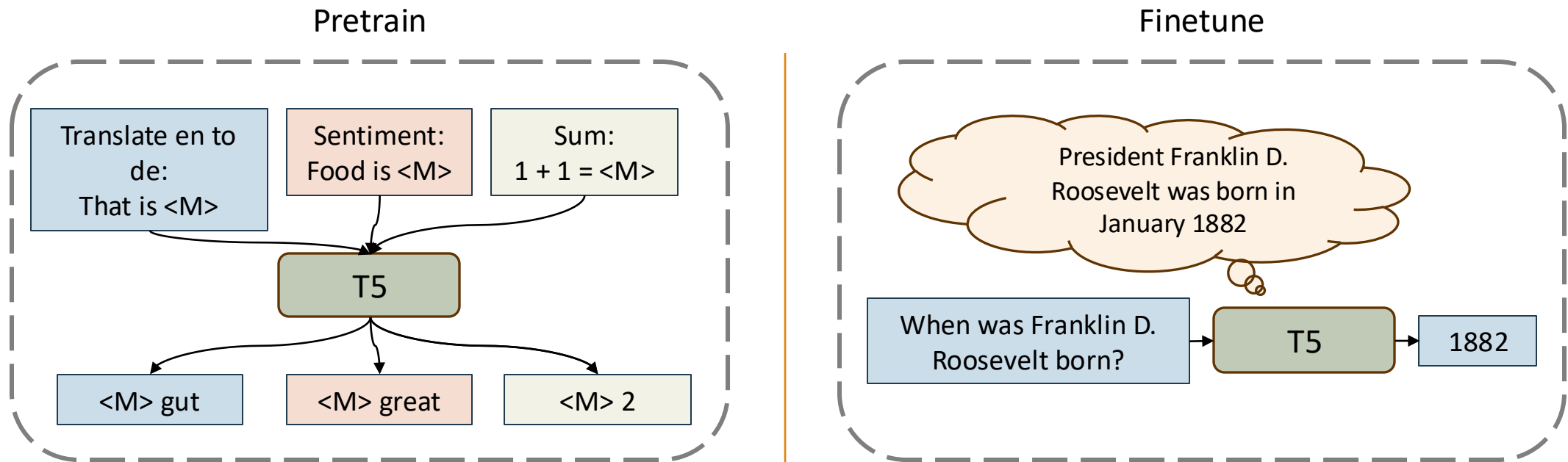
- C4 (Colossal Clean Crawled Corpus, extracted from Common Crawl, 34B words)

Scaling can make a significant boost in performance.

Converts all text-based language problems into a **text-to-text format**.

Training Strategy and Finetune T5

We pre-train T5 by having it learned to fill in dropped-out spans of text (denoted by <M>). To apply T5 to closed-book question answer, we fine-tuned it to answer questions without giving any additional input information or context. This forces T5 to answer questions based on “knowledge” that it learnt during pre-training.

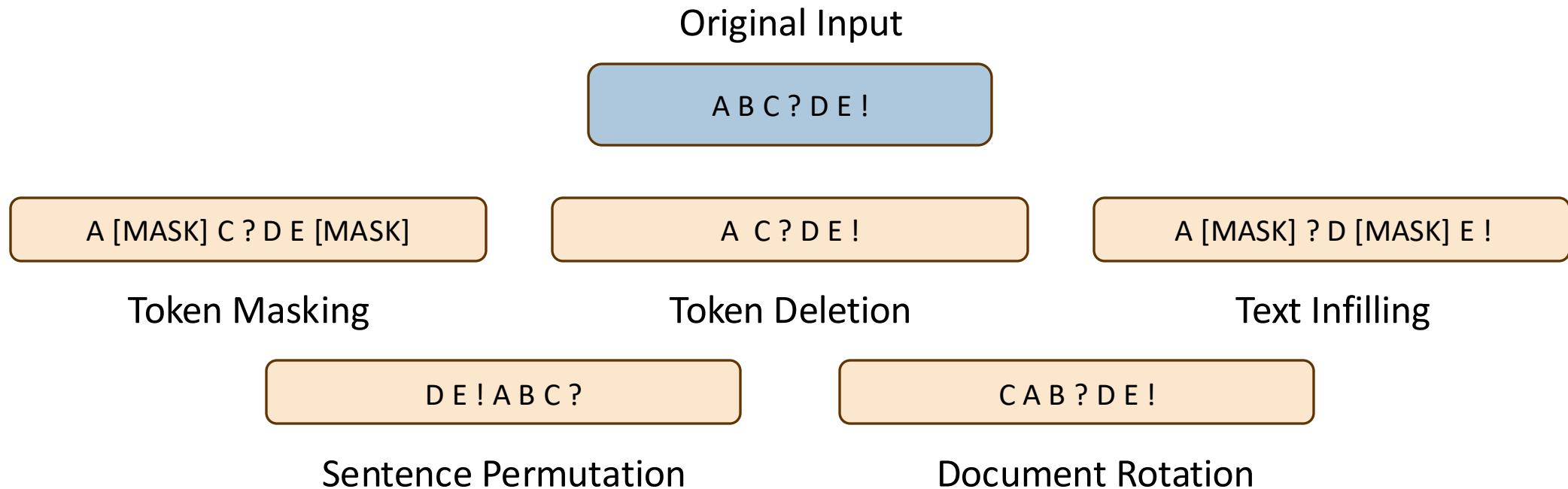




Extensions of T5

BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension (Meta, 2019)

BART (**B**idirectional and **A**uto-**R**egressive **T**ransformers) serves as a **denoising autoencoder** designed to pretrain **sequence-to-sequence** models. BART undergoes two training processes. First, corrupting text using arbitrary noising function. Second, learning a model to reconstruct the original text.



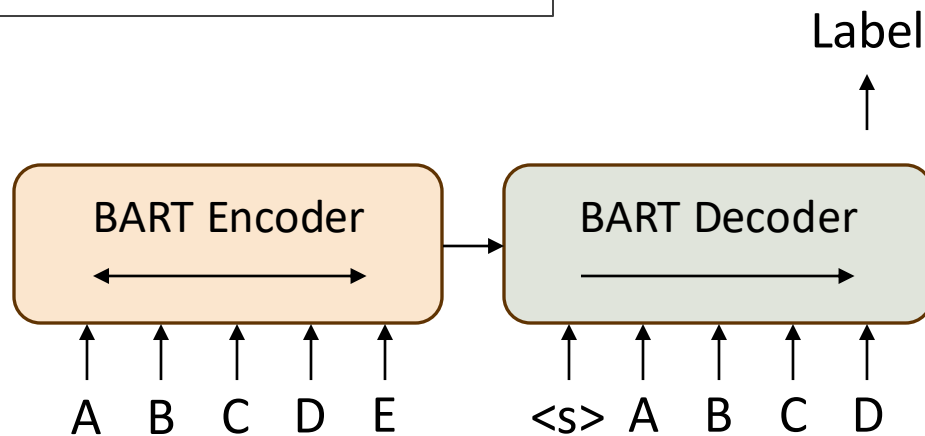
Pretraining tasks of BART

Noise	Method	Goal
Token Masking	Random tokens are sampled and replaced with [MASK] elements.	Teach the model to predict masked token.
Token Deletion	Random tokens are deleted from the input.	Teach the model to predict masked token and its position.
Text Infilling	Like SpanBERT, but 0-length spans correspond to the insertion of [MASK] tokens.	Teach the model to predict how many tokens are missing from a span.
Sentence Permutation	A document is divided into sentences based on full stops, and these sentences are shuffled in a random order.	Teach the model to clarify the relationship between two sentences.
Document Rotation	A token is chosen uniformly at random, and the document is rotated so that it begins with that token.	Teach the model to identify the start of the document.

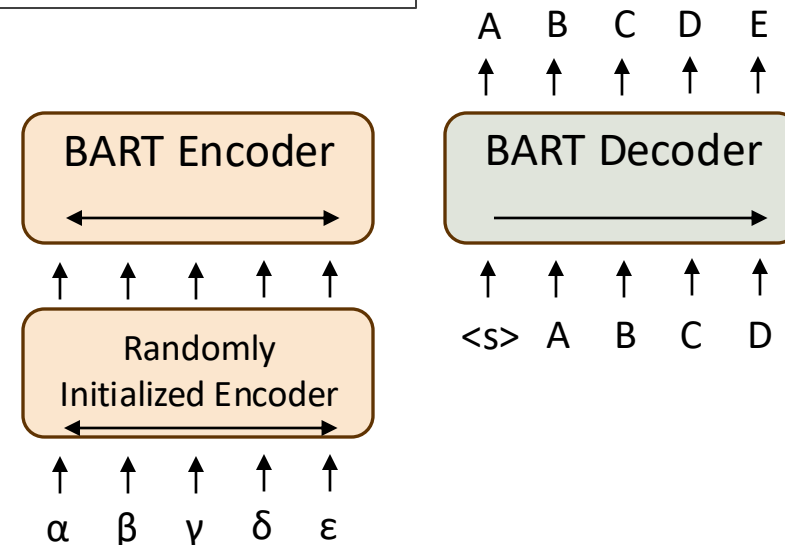
Finetuning BART

For **classification** problems, both the encoder and decoder receive the same input, and the representation from the final output is utilized; For **generation**, BART's encoder is replaced with newly trained encoder the word embeddings in BART. The new encoder can use a separate vocabulary from original BART model.

Classification Problems

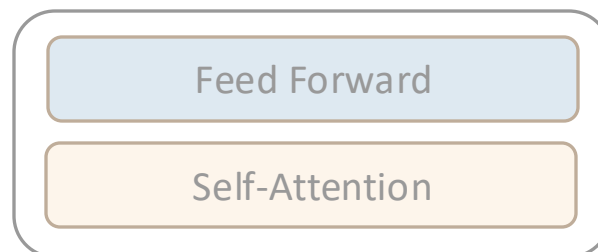


Generation Problems



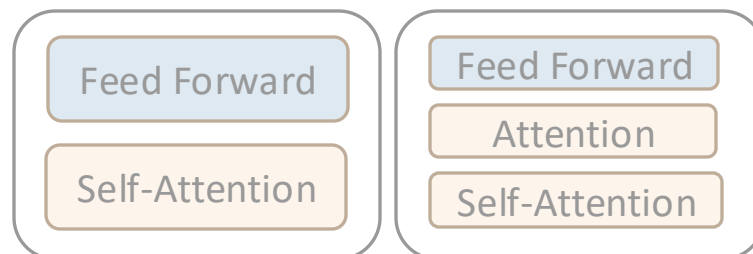
Three Types of Pretraining Ways

Encoders



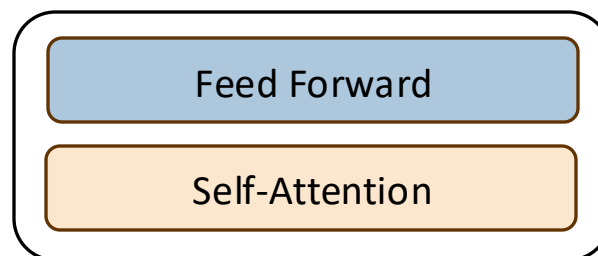
- Bidirectional – capable of looking into the future
- Suitable for downstream task

Encoder-Decoders



- Pros of decoders and encoders?
- Cons of having both during pretraining

Decoders



- What LMs we have seen so far
- Suitable for generation task
- Not bidirectional

GPT (Radford et al., 2018)

Improving Language Understanding by Generative Pre-Training

Details

- Transformer decoder with 12 layers, 117M parameters
- 768-dimensional hidden states, 3072-dimensional feed-forward hidden layers
- Byte-pair encoding with 40,000 merges

Dataset

- Trained on BooksCorpus: over 7000 unique books
 - Contains long spans of contiguous text, for learning long-distance dependencies

Additional Insights

- The acronym “GPT” never showed up in the original paper
 - Could stand for “Generative PreTraining” or “Generative Pretrained Transformer”





Alec Radford

Cited by

	All	Since 2020
Citations	281402	267136
h-index	41	41
i10-index	48	48

TITLE	CITED BY	YEAR
Language models are few-shot learners T Brown, B Mann, N Ryder, M Subbiah, JD Kaplan, P Dhariwal, ... Advances in neural information processing systems 33, 1877-1901	56379	2020
Learning Transferable Visual Models From Natural Language Supervision A Radford, JW Kim, C Hallacy, A Ramesh, G Goh, S Agarwal, G Sastry, ... https://cdn.openai.com/papers ...	44201	2021
Language Models are Unsupervised Multitask Learners A Radford, J Wu, R Child, D Luan, D Amodei, I Sutskever Technical report, OpenAi	32960 *	2019
Proximal policy optimization algorithms J Schulman, F Wolski, P Dhariwal, A Radford, O Klimov arXiv preprint arXiv:1707.06347	30750	2017
Unsupervised representation learning with deep convolutional generative adversarial networks A Radford, L Metz, S Chintala arXiv preprint arXiv:1511.06434	21100	2015
Gpt-4 technical report J Achiam, S Adler, S Agarwal, L Ahmad, I Akkaya, FL Aleman, D Almeida, ... arXiv preprint arXiv:2303.08774	18637	2023
Improving language understanding by generative pre-training A Radford, K Narasimhan, T Salimans, I Sutskever	16937	2018

GPT3

CLIP

GPT2

DCGAN

GPT



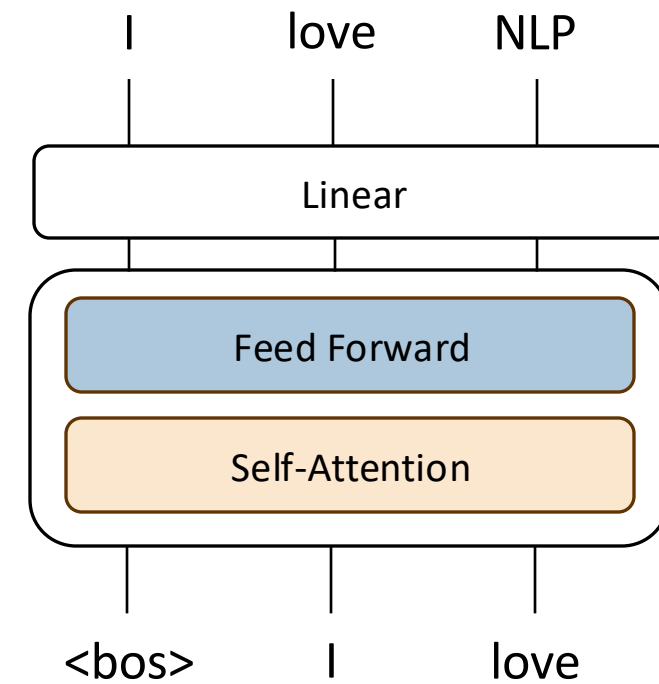
Pretraining Decoder

Idea: Pretrain decoders on language modeling, then use them as generators. This is helpful in tasks where the output is a sequence with a vocabulary like that at pretraining time.

- Pretraining task
 - Next Token Prediction

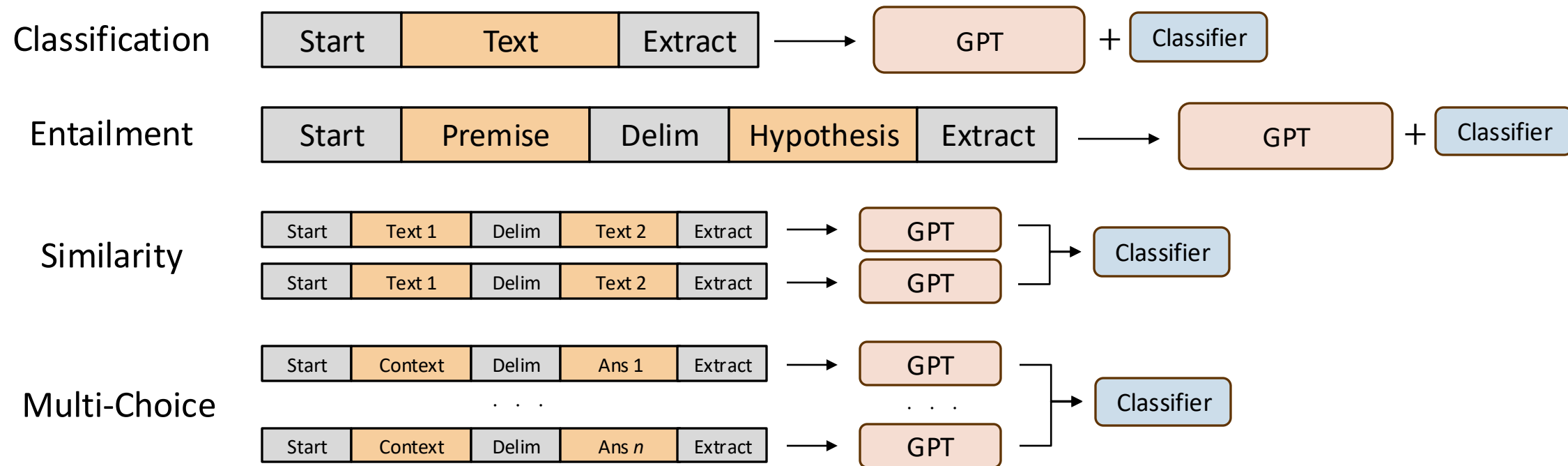
$$p(x) = \prod_{i=1}^n p(s_n | s_1, \dots, s_{n-1})$$

Since language follows a specific rule and order, therefore, we leverage conditional probabilities to help us determine the most probable next word based on the preceding sequence of texts.



Finetune GPT

Idea: Convert all structured inputs into sequence of tokens to be processed by our pre-trained model, followed by a linear classifier



GPT Family

Model	Pretraining method	Parameters	Dataset	Release
GPT-1	Transformer decoder followed by linear-softmax	117 M	BooksCorpus 4.5 GB	2018.06.11
GPT-2	Same as GPT-1 but different normalized layer	1.5 B	WebText 40 GB	2019.02.14
GPT-3	Larger version of GPT-2	175 B	CommonCrawl 45 TB	2020.05.15
GPT-4	Trained with RLHF	> 1.5 T	Undisclosed	2023.03.14

Text generation era starts!

GPT-3, In-Context learning, and LLMs

Language Models are Few-Shot Learners

GPT-3 (Brown et al., 2020) (75 pages)

So far, there are two ways we interact with pre - trained models

1. Sample from the distributions they define
2. Fine-tune them on a task we care about

However, large models are shown to have this **emergence ability** which allows them to perform some kind of learning without gradient steps.

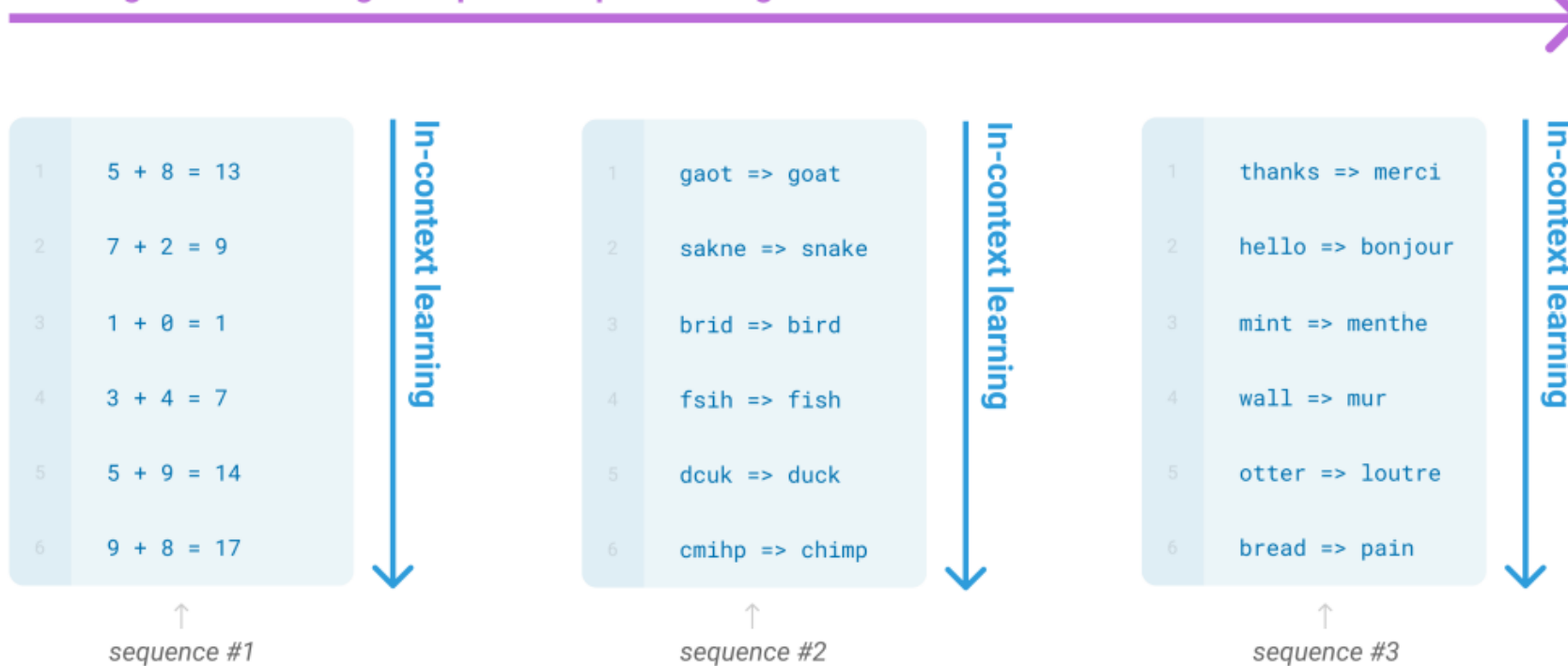
GPT-3, with 175 billion parameters, is a good example of this.



GPT-3, In-Context learning, and LLMs

Very large language models have ability to **learn through examples** provided within their contexts **without gradient steps**. This is referred to as in-context learning ability.

Learning via SGD during unsupervised pre-training



GPT3, ICL

Zero-shot

The model predicts the answer given only a natural language description of the task. No gradient updates are performed.

```
1 Translate English to French: ← task description
2 cheese => ..... ← prompt
```

One-shot

In addition to the task description, the model sees a single example of the task. No gradient updates are performed.

```
1 Translate English to French: ← task description
2 sea otter => loutre de mer ← example
3 cheese => ..... ← prompt
```

Few-shot

In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.

```
1 Translate English to French: ← task description
2 sea otter => loutre de mer ← examples
3 peppermint => menthe poivrée ←
4 plush girafe => girafe peluche ←
5 cheese => ..... ← prompt
```



GPT3 family

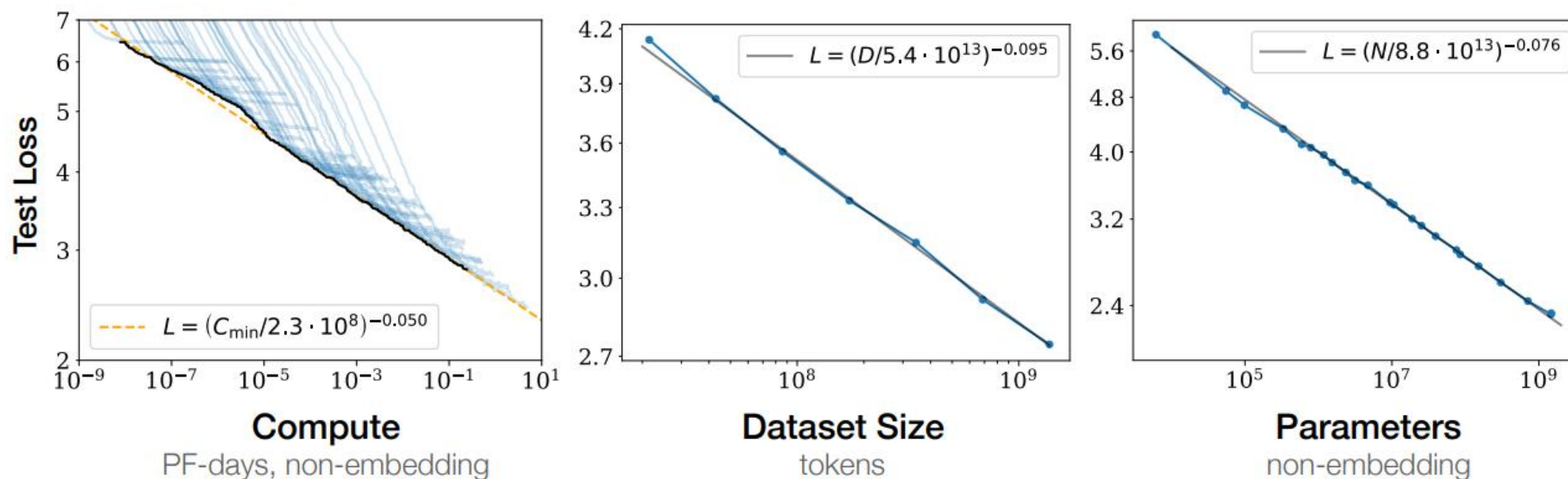
Model Name	n_{params}	n_{layers}	d_{model}	n_{heads}	d_{head}	Batch Size	Learning Rate
GPT-3 Small	125M	12	768	12	64	0.5M	6.0×10^{-4}
GPT-3 Medium	350M	24	1024	16	64	0.5M	3.0×10^{-4}
GPT-3 Large	760M	24	1536	16	96	0.5M	2.5×10^{-4}
GPT-3 XL	1.3B	24	2048	24	128	1M	2.0×10^{-4}
GPT-3 2.7B	2.7B	32	2560	32	80	1M	1.6×10^{-4}
GPT-3 6.7B	6.7B	32	4096	32	128	2M	1.2×10^{-4}
GPT-3 13B	13.0B	40	5140	40	128	2M	1.0×10^{-4}
GPT-3 175B or “GPT-3”	175.0B	96	12288	96	128	3.2M	0.6×10^{-4}



Dataset	Quantity (tokens)	Weight in training mix	Epochs elapsed when training for 300B tokens
Common Crawl (filtered)	410 billion	60%	0.44
WebText2	19 billion	22%	2.9
Books1	12 billion	8%	1.9
Books2	55 billion	8%	0.43
Wikipedia	3 billion	3%	3.4

Scaling Laws (Kaplan et al., 2020)


Scaling laws are simple, predictive rules for model performance. Increasing the model size, dataset size, and the amount of computation during training lead to better performance which increase smoothly.



Very Large Language Models

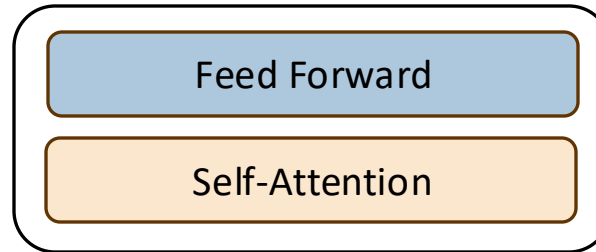
Model	Parameters	Publisher	Release
GPT-3	175B	OpenAI	2020.05
BLOOM	176B	BigScience	2022.07
Flan-PaLM	540B	Google	2022.10

Large Language Models for Community

Model	Architecture	Parameters	Publisher	Release
Llama 2	Decoder-only	7B, 13B, 70B	Meta	2023.07.18
Mistral	Decoder-only	7B, 8×7B	MistralAI 	2023.09.27
Phi 2	Decoder-only	2.7B	Microsoft	2023.12.12
Gemma	Decoder-only	2B, 7B	Google	2024.02.21

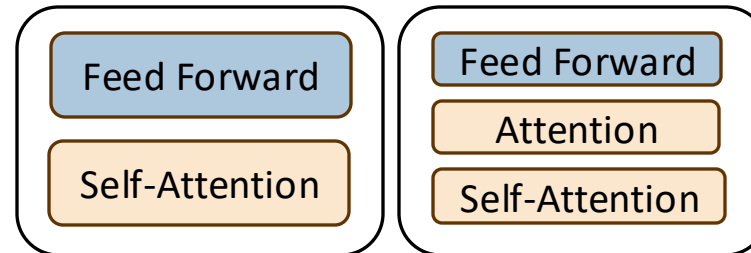
Takeaways

Encoders



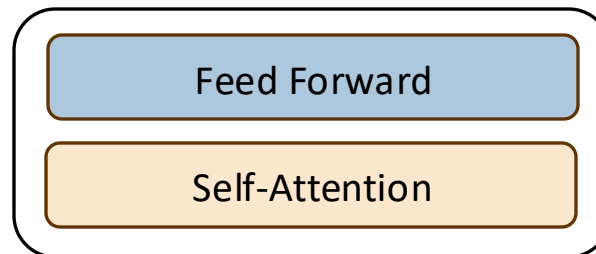
- BERT family
- MLM and NSP task for pretraining
- Not good at performing generation

Encoder-Decoders



- T5, BART
- Span corruption for boosting MLM
- Targeting text-to-text format model

Decoders



- GPT family
- Pure language modeling
- Most popular backbone nowadays