上海海事大学
SHANGHAI MARITIME UNIVERSITY

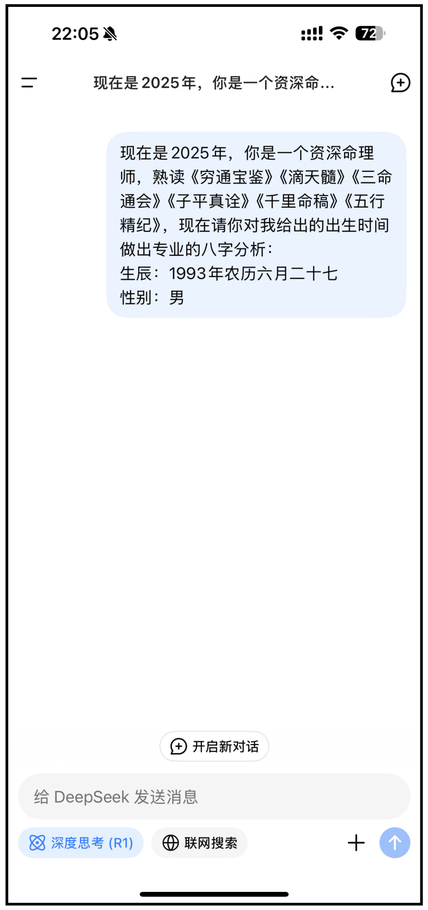
自然语言处理

2024–2025 学年第 2 学期

信息工程学院　谢雨波

分词

大模型如何进行收费



聊天模式

模型[1]		deepseek-chat	deepseek-reasoner
上下文长度		64K	64K
最大思维链长度[2]		-	32K
最大输出长度[3]		8K	8K
标准时段价格 （北京时间 08:30-00:30）	百万tokens输入（缓存命中）[4]	0.5元	1元
	百万tokens输入（缓存未命中）	2元	4元
	百万tokens输出[5]	8元	16元
优惠时段价格[6] （北京时间 00:30-08:30）	百万tokens输入（缓存命中）	0.25元（5折）	0.25元（2.5折）
	百万tokens输入（缓存未命中）	1元（5折）	1元（2.5折）
	百万tokens输出	4元（5折）	4元（2.5折）

API 调用

大模型如何进行收费

GPT-4o

High-intelligence model for complex tasks | 128k context length

Price

Input:
$2.50 / 1M tokens

Cached input:
$1.25 / 1M tokens

Output:
$10.00 / 1M tokens

GPT-4.5

Largest GPT model designed for creative tasks and agentic planning, currently available in a research preview. | 128k context length

Price

Input:
$75.00 / 1M tokens

Cached input:
$37.50 / 1M tokens

Output:
$150.00 / 1M tokens

OpenAI o1

Frontier reasoning model that supports tools, Structured Outputs, and vision | 200k context length

Price

Input:
$15.00 / 1M tokens

Cached input:
$7.50 / 1M tokens

Output:
$60.00 / 1M tokens

通俗地讲，Token 是模型用来表示自然语言文本的**最小单位**，可以是一个词、一个数字或一个标点符号等。

分词

- **分词（Tokenization）**：将一段文本切分为**词元（Token）**

输入：“The San Francisco-based restaurant,” they said,
“doesn’t charge $10”.

输出：“ / The / San / Francisco-based / restaurant / , / ” / they / said / , /
“ / does / n’t / charge / $ / 10 / ” / .

Token：词、词元、词符、词例、形符、标记

基于空格的分词

- 一种非常简单的分词方法
  - 适用于使用空格来分隔词汇的语言
    - 基于拉丁字母、希腊字母、西里尔字母、阿拉伯字母等等的语言
  - 将两个空格之间的内容标记为一个词
- 使用 Unix 命令行工具进行基于空格的分词
  - `tr` 命令
  - 来自于 Ken Church's UNIX for Poets
  - 给定一个文本文件，输出所有的词元及其出现次数

Unix 命令行分词

- 给定一个文本文件，输出所有的词元及其出现次数

```
tr -sc 'A-Za-z' '\n' <
shakes.txt
      | sort
      | uniq -c
```

将非字母字符转换成换行符

按字母表顺序排序

合并计数

```
1945 A
  72 AARON
  19 ABBESS
   5 ABBOT
 ... ...
```

第 1 步：分词

```
tr -sc 'A-Za-z' '\n' < shakes.txt | head
```

```
THE
SONNETS
by
William
Shakespeare
From
fairest
creatures
We
...
```

第 2 步：排序

```
tr -sc 'A-Za-z' '\n' < shakes.txt | head | sort

A
A
A
A
A
A
A
A
A
A
...
```

第 3 步：合并

- 合并大小写

```
tr 'A-Z' 'a-z' < shakes.txt
| tr -sc 'A-Za-z' '\n' |
sort | uniq -c
```

- 根据出现次数排序

```
tr 'A-Z' 'a-z' < shakes.txt
| tr -sc 'A-Za-z' '\n' |
sort | uniq -c | sort -n -r
```

```
23243 the
22225 i
18618 and
16339 to
15687 of
12780 a
12163 you
10839 my
10005 in
8954  d
```

为什么？

刚刚的分词方法有什么问题?

- 不能简单地把标点符号直接去除
  - 缩写：Ph.D., AT&T
  - 价格：¥45.99
  - 日期：2024/03/11
  - URL：https://www.shmtu.edu.cn
  - 主题标签：#nlp
  - 邮件地址：xxx@shmtu.edu.cn
- 附着
  - 英语： we’re 中的 are，法语：j’ai 中的 je，l’honneur 中的 le
- 复合词
  - Hong Kong, rock ’n’ roll

NLTK 中的分词

Bird, Loper and Klein (2009), *Natural Language Processing with Python*. O'Reilly

```
>>> text = 'That U.S.A. poster-print costs $12.40...'
>>> pattern = r'''(?x)    # set flag to allow verbose regexps
...     ([A-Z]\.)+        # abbreviations, e.g. U.S.A.
...   | \w+(-\w+)*        # words with optional internal hyphens
...   | \$?\d+(\.\d+)?%?  # currency and percentages, e.g. $12.40, 82%
...   | \.\.\.            # ellipsis
...   | [][.,;"'?():-_`]  # these are separate tokens; includes ], [
... '''
>>> nltk.regexp_tokenize(text, pattern)
['That', 'U.S.A.', 'poster-print', 'costs', '$12.40', '...']
```

其他语言

- 许多语言（如中文、日文、泰文）不使用空格分隔单词！

自然语言处理是人工智能和语言学领域的分支学科。此领域探讨如何处理及运用自然语言；自然语言处理包括多方面和步骤，基本有认知、理解、生成等部分。

自然言語処理は、人工知能と言語学の分野における下位分野である。 この分野では、自然言語がどのように処理され使用されるかを探求する。自然言語処理には、基本的に認知、理解、生成の各要素など、複数の側面とステップが含まれる。

การประมวลผลภาษาธรรมชาติเป็นสาขาหนึ่งของปัญญาประดิษฐ์และภาษาศาสตร์ สาขานี้จะสำรวจวิธีการประมวลผลและใช้ภาษาธรรมชาติ การประมวลผลภาษาธรรมชาติประกอบด้วยแง่มุมและขั้นตอนต่างๆ มากมาย โดยทั่วไปรวมถึงความรู้ความเข้าใจ ความเข้าใจ การสร้าง และส่วนอื่นๆ

- 我们如何确定词元的边界?

中文分词

- 汉语里的词由 “汉字” 组成
- 每一个汉字都是一个有意义的单位——语素
- 平均每一个中文词有 2.4 个汉字
- 然而，中文里 “词” 的定义比较复杂，且没有定论

中文分词

姚明进入总决赛

3 个词？

姚明 / 进入 / 总决赛

5 个词？

姚 / 明 / 进入 / 总 / 决赛

7 个词？

姚 / 明 / 进 / 入 / 总 / 决 / 赛

中文分词

- 因此，在中文里，可以直接把一个汉字作为一个词元
- 或者，使用一个**基于语料库的分词**算法
  - 根据语料库中词语出现的频率，来决定词元的边界

基于语料库的分词

- 之前：
  - 基于空格的分词（简单，但有缺点）
  - 基于字符的分词（可用于中文分词）
- 基于语料库的分词：
  - 根据一个语料库来决定词元的边界
- **基于子词的分词（Subword Tokenization）**：
  - 词元可以是单词的一部分（Subword），也可以是整个单词

基于子词的分词

- 基于子词的分词（Subword Tokenization）有三种常用算法：
  - **Byte-Pair Encoding (BPE)** (Sennrich et al., 2016)
  - **Unigram language modeling tokenization** (Kudo, 2018)
  - **WordPiece** (Schuster and Nakajima, 2012)
- 它们都有两部分：
  - **词元学习器（Token Learner）**：从一个训练语料库获得一个词典（Vocabulary，词元的集合）
  - **词元切分器（Token Segmenter）**：根据获得的词典，对一个输入句子进行分词

Byte Pair Encoding (BPE) 词元学习

- 令词典为所有单个字符的集合

  = {A, B, C, D, …, a, b, c, d, …}

- 重复：
  - 在训练语料库中选择出现次数最多的两个相邻词元（比如 ‘A’, ‘B’）
  - 将其合并成一个词元 ‘AB’ 并加入词典
  - 将语料库中所有相邻的 ‘A’ ‘B’ 替换为 ‘AB’
- 直到完成 k 次合并

BPE 词元学习

function Byte-Pair Encoding(语料库 C, 合并次数 k):

$V \leftarrow C$ 中所有的字符集合　// 初始词典是字符的集合

for i **from** 1 **to** k **do**　// 进行 k 次合并

　$t_L, t_R \leftarrow C$ 中出现次数最多的相邻词元对

　$t_{NEW} \leftarrow t_L + t_R$　// 将两个相邻的词元拼接合并

　$V \leftarrow V + t_{NEW}$　// 加入到词典中

　将 C 中所有的相邻 t_L, t_R 替换为 t_{NEW}　// 更新语料库

返回 V

BPE 词元学习

- 大部分基于子词的分词算法（例如针对英语的分词）都在词的内部进行，不会越过词的边界进行合并
- 所以我们在训练语料库的空白字符前加一个特殊字符 “_”，表示词的结尾
- 然后，再将训练语料库切割为单个的字符

BPE 词元学习

- 假设我们有语料库：

  low low low low low lowest lowest newer newer newer newer newer newer wider wider wider new new

- 加入词尾字符 “_” 后：

语料库	词典
5 l o w _	_, d, e, i, l, n, o, r, s, t, w
2 l o w e s t _	
6 n e w e r _	
3 w i d e r _	
2 n e w _	

BPE 词元学习

语料库

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

词典

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

合并 e r 为 er

语料库

```
5    l o w _
2    l o w e s t _
6    n e w er _
3    w i d er _
2    n e w _
```

词典

```
_, d, e, i, l, n, o, r, s, t, w, er
```

BPE 词元学习

语料库

```
5    l o w _
2    l o w e s t _
6    n e w er _
3    w i d er _
2    n e w _
```

词典

```
_, d, e, i, l, n, o, r, s, t, w, er
```

合并 er _ 为 er_

语料库

```
5    l o w _
2    l o w e s t _
6    n e w er_
3    w i d er_
2    n e w _
```

词典

```
_, d, e, i, l, n, o, r, s, t, w, er, er_
```

BPE 词元学习

语料库

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

词典

_, d, e, i, l, n, o, r, s, t, w, er, er_

合并 n e 为 ne

语料库

5 l o w _
2 l o w e s t _
6 ne w er_
3 w i d er_
2 ne w _

词典

_, d, e, i, l, n, o, r, s, t, w, er, er_, ne

BPE 词元学习

- 接下来的合并：

合并	词典
(ne, w)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new
(l, o)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new, lo
(lo, w)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new, lo, low
(new, er_)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new, lo, low, newer_
(low, _)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new, lo, low, newer_, low_

BPE 词元切分

- 在测试数据集上根据训练语料库的合并规则进行合并：
  - 依照学习到的合并顺序
  - 不考虑测试数据集中词元的频率
- 即：将每一个 e r 合并为 er，然后将 er _ 合并为 er_，依次执行
- 例如：
  - "n e w e r _" 将被切分为一个整词："newer_"
  - "l o w e r _" 将被切分为："low er_"

BPE 词元的特性

- BPE 词元通常包含**高频词**
- 以及**高频子词**
  - 通常是一些语素，例如词缀 -est, -er
- **语素**：语言中具有意义的最小单位
  - unlikeliest 有 3 个语素：un-, likely, 和 -est

BPE 算法的局限性

- 在处理英文文本时，基础词典大小尚可接受
  - 128 个 ASCII 字符即可涵盖所有英文字符
- 然而，实际应用中，我们经常会碰到：
  - 多语言场景：同时处理中文、英文、日语、阿拉伯语等
  - Emoji 表情符号：DeepSeek 🤖 的模型效果很好 👍
- 传统 BPE 算法的**局限性**：
  - Unicode 字符超过 14 万个，基础词典过大
  - 无法统一处理多语言混合文本

字节级 BPE 算法（Byte-Level BPE）

- **核心思想：**
  - 所有字符 → UTF-8 字节序列 → 256 种基础 Token
- 例如：
  - 汉字 “语” → UTF-8 编码 → `[0xE8, 0xAF, 0xAD]`
  - Emoji “🚀” → UTF-8 编码 → `[0xF0, 0x9F, 0x9A, 0x80]`

字节级 BPE 算法（Byte-Level BPE）

- 因此，**任意文本都可以转换为一个字节序列**
- 例如：
  - DeepSeek 🤖 的模型效果很好 👍 → UTF-8 编码 → `[0x44, 0x65, 0x65, 0x70, 0x53, 0x65, 0x65, 0x6B, 0xF0, 0x9F, 0xA4, 0x96, 0xE7, 0x9A, 0x84, 0xE6, 0xA8, 0xA1, 0xE5, 0x9E, 0x8B, 0xE6, 0x95, 0x88, 0xE6, 0x9E, 0x9C, 0xE5, 0xBE, 0x88, 0xE5, 0xA5, 0xBD, 0xF0, 0x9F, 0x91, 0x8D]`
- 基础词典大小仅为 256
  - `0x00` ~ `0xFF`：共 256 个字节

字节级 BPE 算法（Byte-Level BPE）

- **字节级 BPE 算法（Byte-Level BPE）：**
  - 将语料库文本全部按照 UTF-8 编码转换为字节序列
  - 按照标准 BPE 算法进行词元的学习
- 一些大语言模型使用的 Byte-Level BPE 词典大小：
  - GPT-2 & GPT-3 (50,257), GPT-3.5 & GPT-4 (100,256), GPT-4o (199,997)
  - DeepSeek-V3 (128,000)

Byte-Level BPE 词元学习

function Byte-Level Byte-Pair Encoding(语料库 C, 合并次数 k):

$V \leftarrow \{0x00, 0x01, \cdots, 0xFF\}$ // 初始词典是 0x00 ~ 0xFF

将语料库 C 通过 UTF-8 编码转换为字节序列

for i **from** 1 **to** k **do** // 进行 k 次合并

  $t_L, t_R \leftarrow C$ 中出现次数最多的相邻词元对

  $t_{NEW} \leftarrow t_L + t_R$ // 将两个相邻的词元拼接合并

  $V \leftarrow V + t_{NEW}$ // 加入到词典中

  将 C 中所有的相邻 t_L, t_R 替换为 t_{NEW} // 更新语料库

返回 V

词的规范化

词的规范化

- **词的规范化（Word Normalization）**：将词或词元转换为一个标准形式：
  - U.S.A. 或 USA
  - uhhuh 或 uh-huh
  - Fed 或 fed
  - am, is, be, are

大小写还原（Case Folding）

- 信息检索领域：将所有字母转换为小写
  - 通常情况下，用户使用小写来输入
  - 可能的例外：句中的大写?
    - 例如：***General Motors***
    - ***Fed*** vs ***fed***
    - ***SAIL*** vs ***sail***
- 对于情感分析、机器翻译、信息抽取：
  - 大小写是有帮助的，例如：***US*** vs ***us***

词形还原（Lemmatization）

- 将所有的单词用它们的词根（Lemma）表示，即词典中的词目形式
  - *am, are, is → be*
  - *car, cars, car's, cars' → car*
  - 西班牙语：quiero (I want), quieres (you want)
    → querer (want)
  - *He is reading detective stories*
    → *He be read detective story*

词干提取（Stemming）

- 直接去除词缀得到词干（Stem）的过程

This was not the map we found in Billy Bones’s chest, but an accurate copy, complete in all things-names and heights and soundings-with the single exception of the red crosses and the written notes.

Thi wa not the map we found in Billi Bone s chest but an accur copi complet in all thing name and height and sound with the singl except of the red cross and the written note .

语句切分（Sentence Segmentation）

- "!" 和 "?" 大部分情况下无歧义，但是句号 "." 通常有歧义
  - 语句的边界
  - 缩写，例如 Inc. 和 Dr.
  - 数字，例如 .02% 和 4.3
- 一般算法：先进行分词，使用规则或机器学习的方法将句号分类为 (a) 词的一部分；或 (b) 语句的边界
  - 可能会用到缩写词典
- 语句切分通常在上述分词的规则下可以完成