最新 (/)

团队号 (/channel/index?type=263&name=team)

移动开发 (/channel/index?type=272&name=mobile)

Java (/channel/index?type=257&name=java)

Python (/channel/index?type=268&name=python)

个人号 (/channel/index?type=271&name=person)

前端 (/channel/index?type=264&name=web)

后端 (/channel/index?type=269&name=server)

测试 (/channel/index?type=266&name=test)

更多 (/channel/index?type=265&name=other)

软件定义世界（SDX）(/author/index?authorId=626)

11月21日 1新/0 阅读： 23971 ： 0

简介：

用TensorFlow为图片添加字幕

2017-05-04 06:51 软件定义世界（SDX）

如何使用TensorFlow来构建和训练一个图片字幕生成器

图片字幕生成模型结合了近年来计算机视觉和机器翻译方面的进步，通过使用神经网络来生成现实图片的字幕。对于一个给定的输入图片，神经图像字幕模型被训练来最大化生成一个字幕的可能性。可以被用来产生新颖的图像描述。例如，下面是用MS COCO数据集训练的一个神经图像字幕生成器所产生的字幕。

图1. 来源：Paul Puri。图片来自MS COCO数据集

在这篇文章里，我们会介绍一个中级程度的教程，教大家如何使用谷歌的“Show and Tell”模型的一种变形和Flickr30k数据集来训练一个图片字幕生成器。我们使用TensorFlow的框架来构建、训练和测试我们的模型，因为它相对容易使用而且也有一个日益庞大的在线社区。

最新 (/)

团队号 (/channel/index?type=263&name=team)

移动开发 (/channel/index?type=272&name=mobile)

Java (/channel/index?type=258&name=java)

Python (/channel/index?type=268&name=python)

个人号 (/channel/index?type=271&name=person)

前端 (/channel/index?type=264&name=web)

后端 (/channel/index?type=269&name=server)

测试 (/channel/index?type=266&name=test)

更多 (/channel/index?type=270&name=other)

为什么要生成字幕？

近年来在计算机视觉和自然语言处理任务上应用深度神经网络的成功激励着AI研究人员去探索新的研究机会，交叉连接这些之前互相独立的领域。字幕生成模型就必须去对视觉线索和自然语言的理解进行平衡。

这两个传统上不关的领域的交叉有可能在更大的范围内产生变革。这一技术现在已经有一些很直接的应用。比如，为YouTube视频自动生成摘要或是标注未标记的图片。而更多的有创造力的应用则会大幅度提高一个更广泛的人群的生活质量。与传统的计算机视觉试图去让计算机能更好地接触和理解这个世界一样，这一新技术具有进一步让这个世界对人类更加可达与可理解的潜力。它可以是一个导游，甚至可以成为日常生活的一个视觉帮助服务。比如意大利的AI公司Eyra所开发的Horus可穿戴设备所展示的这个场景。

需要一些安装工作

在我们正式开始前，需要先做一些整理工作。

首先，你需要安装TensorFlow。如果这是你第一次使用TensorFlow，我们推荐你先看看这篇文章《你好，TensorFlow!从零开始构建和训练你的第一个TensorFlow图》

你需要安装pandas、OpenCV2和Jupyter库来保证相关的代码可以运行。不过为了简化安装的过程，我们强烈推荐你使用与本文关联的GitHub库里的这个Docker安装指南。

你还需要下载Flickr30k图片文件和图片字幕数据集。我们的GitHub库里有也提供了下载链接。

现在，让我们开始吧！

图片字幕生成模型

图2. 来源：Shannon Shih获取自加州大学伯克利分校机器学习组织。马的图片来自MS COCO

概括来看，这就是我们将要训练的模型。每张图片都可以被一个深度卷积神经网络编码成一个4096维的向量表示。一个语言生成RNN（循环神经网络）将会对这个表示按顺序解码，并生成自然语言的描述。

最新 (/)
团队号 (/channel/index?type=263&name=team)
移动开发 (/channel/index?type=272&name=mobile)
Java (/channel/index?type=257&name=java)
Python (/channel/index?type=268&name=python)
个人号 (/channel/index?type=271&name=person)
前端 (/channel/index?type=264&name=web)
后端 (/channel/index?type=269&name=server)
测试 (/channel/index?type=266&name=test)
更多 (/channel/index?type=270&name=other)

字幕生成是图像分类的一种扩展

作为计算机视觉的一个任务，图片分类有着很长的历史，且有非常多好的模型。分类需要模型能把图像里与形状和物体相关的视觉信息拼接在一起，然后把这个图片分入一个类别。其他的计算机视觉的机器学习模型，诸如物体检查和图片分割等，则不仅对呈现的信息进行识别，还通过学习如何解读二维的空间，并把这两种理解融合起来，从而能判断出图片里分布的物体信息。对于字幕生成，有两个主要的问题：

- 在获取图片里的重要信息时，我们如何基于图像分类模型的成功结果？
- 我们的模型如何能学习去融合对于语言的理解和对于图片的理解？

利用迁移学习

我们可以利用已有的模型来帮助实现图片生成字幕。迁移学习让我们可以把从其他任务上训练出来的神经网络的数据变换应用到我们自己的数据上。在我们的这个场景里，VGG-16图片分类模型是把224 x224像素的图片作为输入，产生4096维度的特征向量表示，用于分类图片。

我们可以利用这些VGG-16模型生成的表示（也叫做图向量）来训练我们的模型。限于本文的篇幅，我们省略了VGG-16的架构，而是直接用已经计算出来的4096维的特征来加快训练的过程。

导入VGG图片特征和图片字幕是相当得简单直接：

```
def get_data(annotation_path, feature_path):
    annotations = pd.read_table(annotation_path, sep='\t', header=None, names=['image', 'caption'])
    return np.load(feature_path,'r'), annotations['caption'].values
```

理解字幕

现在我们已经有了图片的表示了，还需要我们的模型能学会把这些表示解码成能被理解的字幕。因为文字天生的序列特性，我们会利用一个RNN/LSTM网络里的循环特点（想了解更多，请参考这篇“理解LSTM网络”）。这些网络被训练来预测在给定的一系列前置词汇和图片表示的情况下的下一个词。

长短期记忆（LSTM）单元能让这个模型能更好地选择什么样的信息可以用于字幕词汇序列，什么需要记忆，什么信息需要忘掉。TensorFlow提供了一个封装的功能可以对于给定的输入和输出维度生成一个LSTM层。

为了把词汇能变化成适合LSTM的固定长度表示的输入，我们使用一个向量层来把词汇映射成256维的特征（也叫词向量）。词向量帮助我们把词汇表示成向量，同时相似的词向量在语义上也相似。想了解更多关于词向量如何获取不同词汇之间的关系，请看这篇《用深度学习来获取文本语义》。

在这个VGG-16图片分类器里，卷积层抽取出了4096维表示，然后送入最后的softmax层来做分类。因为LSTM单元需要的是256维的文本输入，我们需要把图片表示转化成在句子中所需的这种表示。为了实现这个目标，我们需要加入另外一个向量层来学习把4096维的图片特征映射成256维的文本特征空间。

构建和训练这个模型

全合在一起，Show and Tell模型就大概像这个样子：

图3，来源《Show and Tell：2015 MSCOCO图片字幕大赛所获得的经验教训》

图3中，{$s0$, $s1$, ..., sN}表示我们试着去预测的字幕词汇，{$wes0$, $wes1$, ..., $wesN$-1}是每个词的词向量。LSTM的输出{$p1$, $p2$, ..., pN}是这个模型产生的句子里下一个词的概率分布。模型的训练目标是最小化对所有的词概率取对数后求和的负值。

```
def build_model(self):

# declaring the placeholders for our extracted image feature vectors, our caption, and our mask

# (describes how long our caption is with an array of 0/1 values of length `maxlen`

img = tf.placeholder(tf.float32, [self.batch_size, self.dim_in])

caption_placeholder = tf.placeholder(tf.int32, [self.batch_size, self.n_lstm_steps])

mask = tf.placeholder(tf.float32, [self.batch_size, self.n_lstm_steps])


# getting an initial LSTM embedding from our image_imbedding

image_embedding = tf.matmul(img, self.img_embedding) + self.img_embedding_bias


# setting initial state of our LSTM

state = self.lstm.zero_state(self.batch_size, dtype=tf.float32)
```

```
total_ loss = 0.0

with tf.variable_scope("RNN"):

for i in range(self.n_lstm_steps):

if i > 0:

#if this isn't the first iteration of our LSTM we need to get the word_embedding co
rresponding

#ccaption (i-1)th word in our caption

with tf.device("/cpu:0"):

current_embedding = tf.nn.embedding_lookup(self.word_embedding, caption_plac
eholder[:,i-1]) + self.embedding_bias

else:

#if this is the first iteration of our LSTM we utilize the embedded image as our inp
ut

current_embedding = image_embedding

if i > 0:

# allows us to reuse the LSTM tensor variable on each iteration

tf.get_variable_scope().reuse_variables()

out, state = self.lstm(current_embedding, state)

print (out,self.word_encoding,self.word_encoding_bias)



if i > 0:

#get the one-hot representation of the next word in our caption

labels = tf.expand_dims(caption_placeholder[:, i], 1)

ix_range=tf.range(0, self.batch_size, 1)

ixs = tf.expand_dims(ix_range, 1)

concat = tf.concat([ixs, labels],1)

onehot = tf.sparse_to_dense(

concat, tf.stack([self.batch_size, self.n_words]), 1.0, 0.0)

#perform a softmax classification to generate the next word in the caption
```

最新 (/)

团队号 (/channel/index?type=263&name=team)

移动开发 (/channel/index?type=272&name=mobile)

Java (/channel/index?type=257&name=java)

Python (/channel/index?type=268&name=python)

个人号 (/channel/index?type=271&name=person)

前端 (/channel/index?type=264&name=web)

后端 (/channel/index?type=269&name=server)

测试 (/channel/index?type=266&name=test)

更多 (/channel/index?type=270&name=other)

```
logit = tf.matmul(out, self.word_encoding) + self.word_encoding_bias

xentropy = tf.nn.softmax_cross_entropy_with_logits(logits=logit, labels=onehot)

xentropy = xentropy * mask[:,i]

loss = tf.reduce_sum(xentropy)

total_loss += loss

total_loss = total_loss / tf.reduce_sum(mask[:,1:])

return total_loss, img, caption_placeholder, mask
```

使用推断来生成字幕

完成训练后，我们就获得了一个在给定图片和所有的前置词汇的前提下，可以给出字幕里下一个词概率的模型。那么我们怎么用这个模型来生成字幕？

最简单的方法就是把一张图片作为输入，循环输出下一个概率最大的词，由此生成一个字幕。

```
def build_generator(self, maxlen, batchsize=1):

#same setup as `build_model` function

img = tf.placeholder(tf.float32, [self.batch_size, self.dim_in])

image_embedding = tf.matmul(img, self.img_embedding) + self.img_embedding_bias

state = self.lstm.zero_state(batchsize,dtype=tf.float32)

#declare list to hold the words of our generated captions

all_words = []

print (state,image_embedding,img)

with tf.variable_scope("RNN"):

# in the first iteration we have no previous word, so we directly pass in the image embedding

# and set the `previous_word` to the embedding of the start token ([0]) for the future iterations

output, state = self.lstm(image_embedding, state)

previous_word = tf.nn.embedding_lookup(self.word_embedding, [0]) + self.embed
```

最新 (/)

团队号 (/channel/index?type=263&name=team)

移动开发 (/channel/index?type=272&name=mobile)

Java (/channel/index?type=257&name=java)

Python (/channel/index?type=268&name=python)

个人号 (/channel/index?type=271&name=person)

前端 (/channel/index?type=264&name=web)

后端 (/channel/index?type=269&name=server)

测试 (/channel/index?type=266&name=test)

更多 (/channel/index?type=270&name=other)

```
    for i in range(maxlen):
        tf.get_variable_scope().reuse_variables()
        out, state = self.lstm(previous_word, state)
        # get a one-hot word encoding from the output of the LSTM
        logit = tf.matmul(out, self.word_encoding) + self.word_encoding_bias
        best_word = tf.argmax(logit, 1)
        with tf.device("/cpu:0"):
            # get the embedding of the best_word to use as input to the next iteration of our L
STM
            previous_word = tf.nn.embedding_lookup(self.word_embedding, best_word)
            previous_word += self.embedding_bias
        all_words.append(best_word)

    return img, all_words
```

很多情况下，这个方法都能用。但是“贪婪”地使用下一个概率最大的词可能并不能产生总体上最合适的字幕。

规避这个问题的一个可行的方法就是“束搜索（Beam Search）”。这个算法通过递归的方法在最多长度为t的句子里寻找k个最好的候选者来生成长度为t+1的句子，且每次循环都仅仅保留最好的那k个结果。这样就可以去探索一个更大的最佳的字幕空间，同时还能让推断在可控的计算规模内。在下图的例子里，算法维护了在每个垂直时间步骤里的一系列k=2的候选句子（用粗体字表示）。

最新 (/)

团队号 (/channel/index?type=263&name=team)

移动开发 (/channel/index?type=272&name=mobile)

Java (/channel/index?type=257&name=java)

Python (/channel/index?type=268&name=python)

个人号 (/channel/index?type=271&name=person)

前端 (/channel/index?type=264&name=web)

后端 (/channel/index?type=269&name=server)

测试 (/channel/index?type=266&name=test)

更多 (/channel/index?type=270&name=other)

图4 来源：*Daniel Ricciardelli*

局限和讨论

神经图片字幕生成器给出了一个有用的框架，能学习从图片到人能理解的图片字幕间的映射。通过训练大量的图片-字幕对，这个模型学会提取从视觉特征到相关语义信息的关系。

但是，对于一个静态图片，我们的字幕生成器是关注图片里有利于分类的特征，而这并不一定是有利于字幕生成的特征。为了改进每个特征里与字幕相关的信息量，我们可以把这个图片向量模型（这个用来编码特征的VGG-16模型）作为整个字幕生成模型的一部分。这就可以让我们能更精细地调优图片编码器来更好地承担字幕生成的角色。

而且，如果我们去仔细地观察生成的字幕，就会发现它们其实相当的模糊与普通化。用下面这个图片-字幕对为例：

最新 (/)

团队号 (/channel/index?type=263&name=team)

移动开发 (/channel/index?type=272&name=mobile)

Java (/channel/index?type=257&name=java)

Python (/channel/index?type=268&name=python)

个人号 (/channel/index?type=271&name=person)

前端 (/channel/index?type=264&name=web)

后端 (/channel/index?type=269&name=server)

测试 (/channel/index?type=266&name=test)

更多 (/channel/index?type=270&name=other)

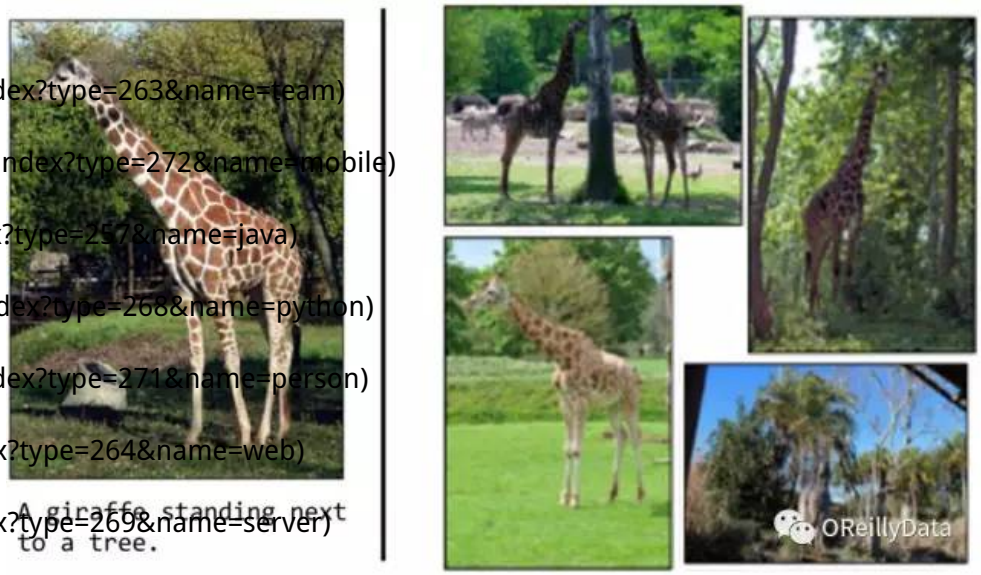

图9：来源：Raul Puri，图片来自MS COCO数据集

这个图片当然是“长颈鹿站立在树旁边”。但是如果看看其他的图片，我们就可能注意到它会对于任何有长颈鹿的图片都生成“长颈鹿站立在树旁边”，因为在训练集里，长颈鹿通常都出现在树的附近。

下一步工作

首先，如果你想改进这里介绍的模型，请阅读以下谷歌的开源“Show and Tell网络”。它可以用Inception-v3图片向量和MS COCO数据集来训练。

目前最前沿的图片字幕模型包含了一个视觉注意力机制。可以让模型在生产字幕时，发现图片里的引起兴趣的区域来有选择地关注图片内容。

同时，如果你有对最前沿的字幕生成器的实现感兴趣，请阅读这个论文《展示、关注和说出：使用视觉注意力的神经图片字幕生成》

注意：别忘了访问GitHub上与这篇文章对应的Python代码和iPython notebook。

这篇博文是O’Reilly和TensorFlow的合作产物。请阅读我们的编辑独立声明。

作者介绍：

Raul Puri

Paul Puri是加州大学伯克利分校CO 2017届毕业的本科生研究人员。Raul已经对多个领域的研究项目做出了贡献，包括但不限于：机器人和自动化、计算机视觉、医疗成图、生物记忆设备等。不过所有这些研究工作都专注于机器学习和机器学习系统在安全、自主驾驶、自然语言处理、计算机视觉和机器人里面的应用。Raul也非常热情于通过教授应用机器学习概念课程来回馈社会。他还是多门伯克利分校机器学习课程的

最新 (/)

团队号 (/channel/index?type=263&name=team)

移动开发 (/channel/index?type=272&name=mobile)

Java (/channel/index?type=265&name=java)

Python (/channel/index?type=268&name=python)

个人号 (/channel/index?type=271&name=person)

前端 (/channel/index?type=264&name=web)

后端 (/channel/index?type=269&name=server)

测试 (/channel/index?type=266&name=test)

更多 (/channel/index?type=270&name=other)

助教和讲师。

Daniel Bricciardelli

Dan Bricciardelli是加州大学伯克利分校的一名本科生研究员。他的研究方向包括用于金融和工业的自然语言处理、计算机视觉、深度主动学习和自动知识发现。Dan在伯克利机器学习组织主席，热心于让机器学习能为技术及非技术学生和专业人员所接触。

This article originally appeared in English: "Caption this, with TensorFlow"

最新 (/)

阅读 4 0

团队号 (/channel/index?type=263&name=team)

移动开发 (/channel/index?type=272&name=mobile)

Java (/channel/index?type=257&name=java)

Python (/channel/index?type=268&name=python)

个人号 (/channel/index?type=271&name=person)

前端 (/channel/index?type=264&name=web)

后端 (/channel/index?type=269&name=server)

测试 (/channel/index?type=266&name=test)

更多 (/channel/index?type=270&name=other)

软件定义世界（SDX）更多文章

干货：揭秘信息可视化图表的设计方法 (/html/626/201705/2649982379/1.html)

【PPT】大数据在保险行业的应用实践 (/html/626/201705/2649982376/1.html)

又读懂机器学习，大数据、自然语言处理、算法全有了..... (/html/626/201705/2649982374/1.html)

麦肯锡万字报告对比中美AI竞争力：学术、产业生态、算法、数据、计算力 (/html/626/201704/2649982372/1.html)

数据开发常用的几种数据预处理和数据整理方法 (/html/626/201704/2649982370/1.html)

猜您喜欢

基于用户的协同过滤算法 (/html/178/201604/2650106997/1.html)

大型网站架构体系的演变 (/html/331/201704/2651688432/1.html)

15张PPT看清：为什么同样很忙，但你始终成不了大牛？ (/html/202/201701/2649715459/1.html)

CDH大数据平台搭建 (/html/344/201606/2652453276/1.html)

震惊小伙伴的单行Python代码 (/html/160/201607/2649639202/1.html)