山东大学 计算机科学与技术 学院

信息检索与数据挖掘 课程实验报告

学号：201600301304	姓名：贾乘兴	班级：人工智能 16

实验题目：文本空间向量模型

实验内容：

本次实验的目标是建立文本的 VSM，建立文本的空间向量模型，对之后的文本处理有了极大的便利，实验步骤为，文本预处理、词频统计、建立词袋模型、计算 tf—idf 值，文本采用的是 20 类文本数据集。

首先要进行文本预处理，文本预处理最终要整理出标准的文本格式，从而可以更好进行词频统计。

首先去除各种标点符号

```
def cutsyms(str):
    new_str = re.sub('[1234567890,.\'\"\t\n*_+=?/|!@#$%^&*()`~<>:;\-\[\]]',"",str)
    return new_str
```

去除标点之后取而代之的是空格，所以可以更加方便的进行去除停用词

```
def cutstopwords(str):
    stopwords = {}.fromkeys([line.rstrip() for line in open('estopwords.txt')])
    segs = str.replace('\n','').lower().split(' ')
    new_str = ''
    for seg in segs:
        if seg not in stopwords:
            new_str = new_str + " " +seg
    return new_str
```

去除停用词后进行 normalization 和 stemming 步骤，即对单词进行归一化与词干提取，这里使用的是 nltk 库的方法

```
def stemming(str):
    s = nltk.stem.SnowballStemmer('english')
    segs = str.replace('\n', '').lower().split(' ')
    new_str = ''
    for seg in segs:
        new_str = new_str + " " + s.stem(seg)
    return new_str
```

经过以上步骤便可以得到标准的方便处理的文本格式，之后我们建立了字典来建立文本词袋，同时统计各词的文档出现频率，最后对 tf—idf 进行计算权重，tf—idf 确立权重是很好的办法，最终得到各个文本的空间向量，并存储。

最终得到的结果储存在文本文件中，使用换行符分离

本实验的全部实验代码如下

```
import os
import chardet
import re
import nltk
import math

# 去除停用词
def cutstopwords(str):
    stopwords = {}.fromkeys([line.rstrip() for line in
open('estopwords.txt')])
    segs = str.replace('\n','').lower().split(' ')
    new_str = ''
    for seg in segs:
        if seg not in stopwords:
            new_str = new_str + " " +seg
    return new_str

# 去除标点
def cutsyms(str):
    new_str = re.sub('[1234567890,.\'\"\t\n*_+=?/|!@#$%^&*()`~<>:;\-\[\]]',"
",str)
    return new_str

# 词干提取
def stemming(str):
    s = nltk.stem.SnowballStemmer('english')
    segs = str.replace('\n', '').lower().split(' ')
    new_str = ''
    for seg in segs:
        new_str = new_str + " " + s.stem(seg)
    return new_str

# 读取文本
def readtxt(path):
    global num_txt
    global num_dict
    num_dict = {}
    num_txt = 0
    all_context = ""
    for dirName, subdirList, fileList in os.walk(path):
        fileList.remove(fileList[0])
        for fname in fileList:
            fname = os.path.join(dirName, fname)
```

```
            f = open(fname, 'rb')
            data = f.read()
            f.close()

            print(chardet.detect(data))
            print(fname)

            fname =
open(fname,'r+',encoding=chardet.detect(data)['encoding'])
            str = fname.read()
            str = cutsyms(str)
            str = cutstopwords(str)
            str = stemming(str)

            strl_ist = str.replace('\n', '').lower().split(' ')
            for seg in strl_ist:
                if seg in num_dict.keys():
                    num_dict[seg] = num_dict[seg] + 1
                else:
                    num_dict[seg] = 1

            all_context = all_context + "\n" + str
            num_txt = num_txt + 1
            fname.close()
    return all_context

# 统计词出现次数
def wordcount(str):
    strl_ist = str.replace('\n','').lower().split(' ')
    count_dict = {}
    for str in strl_ist:
        if str in count_dict.keys():
            count_dict[str] = count_dict[str] + 1
        else:
            count_dict[str] = 1
    # count_list=sorted(count_dict.items(),key=lambda x:x[1],reverse=True)
    count_dict.pop('')
    return count_dict

#全部文本读取
num_txt = 0
num_dict = {}
context = readtxt("/Users/apple/Desktop/ir/news")
#context = readtxt("/Users/apple/Desktop/ir/alltxt")
```

```
#全部文档记数，去除高频低频词
str_dict = wordcount(context)
new_dict = str_dict
str_dict = {}
for seg in new_dict:
    if (new_dict[seg] > 10) & (new_dict[seg] < 1000):
        str_dict[seg] = new_dict[seg]
length = len(str_dict)
print(length)

#数据存储
full_path = '/Users/apple/Desktop/ir/altext01.txt'
file = open(full_path,'a+')
file.write(context)
file.close()

# VSM build
for dirName, subdirList, fileList in
os.walk('/Users/apple/Desktop/ir/news'):
    fileList.remove(fileList[0])
    for fname in fileList:
        fname = os.path.join(dirName, fname)
        f = open(fname, 'rb')
        data = f.read()
        f.close()

        #print(chardet.detect(data))
        #print(fname)

        fname = open(fname, 'r+', encoding=chardet.detect(data)['encoding'])
        ins_str = fname.read()
        fname.close()

        ins_dict = {}
        ins_str = cutsyms(ins_str)
        ins_str = cutstopwords(ins_str)
        ins_str = stemming(ins_str)
        ins_dict = wordcount(ins_str)

        # tf_idf
        sum = 0
        outstr = ""
        for seg in ins_dict.keys():
```

```
            sum = sum + ins_dict[seg]
        for seg in ins_dict.keys():
            tf = ins_dict[seg] / sum
            if seg in  str_dict.keys():
                idf = math.log(num_txt / num_dict[seg])
            else:
                idf = 0
            ins_dict[seg] = tf * idf
            if ins_dict[seg]!=0:
                outstr = outstr + seg + ":" + str(ins_dict[seg]) + " "

        full_path = '/Users/apple/Desktop/ir/vsmresult01.txt'
        file = open(full_path, 'a+')
        file.write(outstr+"\n\n")
        file.close()
```

实验过程中遇到和解决的问题：

（记录实验过程中遇到的问题，以及解决过程和实验结果。可以适当配以关键代码辅助说明，但不要大段贴代码。）

1. 读取文件问题：使用树形读取
2. 词袋过大：去除高频或者低频词

结论分析与体会： 初步建立起词袋模型，对之后的文本处理有了一定对代码与数据基础