特征统计：CountVectorizer+ 分类器：RidgeClassifier 数据：15000 f1分数：0.74

In [2]:

```python
import pandas as pd

from sklearn.feature_extraction.text import CountVectorizer
from sklearn.linear_model import RidgeClassifier
from sklearn.metrics import f1_score

train_df = pd.read_csv('./data/train_set.csv', sep='\t', nrows=15000)

vectorizer = CountVectorizer(max_features=3000)
train_test = vectorizer.fit_transform(train_df['text'])

clf = RidgeClassifier()
clf.fit(train_test[:10000], train_df['label'].values[:10000])

val_pred = clf.predict(train_test[10000:])
print(f1_score(train_df['label'].values[10000:], val_pred, average='macro'))
```

```
0.741494277019762
```

特征统计：TfidfVectorizer（3元+10000特征数）+ 分类器：RidgeClassifier 数据：200000 f1分数：0.87

In [1]:

```python
import pandas as pd

from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import RidgeClassifier
from sklearn.metrics import f1_score

train_df = pd.read_csv('./data/train_set.csv', sep='\t')

tfidf = TfidfVectorizer(ngram_range=(1,3), max_features=3000)
train_test = tfidf.fit_transform(train_df['text'])

clf = RidgeClassifier()
clf.fit(train_test[:10000], train_df['label'].values[:10000])

val_pred = clf.predict(train_test[10000:])
print(f1_score(train_df['label'].values[10000:], val_pred, average='macro'))
```

```
0.8744524667530537
```

特征统计：TfidfVectorizer（3元+10000特征数）+ 分类器：RidgeClassifier 数据：200000 f1分数：0.89

In [2]:

```python
import pandas as pd

from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import RidgeClassifier
from sklearn.metrics import f1_score

train_df = pd.read_csv('./data/train_set.csv', sep='\t')

tfidf = TfidfVectorizer(ngram_range=(1,3), max_features=3000)
train_test = tfidf.fit_transform(train_df['text'])

clf = RidgeClassifier()
clf.fit(train_test[:150000], train_df['label'].values[:150000])

val_pred = clf.predict(train_test[150000:])
print(f1_score(train_df['label'].values[150000:], val_pred, average='macro'))
```

```
0.8945911867541981
```

```
特征统计：TfidfVectorizer（2元+10000特征数）+ 分类器：LogisticRegression（c=4） 数据：200000 f1分数：0.938
```

In [2]:

```python
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import f1_score
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split

train_df = pd.read_csv('./data/train_set.csv', sep='\t')

# vectorizer = CountVectorizer(max_features=3000)
tfidf =TfidfVectorizer(
    sublinear_tf=True,
    strip_accents='unicode',
    ngram_range=(1, 2),
    max_features=10000)
train_val = tfidf.fit_transform(train_df['text'])

x_train = train_val
y_train = train_df['label']

# 可以改变输入维度
x_train_, x_valid_, y_train_, y_valid_ = train_test_split(x_train, y_train, test_size=0.2)


lr= LogisticRegression(C=4, n_jobs=8)
lr.fit(x_train_, y_train_)

val_pred_lr = lr.predict(x_valid_)
print(f1_score(y_valid_, val_pred_lr , average='macro'))
```

```
D:\ProgramData\anaconda\lib\site-packages\sklearn\linear_model\logistic.py:433: Futu
reWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a solver to si
lence this warning.
  FutureWarning)
D:\ProgramData\anaconda\lib\site-packages\sklearn\linear_model\logistic.py:460: Futu
reWarning: Default multi_class will be changed to 'auto' in 0.22. Specify the multi_
class option to silence this warning.
  "this warning.", FutureWarning)
D:\ProgramData\anaconda\lib\site-packages\sklearn\linear_model\logistic.py:1297: Use
rWarning: 'n_jobs' > 1 does not have any effect when 'solver' is set to 'liblinear'.
Got 'n_jobs' = 8.
  " = {}.".format(effective_n_jobs(self.n_jobs)))

0.9385955581526331
```

In [3]:

```python
test_df = pd.read_csv('./data/test_a.csv', sep='\t')
test_=tfidf.transform(test_df['text'])
```

In [4]:

```python
test_pred_lr=lr.predict(test_)
test_pred_lr=pd.DataFrame(test_pred_lr)
test_pred_lr.columns=['label']

```

In [5]:

```
test_pred_lr.to_csv('./output/test_a_pred_lr3.csv', index=None, encoding='utf8')
```

In [3]:

```
from joblib import dump, load
```

In [4]:

```
path='./model/language_detector.model'
dump((clf, tfidf), path)
```

Out[4]:

```
['./model/language_detector.model']
```

In [6]:

```
classifier, vectorizer = load(path)
```