

AIGC算法工程师面经：NLP基础篇——文本解码超全总结

原创 喜欢卷卷的瓦力 瓦力算法学研所 2024年06月07日 09:09 广东

◇◇ 技术总结专栏 ◇◇

作者：vivida



瓦力算法学研所

我们是一个致力于分享人工智能、机器学习和数据科学方面理论与应用知识的公众号。我...
117篇原创内容

公众号

从分词、词表优化、词向量、词频、解码以及构建一个完整的文本分类模型讲解间NLP基础面试题。
本篇主要讲解码部分。

本篇开始继续讲NLP基础，前面三期文章如下，没看过的小伙伴强烈推荐看一下~

AIGC算法工程师面经：NLP基础篇——从分词、词表优化、词向量、词频、解码到设计分类模型
(一)

AIGC算法工程师面经：NLP基础篇——从分词、词表优化、词向量、词频、解码到设计分类模型
(二)

AIGC算法工程师面经：NLP基础篇——从分词、词表优化、词向量、词频、解码到设计分类模型
(三)

本系列将从分词、词表优化、词向量、词频、解码的理论和实践代码讲起，最终会将所有知识化零为整搭建一个完整的文本分类模型。

本篇主要讲解码部分。之前的面试题目只举例了Beam search、Random sample策略，但由于文本解码涉及到的基础内容较多，本篇不严格按照面试问题来，会从概述、算法细节以及简单可上手的代码讲述。

下面是一个快捷目录。

一、前言

1. 文本解码
2. 解码策略分类

二、贪婪采样 (Argmax Decoding)

1. Greedy Search
2. Beam Search
3. Beam Search的优化

4. 总结

三、随机采样 (Stochastic Decoding)

1. Temperature Sampling

2. Top-k Sampling

3. Top-p Sampling (Nucleus Sampling)

4. 总结

四、手撕beam search代码 (简易可直接上手版)

一、前言

1. 文本解码

先用比较通俗的意思解释一下解码：

在生成文本结果的时候，模型的输出是一个时间步一个时间步依次获得的，而且前面时间步的结果还会影响后面时间步的结果。

也就是说，每一个时间步，模型给出的都是基于历史生成结果的条件概率。

那么解码就是：**为了生成完整的句子，融合模型多个时间步的输出，使得最终得到的序列的每一步条件概率连乘起来最大。**

2. 解码策略分类

文本生成中的decoding strategy主要可以分为两大类：

- Argmax Decoding: 主要包括beam search, class-factored softmax等
- Stochastic Decoding: 主要包括temperature sampling, top-k sampling等。

在生成模型中，最终会生成一个大小固定的hidden state h_c ；基于输入句子的hidden state h_c 和先前生成的第1到t-1个词 $x_{1:t-1}$ ；

Decoder会生成当前第t个词的hidden state h_t ，最后通过softmax函数得到第t个词 x_t 的vocabulary probability distribution $P(x|x_{1:t-1})$ 。

两类decoding strategy的主要区别就在于，如何从vocabulary probability distribution $P(x|x_{1:t-1})$ 中选取一个词 x_t ：

- Argmax Decoding的做法是选择词表中probability最大的词，即 $x_t = \operatorname{argmax} P(x|x_{1:t-1})$ ；
- Stochastic Decoding则是基于概率分布 $P(x|x_{1:t-1})$ 随机sample一个词 x_t ，即 $x_t \sim P(x|x_{1:t-1})$ 。

在做seq predction时，需要根据假设模型每个时刻softmax的输出概率来sample单词，合适的sample方法可能会获得更有效的结果。

二、贪婪采样 (Argmax Decoding)

1. Greedy Search

Timestep	1	2	3	4
A	0.5	0.1	0.2	0.0
B	0.2	0.4	0.2	0.2
C	0.2	0.3	0.4	0.2
<eos>	0.1	0.2	0.2	0.6

- 核心思想

每一步取当前最大可能性的结果，作为最终结果。

- 具体方法

获得新生成的词是vocab中各个词的概率，取argmax作为需要生成的词向量索引，继而生成后一个词。

2. Beam Search

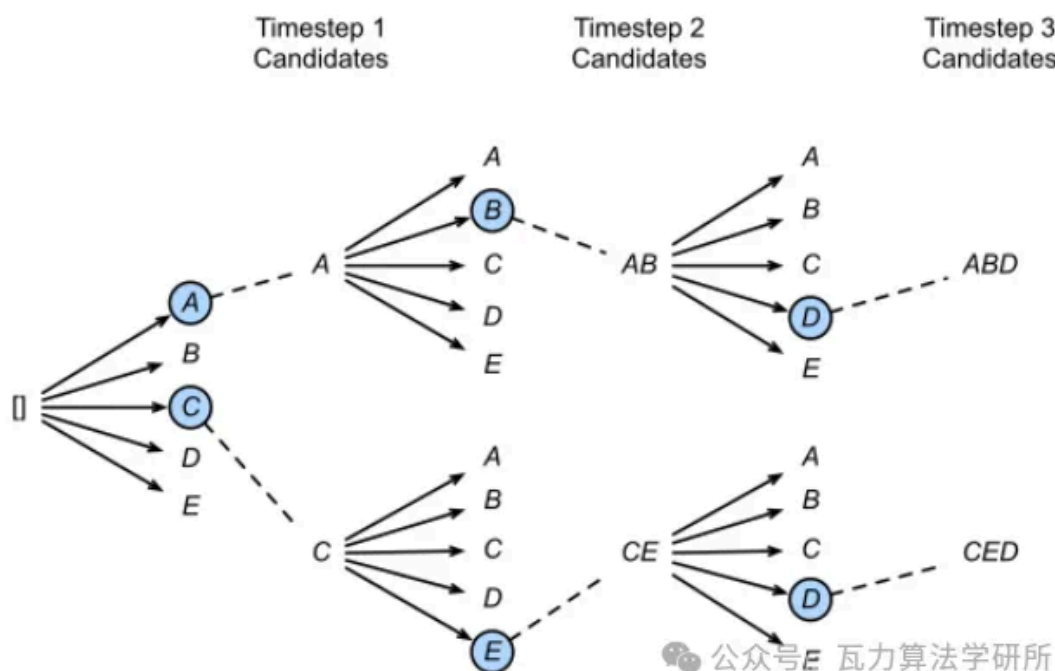
- 核心思想

beam search尝试在广度优先基础上进行进行搜索空间的优化（类似于剪枝）达到减少内存消耗的目的。

- 具体方法

在decoding的每个步骤，都保留着 top K 个可能的候选单词，然后到了下一个步骤的时候，对这 K 个单词都做下一步 decoding，分别选出 top K，然后对这 K^2 个候选句子再挑选出 top K 个句子。以此类推一直到 decoding 结束为止。

- 举例



- 1) 在第一个时间步，A和C是最优的两个，因此得到了两个结果[A],[C]，其他三个就被抛弃了；
- 2) 第二步会基于这两个结果继续进行生成，在A这个分支可以得到5个候选，[AA],[AB],[AC],[AD],[AE]，C也同理得到5个，此时会对这10个进行统一排名，再保留最优的两个，即图中的[AB]和[CE]；
- 3) 第三步同理，也会从新的10个候选里再保留最好的两个，最后得到了[ABD],[CED]两个结果。可以发现，beam search在每一步需要考察的候选数量是贪心搜索的num_beams倍，因此是一种牺牲时间换性能的方法。

3. Beam Search的优化

- 惩罚短句：Length normalization

Beam Search倾向于选择最短的句子，因为在这个连乘操作中，每个因子都是小于1的数，因子越多，最后的概率就越小。

解决这个问题的方法，最后的概率值除以这个生成序列的单词数，这样比较的就是每个单词的平均概率大小。此外，连乘因子较多时，可能会超过浮点数的最小值，可以考虑取对数来缓解这个问题。谷歌给的公式如下：

$$s(Y, X) = \log(P(Y|X)) / lp(Y) + cp(X; Y)$$

$$lp(Y) = \frac{(5 + |Y|)^\alpha}{(5 + 1)^\alpha}$$

$$cp(X; Y) = \beta * \sum_{i=1}^{|X|} \log(\min(\sum_{j=1}^{|Y|} p_{i,j}, 1.0)),$$

公众号E 瓦力算法学研所

其中 $\alpha \in [0, 1]$ ，谷歌建议取值为[0.6, 0.7]之间， α 用于length normalization。

- **Coverage normalization: 惩罚重复**

序列到序列任务中经常会发现一个问题，2016 年， 华为诺亚方舟实验室的论文提到， 机器翻译的时候会存在over translation or undertranslation due to attention coverage。

作者提出coverage-based attention机制来解决coverage 问题。Google machine system 利用了如下的方式进行了length normalization 和 coverage penalty。

跟上述公式差不多， β 用于控制coverage penalty

$$\begin{aligned} s(Y, X) &= \log(P(Y|X)) / lp(Y) + cp(X; Y) \\ lp(Y) &= \frac{(5 + |Y|)^\alpha}{(5 + 1)^\alpha} \\ cp(X; Y) &= \beta * \sum_{i=1}^{|X|} \log(\min(\sum_{j=1}^{|Y|} p_{i,j}, 1.0)), \end{aligned}$$

coverage penalty 主要用于使用 Attention 的场合，通过 coverage penalty 可以让 Decoder 均匀地关注于输入序列 X 的每一个 token，防止一些 token 获得过多的 Attention。

- **End of sentence normalization: 抑制长句**

有的时候我们发现生成的序列一直生成下去不会停止，有的时候我们可以显式的设置最大生成长度进行控制，这里我们可以采用下式来进行约束：

$$ep(X, Y) = \gamma \frac{|X|}{|Y|}$$

其中 $|X|$ 是source的长度， $|Y|$ 是当前target的长度，那么由上式可知，target长度越长的话，上述得分越低，这样就会防止出现生成一直不停止的情况。

4. 总结

Greedy Search和Beam Search存在的问题：

- 容易出现重复的、可预测的词；
- 句子/语言的连贯性差。

三、随机采样 (Stochastic Decoding)

1. Temperature Sampling

- 核心思想

根据单词的概率分布随机采样

- 具体方法

在softmax中引入一个temperature来改变vocabulary probability distribution，使其更偏向high probability words：

$$P(x|x_{1:t-1}) = \frac{\exp(u_t/\text{temperature})}{\sum_v \exp(u_v/\text{temperature})}, \text{temperature} \in [0, 1)$$

另一种表示：假设 $p(x)$ 为模型输出的原始分布，给定一个 temperature 值，将按照下列方法对原始概率分布（即模型的 softmax 输出）进行重新加权，计算得到一个新的概率分布。

$$\pi(x_k) = \frac{e^{\log(p(x_k))/\text{temperature}}}{\sum_{i=1}^n e^{\log(p(x_i))/\text{temperature}}}, \text{temperature} \in [0, 1)$$

当 $\text{temperature} \rightarrow 0$ ，就变成greedy search；当 $\text{temperature} \rightarrow \infty$ ，就变成均匀采样（uniform sampling）。详见论文：[The Curious Case of Neural Text Degeneration]

2. Top-k Sampling

可以缓解生成罕见单词的问题。比如说，我们可以每次只在概率最高的50个单词中按照概率分布做采样。我只保留top-k个probability的单词，然后在这些单词中根据概率做sampling。

- 核心思想

对概率进行降序排序，然后对第k个位置之后的概率转换为0。

- 具体方法

在decoding过程中，从 $P(x|x_{1:t-1})$ 中选取probability最高的前k个tokens，把它们的probability加总得到 $p' = \sum P(x|x_{1:t-1})$ ；

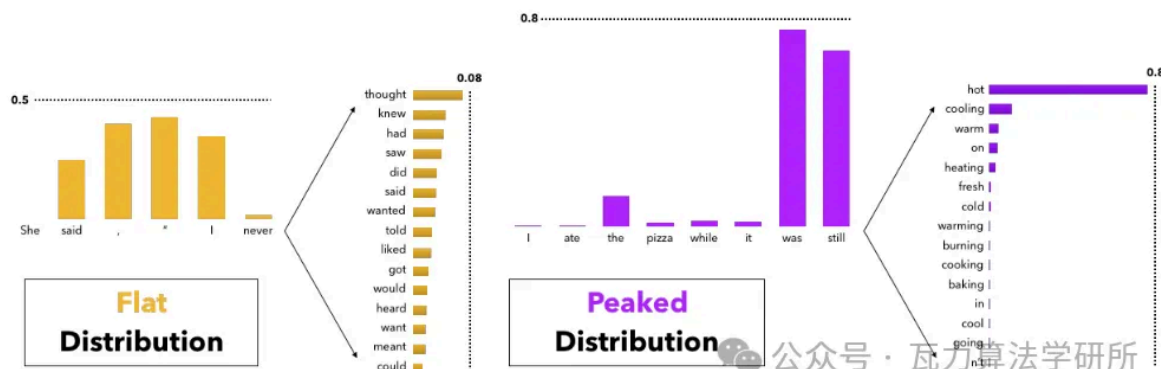
然后将 $P(x|x_{1:t-1})$ 调整为 $P'(x|x_{1:t-1}) = P(x|x_{1:t-1})/p'$ ，其中 $x \in V(k)$ ；

最后从 $P'(x|x_{1:t-1})$ 中sample一个token作为output token。详见论文：[Hierarchical Neural Story Generation]

- 存在的问题

Top-k Sampling存在的主要问题是，**k不太好选**：

常数k是提前给定的值，对于长短大小不一，语境不同的句子，可能有时需要比k更多的tokens；



此外，因为这个概率分布变化比较大，有时候可能很均匀(flat)，有的时候比较集中(peaked)。对于集中的情况还好说，当分布均匀时，一个较小的k容易丢掉很多优质候选词。但如果k定的太大，这个方法又会退化回普通采样。

3. top-p sampling (Nucleus Sampling)

• 核心理念

通过对概率分布进行累加，然后当累加的值超过设定的阈值p，则对之后的概率进行置0。

• 具体方法

提出了Top-p Sampling来解决Top-k Sampling的问题，基于Top-k Sampling，它将 $p' = \sum P(x| x_{1:t-1})$ 设为一个提前定义好的常数 $p' \in (0,1)$ ，而selected tokens根据句子history distribution的变化而有所不同。

详见论文：[The Curious Case of Neural Text Degeneration]

论文有一个效果对比图，对比了这几种采样方式的效果。红字是前后不符，蓝字是重复。Nucleus效果拔群。



WebText

An unprecedented number of mostly young whales have become stranded on the West Australian coast since 2008.

Beam Search, $b=16$

The number of stranded whales has increased by more than 50 per cent in the past year, with the number of stranded whales on the West Australian coast increasing by more than 50 per cent in the past year. The number of whales stranded on the West Australian coast has increased by more than 50 per cent in the past year, with the number of stranded whales on the West Australian coast increasing by more than 50 per cent in the past year.



Pure Sampling

The Australian Food Safety Authority has warned Australia's beaches may be revitalised this year because healthy seabirds and seals have been on the move. More than 50,000 seabirds, sea mammals and seahorses have been swept into the sea by the Holden CS118 and Adelaide Airport CS300 from 2013. A major white-bat and umidauda migration across Australia is under way in Australia for the first time, with numbers reaching an estimated 50,000.

Sampling, $t=0.9$

Last week's intense storms and a series of powerful cyclones have been officially blamed for the deaths of at least nine large fin whales near Whitsundays - the largest loss of any species globally. The fin whales: packed in the belly of one killer whale thrashing madly in fear as another tries to bring it to safety. When the colossal animal breached the waters of Whitsundays, he'd been seen tagged for a decade.

Top-k, $k=640$

Pumping Station #3 shut down due to construction damage Find more at:

www.abc.net.au/environment/species-worry/in-the-top-10-killer-whale-catastrophes-in-history.html

"In the top 10 killer whale catastrophes in history:

1) 1986: Up to 12 orcas struck by lightning; many drowned and many more badly injured.

Top-k, $k=40$, $t=0.7$

The whale's fate was confirmed late last week when the animal was found by fishermen off the coast of Bundaberg. Experts believe the whale was struck by a fishing vessel off the coast of Bundaberg, and died after being sucked into the ocean. The whale's fate was confirmed late last week when the animal was found by fishermen off the coast of Bundaberg.

Nucleus, $p=0.95$

There has been an unprecedented number of calves caught in the nets of whaling stations that operate in WA. Pilot whales continue to migrate to feeding grounds to feed their calves. They are now vulnerable due to the decline of wild populations; they are restricted to one breeding site each year. Image copyright Yoon Bo Kim But, with sharp decline in wild populations the size of the Petrels are shrinking and dwindling population means there will only be room for a few new fowl.



WebText

Poor nutrition has led to a rise in the number of stranded humpback whales on the West Australian coast, veterinary researchers have said. Carly Holyoake, from Murdoch University, at the Australian Veterinary Association's annual conference in Perth on Wednesday, said an unprecedented number of mostly young whales had become stranded on the coast since 2008.

本质上Top-p Sampling和Top-k Sampling都是从truncated vocabulary distribution中sample token，区别在于置信区间的选择。

4. 总结

随机采样存在的问题：

- 生成的句子容易不连贯，上下文比较矛盾。
- 容易生成奇怪的句子，出现罕见词。

四、手撕beam search代码

下面我们来写一个基础版本的beam search函数，由于大部分提供的库写得比较复杂，涉及Beam类以及其中各个 score函数、extend函数，但实际面试过程中可能没法这么写，因此这里总结一个简便又便于理解的版本。

详细步骤如下：

1. 初始化Result列表用来存储每次得到的最大k个概率结果，初始化为[[list(),1]] 1为当前初始化的成绩
2. 遍历最大解码长度S（解码出来S个字），
3. 遍历Result，用来为每个当前为止的最大k个结果解码出候选集
4. 每个解码出的k个结果统一存储在Candidate列表中
5. 按照成绩选取前k个作为Result，继续遍历，直到解码出S长度或者<eos>

代码如下

```
1 from math import log
2 from numpy import array
3 from numpy import argmax
4
5 # 束搜索
6 def beam_search_decoder(data, k):
7     sequences = [[list(), 1.0]]#初始化存储最后结果的列表，存储k个
8     # 遍历序列中的每一步
9     for row in data:#序列的最大长度
10         all_candidates = list()
11         # 扩展每个候选项，即解码当前所得序列的下一个字
12         for i in range(len(sequences)):
13             seq, score = sequences[i]
14             for j in range(len(row)):#计算每个词表中的字的成绩
15                 candidate = [seq + [j], score * -log(row[j])]
16                 all_candidates.append(candidate)
17             # 根据分数排列所有候选项
18             ordered = sorted(all_candidates, key=lambda tup:tup[1])
19             # 选择k个最有可能的
20             sequences = ordered[:k]
21     return sequences
22
23 # 定义一个由10个单词组成的序列，单词来自于大小为5的词汇表
24 data = [[0.1, 0.2, 0.3, 0.4, 0.5],
25         [0.5, 0.4, 0.3, 0.2, 0.1],
26         [0.1, 0.2, 0.3, 0.4, 0.5],
27         [0.5, 0.4, 0.3, 0.2, 0.1],
28         [0.1, 0.2, 0.3, 0.4, 0.5],
29         [0.5, 0.4, 0.3, 0.2, 0.1],
30         [0.1, 0.2, 0.3, 0.4, 0.5],
31         [0.5, 0.4, 0.3, 0.2, 0.1],
32         [0.1, 0.2, 0.3, 0.4, 0.5],
33         [0.5, 0.4, 0.3, 0.2, 0.1]]
34 data = array(data)
```

```
35 # 解码输出序列
36 result = beam_search_decoder(data, 3)
37 # 打印结果
38
39 for seq in result:
40     print(seq)
```

另外的大模型常用的解码代码也可以参考大模型面经之解码策略topk、topp（附代码）这篇。

参考文献

- [1] 《Python深度学习》第8章第1节：8.1 使用LSTM生成文本P228-P234。
- [2] The Curious Case of Neural Text Degeneration: [https://arxiv.org/abs/1904.09751]
- [3] CTRL: A Conditional Transformer Language Model for Controllable Generation: [https://arxiv.org/abs/1909.05858]
- [4] The Curious Case of Neural Text Degeneration [https://arxiv.org/abs/1904.09751]
- [5] Hierarchical Neural Story Generation [https://arxiv.org/abs/1805.04833]

想要获取技术资料的同学欢迎关注公众号，进群一起交流~



喜欢卷卷的瓦力

扫一扫上面的二维码图案，加我为朋友。

添加瓦力微信

算法交流群 · 面试群

大咖分享 · 学习打卡

👤 公众号 · 瓦力算法学研所



瓦力算法学研所

我们是一个致力于分享人工智能、机器学习和数据科学方面理论与应用知识的公众号。我...
117篇原创内容

公众号