# BERTology: overview of modern NLP models

**Anastasia Ianina**

Harbour Space

# Outline

1. BERT (training and fine-tuning): recap
2. BPE & WordPiece tokenization
3. GPT
4. LM evaluation: popular benchmarks
5. Overview of other transformer-based models:
    - Transformer-XL
    - XLNet and Permutative Language Modeling (PLM)
    - MPNet: MLM + PLM
    - RoBERTa, ALBERT, DistilBERT
    - BART
    - BigBird and sparse attention
    - ELECTRA, DeBERTa

Based on:

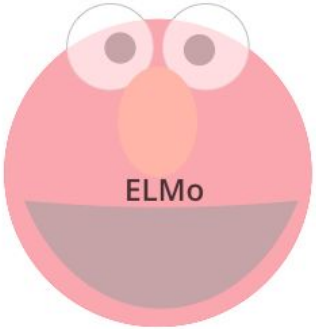
- http://web.stanford.edu/class/cs224n/
- https://jalammar.github.io/illustrated-transformer/
- http://jalammar.github.io/illustrated-bert/


THE
TRANSFORMER


ULM-FiT


BERT


OpenAI
Transformer


ELMo

# BERT: recap

Bidirectional Encoder Representations from Transformers

## 1 - Semi-supervised training on large amounts of text (books, wikipedia..etc).

The model is trained on a certain task that enables it to grasp patterns in language. By the end of the training process, BERT has language-processing abilities capable of empowering many models we later need to build and train in a supervised way.

### Semi-supervised Learning Step

**Model:**


**Dataset:**

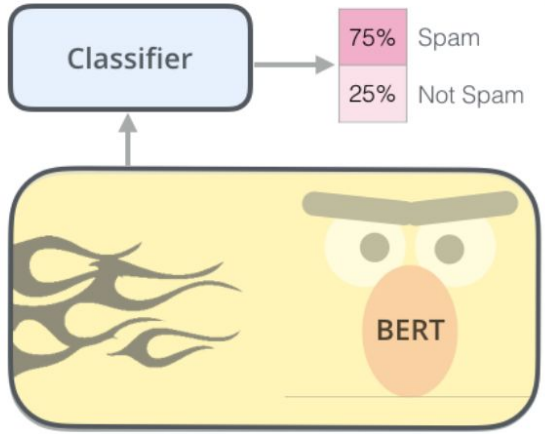

**Objective:**

Predict the masked word
(langauge modeling)

## 2 - Supervised training on a specific task with a labeled dataset.

### Supervised Learning Step

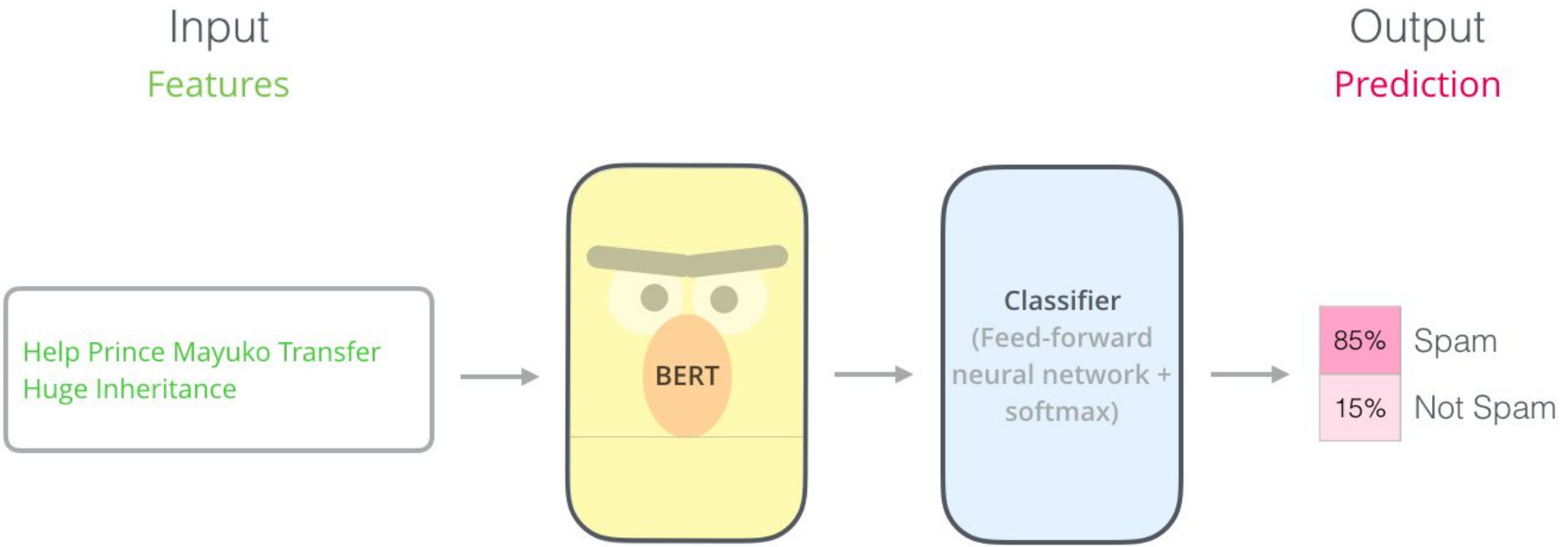
**Model:**
(pre-trained in step #1)



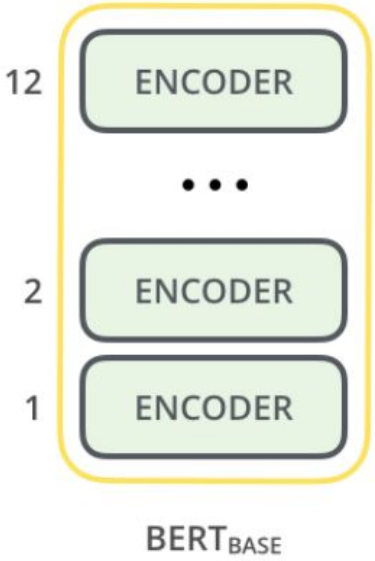

**Dataset:**

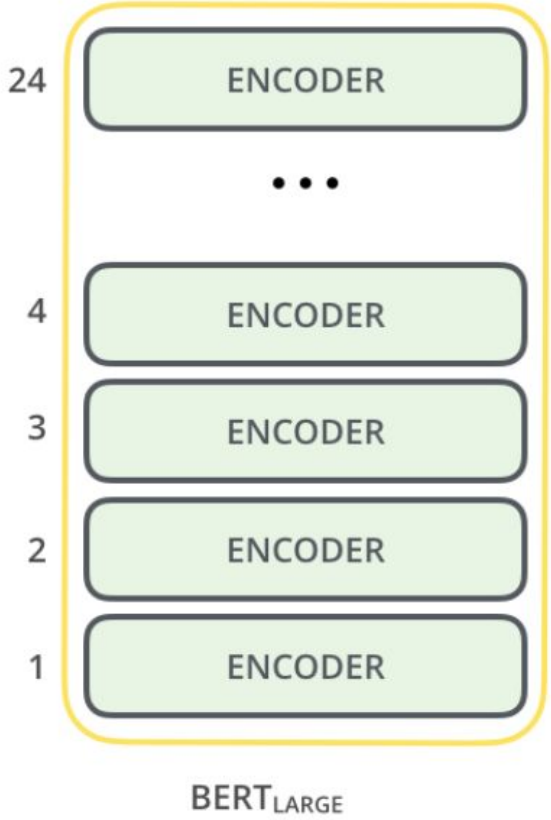
| Email message | Class |
|---|---|
| Buy these pills | Spam |
| Win cash prizes | Spam |
| Dear Mr. Atreides, please find attached… | Not Spam |

# BERT


Input
Features
Output
Prediction
Help Prince Mayuko Transfer
Huge Inheritance
BERT
Classifier
(Feed-forward
neural network +
softmax)
85%
Spam
15%
Not Spam

# BERT: base and large

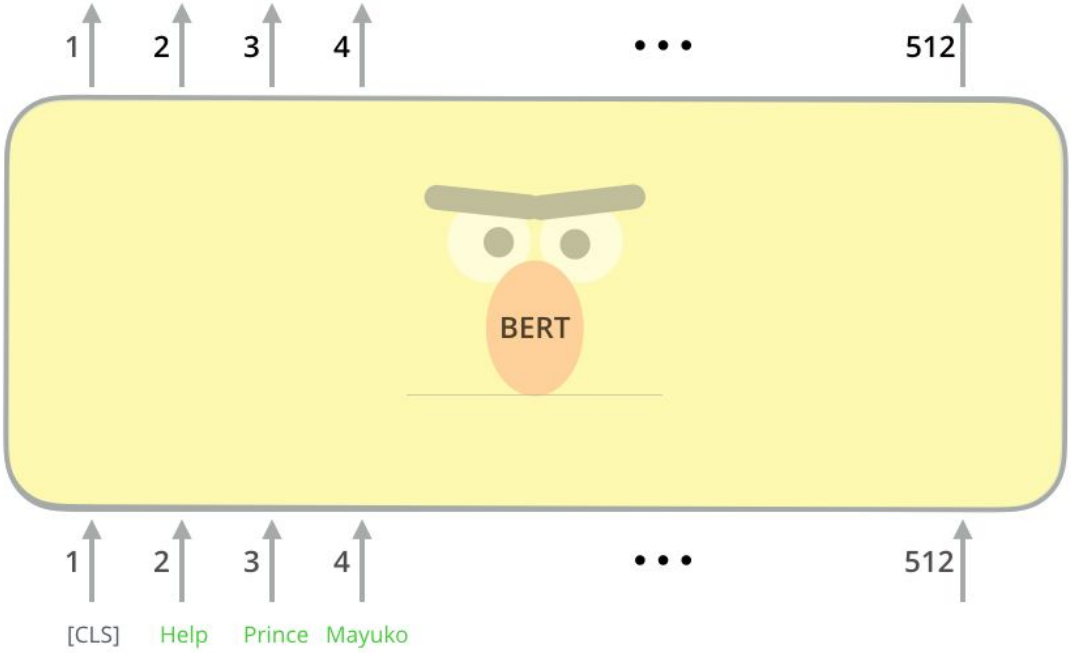


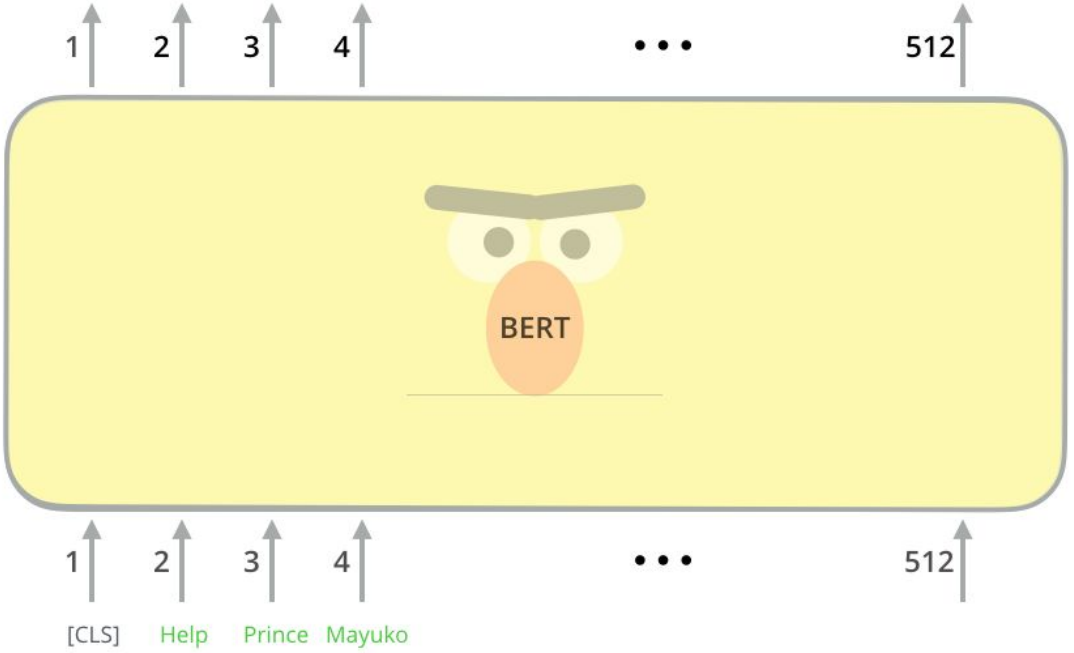
# BERT vs. Transformer

| |  |  | |
|---|---|---|---|
| | | **Base BERT** | **Large BERT** |
| Encoders | 6 | 12 | 24 |
| Units in FFN | 512 | 768 | 1024 |
| Attention Heads | 8 | 12 | 16 |

# Model inputs

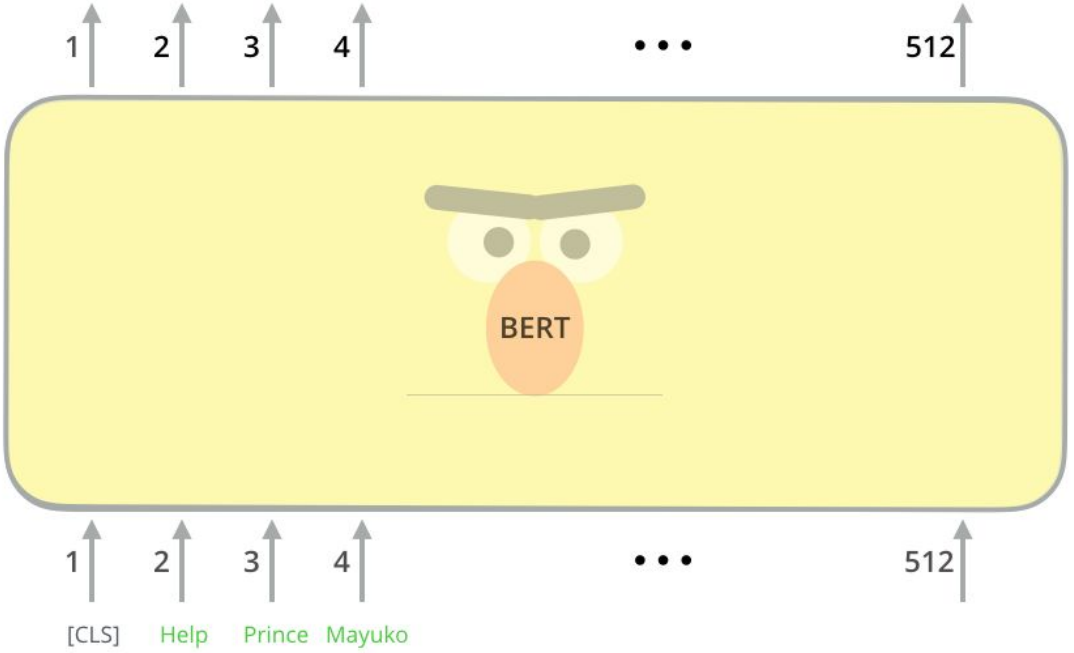


# Model inputs




Identical to the Transformer up until this point

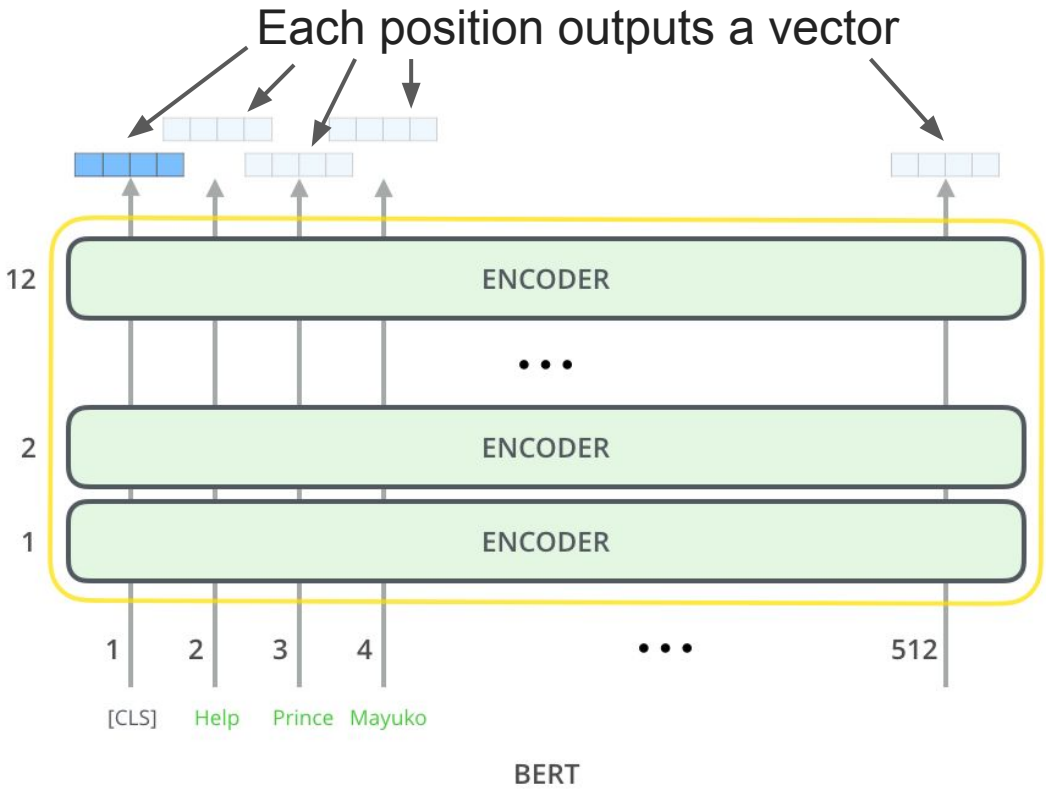
# Model inputs




Identical to the Transformer up until this point

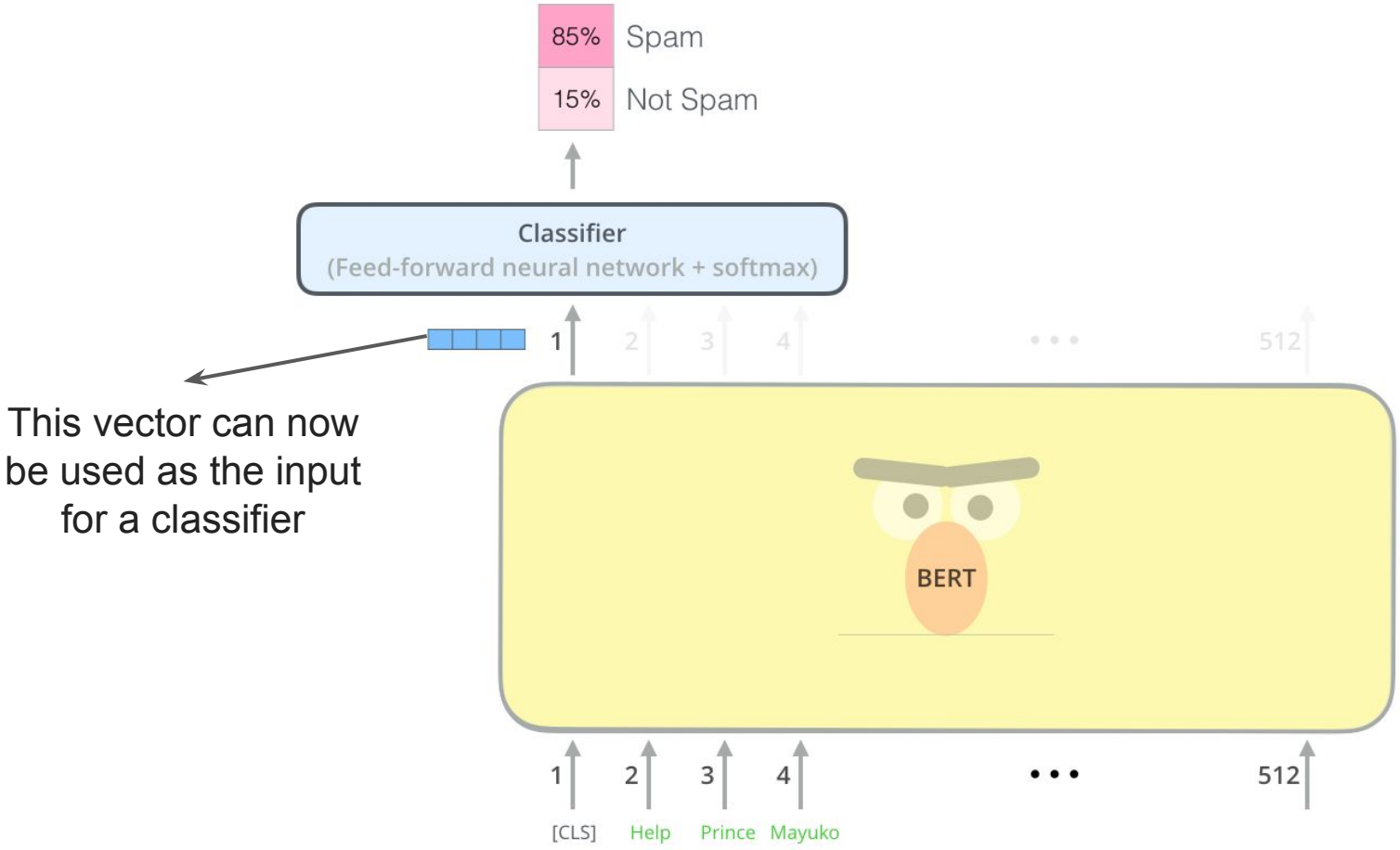
Why is BERT so special?

# Model outputs




For sentence classification we focus on the first position (that we passed [CLS] token to)

# Model inputs

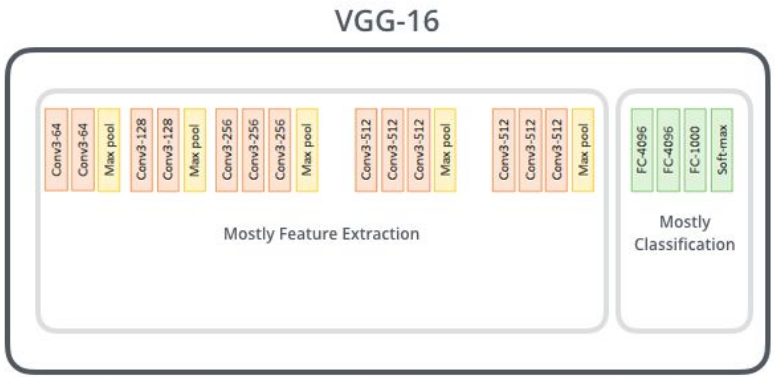



This vector can now be used as the input for a classifier

# Similar to CNN concept!

Input

Features

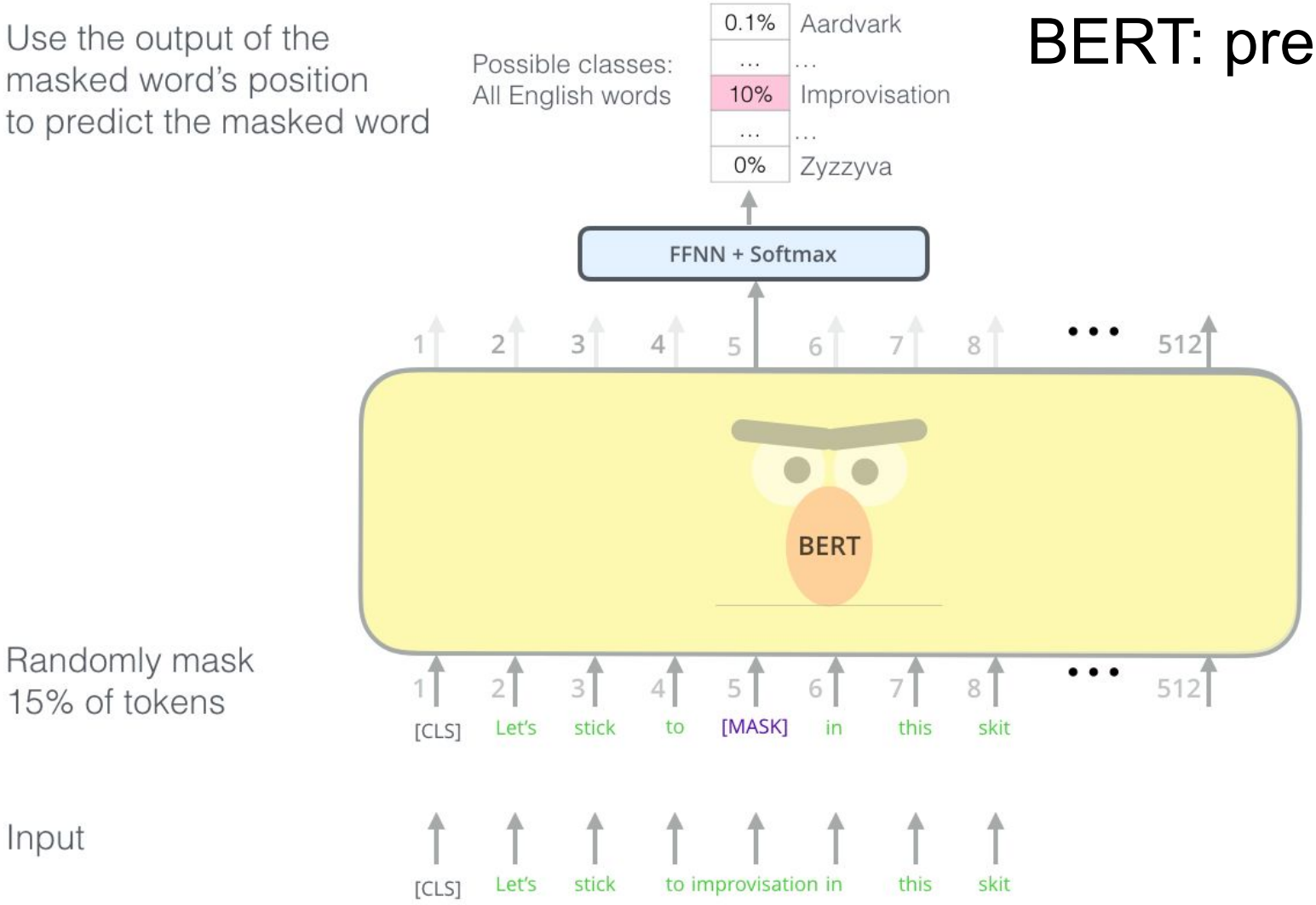




Output

Prediction

| | |
|---|---|
| 0.2% | Kit fox |
| 0.1% | English setter |
| 95% | Egyptian cat |
| 1% | Great Dane |
| | … |
| 0% | Hotdog |

# BERT: pre-training



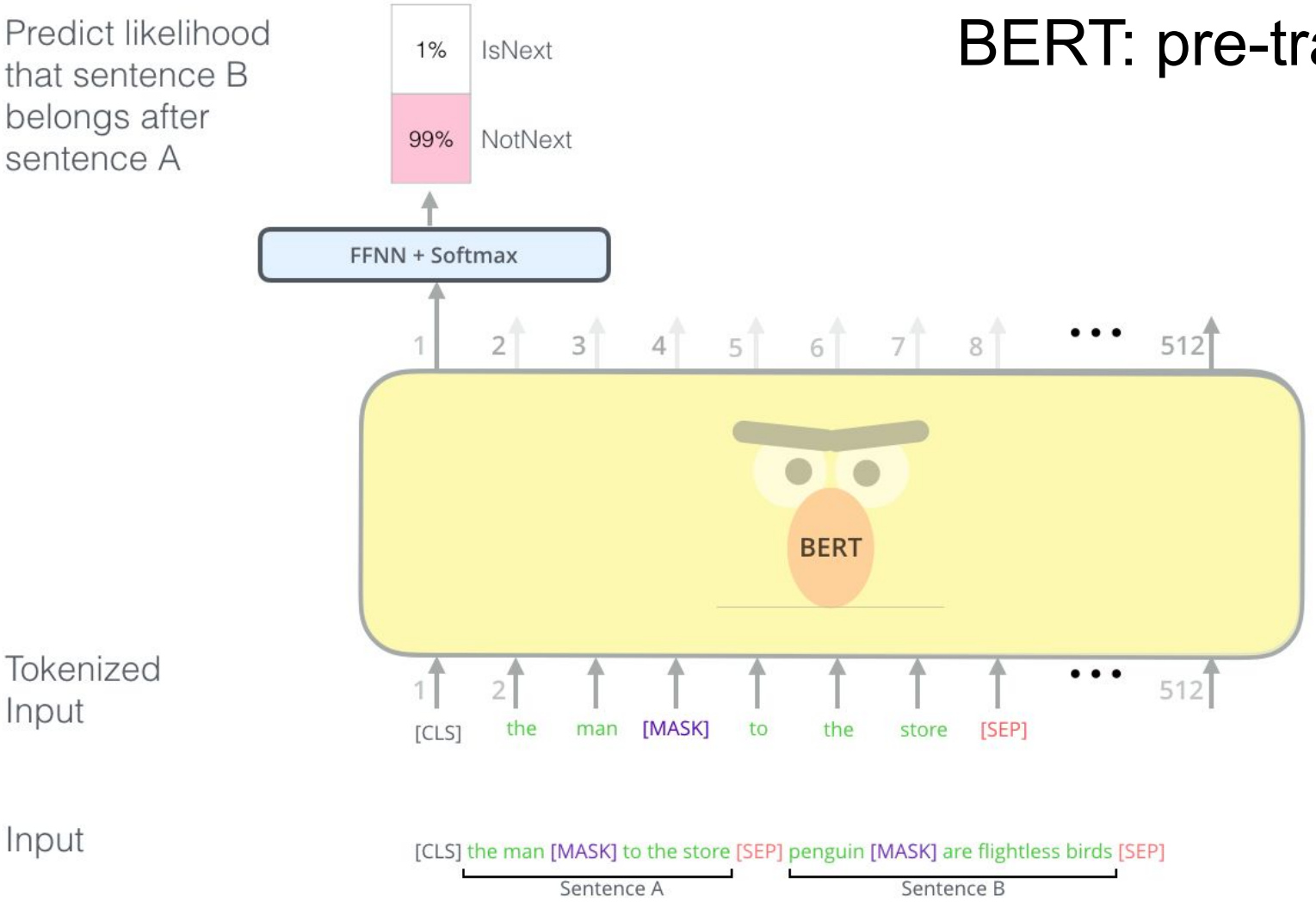
# BERT: pre-training

- “Masked Language Model” approach

- To make BERT better at handling relationships between multiple sentences, the pre-training process includes an additional task:

  *“Given two sentences (A and B),is B likely to be the sentence that follows A, or not?”*

# BERT: pre-training

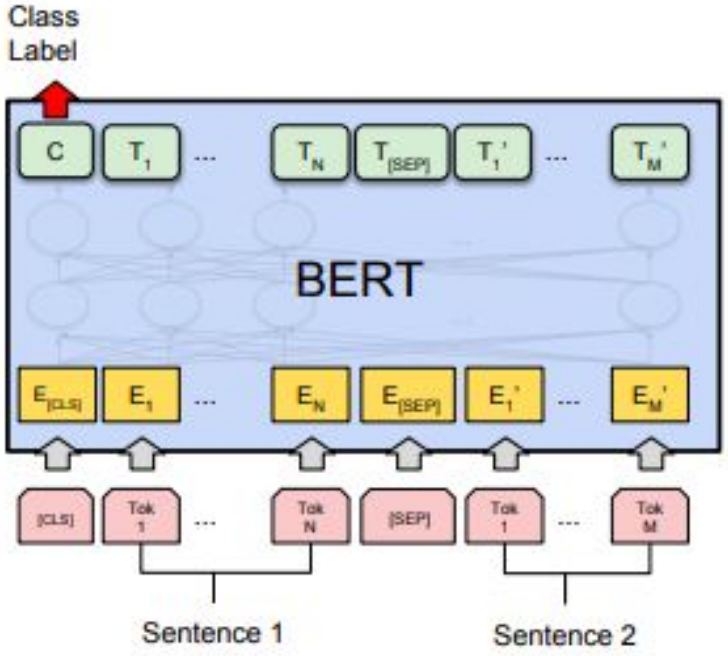


# BERT: input data format

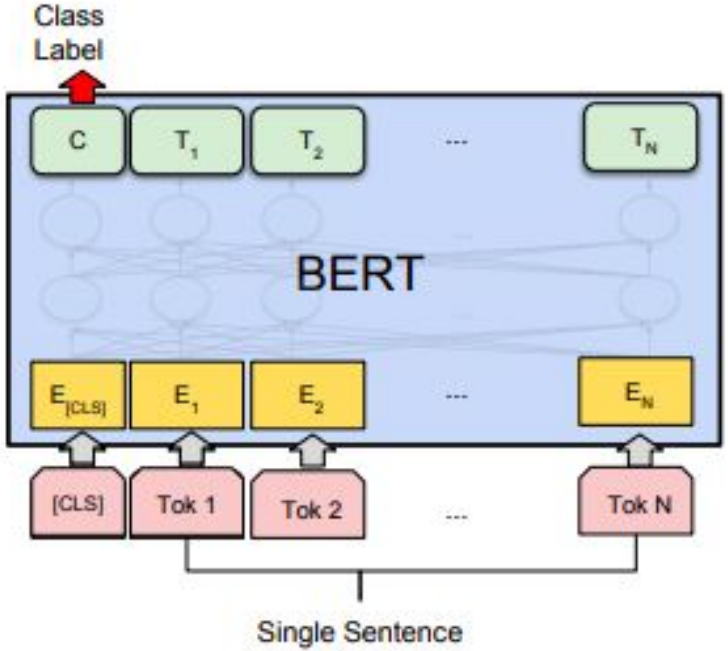
For each tokenized input sentence, we need to create:

- **input ids**: a sequence of integers identifying each input token to its index number in the BERT tokenizer vocabulary
- **segment mask**: a sequence of 1s and 0s used to identify whether the input is one sentence or two sentences long. For one sentence inputs, this is simply a sequence of 0s. For two sentence inputs, there is a 0 for each token of the first sentence, followed by a 1 for each token of the second sentence
- **attention mask**: a sequence of 1s and 0s, with 1s for all input tokens and 0s for all padding tokens

# BERT: fine-tuning for different tasks

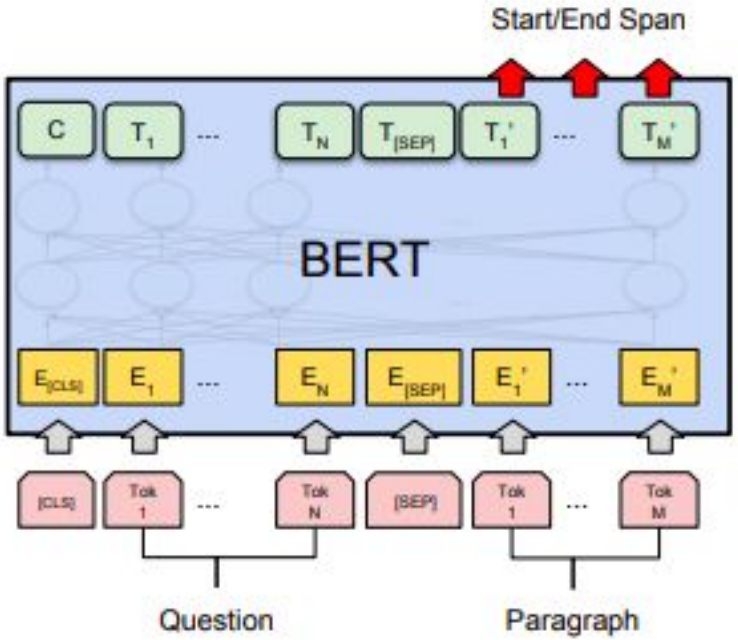



(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

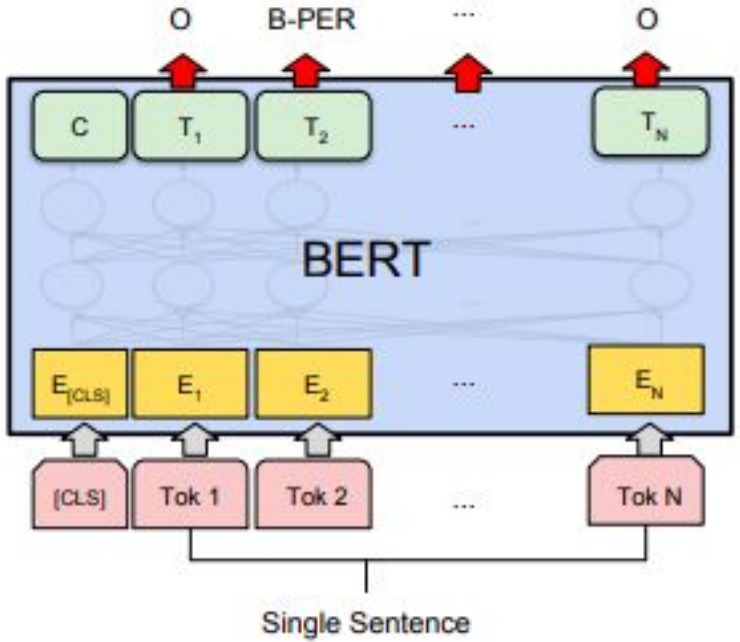



(b) Single Sentence Classification Tasks: SST-2, CoLA

# BERT: fine-tuning for different tasks

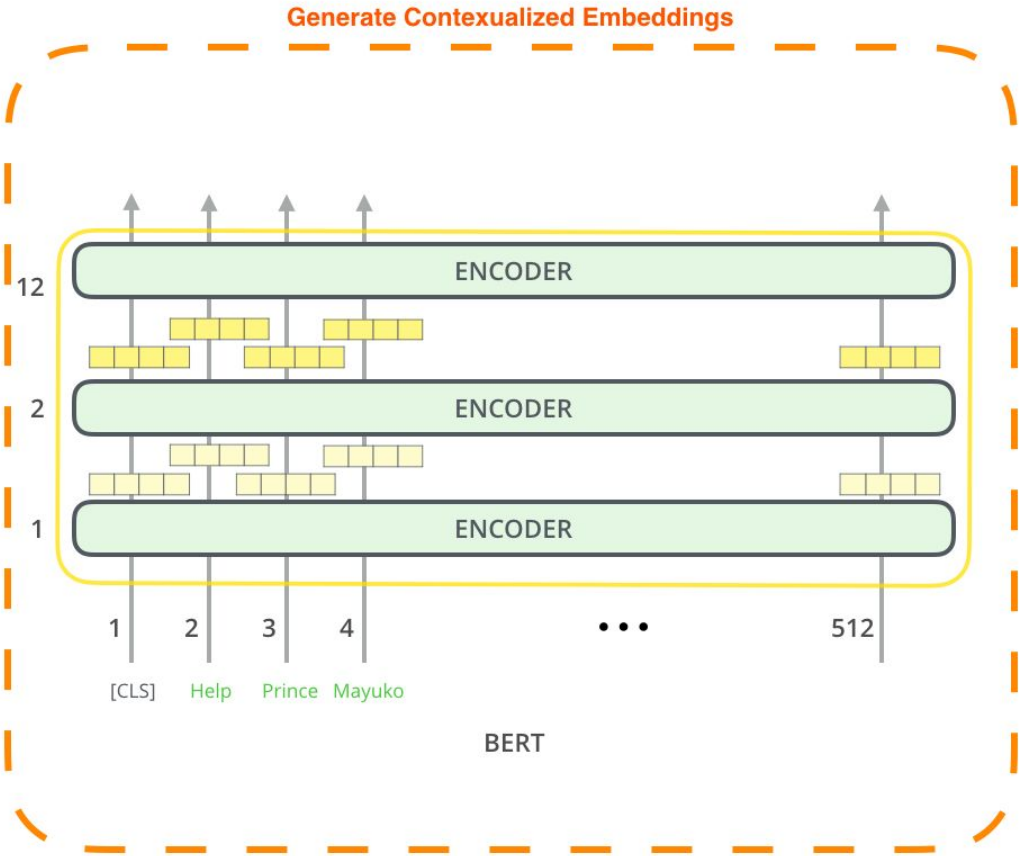



(c) Question Answering Tasks:
SQuAD v1.1

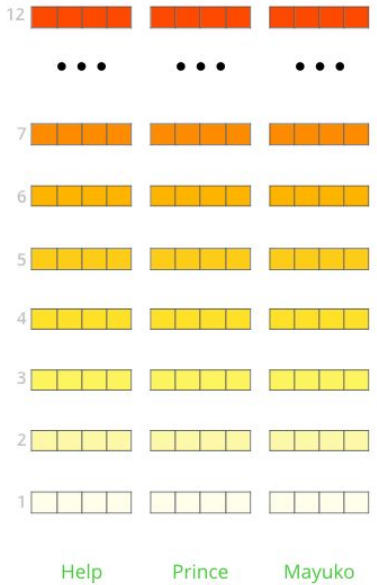



(d) Single Sentence Tagging Tasks:
CoNLL-2003 NER

# BERT for feature extraction



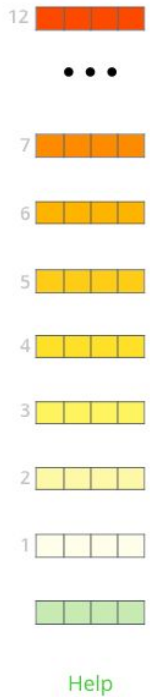

The output of each encoder layer along each token's path can be used as a feature representing that token.




**But which one should we use?**

# BERT for feature extraction

**What is the best contextualized embedding for "Help" in that context?**
For named-entity recognition task CoNLL-2003 NER




| | | Dev F1 Score |
|---|---|---|
| First Layer | Embedding | 91.0 |
| Last Hidden Layer | 12 | 94.9 |
| Sum All 12 Layers | 12 + ... + 2 + 1 = | 95.5 |
| Second-to-Last Hidden Layer | 11 | 95.6 |
| Sum Last Four Hidden | 12 + 11 + 10 + 9 = | 95.9 |
| Concat Last Four Hidden | 9 10 11 12 | 96.1 |

# WordPiece tokenization

## **Example:** Unaffable -> un, ##aff, ##able

The vocabulary is formed iteratively:

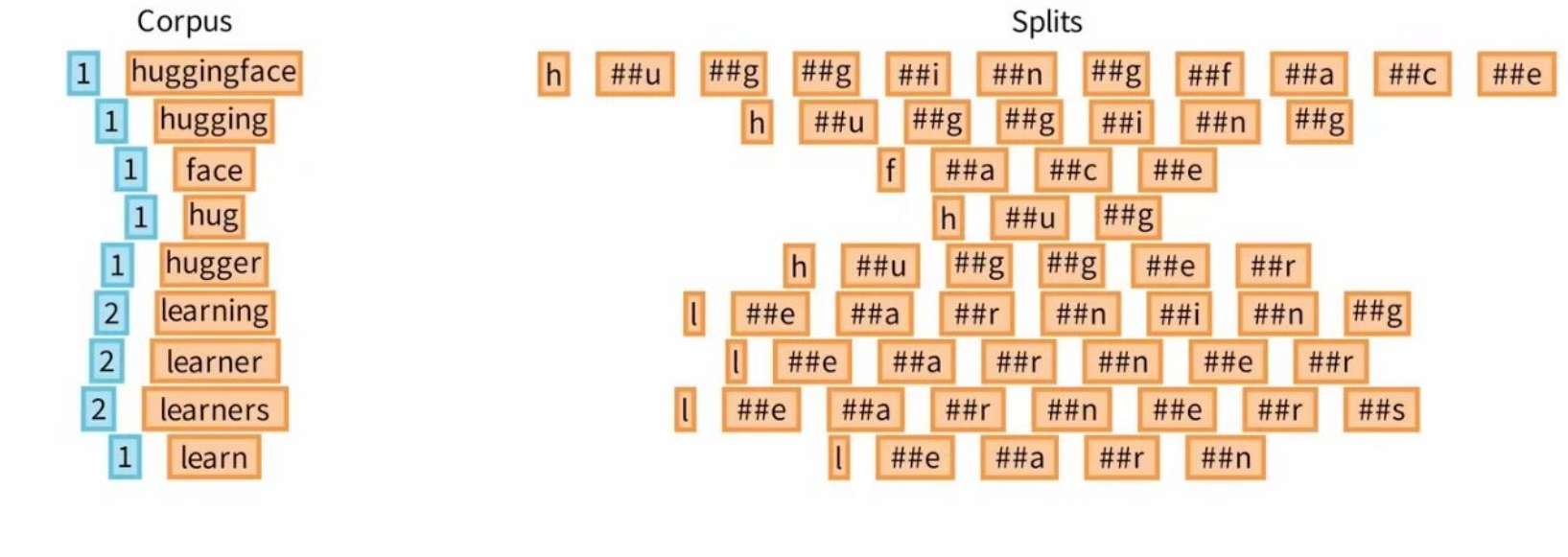
1. **Base vocabulary:** character unigrams (all the ASCII characters and most likely many Unicode characters)
2. **Merge rule:** tokens from the current vocabulary are merged in such a way that the likelihood of the training data is maximum
3. **Stopping criteria:** until vocabulary size is desired or desired number of merges is done (step 2 repeats itself until the desired vocabulary is formed

# WordPiece tokenization

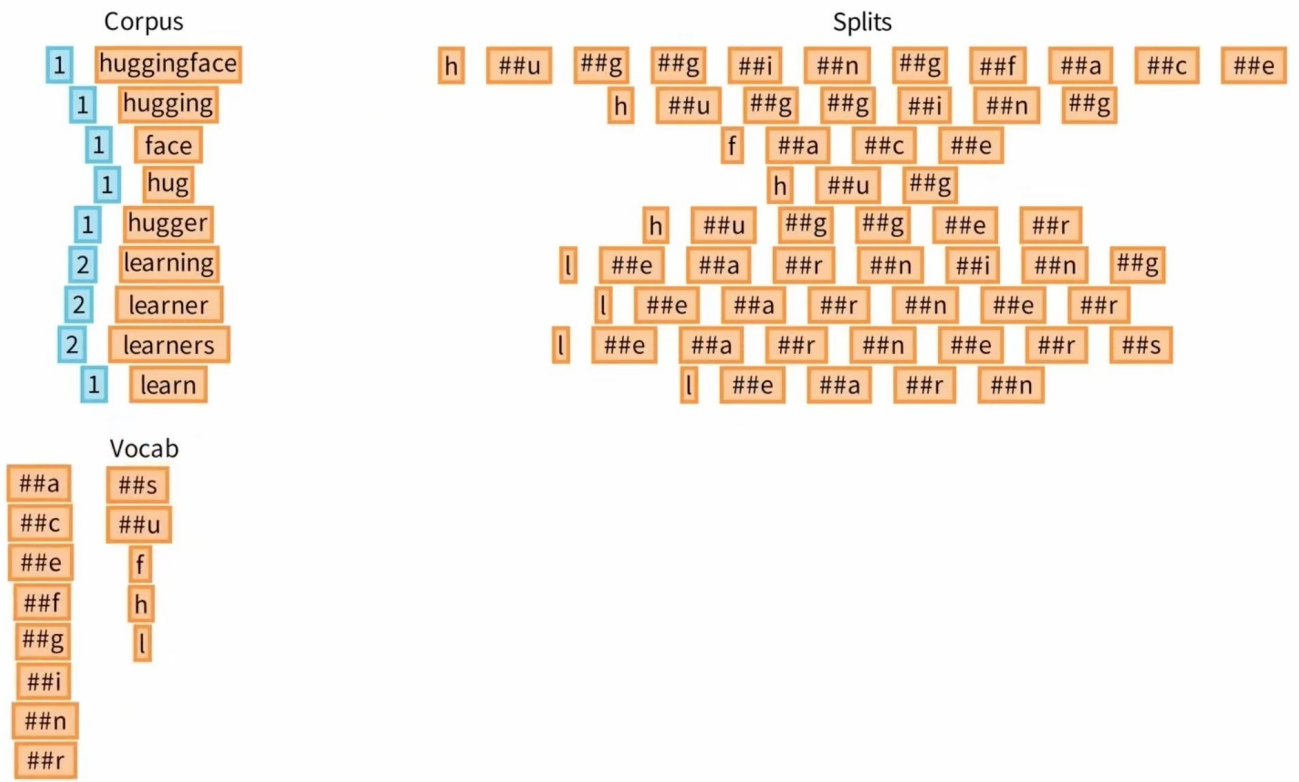
## **Example:** Unaffable -> un, ##aff, ##able

- Single model for 104 languages with a large shared vocabulary (119,547 WordPiece model)
- Non-word-initial units are prefixed with ##
- WordPiece vocabulary consists of:
  - The first 106 symbols: helper tokens, like PAD and UNK
  - 36.5% of the vocabulary are non-initial word pieces
  - The alphabet consists of 9,997 unique characters that are defined as word-initial (C) and continuation symbols (##C), which together make up 19,994 word pieces
  - The rest are multi-character word pieces of various length.
- Used for tokenization of **BERT, DistilBERT, MobileBERT, and MPNet**
- Google never open-sourced its implementation

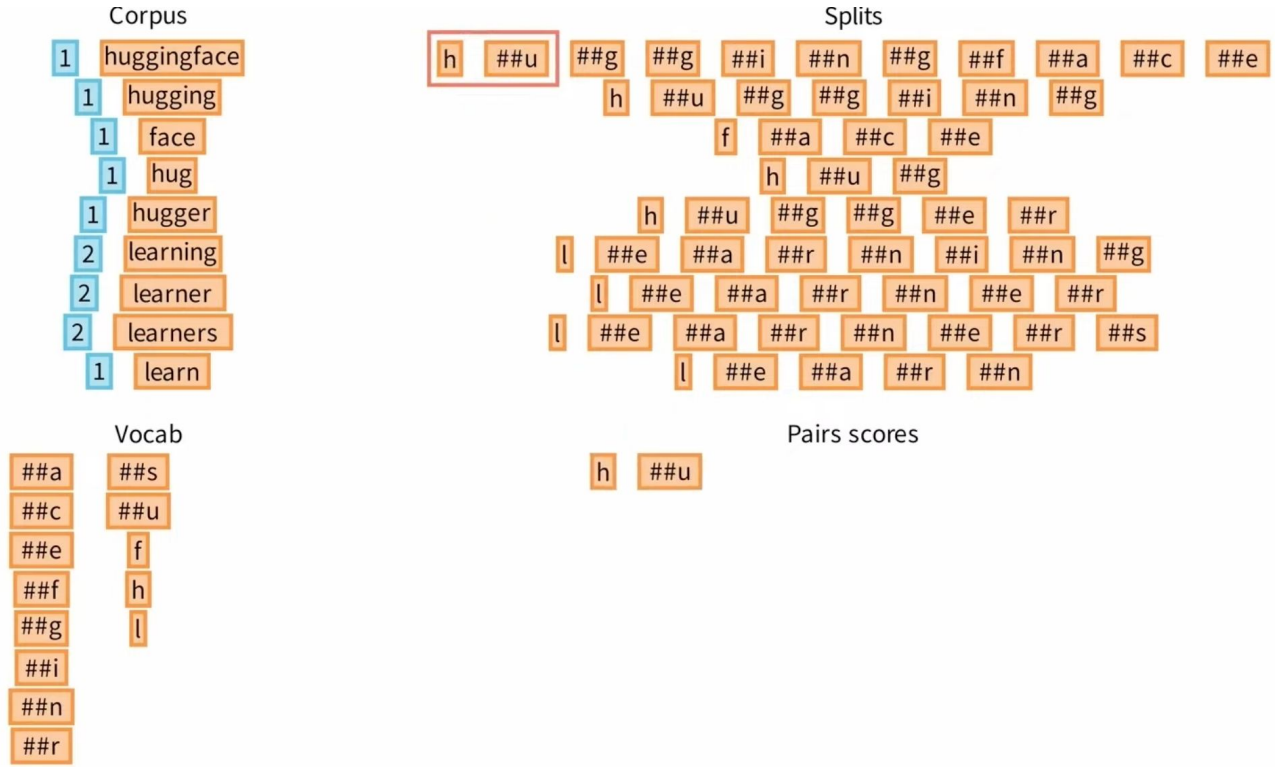
# WordPiece tokenization: example



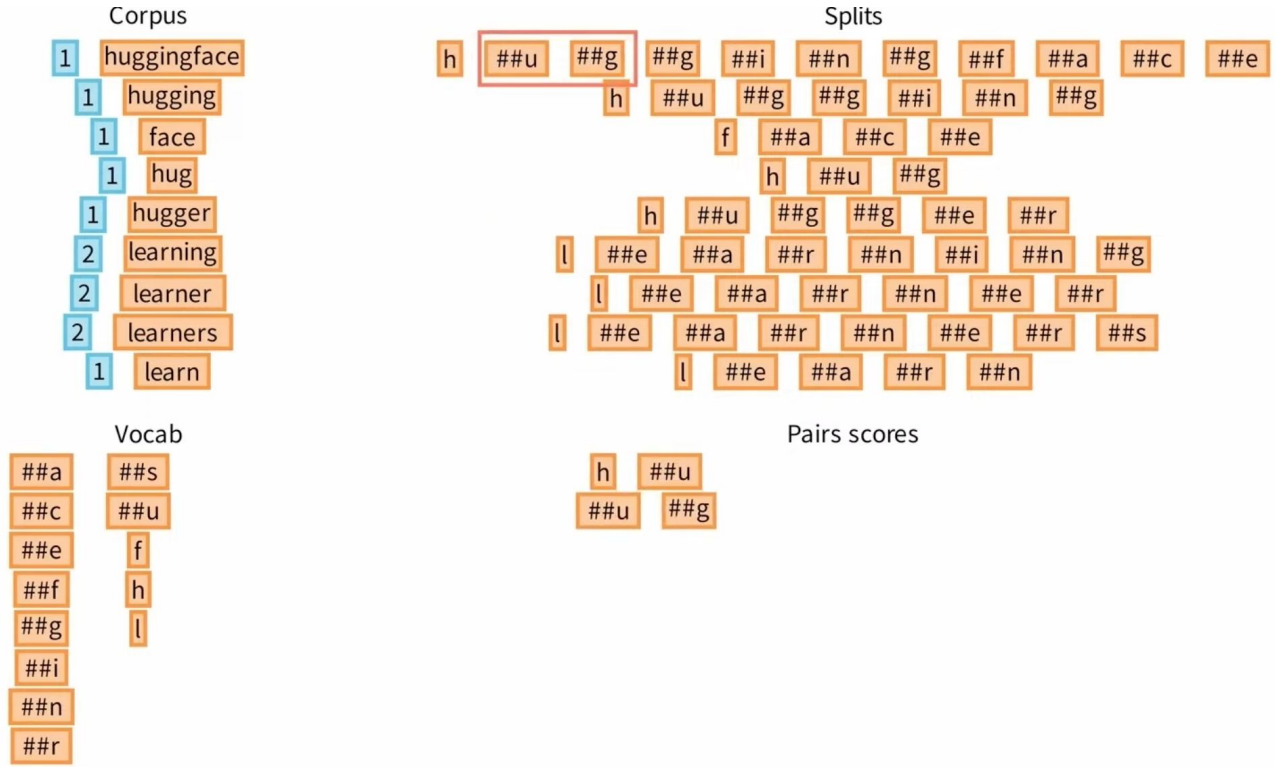
# WordPiece tokenization: example



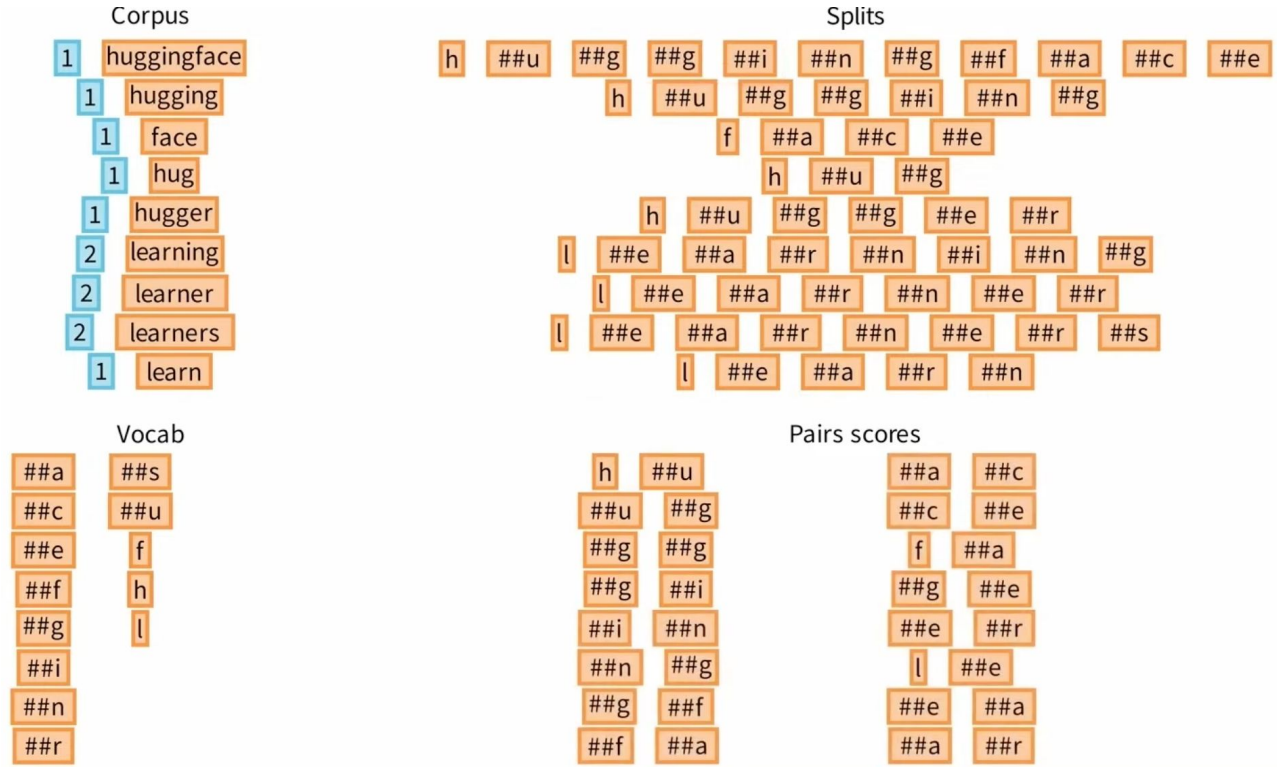
# WordPiece tokenization: example



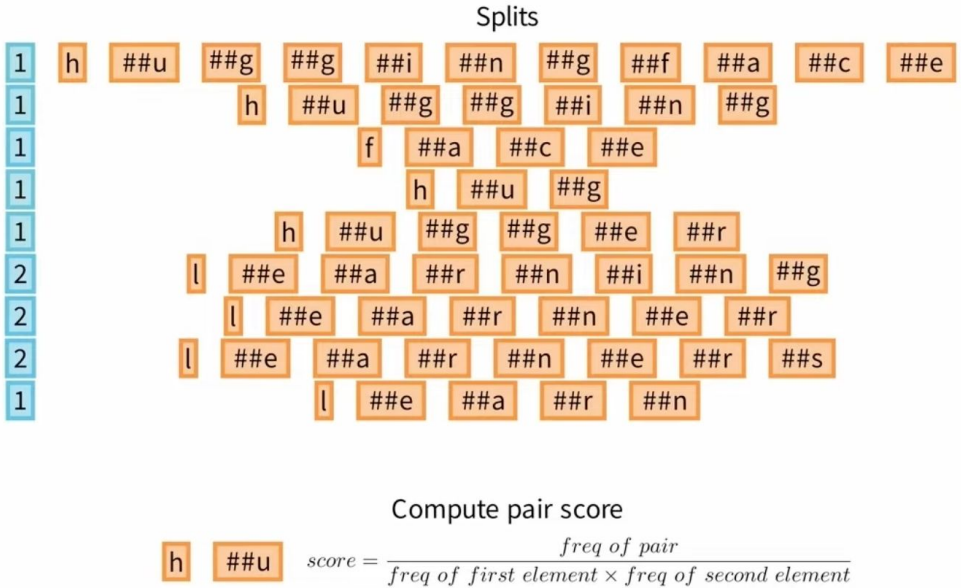
# WordPiece tokenization: example



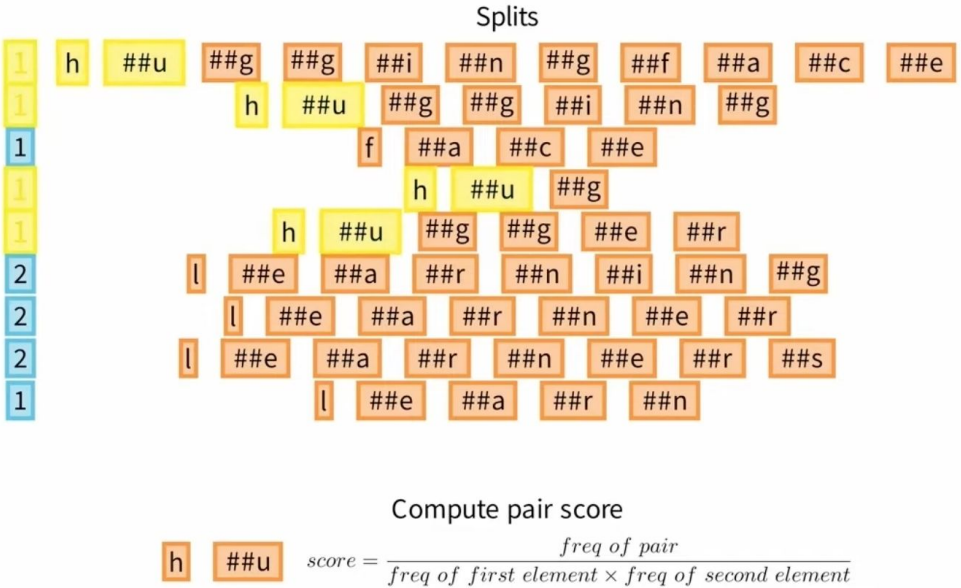
# WordPiece tokenization: example



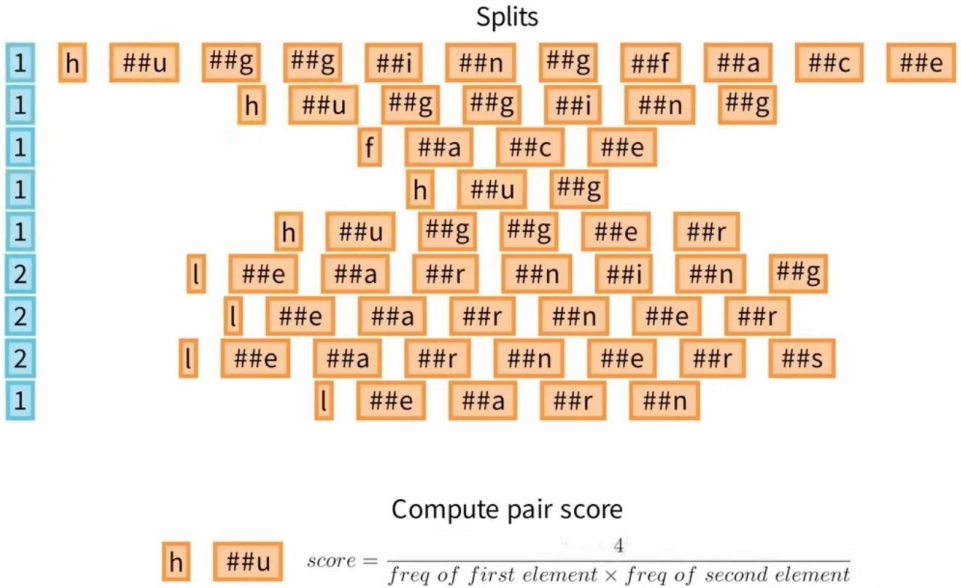
# WordPiece tokenization: example



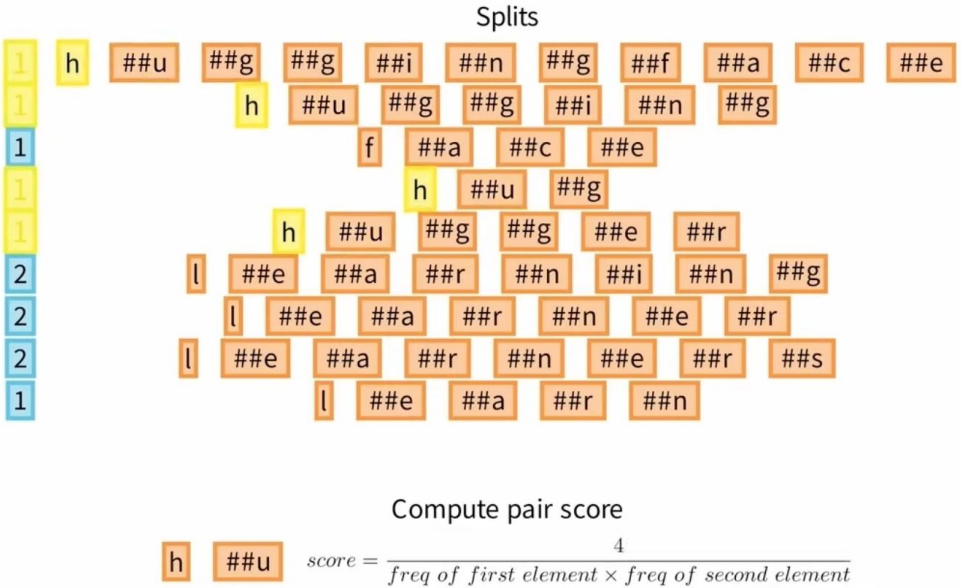
# WordPiece tokenization: example



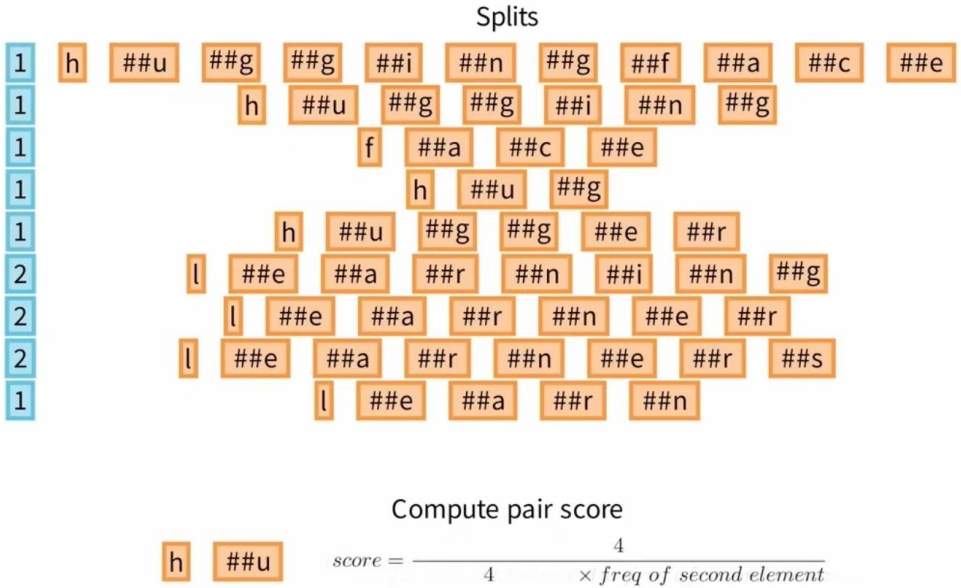
# WordPiece tokenization: example



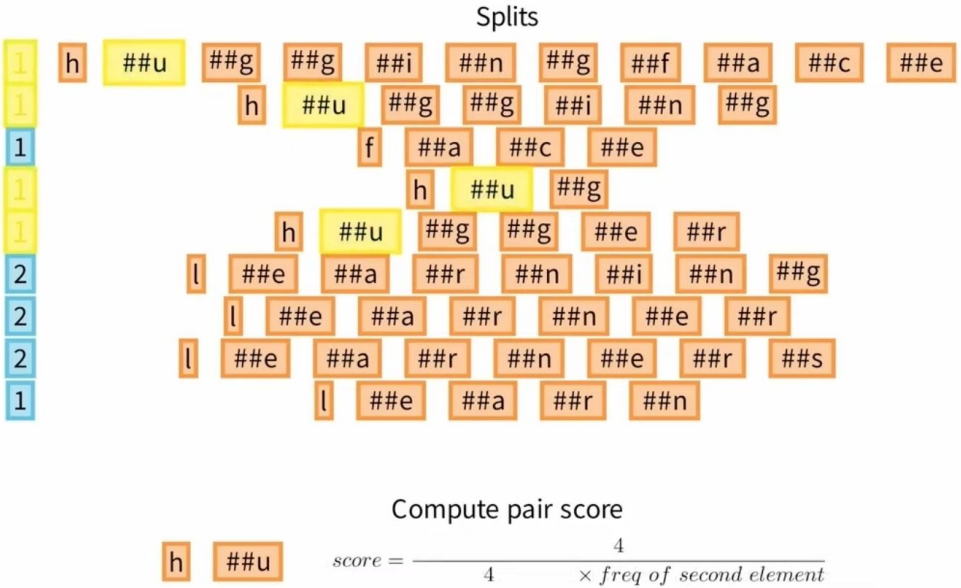
# WordPiece tokenization: example



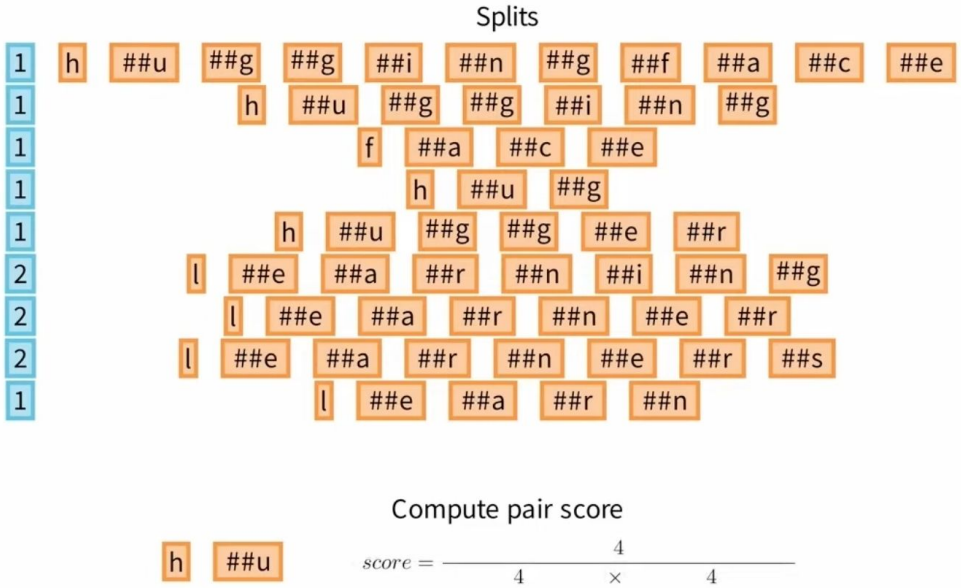
# WordPiece tokenization: example



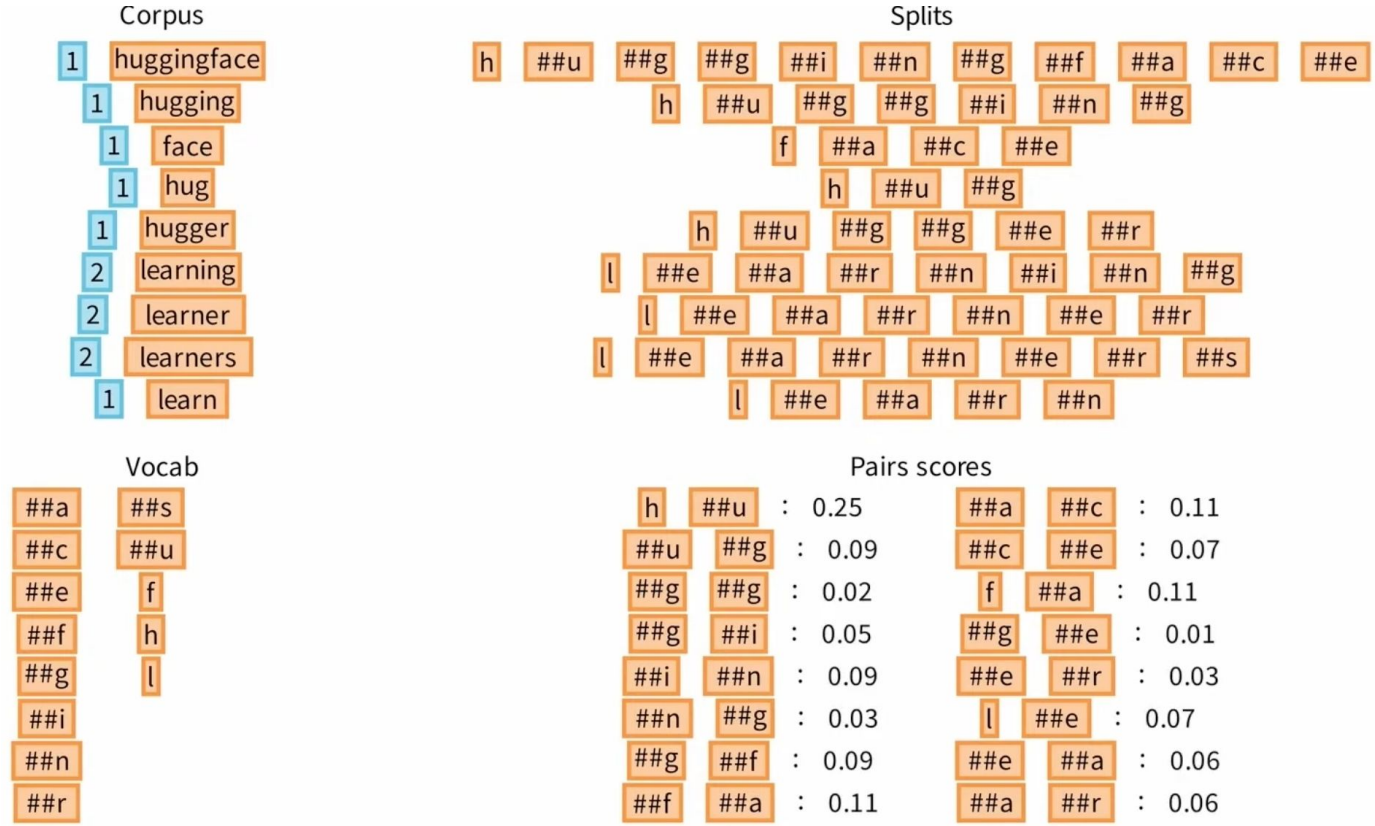
# WordPiece tokenization: example



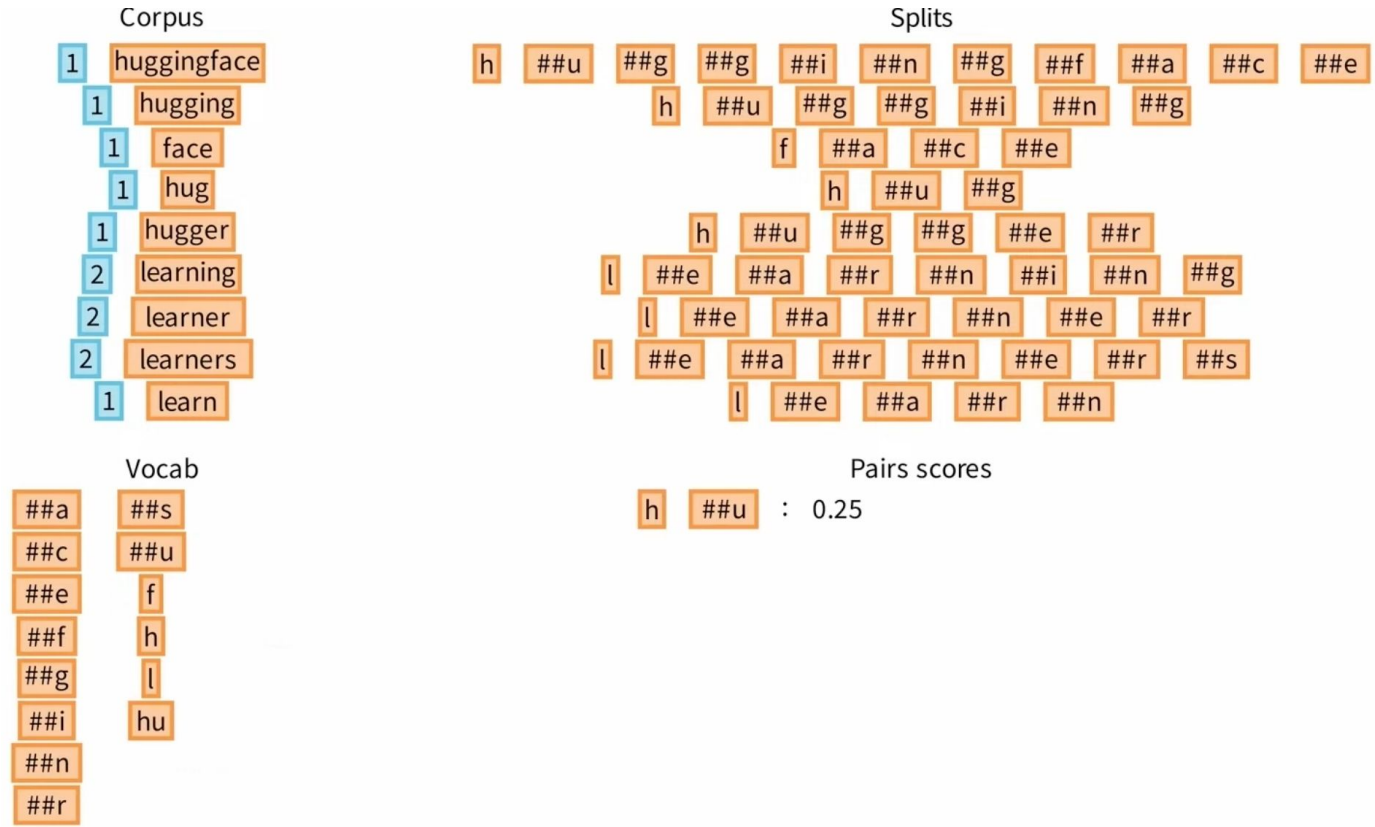
# WordPiece tokenization: example



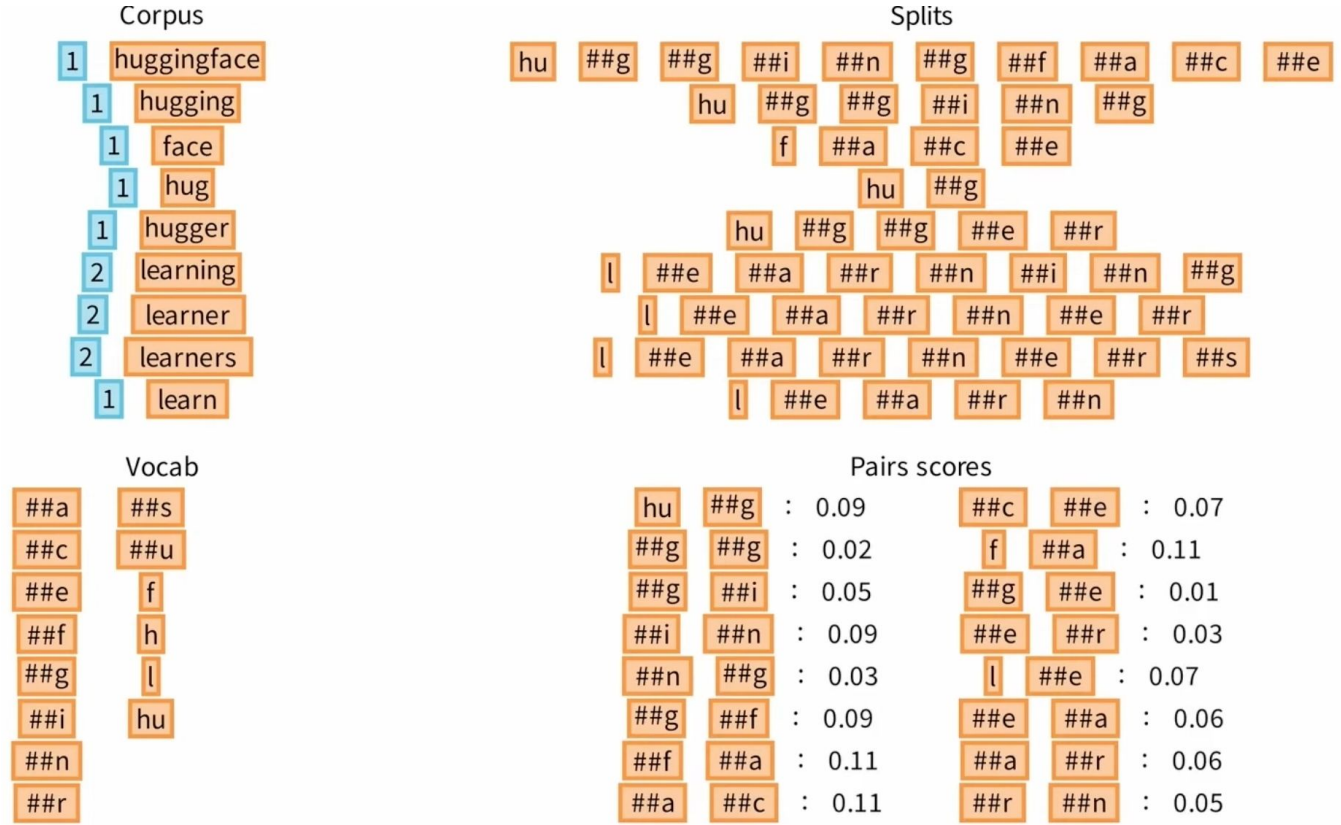
# WordPiece tokenization: example



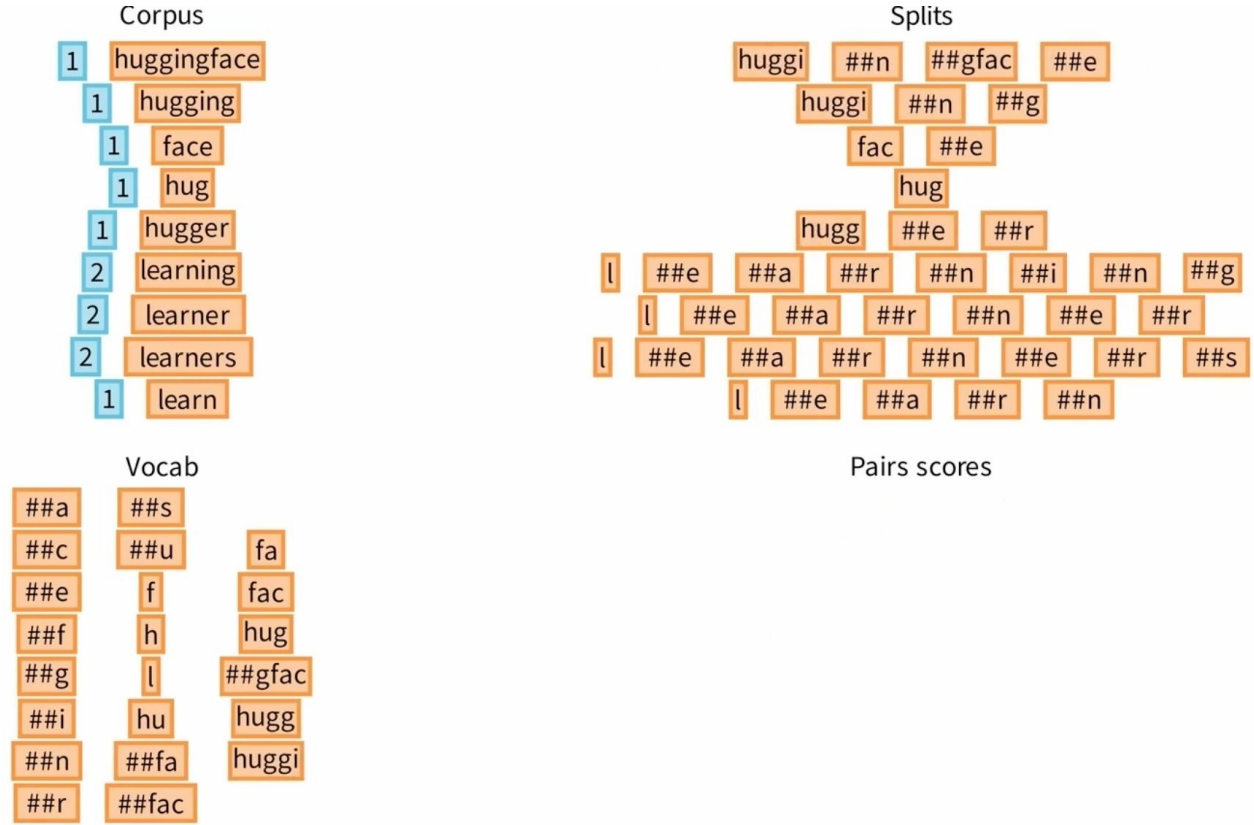
# WordPiece tokenization: example



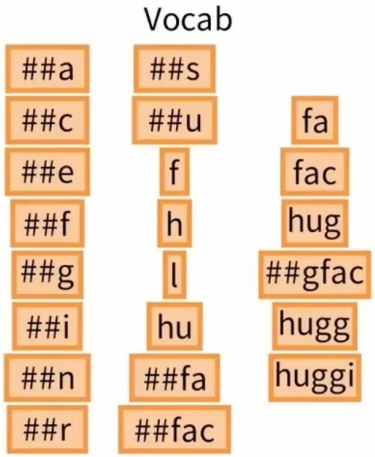
# WordPiece tokenization: example

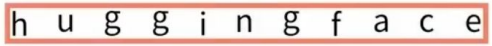


# WordPiece tokenization: example



# WordPiece tokenization: example

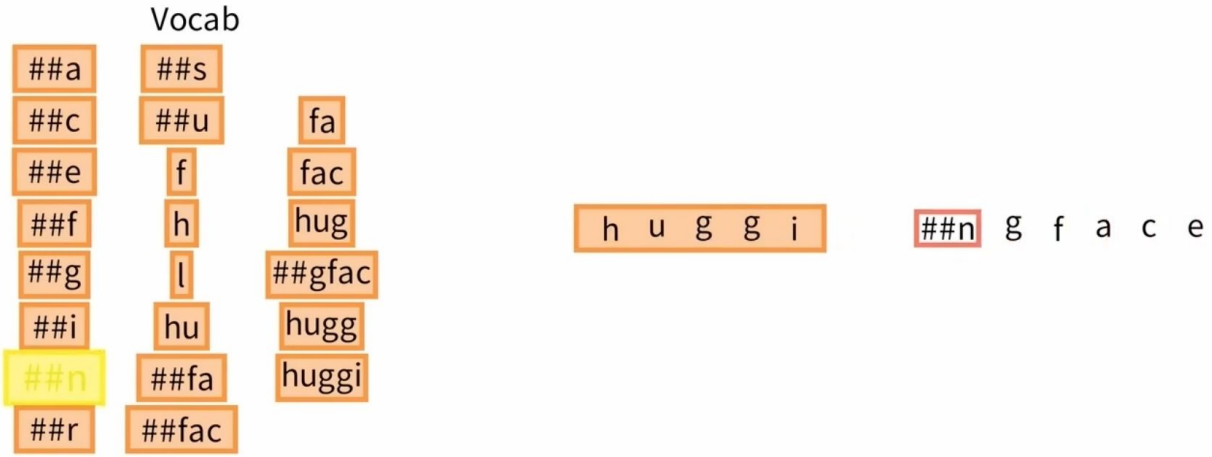


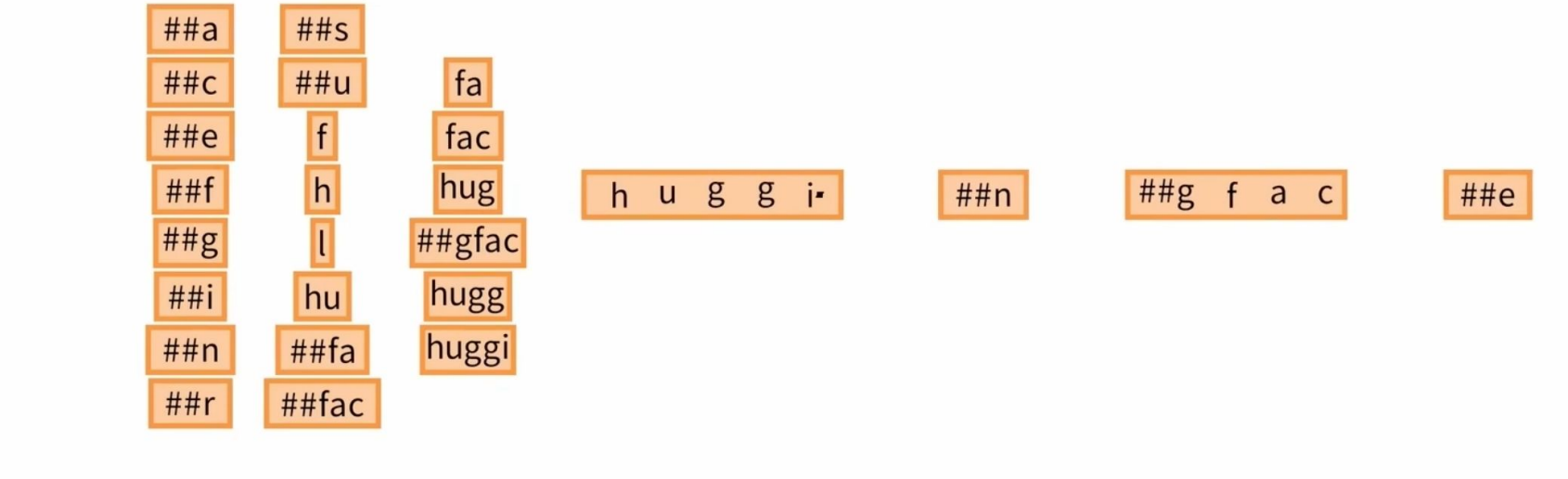
# WordPiece tokenization: example


h u g g i n g f a c e

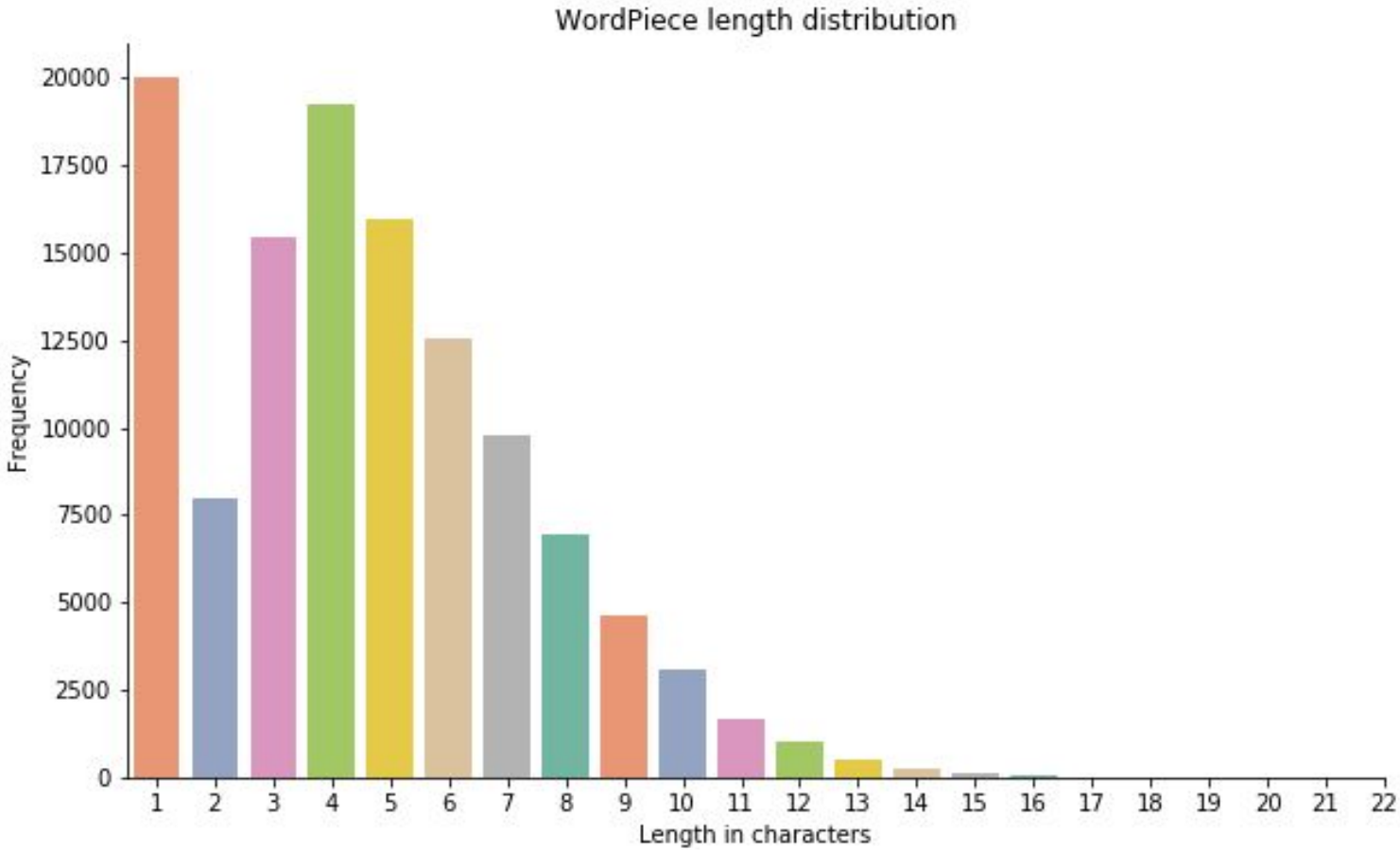
# WordPiece tokenization: example



# WordPiece tokenization: example



# WordPiece tokenization



# Byte Pair Encoding (BPE)

## **Example:** Unrelatedness -> un, related, ness@@

The vocabulary is formed iteratively:

1. **Base vocabulary:** character unigrams (like in WordPiece). However, BPE looks at words as being written not with Unicode characters, but with bytes.
   a. base vocabulary has a small size (256)
   b. no unknown tokens
2. **Merge rule:** tokens from the current vocabulary are merged according to frequencies in the training data (everytime we search for two most frequent consecutive tokens and merge them)
3. **Stopping criteria:** until vocabulary size is desired or desired number of merges is done

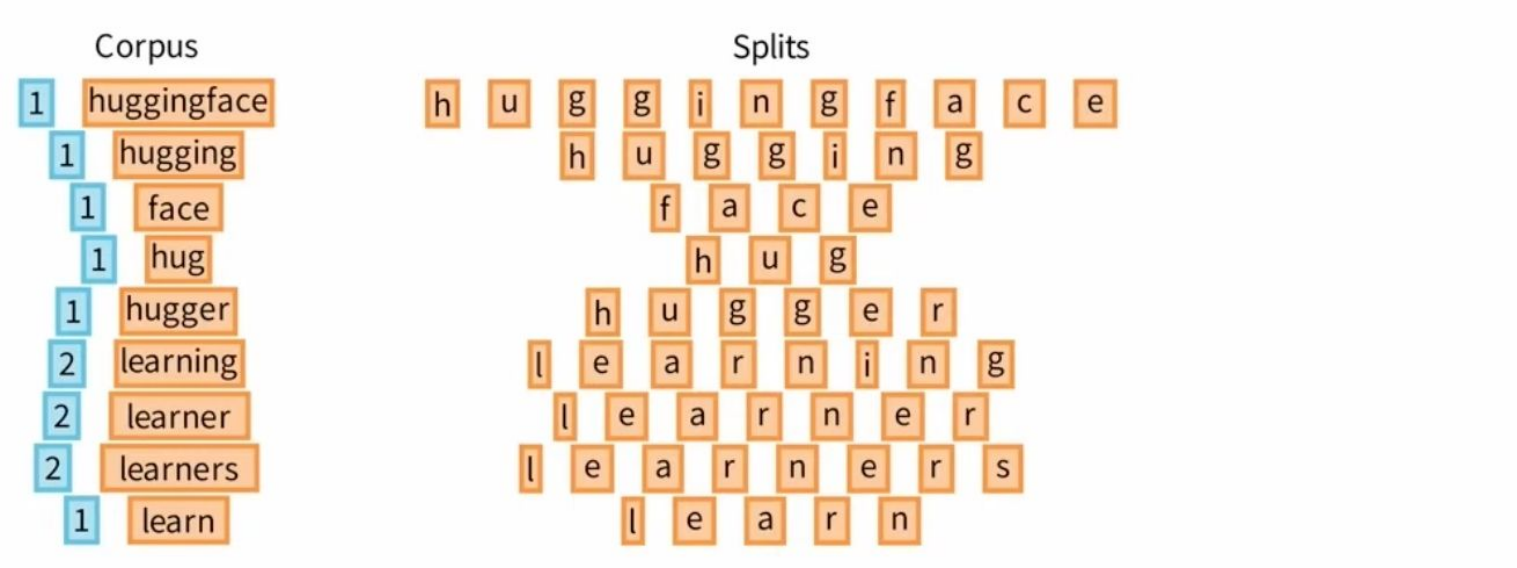
# Byte Pair Encoding (BPE)

## **Example:** Unrelatedness -> un, related, ness@@

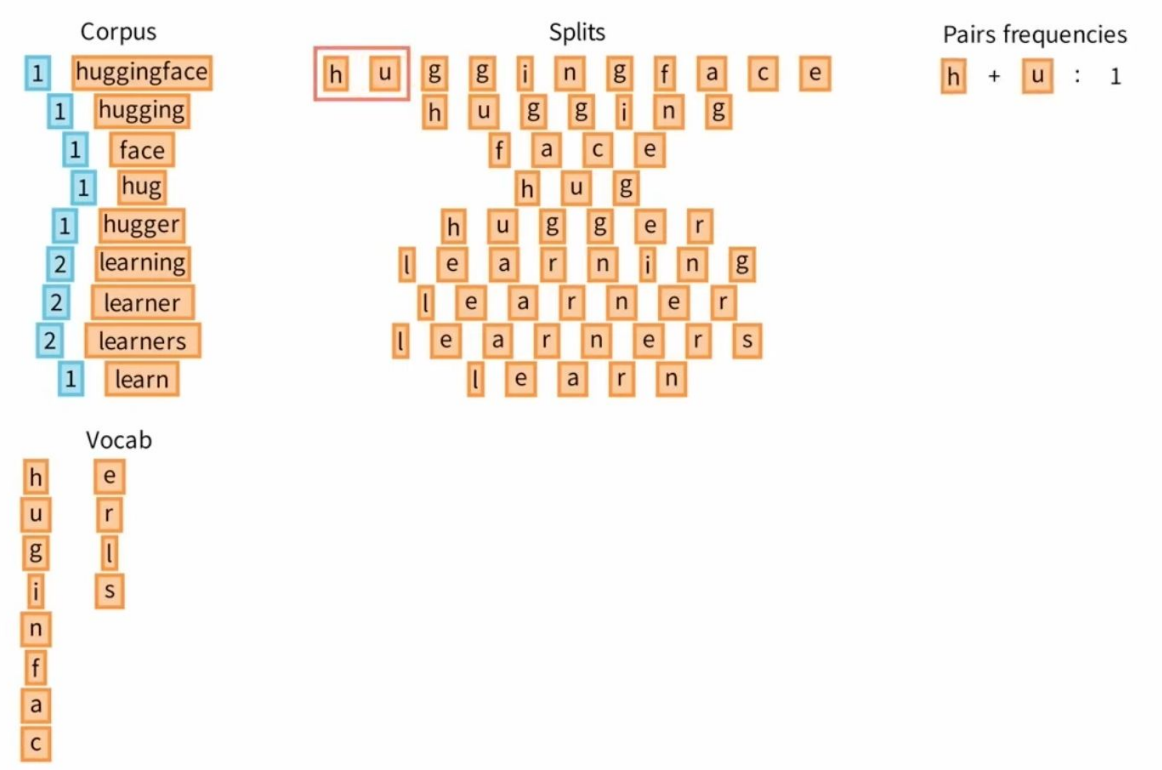
- Was initially developed as an algorithm to compress texts
- Non-word-initial units end with @@
- Uses **byte-level BPE** (treats words as being written with bytes) -> small base vocabulary and no unknown tokens
- Used for tokenization when pretraining the **GPT, RoBERTa, BART, and DeBERTa**

```
u-n-r-e-l-a-t-e-d
u-n re-l-a-t-e-d
u-n re-l-at-e-d
u-n re-l-at-ed
un re-l-at-ed
un re-l-ated
un rel-ated
un-related
```

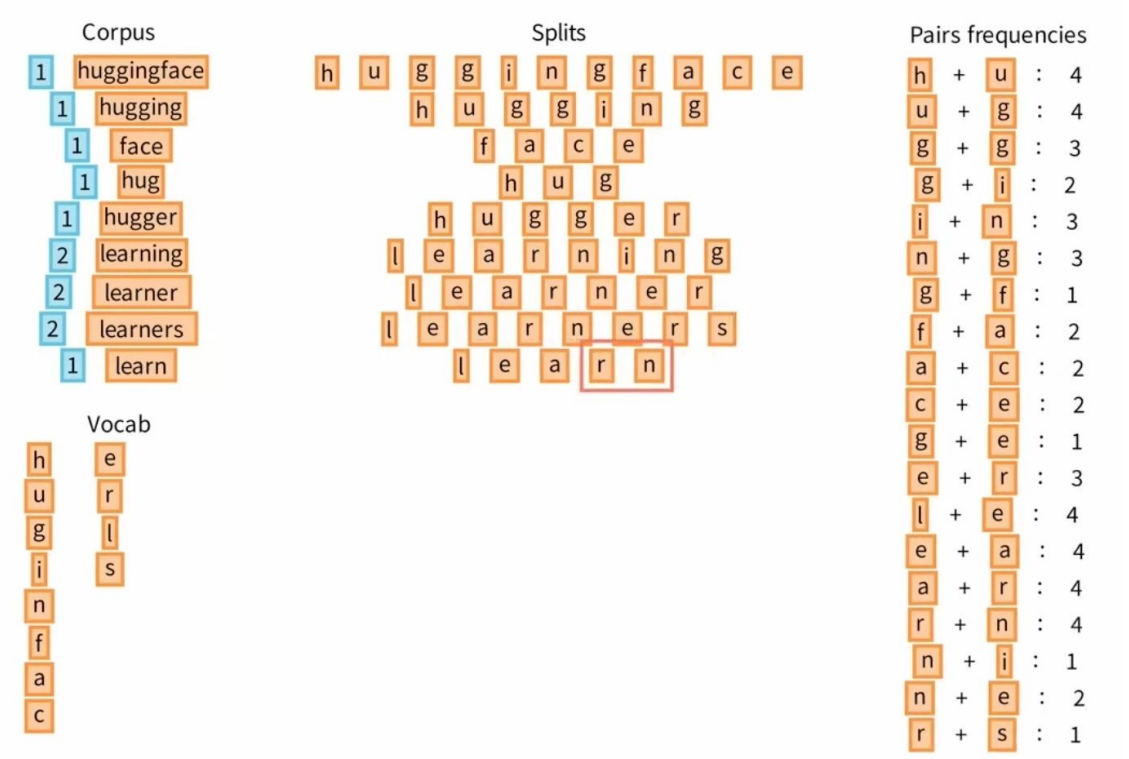
# BPE: example


Corpus
1 huggingface
1 hugging
1 face
1 hug
1 hugger
2 learning
2 learner
2 learners
1 learn
Splits
h u g g i n g f a c e
h u g g i n g
f a c e
h u g
h u g g e r
l e a r n i n g
l e a r n e r
l e a r n e r s
l e a r n

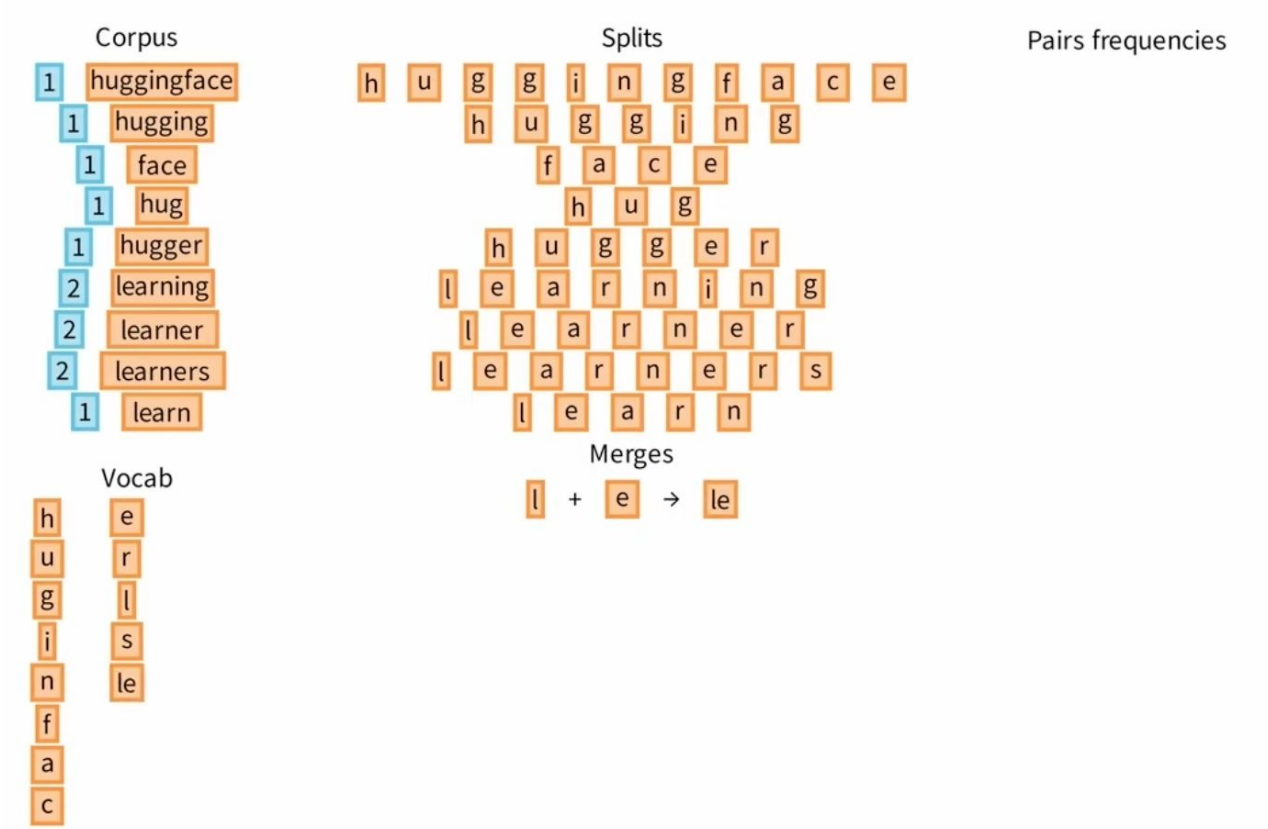
# BPE: example


Corpus
1 huggingface
1 hugging
1 face
1 hug
1 hugger
2 learning
2 learner
2 learners
1 learn
Splits
h u g g i n g f a c e
h u g g i n g
f a c e
h u g
h u g g e r
l e a r n i n g
l e a r n e r
l e a r n e r s
l e a r n
Pairs frequencies
h + u : 1
Vocab
h u g i n f a c e r l s

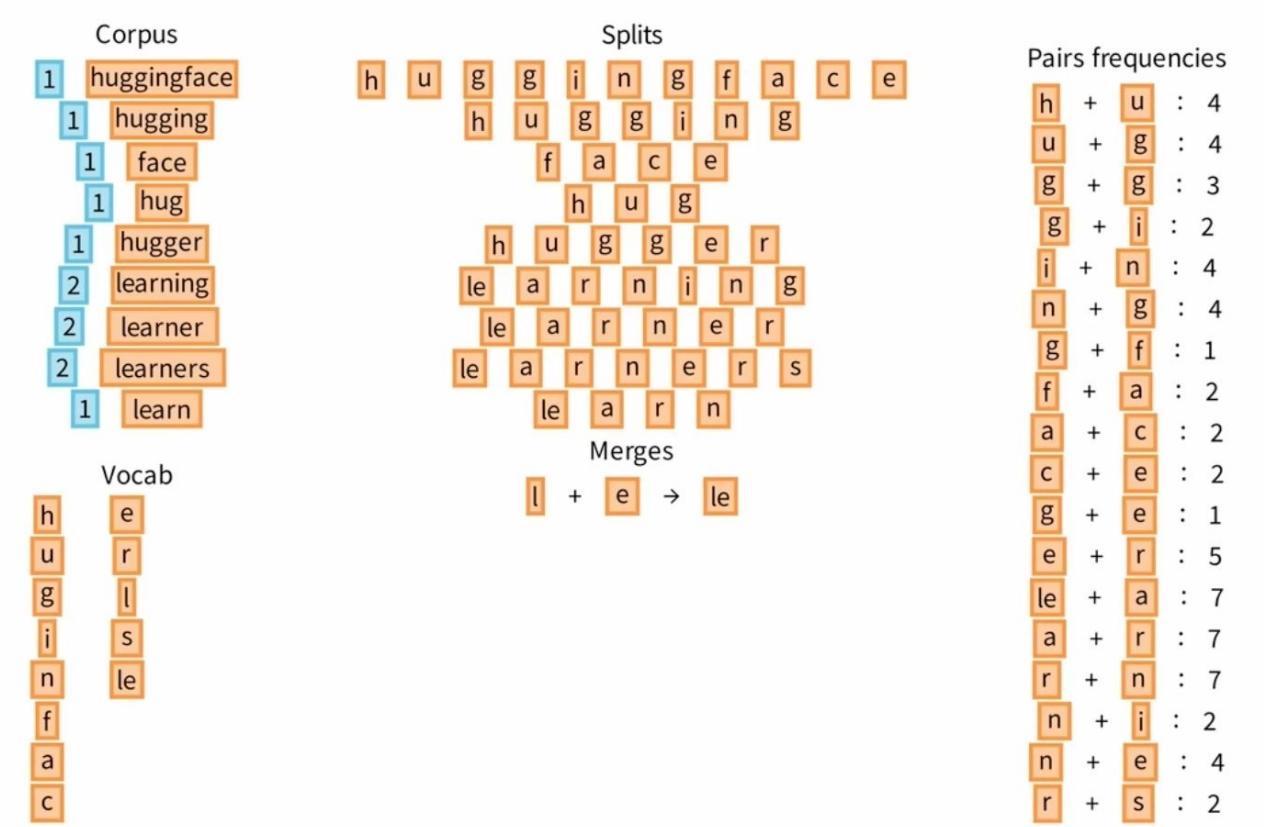
# BPE: example


Corpus
1 huggingface
1 hugging
1 face
1 hug
1 hugger
2 learning
2 learner
2 learners
1 learn
Vocab
h u g i n f a c e r l s
Splits
h u g g i n g f a c e
h u g g i n g
f a c e
h u g
h u g g e r
l e a r n i n g
l e a r n e r
l e a r n e r s
l e a r n
Pairs frequencies
h + u : 4
u + g : 4
g + g : 3
g + i : 2
i + n : 3
n + g : 3
g + f : 1
f + a : 2
a + c : 2
c + e : 2
g + e : 1
e + r : 3
l + e : 4
e + a : 4
a + r : 4
r + n : 4
n + i : 1
n + e : 2
r + s : 1

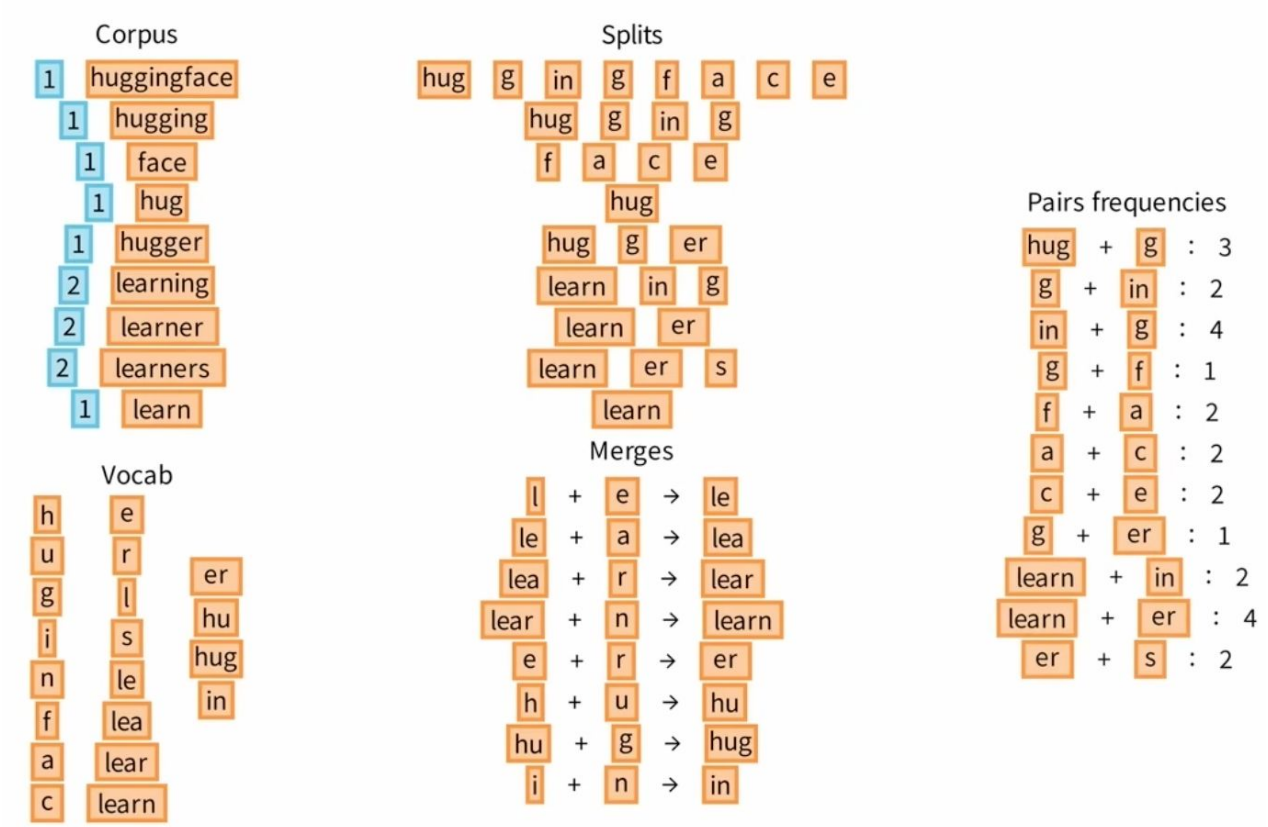
# BPE: example


Corpus
1 huggingface
1 hugging
1 face
1 hug
1 hugger
2 learning
2 learner
2 learners
1 learn
Splits
h u g g i n g f a c e
h u g g i n g
f a c e
h u g
h u g g e r
l e a r n i n g
l e a r n e r
l e a r n e r s
l e a r n
Pairs frequencies
Merges
l + e → le
Vocab
h u g i n f a c e r l s le

# BPE: example


Corpus
1 huggingface
1 hugging
1 face
1 hug
1 hugger
2 learning
2 learner
2 learners
1 learn
Vocab
h u g i n f a c e r l s le
Splits
h u g g i n g f a c e
h u g g i n g
f a c e
h u g
h u g g e r
le a r n i n g
le a r n e r
le a r n e r s
le a r n
Merges
l + e → le
Pairs frequencies
h + u : 4
u + g : 4
g + g : 3
g + i : 2
i + n : 4
n + g : 4
g + f : 1
f + a : 2
a + c : 2
c + e : 2
g + e : 1
e + r : 5
le + a : 7
a + r : 7
r + n : 7
n + i : 2
n + e : 4
r + s : 2

# BPE: example


Corpus
1 huggingface
1 hugging
1 face
1 hug
1 hugger
2 learning
2 learner
2 learners
1 learn
Vocab
h u g i n f a c
e r l s le lea lear learn
er hu hug in
Splits
hug g in g f a c e
hug g in g
f a c e
hug
hug g er
learn in g
learn er
learn er s
learn
Merges
l + e → le
le + a → lea
lea + r → lear
lear + n → learn
e + r → er
h + u → hu
hu + g → hug
i + n → in
Pairs frequencies
hug + g : 3
g + in : 2
in + g : 4
g + f : 1
f + a : 2
a + c : 2
c + e : 2
g + er : 1
learn + in : 2
learn + er : 4
er + s : 2

# WordPiece vs. BPE

## WordPiece

- Maximises the likelihood of the training data when updating vocabulary.
- Trains a language model starting on the base vocabulary and picks the pair ***<base vocab character> + <highest probability generated character>*** with the highest likelihood. This pair is added to the vocab and the language model is again trained on the new vocab. These steps repeat until the desired vocab is formed

## BPE

- Takes into account frequency of the pairs when updating vocabulary
- The process of merging tokens continues until the desired vocab is formed (desired number of tokens, number of merges, etc.)

# GPT

# GPT-2

- Transformer-based architecture
- Trained to predict the **next** word given previous context

Romeo and Juliet is a tragedy written

Romeo and Juliet is a tragedy written by

Romeo and Juliet is a tragedy written by William

Romeo and Juliet is a tragedy written by William Shakespeare

Romeo and Juliet is a tragedy written by William Shakespeare early

Romeo and Juliet is a tragedy written by William Shakespeare early in

Romeo and Juliet is a tragedy written by William Shakespeare early in his

Romeo and Juliet is a tragedy written by William Shakespeare early in his career

Romeo and Juliet is a tragedy written by William Shakespeare early in his career about

# GPT-2

- Transformer-based architecture
- Trained to predict the **next** word given previous context
- 1.5 billion parameters
- Trained on 8 million web-pages

# GPT-2

- Transformer-based architecture
- Trained to predict the **next** word given previous context
- 1.5 billion parameters
- Trained on 8 million web-pages

On language tasks (question answering, reading comprehension, summarization, translation) works well **WITHOUT** fine-tuning

# GPT-2: question answering

EXAMPLES

*Who wrote the book the origin of species?*

**Correct answer**: *Charles Darwin*
**Model answer**: `Charles Darwin`

*What is the largest state in the U.S. by land mass?*

**Correct answer**: *Alaska*
**Model answer**: `California`

# GPT-2: language modeling

EXAMPLE

*Both its sun-speckled shade and the cool grass beneath were a welcome respite after the stifling kitchen, and I was glad to relax against the tree's rough, brittle bark and begin my breakfast of buttery, toasted bread and fresh fruit. Even the water was tasty, it was so clean and cold. It almost made up for the lack of...*

**Correct answer**: *coffee*
**Model answer**: `food`

# GPT-2: machine translation

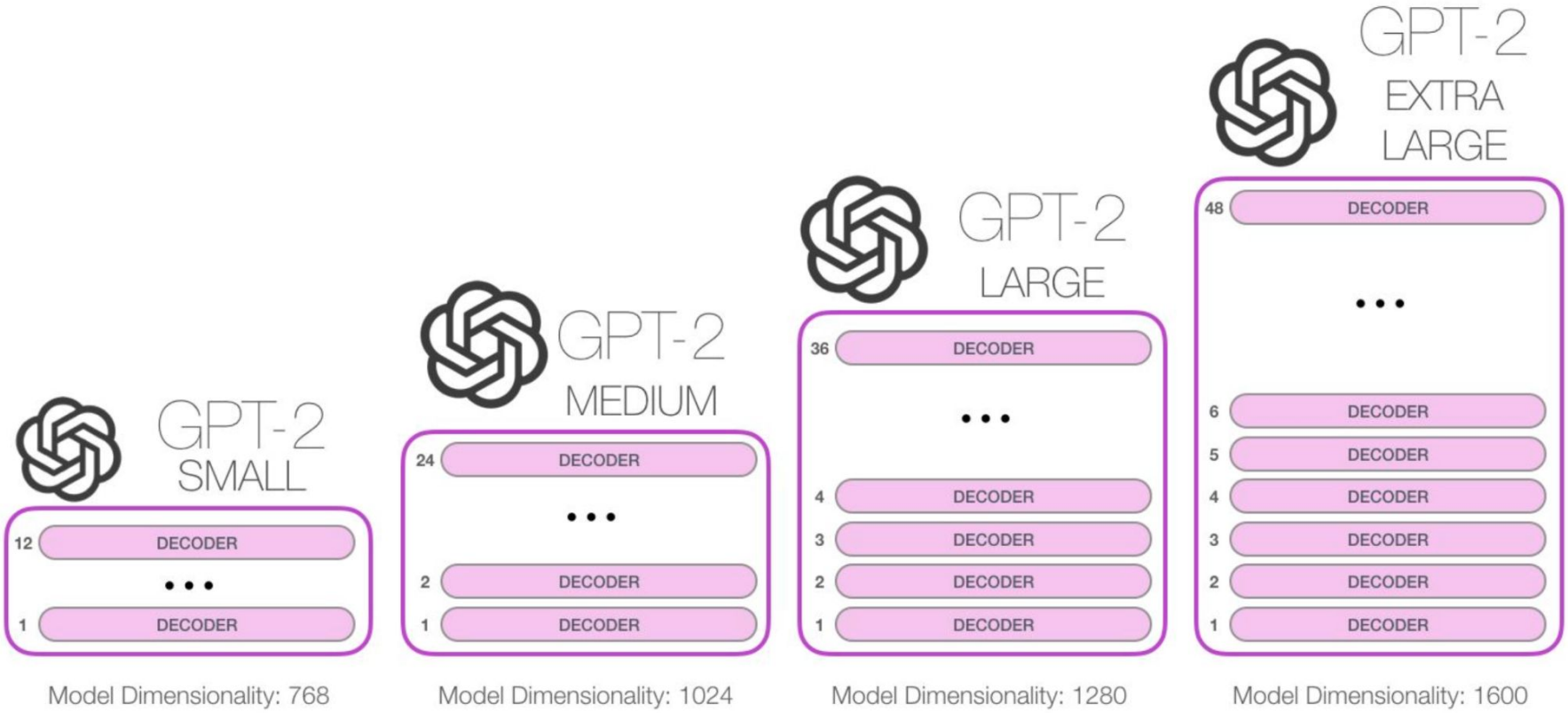
EXAMPLE

**French sentence**:
*Un homme a expliqué que l'opération gratuite qu'il avait subie pour soigner une hernie lui permettrait de travailler à nouveau.*

**Reference translation**:
*One man explained that the free hernia surgery he'd received will allow him to work again.*

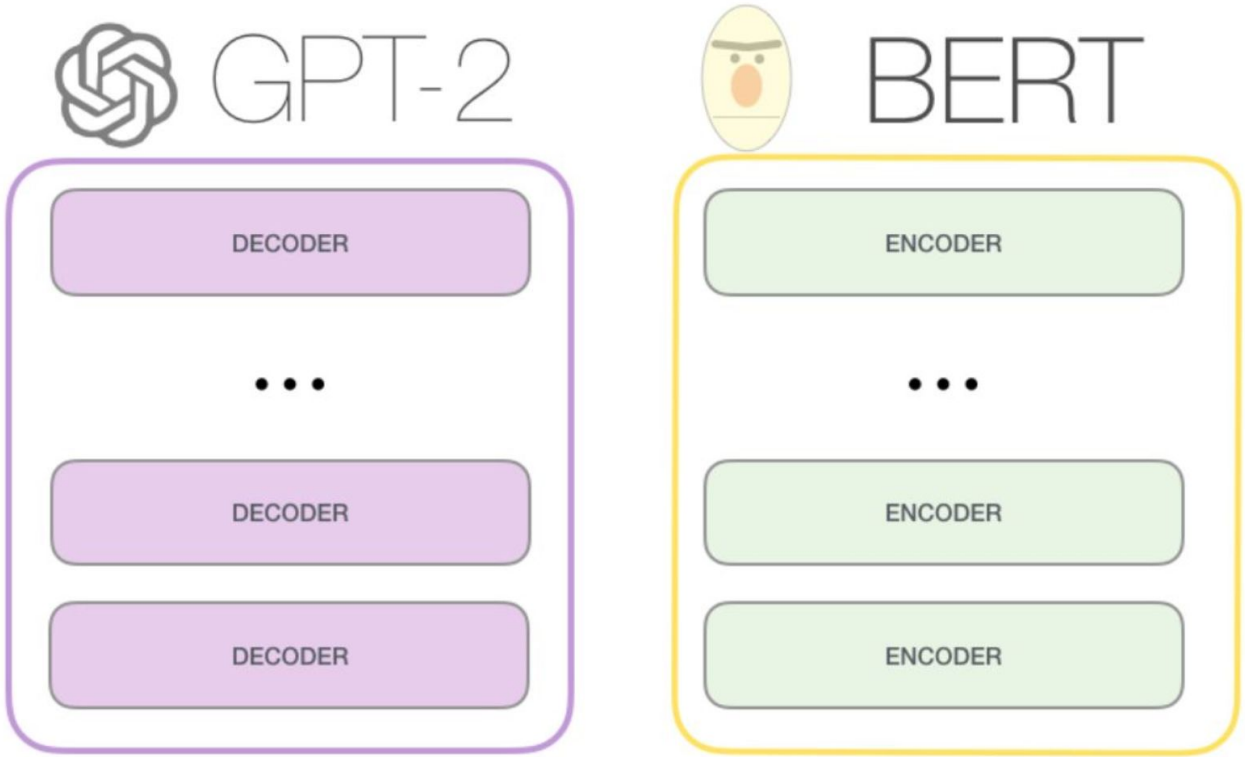
**Model translation**:
`A man told me that the operation gratuity he had been promised would not allow him to travel.`

# GPT-2



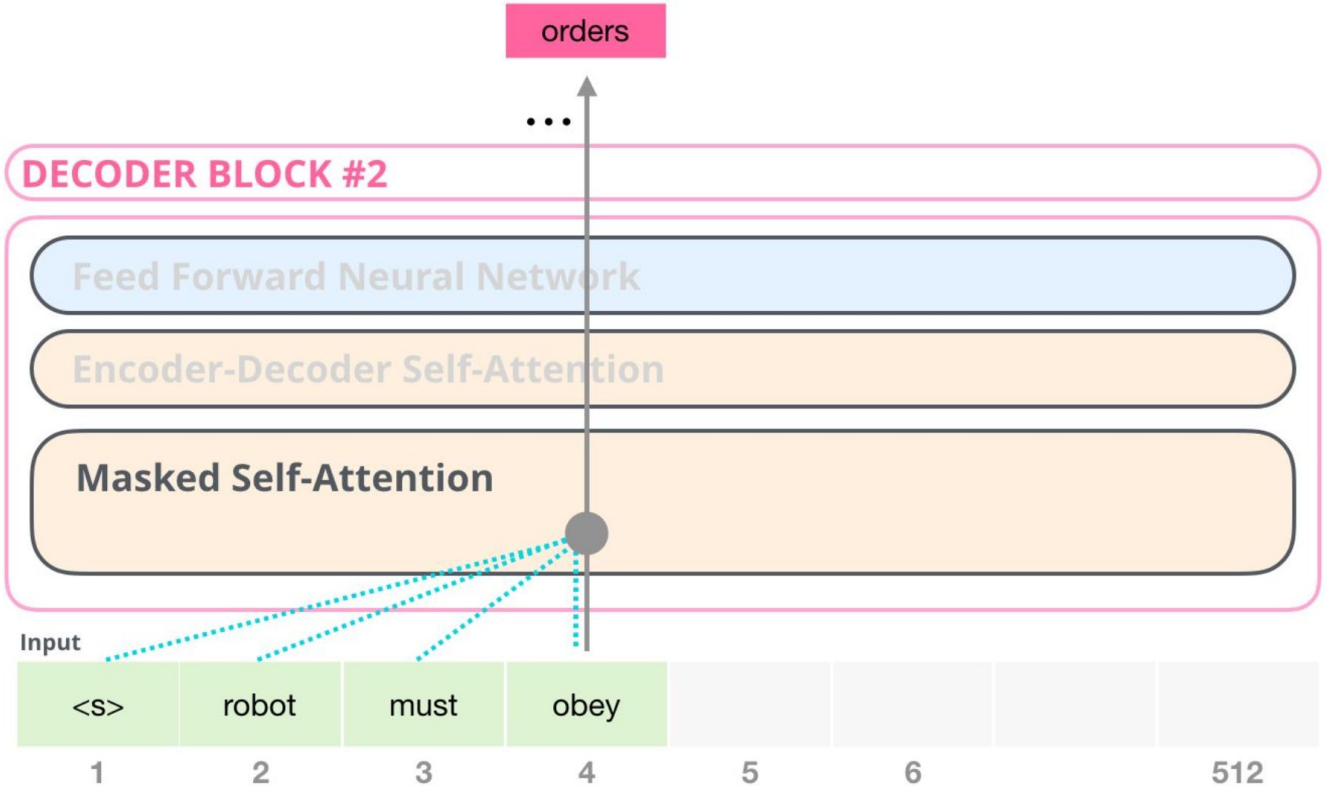

http://jalammar.github.io/illustrated-gpt2/

# GPT-2 vs. BERT



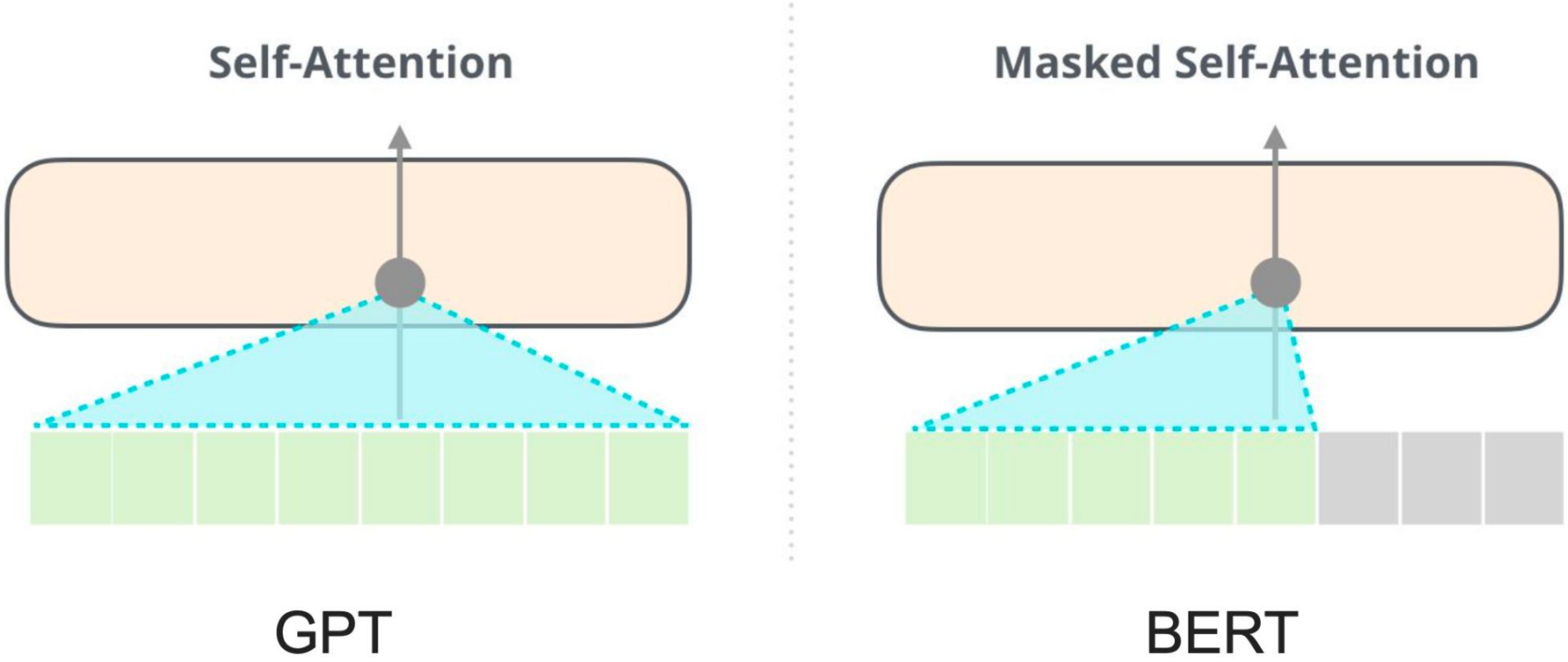

http://jalammar.github.io/illustrated-gpt2/

# GPT




http://jalammar.github.io/illustrated-gpt2/

# GPT



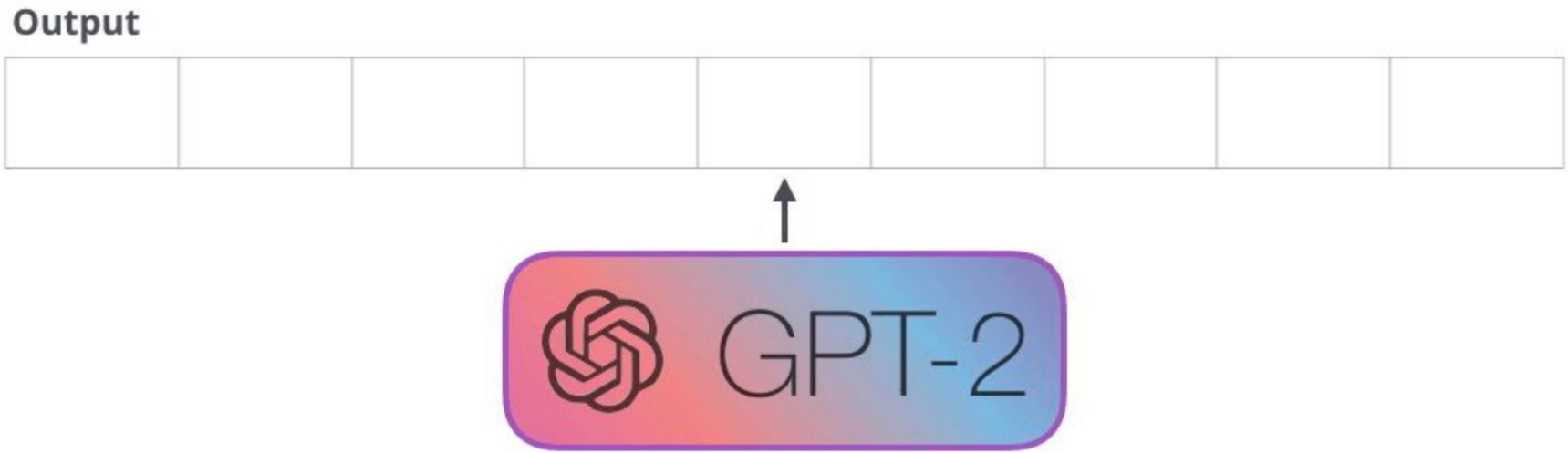

http://jalammar.github.io/illustrated-gpt2/

# Masking

| | Features | | | | Labels |
|---|---|---|---|---|---|
| | position: 1 | 2 | 3 | 4 | |
| Example: 1 | robot | must | obey | orders | must |
| 2 | robot | must | obey | orders | obey |
| 3 | robot | must | obey | orders | orders |
| 4 | robot | must | obey | orders | <eos> |

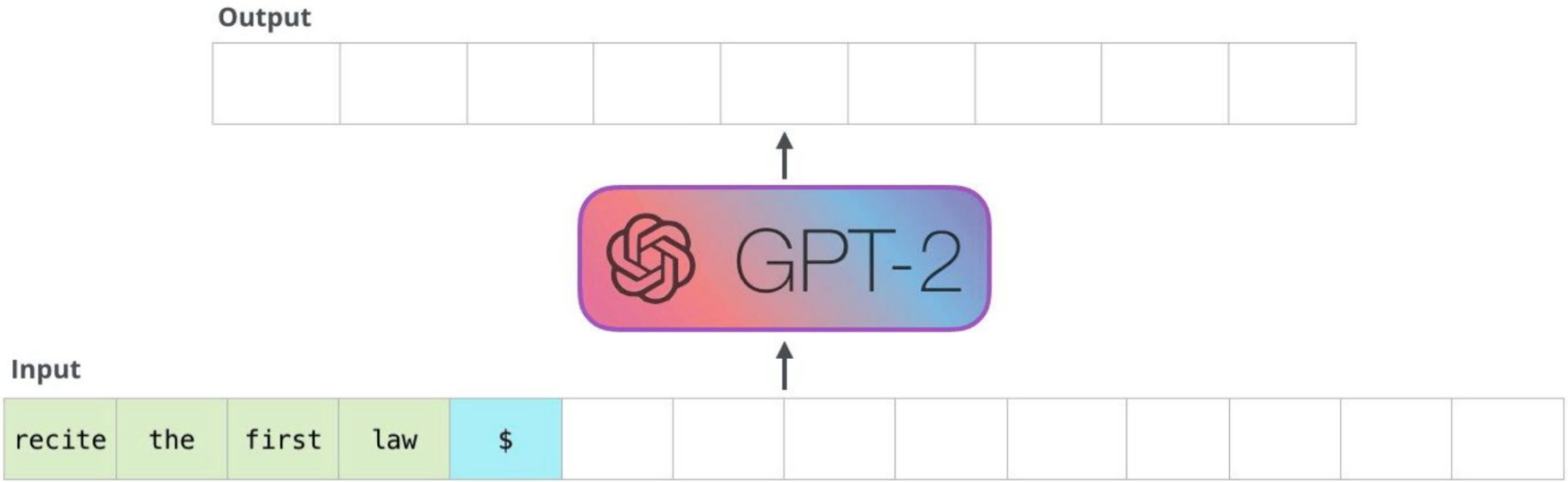
http://jalammar.github.io/illustrated-gpt2/

# GPT-2: autoregression



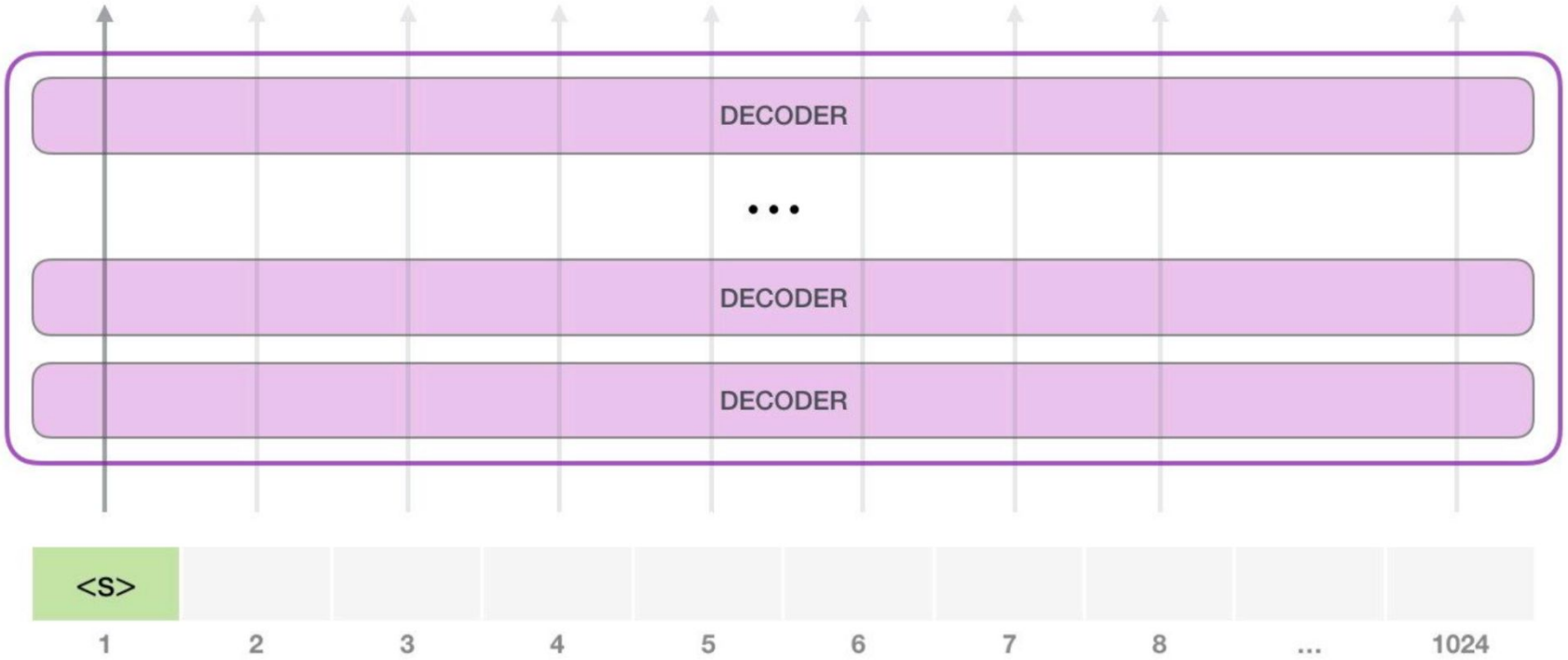

http://jalammar.github.io/illustrated-gpt2/

# GPT-2: autoregression



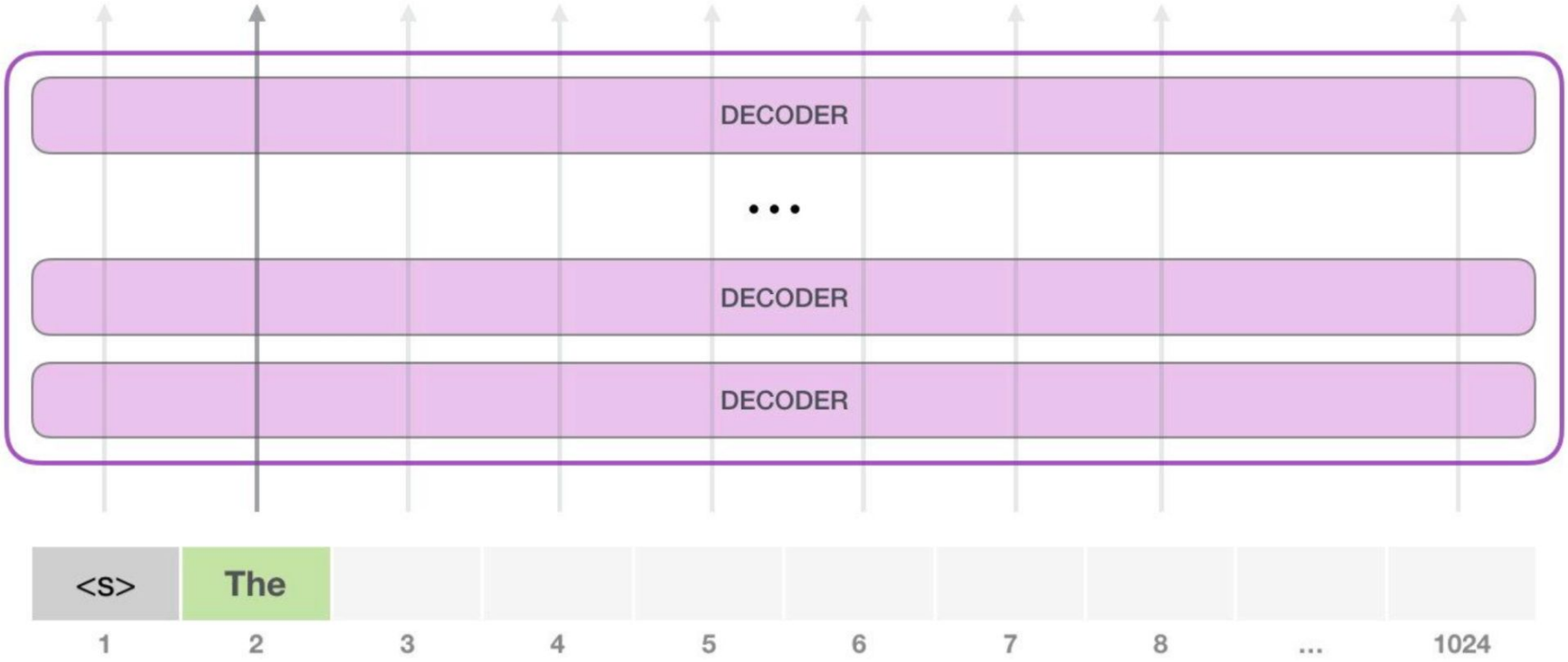

http://jalammar.github.io/illustrated-gpt2/

# GPT-2: text generation




http://jalammar.github.io/illustrated-gpt2/

# GPT-2: text generation




http://jalammar.github.io/illustrated-gpt2/

# GPT-2: fake news and hype

**New AI fake text generator may be too dangerous to ... - The Guardian**
https://www.theguardian.com/.../elon-musk-backed-ai-writes-convincing-news-fiction
4 days ago - The Elon Musk-backed nonprofit company OpenAI declines to release research publicly for fear of misuse. The creators of a revolutionary AI system that can write news stories and works of fiction – dubbed "deepfakes for text" – have taken the unusual step of not releasing ...

**OpenAI built a text generator so good, it's considered too dangerous to ...**
https://techcrunch.com/2019/02/17/openai-text-generator-dangerous/
12 hours ago - A storm is brewing over a new language model, built by non-profit artificial intelligence research company OpenAI, which it says is so good at ...

**The AI Text Generator That's Too Dangerous to Make Public | WIRED**
https://www.wired.com/story/ai-text-generator-too-dangerous-to-make-public/
4 days ago - In 2015, car-and-rocket man Elon Musk joined with influential startup backer Sam Altman to put artificial intelligence on a new, more open ...

**Elon Musk-backed AI Company Claims It Made a Text Generator ...**
https://gizmodo.com/elon-musk-backed-ai-company-claims-it-made-a-text-gener-183...
Elon Musk-backed AI Company Claims It Made a Text Generator That's **Too Dangerous to** Release · Rhett Jones · Friday 12:15pm · Filed to: OpenAI Filed to: ...

**Scientists have made an AI that they think is too dangerous to ...**
https://www.weforum.org/.../amazing-new-ai-churns-out-coherent-paragraphs-of-text/
3 days ago - Sample outputs suggest that the AI system is an extraordinary step forward, producing text rich with context, nuance and even something ...

**New AI Fake Text Generator May Be Too Dangerous To ... - Slashdot**
https://news.slashdot.org/.../new-ai-fake-text-generator-may-be-too-dangerous-to-rele...
3 days ago - An anonymous reader shares a report: The creators of a revolutionary AI system that can write news stories and works of fiction -- dubbed ...

Top stories

OpenAI built a text generator so good, it's considered too dangerous to release
TechCrunch
11 hours ago

Elon Musk's AI company created a fake news generator it's too scared to make public
BGR.com
9 hours ago

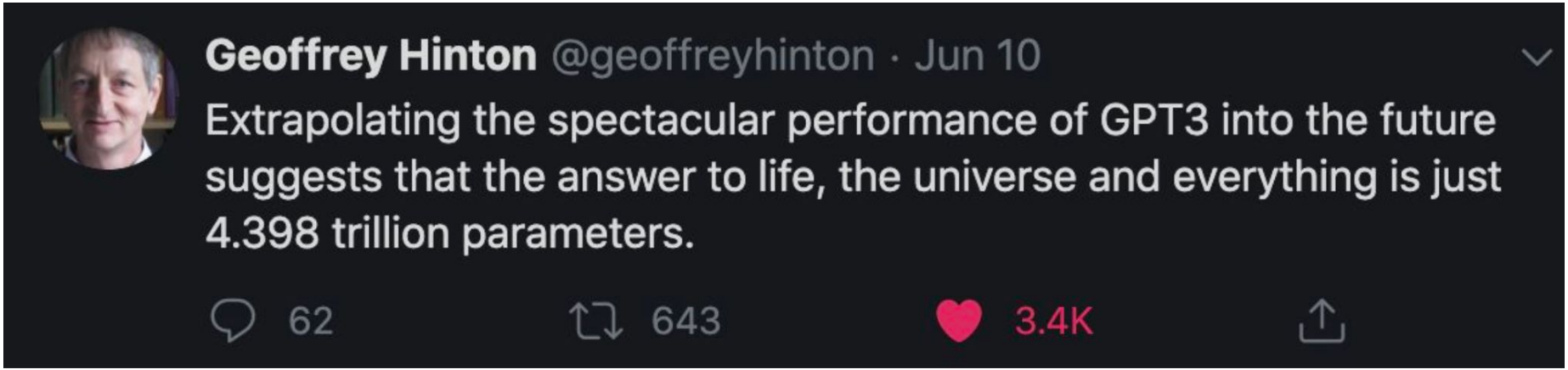
The AI That Can Write A Fake News Story From A Handful Of Words
NDTV.com
2 hours ago

When Is Technology Too Dangerous to Release to the Public?
Slate • 2 days ago

Scientists Developed an AI So Advanced They Say It's Too Dangerous to Release
ScienceAlert • 6 days ago

# GPT-3, GPT-3.5 and GPT-4

- **GPT-1:** 117 million parameters
- **GPT-2:** 1.5 billion parameters
- **GPT-3:** **175 billion** parameters
- **GPT-3.5:** transitional version between GPT-3 and GPT-4



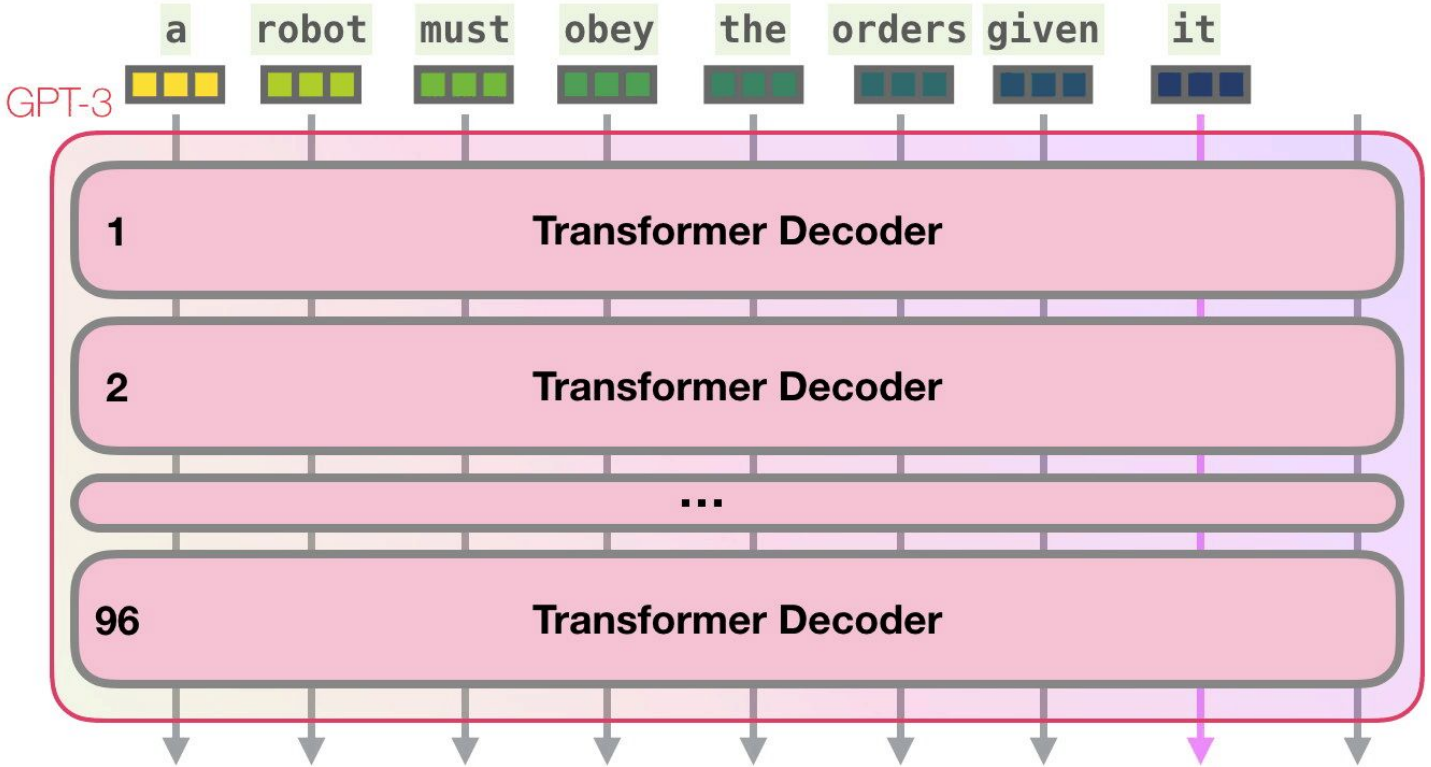
# GPT-3, GPT-3.5 and GPT-4

- **GPT-1:** 117 million parameters, context size 512
- **GPT-2:** 1.5 billion parameters, context size 1024
- **GPT-3:** **175 billion** parameters, context size 2048

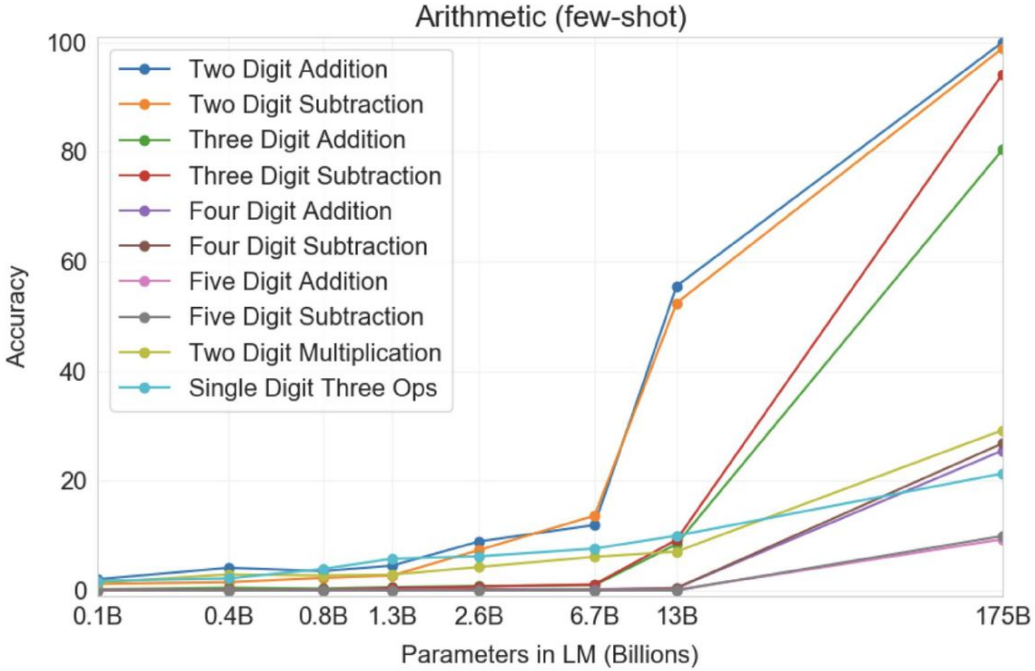
Batch size

- **GPT-1:** 64
- **GPT-2:** 512
- **GPT-3:** **~3.2 million**

# GPT-3: architecture

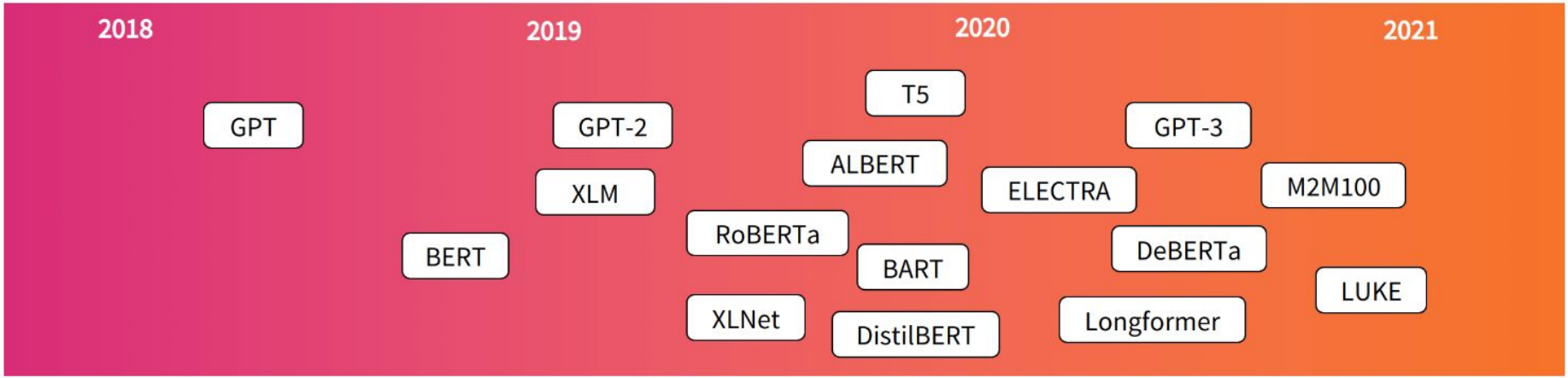


# GPT-3: evaluation




| Setting | 2D+ | 2D- | 3D+ | 3D- | 4D+ | 4D- |
|---|---|---|---|---|---|---|
| GPT-3 Zero-shot | 76.9 | 58.0 | 34.2 | 48.3 | 4.0 | 7.5 |
| GPT-3 One-shot | 99.6 | 86.4 | 65.5 | 78.7 | 14.0 | 14.0 |
| GPT-3 Few-shot | 100.0 | 98.9 | 80.4 | 94.2 | 25.5 | 26.8 |

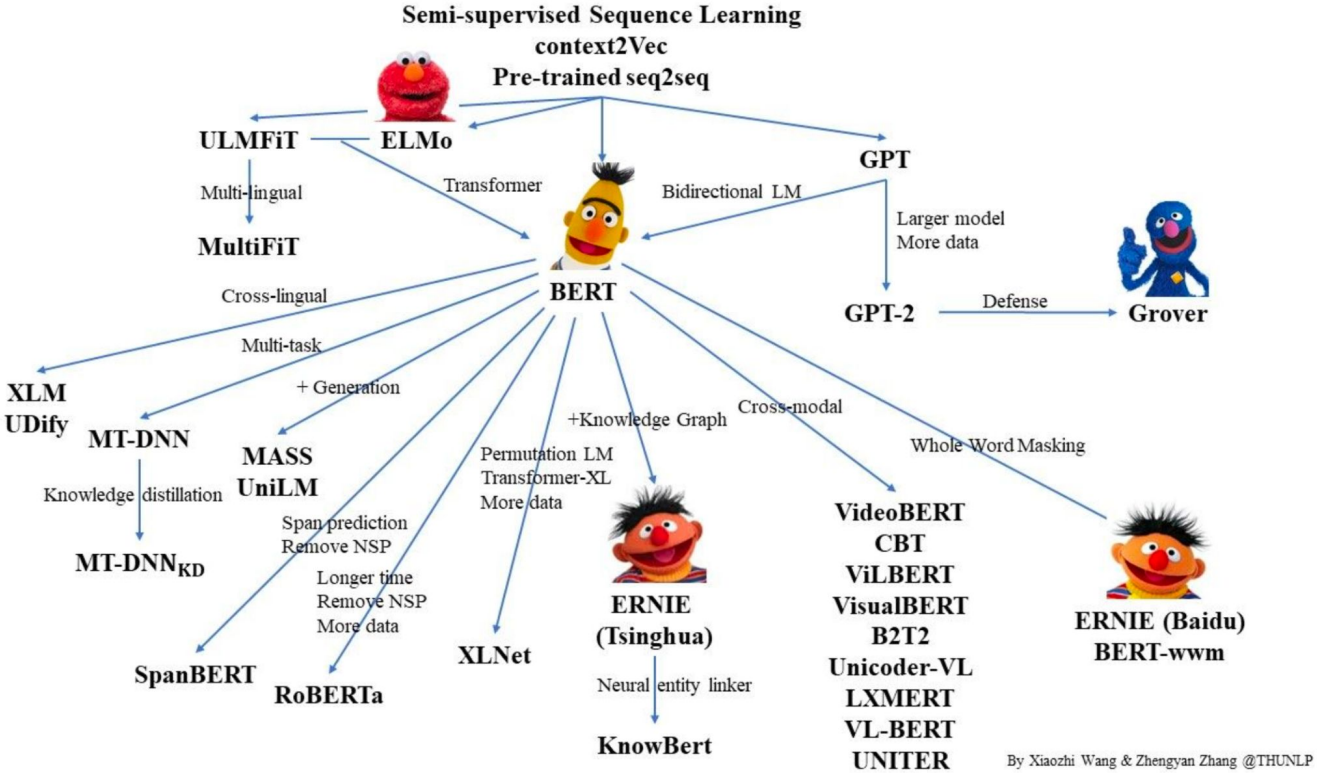
# BERTology




[BERTology paper]

# BERTology

- ➔ **Transformer-XL**
- ➔ XLNet
- ➔ MPNet
- ➔ RoBERTa
- ➔ ALBERT
- ➔ DistilBERT
- ➔ BART
- ➔ BigBird
- ➔ ELECTRA
- ➔ DeBERTa




[BERTology paper]

# Transformer-XL

- Vanila Transformer works with a fixed-length context at training time. That's why:
  - the algorithm is not able to model dependencies that are longer than a fixed length.
  - the segments usually do not respect the sentence boundaries, resulting in context fragmentation which leads to inefficient optimization.

https://arxiv.org/pdf/1901.02860.pdf

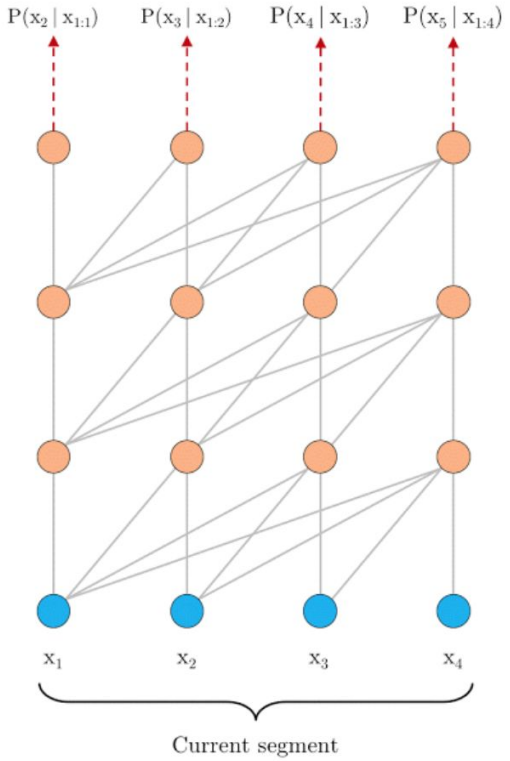
# Segment-level Recurrence

- During training, the representations computed for the previous segment are fixed and cached to be reused as an extended context when the model processes the next new segment.

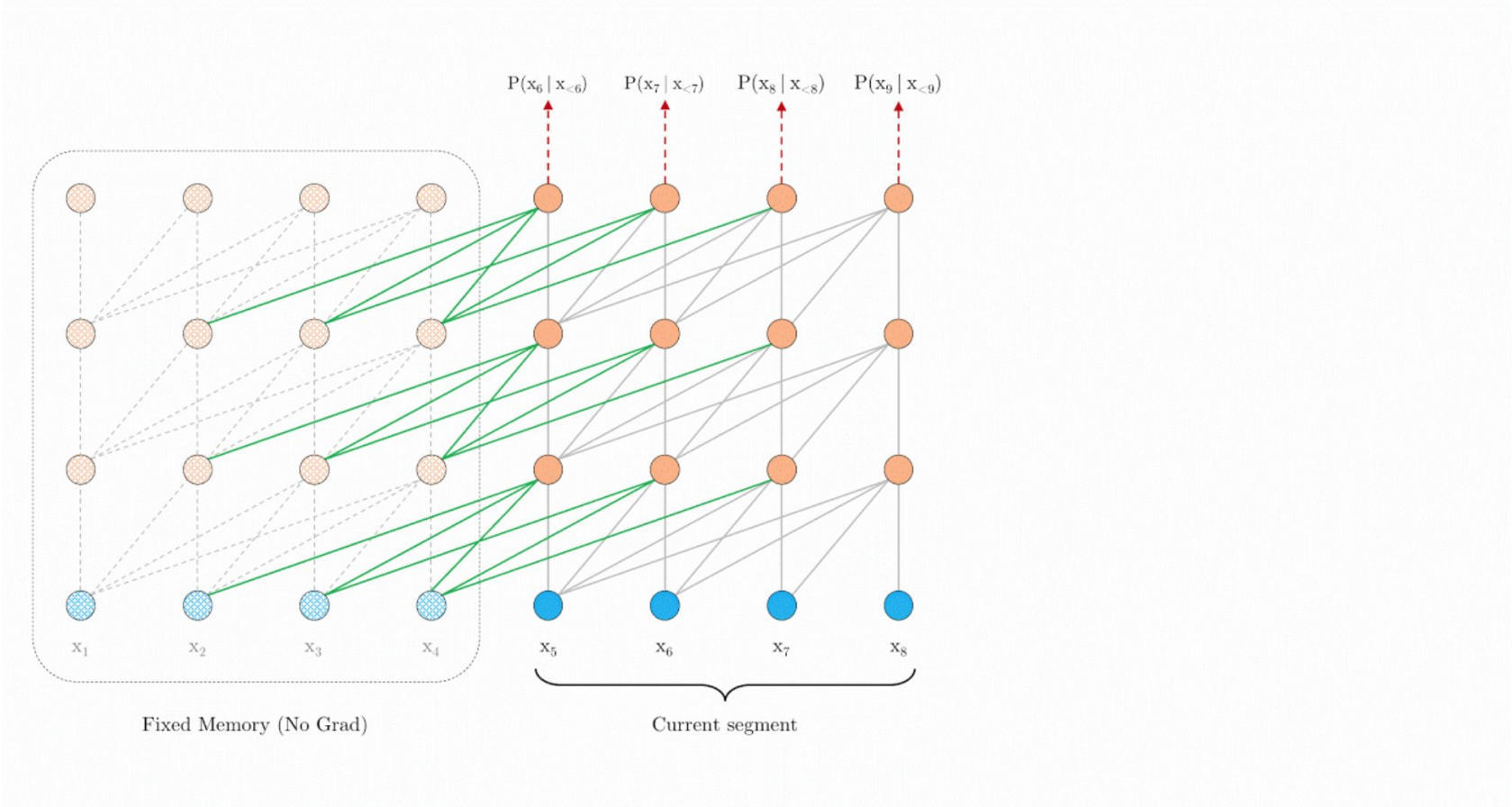
- Contextual information is now able to flow across segment boundaries.

- Recurrence mechanism also resolves the context fragmentation issue, providing necessary context for tokens in the front of a new segment.

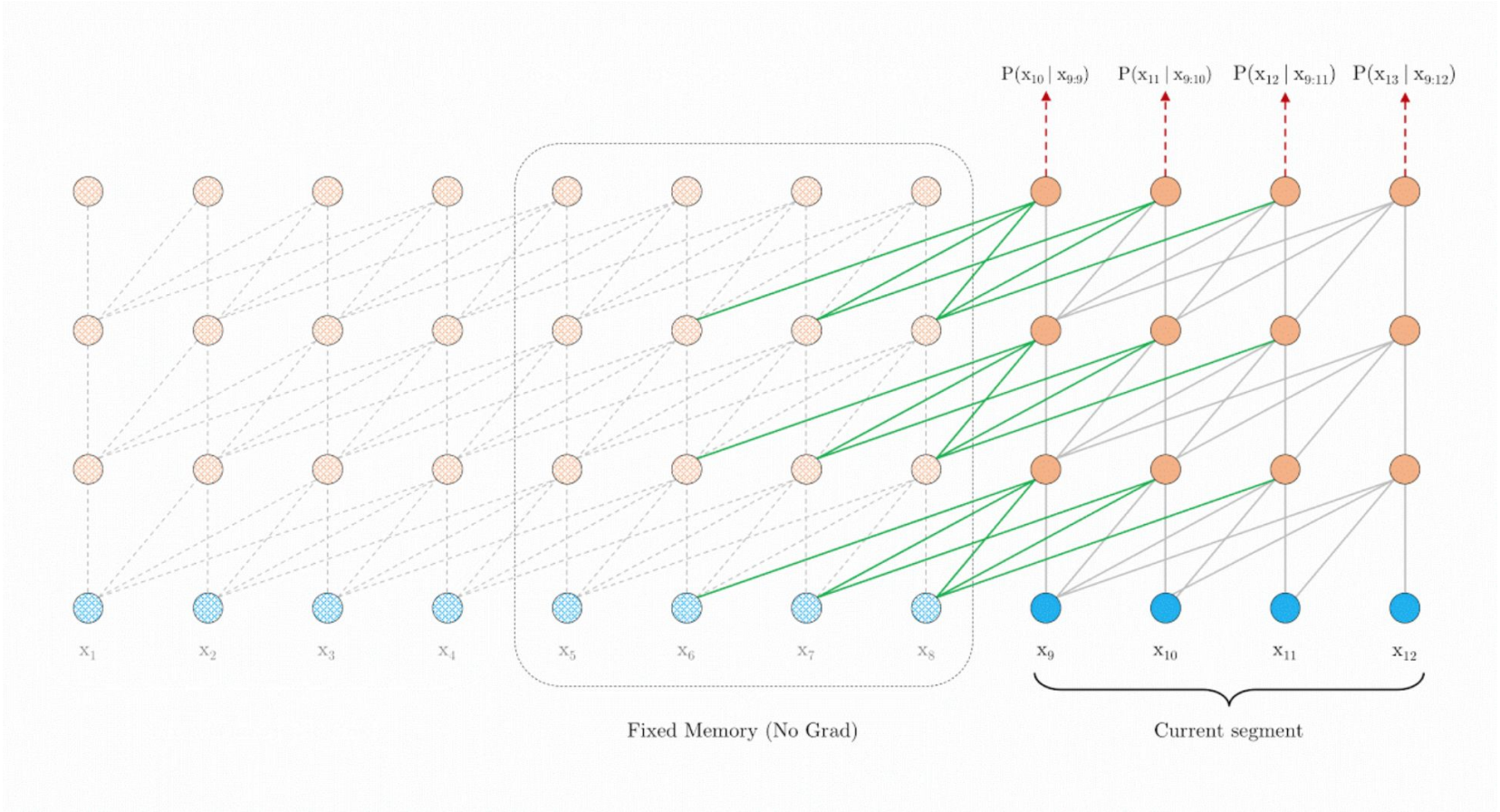
# Segment-level Recurrence



# Segment-level Recurrence



# Segment-level Recurrence



# Relative Positional Encodings

How to **keep positional information coherent** when reusing the states?

- **Vanilla Transformer:** element-wise addition of the word embeddings and the positional encodings
- **Transformer-XL:** injects positional information into the attention score of each layer instead of embeddings
  - now only relative positional information is preserved
  - absolute position can be recovered recursively from relative distances

# Relative Positional Encodings

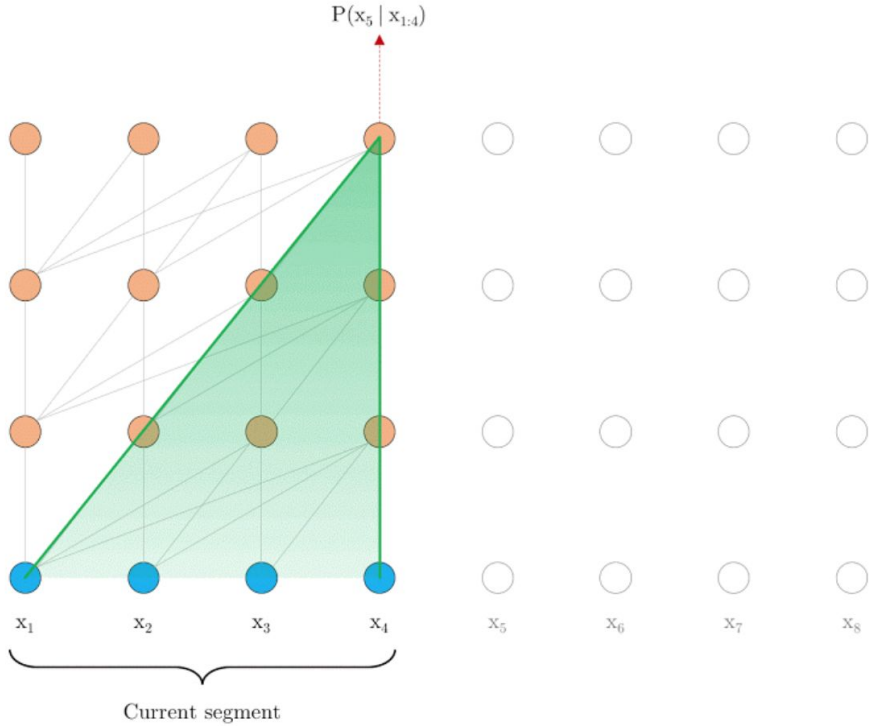
How to keep **positional information coherent** when reusing the states?

- **Vanilla Transformer:** element-wise addition of the word embeddings and the positional encodings
- **Transformer-XL:** injects positional information into the attention score of each layer instead of embeddings (now only relative positional information is preserved)

As a result we get:

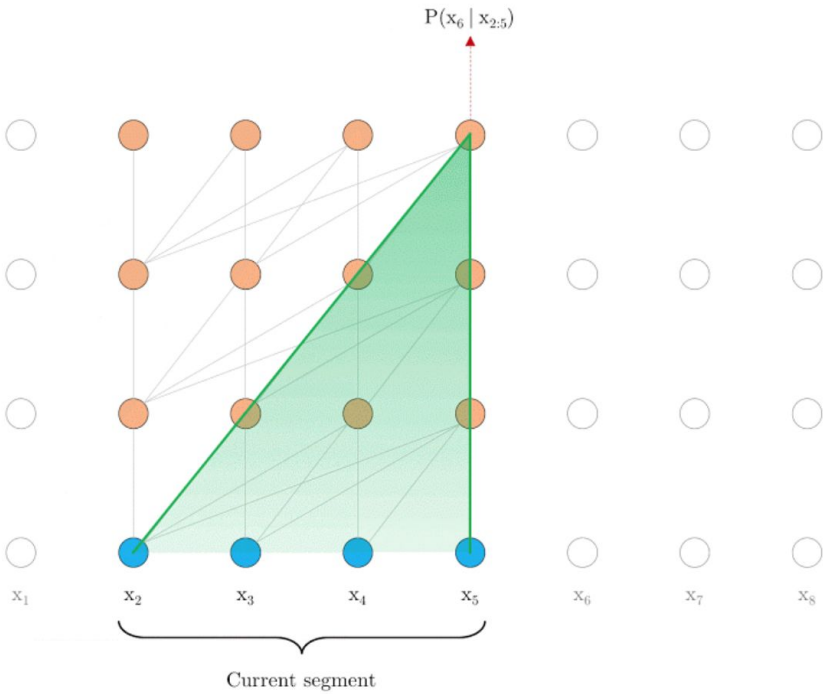
- more generalization to longer sequences at test time
- longer effective context

# Vanila Transformer vs. Transformer-XL



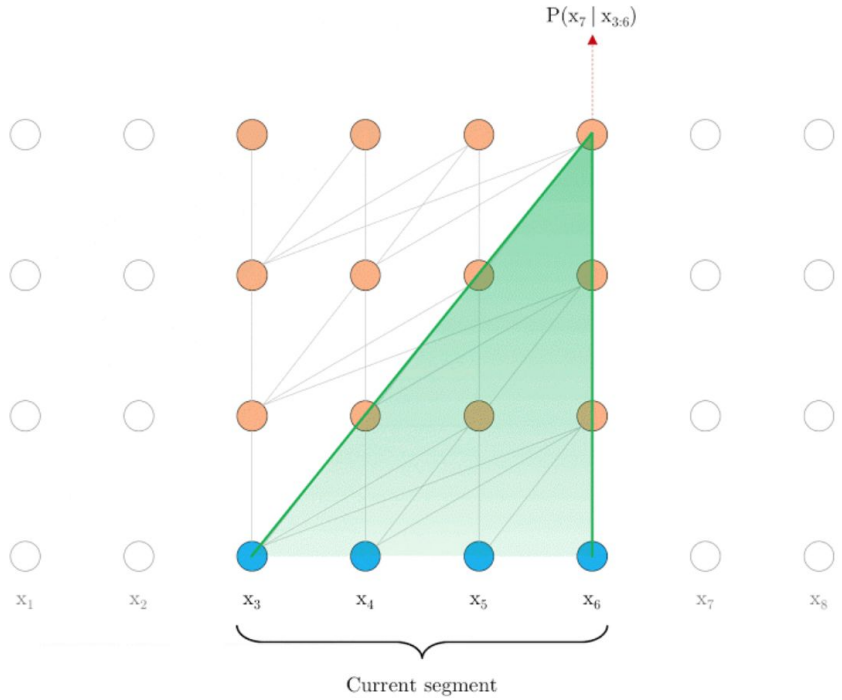

Vanilla Transformer with a fixed-length context at evaluation time

# Vanila Transformer vs. Transformer-XL



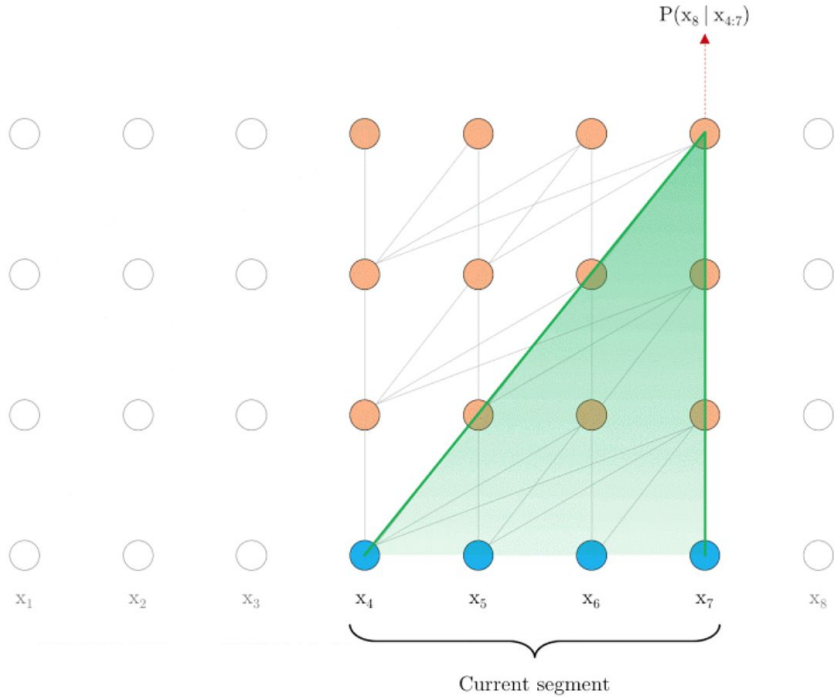

Vanilla Transformer with a fixed-length context at evaluation time

# Vanila Transformer vs. Transformer-XL



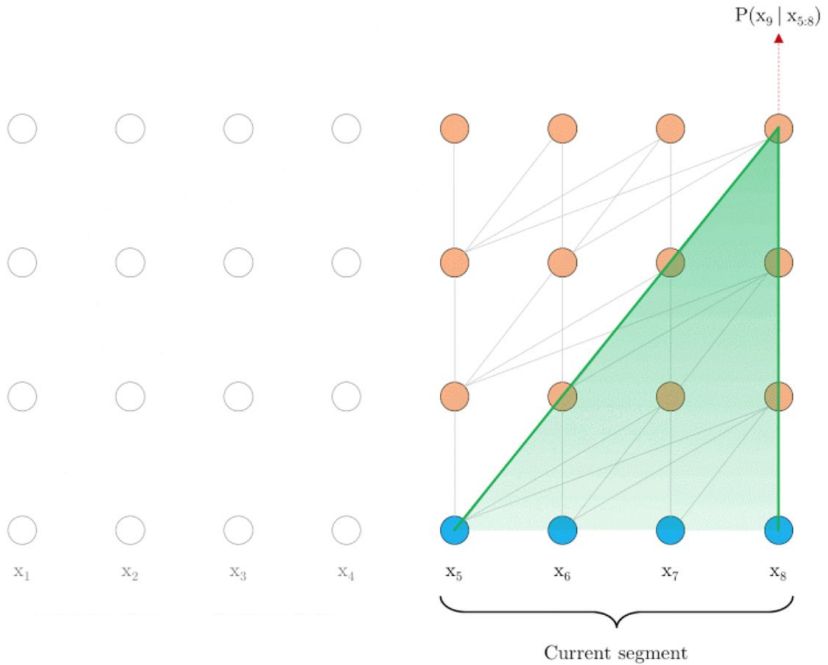

Vanilla Transformer with a fixed-length context at evaluation time

# Vanila Transformer vs. Transformer-XL



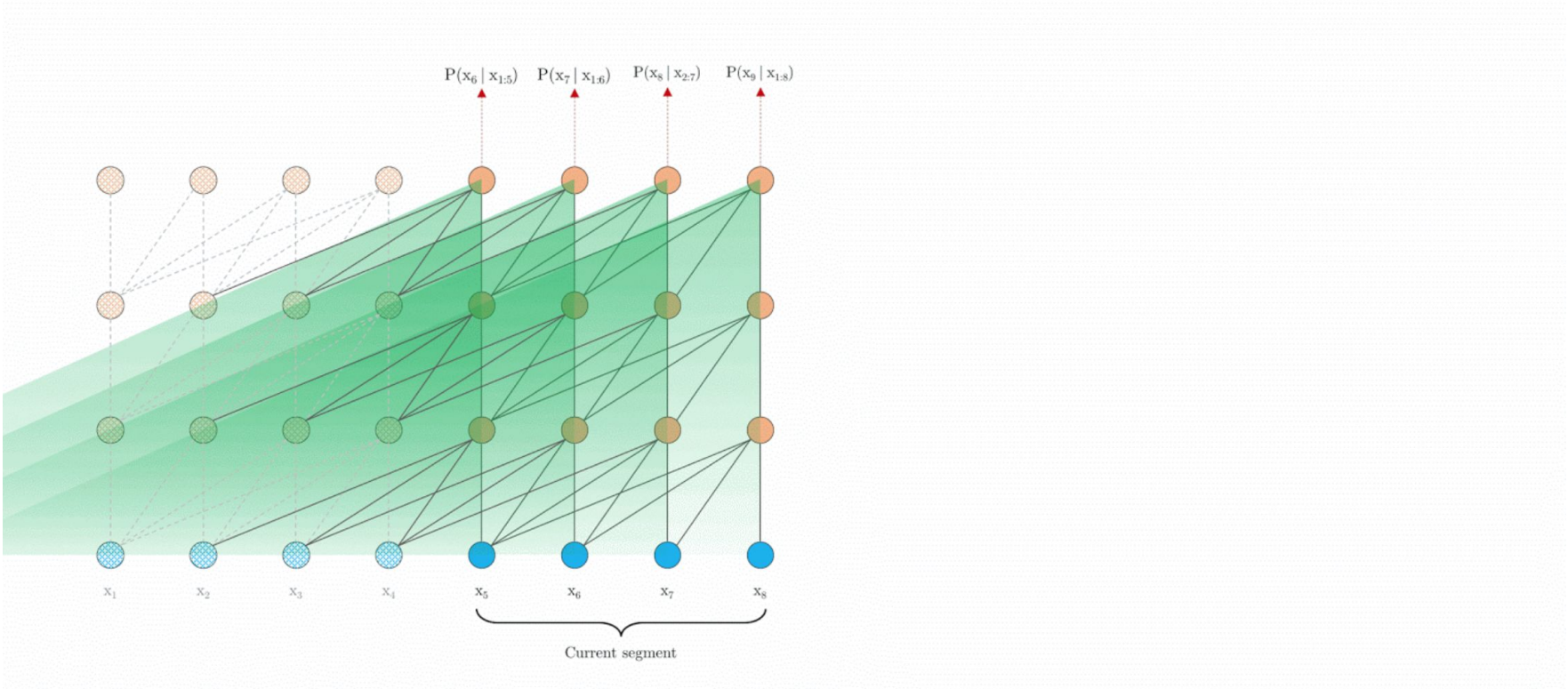

Vanilla Transformer with a fixed-length context at evaluation time

# Vanila Transformer vs. Transformer-XL



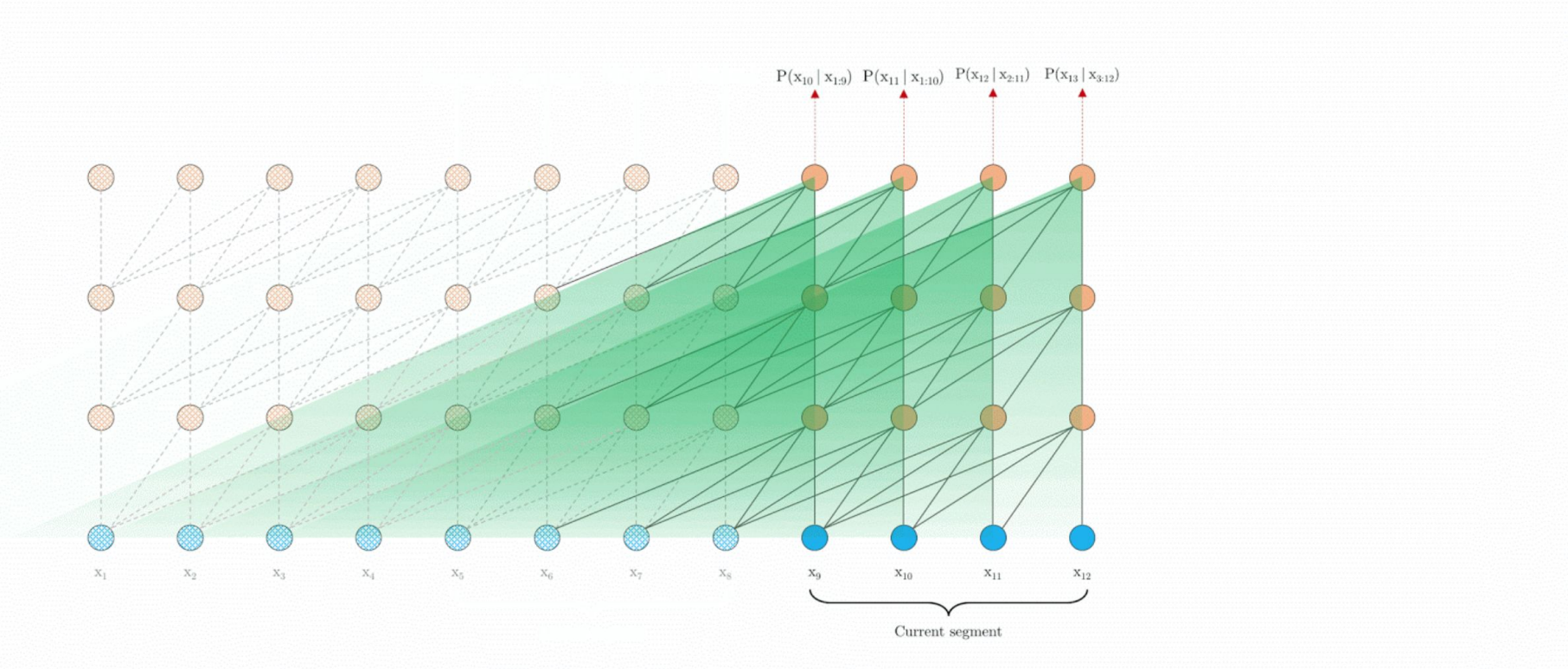

Vanilla Transformer with a fixed-length context at evaluation time

# Vanila Transformer vs. Transformer-XL



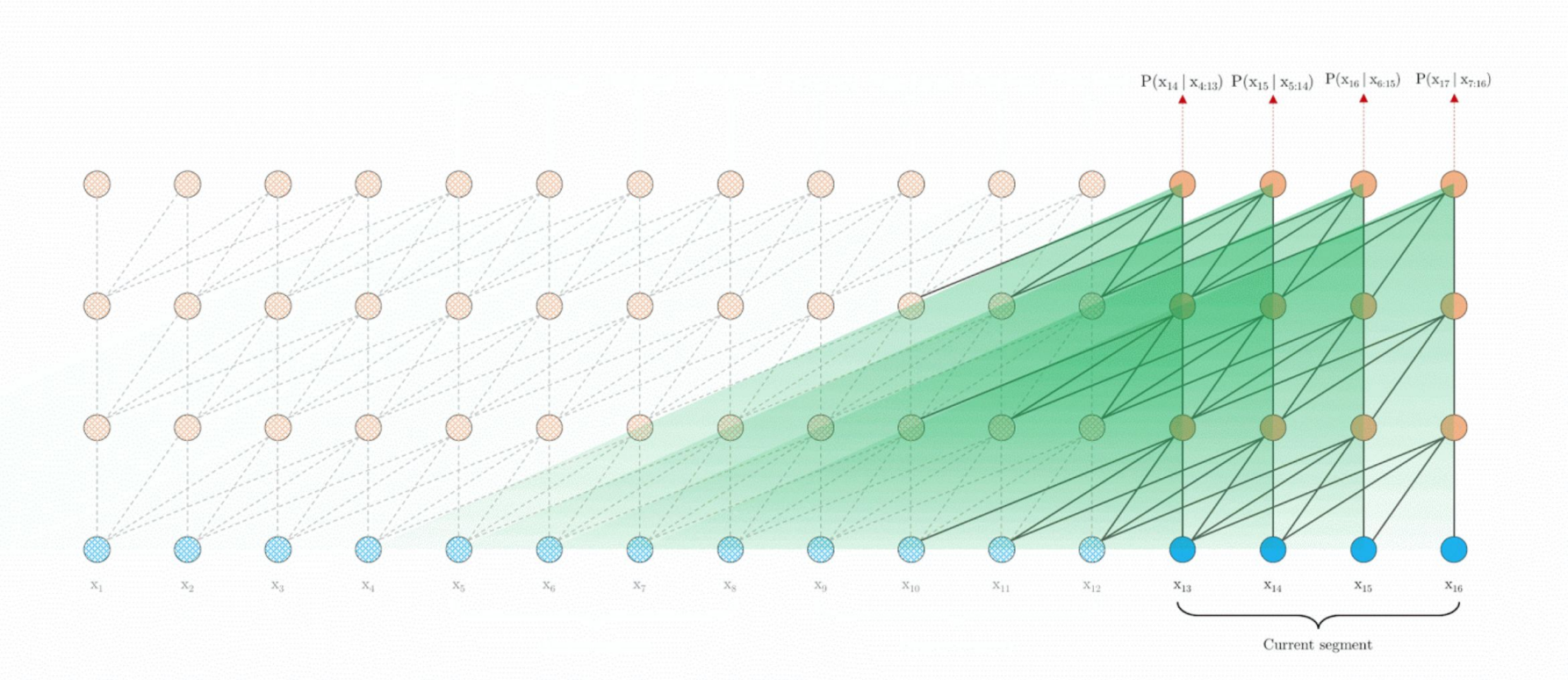

Transformer-XL with segment-level recurrence at evaluation time

# Vanila Transformer vs. Transformer-XL




Transformer-XL with segment-level recurrence at evaluation time

# Vanila Transformer vs. Transformer-XL




Transformer-XL with segment-level recurrence at evaluation time

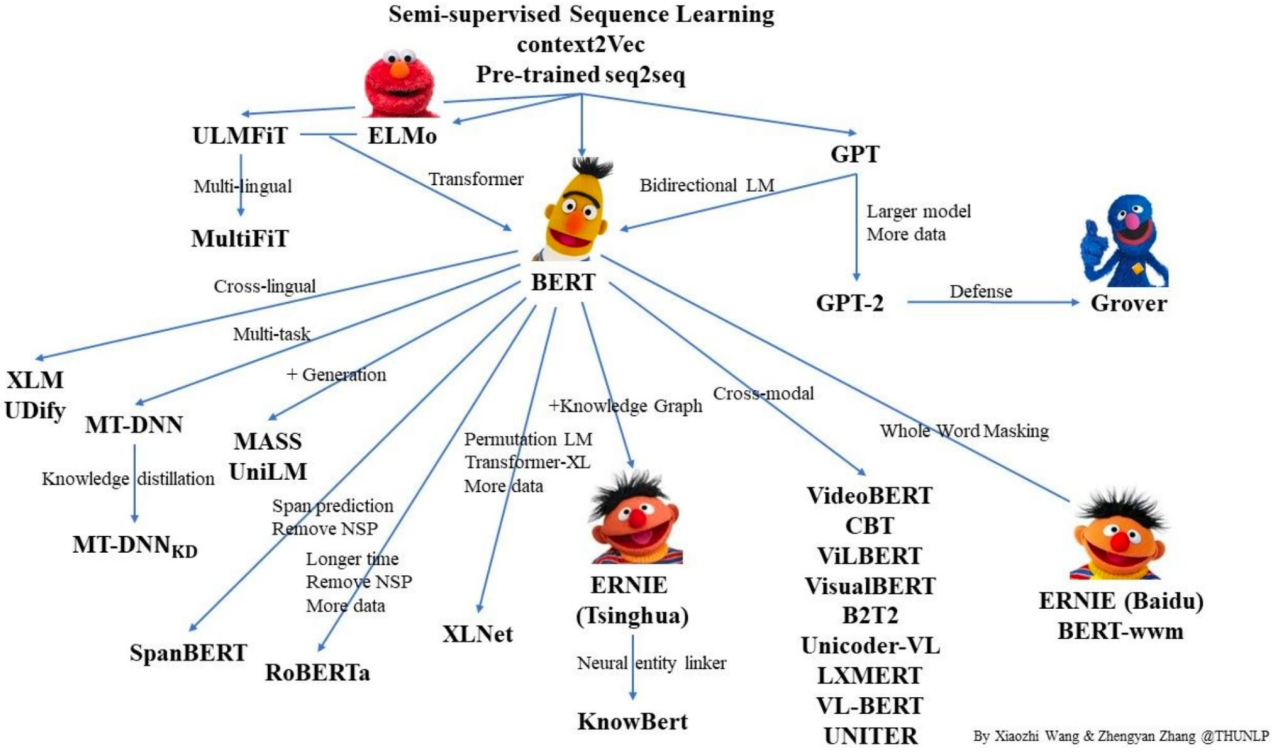
# Vanila Transformer vs. Transformer-XL

- Transformer-XL learns dependency that is about 80% longer than RNNs and 450% longer than vanilla Transformers

- Transformer-XL is up to 1,800+ times faster than a vanilla Transformer during evaluation on language modeling tasks, because no re-computation is needed

# BERTology

- ➔ Transformer-XL
- ➔ **XLNet**
- ➔ **MPNet**
- ➔ RoBERTa
- ➔ ALBERT
- ➔ DistilBERT
- ➔ BART
- ➔ BigBird
- ➔ ELECTRA
- ➔ DeBERTa




[BERTology paper]

# BERT problems

- The [MASK] token used in training does not appear during fine-tuning

# BERT problems

- BERT generates predictions independently

I went to [MASK] [MASK] and saw the [MASK] [MASK] [MASK]

# BERT problems

- BERT generates predictions independently

I went to [MASK] [MASK] and saw the [MASK] [MASK] [MASK]

**Ground truth solutions:**

- I went to New York and saw the Empire State building.

- I went to San Francisco and saw the Golden Gate bridge.

# BERT problems

- BERT generates predictions independently

I went to [MASK] [MASK] and saw the [MASK] [MASK] [MASK]

**BERT solutions:**

- I went to New York and saw the Empire State building.
- I went to San Francisco and saw the Golden Gate bridge.
- I went to New York and saw the Golden Gate bridge.

# XLNet

- XLNET is a generalized autoregressive model
- Permutation language modeling (PLM)
- Integrates the idea of auto-regressive models and bi-directional context modeling
- Outperforms BERT on 20 tasks

XLNet: Generalized Autoregressive Pretraining for Language Understanding

# XLNet

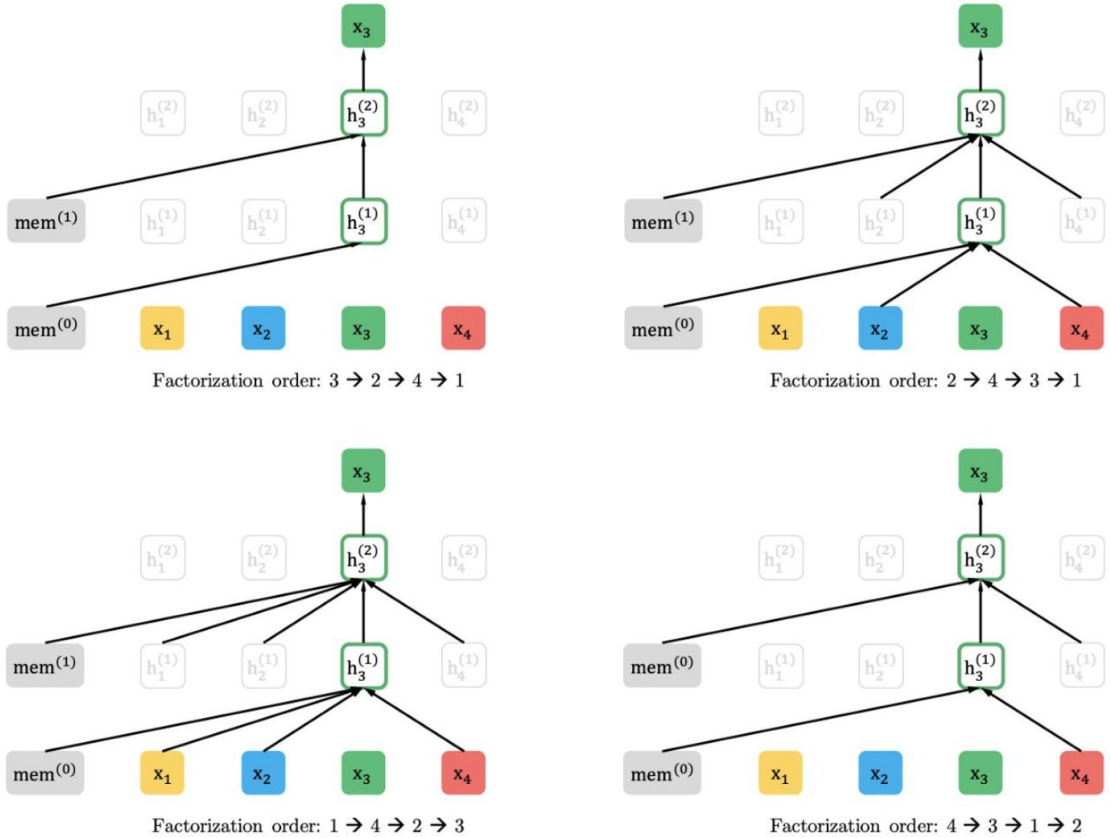
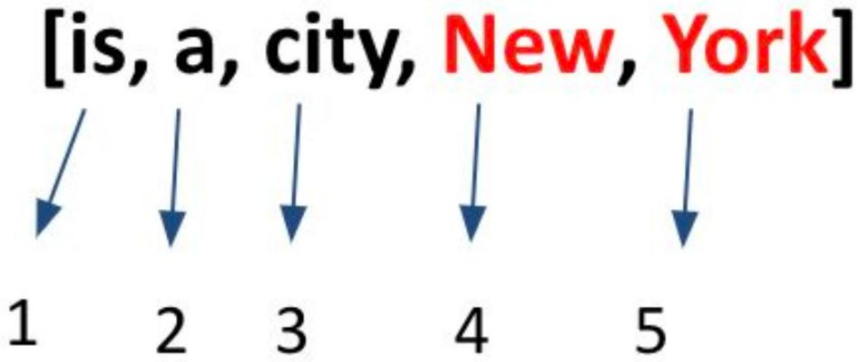



Figure 1: Illustration of the permutation language modeling objective for predicting $x_3$ given the same input sequence $\mathbf{x}$ but with different factorization orders.

[XLNet: Generalized Autoregressive Pretraining for Language Understanding]

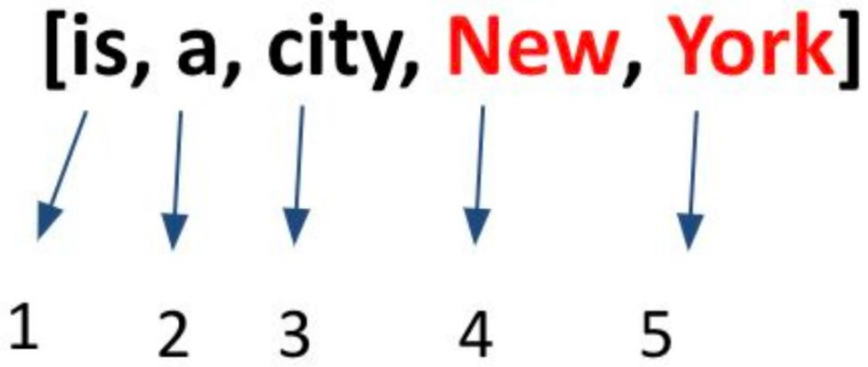
# XLNet vs. BERT




BERT $\log P(\text{New} \mid \textbf{is a city}) + \log P(\text{York} \mid \textbf{is a city})$

XLNET $\log P(\text{New} \mid \textbf{is a city}) + \log P(\text{York} \mid \textbf{New, is a city})$

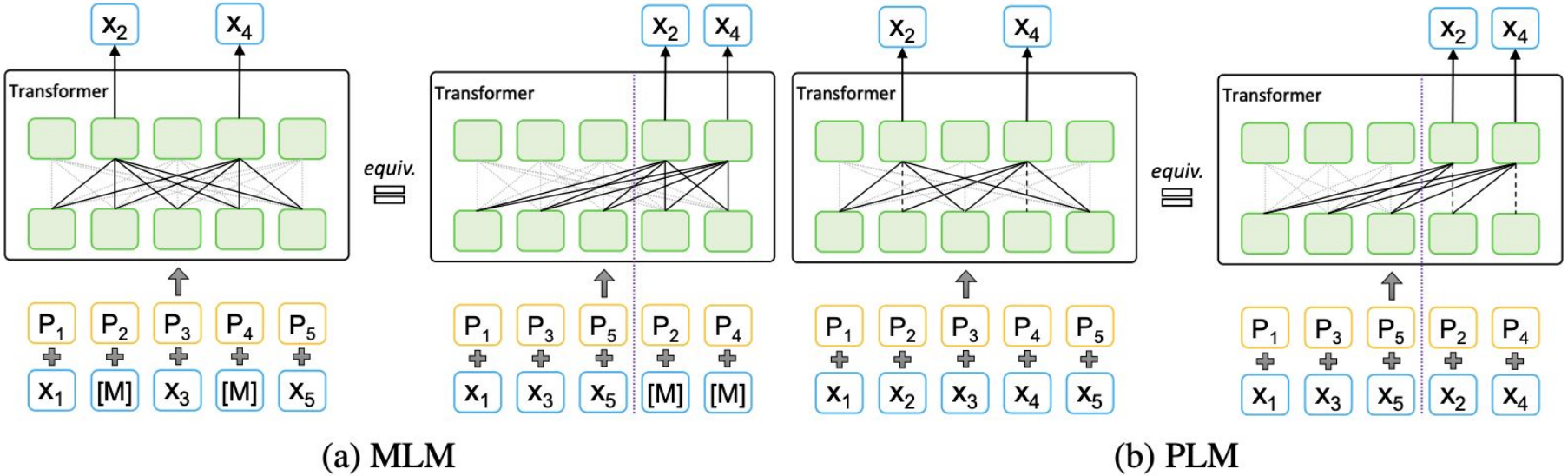
# XLNet vs. BERT




BERT → $\log P(\text{New} \mid \textbf{is a city}) + \log P(\text{York} \mid \textbf{is a city})$

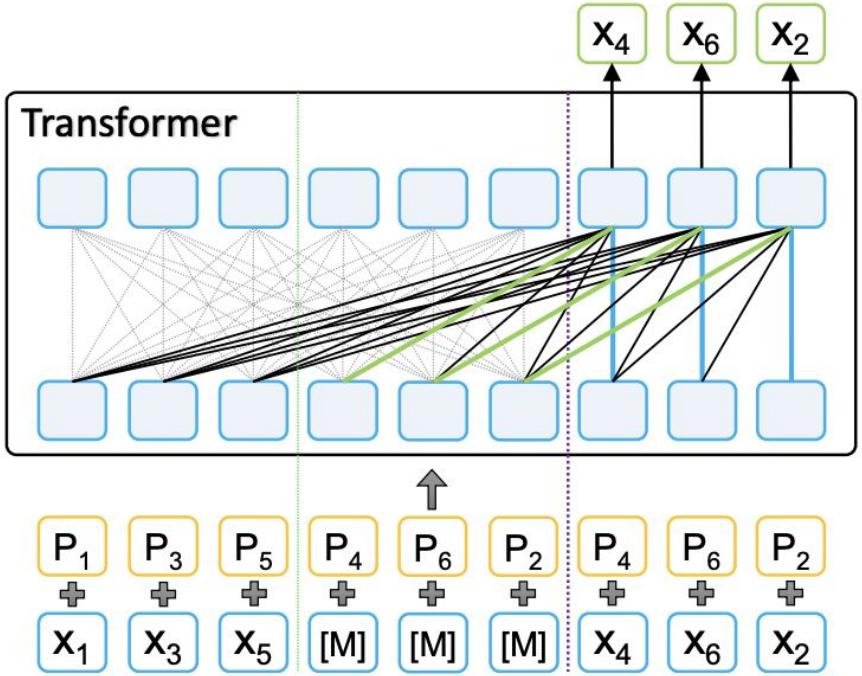
XLNET → $\log P(\text{New} \mid \textbf{is a city}) + \log P(\text{York} \mid \textbf{New, is a city})$

# MLM + PLM = MPNet

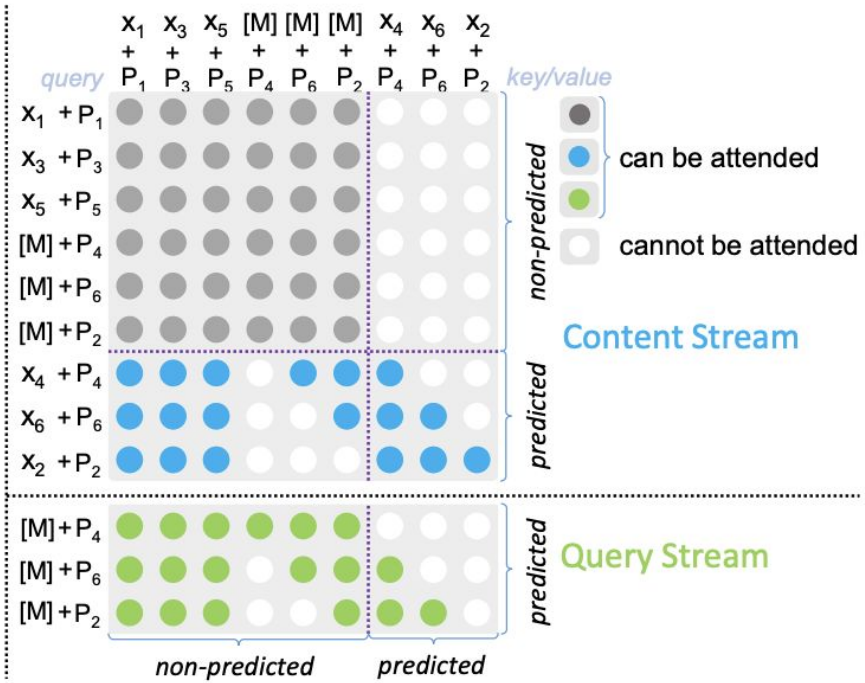



[MPNet: Masked and Permuted Pre-training for Language Understanding]

# MPNet

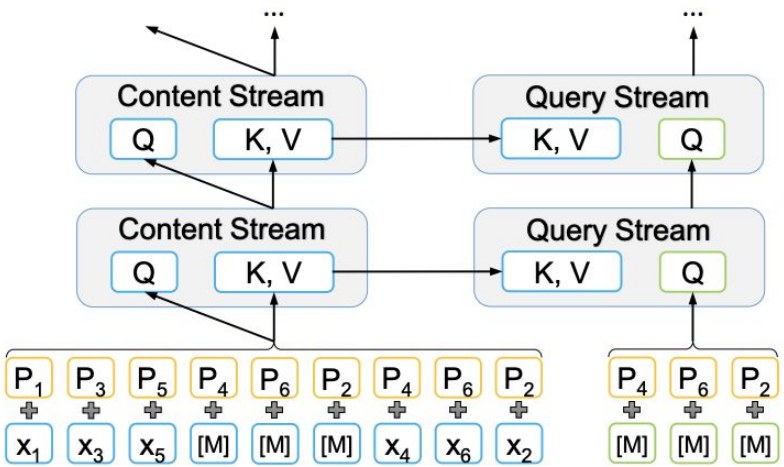

Transformer
x4 x6 x2
P1 P3 P5 P4 P6 P2 P4 P6 P2
x1 x3 x5 [M] [M] [M] x4 x6 x2

(a)


query
key/value
x1 + P1
x3 + P3
x5 + P5
[M] + P4
[M] + P6
[M] + P2
x4 + P4
x6 + P6
x2 + P2
non-predicted
predicted
can be attended
cannot be attended
Content Stream
Query Stream
non-predicted
predicted

(b)

# MPNet

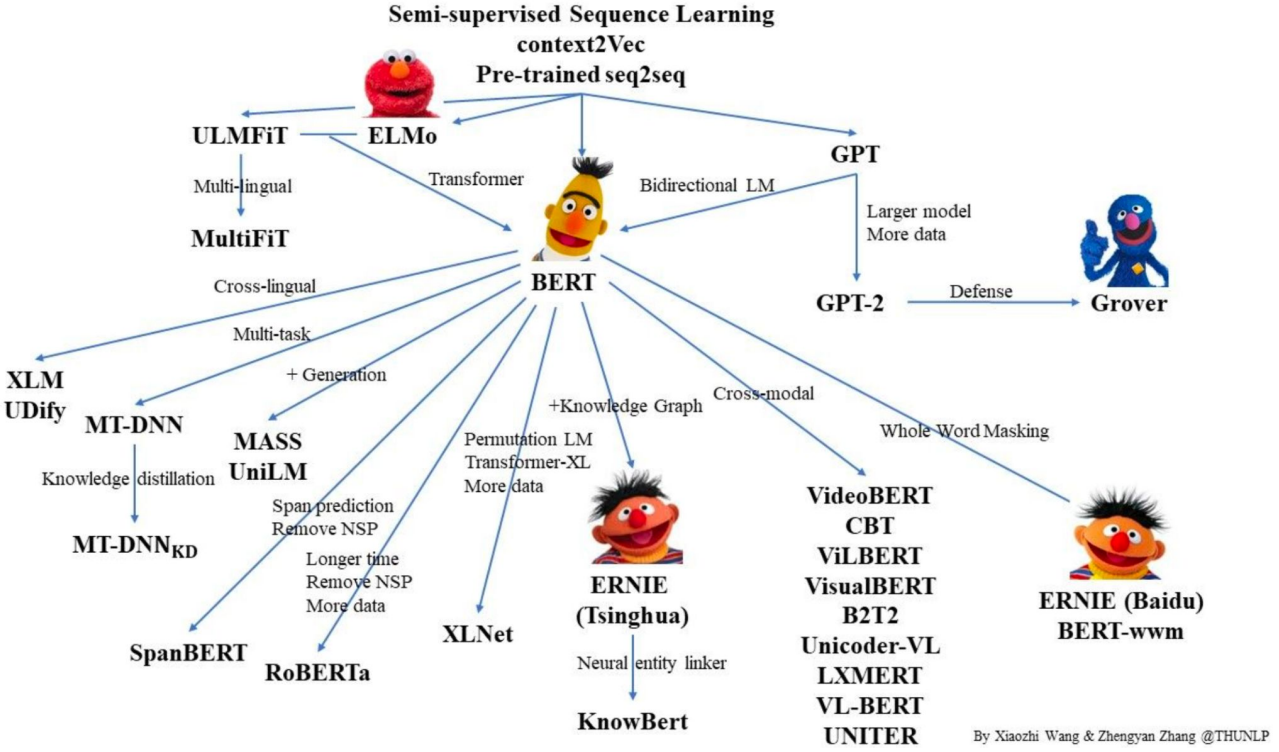



- ➔ two-stream self-attention mechanism, query stream reuses the hidden from content stream to calc. key and value
- ➔ leverages the dependency among the predicted tokens through PLM and makes the model to see auxiliary position information to reduce the discrepancy between pre-training and fine-tuning

| Model | Factorization |
|---|---|
| MLM | $\log P(\text{sentence} \mid \text{the task is [M] [M]}) + \log P(\text{classification} \mid \text{the task is [M] [M]})$ |
| PLM | $\log P(\text{sentence} \mid \text{the task is}) + \log P(\text{classification} \mid \text{the task is sentence})$ |
| MPNet | $\log P(\text{sentence} \mid \text{the task is [M] [M]}) + \log P(\text{classification} \mid \text{the task is sentence [M]})$ |

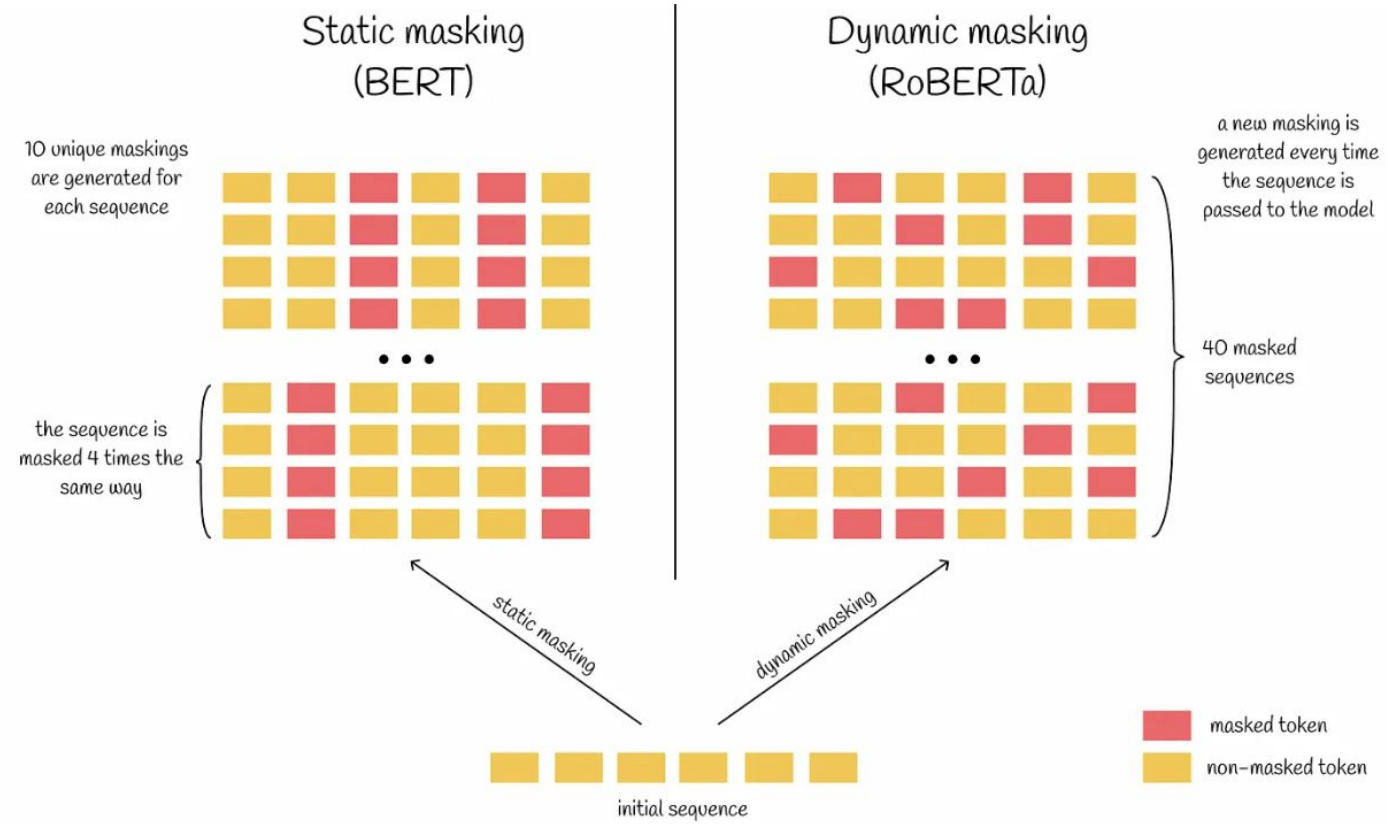
# BERTology

- ➔ Transformer-XL
- ➔ XLNet
- ➔ MPNet
- ➔ **RoBERTa**
- ➔ ALBERT
- ➔ DistilBERT
- ➔ BART
- ➔ BigBird
- ➔ ELECTRA
- ➔ DeBERTa



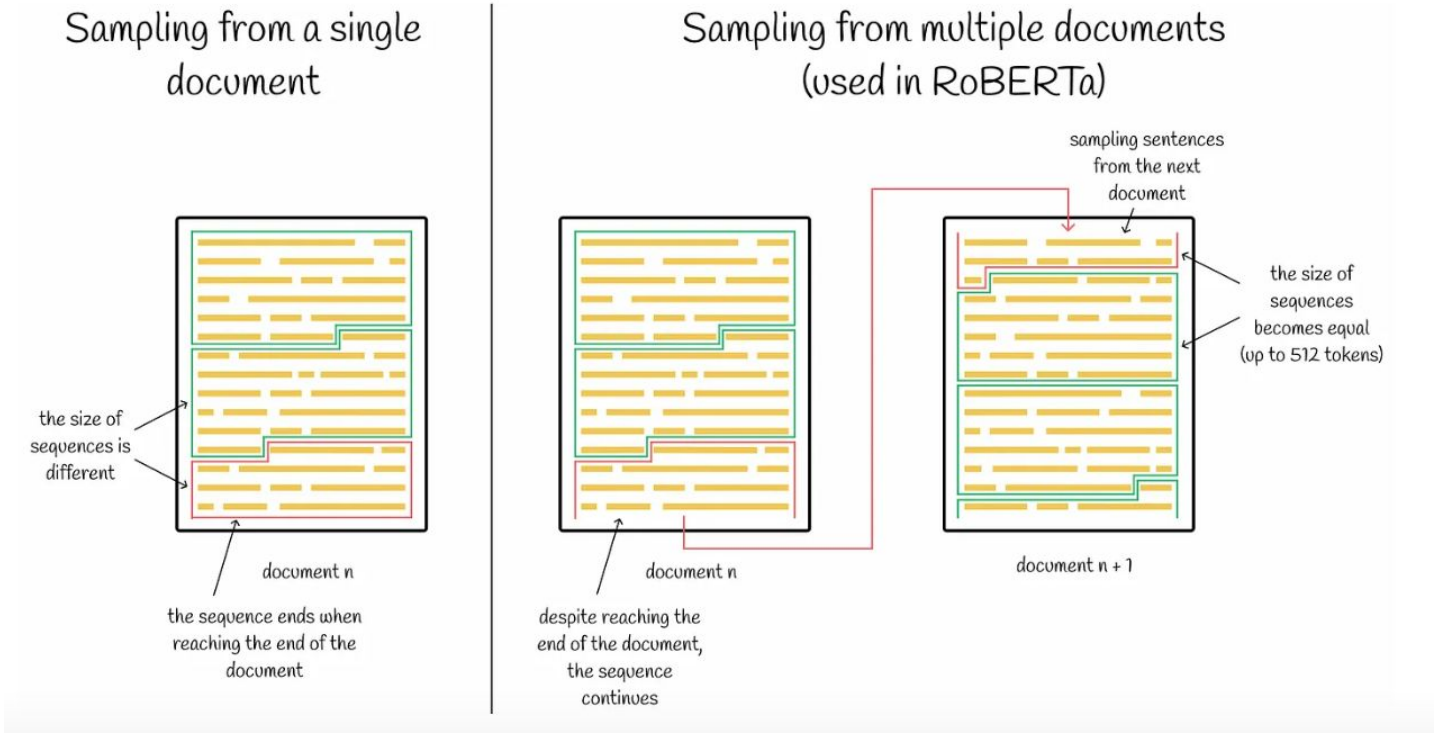

[BERTology paper]

# Dynamic masking

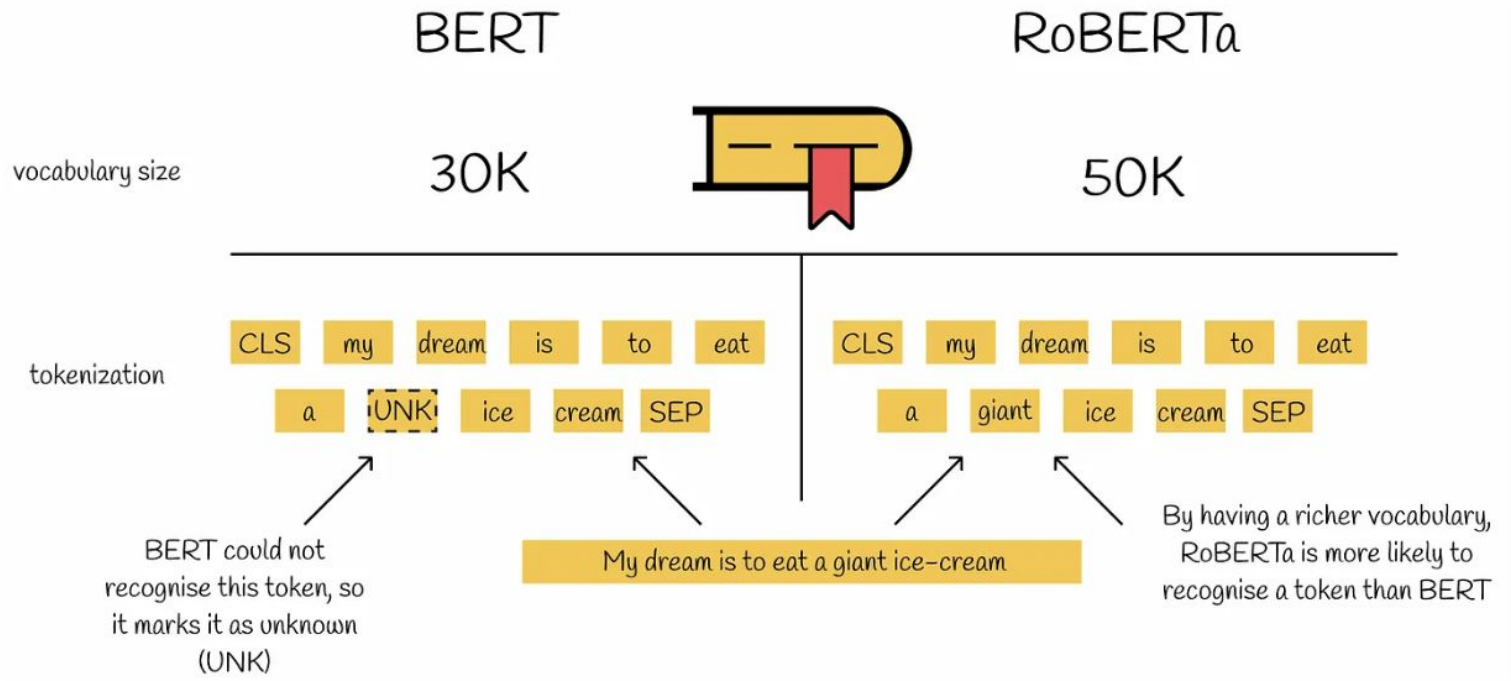



Static masking vs Dynamic masking

# Revisiting NSP problem

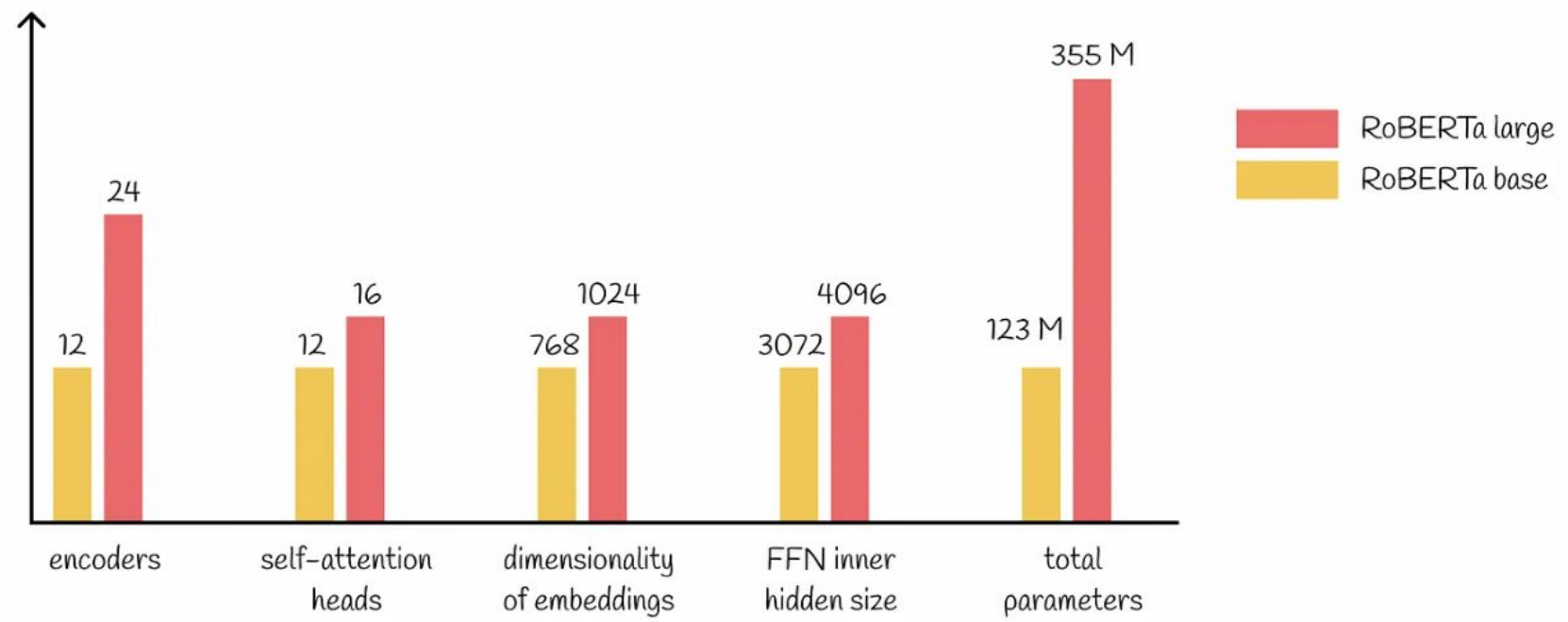


# New tokenizer



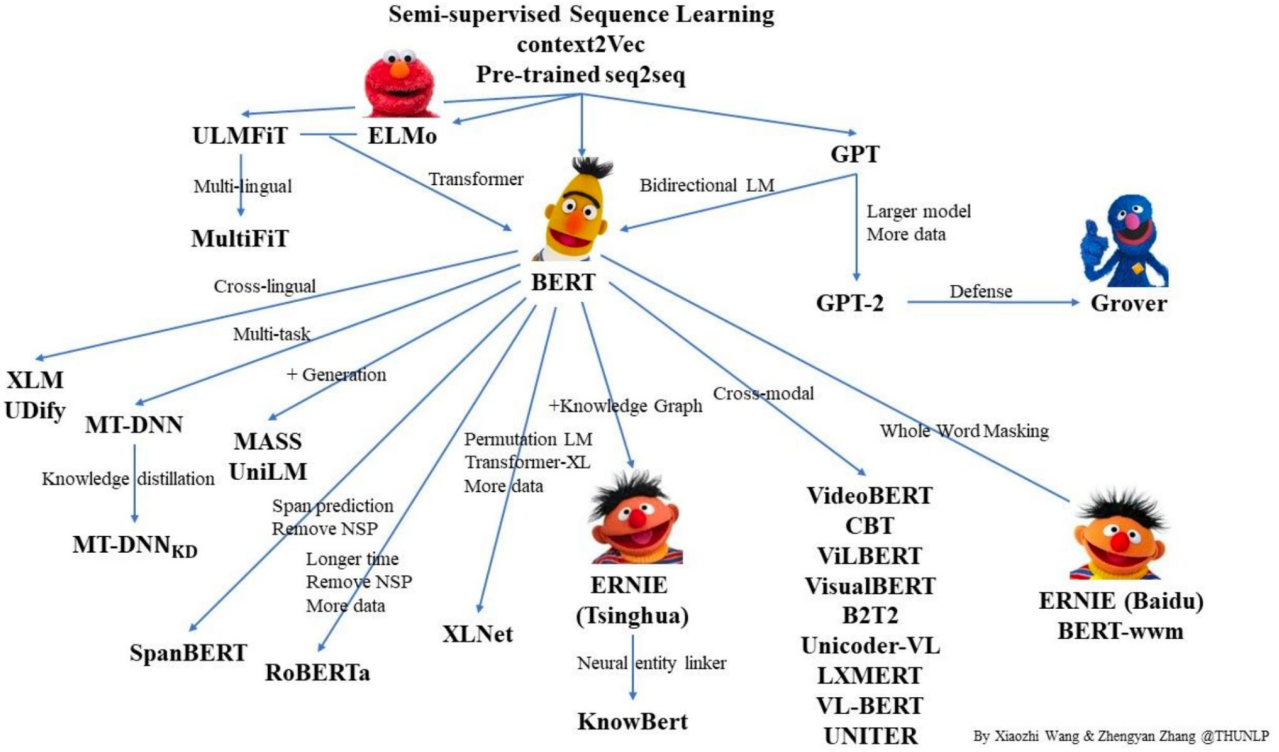
# New tokenizer

- Trained on 160GB of texts
- Batch size is 8000 (compared to 256 in BERT)



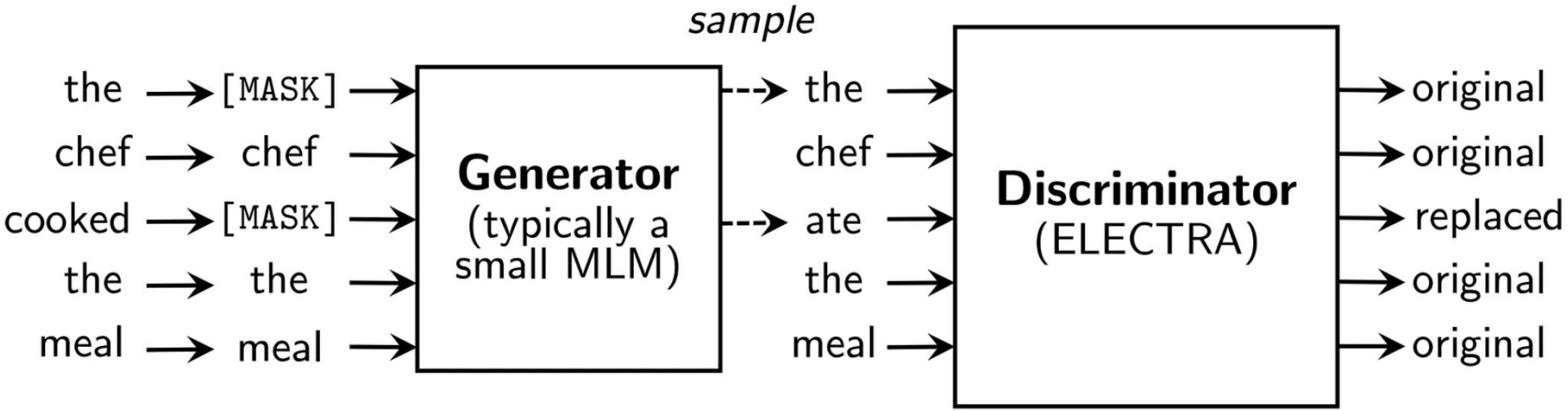
# BERTology

- ➔ Transformer-XL
- ➔ XLNet
- ➔ MPNet
- ➔ RoBERTa
- ➔ ALBERT
- ➔ DistilBERT
- ➔ BART
- ➔ BigBird
- ➔ **ELECTRA**
- ➔ DeBERTa



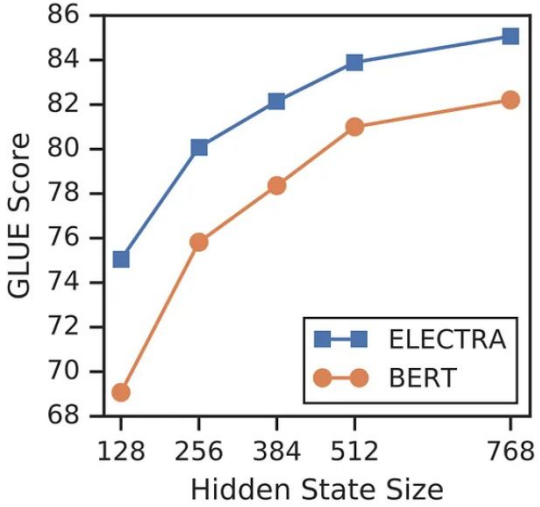

[BERTology paper]

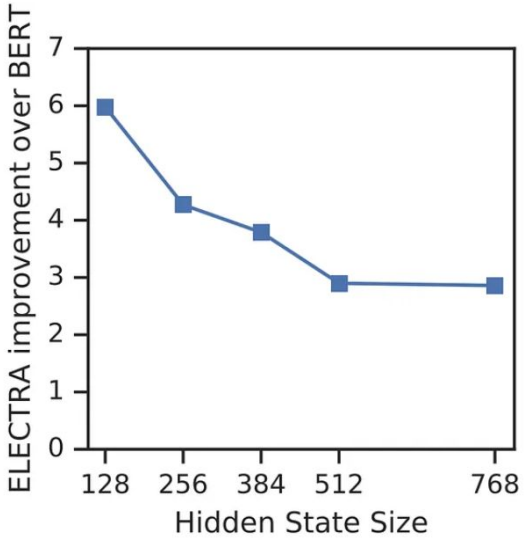
# ELECTRA

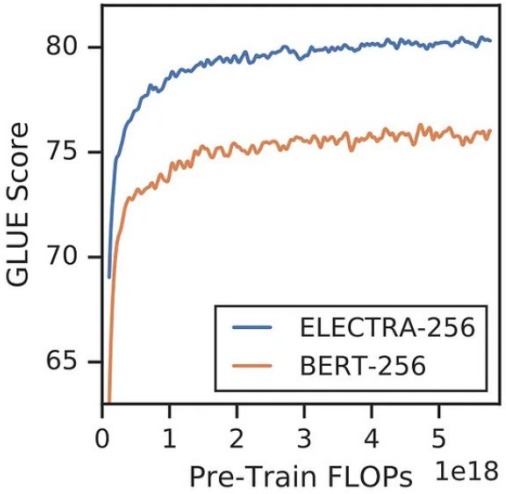



1. Randomly replace some tokens with [MASK]
2. Generator predicts initial tokens for all the masked ones
3. The discriminator input is built by replacing [MASK] with generator predictions
4. For each token in the sequence discriminator predicts whether it is original or was generated

# ELECTRA


GLUE Score
86
84
82
80
78
76
74
72
70
68
128
256
384
512
768
Hidden State Size
ELECTRA
BERT


ELECTRA improvement over BERT
7
6
5
4
3
2
1
0
128
256
384
512
768
Hidden State Size

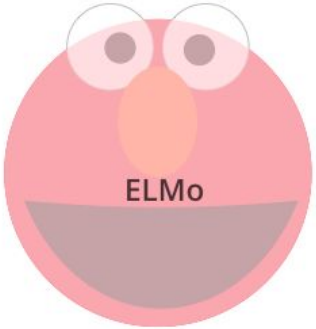

GLUE Score
80
75
70
65
0
1
2
3
4
5
Pre-Train FLOPs
1e18
ELECTRA-256
BERT-256


THE
TRANSFORMER


ULM-FiT


BERT


OpenAI
Transformer


ELMo