动手实战中文文本中的关键字提取

Python实验中心 2019-01-27

前言

关键词提取就是从文本里面把跟这篇文章意义最相关的一些词语抽取出来。这个可以追溯到文献检索初期，关键词是为了文献标引工作，从报告、论文中选取出来用以表示全文主题内容信息的单词或术语，在现在的报告和论文中，我们依然可以看到关键词这一项。因此，关键词在文献检索、自动文摘、文本聚类/分类等方面有着重要的应用，它不仅是进行这些工作不可或缺的基础和前提，也是互联网上信息建库的一项重要工作。

关键词抽取从方法来说主要有两种：

- 第一种是关键词分配：就是给定一个已有的关键词库，对于新来的文档从该词库里面匹配几个词语作为这篇文档的关键词。
- 第二种是关键词提取：针对新文档，通过算法分析，提取文档中一些词语作为该文档的关键词。

目前大多数应用领域的关键词抽取算法都是基于后者实现的，从逻辑上说，后者比前者在实际应用中更准确。

下面介绍一些关于关键词抽取的常用和经典的算法实现

基于TF-IDF算法进行关键词提取

在信息检索理论中，TF-IDF 是 Term Frequency - Inverse Document Frequency 的简写。TF-IDF 是一种数值统计，用于反映一个词对于语料中某篇文档的重要性。在信息检索和文本挖掘领域，它经常用于因子加权。TF-IDF 的主要思想就是：如果某个词在一篇文档中出现的频率高，也即 TF 高；并且在语料库中其他文档中很少出现，即 DF 低，也即 IDF 高，则认为这个词具有很好的类别区分能力。

TF 为词频（Term Frequency），表示词 t 在文档 d 中出现的频率，计算公式：

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

其中，ni,j是该词 ti在文件 dj 中的出现次数，而分母则是在文件 dj 中所有字词的出现次数之和。

IDF 为逆文档频率（Inverse Document Frequency），表示语料库中包含词 t 的文档的数目的倒数，计算公式：

$$\mathrm{idf_i} = \log \frac{|D|}{|\{j : t_i \in d_j\}|}$$

其中，|D|表示语料库中的文件总数，|{j:ti∈dj}| 包含词 ti的文件数目，如果该词语不在语料库中，就会导致被除数为零，因此一般情况下使用 1+|{j:ti∈dj}|

TF-IDF 在实际中主要是将二者相乘，也即 TF * IDF， 计算公式：

$$\mathrm{tfidf_{i,j}} = \mathrm{tf_{i,j}} \times \mathrm{idf_i}$$

因此，TF-IDF 倾向于过滤掉常见的词语，保留重要的词语。例如，某一特定文件内的高频率词语，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。

好在 jieba 已经实现了基于 TF-IDF 算法的关键词抽取，通过命令 `import jieba.analyse` 引入，函数参数解释如下：

```
jieba.analyse.extract_tags(sentence, topK=20, withWeight=False, allowPOS=())
```

- sentence：待提取的文本语料；
- topK：返回 TF/IDF 权重最大的关键词个数，默认值为 20；
- withWeight：是否需要返回关键词权重值，默认值为 False；
- allowPOS：仅包括指定词性的词，默认值为空，即不筛选。

接下来看例子，我采用的语料来自于百度百科对人工智能的定义，获取 Top20 关键字，用空格隔开打印：

```
import jieba.analyse
sentence  = "人工智能（Artificial Intelligence），英文缩写为AI。它是研究、开发
keywords = "  ".join(jieba.analyse.extract_tags(sentence , topK=20, withWeigh
print(keywords)
```

执行结果：

> 人工智能 智能 2017 机器 不同 人类 科学 模拟 一门 技术 计算机 研究 工作 Artificial Intelligence AI 图像识别 12 复杂 流行语

下面只获取 Top10 的关键字，并修改一下词性，只选择名词和动词，看看结果有何不同？

```
keywords =(jieba.analyse.extract_tags(sentence , topK=10, withWeight=True, al
```

执行结果：

> [('人工智能', 0.9750542675762887), ('智能', 0.5167124540885567), ('机器', 0.20540911929525774), ('人类', 0.17414426566082475), ('科学', 0.17250169374402063), ('模拟', 0.15723537382948452), ('技术', 0.14596259315164947), ('计算机', 0.14030483362639176), ('图像识别', 0.12324502580309278), ('流行语', 0.11242211730309279)]

基于TexRank算法进行关键词提取

TextRank 是由 PageRank 改进而来，核心思想将文本中的词看作图中的节点，通过边相互连接，不同的节点会有不同的权重，权重高的节点可以作为关键词。这里给出 TextRank 的公式：

$$WS(V_i) = (1-d) + d * \sum_{V_j \in In(V_i)} \frac{w_{ji}}{\sum_{V_k \in Out(V_j)} w_{jk}} WS(V_j)$$

节点 i 的权重取决于节点 i 的邻居节点中 i-j 这条边的权重 / j 的所有出度的边的权重 * 节点 j 的权重，将这些邻居节点计算的权重相加，再乘上一定的阻尼系数，就是节点 i 的权重，阻尼系数 d 一般取 0.85。

TextRank 用于关键词提取的算法如下：

（1）把给定的文本 T 按照完整句子进行分割，即:

$$T=[S_1 , S_2 , \cdots , S_m]$$

（2）对于每个句子，进行分词和词性标注处理，并过滤掉停用词，只保留指定词性的单词，如名词、动词、形容词，其中 ti,j是保留后的候选关键词。

$$S_i=[t_{i,1} , t_{i,2} , \cdots , t_{i,n}]$$

（3）构建候选关键词图 G = (V,E)，其中 V 为节点集，由（2）生成的候选关键词组成，然后采用共现关系（Co-Occurrence）构造任两点之间的边，两个节点之间存在边仅当它们对应的词汇在长度为 K 的窗口中共现，K 表示窗口大小，即最多共现 K 个单词。

（4）根据 TextRank 的公式，迭代传播各节点的权重，直至收敛。

（5）对节点权重进行倒序排序，从而得到最重要的 T 个单词，作为候选关键词。

（6）由（5）得到最重要的 T 个单词，在原始文本中进行标记，若形成相邻词组，则组合成多词关键词。

同样 jieba 已经实现了基于 TextRank 算法的关键词抽取，通过命令 `import jieba.analyse` 引用，函数参数解释如下：

```
jieba.analyse.textrank(sentence, topK=20, withWeight=False, allowPOS=('ns', 'n', 'v
```

直接使用，接口参数同 TF-IDF 相同，注意默认过滤词性。

接下来，我们继续看例子，语料继续使用上例中的句子。

```
    result = "    ".join(jieba.analyse.textrank(sentence, topK=20, withWeight=Fals
```

执行结果：

> 智能 人工智能 机器 人类 研究 技术 模拟 包括 科学 工作 领域 理论 计算机 年度 需要 语言 相似 方式 做出 心理学

如果修改一下词性，只需要名词和动词，看看结果有何不同？

```
    result = "    ".join(jieba.analyse.textrank(sentence, topK=20, withWeight=Fals
```

执行结果：

> 智能 人工智能 机器 人类 技术 模拟 包括 科学 理论 计算机 领域 年度 需要 心理学 信息 语言 识别 带来 过程 延伸

基于LDA主题模型进行关键词提取

其实，使用 LDA 获取文本关键词在我的第一次 Chat《NLP 中文短文本分类项目实践（上）》已经讲过了，为了保持内容的完整性，在这里我继续写一下。

语料是一个关于汽车的短文本，下面通过 Gensim 库完成基于 LDA 的关键字提取。整个过程的步骤为：文件加载 -> jieba 分词 -> 去停用词 -> 构建词袋模型 -> LDA 模型训练 -> 结果可视化。

```
#引入库文件
import jieba.analyse as analyse      import jieba      import pandas as pd
%matplotlib inline      #设置文件路径
dir = "D://ProgramData//PythonWorkSpace//study//"
file_desc = "".join([dir,'car.csv'])
stop_words = "".join([dir,'stopwords.txt'])      #定义停用词
stopwords=pd.read_csv(stop_words,index_col=False,quoting=3,sep="\t",names=['s
stopwords=stopwords['stopword'].values      #加载语料
df = pd.read_csv(file_desc, encoding='gbk')      #删除nan行
df.dropna(inplace=True)
lines=df.content.values.tolist()      #开始分词
sentences=[]      for line in lines:           try:
            segs=jieba.lcut(line)
            segs = [v for v in segs if not str(v).isdigit()]#去数字
            segs = list(filter(lambda x:x.strip(), segs))     #去左右空格
            segs = list(filter(lambda x:x not in stopwords, segs)) #去掉停用i
            sentences.append(segs)           except Exception:
            print(line)                 continue
#构建词袋模型
dictionary = corpora.Dictionary(sentences)
corpus = [dictionary.doc2bow(sentence) for sentence in sentences]      #lda模
lda = gensim.models.ldamodel.LdaModel(corpus=corpus, id2word=dictionary, num_t
print(lda.print_topic(1, topn=5))      #我们打印所有5个主题，每个主题显示8个词
for topic in lda.print_topics(num_topics=10, num_words=8):
     print(topic[1])
```

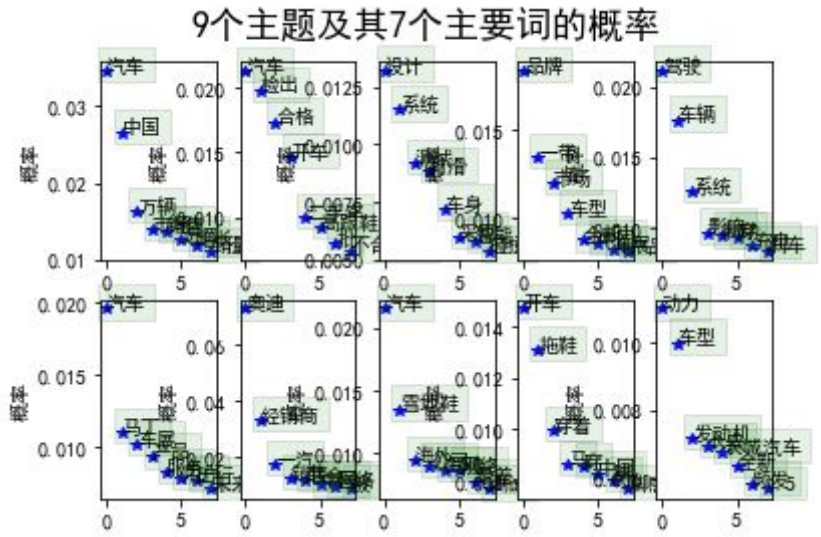
执行结果如下图所示：

```
0.021*"汽车" + 0.020*"检出" + 0.017*"合格" + 0.015*"开车" + 0.010*"二手车"
0.035*"汽车" + 0.027*"中国" + 0.016*"万辆" + 0.014*"市场" + 0.014*"销售" + 0.013*"进口" + 0.012*"增长" + 0.011*"销量"
0.021*"汽车" + 0.020*"检出" + 0.017*"合格" + 0.015*"开车" + 0.010*"二手车" + 0.009*"高跟鞋" + 0.008*"批" + 0.007*"不合格率"
0.013*"设计" + 0.011*"系统" + 0.009*"测试" + 0.009*"打滑" + 0.007*"车身" + 0.006*"采用" + 0.006*"智能" + 0.005*"碰撞"
0.018*"品牌" + 0.013*"一带" + 0.012*"市场" + 0.010*"车型" + 0.009*"全新" + 0.009*"中国" + 0.008*"车展" + 0.008*"产品"
0.021*"驾驶" + 0.018*"车辆" + 0.013*"系统" + 0.010*"影响" + 0.009*"制动" + 0.009*"穿" + 0.009*"充电" + 0.008*"刹车"
0.019*"汽车" + 0.011*"马丁" + 0.010*"车展" + 0.009*"产品" + 0.008*"服务" + 0.008*"用户" + 0.008*"出行" + 0.007*"未来"
0.073*"奥迪" + 0.033*"经销商" + 0.017*"一汽" + 0.012*"销售" + 0.012*"联会" + 0.010*"中国" + 0.009*"网络" + 0.008*"被"
0.021*"汽车" + 0.013*"雪地鞋" + 0.009*"海外" + 0.009*"公司" + 0.009*"驾驶" + 0.009*"网络" + 0.008*"最差" + 0.007*"测试"
0.015*"开车" + 0.013*"拖鞋" + 0.010*"穿着" + 0.009*"马" + 0.009*"穿" + 0.008*"中国" + 0.008*"脚" + 0.008*"脚感"
0.011*"动力" + 0.010*"车型" + 0.007*"发动机" + 0.007*"长安" + 0.007*"长城汽车" + 0.006*"全新" + 0.006*"频发" + 0.006*"1.5"
```

```
#显示中文matplotlib
plt.rcParams['font.sans-serif'] = [u'SimHei']
plt.rcParams['axes.unicode_minus'] = False
# 在可视化部分，我们首先画出了九个主题的7个词的概率分布图
num_show_term = 8 # 每个主题下显示几个词
num_topics    = 10
for i, k in enumerate(range(num_topics)):
```

```
        ax = plt.subplot(2, 5, i+1)
        item_dis_all = lda.get_topic_terms(topicid=k)
        item_dis = np.array(item_dis_all[:num_show_term])
        ax.plot(range(num_show_term), item_dis[:, 1], 'b*')
        item_word_id = item_dis[:, 0].astype(np.int)
        word = [dictionary.id2token[i] for i in item_word_id]
        ax.set_ylabel(u"概率")              for j in range(num_show_term):
              ax.text(j, item_dis[j, 1], word[j], bbox=dict(facecolor='green',al
plt.suptitle(u'9个主题及其7个主要词的概率', fontsize=18)
plt.show()
```

执行结果如下图所示：



基于 pyhanlp 进行关键词提取

除了 jieba，也可以选择使用 HanLP 来完成关键字提取，内部采用 TextRankKeyword 实现，语料继续使用上例中的句子。

```
from pyhanlp import *
result = HanLP.extractKeyword(sentence, 20)       print(result)
```

执行结果：

> [人工智能, 智能, 领域, 人类, 研究, 不同, 工作, 包括, 模拟, 新的, 机器, 计算机, 门, 科学, 应用, 系统, 理论, 技术, 入选, 复杂]

总结

本节内容的重点就是掌握关键字提取的基本方法，常规的关键词提取方法如上所述，当然还有其他算法及其改进，有深入研究需求的，可以下载关键字提取方面的论文阅读。

参考文献

1. Mihalcea R, Tarau P. TextRank: Bringing order into texts[C]//Proceedings of EMNLP. 2004, 4(4): 275.
2. Witten I H, Paynter G W, Frank E, et al. KEA: Practical automatic keyphrase extraction[C]//Proceedings of the fourth ACM conference on Digital libraries. ACM, 1999: 254-255.
3. Chien L F. PAT-tree-based keyword extraction for Chinese information retrieval[C]//ACM SIGIR Forum. ACM, 1997, 31(SI): 50-58.