大语言模型Tokenization技术介绍
Introduction to Tokenization Technology for LLMs

刘群 LIU Qun

Huawei Noah's Ark Lab

大模型技术洞察交流

2024.06.20

NOAH'S ARK LAB

HUAWEI

Content

什么是Token、Tokenizer和Tokenization？

- Token在NLP中被普遍使用，通常是指构成语言序列的最小单位
- 早期NLP中Token通常都是词，所以token翻译成词例，指词的一次出现，与Type（词形）对立，指与出现次数无关的词的形式
- 在NLP进入深度学习以后，token通常比“词（Word）”更小，大部分情况下是“子词（subword）“，也可以指字符（character）甚至比字符更小的单位字节（byte）
- 在多模态大模型中，token也可能是图像、音频或者视频的切割单位
- 在计算机网络中，token是网络中传说信号的基本单位，通常翻译成令牌
- 由于token这个词的含义非常广泛，目前还没有同意的翻译，我们通常直接用英文

开放词表假设和封闭词表假设

- 在深度学习引入NLP之前，NLP通常采用开放词表假设（Open Vocabulary Assumption），也就是词表是可以任意添加的，tokenizer通常只需要把“词”切开就行，这对于英文来说比较简单，但对于中文这样的语言来说，词语切分还是比较复杂的，中文词语切分因此成为了一项独立的技术。
- 在深度学习引入NLP之后，由于所有词都在神经网络模型中被表现为词嵌入向量（embedding），那么在把词嵌入向量还原成词的时候，需要采用一种softmax操作，在整个词表Vocabulary上选择一个词，这时候我们必须把词表限制在一个有限集合中，所以只能采用封闭词表假设（Closed Vocabulary Assumption）。
- 真实自然语言的词（word）都是开放集合，词语出现的频率（frequency）符合Zipf定律。

封闭词表假设问题的解决方案

- 为了解决封闭词表假设和真实自然语言词表不受限问题，通常采用以下两种策略之一：
  - 早期的一种做法是，引入一个特殊token，通常记为UNK，表示unknown word。所有集外词（Out-of-Vocabulary）都被标记为UNK。这种方法带来的主要问题是：
    - UNK的总频率不低，但其embedding毫无意义；
    - 语言生成的时候经常会输出UNK，不可接受。
  - 现在主流做法是，引入子词（subword）级别的token，如果一个单词不在词表中，就切分成子词，子词最小可以是一个字符，甚至一个字节。
    - 子词级别的tokenization较好的解决了封闭词表假设带来的问题。
    - 当然也会有一些新问题，后面会讨论到。

Content

基于子词（Subword）的Tokenization方法

- 基本思想：
  - 词表只收录固定的子词的集合，这里把完整的词word也当初子词subword的特例；
  - 任何一个词，如果不在词表中，就需要被切分成子词序列subword sequence。
- 优点：
  - 彻底解决了集外词乏乏乖问题，任何一个单词都可以切分成子词序列。最坏情况下，所有子词都匹配不上，就会切分成字符序列；
  - 大部分常用词都可以作为独立子词收入词表，这样的话原始的基于词的模型无需做任何改动，序列长度增加很少，模型复杂度和效率都几乎没有增加。
- 问题：
  - 词表构建（词表训练）；
  - 子词切分（tokenization）。

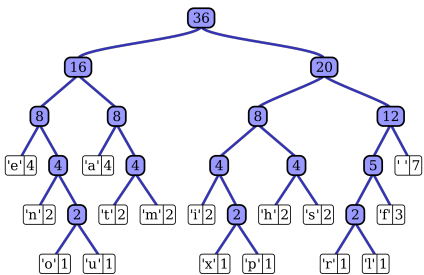
基于语言学子词的Tokenization方法

- 很多语言（如英语）在语言学上是存在子词级别的单位的，如词根stem、前缀prefix、后缀suffix等。
- 基于语言学子词的切分方法，切出的token具有明确的意义，理论上很合理。
- 缺点之一是，语言学上词根stem仍然是开放集合，另外，现在各种语言都含有大量外来词，通常都无法切分成词根词缀。
- 另一个缺点是，语言学切分方法跟具体语言有关，每种语言需要开发各自的子词切分系统，而且很多语言并没有可用的子词切分系统。
- 因此，基于语言学子词的切分方法并没有获得广泛应用。

Byte-Pair Encoding（BPE）子词切分算法

- BPE最早是作为一种压缩算法提出来的，Rich Sennrich(2015)首次提出将BPE算法用于子词级切分，以解决神经机器翻译的集外词问题，取得了很大的成功，并成为基于神经网络的NLP模型中最为流行的子词切分方法。
- BPE vs Hoffman Encoding：

Iteration	Sequence	Vocabulary
0	a b a b c a b c	{a, b, c}
1	ab ab c ab c	{a, b, c, ab}
2	ab abc abc	{a, b, c, ab, abc}
3	ababc abc	{a, b, c, ab, abc, ababc}
4	ababcabc	{a, b, c, ab, abc, ababc, ababcabc}



BPE词表训练算法

1. 预切分文本（通常按照空格或者回车字符进行预切分），得到一部词典Dictionary of Words，词典中包含文本中出现的所有词Words，以及该词Word出现的频率；
2. 将词典中所有的词Word切分成基本字符的序列，得到所有词的初始切分结果；
3. 初始化子词词表Vocabulary，词表只包含所有的基本字符（如字母、数字、标点、汉字等）构成的子词subword；
4. 重复以下过程，每一步将子词词表中的两个子词subword合并merge得到一个新的子词new subword加到词表末尾，直到子词词表Vocabulary的长度达到预设值：
   - 4.1 在词典中统计当前切分结果中所有相邻子词对儿subword pair的出现频率，选择出现频率最高的子词对儿top pair；
   - 4.2 将上述子词对儿top pair合并merge成一个新的子词new subword，并到子词词表Vocabulary末尾；
   - 4.3 更新词典中所有词的切分结果，将其中出现的上述子词对儿top pair都合并merge成新子词new subword；
5. 返回子词词表Vocabulary。

BPE词表训练示例

freq	step 0	step 1	step 2	step 3	step 4	step 5
5	l o w </w>	l o w </w>	l o w </w>	l o w </w>	lo w </w>	low </w>
2	l o w e r </w>	l o w e r </w>	l o w e r </w>	l o w e r </w>	lo w e r </w>	low e r </w>
6	n e w e s t </w>	n e w es t </w>	n e w est </w>	n e w est</w>	n e w est</w>	n e w est</w>
3	w i d e s t </w>	w i d es t </w>	w i d est </w>	w i d est</w>	w i d est</w>	w i d est</w>

freq	7	7	13	16	17	2	6	9	9	3	3	9	9	9	7	7
step 0	l	o	w	</w>	e	r	n	s	t	i	d					
step 1	l	o	w	</w>	e	r	n	s	t	i	d	e s				
step 2	l	o	w	</w>	e	r	n	s	t	i	d	e s	es t			
step 3	l	o	w	</w>	e	r	n	s	t	i	d	e s	es t	est </w>		
step 4	l	o	w	</w>	e	r	n	s	t	i	d	e s	es t	est </w>	l o	
step 5	l	o	w	</w>	e	r	n	s	t	i	d	e s	es t	est </w>	l o	lo w

BPE子词切分过程示例，上表为文本片段词典Dictionary of Words，下表为子词词表Vocabulary。

表中</w>是一个特殊字符，用于表示一个单词的结尾。step 0是初始化的词典和子词词表，其他每一个step将出现频率最高subword pair合并成一个新的子词new subword，并加在子词词表末尾。

BPE子词切分算法

1. 预切分文本，按照训练词表相同的方式，得到一个词序列Sequence of Words；
2. 将词序列的所有词切分成基本字符的序列，得到词语序列的初始切分结果，每个基本字符就是一个子词subword；
3. 重复以下过程，直到无法再对切分结果中的子词进行合并：
   - 3.1 在切分结果中，考察所有相邻子词对儿subword pair，确定它们合并merge后的候选新子词candidate subword在词表Vocabulary中的位置；
   - 3.2 如果所有相邻子词对儿合并merge后得到的候选新子词candidate subword在词表中都找不到相应的subword，那么切分结束，退出；
   - 3.3 在上述候选新子词candidate subword中，选择出现词表Vocabulary中出现位置最靠前的那个top subword；
   - 3.4 使用top subword更新词典中所有文本片段的切分结果，将其中出现的top subword对应的子词对儿subword pair都合并merge成top subword；

WordPiece子词切分算法

- Google在Schuster & Nakajima(2012)中就提出了WordPiece算法的思想，远早于BPE，但并没有命名这个算法，更没有开源。
- Google在论文Schuster et al.(2016)中首次提到在神经机器翻译系统中使用了Schuster & Nakajima(2012)的算法做文本的子词切分，并命名为WordPiece算法，这时已经晚于BPE了。
- 后来Google在BERT中再次使用了WordPiece，使得WordPiece影响大增。但Google并没有开源WordPiece的词表构造算法，也没有公开源代码。好在BERT的代码中还是提供了WordPiece的Tokenizer的调用接口，所以如果研究者们不想构造自己的子词词表，还是可以直接使用BERT提供的词表和词例切分工具做研究的。
- 后来Huggingface的工具包中也提供了一个模拟WordPiece模型的训练工具，但他们并没有真正使用Schuster & Nakajima(2012)中的算法，实际上还是使用BPE方法来构造了一个子词词表，只是把生成的词表改成了BERT提供的WordPiece数据格式，主要是迁就BERT的词表格式。
- 现在WordPiece基本已经没有人使用了，Google的LLM也不再使用。

WordPiece子词切分算法的基本思想

- WordPiece的子词词表构造算法跟BPE非常相似，也是一个从基本字符集开始逐渐合并的过程，每次合并都在词表中增加一个新的子词。
- 区别在于，BPE在合并两个子词的时候，只需要计算两个子词连续出现的概率，而WordPiece则复杂得多。WordPiece首先需要根据给定的子词词表构造一个n-gram语言模型，然后用这个语言模型可以计算整个训练数据出现的似然率（likelihood）。合并的时候，考虑所有的子词对儿（subword pair），根据该子词对儿合并后得到的新的语言模型计算整个训练数据的似然率，选择导致整个训练数据似然率最高的那个子词对儿进行合并。这个过程比较复杂，计算代价很高。Google估计是设计了某种优化算法才有可能实现高效的词表构建，但这个算法没有公开。
- WordPiece的子词切分Tokenization应该是使用了基于n-gram语言模型来搜索最佳切分。
- WordPiece试图基于n-gram语言模型来找到最合理的词表训练和切分方法，理论上很漂亮，但可能实现上比较复杂，Google最终也没有公开源代码，具体实现细节也还不清楚。最终被埋没，有点可惜。

Unigram子词切分算法

- Google还提出了另外一种词表构造方法，叫做Unigram方法(Kudo 2018)。
- Unigram跟WordPiece一样，也采用了语言模型来决定词表中是否收录某个子词。WordPiece原始论文Schuster & Nakajima(2012)并没有明确说明所使用的是哪一个n-gram语言模型，而Unigram方法使用的是一元语言模型，这也是这种方法取名为Unigram的原因。
- 不过跟BPE和WordPiece不同的是，Unigram构造词表的方式不是先构造字符级别词表，再通过合并操作逐渐往词表中增加新的子词，相反，Unigram先构造一个大的子词词表（比如收录语料库中所有词），然后再逐渐删除词表中的子词（把罕见的子词切分成两个更常见的子词），直到词表长度达到预定的值。
- Unigram词表构造方法在一个开源的工具SentencePiece（后面会提到）中提供了具体实现，但运行效率比较低，实际上使用这种方法的人比较少。

SentencePiece: 与语言无关的子词切分方法

- 仔细研究上面介绍的子词切分方法，我们会看到，我们在做子词切分（subword tokenization）之前，还需要先做一次词语切分（word tokenization），以得到一个初始的词典（也就是前面BPE词表构造方法中的Dictionary）。但词语切分也并不是一件简单的事情，比如：
  - 每一种语言都有各自的词语切分问题，词语切分无法做到语言无关。
  - 在中日韩等语言中，字符（主要是汉字）之间是没有空格的，需要专门引入外部的词语切分工具进行词语切分；
- SentencePiece(Kudo & Richardson 2018)就是为了解决这一问题提出的子词切分方法。

SentencePiece: 与语言无关的子词切分方法

- SentencePiece的基本思想是，对于输入的文本无需先做词语切分，直接在输入文本串的基础上构造子词词表和进行子词切分。直观地理解，可以认为输入的每个句子（甚至段落）就是一个单词，然后再构造词表进行子词切分。
- 在这个方法中，空格不作为词语之间的分隔符，也就是说，空格与其他任何字符同等对待。按照这种方式得到的子词词表，子词中出现空格，或者子词中同时出现字母、数字或者标点符号都是很常见的。
- 论文Kudo & Richardson (2018)的实验表明，采用SentencePiece的方法，无需先做词语切分，可以取得跟先做词语切分方法可比的效果。
- SentencePiece解决了LLM Tokenizer适应多语言的问题，目前被广泛采用。

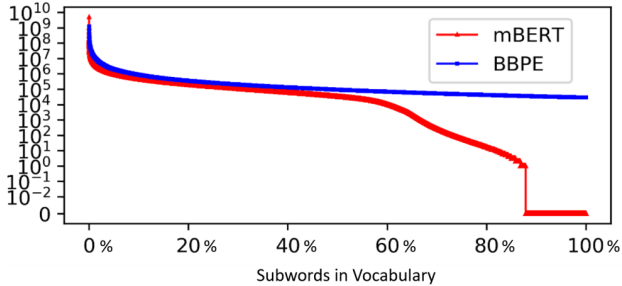
SentencePiece: 与语言无关的子词切分方法

- SentencePiece和WordPiece这两个名词有点容易造成混淆，让人以为SentencePiece是跟WordPiece一样的一种子词切分方法。实际上这是两个不同层次的概念。WordPiece和BPE、Unigram是类似的技术，都是用于构造子词词表和做子词切分的方法。SentencePiece方法只是强调子词切分之前无需做词语切分，直接从句子开始构造子词词表和进行子词切分，但并没有规定子词词表构造和子词切分的算法。所以即使采用SentencePiece方法，还是需要选择WordPiece、BPE或者Unigram来做子词词表构造和子词切分的。
- 另外要注意的是，SentencePiece不仅仅是一种子词切分方法，也是一套开源的工具。这套工具提供了两个子词词表构造的工具：BPE和Unigram。

Byte-level BPE (BBPE): 解决大字符集语言的OOV问题

- 基于子词切分的方法，对于中日韩这样的大字符集语言来说，还是有问题需要解决：
  - 基本字符太多，甚至可能超过预定的词表大小，导致一些字符无法收入子词词表，仍然会有OOV问题；
  - 即使字符集足够大，把所有字符都收入子词词表，这会导致词表中收入大量罕见字，词表空间利用率大大降低，而在解码的时候，需要在词表的所有子词空间上计算softmax，大量的罕见字出现概率极低，这样也导致解码的时间效率大大降低。
- Byte-level BPE（简称BBPE）为这一问题的解决提供了有效的办法。
- BBPE的基本思想是：把所有的字符表示成UTF-8格式，把这个UTF-8字符串当成一个字节组成的序列，然后以一个字节为最小的组成单位，来构造BPE子词词表和进行子词切分。这样的话，BPE最基本的组成单位最多只有256个，不存在大量罕见字符占用子词词表空间的问题。
- SentencePiece和BBPE结合，基本上为多语言大字符集的tokenization提供了理论上较为完美的解决方案。目前已经被主流的LLM所采用。但在应用中还是有一些小问题。

Byte-level BPE (BBPE): 解决大字符集语言的OOV问题



mBERT词表和BBPE词表中的子词在训练数据中出现的概率分布

从图中可以看出，mBERT词表的频率分布在超过词表长度60%以后就快速下降，在超过83%以后直接降到0。说明mBERT的词表使用是非常低效的。即使这样mBERT仍然存在OOV问题；我们在使用中发现，著名作家“章诒和”的“诒”如果写成繁体“詒”，就无法被mBERT识别。

Content

面向LLM的Tokenizer的评价指标

- 面向LLM的Tokenizer的评价指标
  - 词表大小：词表越大，占用参数空间越多。
  - 压缩率：可以用Fertility来度量，用于计算一个词Word平均被切分成多少个token，越少越好。压缩率越高，同样长的上下文序列就可以处理更长的文本。
- 词表大小和压缩率是两个互相矛盾的指标。为了达到更大的压缩率，通常需要使用更大的词表。
- 理论上，词表的参数空间和Transformer模型本身的参数空间是可以互换的。
  - 词表大小可以做到很小，只要充分训练，模型依然可以得到很好的性能。
  - 论文Xu et al. (2018)把词表搜索问题转成了一种最优运输问题，论证了模型存在一个最右的词表大小。该论文在机器翻译上的实验表明，目前主流的神经翻译系统词表都太大了，把词表缩小到三分之一性能可能还能略有提高。但这个任务跟LLM有所不同，因为LLM是面向通用的语言生成任务的，并不像机器翻译那样有个单一的评价指标。
  - 词表大小和压缩率不仅仅是影响模型性能，也影响模型效率。压缩率越高，模型效率越高。

Content

词表分布问题

- 词表中的subword在各个语种、各个领域上存在一个分布，这种分布是由词表的训练数据决定的。
- GPT-3/4的词表训练数据中以英文为主，导致对其他语种覆盖不足
  - 以中文为例，GPT-3最初的训练数据中中文所占比例大约为千分之一，这就导致词表中中文所占比例很低，实际只有几百个subword是中文
  - 根据BBPE的切分原则，绝大部分汉字都被切分成3个或者2个token，而一个英文单词平均大约被切分成1.2个token。所以GPT-3/4的tokenizer对中文是非常不友好的，中文压缩率极低，按token数计算的费用也比英文高得多。
  - GPT-4o改进了tokenizer，新的tokenizer加入了很多多语言subword，但中文的subword数量依然只有几千个，远远少于英文的subword数量（数万个）。
  - 国产的LLM的词表通常都会包含数万个中文subword，因此中文的压缩率会高得多。

词表扩增问题

- 词表的训练数据通常比较有限，跟模型的预训练数据相比会小得多，而且预训练数据会不断扩充，这就会导致词表的分布不合理问题：
  - 新增加的语种（如阿拉伯语、俄语、新的编程语言）或者领域（如医疗），可能会出现很低的压缩率，降低模型的性能和效率。
- 采用BPE的词表训练方法，可以对词表进行增量式扩充，可以较好解决上述问题。
- 经分析表明，千问的新模型词表就是直接从Llama3的词表扩充而来，因为词表前面大部分subword和排序都是完全相同的，只是后面扩增了中文的subword。

词表训练不足问题

- 按照BPE的词表训练方法，有可能导致位置靠前的词，实际出现的频率反而低：
  - 举例来说，一个词表中有【蜻蜓】和【蜻】【蜓】三个subword，但训练数据中【蜻蜓】两个字几乎都是同时出现的，导致【蜻】和【蜓】这两个subword的概率极低，训练严重不足。
- 词表训练不足会导致严重的后果：
  - 一旦数据中出现这种严重训练不足的token，比如单独出现【蜻】和【蜓】，模型就会出现无法预料的异常输出。
- GPT-4也存在这种词表训练不充分问题，大家可以偶尔观察到GPT-4会输出无厘头的结果。

词表训练不足问题

- Provilkov et al. (2019)提出了解决词表训练不足的一种方法BPE-dropout，其基本思想是：
  - 在语言模型训练阶段做子词切分（Subword Tokenization）的时候，以一定的概率随机跳过一些子词合并（merge）操作，这样通过增加子词切分的随机性，就可以使一些原来训练不充分的低频子词出现概率大大增加，从而学到更好的子词表示。
- 现在模型训练数据越来越大，大家不太重视词表训练不足问题，不过这个问题依然存在。

词表训练不合理问题

- 观察BPE训练出来的中文词表，通常会出现一些很长的subword：

```
185118 _日本毛片免费视频观看
116852  中国福利彩票天天
128031 久久免费热在线精品
154809 无码不卡高清免费v
172750  大发快三大小单双
177431 给主人留下些什么吧
181679  qq的天天中彩票
```

- 原因分析：
  - 词表训练数据没有经过严格的清洗
  - 中文词表没有预切分成单词（所以英文词表不可能出现类似问题）

Tokenizer切分不合理问题

Tokenizer切分不一致问题

- 基于SentencePiece构造的Tokenizer有一个比较严重的问题，就是一个英文单词前面有空格和没有空格的时候，会被切分成不同的token：
  - "I have a book but I do not have a pen"中的两个【I】
  - "x = a + b + c*a"中的两个【a】

Content

基于子词Subword的Tokenization方法的缺陷

- 基于子词Subword的Tokenization方法的缺陷
  - LM不知道一个单词由哪些字母组成
  - LM不知道一个单词由几个字母
  - LM不知道一个单词的第一个字母是什么、最后一个字母是什么
  - LM不知道一个句子有多少个单词（word和token数量不一致）
- GPT-4在发布的时候把写藏头文当成一种涌现能力来展示，说明LM很难掌握一个单词的首字母。

基于字符的Tokenization方法

- 基于字符的Tokenization方法理论上可以完美解决上面的问题
- 但基于字符做Tokenization会导致序列长度增加一个数量级，实现上无法接受。

- 已有一些基于字符的Tokenization工作，但应用很少。

Content

Thank you!

把数字世界带入每个人、每个家庭、
每个组织，构建万物互联的智能世界。

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

HUAWEI