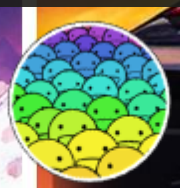
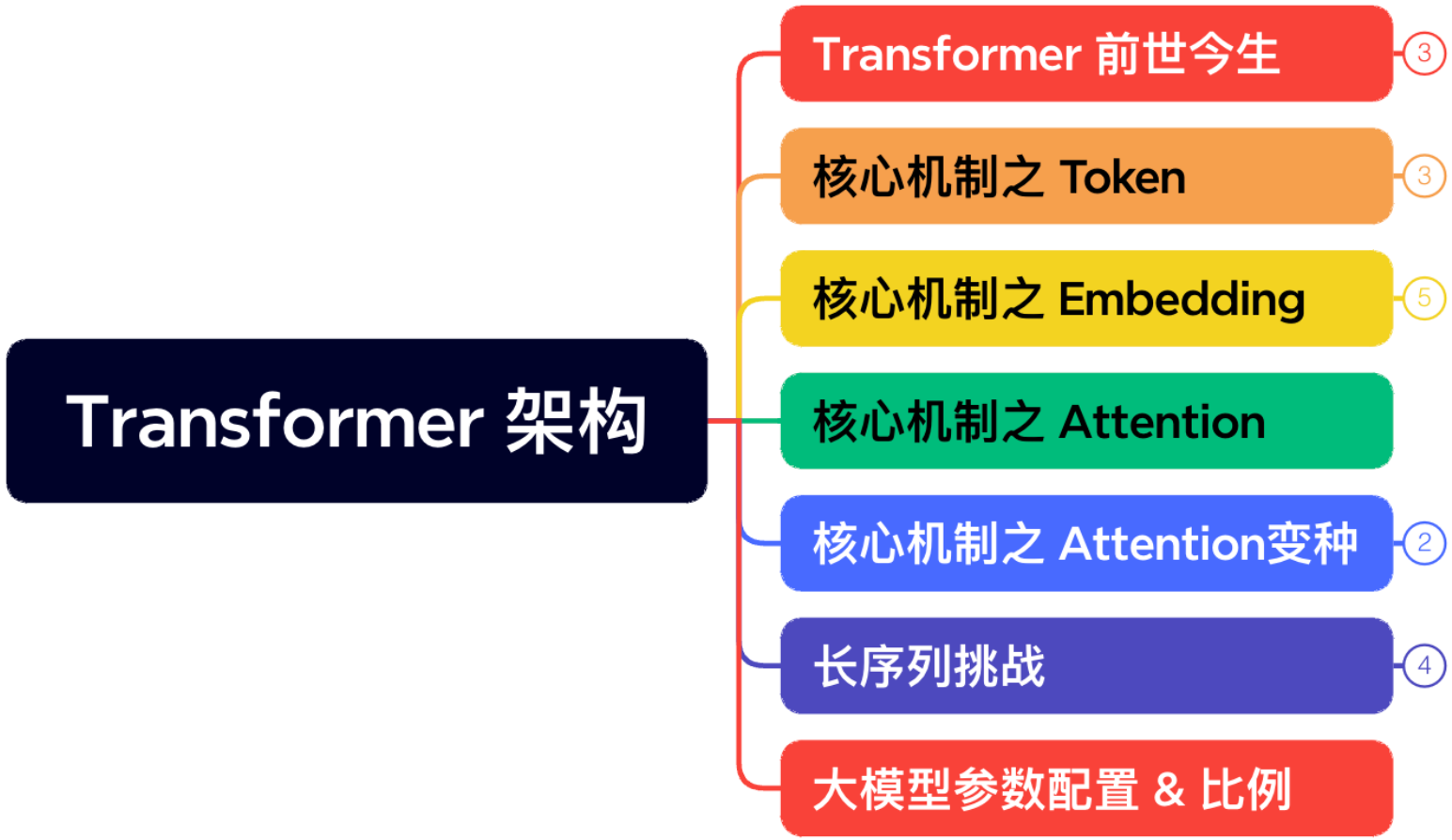



核心机制之 Tokenizer

ZOMI

Contenet


Transformer 架构
Transformer 前世今生 3
核心机制之 Token 3
核心机制之 Embedding 5
核心机制之 Attention
核心机制之 Attention变种 2
长序列挑战 4
大模型参数配置 & 比例


好好想想

思考

- 到底 Token 和 Tokenizer 是什么关系?
- 为什么 Transformer 模型需要先对文本进行 Tokenization?
- 为什么单词直接按空格，或者直接分词不够用?
  - Using/Huawei/Ascend/chips/anywhere/ in/the/world/violates/US/export/control/regulations
  - 在/世界/任何/地方/使用/华为/昇腾/芯片/均/违反/美国/出口/管制/法规

Content

1. Tokenizer 的基本概念
2. Tokenizer 的完整流程
3. 分词算法深度解析
4. Tokenizer 评估指标
5. 演示：Tokenizer 实践

01

Tokenizer 基本概念

什么是Token？Token aka 词元

- 语言模型为何需要将文本转化为数字
- ASCII编码与Token化的本质区别：从单字符映射到语义单元

Token 和单词的关系

- 一个 Token 能代表几个单词和汉字?

英文 Token：单词的“拼图”

- 一个 Token 大约对应 0.75 个单词 或 3-4 个字母。
  - “unhappiness” 可能被切成 “un”、“happi”、“ness” 三个 Token，仿佛在玩拼图游戏。
  - OpenAI 官方称，1000 个 Token 大约能装下 750 个英文单词，相当于一篇短篇小说的开头。

中文 Token：汉字的“打包”

- 在中文中，一个 Token 通常对应 1 到 1.8 个汉字。
  - “你好，世界！” 可能被切成 [‘你’，‘好’，‘，’，‘世’，‘界’，‘！’]，共 6 个 Token。

Tokenization 核心任务

- 将文本映射为ID序列，供模型处理，作为 Transformer 结构的输入。

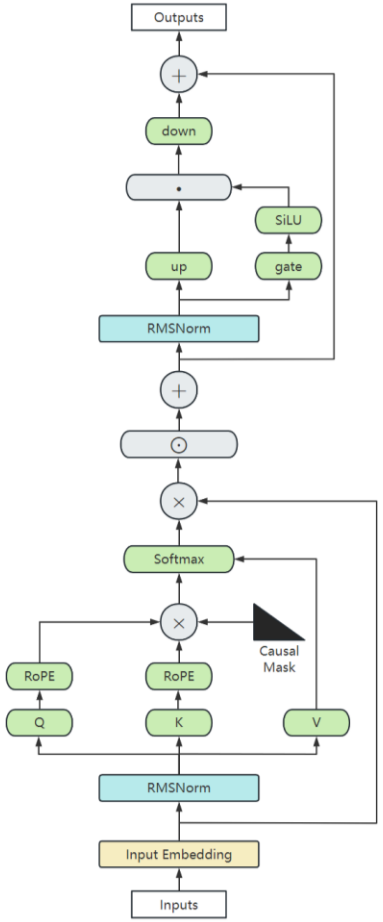
"This is a input text."

Tokenization

[CLS]	This	is	a	input	.	[SEP]
101	2023	2003	1037	7953	1012	102

Embeddings

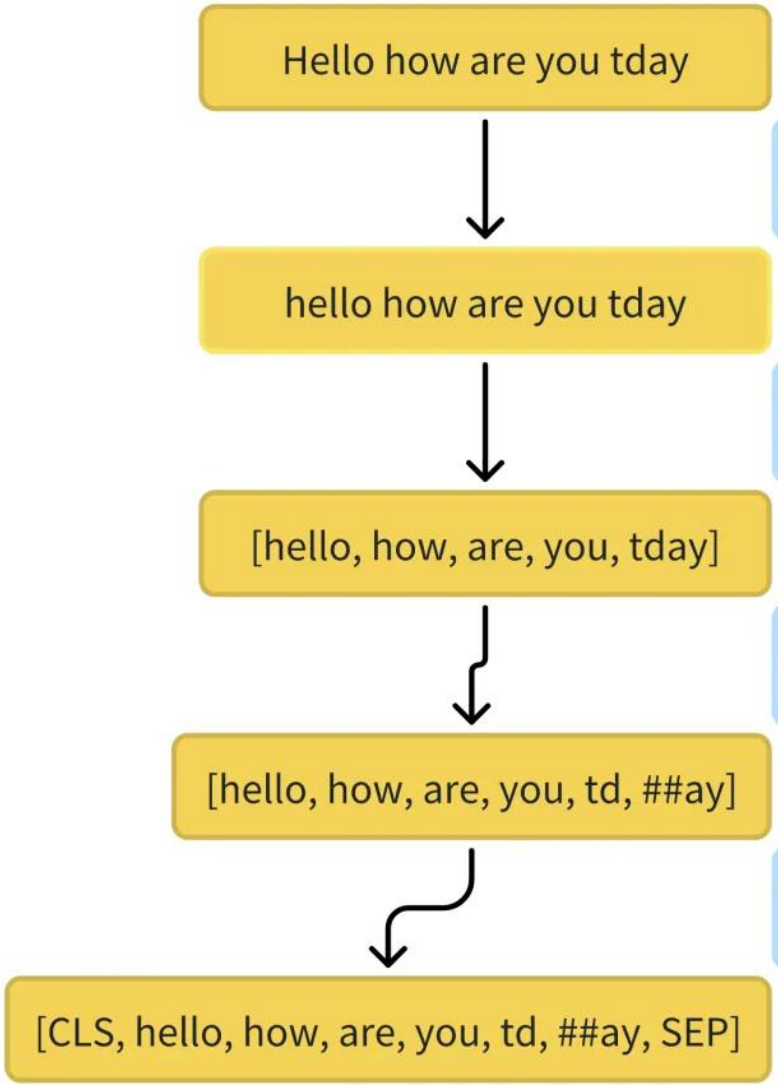
0.0390,	-0.0558,	-0.0440,	0.0119,	0069,	0.0199,	-0.0788,
-0.0123,	0.0151,	-0.0236	-0.0037,	0.0057,	-0.0095,	0.0202
0.0208,	0.0031,	-0.0283,	-0.0402,	-0.0016,	-0.0099	-0.0352,
...	....	...	...	...	...	...



02

Tokenizer 完整流程

预处理四步法



Normalization

- Normalization：文本清洗、数字处理

Pre-Tokenization

- Pre-tokenization：基于规则初步分割

Model

- 核心算法模型：BPE/WordPiece等子词拆分

Post-Tokenization

- Post-processing：Padding、Truncation、特殊标记

预处理四步法Step1： Normalization

- **文本清洗**
  - 去除无用字符：移除文本中的特殊字符、非打印字符等，只保留对分词和模型训练有意义的内容。
  - 去除额外空白：消除文本中多余的空格、制表符、换行符等，统一文本格式。
- **标准化写法**
  - 统一大小写：将所有文本转换为小写或大写，减少大小写变体带来的影响。
  - 数字标准化：将数字统一格式化，如 2025-> 2, 0 , 2, 5 便于推理模型理解。
  - 字符标准化：确保文本采用统一的字符编码（如UTF-8），处理或转换特殊字符和符号。
- **安全与规范化**

预处理四步法 Step2： Pre-tokenization

- Pre-tokenization 基于一些简单的规则进行初步的文本分割
- 将文本初步拆解为更小的单元，如句子和词语：
  - 英文等使用空格分隔的语言来说，这一步相对直接
  - 但对于中文等无空格分隔的语言，则直接使用 Modeling 进入下一步

预处理四步法 Step3： Model

- 根据选定的模型算法（BPE，WordPiece，Unigram、SentencePiece）进行处理
- 通过大量文本数据，根据模型算法规则生成词汇表（Vocabulary）
- 然后依据词汇表，将文本拆分为 Token

预处理四步法 Step4： Post-tokenization

- **序列填充与截断**：保证输入序列的长度一致，对过长序列进行截断，对过短序列进行填充。
- **特殊Token添加**：根据模型需求，在序列的适当位置添加特殊 Token（如[CLS], [SEP]）。
- **构建注意力掩码**：对于需要的模型，构建注意力掩码以区分实际 Token 和填充 Token。

关键参数：词汇表大小的权衡

- 大词汇表（覆盖更多词） vs 小词汇表（计算效率高）
  1. 大词表增加模型在进行词嵌入查找和生成输出时的计算负担，从而减慢处理速度；
  2. 大词表可提高模型覆盖不同词汇和表达的能力，有助于模型更好地理解和生成文本；
  3. 大词表也可能导致模型在某些Token上的训练不足，影响其泛化能力。

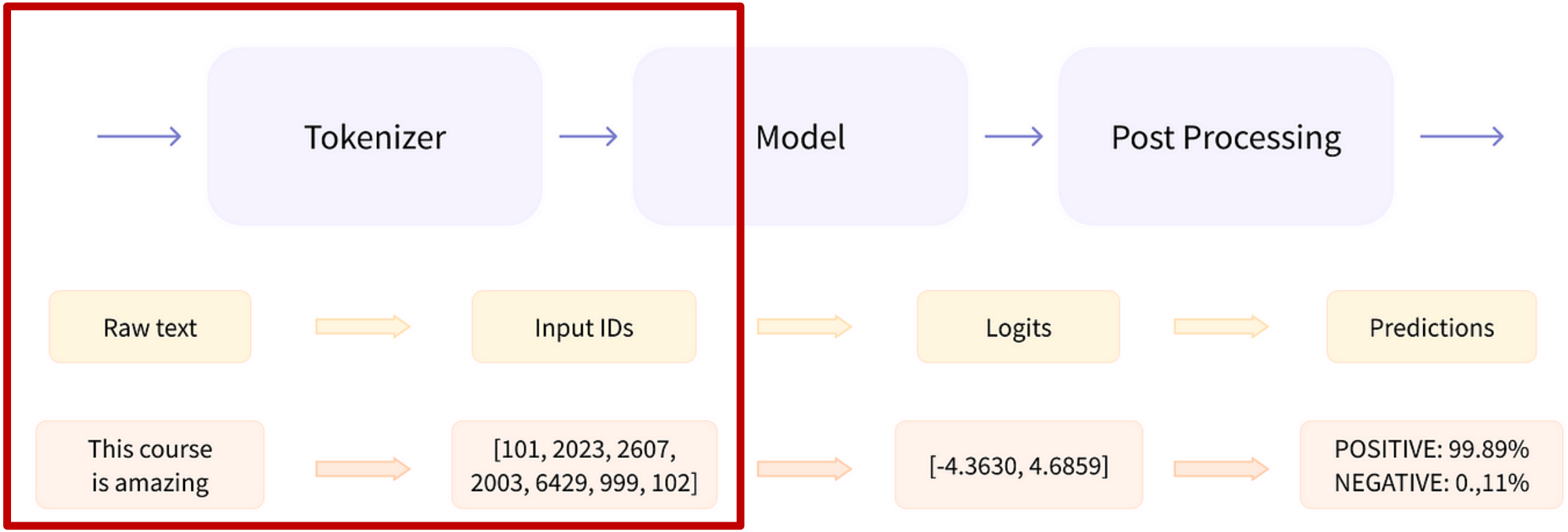
Tokenizer 示例

- https://tiktokenizer.vercel.app/

- 输入："Hello, world!" → 输出：["Hello", ",", "world", "!"] → [102, 12, 356, 8]

Qwen3-8B 分词流程

- 文本 → 字节级编码 → 合并高频字符对 → 映射到扩展词表 → 输出 Token ID 序列



03

分词算法深度解析

分词方法演进史

- Word-based （基于单词）：
  - 按照词进行分词， 如英文Today is sunday. 则根据空格或标点进行分割 [today, is, sunday, .]
- Character-based （基于字符）：
  - 按照单字符进行分词，以 char 为最小粒度， Today is sunday. → [t, o, d, a, y, I, s, s, u, n, d, a, y, .]
- Subword-based（子词分词）：
  - 按照词的 subword 进行分词，Today is sunday. → [today, is, sun, day, .]

分词方法演进史 Word-based

- **优点：**简单直观，符合人类语言习惯。
- **缺点：**
  - 词汇表爆炸（如英语10万+单词）、OOV（Out-of-Vocabulary）问题严重。
  - 不能学习到词缀之间的关系，例如：dog与dogs，happy与unhappy

- **示例：**“unhappiness” → 1 个 token（无法拆解前缀un-和后缀-ness）。

分词方法演进史 Character-based

- **优点：**词汇表极小（如26个字母 + 标点）。
- **缺点：**
  - 丢失语义信息，模型需自行学习长距离依赖（如“h”, “a”, “p”, “p”, "y"）。
  - 每个token的信息密度低
  - 序列过长，解码效率很低

分词方法演进史 Subword-based

- **核心思想：** 将高频词切分为更细粒度的子词，低频词保留为完整词或拆解为字符。
- **优势：**
  - 平衡词汇表大小与语义表达能力，解决OOV问题。
  - 词表规模适中，解码效率较高，能学习到词缀之间的关系。

- **典型算法：**

分词方法	典型模型
BPE （Byte Pair Encoding）	GPT，RoBERTa, BART, GLM, Qwen, DeepSeek, LLAMA, KIMI
WordPiece	BERT, DistilBERT， MobileBERT
Unigram	AlBERT, T5, mBART, XLNet

Byte Pair Encoding (BPE)

Neural Machine Translation of Rare Words with Subword Units

1. **构建初始词表：**
   - 首先按字符粒度分词，将文本中的每个字符视为一个单独的token，形成初始词汇表。
2. **识别最频繁出现的标记对：**
   - 扫描语料库，找到出现频率最高的token对（字符或子词）。
3. **合并最频繁出现的 token 对：**
   - 将2中频率最高 token 对，合并成为一个新 token。
   - 将新 token 添加到词表，并且删除被新 token 完全覆盖的 token，完成词汇表的更新。
   - 同时将 token 对的合并加入到合并规则列表中。
4. **重复步骤 2 和 3：**
   - 继续合并最频繁出现的token对，直到达到指定的词汇量或直到token对对不再频繁出现为止。

Byte Pair Encoding (BPE)

- 统计输入中所有出现的单词，并在每个单词后加一个单词结束符：
  - </w> -> ['hello</w>': 6, 'world</w>': 8, 'peace</w>': 2]
- 将所有单词拆成单字：
  - -> {'h': 6, 'e': 10, 'l': 20, 'o': 14, 'w': 8, 'r': 8, 'd': 8, 'p': 2, 'a': 2, 'c': 2, '</w>': 3}
- 合并最频繁出现的单字：
  - (l, o) -> {'h': 6, 'e': 10, 'lo': 14, 'l': 6, 'w': 8, 'r': 8, 'd': 8, 'p': 2, 'a': 2, 'c': 2, '</w>': 3}
- 合并最频繁出现的单字：
  - (lo, e) -> {'h': 6, 'lo': 4, 'loe': 10, 'l': 6, 'w': 8, 'r': 8, 'd': 8, 'p': 2, 'a': 2, 'c': 2, '</w>': 3}
- 反复迭代直到满足停止条件

Byte Pair Encoding (BPE)

- 给定单词序列：
  - ["the</w>", "highest</w>", "mountain</w>"]
- 从一个很大的 corpus 中排好序的 subword 如下，长度 6 5 4 4 4 4 2：
  - ["errrr</w>", "tain</w>", "moun", "est</w>", "high", "the</w>", "a</w>"]
- 迭代结果：
  - "the</w>" -> ["the</w>"]
  - "highest</w>" -> ["high", "est</w>"]
  - "mountain</w>" -> ["moun", "tain</w>"]

04

Tokenizer 评估指标

Tokenizer 评估指标

- 词汇表覆盖率（能否处理多语言/专业术语）
- token 数量（过长导致计算浪费，过短丢失信息）
- OOV 率（Out-Of-Vocabulary Rate）

05

Tokenizer 实践

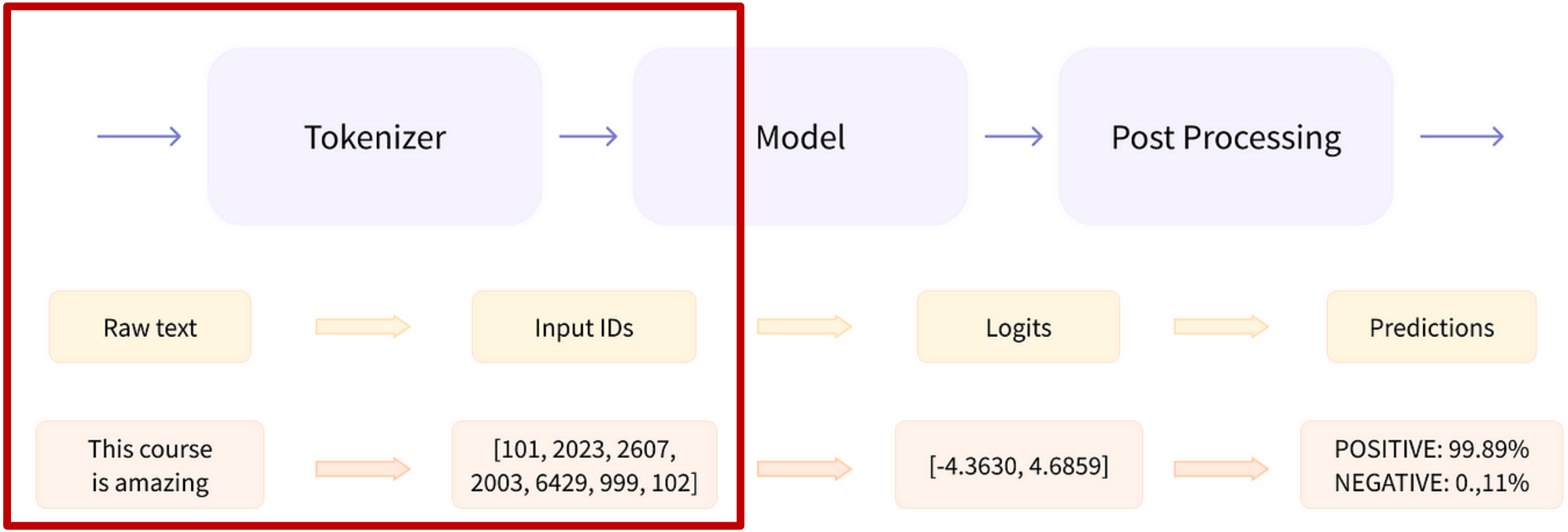
Qwen3-8B 大模型分词算法

- https://huggingface.co/Qwen/Qwen3-8B

- **算法：** Qwen3 采用字节级 BPE（BBPE）
- **词表：** 基于开源分词框架 tiktoken 的 cl100k 基础词表（~10 万词）初始化
- **中文扩展：** cl100k 基础上，添加高频中文字词、成语短语，最终词表扩展至 152,000+ Token
- **数字处理**：连续数字拆分为单个数字（2025→2、0、2、5），提升模型对推理任务泛化能力

Qwen3-8B 分词流程

- 文本 → 字节级编码 → 合并高频字符对 → 映射到扩展词表 → 输出 Token ID 序列



总结与思考

要点回顾 Summary

1. Tokenizer 是 Transformer 大模型理解文本/图像的第一道关卡，作为模型的输入。
2. Subword 方法（BPE、WordPiece）是当前大模型的主流，平衡语义表达与计算效率。

思考 Question

- 为什么 output token 的价格比 input token 更贵?

旗舰模型

旗舰模型	通义千问-Max 适合复杂任务，能力最强	通义千问-Plus 效果、速度、成本均衡	通义千问-Turbo 适合简单任务，速度快、成本低
最大上下文长度 （Token数）	32,768	131,072	1,008,192
最低输入价格 （每百万Token）	$1.6	$0.4	$0.05
最低输出价格 （每百万Token）	$6.4	$1.2	$0.2

https://www.alibabacloud.com/help/zh/model-studio/models

Thank you

把AI系统带入每个开发者、每个家庭、每个组织，构建万物互联的智能世界

Bring AI System to every person, home and organization for a fully connected, intelligent world.

ZOMI

GitHub https://github.com/chenzomi12/AIInfra

引用与参考

- https://www.youtube.com/watch?v=sOPDGQjFcuM&t=1s
- https://arxiv.org/abs/2308.00951
- https://arxiv.org/abs/2106.05974
- https://zhuanlan.zhihu.com/p/652536107
- https://github.com/jingyaogong/minimind/issues/111

- PPT 开源在：https://github.com/chenzomi12/AIInfra