arXiv：1808.09772v2 [cs.CL] 2018年8月30日

关于NLP深度学习的注意事项

Antoine J.-P. Tixier

计算机科学系（DaSciM团队）法国Palaiseau的ÉcolePolytechnique [安东尼. tix](mailto:antoine.tixier-1@colorado.edu)[1 colorado.edu @。](mailto:ier-1@colorado.edu)

最后更新时间星期五31*圣*2018年8月（首次上载2017年3月23日）

**内容**

[7 注意 14 2](#_Toc523947603)

[1 放弃 2](#_Toc523947604)

[2 码 2](#_Toc523947605)

[3 IMDB电影评论数据集 2](#_Toc523947606)

[3.1 概观 2](#_Toc523947607)

[3.2 二进制分类目标函数 2](#_Toc523947608)

[4 范式转换 3](#_Toc523947609)

[4.1 功能嵌入 3](#_Toc523947610)

[4.2 功能嵌入的好处 3](#_Toc523947611)

[4.3 结合核心功能 3](#_Toc523947612)

[5 卷积神经网络（CNN） 4](#_Toc523947613)

[5.1 局部不变性和组合性 4](#_Toc523947614)

[5.2 卷积和汇集 4](#_Toc523947615)

[5.2.1 输入 4](#_Toc523947616)

[5.2.2 卷积层 5](#_Toc523947617)

[5.2.3 汇集层 5](#_Toc523947618)

[5.2.4 文件编码 6](#_Toc523947619)

[5.2.5 Softmax层 6](#_Toc523947620)

[5.3 参数数量 7](#_Toc523947621)

[5.4 可视化和理解内部表示和预测 7](#_Toc523947622)

[5.4.1 文档嵌入 7](#_Toc523947623)

[5.4.2 预测区域识别 8](#_Toc523947624)

[5.4.3 显着性图 8](#_Toc523947625)

[6 递归神经网络（RNN） 9](#_Toc523947626)

[6.1 RNN框架 9](#_Toc523947627)

[6.1.1 语言建模 10](#_Toc523947628)

[6.2 LSTM单位 11](#_Toc523947629)

[6.2.1 内层 11](#_Toc523947630)

[6.2.2 忘记/学习 12](#_Toc523947631)

[6.2.3 香草RNN比喻 12](#_Toc523947632)

[6.3 门控循环单元（GRU） 12](#_Toc523947633)

[6.4 lstm RNN VS VS格鲁 13](#_Toc523947634)

[7 注意 14](#_Toc523947635)

### 注意 [14](#_bookmark67)

* 1. 编码器 - 解码器注意 [14](#_bookmark68)
     1. 编码器 - 解码器概述 [14](#_bookmark69)
     2. 编码器 [15](#_bookmark71)
     3. 解码器 [15](#_bookmark73)
     4. 全球关注 [16](#_bookmark76)
     5. 当地的关注 [17](#_bookmark80)
  2. 自我关注 [18](#_bookmark83)
     1. 与seq2seq关注的区别 [18](#_bookmark85)
     2. 分层关注 [19](#_bookmark87)

# **放弃**

写这些笔记是我学习过程的一部分，所以这是一项正在进行的工作。 为了编写本文档的当前版本，我主要从原始的2D CNN论文中提取信息

[[16](#_bookmark104)]和斯坦福大学的CS231n CNN课程笔记[1](#_bookmark6)，张和华莱士从业者对NLP的CNN指南[[26](#_bookmark114)]，关于CNN的文章分类的开创性论文[[13](#_bookmark101), [14](#_bookmark102)]，丹尼布里茨的教程[2](#_bookmark7) 关于RNNs，Chris Colah的帖子[3](#_bookmark8) 了解LSTM单位以及GRU单位的开创性论文[[2](#_bookmark90), [4](#_bookmark92)]，编码器 - 解码器架构[[2](#_bookmark90), [24](#_bookmark112)]和关注[[20](#_bookmark108), [1](#_bookmark89)]。 最后但并非最不重要的是，Yoav Golderg的NLP神经网络入门[[8](#_bookmark96)和Luong，Cho和Manning关于神经机器翻译的教程[4](#_bookmark9) 证明非常有用。

# **码**

我在Keras中实现了本文档中描述的一些模型，并在IMDB电影评论数据集上进行了测试。 代码可以在我的GitHub上找到： [github.com https：／／／ Tixierae/deep\_learning\_NLP](https://github.com/Tixierae/deep_learning_NLP)。 同样，这是一项正在进行的工作。

# **IMDB电影评论数据集**

## 概观

任务是对来自Internet电影数据库（IMDB）数据集的评论执行二进制分类（正/负）[5](#_bookmark10)，这被称为情绪分析或意见挖掘。 数据集包含50K电影评论，标记为极性。 数据分为50％用于培训，50％用于测试。 我的GitHub上的imdb\_preprocess.py脚本清理评论并将它们放入适合传递给神经网络的格式中：每个评论都是来自大小为V的字典的单词索引（整数）列表，其中最常见的单词具有索引1 。

## 二进制分类目标函数

我们的模型将学会最小化的目标函数是对数损失，也称为交叉熵。 更确切地说，在具有2个类（例如0和1）的二进制分类设置中，日志丢失定义为：

1  ( ( ) ( ))

*N*

logloss = - *y* logp + 1 - y log 1 - p(1)

*N*

*i*=1

*i*

*i*

*i*

*i*

1[http：/ / / / cs231n.github.io卷积网络](http://cs231n.github.io/convolutional-networks/)

2[http：/ / / / / www.wildml.com 09年经常性的神经网络教程第1节到RNNs /](http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/) 3[http：/ / / / colah.github.io帖子2015年覆盖lstms／08](http://colah.github.io/posts/2015-08-Understanding-LSTMs/) 4[http：/ / / / / sites.google.com acl16nmt家居网站](https://sites.google.com/site/acl16nmt/home)

5[ai.stanford.edu http：/ / / / /数据/情绪amaas ~](http://ai.stanford.edu/%7Eamaas/data/sentiment/)

其中N是观测数，p*i*是分配给1类的概率 *Pi* 是指定为0​​级的概率，y*i*是i*日*观测值（0或1）的真实标签。 您可以看到，只有与每个观察的真实标签相关联的术语才会影响总体得分。 对于给定的观察，假设真实标签是1，并且模型分配给该类的概率是0.8（相当好的预测），则对数损失将等于log（0.8）= 0.22。 如果预测略差但不完全关闭，比如p*i*= 0.6，则对数损失将等于0.51，对于0.1，对数损失将达到2.3。 因此，模型距离事实越远，受到的惩罚就越大。 显然，一个完美的预测

*−*

( )*−*

（右级的概率为1）获得空分。

# **范式转换**

## 功能嵌入

与将核心特征及其组合视为特征空间的独特维度的传统机器学习模型相比，深度学习模型通常将核心特征（仅核心特征）作为向量嵌入低维连续空间中，其中维度表示共享潜在概念[[8](#_bookmark96)]。 嵌入随机初始化或从预训练中获得[6](#_bookmark16)。 然后可以像其他模型参数一样在训练期间更新它们，或者保持静态。

## 功能嵌入的好处

将特征映射到密集连续向量的主要优点是能够捕获特征之间的相似性，因此可以概括。 例如，如果模型在训练期间从未见过“奥巴马”这个词，但是遇到了“总统”这个词，通过知道这两个词是相关的，它就能将它所学到的“总统”转移到涉及“奥巴马”的案件。 对于传统的单热矢量，这两个特征将被认为是正交的，并且它们之间不能共享预测能力[7](#_bookmark17)。 此外，从庞大的稀疏空间到密集紧凑的空间，可以降低计算成本和拟合模型所需的数据量，因为学习的参数较少。

## 结合核心功能

与传统ML中所做的不同，核心特征的组合不被编码为特征空间的新维度，而是被编码为要组合的核心特征的向量的和，平均或级联。 无论训练示例的大小如何（例如，文档中的单词数），求和或求平均是总是获得固定大小的输入向量的简单方法。 但是，这两种方法都完全忽略了功能的排序。 例如，在这种情况下，并使用unigrams作为特征，两个句子“John比Mary快”和“Mary比John快”具有完全相同的表示。 另一方面，使用串联允许跟踪排序，但填充和截断[8](#_bookmark18) 需要使用以便为每个训练示例连接相同数量的向量。 例如，无论其大小如何，集合中的每个文档都可以转换为具有相同的固定长度s：较长的文档被截断为其第一个（或最后一个，中间...）的单词，较短的文档被填充用一个特殊的零向量来弥补缺失的单词[[26](#_bookmark114), [14](#_bookmark102)].

6在NLP中，经常使用从非常大的语料库中用Word2vec或GloVe获得的预训练的单词向量。 例如，可以从中获取Google News word2vec向量 [http：／／code.google.com word2vec /档案/ P / /](https://code.google.com/archive/p/word2vec/)，在“预先训练的单词和短语向量”部分下

7注意，单热矢量可以作为输入传递给神经网络。 但是，网络隐含地学习了

在第一层中嵌入特征

8[http：/ / / / / keras.io序列预处理](https://keras.io/preprocessing/sequence/)

# **卷积神经网络（CNN）**

## 局部不变性和组合性

最初灵感来自对猫的视觉皮层的研究[[12](#_bookmark100)]，CNNs是在计算机视觉中开发的，用于处理常规网格，如图像[[16](#_bookmark104)]。 它们是前馈神经网络，其中层中的每个神经元接收来自前一层中的神经元邻域的输入。 通过将较低级别的基本特征组合到更高级别的特征中，这些邻域或本地感知字段允许CNN以分层方式识别越来越复杂的模式。 此属性称为组合性。 例如，可以从原始像素推断边缘，然后可以使用边缘来检测简单的形状，最后可以使用形状来识别对象。 此外，图像中特征的绝对位置无关紧要。 仅捕获各自的位置对于组成更高级别的模式非常有用。 因此，模型应该能够检测到特征，而不管其在图像中的位置如何。 此属性称为局部不变性。 组合性和局部不变性是CNN的两个关键概念。 CNN在计算机视觉方面已经达到了很好的表现[[15](#_bookmark103)]，但不难理解，由于组合性和局部不变性，它们在NLP中也能做得很好。 实际上，在NLP中，高阶特征（n-gram）可以像CV一样从低阶特征构造，并且在本地排序是至关重要的（“不坏，非常好”，“不好，非常差”，“不建议“），但不建议在文档级别。 实际上，在试图确定电影评论的极性时，我们并不关心在文档的开头或结尾是否“不坏，非常好”。 我们只需要捕捉“不”在“坏”之前的事实，依此类推。 请注意，CNN无法编码远程依赖关系，因此，对于某些任务，例如语言建模，长距离依赖很重要，周期性架构，如

LSTM是首选。

## 卷积和汇集

虽然最近的工作表明卷积层可以直接堆叠在彼此之上[[23](#_bookmark111)]，CNN的基本结构是卷积层，后面是汇集层。 在下文中，我们将详细说明这两个层如何相互作用，例如使用短文档分类的NLP任务（参见图2）。 [1](#_bookmark30)).

### 输入

我们可以将文档表示为实矩阵AR*s x d*，其中s是文档长度，d是单词嵌入向量的维度。 由于s必须在集合级别固定，但文档大小不同，我们会将较长的文档截断为它们的第一个单词，并根据需要用特殊的零向量填充较短的文档。 单词向量可以随机初始化或者预先训练。 在后一种情况下，它们可以在培训期间更新或不更新（“非静态”与“静态”方法[[14](#_bookmark102)]).

*∈*

将A视为图像具有误导性，因为只有一个空间维度。 嵌入向量实际上不是输入本身的一部分，它们只是表示共享潜在空间中输入元素的坐标。 在计算机视觉中，术语“通道”通常用于指代此深度维度（不要与网络中隐藏层的数量相混淆）。 如果我们处理图像，我们将有两个空间维度加上深度。 输入将是维度的张量（宽度高度n\_channels），即，2D矩阵，其中在颜色（RGB）和灰度级图像的情况下，每个条目将分别与长度为3或1的向量相关联。

*× ×*

### 卷积层

卷积层是线性操作，然后是非线性变换。 线性操作包括将（在元素上）乘以在输入文档上应用的1D窗口的每个实例的过滤器，表示为参数矩阵。 过滤器就像窗口一样，只有一个空间维度，但它完全延伸到输入深度（单词嵌入空间的d维度）。 如果h是窗口大小，则与过滤器相关联的参数矩阵W因此属于R*h x d*。 W在训练期间随机初始化并学习。

窗口在输入上的实例化称为区域或感知字段。 有（S-H）/ stride + 1，其中stride对应于我们在每一步滑动窗口的单词数。 步幅为1，因此有s - h + 1个感受野。 该

*∈*

因此，给定滤波器的卷积层的输出是其元素为的向量o R*S-H + 1*

计算如下：

*oi*= W·A [i：i + h - 1，：] (2)

其中A [i：i + h 1，：] R*h x d*是i*日*区域矩阵，并且是返回两个矩阵的行方点积的和的运算符。 请注意，对于给定的过滤器，相同的W应用于窗口的所有实例化，无论它们在文档中的位置如何。 换句话说，过滤器的参数在感知域之间共享。 这正是为模型提供空间不变性的原因，因为过滤器经过训练以识别模式所在的任何位置。 它还大大减少了模型的参数总数。

*− ∈ ·*

然后，非线性激活函数f，例如ReLU[9](#_bookmark27) （max（0，x））或tanh（*e2x−1*）被应用

*e*2*x*+1

elementwise to o，返回与过滤器关联的所谓的特征映射c R*S-H + 1*：

*∈*

其中b∈R是可训练的偏差。

*ci= f（oi）+ b* (3)

对于短句分类，最佳区域大小通常在1到10之间，实际上，n*f*过滤器（n*f*[100,600]）应用于每个区域，使模型具有学习能力每个区域的不同，互补特征[[26](#_bookmark114)]。 由于每个滤波器生成特征映射，因此每个区域嵌入到n*f*维空间中。 此外，在最佳区域周围使用不同大小的区域可以提高性能[[26](#_bookmark114)]。 在这种情况下，创建不同的并行分支（每个区域大小一个），并在汇集后连接输出，如图4所示。 [1](#_bookmark30)。 性能和成本随着n*f*增加到某一点，之后模型开始过度拟合。

*∈*

### 汇集层

输入文档中功能的确切位置无关紧要。 重要的只是某些特征是否存在。 例如，要将评论归类为肯定，在文档的开头或结尾出现“有史以来最好的电影”并不重要。 要将这种稳健性注入模型，全局k-max池 [10](#_bookmark28) 受雇了。 该方法从每个特征图中提取k个最大值并将它们连接起来，从而形成最终向量，其大小在训练期间始终保持不变。 简短的句子

9与tanh相比，ReLu是可承受的（在负区域中由许多零值引起的稀疏性）并且更好地消除消失的梯度问题，因为在正区域中，梯度是恒定的，而随着tanh它变得越来越小

10池化也可以在小区域本地应用，但对于短文本分类，全局池化工作

更好[[26](#_bookmark114)].

分类，[[26](#_bookmark114)]发现k = 1远远优于高阶策略。 他们还报告说使用最大值比使用平均值要好得多，这是有道理的，因为我们只想从每个特征图中提取最显着的特征。

激活函数卷积

3个区域大小：（2,3,4）每个区域大小的2个过滤器

共6个过滤器

句子矩阵7×5

1-max池

每个区域大小的2个要素图

2节课

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

d=5



具有softmax和dropout的仿射层

连接6个条目以形成单个特征向量

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

I

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

非常喜欢这部电影

|  |
| --- |
|  |
|  |
|  |
|  |
|  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |

!

|  |
| --- |
|  |
|  |
|  |
|  |
|  |

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

图1：用于（短）文档分类的CNN架构，取自Zhang和Wallace（2015）[[26](#_bookmark114)]. *s* = 7,



{ 1

*d* = 5. 3 re{gions o1f各自的尺寸h = 2,3,4被认为具有相应的输出矢量resp。 长度

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |

*s* - h + 1 =

6*,* 5*,* 4

对于每个滤波器（在卷积之后产生，未示出）。 每个区域大小有2个过滤器。 对于

三个区域大小，过滤器分别为。 与元素应用f和添加偏差后的长度特征图相关联）。 使用1-max池。

6*,* 5*,* 4

（输出向量

### 文件编码

{ 1

如图所示。 [1](#_bookmark30)CNN架构从高层看事物，将输入的每个滤波版本连接到最终特征向量中的单个神经元。 该向量可以看作输入文档的嵌入或编码。 这是模型的主要贡献，我们感兴趣的东西。架构的其余部分仅取决于任务。

### Softmax层

由于此处的目标是对文档进行分类，因此将softmax函数应用于文档编码以输出类概率。 然而，不同的任务会要求不同的架构：确定两个句子是否是释义，例如，需要两个CNN编码器共享权重，具有最终能量函数和对比度损失（la暹罗语[][3](#_bookmark91)]）; 对于翻译或摘要，我们可以使用LSTM语言模型解码器，以输入文档的CNN编码为条件（àlaseq-to-seq [[24](#_bookmark112)]）等

回到我们的分类设置，softmax转换向量xR*K*成一个

*∈*

累加到一的正浮点数的向量，即，到类的概率分布

预测：

softmax（x*i*）=

*e×i*

*K*

*j*=1

*e×j*

(4)

在二元分类的情况下，我们可以得到一个只有一个神经元和一个sigmoid函数的输出层（σ（x）=，而不是具有softmax的两个神经元的最终输出层，其中每个神经元代表两个类中的一个。 1-x）。 在这种情况下，神经元输出属于两个类别之一的概率，并且基于σ（x）是大于还是小于0.5（假设相等的先验）来做出关于要预测的类别的决定。 这些

1+*e*

=

两种方法是等价的。 事实上，1

*−x*1+*e*

*eX*

*Ex+ E*0

。 因此，单神经元S形层可以

被视为双神经元softmax层，其中一个神经元永远不会激活并且其输出始终等于零。

## 参数数量

CNN的可训练参数总数是以下术语的总和：

**字**嵌入矩阵（仅当非静态模式时）:( V + 1） *d*，其中V是词汇量的大小。 我们为零填充向量添加一行。

* *×*

**卷积层：h** *d nf*+ n*f*（每个滤波器中的条目数乘以滤波器数量加上偏差）。

* *× ×*

**softmax层：n***f* 1 + 1（完全连接的层，输出尺寸为1，偏置为1）。

* *×*

## 可视化和理解内部表示和预测

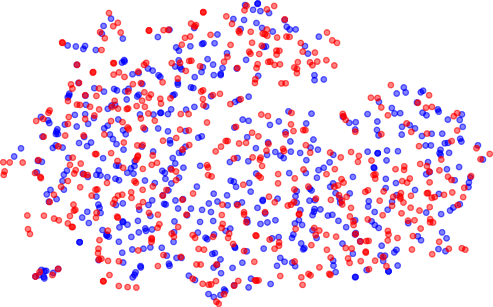
### 文档嵌入

验证我们的模型是否有效学习的快速简便方法是检查其内部文档表示是否有意义。 回想一下，馈送到softmax层的特征向量可以看作输入文档的n*f*维编码。 通过在文档子集的体系结构中以该精确级别收集模型的中间输出，并将向量投影到低维地图，我们可以因此可视化嵌入和标签之间是否存在任何相关性。 Fig.s [2](#_bookmark36) 和 [3](#_bookmark36) 证明确实，我们的模型正在学习有意义的文档表示。

基于CNN的doc嵌入的t-SNE可视化（来自测试集的前1000个文档）

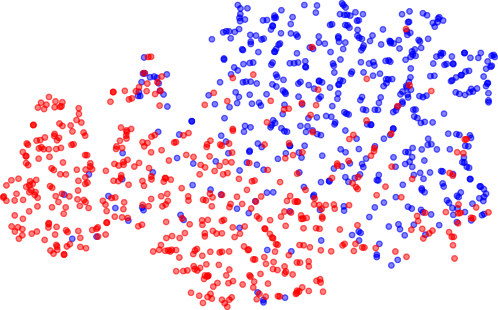
基于CNN的doc嵌入的t-SNE可视化（来自测试集的前1000个文档）

30



0

1



0

1

30

20

20

10

10

0

0

10

10

20

20

30

30

30 20 10 0 10 20 30 30 20 10 0 10 20 30

图2：训练前的Doc嵌入。 图3：2个时期后的Doc嵌入。

### 预测区域识别

这种方法见[第5节和第6节] [[13](#_bookmark101)]。 回想一下，在我们通过应用池来丢失位置信息之前，大小为h的n个*f*过滤器中的每一个都与大小为（S-H）/ stride + 1（特征映射）的向量相关联，其条目表示卷积的输出。在应用非线性和添加偏置之后，在输入中具有相应感受野的滤波器。 因此，每个感受野嵌入n*f*维空间。 因此，在训练之后，我们可以通过检查与感知域嵌入相对应的模型的中间输出（在汇集层之前），并通过查找区域来识别给定文档中对其类别最具预测性的区域。有最高的规范。 例如，负面IMDB评论的一些最具预测性的区域是：“有史以来最糟糕的电影”，“不要浪费你的钱”，“写得不好，表现不佳”，“糟糕的画面质量”。 相反，一些非常具有积极性的地区是：“惊人的配乐”，“视觉上美丽”，“酷酷的旅程”，“结束相当令人满意”......

### 显着性图

另一种了解模型如何发布预测的方法由[[[22](#_bookmark110)]并通过[适用于NLP][17](#_bookmark105)]。 该想法是基于输入文档对预测的影响对输入文档AR*s x d*的元素进行排名。 可以通过模型CNN的输出的一阶偏导数的大小给出近似：CNN（A）相对于A的每一行a：

*∈*

*1→*

*∂*（CNN）

1

1

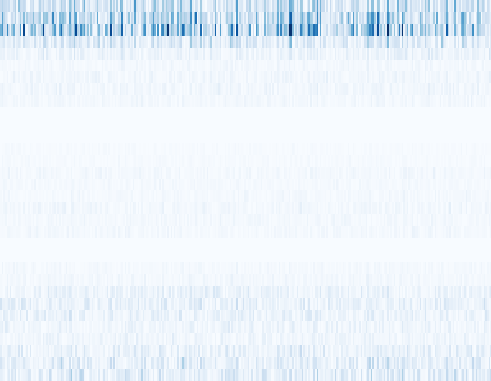
显着性（a）= *|*

(5)

1 *∂a a*1

解释是我们确定A中哪些词需要改变最少，以最大程度地改变班级分数。 可以通过执行单个反向传播通过（基于预测，而不是像训练期间的损失）来获得导数。 Fig.s [4](#_bookmark40) 和 [5](#_bookmark41) 分别显示负面和正面评论的显着性图示例。

最美观



曾经见过我的生活多在这里学习如何发挥

相机彩色服装

设置拍摄工作进入官方网站英语法语也给想法纯粹的美

0.020

0.015



0.010

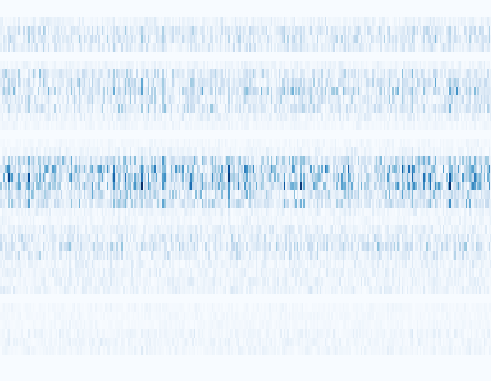
0.005

载 0.000

0 50 100 150 200 250

图4：IMDB测试集的文档1的显着性图（真实标签：正面）

国王活着有缺陷的人为混乱，它脱落了



自我放纵的借口运输束演员沙漠在哪里可以

扮演重大思想

愚蠢的过度紧张

戏剧完全无视真正的力量人性市场的吸引力甚至常识

艺术其他的单词电影不相信你只看

0.00020

0.00015



0.00010

0.00005

批评者 0.00000

0 50 100 150 200 250

图5：IMDB测试集文档15的显着性图（真实标签：负面）

# **递归神经网络（RNN）**

我们首先介绍整个RNN框架，然后介绍两种在实践中广泛使用的单元：LSTM和GRU。 有关RNN，LSTM及其应用的详细介绍可以在[[19](#_bookmark107)].

## RNN框架

虽然CNN天生擅长处理网格，但RNN专门开发用于序列[[6](#_bookmark94)]。 一些示例包括时间序列，或者在NLP中，单词（字符序列）或句子（单词序列）。 CNN确实允许捕获一些订单信息，但它仅限于本地模式，并且忽略了远程依赖性[[8](#_bookmark96)]。 如图所示。 [6](#_bookmark45)，RNN可以被视为共享相同参数的简单神经层链。

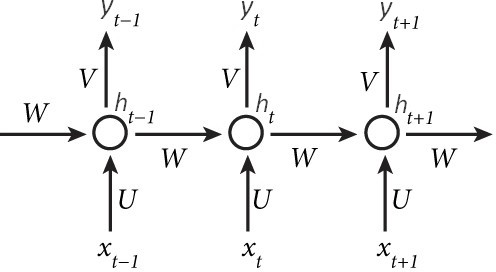


图6：展开的RNN的3个步骤（改编自 [*Denny Britz的博客*](http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-2-implementing-a-language-model-rnn-with-python-numpy-and-theano/)。 每个圆圈代表一个RNN单位（见方程式） [6](#_bookmark47) & [7](#_bookmark48)).

隐藏状态h0初始化为全零，并返回隐藏状态的有序列表

*h*1*，...，hT*

作为

以及输出向量y1，...，y*T*的有序列表。 输出向量可以用作输入

其他RNN单元，在考虑深层架构时（垂直堆叠的多个RNN层，如图1所示）。 [7](#_bookmark46)）。 隐藏状态或多或少地对应于网络的“短期”存储器。 注意，每个训练示例是其自身的完整x1，...，x*T*序列，并且可以根据任务与标签相关联。 例如，对于简短的文件分类，

l

l

从高层次开始，RNN被输入一个有序的输入向量列表x1，...，x*T*l，如同litial

序列将与标签相关联，而对于语言建模，我们只需解析所有序列，重复预测下一个单词。

𝑦𝑡−1 𝑦𝑡 𝑦𝑡+1

ℎ3

𝑡−1

ℎ3

𝑡

ℎ3

𝑡+1

ℎ2

𝑡−1

ℎ2

𝑡

ℎ2

𝑡+1

ℎ1

𝑡−1

ℎ1

𝑡

ℎ1

𝑡+1

. . .

. . .

. . .

. . .

. . .

. . .

𝑥𝑡−1 𝑥𝑡 𝑥𝑡+1

图7：展开的深RNN的3个步骤。 每个圆圈代表一个RNN单位。 内层（1和2）中每个单元的隐藏状态用作上述层中相应单元的输入。

在序列中的任何步骤t，隐藏状态h*t*根据先前隐藏状态定义

*ht−1*和当前输入向量x*t*以下列递归方式：

*ht= f（U xt+ W ht−1+ b）* (6)

其中f是非线性，例如tanh（应用元素），x*t*∈R*吵闹*，U∈R*(H) x 从*和WR*^ h×高*是所有时间步长共享的参数矩阵，h*t*，h *t−1*和b属于R*H*。 如果将单热矢量作为输入传递，则d*in*可以是词汇表的大小，或维度

*∈*

使用共享功能时的嵌入空间。 H是隐藏层的维度。 通常，H 100.该层越大，存储器的容量越大，计算成本越高。

*∼*

输出向量y*t* R*怀疑*转换当前隐藏状态h*t* R*H*的方式

*∈ ∈*

取决于最后的任务。 对于分类，计算如下：

*yt*= softmax（V h*t*） (7)

其中VR*DOUT×高*是在所有时间步骤中共享的参数矩阵。 d*出*取决于类别的数量。 例如，对于3级文档分类，d*出*= 3，对于单词级语言模型，d*出*= V.

*∈*

*| |*

注意，当垂直堆叠多个RNN层（深RNN架构）时，下面单元的隐藏状态直接连接到上面的单位，即x*标签*= y*tbelow*和y*tbelow*= h*tbelow* 。 输出层（Eq。 [7](#_bookmark48)）位于堆栈顶部。

### 语言建模

语言建模是分类的一种特殊情况，其中训练模型以预测句子中的下一个单词或字符。 在每个时间步t，输出向量给出词汇中所有单词/字符的x*t*概率分布，以前一个为条件

序列中的单词/字符，即P [x*t*| x*t−1*，...，x1]。 在测试时，全序列{x1，...，x*T*}的概率由所有条件概率的乘积给出：

*T*

l TT

*P {x*1*，...，xT} = P [x*1*]× P [xt| xt−1，...，x*1*]* (8)

*t*=2

语言模型还可用于生成任意大小的文本，方法是重复采样所需的时间步长数（对于字符级粒度）或直到

选择了特殊的句末标记[11](#_bookmark54) （用于字级粒度）。

例如，对于字符级语言模型，T很容易超过20或25.这极大地放大了众所周知的消失和爆炸梯度问题的不利影响，这可以防止学习远程依赖性[12](#_bookmark55)。 请注意，这个问题也可以通过前馈神经网络（例如多层感知器）来体验，但由于它们固有的深层倾向，它会因RNN而变得更糟。

## LSTM单位

实际上，每当人们使用RNN时，他们就会使用LSTM或GRU单元（参见下一小节），因为这些单元的设计使其能够逃脱消失/爆炸的梯度并在更长的时间段内跟踪信息[[11](#_bookmark99)].

如图所示。 [8](#_bookmark52)，与基本RNN单元相比，LSTM单元中改变的两件事是（1）存在作为显式存储器的单元状态（c*t*），以及（2）如何计算隐藏状态。 使用vanilla RNN，隐藏状态用单个计算

层为h*t*= tanh（U x*t*+ W h*t−1*+ b）（参见方程式2）。 [6](#_bookmark47)）。 然而，对于LSTM单元，隐藏状态由四个交互层计算，这使得网络具有记忆或忘记的能力

有关序列中前面元素的具体信息。

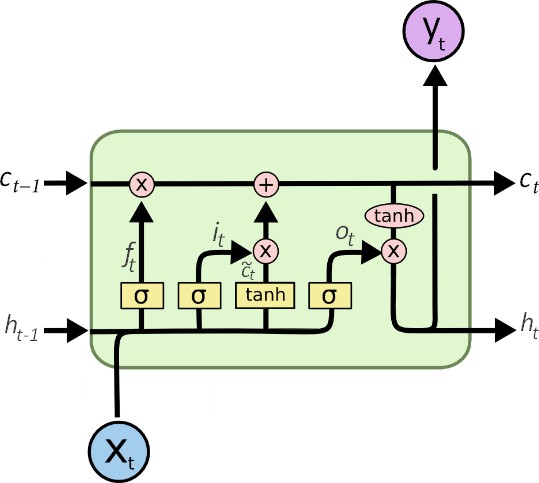


图8：LSTM单元。 改编自 [*Chris Colah的博客*](http://colah.github.io/posts/2015-08-Understanding-LSTMs/).

### 内层

这四层是：

* + - 1. 遗忘门层：f*t*=σU*f*x*t*+ W*f*h*t−1*+ b*f*

( )

( )

* + - 1. 输入栅极层：i*t*=σU*i*x*t*+ W*i*h*t−1*+ b*i*

( )

* + - 1. 候选值计算层：c〜*t*= tanh U*c*x*t*+ W*c*h*t−1*+ b*c*

[( )](http://karpathy.github.io/2015/05/21/rnn-effectiveness/)

* + - 1. 输出栅极层：o*t*=σU*o*x*t*+ W*o*h*t−1*+ b*o*

11见[[9](#_bookmark97)]和 [karpathy.github.io http：/ / / / / / RNN 05年21号-效果](http://karpathy.github.io/2015/05/21/rnn-effectiveness/)

12wildml.com/2015/10/recurrent-neural-networks-tutorial-part-3-backpropagation-through-time-and-

消失-梯度/

由于sigmoid函数（σ）的元素应用，遗忘，输入和输出门层（上面的1,2和4）生成的向量的条目都包含在0和1之间，并且接近0或接近0当这些层中的一个与另一个矢量相乘时，它因此充当仅选择该矢量的特定比例的滤波器。 这正是这些层被称为门的原因。 两种极端情况是当所有条目等于1时 - 完整向量传递 - 或0-无任何传递。 请注意，3个遗忘，输入和输出门以完全相同的方式计算，只有参数变化。 但是，这些参数在所有时间步骤中共享。

### 忘记/学习

通过考虑新的训练示例x*t*和当前隐藏状态h*t−1*，遗忘门层f*t*确定应该忘记前一个小区状态c*t−1*的多少（内存应该被释放），同时从相同的输入，输入门

第i层*t*决定应将多少候选值c~*t*写入存储器，或者换句话说，应该学习多少新信息。 组合两个过滤器的输出会更新单元状态：

*ct= ft◦ct−1+ it◦~t* (9)

其中表示元素乘法（Haddamard乘积）。 这样，重要信息不会被新输入覆盖，但可以长时间保存在它们旁边。 最后，激活h*t*由更新的存储器计算，由输出门层o*t*调制：

*◦*

*ht= tanh ct◦ot* (10)

( )

输出门允许该单元仅在发现存储器内信息与当前时间步长相关时才激活。 最后，和之前的简单RNN一样，输出向量作为新隐藏状态的函数计算：

*yt*= softmax（V h*t*） (11)

### 香草RNN比喻

如果我们决定忘记前一个状态的所有内容（f*t*的所有元素都为空），要学习所有新信息（i的所有元素*t*等于1），并记住整个单元格

状态传递到下一个时间步（o的所有元素*t*等于1），我们有c*t*=

tanh(U*c*x*t*+ W*c*h*t−1*+ b*c)*，因此我们回到(vanill(RNN单位，on)差异

〜*t*=

因为我们最终得到h*t*= tanh tanh U*c*x*t*+ W*c*h*t−1*+ b*c*h*t*= tanh U*c*x *t*+ W*c*h*t−1*+ b*c*与经典RNN情况类似。

( )

## 门控循环单元（GRU）

代替

如图所示。 [9](#_bookmark60)，GRU单位[[2](#_bookmark90)]是一个简化的LSTM单元，只有两个门（复位和更新），并且没有明确的存储器c*t*。

* + 1. 复位栅极层：r*t*=σ(U*r*x*t*+ W*r*h*t−1*+ b*r)*
    2. 更新栅极层：z*t*=σ(U*z*x*t*+ W*z*h*t−1*+ b*z)*

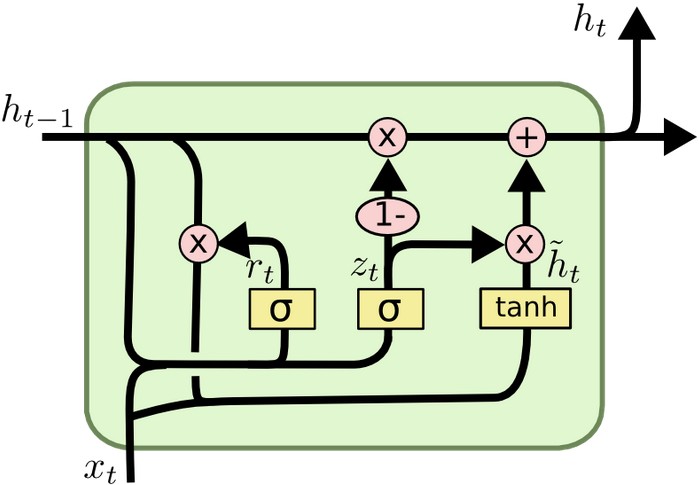


图9：GRU单元。 取自 [*Chris Colah的博客*](http://colah.github.io/posts/2015-08-Understanding-LSTMs/).

候选隐藏状态计算如下：

*h*˜*t = tanh Uhxt+ Wh（rt◦ht−1）+ bh* [13](#_bookmark65) (12)

( )

当复位门的所有元素接近零时，来自先前时间步长的信息（存储在h*t−1*中）被丢弃，因此候选隐藏状态仅基于当前输入x*t*。 最终以类似于LSTM单元的方式获得新的隐藏状态

状态，通过在先前的隐藏状态和候选隐藏状态之间进行线性插值：

*ht= zt◦ht−1+（1 - zt）◦*˜*t* （13）唯一的区别是，这次，更新门z*t*用作遗忘门，并确定应该忘记的前一隐藏状态的信息部分，并且输入门耦合在遗忘门上。

## lstm RNN VS VS格鲁

基本的RNN单元在每个时间步骤暴露其完全隐藏状态（参见Eq。 [6](#_bookmark47)），随着时间的推移，旧的投入的影响很快就会被最近投入的影响所取代。 因此，RNN无法通过多个步骤记住重要功能。 实际上，我们之前已经表明RNN类似于LSTM，其中对于所有t，f*t*=*1*0，i*t*=*1*1和o*t*=*1*1（我们忘记了过去的一切，并学习了关于现在的一切。

另一方面，由于使用了显式存储器（单元）和门控机制，LSTM单元能够控制过去的哪部分信息应保存在存储器中（忘记门f*t*），应将当前输入的哪部分信息写入存储器（输入门i*t*），以及下一个时间步应将多少存储器暴露给更高层的单元（输出门o*t* ））。

GRU还具有门控机制，但没有明确的内存（没有单元状态）。 因此，GRU的选通机制更简单，没有输出门：旧信息和新信息之间的线性插值直接注入到新的隐藏状态中

过滤（见Eq。 [13](#_bookmark63)）。 另一个不同之处在于，当计算候选值时，GRU通过其复位门r*t*调制来自先前激活h*t−1*的信息流（参见方程式1）。 [12](#_bookmark62)），在LSTM单元中，〜*t*基于原始h*t−1*。 最后但并非最不重要的，在

GRU，旧信息和新信息之间的平衡仅由更新门z*t*完成

13应该注意的是[原始配方][2](#_bookmark90)]使用r*t*◦（W*h*h*t−1*）。 在这里，我们采用[[4](#_bookmark92)]，W*h*（r*t*◦h*t−1*）。 根据 [[4](#_bookmark92)]，这两种配方的表现相当。

（见Eq。 [13](#_bookmark63)），而LSTM单元有两个独立的忘记和输入门。

虽然LSTM和GRU单元明显优于基本RNN单元[[4](#_bookmark92)]，没有证据证明哪一个最好[[10](#_bookmark98), [4](#_bookmark92)]。 但是，由于GRU更简单，因此更容易实现，更高效，参数更少，因此需要更少的训练数据。

# **注意**

关注机制[[1](#_bookmark89)]是在用于神经机器翻译（NMT）的编码器 - 解码器架构的背景下开发的[[2](#_bookmark90), [24](#_bookmark112)]，并迅速应用于自然相关的任务，如图像字幕（将图像翻译成句子）[[25](#_bookmark113)]和摘要（翻译成更紧凑的语言）[[21](#_bookmark109)]。 从高级别开始，通过允许解码器在多个向量上购买它所需要的东西，注意力使编码器免于必须将输入嵌入到单个固定长度向量中的负担，因此允许保留更多信息[[1](#_bookmark89)].

今天，注意力在深度学习模型中无处不在，并且不仅在编码器 - 解码器环境中使用。 值得注意的是，仅针对编码器提出了注意装置，以解决诸如文档分类之类的任务[[27](#_bookmark115)]或代表学习[[5](#_bookmark93)]。 这种机制被认为是自我或内在的关注。

在下文中，我们将首先介绍NMT编码器 - 解码器的原始背景，使用[[...]引入的一般框架。[20](#_bookmark108)]，然后介绍自我关注。

* 1. **编码器 - 解码器注意**
     1. **编码器 - 解码器概述**

从很高的层面来看，如图1所示。 [10](#_bookmark70)，编码器将输入嵌入到矢量中，并且解码器从该矢量生成一些输出。

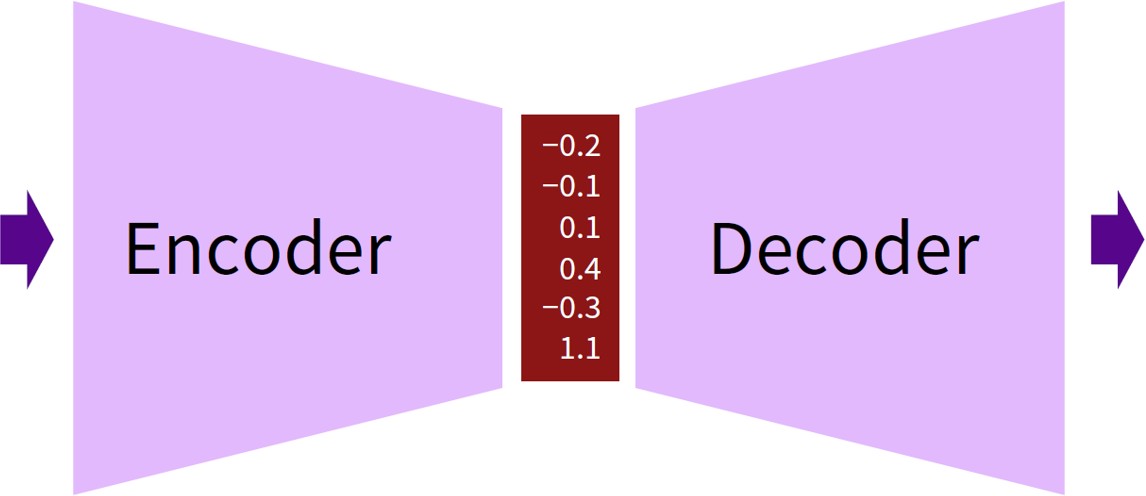


图10：编码器 - 解码器架构概述。 取自 [sites.google.com https：／／／](https://sites.google.com/site/acl16nmt/home) [网站/ acl16nmt /家](https://sites.google.com/site/acl16nmt/home)

在神经机器翻译（NMT）中，输入和输出是单词序列，分别是x = x1,. . . ，x*德克萨斯州*和y = y1,. . . ，y*泰*。 x和y通常被称为源和目标句子。 当输入和输出都是序列时，编码器 - 解码器架构有时被称为序列到序列（seq2seq）[[24](#_bookmark112)]。 由于编码器 - 解码器架构在任何地方都是可微分的，因此可以通过并行语料库上的最大似然估计（MLE）同时优化它们的参数θ。 这种培训方式称为端到端。

( ) ( )

argmax*θ (*

（X，Y）∈corpus

log p（y | x;θ）1 (14)

这里，我们想要最大化的函数是正确翻译的对数概率。

* + 1. **编码器**

源句可以由任何模型嵌入（例如，CNN，完全连接）。 通常对于MT而言，编码器是深RNN。 Bahdanau等。 [[1](#_bookmark89)]最初使用双向深度RNN。 这样的模型由两个深度单向RNN组成，除了字嵌入矩阵之外具有不同的参数。 第一前向RNN从左到右处理源句子，而第二前向RNN从右到左处理它。 在每个时间步t连接两个句子嵌入以获得双向RNN的内部表示：

*ht= h1t; Ht* (15)

l

双向RNN在编码源词时考虑整个上下文，而不仅仅是前面的词。 结果，h*t*偏向以字x为中心的小窗口*t*，而对于单向RNN，h*t*偏向x*t*和紧接在其之前的字。 专注于围绕x*t*的小窗口可能是有利的，但似乎并不重要。 的确，Luong等人。 [[20](#_bookmark108)]使用通常的单向深度RNN编码器获得最先进的结果。 在下文中，将写入编码器的隐藏状态¯t。 它们有时被称为文献中的注释。

* + 1. **解码器**

虽然可以使用不同的模型作为编码器，但是在NMT中，解码器通常是单向RNN，因为该模型自然地适应于生成任务的顺序性质，并且通常是深度（堆叠）。 解码器一次一步地生成目标句子的每个单词。

**关**键的想法。 使解码器仅使用由编码器产生的最后一个注释h*德克萨斯州*来生成输出，迫使编码器尽可能多地将信息拟合到h*德克萨斯州*中。 由于h*德克萨斯州*是单个固定大小的矢量，因此其容量有限，因此丢失了一些信息。 另一方面，注意机制允许解码器考虑整个序列h1,. . . ，h*德克萨斯州*由编码器在生成过程的每个步骤产生的注释。 结果，编码器能够通过在其所有注释中分发信息来保留更多信息，知道解码器将能够稍后决定它应该注意哪些向量。

( )

更确切地说，目标语句y =（y1，...，y*泰*）基于分布一次生成一个单词y*t*：

l ( )

*P yt| {y*1*，...，yt−1}，ct= softmax Ws˜t* (16)

其中˜t，注意隐藏状态，计算如下：

˜*t* = tanh W*c*c*t*; H*t* (17)

( l)

*ht*是解码器的隐藏状态（当解码器是堆叠RNN时顶层的隐藏状态）并且从先前生成的目标字{y1，...中提供信息abl， Y*t−1*}，

*ct*是源上下文向量，和

;

是串联。 W*s*和W*c*是可训练的基质

参数。 为简单起见，未显示偏差。 如图所示。 [11](#_bookmark75)，上下文向量c*t*可以用两种方式计算：全局和局部。 我们将在接下来的两个小节中描述每种方法。

**关**于光束搜索的注释。 尝试词汇表中所有可能的单词组合来找到具有最高联合概率的目标句子是难以处理的。 但另一方面，以纯贪婪的方式生成y，即每次选择最可能的单词，都是非常不理想的。 在实践中，用梁探索了一定数量的候选翻译

*搜*索，一种启发式搜索算法[[7](#_bookmark95)]。 较大的K值会产生更好的目标句子，但会降低解码速度。

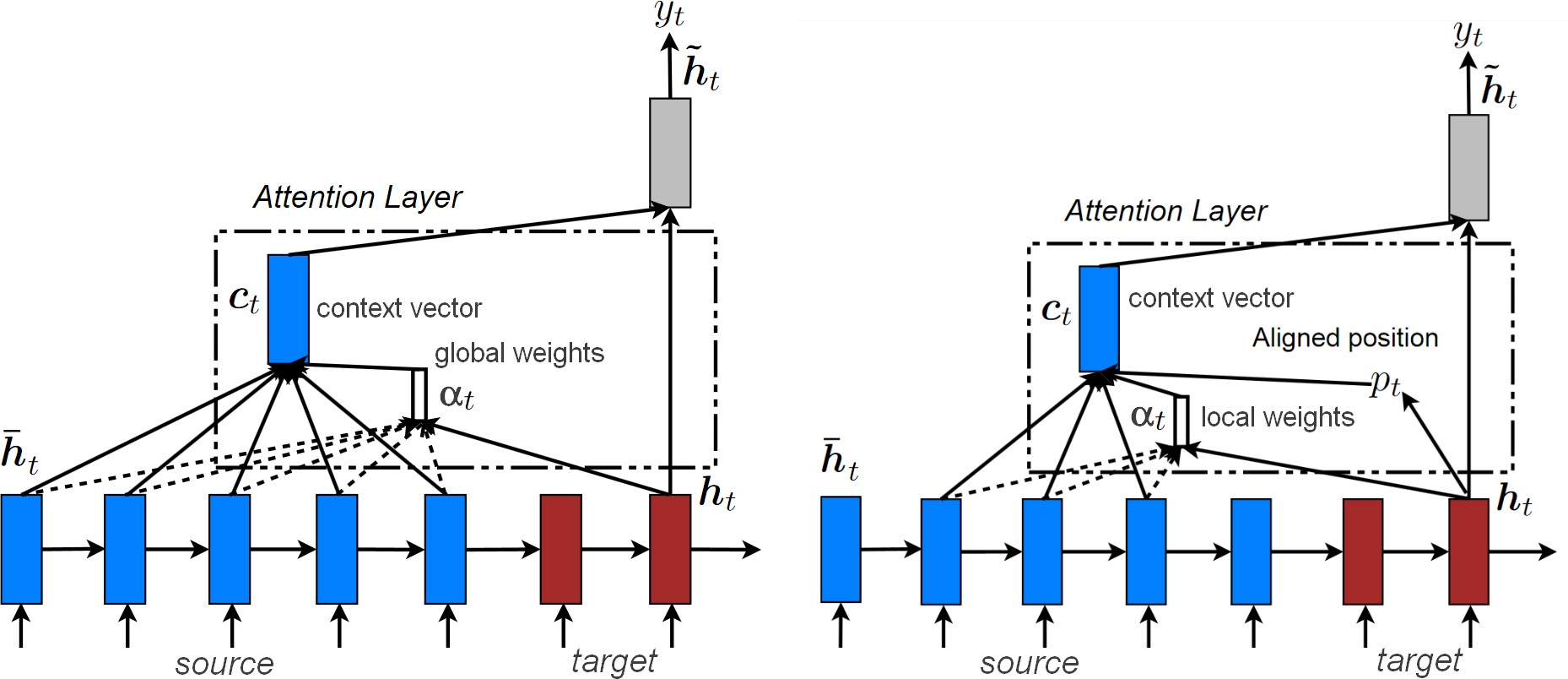


图11：全球（左）与本地关注（右）。 改编自 [[20](#_bookmark108)].

* + 1. **全球关注**

这里，上下文向量c*t*被计算为源句的注释的完整列表h¯i（即，编码器的隐藏状态）的加权和。 有T*x*注释。 每一个都是一个大小的向量，它是编码器隐藏层中单元的数量。 c*t*与任何注释具有相同的大小。 对齐矢量α*t*的大小等于源句的大小T*x*，因此它是可变的。

*德克萨斯州*

*ct=* *αt,i¯我* (18)

*i*=1

通过将softmax应用于比对的输出来计算比对矢量α*t*

当前目标隐藏状态h*t*和所有源隐藏状态¯之间的操作（得分（））：

*αt，i =*

*T*x i*I*= 1

(得分（h*t*，¯iI）)

(19)

换句话说，α*t*是所有源隐藏状态的概率分布（其系数都在0和1之间，总和为1），并指示源句中的哪些单词是

口

exp(得分（h*t*，h¯i）)

最有可能帮助预测下一个单词。 得分（）理论上可以是任何比较函数。 Luong等。 [[20](#_bookmark108)用点积（得分（h*t*，¯i）= h*t ¯*i）进行实验，这是一种具有参数矩阵的更通用的配方（得分（h*t*，¯i）= h*t*W*α*h¯i），并且完全连接

层。 他们发现dot更适合全球关注，而一般情况对于本地人来说更优越

注意。 图1中提供了全球关注的概述。 [12](#_bookmark78).

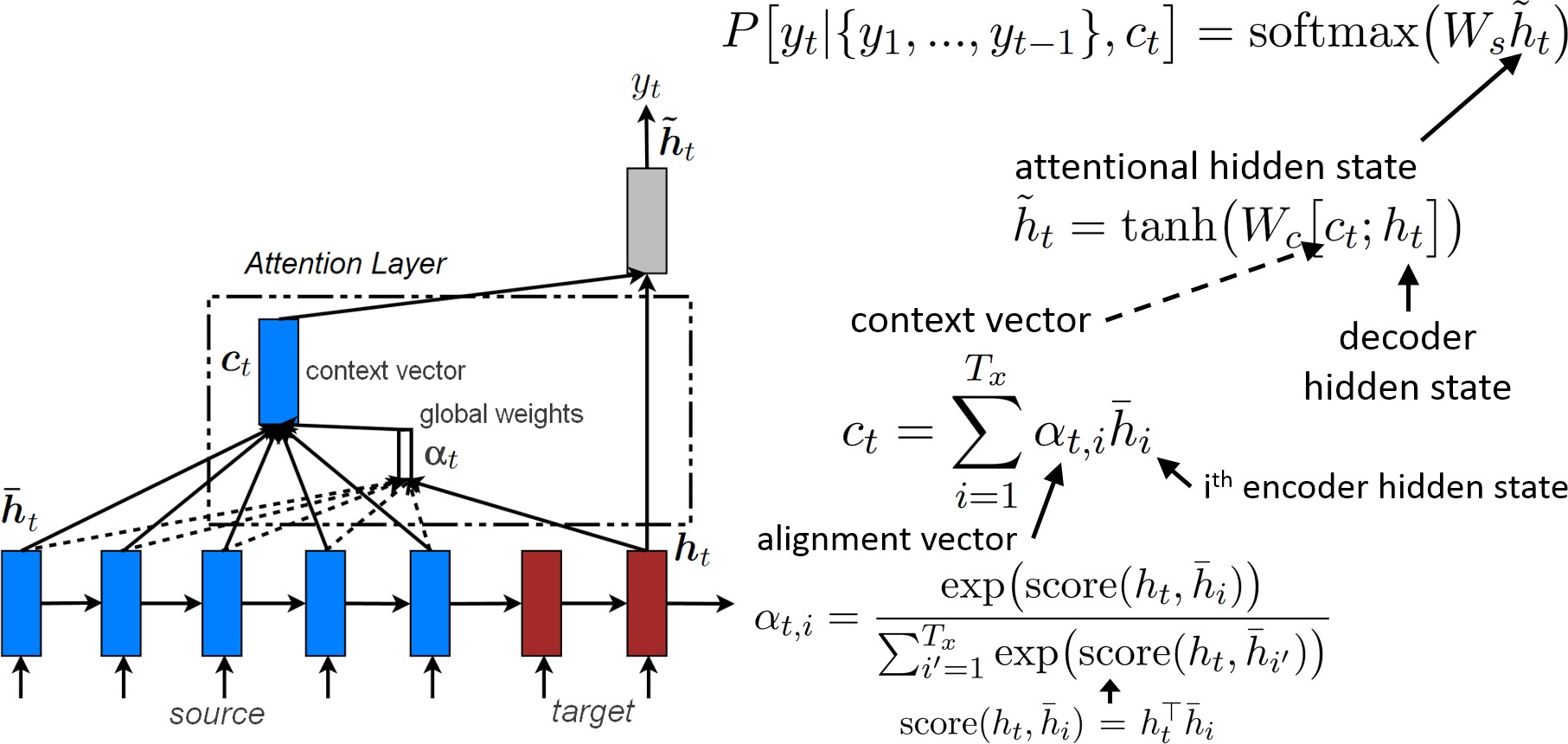


图12：全球关注机制摘要[[20](#_bookmark108)].

* + 1. **当地的关注**

考虑源句中的所有单词以生成每个单个目标单词是昂贵的，并且可能不是必需的。 为了解决这个问题，Luong等人。 [[20](#_bookmark108)]建议只关注固定尺寸2D + 1注释的小窗口：

*ct=*

*pt*+*D*

*I = PT-d*

*αt,iH*¯*我* (20)

*D*由用户规定，并且窗口居中的位置p*t*被设置为t（单调对齐）或者基于关于先前生成的目标词的信息{y（由可预测对齐）确定（y）。 TF914），...，y*t−1*}存储在h*t*中：

( )

*pt= Tx·σv的pTtanh（Wpht）* (21)

其中T*x*是源句的长度，σ是S形函数，v*p*和W*p*是可训练的参数。 对齐权重的计算方式类似于全局关注（Eq。 [19](#_bookmark77)），增加了一个以p*t*为中心且标准偏差的正态分布项

*D/*2:

exp(得分（h*t*，¯i）) ( （*i - pt）*2\

*t,i p + D.* 口

)

*α* =

*t*

*II= PT-d*

*∈ ∩* l *∈ ∩ −*

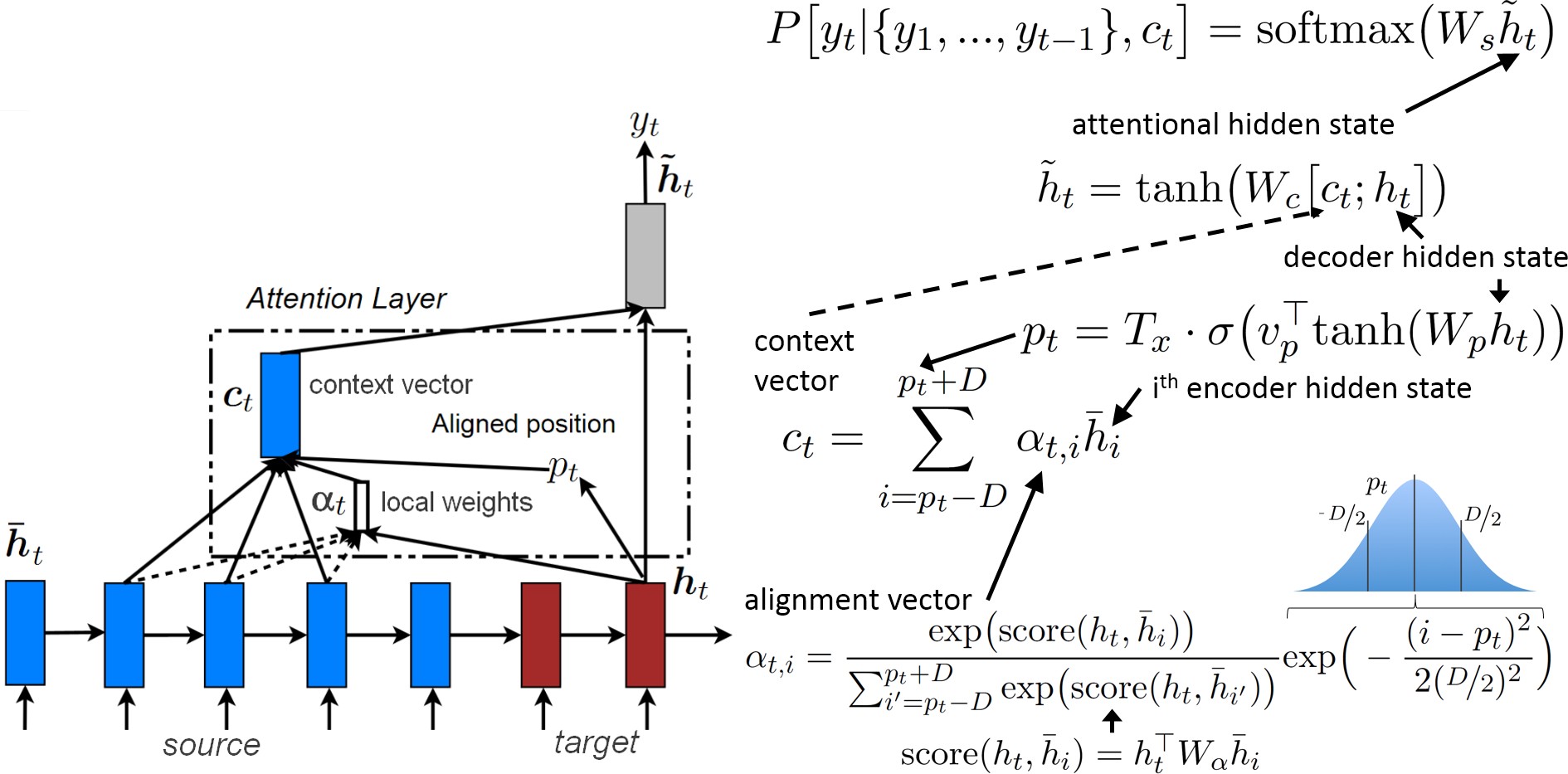
(得分（h*t*，¯iI）

口

*−* 2(*D/*2)2

(22)

注意p*t*R 0，T*x*和i N p*t*D，p*t*+ D]。 当我离开窗口p的中心*t*时，高斯项的加法使得对准权重衰减，即，它更加重视p*t*附近的注释。 此外，与全局关注不同，α*t*的大小是固定的并且等于2D + 1，因为仅考虑窗口内的注释。 实际上，局部注意力可以被视为全局关注，其中对齐权重乘以截断的正态分布（即，在窗口外返回零）。 图1中提供了本地关注的摘要。 [13](#_bookmark81).



图*13：具有预测对齐机制的局部注意力摘要[*[20](#_bookmark108)].

* 1. **自我关注**

我们在一个更简单的设置中使用单个RNN编码器将序列x1作为输入. . . ，x*T*

*h*1*,. . . ，hT*

( )

长度T。 像往常一样，RNN将输入序列映射到注释序列

.

目标与编码器 - 解码器环境中的注意力完全相同：而不是

将最后一个注释h*T*作为整个序列的综合概要，其倾向于信息丢失，通过考虑所有时间步骤的注释来计算新的隐藏表示。 为此目的，2016/2017年文献中出现了自我关注或内心关注机制，例如[[27](#_bookmark115), [18](#_bookmark106)]。 接下来我们使用[[[27](#_bookmark115)].

如公式所示。 [23](#_bookmark84)，注释h*t*首先传递到密集层。 然后通过将致密层的输出u*t*与可训练的上下文矢量u（随机初始化）进行比较并用softmax进行归一化来导出对准系数α*t*。 最终获得注意向量s作为注释的加权和。

*你t*= tanh（W h*t*）

exp（得分（u*t*，u））

*αt=*

*T* exp（得分（u，u））*t*

*吨I*= 1

(23)

*T*

*s* = *αtHt*

*t*=1

得分在理论上可以是任何对齐函数。 一种直接的方法是使用得分（u*t*，u）= u*t*u。 平均向量可以被解释为最佳单词的表示。 当面对一个新的例子时，该模型使用这些知识来决定它应该是哪个单词

注意。 在训练期间，通过反向传播，模型更新上下文向量，即，它调整其最佳单词的内部表示。

* + 1. **与seq2seq关注的区别**

上面的自我关注定义中的上下文向量与seq2seq注意中使用的上下文向量无关！ 在seq2seq中，上下文向量c*t*等于加权和

*Tx i*=1

*αt,i¯*i，用于计算注意隐藏状态为˜t = tanh(W*c*c*t*; H*TL）*。 在

然而，由于没有解码器，因此当使用score（）执行对齐时，上下文向量简单地用作解码器的隐藏状态的替代。 因此，在自我关注中，对齐向量α表示每个输入单词相对于最佳单词（平均）的相似性，而在seq2seq注意中，α表示每个源单词在生成目标的下一个元素时的相关性句子。

* + 1. **分层关注**

图3中所示的体系结构提供了一个简单，良好的例子，说明自我关注在实践中如何有用。 [14](#_bookmark88)。 在这种架构中，自我关注机制发挥了两次：在单词级别和句子级别。 这种方法有两个原因：首先，它与文档的自然层次结构（单词）相匹配 句子文件）。 其次，在计算文档的编码时，它允许模型首先确定哪些单词在每个句子中是重要的，然后，哪些句子总体上是重要的。 通过能够通过句子注意系数重新调整单词注意系数，该模型捕获这样的事实：当在给定句子中找到时，给定的单词实例可能非常重要，但是同一单词的另一个实例可能不是

*→*

*→*

在另一句话中找到的那个重要。

句子编码器

文件编码器



*文件向量*

*自我关注*

**…**

*句子注释*

bidirGRU

单词1单词2单词3 ...单词T.

…

*单词向量*

bidirGRU

…

*单词注释*

*句子矢量*

*自我关注*

*句子向量 *

句子编码器

句子编码器

句子编码器

*单词向量* **…**

**…**

**句子N.**

**第2句**

**第1句**

图14：分层注意架构[[27](#_bookmark115)].

**参考**

1. Bahdanau，Dzmitry，Kyunghyun Cho和Yoshua Bengio。 “通过联合学习对齐和翻译来进行神经机器翻译。” arXiv preprint arXiv：1409.0473（2014）。 [2](#_bookmark0), [14](#_bookmark66), [15](#_bookmark72)
2. Cho，K.，VanMerriënboer，B.，Gulcehre，C.，Bahdanau，D.，Bougares，F.，Schwenk，H.，Bengio，Y。（2014）。 学习使用RNN编码器 - 解码器进行统计机器翻译的短语表示。 arXiv preprint arXiv：1406.1078。 [2](#_bookmark0), [12](#_bookmark56), [13](#_bookmark61), [14](#_bookmark66)
3. Chopra，Sumit，Raia Hadsell和Yann LeCun。 “有条理地学习相似性度量，应用于面部验证。” 计算机视觉和模式识别，2005年。

CVPR 2005. IEEE计算机学会会议。 卷。 1. IEEE，2005。 [6](#_bookmark29)

1. Chung，J.，Gulcehre，C.，Cho，K。，＆Bengio，Y。（2014）。 门控递归神经网络在序列建模中的实证评价。 arXiv preprint arXiv：1412.3555。 [2](#_bookmark0), [13](#_bookmark61), [14](#_bookmark66)
2. Conneau，A.，Kiela，D.，Schwenk，H.，Barrault，L。，＆Bordes，A。（2017）。 从自然语言推理数据监督学习通用句子表示。 arXiv preprint arXiv：1705.02364。 [14](#_bookmark66)
3. Elman，JL（1990）。 及时发现结构。 认知科学，14：2,179-211。 [9](#_bookmark42)
4. Freitag，Markus和Yaser Al-Onaizan。 “用于神经机器翻译的光束搜索策略。” arXiv preprint arXiv：1702.01806（2017）。 [16](#_bookmark74)
5. Goldberg，Y。（2015）。 自然语言处理神经网络模型的入门读物。

Journal of Artificial Intelligence Research，57,345-420。 [2](#_bookmark0), [3](#_bookmark11), [9](#_bookmark42)

1. Graves，A。（2013）。 用递归神经网络生成序列。 arXiv preprint arXiv：1308.0850。 [11](#_bookmark50)
2. 格雷夫，克劳斯等人。 “LSTM：搜索空间奥德赛。” 关于神经网络和学习系统的IEEE事务28.10（2017）：2222-2232。 [14](#_bookmark66)
3. Hochreiter，S.，Schmidhuber，J。（1997）。 长期短暂记忆。 神经计算，9（8），1735 [11](#_bookmark50)
4. Hubel，David H.和Torsten N. Wiesel（1962）。 猫的视觉皮层中的感受野，双目互动和功能结构。 The Journal of physiology 160.1：106-154。 [4](#_bookmark20)
5. Johnson，R.，Zhang，T。（2015）。 用卷积神经网络有效利用词序进行文本分类。 出现：NAACL-2015，（2011）。 [2](#_bookmark0), [8](#_bookmark38)
6. Kim，Y。（2014）。 句子分类的卷积神经网络。 2014年自然语言处理经验方法会议论文集（EMNLP 2014），1746-1751。 [2](#_bookmark0), [3](#_bookmark11), [4](#_bookmark20)
7. Krizhevsky，Alex，Ilya Sutskever和Geoffrey E. Hinton。 “使用深度卷积神经网络进行Imagenet分类。” 神经信息处理系统的进展。 2012。 [4](#_bookmark20)
8. LeCun，Y.，Bottou，L.，Bengio，Y。和Haffner，P。（1998）。 基于梯度的学习应用于文档识别。 IEEE，86（11），2278-2324的会议记录。 [2](#_bookmark0), [4](#_bookmark20)
9. Li，J.，Chen，X.，Hovy，E。和Jurafsky，D。（2015）。 在nlp中可视化和理解神经模型。 arXiv preprint arXiv：1506.01066。 [8](#_bookmark38)
10. 林，周汉，等。 “一种结构化的自我痴迷句子嵌入。” arXiv preprint arXiv：1703.03130（2017）。 [18](#_bookmark82)
11. Lipton，Zachary C.，John Berkowitz和Charles Elkan。 “对序列学习的递归神经网络进行评论。” arXiv preprint arXiv：1506.00019（2015）。 [9](#_bookmark42)
12. Luong，Minh-Thang，Hieu Pham和Christopher D. Manning。 “基于注意力的神经机器翻译的有效方法。” arXiv preprint arXiv：1508.04025（2015）。 [2](#_bookmark0), [14](#_bookmark66), [15](#_bookmark72), [16](#_bookmark74), [17](#_bookmark79), [18](#_bookmark82)
13. Rush，Alexander M.，Sumit Chopra和Jason Weston。 “抽象句子摘要的神经注意模型。” arXiv preprint arXiv：1509.00685（2015）。 [14](#_bookmark66)
14. Simonyan，K.，Vedaldi，A。和Zisserman，A。（2013）。 Simonyan，Karen，Andrea Vedaldi和Andrew Zisserman。 “深入卷积网络：可视化图像分类模型和显着性图。” arXiv preprint arXiv：1312.6034（2013）。 arXiv preprint arXiv：1312.6034。 [8](#_bookmark38)
15. Springenberg，Jost Tobias，et al。 “为简单而努力：全部卷积网。” arXiv preprint arXiv：1412.6806（2014）。 [4](#_bookmark20)
16. Sutskever，Ilya，Oriol Vinyals和Quoc V. Le。 “用神经网络进行序列学习的序列。” 神经信息处理系统的进展。 2014年 [2](#_bookmark0), [6](#_bookmark29), [14](#_bookmark66)
17. Xu，K.，Ba，J.，Kiros，R.，Cho，K.，Courville，A.，Salakhudinov，R。，...＆Bengio，Y。（2015，June）。 显示，参与和讲述：通过视觉注意生成神经图像标题。 在国际机器学习会议（第2048-2057页）。 [14](#_bookmark66)
18. 张，叶和拜伦华莱士。 “用于句子分类的卷积神经网络（和实践者指南）的敏感性分析。” arXiv preprint arXiv：1510.03820（2015）。 [2](#_bookmark0), [3](#_bookmark11), [5](#_bookmark25), [6](#_bookmark29)
19. 杨子超等 “用于文档分类的分层注意网络。” 2016年计算语言学协会北美分会会议记录：人类语言技术。 2016年 [14](#_bookmark66), [18](#_bookmark82), [19](#_bookmark86)