2.6 英文分词与词元切分

林洲汉
上海交大电院

2023年秋季学期

目录

目录

简单句子的切分

You are what you eat.

From the above experiments we conclude that for sentence splitting the corpus being used for training is not so critical because sentence annotations are not really controversial within different biomedical subdomains.

简单按照空格和标点符号切分即可。

问题与需求：缩写词

I'm right, aren't I?

切分方法一：将缩写后的词作为单独的一个新词

I’m / right / , / aren't / I / ?

can't, don't, couldn't, shouldn't, wouldn’t…

切分方法二：将缩写后的词拆开

I / ’m / right / , / are / n't / I / ?

O'neil, o'clock, O'reilly…

问题与需求：数字、符号

The total revenue rises above $300,000,000. (c.f. Fig. 3)

切分方法一：将数字和符号按照标点切开

$ / 300 / , / 000 / , / 000	**Fig. / 3**
失去了数字的意义	Fig-3; Table-2

切分方法二：不拆开

$300,000,000	**Fig. 3**
$300,000,001; $300,000,002; $300,000,003; …	Fig. 1; Fig. 2 Table 3; Chart 4 …

目录

词级别的词元切分（word-level tokenization）

基于规则的方法

- 使用正则表达式做匹配，不断修改正则表达式的数量和形式，使得在给定语料上的切分效果趋于完善

基于学习的方法

- 使用序列标注模型，在给定语料上像学习词性标注与中文分词那样学习词元的切分。

词级别的词元切分（word-level tokenization）

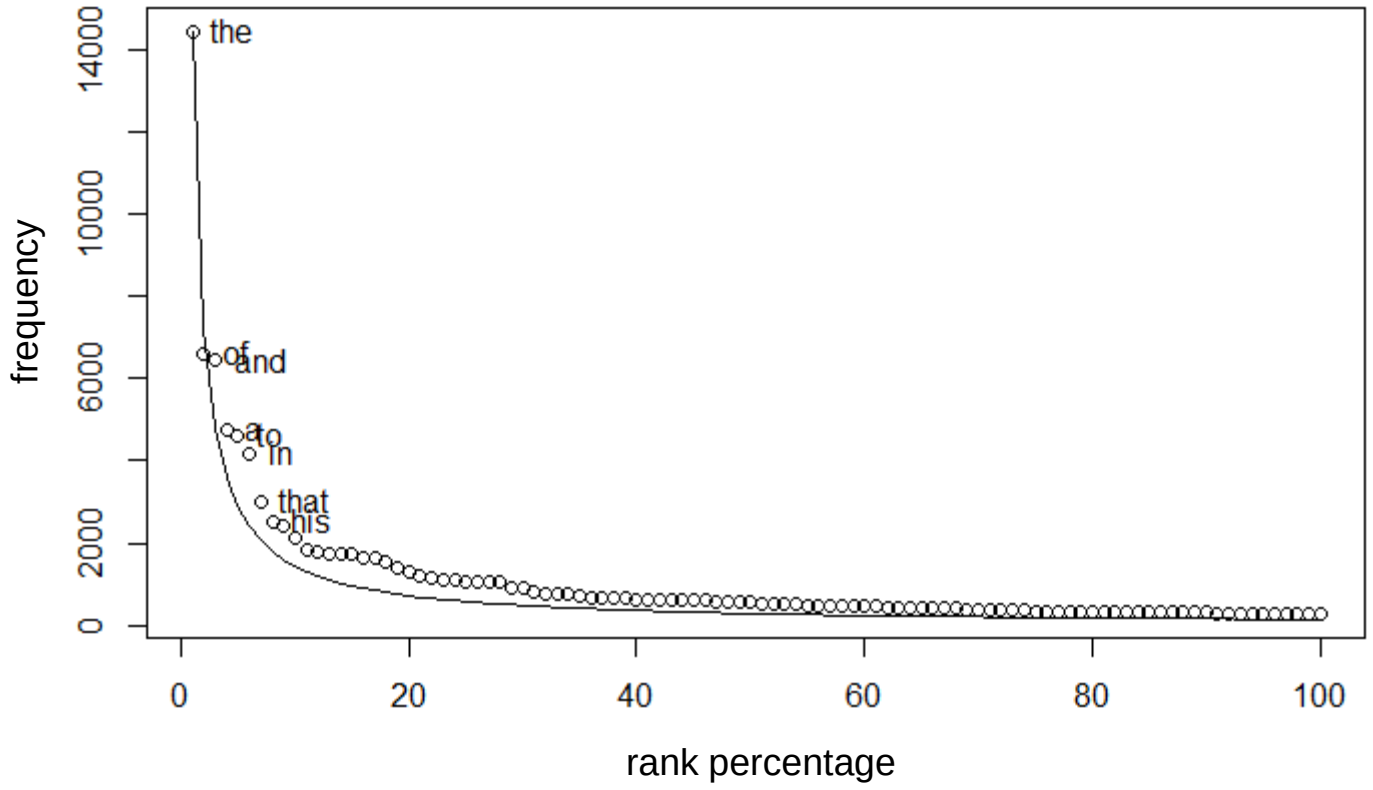
```
# starting quotes
STARTING_QUOTES = [
    (re.compile(r"^\""), r'``'),
    (re.compile(r"(``)"), r' \1 '),
    (re.compile(r"([ \(\[{<])(\"|\'{2})"), r"\1 `` "), ]

# punctuation
PUNCTUATION = [
    (re.compile(r"([:,])([^\d])"), r' \1 \2'),
    (re.compile(r"([:,])$"), r' \1 '),
    (re.compile(r"\.\.\."), r' ... '),
    (re.compile(r"[;@#$%&]"), r' \g<0> '),
    (re.compile(r'([^\.])(\.)([\]\)}>"\']*)\s*$'), r"\1 \2\3 ", ), # Handles the final period.
    (re.compile(r"[?!]"), r' \g<0> '),
    (re.compile(r"([^'])' "), r"\1 ' "), ]

# Pads parentheses
PARENS_BRACKETS = (re.compile(r"[\]\[\(\)\{\}\<\>]"), r' \g<0> ')

# Optionally: Convert parentheses, brackets and converts them to PTB symbols.
CONVERT_PARENTHESES = [
    (re.compile(r"\("), "-LRB-"),
    (re.compile(r"\)"), "-RRB-"),
    (re.compile(r"\["), "-LSB-"),
    (re.compile(r"\]"), "-RSB-"),
    (re.compile(r"\{"), "-LCB-"),
    (re.compile(r"\}"), "-RCB-"), ]

DOUBLE_DASHES = (re.compile(r"--"), r' -- ')

# ending quotes
ENDING_QUOTES = [
    (re.compile(r'"'), " '' "),
    (re.compile(r"(\S)(\'\')"), r"\1 \2 "),
    (re.compile(r"([^' ])('[sS]|'[mM]|'[dD]|') "), r"\1 \2 "),
    (re.compile(r"([^' ])('ll|'LL|'re|'RE|'ve|'VE|n't|N'T) "), r"\1 \2 "), ]
```

目录

亚词级别的词元切分（subword-level tokenization）

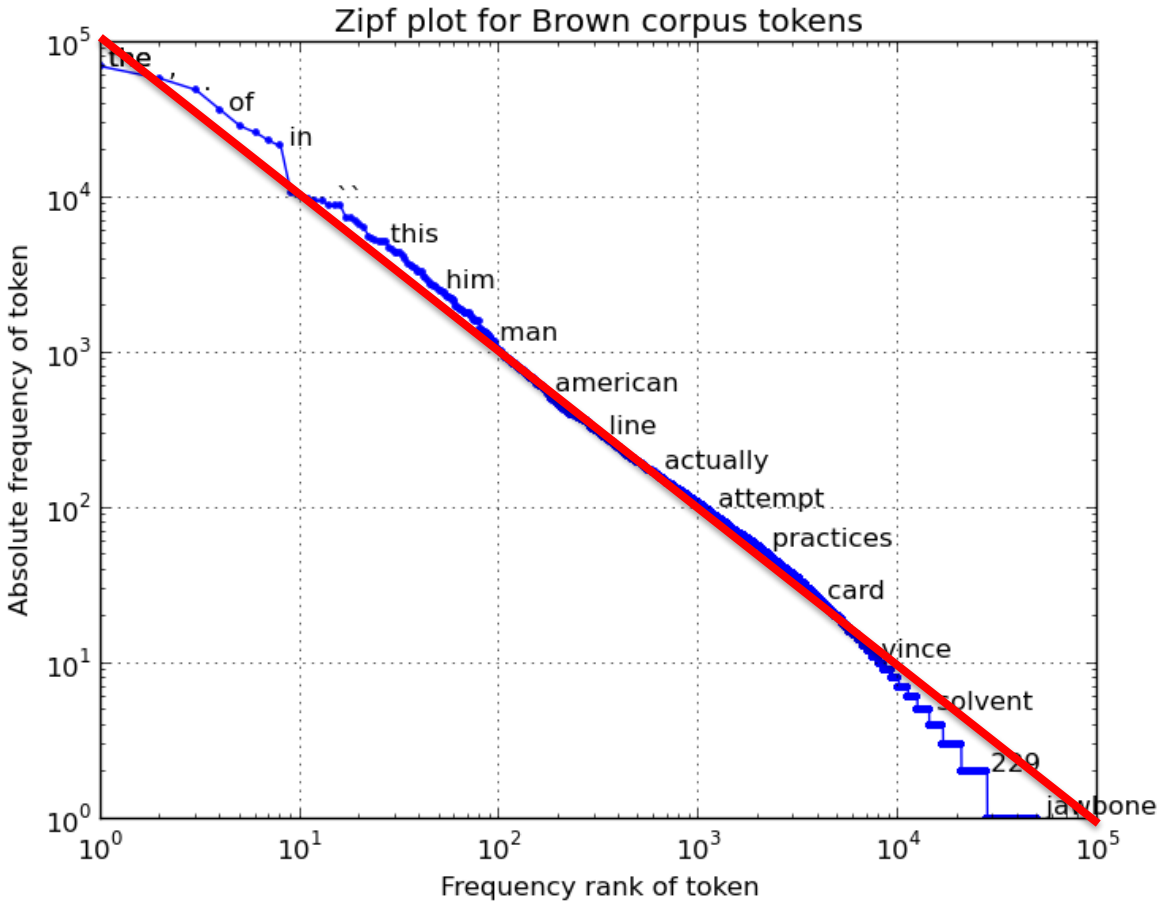
对于英语、法语、德语等语言而言，词并不是一个特别合适的切分单元

长尾分布



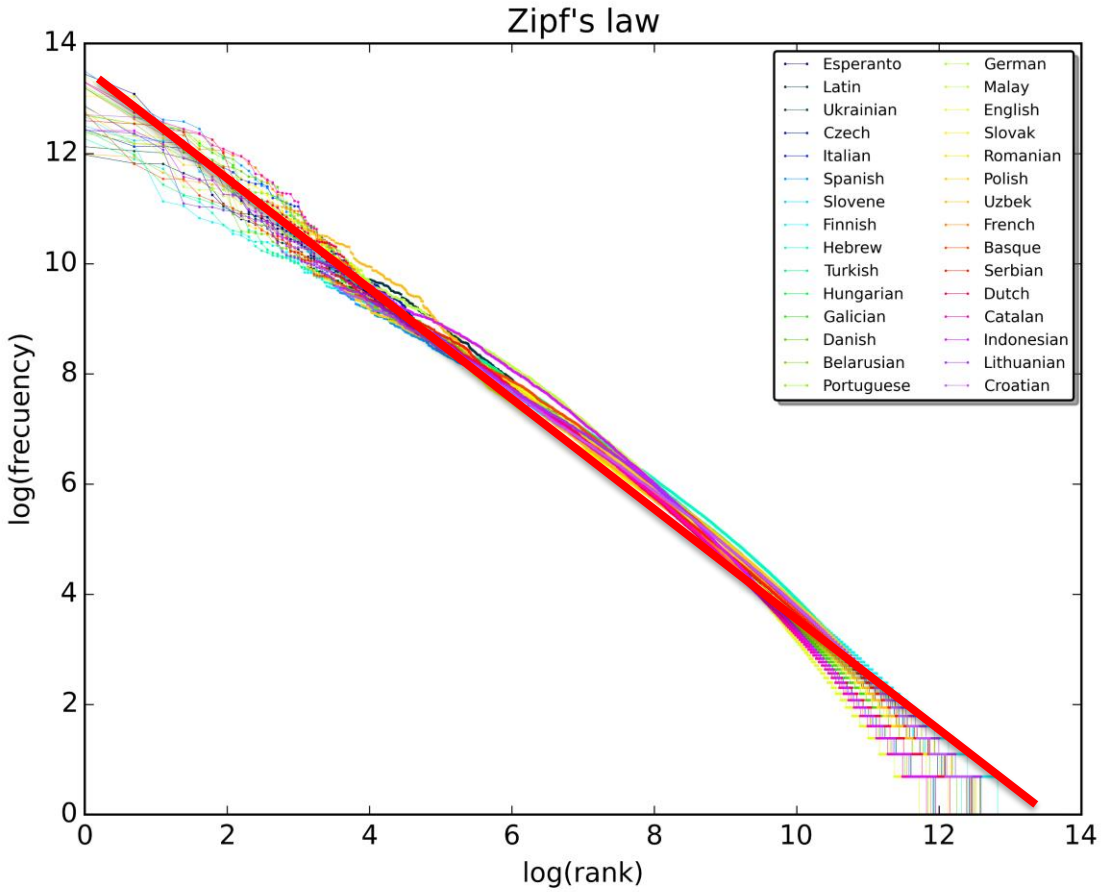
亚词级别的词元切分（subword-level tokenization）

如果我们使用双对数轴线：

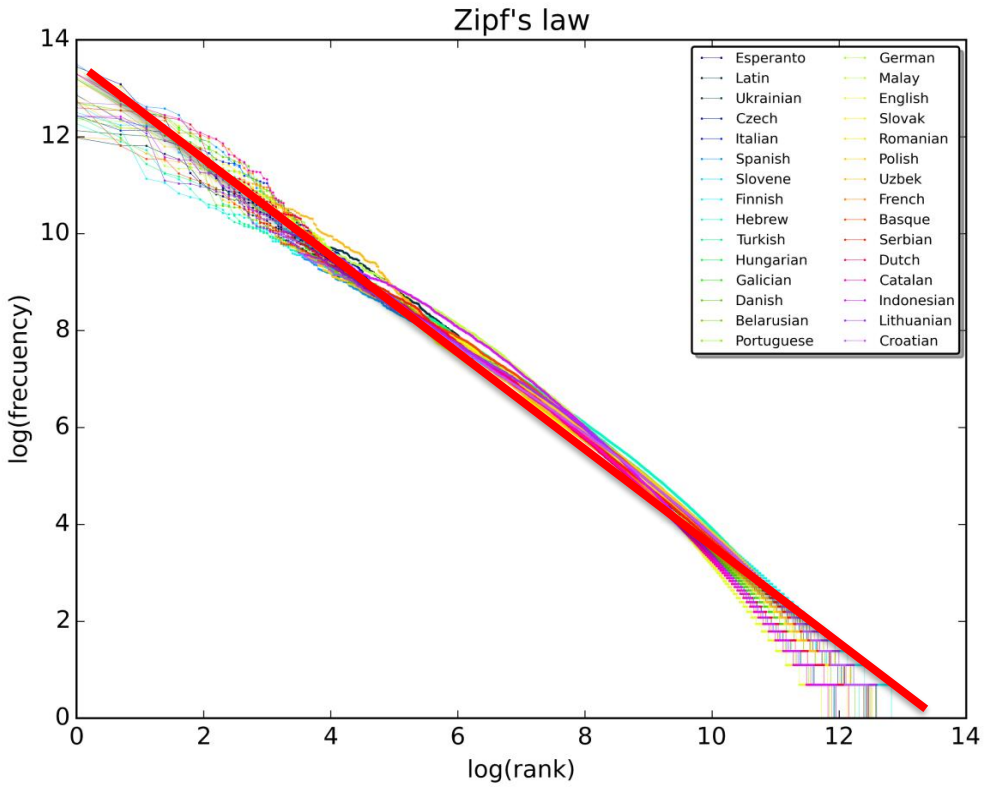


亚词级别的词元切分（subword-level tokenization）

同样的情况广泛存在于各种语言。比如对30种语言的维基百科进行词频统计：



亚词级别的词元切分（subword-level tokenization）



总的来说，几乎所有语言中词频的分布都大致符合

“Zipf's Law”，

即对于包含有N个不同单词的语料集，其第k个最常见单词饭归一化频度为:

$$f(k) = \frac{\frac{1}{k^s}}{\sum_{i=1}^{N} \frac{1}{i^s}}$$

其中s为参数，随具体语言的不同而不同。

这样的分布对应到双对数轴线上，就是一条直线。

亚词级别的词元切分（subword-level tokenization）

对于英语、法语、德语等语言而言，词并不是一个特别合适的切分单元

- 对于一个中等到大型的语料集而言，其涉及到的词汇将达到30~60万。
- 即便刨除只出现一两次的罕见词，词汇表仍可轻松超越10万。且大量的未登录词会造成模型在具体任务上性能的下降。

然而大量的单词之间存在互相联系，或是同一个词的不同屈折变化，或是共享部分词根

亚词级别的词元切分（subword-level tokenization）

对于英语、法语、德语等语言而言，词并不是一个特别合适的切分单元

- 同一个词的不同屈折变化

例如：

response
responsive
responsible
responsibility
…

- 不同词之间的联系

例如：

contract
contradictory
contrastive
inventive
responsive
bilateral
bicycle
reserve
preserve
conserve
…

亚词级别的词元切分（subword-level tokenization）

对于英语、法语、德语等语言而言，词并不是一个特别合适的切分单元

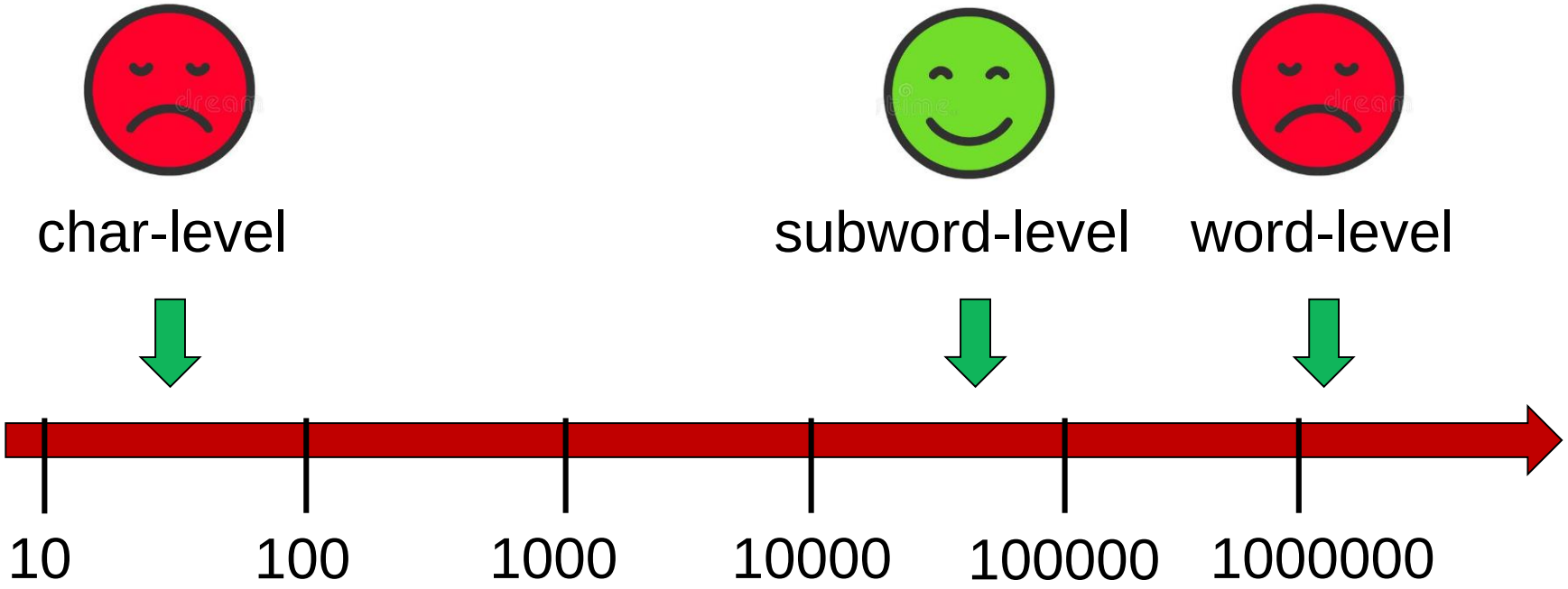
然而，除了词之外，字母也不是很好的切分单元。

- 各个语言，主要是英语等欧洲语言中，字母表的数目在几十的量级，即便考虑某些语言的accent（如àáâãäåéèëïìöö等）字母表也很难超过100。

亚词级别的词元切分（subword-level tokenization）

对于英语、法语、德语等语言而言，词并不是一个特别合适的切分单元

然而，除了词之外，字母也不是很好的切分单元。



BPE: Byte-Pair Encoding

BPE是2016年提出的亚词级别词元切分方法[1]，其使用的算法直接继承自1994年的数据压缩领域的论文[2]。

该算法从数据中自底向上地组合字母来形成词元，词元数目精确可控。

通过适当调整参数，可以得到类似于“词根词缀”的英文亚词单元。

1. Sennrich, Rico, Barry Haddow, and Alexandra Birch. "Neural Machine Translation of Rare Words with Subword Units." In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 1715-1725. 2016.
2. Gage, Philip. "A new algorithm for data compression." The C Users Journal 12, no. 2 (1994): 23-38.

BPE: Byte-Pair Encoding：算例

{ ‘low_’ : 5, ‘lowest_’ : 2, ‘newer_’ : 3, ‘wider_’ : 6}

1. 初始词汇表：将所有出现的单个字母全都包含进来

l o w e s t n r i d _

BPE: Byte-Pair Encoding：算例

{ ‘low_’ : 5, ‘lowest_’ : 2, ‘newer_’ : 3, ‘wider_’ : 6}

1. 初始词汇表：将所有出现的单个字母全都包含进来
2. 迭代：（1）将现有语料库用现有词汇表表示；
   （2）再将其中最常见的一个bi-gram捏合成一个新的token

l o w e s t n r i d _ +

BPE: Byte-Pair Encoding：算例

{ ‘low_’ : 5, ‘lowest_’ : 2, ‘newer_’ : 3, ‘wider_’ : 6}

1. 初始词汇表：将所有出现的单个字母全都包含进来
2. 迭代：（1）将现有语料库用现有词汇表表示；
   （2）再将其中最常见的一个bi-gram捏合成一个新的token

l o w e s t n r i d _ + r_

BPE: Byte-Pair Encoding

{ ‘low_’ : 5, ‘lowest_’ : 2, ‘newer_’ : 3, ‘wider_’ : 6}

1. 初始词汇表：将所有出现的单个字母全都包含进来
2. 迭代：（1）将现有语料库用现有词汇表表示；
   （2）再将其中最常见的一个bi-gram捏合成一个新的token

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

BPE: Byte-Pair Encoding：算例

{ ‘low_’ : 5, ‘lowest_’ : 2, ‘newer_’ : 3, ‘wider_’ : 6}

1. 初始词汇表：将所有出现的单个字母全都包含进来
2. 迭代：（1）将现有语料库用现有词汇表表示；
   （2）再将其中最常见的一个bi-gram捏合成一个新的token

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

BPE: Byte-Pair Encoding：算例

{ ‘low_’ : 5, ‘lowest_’ : 2, ‘newer_’ : 3, ‘wider_’ : 6}

1. 初始词汇表：将所有出现的单个字母全都包含进来
2. 迭代：（1）将现有语料库用现有词汇表表示；
   （2）再将其中最常见的一个bi-gram捏合成一个新的token

l o w e s t n r i d _ + er_ lo

BPE: Byte-Pair Encoding：算例

{ ‘low_’ : 5, ‘lowest_’ : 2, ‘newer_’ : 3, ‘wider_’ : 6}

1. 初始词汇表：将所有出现的单个字母全都包含进来
2. 迭代：（1）将现有语料库用现有词汇表表示；
   （2）再将其中最常见的一个bi-gram捏合成一个新的token

l o w e s t n r i d _ + er_ low

BPE: Byte-Pair Encoding：算例

{ ‘low_’ : 5, ‘lowest_’ : 2, ‘newer_’ : 3, ‘wider_’ : 6}

1. 初始词汇表：将所有出现的单个字母全都包含进来
2. 迭代：（1）将现有语料库用现有词汇表表示；
   （2）再将其中最常见的一个bi-gram捏合成一个新的token
3. 终止：迭代达到预先设定的最大次数时停止

最终词汇表中的词元数目为：初始词汇数+迭代次数

l o w e s t n r i d _ + er_ low

BPE: Byte-Pair Encoding：算法流程

1. 初始词汇表：将所有出现的单个字母全都包含进来
2. 迭代：（1）将现有语料库用现有词汇表表示；
   （2）再将其中最常见的一个bi-gram捏合成一个新的token
3. 终止：迭代达到预先设定的最大次数时停止

最终词汇表中的词元数目为：初始词汇数+迭代次数

BPE: Byte-Pair Encoding：优点

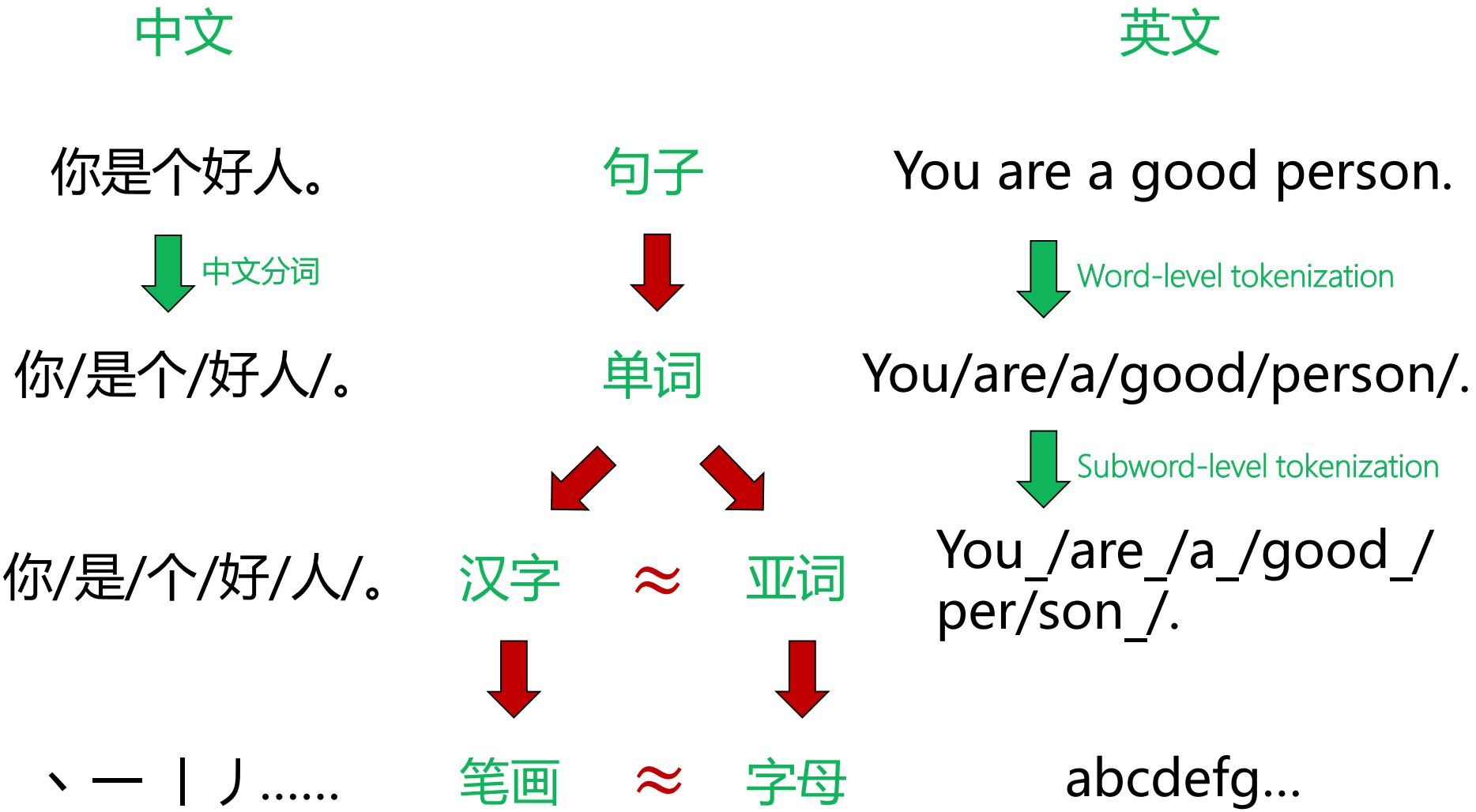
1. 解决了字母级与单词级的词元切分各自的问题，且词元数量精确可控。

2. 解决了未登录词的问题：无论设置的词元数量是多少，总是可以有办法使用有限的词元覆盖所有可能出现的单词。

3. 算法复杂度低，可以在大规模语料上实用。

4. 是目前绝大部分NLP模型采用的词元切分方法。

中英文对比


中文
英文
你是个好人。
句子
You are a good person.
中文分词
Word-level tokenization
你/是个/好人/。
单词
You/are/a/good/person/.
Subword-level tokenization
你/是/个/好/人/。
汉字
≈
亚词
You_/are_/a_/good_/per/son_/.
丶一丨丿……
笔画
≈
字母
abcdefg…

目录

词元切分：常用工具

词级别的词元切分（word-level tokenization）

序号	库名/函数名	描述
1	nltk.tokenize.word_tokenize	基于正则表达式的词级别词元切分
2	nltk.tokenize.wordpunct_tokenize	基于标点的词级别词元切分
3	spacy.tokenizer.Tokenizer	基于正则表达式的词级别词元切分
4	mosestokenizer	基于正则表达式的词级别词元切分

亚词级别的词元切分（subword-level tokenization）

序号	库名/函数名	描述
1	sentencepiece	使用BPE从句子直接切分出亚词
2	wordpiece	使用BPE从词切分出亚词