好未来 Future Camp: NLP Week2 报告

吕晴

Project repo: https://github.com/veronica320/FutureCampWeek2

1 任务一：模型原理笔记

参考资料来源：维基百科，http://www.iro.umontreal.ca/bengioy/talks/DL-Tutorial-NIPS2015.pdf，http://www.wangqingbaidu.cn/article/dlp1516351259.html

1.1 CNN

- 一种前馈神经网络，可以响应一部分覆盖范围内的周围单元
- 由一个或多个卷积层和顶端的全连通层（对应经典的神经网络）组成，同时也包括关联权重和池化层（pooling layer）
- 卷积运算的目的是提取输入的不同特征，第一层卷积层可能只能提取一些低级的特征如边缘、线条和角等层级，更多层的网路能从低级特征中迭代提取更复杂的特征
- 相比较其他深度、前馈神经网络，卷积神经网络需要考量的参数更少，在图像和语音识别方面往往能够给出更好的结果

1.2 RNN

- 时间递归神经网络（又称循环神经网络），神经元间连接构成矩阵
- 单纯递归神经网络因为无法处理随着递归，权重指数级爆炸或消失的问题，难以捕捉长期时间关联，而结合不同的 LSTM 可以很好解决这个问题
- 时间递归神经网络可以描述动态时间行为，因为和前馈神经网络接受较特定结构的输入不同，RNN 将状态在自身网络中循环传递，因此可以接受更广泛的时间序列结构输入
- 用两个复发神经网络双向读取一个序列可以使人工智能获得“注意力”：简单的做法是将一个句子分别从两个方向编码为两个隐藏状态 $\vec{h}$，然后将其拼接在一起作为整体隐藏状态

1.3 语义匹配模型

- DSSM
  - 基于深度网络的语义模型，核心思想是把 query 以及 doc 映射到一个相同维度的语义空间，通过最大化两个空间的 cosine 相似度，达到信息检索的目的

    * 设计初衷是为了做信息检索（IR，应用到搜索引擎），但是后来逐渐被演化到不同数据源的语义空间映射问题，包括但是不仅仅包括信息检索、图文匹配、Caption、Sent2Vec
    * 致力于解决传统算法中无法解决的两个问题：数据规模 (在现有的 Web Data，数据规模是相当大的，传统方法对对大规模数据的扩展性很差，而且不能处理没有见过的单词), 及语义空间映射（根据不同的任务，引入具体场景的监督信息）
  - CDSSM
    * 基于卷积的 DSSM，由于原始的 DSSM 所有的单词相当于是单独来看，而且 Query 和 Doc 都是 Bag of words，句子之间的联系就被词袋模型给抹杀掉了，所以作者通过引入卷积操作，尽量地捕捉单词的相关性
    * 对单词的处理与原始的 DSSM 相同，是基于 Trigram 的 Hash 算法。然后通过卷积核，把一个 Doc(或 Query) 卷积出若干的 feature map，对应维度进行 MaxPooling，最后降维到 128 维的语义空间

2 任务二：语义匹配计算

2.1 任务描述

利用好未来提供的题库数据，进行两个题目之间的语义匹配计算。我们提供了小学数学题目的标注数据，包括题干 pair 的训练集和测试集，每对 pair 被标注了 1（正，相关）or 0（负，不相关），每一道题与多道题组成 pair（如：题目 a 与多道题被标注为 1，同时又与相同数目的其他题目被标注为 0）。

2.2 预处理

- 训练语料中，由于每道问题词数不相同，在作为神经网络输入时会出现向量长度不等的情况。在微软关于 DSSM 和 CDSSM 两篇论文中，作者分别用了 word hashing+bag of words/max pooling 的方法来解决这个问题；但中文中，由于不会出现维数爆炸问题，所以无需做 word hashing，因此需要用其他方法来解决长度不等的问题。这里我人为设置了一个句子的最大长度 maxlen 参数（default=20），对于不足的句子用 padding 补齐。
- 为了加快训练时加载词向量的速度，对下载好的预训练词向量（来源：https://github.com/Embedding/Chinese-Word-Vectors），我先做了过滤，只保留训练语料和测试语料中出现的词语。

2.3 Baseline 模型

2.3.1 模型结构

仿照微软 DSSM 的论文（Huang 2013），我构建了一个略有区别的 DSSM 模型作为 baseline，结构图如下：

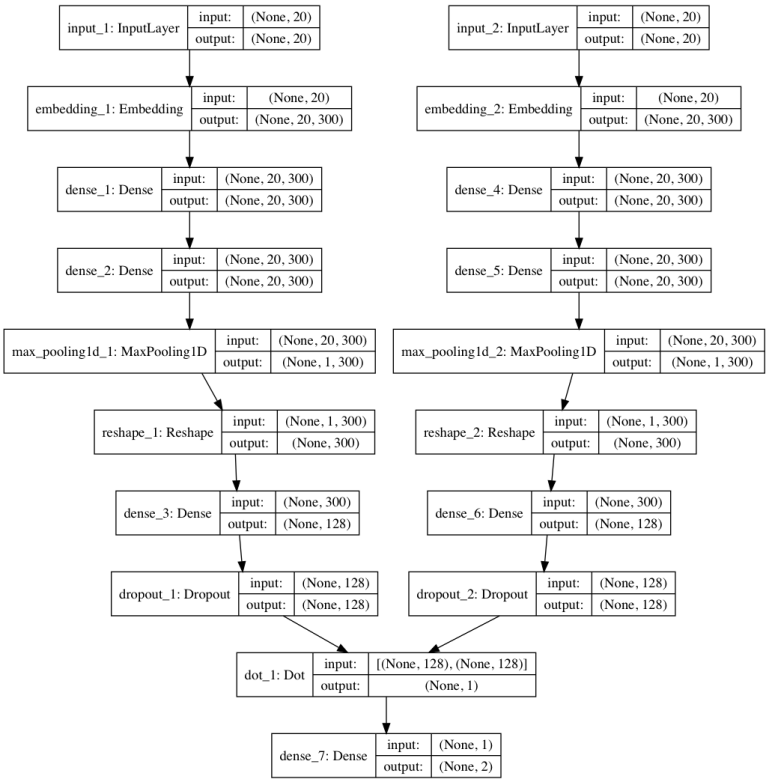

图 1: Baseline 模型结构（参考微软 DSSM）

其中，每条数据的两道题目分别作为 input_1 和 input_2，同时输入（由于之前采用了默认的 maxlen=20，所以这里的输入长度被限制为了 20）。之后是 Embedding 层，这里不采用预训练词向量，而是由模型自己学习 embedding。接下来是两个全连层，参考了微软论文的结构。有略微不同的是，由于之前没有做 word hashing，这里需要另外降维，所以我加了一层 Pooling。最后，再映射到 128 维的语义空间中，得到两个问题对应的语义向量。对其求 cosine 相似度，再用 Softmax 映射到二分类，即得到输出（loss function 选择了交叉熵）。

2.3.2 几点说明

* 对于语料中的数学符号，我在这里并没有过滤。如果使用预训练词向量，也许过滤更好，因为预训练词向量中基本不包括数学符号词条。但由于这里 Baseline 模型我用的是自训练词向量，数学符号的 embedding 也许是可以学习到的，因此这里就没有做过滤。
* 模型结构图中倒数第三层 (dropout) 虽然我加进去了而原论文没有，但 Baseline 模型我暂时把 dropout ratio 设成了 0，所以与原论文等价。
* 其他相关参数：maxlen=20，l2 regularization ratio=0.01，learning rate=0.001, embedding dim=300, train val split = 0.1, batch size=200, epoch=30, patience for early stopping=10

2.3.3 模型表现

在训练过程中，用 10% 的训练数据做 validation，记录模型在 training 和 validation set 上各个 epoch 中的 loss 和 accuracy：

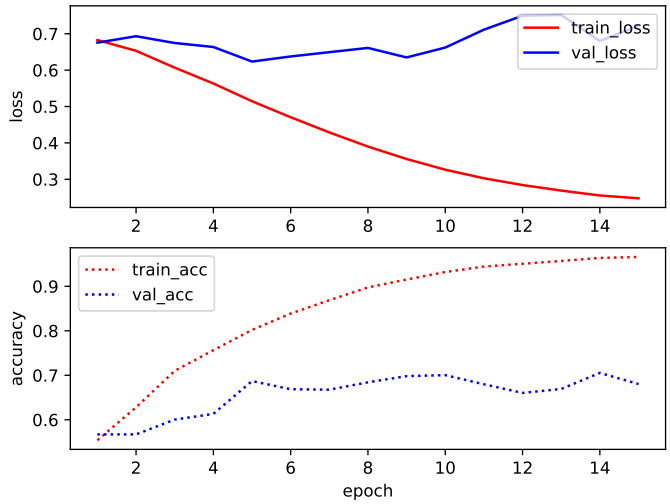

图 2: Baseline 模型训练过程中的表现

可以发现，overfitting 相当严重，虽然 val accuracy 在最后相比一开始略有提高，但整体趋势波动，且 loss 出现了不降反升的趋势。

在 test set 上评估最终模型的表现（最终模型取为各个 epoch 中 val loss 最优的模型）：

模型	Accuracy	Precision	Recall	F1 score
Baseline	0.654	0.629	0.75	0.684

表 1: Baseline 模型在测试集上的表现

可喜的是至少还是显著 above chance 的，recall 相比于 Precision 更高，说明有 overpredict positive 的倾向。

2.4 改进 Baseline 模型

在 Baseline 模型的基础上，尝试从以下几个方面做了改进：

2.4.1 数据清洗和词向量

在 Baseline 模型中对语料中的数学符号没有进行过滤，这里尝试了一下另外两种过滤的方案，分别称为“粗过滤”“细过滤”：

* “粗过滤”：即过滤掉所有非中文字符，包括标点符号。猜想这种方案可能会在使用预训练词向量时比较有用，因为预训练词向量大多只包含中文词语。而若要同时用双语词向量模型，则需要对其中一种语言的向量空间进行映射，否则无法保证两空间的可比性。

* “细过滤”：参考上周朱渝珊同学的作业，在中文词语之外，还保留有意义的 latex 数学符号（frac，sum，div，times 等）、运算符号（+，-）等。猜想这种方案可能会在使用自训练词向量时效果更好，原因如上。

使用与 Baseline 同样的参数，两种改进方案的表现与其对比如下（粗过滤使用了预训练词向量，而细过滤则没有）：

模型	Accuracy	Precision	Recall	F1 score
Baseline	0.654	0.629	0.75	0.684
Baseline+ 粗过滤	0.675	0.686	0.643	0.664
Baseline+ 细过滤	0.678	0.707	0.605	0.652

表 2: Baseline 模型与过滤改进模型在测试集上的表现对比

发现两个改进模型的 Accuracy 和 Precision 相比于 Baseline 均有提高，且细过滤比粗过滤提高更多；但同时，Recall 都降低了，从而 F score 也降低了，且细过滤比粗过滤也相应地降低更多。这似乎说明过滤精度越高，模型在预测正样本时就更倾向于谨慎。综合考虑，选择表现比较均衡的 Baseline+ 粗过滤作为当前最优模型。

2.4.2 DropOut 层

观察到 Baseline 模型的 overfitting 问题，于是尝试了一下在 2.4.1 的最优模型基础上的 ccosine 相似度之前加入一个 DropOut 层（即图 1 所示的结构，其中 dropout ratio=0.1）。实验结果如下：

模型	Accuracy	Precision	Recall	F1 score
Baseline+ 粗过滤（2.4.1）	0.675	0.686	0.643	0.664
Baseline+ 细过滤 +Dropout	0.662	0.628	0.79	0.7

表 3: 2.4.1 节最优模型与 Dropout 改进模型在测试集上的表现对比

发现加入 Dropout 层之后，Accuracy 略有下降，同时虽然 Precision 下降了，但 Recall 大幅上升，故 F score 最终有比较显著的提升。暂且将改进后的模型作为当前最优模型。

2.4.3 参数调整

观察以上模型的训练过程，发现 overfitting 问题依然比较严重：大多数情况下，在第 10 个 epoch 左右模型的 val loss 就陷入了瓶颈，之后略微波动，而 train loss 稳步下降，经常有 early stopping 的情况。于是，在 2.4.2 的模型基础上，尝试对其中一些可能影响 overfitting 的参数进行了以下范围的调整：

* l2:[0.01, 0.05]
* drop out ratio:[0.0, 0.1, 0.2]
* batch size:[100, 200, 400]

* maxlen:[10, 20, 30]

经过实验，发现这一范围内最优模型表现如下：

模型	Accuracy	Precision	Recall	F1 score
Baseline+ 粗过滤 +Dropout（2.4.2）	0.662	0.628	0.79	0.7
Baseline+ 细过滤 +Dropout+ 调参	0.743	0.753	0.716	0.734

表 4: 2.4.2 节最优模型与调参改进模型在测试集上的表现对比

参数：l2=0.05, drop out ratio=0.1, batch size=400, maxlen=10

除 recall 外，当前模型相比于 2.4.2 节模型在三个指标上均有提升，发现增大 l2，dropout ratio，和 batch size 的确有助于减小 overfitting 带来的影响。没有预料掉的一点是，句子最大长度 maxlen 这个参数，之前 baseline 中默认值设为 20 是为了训练效率考虑。本来以为 20 已经是个比较短的值了，因为训练语料中不乏相当长的句子，但实验中似乎设为 10 时反而表现更好。

2.4.4 其他模型结构

除了 DSSM 之外，我还试验了两个其它的网络结构：

* CDSSM
  参考微软论文（Shen 2014）和 github 开源项目（github.com/airalcorn2），搭建了如下模型：

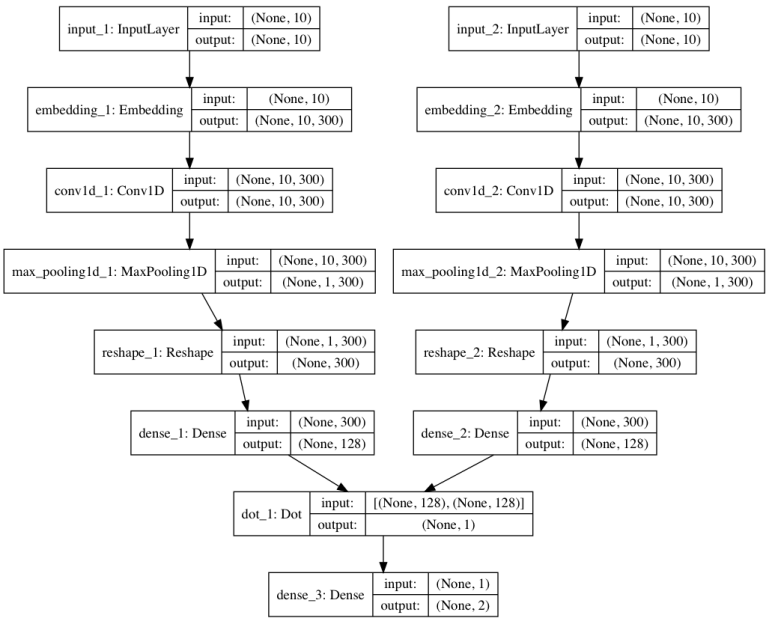

图 3: CDSSM 模型结构（参考微软论文）

同样，与原论文不同的是，对于中文语料并未做 word hashing，另外由于此处是二分类任务，在最后一层做 softmax 时也不需要原来的参数 gamma。

模型在测试集上的表现如下：

模型	Accuracy	Precision	Recall	F1 score
CDSSM	0.685	0.681	0.694	0.687

表 5: CDSSM 在测试集上的表现

模型参数与 2.4.3 节中最优模型相同，表现并不如后者，可能是因为没来得及调参...

* CGRU

之前 CDSSM 的一个理论缺陷在于，截取到的语序和上下文信息收到卷积窗口大小的限制，而无法获知长距离依赖的信息。于是，在 CDSSM 的基础上，我又尝试了一下 RNN 结构，两层 CNN+ 一层双向 GRU，称为 CGRU。其结构如下：

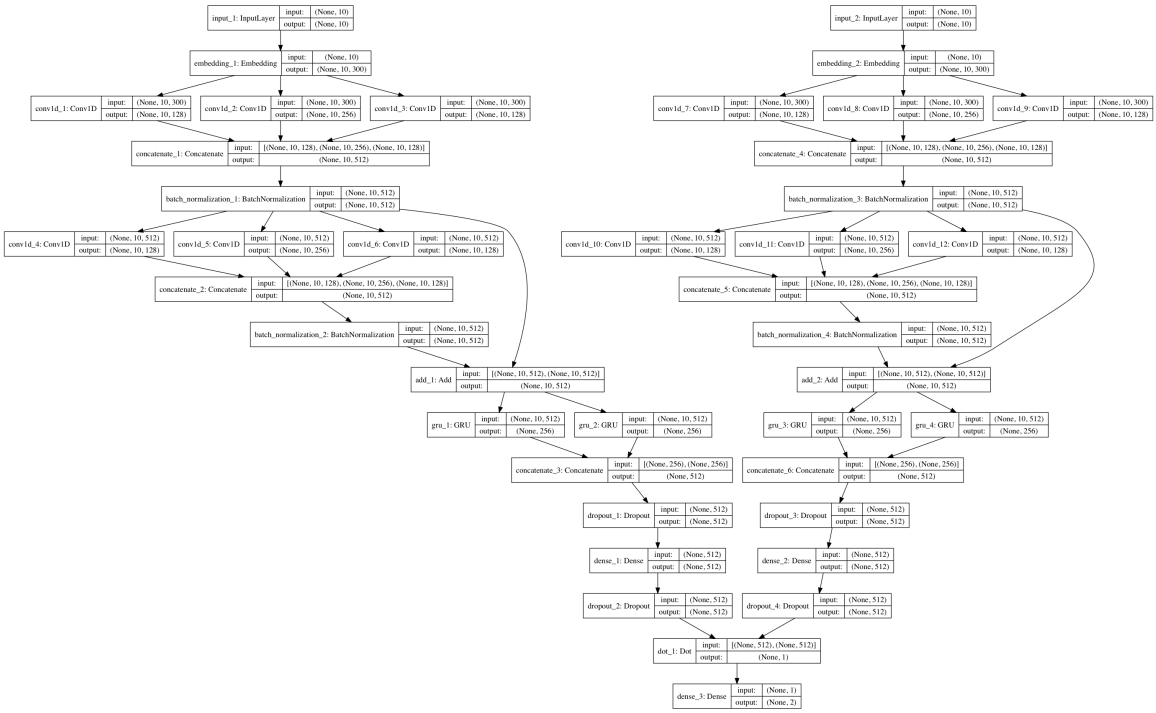

图 4: CGRU 模型结构（参考微软 CDSSM）

（其实两边结构是对称的，只是画的问题）。