

CSDN新首页上线啦，邀请你来立即体验！(http://blog.csdn.net/)

立即体
验

CSDN

博客 (//blog.csdn.net/DefaultHome) 学院 (//edu.csdn.net?ref=toolbar)

下载 (//download.csdn.net?ref=toolbar) GitChat (//gitbook.cn/?ref=csdn)

更多 ▾



4



weixin_3506... (//my.csdn.net?ref=toolbar)

(//write.blog.csdn.net/postedit/activity?ref=toolbar)source=csdnblog

深度学习笔记——Attention Model (注意力模型) 学习总结

原创

2017年08月06日 21:49:46

7779



mpk_no1 (http://blog.csdn.net/mpk_no1)

+ 关注

(http://blog.csdn.net/mpk_no1)

码云

原创

22

粉丝

21

喜欢

1

未开通
(https://github.com/utm_sourc

深度学习里的Attention model其实模拟的是人脑的注意力模型，举个例子来说，当我们观赏一幅画时，虽然我们可以看到整幅画的全貌，但是在我们深入仔细地观察时，其实眼睛聚焦的就只有很小的一块，这个时候人的大脑主要关注在这一小块图案上，也就是说这个时候人脑对整幅图的关注并不是均衡的，是有一定的权重区分的。这就是深度学习里的Attention Model的核心思想。

AM刚开始也确实是应用在图像领域里的，AM在图像处理领域取得了非常好的效果！于是，就有人开始研究怎么将AM模型引入到NLP领域。最有名的当属“Neural machine translation by jointly learning to align and translate”这篇论文了，这篇论文最早提出了Soft Attention Model，并将其应用到了机器翻译领域。后续NLP领域使用AM模型的文章一般都会引用这篇文章（目前引用量已经上千了！！）

如下图所示，机器翻译主要使用的是Encoder-Decoder模型，在Encoder-Decoder模型的基础上引入了AM，取得了不错的效果：

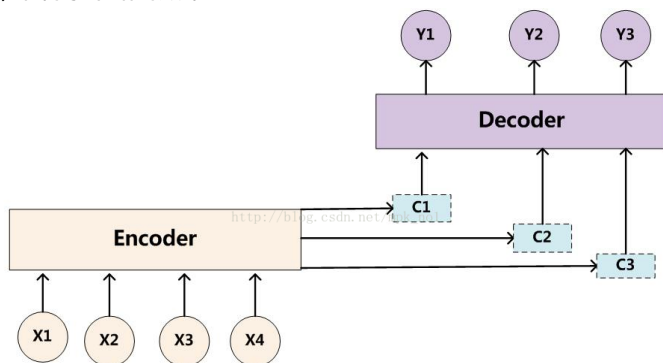


图2 引入AM模型的Encoder-Decoder框架

他的最新文章

更多文章 (http://blog.csdn.net/mpk_no1)

8大经典排序算法及其实现代码 (http://blog.csdn.net/mpk_no1/article/details/76397713)

频繁项挖掘-Apriori算法 (http://blog.csdn.net/mpk_no1/article/details/75929458)

自动编码器AutoEncoder学习总结 (http://blog.csdn.net/mpk_no1/article/details/75201582)

关键词提取方法学习总结 (TF-IDF、Topic-model、RAKE) (http://blog.csdn.net/mpk_no1/article/details/75201546)

Lucene学习总结 (http://blog.csdn.net/mpk_no1/article/details/75201515)

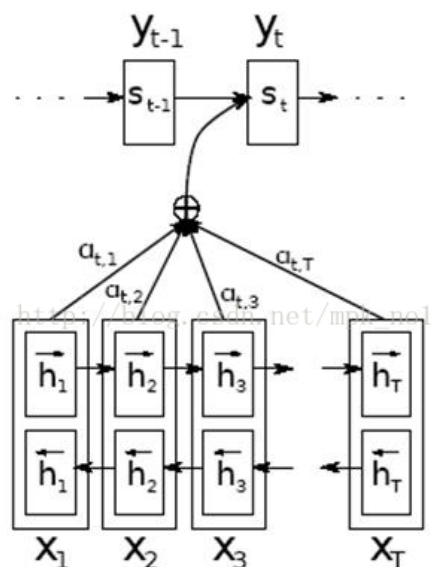
内容举报

返回顶部

Soft Attention Model :

这里其实是上面图的拆解, 我们前面说过, “Neural machine translation by jointly learning to align and translate”这篇论文提出了soft Attention Model, 并将其应用到了机器翻译上面。其实, 所谓Soft, 意思是在求注意力分配概率分布的时候, 对于输入句子X中任意一个单词都给出个概率, 是个概率分布。

即上图中的 c_i 是对Encoder中每一个单词都要计算一个注意力概率分布, 然后加权得到的。如下图所示:



其实有Soft AM, 对应也有一个Hard AM。既然Soft是给每个单词都赋予一个单词对齐概率, 那么如果不这样做, 直接从输入句子里面找到某个特定的单词, 然后把目标句子单词和这个单词对齐, 而其它输入句子中的单词硬性地为0, 这就是Hard Attention Model的思想。Hard AM在图像里证明有用, 但是在文本里面用处不大, 因为这种单词一一对齐明显要求太高, 如果对不齐对后续处理负面影响很大。但是, 斯坦福大学的一篇paper“Effective Approaches to Attention-based Neural Machine Translation”提出了一个混合Soft AM 和Hard AM的模型, 论文中, 他们提出了两种模型: Global Attention Model和Local Attention Model, Global Attention Model其实就是Soft Attention Model, Local Attention Model本质上是Soft AM和Hard AM的一个混合。一般首先预估一个对齐位置 P_t , 然后在 P_t 左右大小为D的窗口范围来取类似于Soft AM的概率分布。

Global Attention Model和Local Attention Model

相关推荐

自然语言处理中的Attention Model : 是什么及为什么 (<http://blog.csdn.net/malefactor/article/details/50550211>)

以Attention Model为例谈谈两种研究创新模式 (<http://blog.csdn.net/malefactor/article/details/50583474>)

浅谈Attention-based Model【原理篇】 (<http://blog.csdn.net/wuzqChom/article/details/75792501>)

深度学习方法(九): 自然语言处理中的Attention Model/注意力模型 (<http://blog.csdn.net/xbinworld/article/details/54607525>)



内容举报

返回顶部

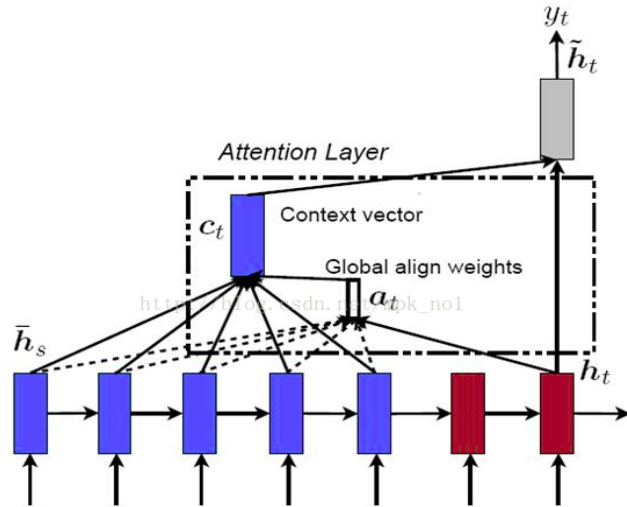


图3. Global Attention Model

Global AM其实就是soft AM，Decoder的过程中，每一个时间步的Context vector需要计算Encoder中每一个单词的注意力权重，然后加权得到。



西班牙跟团游

ui界面设计工具

透明手机多少钱

moto新款手机

他的热门文章

深度学习笔记——Attention Model (注意力模型) 学习总结 (http://blog.csdn.net/mpk_no1/article/details/72862348)

📖 7417

深度学习笔记——Word2vec和Doc2vec原理解并结合代码分析 (http://blog.csdn.net/mpk_no1/article/details/72458003)

📖 3384

深度学习笔记——Word2vec和Doc2vec应用举例：词和句子的相似度计算 (http://blog.csdn.net/mpk_no1/article/details/72510677)

📖 3078

关键词提取方法学习总结 (TF-IDF、Topic-model、RAKE) (http://blog.csdn.net/mpk_no1/article/details/75201546)

📖 2591

深度学习笔记——TensorFlow学习总结 (三) 使用tf.nn.nn_module实现的神经网络进阶

📖 1637



内容举报



返回顶部

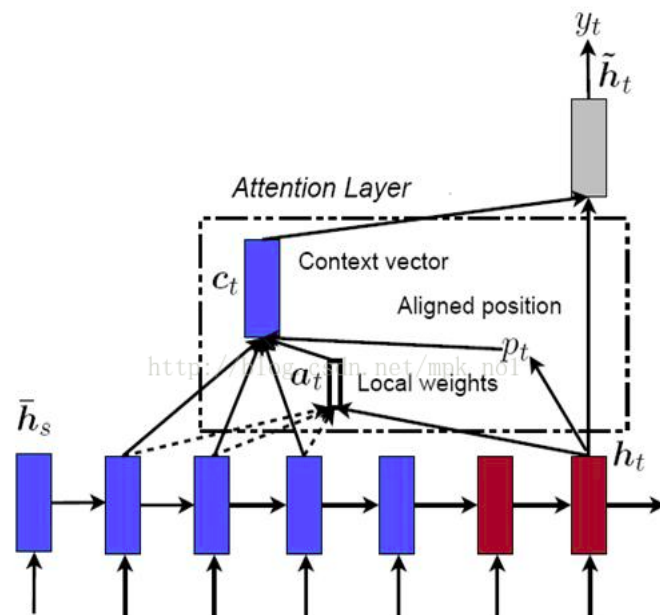


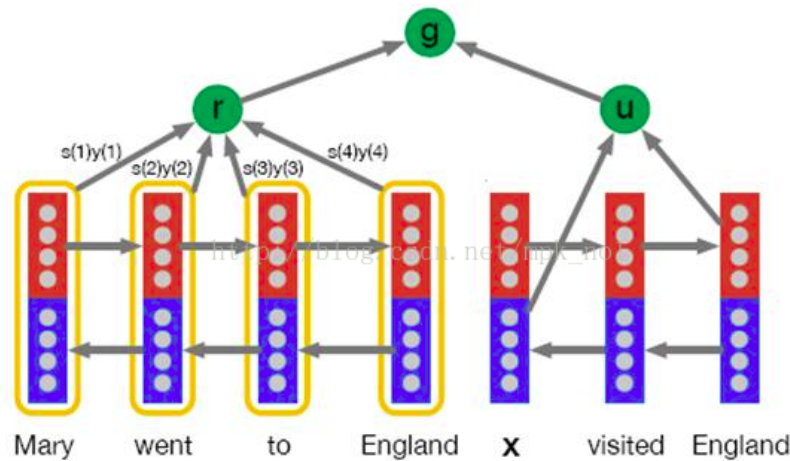
图4. LocalAttention Model

Local AM则是首先找到一个对其位置，然后在对其位置左右一个窗口内来计算注意力权重，最终加权得到Context vector。这其实是Soft AM和Hard AM的一个混合折中。

静态AM

其实还有一种AM叫做静态AM。所谓静态AM，其实是指对于一个文档或者句子，计算每个词的注意力概率分布，然后加权得到一个向量来代表这个文档或者句子的向量表示。跟soft AM的区别是，soft AM在Decoder的过程中每一次都需要重新对所有词计算一遍注意力概率分布，然后加权得到context vector，但是静态AM只计算一次得到句子的向量表示即可。（这其实是针对于不同的任务而做出的改变）





强制前向AM

Soft AM在逐步生成目标句子单词的时候，是由前向后逐步生成的，但是每个单词在求输入句子单词对齐模型时，并没有什么特殊要求。强制前向AM则增加了约束条件：要求在生成目标句子单词时，如果某个输入句子单词已经和输出单词对齐了，那么后面基本不太考虑再用它了，因为输入和输出都是逐步往前走的，所以看上去类似于强制对齐规则在往前走。

看了这么多AM模型以及变种，那么我们来看一看AM模型具体怎么实现，涉及的公式都是怎样的。

我们知道，注意力机制是在序列到序列模型中用于注意编码器状态的最常用方法，它同时还可用于回顾序列模型的过去状态。使用注意力机制，系统能基于隐藏状态 s_1, \dots, s_m 而获得环境向量 (context vector) c_i ，这些环境向量可以和当前的隐藏状态 h_i 一起实现预测。环境向量 c_i 可以由前面状态的加权平均数得出，其中状态所加的权就是注意力权重 a_i ：

$$c_i = \sum_j a_{ij} s_j$$

$$a_i = \text{softmax}(f_{\text{att}}(h_i, s_j))$$

注意力函数 $f_{\text{att}}(h_i, s_j)$ 计算的是目前的隐藏状态 h_i 和前面的隐藏状态 s_j 之间的非归一化分配值。

而实际上，注意力函数也有很多种变体。接下来我们将讨论四种注意力变体：加性注意力 (additive attention)、乘法 (点积) 注意力 (multiplicative attention)、自注意力 (self-attention) 和关键值注意力 (key-value attention)。

加性注意力 (additive attention)



内容举报



返回顶部

加性注意力是最经典的注意力机制 (Bahdanau et al., 2015) [15], 它使用了有一个隐藏层的前馈网络 (全连接) 来计算注意力的分配:

$$f_{att}(\mathbf{h}_i, \mathbf{s}_j) = \mathbf{v}_a^\top \tanh(\mathbf{W}_a[\mathbf{h}_i; \mathbf{s}_j])$$

也就是: $f_{att}(\mathbf{h}_i, \mathbf{s}_j) = \mathbf{v}_a^\top \tanh(\mathbf{W}_1 \mathbf{h}_i + \mathbf{W}_2 \mathbf{s}_j)$

乘法 (点积) 注意力 (multiplicative attention)

乘法注意力 (Multiplicative attention) (Luong et al., 2015) [16] 通过计算以下函数而简化了注意力操作:

$$f_{att}(\mathbf{h}_i, \mathbf{s}_j) = \mathbf{h}_i^\top \mathbf{W}_a \mathbf{s}_j$$

加性注意力和乘法注意力在复杂度上是相似的, 但是乘法注意力在实践中往往要更快速、具有更高效的存储, 因为它可以使用矩阵操作更高效地实现。两个变体在低维度 d_h 解码器状态中性能相似, 但加性注意力机制在更高的维度上性能更优。

自注意力 (self-attention)

注意力机制不仅能用来处理编码器或前面的隐藏层, 它同样还能用来获得其他特征的分布, 例如阅读理解任务中作为文本的词嵌入 (Kadlec et al., 2017) [37]。然而, 注意力机制并不直接适用于分类任务, 因为这些任务并不需要情感分析 (sentiment analysis) 等额外的信息。在这些模型中, 通常我们使用 LSTM 的最终隐藏状态或像最大池化和平均池化那样的聚合函数来表征句子。

自注意力机制 (Self-attention) 通常也不会使用其他额外的信息, 但是它能使用自注意力关注本身进而从句子中抽取相关信息 (Lin et al., 2017) [18]。自注意力又称作内部注意力, 它在很多任务上都有十分出色的表现, 比如阅读理解 (Cheng et al., 2016) [38]、文本继承 (textual entailment/Parikh et al., 2016) [39]、自动文本摘要 (Paulus et al., 2017) [40]。

$$\mathbf{A} = \text{softmax}(\mathbf{V}_a \tanh(\mathbf{W}_a \mathbf{H}^\top))$$

$$\mathbf{C} \Leftarrow \mathbf{A} \mathbf{H}$$

关键值注意力 (key-value attention)

关键值注意力 (Daniluk et al., 2017) [19] 是最近出现的注意力变体机制, 它将形式和函数分开, 从而为注意力计算保持分离的向量。它同样在多种文本建模任务 (Liu & Lapata, 2017) [41] 中发挥了很大的作用。具体来说, 关键值注意力将每一个隐藏向量 \mathbf{h}_i 分离为一个键值 \mathbf{k}_i 和一个向量 \mathbf{v}_i : $[\mathbf{k}_i; \mathbf{v}_i] = \mathbf{h}_i$ 。键值使用加性注意力来计算注意力分布 \mathbf{a}_i :

$$\mathbf{a}_i = \text{softmax}(\mathbf{v}_a^\top \tanh(\mathbf{W}_1 [\mathbf{k}_{i-L}; \dots; \mathbf{k}_{i-1}] + (\mathbf{W}_2 \mathbf{k}_i) \mathbf{1}^\top))$$

其中 L 为注意力窗体的长度, $\mathbf{1}$ 为所有单元为 1 的向量。然后使用注意力分布值可以求得环境表征 \mathbf{c}_i :

$$\mathbf{c}_i = [\mathbf{v}_{i-L}; \dots; \mathbf{v}_{i-1}] \mathbf{a}^\top$$



4



内容举报



返回顶部

其中环境向量 c_i 将联合现阶段的状态值 v_i 进行预测。

最后的最后，再加一个自己用keras实现的简单的静态AM（自注意力）层的代码吧：

```
[python]
1. from keras import backend as K
2. from keras.engine.topology import Layer
3. from keras import initializers, regularizers, constraints
4.
5. class Attention_layer(Layer):
6.     """
7.         Attention operation, with a context/query vector, for temporal data.
8.         Supports Masking.
9.         Follows the work of Yang et al. [https://www.cs.cmu.edu/~diyi/docs/naacl16.pdf]
10.        "Hierarchical Attention Networks for Document Classification"
11.        by using a context vector to assist the attention
12.        # Input shape
13.            3D tensor with shape: `(samples, steps, features)`.
14.        # Output shape
15.            2D tensor with shape: `(samples, features)`.
16.        :param kwargs:
17.            Just put it on top of an RNN Layer (GRU/LSTM/SimpleRNN) with return_sequences=True.
18.            The dimensions are inferred based on the output shape of the RNN.
19.        Example:
20.            model.add(LSTM(64, return_sequences=True))
21.            model.add(AttentionWithContext())
22.        """
23.
24.    def __init__(self,
25.                 W_regularizer=None, b_regularizer=None,
26.                 W_constraint=None, b_constraint=None,
27.                 bias=True, **kwargs):
28.
29.        self.supports_masking = True
30.        self.init = initializers.get('glorot_uniform')
31.
32.        self.W_regularizer = regularizers.get(W_regularizer)
33.        self.b_regularizer = regularizers.get(b_regularizer)
34.
35.        self.W_constraint = constraints.get(W_constraint)
36.        self.b_constraint = constraints.get(b_constraint)
37.
38.        self.bias = bias
39.        super(Attention_layer, self).__init__(**kwargs)
40.
41.    def build(self, input_shape):
```



内容举报



返回顶部



4



```

42.         assert len(input_shape) == 3
43.
44.         self.W = self.add_weight((input_shape[-1], input_shape[-1]),
45.                                   initializer=self.init,
46.                                   name='{}_W'.format(self.name),
47.                                   regularizer=self.W_regularizer,
48.                                   constraint=self.W_constraint)
49.
50.         if self.bias:
51.             self.b = self.add_weight((input_shape[-1],),
52.                                       initializer='zero',
53.                                       name='{}_b'.format(self.name),
54.                                       regularizer=self.b_regularizer,
55.                                       constraint=self.b_constraint)
56.
57.         super(Attention_layer, self).build(input_shape)
58.
59.     def compute_mask(self, input, input_mask=None):
60.         # do not pass the mask to the next layers
61.         return None
62.
63.     def call(self, x, mask=None):
64.         uit = K.dot(x, self.W)
65.
66.         if self.bias:
67.             uit += self.b
68.
69.         uit = K.tanh(uit)
70.
71.         a = K.exp(uit)
72.
73.         # apply mask after the exp. will be re-normalized next
74.         if mask is not None:
75.             # Cast the mask to floatX to avoid float64 upcasting in theano
76.             a *= K.cast(mask, K.floatx())
77.
78.         # in some cases especially in the early stages of training the sum may be almost zero
79.         # and this results in NaN's. A workaround is to add a very small positive number to the
80.         # a /= K.cast(K.sum(a, axis=1, keepdims=True), K.floatx())
81.         a /= K.cast(K.sum(a, axis=1, keepdims=True) + K.epsilon(), K.floatx())
82.         print a
83.         # a = K.expand_dims(a)
84.         print x
85.         weighted_input = x * a
86.         print weighted_input
87.         return K.sum(weighted_input, axis=1)
88.
89.     def compute_output_shape(self, input_shape):

```



内容举报



返回顶部


```
89.         return (input_shape[0], input_shape[-1])
```

版权声明：本文为博主原创文章，未经博主允许不得转载。



4



发表你的评论

(http://my.csdn.net/weixin_35068028)



u012442157 (/u012442157) 2017-11-15 12:14

3楼

(/u012442157)下 (Kadlec et al., 2017) [37]是引用的哪一篇论文

回复



Math2046 (/Math2046) 2017-11-04 15:20

2楼

(/Math2046)您好，我想知道这篇文章翻译的英文文献名字，里面涉及到的参考文献是什么，谢谢。

回复



m0_37143327 (/m0_37143327) 2017-10-29 16:22

1楼

(/m0_37143327)我想问一下，AM模型最初是应用在图像处理的哪些领域，或者说博主有没有些图像领域的论文推荐？

谢谢

回复

相关文章推荐

自然语言处理中的Attention Model：是什么及为什么 (<http://blog.csdn.net/malefactor/article...>)

要是关注深度学习在自然语言处理方面的研究进展，我相信你一定听说过Attention Model（后文有时会简称AM模型）这个词。AM模型应该说是过去一年来NLP领域中的重要进展之一，在很多场景被证明有...



内容举报



返回顶部



malefactor (<http://blog.csdn.net/malefactor>) 2016年01月20日 18:26 58071

以Attention Model为例谈谈两种研究创新模式 (<http://blog.csdn.net/malefactor/article/detail...>)

各位观众朋友好，也许此刻您刚打开电梯.....上了年纪的读者估计能看出上面一句是引用了韩乔生老先生的名言，我写东西就喜欢用名人名言开头，这好习惯这么多年怎么也改不了。您问韩乔生是谁？恭喜您，作为90后您以...



malefactor (<http://blog.csdn.net/malefactor>) 2016年01月26日 00:18 21394



太任性！学AI的应届学弟怒拒20K Offer，他想要多少钱？

AI改变命运呀！！前段时间在我司联合举办的校招招聘会上，一名刚刚毕业的学弟陆续拒绝2份Offer，企业给出18K、23K高薪，学弟拒绝后直接来了一句...

(http://www.baidu.com/cb.php?c=lgF_pyfqHmknjnvPjn0lZ0qnfK9ujYzP1f4PjDs0Aw-5Hc3rHnYnHb0TAq15HfLPWRznjb0T1d-rjubnWnkn1PbuW7bmVPh0AwY5HDdnHfznWRYPW00lgF_5y9YIZ0IQzq-uZR8mLPbUB48ugfElAqspynEmybz5LNYUNq1ULNzmvRqmhkEu1Ds0ZFb5HD0mhYqn0KsTWYs0ZNGujYkPHTYn1mk0AqGujYknWb3rjDY0APGujYLnWm4n1c0ULI85H00TZbqnWC)

浅谈Attention-based Model【原理篇】 (<http://blog.csdn.net/wuzqChom/article/details/757...>)

转载请标明出处：<http://blog.csdn.net/wuzqchom/article/details/75792501> 计划分为三个部分：浅谈Attention-based Model【...】



wuzqChom (<http://blog.csdn.net/wuzqChom>) 2017年07月22日 19:24 1662

深度学习方法（九）：自然语言处理中的Attention Model注意力模型 (<http://blog.csdn.net/xb...>)

上一篇博文深度学习方法（八）：Encoder-Decoder模型，基本Sequence to Sequence模型描述了基本的Encoder-Decoder模型，在作为翻译模型的时候，这种基本的Enc...



xbinworld (<http://blog.csdn.net/xbinworld>) 2017年02月04日 00:27 6505

浅谈Attention-based Model【源码篇】 (<http://blog.csdn.net/wuzqChom/article/details/779...>)



内容举报



返回顶部

转载请标明出处：http://blog.csdn.net/wuzqchom/article/details/77918780此为本人阅读tensorflow源码的记录，主要在一些步骤上加了一些注释和少...



wuzqChom (http://blog.csdn.net/wuzqChom) 2017年09月09日 22:54 694



自然语言处理中的Attention Model：是什么及为什么 (http://blog.csdn.net/jdbc/article/detail...)

要是关注深度学习在自然语言处理方面的研究进展，我相信你一定听说过Attention Model（后文有时会简称AM模型）这个词。AM模型应该说是过去一年来NLP领域中的重要进展之一，在很多场景被证明有...



jdbc (http://blog.csdn.net/jdbc) 2016年01月26日 11:16 2336

【深度学习】聚焦机制DRAM(Deep Recurrent Attention Model)算法详解 (http://blog.csdn.n...)

Visual Attention基础，Multiple object recognition with visual attention算法解读。



shenxiaolu1984 (http://blog.csdn.net/shenxiaolu1984) 2016年06月28日 22:14 6483

attention 机制 (http://blog.csdn.net/qq_26609915/article/details/52086772)

attention 机制什么是attentionattention机制是（非常）松散地基于人类的视觉注意机制。就是按照“高分辨率”聚焦在图片的某个特定区域并以“低分辨率”感知图像的周边区域的模式，然后...



qq_26609915 (http://blog.csdn.net/qq_26609915) 2016年08月01日 16:24 5285

深度学习中的Attention模型介绍及其进展 (http://blog.csdn.net/jteng/article/details/52864401)

近期对深度学习中的Attention模型进行了深入研究，该模型在图像识别、语音识别和自然语言处理三大深度学习的热门领域均有广泛的使用，是2014和2015年深度学习领域的重要进展。现对其原理、主要应用...



jteng (http://blog.csdn.net/jteng) 2016年10月20日 15:29 7022





内容举报



返回顶部

阅读理解任务中的Attention-over-Attention神经网络模型原理及实现 (<http://blog.csdn.net/liuchongge>)

本文是“Attention-over-Attention Neural Networks for Reading Comprehension”的阅读笔记。这篇论文所处理的任务是阅读理解里面的完形填空...

 liuchongge (<http://blog.csdn.net/liuchongge>) 2017年06月06日 09:24  2569





4





深度学习笔记(六)：Encoder-Decoder模型和Attention模型 (<http://blog.csdn.net/u014595019>)

这两天在看attention模型，看了下知乎上的几个回答，很多人都推荐了一篇文章Neural Machine Translation by Jointly Learning to Align and ...

 u014595019 (<http://blog.csdn.net/u014595019>) 2016年10月15日 23:09  19550



Attention注意力机制--原理与应用 (<http://blog.csdn.net/joshuaxx316/article/details/706653...>)

注意力机制即Attention mechanism在序列学习任务上具有巨大的提升作用，在编解码器框架内，通过在编码段加入A模型，对源数据序列进行数据加权变换，或者在解码端引入A模型，对目标数据进行加权...

 joshuaxx316 (<http://blog.csdn.net/joshuaxx316>) 2017年04月24日 21:51  9574



attention model (<http://blog.csdn.net/apsvfb/article/details/59110169>)

How did I select papers?First, I tried to search for "attention" in CVPR2014-2016, ICCV2009-2015 an...

 apsvfb (<http://blog.csdn.net/apsvfb>) 2017年03月01日 16:16  511

王小草【深度学习】笔记第七弹--RNN与应用案例：注意力模型与机器翻译 (http://blog.csdn.net/sinat_33761963)

标签（空格分隔）：王小草深度学习笔记1. 注意力模型1.2 注意力模型概述注意力模型（attention model）是一种用于做图像描述的模型。在笔记6中讲过RNN去做图像描述，但是精准度可能差强...

 sinat_33761963 (http://blog.csdn.net/sinat_33761963) 2016年12月08日 17:12  3047

视觉注意力的循环神经网络模型 (http://blog.csdn.net/Leo_Xu06/article/details/53491400)





内容举报





返回顶部

视觉注意力的循环神经网络模型

 Leo_Xu06 (http://blog.csdn.net/Leo_Xu06) 2016年12月07日 20:43  3988



对Attention is all you need 的理解 (<http://blog.csdn.net/mijiaoxiaosan/article/details/7325...>)

对谷歌Attention is all you need 的理解。

 mijiaoxiaosan (<http://blog.csdn.net/mijiaoxiaosan>) 2017年06月14日 19:24  5994



机器翻译之Facebook的CNN与Google的Attention (http://blog.csdn.net/Young_Gy/article/d...)

机器翻译的常用架构是seq2seq，可是seq2seq中的核心模型RNN是序列模型，后面的计算依赖于前面的计算，如何并行提高效率很是苦恼。最近，Facebook和Google的研究人员分别尝试用CNN...

 Young_Gy (http://blog.csdn.net/Young_Gy) 2017年07月03日 16:24  732



一种通过self-attention机制生成多维度aspect的句向量模型 (<http://blog.csdn.net/guoyuhaoa...>)

这篇博客主要借鉴了IBM沃森实验室的2017年ICLR会议的论文《A STRUCTURED SELF-ATTENTIVE SENTENCE EMBEDDING》。根据self-attention机制生...

 guoyuhaoaaa (<http://blog.csdn.net/guoyuhaoaaa>) 2017年12月03日 15:14  83

以Attention Model为例谈谈两种研究创新模式 (<http://blog.csdn.net/jdbc/article/details/5058...>)

本文主要介绍以Attention Model为例谈谈两种研究创新模式。

 jdbc (<http://blog.csdn.net/jdbc>) 2016年01月26日 11:19  1338

attention 机制入门 (<http://blog.csdn.net/aliceyangxi1987/article/details/76284315>)

在下面这两篇文章中都有提到 attention 机制：使聊天机器人的对话更有营养 如何自动生成文章摘要今天来看看 attention 是什么。下面这篇论文算是在NLP中第一个使用attenti...



4



内容举报



返回顶部