深度学习与神经网络第三次课程项目

王逸群 19307110397

2022.5

1 数据集介绍

本项目使用 MSCOCO(Microsoft Common Objects in Context) 数据集的子集，并使用 NOC(Novel Object Captioning) 分割方法分割数据集。数据集中的每张图片都附带 5 句人工标注的句子。巴士、瓶子、沙发、微波炉、披萨、球拍、行李箱、斑马等 8 件物品不出现在训练集中，但会出现在验证集和测试集中。

2 模型学习

2.1 Deep Compositional Captioning

Lisa Anne 的 DCC(Deep Compositional Captioning) 模型架构图如图1所示。可以看到，模型由三个部分组成。

首先，词汇分类器的主体是一个卷积神经网络。其中，分类的目标概念是通过词性标注得到的在图像文本对中频繁出现的形容词、动词和名词，以及一些图像文本对以外的概念；而分类器的输出是一个特征，每个元素表示给定概念在图像中出现的概率。具体

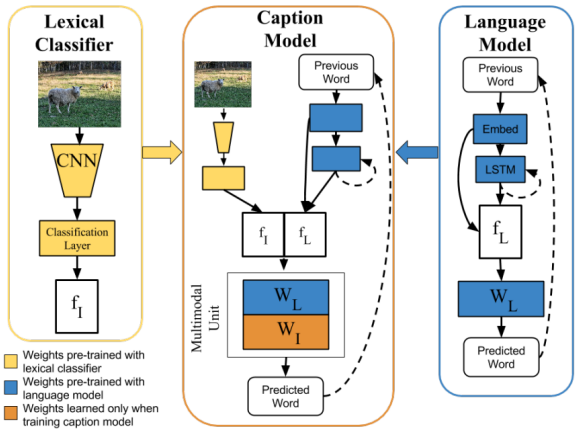

图 1: DCC 模型架构图

操作时，词汇分类器先在 ILSVRC(ImageNet Large Scale Visual Recognition Challenge) 训练集上预训练，再进行精调。

接着，语言模型由词汇编码、长短期记忆网络、词汇预测层组成。其中，生成的特征由词汇编码和长短期记忆网络的输出连接而成。

在词汇分类器和语言模型的基础上，图像说明模型由两者组合而成。其中，词汇分类器的输出特征与语言模型的输出特征相互连接，经过多模态单元即一个线性层和一个 Softmax 层后，输出每一个单词是下一个单词的概率分布。

模型训练完毕后，还要进行知识迁移。

2.2 Neural Baby Talk

本项目使用 Jiasen Lu 的 NBT(Neural Baby Talk) 模型。架构图如图2所示。

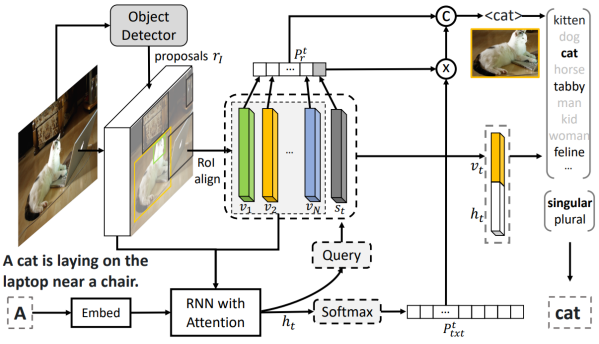

图 2: NBT 模型架构图

该模型将单词分为两类进行处理，一类是可视单词，基于特定的图片区域；另一类是文本单词，是图像说明中的其它单词。针对前者，还需要对其复数形式和细粒度形式进行判断。

3 评价指标

3.1 F1

将输出句子与标注句子中的单词分为四类：

名称	字母	含义
真正例	TP	在输出句子和标注句子中都出现
假正例	FP	在输出句子中出现但在标注句子中不出现
假负例	FN	在输出句子中不出现但在标注句子中出现
真负例	TP	在输出句子和标注句子中都不出现

表 1: 输出句子与标注句子中的四类单词

定义精度、召回率、F1：

$$P = \frac{TP}{TP + FP}$$
$$R = \frac{TP}{TP + FN}$$
$$F_1 = \frac{2}{\frac{1}{P} + \frac{1}{R}}$$

3.2 CIDEr

CIDEr 可以计算输出句子和一组标注句子的匹配程度。

首先，输出句子和标注句子的所有单词都被简化为词干，进而输出句子和标注句子可以由一元至 N 元词汇表示。默认情况下，$N = 4$。

接着，计算每个词元的 TF-IDF 权重。具体而言，记 I 表示待评判的输出句子集合。对于第 i 句输出句子 c_i，有一组标注句子 $S_i = \{s_{ij}\}$。进而对于第 k 个词元，记 $h_k(c_i)$ 表示其在输出句子 c_i 中出现的次数，$h_k(s_{ij})$ 表示其在标注句子 s_{ij} 中出现的次数，于是 TF-IDF 权重为：

$$g_k(c_i) = \frac{h_k(c_i)}{\sum_l h_l(c_i)} \log\left(\frac{|I|}{\sum_p \min(1, h_k(c_p))}\right)$$
$$g_k(s_{ij}) = \frac{h_k(s_{ij})}{\sum_l h_l(s_{ij})} \log\left(\frac{|I|}{\sum_p \min(1, \sum_q h_k(s_{pq}))}\right)$$

在此基础上，再对于 n 元词汇，计算输出句子和标注句子之间的 TF-IDF 的平均余弦相似度；最后，将这些余弦相似度加权平均得到 CIDEr；

$$CIDEr_n(c_i, S_i) = \frac{1}{|S_i|} \sum_j \frac{g^n(c_i) \cdot g^n(s_{ij})}{||g^n(c_i)||||g^n(s_{ij})||}$$
$$CIDEr(c_i, S_i) = \sum_n w_n CIDEr_n(c_i, S_i)$$

3.3 METEOR

对于给定的输出句子和标注句子，首先进行单词之间的对齐，使得每个单词映射到另一个句子中的至多一个单词。对齐包括若干个步骤，每个步骤包括两个阶段。

在第一阶段，若干外部模块根据不同的标准建立起所有可能的映射。比如，`exact`模块当两个单词完全相同时建立映射，`porter stem`模块当两个单词词干相同时建立映射，`WN synonymy`模块当两个单词是同义词时建立映射。默认情况下，按顺序使用这三种模块。在第二阶段，第一阶段得到的所有可能的映射中的一个子集成为真正的映射，该子集尽可能大，且尽可能产生较少的交叉。

得到单词之间的映射后，使用与3.1节相似的方法得到精度和召回率，并计算：

$$F_{mean} = \frac{10}{\frac{1}{P} + \frac{9}{R}}$$

由于以上仅进行了单词之间的对齐，接下来需要再计算一个惩罚项，体现词组甚至句子之间的对齐。具体而言，将输出句子中的单词聚合成尽可能少的块，使得每一块中相邻的单词都映射到标注句子中相邻的单词。这样一来，块越少，对齐的词组越长。进而得到惩罚项和 MEREOR：

$$Penalty = \frac{1}{2} \times (\frac{\#chunks}{\#unigrams\_matched})^3$$
$$METEOR = F_{mean} \times (1-Penalty)$$

3.4 SPICE

SPICE 也可以计算输出句子和一组标注句子的匹配程度。

首先，对于每一句话，计算其句法树，如图3上方所示；并由句法树导出场景图，如图3右侧所示。其中，实体、属性和联系分别被标注，图中显示为红色、绿色和蓝色。

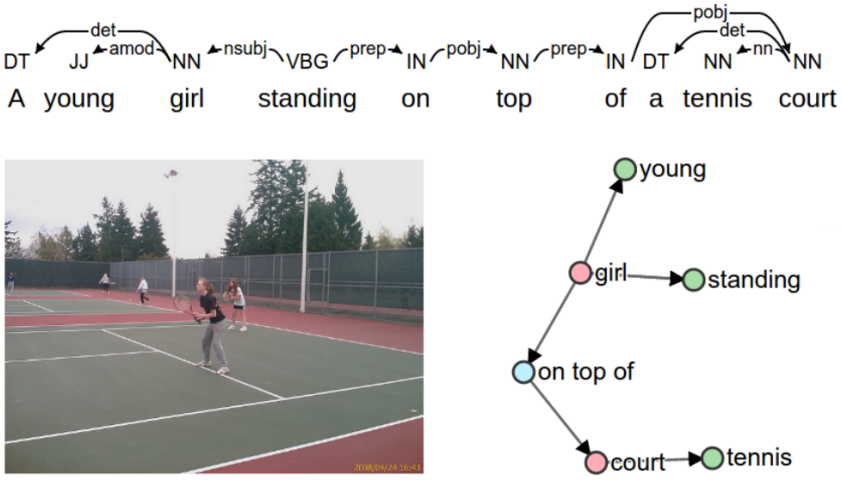

图 3: 句法树与场景图说明

具体而言，给定实体集合 C、属性集合 A、联系集合 R，对于句子 c，记 $O(c) \subseteq C$ 是 c 中出现的实体集合，$K(c) \subseteq O(c) \times A$ 是 c 中出现的实体属性集合，$E(c) \subseteq O(c) \times R \times O(c)$ 是 c 中出现的实体联系集合，则场景图为 $G(c) = \langle O(c), E(c), K(c) \rangle$。

记输出句子 c 的场景图为 $G(c)$，每句标注句子 s_j 的场景图为 $G(s_j)$。将 $G(s_j)$ 中同义实体的节点合并，得到标注句子集合 S 的场景图 $G(S)$，如图4所示。

最后，在场景图 $G(c)$ 和 $G(S)$ 的基础上，使用3.3中的对齐方法，计算精度、召回率、F1，其中 F1 即为 SPICE。

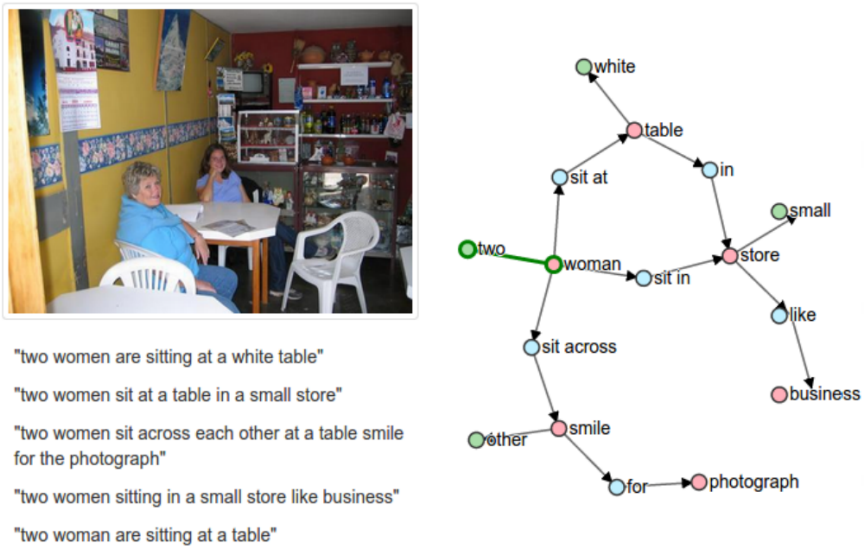

图 4: 场景图合并说明

4 实验过程中遇到的问题与解决

4.1 尝试安装 Caffe

- 运行`transfer.sh`时，报错找不到模块`caffe`，通过网络得知需要使用`make _caffe`命令编译 C++ 程序文件；编译时，又显示找不到 C 语言头文件`Python.h`，通过网络得知需要安装对应的文件，操作后并未解决问题；再通过网络得知 C 语言包含头文件的搜索路径，确保以下路径中都有`Python.h`：

```
1 /usr/include/
2 /usr/local/include/
3 /usr/local/miniconda3/include/
4 /usr/local/miniconda3/envs/DCC/include/
```

  仍然没有解决问题；最后通过网络得知，使用`gcc`命令编译 C++ 程序文件时，可以通过参数`-I`指定头文件的搜索路径，解决问题。

- 继续编译 C++ 程序文件，又显示找不到 C 语言头文件`numpy/arrayobject.h`，与上面的问题相似,通过网络得知需要运行`sudo apt-get install python-numpy`安装对应的文件，解决问题。

- 继续编译 C++ 程序文件，又显示找不到 C 语言头文件`caffe/proto/caffe.pb.h`，通过网络得知需要使用`protoc`从`src/caffe/proto/caffe.proto`生成`caffe.pb.h`和 `caffe.pb.cc`，解决问题。至此，C++ 程序文件成功编译。

- 继续运行`transfer.sh`时，依然报错找不到模块`caffe`，通过网络得到一种解决方法，修改`PYTHONPATH`，操作后并未解决问题。

- 在室友的指导下，转而使用Makefile编译，即使用make命令；编译时，报错无法创建链接，通过网络推测与服务器远程主页有关，于是将文件夹移动至服务器根目录，该报错信息不再出现。

- 继续使用make命令编译，显示找不到 C 语言头文件hdf5.h，通过网络指示先安装对应的文件，操作后并未解决问题，使用find命令查找路径，并在Makefile.config文件中修改INCLUDE_DIRS := $(PYTHON_INCLUDE) /usr/local/include路径，该报错信息不再出现。

- 继续使用make命令编译，显示找不到 C 语言头文件lmdb.h，通过网络指示安装对应的文件，与上面的问题相似，使用find命令查找路径，并在Makefile.config文件中修改INCLUDE_DIRS := $(PYTHON_INCLUDE) /usr/local/include路径，该报错信息不再出现。

- 继续使用make命令编译，显示不支持 GPU 架构，通过网络指示在Makefile.config文件中修改CUDA_ARCH，该报错信息不再出现。

- 继续使用make命令编译，显示如下报错信息：

```
1 /usr/bin/ld: cannot find -lhdf5_hl
2 /usr/bin/ld: cannot find -lhdf5
3 /usr/bin/ld: cannot find -lleveldb
4 /usr/bin/ld: cannot find -lsnappy
```

  针对前两项，通过网络指示在Makefile文件中修改LIBRARIES变量；针对后两项，通过网络指示，与最早的问题类似，安装对应的文件，该报错信息不再出现。

- 继续使用make命令编译，显示存在未定义的引用，通过网络指示在Makefile文件中修改LIBRARIES变量，该报错信息不再出现。至此，成功使用make命令编译。

- 继续运行transfer.sh时，依然报错找不到模块caffe，通过网络指示，运行命令make pycaffe，并修改PYTHONPATH，该报错信息不再出现。

- 继续运行transfer.sh时，又报错找不到模块skimage.io，通过网络得知需要运行sudo apt-get install python-skimage安装对应的文件，操作后并未解决问题；再通过网络得知可以使用dpkg命令获取skimage的安装路径；根据之前的经验，修改PYTHONPATH，该报错信息不再出现。

4.2 尝试运行 NBT 代码

- 运行docker命令时，显示找不到docker命令，通过网络指示操作，显示找不到yum命令；再通过网络指示安装yum命令，报错找不到模块rpm；再通过网络指示安装yum命令，操作后并未解决问题；再通过网络指示安装yum命令，成功；通过网络指示安装docker命令，成功，但是不能启动docker。

- 使用pip命令安装pycocotools模块时报错，通过网络指示，成功安装；使用pip命令安装stanfordcorenlp模块时报错，通过网络指示，用相似的方法成功安装。
- 运行main.py程序时，报错torch.utils.ffi模块已弃用，在室友的指导下，通过网络指示，更换torch的版本为0.4.0，torchvision的版本为0.2.2，torchtext的版本为0.2.3，tensorflow的版本为1.15.0，该报错信息不再出现。
- 运行main.py程序时，使用torchtext.vocab.GloVe下载glove.6B.zip，报错拒绝连接，通过网络指示，手动下载glove.6B.zip，并移动到.vector_cache文件夹。
- 运行main.py程序时，读取h5文件会频繁且随机地报错，相同的程序在室友的服务器上可以正常运行，因此之后在室友的帮助下在室友的服务器上完成程序运行。

5 数据预处理

- 使用 Karpathy 对 MSCOCO 数据集的预处理：将所有单词转化为小写，丢弃非字母数字的字符，保留在训练集中至少出现 5 次的单词。
- 使用 Faster-RCNN 对 MSCOCO 数据集进行目标检测。

6 模型预训练

- 以 ResNet-101 为初始架构，在 ImageNet 上预训练模型。

7 超参数设置

- beam size：训练时为 1，测试时为 3；
- 梯度截断：0.1；
- 学习率：初始值为 0.0005，之后每 3 个回合下降为 0.8 倍；
- 权重衰减：无；
- 优化器：参数为 0.9 和 0.999 的 Adam 优化器；
- 回合数：30；
- 批量大小：20。

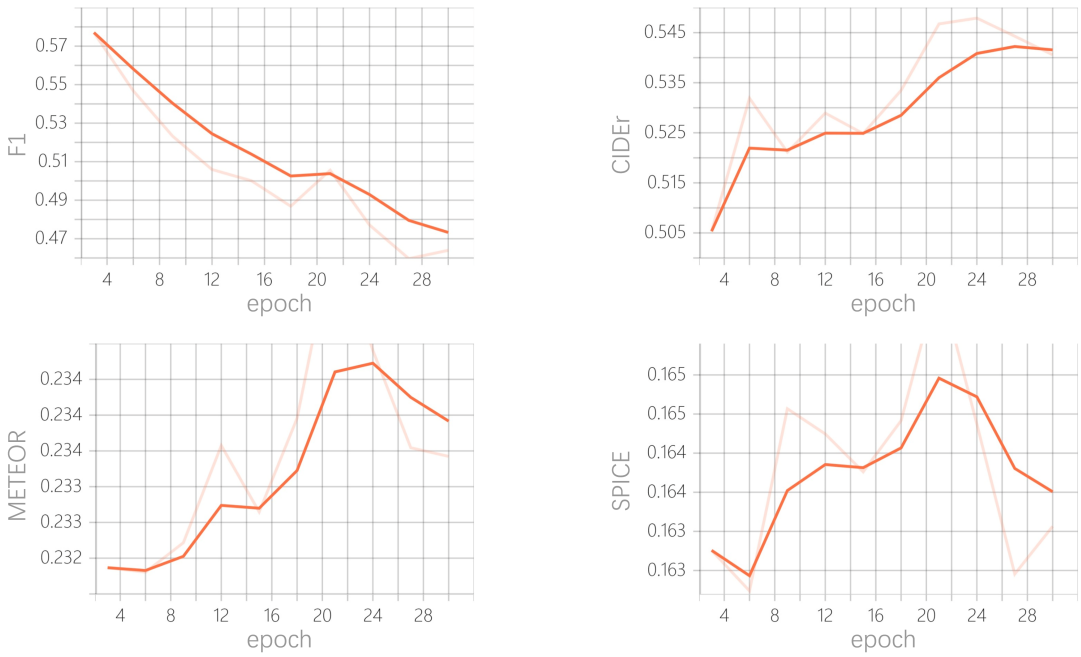

图 5: 实验结果

	巴士	瓶子	沙发	微波炉	披萨	球拍	行李箱	斑马	平均
F1	0.683	0.075	0.300	0.584	0.347	0.146	0.470	0.888	0.437
CIDEr	0.523	0.804	0.685	0.543	0.510	0.334	0.626	0.426	0.556
METEOR	0.223	0.222	0.259	0.244	0.209	0.242	0.213	0.231	0.231
SPICE	0.174	0.161	0.185	0.161	0.150	0.166	0.146	0.172	0.164

表 2: 实验结果

8 实验结果

实验结果如图5和表2所示。

可以看到，随着训练过程的推进，CIDEr、METEOR、SPICE 等评价指标都有上升的趋势，而 F1 有下降的趋势。前者能够比较好地评价句子的语义信息，较有参考价值；而后者则相反，参考价值较低。