**Msg\_cluster**

说明：本项目为python语言实现的基于simhash短信相似度聚类工具

**环境依赖**

* python
* python-jieba
* bash

**目录说明**

data 文件夹

-stopwords.txt （停用词表）

-userdict.txt（用户关键词）

data/temp 文件夹 （存放中间结果文件和文件夹，文件中每一行均表示一个文档）

-\*.src 短信的原始文本（有数字日期）

-\*.ori 过滤掉数字字母，可用于检测的原始文本（去噪）

-\*.token 中文分词结果

-word.dict 根据分词结果生成的特征词典

-\*.feat 特征向量文件

-\*.fprint Simhash信息指纹文件

-\*.min Minhash 文件

-\*.cluster 聚类结果详情文件

-\*.summary 聚类结果摘要文件

src/ 文件夹

源程序

**代码使用说明**

**对短信进行聚类分析**

**短信聚类 (cluster\_msg.py)**

brief: 对短信列表按simhash+minhash聚类

INPUT: 停用词表 + 用户关键词表 + 短信列表

OUTPUT: 将聚类结果保存到\*.cluster文件中，将聚类摘要保存到\*.summary中，并生成特征词典文件

usage:

cluster\_msg.py <stopword\_path> <user\_dict> <message.txt> <cluster\_path> <summary\_path>

e.g.

cluster\_msg.py ../data/stopwords.txt ../data/userdict.txt ../data/msg2.txt ../data/msg2.cluster ../data/msg2.summary

{Note: 用户关键词表指定了重点关注的词及权重，分词时不会被切割，计算simhash时会乘上该词的权重，权重越大，对分类的聚合影响越大}

**详细处理流程（单步）**

**去噪 (msg-filter.sh)**

brief: 原始文本的初步去噪（去特殊符号、英文字母、数字 ...），消除连续空格以及删除空白行

INPUT: 待去噪文本 (.src)

OUTPUT: 去噪后的文本 (.ori)

usage:

msg\_filter.sh <\*.src> <\*.ori>

e.g.

msg-filter.sh data/temp/all.src data/temp/all.ori

**预处理**

**中文分词(tokens.py)**

brief: 采用Jieba分词器对去噪后的原始文本进行中文分词

INPUT: 去噪后的文本 (.ori)

OUTPUT: 中文分词结果 (.token)

usage:

./tokens.py -s/-m <\*.ori/inputfolder> <\*.token/outputfolder> c/s[mode] <stopword.list>

-s[single]/-m[multiple] 对单个文本文件 (\*.ori) 或对文本文件目录进行分词

-s <\*.ori> <\*.token>

-m <inputfolder> <outputfolder> {Note: 采用-m模式时，原始文本名最好以.ori结尾}

c/s[mode] Jieba分词器模式选择

c模式 jieba.cut(...)

s模式 jieba.cut\_for\_search()

e.g.

tokens.py -s ../data/temp/all.ori ../data/temp/all.token c ../data/stopwords.txt

**生成特征词典 (workdict.py)**

brief: 根据分词结果文件或目录，生成以词频降序排列的特征词典

INPUT: 中文分词结果 (.token)

OUTPUT:生成的特征词典，词典格式如下：ID + 特征词 + 词频

usage:

worddict.py <input\_folder/\*.token> <output\_file>

e.g.

worddict.py ../data/temp/all.token ../data/temp/word.dict

**生成特征向量 (features.py)**

brief: 根据分词结果和特征词典，生成特征向量文件

INPUT: 第一步处理中分词后的文本 + 第二步生成的特征词典

OUTPUT: 以行为单位生成各文档的特征向量：id1:nonzero-tf id2:nonzero-tf ...

usage:

feature.py -s/-m <word\_dict\_path> <tokens\_file/tokens\_folder> <feature\_file/feature\_folder>

-s[single]/-m[multiple] 对单个分词文件 (\*.token) 或对分词文件目录生成特征向量

e.g.

feature.py -s ../data/temp/word.dict ../data/temp/all.token ../data/temp/all.feat

**生成Simhash指纹 (simhash.py)**

brief: 根据分词结果和特征词典，生成信息指纹文件

INPUT: 特征词典 + 特征向量文件

OUTPUT: 信息指纹文件

usage:

simhash.py <word\_dict\_path> <\*.feat> <\*.fprint>

e.g.

simhash.py ../data/temp/word.dict ../data/temp/all.feat ../data/temp/all.fprint

**生成Minhash指纹 (minhash.py)**

brief: 特征词典 + 特征向量文件，生成用于分箱的minhash文件

INPUT: 特征词典 + 特征向量文件

OUTPUT: minhash文件

usage:

minhash.py <word\_dict\_path> <\*.feat> <\*.min>

e.g.

minhash.py ../data/temp/word.dict ../data/temp/all.feat ../data/temp/all.min

**对短信进行聚类 (cluster.py)**

brief: 根据分词结果和特征词典，生成聚类结果详情文件和摘要文件

INPUT: 特征词典 + 特征向量文件

OUTPUT: cluster文件 + summary文件

usage:

cluster.py <word\_dict\_path> <\*.feat> <\*.cluster> <\*.summary>

e.g.

minhash.py ../data/temp/word.dict ../data/temp/all.feat ../data/temp/all.cluster ../data/temp/all.summary