作业五 相似度计算
任务描述：采用word2vec方法，进行句子相似度计算训练。

给出一个有关句子相似度的二分类数据集msr_paraphrase（包含train、test、README三个文件），其中第一列数字1代表相似，0代表不相似。
选择文件train中的string1&2部分作为训练语料，选择文件test计算句子相似度，然后与标注结果比较，输出你认为合适的分类阈值，以及该阈值下的准确率Accuracy，精确率Precision，召回率Recall和F1值（精确到小数点后两位）。

句向量相似度计算方式：
首先对句子分词，获取每个词的词向量，然后将所有的词向量相加求平均，得到句子向量，最后计算两个句子向量的余弦值(余弦相似度)。

Word2vec部分，使用Gensim的Word2Vec训练模型，自行调整合适的参数。
注意可能会出现word xx not in vocabulary的情况，这是由于训练时默认参数min_count=5，会过滤掉低频词。可以手动设定min_count=1，或者在计算句向量时，遇到低频词则忽略。自行选择其一，并注释。
gensim镜像安装方式
pip install -i https://pypi.tuna.tsinghua.edu.cn/simple (https://pypi.tuna.tsinghua.edu.cn/simple) gensim
导入方式from gensim.models import word2vec

一、函数定义&导包

1. $getTokenized(sentence)->list$:输入一个句子，返回其标准化之后的分词列表
2. $getSentenceVector(tokenization, trainModel)->np.ndarray$:输入一个句子的分词列表、训练模型, 返回它的词向量
3. $getCosineSimilarity(vector1, vector2)->float$:输入两个句子的句向量，返回他们的余弦相似度

In [68]:

```
import csv
import nltk
import re
import numpy as np
import nltk.tokenize as tk
from nltk.corpus import stopwords
from gensim.models import Word2Vec
from scipy.linalg import norm
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

#函数1：将句子转换成标准规范化的分词形式
def getTokenized(sentence:str)->list:
    pattern=re.compile("[^a-zA-Z0-9\n ]")              #数字字符的正则匹配
    sentence = re.sub(pattern,"",sentence).lower()     #将所有非数字字符的符号转化为空，大小写转换
    Tokenization = tk.word_tokenize(sentence)          #分词，标记化
    Tokenization = [w for w in Tokenization if(w not in stopwords.words('english'))]  #去停用词
    return Tokenization


#函数2: 输入一个句子的分词列表、训练模型，返回它的句子向量
def getSentenceVector(tokenization:list, trainModel:Word2Vec)->np.ndarray:
    vector = np.zeros(100)
    for word in tokenization:
        vector += trainModel[word]    #trainModel即为该单词的词向量
    if len(tokenization) != 0:
        vector /= len(tokenization)       #求平均值，得到句子向量
    return vector

#函数3:  输入两个句子的句向量，返回他们的余弦相似度
def getCosineSimilarity(vector1:np.ndarray, vector2:np.ndarray)->float:
    # np.dot :向量点乘
    # norm:   向量的模
    if norm(vector1) == 0 and norm(vector2) == 0:
        return 1
    elif norm(vector1) == 0 or norm(vector2) == 0:
        return 0
    return np.dot(vector1, vector2) / (norm(vector1) * norm(vector2))
```

二、数据读取&分词处理

- 调用系统的csv读取文本,然后转换成list存储在data中
- 如果open的时候不注明编码方式为utf-8,编译器会抛出一个gbk decode异常
- WordsList1[i]表示第i行的string1的分词列表，WordsList2同理
- 设置了部分提示信息来表示程序的运行进度

In [21]:

```
#data[i][0]-data[i][4]分别表示第i项数据的 编号,ID1,ID2,字符串1，字符串2
data = open("msr_train.csv","r", encoding="utf-8")    #读取全部数据
data = list(csv.reader(data))[1:]                     #去除表头
WordsList1 = []
WordsList2 = []

print("正在获取分词:")
cmp = len(data)//10
for i in range (len(data)):
    if i == cmp:
        print("    进度:%d%%"%(cmp*100//len(data)))
        cmp += len(data)//10
    WordsList1.append(getTokenized(data[i][3]))
    WordsList2.append(getTokenized(data[i][4]))
print("计算分词已完成")
```

```
正在获取分词:
    进度:9%
    进度:19%
    进度:29%
    进度:39%
    进度:49%
    进度:59%
    进度:69%
    进度:79%
    进度:89%
    进度:99%
计算分词已完成
```

三、Word2Vec模型训练

- WordsList为传入训练参数
- min_count是对词进行过滤，频率小于min-count的单词则会被忽视
- sg=1是skip-gram算法，对低频词敏感
- window是句子中当前词与目标词之间的最大距离，3表示在目标词前看3-b个词，后面看b个词（b在0-3之间随机）。
- size是输出词向量的维数，值太小会导致词映射因为冲突而影响结果，值太大则会耗内存并使算法计算变慢
- negative和sample可根据训练结果进行微调，sample表示更高频率的词被随机下采样到所设置的阈值，默认值为1e-3。
- hs=1表示层级softmax将会被使用，默认hs=0且negative不为0，则负采样将会被选择使用。

In [74]:

```
Model1 = Word2Vec(WordsList1, min_count= 1, sg = 1, window = 5, size = 100)
Model2 = Word2Vec(WordsList2, min_count= 1, sg = 1, window = 5, size = 100)
print("Word2Vec模型训练已完成")
```

```
Word2Vec模型训练已完成
```

四、计算阈值

- 采用全部句子余弦相似度的平均值作为阈值

In [75]:

```
AllSimilarity = []
sumValue = 0
for i in range(len(data)):
    vector1 = getSentenceVector(WordsList1[i],Model1)
    vector2 = getSentenceVector(WordsList2[i],Model2)
    CosineSimilarity = getCosineSimilarity(vector1,vector2)
    AllSimilarity.append(CosineSimilarity)
    sumValue += CosineSimilarity
averageValue = sumValue / len(AllSimilarity)
print("设置阈值为:",averageValue)
```

```
<ipython-input-68-fdbd135ebef5>:24: DeprecationWarning: Call to deprecated `__getite
m__` (Method will be removed in 4.0.0, use self.wv.__getitem__() instead).
  vector += trainModel[word]    #trainModel即为该单词的词向量
```

设置阈值为: 0.9279506618798747

五、与标注结果对比

- StandardResult为msr_train.csv中的标准结果
- MyResult为训练的结果

In [76]:

```
StandardResult = [int(e[0]) for e in data]
MyResult = []
for value in AllSimilarity:
    if value >= averageValue:
        MyResult.append(1)
    else:
        MyResult.append(0)

print("准确率Accuracy: %.2f"%accuracy_score(StandardResult, MyResult))
print("精确率Precision: %.2f"%precision_score(StandardResult, MyResult))
print("召回率Recall: %.2f"%recall_score(StandardResult, MyResult))
print("F1值: %.2f"%f1_score(StandardResult, MyResult))
```

准确率Accuracy: 0.68
精确率Precision: 0.68
召回率Recall: 0.99
F1值: 0.81