【NLP基础】NLP关键字提取技术之LDA算法原理与实践

原创 Anu0225 深度学习自然语言处理 2019-07-22

> 阅读大概需要11分钟 🤤
> 跟随小博主，每天进步一丢丢 😁

引文

人们是如何从大量文本资料中便捷得浏览和获取信息？答案你肯定会说通过关键字。仔细想想，我们人类是怎么提取关键词？我们从小就接触语言，语法，当听到或者看到一句话时，我们大脑自动会对这句话按规则分词（小学是不是做过断句的训练），还记得语文老师讲过，一句话中主语（名词），谓语（动词），宾语（名词）通常就是重点，这样我们大脑从小就会根据词性和语法对句中词进行打标签，训练分类器，随着我们接触到的语料越来越多，分类器也越来越准确（如果你是从事语言学的，那你的分类器就更准）。仅仅通过词性和语法，会在长文本中出现一个问题，因为一篇文章中会出现很多主语，谓语，宾语，不可能所有的这些词都是关键词，这样我们大脑是怎么处理的，如果我们对一篇文章的背景和主题很熟悉的话，我们会很准确得从一篇文章中提取关键词，但当我们接触一篇比较陌生的文章，我们往往很难准确提取关键词。

算法

上面其实对应的是机器学习的两种方法：监督学习和无监督学习。**监督学习的关键字提取方法是通过分类的方式进行，通过打标签，训练分类器，从而实现关键字提取，但缺点就是需要大批量的标注数据，人工成本太高。**相对于监督学习，无监督学习的方法就无需标注数据，常用的无监督关键词提取算法包括：TF-IDF算法、TextRank算法和主题模型算法（LDA、LSA、LSI），现重点介绍LDA算法，其他算法后续再讲.

我不喜欢讲大多学术上比较难懂的词，下面我将**通俗得去讲解LDA算法原理。**通常我们可以定义主题是一种关键词集合，如果一篇文章出现这些关键词，我们可以直接判断这篇文章属于某种主题。但这种定义主题会有个弊端,比如一篇文章出现了一个球星的名字，那么这篇文章的主题就是体育。可能你马上反驳说不一定，文章确实有球星的名字，但是里面全部在讲球星的性丑闻，和篮球没半毛钱关系，此时主题是娱乐还差不多。所以一个词不能硬性地扣一个主题的帽子，如果说一篇文章出现了某个球星的名字，我们只能说有很大概率他属于体育的主题，但也有小概率属于娱

乐的主题。**同一个词，在不同的主题背景下，它出现的概率是不同的**。LDA认为文章都是用基本的词汇组合而成，LDA通过词汇的概率分布来反映主题！

由此可以定义LDA的生成过程：
1.对每篇文档，在主题分布中抽取一个主题
2.对抽到的主题所对应的单词分布中随机抽取一个单词
3.重复上述过程直至遍历整篇文档中的每个单词
4.经过以上三步，就可以看一下两个分布的乘积，是否符合给定文章的分布，以此来调整。

LDA的训练就是根据现有的数据集生成 **文档-主题分布矩阵** 和 **主题-词分布矩阵**。
所以LDA的核心，其实就是这个公式

$$P(词 | 文档) = P(词 | 主题)\ P(主题 | 文档)$$

实练

上面说了这么多，下面我们通过代码去实现吧，Gensim中有实现好的训练方法，直接调用即可。Gensim是一款开源的第三方Python工具包，用于从原始的非结构化文本中，无监督地学习到文本隐层的主题向量表达。

训练一个关键词提取算法需要以下步骤：

- 加载已有的文档数据集
- 加载停用词表
- 对数据集中的文档进行分词
- 根据停用词表，过滤干扰词
- 根据训练集训练算法

（很多博客上都是通过jieba分词，但我个人认为结巴分词不是很准确，如果分词都不准确，那怎么提取准确的关键词呢），个人采用pyhanlp的感知机算法进行分词，这是通过多次工作实践，感觉分词最准确的一种算法。

a.导入相关库

```
import math
import numpy as np
from pyhanlp import *
import functools
```

```
from gensim import corpora,models
```

b.定义好停用词表的加载方法

```
def get_stopword_list():
    stop_word_path='stopwords.txt'
    stopword_list=[sw.replace('\n','') for swin open(stop_word_path).readlir
    return stopword_list
```

c.定义一个分词方法

```
def seg_to_list(sentence,pos=False):
    seg_list = HanLP.newSegment(“perceptron”).seg(sentence)
    return seg_list
```

d.定义干扰词过滤方法：根据分词结果对干扰词进行过滤

```
def word_filter(seg_list,pos=False):
   stopword_list=get_stopword_list()
   filter_list = [str(s.word) for sin seg_list if not s.word in stopword_lis
   return filter_list
```

e.加载数据集，对数据集中的数据分词和过滤干扰词，每个文本最后变成一个非干扰词组成的词语列表

```
defload_data(pos=False):
    doc_list=[]
    ll =[]
    for line in open('corpus.txt','r',encoding='utf-8'):
        ll.append(line.strip())
    content=’’.join(ll)
    seg_list=seg_to_list(content,pos)
    filter_list=word_filter(seg_list,pos)
    doc_list.append(filter_list)
    return doc_list
```

f.训练LDA模型

```
# doc_list：加载数据集方法的返回结果
```

```
# keyword_num：关键词数量
# model：主题模型的具体算法
# num_topics：主题模型的主题数量
class TopicModel(object):
    def __init__(self,doc_list,keyword_num,model='LDA',num_topics=4):
        #使用gensim的接口，将文本转换为向量化的表示
        self.dictionary=corpora.Dictionary(doc_list)
        #使用BOW模型向量化
        corpus=[self.dictionary.doc2bow(doc) for doc in doc_list]
        #对每个词，根据TF-IDF进行加权，得到加权后的向量表示
        self.tfidf_model=models.TfidfModel(corpus)
        self.corpus_tfidf=self.tfidf_model[corpus]
        self.keyword_num=keyword_num
        self.num_topics=num_topics
        self.model =self.train_lda()

        #得到数据集的 主题-词分布
        word_dic=self.word_dictionary(doc_list)
        self.wordtopic_dic=self.get_wordtopic(word_dic)

    def train_lda(self):
        lda=models.LdaModel(self.corpus_tfidf,num_topics=self.num_topics,id2
        return lda

    def get_wordtopic(self,word_dic):
        wordtopic_dic={}
        for word in word_dic:
            single_list=[word]
            wordcorpus=self.tfidf_model[self.dictionary.doc2bow(single_list)
            wordtopic=self.model[wordcorpus]
            wordtopic_dic[word]=wordtopic
        return wordtopic_dic

    def get_simword(self,word_list):
        sentcorpus=self.tfidf_model[self.dictionary.doc2bow(word_list)]
        senttopic=self.model[sentcorpus]
    # 余弦相似度计算
    def calsim(l1,l2):
        a, b, c = 0.0, 0.0, 0.0
        for t1,t2 in zip(l1,l2):
```

```
            x1=t1[1]
            x2=t2[1]
            a += x1 * x1
            b += x1 * x1
            c += x2 * x2
        sim=a/math.sqrt(b*c) if not (b*c)==0 else 0.0
        return sim

    sim_dic={}
    for k,v in self.wordtopic_dic.items():
        if k not in word_list:
             continue
        sim=calsim(v,senttopic)
        sim_dic[k]=sim
        for k,v insorted(sim_dic.items(),key=functools.cmp_to_key(cmp),rever
            print(k+"/",end='')
        print()
     #词空间构建方法和向量化方法，在没有gensim接口时的一般处理方法
    def word_dictionary(self,doc_list):
        dictionary=[]
        for doc in doc_list:
            dictionary.extend(doc)
        dictionary=list(set(dictionary))
        return dictionary

    def doc2bowvec(self,word_list):
        vec_list=[1 if word in word_list else 0 for word in self.dictionary]
        return vec_list
```

g.调用主函数，对目标文本进行关键词提取

```
if __name__ == '__main__':
    text = '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
```

LDA模型结果:

```
重点/许嘉璐/行动/签约/百万/理事长/爱心/款物/晋江市/接受/
```

总体来说结果还算准确。