重庆邮电大学

学生实验实习报告册

学年学期：	2022-2023 学年(秋)学期
课程名称：	行业大数据分析综合实践
学生学院：	计算机学院/人工智能学院
专业班级：	数据科学与大数据技术
学生学号：	XXXXXX
学生姓名：	陈浩如
联系电话：	19823324153

重庆邮电大学教务处印制

目　录

教师评阅记录表

【重要说明】

- 学生提交报告册最终版时，**必须包含此页**，否则不予成绩评定。
- 本报告册模板内容格式除确实因为填写内容改变了布局外，**不得变更其余部分的格式**，否则不予成绩评定。

报告是否符合考核规范	☑ 符合　☐ 不符合
报告格式是否符合标准	☑ 符合　☐ 不符合
报告是否完成要求内容	☑ 是　☐ 否
报告评语：	
报告成绩：	
评阅人签名（签章） 2023 年 1 月 8 日	

实验或实习报告

课程名称	行业大数据分析综合实践	课程编号	A2040870
开课学院	计算机学院/人工智能学院		
指导教师	XX		
实验实习地点	综合实验大楼 B517		

学号	姓名
*****	*****

实验一

一、实验题目及内容

课后作业（二）3：利用 jieba 对《网络新闻实体发现与情感识别》案例数据进行分词

二、实验过程步骤（注意是主要关键步骤，适当文字+截图说明）、实验结果及分析

1.读入数据：

由于数据是 json 模式存储，故利用 codecs 库进行 utf-8 编码的流读入，按行处理 json 数据后保存为 DataFrame 格式返回。

```
def read_data(filepath):
    #利用codecs库打开文件
    f = codecs.open(filepath, 'r', 'utf-8')
    #临时列表存储读入的json文件
    data = []
    for line in f.readlines():
        #json.loads读入为字典
        news = json.loads(line.strip())
        data.append(news)
    #转化成DataFrame格式
    return pd.DataFrame(data)
```

将训练集和测试集读入，并将标题和文章合并，分别存为 data_train 和 data_test。

```
train = read_data('internet-train.txt')
test = read_data('internet-test.txt')
data_train = train['title'] + train['content']
data_test = test['title'] + test['content']
```

2. Jieba 词库导入自定义词典

由于此项目对于实体分词的准确性要求很高，所以需要导入常用中文词典。同时，对 train 数据中出现过的标签实体，在去重后保存为自定义词典。这是因为通常项目的文章选取应该是同分布的，虽然测试集的选取是随机的，但是在导入训练集的标签实体后，由于同分布的原因，会大大增加实体

分词准确率。

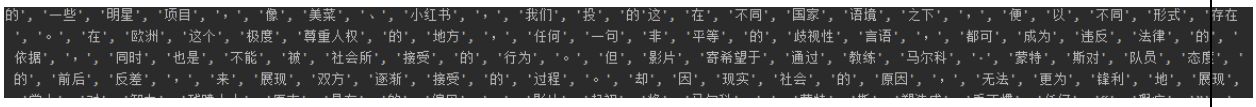
```
jieba.load_userdict('userditc.txt') #导入常用中文词典
jieba.load_userdict('entities.txt') #导入训练集标签去重实体集
```

3. jieba 分词

对训练集和测试集的文本数据分词，保存为列表形式。同时由于加载了一定的训练集标签去重实体集及常用中文词典，应当关闭 cut_all 模式，因为如果开启后，会出现重复分词的大词，影响语序。

```
for t in data_train:
    s_l = jieba.cut(t, cut_all=False)
    train_data.append(' '.join(s_l).split())
for t in data_test:
    s_l = jieba.cut(t, cut_all=False)
    test_data.append(' '.join(s_l).split())
```

4. 实验结果



可以看到分词效果较为明显，但存在大量无意义的语句，比如标点符号和一系列语气动词，故考虑读入 stop_words 加入停用词，再尝试分词。

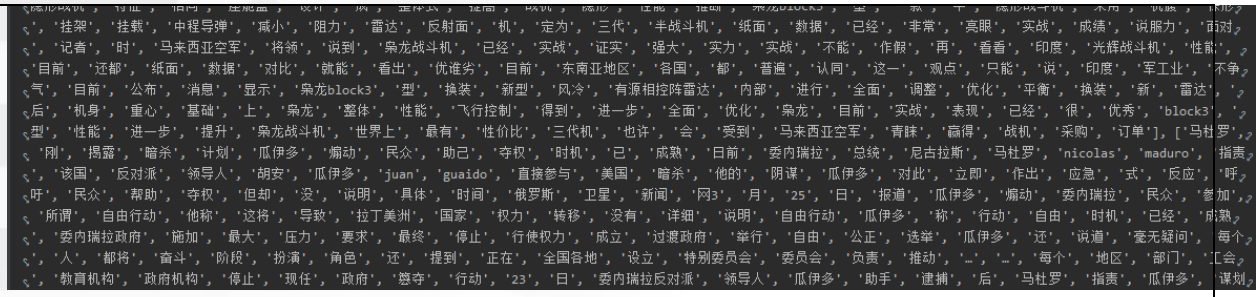
读入 stop_words 代码如下

```
def get_stopwords():
    with open('stop_words.txt', 'r', encoding='utf-8') as f:
        stop_words = [word.strip('\n') for word in f.readlines()]
    return stop_words
```

修改 jieba 分词过程，将在 stop_words 中的词语丢去。

```
for t in data_train:
    s_l = jieba.cut(t, cut_all=False)
    tmp = ' '.join(s_l).split()
    x = [word for word in tmp if word not in stop_word]
    train_data.append(x)
for t in data_test:
    s_l = jieba.cut(t, cut_all=False)
    tmp = ' '.join(s_l).split()
    x = [word for word in tmp if word not in stop_word]
    test_data.append(x)
```

输出实验结果如下



可以看到分词效果极佳，为接下来训练 word2vec 做好了准备。

5.实验分析

NLP 任务中，最重要的就是将文本数据转换成数字，再丢入模型训练。在第一步文本转换数字的时候，多数任务以词为单位，故中文分词 jieba 库就大放光彩。Jieba 库支持自定义词典，以及多种切割模式。

在做分词的时候，要多注意分词的准确率，根据任务的具体情况，添加不同的自定义词典，来达到更好的分词效果。

实验二

一、实验题目及内容

课后作业（二）4：利用 Gensim 训练《网络新闻实体发现与情感识别》案例数据的 word2vec 词向量。

二、实验过程步骤（注意是主要关键步骤，适当文字+截图说明）、实验结果及分析

1. 数据准备

在实验一中准备好的训练集和测试集的所有文本分词结果列表，将其合并为总词集。

```
data = train_data + test_data
```

2. Word2Vec 训练

将数据集送入 Word2Vec 中训练，由于该项目语序较为重要，故将 window 开长，加强上下文与对应词的联系，同时线程开到实验用机的 CPU 最大线程 8。

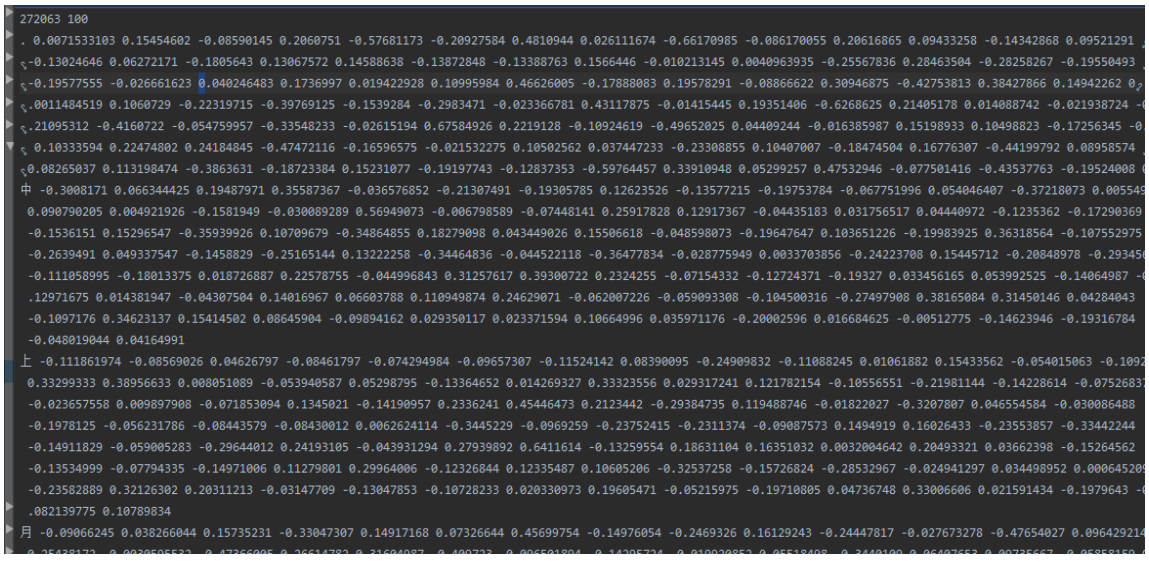
```
model = gensim.models.Word2Vec(data, sg = 1, vector_size=100, window=5, min_count=2, negative=3, sample=0.001, hs=1, workers=8)
```

3. Word2Vec 模型保存

```
model.wv.save_word2vec_format('word2vec_model.txt', binary=False)
```

4. 实验结果

开启训练好的模型查看，维度是设定好的 100 维。训练效果较好。



5. 实验分析

Word2Vec 是目前最主流的训练词向量的算法之一，虽然运行速度没有 FastText 那么快，但准确率和词相似度都更高。在训练时要根据项目的具体情况调整训练参数，如果上下文更重要，就要加大 window 取值。本实验选择了 Cbow 模型。负采样设置为 3，之后可以根据项目的具体需求调整参数的设置。

实验三

一、实验题目及内容

课后作业（三）2：设计和实现一套《020 商铺食品安全相关评论发现》项目的机器学习解决方案，并在线上进行结果提交（**本题基准参考成绩：0.8833**）

二、技术方案（文字描述+图表说明为主，**不得大段粘贴代码**）、实验结果及分析（技术方案的书写可以适当参考 QQ 群文件中的“R3.技术方案撰写参考文档”）

1. 线上测试成绩（给出最优成绩和线上测评 ID）

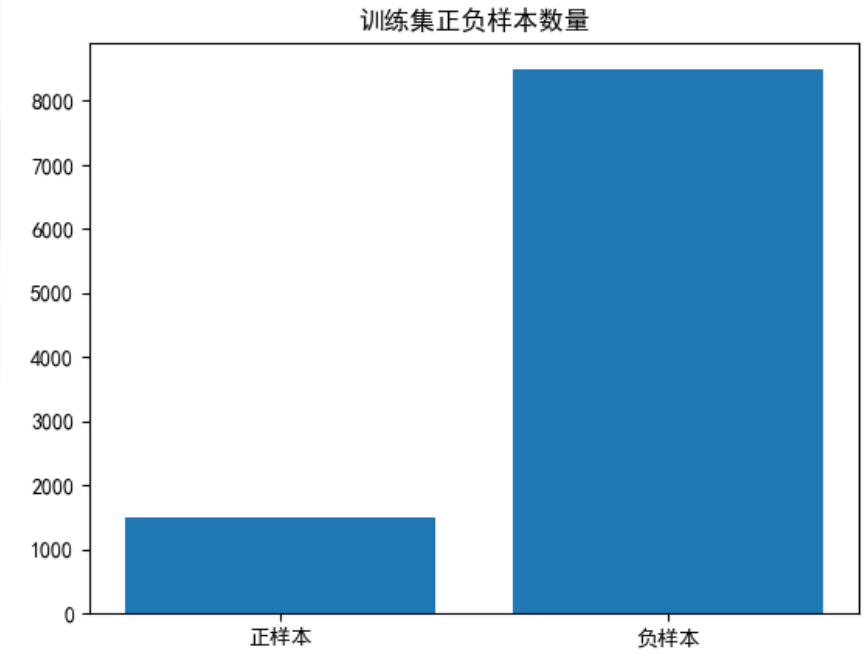
2. 摘要

在新型餐饮模式的发展下，外卖行业兴起。但随着外卖行业的兴起，食品安全也不可忽视。本次文本分类任务针对外卖评论是否涉及食品安全进行二分类。本次二分类任务出于学习的目的，基于一万条训练样本，共使用三种模型（TF-IDF + XGBOOST、TF-IDF+SVM、BOW+SVM），得到了三个分类模型，在模型训练的途中，选择了五折交叉验证的方法进行模型融合。最后，线上取到了 F1=0.9190 的成绩。

3. 数据分析与数据预处理

3.1 数据可视化分析

(1)对训练集中正样本和负样本做可视化统计，输出如下



可见整个训练集中关于食品安全问题的评论还是占少数。

(2)对训练集中每条正样本词集做交集，观察哪些词语在正样本中出现次数最多。

绘制词云图后发现，上述明显与食品卫生安全强相关的词语导致了标签为正。

(3)判断这些高频食品安全词语在负样本中出现次数

```
cnt = 0
for i in neg['comment']:
    x = ' '.join(jieba.cut(i)).split()
    x = [y for y in x if y not in stop_word]
    ok = 0
    for j in x:
        if j in out:
            ok = 1
            print(j)
    if ok == 1:
        cnt += 1

print('食品安全高频词在负样本中出现%d次'%cnt)
```

```
发现
吐
发现
发现
发现
死
发现
死
食品安全高频词在负样本中出现108次

Process finished with exit code 0
```

可以看到这些高频词在负样本中极少出现，合理确定为特殊词语，在后续导入用户词典的时候应该将高频词加入 jieba 分词词典，这样能提高准确率。

3.2 数据预处理

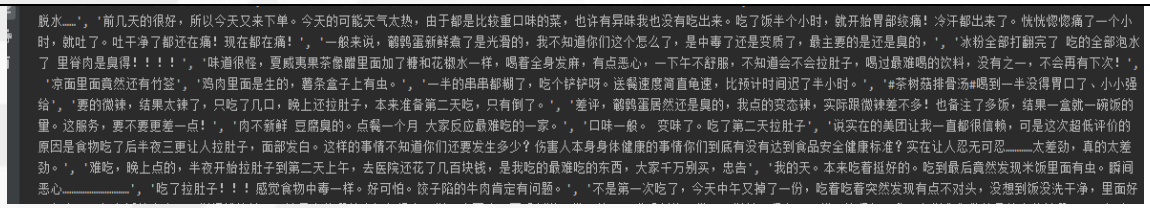
训练集和测试集的 comment 中存在大量表情、颜文字、图片标签、html 标签等无用数据。在调用 filter 函数通过正则匹配对这些文字过滤后，达到了较好的效果。

Filter 函数如下：

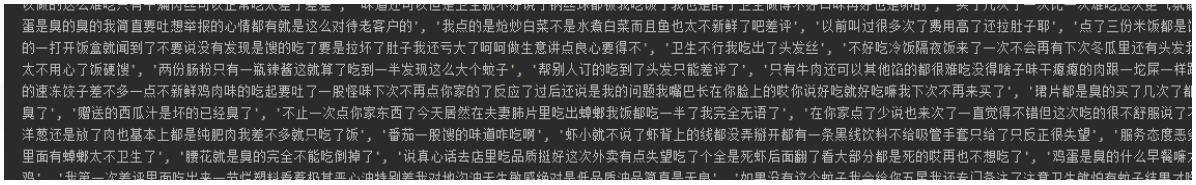
```
def filter(text):
    text = re.sub(r"[A-Za-z0-9\!\=\? \%\[\]\,\ (\) \>\<:&lt;\/#\. -----\_]", "", text)
    text = text.replace('图片', '')
    text = text.replace('\xa0', '')

    cleanr = re.compile('<.*?>')
    text = re.sub(cleanr, ' ', text)
    text = re.sub(r"\\【.*?】+|\\《.*?》+|\\#.*?#+|[.!/_,$&%^*()~<>+?@|:♥@~{}#]+|[—! \\\, 。=？ 、：“”‘’￥…… （） 《》 【】]",'',text)
    text = text.strip()
    return text
```

过滤前的文本如下，存在大量标点符号和无用的表情图。



过滤后文本就显得较为干净，适用于 jieba 分词



4. 算法实现与算法优化

机器学习中最重要的一步便是特征工程，但在 NLP 任务中很难以数字特征来构建新的特征，一般按照词序、词频、词向量等构建特征工程，项目中采用了三种构建特征工程的方法，TF-IDF、词向量、BOW，其中词向量方法在机器学习时的效果很差，故本项目只采用了 TF-IDF、BOW 两种构建特征工程的方法。

模型构建过程中，由于本项目是分类任务，在分类任务上发挥出色的有逻辑回归、SVC 分类器、XGBoost 分类算法。其中 SVC 和 XGBoost 的性能更佳，故本项目采用了这两种模型。

在对两种特征工程和两种模型选择的组合实验下，选择了 TF-IDF 和 SVM、TF-IDF 和 XGBoost、BOW 和 SVM 三种方案。

(1)TF-IDF 和 SVM

使用 TF-IDF 模型综合考虑上下文词频关系，以及一小段滑窗内的上下文联系。再使用 SVM 的二分类专属模型 SVC 完成文本分类任务。

在模型调优也就是参数选择方面，通过网格搜索的办法，找到最优的参数。

```
param_grid = [
{'kernel': ['linear', 'sigmod', 'poly'], 'C': [0.7, 0.8, 0.9, 1.0, 1.1,1.2]},
{'gamma': [0.001, 0.002,0.01,0.003,0.1,0.004], 'coef0': [0.0,0.1,0.2,0.01,0.001], 'degree': [1, 2, 3, 4]},
]
model = SVC()
grid_search = GridSearchCV(model, param_grid, cv=5,
                           scoring='neg_mean_squared_error')

grid_search.fit(X_train, Y_train)
print(grid_search.best_estimator_)
```

网格搜索后得到最优参数如下

```
model = SVC(C=0.9, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
            decision_function_shape='ovr', degree=3, gamma=0.001, kernel='linear',
            max_iter=-1, random_state=None, shrinking=True,tol=0.001, verbose=False)
```

进行 TF-IDF 及 SVC 模型训练。

```
def TFIDF_SVM():
    model = SVC(C=0.9, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
                decision_function_shape='ovr', degree=3, gamma=0.001, kernel='linear',
                max_iter=-1, random_state=None, shrinking=True, tol=0.001, verbose=False)
    model.fit(X_train_TFIDF, Y_train_TFIDF)
    preds = model.predict(X_test_TFIDF)

    return preds.tolist()
```

得到 F1 分数为 0.91904765000

(2)TF-IDF 和 XGBoost

使用 TF-IDF 模型综合考虑上下文词频关系，以及一小段滑窗内的上下文联系。再使用集成学习大杀器 XGBoost 完成文本分类任务。

XGBoost 由于训练较为特殊，且难以调参，便没有使用网格搜索的方法，根据经验调整了迭代轮数，最深层数，叶节点最大个数，学习率和 gamma，完成了训练。

```
def TFIDF_XGBOOST():
    params = {'booster': 'gbtree', 'objective': 'binary:logistic', 'eval_metric': 'auc', 'gamma': 0.1, 'min_child_weight': 1.1,
              'eta': 0.01, 'tree_method': 'gpu_hist', 'seed': 0, 'nthread': 12, 'predictor': 'gpu_predictor'}
    tr = X_train_TFIDF
    te = X_test_TFIDF
    train_data = xgb.DMatrix(tr, label=Y_train_TFIDF)
    test_data = xgb.DMatrix(te)
    w = [(train_data, 'train')]

    model = xgb.train(params, train_data, num_boost_round=5000, evals=w, early_stopping_rounds=50)
    predict = pd.DataFrame(model.predict(test_data, validate_features=False), columns=['prob'])
    predict = predict['prob'].tolist()
    res = [1 if x >= 0.5 else 0 for x in predict]
    return res
```

由于 XGBoost 输出预测结果表示该样本为正例的概率，所以设定阈值 0.5，若概率在 0.5 以上便标记为真。

得到 F1 分数为 0.85641026000

(3)BOW 和 SVM

使用 BOW 词袋算法，通过统计每个词在文本中出现的次数，之后可以得到该文本基于词的特征，将各个文本样本的这些词与对应的词频放在一起，这就用词向量表示了这个文本。再使用 SVM 的二分类专属模型 SVC 完成文本分类任务。

得到 F1 分数为 0.87410920000

5. 模型融合

本实验拟用五折交叉验证对同类模型进行融合，分别对三种不同的模型进行五折交叉验证。

同时尝试将三个模型通过投票法进行模型融合，投票法代码如下。

```
result = []
for i in range(len(pred1)):
    result.append(pred1[i] + pred2[i] + pred3[i])

y_test = [1 if x >= 2 else 0 for x in result]
```

最终得到模型融合后的 F1 分数为 0.89447240000

可以看到模型融合的效果远不如 TF-IDF 和 SVM 单个模型的效果，考虑到可能是 TF-IDF 与 XGBoost 没有很好的结合，导致了单个模型效果太差影响了总模型的效果。

6. 实验总结

本项目是基于传统机器学习方法完成的自然语言处理的二分类任务。共构建和使用了三组模型，进行五折交叉验证完成模型融合任务。但在模型融合后发现融合效果不如单个模型的效果，所以在模型融合时应该选择三个评分都较为接近的模型才能融合，否则只会出现拖累的情况。同时本项目使用的两种构建特征工程的方法：TF-IDF 和 BOW，他们在本项目中的适用情况也不同。考虑到本项目作为食品安全预测项目，在数据分析中挖掘的多个特殊关键词起到了决定性作用，而 BOW 难以计算特殊关键词的重要性及在文中的作用，故 TF-IDF 得到了更高的评分。

同时，项目中也尝试过使用 Word2Vec 和 FastText 构建词向量，但在机器学习任务中难以发挥优势。通过本项目，学习到了使用传统机器学习方法解决 NLP 任务的办法，还学习到针对比例不均衡样本进行特殊的概率阈值划分。

实验四

一、实验题目及内容

课后作业（四）2：设计和实现一套《020 商铺食品安全相关评论发现》项目的深度学习解决方案，并在线上进行结果提交（**本题基准参考成绩：0.8833**）

二、技术方案（文字描述+图表说明为主，**不得大段粘贴代码**）、实验结果及分析（技术方案的书写可以适当参考 QQ 群文件中的“R3.技术方案撰写参考文档”）

1. 线上测试成绩（给出最优成绩和线上测评 ID）

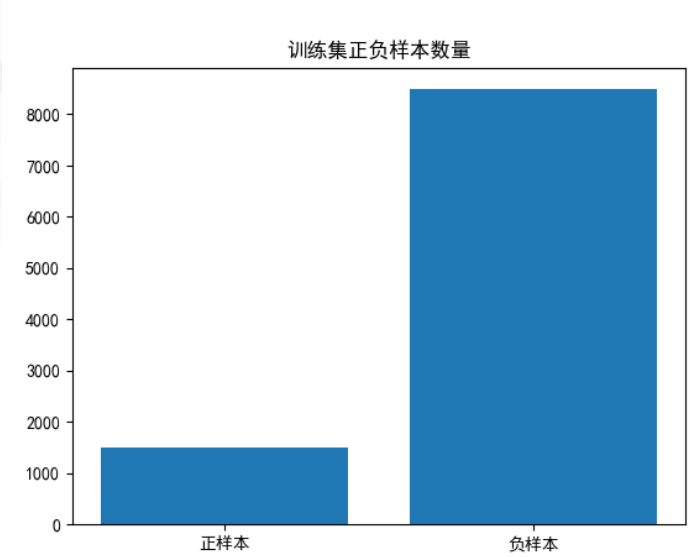
2. 摘要

本次文本分类任务针对外卖评论是否涉及食品安全进行二分类。本次实验在横向对比卷积神经网络、循环神经网络、BiLSTM 模型和调整参数后，选用 BiLSTM+CNN 作为模型。首先利用 word2vec 进行词嵌入将文本训练为词向量，再通过 BiLSTM 模型提取上下文关系，最后通过卷积神经网络进行表征数据学习，同时引入残差学习的概念，添加了残差块，进行文本分类。由于本项目与特殊词语强相关，带有“安全”意味的词语比上下文关系更重要，故在横向对比后，在提取上下文关系后，使用了卷积神经网络，完成了文本二分类任务。

3. 数据分析与数据预处理

3.1 数据可视化分析

(1)对训练集中正样本和负样本做可视化统计，输出如下



可见整个训练集中关于食品安全问题的评论还是占少数。

(2)对训练集中每条正样本词集做交集，观察哪些词语在正样本中出现次数最多。

绘制词云图后发现，上述明显与食品卫生安全强相关的词语导致了标签为正。

(3)判断这些高频食品安全词语在负样本中出现次数

```
cnt = 0
for i in neg['comment']:
    x = ' '.join(jieba.cut(i)).split()
    x = [y for y in x if y not in stop_word]
    ok = 0
    for j in x:
        if j in out:
            ok = 1
            print(j)
    if ok == 1:
        cnt += 1

print('食品安全高频词在负样本中出现%d次'%cnt)
```

```
发现
吐
发现
发现
发现
死
发现
死
食品安全高频词在负样本中出现108次

Process finished with exit code 0
```

可以看到这些高频词在负样本中极少出现，合理确定为特殊词语，在后续导入用户词典的时候应该将高频词加入 jieba 分词词典，这样能提高准确率。

3.2 数据预处理

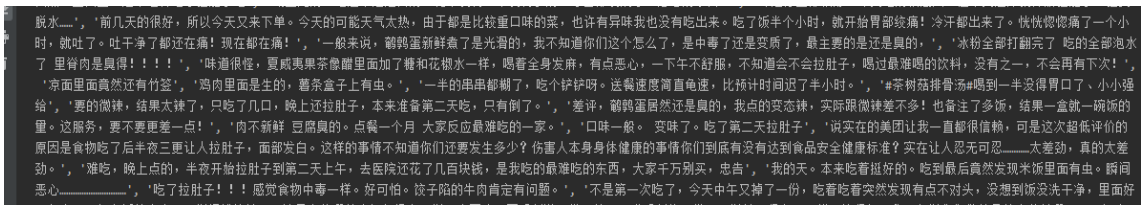
训练集和测试集的 comment 中存在大量表情、颜文字、图片标签、html 标签等无用数据。在调用 filter 函数通过正则匹配对这些文字过滤后，达到了较好的效果。

Filter 函数如下：

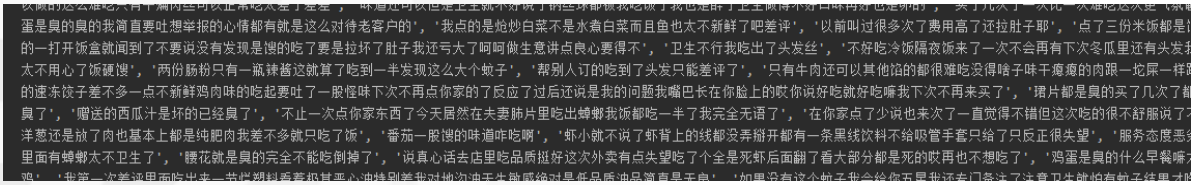
```
def filter(text):
    text = re.sub(r"[A-Za-z0-9\!\=\? \%\[\]\,\ (\) \>\<:&lt;\/#\. -----\_]", "", text)
    text = text.replace('图片', '')
    text = text.replace('\xa0', '')

    cleanr = re.compile('<.*?>')
    text = re.sub(cleanr, ' ', text)
    text = re.sub(r"\\【.*?】+|\\《.*?》+|\\#.*?#+|[.!/_,$&%^*()～<>+?@|:♥@~{}#]+|[—！\\，。=？、：“”‘’￥……（）《》【】]",'',text)
    text = text.strip()
    return text
```

过滤前的文本如下，存在大量标点符号和无用的表情图。



过滤后文本就显得较为干净，适用于 jieba 分词



4. 算法实现与算法优化

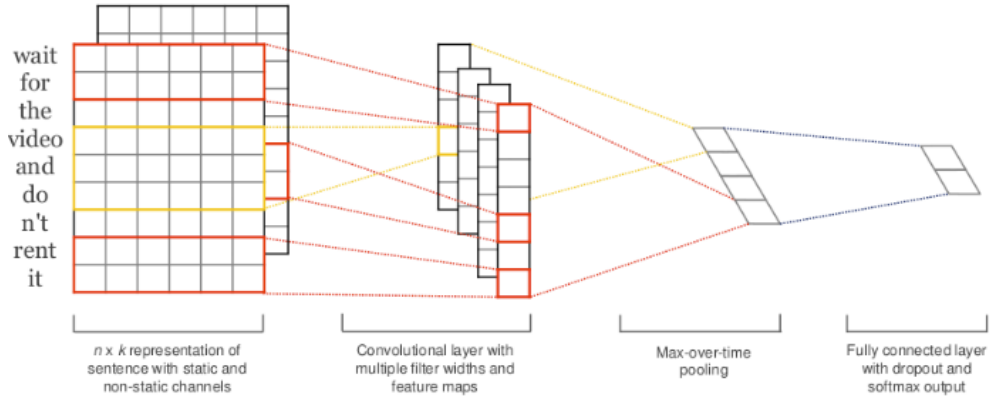
4.1.词向量训练

本实验拟用 Word2Vec 词向量训练模式对文本进行训练，训练维度 128 维。

4.2.模型搭建

分析本项目的具体任务，以及进行数据分析后，发现本文的标签与一些特殊的涉及“食品安全”的特殊词语强相关，比如说“苍蝇”、“头发”等。在经分析后，考虑到特殊词语的重要性，上下文关系就显得没那么重要。而卷积神经网络虽然有强大的表征学习能力，但对上下文关系提取能力不足，但正适用于本项目的情况。故在 BiLSTM 网络结构提取上下文信息后，接上了 CNN 网络。下面进行横向对比实验。

(1)TextCNN:



TextCNN 具体结构图

在通过 TextCNN 对特征进行卷积处理后，通过调整卷积核大小、dropout 的概率、学习率、卷积核个数，线上成绩达到了 **0.90464544000**

(2)RNN 和 LSTM:

在处理循环神经网络和长短时记忆网络中，经过不断的参数调优，也难以发挥到比 TextCNN 更好

的效果。因为本项目的特性决定更适用于 TextCNN，两者线上成绩如下。

result_RNN.csv

所在赛程	状态 / 得分		提交时间
初赛 - A榜	0.89330024000	查看日志	2022/04/13 14:31

备注：无备注信息

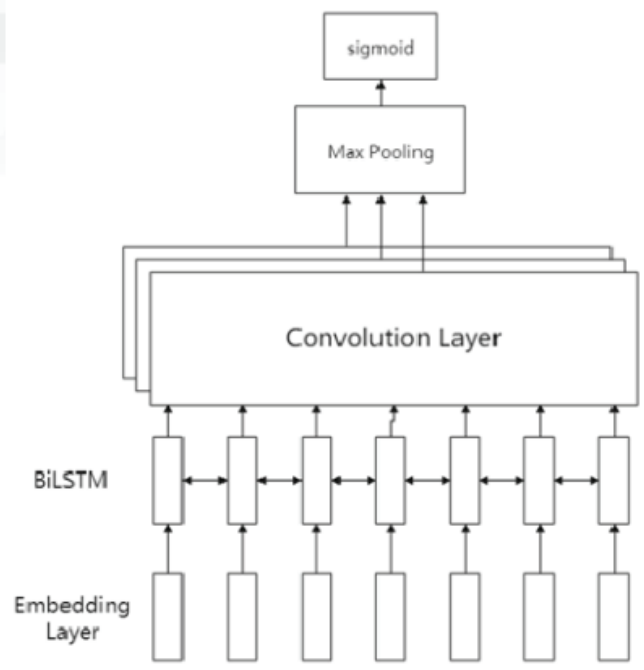
result_LSTM.csv

所在赛程	状态 / 得分		提交时间
初赛 - A榜	0.88395065000	查看日志	2022/04/13 19:20

备注：无备注信息

(3)BiLSTM+CNN

在利用长短时记忆网络结构提取上下文关系后，使用了类似于 TextCNN 的卷积神经网络来进行表征学习。网络具体结构图如下。



其中 BiLSTM 采用了 2 层网络结构，且加入了 Dropout。由于长短时记忆网络能够很好地学习到上下文关系，而卷积神经网络能够很好地将表征学习到。故本网络结构发挥了不错的效果。线上评分如下：

result_LSTM.csv

所在赛程	状态 / 得分		提交时间
初赛 - A榜	0.87439600000	查看日志	2022/04/26 16:06

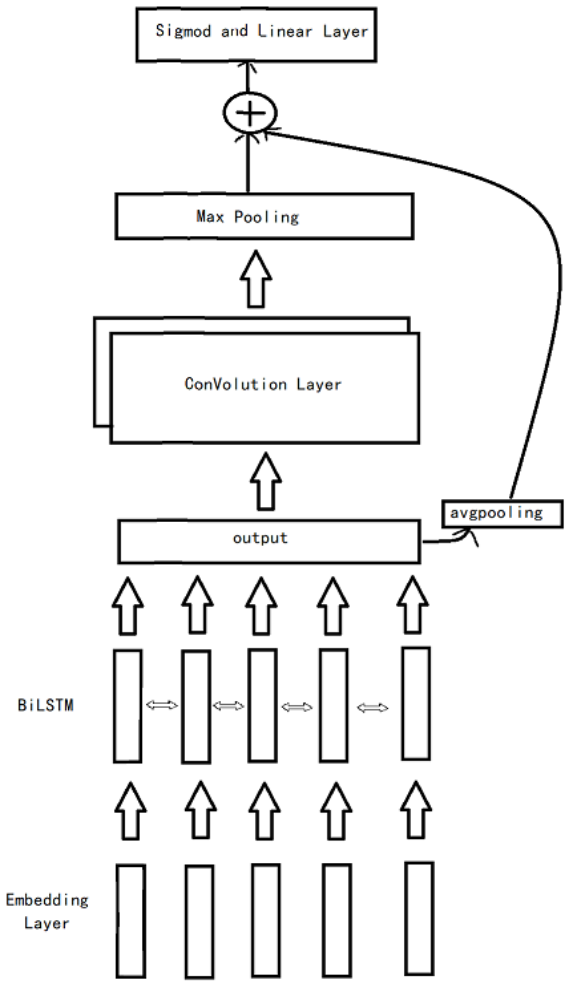
备注：LSTM+TextCNN

最终也选用了这种网络结构。

4.3 模型调优

由上述各网络结构横向对比以及线上评分的情况综合来看，虽然 BiLSTM+TextCNN 网络在理论上是能够发挥更有效的性能的，但线上评分却是最低的。在查阅了资料后，发现很可能是由于多层网络堆叠（4 层 LSTM+卷积层）导致梯度消失或者梯度爆炸而导致该模型的评分略低。故引入残差学习来解决多层网络堆叠导致的梯度相关问题。

在调整网络结构后，新的网络结构如下：



由于在卷积层输出后，数据减少了一个维度，本实验采用了将 BiLSTM 的输出进行平均池化，既保留了数据情况，也降低了维度，以此能和后续卷积层的输出进行残差块相加，做到了加深网络结构的同时，还能防止梯度问题。

同时对学习率，卷积核大小和卷积核个数进行调整，最后完成了参数调优。

5. 模型融合

本实验采用五折交叉验证，将训练样本随机划分成五份，训练出五个不一样的网络，再将预测概率加权平均后四舍五入取得预测结果。

```
#五折交叉训练
for i in range(1, 5):
    print('第%d折训练'%i)
    output = train(i, epoch = 5)
    if len(pred) == 0:
        pred = [x / 5 for x in output]
    else:
        for j in range(len(pred)):
            pred[j] = pred[j] + output[j] / 5

label = []
for i in pred:
    if i >= 0.5:
        label.append(1)
    else:
        label.append(0)
```

最后线上成绩如下：

6. 实验总结

在本次实验中，学习到了几种不同的网络结构，TextCNN、RNN、双向 LSTM 模型各有各的优点和缺点。在本次项目中，TextCNN 的强大表征学习能力很适用于本次项目，但考虑到 TextCNN 并不能很好地解决上下文联系的问题，故本实验尝试在卷积层前加入了长短时记忆网络。之后出现的网络结构堆叠导致梯度问题，虽然最后成绩和单个 TextCNN 进行调优后相差无几。但在本次实验中，我通过搭建一个自己的网络，了解到了网络层之间的关系，以及解决网络层数过多导致的问题。在本次实验中，我学会了 pytorch 的应用，并能用 pytorch 创造一些简单的网络模型。在搭建模型的过程中，才能理解每层网络以及网络结构拼接具体发挥的作用，了解了如何用深度学习网络来处理一个简单的 NLP 文本分类任务。也许在设计自己网络的过程中，有很多错误使用，但这也是一个经验积累的过程。

实验五

一、实验题目及内容

课后作业（六）：将第六章所有任务串联起来，对《网络新闻实体发现与情感识别》项目任务进行预测、调试和优化，并利用测试集计算模型 F1-score 评分(评分计算方法见第六章“2.1 项目介绍”及“2.2 实验任务 32”)。(**本题基准参考成绩：0.40790**)

二、技术方案（文字描述+图表说明为主，**不得大段粘贴代码**）、实验结果及分析（技术方案的书写可以适当参考 QQ 群文件中的“R3.技术方案撰写参考文档”）

1.测试集 F1-score 评分

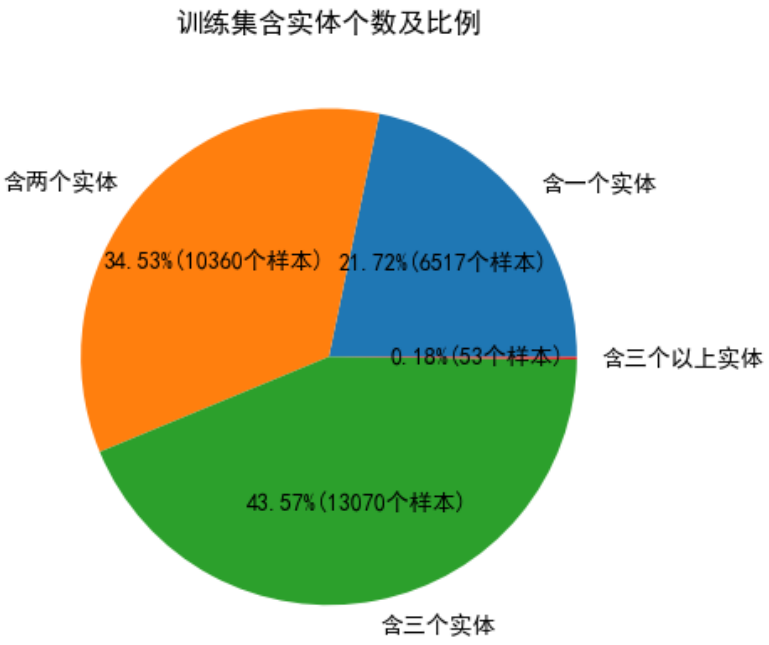
```
entity score =  0.5530688095237317
emotions score =  0.28813206349207293
score =  0.42060043650790235
```

2. 摘要

本次项目抽象为预测文章关键词及关键词情感的任务。在研究 baseline 后，提出了多任务结合的方式。任务一通过分词和词向量预测每个样本的关键词，实质上跑了一个文本二分类任务。任务二在任务一的基础上，对任务一预测出的实体提取关键句。然后将关键句拼接成大长句，用 bert 预训练模型 encode 得到句向量后跑文本三分类任务。同时任务中采用了 jieba 分词改进、关键句中其他实体”<unk>”化等优化措施。相较于 baseline，本方案在任务一中有着较大的提升。

3. 数据分析与数据预处理

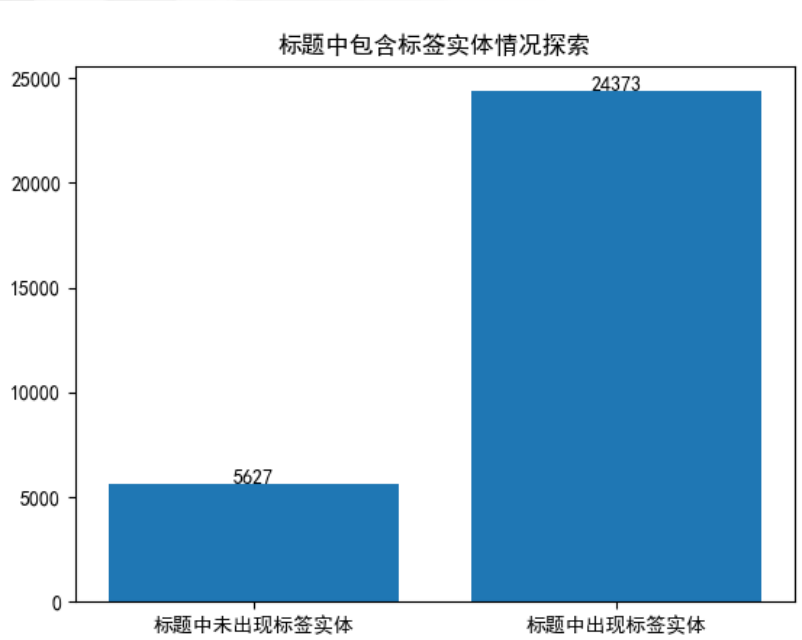
3.1 训练集实体数据探索



可以看到在 30000 条训练集中，99.82%的样本都是包含三个及以下的实体，故在训练数据时，单个样本实体过多不会对模型训练和评测造成影响。

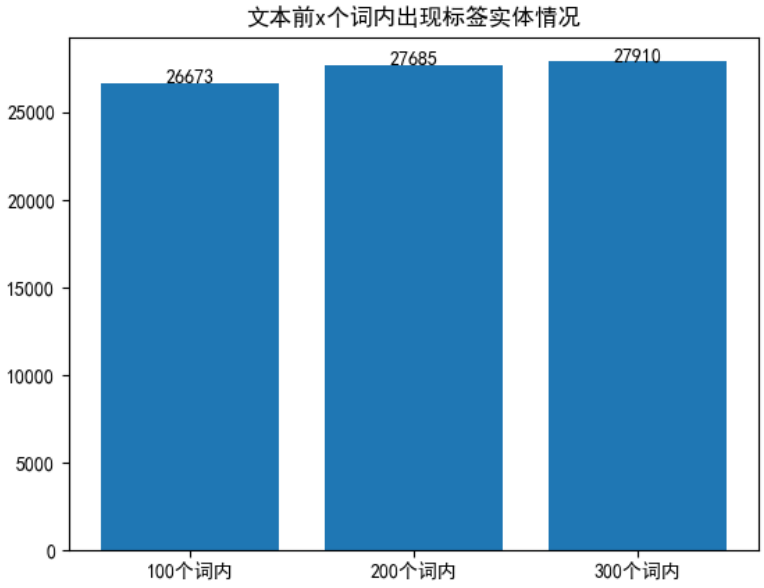
3.2 训练集标签实体数据探索

将训练集的 Title 列清洗并分词后，探索训练集共有多少条样本在标题中就出现了标签实体。



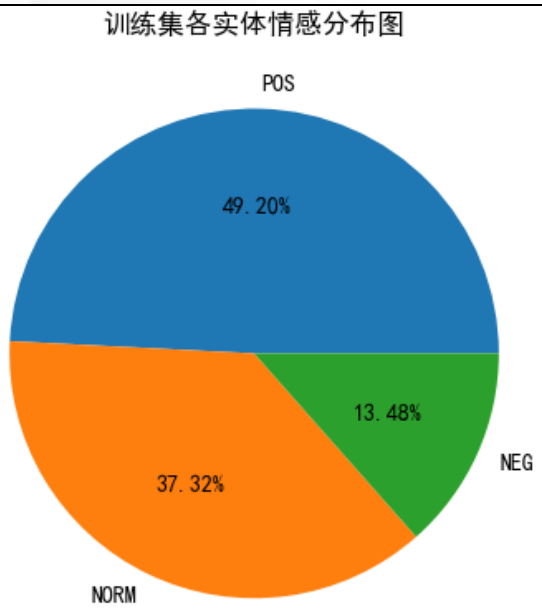
如图所示，标题由于是全文浓缩的精华，出现标签实体的概率更大，接近 5/6 的样本标题中都出现了实体集，所以可以在后续模型训练中，将标题拼接到文章最前列，或者对标题的预测概率加大权重，来更好地预测标签。

接下来探索不同文本长度中，标签实体分布的情况。



可以看到，在文章前 100 个词内有 5/6 的样本出现了标签实体，然而当文章长度逐次增加 100 词时，效果却较为收敛。考虑到新闻文体写作时，会开门见山，将文章重点在开头就展现出来，而后续的文章，都是对其的具体描写。故将文章重点放在前 100 个词即可，后续文章只会徒增样本量、计算量，增加误差。

3.3 训练集实体标签情感数据探索



可以看到三种情况的标签分布不均匀，这对预测结果有一定影响。

3.4 数据预处理

(1)jieba 分词优化

由于本项目任务一要求实体标签命中率，而标签命中与标签本身息息相关，但在模型调试的过程中发现了大量预测实体和真实实体极其接近，甚至只差一个字，这就牵扯到了 jieba 分词时的命中率。

故在预分词阶段，扫描所有标签实体，将其加入 jieba 的用户自定义词典中，即可增加实体命中率。

导出的用户词典如下

```
s = set()
for i in train['coreEntityEmotions']:
    for j in i.keys():
        s.add(j)
```

```
3189 残疾儿童看护者
3190 视频直播平台
3191 新快海涛国际影视文化传媒
3192 苹果折叠屏
3193 区块链
```

(2)文本数据清洗

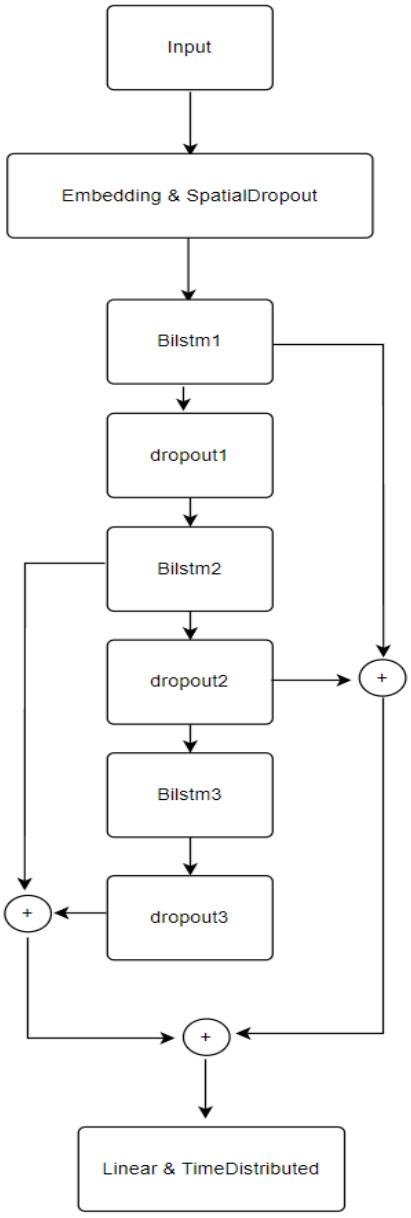
采用 re 库正则表达式去除 html 标签，然后将大写字母转化为小写字母。再导入中文第三方库 longconv，将繁体化为简体后分词。

4. 算法实现与算法优化

4.1 任务一 实体预测

首先将数据清洗和预处理，在 jieba 分词优化后对文本进行分词，将预设 max_len 调为 100，进行 Word2Vec、FastText、Bert 三者词向量训练并拼接，文本序列化后丢入模型训练。在模型训练和预测后，会对每个词都给出一个其为实体词的概率，设定合理阈值取概率前三大或者未达到阈值里最大的词作为当前样本的实体词。最后保存为 pkl 文件，留到任务二继续使用该预测出来的实体词。

任务一网络结构具体如下。



最后得到任务一的实体命中率为 **0.55306**。可以看到相较于 baseline 的实体命中率有较大提升。

4.2 任务二 实体情感识别

在得到任务一的预测实体后，借助 jieba 分词重新分词，且将感叹号、分号、问号等表示句子结束的标点符号换为句号。再按句号划分句子，提取出每句话。

对每个样本的每个实体，分别找到出现过实体的所有关键句子，然后取第一句、最后一句、最中间一句共三句关键句。若是有实体的关键句不足三句，则将关键句的前后句当作关键句。最后将选中的三句关键句拼成一句话，送入 bert 预训练模型 encode 得到句向量。此处将句子的情感当作实体的情感。

在上述处理过程中，出现了一句话中出现了多个实体的情况，如：

Entity : 电影，电视剧

Content: 电影比电视剧好看。

此时就不能以句子情感代替实体情感，因为电影和电视剧在此处明显为 POS 和 NEG，句向量不能代表任何一个实体。

优化方法： 在对电影实体词提取关键句时，将电视剧和其他实体打为‘<unk>’，如：

Content: 电影比<unk>好看。 Entity: 电影 label: POS

Content: <unk>比电视剧好看。 Entity: 电视剧 label: NEG

在处理完后，送入模型训练和预测，模型是一个三分类模型。网络结构是 2 层 BiLstm 交叉接入 dropout。

最终得到情感评分为：0.2888

可以看到其实任务二采用 BiLSTM 网络的效果比较差，主要是在提取关键句后，每个样本都只有一个简单的句向量，而 BiLSTM 的长处是提取上下文关系，故此处改完卷积神经网络更为合适，但限于时间的因素，并没有成功实现卷积神经网络。

4.3 算法优化：

本次项目中主要的优化有 jieba 分词优化，以及各个主要参数的调优。同时对网络结构做了各种尝试，最后在加入残差学习后，实体命中率有效提升。

5. 模型融合

任务一和任务二的模型融合均采用五折交叉验证的方式进行模型融合，且在五折交叉验证的过程中将验证集预测结果记录下来，以做最后的测试集评分统计。

6. 实验总结

在本次任务中，通过学习指导书的 baseline 的整体项目流程后，对整个项目具体怎么做有了一定的了解。同时也尝试使用了多任务的方式来进行本次实验任务。从任务一的分词及网络结构的调整中，学会了一系列优化技巧。在任务二中，提出了一套自己的情感识别的方案，虽然效果并不是非常理想，考虑到任务二中每个样本只有一个句向量，故 BiLSTM 网络结构难以发挥作用，之后可以换成更适合这种情况的 CNN，提取表征信息。同时，任务一也进行了 BIO 的标注方式，尝试了 BiLSTM+CRF 模型做 ner 任务，但由于无法处理 CRF 模型输出的各个实体标准后如何取前三个最有可能的实体的问题，放弃了该方案，这是技术力上的不足，还可以弥补。以此看来，还具有很多需要学习的地方和需要完善的地方。