Homework1 报告

霍瑜 201605130119

第一步：分割单词，单词标准化，去停用词

问题 1：根据哪些字符来分割？

理想情况下在英文文档中，应该以空格，换行等空字符来分割，但在此数据集中存在标点符号后无空格的情况，这就会导致标点符号连接的两个单词会被分到一起，所以我们需要把标点符号也加入到分割符中。但这又会导致另一个问题，类似 U.S.A 的字符串，如果采用上述方法，会把 U.S.A 分割为三个单字符，也就损坏了信息。
但文档中标点符号后无空格的情况多与类似 U.S.A 的情况，所以综合考虑还是把标点符号也加入分隔符。

问题 2：如何单词标准化？

方案：去掉所有非拉丁字母的字符并把大写转小写。
本来我还想着保留数字，但之后发现保留数字的单词很杂乱，有很多无意义的内容

问题 3：去停用词

网上找了个停用词表，包含 891 个单词，直接相减

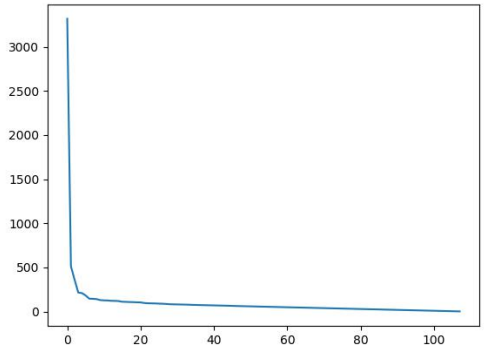
代码见 get_word.py
经过上述处理后的文档信息见 article1.txt，其中每行表示一个文档，每行由多个空格分开的单词组成

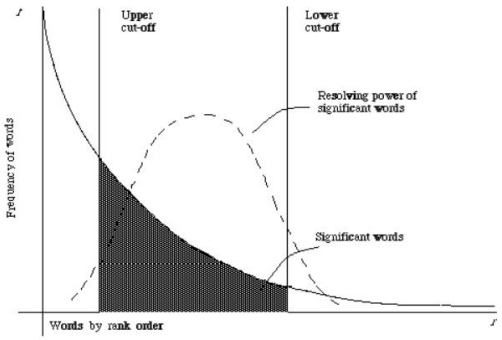
第二步：去除无用的单词

问题 1：Zipf's law

利用数据我验证了一下 Zipf's law。

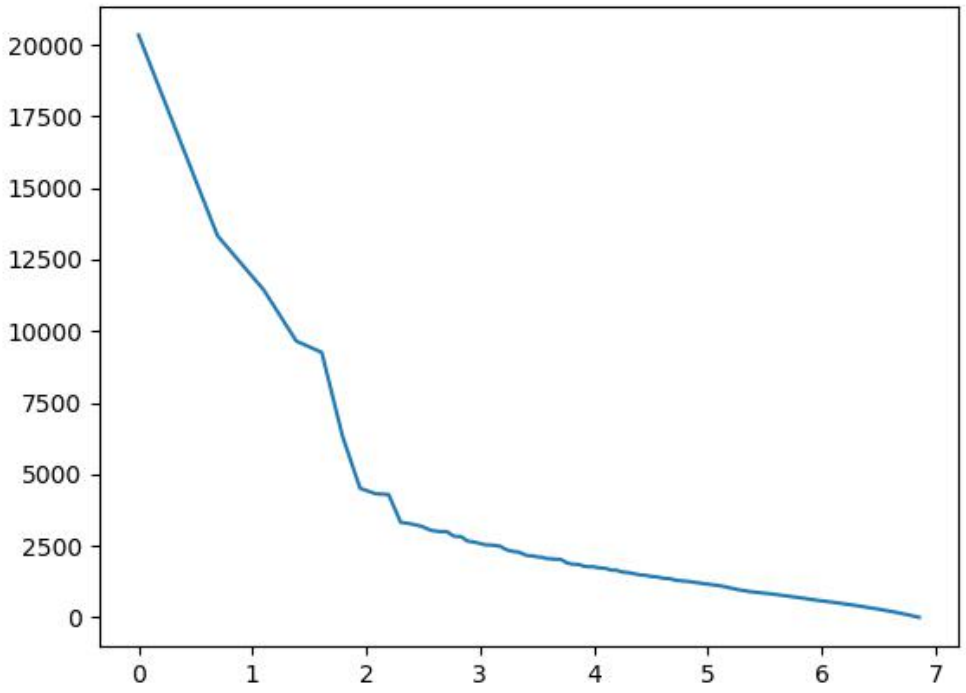
下图：横坐标表示单词词频的排名，纵坐标表示单词出现的次数
左：我跑出的图　右：课上 PPT 中的图





差别有点大！！！！
个人认为是数据量太小，导致很多的单词都只出现了少数次，才会显得曲线过早的逼近零

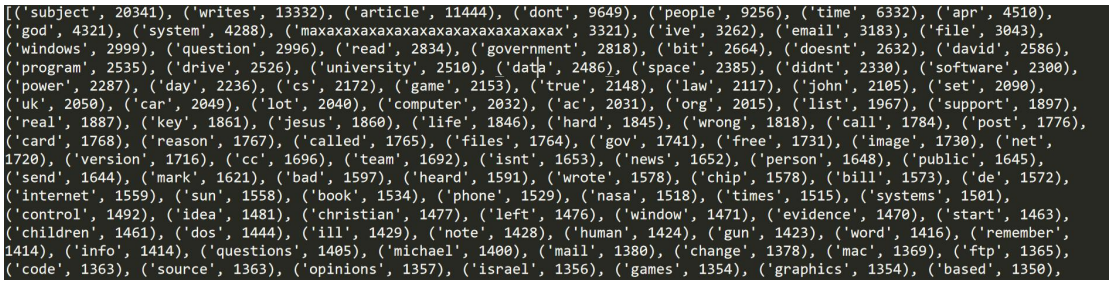
为扩大效果，我对横坐标取了 log，得到下图：



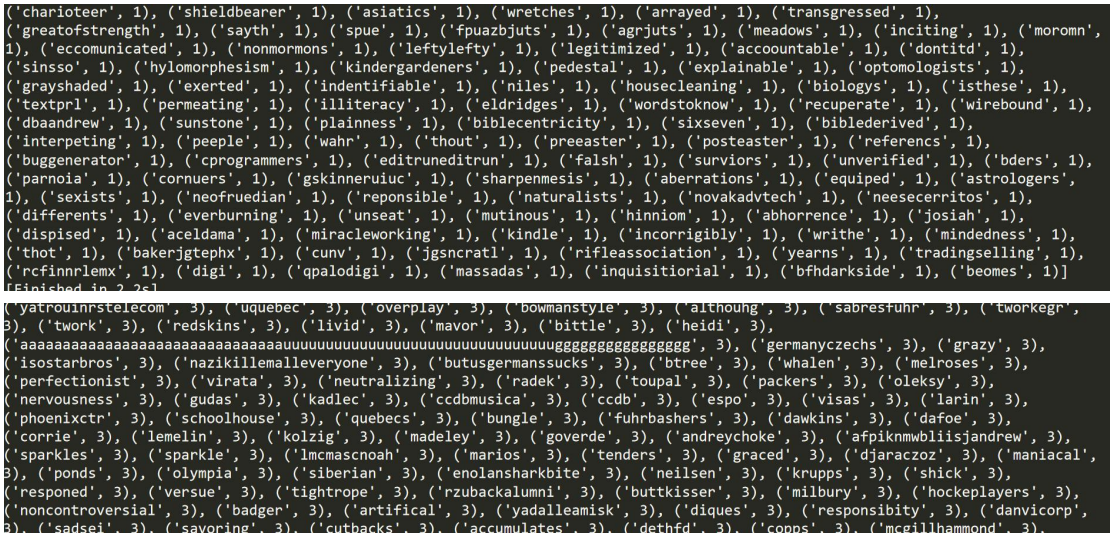
总的来看，还算 ok

问题 2：选取的词频范围？

高频单词：

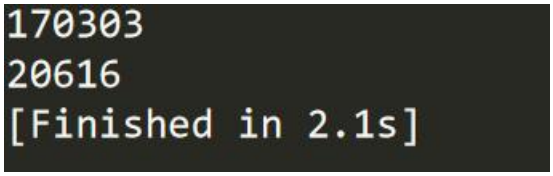


低频单词：



由于已经去除了停用词，也就可以假定不存在词频特高且无用的单词，于是我就只对下限做了限制，综合词典大小，词意等方面考虑，最后选择去除词频小于等于 10 的单词

词典大小降了 8.5 倍左右



代码见 get_word1.py
经过上述处理后的文档信息见 article2.txt

第三步：词形还原，词干提取

使用 NLTK 处理库

问题 1：选择词形还原还是词干提取？

简单试了试 NLTK 库里的 steming 和 lemmatization，发现词形还原的效果很差，连简单的 drove，driving 都处理不了，于是果断选区词干提取。

代码见 get_word2.py
经过上述处理后的文档信息见 article3.txt

第四步：TF-IDF 计算

$$\mathrm{tf}_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}}$$

$$\mathrm{idf}_i = \lg \frac{|D|}{|\{j : t_i \in d_j\}|}$$

利用公式计算出 tf，idf，直接相乘

代码见 get_vector.py
经过上述处理后的文档信息见 vector.txt(由于太大，未上传到 github)

问题：词典两万多个单词，那么表示一个文档的向量就有两万维，导致很多都是零，这是 VSM 很大的一个问题

第五步：计算文档相似度（即向量距离）

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

采用向量夹角 cos 值表示，尝试计算了一下第一个文档和第二个文档的相似度：

```
22 print(calc(vector[0], vector[1], len(vector[0])))
23
```

```
0.39727246021557594
[Finished in 15.1s]
```

代码见 calc_distance.py