实验四 电影评论情感分类

一、 实验目的

1. 进一步加深对卷积神经网络基本原理的理解。
2. 掌握卷积神经网络处理文本的各项技术。
3. 掌握文本分类模型 Text-CNN 的架构和原理。

二、 实验要求

1. 任选一个深度学习框架建立 Text-CNN 模型（本实验指导书以 TensorFlow 为例）。
2. 实现对中文电影评论的情感分类，实现测试准确率在 83%以上。
3. 也可采用 LSTM 实现，实现测试准确率高于卷积神经网络。
4. 按规定时间在课程网站提交实验报告、代码以及 PPT。

三、 实验原理

Text-CNN 和传统的 CNN 结构类似，具有词嵌入层、卷积层、池化层和全连接层的四层结构，如图 1 所示。

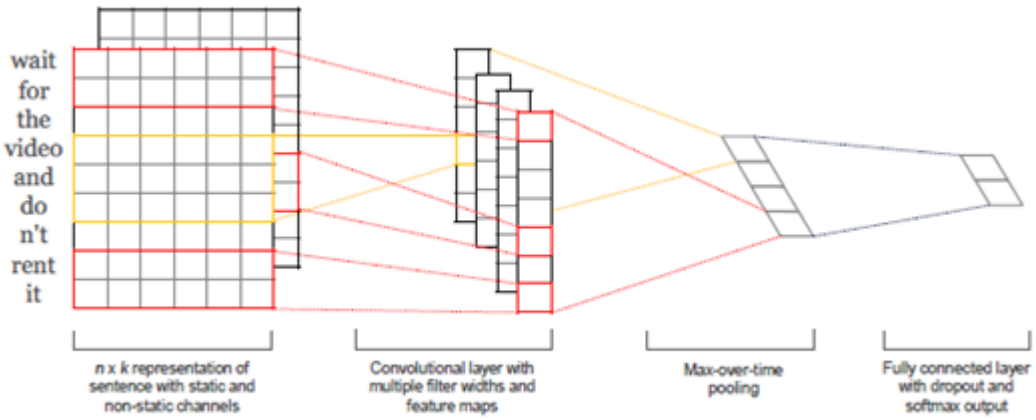

图 1 Text-CNN 网络结构

如图 1 中所示，Text-CNN 的词嵌入层（Word embedding）使用二维矩阵来表示长文本。词嵌入将输入文本的每个词语通过空间映射，将独热表示（One-Hot Representation）转换成分布式表示（Distributed Representation），进而可以

使用低维的词向量来表示每一个词语。经过词嵌入，每个单词具有相同长度的词向量表示。将各个词语的向量表示连起来便可以得到二维矩阵。得到词向量的方式有多种，常用的是 Word2vec 方法。若使用预训练好的词向量，在训练模型的时候可以选择更新或不更新词向量，分别对应嵌入层状态为 Non-static 和 Static。

Text-CNN 的卷积层是主要部分，卷积核的宽度等于词向量的维度，经卷积后可以提取文本的特征向量。与在图像领域应用类似，Text-CNN 可以设置多个卷积核以提取文本的多层特征，长度为 N 的卷积核可以提取文本中的 N-gram 特征。

Text-CNN 的池化层一般采取 Max-over-time pooling，输出最大值，从而判断词嵌入中是否含 N-gram。

Text-CNN 的全连接层采用了 Dropout 算法防止过拟合，并使用 Softmax 函数输出各个类别的概率。

算法具体原理可阅读：Kim Y .2014--《Convolutional Neural Networks for Sentence Classification》一文（已放到课程网站这个实验作业的文件夹中）。

四、 实验所用工具及数据集（以 Tensorflow 为例）

1. 主要工具

Python 3.5+、TensorFlow 1.3.0、Numpy 1.13.1、jieba 0.39

2. 数据集

1) 训练集：包含 2W 条左右中文电影评论，其中正负向评论各 1W 条左右。

2) 验证集：包含 6K 条左右中文电影评论，其中正负向评论各 3K 条左右。

3) 测试集：包含 360 条左右中文电影评论，其中正负向评论各 180 条左右。

4) 预训练词向量。中文维基百科词向量 word2vec。

数据集已经放到课程网站这个实验作业的文件夹中。

五、 实验步骤与方法

1. 加载本实验所有函数库

2. 数据预处理

① 设置分类类别以及类别对应词典{pos:0, neg:1};

② 构建词汇表并存储，形如{word: id};

```
# only one start
def build_word2id(file):
    """
    :param file: word2id #保存地址
    :return: None
    """
    word2id = {'_PAD_': 0}
    path = ['./data/train.txt', './data/validation.txt']
    print(path)
    for _path in path:
        with open(_path, encoding='utf-8') as f:
            for line in f.readlines():
                sp = line.strip().split()
                for word in sp[1:]:
                    if word not in word2id.keys():
                        word2id[word] = len(word2id)

    with open(file, 'w', encoding='utf-8') as f:
        for w in word2id:
            f.write(w+'\t')
            f.write(str(word2id[w]))
            f.write('\n')
```

③ 加载上述构建的词汇表;

④ 基于预训练好的 word2vec 构建训练语料中所含词语的 word2vec;

```
def build_word2vec(fname, word2id, save_to_path=None):
    """
    :param fname # 预训练的 word2vec.
    :param word2id # 语料文本中包含的词汇集.
    :param save_to_path # 保存训练语料库中的词组对应的 word2vec 到本地
    :return  #语料文本中词汇集对应的 word2vec 向量{id: word2vec}
    """
```

```
    import gensim
    n_words = max(word2id.values()) + 1
    model = gensim.models.KeyedVectors.load_word2vec_format(fname,
binary=True)
    word_vecs = np.array(np.random.uniform(-1., 1., [n_words, model.vector_size]))
    for word in word2id.keys():
      try:
        word_vecs[word2id[word]] = model[word]
      except KeyError:
        pass
    if save_to_path:
      with open(save_to_path, 'w', encoding='utf-8') as f:
        for vec in word_vecs:
          vec = [str(w) for w in vec]
          f.write(' '.join(vec))
          f.write('\n')
    return word_vecs
```

⑤ 加载上述构建的 word2ve;

⑥ 加载语料库：train/dev/test;

⑦ 生成批处理 id 序列。

经过数据预处理，数据的格式如下：

x: [1434, 5454, 2323, ..., 0, 0, 0]

y: [0, 1]

x 为构成一条评论的词所对应的分类 id。 y 为 onehot 编码: pos-[1, 0], neg-[0, 1]

3. 建立 Text-CNN 模型

① 配置模型相关参数，在 COINFIG 类中完成

② 使用 TensorFlow 框架完成 Text-CNN 模型的建立

```
class CONFIG():
    update_w2v = True        # 是否在训练中更新 w2v
    vocab_size = 59290       # 词汇量，与 word2id 中的词汇量一致
    n_class = 2              # 分类数：分别为 pos 和 neg
```

```
max_sen_len = 75          # 句子最大长度
embedding_dim = 50        # 词向量维度
batch_size = 100          # 批处理尺寸
n_hidden = 256            # 隐藏层节点数
n_epoch = 10              # 训练迭代周期，即遍历整个训练样本的次数
opt = 'adam'              # 训练优化器：adam 或者 adadelta
learning_rate = 0.001     # 学习率；若 opt=‘adadelta'，则不需要定义学习率
drop_keep_prob = 0.5      # dropout 层，参数 keep 的比例
num_filters = 256         # 卷积层 filter 的数量
kernel_size = 4           # 卷积核的尺寸；nlp 任务中通常选择 2,3,4,5
print_per_batch = 100     # 训练过程中,每 100 词 batch 迭代，打印训练信息
  save_dir = './checkpoints/' # 训练模型保存的地址
  train_path = './data/train.txt'
  dev_path = './data/validation.txt'
  test_path = './data/test.txt'
  word2id_path = './data/word_to_id.txt'
  pre_word2vec_path = './data/wiki_word2vec_50.bin'
  corpus_word2vec_path = './data/corpus_word2vec.txt'
```

4. 模型训练与验证

使用训练集和验证集完成模型训练、验证。返回训练、验证损失和准确率。

5. 模型测试

使用测试集完成模型的测试。通过准确率、召回率、F1-分数、混淆矩阵指标来评估模型的性能。

6. 预测

使用 predict 函数完成电影评论的情感分类。