实验报告

计算语言学大作业（QA—Answer Selection）

谭鑫（1601111294）

一、 实验环境

操作系统	OS X EI Capitan
处理器	2.9 GHz Intel Core i5
内存	8 GB 1867 MHz DDR3

二、实验数据

实验数据在data文件中给出

1. ../data/wiki 文件下包含了训练集、开发集以及测试集数据，每类数据有包含了三种类型的文件：

1）.tsv 文件：对于问题和答案描述的原始文件（包含：QuestionID Question DocumentID DocumentTitle SentenceID Sentence Label）

2）.ref文件：对应.tsv文件中的QuestionID SentenceID Label

3）.qtype文件：问题类型的描述文件（Location Human Numeric Abbreviation Entity Description）

2. ../data/GoogleNews-vectors-negative300.bin:

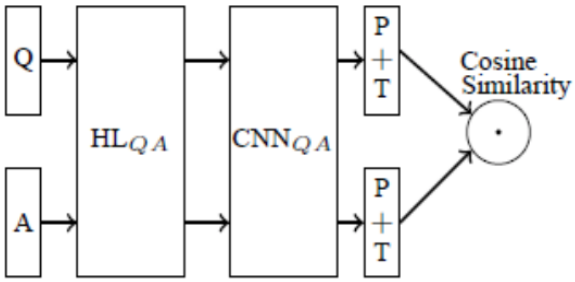
word2vec 是 Google 在 2013 年年中开源的一款将词表征为实数值向量的高效 工具，采用的模型有 CBOW(Continuous Bag-Of-Words，即连续的词袋模型)和 Skip-Gram 两种。word2vec 代码链接为:https://code.google.com/p/word2vec/。word2vec 通过训练，可以把对文本内容的处理简化为 K 维向量空间中的向量运算，而向量空间上的相似度可以用来表示文本语义上的相似度。

三、编译运行

命令以shell脚本的格式给出：../code/go_wiki.sh

四、实验原理

基于CNN（theano）



首先问题和答案通过word2vec转换为词向量。然后Q&A共用一个网络，网络中包括HL，CNN，P+T和Cosine_Similarity，HL是一个g(W*X+b)的非线性变换，CNN是卷积神经网络层，P是max_pooling，T是激活函数Tanh，最后的Cosine_Similarity表示将Q&A输出的语义表示向量进行相似度计算。

1.将问题和答案向量化

```
#将问题和答案向量化
def make_cnn_data(revs, word_idx_map, max_l=50, filter_h=3, val_test_splits=[2,3]):
    """
```

```
# define model architecture
index = T.lscalar()
lx = T.matrix('lx')
rx = T.matrix('rx')
y = T.ivector('y')
Words = theano.shared(value = U, name = "Words")
# input: word embeddings of the mini batch
llayer0_input = Words[T.cast(lx.flatten(),dtype="int32")].reshape((lx.shape[0],1,lx.shape[1],Words.shape[1]))
# input: word embeddings of the mini batch
rlayer0_input = Words[T.cast(rx.flatten(),dtype="int32")].reshape((rx.shape[0],1,rx.shape[1],Words.shape[1]))
```

2. 定义模型的输入，将问题和答案作为两个特征分开考虑

```
conv_layer = QALeNetConvPoolLayer(rng, linp=llayer0_input, rinp=rlayer0_input,
                        filter_shape=filter_shape, poolsize=pool_size)
```

3. 对问题和答案进行卷积和池化操作

```
#将输入特征与filter卷积，conv.conv2d函数
lconv_out = conv.conv2d(input=linp, filters=self.W)
rconv_out = conv.conv2d(input=rinp, filters=self.W)

#self.b.dimshuffle('x', 0, 'x', 'x'):将self.b一维向量转换成shape(1, filter_shape[0], 1, 1)四维
#激活函数（每组四个特征进行求和，加权，加偏置）
lconv_out_tanh = T.tanh(lconv_out + self.b.dimshuffle('x', 0, 'x', 'x'))
rconv_out_tanh = T.tanh(rconv_out + self.b.dimshuffle('x', 0, 'x', 'x'))

#池化操作
self.loutput = downsample.max_pool_2d(input=lconv_out_tanh, ds=self.poolsize, ignore_border=True,
    mode="average_exc_pad")
self.routput = downsample.max_pool_2d(input=rconv_out_tanh, ds=self.poolsize, ignore_border=True,
    mode="average_exc_pad")
self.params = [self.W, self.b]
```

4. 生成的llayer_inputs以及rlayer1_inputs是一个python的list，使用concatenate将list的多个tensor拼接起来（list中的每个tensor表示一种大小的filter卷积的结果）

```
for conv_layer in conv_layers:
    test_llayer0_output, test_rlayer0_output = conv_layer.predict(test_llayer0_input, test_rlayer0_input)
    test_lpred_layers.append(test_llayer0_output.flatten(2))
    test_rpred_layers.append(test_rlayer0_output.flatten(2))
test_llayer1_input = T.concatenate(test_lpred_layers, 1)
test_rlayer1_input = T.concatenate(test_rpred_layers, 1)
```

5. 模型的低层由卷基层和最大池化曾组成，高层是一个全连接的MLP神经网络（隐层+逻辑回归，ANN），高层的输入是下层特征图的集合。针对给定的epoch数（epoch=5）模型进行训练，将上一步卷积的结果作为逻辑回归的输入，进行预测。

```
# 使用logistic regression进行模型的预测，返回分类
test_y_pred = classifier.predict(test_llayer1_input, test_rlayer1_input)
test_model = theano.function([lx, rx], test_y_pred)
```

6. 最终按照预测结果的大小对每个问题的答案分别进行排名

五、实验结果

开发集和测试集的训练结果在../pred/wiki/cnn-dev.rank和../pred/wiki/cnn-test.rank文件中给出。

开发集的评测结果如下：

```
Final Evaluation Score:
MAP: 0.646533393855
MRR 0.658983003328
```

六、参考文献

[1]Feng M, Xiang B, Glass M R, et al. Applying deep learning to answer selection: A study and an open task[C]// Automatic Speech Recognition and Understanding. IEEE, 2015.
[2]Yang Y, Yih W T, Meek C. WikiQA: A Challenge Dataset for Open-Domain Question Answering[C]// Conference on Empirical Methods in Natural Language Processing. 2015.
[3]Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. In Proceedings of Workshop at ICLR, 2013.
[4]Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed Representations of Words and Phrases and their Compositionality. In Proceedings of NIPS, 2013.