# 人工智能学习体会

**———2015201974 左笑晨**

【**摘要**】：**首先说一说在AI小车制作过程中我做的一些事情，然后重点总结一下本学期在学习人工智能课程的同时自己亲自动手实践的一些问题，主要是在自然语言处理领域的两个问题：文本匹配问题和命名实体识别问题(采用的数据是京东和天猫商品标题)**

## 一、AI小车的制作

本次小车制作的过程大致分为三个阶段，第一个阶段主要是小车的拼装和自动避障功能的完成。这一阶段在代码部分没有什么困难，我们用的是课堂PPT上的代码，主要的过程就是小车的拼接。其实拼接小车对我们而言反而是一项挑战，由于长期与代码打交道，实际的动手能力比较差，再加上之前并没有接触过类似的任务，车子总是跑不起来，大部分的情况是接线部分的接触不良，以及电机驱动板连接端口总是出故障，最后四个电机端口都坏掉了。第二个阶段我们分步实现了我们最终要实现的功能的几个小部分。我们的最终功能是让小车自由行驶，并在行驶途中记录下来路线与沿途遇到的动物，实时绘制路线并标记拐点处识别出来的动物类别。我们把任务分成三个部分完成，分别是动物类别的识别、小车行进路线的实时绘制和电脑连接蓝牙串口模块。其中数据标记部分是我们大家共同完成的，我们收集了8种动物各100张图片并进行了标记，进行灰度化处理之后用最简单的CNN进行训练，最终模型对部分拍摄效果较好的照片识别结果可以接受，基本可以识别出八种常见的动物。

## 二、文本匹配问题

### 1、MatchPyramid Model

在文本匹配问题实现开始，我选择尝试的是MatchPyramid Model。模型的网络结构如图1所示。首先计算两段文本的相似度矩阵（将商品标题分词之后，用word2vec产生各词向量，两文本各词之间两两计算词向量的余弦值），在得到的相似度矩阵上用卷积池化操作，将最终池化后的矩阵平铺(flatten)成一维向量放入多层感知机中训练最终输出匹配分数，并在同一个batch内做排序，通过计算MAP（mean average precision）评价模型的准确性。

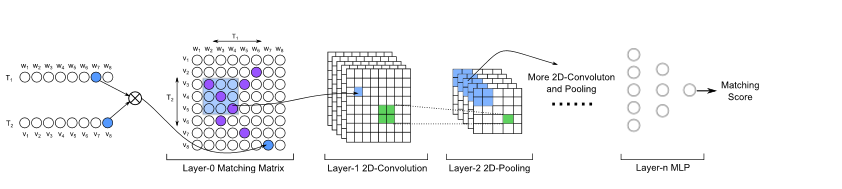
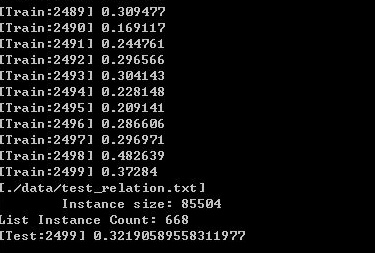


图1

通常情况下，由于自然语言处理问题通常是处理序列，因而大多采用RNN的网络结构，而此模型中通过在两文本之间的相似度矩阵上引入CNN的应用，从而可以挖掘出两段文本更深层次的语义信息，最终运行结果如下：



最终的MAP值大约在0.3左右。比直接用solr做检索得到的结果0.27好。

### 2、ARC-I模型

继MatchPyramid Model之后，在ARC-I模型我中进一步找到了CNN在文本匹配领域应用上的一些特点。

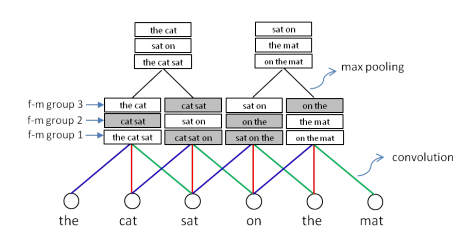


图2

如图2所示，ARC-I模型的第一层卷积是对其中一段文本上的一维卷积，卷积核的尺寸固定，如上图中卷积核的尺寸为3。此处的卷积操作可以理解成自然语言处理问题中的n-gram模型，n的值即为卷积核的尺寸。多个不同的卷积核提取了文本不同角度的信息，如图2中的f-m group1-3所示。

在普通的文本分类问题中也可以用这样的一维卷积，应用多种尺寸的卷积核，相当于组合了各种n-gram模型的结果，可以提取文本的多层次信息。由于在匹配问题中，需要保留一段文本中词语的顺序信息，以便于更精确的与另一个文本向量计算相似度，所以不能在一维卷积之后的池化降维后直接降至一维，正如图2中的池化操作中池化核的尺寸为2。若要对降维后的结果进一步做卷积操作，将第一层提取出的特征进行组合，第一个卷积层的卷积核尺寸必须相同，从而可以得到第二层卷积的输入矩阵。第二个卷积层为二维卷积，因为输入矩阵的每一行是不同卷积核对相同词提取的特征，彼此之间相互独立，可以进行组合。模型的结构如图3所示：

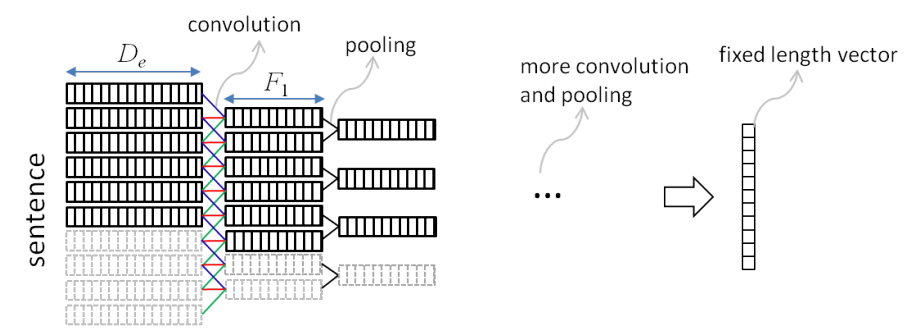


图3

在进行多次卷积池化操作之后，将最终的到的矩阵平铺得到了文本的表示向量，将两段文本得到的表示向量连接在一起，放入MLP中计算得到最终的匹配分数，模型总体结构如图4所示：

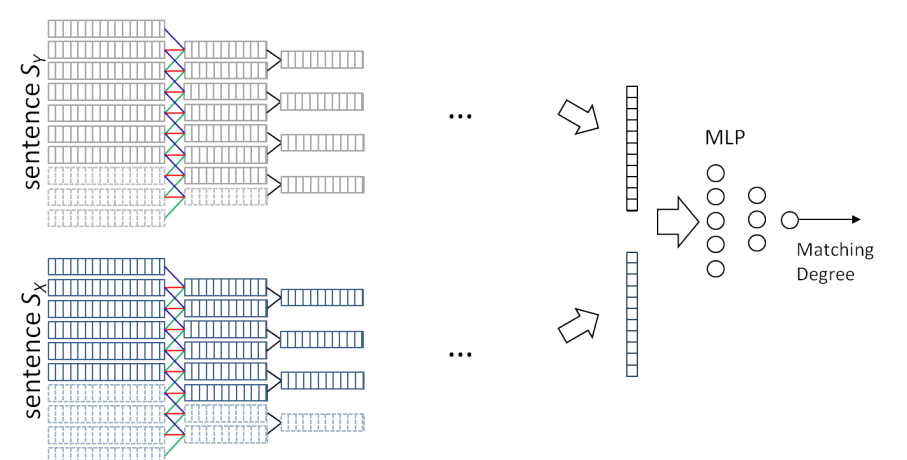


图4

ARC-I模型最终得到的结果如下：

|  |  |  |  |
| --- | --- | --- | --- |
|  | Accuracy | Precise | Recall |
| Value | 87% | 44% | 80% |

注：这里的Precise值有些异常，原因在于训练集中正负例是按照1：1投放的，测试集为了和solr结果做对比，用solr检索了排名前20的结果，大部分结果中前20条只有1到2条是匹配的，其余都是不匹配的，所以模型的Accuracy偏高的同时Precise值有些低。

## 3、**Convolutional Neural Tensor Network(CNTN)**

在ARC-I模型的基础上还可以做出一些改进。在得到两段文本的表示向量之后，ARC-I模型只采用了简单的向量拼接得到整体向量。将MLP换成Tensor层，就是CNTN模型。Tensor层的引入是为了更好地连接两段文本的主题向量，具体结构如图5所示：

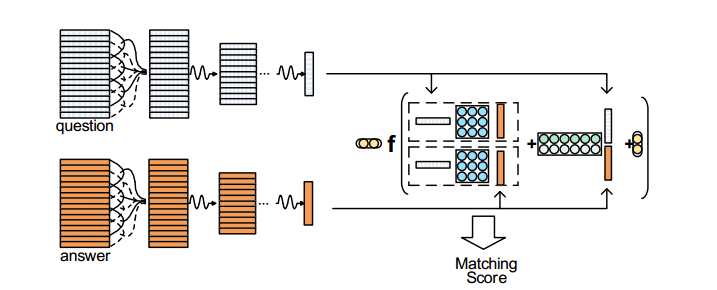


图5

具体Tensor层的公式如下：

其中a和b表示两段文本，M是三维张量，V是二维张量，M、V、b和u都是需要训练得到的参数，最终得到的是两段文本匹配的得分。加入Tensor层之后训练得到的结果为：

|  |  |  |  |
| --- | --- | --- | --- |
|  | Accuracy | Precise | Recall |
| Value | 90% | 51% | 88% |

### 4、ARC-II模型

Tensor层加入的原因就在于使用ARC-I模型是，两个文本在最终形成主题向量之前没有交流，可能导致匹配的不准确。因此，ARC-II在ARC-I的网络结构的基础上，将两段文本在第一个卷积层时就同步运算，因此ARC-I每一个卷积核形成的feature map是一个向量，在ARC-II中，每一个卷积核运算形成的feature map则是一个矩阵，如下图6所示：

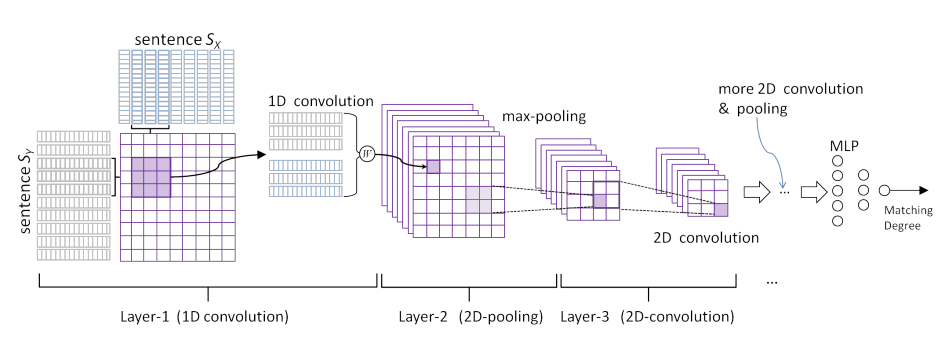


图6

ARC-II的结果出奇得好：

|  |  |  |  |
| --- | --- | --- | --- |
|  | Accuracy | Precise | Recall |
| Value | 94.6% | 71.2% | 89.4% |

### 5、Attention-Based CNN(ABCNN)

一般提到Attention机制，首先想到的是RNN，原因在于序列化信息的匹配过程中，其中某一序列的所有词对于另一个序列中的某一个词的重要程度并不相同。比如如下两个商品标题：

a. 汇源 100 果汁 系列 （ 苹果 ） 200ml 饮料 单盒

b. 韩国 进口 果汁 饮料 乐天 芒果汁 180ml

其中标题b中的“芒果汁”一词显然更应该关注标题a中的“苹果”一词，而不是“系列”一词。也就是处理“芒果汁”一词时，“苹果”和“系列”这两个词应被给予不同的关注度，也就是我们所说的Attention。在RNN中，比如自动应答的生成模型，在生成应答中的某个词时，会计算一个Attention向量，维数和问题次数相同，表示问题中各个词的重要程度。而在CNN中，很自然就可以想到，这里的Attention会以权重矩阵的形式存在。如图7所示：

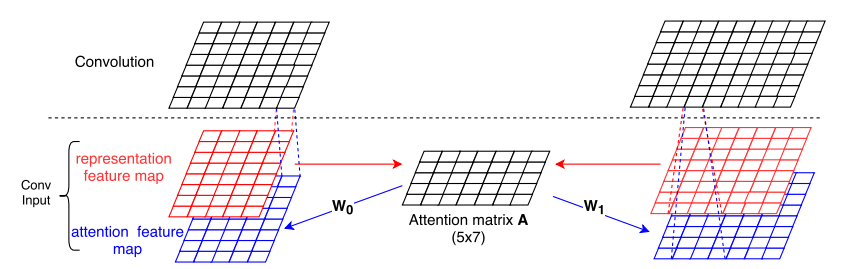


图7

图中间的矩阵是用两段文本的相似度矩阵作为Attention矩阵。红色的矩阵为正常的词向量矩阵，蓝色的矩阵是经中间的Attentioin矩阵计算出的Attention特征矩阵，将这两部分结合起来放入卷积层操作，就实现了Attention Based CNN。其中蓝色矩阵由Attention矩阵乘以W0，W1矩阵得来，W0，W1矩阵中的值都作为模型参数参加训练。而红色矩阵与蓝色矩阵的结合方式论文中没有提及。在实现过程中，我采用的是将蓝色矩阵的一列加和作为红色矩阵对应列向量的权值。这种方法是在卷积层加入了Attention机制。另一种方法是在池化层加入Attentioin机制，结构如图8所示：

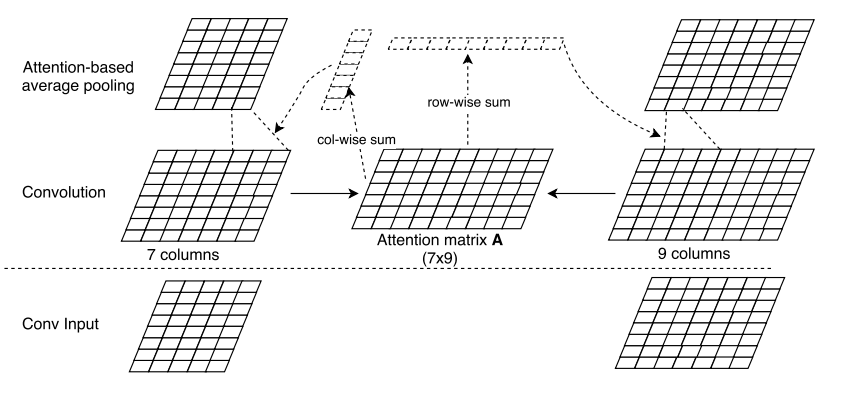


图8

这种结构与第一种结构类似，加入Attention的方法也是一样的，只不过在卷积之后的池化操作加入Attention，可以利用卷积操作的优点，组合局部信息，找到序列间隔更远、抽象程度更高的词组之间的关系，提高Attention机制的价值。由上述两种方法，自然可以想到第三种模型，那就是同时在卷积层与池化层添加Attention，总体结构图如下图9：

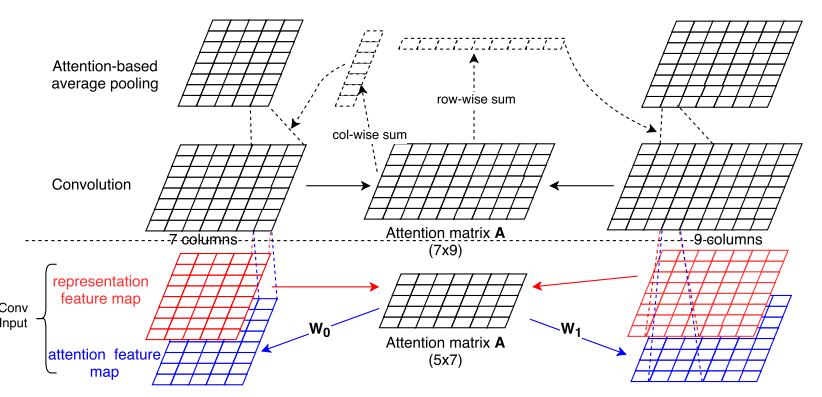


图9

在实验过程中，我只在卷积层之前加入了Attention机制，最后的评价指标有所改变（由于发现采用分类问题的思路可能不是很科学，所以采用了排序问题的思路，计算了下述指标，结果如下表）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| MAP | MRR | P@1 | P@5 | P@10 |
| 0.19 | 0.227 | 0.08 | 0.4 | 0.57 |

注：之前的MatchPyramid Model也采用了排序问题的评价指标，但两者使用的数据不同，所以最终的结果也有所不同，无法直接比较。采用此方法的结果比用solr检索结果稍差，这也是实验始料不及的。

### 6.Kernel Based Neural Ranking Model(K-NRM)

终于跳出了CNN的思路，开始尝试一种新的结构。K-NRM结构的核心在于kernel的应用。以往的文本匹配方法大多追求实匹配（exact matching），通过word2vec得到的词向量之间的相似度来衡量文本的相关关系。相似度高则倾向于相关，这在某些场合内是不太科学的。例如论文中提到的一个例子：某人想要查询一家匹兹堡的宾馆(pittsburgh hotel），由于波士顿(boston)和匹兹堡(pittsburgh）相似度高（原因可以研究word2vec的训练方法得知）,宾馆(hotel）和汽车旅店(motel)的相似度高，因此查询者要是得到了有关pittsburgh motel的结果或许会很开心，但是得到boston hotel的结果则不然了。

上面的例子说明一味追求实匹配并不一定会得到好的结果，因而可以尝试软匹配(soft match）。在该结构中，通过kernel指导词向量的训练，在不同层次相似度上提取soft-tf值作为特征进行learning to rank，从而实现了软匹配。模型的结构如下图10所示：

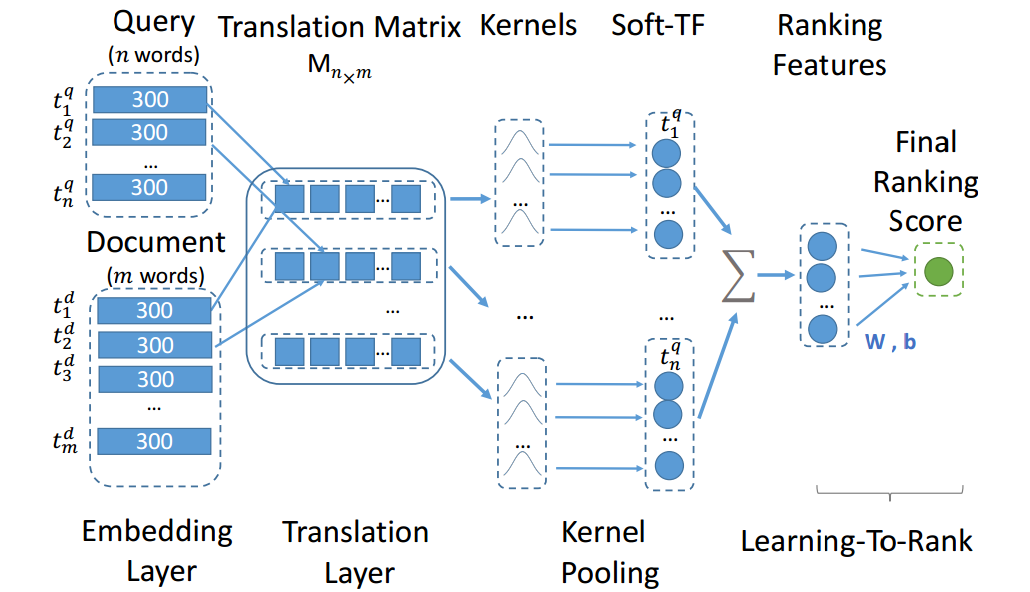


图10

上图提取自一篇关于信息检索的论文，所以Embedding Layer标注的是Query 和Document。在讨论文本匹配时，可以看成两段文本a和b。将Embedding后的词向量计算得到的相似度矩阵作为Translation层的输出，之后的kernel pooling层用来提取文本各词的soft-tf特征。采用高斯核函数，设置不同的相似度来实现多层次匹配，将提取的特征加和作为排序的特征。在实践过程中，我采用了idf值作为加和权重。最后经过输出层得出最终的匹配得分。由于这一部分的代码仍在实现，还没有得到具体结果。

## 三、命名实体识别问题

命名实体识别问题(NER)研究的不多，是在做文本匹配的时候考虑到在文本上做ner之后再匹配也许可以得到更好的效果，所以着手实现了一个ner模型，采用的结构是bi-lstm-crf，模型的结构如图11所示：

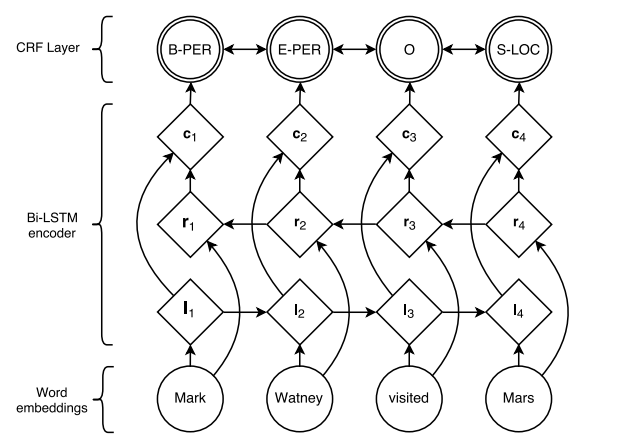


图11

最外层仍然是embedding层，由于原始数据的分词存在问题，可能影响ner的结果，我将原始数据的分词符去掉，并且将手动分好的词和所有的商品品牌名称加入jieba词典，用jieba再次分词并用word2vec得到词向量。将得到的词向量序列放入双向的lstm中训练得到k个t维向量。其中k为句子的词语个数，t为标记符号数量。将结果放入crf层来标记，得到每个词被标记成各个tag的概率，选出最高的标记序列。论文中将crf层表述成各标记之间的转换概率矩阵，将序列的转换概率之和与bi-lstm层输出的向量加和之后放入感知机中训练得到各标记概率。最终的结果很不错，如下图12所示：

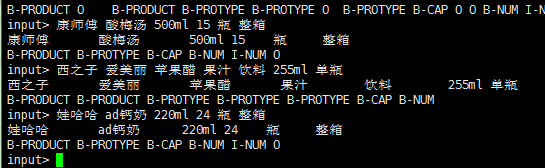


图12

## 参考文献：

1、Neural Architectures for Named Entity Recognition. *Proceedings of NAACL-HLT 2016, pages 260–270*

2、ABCNN: Attention-Based Convolutional Neural Network for Modeling Sentence Pairs.*arXiv:1512.05193v3 [cs.CL] 9 Apr 2016*

3、Convolutional Neural Tensor Network Architecturefor Community-based Question Answering. *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI 2015)*

4、End-to-End Neural Ad-hoc Ranking with Kernel Pooling. *SIGIR’17, August 7-11, 2017, Shinjuku, Tokyo, Japan*

5、Convolutional Neural Network Architectures for Matching Natural Language Sentences.

6、深度文本匹配综述. *《计算机科学报》2017年4月*