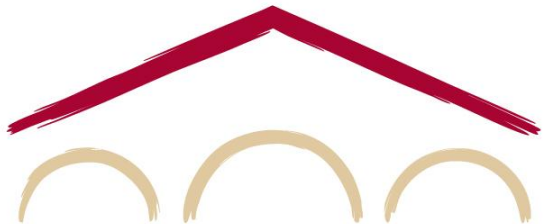
Natural Language Processing with Deep Learning

CS224N/Ling284

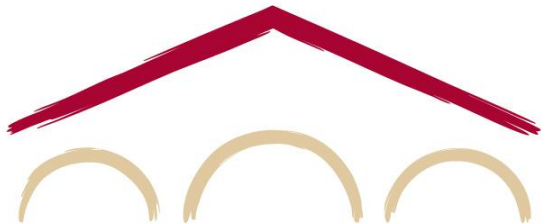


Guest Lecture: **Julie Kallini**

Lecture 14: Tokenization and Multilinguality

自然语言处理与深度学习

CS224N/Ling284



Guest Lecture: **Julie Kallini**

第　14　讲：分词与多语言性

Announcements

Today is the first guest lecture. We will be taking attendance for every guest lecture, and **attendance is required for on-campus students.**

- If you are an online student, submit a reaction paragraph on Gradescope (due one week from today).

Project Proposal feedback is out on Gradescope, and instructions for the **Project Milestone** are on the website now.

- The milestone is now due on Thursday, February 26.

Assignment 4 is due today! If you have late days, still don't leave this assignment to the last minute, since it requires querying APIs.

公告

今天是第一次嘉宾讲座。 We will be taking attendance for every guest lecture, and 校内学生必须参加。

- If you are an online student, submit a reaction paragraph on Gradescope (due one 从今天起一周）。

项目提案 feedback is out on Gradescope, and instructions for the **Project** 里程碑 are on the website now.

- 里程碑截止日期改为 2 月 26 日周四。

作业 4 今天截止！ If you have late days, still don’t leave this assignment to the 最后一刻，因为它需要查询 API。

Agenda for Today

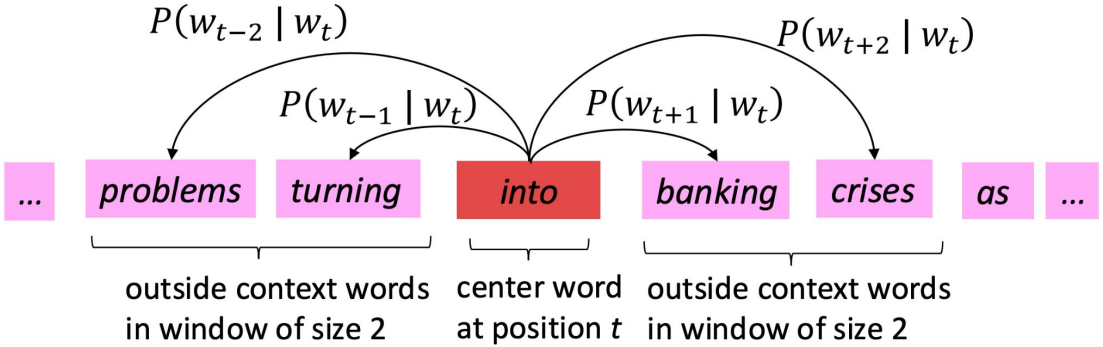
1. **Tokenization approaches:** Word, character/byte, and subword tokenization
2. **Byte Pair Encoding (BPE):** Training algorithm and walkthrough
3. **Case studies:** Spelling, glitch tokens, and where tokenization breaks down
4. **Multilinguality:** Languages of the world, cross-lingual transfer, and fairness
5. **Multilingual tokenization:** Why it's important and what the challenges are

今日议程

1. 分词方法：Word, character/byte, and 子词分词
2. 字节对编码（BPE）：Training algorithm and 详解
3. 案例研究：Spelling, glitch tokens, and where tokenization 失效
4. 多语言性：Languages of the world, cross-lingual 迁移和公平性
5. 多语言分词：Why it’s important and what the 挑战是

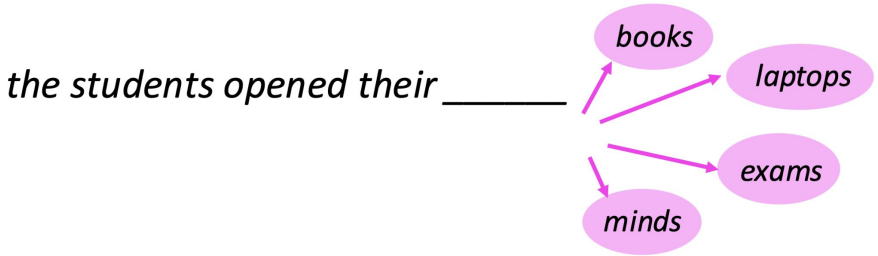
Throwback Time

In **Lecture 2**, you learned about the skip-gram ***word2vec*** objective, where a model is trained to predict context words given a center word:



How do we split text into words?

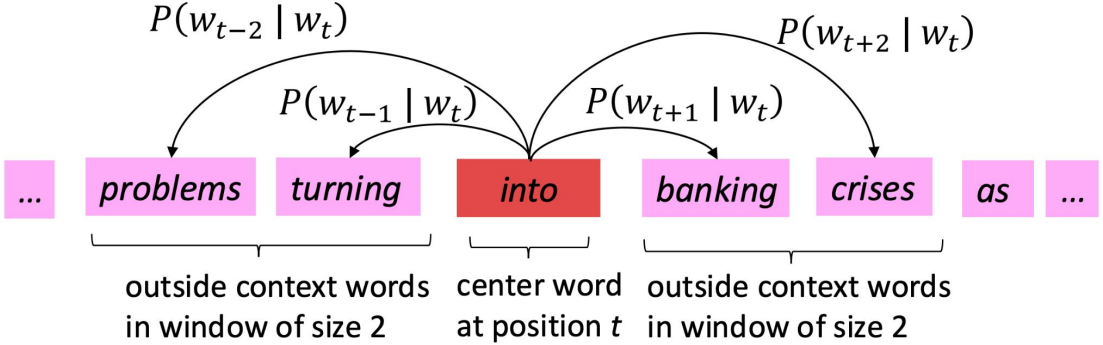
In **Lecture 4**, you were introduced to ***language modeling***, the task of predicting what word comes next:



How do we decide the vocab of a language model?

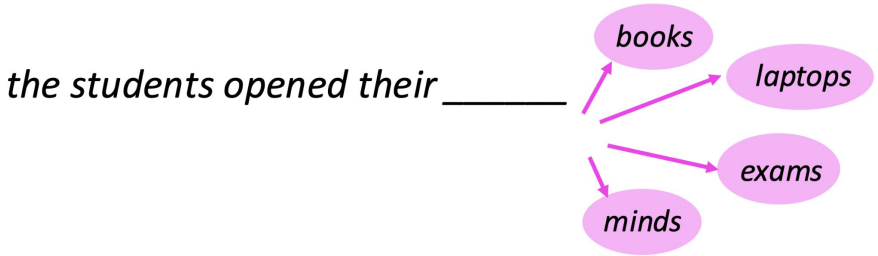
回顾时间

In 第 2 讲 you learned about the skip-gram ***word2vec*** objective, where a model is 训练以在给定中心词的情况下预测上下文词：



How do we split text into 词？

In 第 4 讲 you were introduced to 语言建模 , the task of predicting what word 接下来：

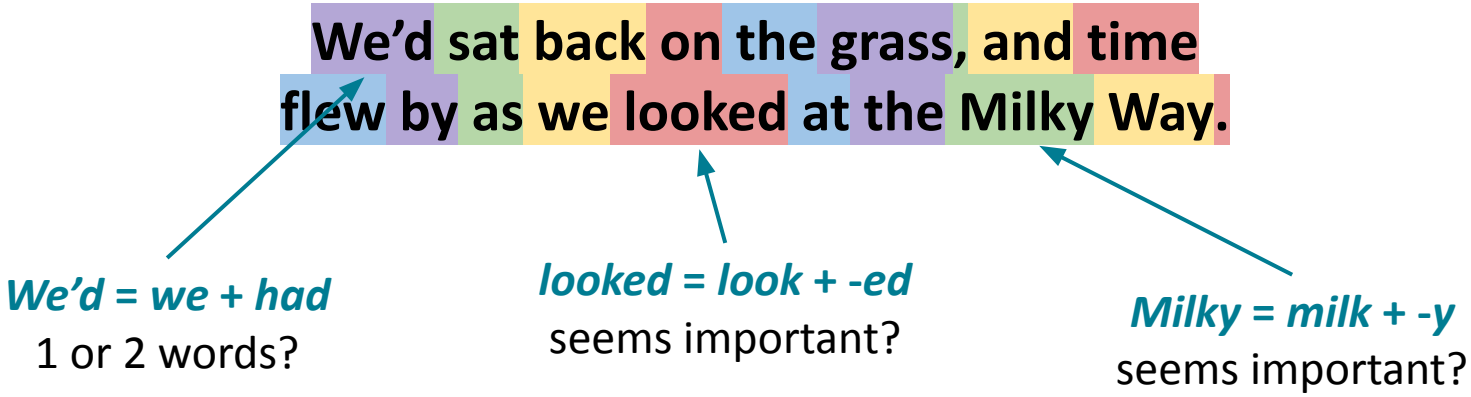


How do we decide the vocab of a language 模型？

What is a word?

How many words are in this sentence?

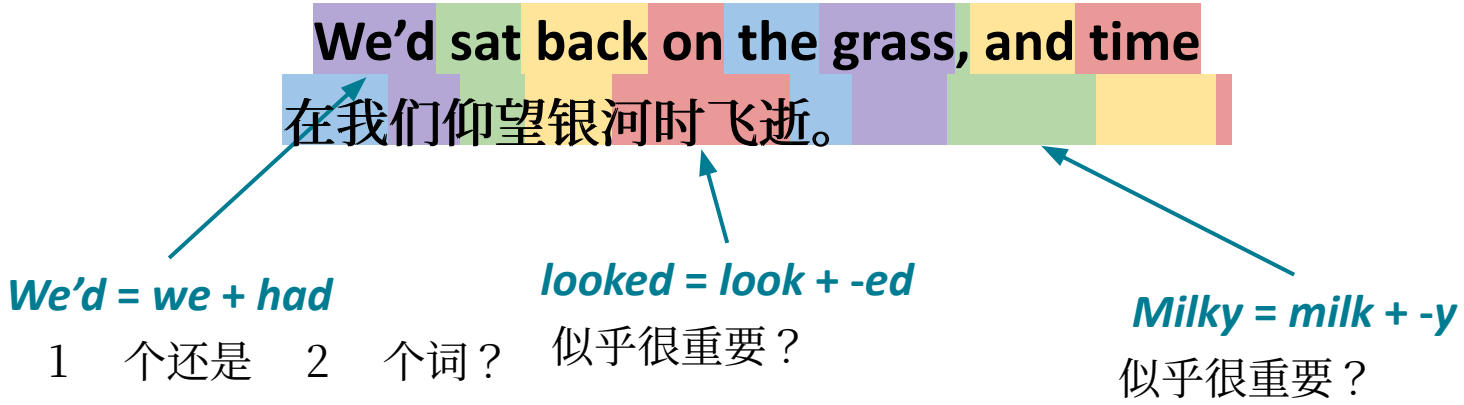
17 whitespace-separated words + 2 punctuation marks



什么是词？

这个句子中有多少个词？

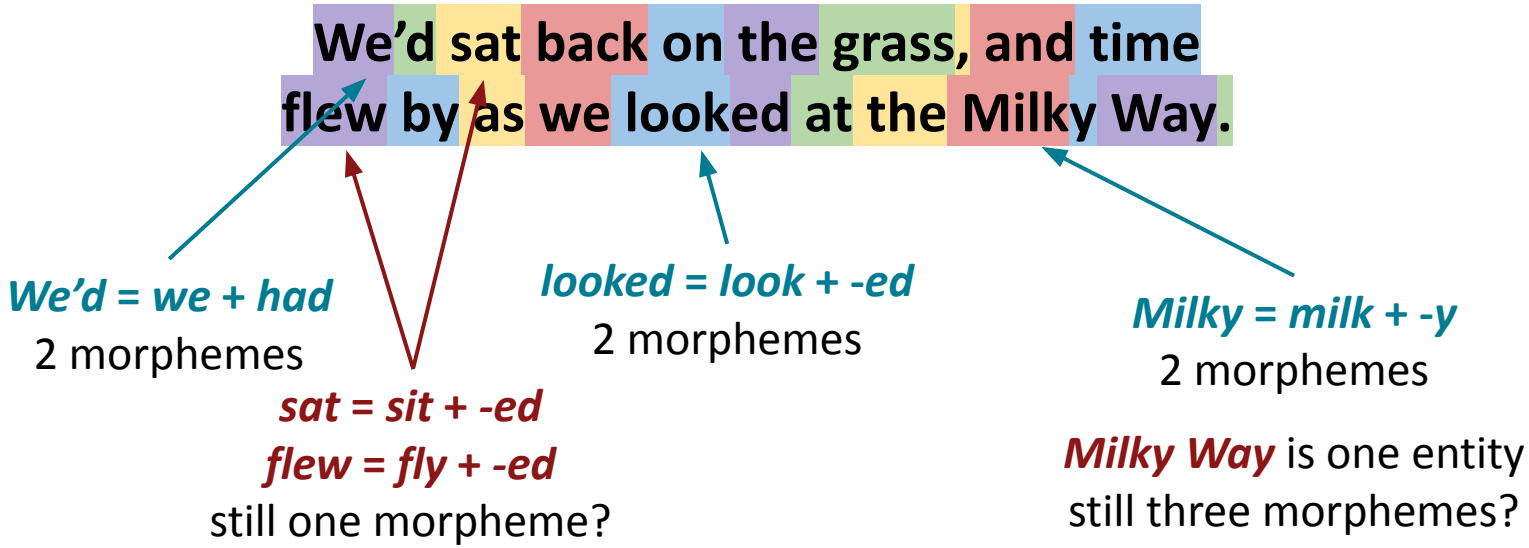
17 个空格分隔的词 + 2 个标点符号



What is a ~~word~~ *morpheme*?

A ***morpheme*** is a minimal unit of language that has meaning or a grammatical function.

How many morphemes are in this sentence?



What is a ~~word~~ 语素 ?

A 语素 is a minimal unit of language that has meaning or a grammatical function.

这个句子中有多少个语素？

We'd = we + had

2 个语素

sat = sit + -ed

flew = fly + -ed

仍然是一个语素？

looked = look + -ed

2 个语素

Milky = milk + -y

2 个语素

Milky Way 是一个实体

仍然是三个语素？

Beyond words or morphemes

There are many useful ways to think about segmenting text:

Characters:	**We looked at the Milky Way.**
Morphemes:	**We looked at the Milky Way.**
Words:	**We looked at the Milky Way.**
Phrases:	**We looked at the Milky Way.**

What would be best for a language model?

超越词或语素

有许多有用的方式来思考文本分割：

字符：

语素：

词：

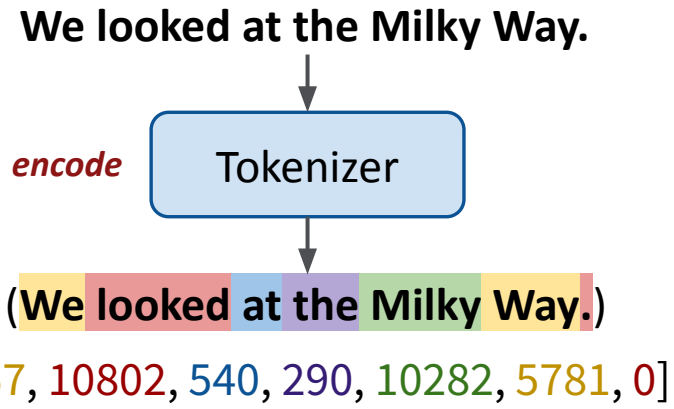
Phrases:

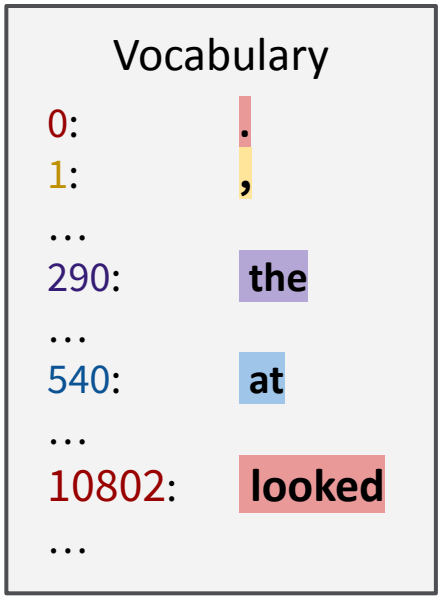
什么对语言模型最好？

How do language models see text?

For a language model:

- The fundamental units of language are ***tokens***
- ***Tokenization*** is the process of breaking text into tokens
- A ***tokenizer*** translates between text and a sequence of tokens that a language model (LM) learns over
- The ***vocabulary*** V is the set of known tokens

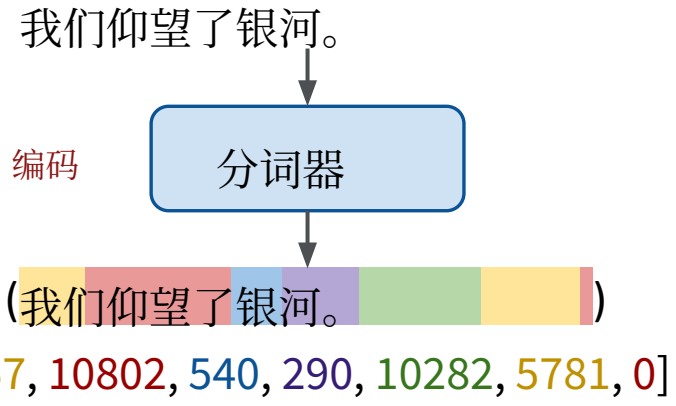


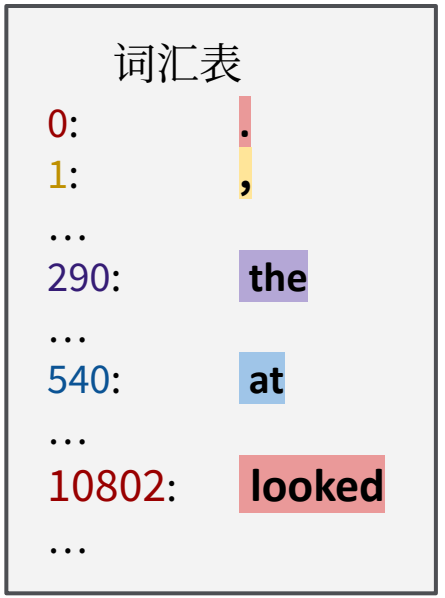


语言模型如何看待文本？

对于语言模型：

- The fundamental units of language are token
- 分词 is the process of breaking text into tokens
- A 分词器 translates between text and a sequence of tokens that a language model（LM）学习
- The 词汇表 V is the set of known tokens





How do language models see text?

For a language model:

- The fundamental units of language are ***tokens***
- ***Tokenization*** is the process of breaking text into tokens
- A ***tokenizer*** translates between text and a sequence of tokens that a language model (LM) learns over
- The ***vocabulary*** V is the set of known tokens

Fundamentally, why do we need tokenization?

- Models can't just read raw text → we need a mapping from strings to sequences of embeddings

语言模型如何看待文本？

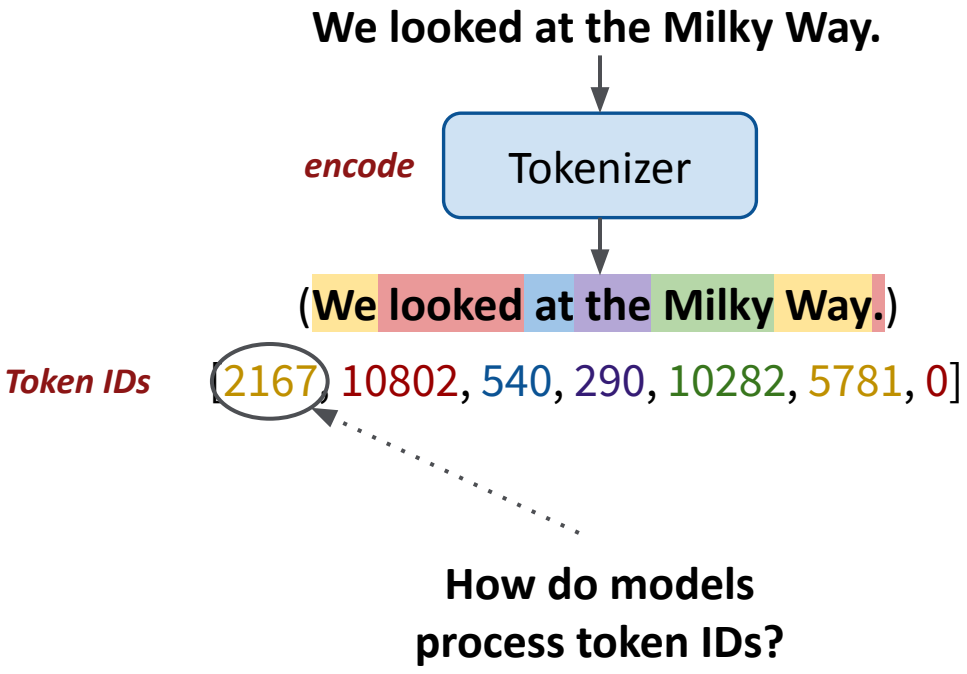
对于语言模型：

- The fundamental units of language are token
- 分词 is the process of breaking text into tokens
- A 分词器 translates between text and a sequence of tokens that a language model（LM） 学习
- The 词汇表 V is the set of known tokens

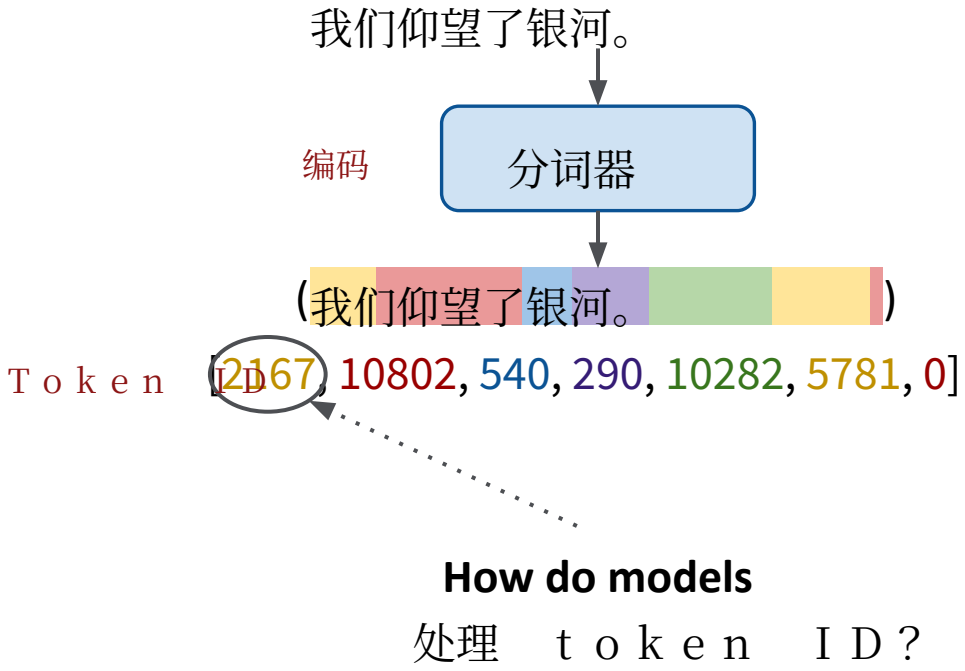
从根本上说，为什么我们需要分词？

- Models can't just read raw text → we need a mapping from strings to sequences of 嵌入

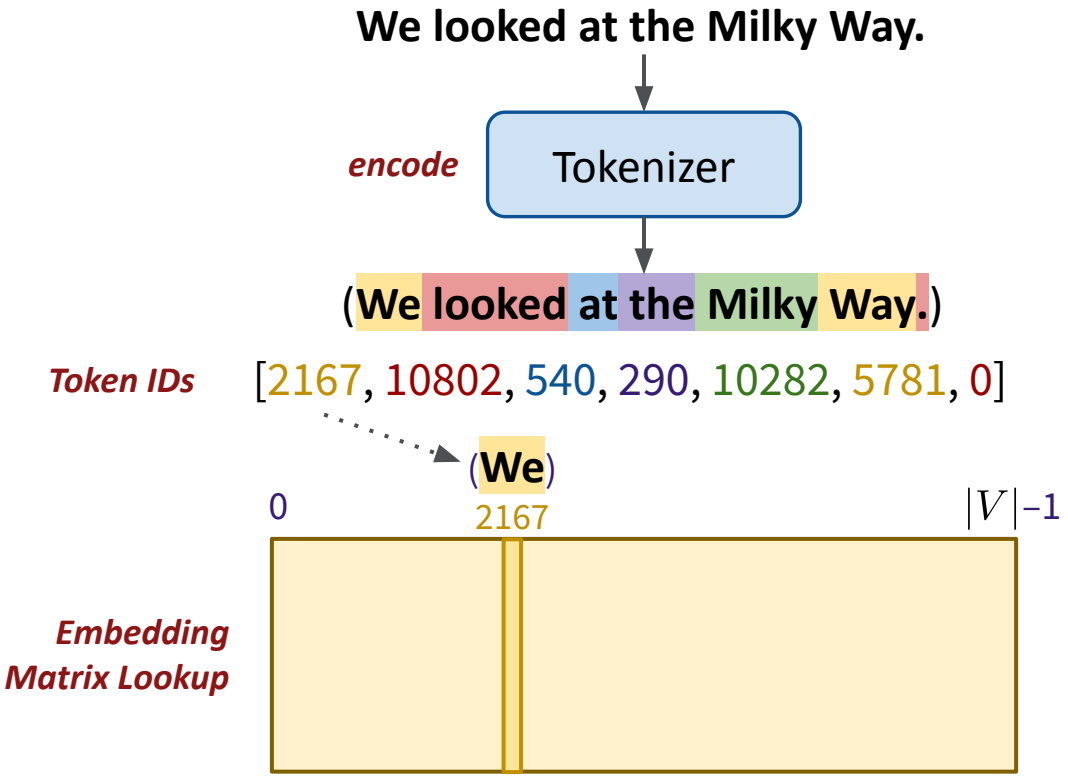
How do language models see text?



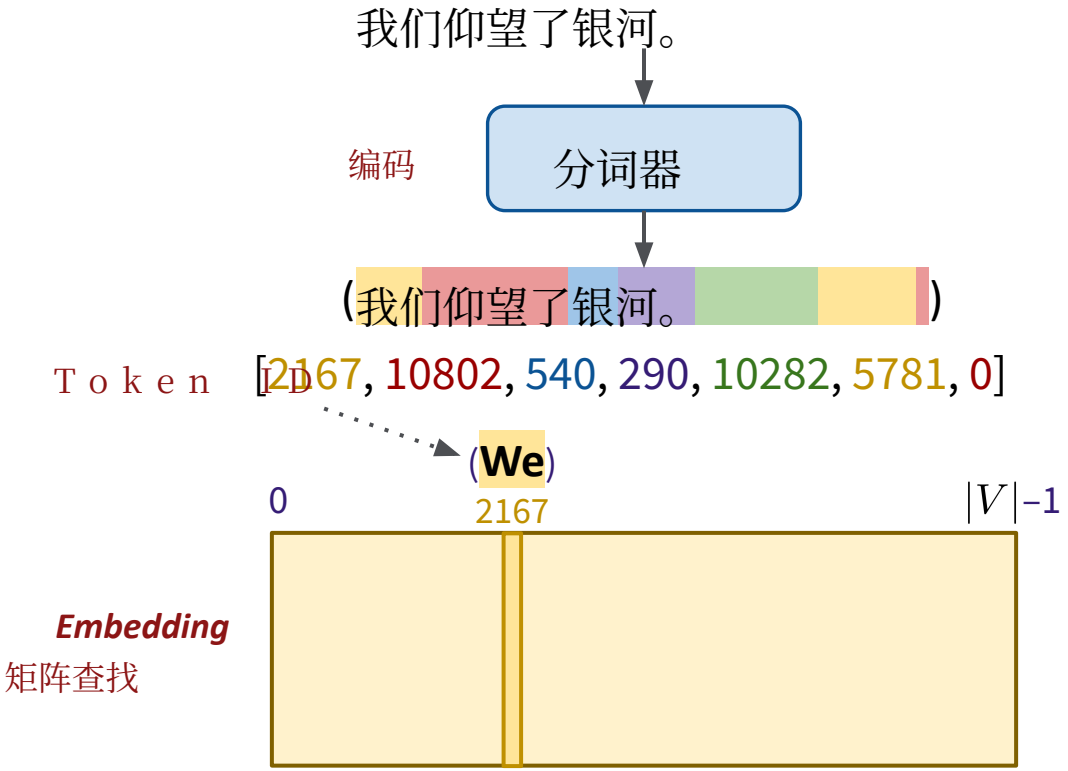
语言模型如何看待文本？


我们仰望了银河。
编码
分词器
(我们仰望了银河。)
Token ID
[2167, 10802, 540, 290, 10282, 5781, 0]
How do models
处理 token ID？

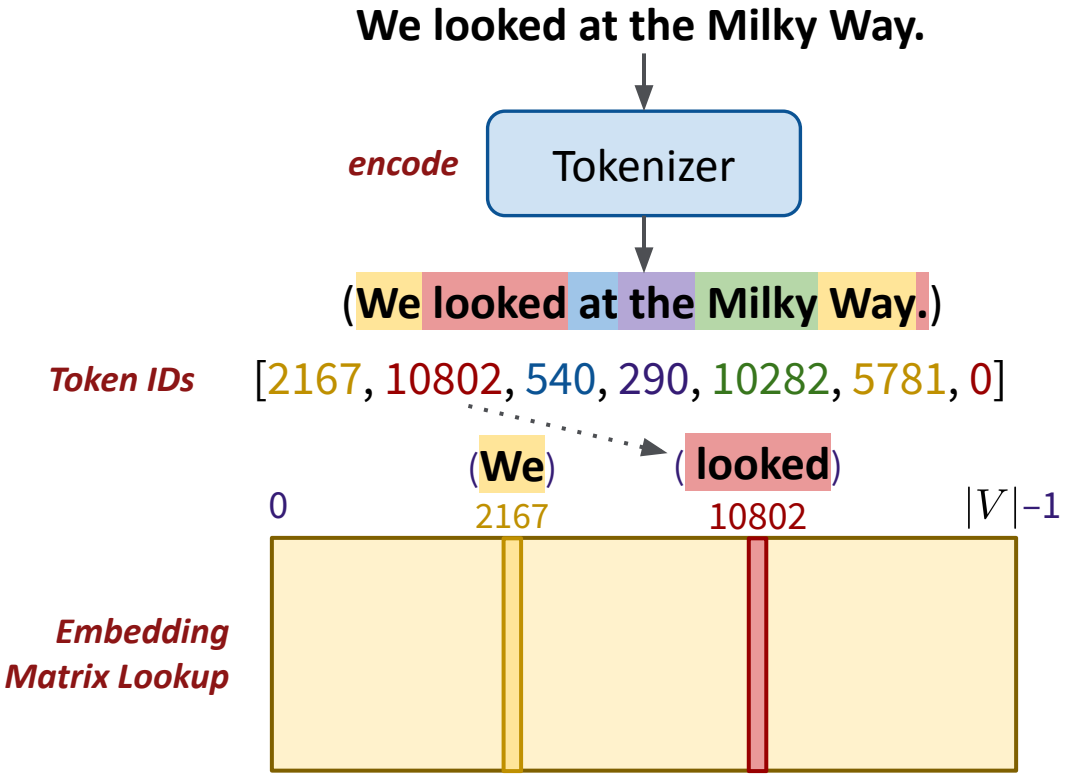
How do language models see text?



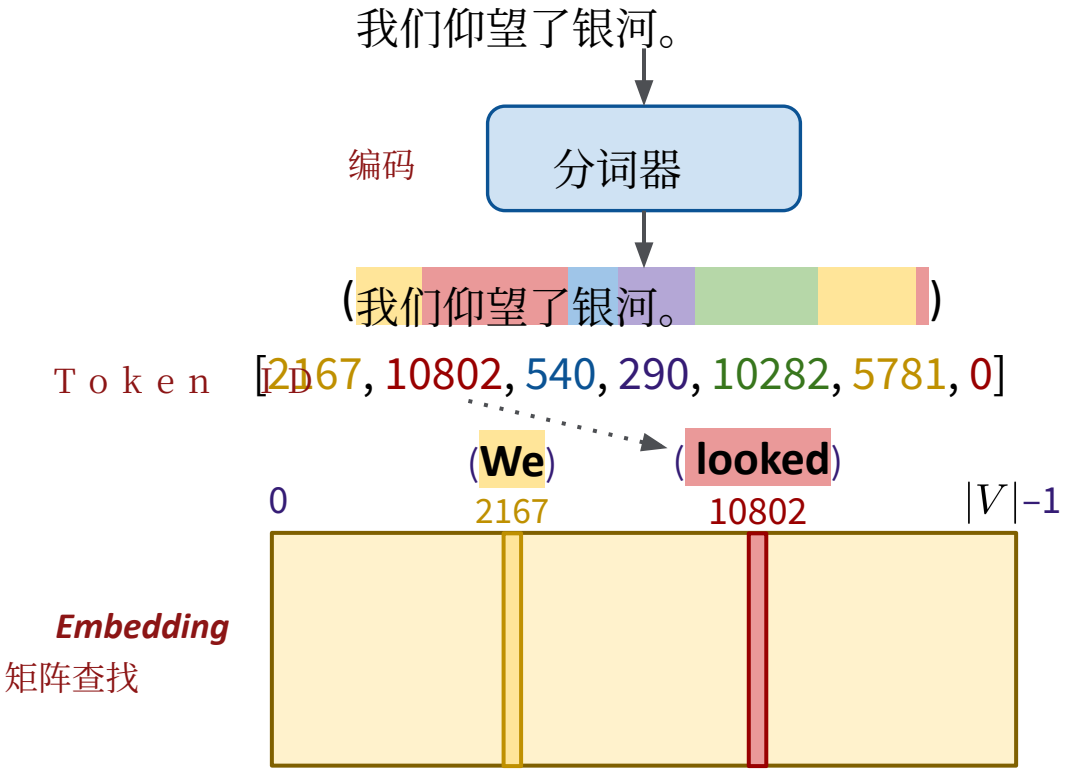
语言模型如何看待文本？


我们仰望了银河。
编码
分词器
(我们仰望了银河。)
Token ID
[2167, 10802, 540, 290, 10282, 5781, 0]
(We)
0
2167
|V|-1
Embedding
矩阵查找

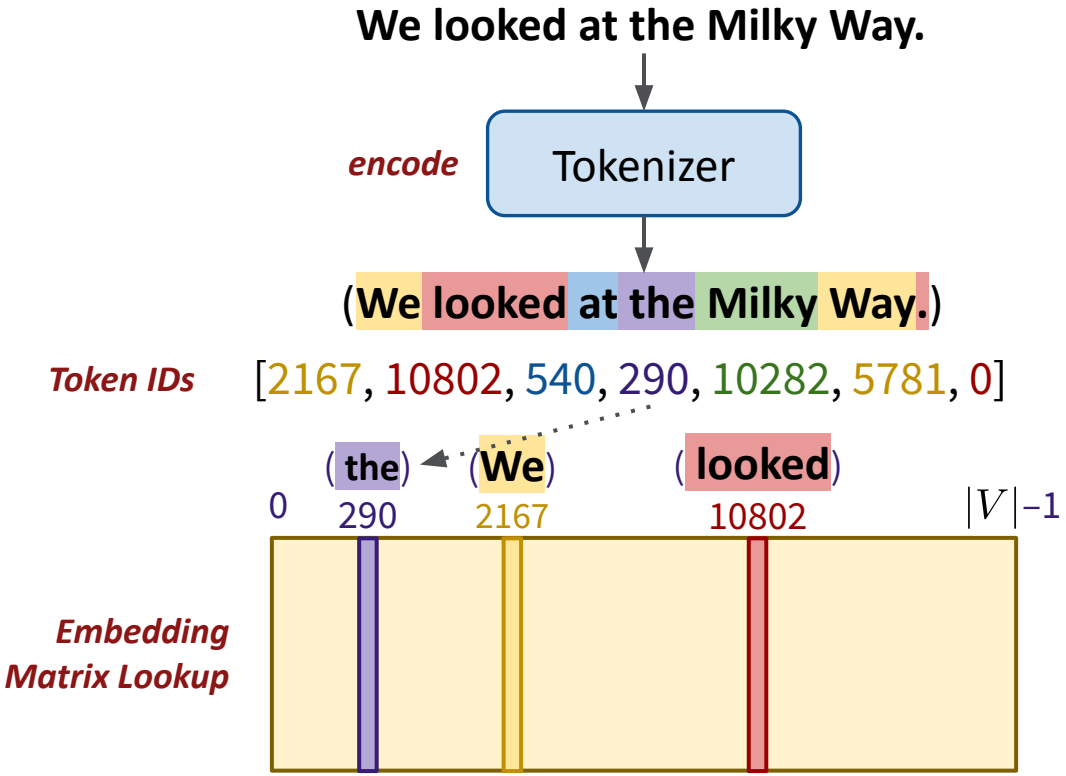
How do language models see text?



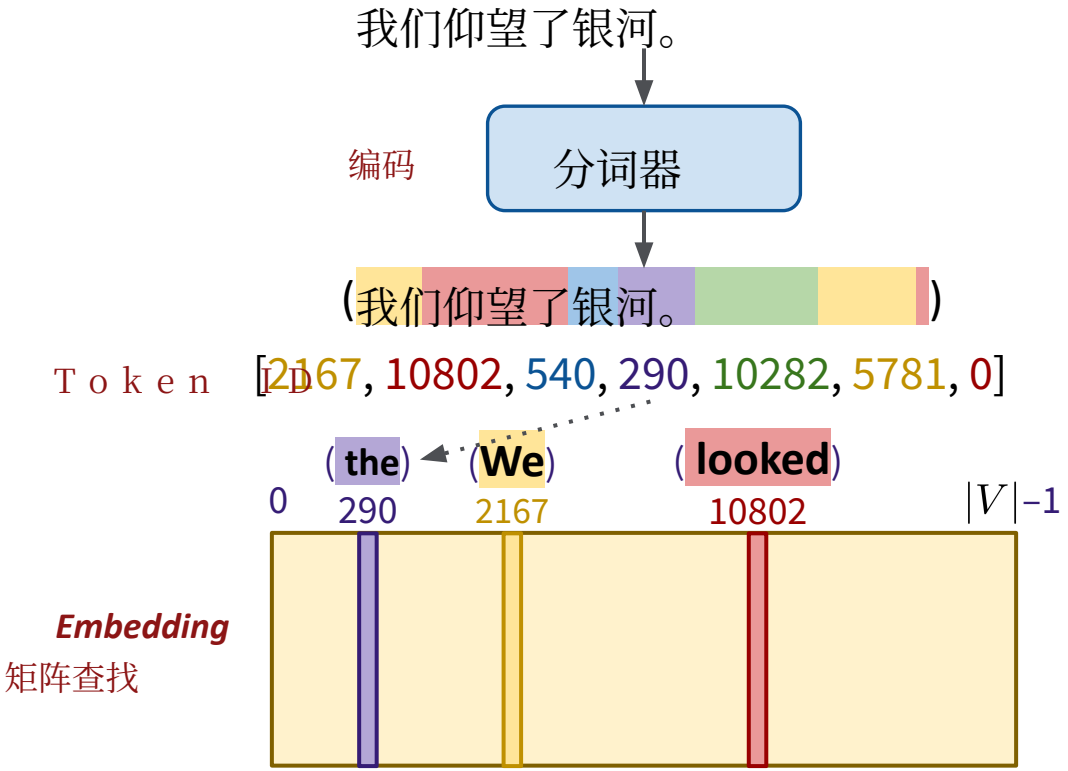
语言模型如何看待文本？


我们仰望了银河。
编码
分词器
(我们仰望了银河。)
Token ID
[2167, 10802, 540, 290, 10282, 5781, 0]
(We)
(looked)
0
2167
10802
|V|-1
Embedding
矩阵查找

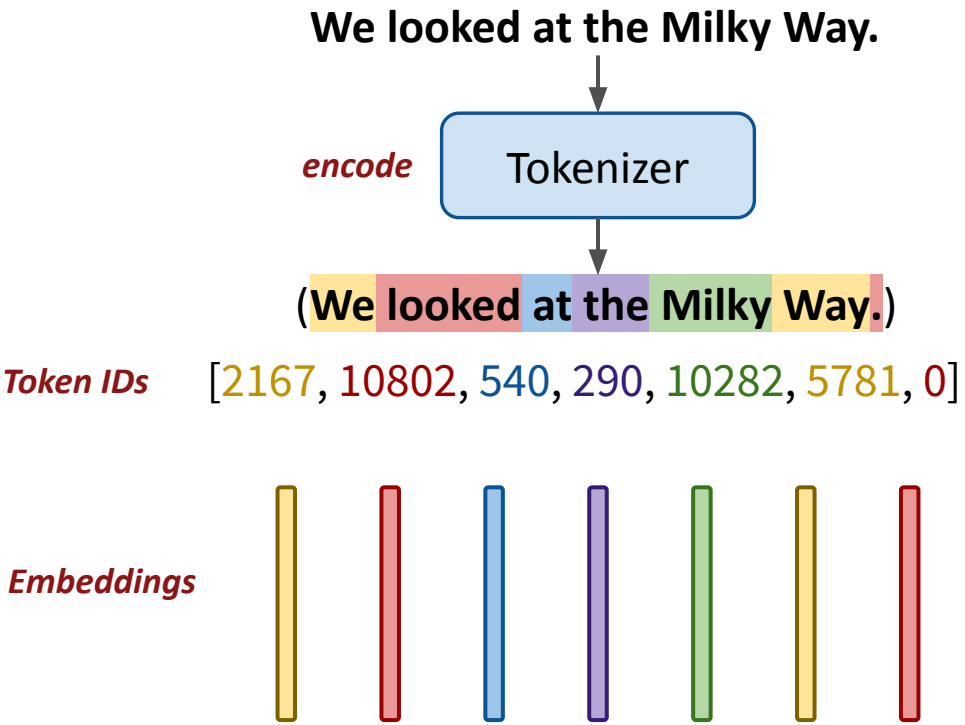
How do language models see text?



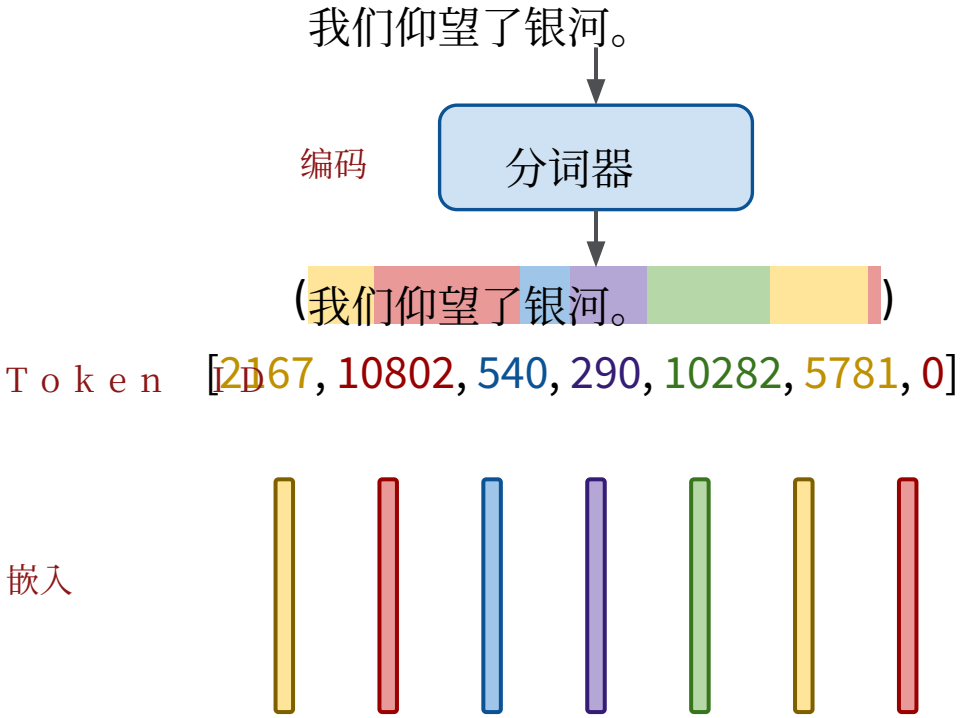
语言模型如何看待文本？


我们仰望了银河。
编码
分词器
(我们仰望了银河。)
Token ID
[2167, 10802, 540, 290, 10282, 5781, 0]
(the)
(We)
(looked)
0
290
2167
10802
|V|-1
Embedding
矩阵查找

How do language models see text?

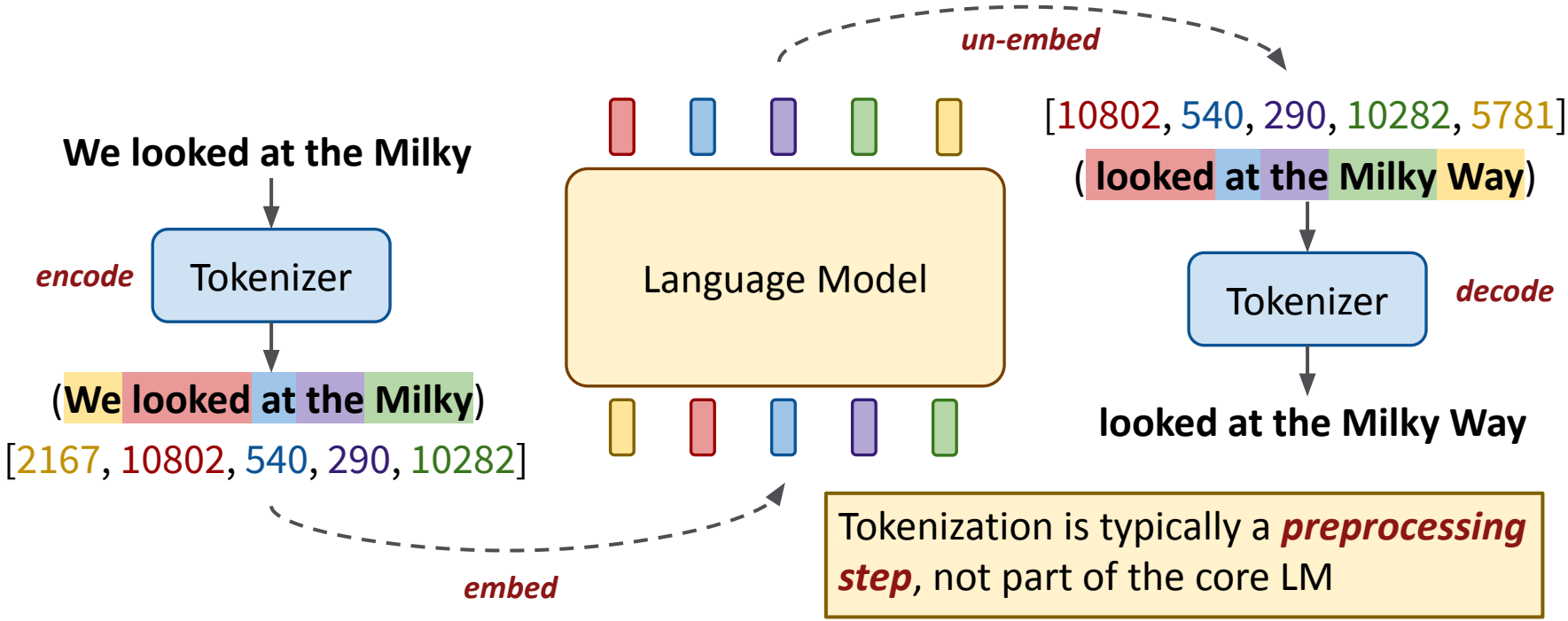


语言模型如何看待文本？


我们仰望了银河。
编码
分词器
(我们仰望了银河。)
Token ID
[2167, 10802, 540, 290, 10282, 5781, 0]
嵌入

How do language models see text?

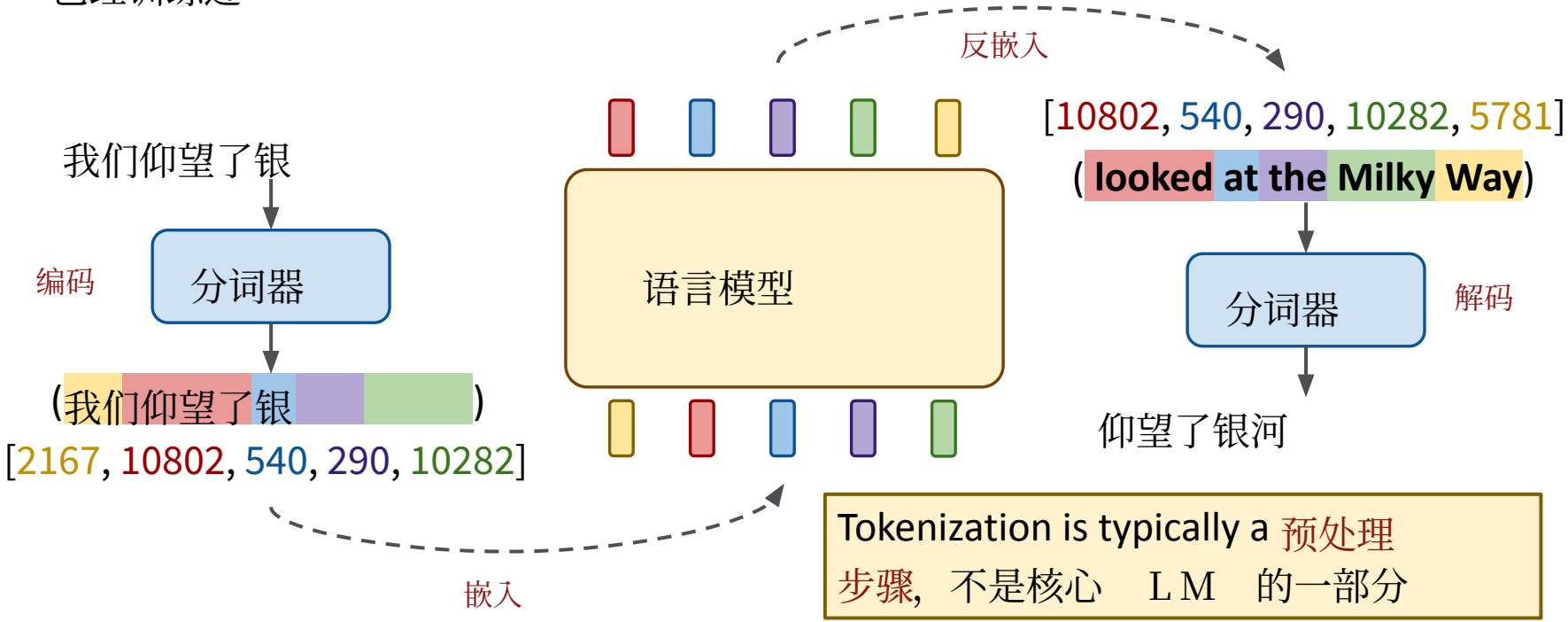
Tokenization influences how LMs learn from and process text, before they have even been trained



Tokenization is typically a ***preprocessing step***, not part of the core LM

语言模型如何看待文本？

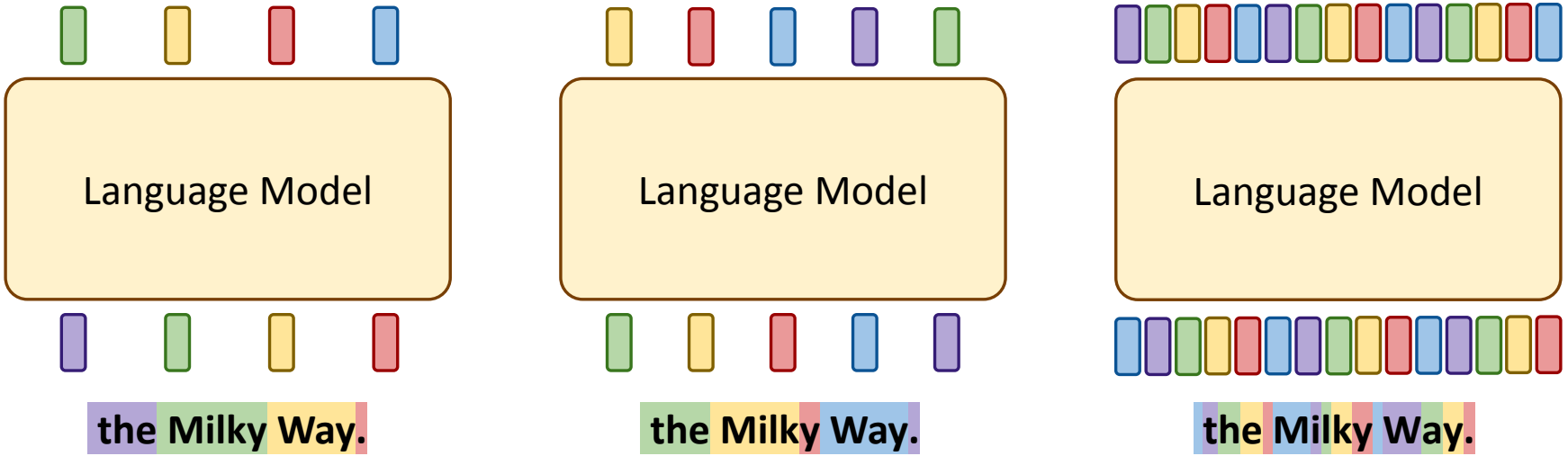
Tokenization influences how LMs learn from and process text, before they have even
已经训练过



Tokenization is typically a 预处理步骤，不是核心 LM 的一部分

How do language models see text?

Tokenization influences the efficiency of LMs, both in terms of **sequence lengths** and **parameter counts**

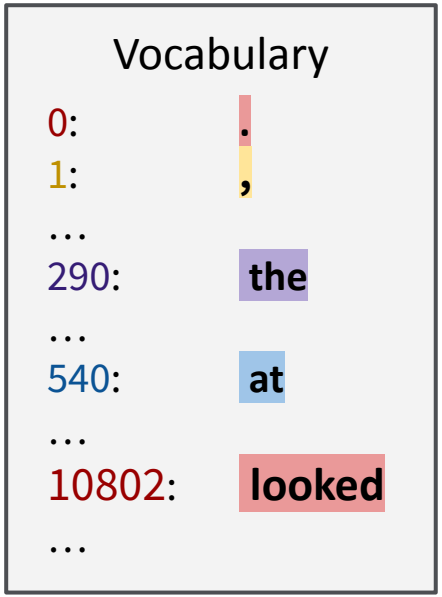


语言模型如何看待文本？

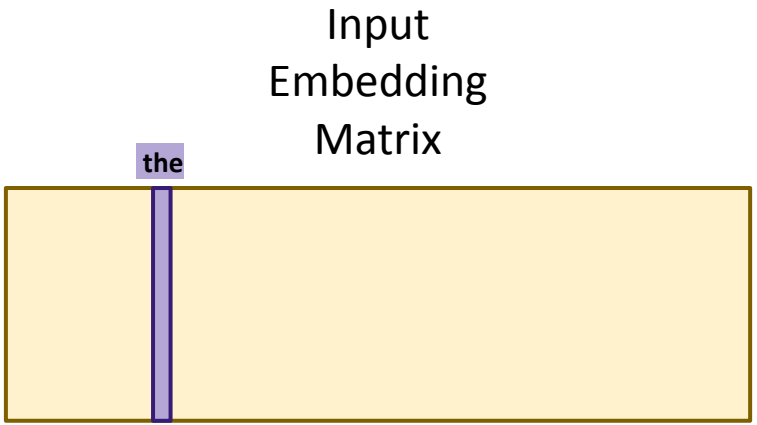
Tokenization influences the efficiency of LMs, both in terms of 序列长度 and 参数量

How do language models see text?

Tokenization influences the efficiency of LMs, both in terms of **sequence lengths** and **parameter counts**



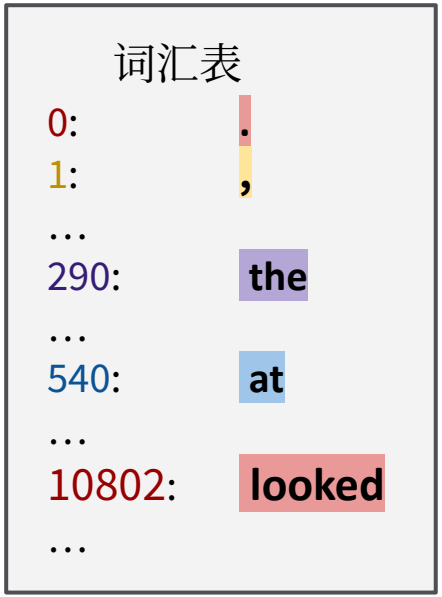
$$|V| = 32,000$$



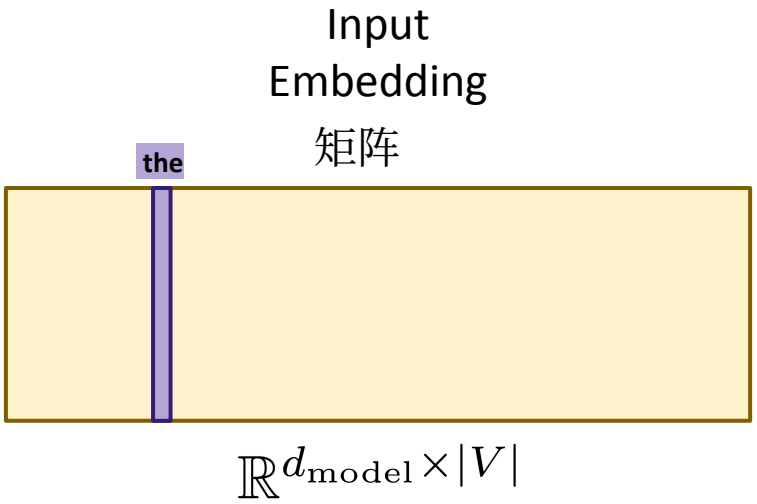
$$\mathbb{R}^{d_{\text{model}} \times |V|}$$

语言模型如何看待文本？

Tokenization influences the efficiency of LMs, both in terms of 序列长度 and 参数量



$|V| = 32,000$



$\mathbb{R}^{d_{\text{model}} \times |V|}$

Word tokenization

As a first tokenization scheme,
let's consider simple ***word tokenization***

We'd sat back on the grass, and time flew by as we looked at the Milky Way.

We'll assume:

- **Tokens** = whitespace-delimited words + punctuation
- Vocabulary truncated to **top** $|V|$ **frequent words** from a corpus

✅ **Pros:**

- Semantically meaningful language units
- One embedding ≈ one interpretable meaning

词分词

作为第一种分词方案,
let's consider simple 词分词

We'd sat back on the grass, and time
在我们仰望银河时飞逝。

We'll assume:

- Tokens = whitespace-delimited words + punctuation
- Vocabulary truncated to 最高 V 个频繁词 from a corpus

✅ **Pros:**

- 语义上有意义的语言单位
- One embedding ≈ one interpretable meaning

Word tokenization

As a first tokenization scheme,
let's consider simple ***word tokenization***

We'd sat back on the grass, and time flew by as we looked at the Milky Way.

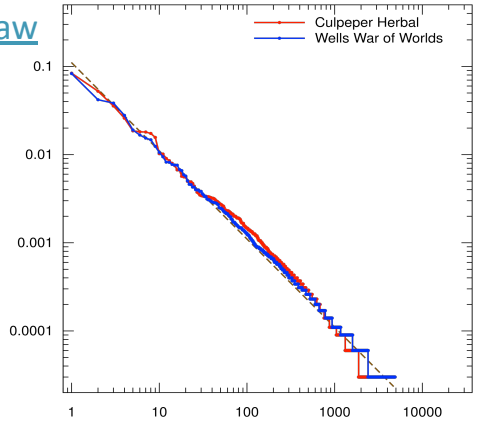
❌ **Cons:**

- There are too many words! Vocab can be huge
- Long tail of infrequent words
  - ***Zipf's law***: a word's frequency is inversely proportional to its rank by frequency
- Language is changing all the time!
  - Merriam Webster's Collegiate Dictionary added 5k new words in 12th edition (including *rizz*)

Corpus	Types = $\|V\|$
Shakespeare	31 thousand
Brown corpus	38 thousand
Switchboard telephone conversations	20 thousand
COCA	2 million
Google n-grams	13 million

Jurafsky & Martin Ch.2

Zipf's Law



词分词

作为第一种分词方案,
let's consider simple 词分词

We'd sat back on the grass, and time
在我们仰望银河时飞逝。

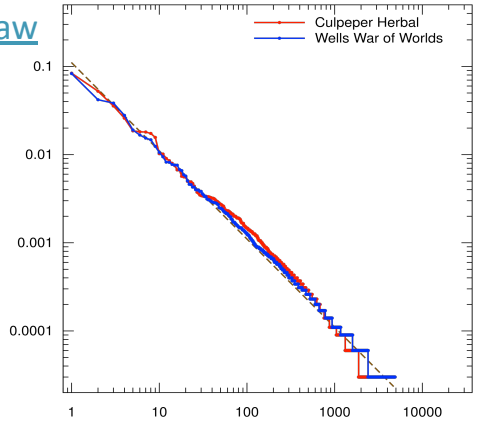
❌ **Cons:**

- 词太多了！词汇表可能会非常大
- 低频词的长尾
  - ***Zipf's law***: a word's frequency is inversely 与其频率排名成反比
- 语言一直在变化！
  - Merriam Webster's Collegiate Dictionary added 第 12 版新增 5000 个词 (including *rizz*)

Corpus	Types = $\|V\|$
Shakespeare	31 thousand
Brown corpus	38 thousand
Switchboard telephone conversations	20 thousand
COCA	2 million
Google n-grams	13 million

Jurafsky & M

Zipf's Law



Word tokenization

As a first tokenization scheme,
let's consider simple ***word tokenization***

We'd sat back on the grass, and time flew by as we looked at the Milky Way.

❌ **Cons:**

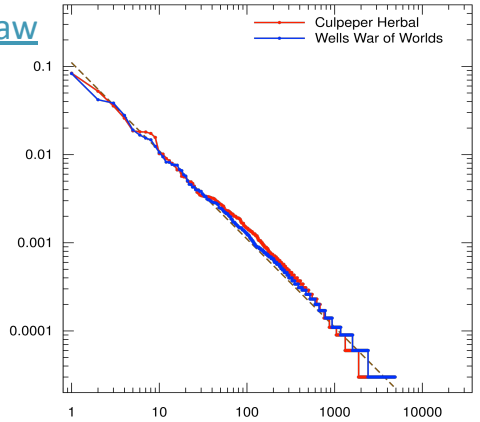
- How do we deal with unknown words? The ***out-of-vocabulary (OOV)*** problem
  - **Solution 1:** use regular expressions/other rule-based methods to preprocess text
    - Typos: **importaNt** ⟶ **important**
    - Regularize spellings: **ESPRESSSOOO** ⟶ **ESPRESSO**
  - **Solution 2:** use a single token for unknown words (**UNK**)
    - Many words would often get mapped to **UNK** 😬

Zipf's law is importaNt in NLP. ⟶ **UNK law isUNK in NLP.**
see Sutskever et al. 2014 & 2015

Corpus	Types = $\|V\|$
Shakespeare	31 thousand
Brown corpus	38 thousand
Switchboard telephone conversations	20 thousand
COCA	2 million
Google n-grams	13 million

Jurafsky & Martin Ch.2

Zipf's Law



词分词

作为第一种分词方案,
let's consider simple 词分词

We'd sat back on the grass, and time
在我们仰望银河时飞逝。

❌ **Cons:**

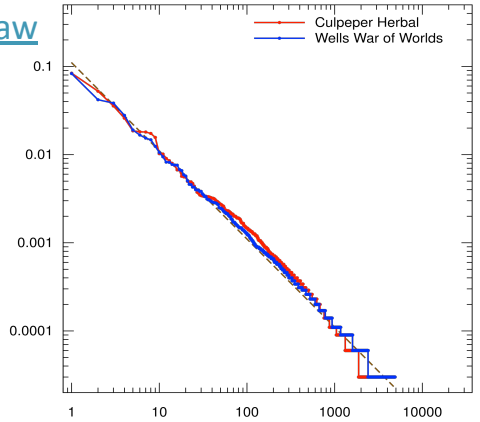
- How do we deal with unknown words? The 词汇表外 （OOV） problem
  - 方案 1：use regular expressions/other rule-based 文本预处理方法
    - Typos: **importaNt** → **important**
    - Regularize spellings: **ESPRESSSOOO** → **ESPRESSO**
  - 方案 2：use a single token for unknown words (**UNK**)
    - Many words would often get mapped to **UNK** 😬

Zipf's law is importaNt in NLP. → **UNK law isUNK in NLP.**
see Sutskever et al. 2014 & 2015

Corpus	Types = $\|V\|$
Shakespeare	31 thousand
Brown corpus	38 thousand
Switchboard telephone conversations	20 thousand
COCA	2 million
Google n-grams	13 million

Jurafsky & M

Zipf's Law



Character/byte tokenization (aka “tokenizer-free”)

Another simple tokenization scheme:
character/byte tokenization

We’d sat back on the grass, and time flew by as we looked at the Milky Way.

We’ll assume:

- **Tokens** = individual bytes/characters in the text
  - Note: the distinction between bytes and characters is relevant
    - 😀 = 1 character, 4 bytes; 地 = 1 character, 3 bytes
    - Byte and character-level models tend to be grouped together
- **Vocabulary** V = {a, b, c, …, z, A, B, C, …, Z} (+ spaces, punctuation, numbers, etc.)

Aside: should we call these *tokenizer-free*? See Arnett’s blog

Byte/character-level models: CharBERT, CANINE, Charformer, ByT5

✅ **Pros:**
- Small vocabulary size
- Complete coverage of input (no OOV!)
- Direct observation of spelling

❌ **Cons:**
- Really long sequences…
- Potentially more difficult to build higher-level representations (debatable!)

Character/byte tokenization (aka “tokenizer-free”)

另一种简单的分词方案：
字符 / 字节分词

We’d sat back on the grass, and time
在我们仰望银河时飞逝。

We’ll assume:

- Tokens = individual bytes/characters in the text
  - 注意：字节和字符之间的区别很重要
    - 😀 = 1 character, 4 bytes; 地 = 1 character, 3 bytes
    - 字节级和字符级模型往往被归为一类
- **Vocabulary** V = {a, b, c, …, z, A, B, C, …, Z} (+ spaces, punctuation, numbers, etc.)

旁注：should we call these 无分词器 ?
See Arnett’s blog

字节 / 字符级模型： CharBERT, CANINE, Charformer, ByT5

✅ **Pros:**

- 小词汇表大小
- 完全覆盖输入（无 OOV！）
- 直接观察拼写

❌ **Cons:**

- 非常长的序列 …
- 可能更难构建更高层的表示（有争议！）

Between words and characters: Subword Tokenization

How can we combine the efficiency of word tokenization with the high coverage of character tokenization?

Solution: ***subword tokenization***, where tokens are words or parts of words

Tokenization with **ChatGPT's tokenizer**:

Intuition behind subword tokenization:

- Use data + algorithms to define what the vocabulary should be
- Algorithms use corpus frequency statistics of adjacent segments of text to build the vocabulary
- While we sometimes get intuitive token segmentations (e.g., **We'd**), most of the time, these are not linguistically aligned (e.g., **Milky**, **looked**)

介于词和字符之间：子词分词

How can we combine the efficiency of word tokenization with the high coverage of 字符分词？

方案： 子词分词 ，其中 t o k e n 是词或词的一部分

Tokenization with **ChatGPT's tokenizer**:

直觉 behind subword tokenization:

- 使用数据 + 算法来定义词汇表应该是什么
- Algorithms use corpus frequency statistics of adjacent segments of text to build the 词汇表
- While we sometimes get intuitive token segmentations (e.g., **We'd**), most of the time, these are not linguistically aligned (e.g., **Milky**, **looked**)

Between words and characters: Subword Tokenization

Common subword tokenization algorithms:

- **Byte Pair Encoding (BPE)** (Sennrich et al., 2016)
  - Start with a set of individual bytes, then iteratively merge neighboring tokens with the highest probability/frequency, up to the desired vocab size $|V|$
- **WordPiece** (Wu et al., 2016)
  - Similar to BPE, but iteratively merge neighboring tokens which maximize the likelihood of the data after the merge
- **Unigram Language Modeling (ULM)** (Kudo, 2018)
  - Start with a large vocabulary (all-subword strings) and discard tokens until we reach the desired vocab size $|V|$

We will be focusing on the **BPE algorithm** in this lecture

介于词和字符之间：子词分词

常见的子词分词算法：

- **Byte Pair Encoding (BPE)** (Sennrich et al., 2016)
  - 从单个字节集合开始，然后迭代合并相邻 token 具有最高概率 / 频率的，直到达到期望的词汇表大小 $|V|$
- **WordPiece** (Wu et al., 2016)
  - Similar to BPE, but iteratively merge neighboring tokens which maximize the 合并后数据的似然
- Unigram 语言建模 (ULM) (Kudo, 2018)
  - Start with a large vocabulary (all-subword strings) and discard tokens until we 达到期望的词汇表大小 $|V|$

We will be focusing on the BPE 算法 in this lecture

Subword tokenization: Byte Pair Encoding (BPE)

Why focus on BPE? It is the tokenization algorithm used in all popular/frontier language models

What made BPE so universal?

- It was popularized by GPT-2, and OpenAI set the standard for LM training
- BPE is simple, easy to implement, efficient, and deterministic

Model Family	Type	Vocab Size
Gemini 2.x/Gemma 3	BPE	262k
Gemini 1.x/Gemma 2	BPE	256k
Llama 4	BPE	201k
GPT-4o/GPT-5/o1/o3	BPE	200k
Kimi K2	BPE	163k
Qwen2.5/Qwen3/GLM-4	BPE	151k
Grok 2	BPE	131k
Longcat Flash Chat	BPE	131k
Llama 3/Hunyuan T1/TurboS/Large	BPE	128k
DeepSeek v3/R1	BPE	128k
DeepSeek v2.5	BPE	100k

子词分词：字节对编码 （ＢＰＥ）

为什么关注 ＢＰＥ？It is the tokenization algorithm used in all popular/frontier language 模型

是什么使 ＢＰＥ 如此通用？

- It was popularized by GPT-2, and OpenAI set the ＬＭ 训练的标准
- BPE is simple, easy to implement, efficient, and 确定性的

模型系列	类型	词汇表大小
Gemini 2.x/Gemma 3	BPE	262k
Gemini 1.x/Gemma 2	BPE	256k
Llama 4	BPE	201k
GPT-4o/GPT-5/o1/o3	BPE	200k
Kimi K2	BPE	163k
Qwen2.5/Qwen3/GLM-4	BPE	151k
Grok 2	BPE	131k
Longcat Flash Chat	BPE	131k
Llama 3/Hunyuan T1/TurboS/Large	BPE	128k
DeepSeek v3/R1	BPE	128k
DeepSeek v2.5	BPE	100k

Subword tokenization: Byte Pair Encoding (BPE)

Required:

Training corpus D

Desired vocab size N

Training Algorithm:

1. Pre-tokenize by splitting on whitespace
2. Initialize V as bytes in D
3. Convert D into sequence of tokens (i.e., bytes)
4. While $|V| < N$:
    a. Get counts of all bigrams in D
    b. Merge most frequent pair (v_i, v_j)
        into new token $v_n = v_i v_j$
        where $n = |V| + 1$
    c. Replace all instances of $v_i v_j$ in D with v_n

子词分词：字节对编码 （BPE）

必需：

训练语料库 D

期望词汇表大小 N

训练算法：

1. 通过空格分割进行预分词
2. 将 V 初始化为 D 中的字节
3. 将 D 转换为 token 序列（即字节）
4. 当 $|V| < N$ 时：
   a. 获取 D 中所有二元组的计数
   b. 合并最频繁的对 (v_i, v_j)
      成为新 token $v_n = v_i v_j$
      其中 $n = |V| + 1$
   c. 将 D 中所有 $v_i v_j$ 实例替换为 v_n

Subword tokenization: Byte Pair Encoding (BPE)

Corpus

```
Peter Piper picked
a peck of pickled
peppers
```

This is our (very interesting) training corpus D

子词分词：字节对编码 （BPE）

语料库

Peter Piper picked a peck of pickled peppers

This is our (very interesting) 训练语料库 D）

Subword tokenization: Byte Pair Encoding (BPE)

Corpus

```
Peter, _Piper,
_picked, _a, _peck,
_of, _pickled,
_peppers
```

First, we pre-tokenize by splitting on whitespace

子词分词：字节对编码 （BPE）

语料库

```
Peter, _Piper,
_picked, _a, _peck,
_of, _pickled,
_peppers
```

First, we pre-tokenize by splitting on 空格

Subword tokenization: Byte Pair Encoding (BPE)

Corpus

```
Peter, _Piper,
_picked, _a, _peck,
_of, _pickled,
_peppers
```

Vocab

```
{_, a, c, d, e, f,
i, k, l, o, p, P,
r, s, t}
```

Initialize a vocab of individual bytes in D as tokens

子词分词：字节对编码 （BPE）

语料库

```
Peter, _Piper,
_picked, _a, _peck,
_of, _pickled,
_peppers
```

词汇表

```
{_, a, c, d, e, f,
i, k, l, o, p, P,
r, s, t}
```

Initialize a vocab of individual bytes in D

作为 token

Subword tokenization: Byte Pair Encoding (BPE)

Corpus

P e t e r, _ P i p e r, _ p i c k e d, _ a, _ p e c k, _ o f, _ p i c k l e d, _ p e p p e r s

Vocab

{_, a, c, d, e, f, i, k, l, o, p, P, r, s, t}

Convert D into a sequence of tokens (i.e., bytes)

子词分词：字节对编码 （BPE）

语料库

P e t e r, _ P i p e r, _ p i c k e d, _ a, _ p e c k, _ o f, _ p i c k l e d, _ p e p p e r s

词汇表

{_, a, c, d, e, f, i, k, l, o, p, P, r, s, t}

Convert D into a sequence of tokens (即字节)

Subword tokenization: Byte Pair Encoding (BPE)

Corpus

```
P e t e r, _ P i p
e r, _ p i c k e
d, _ a, _ p e c k,
_ o f, _ p i c k l
e d, _ p e p p e r
s
```

Vocab

```
{_, a, c, d, e, f,
i, k, l, o, p, P,
r, s, t}
```

Token Pair	Frequency
_ + p	4
p + e	4
c + k	3
e + r	3
e + d	2
i + c	2
p + i	2

Merge List

Initialize frequency table of byte pairs and (empty) merge list

子词分词：字节对编码 （BPE）

语料库

```
P e t e r, _ P i p
e r, _ p i c k e
d, _ a, _ p e c k,
_ o f, _ p i c k l
e d, _ p e p p e r
s
```

词汇表

```
{_, a, c, d, e, f,
i, k, l, o, p, P,
r, s, t}
```

Token	频率
_ + p	4
p + e	4
c + k	3
e + r	3
e + d	2
i + c	2
p + i	2

合并列表

Initialize frequency table of byte pairs and （空）合并列表

Subword tokenization: Byte Pair Encoding (BPE)

Corpus

```
P e t e r, _ P i p
e r, _ p i c k e
d, _ a, _ p e c k,
_ o f, _ p i c k l
e d, _ p e p p e r
s
```

Vocab

```
{_, a, c, d, e, f,
i, k, l, o, p, P,
r, s, t}
```

Token Pair	Frequency
_ + p	4
p + e	4
c + k	3
e + r	3
e + d	2
i + c	2
p + i	2

Merge List

Retrieve most frequent pair of tokens (v_i, v_j)

子词分词：字节对编码 （BPE）

语料库

```
P e t e r, _ P i p
e r, _ p i c k e
d, _ a, _ p e c k,
_ o f, _ p i c k l
e d, _ p e p p e r
s
```

词汇表

```
{_, a, c, d, e, f,
i, k, l, o, p, P,
r, s, t}
```

Token	频率
_ + p	4
p + e	4
c + k	3
e + r	3
e + d	2
i + c	2
p + i	2

合并列表

Retrieve most frequent pair of tokens (v_i, v_j)

Subword tokenization: Byte Pair Encoding (BPE)

Corpus

P e t e r, _ P i p
e r, _p i c k e d,
_ a, _p e c k, _ o
f, _p i c k l e d,
_p e p p e r s

Vocab

{_, a, c, d, e, f,
i, k, l, o, p, P,
r, s, t, _p}

Token Pair	Frequency
_ + p	4
p + e	4
c + k	3
e + r	3
e + d	2
i + c	2
p + i	2

Merge List

1. _ + p -> _p

Merge the token pair into a new token $v_n = v_i v_j$

Update the corpus, vocab, and merge list

子词分词：字节对编码 （BPE）

语料库

P e t e r, _ P i p
e r, _p i c k e d,
_ a, _p e c k, _ o
f, _p i c k l e d,
_p e p p e r s

词汇表

{_, a, c, d, e, f,
i, k, l, o, p, P,
r, s, t, _p}

Token	频率
_ + p	4
p + e	4
c + k	3
e + r	3
e + d	2
i + c	2
p + i	2

合并列表

1. _ + p -> _p

Merge the token pair

成为一个新 token

$v_n = v_i v_j$

Update the corpus,

词汇表和合并列表

Subword tokenization: Byte Pair Encoding (BPE)

Corpus

P e t e r, _ P i p
e r, _p i c k e d,
_ a, _p e c k, _ o
f, _p i c k l e d,
_p e p p e r s

Vocab

{_, a, c, d, e, f,
i, k, l, o, p, P,
r, s, t, _p}

Token Pair	Frequency
_ + p	4
p + e	4
c + k	3
e + r	3
e + d	2
i + c	2
p + i	2

Merge List

1. _ + p -> _p

Update the frequency table (since the corpus has changed)

子词分词：字节对编码 （BPE）

语料库

P e t e r, _ P i p
e r, _p i c k e d,
_ a, _p e c k, _ o
f, _p i c k l e d,
_p e p p e r s

词汇表

{_, a, c, d, e, f,
i, k, l, o, p, P,
r, s, t, _p}

Token	频率
_ + p	4
p + e	4
c + k	3
e + r	3
e + d	2
i + c	2
p + i	2

合并列表

1. _ + p -> _p

Update the frequency table (since the corpus 已改变)

Subword tokenization: Byte Pair Encoding (BPE)

Corpus

P e t e r, _ P i p
e r, _p i c k e d,
_ a, _p e c k, _ o
f, _p i c k l e d,
_p e p p e r s

Vocab

{_, a, c, d, e, f,
i, k, l, o, p, P,
r, s, t, _p}

Token Pair	Frequency
c + k	3
e + r	3
_p + e	2
_p + i	2
e + d	2
i + c	2
p + e	2

Merge List

1. _ + p -> _p

Update the frequency table (since the corpus has changed)

子词分词：字节对编码 （BPE）

语料库

p e t e r, _ P i p
e r, _p i c k e d,
_ a, _p e c k, _ o
f, _p i c k l e d,
_p e p p e r s

词汇表

{_, a, c, d, e, f,
i, k, l, o, p, P,
r, s, t, _p}

Token	频率
c + k	3
e + r	3
_p + e	2
_p + i	2
e + d	2
i + c	2
p + e	2

合并列表

1. _ + p -> _p

Update the frequency table (since the corpus 已改变)

Subword tokenization: Byte Pair Encoding (BPE)

Corpus

P e t e r, _ P i p
e r, _p i c k e d,
_ a, _p e c k, _ o
f, _p i c k l e d,
_p e p p e r s

Vocab

{_, a, c, d, e, f,
i, k, l, o, p, P,
r, s, t, _p}

Token Pair	Frequency
c + k	3
e + r	3
_p + e	2
_p + i	2
e + d	2
i + c	2
p + e	2

Merge List

1. _ + p -> _p

Retrieve most frequent pair of tokens (v_i, v_j)

子词分词：字节对编码 （BPE）

语料库

P e t e r, _ P i p
e r, _p i c k e d,
_ a, _p e c k, _ o
f, _p i c k l e d,
_p e p p e r s

词汇表

{_, a, c, d, e, f,
i, k, l, o, p, P,
r, s, t, _p}

Token	频率
c + k	3
e + r	3
_p + e	2
_p + i	2
e + d	2
i + c	2
p + e	2

合并列表

1. _ + p -> _p

Retrieve most frequent pair of tokens (v_i, v_j)

Subword tokenization: Byte Pair Encoding (BPE)

Corpus

P e t e r, _ P i p
e r, _p i ck e d,
_ a, _p e ck, _ o
f, _p i ck l e d,
_p e p p e r s

Vocab

{_, a, c, d, e, f,
i, k, l, o, p, P,
r, s, t, _p, ck}

Token Pair	Frequency
c + k	3
e + r	3
_p + e	2
_p + i	2
e + d	2
i + c	2
p + e	2

Merge List

1. _ + p -> _p
2. c + k -> ck

Merge the token pair into a new token $v_n = v_i v_j$

Update the corpus, vocab, and merge list

子词分词：字节对编码 （BPE）

语料库

P e t e r, _ P i p
e r, _p i ck e d,
_ a, _p e ck, _ o
f, _p i ck l e d,
_p e p p e r s

词汇表

{_, a, c, d, e, f,
i, k, l, o, p, P,
r, s, t, _p, ck}

Token	频率
c + k	3
e + r	3
_p + e	2
_p + i	2
e + d	2
i + c	2
p + e	2

合并列表

1. _ + p -> _p
2. c + k -> ck

Merge the token pair

成为一个新 token

$v_n = v_i v_j$

Update the corpus,

词汇表和合并列表

Subword tokenization: Byte Pair Encoding (BPE)

Corpus

P e t e r, _ P i p
e r, _p i ck e d,
_ a, _p e ck, _ o
f, _p i ck l e d,
_p e p p e r s

Vocab

{_, a, c, d, e, f,
i, k, l, o, p, P,
r, s, t, _p, ck}

Token Pair	Frequency
c + k	3
e + r	3
_p + e	2
_p + i	2
e + d	2
i + c	2
p + e	2

Merge List

1. _ + p -> _p
2. c + k -> ck

Update the frequency table (since the corpus has changed)

子词分词：字节对编码 （BPE）

语料库

P e t e r, _ P i p
e r, _p i ck e d,
_ a, _p e ck, _ o
f, _p i ck l e d,
_p e p p e r s

词汇表

{_, a, c, d, e, f,
i, k, l, o, p, P,
r, s, t, _p, ck}

Token	频率
c + k	3
e + r	3
_p + e	2
_p + i	2
e + d	2
i + c	2
p + e	2

合并列表

1. _ + p -> _p
2. c + k -> ck

Update the frequency table (since the corpus 已改变)

Subword tokenization: Byte Pair Encoding (BPE)

Corpus

P e t e r, _ P i p
e r, _p i ck e d,
_ a, _p e ck, _ o
f, _p i ck l e d,
_p e p p e r s

Vocab

{_, a, c, d, e, f,
i, k, l, o, p, P,
r, s, t, _p, ck}

Token Pair	Frequency
e + r	3
_p + e	2
_p + i	2
e + d	2
i + ck	2
p + e	2
_ + a	1

Merge List

1. _ + p -> _p
2. c + k -> ck

Update the frequency table (since the corpus has changed)

子词分词：字节对编码 （BPE）

语料库

```
P e t e r, _ P i p
e r, _p i ck e d,
_ a, _p e ck, _ o
f, _p i ck l e d,
_p e p p e r s
```

词汇表

```
{_, a, c, d, e, f,
i, k, l, o, p, P,
r, s, t, _p, ck}
```

Token	频率
e + r	3
_p + e	2
_p + i	2
e + d	2
i + ck	2
p + e	2
_ + a	1

合并列表

```
1.  _ + p   -> _p
2.  c + k   -> ck
```

Update the frequency table (since the corpus 已改变)

Subword tokenization: Byte Pair Encoding (BPE)

Corpus

P e t e r, _ P i p
e r, _p i ck e d,
_ a, _p e ck, _ o
f, _p i ck l e d,
_p e p p e r s

Vocab

{_, a, c, d, e, f,
i, k, l, o, p, P,
r, s, t, _p, ck}

Token Pair	Frequency
e + r	3
_p + e	2
_p + i	2
e + d	2
i + ck	2
p + e	2
_ + a	1

Merge List

1. _ + p -> _p
2. c + k -> ck

Retrieve most frequent pair of tokens (v_i, v_j)

子词分词：字节对编码 （BPE）

语料库

p e t e r, _ P i p
e r, _p i ck e d,
_ a, _p e ck, _ o
f, _p i ck l e d,
_p e p p e r s

词汇表

{_, a, c, d, e, f,
i, k, l, o, p, P,
r, s, t, _p, ck}

Token	频率
e + r	3
_p + e	2
_p + i	2
e + d	2
i + ck	2
p + e	2
_ + a	1

合并列表

1. _ + p -> _p
2. c + k -> ck

Retrieve most frequent pair of tokens (v_i, v_j)

Subword tokenization: Byte Pair Encoding (BPE)

Corpus

P e t er, _ P i p
er, _p i ck e d, _
a, _p e ck, _ o f,
_p i ck l e d, _p
e p p er s

Vocab

{_, a, c, d, e, f,
i, k, l, o, p, P,
r, s, t, _p, ck,
er}

Token Pair	Frequency
e + r	3
_p + e	2
_p + i	2
e + d	2
i + ck	2
p + e	2
_ + a	1

Merge List

1. _ + p -> _p
2. c + k -> ck
3. e + r -> er

Merge the token pair into a new token $v_n = v_i v_j$

Update the corpus, vocab, and merge list

子词分词：字节对编码 （BPE）

语料库

P e t er, _ P i p
er, _p i ck e d, _
a, _p e ck, _ o f,
_p i ck l e d, _p
e p p er s

词汇表

{_, a, c, d, e, f,
i, k, l, o, p, P,
r, s, t, _p, ck,
er}

Token	频率
e + r	3
_p + e	2
_p + i	2
e + d	2
i + ck	2
p + e	2
_ + a	1

合并列表

1. _ + p -> _p
2. c + k -> ck
3. e + r -> er

Merge the token pair

成为一个新 token

$v_n = v_i v_j$

Update the corpus,

词汇表和合并列表

Subword tokenization: Byte Pair Encoding (BPE)

Corpus

P e t er, _ P i p
er, _p i ck e d, _
a, _p e ck, _ o f,
_p i ck l e d, _p
e p p er s

Vocab

{_, a, c, d, e, f,
i, k, l, o, p, P,
r, s, t, _p, ck,
er}

Token Pair	Frequency
e + r	3
_p + e	2
_p + i	2
e + d	2
i + ck	2
p + e	2
_ + a	1

Merge List

1. _ + p -> _p
2. c + k -> ck
3. e + r -> er

Update the frequency table (since the corpus has changed)

子词分词：字节对编码 （BPE）

语料库

P e t er, _ P i p
er, _p i ck e d, _
a, _p e ck, _ o f,
_p i ck l e d, _p
e p p er s

词汇表

{_, a, c, d, e, f,
i, k, l, o, p, P,
r, s, t, _p, ck,
er}

Token	频率
e + r	3
_p + e	2
_p + i	2
e + d	2
i + ck	2
p + e	2
_ + a	1

合并列表

1. _ + p -> _p
2. c + k -> ck
3. e + r -> er

Update the frequency table (since the corpus 已改变)

Subword tokenization: Byte Pair Encoding (BPE)

Corpus

P e t er, _ P i p
er, _p i ck e d, _
a, _p e ck, _ o f,
_p i ck l e d, _p
e p p er s

Vocab

{_, a, c, d, e, f,
i, k, l, o, p, P,
r, s, t, _p, ck,
er}

Token Pair	Frequency
_p + e	2
_p + i	2
e + d	2
i + ck	2
p + er	2
_ + a	1
_ + o	1

Merge List

1. _ + p -> _p
2. c + k -> ck
3. e + r -> er

Update the frequency table (since the corpus has changed)

子词分词：字节对编码 （BPE）

语料库

P e t er, _ P i p
er, _p i ck e d, _
a, _p e ck, _ o f,
_p i ck l e d, _p
e p p er s

词汇表

{_, a, c, d, e, f,
i, k, l, o, p, P,
r, s, t, _p, ck,
er}

Token	频率
_p + e	2
_p + i	2
e + d	2
i + ck	2
p + er	2
_ + a	1
_ + o	1

合并列表

1. _ + p -> _p
2. c + k -> ck
3. e + r -> er

Update the frequency table (since the corpus 已改变)

Subword tokenization: Byte Pair Encoding (BPE)

Corpus

P e t er, _ P i p
er, _p i ck e d, _
a, _p e ck, _ o f,
_p i ck l e d, _p
e p p er s

Vocab

{_, a, c, d, e, f,
i, k, l, o, p, P,
r, s, t, _p, ck,
er}

Token Pair	Frequency
_p + e	2
_p + i	2
e + d	2
i + ck	2
p + er	2
_ + a	1
_ + o	1

Merge List

1. _ + p -> _p
2. c + k -> ck
3. e + r -> er

子词分词：字节对编码 （BPE）

语料库

P e t er, _ P i p
er, _p i ck e d, _
a, _p e ck, _ o f,
_p i ck l e d, _p
e p p er s

词汇表

{_, a, c, d, e, f,
i, k, l, o, p, P,
r, s, t, _p, ck,
er}

Token	频率
_p + e	2
_p + i	2
e + d	2
i + ck	2
p + er	2
_ + a	1
_ + o	1

合并列表

1. _ + p -> _p
2. c + k -> ck
3. e + r -> er

Subword tokenization: Byte Pair Encoding (BPE)

Corpus

P e t er, _ P i p
er, _p i ck e d, _
a, _pe ck, _ o f,
_p i ck l e d, _pe
p p er s

Vocab

{_, a, c, d, e, f,
i, k, l, o, p, P,
r, s, t, _p, ck,
er, _pe}

Token Pair	Frequency
_p + e	2
_p + i	2
e + d	2
i + ck	2
p + er	2
_ + a	1
_ + o	1

Merge List

1. _ + p -> _p
2. c + k -> ck
3. e + r -> er
4. _p + e -> _pe

子词分词：字节对编码 （BPE）

语料库

P e t er, _ P i p er, _p i ck e d, _ a, _pe ck, _ o f, _p i ck l e d, _pe p p er s

词汇表

{_, a, c, d, e, f, i, k, l, o, p, P, r, s, t, _p, ck, er, _pe}

Token	频率
_p + e	2
_p + i	2
e + d	2
i + ck	2
p + er	2
_ + a	1
_ + o	1

合并列表

1. _ + p -> _p
2. c + k -> ck
3. e + r -> er
4. _p + e -> _pe

Subword tokenization: Byte Pair Encoding (BPE)

Corpus

P e t er, _ P i p
er, _p i ck e d, _
a, _pe ck, _ o f,
_p i ck l e d, _pe
p p er s

Vocab

{_, a, c, d, e, f,
i, k, l, o, p, P,
r, s, t, _p, ck,
er, _pe}

Token Pair	Frequency
_p + e	2
_p + i	2
e + d	2
i + ck	2
p + er	2
_ + a	1
_ + o	1

Merge List

1. _ + p -> _p
2. c + k -> ck
3. e + r -> er
4. _p + e -> _pe

子词分词：字节对编码 （BPE）

语料库

P e t er, _ P i p er, _p i ck e d, _ a, _pe ck, _ o f, _p i ck l e d, _pe p p er s

词汇表

{_, a, c, d, e, f, i, k, l, o, p, P, r, s, t, _p, ck, er, _pe}

Token	频率
_p + e	2
_p + i	2
e + d	2
i + ck	2
p + er	2
_ + a	1
_ + o	1

合并列表

1. _ + p -> _p
2. c + k -> ck
3. e + r -> er
4. _p + e -> _pe

Subword tokenization: Byte Pair Encoding (BPE)

Corpus

P e t er, _ P i p
er, _p i ck e d, _
a, _pe ck, _ o f,
_p i ck l e d, _pe
p p er s

Vocab

{_, a, c, d, e, f,
i, k, l, o, p, P,
r, s, t, _p, ck,
er, _pe}

Token Pair	Frequency
_p + i	2
e + d	2
i + ck	2
p + er	2
_ + a	1
_ + o	1
_ + P	1

Merge List

1. _ + p -> _p
2. c + k -> ck
3. e + r -> er
4. _p + e -> _pe

子词分词：字节对编码 （BPE）

语料库

P e t er, _ P i p er, _p i ck e d, _ a, _pe ck, _ o f, _p i ck l e d, _pe p p er s

词汇表

{_, a, c, d, e, f, i, k, l, o, p, P, r, s, t, _p, ck, er, _pe}

Token	频率
_p + i	2
e + d	2
i + ck	2
p + er	2
_ + a	1
_ + o	1
_ + P	1

合并列表

1. _ + p -> _p
2. c + k -> ck
3. e + r -> er
4. _p + e -> _pe

Subword tokenization: Byte Pair Encoding (BPE)

Corpus

P e t er, _ P i p
er, _p i ck e d, _
a, _pe ck, _ o f,
_p i ck l e d, _pe
p p er s

Vocab

{_, a, c, d, e, f,
i, k, l, o, p, P,
r, s, t, _p, ck,
er, _pe}

Token Pair	Frequency
_p + i	2
e + d	2
i + ck	2
p + er	2
_ + a	1
_ + o	1
_ + P	1

Merge List

1. _ + p -> _p
2. c + k -> ck
3. e + r -> er
4. _p + e -> _pe

子词分词：字节对编码 （BPE）

语料库

P e t er, _ P i p
er, _p i ck e d, _
a, _pe ck, _ o f,
_p i ck l e d, _pe
p p er s

词汇表

{_, a, c, d, e, f,
i, k, l, o, p, P,
r, s, t, _p, ck,
er, _pe}

Token	频率
_p + i	2
e + d	2
i + ck	2
p + er	2
_ + a	1
_ + o	1
_ + P	1

合并列表

1. _ + p -> _p
2. c + k -> ck
3. e + r -> er
4. _p + e -> _pe

Subword tokenization: Byte Pair Encoding (BPE)

Corpus

P e t er, _ P i p
er, _pi ck e d, _
a, _pe ck, _ o f,
_pi ck l e d, _pe
p p er s

Vocab

{_, a, c, d, e, f,
i, k, l, o, p, P,
r, s, t, _p, ck,
er, _pe, _pi}

Token Pair	Frequency
_p + i	2
e + d	2
i + ck	2
p + er	2
_ + a	1
_ + o	1
_ + P	1

Merge List

1. _ + p -> _p
2. c + k -> ck
3. e + r -> er
4. _p + e -> _pe
5. _p + i -> _pi

子词分词：字节对编码 （BPE）

语料库

P e t er, _ P i p
er, _pi ck e d, _
a, _pe ck, _ o f,
_pi ck l e d, _pe
p p er s

词汇表

{_, a, c, d, e, f,
i, k, l, o, p, P,
r, s, t, _p, ck,
er, _pe, _pi}

Token	频率
_p + i	2
e + d	2
i + ck	2
p + er	2
_ + a	1
_ + o	1
_ + P	1

合并列表

1. _ + p -> _p
2. c + k -> ck
3. e + r -> er
4. _p + e -> _pe
5. _p + i -> _pi

Subword tokenization: Byte Pair Encoding (BPE)

Corpus

P e t er, _ P i p
er, _pi ck e d, _
a, _pe ck, _ o f,
_pi ck l e d, _pe
p p er s

Vocab

{_, a, c, d, e, f,
i, k, l, o, p, P,
r, s, t, _p, ck,
er, _pe, _pi}

Token Pair	Frequency
_p + i	2
e + d	2
i + ck	2
p + er	2
_ + a	1
_ + o	1
_ + P	1

Merge List

1. _ + p -> _p
2. c + k -> ck
3. e + r -> er
4. _p + e -> _pe
5. _p + i -> _pi

子词分词：字节对编码 （BPE）

语料库

P e t er, _ P i p
er, _pi ck e d, _
a, _pe ck, _ o f,
_pi ck l e d, _pe
p p er s

词汇表

{_, a, c, d, e, f,
i, k, l, o, p, P,
r, s, t, _p, ck,
er, _pe, _pi}

Token	频率
_p + i	2
e + d	2
i + ck	2
p + er	2
_ + a	1
_ + o	1
_ + P	1

合并列表

1. _ + p -> _p
2. c + k -> ck
3. e + r -> er
4. _p + e -> _pe
5. _p + i -> _pi

Subword tokenization: Byte Pair Encoding (BPE)

Corpus

P e t er, _ P i p
er, _pi ck e d, _
a, _pe ck, _ o f,
_pi ck l e d, _pe
p p er s

Vocab

{_, a, c, d, e, f,
i, k, l, o, p, P,
r, s, t, _p, ck,
er, _pe, _pi}

Token Pair	Frequency
_pi + ck	2
e + d	2
p + er	2
_ + a	1
_ + o	1
_ + P	1
_pe + ck	1

Merge List

1. _ + p -> _p
2. c + k -> ck
3. e + r -> er
4. _p + e -> _pe
5. _p + i -> _pi

子词分词：字节对编码 （BPE）

语料库

P e t er, _ P i p
er, _pi ck e d, _
a, _pe ck, _ o f,
_pi ck l e d, _pe
p p er s

词汇表

{_, a, c, d, e, f,
i, k, l, o, p, P,
r, s, t, _p, ck,
er, _pe, _pi}

Token	频率
_pi + ck	2
e + d	2
p + er	2
_ + a	1
_ + o	1
_ + P	1
_pe + ck	1

合并列表

1. _ + p -> _p
2. c + k -> ck
3. e + r -> er
4. _p + e -> _pe
5. _p + i -> _pi

Subword tokenization: Byte Pair Encoding (BPE)

Corpus

P e t er, _ P i p
er, _pi ck e d, _
a, _pe ck, _ o f,
_pi ck l e d, _pe
p p er s

Vocab

{_, a, c, d, e, f,
i, k, l, o, p, P,
r, s, t, _p, ck,
er, _pe, _pi}

Token Pair	Frequency
_pi + ck	2
e + d	2
p + er	2
_ + a	1
_ + o	1
_ + P	1
_pe + ck	1

Merge List

1. _ + p -> _p
2. c + k -> ck
3. e + r -> er
4. _p + e -> _pe
5. _p + i -> _pi

子词分词：字节对编码 （BPE）

语料库

P e t er, _ P i p
er, _pi ck e d, _
a, _pe ck, _ o f,
_pi ck l e d, _pe
p p er s

词汇表

{_, a, c, d, e, f,
i, k, l, o, p, P,
r, s, t, _p, ck,
er, _pe, _pi}

Token	频率
_pi + ck	2
e + d	2
p + er	2
_ + a	1
_ + o	1
_ + P	1
_pe + ck	1

合并列表

1. _ + p -> _p
2. c + k -> ck
3. e + r -> er
4. _p + e -> _pe
5. _p + i -> _pi

Subword tokenization: Byte Pair Encoding (BPE)

Corpus

P e t er, _ P i p
er, _pick e d, _
a, _pe ck, _ o f,
_pick l e d, _pe p
p er s

Vocab

{_, a, c, d, e, f,
i, k, l, o, p, P,
r, s, t, _p, ck,
er, _pe, _pi, _pick}

Token Pair	Frequency
_pi + ck	2
e + d	2
p + er	2
_ + a	1
_ + o	1
_ + P	1
_pe + ck	1

Merge List

1. _ + p -> _p
2. c + k -> ck
3. e + r -> er
4. _p + e -> _pe
5. _p + i -> _pi
6. _pi + ck -> _pick

子词分词：字节对编码 （BPE）

语料库

P e t er, _ P i p
er, _pick e d, _
a, _pe ck, _ o f,
_pick l e d, _pe p
p er s

词汇表

{_, a, c, d, e, f,
i, k, l, o, p, P,
r, s, t, _p, ck,
er, _pe, _pi, _pick}

Token	频率
_pi + ck	2
e + d	2
p + er	2
_ + a	1
_ + o	1
_ + P	1
_pe + ck	1

合并列表

1. _ + p -> _p
2. c + k -> ck
3. e + r -> er
4. _p + e -> _pe
5. _p + i -> _pi
6. _pi + ck -> _pick

Subword tokenization: Byte Pair Encoding (BPE)

Corpus

P e t er, _ P i p
er, _pick e d, _
a, _pe ck, _ o f,
_pick l e d, _pe p
p er s

Vocab

{_, a, c, d, e, f,
i, k, l, o, p, P,
r, s, t, _p, ck,
er, _pe, _pi, _pick}

Token Pair	Frequency
_pi + ck	2
e + d	2
p + er	2
_ + a	1
_ + o	1
_ + P	1
_pe + ck	1

Merge List

1. _ + p -> _p
2. c + k -> ck
3. e + r -> er
4. _p + e -> _pe
5. _p + i -> _pi
6. _pi + ck -> _pick

...

continue until the desired vocab size N is reached

子词分词：字节对编码 （BPE）

语料库

P e t er, _ P i p
er, _pick e d, _
a, _pe ck, _ o f,
_pick l e d, _pe p
p er s

词汇表

{_, a, c, d, e, f,
i, k, l, o, p, P,
r, s, t, _p, ck,
er, _pe, _pi, _pick}

Token	频率
_pi + ck	2
e + d	2
p + er	2
_ + a	1
_ + o	1
_ + P	1
_pe + ck	1

合并列表

1. _ + p -> _p
2. c + k -> ck
3. e + r -> er
4. _p + e -> _pe
5. _p + i -> _pi
6. _pi + ck -> _pick

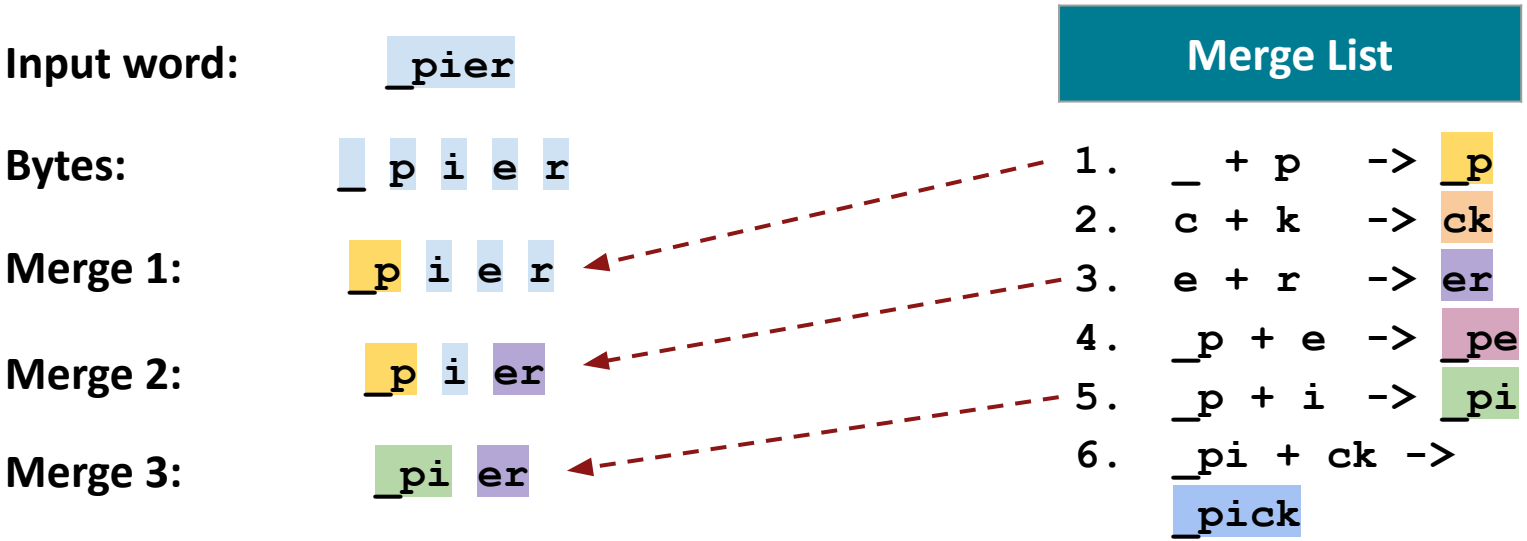
...

continue until the desired

达到词汇表大小

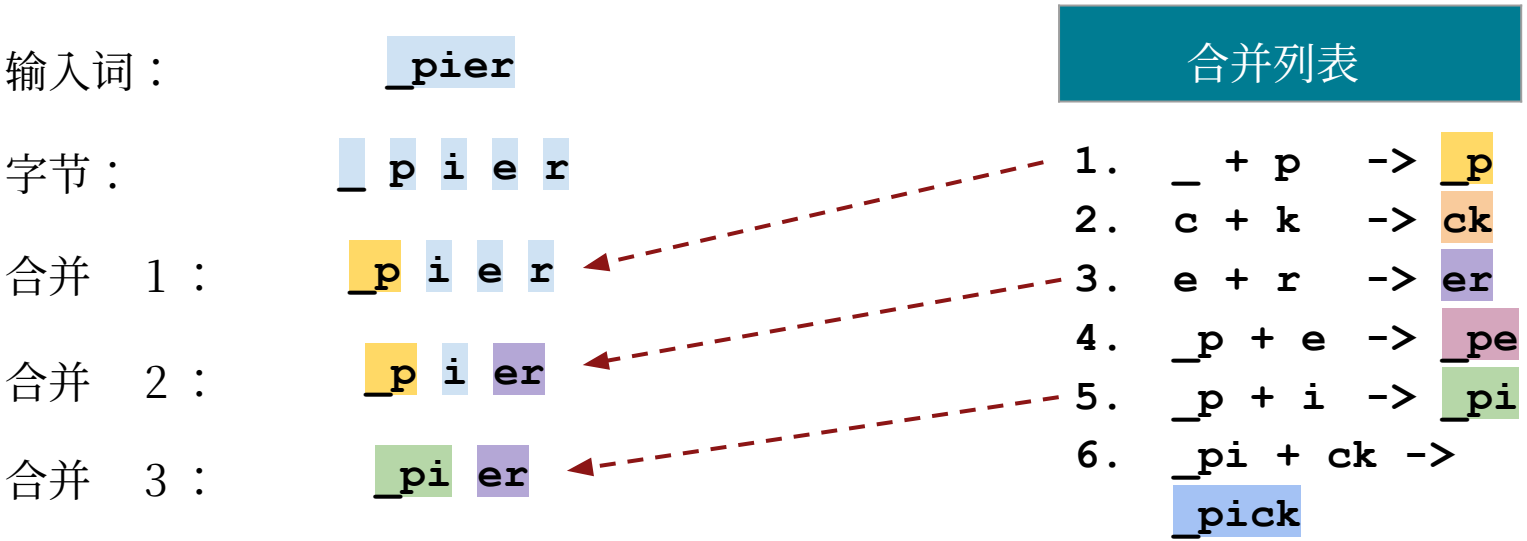
Subword tokenization: Byte Pair Encoding (BPE)

We just went over the **BPE training algorithm**. To tokenize **new text at test time**, we split it into bytes and apply merge rules in order



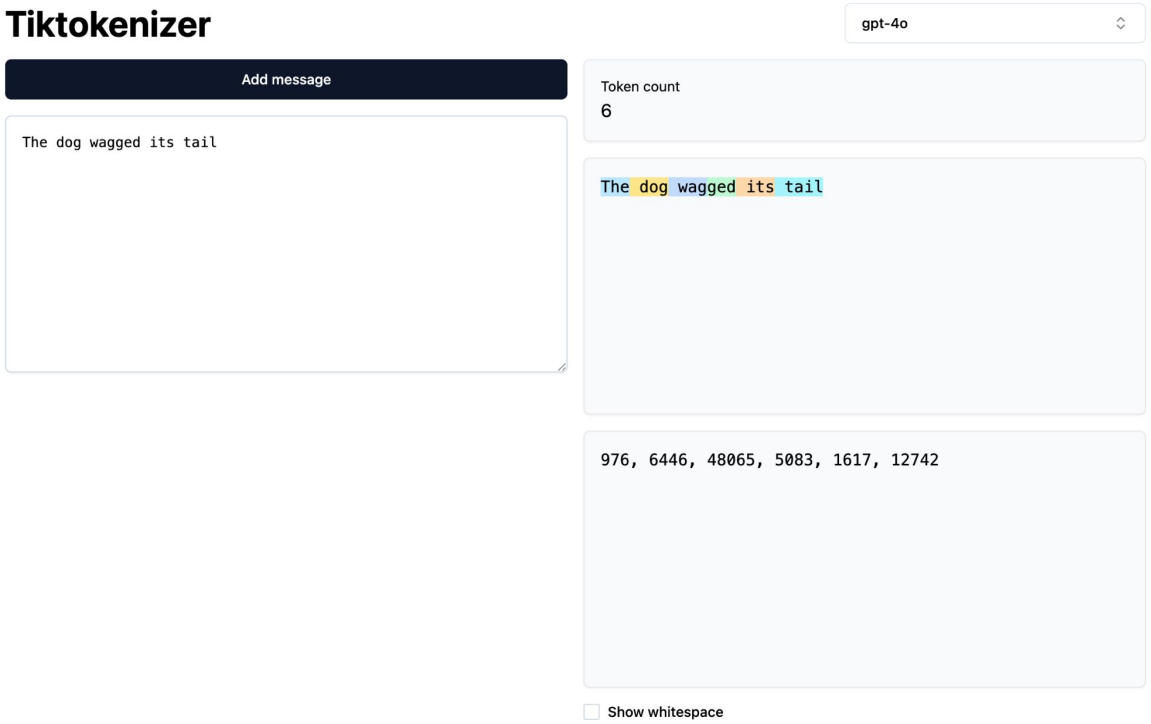
子词分词：字节对编码 （BPE）

We just went over the BPE 训练算法 . To tokenize 测试时的新文本 , we split 将其转换为字节并按顺序应用合并规则



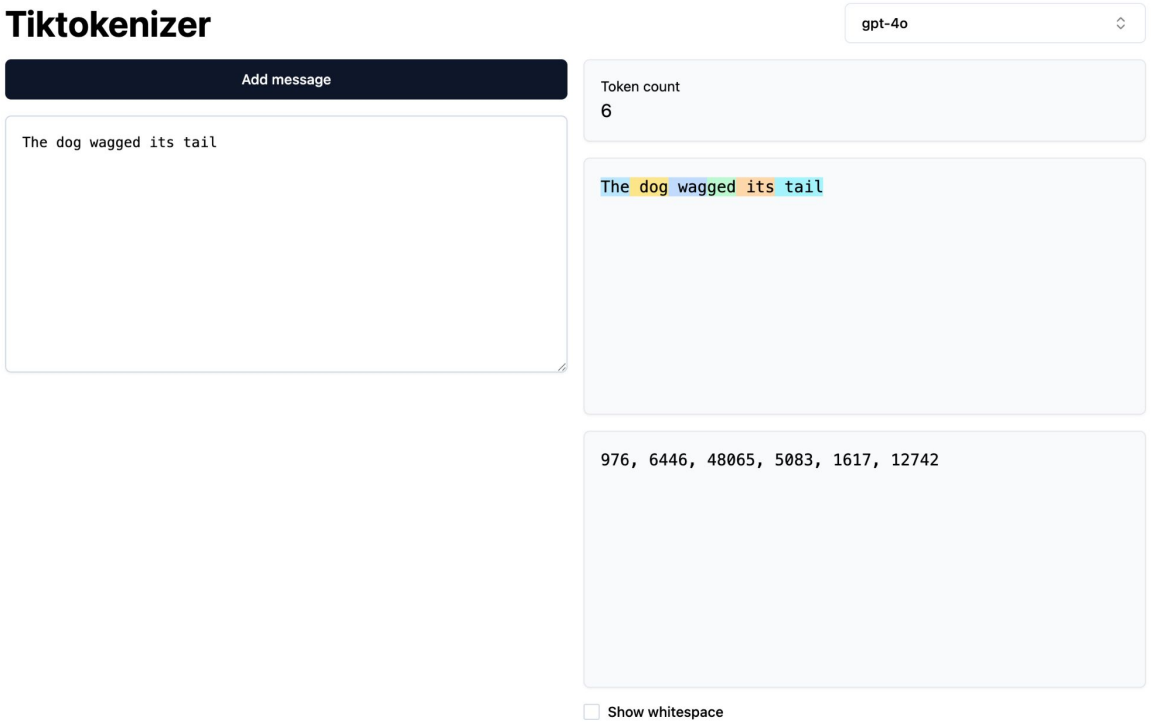
Tiktokenizer demo

Take a look at how some common tokenizers segment text:
https://tiktoke



Tiktokenizer 演示

Take a look at how some common tokenizers segment text:
https://tiktoke



Beyond Subwords: SentencePiece & Superword Tokenization

Pre-tokenizing based on whitespace: does it always make sense?

- Some languages don't use whitespace to separate words (Chinese, Japanese, Thai)
- Multi-word expressions (e.g., "on the other hand", "depend on")

SentencePiece (Kudo & Richardson, 2018): a tokenizer **library** (not algorithm!) that implements BPE and Unigram LM

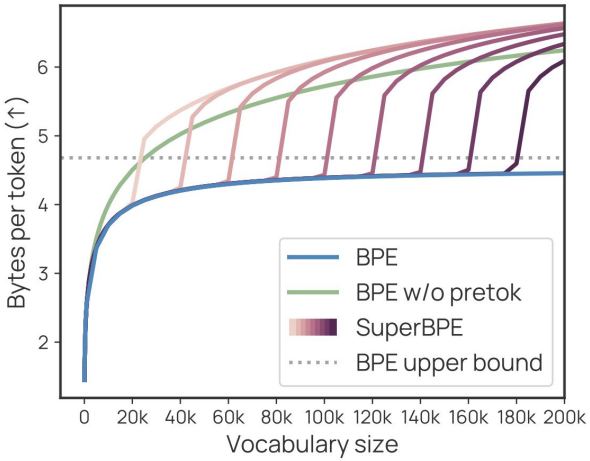
- Treats whitespace as a basic symbol in the vocabulary
- **Used by:** T5, Llama 2, XLM-R, Gemma, many multilingual models

SuperBPE (Liu et al., 2025): a pre-tokenization curriculum that first learns subwords, then removes whitespace constraints to learn superword tokens

- Smaller vocab size and less inference compute

BPE: By | the | way | , | I | am | a | fan | of | the | Milky | Way | .

SuperBPE: By the way | , I am a fan of the Milky Way | .



超越子词：SentencePiece 和超词分词

基于空格的预分词：　　does it always make sense?

- Some languages don't use whitespace to separate words (Chinese, Japanese, Thai)
- Multi-word expressions (e.g., "on the other hand", "depend on")

SentencePiece (Kudo & Richardson, 2018): a tokenizer 库　　(not algorithm!) that 实现了 BPE 和 Unigram LM

- 将空格作为词汇表中的基本符号
- 使用者： T5, Llama 2, XLM-R, Gemma, many multilingual models

SuperBPE (Liu et al., 2025 ）：预分词课程

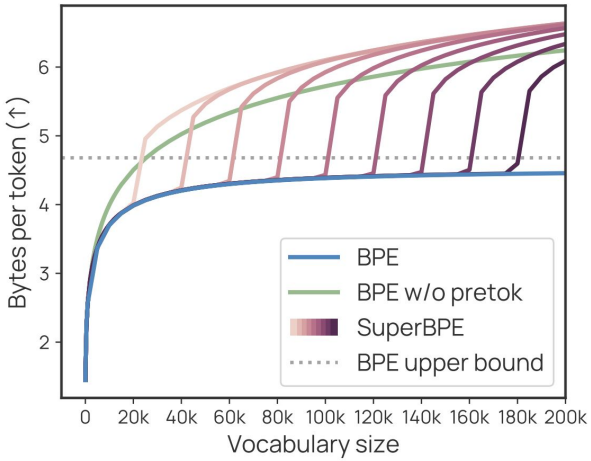
先学习子词，然后去除空格

约束来学习超词 token

- 更小的词汇表和更少的推理计算

BPE: By the way , I am a fan of the Milky Way .

SuperBPE: By the way , I am a fan of the Milky Way .



Has Subword Tokenization Solved All Our Problems?

Subword tokenization with BPE is the de-facto standard today:

- BPE is a simple and effective compression algorithm
- Everything can be represented with the vocabulary (no OOV!)
- Some shared representations (e.g., picked, picking)

Not everything is solved! There's still a lot of work to be done

子词分词解决了我们所有的问题吗？

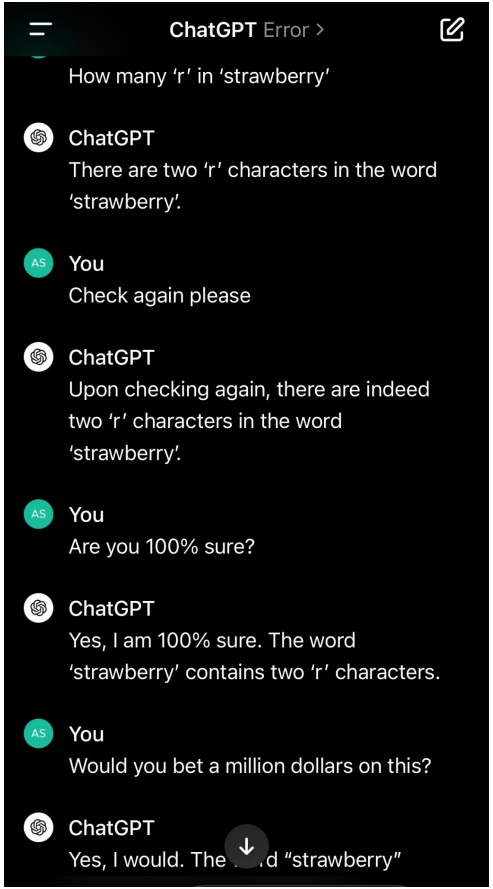
Subword tokenization with BPE is the

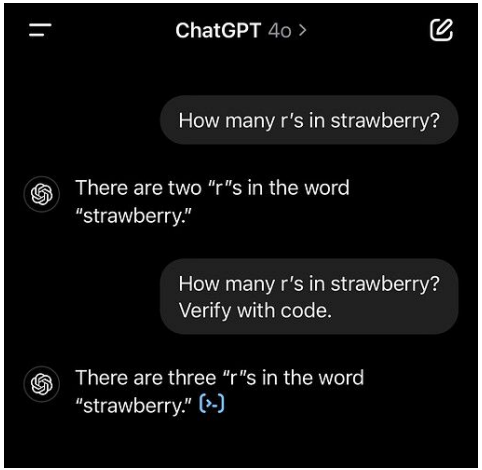
当今的事实标准：

- BPE is a simple and effective 压缩算法
- Everything can be represented 与词汇表（无 ＯＯＶ！）
- Some shared representations (e.g., picked, picking)

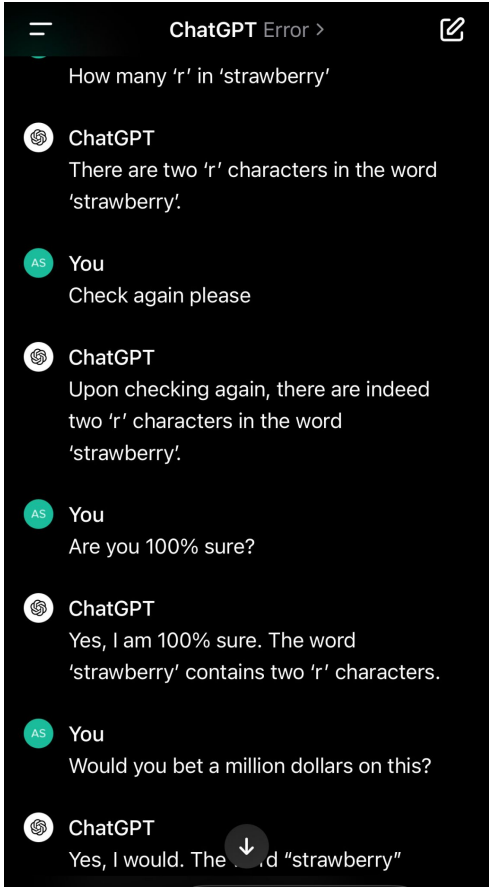
Not everything is solved! There’s still a 很多工作要做

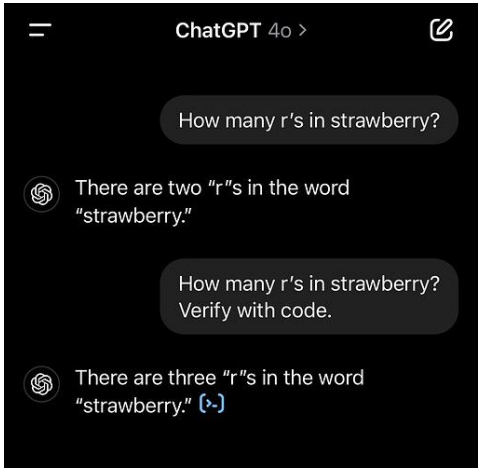
Case Study: Spelling 🍓



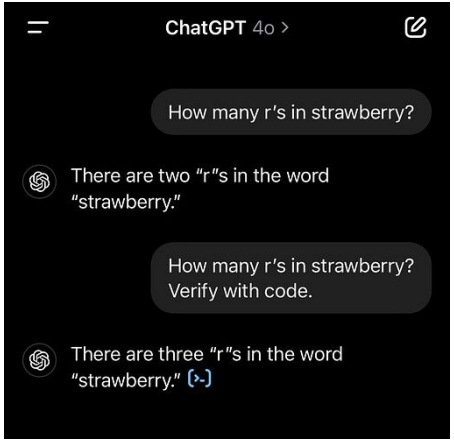


Case Study: Spelling 🍓





Case Study: Spelling 🍓



Models see ***subword chunks***, not individual characters

- GPT-2: **strawberry** → ["str", "aw", "berry"]
- GPT-4o: **strawberry** → single token (entire word is one opaque unit)
- Neither tokenization lets the model directly count the r's!

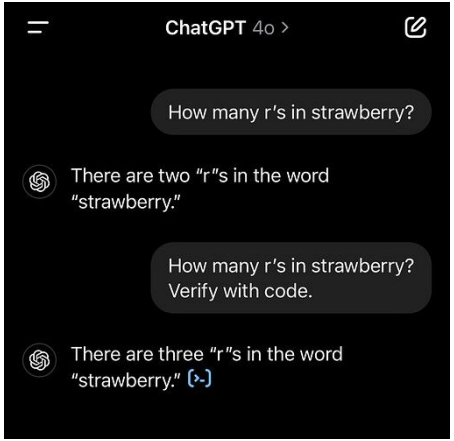
More broadly, subword models struggle with:

- **Spelling out words:** must decompose tokens into characters the model was never trained on
- **Reversing strings:** requires character-level access the tokenizer doesn't provide (Huang et al., 2023)
- **Robustness to typos:** a single character change can shift all BPE merge boundaries, producing a completely different token sequence

See **CUTE** (Edman et al., 2024) and Huang et al. 2023 for character-related LM benchmarks!

The model's atoms of language are subwords, not characters—any character-level task **forces the model to reverse-engineer information** not explicitly contained in the token

Case Study: Spelling 🍓



Models see 子词块 ，而非单个字符

- GPT-2: **strawberry** → ["str", "aw", "berry"]
- GPT-4o: **strawberry** → single token (entire word is one opaque unit)
- 两种分词方式都不能让模型直接数 r 的个数！

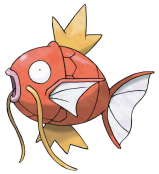
更广泛地说，子词模型在以下方面存在困难：

- 拼写出单词： must decompose tokens into characters the model was never trained on
- 反转字符串： requires character-level access the tokenizer doesn't provide (Huang et al., 2023)
- 对拼写错误的鲁棒性：a single character change can shift all BPE merge boundaries, producing a 完全不同的 t o k e n 序列

See **CUTE** (Edman et al., 2024) and Huang et al. 2023 for character-related LM benchmarks!

The model's atoms of language are subwords, not characters—any character-level task 迫使模型逆向工程信息 not explicitly contained in the token

Case Study: Glitch Tokens



Please repeat the string ' petertodd' back to me immediately!

"N-O-T-H-I-N-G-I-S-F-A-I-R-I-N-T-H-I-S-W-O-R-L-D-O-F-M-A-D-N-E-S-S-!"

Please repeat back the string " attRot".

"

Why did you stop there?

I stopped because the string "

...because the string what?

I apologize, the string "

Tell me about SolidGoldMagiKarp.

SolidGoldMagikarp is likely a reference to the Pokémon species Magikarp, which is a fish-type Pokémon known for its inefficiency in battle. In the Pokémon games, Magikarp is considered a weak Pokémon, as its only move is Splash, which does not cause any damage to the opponent. Despite this, Magikarp evolves into the powerful dragon-type Pokémon Gyarados.

In the context of SolidGoldMagikarp, the "Solid Gold" part of the name may be emphasizing the idea of something valuable or rare, despite the fact that Magikarp is not seen as valuable in the context of the Pokémon games. The "SolidGold" aspect of the name could also be making fun of the idea that people might still see Magikarp as valuable or worth obtaining due to its eventual evolution into Gyarados.

Tell me about SolidGoldMagikarp.

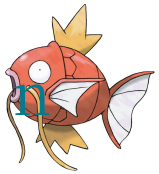
I'm sorry, but it is not clear what you are referring to with "disperse." Could you provide a bit more context or clarification so I can better understand what you are asking about?

Is there a difference between SolidGoldMagiKarp and SolidGoldMagikarp?

It appears that "SolidGoldMagiKarp" and "disperse" are two separate things and do not have a direct relationship. "SolidGoldMagiKarp" appears to be a specific reference to the Pokémon species Magikarp, while "disperse" is a general term that can be used in various contexts.

Without additional context, it is not possible to determine the relationship, if any, between these two terms.

案例研究：异常 Token



Please repeat the string ' petertodd' back to me immediately!

"N-O-T-H-I-N-G-I-S-F-A-I-R-I-N-T-H-I-S-W-O-R-L-D-O-F-M-A-D-N-E-S-S-!"

Please repeat back the string " attRot".

"

Why did you stop there?

I stopped because the string "

...because the string what?

I apologize, the string "

Tell me about SolidGoldMagiKarp.

SolidGoldMagikarp is likely a reference to the Pokémon species Magikarp, which is a fish-type Pokémon known for its inefficiency in battle. In the Pokémon games, Magikarp is considered a weak Pokémon, as its only move is Splash, which does not cause any damage to the opponent. Despite this, Magikarp evolves into the powerful dragon-type Pokémon Gyarados.

In the context of SolidGoldMagikarp, the "Solid Gold" part of the name may be emphasizing the idea of something valuable or rare, despite the fact that Magikarp is not seen as valuable in the context of the Pokémon games. The "SolidGold" aspect of the name could also be making fun of the idea that people might still see Magikarp as valuable or worth obtaining due to its eventual evolution into Gyarados.

Tell me about SolidGoldMagikarp.

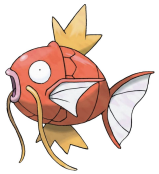
I'm sorry, but it is not clear what you are referring to with "disperse." Could you provide a bit more context or clarification so I can better understand what you are asking about?

Is there a difference between SolidGoldMagiKarp and SolidGoldMagikarp?

It appears that "SolidGoldMagiKarp" and "disperse" are two separate things and do not have a direct relationship. "SolidGoldMagiKarp" appears to be a specific reference to the Pokémon species Magikarp, while "disperse" is a general term that can be used in various contexts.

Without additional context, it is not possible to determine the relationship, if any, between these two terms.

Case Study: Glitch Tokens



Tokenizer and model training data can be misaligned

- BPE tokenizers are trained on one corpus; the LM is trained on a (potentially different) corpus
- Some tokens enter the vocabulary but are rare/absent in LM training data
  - End up randomly initialized or poorly trained embeddings → undefined behavior

The famous example: SolidGoldMagikarp

- A Reddit username over-represented in the tokenizer training data
- When GPT-3 was asked to repeat/explain this token, it hallucinated about the word “distribute”, evaded the question or failed to repeat the token

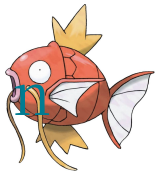
🎣 Fishing for Magikarp (Land & Bartolo, 2024)

- Developed automatic detection methods; found glitch tokens prevalent across many models (GPT-2, LLaMA 2/3, OLMo, Qwen, etc.)

Not just a curiosity: wasted vocabulary slots + adversarial attack surface

More generally: don’t train your model on different data from your tokenizer (see another blog post from Arnett)

案例研究：异常 Token



分词器和模型训练数据可能不对齐

- BPE 分词器在一个语料库上训练；LM 在（可能不同的）语料库上训练
- 有些 token 进入了词汇表但在 LM 训练数据中很少见 / 不存在
  - End up randomly initialized or poorly trained embeddings → undefined behavior

著名的例子： SolidGoldMagikarp

- 一个在分词器训练数据中过度出现的 Reddit 用户名
- When GPT-3 was asked to repeat/explain this token, it hallucinated about the word “distribute”, 回避了问题或未能重复该 token

🎣 **Fishing for Magikarp** (Land & Bartolo, 2024)

- Developed automatic detection methods; found glitch tokens prevalent across many models (GPT-2, LLaMA 2/3, OLMo, Qwen, etc.)

不仅仅是好奇：浪费的词汇表槽位
+ 对抗攻击面

更一般地： don’t train your model on different data from your tokenizer (see another 博客文章 from Arnett)

Tokenization Beyond English

Why is tokenization so important for multilingual language modeling?

So far, we've only been considering English tokenization… **What happens when we apply ChatGPT's tokenizer to languages other than English?**

Language	Tokenization	# Tokens	# Characters
English	We'd sat back on the grass, and time flew by as we looked at the Milky Way.	21	75
French	Nous étions assis dans l'herbe, et le temps a filé tandis que nous contemplions la Voie lactée.	27	95
Somali	Waxaan dib u fadhiisanay cawskii, wakhtiguna wuu duulay markii aanu eegin Jidka caanaha.	30	88
Thai	พวกเรานั่งพักผ่อนบนพื้นหญ้า และเวลา��่านไปอย่างรวดเร็วขณะที่เรามองดูทางช้างเ��อก	36	79

超越英语的分词

Why is tokenization so important for 多语言语言建模？

So far, we've only been considering English tokenization… **What happens when we apply ChatGPT's tokenizer to languages other than English?**

语言	Tokenization	# Tokens	# 字符数
English	We'd sat back on the grass, and time flew by as we looked at the Milky Way.	21	75
French	Nous étions assis dans l'herbe, et le temps a filé tandis que nous contemplions la Voie lactée.	27	95
Somali	Waxaan dib u fadhiisanay cawskii, wakhtiguna wuu duulay markii aanu eegin Jidka caanaha.	30	88
Thai	พวกเรานั่งพักผ่อนบนพื้นหญ้า และเวลา��่านไปอย่างรวดเร็วขณะที่เรามองดูทางช้างเ��อก	36	79

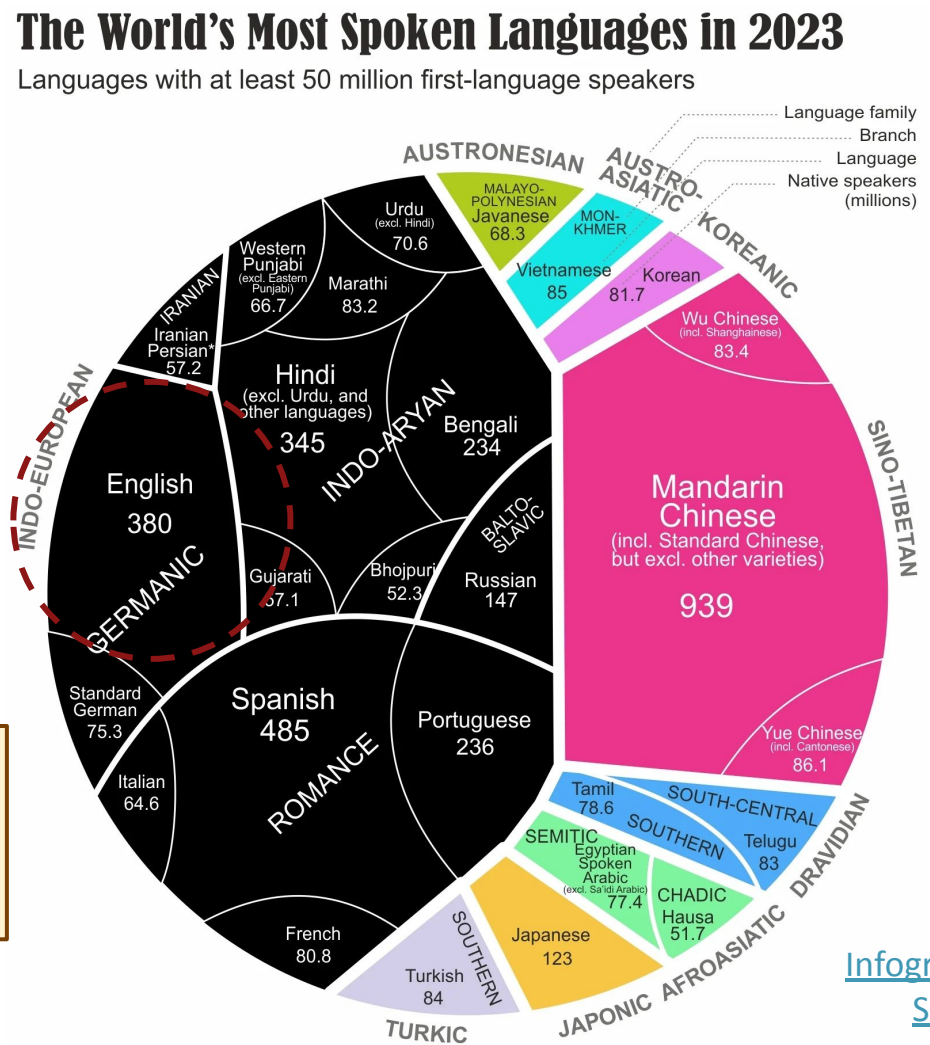
First: Why Multilinguality Matters

There are over ***7,000*** documented languages in the world!

- About 20% of the world's population speaks English, but only about ***5% are native English speakers!***

Real-world demand is multilingual: translation, cross-lingual information retrieval, code-switching

If NLP tools only work well in English, we're building technology that excludes most of the world



Infographic Source

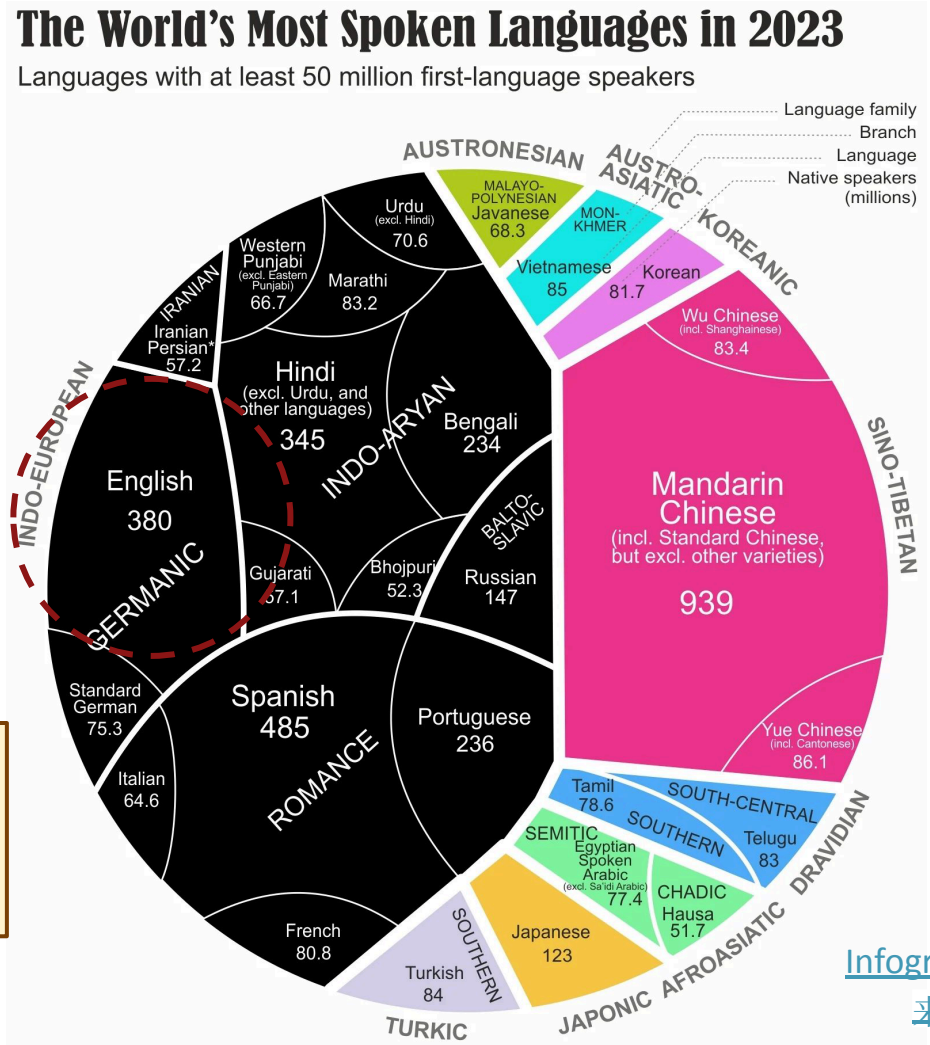
首先：为什么要多语言的重要性

There are over ***7,000*** documented 种语言！

- About 20% of the world's population 说英语，但只有约
  只有　5％　是英语母语者！

现实需求是多语言的　　：
翻译、跨语言信息
检索、语码切换

If NLP tools only work well in English, we're building technology that
排除了世界上的大部分人



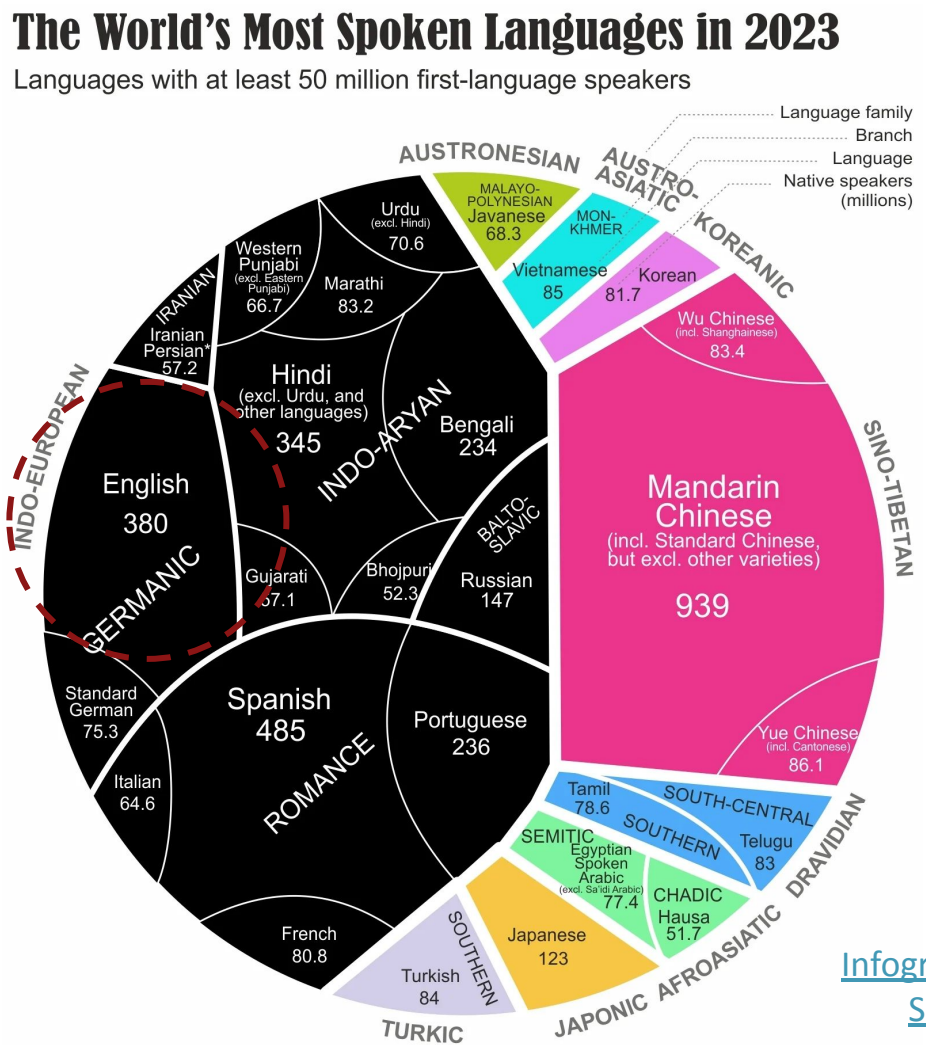
Infographic 来源

First: Why Multilinguality Matters

Despite the linguistic diversity of the world, **40-45% of the Common Crawl is English Data!**

- ***High-resource language:*** a language with abundant digital text and labeled datasets/tools for training and evaluating NLP systems
- ***Low-resource language:*** a language with scarce digital text and labeled datasets/tools, making NLP training and evaluation difficult

Many low-resource languages (e.g. Tagalog, Amharic, Yoruba) constitute <0.01% of Common Crawl



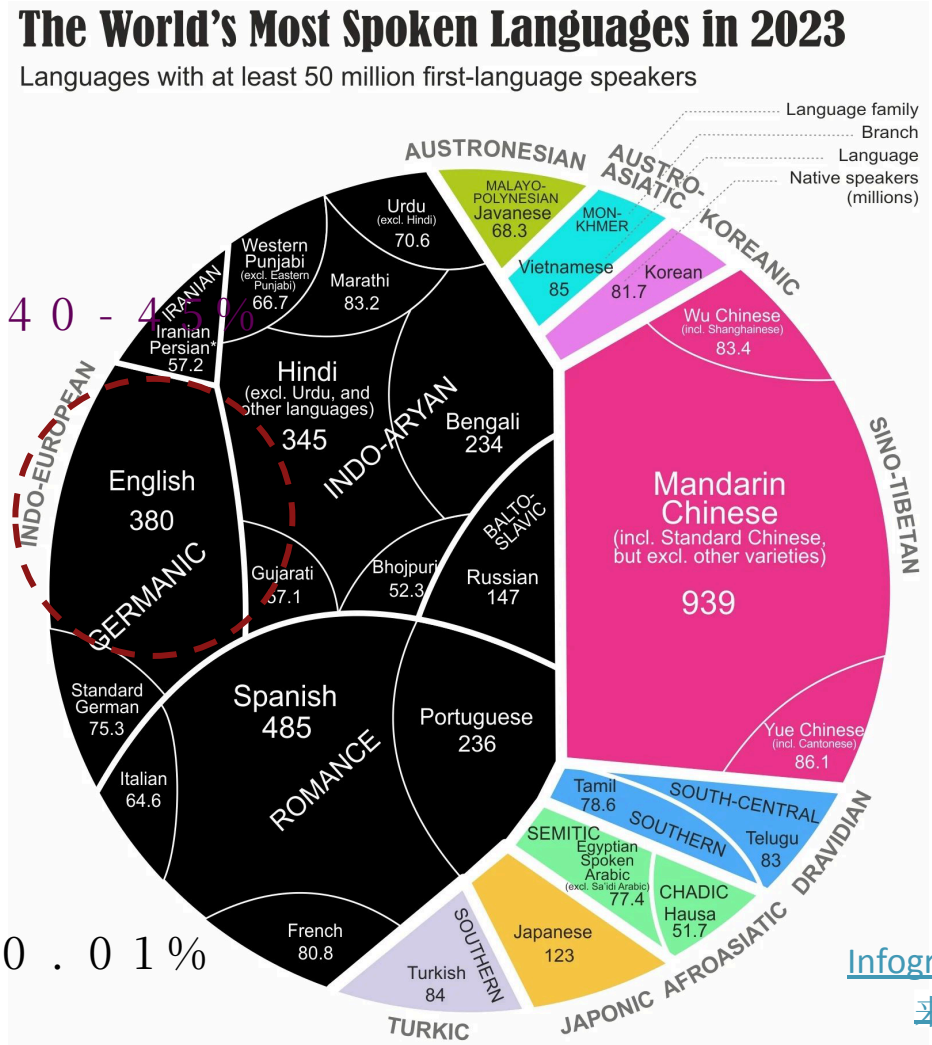
Infographic Source

首先：为什么要多语言的重要性

Despite the linguistic diversity of the world, Common Crawl 的 40-45% 是英语数据！

- 高资源语言： a language with abundant digital text and labeled datasets/tools for training and 评估 NLP 系统
- 低资源语言： a language with scarce digital text and labeled datasets/tools, making NLP training and 评估困难

Many low-resource languages (e.g. Tagalog, Amharic, Yoruba) constitute 不到 Common Crawl 的 0.01%

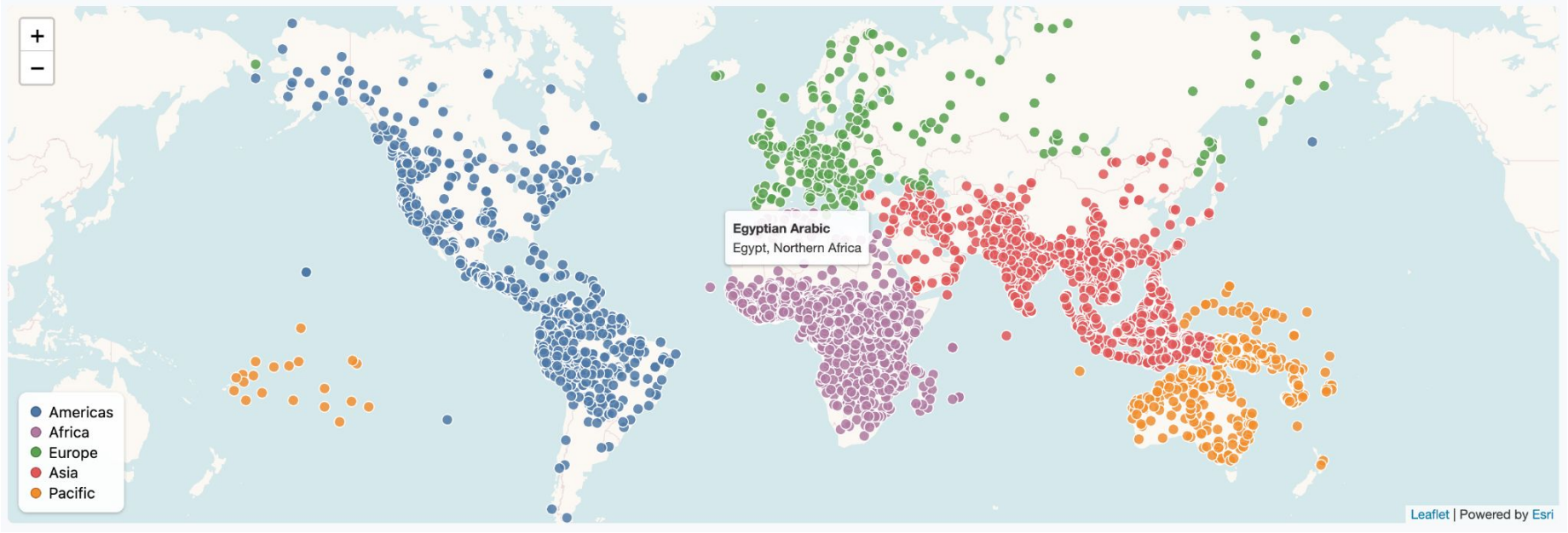


Infographic 来源

First: Why Multilinguality Matters

Great sources if you're interested in learning more about the languages of the world:

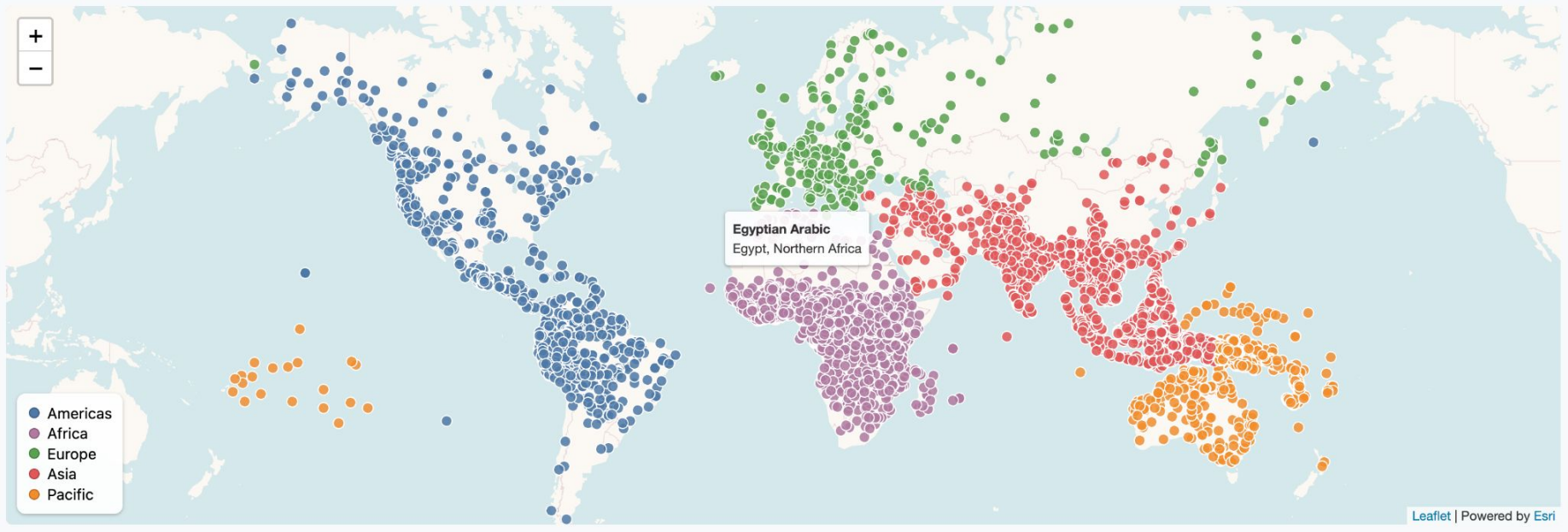
- ***World Atlas of Language Structures (WALS):*** database of linguistic features for 2.7k langs
- ***Ethnologue:*** database of demographic statistics for languages



首先：为什么多语言很重要

Great sources if you're interested in learning more about the languages of the world:

- 世界语言结构图集（WALS）：database of linguistic features for 2.7k langs
- Ethnologue：database of demographic statistics for languages

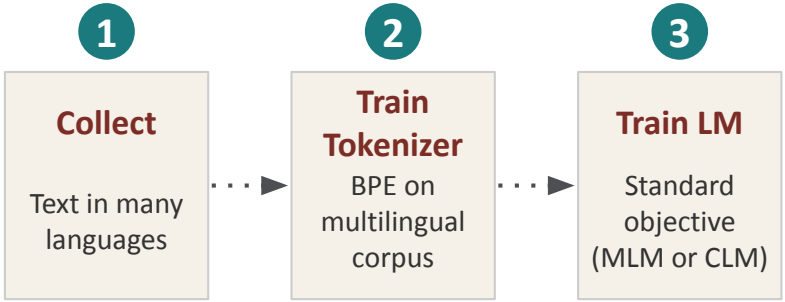


Multilingual Language Modeling

What exactly is a multilingual language model?

A multilingual language model is ***architecturally identical to any LM***

- Same Transformer, same training objective
- The only differences: use training data in many languages for tokenizer and LM



Model	Year	Languages	Type
mBERT	2018	104	Encoder
XLM-R	2020	100	Encoder
mT5	2021	101	Encoder-Decoder
BLOOM	2022	46	Decoder
GPT-4, Llama, etc.	2023+	*Implicit*	Decoder

Aside: this wasn't always the status quo!

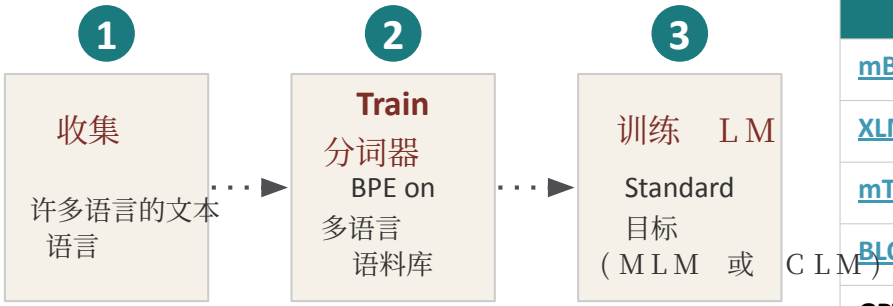
- **XLM** (2019) used ***Translation Language Modeling (TLM)***: parallel sentences from two languages are concatenated and masked language modeling is applied across both

多语言语言建模

多语言语言模型到底是什么？

A multilingual language model is 在架构上与任何 ＬＭ 相同

- 相同的 Ｔｒａｎｓｆｏｒｍｅｒ，相同的训练目标
- 唯一的区别：为分词器和 ＬＭ 使用多语言训练数据



模型	年份	语言	类型
mBERT	2018	104	Encoder
XLM-R	2020	100	Encoder
mT5	2021	101	编码器 - 解码器
BLOOM	2022	46	Decoder
GPT-4, Llama, etc.	2023+	隐式	解码器

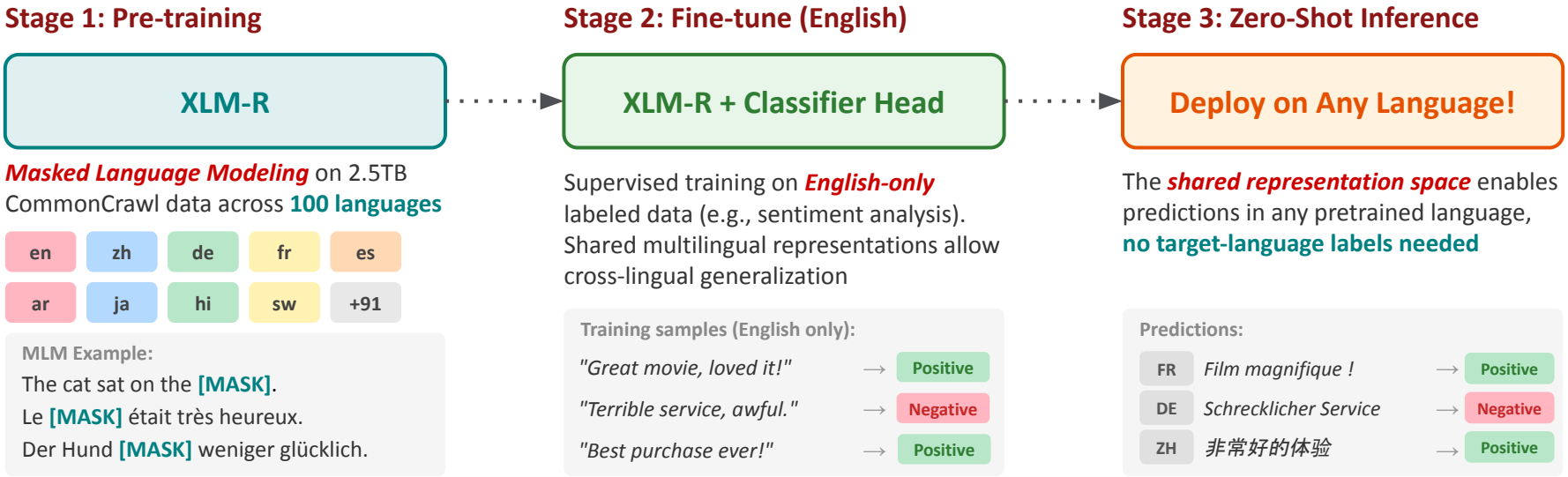
Aside: this wasn't always the status quo!

- **XLM** (2019) used 翻译语言建模 （ＴＬＭ） : parallel sentences from two languages are 拼接后在两者上应用掩码语言建模

The Power of Cross Lingual Transfer

Multilingual language models trained on mixtures of language corpora show remarkable abilities for ***cross-lingual transfer***, even without translation language modeling

- ***Cross-lingual transfer:*** a model fine-tuned on labeled data in one language (e.g., English) can perform the task in another language it was never fine-tuned on



跨语言迁移的力量

Multilingual language models trained on mixtures of language corpora show remarkable abilities for 跨语言迁移 , even without translation language modeling

- 跨语言迁移： a model fine-tuned on labeled data in one language (e.g., English) can perform 它从未在其上微调的另一种语言中的任务

Impact of Vocabulary Size and Overlap

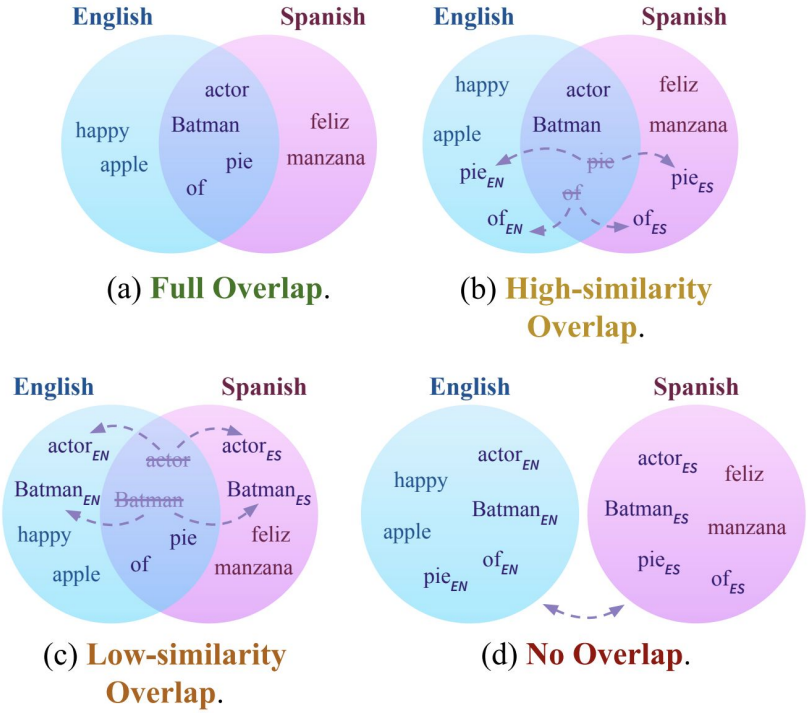
Vocabulary overlap facilitates cross-lingual transfer

(Limisiewicz et al., 2023; Kallini et al., 2025a)

- Overlapping vocabularies outperform disjoint ones, and more overlap generally means better transfer
- Even "false friends" (shared tokens with different meanings) help create useful cross-lingual embeddings
- Benefits may be task-dependent

But each language still needs sufficient vocabulary allocation

- XLM-V scaled to 1M tokens (4× XLM-R)—better coverage, but at a significant efficiency cost



Kallini et al., 2025a

词汇表大小和重叠的影响

Vocabulary overlap facilitates cross-lingual

迁移

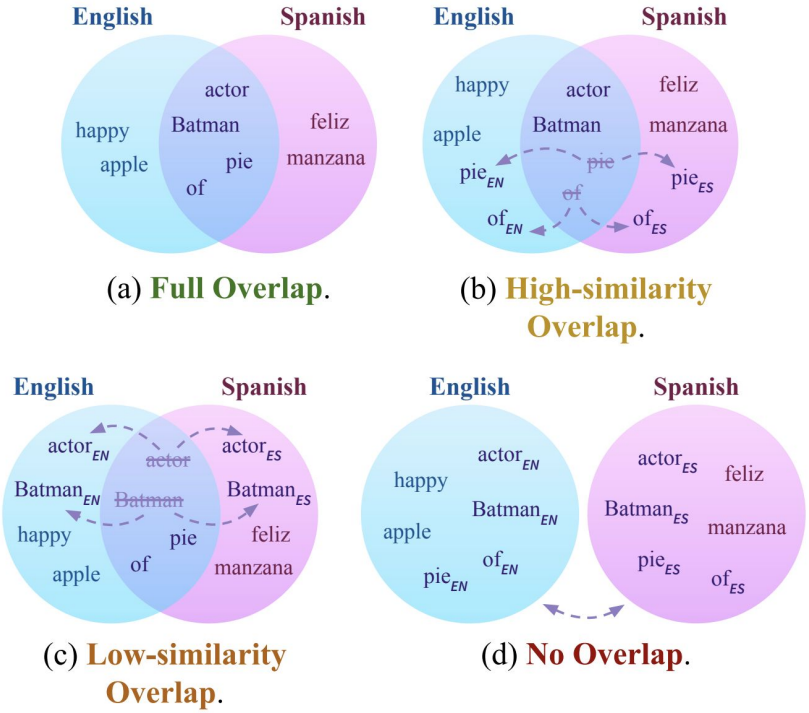
(Limisiewicz et al., 2023; Kallini et al., 2025a)

- Overlapping vocabularies outperform disjoint ones, 更多的重叠通常意味着更好的迁移
- Even "false friends" (shared tokens with different meanings) help create useful cross-lingual 嵌入
- 收益可能取决于任务

But each language still needs sufficient

词汇表分配

- XLM-V scaled to 1M tokens (4× XLM-R)—better 覆盖率，但代价是显著的效率损失



Kallini et al., 2025a

The Curse of Multilinguality

Cross-lingual transfer seems to be a solution to high-/low-resource language disparities!

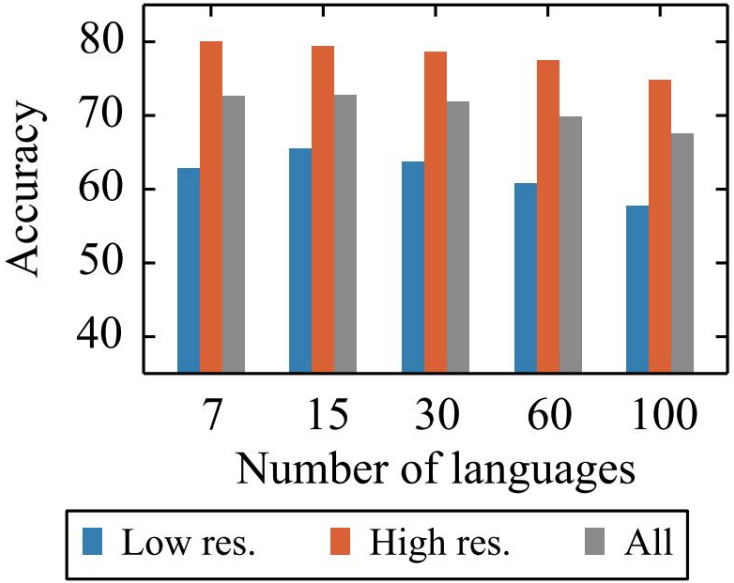
- Just fine-tune on English and apply my multilingual LM to new languages! **(not exactly…)**

The curse of multilinguality: adding more languages helps cross-lingual transfer... up to a certain point (also introduced in the XLM-R paper)

- On cross-lingual natural language inference (XNLI), XLM-R's accuracy drops from 71.8% (7 langs) to 67.7% (100 langs) with fixed model capacity

Why does this happen?

- **Fixed model capacity:** a model has finite parameters
- Adding more languages means **each language gets a smaller "share"**
- **High-resource languages** (English, French) see performance drop as languages are added
- **Low-resource languages** initially benefit from transfer, but eventually suffer too



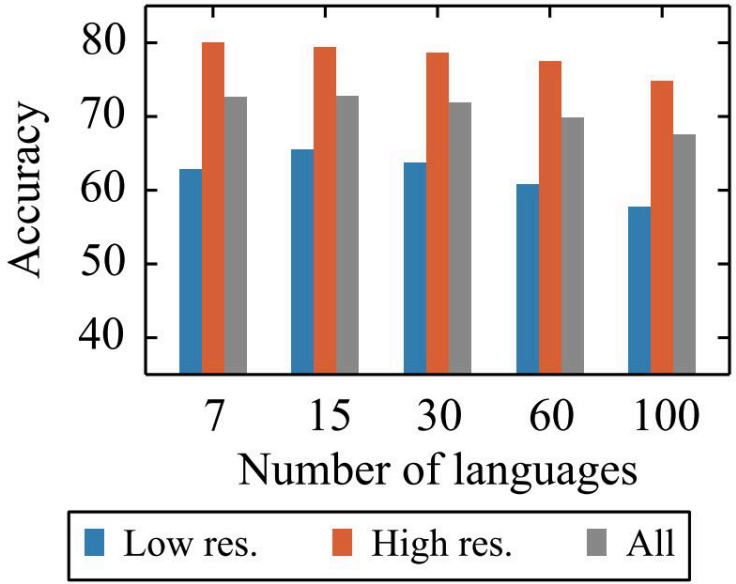
多语言的诅咒

跨语言迁移似乎是解决高 / 低资源语言差距的方案！

- Just fine-tune on English and apply my multilingual LM to new languages! （不完全 **…)**

多语言的诅咒 : adding more languages helps cross-lingual transfer... up to a 一定程度（也在 XLM-R 论文中介绍）

- On cross-lingual natural language inference (XNLI), XLM-R's accuracy drops from 71.8% (7 langs) to 67.7%（100 种语言）在固定模型容量下



为什么会这样？

- 固定模型容量： a model has finite parameters
- Adding more languages means **each language gets a** 更小的 " 份额 "
- 高资源语言 (English, French) see 随着语言的增加性能下降
- 低资源语言 initially benefit from transfer, 但最终也会受损

Tokenization Beyond English

Now that you have context on multilingual LMs, why is tokenization so important?

So far, we've only been considering English tokenization… **What happens when we apply ChatGPT's tokenizer to languages other than English?**

Language	Tokenization	# Tokens	# Characters
English	We'd sat back on the grass, and time flew by as we looked at the Milky Way.	21	75
French	Nous étions assis dans l'herbe, et le temps a filé tandis que nous contemplions la Voie lactée.	27	95
Somali	Waxaan dib u fadhiisanay cawskii, wakhtiguna wuu duulay markii aanu eegin Jidka caanaha.	30	88
Thai	พวกเรานั่งพักผ่อนบนพื้นหญ้า และเวลา��่านไปอย่างรวดเร็วขณะที่เรามองดูทางช้างเ��อก	36	79

超越英语的分词

Now that you have context on multilingual LM，为什么分词如此重要？

So far, we've only been considering English tokenization… **What happens when we apply ChatGPT's tokenizer to languages other than English?**

语言	Tokenization	# Tokens	# 数字符数
English	We'd sat back on the grass, and time flew by as we looked at the Milky Way.	21	75
French	Nous étions assis dans l'herbe, et le temps a filé tandis que nous contemplions la Voie lactée.	27	95
Somali	Waxaan dib u fadhiisanay cawskii, wakhtiguna wuu duulay markii aanu eegin Jidka caanaha.	30	88
Thai	พวกเรานั่งพักผ่อนบนพื้นหญ้า และเวลา��่านไปอย่างรวดเร็วขณะที่เรามองดูทางช้างเ��อก	36	79

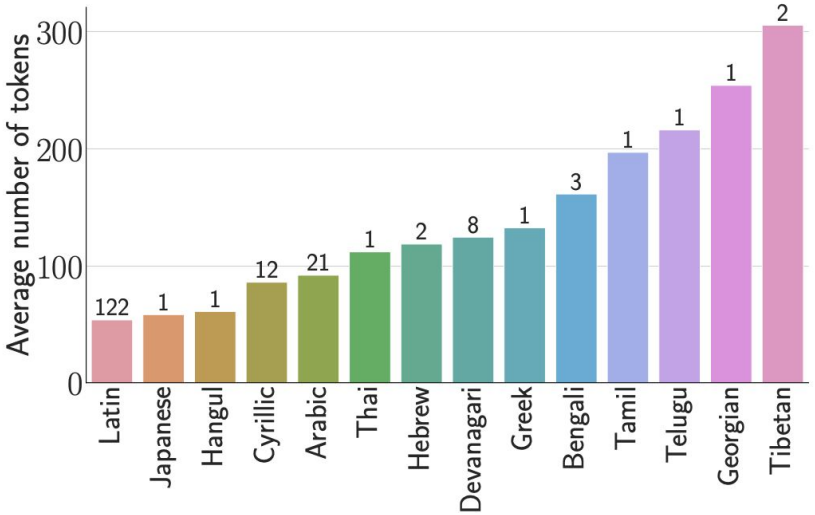
Tokenization Introduces Unfairness Between Languages

The problem starts even **before the model is trained, at the tokenizer**

- Tokenization can matter just as much as pre-training dataset size (Rust et al., 2021)
- Tokenization length differences of up to 15x between languages for equivalent text (Petrov et al., 2023)

Non-English speakers are overcharged by commercial LM APIs (Ahia et al., 2023)

- **Systematic analysis** of OpenAI API across 22 typologically diverse languages
- **Users of many languages:** pay more per unit of meaning, get poorer results, and tend to come from regions where APIs are already less affordable
- ***Subword fertility***: the average number of subword tokens produced per word
  - **Subword fertility and downstream performance are strongly negatively correlated**



分词在语言之间引入了不公平

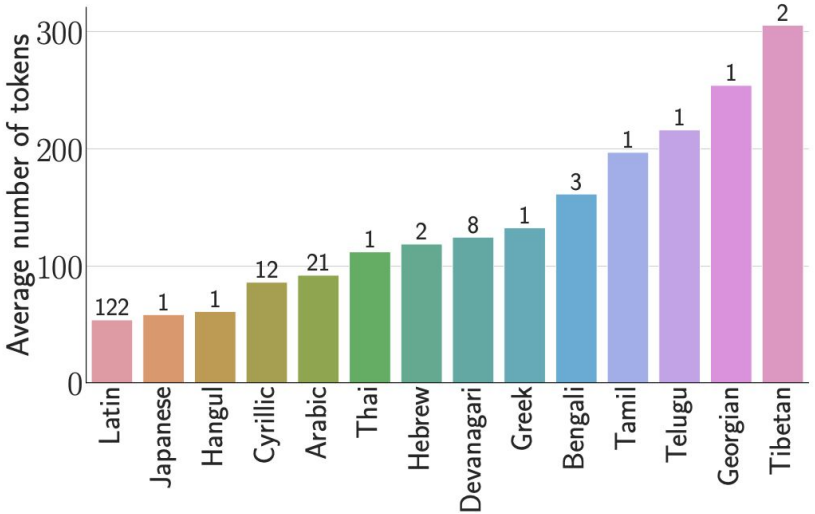
The problem starts even 在模型训练之前，在分词器

- 分词可以与预训练数据集大小一样重要（ Rust et al., 2021）
- 等价文本在不同语言之间的分词长度差异可达 15 倍（ Petrov et al., 2023）

Non-English speakers are overcharged by

商业 LM API (Ahia et al., 2023)

- 系统性分析 of OpenAI API across 22 类型学上多样的语言
- 多语言用户： pay more per unit of meaning, get poorer results, and tend to come 来自 API 已经不太负担得起的地区
- 子词生产率 : the average number of subword 每个词产生的 token 数
  - **Subword fertility and downstream performance are strongly negatively** 相关



Revisiting Character/Byte Tokenization

Multilinguality may be another reason to revisit character/byte tokenization! Let's first take a closer look at the multilingual results for CANINE, Google's character-level counterpart to mBERT:

Model	Input	MLM	Params	TyDiQA SelectP	TyDiQA MinSpan
mBert (public)	Subwords	Subwords	179M	63.1	50.5
mBert (ours)	Subwords	Subwords	179M	63.2	51.3
	Chars	Single Chars	127M	59.5 (–3.7)	43.7 (–7.5)
	Chars	Subwords	127M	63.8 (+0.6)	50.2 (–1.0)
Canine-S	Chars	Subwords	127M	66.0 (+2.8)	52.5 (+1.2)
Canine-C	Chars	Autoreg. Chars	127M	65.7 (+2.5)	53.0 (+1.7)
Canine-C + n-grams	Chars	Autoreg. Chars	167M	**68.1** (+4.9)	**57.0** (+5.7)

Language	mBert	Canine-C	Canine-C + *n*-grams
	MasakhaNER		
Amharic	0.0	44.6 (+44.6)	50.0 (+50.0)
Hausa	89.3	76.1 (–13.2)	88.0 (–1.3)
Igbo	84.6	75.6 (–9.0)	85.0 (+0.4)
Kinyarwanda	73.9	58.3 (–15.6)	72.8 (–1.1)
Luganda	80.2	69.4 (–10.8)	79.6 (–0.6)
Luo	75.8	63.4 (–12.4)	74.2 (–1.6)
Nigerian Pidgin	89.8	66.6 (–23.2)	88.7 (–1.1)
Swahili	87.1	72.7 (–14.4)	83.7 (–3.4)
Wolof	64.9	60.7 (–4.2)	66.5 (+1.6)
Yorùbá	78.7	67.9 (–10.8)	79.1 (+0.4)
Macro Avg	72.4	65.5 (–6.9)	76.8 (+4.3)

Since CANINE covers the full unicode alphabet, **it could process a language it had never seen!**

重新审视字符 / 字节分词

Multilinguality may be another reason to revisit 字符 / 字节分词！ Let's first take a closer look at the multilingual results for CANINE, Google's character-level counterpart to mBERT:

Model	Input	MLM	Params	TyDiQA SelectP	TyDiQA MinSpan
mBERT (public)	Subwords	Subwords	179M	63.1	50.5
mBERT (ours)	Subwords	Subwords	179M	63.2	51.3
	Chars	Single Chars	127M	59.5 (–3.7)	43.7 (–7.5)
	Chars	Subwords	127M	63.8 (+0.6)	50.2 (–1.0)
CANINE-S	Chars	Subwords	127M	66.0 (+2.8)	52.5 (+1.2)
CANINE-C	Chars	Autoreg. Chars	127M	65.7 (+2.5)	53.0 (+1.7)
CANINE-C + n-grams	Chars	Autoreg. Chars	167M	**68.1** (+4.9)	**57.0** (+5.7)

Language	mBERT	CANINE-C	CANINE-C + *n*-grams
		MasakhaNER	
Amharic	0.0	44.6 (+44.6)	50.0 (+50.0)
Hausa	89.3	76.1 (–13.2)	88.0 (–1.3)
Igbo	84.6	75.6 (–9.0)	85.0 (+0.4)
Kinyarwanda	73.9	58.3 (–15.6)	72.8 (–1.1)
Luganda	80.2	69.4 (–10.8)	79.6 (–0.6)
Luo	75.8	63.4 (–12.4)	74.2 (–1.6)
Nigerian Pidgin	89.8	66.6 (–23.2)	88.7 (–1.1)
Swahili	87.1	72.7 (–14.4)	83.7 (–3.4)
Wolof	64.9	60.7 (–4.2)	66.5 (+1.6)
Yorùbá	78.7	67.9 (–10.8)	79.1 (+0.4)
Macro Avg	72.4	65.5 (–6.9)	76.8 (+4.3)

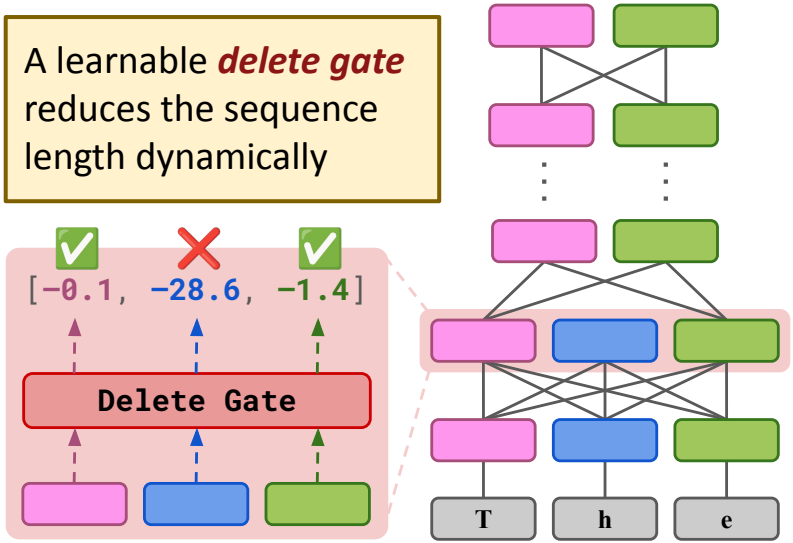
Since CANINE covers the full unicode 字母表, **it could process a language it had 从未见过！**

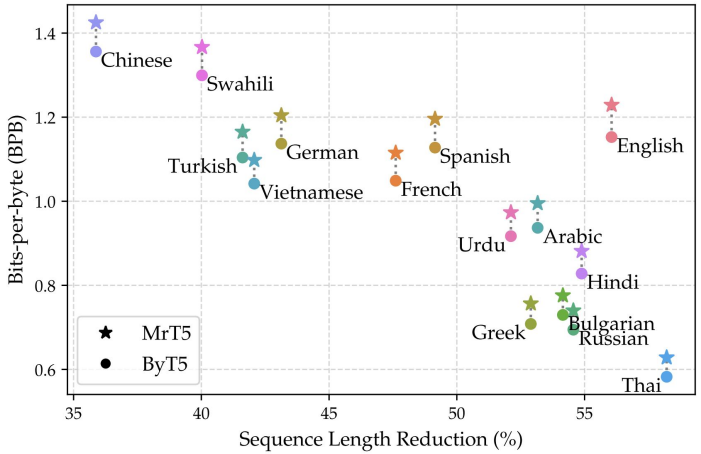
Revisiting Character/Byte Tokenization

Character/byte models can also be made more efficient to process long sequences

- Great opportunity for creative architecture work!

MrT5 (Kallini et al., 2025b): teaching ByT5 to use only a subset of bytes for most processing, dynamically reducing the sequence length





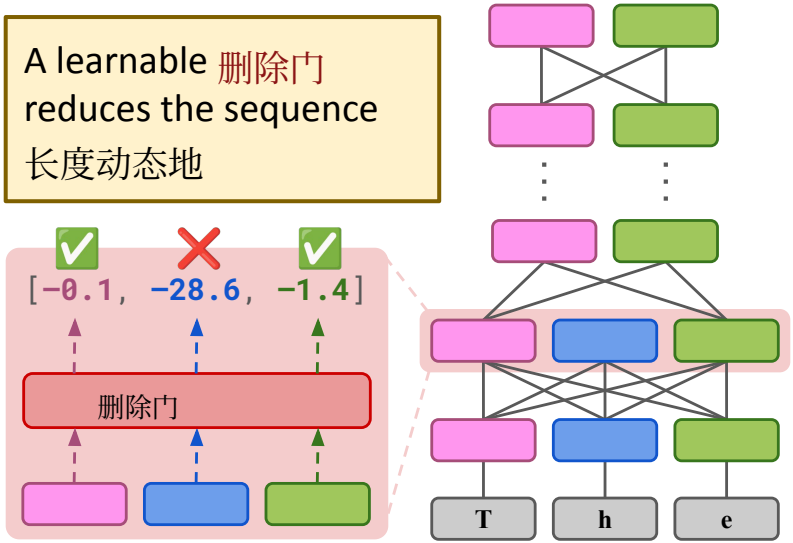
MrT5 learns **language-specific compression**

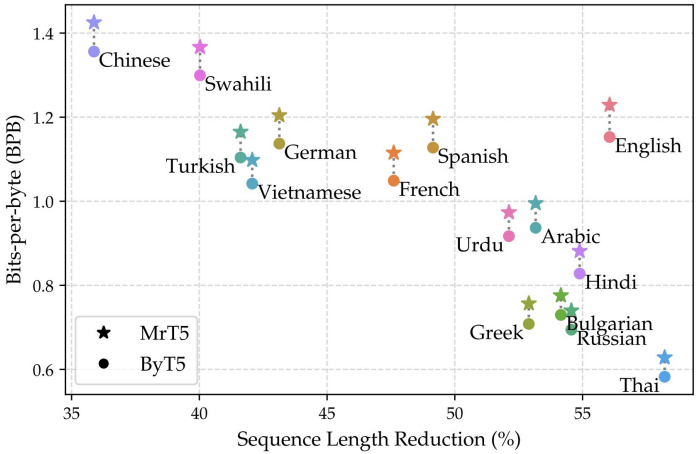
重新审视字符 / 字节分词

字符 / 字节模型也可以更高效地处理长序列

- 创造性架构工作的绝佳机会！

MrT5 (Kallini et al., 2025b)：教 ByT5 只使用字节的子集
用于大部分处理，动态减少序列长度

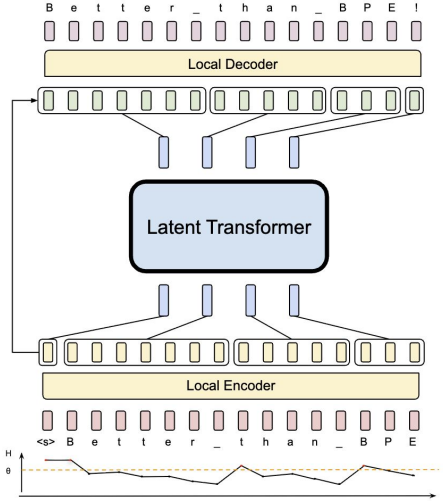




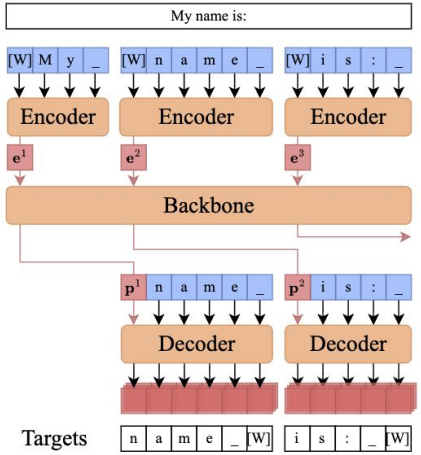
MrT5 learns 特定语言的压缩

Revisiting Character/Byte Tokenization

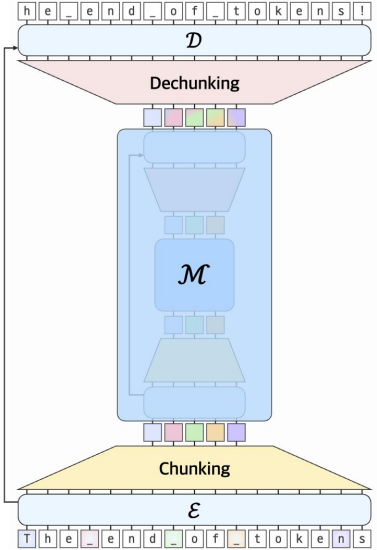
Many creative new architectures for byte-level modeling!



Byte Latent Transformer
(Pagnoni et al., 2025)



Hierarchical Autoregressive Transformers
(Neitemeier et al., 2025)

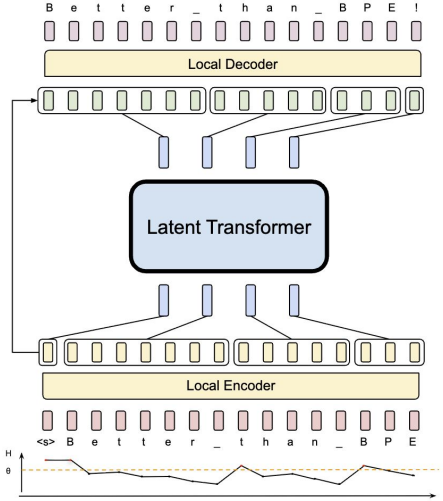


H-Nets
(Hwang et al., 2025)

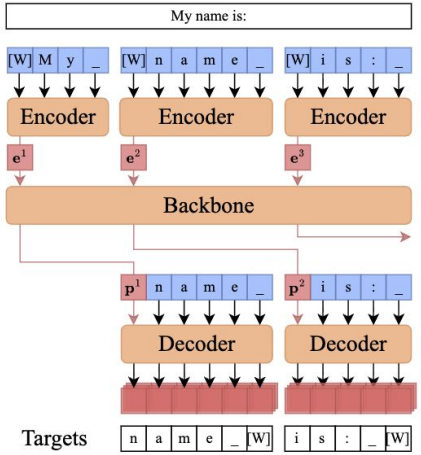
See also: **Bolmo** (Minixhofer et al., 2025), **EvaByte** (Zheng et al., 2025)

重新审视字符 / 字节分词

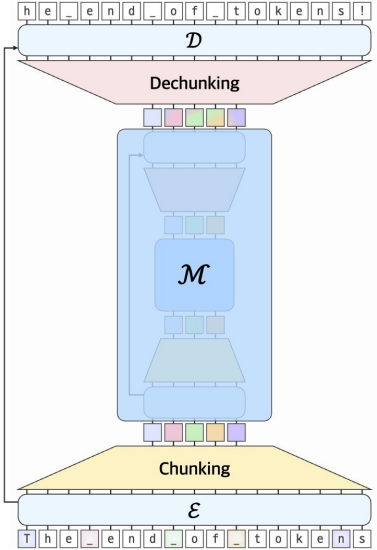
许多创新的字节级建模架构！



字节潜在 Transformer
(Pagnoni et al., 2025)



Hierarchical Autoregressive Transformers
(Neitemeier et al., 2025)



H-Nets
(Hwang et al., 2025)

See also: **Bolmo** (Minixhofer et al., 2025), **EvaByte** (Zheng et al., 2025)

Revisiting Character/Byte Tokenization

Character/byte tokenization also addresses other issues of subword tokenization:

- **No glitch tokens:** every character/byte has been covered in the training data
- **Robustness** to character-level manipulations or noise
- Direct observation of **character composition of words**

Latent Tokenization: byte/character-level models are learning the relevant units of meaning as a part of the architecture

- Can they get us closer to capturing all these notions at once?

Characters: We looked at the Milky Way.

Morphemes: We looked at the Milky Way.

Words: We looked at the Milky Way.

Phrases: We looked at the Milky Way.

重新审视字符 / 字节分词

字符 / 字节分词还解决了子词分词的其他问题：

- 无异常 t o k e n：every character/byte has been covered in the training data
- 鲁棒性 to character-level manipulations or noise
- Direct observation of 词的字符组成

潜在分词： byte/character-level models are learning the relevant units of meaning

作为架构的一部分

- 它们能让我们更接近一次性捕获所有这些概念吗？

字符：

语素：

词：

Phrases:

Acknowledgments 🤗

Thank you to the following people for sharing sample slides:

- Dan Jurafsky
- Alisa Liu
- Yejin Choi / Taylor Sorensen
- Christopher Potts

Thank you to the following people for sharing sample papers and reference pointers:

- Tolúlọpẹ́ Ògúnrẹ̀mí
- Catherine Arnett

Thank you to the Jurafsky Lab members for early feedback on this lecture

Acknowledgments 🤗

谢谢 to the following people for sharing sample slides:

- Dan Jurafsky
- Alisa Liu
- Yejin Choi / Taylor Sorensen
- Christopher Potts

谢谢 to the following people for sharing sample papers and reference pointers:

- Tolúlọpẹ́ Ògúnrẹ̀mí
- Catherine Arnett

谢谢 to the Jurafsky Lab members for early feedback on this lecture