**Gensim常见用法：**①Word2Vec训练词向量②生成主题向量、词典③计算TF-IDF④文本相似度分析

**语料库预处理：**

一行一个文档或句子，将文档或句子分词（以空格分割，英文可以不用分词，英文单词之间已经由空格分割，中文预料需要使用分词工具进行分词

（或创建一个list，其中的元素为每篇文本的字符串，利用jieba对list中的每篇文本进行分词并去除停用词，生成一个以各篇文本分词结果集为元素的list）

文本相似度分析

**制作词袋:**

from gensim import corpora

dictionary=corpora.dictionary(list)

**生成主题向量：**

theme\_vector=dictionary.doc2bow(某篇文本的分词结果)

#将语料库中文章的主题向量组合为list,就得到了**主题向量库:**

corpus=[dictionary.doc2bow(text for text in textwarehouse)]

**查看每个词对应id：**

dictionary.token2id

**生成主题向量库潜在语义索引（LSI）：**

lsi=models.LsiModel(主题向量库)

**索引映射：**lsi[主题向量库]

**计算输入文本与主题向量库中各向量相似度：**

lsi[某篇文本的主题向量]

计算文本中各词的TF-IDF值

[**https://www.baidu.com/link?url=3X-VcSzkt6OEZs6R6Bez3s19YVf19T0-S20-cqi45YiEqM2-Ac1xOmMFrsWenJJz&wd=&eqid=a1b856350003831f000000035f21313f**](https://www.baidu.com/link?url=3X-VcSzkt6OEZs6R6Bez3s19YVf19T0-S20-cqi45YiEqM2-Ac1xOmMFrsWenJJz&wd=&eqid=a1b856350003831f000000035f21313f)

from gensim import models

tfidf=models.TfidfModel(主题向量库)

tfidf[某篇文本的主题向量]

#gensim会自动去除停用词，比如the

#gensim会自动去除单个字母，比如i

#gensim会去除词袋中没有的词，也就是说不能训练没有出现在语料库中的词，所以通过gensim并不能计算每个单词的tfidf值

Word2Vec

**词向量训练：**

from gensim.models import Word2Vec

sentences = word2vec.LineSentence(分好词且一行一句的语料库txt路径) #option1

sentences = 一个分好词的list，如[["cat", "say", "meow"], ["dog", "say", "woof"]]#option2

model = Word2Vec(sentences , size=100, window=5, min\_count=1, workers=3)

#size：词向量维度，默认100

#window：窗口大小，即词向量上下文最大距离，默认为5

#min\_count: 需要计算词向量的最小词频。这个值可以去掉一些很生僻的低频词，默认是5。

#negative：即使用Negative Sampling时负采样的个数，默认是5。

#cbow\_mean: 仅用于CBOW模型，为0，则算法中隐藏层词向量的为上下文的词向量之和，为1则为上下文的词向量的平均值。

#iter: 随机梯度下降法中迭代的最大次数，默认是5

#alpha: 是初始的学习速率，在训练过程中会线性地递减到min\_alpha。默认0.025

#min\_alpha:最小的迭代步长值

#workers：用于控制训练的并行数,默认为3

**创建word-vector查询表：**

{word:vector for word,vector in zip(w2v\_model.wv.index\_to\_key,w2v\_model.wv.vectors)}

**获取词向量结果：**model.wv.vectors

**查看两个词相似程度：**model.wv.similarity('中文词1'.decode('utf-8'), '中文词2'.decode('utf-8'))

**找出异类：**model.wv.doesnt\_match(u"沙瑞金 高育良 李达康 刘庆祝".split())

**找出某个词向量最近的词集合：**



**利用t-sne对词向量进行可视化**

from sklearn.manifold import TSNE

tsne = TSNE(n\_components=2, verbose=1, perplexity=40, n\_iter=250)

tsne\_results = tsne.fit\_transform(glove\_vectors.vectors)

x,y=zip(\*tsne\_results)

plt.figure(figsize=(12,12))

sns.scatterplot(x,y)

for i in range(len(x)):plt.text(x[i],y[i], model.wv.index\_to\_key[i],size='small')

plt.show()