

# NLP文本分类 新闻分类

一娃的成长 一娃的成长 4月16日

记一次中文新闻文本分类的实验记录，本实验主要是对以下几个网页内容的整合及结果复现：

[https://www.sohu.com/a/165903757\\_176628](https://www.sohu.com/a/165903757_176628)

<https://baijiahao.baidu.com/s?id=1641172818761365604&wfr=spider&for=pc>

## 0. 实验环境

代码环境：Python 3.6

分词工具：jieba

框架工具：sklearn gensim

## 1. 数据准备

### 1.1 训练及测试数据

采用开源的清华新闻数据集 THUCNews，数据下载链接  
<http://thuctc.thunlp.org/message>

本次实验只处理'财经','股票','科技','社会','游戏'五个类别的部分数据，其中每个类别前1000条作为训练数据，后200条作为测试数据。

### 1.2 停用词文件

[https://github.com/foowaa/Chinese\\_from\\_dongxiexidian](https://github.com/foowaa/Chinese_from_dongxiexidian)

### 1.3 数据提取与清洗

prepare\_data.py

```
1  # encoding: utf-8
2  import os
3  import jieba
4  from multiprocessing import Process, Queue
5  import multiprocessing
6
7  def strQ2B(ustring):
8      """全角转半角"""
9      rstring = ""
10     for uchar in ustring:
11         inside_code=ord(uchar)
```

```
12         if inside_code == 12288: #全角空格直接转
13             inside_code = 32
14         elif (inside_code >= 65281 and inside_code <= 65374): #全角字符 (除空
15             inside_code -= 65248
16         rstring += chr(inside_code)
17     return rstring
18 # datadir include category subdir
19 # return tuple list [(filepath, category)]
20 def get_data_tuples(datadir, categories, operation, cnt_per_type):
21     tuples = []
22     for _, dirs, _ in os.walk(datadir):
23         for d in dirs:
24             if d in categories:
25                 files = os.listdir(datadir+'/'+d)
26                 if operation == 'train':
27                     for f in files[:cnt_per_type]:
28                         f = datadir+'/'+d+'/'+f
29                         tuples.append((f,d))
30                 if operation == 'test':
31                     for f in files[-cnt_per_type-1:-1]:
32                         f = datadir + '/' + d + '/' + f
33                         tuples.append((f,d))
34
35     return tuples
36
37 def get_stop_words(fpath):
38     ignore_words= []
39     with open(fpath, 'r', encoding='utf-8') as fstop:
40         for l in fstop.readlines():
41             l = l.replace('\n', '')
42             ignore_words.append(l)
43
44     return ignore_words
45
46 def cleanup_sentence(q, ftuples, stop_words):
47     for f, k in ftuples:
48         with open(f, 'r', encoding='utf-8') as fr:
49             text = fr.read().lower()
50             text = text.replace("\t", " ").replace("\n", " ")
51             seg_text = jieba.cut(text)
```

```
52         segout = " ".join(seg_text)
53         words = segout.split()
54         outline = [] # " ".join(outline.split())
55         for i in words:
56             if i not in stop_words:
57                 outline.append(i)
58         outline = " ".join(outline) + "\t__label__" + k + "\n"
59         q.put(outline)
60         # print("segment file: %s" % f)
61         # print(outline)
62         # break
63
64 if __name__ == '__main__':
65     stop_words = get_stop_words('./stopwords.dat')
66
67     train_categories = ['财经', '股票', '科技', '社会', '游戏']
68     test_categories = train_categories
69
70     train_tuples = get_data_tuples('../data/THUCNews', train_categories, 'train')
71     test_tuples = get_data_tuples('../data/THUCNews', test_categories, 'test')
72
73     q = Queue()
74     procs = []
75     proc_cnt = multiprocessing.cpu_count()
76
77     if len(train_tuples) <= proc_cnt:
78         proc_cnt = len(train_tuples)
79     task_piece = len(train_tuples) / proc_cnt
80     task_reserve = len(train_tuples) % proc_cnt
81
82     e = 0
83     for i in range(multiprocessing.cpu_count()):
84         b = e
85         e += task_piece
86         if i < task_reserve:
87             e += 1
88         p = Process(target=cleanup_sentence, args=(q, train_tuples[int(b):int(e)]))
89         p.start()
90         procs.append(p)
91
```

```
92     with open("news_train.txt", "w", encoding='utf-8') as ftrain:
93         i = 0
94         while i < len(train_tuples):
95             line = q.get()
96             ftrain.write(line)
97             #ftrain.flush()
98             i += 1
99
100     for t in procs:
101         t.join()
102     procs.clear()
103
104     if len(test_tuples) <= proc_cnt:
105         proc_cnt = len(test_tuples)
106     task_piece = len(test_tuples) / proc_cnt
107     task_reserve = len(test_tuples) % proc_cnt
108
109     e = 0
110     for i in range(multiprocessing.cpu_count()):
111         b = e
112         e += task_piece
113         if i < task_reserve:
114             e += 1
115         p = Process(target=cleanup_sentence, args=(q, test_tuples[int(b):int(e)]))
116         p.start()
117         procs.append(p)
118
119     with open("news_test.txt", "w", encoding='utf-8') as f:
120         i = 0
121         while i < len(test_tuples):
122             line = q.get()
123             f.write(line)
124             i += 1
125
126     for t in procs:
127         t.join()
128     procs.clear()
129
```

## 2. 训练&测试

### 2.1 训练过程

此过程以两种向量空间模型Doc2Vec和Tf-Idf为例，分别使用 LogisticRegression RandomForestClassifier XGBoost 三种机器学习算法进行对比测试。

transformers.py

```
1  # encoding: utf-8
2
3  from gensim.models.doc2vec import Doc2Vec, TaggedDocument
4  from sklearn.base import BaseEstimator
5  from sklearn import utils as skl_utils
6  from tqdm import tqdm
7  import multiprocessing
8  import numpy as np
9
10 class Doc2VecTransformer(BaseEstimator):
11     def __init__(self, vector_size=100, learning_rate=0.02, epochs=5):
12         self.learning_rate = learning_rate
13         self.epochs = epochs
14         self._model = None
15         self.vector_size = vector_size
16         self.workers = multiprocessing.cpu_count() - 1
17
18     def fit(self, df_x, df_y=None):
19         tagged_x = [TaggedDocument(row.split(), [index]) for index, row in df_x.iterrows()]
20         model = Doc2Vec(documents=tagged_x, vector_size=self.vector_size, workers=self.workers)
21         for epoch in range(self.epochs):
22             model.train(skl_utils.shuffle([x for x in tqdm(tagged_x)]), total_epochs=self.epochs)
23             model.alpha -= self.learning_rate
24             model.min_alpha = model.alpha
25             self._model = model
26
27         return self
28
29     def transform(self, df_x):
30         return np.asmatrix(np.array([self._model.infer_vector(x.split()) for x in df_x.iterrows()]))
31
```

```
32
33 # tf-idf
34 from sklearn.feature_extraction.text import TfidfVectorizer
35
36 class Text2TfIdfTransformer(BaseEstimator):
37     def __init__(self):
38         self._model = TfidfVectorizer()
39         pass
40
41     def fit(self, df_x, df_y=None):
42         # df_x = df_x.apply(lambda x : clean_text(x))
43         self._model.fit(df_x)
44
45         return self
46
47     def transform(self, df_x):
48         return self._model.transform(df_x)
49
50
51 from sklearn.pipeline import Pipeline
52 from sklearn.linear_model import LogisticRegression
53 from sklearn.model_selection import cross_val_score
54
55
56 if __name__ == "__main__":
57     in_x = []
58     in_y = []
59     with open('news_train.txt', 'r', encoding='utf-8') as f:
60         for l in f.readlines():
61             l = l.replace('\n', '')
62             in_x.append(l.split('\t')[0])
63             in_y.append(l.split('\t')[1].replace('__label__', ''))
64
65     # doc2vec_trf = Doc2VecTransformer()
66     # doc2vec_features = doc2vec_trf.fit(in_x).transform(in_x)
67     # print(doc2vec_features)
68     #
69     # pl_log_reg = Pipeline(steps=[('doc2vec', Doc2VecTransformer()),
70     #     ('log_reg', LogisticRegression(multi_class='auto', solver='liblinear'))
71     # ])
```

```
72 # scores = cross_val_score(pl_log_reg, in_x, in_y, cv=5, scoring='accuracy')
73 # print('Accuracy for Logistic Regression: ', scores.mean())
74
75 tfidf_transformer = Text2TfidfTransformer()
76 tfidf_vectors = tfidf_transformer.fit(in_x).transform(in_x)
77 print("tf-idf vector shape: ", tfidf_vectors.shape)
78
79 pl_log_reg_tfidf = Pipeline(steps=[('tfidf', Text2TfidfTransformer()),
80                                   ('log_reg', LogisticRegression(multi_class='auto', solver='liblinear'))]
81                                )
82 scores = cross_val_score(pl_log_reg_tfidf, in_x, in_y, cv=5, scoring='accuracy')
83 print('Accuracy for Tf-Idf & Logistic Regression: ', scores.mean())
```

### classify\_cmp.py

```
1 # encoding: utf-8
2
3 from transformers import *
4
5 from sklearn.pipeline import Pipeline
6 from sklearn.linear_model import LogisticRegression
7 from sklearn.model_selection import cross_val_score
8 from sklearn.ensemble import RandomForestClassifier
9 from sklearn.metrics import accuracy_score
10 import xgboost as xgb
11 import random
12
13
14 class EmptyTransformer():
15     def __init__(self):
16         pass
17
18     def fit(self, df_x, df_y=None):
19         # print("do fit ...")
20         return self
21
22     def transform(self, in_x):
23         # print("do transform ...")
24         return in_x
```

```
25
26
27 if __name__ == "__main__":
28     in_x = []
29     in_y = []
30     with open('news_train.txt','r',encoding='utf-8') as f:
31         for l in f.readlines():
32             l = l.replace('\n','')
33             in_x.append(l.split('\t')[0])
34             in_y.append(l.split('\t')[1].replace('__label__', ''))
35
36     randnum = random.randint(0, 100)
37     random.seed(randnum)
38     random.shuffle(in_x)
39     random.seed(randnum)
40     random.shuffle(in_y)
41
42     doc2vec_trf = Doc2VecTransformer()
43     doc2vec_features = doc2vec_trf.fit(in_x).transform(in_x)
44     # print(doc2vec_features)
45     print("doc2vec vector shape: ", doc2vec_features.shape)
46
47     # pl_log_reg = Pipeline(steps=[('doc2vec',Doc2VecTransformer()),
48     #     ('log_reg', LogisticRegression(multi_class='auto', solver='liblinear'))]
49     #
50     # scores = cross_val_score(pl_log_reg, in_x, in_y, cv=5, scoring='accuracy')
51     # print('Accuracy for Logistic Regression: ', scores.mean())
52
53     classes = list(set(in_y))
54     label_y = []
55     output_empty = [0] * len(classes)
56     for y in in_y:
57         output_row= list(output_empty)
58         output_row[classes.index(y)]= 1
59         label_y.append(output_row)
60
61     test_x = []
62     test_y = []
63     with open('news_test.txt','r',encoding='utf-8') as f:
64         for l in f.readlines():
```



```
65         l = l.replace('\n', '')
66         test_x.append(l.split('\t')[0])
67         test_y.append(l.split('\t')[1].replace('__label__', ''))
68
69     randnum = random.randint(0, 100)
70     random.seed(randnum)
71     random.shuffle(test_x)
72     random.seed(randnum)
73     random.shuffle(test_y)
74
75     test_doc2vec_features = doc2vec_trf.transform(test_x)
76     print("test_doc2vec vector shape: ", test_doc2vec_features.shape)
77
78     test_label_y = []
79     for y in test_y:
80         output_row= list(output_empty)
81         output_row[classes.index(y)]= 1
82         test_label_y.append(output_row)
83
84     pl_log_reg = Pipeline(steps=[('doc2vec', EmptyTransformer()),
85                                  ('log_reg', LogisticRegression(multi_class='auto', solver='liblinear'))]
86     scores = cross_val_score(pl_log_reg, doc2vec_features, in_y, cv=5, scoring='accuracy')
87     print('Accuracy for Logistic Regression Classifier : ', scores.mean())
88
89     pl_random_forest = Pipeline(steps=[('doc2vec', EmptyTransformer()),
90                                         ('random_forest', RandomForestClassifier())])
91     scores = cross_val_score(pl_random_forest, doc2vec_features, in_y, cv=5, scoring='accuracy')
92     print('Accuracy for RandomForest Classifier : ', scores.mean())
93
94     pl_xgb = Pipeline(steps=[('doc2vec', EmptyTransformer()),
95                               ('xgb_boost', xgb.XGBClassifier(objective='binary:logistic'))])
96     scores = cross_val_score(pl_xgb, doc2vec_features, in_y, cv=5, scoring='accuracy')
97     print('Accuracy for XGBoost Classifier : ', scores.mean())
98
99     tfidf_transformer = Text2TfIdfTransformer()
100    tfidf_vectors = tfidf_transformer.fit(in_x).transform(in_x)
101    print("tf-idf vector shape: ", tfidf_vectors.shape)
102
103    test_tfidf_features = tfidf_transformer.transform(test_x)
104    print("test_tf-idf vector shape: ", test_tfidf_features.shape)
```

```
105
106 pl_log_reg_tf_idf = Pipeline(steps=[('tfidf', EmptyTransformer()),
107     ('log_reg', LogisticRegression(multi_class='auto', solver='liblinear'))]
108 scores = cross_val_score(pl_log_reg_tf_idf, tfidf_vectors, in_y, cv=5, scoring='accuracy')
109 print('Accuracy for Tf-Idf & Logistic Regression: ', scores.mean())
110
111 pl_random_forest_tf_idf = Pipeline(steps=[('tfidf', EmptyTransformer()),
112     ('random_forest', RandomForestClassifier())]
113 scores = cross_val_score(pl_random_forest_tf_idf, tfidf_vectors, in_y, cv=5, scoring='accuracy')
114 print('Accuracy for Tf-Idf & RandomForest : ', scores.mean())
115
116 pl_xgb_tf_idf = Pipeline(steps=[('tfidf', EmptyTransformer()),
117     ('xgboost', xgb.XGBClassifier(objective='binary'))]
118 scores = cross_val_score(pl_xgb_tf_idf, tfidf_vectors, in_y, cv=5, scoring='accuracy')
119 print('Accuracy for Tf-Idf & XGBoost Classifier : ', scores.mean())
```

## 2.2 测试结果

```
1 doc2vec vector shape: (5000, 100)
2 test_doc2vec vector shape: (1000, 100)
3 Accuracy for Logistic Regression Classifier : 0.6442
4 Accuracy for RandomForest Classifier : 0.6432
5 Accuracy for XGBoost Classifier : 0.6522
6 tf-idf vector shape: (5000, 104198)
7 test_tf-idf vector shape: (1000, 104198)
8 Accuracy for Tf-Idf & Logistic Regression: 0.9064
9 Accuracy for Tf-Idf & RandomForest : 0.8942
10 Accuracy for Tf-Idf & XGBoost Classifier : 0.9086
```

## 3. 结果分析

实验结束，引用原文的结果分析：

尽管在自然语言处理中，“DocVec”模型比“Tf-Idf”模型更高级，但我们的案例证明，后者效果更佳。我们分别使用了基于线性、袋状以及推进型的分类器。

原因可以这么理解。在我们的数据集中，每一个“文本”领域包含了一些决定其类别的高频单词/标记。因此，应用一个对语境/上下文敏感模型会使问题更为复杂、（或者）混淆信息。某些文本类别包含一些高频出现的标记，这些标记提供了大量数值以定义“Tf-Idf”模型。

同时，“文本”是细分领域的。比如，“布莱尔 (blair)”一词更可能出现在“政治”类别，而非“运动”类别。因此，像这样的词对“Tf-Idf”模型起了作用。

而且，“Doc2Vec”模型更适合应用于语法正确的文本中。大量案例和数据科学家的实验证明，虽然“Tf-Idf”模型次于“DocVec”模型，但前者对于细分领域的文本分类更为有效。

实验代码GitHub地址：

<https://github.com/cddypang/nlp-learning>