1 准备数据

In [1]:

```
# 选取 科技、汽车、娱乐、军事、运动
import jieba
import pandas as pd

df_technology = pd.read_csv("./origin_data/technology_news.csv", encoding='utf-8')

df_technology = df_technology.dropna()

df_car = pd.read_csv("./origin_data/car_news.csv", encoding='utf-8')
df_car = df_car.dropna()

df_entertainment = pd.read_csv("./origin_data/entertainment_news.csv", encoding='utf-8')
df_entertainment = df_entertainment.dropna()

df_military = pd.read_csv("./origin_data/military_news.csv", encoding='utf-8')
df_military = df_military.dropna()

df_sports = pd.read_csv("./origin_data/sports_news.csv", encoding='utf-8')
df_sports = df_sports.dropna()

df_society= pd.read_csv("./origin_data/society_news.csv", encoding='utf-8')
df_society= df_sports.dropna()
```

In [2]:

```
# 每类数据取10000条
technology = df_technology.content.values.tolist()[0:10000]
car = df_car.content.values.tolist()[0:10000]
entertainment = df_entertainment.content.values.tolist()[0:10000]
military = df_military.content.values.tolist()[0:10000]
sports = df_sports.content.values.tolist()[0:10000]
society = df_society.content.values.tolist()[0:10000]
```

In [3]:

```
' '.join(jieba.lcut(technology[0]))
```

```
Building prefix dict from the default dictionary ...
Loading model from cache C:\Users\LIUJIA~1\AppData\Local\Temp\jieba.cache
Loading model cost 0.719 seconds.
Prefix dict has been built successfully.
```

Out[3]:

```
'\u3000 \u3000 , 中新网 , 1 月 7 日电 \xa0  恰逢 CES  2017 拉开 大幕 ,  却 惊闻 “ AlphaGo 升级版 ” 的 Master 迎来 60 连胜 ,  人类 顶尖 围棋 手 在 一周 内 纷纷 败给 这个 谷歌 旗下 DeepMind 团队 打造 的 “ 围棋 大脑 ” ,  显然 也 为 聚焦 于 人工智能 的 本届 CES 增添 了 声势 。 而 首次 参展 ,  并 致力于 打造 “ 原创 AI 大脑 ” 的 中国 深度 学习 领军 企业 的 商汤 科技 ,  在 人工智能 的 浪潮 之巅 ,  及 众多 业界 前辈 和 巨匠 面前 ,  将会 交出 一份 怎样 的 答卷 呢 ？'
```

2 去停用词

In [4]:

```
# 加载停用词转为numpy
stopwords=pd.read_csv("origin_data/stopwords.txt",index_col=False,quoting=3,sep="\t",names=['s
stopwords=stopwords['stopword'].values  #numpy
```

In [5]:

```
# 定义一个类,去停用词
sentences=[] #存放所有的数据
# sentences_train=[]
# sentences_test=[]
def preprocess_texts(target_path,catergary,texts):
    with open(target_path + catergary + '.txt', 'w' ) as f:
        for text in texts:
            words=jieba.lcut(text)
            words=list(filter(lambda x:len(x)>1, words))
            words=list(filter(lambda x:  x not in stopwords, words))
            if len(words):
                sentences.append((' '.join(words),catergary))
                f.write(' '.join(words)+'\n')
        print(len(sentences))
#     sentences_train+=sentences[:7000]
#     sentences_test+=sentences[:3000]
```

In [6]:

```
preprocess_texts ('./processed_data/', 'technology' ,technology  )
preprocess_texts ('./processed_data/', 'car' ,car )
preprocess_texts ('./processed_data/', 'entertainment' ,entertainment  )
preprocess_texts ('./processed_data/', 'military ' ,military   )
preprocess_texts ('./processed_data/', 'sports ' ,sports  )
preprocess_texts ('./processed_data/', 'society' ,society )
```

```
9954
19898
29841
39768
49692
59616
```

In [300]:

```
# sentences_train=[]
# sentences_test=[]
# for i in range(5):
#     sentences_train+=sentences[i*10000:i*10000+7000]
#     sentences_test+=sentences[7000+i*10000:(i+1)*10000]
```

In [7]:

```
len(sentences)
```

Out[7]:

```
59616
```

In [8]:

```
for i in sentences:
    if len(i[0])==0:  print( "false")
```

In [9]:

```
import random
random.shuffle(sentences)
```

3 划分数据集

In [10]:

```
from sklearn.model_selection import train_test_split
x_train_, y_train_ = zip(*sentences)
# x_test, y_test = zip(*sentences_test)
# print(len(x_train))
x_train, x_val, y_train, y_val= train_test_split(x_train_, y_train_, random_state=2020)
```

4 训练和预测数据

In [11]:

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from gensim.models import Word2Vec
vec = CountVectorizer(
    analyzer='word',      # 以词为单位
    max_features=4000,  # 选取4000个主要词构建预料
)
tfv = TfidfVectorizer(lowercase=True, decode_error='ignore')
# w2v = Word2Vec(x_train, size=300, window=5, iter=15, workers=12, seed=2020)

tfv.fit(x_train)
vec.fit(x_train)
```

Out[11]:

```
CountVectorizer(analyzer='word', binary=False, decode_error='strict',
                dtype=<class 'numpy.int64'>, encoding='utf-8', input='content',
                lowercase=True, max_df=1.0, max_features=4000, min_df=1,
                ngram_range=(1, 1), preprocessor=None, stop_words=None,
                strip_accents=None, token_pattern='(?u)\\b\\w\\w+\\b',
                tokenizer=None, vocabulary=None)
```

In [12]:

```
# one-hot预测分数
from sklearn.naive_bayes import MultinomialNB
classifier = MultinomialNB()
classifier.fit(vec.transform(x_train), y_train)
classifier.score(vec.transform(x_val), y_val)
```

Out[12]:

```
0.6565351583467526
```

In [26]:

```
# tf-idf预测分数
from sklearn.naive_bayes import MultinomialNB
classifier = MultinomialNB()
classifier.fit(tfv.transform(x_train), y_train)
# classifier.score(tfv.transform(x_train), y_train)
classifier.score(tfv.transform(x_val), y_val)
```

Out[26]:

0.8236491322240115

In [14]:

```
# 构建word2vec模型
from gensim.models import Word2Vec
w2v = Word2Vec(x_train_, size=60, iter=10, seed=2020,min_count=5 )
```

In [15]:

```
import numpy as np
x_train_w2v=np.zeros((len(x_train),60))
x_train_w2v_ave=np.zeros((len(x_train),60))
i=0
for words in list(x_train):
    n=0
    for word in words:
        if word in w2v:
            n+=1
            vect=w2v[word]
            x_train_w2v[i,:]+=vect
    x_train_w2v_ave[i,:]=x_train_w2v[i,:]/n
    i+=1
```

```
D:\ProgramData\anaconda\envs\tf2.0\lib\site-packages\ipykernel_launcher.py:8: Deprec
ationWarning: Call to deprecated `__contains__` (Method will be removed in 4.0.0, us
e self.wv.__contains__() instead).

D:\ProgramData\anaconda\envs\tf2.0\lib\site-packages\ipykernel_launcher.py:10: Depre
cationWarning: Call to deprecated `__getitem__` (Method will be removed in 4.0.0, us
e self.wv.__getitem__() instead).
  # Remove the CWD from sys.path while we load stuff.
```

In [16]:

```
x_val_w2v=np.zeros((len(x_val),60))
x_val_w2v_ave=np.zeros((len(x_val),60))
i=0
for words in list(x_val):
    if len(words)==0: print( i,"false")
    n=0
    for word in words:
        if word in w2v:
            n+=1
            vect=w2v[word]
            x_train_w2v[i,:]+=vect
#     if n==0: print("false")
    x_val_w2v_ave[i,:]=x_train_w2v[i,:]/n
    i+=1
```

```
D:\ProgramData\anaconda\envs\tf2.0\lib\site-packages\ipykernel_launcher.py:8: Deprec
ationWarning: Call to deprecated `__contains__` (Method will be removed in 4.0.0, us
e self.wv.__contains__() instead).

D:\ProgramData\anaconda\envs\tf2.0\lib\site-packages\ipykernel_launcher.py:10: Depre
cationWarning: Call to deprecated `__getitem__` (Method will be removed in 4.0.0, us
e self.wv.__getitem__() instead).
  # Remove the CWD from sys.path while we load stuff.
```

In [17]:

```
# x_train_df=pd.DataFrame(x_train_w2v_ave)
# y_train_df=pd.DataFrame(y_train)
# x_val_df=pd.DataFrame(x_val_w2v_ave)
# y_val_df=pd.DataFrame(y_val)
```

In [18]:

```
# from sklearn.preprocessing import LabelEncoder
# labelEncoder = LabelEncoder()
# y_train_label = labelEncoder.fit_transform(y_train_df)
# y_train[0:10]
# y_train_label=pd.DataFrame(y_train_label)
# y_train_label
```

In [19]:

```
from sklearn.linear_model import LogisticRegression
classifier =  LogisticRegression()
classifier.fit(x_train_w2v_ave, y_train)
classifier.score(x_val_w2v_ave, y_val)
```

Out[19]:

```
0.42706655931293613
```

In [20]:

```
from sklearn.naive_bayes import  GaussianNB
classifier =  GaussianNB()
classifier.fit(x_train_w2v_ave, y_train)
classifier.score(x_val_w2v_ave, y_val)
```

Out[20]:

0.3882850241545894

In [21]:

```
#
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import accuracy_score, precision_score
import numpy as np

def stratifiedkfold_cv(x, y, clf_class, shuffle=True, n_folds=5, **kwargs):
    stratifiedk_fold = StratifiedKFold(n_splits=n_folds, shuffle=shuffle)
    y_pred = y[:]
    for train_index, test_index in stratifiedk_fold.split(x, y):
        X_train, X_test = x[train_index], x[test_index]
        y_train = y[train_index]
        clf = clf_class(**kwargs)
        clf.fit(X_train,y_train)
        y_pred[test_index] = clf.predict(X_test)
    return y_pred
```

In [35]:

```
x_train_, y_train_ = zip(*sentences)
# x_train_=x_train_[:-1]
# y_train_=y_train_[:-1]
# len(x_train_)
```

In [36]:

```
# one-hot交叉验证
NB = MultinomialNB
print(precision_score(y_train_, stratifiedkfold_cv(vec.transform(x_train_),np.array(y_train_),N
```

0.6830925308820235

In [38]:

```
# tfidf交叉验证
NB = MultinomialNB
print(precision_score(y_train_, stratifiedkfold_cv(tfv.transform(x_train_),np.array(y_train_),N
```

0.7269982223098465

In [47]:

```
# w2v交叉验证
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score
LR = LogisticRegression()

score_ndarray = cross_val_score(LR,x_train_w2v_ave, np.array(y_train), cv=5)
print(score_ndarray.mean())
# print(precision_score(y_train, stratifiedkfold_cv( x_train_w2v_ave, np.array(y_train), LR),
```

```
0.6345501404018736
```