文档归类项目开题报告

1. 项目背景

文档分类，是将一个文档分配到一个或者多个类别中。它可以通过人工分类完成，也可以通过计算机算法实现。

文档可以根据主题进行分类，也可以根据其它属性（如文档类型、作者、出版时间等）进行分类。根据主题进行分类主要有两种方法：基于内容的方法和基于请求的方法。

基于内容的分类方法是通过特殊主题上的不同权重来决定该文档被分到哪个类别中的。一般来说，在图书馆中，当一个文档被划分到某个类别时，这个文档中至少要有 20%的内容是关于这个类的。在自动分类的领域，这个标准可能是一些给定单词在文档中出现的频率。[1]

文档自动分类的任务可以分为三类：监督式学习的文档分类，这需要人工反馈数据的一些外在机制。非监督式学习的文档分类（也被称作文档聚类），这类任务完全不依靠外在人工机制。和半监督式学习的文档分类，是前两类的结合，它其中有一部分的文档是由人工标注的，这有一些相关方面的具有许可证的软件 [2] 。

2. 问题描述

本项目需要解决的问题是通过运用机器学习和自然语言处理技术，寻找一种能自动进行新闻文档分类的算法。训练数据和测试数据均有标签，属于监督学习，需要将文档分为 20 个类别，为多分类任务。

3. 输入数据

本项目的输入数据是 20 种新闻数据集。该数据集大约有 18000 条新闻，比较均衡地分成了 20 类，是比较常用的文本数据之一，既可以从官方网站下载，也可利用 sklearn 工具包下载。

新闻类别如下：

'alt.atheism',

'comp.graphics',

'comp.os.ms-windows.misc',

'comp.sys.ibm.pc.hardware',

'comp.sys.mac.hardware',

'comp.windows.x',

'misc.forsale',

'rec.autos',

'rec.motorcycles',

'rec.sport.baseball',

'rec.sport.hockey',

'sci.crypt',

'sci.electronics',

'sci.med',

'sci.space',

'soc.religion.christian',

'talk.politics.guns',

'talk.politics.mideast',

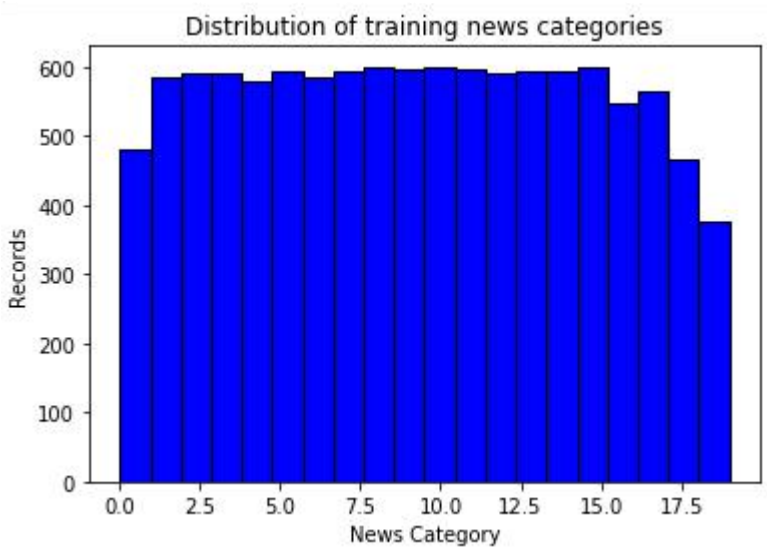
'talk.politics.misc',

'talk.religion.misc'

训练集中各个类别的记录数如下：

[(10, 600), (15, 599), (8, 598), (9, 597), (11, 595), (7, 594), (13, 594), (5, 593), (14, 593), (2, 591), (12, 591), (3, 590), (6, 585), (1, 584), (4, 578), (17, 564), (16, 546), (0, 480), (18, 465), (19, 377)]

分布如下图，可以看出分布比较均匀。



抽样展示一条新闻记录样本，如下：

From: lerxst@wam.umd.edu (where's my thing)

Subject: WHAT car is this!?

Nntp-Posting-Host: rac3.wam.umd.edu

Organization: University of Maryland, College Park

Lines: 15

I was wondering if anyone out there could enlighten me on this car I saw the other day. It was a 2-door sports car, looked to be from the late 60s/ early 70s. It was called a Bricklin. The doors were really small. In addition, the front bumper was separate from the rest of the body. This is all I know. If anyone can tellme a model name, engine specs, years of production, where this car is made, history, or whatever info you have on this funky looking car, please e-mail.

Thanks,

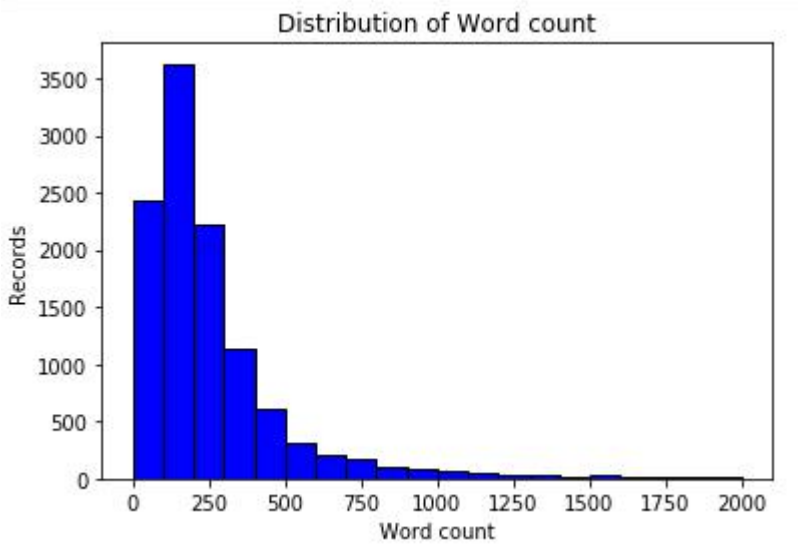
- IL

---- brought to you by your neighborhood Lerxst ----

最长新闻的单词数： 20235

最短新闻的单词数： 9

新闻的长度分布如下：



从图可以看出，大部分新闻长度在 500 个单词以内。

频度最高的 10 个单词：

[('', 660661), ('the', 115263), ('to', 64169), ('of', 61504), ('a', 52312), ('and', 47081), ('is', 36661), ('in', 34707), ('that', 31310), ('I', 31184)]

4. 解决办法

本项目将分别使用决策树模型、朴素贝叶斯模型和一维卷积神经网络算法，对数据集分别进行训练，并根据评估指标找出最优算法。

5. 基准模型

斯坦福大学的一个团队在该数据集上取得的正确率在 0.85 左右，本项目的基准为在测试集上验证的正确率 0.85。

6. 评估指标

本项目是一个多分类问题，这里采用正确率 Accuracy 作为评估指标。正确率等于被分对的样本数除以所有的样本数，正确率越高，分类器越好。

$$Accuracy = \frac{\sum_{i=1}^{n} I(y_i = \hat{y}_i)}{n}$$

y_i 表示第 i 个样本的标签，$\hat{y}_i$ 表示第 i 个样本的预测值。当样本标签和预测值相等时，I 等于 1，否则等于 0。

7. 设计大纲

数据预处理：

去掉标点、空格、数字， - 等特殊字符；

将非专有名词都转换成小写字母；

词向量的训练方式：

采用 Word2Vec 词向量，分别使用

1，尝试通过自己训练；

2，采用 Google 预训练的 300 维新闻语料的词向量 googlenews-vecctors-negative300.bin

最后比较训练结果，采用正确率高的方式。

7.1. 导入并探索数据

```
from sklearn.datasets import fetch_20newsgroups
from sklearn.naive_bayes import MultinomialNB
from sklearn.tree import DecisionTreeClassifier
from sklearn import metrics
from sklearn.feature_extraction.text import TfidfVectorizer
from pprint import pprint
from collections import Counter
import matplotlib.pyplot as plt
import nltk

# 导入数据集
newsgroups_train = fetch_20newsgroups(subset='train')
newsgroups_test = fetch_20newsgroups(subset='test')
```

```
# 探索数据集
# 打印训练集新闻类别
print("训练集新闻类别：")
pprint(list(newsgroups_train.target_names))
# 打印测试集新闻类别
# print("测试集新闻类别：")
# pprint(list(newsgroups_train.target_names))
# 训练集的记录数
print("训练集新闻记录数：",newsgroups_train.filenames.shape[0])
# 测试集的记录数
print("测试集新闻记录数：",newsgroups_test.filenames.shape[0])
# 训练集前 10 条记录的类别
print("训练集前 10 条记录的类别：",newsgroups_train.target[:10])
# 统计训练集各个类别的数据量
category_counter = Counter(newsgroups_train.target)
print(category_counter.most_common(len(newsgroups_train.target_names)))

plt.hist(newsgroups_train.target, bins=len(newsgroups_train.target_names),
         facecolor="blue", edgecolor="black")
# 显示横轴标签
plt.xlabel("News Category")
# 显示纵轴标签
plt.ylabel("Records")
# 显示图标题
plt.title("Distribution of training news categories")
plt.show()

# 打印一条训练记录样本
print(newsgroups_train.data[0])
```

```
# 词频统计
dictionary = {}
word_count = {}
for i, text in enumerate(newsgroups_train.data):
    fredist = nltk.FreqDist(text.split(" "))
    for localkey in fredist.keys():
        if localkey in dictionary.keys(): # 检查当前词频是否在字典中存在
            dictionary[localkey] = dictionary[localkey] + fredist[localkey] # 如果存在，将词频累加，并更新字典值
        else: # 如果字典中不存在
            dictionary[localkey] = fredist[localkey] # 将当前词频添加到字典中
    word_count[i]=fredist.N()
print(sorted(dictionary.items(), key = lambda x:x[1], reverse = True)[:10]) # 根据词频字典值排序，并打印

print("最长新闻的单词数：", max(word_count.values()))
print("最短新闻的单词数：", min(word_count.values()))

# 新闻的长度分布
plt.hist(word_count.values(), bins=20,
         facecolor="blue", edgecolor="black", range=(0,2000))
# 显示横轴标签
plt.xlabel("Word count")
# 显示纵轴标签
plt.ylabel("Records")
# 显示图标题
plt.title("Distribution of Word count")
plt.show()
```

```
# 向量化
vectorizer = TfidfVectorizer()
vectors_train = vectorizer.fit_transform(newsgroups_train.data)
vectors_test = vectorizer.transform(newsgroups_test.data)
print(vectors_train.shape)
```

7.2. 实现决策树算法，并评估模型

```
clf_DecisionTree = DecisionTreeClassifier(random_state=0)
clf_DecisionTree.fit(vectors_train, newsgroups_train.target)
pred_DecisionTree = clf_DecisionTree.predict(vectors_test)
accuracy_DecisionTree = metrics.accuracy_score(newsgroups_test.target,
pred_DecisionTree)
print("决策树正确率：", accuracy_DecisionTree)
```

7.3. 实现朴素贝叶斯算法，并评估模型

```
clf_MultinomialNB = MultinomialNB(alpha=.01)
clf_MultinomialNB.fit(vectors_train, newsgroups_train.target)
pred_MultinomialNB = clf_MultinomialNB.predict(vectors_test)
accuracy_MultinomialNB = metrics.accuracy_score(newsgroups_test.target,
pred_MultinomialNB)
print("朴素贝叶斯正确率：", accuracy_MultinomialNB)
```

7.4. 实现一维卷积神经网络算法，并评估模型

```
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.utils import to_categorical
from keras.layers import Dense, Flatten, Dropout
```

```
from keras.layers import Conv1D, MaxPooling1D, Embedding
from keras.models import Sequential
import numpy as np

MAX_SEQUENCE_LENGTH = 100
EMBEDDING_DIM = 200

tokenizer = Tokenizer()
tokenizer.fit_on_texts(newsgroups_train.data)
sequences = tokenizer.texts_to_sequences(newsgroups_train.data)
word_index = tokenizer.word_index
word_index_length = len(word_index)
print('Found %s unique tokens.' % word_index_length)
data = pad_sequences(sequences, maxlen=MAX_SEQUENCE_LENGTH)
labels = to_categorical(np.asarray(newsgroups_train.target))
print('Shape of data tensor:', data.shape)
print('Shape of label tensor:', labels.shape)

model = Sequential()
model.add(Embedding(len(word_index) + 1, EMBEDDING_DIM,
input_length=MAX_SEQUENCE_LENGTH))
model.add(Dropout(0.2))
model.add(Conv1D(200, 3, padding='valid', activation='relu', strides=1))
model.add(MaxPooling1D(3))
model.add(Flatten())
model.add(Dense(EMBEDDING_DIM, activation='relu'))
model.add(Dense(labels.shape[1], activation='softmax'))
model.summary()
model.compile(loss='categorical_crossentropy',
              optimizer='adam',
```

```
                    metrics=['acc'])
print(model.metrics_names)
model.fit(data, labels, epochs=2, batch_size=128)


# 验证
tokenizer_test = Tokenizer()
tokenizer_test.fit_on_texts(newsgroups_test.data)
sequences_test = tokenizer_test.texts_to_sequences(newsgroups_test.data)
x_test = pad_sequences(sequences_test, maxlen=MAX_SEQUENCE_LENGTH)
y_test = to_categorical(np.asarray(newsgroups_test.target))
print(model.evaluate(x_test, y_test))
```

7.5. 比较各个模型结果，总结。

引用

[1]. https://en.wikipedia.org/wiki/Document_classification

[2].https://baike.baidu.com/item/%E6%96%87%E6%A1%A3%E5%88%86%E7%B1%BB/22776999?fr=aladdin

[3]. Library of Congress (2008). The subject headings manual. Washington, DC.: Library of Congress, Policy and Standards Division.

[4]. https://nlp.stanford.edu/wiki/Software/Classifier/20_Newsgroups