Web 实验报告 Lab1

金哲欣 PB17111663

许世晨 PB17030846

1. 分词

1.1 分词工具包

我们分别使用了两个分词工具包：pkuseg 和 jieba。

jieba 分词特点：

- 支持三种分词模式：
  - 精确模式，试图将句子最精确地切开，适合文本分析；
  - 全模式，把句子中所有的可以成词的词语都扫描出来, 速度非常快，但是不能解决歧义；
  - 搜索引擎模式，在精确模式的基础上，对长词再次切分，提高召回率，适合用于搜索引擎分词。
- 支持繁体分词。
- 支持自定义词典。
- MIT 授权协议。

pkuseg 分词特点：

- 多领域分词。不同于以往的通用中文分词工具，此工具包同时致力于为不同领域的数据提供个性化的预训练模型。根据待分词文本的领域特点，用户可以自由地选择不同的模型。 目前支持了新闻领域，网络领域，医药领域，旅游领域，以及混合领域的分词预训练模型。在使用中，如果用户明确待分词的领域，可加载对应的模型进行分词。如果用户无法确定具体领域，推荐使用在混合领域上训练的通用模型。
- 更高的分词准确率。相比于其他的分词工具包，当使用相同的训练数据和测试数据，pkuseg可以取得更高的分词准确率。
- 支持用户自训练模型。
- 支持用户使用全新的标注数据进行训练。
- 支持词性标注。

经过测试，我们最终选择了 jieba 分词的搜索引擎模式。

1.2 停用词处理

我们利用分词工具包中的词性标注功能，把量词、代词、副词、介词、助词、语气词、叹词、拟声词、标点等词性的词进行删去，但是最终结果并没有什么改善。

最终决定仅仅删去标点，包括：

```
['~', '`', '!', '@', '#', '$', '%', '^', '&', '*', '(', ')', '_', '-', '+',
'=', '[', ']', '{', '}', '\\', '|', ';', ':', '\'', '"', ',', '<', '>', '.',
'/', '?', ' ', '。', '、', '〃', '〃', '【', '】', '《', '》']
```

1.3 英文处理

中英文分离

由于现有的中文分词工具包对中英混合文本的分词并不准确，所以我们使用空格在文本中把中英文分割开来，以保证分词工具包能够顺利区分中英文。

英文小写化

为了避免大小写对英文分词统计的影响，我们把英文都转换为小写。

1.4 数字处理

由于分词工具有时候不会把中文和数字分离，导致词条统计不完全，数词统计也不完全，我们使用空格分割数字和中文。同时，有些标点与数字连接有特定含义，比如 `2-2` 、`23:19` 等，我们把 `:` 和 `-` 作为数字的连接符，不对其进行分离。

1.5 自定义词典

专有名词

专有名词	专有名词	专有名词
道聚城	枪神纪	朱自清
刘诗诗	硫酸根	口算
化工制图	李蒽熙	韩安冉
画法几何及工程制图	幼儿园	家长会
中班	三角形	盐城赶集网
百度知道	百度文库	文化生活
家乡的桥	综合实践	社会实践
ie	浏览器	四年级
六年级	清理器	机械设计基础
长媳难为	科学家	地级市
县级市	i5	i7
语文园地	中国梦	3d
久保田	收割机	走光
世界地球日	月考	photoshop

错别字

原词	纠正
道具城	道聚城
和迅网	和讯网
无限活力	无限火力

近义词

原词	近义词
黑板报	手抄报
手抄报	黑板报
来历	由来
鲁滨孙	鲁滨逊
评课稿	说课稿
申请材料	申请书
1折	一折
改选	竞选
潜血	隐血
启发	启示
构思	构想
滑冰	溜冰
译文	翻译

1.6 Title 权重

由于文档的标题是文档内容的归纳和浓缩，所以我们认为标题应该增大权重，最后将标题复制100次，与文本内容拼接之后进行分词。

2. 计算 tf-idf

2.1 计算 tf

尝试了4种词项频率计算方式.

1. $w_{t,d} = tf_{t,d}$

这是最为平凡的一种计算方式， 直接使用关键词在文档中出现的次数作为权重。 一方面，我们期待这种朴素的计算方法可能会在小规模的文档集合中产生不错的效果（事实证明，这种期待是不明智的）；另一方面，这个平凡的计算方式能够为后面的几种计算方式起到baseline的作用。

2. $w_{t,d} = 1 + \ln(tf_{t,d})$

我们之后使用了教材中的词频计算方式。$ln()$ 变换避免了一个词的权重随着词语出现的次数线性增长，使得词语权重的增加速率随着词语次数的增加而衰减， 基本符合现实中文档相关性随着关键词次数增加而发生变化的情况。毕竟， 对于一篇关键词出现1,000次的文档和关键词出现100次的文档，我们不能说前者的相关性是后者的10倍。但是这样做的问题在于，这个公式没有考虑文档的长度对结果的影响。假如一篇文档的相关性并不高，但是这篇文档的长度很长，也会导致关键词在这篇文档中大量出现，总而使得计算得到的权重很大。这是不符合事实的。

3. $w_{t,d} = \frac{tf_{t,d}}{N_d}$

这是一种较为主流的计算方式，将权重确定为关键词在文档中出现的次数与文档总长的比值。一方面，随着关键词出现次数的增加，计算得到的权重也会随之增加；另一方面，对于一篇相关性不高但是长度很长的文档，这种计算方法也不会为这个关键词计算出很大的权重值，因为我们对词频除以了文档的总长度，使得文章的长度不至于产生太大的影响。

4. $w_{t,d} = \frac{1+\log_{10}(tf_{t,d})}{\log_{10}(N_d)}$

在第3中方法中，一个文档对于某关键词的权重和这个关键词在文档中的密度是成正比的。但现实世界中往往未必如此。我们假设，权重的变化应该更加平滑而舒缓，所以我们对词频和文档长度做了ln映射之后再相除。为了避免出现0权重，我们在分子中$+1$. 这样以来，我们得到了一个既考虑了文档长度对权重的影响，同时变化也较为舒缓的权重计算方法。当然，以上公式的效果还需要在实践中检验。

> 注：N_d 为文档d中的总词数。

2.2 计算 idf

尝试了2种逆文档频率计算方式：

1. $idf_t = \log_{10}(\frac{N}{df_t})$
2. $idf_t = \ln(\frac{N}{df_t})$

最终，我们选择了第二种 tf 计算方式和第二种 idf 计算方式。

3. 计算相似度

尝试了两种相似度的计算方式：

1. $cos(D, Q)$
2. $D \cdot Q^T$

经过测试，第2种计算方式效果更好，事实上第1种方式只是比第2种方式多除了 $|D| \cdot |Q|$ ，猜测可能是因为第1种方式会受到文章长度的影响，导致相似度计算表现不佳。当然，第2中方式也不是没有缺憾的。在第2种计算方式中，文档中一切不在query中的词语，都没有对相似度的计算产生任何影响，这与现实世界的情况是不太符合的。只能说在实验使用的数据集里，文档长度所导致的误差比无视无关词语带来的误差更为显著。

4. 重排序

由于增大 title 权重会导致文本相关性的排序更多依赖文本标题，一定程度上导致F1虽然高，但是NDCG值较低，所以我们对文档重新进行 title 权重为1的分词，然后重新计算 tf-idf 和相关度，重新排序之后得到最终结果。

5. 结果

截至 2019.12.1 15:07 我们的排名是第一：

filename	F1	NDCG@20
PB17111663_金哲欣_lab1_submission_11.csv	0.8257975573824473	0.9016877043260646
PB16001810_蒋硕轩_lab1_submission_5.csv	0.8220967012009793	0.8978684449473255
PB16001707_王梓涵_lab1_submission_8.csv	0.8220967012009793	0.8978677716716333