以字为单位的自然语言模型

Yoon Kim
School of Engineering and Applied Sciences
Harvard University
yoonkim@seas.harvard.edu

Yacine Jernite
School of Engineering and Applied Sciences
srush@seas.harvard.edu
Harvard University

David Sontag
Courant Institute of Mathematical Sciences
New York University
jernite@cs.nyu.edu

Alexander M. Rush
Courant Institute of Mathematical Sciences
New York University
dsontag@cs.nyu.edu

摘要

我们描述了一个只依赖以字为单位输入的自然语言模型。我们的预测依然基于以词为单位。我们的模型对于字使用了卷积神经网络 (CNN) 和高速网络，模型的输出将作为输入，输入到由长短期记忆人工神经网络神经元 (LSTM) 组成的循环神经网络 (RNN)。对于 Penn Treebank 英文数据集，我们的模型比现存的主流模型的参数少 60%，却仍然能取得不相上下多结果。对于内含丰富词法的语言（比如阿拉伯语、捷克语、法语、德语西班牙语、俄语），我们的模型的在需要很少的参数的情况下，表现超过 LSTM 的基准线。我们的结果表明，对于很多语言来说，以字为单位的输入，已经足够对语言建模。进一步分析由我们模型的产生的词的表示表明，我们的模型能够基于字编码并且包含语义信息。

1 介绍

语言建模是人工智能和自然语言处理最基础的任务。一个语言模型是对于一系列词的概率分布和转换方法。其中，转换方法通常包括通过计数和子序列拟合对马尔科夫链第 n 次序的假设和估测 n-gram 的概率。基于计数的模型比较容易训练，但是对于稀有词，可能由于数据稀疏而不能很好的估计。

神经语言模型为了解决 n-gram 数据稀疏问题通过参数把词向量化（词嵌入），并把向量词作为神经网络的输入。向量化所使用的参数在训练中学习到。通过神经语言模型获得的词嵌入具有语义上比较接近的词在向量空间比较接近这一性质。

虽然神经语言模型的表现优于计数语言模型，但是该模型不能表示语素信息。比如，该模型不能发现相同前缀的词（例如 eventfull,eventfully,uneventful,uneventfully）是结构上有关联的，他们应该在向量空间上结构相关。因此，稀有词的嵌入可能被错误的估计。这个问题对于词法丰富的语言来说，问题尤其严重

29th Conference on Neural Information Processing Systems (NIPS 2015), Barcelona, Spain.

在这篇文章中，我们提出了一个语言模型，这个模型通过以字为单位的卷积神经网络来获取子词信息。模型的输出将作为 RNN 语言模型的输入。不同于之前使用词素来获取子词信息的模型，我们的模型不需要对词素标记。同样，不同于近期提出的结合词嵌入和字节的模型，我们的模型在输入层不使用词嵌入。因为对于神经语言模型，大多数的参数都是用于生产词嵌入，我们的模型所需的参数相比于之前的模型显著减少。特别适用于对于模型大小有限制的平台。
作为总结，我们的贡献主要有下面几点：

- 对于英语，在 Penn Treebank 英文数据集上，我们的模型比现存的主流模型的参数少 60%，却仍然能取得不相上下多结果
- 对于内含丰富词法的语言（比如阿拉伯语、捷克语、法语、德语、西班牙语、俄语），我们的模型的在需要很少的参数的情况下，表现超过 LSTM 的基准线

2 模型

Figure 1 直观的显示了我们模型的架构。经典的神经语言模型采用词嵌入作为输入，我们的模型采用单层字节 max-over-time 池化卷积神经网络的输出作为输入
作为记号，我们用粗体小写字母标记向量，粗体大写字母标记矩阵，斜体小写字母标记变量，花体大写字母标记集合。为了标记方便，我们假设字和词都已经转换成了索引。

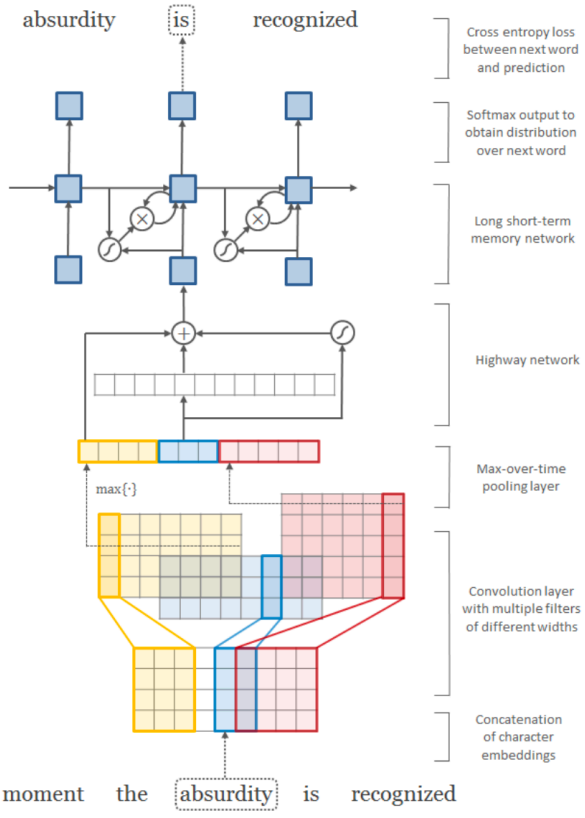

Figure 1: 模型架构

2.0.1 递归神经网络

递归神经网络（RNN）特别适合对于时序模型建模。在每一个时步 t，根据输入向量 $\mathbf{x_i} \in \mathbb{R}^{\mathbf{n}}$、隐式状态 $\mathbf{h_i}$，应用下面的公式产生下一层隐式状态:

$$\mathbf{h_i} = \mathbf{f}((\mathbf{Wx})_\mathbf{i} + \mathbf{Uh_{t-1}} + \mathbf{b}) \tag{1}$$

在这里 $\mathbf{W} \in \mathbb{R}^{\mathbf{mxn}}, \mathbf{U} \in \mathbb{R}^{\mathbf{mxn}}, \mathbf{b} \in \mathbb{R}_\mathbf{m}$ 都是仿射变换的参数，f 是非线性对每个元素的函数。理论上，RNN 中的隐式状态 $\mathbf{h_t}$ 包含了 t 个时步的历史信息。然而在实际中，由于权重指数级爆炸或消失的问题，基本的 RNN 难以捕捉长期时间关联。
长短期记忆（LSTM）通过将 RNN 在每一时步增加一个记忆单元向量 $\mathbf{c_i} \in \mathbb{R}_\mathbf{n}$ 来解决这一问题。具体来讲，LSTM 的每一步利用输入 $\mathbf{x_i}, \mathbf{h_{t-1}}, \mathbf{c_{t-1}}$ 根据下面的公式，计算出 $\mathbf{h_t}, \mathbf{c_t}$:

$$\mathbf{i_t} = \sigma(\mathbf{W^i x_i} + \mathbf{U^i h_{t-1}} + \mathbf{b^i}) \tag{2}$$

$$\mathbf{f_t} = \sigma(\mathbf{W^f x_t} + \mathbf{U^f h_{t-1}} + \mathbf{b^f}) \tag{3}$$

$$\mathbf{o_i} = \sigma(\mathbf{W^o x_t} + \mathbf{U^o h_{t-1}} + \mathbf{b_o}) \tag{4}$$

$$\mathbf{g_t} = \mathbf{tanh}(\mathbf{W^g x_t} + \mathbf{U^g h_{t-1}} + \mathbf{b_g}) \tag{5}$$

$$\mathbf{c_t} = \mathbf{f_t} \odot \mathbf{c_{t-1}} + \mathbf{i_t} \odot \mathbf{g_t} \tag{6}$$

$$\mathbf{h_t} = \mathbf{o_t} \odot \mathbf{tanh}(\mathbf{c_t}) \tag{7}$$

在这里 $\sigma(\cdot)$ 和 $tanh(\cdot)$ 是对每个元素的 sigmoid 函数和双曲正切函数,$\odot$ 是元素间相乘运算符，$\mathbf{i_t}, \mathbf{f_t}, \mathbf{o_t}$ 分别是输入门、遗忘门、输出门。当 $t = 1$ 时，$\mathbf{h_0}$ 和 $\mathbf{c_0}$ 被初始化为零向量。LSTM 需要的参数是 $\mathbf{W_j}, \mathbf{U_j}, \mathbf{b_j}, \mathbf{j} \in \{\mathbf{i}, \mathbf{f}, \mathbf{o}, \mathbf{g}\}$
LSTM 中的记忆单元是随时间变化的，从而缓解了权重指数消失问题。虽然权重指数爆炸仍然是一个问题，但是在实际训练过程中，一些简单的优化策略（例如变化率截断）效果很好。LSTM 在包括语言建模在内的很多任务中显示出比基本 RNN 更好的结果。通过使 $\mathbf{h_t}$ 作为 t 时的输入，可以很容易的将 RNN ／ LSTM 扩展到多层。实际上，在很多任务中，使用多层网络是获得有竞争力结果的关键选择

2.1 Recurrent Neural Network Language Model

记 $\mathscr{V}$ 为固定的词汇集大小。给定历史序列 $w_{1:t} = [w_1, ..., w_t]$, 一个语言模型确定了 w_{t+1} 的概率分布。一个循环神经网络语言模型（RNN-LM) 通过对隐式层应用仿射变换然后再应用 softmax 函数来实现这一点:

$$Pr(w_{t+1} = j \mid w_{1:t}) = \frac{exp(\mathbf{h_t} \cdot \mathbf{p^j} + \mathbf{q^j})}{\sum_{j' \in \mathscr{V}} exp(\mathbf{h_t} \cdot \mathbf{p^{j'}} + \mathbf{q^{j'}})} \tag{8}$$

在这里 $\mathbf{p^j}$ 是 $\mathbf{P} \in \mathbb{R}^{\mathbf{m} \times |\mathscr{V}|}$ 的第 j 列，q^j 是偏差项。同样的，传统的 RNN 语言模型把词作为输入，如果 $w_t = k$，传统模型在时步 t 的输入是 $\mathbf{X} \in \mathbb{R}^{\mathbf{n} \times |\mathscr{V}|}$ 的第 k 列 x^k。我们的模型把词嵌入矩阵 $\mathbf{X}$ 替换成了下面描述的字为单位的卷积神经网络的输出。
我们记 $w_{1:T} = [w_1, ..., w_T]$ 为训练词库的词序列，训练包括最小化该序列的负 log 概率 (NLL), 这一过程通常有啊截断反向传播来实现

$$NLL = -\sum_{t=1}^{T} logPr(w_t \mid w_{1:t-1}) \tag{9}$$

2.1.1 字单位的卷积神经网络

在我们的模型中，t 时刻的输入是字单位的卷积神经网络的输出（CharCNN), 我们将在这里小节里面描述 CharCNN。CNN 在计算机视觉领域的应用已经取得了突出的效果，并且对于多种自然语言处理（NLP）任务也是十分有效的。针对 NLP 应用 CNN 有不同的架构，因为 NLP 需要时序卷积而不是空间卷积。
令 $\mathscr{C}$ 为字的集合，d 为字嵌入的维度，$\mathbf{Q} \in \mathbb{R}^{\mathbf{d} \times |\mathscr{C}|}$ 为字嵌入矩阵。假设词 $k \in \mathscr{V}$ 是由字序列 $[c_1, ..., c_l]$ 组成，l 是词 k 的长度。k 的字表示由矩阵 $\mathbf{C^k} \in \mathbb{R}^{\mathbf{d} \times \mathbf{l}}$ 给出，该矩阵的第 j 列是字 c_j 的字嵌入
我们对 $\mathbf{C^k}$ 和宽度为 w 的过滤器 $\mathbf{H} \in \mathbb{R}^{\mathbf{d} \times \mathbf{w}}$ 应用窄卷积。然后，我们加上偏差，再应用一个非线形函数来获得一个特征图谱 $\mathbf{f^k} \in \mathbb{R}^{\mathbf{l-w+1}}$。$\mathbf{f^k}$ 的第 i 项由下式得出:

$$\mathbf{f^k}[\mathbf{i}] = \mathbf{tanh}(\langle \mathbf{C^k}[*, \mathbf{i} : \mathbf{i + w - 1}], \mathbf{H} \rangle + \mathbf{b}) \tag{10}$$

在这里 $bfC^k[*, i:i+w-1]$ 是 $\mathbf{C^k}$ 的 i 到 $i+w-1$ 列，$\langle \mathbf{A}, \mathbf{B} = \mathbf{Tr}(\mathbf{AB^T}) \rangle$ 是 Frobenius 内积。最后我们应用 max-over-time 作为过滤器 $\mathbf{H}$ 对应的特征

$$y^k = max_i \mathbf{f^k}[\mathbf{i}] \tag{11}$$

这样做的目的是对于一个给定的过滤器，获取最重要的特征，也就是具有最高值的特征。一个过滤器本质上是选出一个字 n 元语法，n 的大小由过滤器宽度决定。
我们通过一个特征由一个过滤矩阵得出描述了这个过程。我们的 CharCNN 使用多个不同宽度的过滤器获得 k 的特征向量。所以如果我们有 h 个过滤器 $\mathbf{H_1}, ...\mathbf{H_h}$, 就有 k 的输入表示 $\mathbf{y^k} = [\mathbf{y_1^k}, ..., \mathbf{y_h^k}]$，对于很多 NLP 应用 h 通常在 $[100, 1000]$ 这个范围内选择。

2.2 高速网络

我们只需要在 RNN-LM 中在 t 时刻，用 $\mathbf{y^k}$ 替换 $\mathbf{x^k}$，正如我们稍后将要证明的，这样就已经能有很高效的表示了。我们也可以对 $\mathbf{y^k}$ 使用一个多层感受器（MLP), 来建模过滤器选择的 n 元语法的交集，但是我们发现这样的表现不是很好。
我们通过将 $\mathbf{y^k}$ 运行在高速网络上进一步提高的我们的表现。高速网络是一层的 MLP 应用了仿射变换之后再施加一个非线形函数，从而得到一组新的特征。

$$\mathbf{z} = \mathbf{g}(\mathbf{Wy} + \mathbf{b}) \tag{12}$$

一层高速网络实现了做了下面的过程：

$$\mathbf{z} = \mathbf{t} \odot \mathbf{g}(\mathbf{W_H y} + \mathbf{b_H}) + (\mathbf{1} - \mathbf{t}) \odot \mathbf{y} \tag{13}$$

在这里 g 是一个非线性函数,$\mathbf{t} = \sigma(\mathbf{W_T y} + \mathbf{b_T})$ 被称作转换门，$(\mathbf{1} - \mathbf{t})$ 被称作进位门。与 LSTM 网络中的记忆单元相似，高速网络会使一些维度的输入直接作为输出。在构建的时候，$\mathbf{y}$ 和 $\mathbf{z}$ 的维度必须相同，$\mathbf{W_T}$ 和 $\mathbf{W_T}$ 都是方阵。

3 实验设置

根据语言建模的标准，我们使用复杂度（PPL) 来衡量我们模型的表现。对于一个序列 $[w_1, ..., w_T]$，一个模型的复杂度由下式给出：

$$PPL = exp(\frac{NLL}{T}) \tag{14}$$

在这里 NLL 是对测试集来计算的。我们针对不同的语言和不同大小的词汇集进行了测试。
针对 Penn Treebank（PTB）数据集，我们进行了高阶参数查询、模型检查和分离学习。使用 0-20 进行标准学习、21-22 进行验证、23-24 进行测试。PTB 有一百万个标签，$|\mathscr{V}| = 10k$, 已经被广泛的用于语言建模。
在针对 PTB 数据集优化高阶参数之后，我们将模型应用于很多词法丰富的语言：捷克语、德语、法语、西班牙语、俄语、阿拉伯语。尽管未经处理的数据是公开可用的，我们从作者那里获取到了预处理过的数据，他们的 NLM 充当了我们模型的基准。我们在每个语言有一百万个标签的小数据集（DATA-S）和 $|\mathscr{V}| = 10k$ 远大于 PTB 的大数据集（DATA-L）上都进行了训练。
在这些数据集中，只有只出现一次的词汇会被 <unk> 替换，所以我们有效的使用了数据集上包含的词汇集。

3.1 优化

我们使用截断反向传播来训练我们的模型。我们使用 stochastic gradient descent 反向传播 35 个时步。学习率初始设为 1.0，当 PPL 没有在每个 epoch 在验证集下降超过 1.0 时，学习率变为原来的一半。
在 DATA-S 上，我们令 batch 大小为 20，在 DATA-L 中我们使用 batch 大小为 100。变化率是每个 batch 的平均。我们对于非阿拉伯语训练 25 个 epoch，对于阿拉伯语训练 30 个 epoch，选出在验证集上表现最好的模型。模型随机初始化为服从 [-0.05,0.05] 上的均匀分布。
我们在 LSTM 从输入到隐式层和 softmax 到输出层使用概率为 0.5 的丢弃法进行正规化学

	DATA-S			DATA-L		
	$\|\mathcal{V}\|$	$\|\mathcal{C}\|$	T	$\|\mathcal{V}\|$	$\|\mathcal{C}\|$	T
English (EN)	10 k	51	1 m	60 k	197	20 m
Czech (CS)	46 k	101	1 m	206 k	195	17 m
German (DE)	37 k	74	1 m	339 k	260	51 m
Spanish (ES)	27 k	72	1 m	152 k	222	56 m
French (FR)	25 k	76	1 m	137 k	225	57 m
Russian (RU)	62 k	62	1 m	497 k	111	25 m
Arabic (AR)	86 k	132	4 m	–	–	–

Figure 2: 词汇集统计数据

习。我们进一步限制变化率的范数小于 5。如果 L_2 规范化的变化率超过 5，在更新之前，我们设置他为 5。变化率范数限制对于训练模型是很重要的。我们的选择很大程度上是受 Zaremba et al. 做的工作的启发。
最后，为了加速 DATA-L 上的学习，我们使用了 hierarchical softmax。Hierarchical softmax 是一个训练 $|\mathcal{V}|$ 大小不同的语言模型的常用策略。我们选择 $c = \lceil\sqrt{|\mathcal{V}|}\rceil$ 个簇，随机的把 $\mathcal{V}$ 分成大小相同的子集 $\mathcal{V}_1, ..., \mathcal{V}_c$, 这样就有：

$$Pr(w_{t+1} = j \mid w_{1:t}) = \frac{exp(\mathbf{h_t} \cdot \mathbf{s^r} + \mathbf{t^r})}{\sum_{r'=1} cexp(\mathbf{h_t} \cdot \mathbf{s^{r'}} + \mathbf{t^{r'}})} \times \frac{exp(\mathbf{h_t} \cdot \mathbf{p_r^j} + \mathbf{q_r^j})}{\sum_{j' \in \mathcal{V}_r} exp(\mathbf{h_t} \cdot \mathbf{p_r^{j'}} + \mathbf{q_r^{j'}})} \tag{15}$$

在这里 r 是簇的索引，有 $j \in \mathcal{V}_r$。上式的第一项是选择第 r 簇的概率，第二项是已知第 r 簇被选的情况下选中词 j 的概率。

4 实验结果

4.1 English Penn Treebank

		Small	Large
CNN	d	15	15
	w	$[1, 2, 3, 4, 5, 6]$	$[1, 2, 3, 4, 5, 6, 7]$
	h	$[25 \cdot w]$	$[\min\{200, 50 \cdot w\}]$
	f	tanh	tanh
Highway	l	1	2
	g	ReLU	ReLU
LSTM	l	2	2
	m	300	650

Figure 3: 小模型和大模型的架构

我们为了评估性能和大小的取舍，对我们的模型训练了两个版本。小模型（LSTM-Char-Small）和大模型 (LSTM-Char-Large) 的架构在表 2 中进行了总结。
和其他基准一样，我们也用词嵌入训练了两个比较模型（LSTM-Word-Small,LSTM-Word-Large)。LSTM-Word-Small 使用了 200 个隐式单元。LSTM-Word-Large 使用了 650 个隐式单元。词嵌入的大小分别是 200 和 650. 这些大小的选择是为了保持和字单位模型相似的参数数目。
正如从表 3 中看到的，我们的模型在参数少 60% 的情况下, 与现在领先的模型取得差不多的表现。我们的小模型明显优于其他大小相似的 NLM。

	PPL	Size
LSTM-Word-Small	97.6	5 m
LSTM-Char-Small	92.3	5 m
LSTM-Word-Large	85.4	20 m
LSTM-Char-Large	78.9	19 m
KN-5 (Mikolov et al. 2012)	141.2	2 m
RNN† (Mikolov et al. 2012)	124.7	6 m
RNN-LDA† (Mikolov et al. 2012)	113.7	7 m
genCNN† (Wang et al. 2015)	116.4	8 m
FOFE-FNNLM† (Zhang et al. 2015)	108.0	6 m
Deep RNN (Pascanu et al. 2013)	107.5	6 m
Sum-Prod Net† (Cheng et al. 2014)	100.0	5 m
LSTM-1† (Zaremba et al. 2014)	82.7	20 m
LSTM-2† (Zaremba et al. 2014)	78.4	52 m

Figure 4: 小模型和大模型的架构

4.2 其他语言

我们模型在 English PTB 上的优异表现表明可以使用它来处理更大规模大数据。然而，英语从词法角度讲是相对简单的，所以我们下一部分的结果主要集中在有更丰富词法的语言上。

我们将我们的结果同 morphological logbilinear (MLBL) 模型相比较。MLBL 通过在输入和输出阶段的语素嵌入也考虑了子词信息。因为在同 MLBL 比较时，我们使用的 LSTM 比 MLBL 使用的 feed-forward 有众所周知的更优异的表现，我们又训练了一个 LSTM 版本的语素 NLM。这个版本中，传入 LSTM 的词表示是词的语素嵌套的和。具体来讲，假设 $\mathscr{M}$ 是一个语言的语素集，$\mathbf{M} \in \mathbb{R}^{\mathbf{n} \times |\mathscr{M}|}$ 是语素嵌入矩阵,$\mathbf{m^j}$ 是 $\mathbf{M}$ 的第 j 列，给定词 k, 下式为输入 LSTM 的词的表示：

$$\mathbf{x^k} + \sum_{\mathbf{j} \in \mathscr{M}^\mathbf{k}} \mathbf{m^j} \tag{16}$$

在这里 $\mathbf{x^k}$ 是词嵌入，$\mathscr{M}_k \subset \mathscr{M}$ 是词 k 的词素集。词素通过在预处理阶段运行一个词素标记非监督算法来获得。词嵌入本身是在词素嵌入的基础上的。词素嵌入对于小模型和大模型的大小分别是 200/650。我们进一步训练词单位的 LSTM 作为另一个基准.

从表 4 中可以看出，我们的字单位的模型表现明显优于词单位的其他模型，尽管我们的模型更小。字模型同样优于词素模型（MLBL 和 LSTM 架构），注意到词素模型需要更多的参数，因为词嵌入作为输入的一部分。

由于内存限制，我们在 DATA-L 上只训练了小模型。有趣的是，在西班牙语、法语和英语中，我们没有观察到词模型和语素模型显著的区别。字模型同样比词和语素取得更好多结果。在英语训练中，对于 $\mathscr{V}$ 很大大时候，我们同样也观察到复杂度显著的减少。作为这一小节的总结，我们要指出，对于不同语言，我们使用同样的架构，并且对于高阶参数，我们并没有针对特定的语言有特定的参数调整

5 讨论

5.1 学习到的词表示

我们研究了模型在 PTB 数据集上学习到的词的表示。表 6 显示了词单位和字单位模型学习到的近邻词表示。对于字单位模型，我们比较了使用高速网络前后的词表示。

在进入高速网络层之前，词表示看起来只依赖于表面形式，例如单词 you 的最近邻是 your, young, four, youth。高速网络似乎能够编码和拼写无关的语义信息。在经过高速网络之后，you 的最近邻是 we，we 的拼写和 you 差很多。另一个例子是，while 和 though，他们看起来拼写很不一样，但是我们的模型能够把他们放置到临近的地方。我们的模型也犯了很多明显的错误（比如 his 和 hhs），尽管可能是由于数据集太小造成的，还是揭示了我们模型的

		DATA-S					
		CS	DE	ES	FR	RU	AR
Botha	KN-4	545	366	241	274	396	323
	MLBL	465	296	200	225	304	–
Small	Word	503	305	212	229	352	216
	Morph	414	278	197	216	290	230
	Char	401	260	182	189	278	196
Large	Word	493	286	200	222	357	172
	Morph	398	263	177	196	271	**148**
	Char	**371**	**239**	**165**	**184**	**261**	**148**

Figure 5: 对于 DATA-S 在测试集上的复杂度

		DATA-L					
		CS	DE	ES	FR	RU	EN
Botha	KN-4	862	463	219	243	390	291
	MLBL	643	404	203	227	**300**	273
Small	Word	701	347	186	202	353	236
	Morph	615	331	189	209	331	233
	Char	**578**	**305**	**169**	**190**	313	**216**

Figure 6: 对于 DATA-L 在测试集上的复杂度

一些局限。

5.2 学习到的字 n 元语法表示

正如前面讨论到的，CharrCNN 的每个筛选器对于识别特定字的 n 元语法是必要的。我们最初的期待是每个筛选器能够学习在不同的语素上激活，然后根据识别出的语素实现语义表达。然而，通过研究筛选器选出的 n 元语法，我们发现它们并没有和合法的语素建立起联系。

为了能够建立起更好的关于字模型学了的什么的直觉，通过主成分分析，我们绘出了学习到的 n 元语法的表示。我们把每个字 n 元语法送入 CharCNN，然后利用 CharCNN 的输出作为对应的字的固定维度的 n 元语法表示。正如图表 2 显示的，我们的模型学习到了如何区分前缀（红）、后缀（蓝）和其他（灰）。我们也发现这个表达对于包含连字符的字 n 元语法表示特别敏感，我们的推测是因为连字符通常预示着这个词是演讲的一部分。

5.3 高速网络层

我们通过隔离分析定量的研究高速网络的作用。我们移除模型的一层高速网络层，发现模型的性能严重下降。因为性能的不同可能是由于模型大小减少所造成的，我们也训练了一个将 $\mathbf{y}^{\mathbf{k}}$ 送入一个一层 MLP，然后将它作为 LSTM 的输入。尽管可能是因为优化的问题，但是我们发现 MLP 效果也很差。

我们推测高速网络对于 CNN 来说是必要的。CNN 已经在很多 NLP 任务中被证明是有效的方法。我们推测将高速网络结合进 CNN 中将会取得更好的结果。

我们也发现：（1）有一到两层高速网络是很重要的，但是进一步增加高速网络层数，并没有取得更优的结果。(2) 在最大池化前设置更多的卷积网络并没有帮助。（3）高速网络对于使用词嵌入作为输入的模型并没有帮助。

	In Vocabulary					Out-of-Vocabulary		
	while	*his*	*you*	*richard*	*trading*	*computer-aided*	*misinformed*	*looooook*
LSTM-Word	*although*	*your*	*conservatives*	*jonathan*	*advertised*	–	–	–
	letting	*her*	*we*	*robert*	*advertising*	–	–	–
	though	*my*	*guys*	*neil*	*turnover*	–	–	–
	minute	*their*	*i*	*nancy*	*turnover*	–	–	–
LSTM-Char (before highway)	*chile*	*this*	*your*	*hard*	*heading*	*computer-guided*	*informed*	*look*
	whole	*hhs*	*young*	*rich*	*training*	*computerized*	*performed*	*cook*
	meanwhile	*is*	*four*	*richer*	*reading*	*disk-drive*	*transformed*	*looks*
	white	*has*	*youth*	*richter*	*leading*	*computer*	*inform*	*shook*
LSTM-Char (after highway)	*meanwhile*	*hhs*	*we*	*eduard*	*trade*	*computer-guided*	*informed*	*look*
	whole	*this*	*your*	*gerard*	*training*	*computer-driven*	*performed*	*looks*
	though	*their*	*doug*	*edward*	*traded*	*computerized*	*outperformed*	*looked*
	nevertheless	*your*	*i*	*carl*	*trader*	*computer*	*transformed*	*looking*

Figure 7: 最近邻词在字单位和词单位经过高速网络后的表达

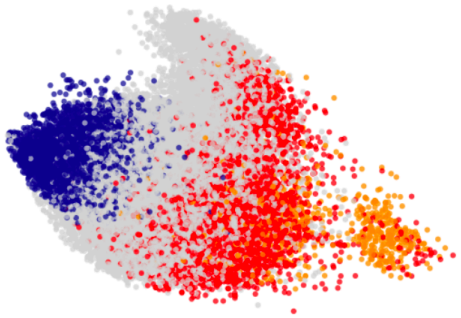

Figure 8: 字的 n 元语法的点图

5.4 词汇量大小的影响

我们接着学习了词汇集的大小对于不同模型结果的影响。我们使用 DATA-L 中的德语（DE）数据集，实用不同的词汇集大小，计算从词单位和字单位复杂率减少的大小。为了控制词汇量的变化，我们选出最常用的 k 个词，把他们替换为 <unk>。因为在前面的实验中，字模型没有使用 <unk> 的表面形式，只是把它当成另外一个标签。尽管表 8 表明随着词汇量的增加，复杂度减少的没有那么明显，我们还是发现字模型在所有的场景下都还是优于词模型。

5.5 进一步观察

我们在进一步的的实验和观察中发现了下面的情况：

- 结合词嵌入和 CharCNN 的输出的词表示结果稍微糟糕一些。这个结果很让我们意外，因为有相关报道表明这样的结合在演讲标记和实体识别中有明显的改善。尽管我们的实验结果可能是由于我们的实验设置的不够合理，但是还是表明，对于一些任务，词嵌入式无用的，字输入就已经足够了

- 尽管我们的模型需要卷积操作，所以相对于只是进行简单词查找的词模型慢一些，但是我们还是发现这种速度的区别可以通过 GPU 实现的优化来控制的。比如，在 PTB 数据集上，大的字单位模型训练速度为 1500 标签每秒，相对应的，词模型训练速度为 3000 标签每秒。为了评分，因为 CharCNN 的输出可以被预先训练出来，所以我们的模型可以和词模型有相同的运行时间。然而，这样会增加模型的大小，所以这是一个在运行时速度和内存之间的取舍问题。

	LSTM-Char	
	Small	Large
No Highway Layers	100.3	84.6
One Highway Layer	92.3	79.7
Two Highway Layers	90.1	78.9
One MLP Layer	111.2	92.6

Figure 9: 小模型和大模型在使用/不使用高速网络的复杂度对比

		$\lvert\mathcal{V}\rvert$			
		10 k	25 k	50 k	100 k
T	1 m	17%	16%	21%	–
	5 m	8%	14%	16%	21%
	10 m	9%	9%	12%	15%
	25 m	9%	8%	9%	10%

Figure 10: 不同的词汇量下，字模型比词模型在复杂度减少量

6 总结

我们引入了只使用字单位输入的神经语言模型。预测还是在词单位进行的，尽管模型有更少的参数，我们的模型表现超出使用词单位和使用语速单位的模型的基准线。我们的模型对于词嵌入作为输入在神经语言建模中的必要性。
对于从字模型中获取到的词的表示进行分析之后揭示了这个模型能够只从字编码丰富的语义信息和拼写信息。使用 CharCNN 和高速网络层对于表达学习还是有很大帮助的。
到目前为止，由于序列化处理词作为自然语言处理比较普世的方法，如果这篇论文的方法能够对其他任务有所帮助是很好的。

7 致谢

我们非常感谢 Jan Botha，他为我们提供了预处理好的数据和模型的结果。