A I

机器学习课程设计

浙江工业大学

答辩人：童文韬、王子平、钱浩天

目录 /CONTENT

第一部分

课程设计介绍

LOREM IPSUM DOLOR SIT AMET CONSECTETUR ADIPISICING ELIT, SED DO EIUSMOD TEMPOR INCIDIDUNT

所给数据举例展示如下


新
万州烤鱼
特色烤鱼 干锅 中餐


谐和超市


严氏诊所
国家正规医疗机构
社保定点刷卡
入口
出口


友好房产
出售 免费登记 出租

第二部分

研究思路与方法

LOREM IPSUM DOLOR SIT AMET CONSECTETUR ADIPISICING ELIT, SED DO EIUSMOD TEMPOR INCIDIDUNT

研究内容与方法

首先，这类问题有一种专门的机器学习类别，叫场景分类，也就是通过分析图片的图像语义来对图进行分类,例如下面两张图,就可以用场景分类判断出他们是工厂，因为从图像上看他们有很多类似的地方

1

场景分类算法

包括哪些呢?

- 传统的场景分类算法
- 底层特征
- 中层语义
- 高层特征

- 基于学习特征的场景分类
- Alex-Net
- Places-CNN
- DeCAF
- DUCA
- MR-CNNs

但是这个题目用场景分类算法可能不是很好
例如下面两张图

2

基于OCR的店铺分类

包括哪些呢?

种类

- 场景OCR识别:
- FCN
- EAT
- RARE
- CTPN

步骤

- 首先通过OCR模型提取出店铺中的一些关键字
- 然后进行自然语言处理
- 最后通过给定标签进行分类

3

NLP

包括哪些呢?

- NB,SVM
- 深度学习

- 这里NB,SVM更好，容易实现并且由于店铺名称字数少，语义较为简单，没有必要用神经网络,用NB,或者SVM即可

4

设计的总体思路

由之前思路进行总结

①首先就是通过OCR提取出店铺名称等字样

②然后对提取出的文字进行自然语言处理，分析语义，通过给定标签训练自然语言处理模型，并且定义一个词袋

③最后进行测试，观察模型效果

第三部分

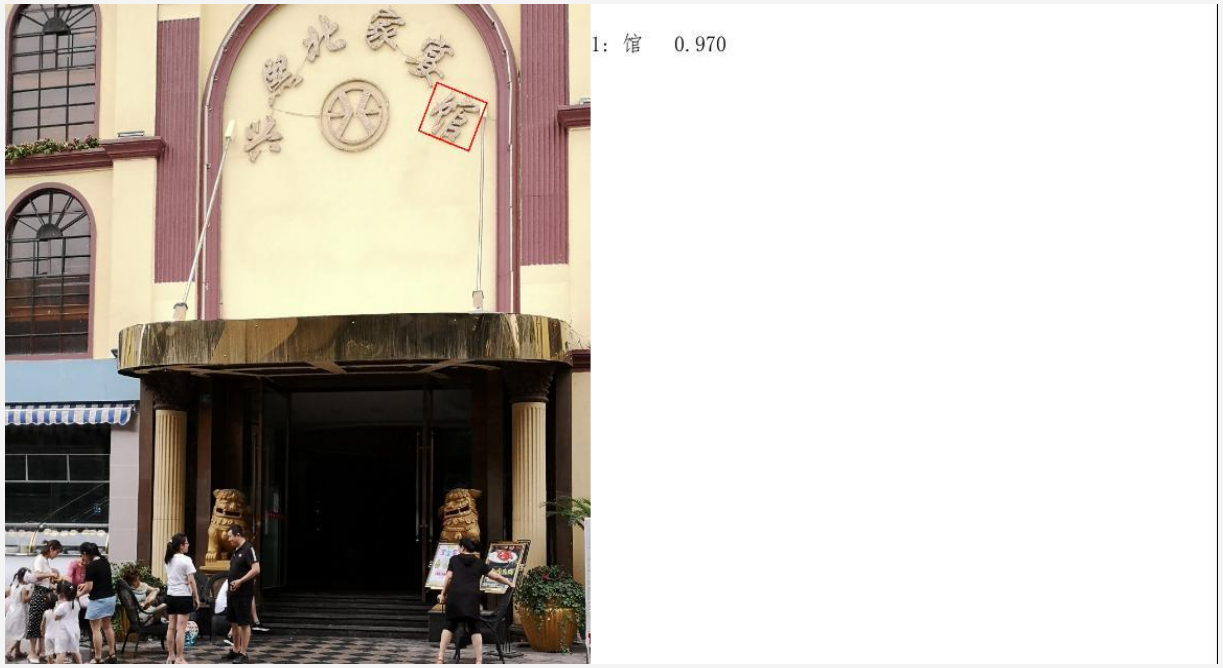
研究步骤与成果

LOREM IPSUM DOLOR SIT AMET CONSECTETUR ADIPISICING ELIT, SED DO EIUSMOD TEMPOR INCIDIDUNT

一.Ocr部分

- 说到Ocr，就会有很多很多的问题,例如下图:(扭曲的文字)

- Ocr的结果:




```
[[[737.0, 127.0], [822.0, 160.0], [791.0, 238.0], [706.0, 205.0]], ('馆', 0.9695794)]
```

又例如:繁体字，目前看来效果不错


```
[[[99.0, 30.0], [639.0, 57.0], [632.0, 209.0], [91.0, 182.0]], ('国银行', 0.9822061)]
[[[692.0, 84.0], [986.0, 88.0], [985.0, 179.0], [691.0, 175.0]], ('BAN', 0.99814075)]
[[[47.0, 591.0], [155.0, 595.0], [155.0, 619.0], [46.0, 616.0]], ('高速不排队', 0.99852836)]
[[[234.0, 600.0], [343.0, 600.0], [343.0, 620.0], [234.0, 620.0]], ('高速不排队', 0.99788415)]
[[[463.0, 609.0], [547.0, 613.0], [545.0, 655.0], [461.0, 650.0]], ('银行', 0.94691837)]
[[[45.0, 629.0], [154.0, 632.0], [154.0, 653.0], [44.0, 650.0]], ('免费送ETC', 0.9977336)]
[[[230.0, 633.0], [344.0, 635.0], [343.0, 655.0], [230.0, 653.0]], ('免费送ETC', 0.99775666)]
[[[416.0, 636.0], [448.0, 636.0], [448.0, 656.0], [416.0, 656.0]], ('免费法E', 0.6785805)]
[[[479.0, 667.0], [539.0, 667.0], [539.0, 681.0], [479.0, 681.0]], ('CHINA', 0.98278254)]
```

有遮挡，而且图像也很模糊

```
[2021/05/23 15:47:14] root INFO: dt_boxes num : 3, elapse : 1.00929856300354
[2021/05/23 15:47:14] root INFO: cls num  : 3, elapse : 0.04886960983276367
[2021/05/23 15:47:14] root INFO: rec_res num  : 3, elapse : 0.07679533958435059


Process finished with exit code 0
```

倾斜文字(目前看来效果不错)


```
[[[508.0, 362.0], [641.0, 377.0], [634.0, 438.0], [501.0, 423.0]], ('米', 0.66151804)]
[[[102.0, 491.0], [444.0, 381.0], [464.0, 441.0], [122.0, 551.0]], ('REDDRAGONFL', 0.98374104)]
[[[303.0, 491.0], [466.0, 448.0], [482.0, 510.0], [319.0, 552.0]], ('红蜻蜓', 0.71889377)]
```

2.自然语言处理模块

这个项目我们初始设定是用SVM

首先要将文本转化成特征向量，贴上标签进行训练

如何转换?

代码展示

研究步骤与成果

首先需要读入文本数据

```
def nlp_with_svm_train():
    x = []
    y = []
    x_canteen=[]
    x_shop=[]
    x_service=[]
    x_others=[]
    file_canteen = open('./result_ocr/canteen.txt', 'r', encoding='gbk')
    lines_canteen = file_canteen.readlines()
    for line in lines_canteen:
        temp = ""
        for db in line.split():
            temp = temp + db + " "
        x_canteen.append(temp)
    num_canteen = len(x_canteen)

    file_shop = open('./result_ocr/shop.txt', 'r', encoding='gbk')
    lines_shop = file_shop.readlines()
```

给数据加标签


```
for i in range(num_canteen):
    y.append(0)

for i in range(num_shop):
    y.append(1)

for i in range(num_service):
    y.append(2)

for i in range(num_others):
    y.append(3)

for element in x_canteen:
    x.append(element)

for element in x_shop:
    x.append(element)

for element in x_service:
    x.append(element)

for element in x_others:
    x.append(element)
```

```
for line in lines_shop:
    temp = ""
    for db in line.split():
        temp = temp + db + " "
    x_shop.append(temp)
num_shop = len(x_shop)
# 得到数据标签
file_service = open('./result_ocr/service.txt', 'r', encoding='gbk')
lines_service = file_service.readlines()
for line in lines_service:
    temp = ""
    for db in line.split():
        temp = temp + db + " "
    x_service.append(temp)
num_service = len(x_service)

file_others = open('./result_ocr/others.txt', 'r', encoding='gbk')
lines_others = file_others.readlines()
for line in lines_others:
    temp = ""
    for db in line.split():
        temp = temp + db + " "
    x_others.append(temp)
num_others = len(x_others)
```

研究步骤与成果

训练的过程

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2)  # 随机划分，训练过程暂时没有使用测试数据
stopword_file = open("./result_ocr/stopword.txt", 'r')  # stopwords.txt是停用词存储所在的文件
stopword_content = stopword_file.read()
stopword_list = stopword_content.splitlines()
stopword_file.close()
count_vect = CountVectorizer(stop_words=stopword_list, token_pattern=r"(?u)\b\w+\b")
train_count = count_vect.fit_transform(x_train)

"""
tf-idf chi特征选择；类似将自然语言转成机器能识别的向量
"""
tfidf_trainformer = TfidfTransformer()
train_tfidf = tfidf_trainformer.fit_transform(train_count)
select = SelectKBest(chi2, k='all')
train_tfidf_chi = select.fit_transform(train_tfidf, y_train)

svc = SVC(C=1.0, kernel='rbf', gamma=0.5)
svc.fit(train_tfidf, y_train)  # 模型训练
print("train accurancy:", svc.score(train_tfidf, y_train))  # important 准确值
train_pre = svc.predict(train_tfidf)  # 预测值（结果内容是识别的具体值）
print(classification_report(train_pre, y_train))  # 输出分类报告（大概就是准确率、召回率）
```

研究步骤与成果


```
def test_model():
    x_test= []
    file1_test = open("./result_ocr/test.txt", 'r',
encoding='gbk')
    lines_test = file1_test.readlines()
    for line in lines_test:
        temp = ""
        for db in line.split():
            temp = temp + db + " "
        x_test.append(temp)
    num_test = len(x_test)
```

```
    with open('svm.pickle', 'rb') as svm:
        svm1 = pickle.load(svm)

    with open('count_vect.pickle', 'rb') as count_vect:
        count_vect1 = pickle.load(count_vect)

    with open('tfidf_trainformer.pickle', 'rb') as tfidf_trainformer:
        tfidf_trainformer1 = pickle.load(tfidf_trainformer)

    """
    停用词处理等
    """
    test_count = count_vect1.transform(x_test)

    """
    特征选择
    """
    test_tfidf = tfidf_trainformer1.transform(test_count)
nlp_with_svm_train()
```

研究步骤与成果

支持向量机训练结果
(高斯核):

train accurancy: 0.9575471698113207

	precision	recall	f1-score	support
-2	0.22	1.00	0.36	20
-1	1.00	1.00	1.00	504
1	1.00	0.92	0.96	541
2	1.00	0.95	0.97	631
accuracy			0.96	1696
macro avg	0.80	0.97	0.82	1696
weighted avg	0.99	0.96	0.97	1696

支持向量机训练结果
(线性核):

train accurancy: 1.0

	precision	recall	f1-score	support
-2	1.00	1.00	1.00	85
-1	1.00	1.00	1.00	482
1	1.00	1.00	1.00	519
2	1.00	1.00	1.00	610
accuracy			1.00	1696
macro avg	1.00	1.00	1.00	1696
weighted avg	1.00	1.00	1.00	1696

实验表明SVM采用线性核函数的效果远远好于采用高斯核函数

进一步采用神经网络算法进行实验

```
Training and evaluating...
Epoch: 1
Iter:      0,  Train Loss:    1.4,  Train Acc:   3.12%,  Val Loss:    1.4,  Val Acc:  17.65%,  Time: 0:00:01 *
Epoch: 2
Iter:    100,  Train Loss:    1.4,  Train Acc:  34.38%,  Val Loss:    1.5,  Val Acc:  20.59%,  Time: 0:00:05 *
Epoch: 3
Epoch: 4
Iter:    200,  Train Loss:    1.1,  Train Acc:  28.12%,  Val Loss:    1.5,  Val Acc:  29.41%,  Time: 0:00:08 *
Epoch: 5
Iter:    300,  Train Loss:    1.3,  Train Acc:  34.38%,  Val Loss:    1.4,  Val Acc:  20.59%,  Time: 0:00:12
Epoch: 6
Epoch: 7
Iter:    400,  Train Loss:    1.2,  Train Acc:  40.62%,  Val Loss:    1.5,  Val Acc:  20.59%,  Time: 0:00:15
Epoch: 8
Iter:    500,  Train Loss:    1.2,  Train Acc:  46.88%,  Val Loss:    1.5,  Val Acc:  20.59%,  Time: 0:00:18
Epoch: 9
Epoch: 10
Iter:    600,  Train Loss:    1.4,  Train Acc:  37.50%,  Val Loss:    1.5,  Val Acc:  20.59%,  Time: 0:00:20
Epoch: 11
Iter:    700,  Train Loss:    1.1,  Train Acc:  50.00%,  Val Loss:    1.5,  Val Acc:  20.59%,  Time: 0:00:23
Epoch: 12
Epoch: 13
Iter:    800,  Train Loss:    1.2,  Train Acc:  37.50%,  Val Loss:    1.5,  Val Acc:  20.59%,  Time: 0:00:26
Epoch: 14
Iter:    900,  Train Loss:    1.2,  Train Acc:  43.75%,  Val Loss:    1.5,  Val Acc:  20.59%,  Time: 0:00:28
Epoch: 15
Epoch: 16
Iter:   1000,  Train Loss:    1.3,  Train Acc:  37.50%,  Val Loss:    1.5,  Val Acc:  20.59%,  Time: 0:00:31
Epoch: 17
Iter:   1100,  Train Loss:    1.2,  Train Acc:  43.75%,  Val Loss:    1.5,  Val Acc:  20.59%,  Time: 0:00:33
Epoch: 18
Epoch: 19
Iter:   1200,  Train Loss:    1.3,  Train Acc:  34.38%,  Val Loss:    1.5,  Val Acc:  20.59%,  Time: 0:00:36
No optimization for a long time, auto-stopping...

C:\Users\TWT\Desktop\text-classification-cnn-master>
```

其实可以很明显地看到，数据出现了过拟合的现象

word2vec使用：缺乏数据集

神经网络缺乏数据集就会产生很严重的过拟合现象,并且word2vec如果缺数据的话会导致很多词语他不知道是什么有意思

实验结果并不是很好

太缺数据集了

卷积神经网络的训练结果:

实验结果并不是很好

最后还是决定用支持向量机

并且对数据进行了一定的预处理,用jieba库对文本进行分词
实验结果如上

总排行　历史周（Top3）　大屏排行榜NEW

17	--	赛队9b583	0.7075	21	2021-06-22 08:42
1	--	无法识别	0.9016	1	2021-05-08 23:20
2	--	SiegeLions	0.8884	13	2021-05-09 22:45
3	--	摸式识别	0.8884	5	2021-05-09 06:32
4	--	赛队f3c4b	0.8634	8	2021-06-21 16:58
5	--	DeeperMind（王雨钰 孙博文）	0.8458	16	2021-05-10 11:28
6	--	Capcom	0.8442	8	2021-05-09 13:38
7	--	SYS队	0.8284	10	2021-05-09 23:57
8	--	77rising	0.8087	9	2021-05-10 10:18
9	--	爬藤雷克格尼生提姆4	0.8079	4	2021-05-09 23:49
10	--	赛队95fe7	0.8064	1	2021-05-08 14:31

THANK YOU

恳请各位同学及老师批评指正