CANINE: 一个高效的免词元化编码器

摘要

基于流水线的 NLP 系统在很大程度上已被端到端神经建模所取代，但几乎所有常用模型仍然需要明确的标记化步骤。虽然最近基于数据派生的子词词典的标记化方法比手动设计的标记器更健壮，但这些技术并不同样适用于所有语言，并且任何固定词汇表的使用可能会限制模型的适应能力。在本文中，我们介绍了 CANINE，这是一种直接对字符序列进行操作的神经编码器——无需显式标记化或词汇化，以及一种直接对字符进行操作的预训练策略。为了有效且高效地使用其更细粒度的输入，CANINE 将减少输入序列长度的下采样与对上下文进行编码的深度转换器堆栈相结合。CANINE 在 TYDI QA（一项多语言基准测试）上的性能优于同类 mBERT 模型 5.7 F1%，同时模型参数较少。

关键词：无标记化；CANNIE；下采样

1 引言

世界上存在海量的语言与词汇，在处理多语言场景时，传统预训练模型采用的 Vocabulary 和 Tokenization 方案难免会遇到 Out of Vocabulary 和 Unkonw Token 的情况。Canine 提供了 Tokenization-free 的预训练模型方案，提高了模型在多语言任务下的能力。

2 相关工作

2.1 子词词元化的改进

人们提出了对标准子词词元化的进一步改进的方法，例如 BPE[1]、WordPiece[2]和 SentencePiece[3]。训练期间的确定性分割使得我们不能对单词的形态进行充分利用，基于此，子词正则化[4]和 BPE-dropout[5]在训练输入的多个词元化结果中随机取样，子词词汇固有的模糊性解释了上述做法的合理性。最近，这种方法被进一步扩展，加强了在不同区段上进行预测的一致性[6]。Unigram LM 自上而下地构建词汇表，相较于 BPE，它在预训练编码器上可以更好地与形态学保持一致。

也有其他人构建了多种粒度的混合模型，他们将字符与词元[7]或者不同的子词词汇结合起来[8]以期待更好的表现效果。

2.2 字符级模型

从 NLP 的大趋势来看，字符级的 n-gram 模型大多已经被神经网络取代。虽然字符级模型通常落后于词级模型，但是字符级特征对于语言的丰富形态很重要，特别是在低资源的环境下[9]。

对于语言建模，字符语言模型使用了 vanilla RNN 架构，以纯粹的无词元化方式产生字符序列的分布[10]。Chung 等人[11]联合训练了一个子模块，将字符级输入在堆叠式 LSTM 的每一层分割成较大的跨度。由于在性能上始终落后于词级，人们的注意力从纯粹的 CLMs 转移到纯字符意识的模型，仍然依赖于传统的词元化。一些混合模型在字符水平上处理输入，但预测阶段的词汇却来自封闭的词汇表。其他一些模型在输入端重新引入了显式的词元化，要么是产生了大量的字符序列，这些字符序列形成了一个开放词汇表，要么在封闭词汇生成器产生罕见或未知的词元时使用纯字符生成器。特别是在像 BPE 这样固有的含糊不清的子词词汇表普及之后，一些研究从单一的输入分割种跳出来，将所有可能的分割边缘化[12]。

在 Transformer[13]取代 RNNs 成为 NLP 中的主流架构后，字符级模型也随之而来。Al-Rfou 等人的研究向我们展示了字节级的 vanilla Transformer 的表现明显低于词级[14]。Radford 等人也报告了类似的发现[15]。虽然差距在缩小[16]，但子词级 Transformer 仍是纯语言建模的主流。

2.3 多语言模型

深度预训练的多语言模型在多语言 NLP 任务起主导作用，其子词词汇是跨语言共享的。这类模型参考了单语言模型的架构，并在 100 多种语言中进行联合训练，同时使用无监督的 LM 损失。

3 本文方法

3.1 本文方法概述

CANINE 主要由三个部分组成：(1) 一种用于嵌入文本的无词汇技术；(2) 一种通过下采样和上采样实现的高效字符级模型；(3) 一种在字符级模型上进行掩码语言建模的有效手段。

3.2 模型

CANINE 的设计是对现代编码器（如 GPT、(m)BERT、XLM 和 XLM-R）中的深层 Transformer 堆栈进行最小的修改，因此其结构很容易被该系列的其他模型所采用。这种字符模型的最简单实现是在每个输入位置上用字符代替子词。然而，在相同的输入文本下，这种方法需要更多的序列位置，从而导致前馈层的计算量直线上升，而在 self-attention 层的计算量则呈四倍上升。

CANINE 模型的总体形式是由一个下采样函数 DOWN、一个主编码器 ENCODE 和一个上采样函数 UP 组成；给定一个长度为 n、维度为 d 的字符嵌入序列 $\mathbf{e} \in \mathbb{R}^{n \times d}$ 的输入，有：

$$\mathbf{Y}_{seq} \leftarrow \text{Up}(\text{ENCODE}(\text{DOWN}(\mathbf{e}))$$

其中 $\mathbf{Y}_{seq} \in \mathbb{R}^{n \times d}$ 是序列预测任务的最终表现形式。同样地，对于分类任务，模型只需使用主编码器的第 0 个元素，即：

$$\mathbf{Y}_{cls} \leftarrow [\text{Up}(\text{ENCODE}(\text{DOWN}(\mathbf{e}))]_0$$

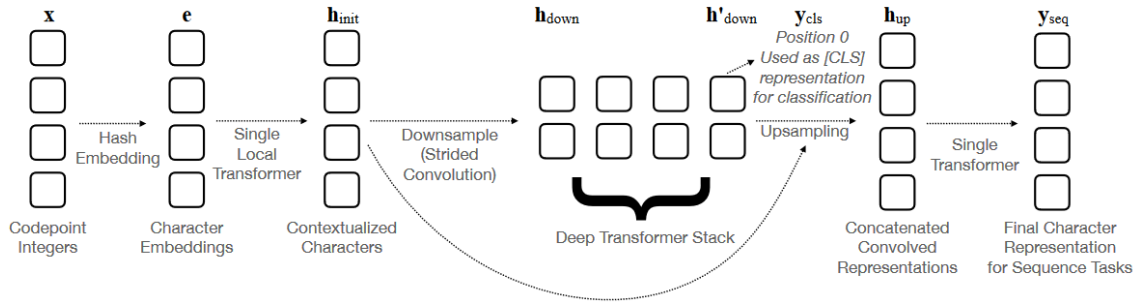

图 1: CANINE 神经结构

预处理。和现有模型一样，CANNIE 的输入最终必须被表示为一个整数序列，但由于 Unicode 已经将字符定义和标准化，通常成百上千的预处理代码可以被一个非常简单的程序所取代：只需遍历输入字符串种的字符，并返回其代码点的整数值。此外，由于代码点值是 Unicode 标准的一部分，这是被公开记录的，且已经被编程语言所支持，它们不会随着时间的推移而变化，这与基于词汇的任意 ID 不同。

字符哈希嵌入。CANNIE 使用散列法来使得用相对较少的参数嵌入 Unicode 码点的全部空间，考虑到不同的码点可能有完全相同的表示，我们对每个码点应用多个散列函数，并将各种散列值的表示

拼接起来。

更形式化地，给定一个码点 $x_i \in \mathbb{N}$，我们应用 K 个散列函数 $H_k : \mathbb{N} \to \mathbb{N}$，在自己嵌入矩阵中查找每个散列结果 $\varepsilon_k \in \mathbb{R}^{B \times d'}$，产生 K 个大小为 $d' = d/K$ 的嵌入，之后会把他们集中到一个大小为 d 的表示上：

$$e_t \leftarrow \bigoplus_{k}^{K} \text{LOOKUP}\left(H_k(x_i)\%B, \varepsilon_k\right)$$

这里的 $\bigoplus$ 表示向量拼接。我们把这些成为字符嵌入 $\mathbf{e} \in \mathbb{R}^{n \times d}$。在实验中，采用 $d = 768, K = 8, B = 16\text{k}$.

虽然每个单独的哈希函数都会受到哈希碰撞的影响，但总体影响很小，因为每个函数只占编码点整体嵌入的一小部分，而且其他哈希函数产生相同碰撞的可能性很小。

因为模型总是包括所有的代码点，所以在微调过程中，有可能学习在预训练中从未见过的字符（以及延伸到单词、脚本等）的表征，同时仍然可以利用预训练学到的单词组成和句子结构。

无词汇的 n-grams。我们也可以重新定义上面的嵌入 e_i，以包括字符 n-grams，这样可以保证没有固定的词汇，使得每个 n-gram 顺序对总嵌入的贡献是相同的：

$$e_i^N \leftarrow \bigoplus_{k}^{K} \sum_{j}^{N} \text{LOOKUP}(H'_k(x_{i\ldots j})\%B, \varepsilon_{j,k})$$

下采样。首先，使用一个单层块局部注意力 transformer 进行编码，这个模型只在每个预定大小的块区内运用自注意力机制，节约了注意力带来的消耗。接下来，使用分层卷积来减少序列位置的数量。给定字符嵌入 $e \in \mathbb{R}^{B \times d}$，序列长度为 n 个字符，维度为 d，我们用一个跨度为 r 的卷积来对序列进行下采样。

$$\mathbf{h}_{\text{init}} \leftarrow \text{LOCALTRANSFORMER}_1(\mathbf{e})$$

$$\mathbf{h}_{\text{down}} \leftarrow \text{STRIDEDCONV}(\mathbf{h}_{\text{init}}, r)$$

Transformer 栈。在下采样之后，CANINE 对产生的下采样位置应用了一个具有 L 层的深度 Transformer，这与 BERT 和其他衍生模型的核心是相同的，在这里会产生很多的计算和参数。

$$\mathbf{h}'_{\text{down}} \leftarrow \text{TRANSFORMER}_L(\mathbf{h}_{\text{down}})$$

$$\mathbf{y}_{\text{cls}} = [\mathbf{h}'_{\text{down}}]_0$$

上采样。上述架构对于分类任务来说是足够的，但有的序列预测任务要求输出与输入序列长度一致，比如标签预测。我们首先将原始字符 transformer 的输出与 transformer 栈产生的下采样表示相连接，从而重建一个按字符排列的输出表示：

$$\mathbf{h}_{\text{up}} \leftarrow \text{CONV}(\mathbf{h}_{\text{init}} \oplus \mathbf{h}'_{\text{down}}, w)$$

$$\mathbf{y}_{\text{seq}} \leftarrow \text{TRANSFORMER}_1(\mathbf{h}_{\text{up}})$$

4 复现细节

4.1 与已有开源代码对比

原文章的主要亮点是对字符对应 Unicode 码点进行处理，与使用子词相比，这样带来的问题是，在相同的输入文本下，这种方法需要更长的序列，这会导致前馈层的计算量直线上升。为了提高模型的效率，使用了上采样和下采样两种策略，但对于 Transformer 堆栈并没有进行优化，使用的仍是 BERT 采用的技术。而传统的 Transformer-base 横型在处理长文本时存在一些问题，因为它们均采用”我全都要看” 的策略，即每一个 token 都要与其他所有 token 进行交互，无论是空间还是时间复杂度都高达 $O(n^2)$。为了解决这个问题，之前有些工作是将长文本切分为若干个较短的文本块，然后逐个处理，例如 Transformer-xL。但这会使得多个文本块之间孤立，因而必然存在大量的信息丢失。当然，我们也可以通过添加一些其它机制来加强文本块之间的交互，但这种新机制实现起来要么很复杂，要么是针对特定任务的，通用性不强。

为了使模型更够更轻松地处理长文本，我们对上述 Transformer 堆栈进行改进，采用 Longformer[17]。Longformer 主要是对 Transformer 传统的自注意力机制进行了改进，具体来说，每一个 token 只对窗口内的 token 进行交互。针对具体任务，Longformer 还在原有的局部注意力机制的基础上增加了一种全局注意力。

具体来讲，Longformer 的提出了三种新的 attention 机制，分别是滑窗机制、空洞滑窗机制、融合全局信息的滑窗机制。

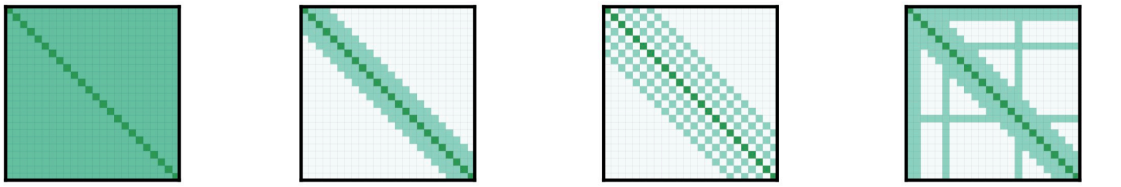

图 2: 几种不同模式的注意力机制的比较

滑窗机制。对于每一个 token，只对其附近的 w 个 token 计算 attention，复杂度为 $O(n^2)$，其中 n 为文本的长度。作者认为，根据应用任务的不同，可以对 Transformer 每一层施以不同的窗口大小 w.

空洞滑窗机制。对每一个 token 进行编码时，普通的滑窗机制只能考虑到长度为 w 的上下文。作者进一步提出空洞滑窗机制（实际上空洞滑窗是 CV 领域中很早就有的一项技术），在不增加计算负荷的前提下，拓宽视野范围。在滑动窗口中，被 attented 到的两个相邻 token 之间会存在大小为 d 的间隙，因此每个 token 的视野范围可达到 $d \times w$。实验表明，由于考虑了更加全面的上下文信息，空洞滑窗机制比普通的滑窗机制表现更佳。

融合全局信息的滑窗机制。我们知道 BERT 类的语言模型在微调时，实现方式略有不同。比如，对于文本分类任务，会在整个输入的前面加上 [CLS] 这个 token；对于 QA 任务，则会将问题与文本进行拼接后进行输入。在 Longformer 中，作者也希望能够根据具体任务的不同，添加少量的全局 attention。对于添加了全局 attention 的 token，在对其编码时要对整个序列做 attention，并且在对其他 token 进行编码时，都要 attend 到它。

4.2 实验环境搭建

paddlenlp==2.3.1

paddlepaddle==2.3.0

absl-py

h5py

4.3 界面分析与使用说明

直接运行代码即可。

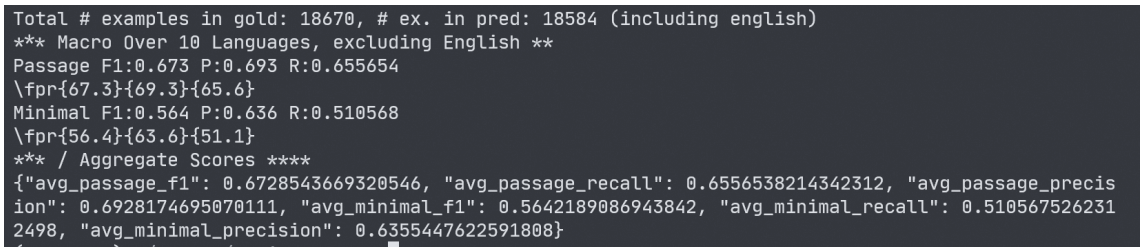

图 3: 代码运行截图

4.4 创新点

我们将 Longformer 中的融合全局信息的滑窗机制用于上述 Transformer 堆栈，这样大大减少了 Transformer 堆栈的计算量和参数量，使得模型能够处理更长文本的同时，进一步提高了模型的效率，同时能够捕捉更全面的上下文信息。

5 实验结果分析

TydiQA 为多语言阅读理解数据集。Tydi 数据库中包含了 18 万 + 篇 wiki 百科语料，20 万 + 文章与问题对，共涉及 11 种不同的语言。Canine 在 TydiQA 上实现了 Selection Passage Task 66% F1 及 Minimum Answer Span Task 58% F1 的精度。

以下复现结果为多次微调、预测、评估后的 macro-F1 平均值。可以看出在 Passage Selection 任务上，自己复现的精度和 CANINE 原文的精度差距仅为 0.05%，可能是由于加入了融合全局信息的滑窗机制，在 Minimal Answer Span 任务上，竟高于 CANINE 精度 2.32%。

表 1: 数据集和复现精度

TydiQA 任务	TydiQA mBert 基线	Canine 精度	自己复现精度
Passage Selection Task (SELECTP)	63.2%	66.0%	65.95%
Minimal Answer Span Task (MINSPAN)	51.2%	52.8%	55.12%

训练过程如下:

表 2: 训练过程

GPU	batch size	acc grad steps	理论 batch size	seed	epoch	TydiQA SelectP F1	TydiQA MinSpan F1
V100	16	1	16	2021	3	66.03%	55.80%
V100	16	1	16	666	3	67.05%	56.23%
V100	16	32	512	5121	0	64.38%	53.67%
V100	16	32	512	555	4	66.32%	54.20%
3090*4	14	9	504	5123	4	65.97%	55.68%

6 总结与展望

本文将在复现 CANINE 的基础上，加入了 Longformer 的融合全局信息的滑窗机制，使得模型能够更全面地捕获上下文信息，同时也减少了计算量和参数量。从复现结果来看，在 Passage Selection 任务上和 CANINE 原文的精度相差仅有 0.05%，在 Minimal Answer Span 任务上的表现则略优于 CANINE，因此可以认为复现较为成功。

CANINE 采用了 tokenization-free 和 vocabulary-free，消除了传统方法在处理多语言时的障碍，同时大大提高了效率，这使得 CANINE 在多语言任务上取得了不错的表现。值得注意的是，在单一语言任务上，CANINE 取得的效果落后于传统方法。因此，如何在使用 tokenization-free 和 vocabulary-free 的基础上，兼顾多语言任务和单语言任务，是我们可以继续探索的一个方向。

参考文献

[1] SENNRICH R, HADDOW B, BIRCH A. Neural machine translation of rare words with subword units [J]. arXiv preprint arXiv:1508.07909, 2015.

[2] WU Y, SCHUSTER M, CHEN Z, et al. Google's neural machine translation system: Bridging the gap between human and machine translation[J]. arXiv preprint arXiv:1609.08144, 2016.

[3] KUDO T, RICHARDSON J. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing[J]. arXiv preprint arXiv:1808.06226, 2018.

[4] KUDO T. Subword regularization: Improving neural network translation models with multiple subword candidates[J]. arXiv preprint arXiv:1804.10959, 2018.

[5] PROVILKOV I, EMELIANENKO D, VOITA E. BPE-dropout: Simple and effective subword regularization[J]. arXiv preprint arXiv:1910.13267, 2019.

[6] WANG X, RUDER S, NEUBIG G. Multi-view subword regularization[J]. arXiv preprint arXiv:2103.08490, 2021.

[7] LUONG M T, MANNING C D. Achieving open vocabulary neural machine translation with hybrid word-character models[J]. arXiv preprint arXiv:1604.00788, 2016.

[8] ZHANG X, LI P, LI H. AMBERT: A pre-trained language model with multi-grained tokenization[J]. arXiv preprint arXiv:2008.11869, 2020.

[9] GARRETTE D, BALDRIDGE J. Learning a part-of-speech tagger from two hours of annotation[C]//Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. 2013: 138-147.

[10] HWANG K, SUNG W. Character-level language modeling with hierarchical recurrent neural networks [C]//2017 ieee international conference on acoustics, speech and signal processing (icassp). 2017: 5720-5724.

[11] CHUNG J, AHN S, BENGIO Y. Hierarchical multiscale recurrent neural networks[J]. arXiv preprint arXiv:1609.01704, 2016.

[12] GRAVE E, SUKHBAATAR S, BOJANOWSKI P, et al. Training hybrid language models by marginalizing over segmentations[C]//Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. 2019: 1477-1482.

[13] VASWANI A, SHAZEER N, PARMAR N, et al. Attention is all you need[J]. Advances in neural information processing systems, 2017, 30.

[14] AL-RFOU R, CHOE D, CONSTANT N, et al. Character-level language modeling with deeper self-attention[C]//Proceedings of the AAAI conference on artificial intelligence: vol. 33: 01. 2019: 3159-3166.

[15] RADFORD A, WU J, CHILD R, et al. Language models are unsupervised multitask learners[J]. OpenAI blog, 2019, 1(8): 9.

[16] CHOE D, AL-RFOU R, GUO M, et al. Bridging the gap for tokenizer-free language models[J]. arXiv preprint arXiv:1908.10322, 2019.

[17] BELTAGY I, PETERS M E, COHAN A. Longformer: The long-document transformer[J]. arXiv preprint arXiv:2004.05150, 2020.