

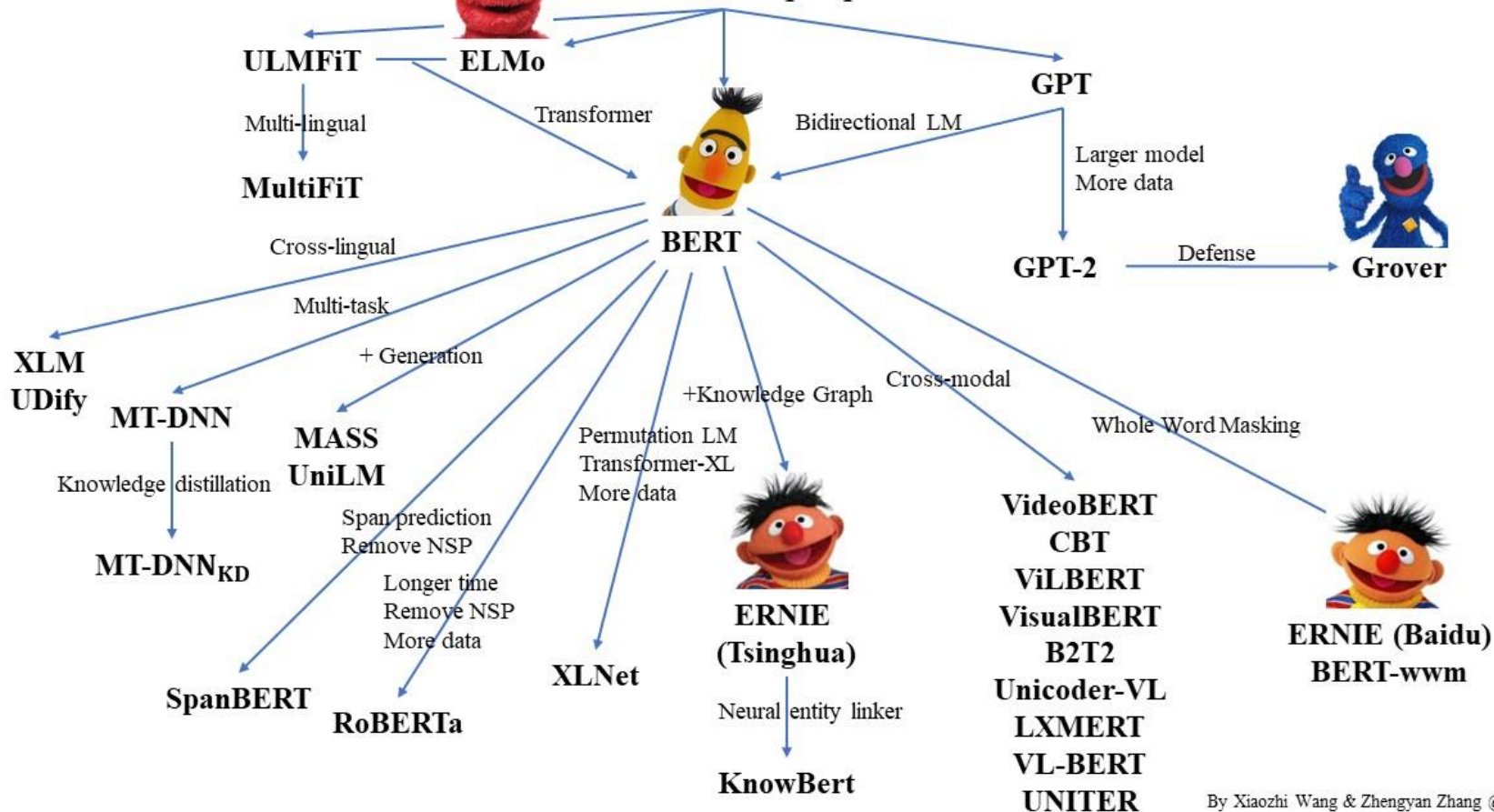
Advanced Pre-training for Language Models



Semi-supervised Sequence Learning

context2Vec

Pre-trained seq2seq



Pre-Training Objectives

- Pre-training with abundantly available data
- Use Self-Supervised Learning
- GPT: Language Modeling
 - Predict next word given left context
- Word2Vec, BERT, RoBERTa: Masked Language Modeling
 - Predict missing word given a certain context

Next Sentence Prediction (NSP)

Sentence 1

Sentence 2

Next Sentence?

I am going outside.

I will be back after 6.

YES

I am going outside.

You know nothing John snow.

NO

Train the CLS and SEP tokens

Input = [CLS] the man went to [MASK] store [SEP] he bought a gallon [MASK] milk [SEP]

Label = IsNext

Input = [CLS] the man [MASK] to the store [SEP] penguin [MASK] are flight ##less birds [SEP]

Label = NotNext

Strategy used in BERT but shown to be ineffective in RoBERTa

Sentence Order Prediction (SOP)

- NSP primarily requires **topic-prediction**
- SOP can be used to focus on inter-sentence coherence (ALBERT)

Input = [CLS] the man went to [MASK] store [SEP] he bought a gallon [MASK] milk [SEP]

Label = CorrectOrder

Input = [CLS] he bought a gallon [MASK] milk [SEP] the man went to [MASK] store [SEP]

Label = InCorrectOrder

Masking Strategy in BERT

- Mask 15% of the input tokens

the man went to the [MASK] to buy a [MASK] of milk

store gallon

↑ ↑

- **Problem 1:** Assumes that the MASK tokens are independent of each other
- **Problem 2:** [MASK] tokens never appear during fine-tuning

Masking Strategy in BERT

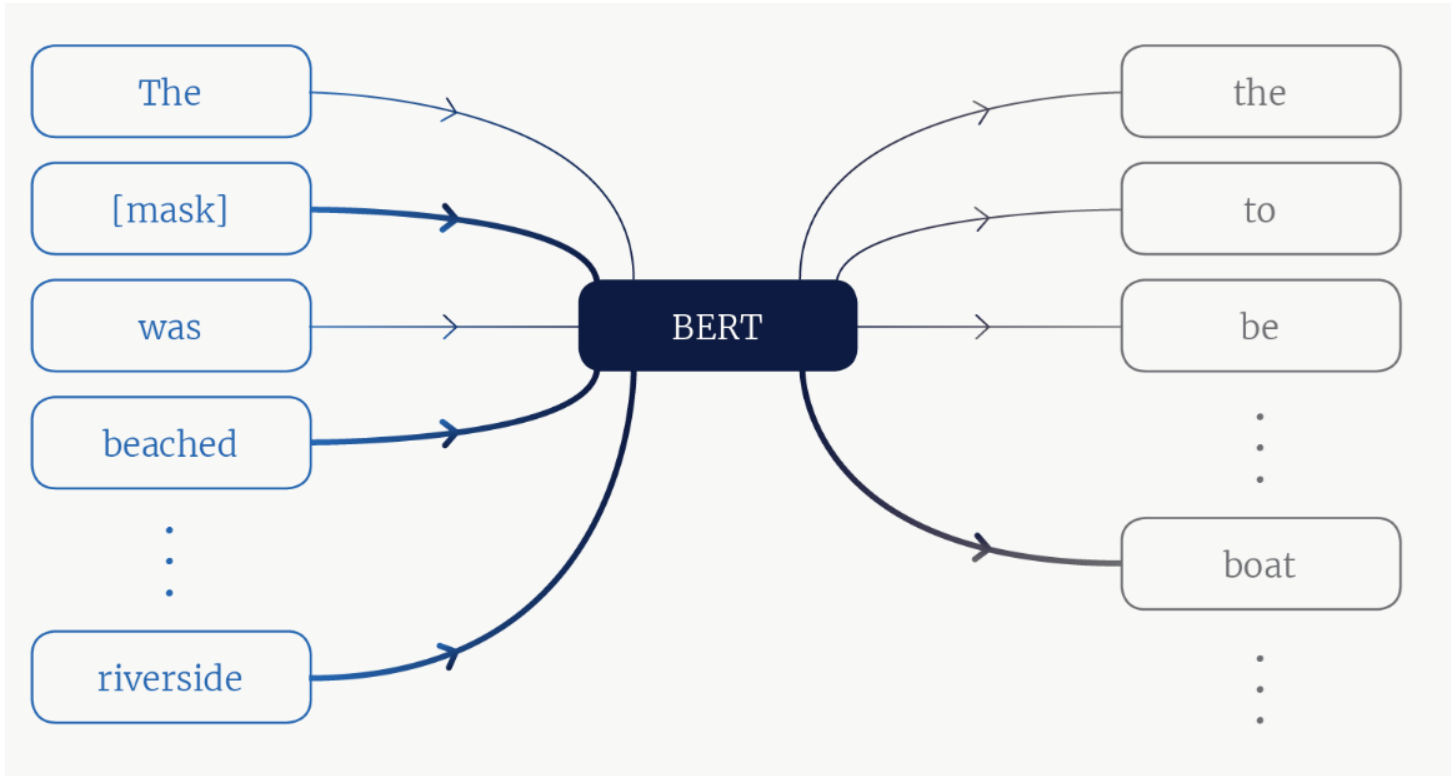
- Mask 15% of the input tokens
- **80% of the time**: Replace the word with the [MASK] token, e.g., my dog is hairy → my dog is [MASK]

Masking Strategy in BERT

- Mask 15% of the input tokens
- **80% of the time**: Replace the word with the [MASK] token, e.g., my dog is hairy → my dog is [MASK]
- **10% of the time**: Replace the word with a random word, e.g., my dog is hairy → my dog is apple

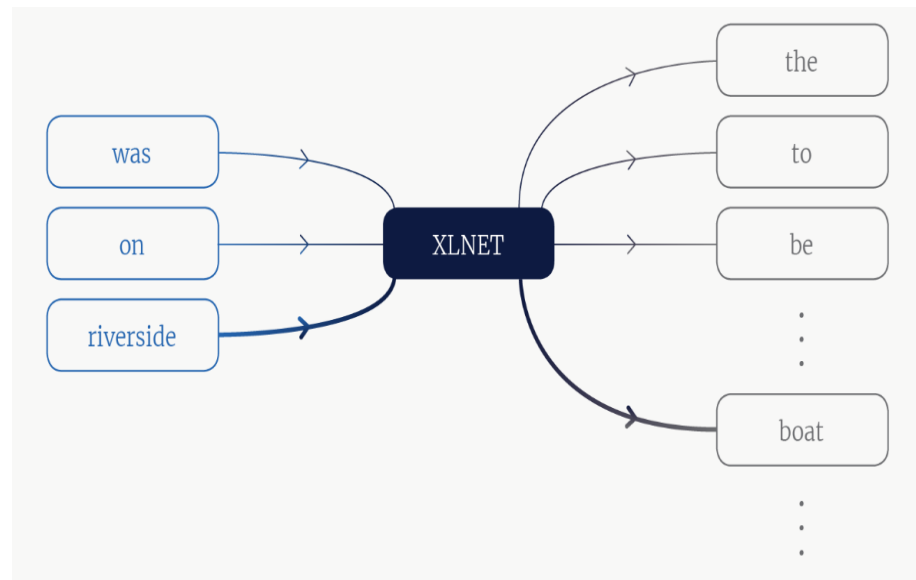
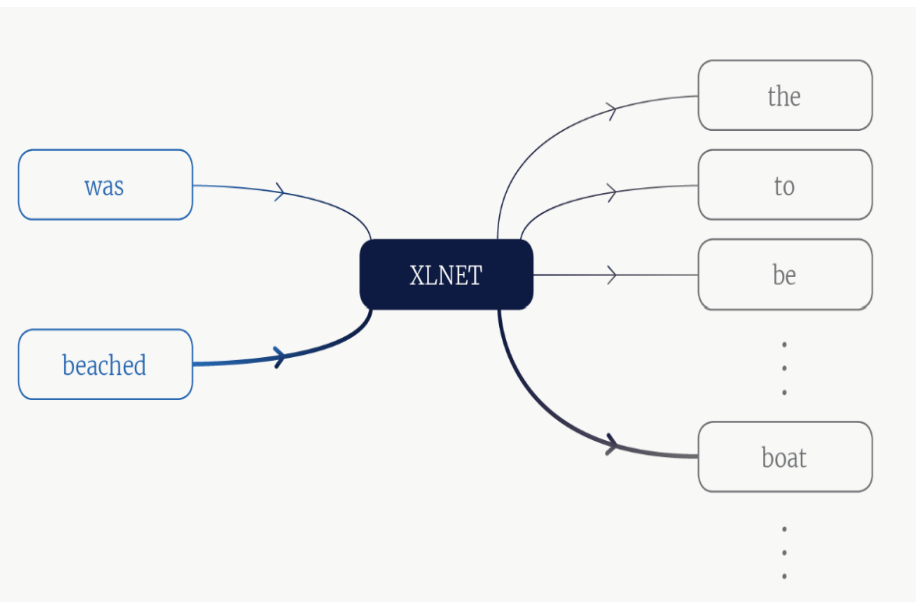
Masking Strategy in BERT

- Mask 15% of the input tokens
- **80% of the time**: Replace the word with the [MASK] token, e.g., my dog is hairy → my dog is [MASK]
- **10% of the time**: Replace the word with a random word, e.g., my dog is hairy → my dog is apple
- **10% of the time**: Keep the word unchanged, e.g., my dog is hairy → my dog is hairy.



The boat was beached on the riverside

Permutation Modeling using XLNET



The boat was beached on the riverside

Permutation Language Modeling

- Predict words given any possible permutation of input words
- Order of words is maintained, as in test-time

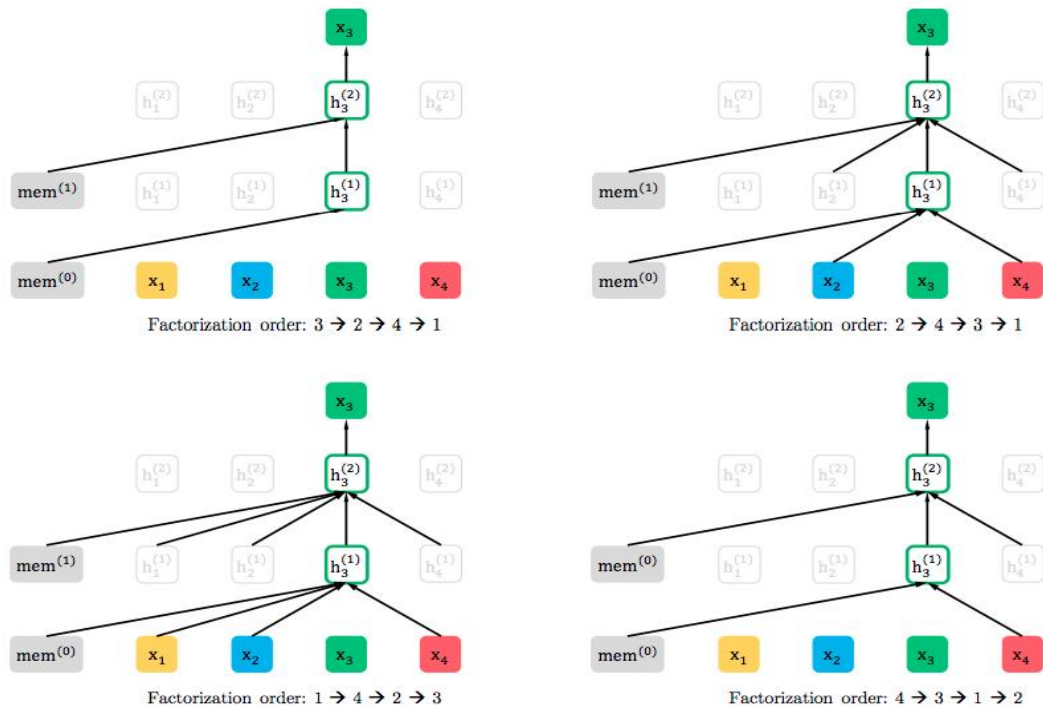


Figure 1: Illustration of the permutation language modeling objective for predicting x_3 given the same input sequence \mathbf{x} but with different factorization orders.

Approximate Computation Time

ULMFit	1 GPU day
ELMo	40 GPU days
BERT	450 GPU days
XLNet	2000 GPU days

Granularity of Masking

- BERT chooses word-pieces but this is sub-optimal
- Philammon → Phil ##am #mon
- Not much information to be gained by predicting at word piece level

Granularity of Masking

- BERT-wwm (**whole word masking**): Always mask entire word
- BERT: Phil ##am [MASK] was a great singer
- BERT-wwm: [MASK] [MASK] [MASK] was a great singer

SpanBERT

- [MASK] Delhi, the Indian Capital is known for its rich heritage.
- Easier to predict “New”, given that we already know Delhi
- Instead of masking individual words, mask contiguous spans
- [MASK] [MASK], the Indian Capital is known for its rich heritage.

Knowledge Masking Strategies in ERNIE

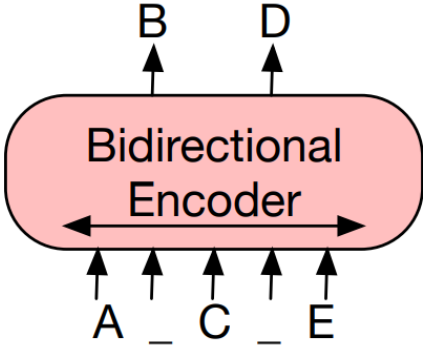
Sentence	Harry	Potter	is	a	series	of	fantasy	novels	written	by	British	author	J.	K.	Rowling
Basic-level Masking	[mask]	Potter	is	a	series	[mask]	fantasy	novels	[mask]	by	British	author	J.	[mask]	Rowling
Entity-level Masking	Harry	Potter	is	a	series	[mask]	fantasy	novels	[mask]	by	British	author	[mask]	[mask]	[mask]
Phrase-level Masking	Harry	Potter	is	[mask]	[mask]	[mask]	fantasy	novels	[mask]	by	British	author	[mask]	[mask]	[mask]

ERNIE 2.0

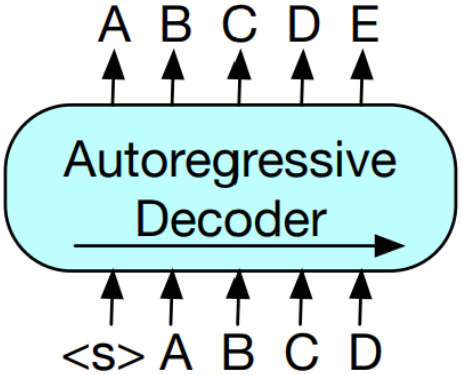
- Word-Level Pre-training:
 - Capitalization Prediction Task
 - Token-Document Relation Prediction Task (Frequency)
- Sentence-Level Pre-training:
 - Sentence Reordering Task
 - Sentence Distance Task
- Semantic-Level Pre-training:
 - Discourse Relation Task
 - IR Relevance Task

Pre-Trained Encoder-Decoder Models

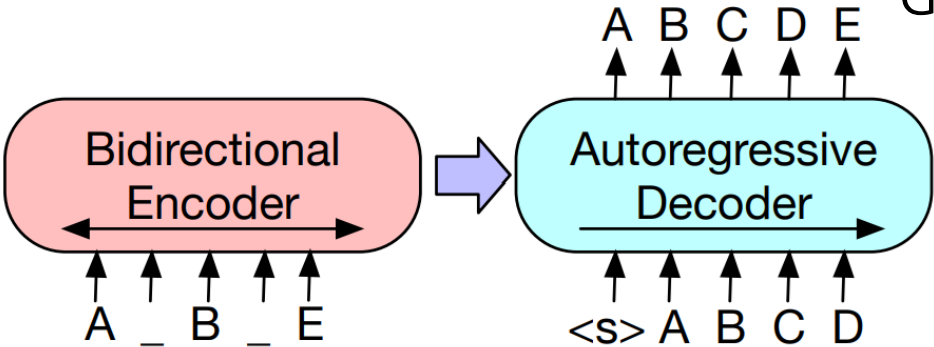
Pre-training Encoder-Decoder models



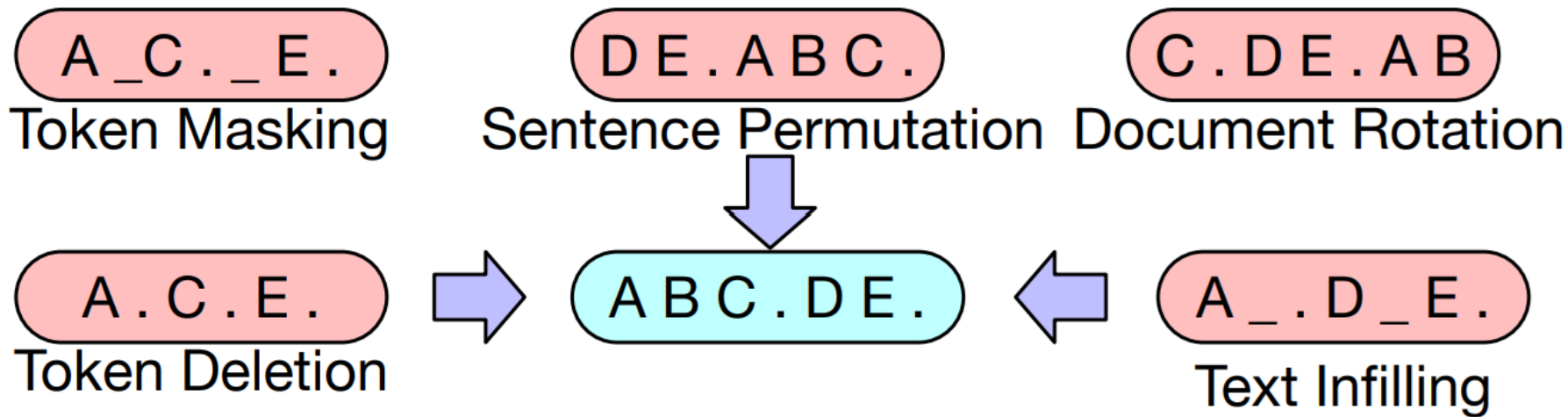
BERT



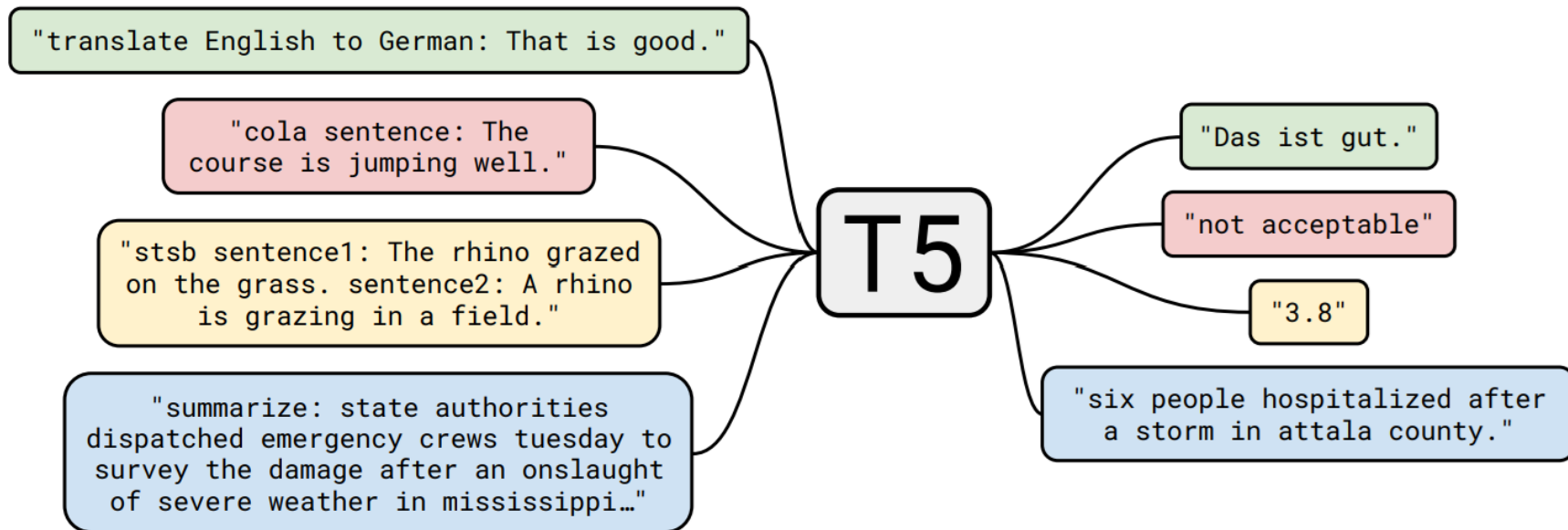
GPT



BART from Facebook



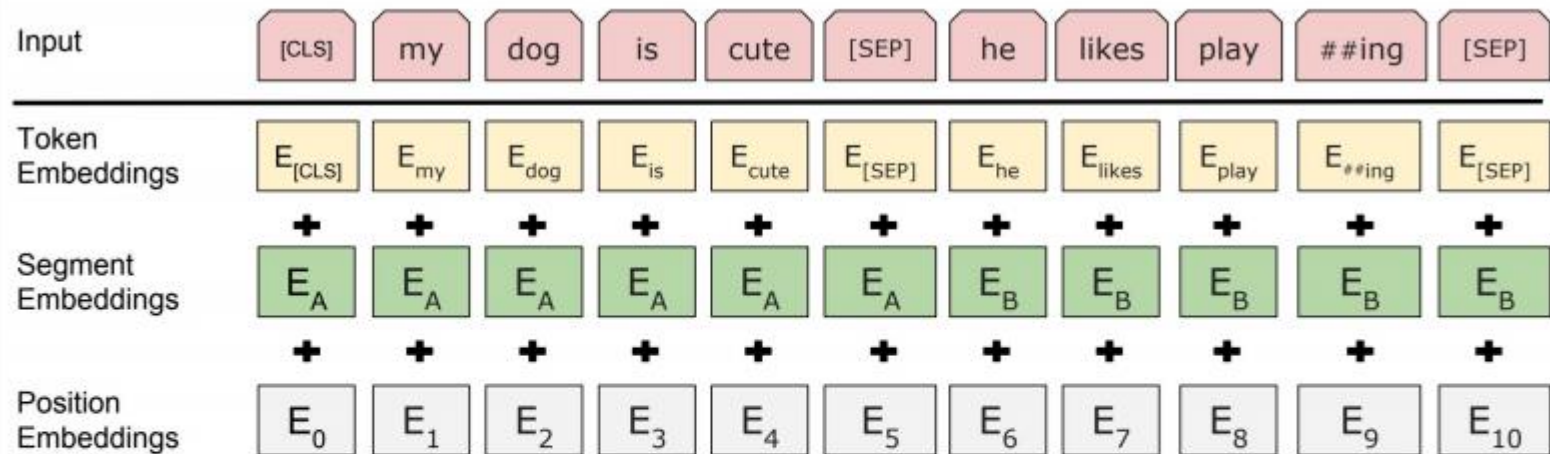
T5 from Google



T5: Unsupervised Pre-training Objectives

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 .	<i>(original text)</i>
Deshuffling	party me for your to . last fun you inviting week Thank	<i>(original text)</i>
MASS-style Song et al. (2019)	Thank you <M> <M> me to your party <M> week .	<i>(original text)</i>
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>

Tokenization Strategies



- Use 30,000 WordPiece vocabulary on input.
- Each token is sum of three embeddings

Byte Pair Encoding (cs224n slides)

- Originally a **compression** algorithm:
 - Most frequent **byte** pair \mapsto a new **byte**.

Replace bytes with character ngrams

(though, actually, some people have done interesting things with bytes)

Rico Sennrich, Barry Haddow, and Alexandra Birch. **Neural Machine Translation of Rare Words with Subword Units**. ACL 2016.

<https://arxiv.org/abs/1508.07909>

<https://github.com/rsennrich/subword-nmt>

<https://github.com/EdinburghNLP/nematus>

Byte Pair Encoding

- A **word segmentation** algorithm:
 - Though done as bottom up clustering
 - Start with a unigram vocabulary of all (Unicode) **characters** in data
 - Most frequent **ngram pairs** \mapsto a new **ngram**

- A **word segmentation** algorithm:
 - Start with a vocabulary of **characters**
 - Most frequent **ngram pairs** \mapsto a new **ngram**

Dictionary

5 l o w
2 l o w e r
6 n e w e s t
3 w i d e s t

Vocabulary

l, o, w, e, r, n, w, s, t, i, d

Start with all characters
in vocab

- A **word segmentation** algorithm:
 - Start with a vocabulary of **characters**
 - Most frequent **ngram pairs** \mapsto a new **ngram**

Dictionary

5 l o w
2 l o w e r
6 n e w e s t
3 w i d e s t

Vocabulary

l, o, w, e, r, n, w, s, t, i, d, e s

Add a pair (e, s) with freq 9

- A **word segmentation** algorithm:
 - Start with a vocabulary of **characters**
 - Most frequent **ngram pairs** \mapsto a new **ngram**

Dictionary

5 l o w
2 l o w e r
6 n e w e s t
3 w i d e s t

Vocabulary

l, o, w, e, r, n, w, s, t, i, d, e s, e s t

Add a pair (e s, t) with freq 9

- A **word segmentation** algorithm:
 - Start with a vocabulary of **characters**
 - Most frequent **ngram pairs** \mapsto a new **ngram**

Dictionary

5 **lo w**
2 **lo w e r**
6 **n e w e s t**
3 **w i d e s t**

Vocabulary

l, o, w, e, r, n, w, s, t, i, d, e, s, e, s, t, **lo**

Add a pair (l, o) with freq 7

Byte Pair Encoding

- Have a target vocabulary size and stop when you reach it
- Do deterministic longest piece segmentation of words
- Segmentation is only within words identified by some prior tokenizer (commonly Moses tokenizer for MT)
- **Automatically decides** vocab for system
 - No longer strongly “word” based in conventional way

Wordpiece model

- Rather than char n-gram count, uses a greedy approximation to maximizing language model log likelihood to choose the pieces
- Add n-gram that maximally reduces perplexity
- [[link1](#)] and [[link2](#)] contain further details

CANINE: Character-level Pre-trained Encoder

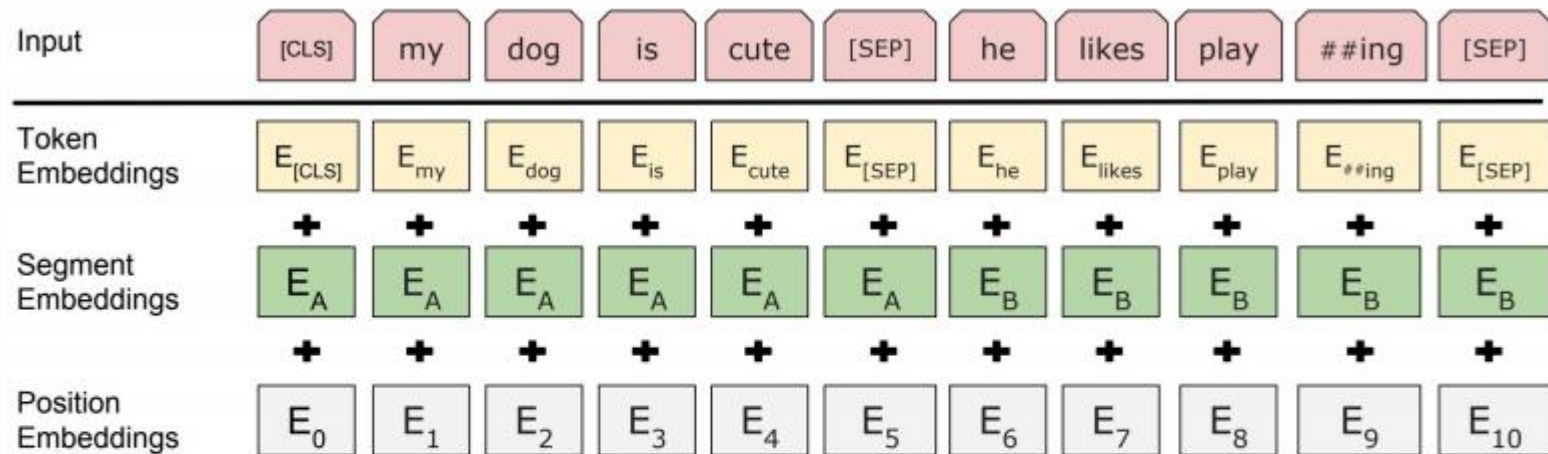
- **Issues:**
- Results in much longer sequence-lengths
- 143K unicode characters

- Down-sample input embeddings (Convolution)
- Hash functions to reduce embedding space

ByT5: Byte-level Seq2Seq

- Operate directly on byte-representations
 - Only 256 input embeddings!
- Embeddings occupy 66% of T5-base
- “Unbalanced”-Seq2Seq
 - 6 layer encoder, 2 layer decoder (3:1)

Positional Embeddings



- Use 30,000 WordPiece vocabulary on input.
- Each token is sum of three embeddings

Absolute Positional Embeddings in BERT

- Transformer architecture suggested using sinusoidal embeddings
- BERT instead learnt absolute positional embeddings
- Fine tuned a randomly initialized embedding matrix
- Size: $512 * 768$
- To reduce training time: 90% of the time trained with sequence size of 128

Relative Positional Embeddings

$$z_i = \sum_{j=1}^n \alpha_{ij} (x_j W^V)$$

$$z_i = \sum_{j=1}^n \alpha_{ij} (x_j W^V + a_{ij}^V)$$

$$e_{ij} = \frac{(x_i W^Q)(x_j W^K)^T}{\sqrt{d_z}}$$

$$e_{ij} = \frac{x_i W^Q (x_j W^K + a_{ij}^K)^T}{\sqrt{d_z}}$$

$$a_{ij}^K = w_{\text{clip}(j-i, k)}^K$$

$$a_{ij}^V = w_{\text{clip}(j-i, k)}^V$$

$$\text{clip}(x, k) = \max(-k, \min(k, x))$$

RPE(j-i, k)

Relative Positional Embeddings in T5

$$Att_{ij} = Q_i * K_j + RPE_{clip}(j-i, 128)$$

Only 32 embeddings, scaled logarithmically

Long Input Transformers

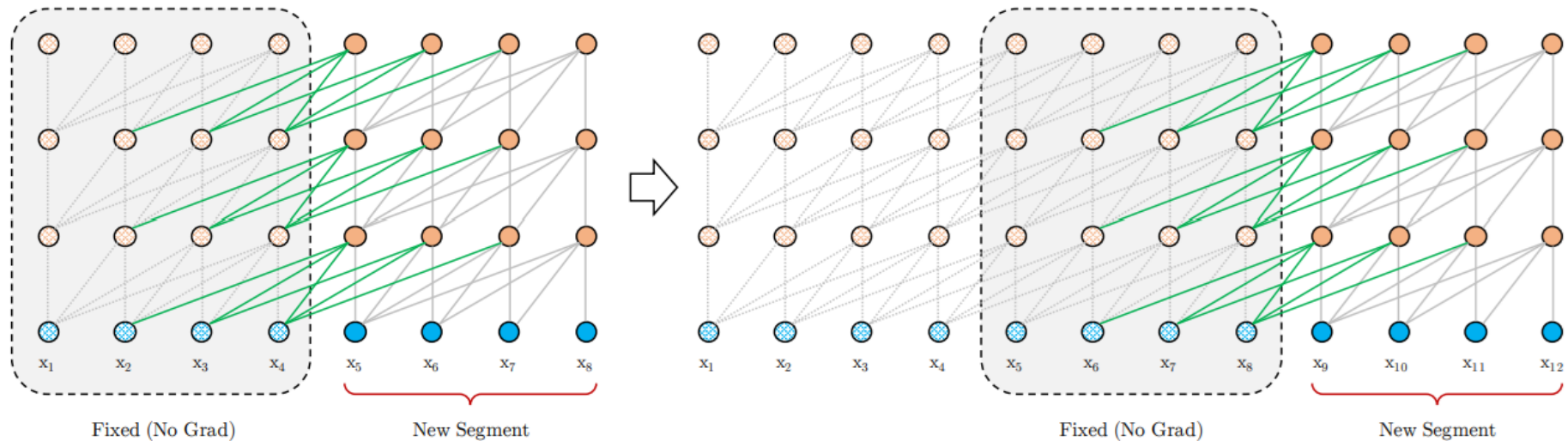
Attention in Transformers

- Every word attends over every other word
- $O(n^2)$ complexity, compared to $O(n)$ complexity of LSTM
- Inherently unscalable to long sequences

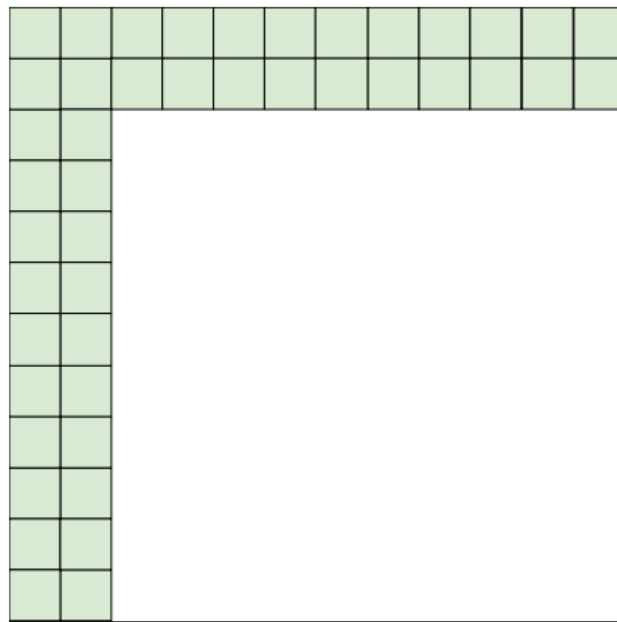
Transformer-XL: Recurrence in Transformers

- Chunk the text into segments
- Attend over current and previous segments
- Don't pass gradient to previous segment
- Faster training and inference (1800x)

Transformer-XL

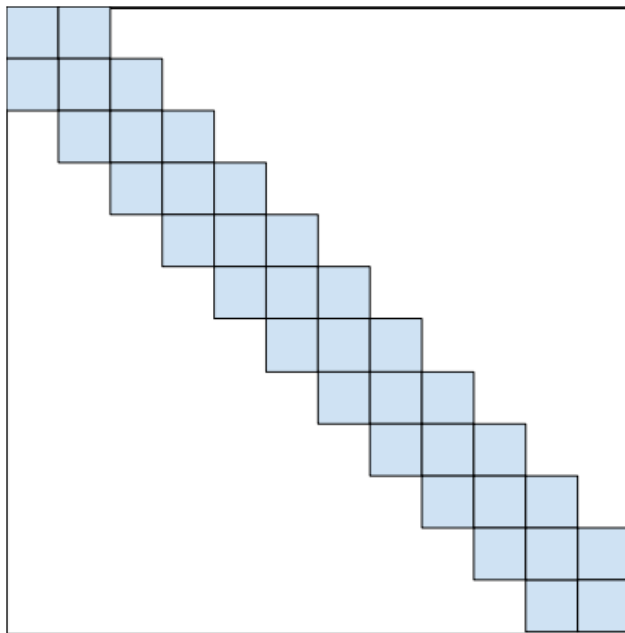


BigBird - Sparse Attention



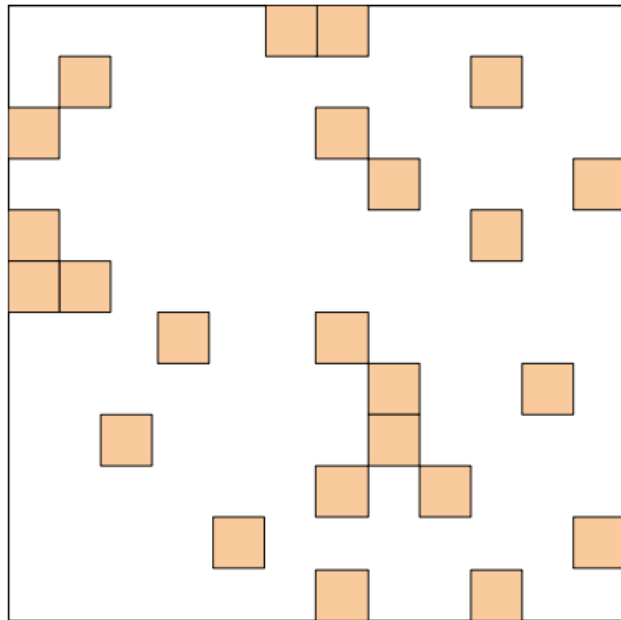
(c) Global Attention

BigBird - Sparse Attention



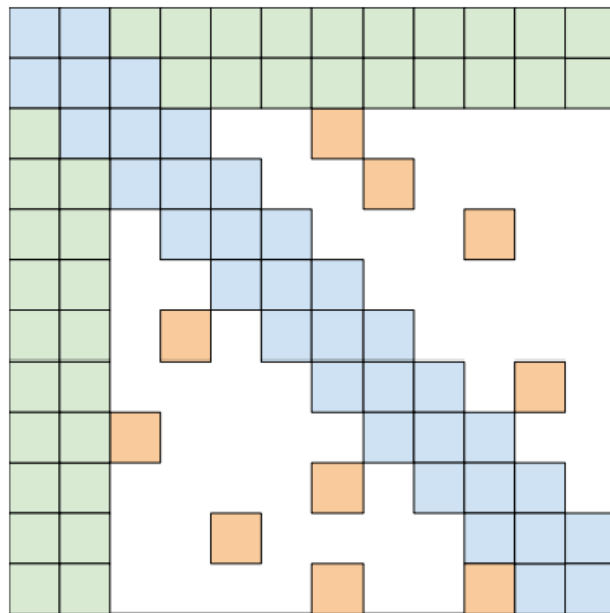
(b) Window attention

BigBird - Sparse Attention



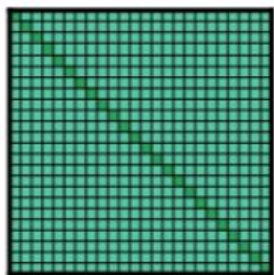
(a) Random attention

BigBird - Sparse Attention (8x faster)

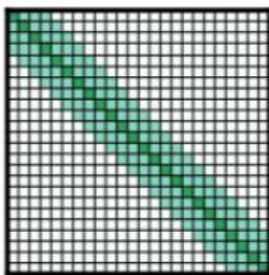


(d) BIGBIRD

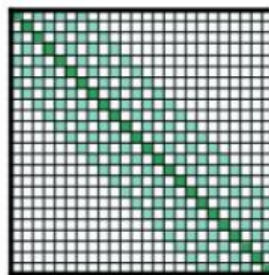
LongFormer



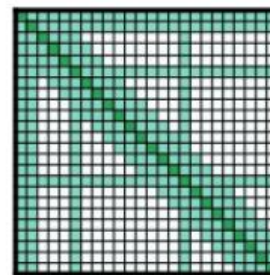
(a) Full n^2 attention



(b) Sliding window attention



(c) Dilated sliding window



(d) Global+sliding window

Figure 2: Comparing the full self-attention pattern and the configuration of attention patterns in our Longformer.