NLP----关键词提取算法（TextRank,TF/IDF）

河北凝讯科技订阅号 2018-11-24

TF/IDF

基本思想：TF是计算一个词在一篇文档中出现的频率，IDF是一个词在多少篇文档中出现过，显然TF越高证明这个词在这篇文章中的代表性就越强，而INF越低则证明这个词在具有越强的区分能力。因此中和这两个数，就能较好地算出文档的关键词。

关键公式

$$tf * idf(i,j) = tf_{ij} * idf_i = \frac{n_{ij}}{\sum_k n_{kj}} * log(\frac{|D|}{1+|D_i|})$$

|D_i|是文档中出现词i的文档数量，|D|是文档数

附上书上抄来的代码

```python
import jieba
import jieba.posseg as psg
import math
import functools

# 停用词表加载方法


def get_stopword_list():
    # 停用词表存储路径，每一行为一个词，按行读取进行加载
    # 进行编码转换确保匹配准确率
    stop_word_path = './data/stopword.txt'
    stopword_list = [sw.replace('\n', '')
                     for sw in open(stop_word_path,encoding = 'utf-8').readlines()]
    return stopword_list

# 分词方法，调用结巴接口


def seg_to_list(sentence, pos=False):
    if not pos:
        # 不进行词性标注的分词方法
        seg_list = jieba.cut(sentence)
    else:
        # 进行词性标注的分词方法
        seg_list = psg.cut(sentence)
    return seg_list

# 去干扰词


def word_filter(seg_list, pos=False):
    stopword_list = get_stopword_list()
    filter_list = []
    # 根据POS参数选择是否词性过滤
    # 不进行词性过滤，则将词性都标记为n，表示全部保留
    for seg in seg_list:
        if not pos:
```

```
            word = seg
            flag = 'n'
        else:
            word = seg.word
            flag = seg.flag
        if not flag.startswith('n'):
            continue
        # 过滤高停用词表中的词，以及长度<2的词
        if not word in stopword_list and len(word) > 1:
            filter_list.append(word)

    return filter_list

# 数据加载，pos为是否词性标注的参数，corpus_path为数据集路径


def load_data(pos=False, corpus_path='./data/corpus.txt'):
    # 调用上面方法对数据集进行处理，处理后的每条数据仅保留非干扰词
    doc_list = []
    for line in open(corpus_path, 'r',encoding = 'utf-8'):
        content = line.strip()
        seg_list = seg_to_list(content, pos)
        filter_list = word_filter(seg_list, pos)
        doc_list.append(filter_list)
    return doc_list


def train_idf(doc_list):
    idf_dic = {}
    #总文档数
    tt_count = len(doc_list)

    #每个词出现的文档数
    for doc in doc_list:
        for word in set(doc):
            idf_dic[word] = idf_dic.get(word,0.0)+1.0
```

```
75
76      #按公示转换为idf值，分母加一进行平滑处理
77      for k,v in idf_dic.items():
78          idf_dic[k]=math.log(tt_count/(1.0+v))
79
80      #对于没有在字典中的词，默认其仅在一个文档中出现，得到默认idf值
81      default_idf = math.log(tt_count/1.0)
82      return idf_dic,default_idf
83
84  def cmp(e1,e2):
85      import numpy as np
86      res = np.sign(e1[1]-e2[1])
87      if res != 0:
88          return res
89      else:
90          a = e1[0]+e2[0]
91          b = e2[0]+e1[0]
92          if a>b:
93              return 1
94          elif a == b:
95              return 0
96          else:
97              return -1
98
99  class TfIdf(object):
100     #统计tf值
101     def get_tf_dic(self):
102         tf_dic = {}
103         for word in self.word_list:
104             tf_dic[word] = tf_dic.get(word,0.0)+1.0
105
106         tt_count = len(self.word_list)
107         for k,v in tf_dic.items():
108             tf_dic[k] = float(v)/tt_count
109
110         return tf_dic
111
112     #四个参数分别是：训练好的idf字典，默认idf值，处理后的待提取文本，关键词数量
```

```
    def __init__(self,idf_dic,default_idf,word_list,keyword_num):
        self.word_list = word_list
        self.idf_dic,self.default_idf = idf_dic,default_idf
        self.tf_dic  = self.get_tf_dic()
        self.keyword_num = keyword_num

    #按公式计算tf_idf
    def get_tfidf(self):
        tfidf_dic = {}
        for word in self.word_list:
            idf = self.idf_dic.get(word,self.default_idf)
            tf = self.tf_dic.get(word,0)

            tfidf  = tf*idf
            tfidf_dic[word] = tfidf

            #根据tf_idf排序，取排名前keyword_num的词作为关键词
        for k ,v in sorted(tfidf_dic.items(),key=functools.cmp_to_key(cmp),reverse=True)[:self.keyword_num]:
            print(k+"/",end='')
        print()

def tfidf_extract(word_list, pos=False, keyword_num=10):
    doc_list = load_data(pos)
    idf_dic, default_idf = train_idf(doc_list)
    tfidf_model = TfIdf(idf_dic, default_idf, word_list, keyword_num)
    tfidf_model.get_tfidf()

if __name__ == '__main__':
    text = '6月19日,《2012年度“中国爱心城市”公益活动新闻发布会》在京举行。' + \
           '中华社会救助基金会理事长许嘉璐到会讲话。基金会高级顾问朱发忠,全国老龄' + \
           '办副主任朱勇,民政部社会救助司助理巡视员周萍,中华社会救助基金会副理事长耿志远,' + \
           '重庆市民政局巡视员谭明政。晋江市人大常委会主任陈健倩,以及10余个省、市、自治区民政局' + \
           '领导及四十多家媒体参加了发布会。□中华社会救助基金会秘书长时正新介绍本年度“中国爱心城' + \
           '市”公益活动将以“爱心城市宣传、孤老关爱救助项目及第二届中国爱心城市大会”为主要内容,重庆市' + \
           '、呼和浩特市、长沙市、太原市、蚌埠市、南昌市、汕头市、沧州市、晋江市及遵化市将会积极参加' + \
           '这一公益活动。□中国雅虎副总编张银生和凤凰网城市频道总监赵耀分别以各自媒体优势介绍了活动' + \
           '的宣传方案。□会上,中华社会救助基金会与“第二届中国爱心城市大会”承办方晋江市签约,许嘉璐理' + \
           '事长接受晋江市参与“百万孤老关爱行动”向国家重点扶贫地区捐赠的价值400万元的款物。晋江市人大' + \
           '常委会主任陈健倩介绍了大会的筹备情况。'

    pos = True
    seg_list = seg_to_list(text, pos)
    filter_list = word_filter(seg_list, pos)

    print('TF-IDF模型结果：')
    tfidf_extract(filter_list)

```

TextRank

基本思路：每个词将自己的分数平均投给附近的词，迭代至收敛或指定次数即可，初始分可以打1

附上代码

```
def get_stopword_list():
    path = './data/stop_words.utf8'
    stopword_list = [sw.replace('\n','') for sw in open(path,'r',encoding='utf8').readlines()]
    return stopword_list

def seg2list(text):
    import jieba
    return jieba.cut(text)

def word_filter(seg_list):
    stopword_list = get_stopword_list()
    filter_list = []
    for w in seg_list:
        if not w in stopword_list and len(w)>1:
            filter_list.append(w)
    return filter_list


str = '6月19日,《2012年度“中国爱心城市”公益活动新闻发布会》在京举行。' + \
            '中华社会救助基金会理事长许嘉璐到会讲话。基金会高级顾问朱发忠,全国老龄' + \
            '办副主任朱勇,民政部社会救助司助理巡视员周萍,中华社会救助基金会副理事长耿志远,' + \
            '重庆市民政局巡视员谭明政。晋江市人大常委会主任陈健倩,以及10余个省、市、自治区民政局' + \
            '领导及四十多家媒体参加了发布会。□中华社会救助基金会秘书长时正新介绍本年度“中国爱心城' + \
            '市”公益活动将以“爱心城市宣传、孤老关爱救助项目及第二届中国爱心城市大会”为主要内容,重庆市' + \
            '、呼和浩特市、长沙市、太原市、蚌埠市、南昌市、汕头市、沧州市、晋江市及遵化市将会积极参加' + \
            '这一公益活动。□中国雅虎副总编张银生和凤凰网城市频道总监赵耀分别以各自媒体优势介绍了活动' + \
            '的宣传方案。□会上,中华社会救助基金会与“第二届中国爱心城市大会”承办方晋江市签约,许嘉璐理' + \
            '事长接受晋江市参与“百万孤老关爱行动”向国家重点扶贫地区捐赠的价值400万元的款物。晋江市人大' + \
            '常委会主任陈健倩介绍了大会的筹备情况。'
win={}
seg_list = seg2list(str)
filter_list = word_filter(seg_list)
#构建投分表，根据窗口
for i in range(len(filter_list)):
    if filter_list[i] not in win.keys():
        win[filter_list[i]]=set()
    if i-5 < 0:
        lindex = 0
    else:
        lindex = i-5
    for j in filter_list[lindex:i+5]:
        win[filter_list[i]].add(j)

# 投票
time = 0
score = {w:1.0 for w in filter_list}
while(time<50):
    for k,v in win.items():
        s = score[k]/len(v)
        score[k] = 0
        for i in v:
            score[i]+=s
    time+=1

l = sorted(score.items(), key=lambda score:score[1],reverse=True)
print(l)


```