Homework 1 VSM 实验报告

实验内容：

1.预处理文本数据集，并得到每个文本的 VSM 表示

实验步骤：

1. Tokenization

分词是将文本分割成若干个单词或短语。使用 Python 的自然语言工具包 NLTK 模块可以调用 word_tokenize 方法方便分词。然而问题是标点符号也被列为单词。之后的步骤中会将这些标点去掉。
以一句话为例：
分词之前：

```
It's not straight-forward to perform so-called 'tokenization' in U.S.A
```

分词之后：

```
['It', "'s", 'not', 'straight-forward', 'to', 'perform', 'so-called', "'tokenization", "'", 'in', 'U.S.A']
```

2. Normalization and Stemming

词干提取去除单词的单复数或时态不同带来的影响。NLTK 中的 PorterStemmer 可以方便的进行 stemming。我发现直接 stemming 之后“U.S.A”变成“u.s.a”了，试了试其他大写单词也变成了小写，所以一开始以为这个 stemming 自动进行 normalization 了。后来突然发现“It”这个单词依然保留大写，所以还是加上了 normalization，将每个单词大写变小写，然后去除非字母的字符。
标准化之后：

```
['it', 's', 'not', 'straightforward', 'to', 'perform', 'socalled', 'tokenization', '', 'in', 'usa']
```

词干提取之后：

```
['it', 's', 'not', 'straightforward', 'to', 'perform', 'socal', 'token', '', 'in', 'usa']
```

3. Stopword filtering

将出现频率过高的或频率过低的这些不具代表性的词语做为停用词，将这些词从文本中删除来减少不必要的统计。用 NLTK 的 stopwords.words('english')可以获得停用词列表，在单词中将停用词过滤掉，顺便把空串去除。
去停用词之后：

```
['straightforward', 'perform', 'socal', 'token', 'usa']
```

4. TF-IDF Weighting

用 TF-IDF weighting 的方法为文章的单词赋予权值。TF 表示单词在文章中的

出现程度，IDF 表示单词在所有文章中的稀有程度，用 TF*IDF 来为作为每个单词的权值来反映这个单词对区分文章内容的影响。
TF 和 IDF 用如下定义方式：

$$tf(t,d) = \begin{cases} 1 + \log c(t,d), if\ c(t,d) > 0 \\ 0, otherwise \end{cases}$$

$$IDF(t) = \log(\frac{N}{df(t)})$$

5. Distance Calculation

用两个文章表示成的向量之间的距离来衡量这两个文章的相似程度。如果单单用欧几里得距离的话会受到向量本身长度的影响，两个更长的向量他们的距离会变得更远，这样长文章的相似程度会更低。于是想法是将向量单位化再求欧式距离，也就相当于求向量之间的夹角了。

$$cosine(d_i, d_j) = \frac{V_{d_i}^T V_{d_j}}{|V_{d_i}|_2 \times |V_{d_j}|_2}$$

实验总结：

这次实验是第一次实验，并不很难，但是因为各种原因拖到了很晚，感觉非常羞愧。在实验中遇到的问题有很多，首先安装 python 就用了不少时间，然后一开始不知道 NLTK，想用 StandfordParser 进行 tokenization，然后这个使用 java 写的，要装 jpype，装好了又发现他的代码是 Python2 写的，各种不会用。后来终于发现了 NLTK，感觉好方便。之后边百度边写终于差不多回忆起 python 的东西了，慢慢变得熟练了起来。以后的实验应该会顺利许多。