# 实验一　VSM

本实验先通过 tokenization, stemming,去 stopwords 等步骤对文档集 **20 Newsgroups dataset** 进行了预处理，然后建立了该文档集的词袋，并对其中每个文档建立了向量空间模型，最后用 TF-IDF 对其进行了改进。

## 1. 读取文档

读取文档时需要用到 python 自带的 os 模块遍历文档集中的每一个文档，以二进制方式打开，借助 chardet 模块的 detect 函数检测每个文档的编码方式，对一个文档中的每一行按其编码方式解码，并 join 为一个字符串。最终得到一个包含所有文档的列表，其中每个元素是由一个文档构成的字符串。

```
# 二进制方式读取，获取字节数据，检测类型
def get_encoding(file):
    with open(file, 'rb') as f:
        return chardet.detect(f.read())['encoding']
```

## 2. 预处理

建立一个文档预处理的函数 filter，借助 nltk 模块的 tokenization，stem，去 stopwords 等功能对文档进行预处理，由于还有一些没用的标点符号、连字符没被处理，自建了一个包含各种标点符号的列表，对出现在其中的符号进行删除，对连字符用空格进行了替换，另外将大写统一为了小写。

```
# 去标点
punctuation = [',', '<', '>', '.', "'s", '`', '~', '!', '@', '#', '$', '%', '^',
               '&', '*', '(', ')', '_', '', '=', '+', '[', ']', '{', '}', '\\', '|',
               ':', ';', "\'", '/', '?',"\"","“","’","``"]
filtered2 = [i for i in filtered1 if i not in punctuation]
```

## 3. 建立词袋并对每个文档统计词频

导入 scikit-learn 的 CountVectorizer，用 fit（）函数建立预处理后的文档集的词袋，并用 transform 函数对每个文档编码词频信息，得到一个所包含单词的稀疏向量。

```
# 对每个文档统计词频
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer()
# 建立词典
vectorizer.fit(sumData)
print(vectorizer.vocabulary_)

lenth = len(sumData)
for i in range(lenth):
    vector = vectorizer.transform([sumData[i]])
```

## 4. 用 TF-IDF 改进

借助 scikit-learn 的 TfidfVectorizer 建立文档集的词汇表并计算 idf 得分，再用 transform 函数对每个文档编码基于 TF-IDF 的词频得分。

```
# 对每个文档用 tf-idf 统计词频
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer1 = TfidfVectorizer()
# 建立词汇表
vectorizer1.fit(sumData)

lenth = len(sumData)
for i in range(lenth):
    vector1 = vectorizer1.transform([sumData[i]])
```

通过这次实验，我加深了对向量空间模型的理解，熟悉了 python 的编程语言。