深度学习第四课：让机器读懂视频

大家好，我是来自PaddlePaddle团队的工程师。深度学习第三课中，曹莹介绍了如何让机器完成简单的写作任务，即利用循环神经网络来完成“文本到文本”的生成任务。在深度学习第四课中，我们会进一步介绍如何让机器读懂视频并用文本进行描述，即更复杂的“视频到文本”的生成任务。

PaddlePaddle最早在百度内部使用的时候，就做过非常多的自然语言处理任务，而自然语言处理任务几乎是RNN的天下，所以PaddlePaddle对RNN支持得非常好。根据深度学习第三课中总结的优势（灵活的序列输入、RNN算的快、可高度定制的RNN单元），本课会进一步展开介绍如下内容：

1. 首先，介绍“视频到文本”的应用场景，给大家一个直观的感受。
2. 其次，依次讲述“文本到文本”、“视频到一句话”、“视频到一段话”的模型优化过程。
3. 最后，介绍PaddlePaddle中的双层序列，并帮助大家看懂双层RNN的配置。

一、应用场景

伴随着信息时代的到来，海量信息在全球被采集、传输和应用。尤其是数码照相机、数码摄像机等数字化产品的出现，让图像和视频进一步成为人们喜闻乐见的交流方式。但视频信息存在数据量大、抽象程度低的特点，并且常常由于缺乏有效的技术导致不能及时处理而浪费。因此，如何让机器读懂视频，是当今的研究热点问题。

让机器读懂视频，即机器能用一个或多个句子来描述视频内容，在视频检索（video retrieval）、视频字幕（video caption）、盲人导航（blind navigation）等领域有广泛的应用。下面举例介绍几个应用场景。

视频检索

高效的视频检索系统，必须具备视频自动检索功能。如在安防监控领域，虽然监控摄像头已经遍布大街小巷，为大多数案件留下了影像资料，给警方破案带来了很大的便利。但是，有了相关视频不等于就找到了目标信息，查找视频、分析视频的工作常常会耗用警方大量的时间和人力。在破案过程中时间是关键，为了争取快一分钟找到线索，公司和学术界都推出了很多高效智能的视频检索软件。

如图1展示在电视剧《生活大爆炸》中检索主人公Sheldon Cooper的示例[1]。从图中可以看出，所有包含该主人公的视频片段都可以被检索到。

图1. 电视剧《生活大爆炸》的视频检索

视频字幕

视频字幕分为两种：

1. 对含有音频的视频，利用语音识别和机器翻译技术，自动生成字幕。Youtube率先应用了这种技术，随便打开一个英文的、没有字幕的视频（如下所示）。点击右下角的字幕按钮，直接就生成了英文字幕。再点击设置中的自动翻译，还有几十种国家的语言可以选择。我们选择简体中文，发现翻译得还挺不错。

2. 对不含音频的视频，利用图像识别和机器翻译技术，自动生成字幕。图2展示了在TACos视频数据集上生成出的多句连续的话。从图中可以看出，这几句话已经能比较准确地描述出这个视频的内容。

一个人在剥水果。

一个人将水果放进碗里。

一个人在切桔子。

一个人将切片放进盘子里。

一个人冲洗水槽里的盘子。

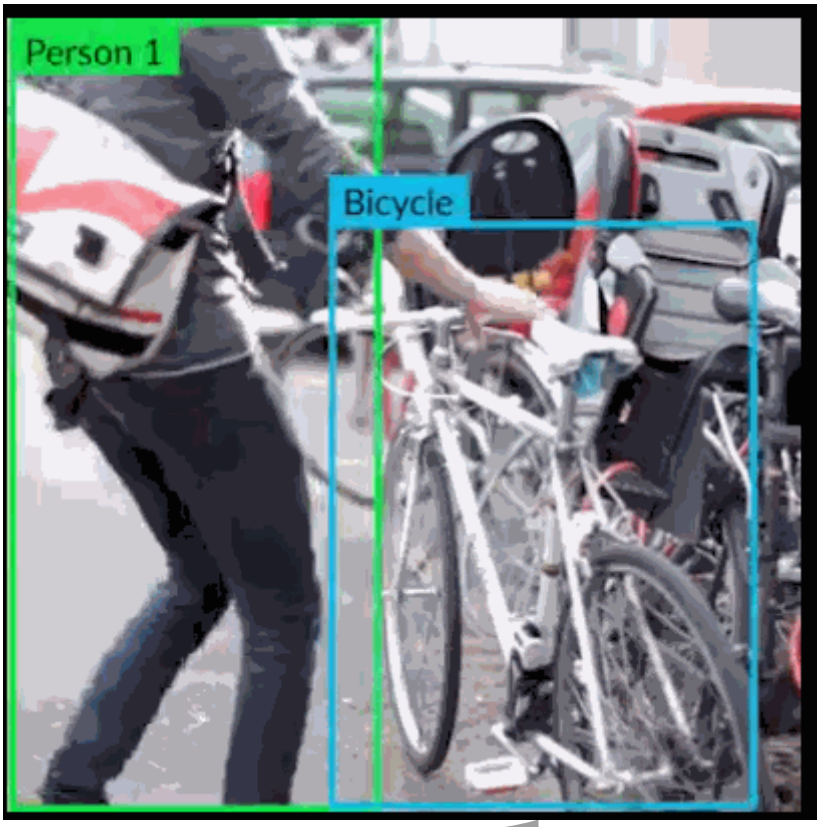
图2. TACos视频数据集上生成的多句连续的话

盲人导航

许多武侠小说、科幻或超级英雄股市中，盲人常常有超级听觉，甚至能“听声辨物”，但这只是大家的想象。盲人最大的障碍之一，就是不知道身边有什么事物。但在深度学习技术的发展下，盲人如今可以用“手机辨物”，从而大大改善生活[2]。

Aipoly Vision手机APP就是这样的一款应用。它的使用方式相当简单，只要对着想“看”的方向拍摄，手机就会自动说出“看”的方向有什么东西。下面视频展示了手机“看”到了“一

个人在骑车”。



二、模型概览

近年来，受到自然语言处理中“文本到文本”模型的启发，“视频到文本”的生成任务取得了一系列成果。这些成果都专注于为一个视频生成一条句子，但对于包含丰富语义的视频，一句话显然不能描述完整。因此，文章[3]使用双层RNN为视频生成一段话，其中每句话的生成都会考虑到前面句子的语义上下文。

下面首先回顾深度学习第三课中的“文本到文本”的模型，再讲述从“视频到一句话”至“视频到一段话”的模型优化过程。

文本到文本

“文本到文本”的模型[4][5]分为标准的编码和解码阶段，如图3所示。

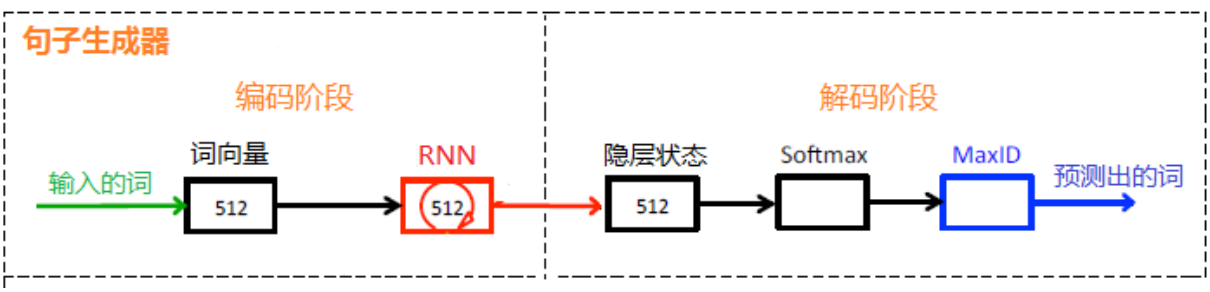


图3. 文本到文本的模型示例

编码阶段包含三步：

- 第一步：将输入句子的每个词用one-hot vector表示，图3中绿色的“输入的词”已经是one-hot vector了。
- 第二步：由于one-hot vector难以刻画词与词之间的语义相关性，因此用一个映射矩阵将其映射为一个固定维度的稠密向量，即词向量。词向量的维度通常为128的倍数。
- 第三步：用RNN来编码源语言词向量序列。其中每个时间步的隐层状态是根据上一个时间步的隐层状态，和当前时间步的词向量输入，然后通过一个非线性激活函数来计算得到的。

解码阶段也包含三步：

- 第一步：计算解码RNN下一个时间步的隐层状态。具体来说，是根据源语言词向量序列的编码信息、真实目标语言序列当前时间步的词，和当前时间步的隐层状态，然后通过一个非线性激活函数计算得到的。
- 第二步：根据隐层状态，用softmax归一化得到词概率。
- 第三步：根据词概率来计算代价（训练过程）或采样出单词（生成过程，如图3中蓝色部分）。

下面的视频介绍了如何用“文本到文本”的模型进行中英翻译的过程。

更详细的“文本到文本”的模型介绍，可参考深度学习教程中的机器翻译章节。

视频到文本

受到“文本到文本”模型的启发，在“视频到文本”的生成任务中：将视频的图像序列看作源语言，将对应的文本序列看作目标语言。给定从视频帧中提取的一系列深度卷积特征（如VggNet[6]、C3D[7]等），编码阶段，视频的编码向量可以是特征序列的平均池化表示[8]、使用注意力机制后的平均池化表示[9]、或特征序列的最后一个向量表示[10]等；解码阶段，则从这些编码向量中解码出不定长的文本句子。

图4展示了“视频到文本”的一个简单配置示例[8]。编码阶段，用卷积神经网络从每个视频帧中提取出特征，再对所有特征做平均池化（mean pooling）操作得到整个视频的编码向量。解码阶段，用两层LSTM网络从视频的编码向量（和上一个生成的词）中生成出完整的文本句子。

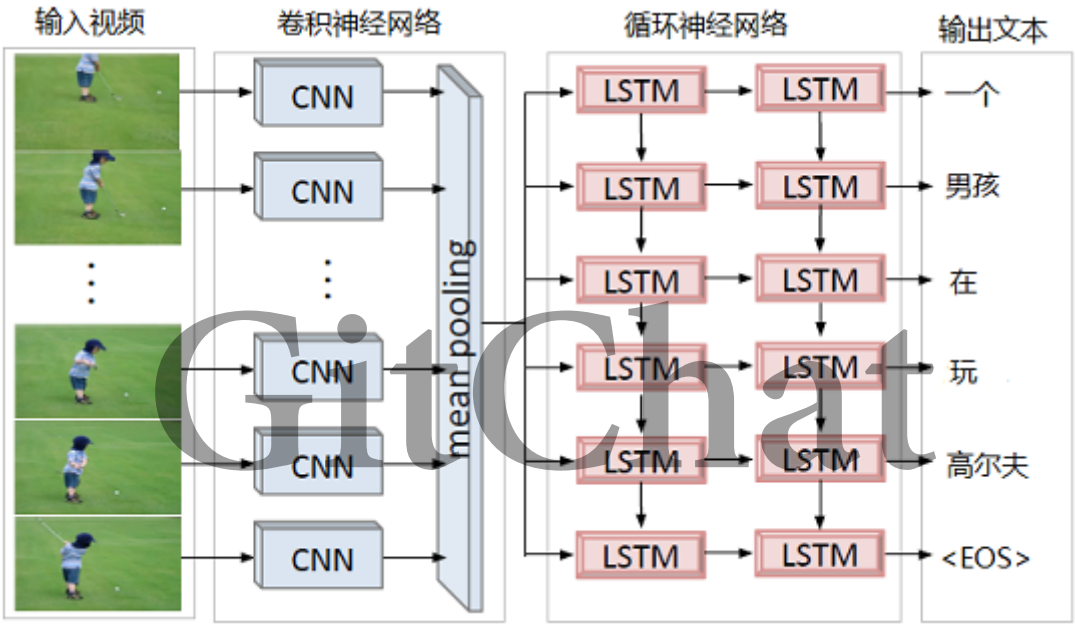


图4. 视频到文本的模型示例

视频到一段话

图4的场景比较简单，可以用一句话“一个男孩在玩高尔夫”来描述；但对图2的场景，很难用一句话来描述“剥、切、洗”这几个动作。因此，文章[3]尝试使用双层RNN为视频生成一段话（如图5所示）。

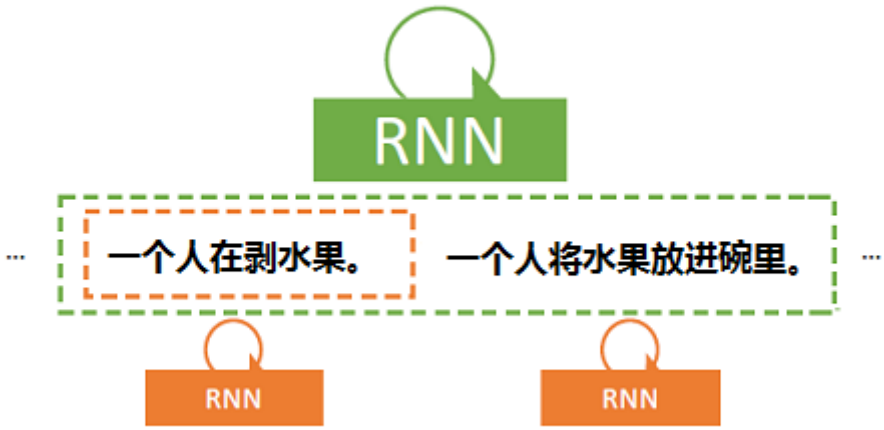


图5. 段落中的话是有语义相关性的

双层RNN，即RNN之间有一层嵌套关系。内层RNN是句子生成器，即将视频中的某一小段（连续的几帧或几十帧）生成一个句子。外层RNN是段落生成器，它考虑句子间的依赖关系，利用段落的历史信息来初始化下一个句子生成器。循环进行，从而生成出一段连续的句子。

图6展示了文章[3]中“视频到一段话”的配置示例。它在“视频到一句话”模型上，结合“文本到文本”模型，变成“（视频+文本）到文本”模型。而引入“文本到文本”模型的原因，是为了利用段落生成器提供的历史信息，来更好地生成下面的句子。

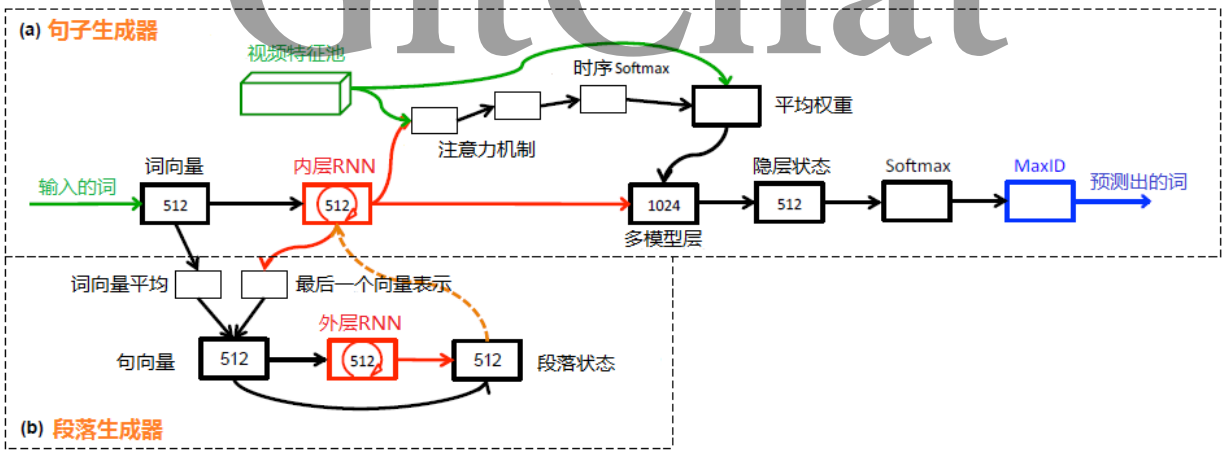


图6. 视频到一段话的模型示例

句子生成器：包含两个输入（绿色表示），分别是文本输入（一句话）和视频输入（视频特征池）。简单的来看：

- 编码阶段，将上一条文本词向量序列通过内层RNN编码后的信息，和当前视频特征序列的平均权重编码信息，用一个多模型层融合在一起。
- 解码阶段，从融合后的编码向量中解码出不定长的文本句子。

段落生成器：作用在句子生成器的编码阶段：

- 首先，对上一条文本词向量序列，将其平均池化后的向量和通过内层RNN编码后的最后一个向量，拼接得到句向量。

- 其次，用外层RNN来编码句向量序列，外层RNN中每个时间步的隐层状态用于初始化下一个内层RNN，即实现了用段落的历史信息来初始化下一个句子生成器的功能。

据我们所知，文章[3]是第一篇利用双层RNN来完成“视频到一段话”的工作，感兴趣的读者可以阅读原文。

三、PaddlePaddle中的RNN

PaddlePaddle目前可支持双层RNN。本节首先介绍双层序列的概念；其次介绍PaddlePaddle中，高度可定制的RNN单元`recurrent_group`；最后使用`recurrent_group`来配置一个简单的双层RNN。

注意：文章[3]的配置也可以在PaddlePaddle平台下运行，但这个配置相对复杂，在本课中就不进行代码讲解，后面会更新在PaddlePaddle/models项目中。

双层序列

在自然语言处理任务中，序列是一种常见的数据类型：

- 一个独立的词语，可以看作是一个非序列输入，或者，我们称之为一个0层的序列。

- 由词语构成的句子，是一个单层序列。

- 若干个句子构成一个段落，是一个双层的序列。双层序列是一个嵌套的序列，它的每一个元素，又是一个单层的序列。这是一种非常灵活的数据组织方式，帮助我们构造一些复杂的输入信息。

高度可定制的RNN单元

`recurrent_group`是PaddlePaddle支持的一种任意复杂的RNN单元。用户只需定义RNN在一个时间步内完成的计算，PaddlePaddle负责完成信息和梯度在时间序列上的传播。

一个简单调用如下：

```
recurrent_group(step, input, reverse)
```

- `step`：核心调用函数，它定义了RNN在一个时间步内完成的计算。可以通过自由组合PaddlePaddle支持的各种layer，来实现用户自定义的功能。

- `input` ：输入层，支持三种不同的输入类型。
    - 数据输入： `recurrent_group` 会对数据输入序列进行时间步上的拆分，然后交给 `step` 函数。双层序列会被拆分为时间步上的单层序列，单层序列会被拆分为时间步上的非序列。这个拆分过程对用户是透明的。
    - 其他两种输入类型请参阅文档。
- `reverse` ：是否以逆序处理输入序列。

双层RNN配置示例

本小节用一对效果完全相同的、分别使用单双层RNN作为网络配置的模型，来展示如何配置一个简单的双层RNN。

输入序列：

- 对于单层RNN，输入数据为单层序列，如 `[4, 5, 2, 0, 9, 8, 1, 4]` 。
- 对于双层RNN，输入数据为双层序列，可在上述单层序列里面，任意对一些连续数据进行组合，如 `[[ [4, 5, 2], [0, 9], [8, 1, 4]]]` 。

单层RNN的网络配置：

- 一个非常简单的全连接RNN。
- 每一个时间步，当前的输入y和上一个时间步的输出rnn_state做了一个全连接。

```
def step(y):
 mem = paddle.layer.memory(name="rnn_state",
size=hidden_dim)
 out = paddle.layer.fc(input=[y, mem],
          size=hidden_dim,
          act=paddle.activation.Tanh(),
          name="rnn_state")
 return out

out = paddle.layer.recurrent_group(name="rnn", step=step,
input=emb)
```

双层RNN的网络配置：

- 内层 `inner_step` ：和单层RNN的几乎一样，除了boot_layer=outer_mem，表示将外层的outer_mem作为内层memory的初始状态。
- 外层 `outer_step` 中：outer_mem记忆了每个单层序列的最后一个向量，即整个双层RNN是将前一个子句的最后一个向量，作为下一个子句memory的初始状态。

- 从输入序列上看：单双层序列的句子是一样的，只是双层序列（使用SubsequenceInput标记）对其又做了划分。因此双层RNN的网络配置中，必须将前一个单层序列的最后一个元素，作为boot_layer传给下一个子句的memory，才能保证和单层RNN的配置中“每个时间步都用了上一个时间步的输出结果”一致。

```
def outer_step(x):
 outer_mem = paddle.layer.memory(name="outer_rnn_state",
size=hidden_dim)
 def inner_step(y):
     inner_mem = paddle.layer.memory(name="inner_rnn_state",
                       size=hidden_dim,
                       boot_layer=outer_mem)
     out = paddle.layer.fc(input=[y, inner_mem],
                           size=hidden_dim,
                           act=paddle.activation.Tanh(),
                           name="inner_rnn_state")
     return out

 inner_rnn_output = paddle.layer.recurrent_group(
     step=inner_step,
     name="inner",
     input=x)
 last = paddle.layer.last_seq(input=inner_rnn_output,
name="outer_rnn_state")

 return inner_rnn_output

out = paddle.layer.recurrent_group(
 name="outer",
 step=outer_step,
 input=SubsequenceInput(emb))
```


小结

人类能够观看一段简短的视频并轻松描绘出视频内容，甚至预测出后续事件的发生。而机器，虽然目前能用一个或多个句子来描述视频内容，但拥有“预测”这样的能力依然是可望不可及。因此，距离真正的“让机器读懂视频”还有很长的路要走。

最后，欢迎大家能关注我们的微信公众号PaddlePaddle，我们会不定期地推出精彩的原创文章、举办丰富的线上/线下分享活动，敬请期待。

参考文献

1. Y. Li, R Wang, Z Huang, et al. Face video retrieval with image query via hashing across euclidean space and riemannian manifold[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015: 4758-4767.

2. http://technews.cn/2016/04/17/app-helps-blind-people-see/.

3. H. Yu, J. Wang, Z. Huang, et al. Video paragraph captioning using hierarchical recurrent neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016.

4. K. Cho, B. Van Merriënboer , C. Gulcehre , et al. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2014: 1724-1734.

5. D. Bahdanau, K. Cho, Y. Bengio. Neural machine translation by jointly learning to align and translate. In Proceedings of ICLR 2015, 2015.

6. K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In International Conference on Learning Representations, 2014.

7. D. Tran, L. D. Bourdev, R. Fergus, L. Torresani, and M. Paluri. C3D: generic features for video analysis. In Proceedings of the IEEE International Conference on Computer Vision, 2015.

8. S. Venugopalan, H. Xu, J. Donahue, M. Rohrbach, R. J. Mooney, and K. Saenko. Translating videos to natural language using deep recurrent neural networks. In Proceedings of the North American Chapter of the Association for Computational Linguistics, 2015.

9. L. Yao, A. Torabi, K. Cho, N. Ballas, C. Pal, H. Larochelle, and A. Courville. Describing videos by exploiting temporal structure. In Proceedings of the IEEE International Conference on Computer Vision, pages 4507–4515, 2015.

10. J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015.