姓名：刘权祥

学号：2019300414

第三次作业

实验目的

1. 根据训练集确定词语的TFIDF和Word2vec词向量（Word2vec可以使用gensim调用函数获取），窗口半径设置为2；
2. 选择10个词根据词向量确定其Top5相似词。

实验内容&原理

数据的初步处理

这次作业给的`input.txt`中，每一行几乎都是一个句子，并且每个句子的长度都差不多，为了能够计算多个文档中的词向量，这里我把**每一行都视为是一个文档**。

给的数据是已经做好分词处理了的，但是里面还有标点符号，所以读取文件之后需要进行初步处理。

这里我根据数据的一些规则，使用正则表达式去除了标点符号。**核心代码如下：**

```
def read_data(filename=None):
    """
    读取数据文件，并且进行处理，去除标点符号
    :param filename: 文件名，string类型
    :return: 文件的内容
    """
    if filename is None:
        return None
    with open(filename) as file:
        # 首先按行分词
        data_raw = file.read().split("\n")
        data = []
        for line_data in data_raw:
            if line_data is None:  # 去除一些异常情况
                continue
            # 因为标点符号一般不会出现在句首，所以这里用下面的方式刚好可以去除标点符号和多余的空格
            line_data = re.sub(r' [^\w\s]', '', line_data)
            data.append(line_data.split(" "))
        return data
```

计算词语的TFIDF

TF-IDF（term frequency–inverse document frequency，词频-逆向文件频率）是一种用于信息检索（information retrieval）与文本挖掘（text mining）的常用加权技术。

TF-IDF是一种统计方法，用以评估一字词对于一个文件集或一个语料库中的其中一份文件的重要程度。字词的重要性随着它在文件中出现的次数成正比增加，但同时会随着它在语料库中出现的频率成反比下降。

TF-IDF的主要思想是：如果某个单词在一篇文章中出现的频率TF高，并且在其他文章中很少出现，则认为此词或者短语具有很好的类别区分能力，适合用来分类。

（1）词频(Term Frequency)

词频（TF）表示词条（关键字）在文本中出现的频率。

这个数字通常会被归一化(一般是词频除以文章总词数), 以防止它偏向长的文件。但是**由于本次作业中的数据集中，每个文档都是一个句子，并且它们的长度差不多相同，所以这里可以不进行归一化处理。**

计算公式如下：

$$tf_{t,d} = log(count(t, d) + 1)$$

其中t是单词，d是文档，而$count(t, d)$是单词t在文档d中的个数。

核心代码：

```
# ==================计算每一个单词的TF==================
word_tf = {}
for t in unique_words:
    for d in range(len(document_list)):
        if t not in document_list[d]:
            word_tf[(t, d)] = 0
        else:
            word_tf[(t, d)] = 1 + math.log(document_list[d].count(t))
```

（2）逆向文件频率(Inverse Document Frequency)

逆向文件频率 (IDF)：某一特定词语的IDF，可以由总文件数目除以包含该词语的文件的数目，再将得到的商取对数得到。如果包含词条t的文档越少, IDF越大，则说明词条具有很好的类别区分能力。

通常计算IDF时，需要让分母+1，防止分母为0，但是**本次作业中的所有单词都来源于文档，所以分母不可能为0，这里就没有进行+1处理。**

计算公式如下：

$$idf_t = log(\frac{N}{df_t})$$

其中t表示单词，N表示文档的总数，而df_t是出现词t的文档频次。

核心代码：

```
    # ==================计算每一个单词的IDF==================
    word_idf = {} # 存储每个词的idf值
    word_df = defaultdict(int) # 存储包含该词的文档数
    for t in unique_words:
        for d in document_list:
            if t in d:
                word_df[t] += 1
        word_idf[t] = math.log(len(document_list) / (word_df[t])) # 单词是从文档里提取出来的，所以不担心分母为0
```

（3）词频-逆向文件频率（TFIDF）

某一特定文件内的高词语频率，以及该词语在整个文件集合中的低文件频率，可以产生出高权重的TF-IDF。因此，TF-IDF倾向于过滤掉常见的词语，保留重要的词语。

计算公式如下：

$$tf\_idf_w = tf_{t,d} \times idf_t$$

核心代码：

```
    # ==================计算每一个单词的TFIDF==================
    word_tfidf = {} # 存储每个词的idf值
    for t in unique_words:
        for d in range(len(document_list)):
            word_tfidf[(t, d)] = word_tf[(t, d)] * word_idf[t]
```

计算词语的Word2vec

Gensim（http://pypi.python.org/pypi/gensim）是一款开源的第三方Python工具包，用于从原始的非结构化的文本中，无监督地学习到文本隐层的主题向量表达。 主要用于主题建模和文档相似性处理，它支持包括TF-IDF，LSA，LDA，和word2vec在内的多种主题模型算法。

使用Gensim训练Word2vec十分方便，**训练步骤如下：**

1）将语料库预处理：一行一个文档或句子，将文档或句子分词（本次作业中我们只需要去除标点符号）；

2）将原始的训练语料转化成一个sentence的迭代器，每一次迭代返回的sentence是一个word的列表。可以使用Gensim中word2vec.py中的LineSentence()方法实现；

3）将上面处理的结果输入Gensim内建的word2vec对象进行训练即可。

由于是调包，所以代码特别简单，三行代码就搞定了：

```
from gensim.models import word2vec

# 训练模型
sentences = word2vec.LineSentence("data/input.txt")
model = word2vec.Word2Vec(sentences, window=2, min_count=1, sg=1)
```

确定词语的Top5相似词

使用函数 similar_by_word 就可以查找词语的相似词。**这里我们只能够查找已经有的词，如果查找不存在的词，会报错。**

代码如下所示：

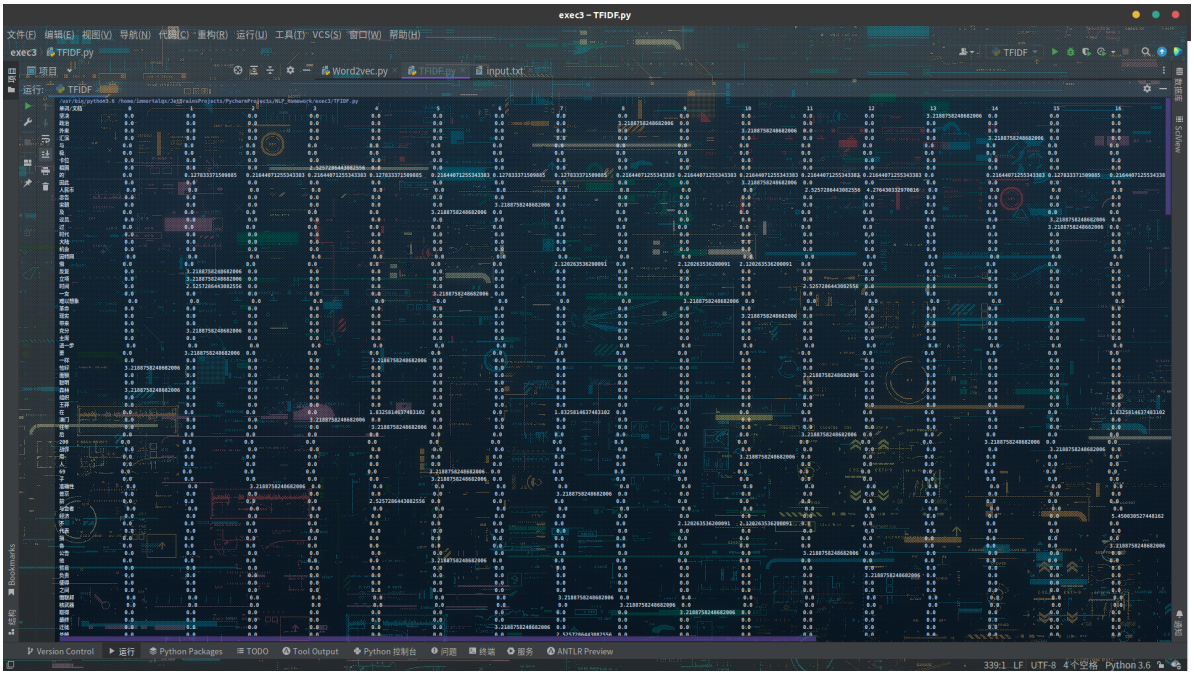
```
# 设定10个词
word_list = ["国家", "主权", "普京", "因特网", "国务卿",
         "总统", "人民", "联合国", "贡献", "主席"]

# 查找Top5相似词
for word in word_list:
    print("\"" + word + "\"" + " 的Top5相似词为：")

    req_count = 6
    for key in model.wv.similar_by_word(word, topn=100):
        # 计数
        req_count -= 1
        if req_count == 0:
            break
        # 忽略长度为1的词
        if len(key[0]) <= 1:
            continue

        print("\t", end="")
        print(key[0], key[1])
    print()
```

实验结果&分析

TFIDF计算结果

这里只计算了前面25行数据的结果。从下图中可以看出来，tf-idf向量是特别长且特别稀疏的。

下面是一张缩放的更小的图，可以看出几乎绝大多数位置都是0。



确定词语的Top5相似词

由于我设置了相似词长度小于等于1就跳过，并且`similar_by_word`中的参数`topn=100`，即在前100个最相似的词中找出5个由至少两个字构成的词语，所以有的词语找不出5个最相似的词。结果如下所示：

```
"国家" 的Top5相似词为：
    根本 0.9385648965835571
    军队 0.9328924417495728
    广大 0.9322903156280518
    民族 0.9314547181129456
    群众 0.9272975921630859

"主权" 的Top5相似词为：
    领土 0.9886884093284607
    民主 0.9777008295059204
    民族 0.9754917025566101
    发展中 0.974288284778595
    军队 0.973956823348999

"普京" 的Top5相似词为：
    克林顿 0.9932552576065063
    会见 0.9932473301887512
    唐家璇 0.9927941560745239
    钱其琛 0.9927697777748108
    朱镕基 0.9902796745300293

"因特网" 的Top5相似词为：
    评论 0.9590295553207397
    当选 0.9586423635482788
    致辞 0.9581052660942078
    孙玉玺 0.9579243659973145
    鲍威尔 0.9577133059501648

"国务卿" 的Top5相似词为：
    十月 0.9969860315322876
    参议院 0.995841920375824
    朱亚衍 0.9956451058387756
    副主席 0.9953170418739319
```

"总统" 的Top5相似词为：
  克林顿 0.980272650718689
  普京 0.9756221175193787
  李鹏 0.9747560024261475

"人民" 的Top5相似词为：
  政府 0.9101632833480835
  利益 0.9041685461997986
  群众 0.8976961970329285
  根本 0.8838081359863281

"联合国" 的Top5相似词为：
  巴基斯坦 0.9956135153770447
  韩国 0.9947172999382019
  伊拉克 0.994083821773529
  曾经 0.9937700033187866
  印度 0.9930779337882996

"贡献" 的Top5相似词为：
  成果 0.9937458634376526
  责任 0.9903661012649536
  方向 0.9902763962745667
  地位 0.9901077747344971
  巨大 0.9893426895141602

"主席" 的Top5相似词为：
  李鹏 0.9561616778373718
  今天 0.9540614485740662
  总统 0.9500451683998108

进程已结束,退出代码0