Tmsvm- Text Mining System based on SVM

基于SVM的文本挖掘系统

|  |  |
| --- | --- |
| 版本 | 0.8 |
| 作者 | 张知临 |
| 联系方式 | zhzhl202@163.com |
| 最后更新 | 2011/11/7 |
| 系统主页 | <http://code.google.com/p/tmsvm/> |

# 简介：

文本挖掘无论在学术界还是在工业界都有很广泛的应用场景。而文本分类是文本挖掘中一个非常重要的手段与技术。现有的分类技术都已经非常成熟，SVM、KNN、Decision Tree、AN、NB在不同的应用中都展示出较好的效果，前人也在将这些分类算法应用于文本分类中做出非常多的工作。但在实际的商业应用中，仍然有很多问题需要待解决，其中之一就是文本分类中的高维性和稀疏性，这个问题将导致curve of dimension，以及过拟合，导致训练的模型在预测未知数据时效果很差。

这个开源系统的目的是集众人智慧，将文本挖掘、文本分类效果非常好的算法实现并有效组织，并将文本分类的过程自动化，从词典生成、特征选择到分类模型参数选择、分类模型训练预测都可以一步完成。

## 主要特征

该系统在封装libsvm、liblinear的基础上，又增加了特征选择、LSA特征抽取、SVM模型参数选择、libsvm格式转化模块以及一些实用的工具。其主要特征如下：

1. 基于Chi的feature selection
2. 基于LSA的feature extraction
3. 文本特征向量的归一化
4. 支持Binary,Tf,Tf\*Idf,tf\*Chi等多种特征权重
5. 利用交叉验证对SVM模型参数自动选择。
6. 封装并完全兼容libsvm、liblinear。
7. 支持macro-average、micro-average、F-measure、Recall、Precision、Accuracy等多种评价指标
8. 将文本直接转化为libsvm、liblinear所支持的格式。
9. 采用python的csc\_matrix支持存储大稀疏矩阵。

## 输入格式：

label value1 [value2]

其中label是定义的类标签，如果是binary classification，建议positive样本为1，negative样本为-1。如果为multi-classification。label可以是任意的整数。

其中value必须为已经分好词的文本。可以利用ICTCLAS等分词工具预先对文本进行分词。

## 调用方法：

本系统提供了多种调用方式：

1、对

$python auto\_tms\_train.py trainfile 对trainfile中数据自动进行SVM模型训练。从词典生成、特征选择、SVM模型参数选择、SVM模型训练一步完成。另外还有其他的参数可以设置。具体调用见

$python tms\_train.py –s (1|2|3|4|5) trainfile 。对trainfile中的数据进行SVM模型训练。并支持分步进行。1为自动进行模型训练，和auto\_tms\_train.py功能相同。2为进行特征选择。3为生成SVM模型的输入格式。4为SVM模型的参数选择。5为SVM模型训练。另外还有其他的参数可以设置。具体调用见

$python tms\_predict.py testfile dic\_path model\_path 。给定词典以及训练好的模型，对testfile进行训练。具体调用见

$python lsa \_tms\_train.py trainfile svmmodel M。指定已经训练好的svm模型以及词典的长度，即可以对trainfile进行LSA模型的训练。现在实现的LSA算法是在SVM模型预测的基础上划定local region，并进行local SVD,所以LSA模型训练的时候需要使用已经训练好的SVM模型。

$python lsa \_predict.py testfile dic\_path model\_path lsa\_path lsa\_model\_path 。给定词典、SVM模型、LSA中u矩阵保存路径、LSA模型路径，即可对testfile进行预测。

$python result\_analysis.py datafile ，对结果数据进行分析。可计算

macro-average,micro-average,f-measure,recall,precision,以及设定特定的阈值(binary classification)计算f-measure,recall,precision，以及对设定范围，对该范围的所有阈值计算f-measure,recall,precision，从而选定最好的阈值。

## 本系统欲解决的问题：

1. 文本分类的高维性和稀疏性
2. Unbalance 样本的问题。实际应用中，各个类别的数据集大小往往是不平衡的，尤其在information filtering 领域。感兴趣的数据集相对于不感兴趣的数据集是非常小
3. Unlabeled 样本的利用问题。
4. 样本的Random sample 问题。

## 程序文件说明

**src:**即该系统的源代码，提供了5个可以在Linux下可以直接调用的程序:

auto\_tms\_train.py：SVM模型自动训练程序。从词典生成、特征选择、SVM模型参数选择、SVM模型训练一步完成。

tms\_train.py：SVM模型训练程序，可自动训练也可以分布训练，通过参数-s 设置。1为自动进行模型训练，和auto\_tms\_train.py功能相同。2为进行特征选择。3为生成SVM模型的输入格式。4为SVM模型的参数选择。5为SVM模型训练。

tms\_predict.py：SVM模型预测程序。

lsa\_tms\_train.py：LSA模型训练程序。

lsa\_predict.py：LSA模型预测程序。

ctm\_train\_model.py是模型训练的主程序，里面提供了多种训练的调用函数。ctm\_predict\_model.py是模型预测的主程序，里面提供了多种预测所用的函数。可以直接调用。

**tools:**提供的一些有用的工具，包括result\_analysis.py等。

**java：**java版本的模型预测程序，其中ctm\_predict.java为为社区帖子监控模型所写的预测程序。tms\_predict.java 为通用的模型预测程序。可以支持多个SVM模型同时预测。

# 程序调用接口

## 自动文本SVM分类模型训练auto\_tms\_train.py

1. **说明**

此函数为文本SVM分类模型**自动**训练程序，给定训练文本及设置相应参数，即可得到训练好的模型。

1. **调用示例：**

*usage:%prog [options] filename*

完整形式：

Python auto\_tms\_train.py –p ../ -i 6,7,8,9,10 –w –d dic.key

–m im. model -t tms.train –a tms.param –r 0.4 –S ^ ../im.train

精简形式：

python auto\_tms\_train.py -i 6,7,8,9,10 ../im.train

其中im.train为训练文本的名称，其里面的内容如下，指定将第6,7,8,9,10位的内容一同放入模型训练。训练后的模型保存在源程序的上一层目录。

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 类别 | 编号 | 时间 | 发送方 | 接收方 | 文本内容 | 分词的的内容 | 链接形式 | 有无QQ | 商品是否属于其中一方 | 商品属性 |
| 1 | qizha1035577 | ######## | 4lile3uo842c | 李英0821 | 你好，这几天本店装修!旺旺不能打字！如要购买商品的请 加我 qq 61517891 联系我购买 ！为表示歉意！一律包邮费加优惠！谢谢 | 你好^^，^^这^^几^^天^^本^^店^^装修^^!^^旺旺^^不^^能^^打字^^！^^如^^要^^购买^^商品^^的^^请^^加^^我^^qq61517891^^联系^^我^^购买^^！^^为^^表示^^歉意^^！^^一律^^包邮费^^加^^优惠^^！^^谢谢 | no\_link\_msg | QQQQQQQ | no\_auction\_exist | no\_auction\_here |

1. **输入格式**

第一列为类别，如果为binary分类。最好Positive为1，Negative为-1。如果为多分类，正常样本为-1，其余的类可以选择1,2,3,4……

其余列为内容，可以有多列内容，然后在训练模型时指定训练的列数(

-i 3,4,5 即指定第4,5,6列训练—默认列数从0开始)。

1. **结果**

模型结果会放在“model”文件夹中，里面有两个文件，默认情况下为dic.key 和 tms.model 。其中dic.key为特征选择后的词典；tms.model为训练好的SVM分类模型。

临时文件会放在“temp”文件夹中。里面有两个文件：tms.param和tms.train。其中tms.param为SVM模型参数选择时所实验的参数。tms.train是供libtms训练器所使用的输入格式。

1. **参数说明**：

*-p，--path，模型保存的路径。默认为 ”../”*

*-i，--indexes,输入文本中训练的模型的部分（从0开始编号）**，默认为[1]，输入时可用 –i 1,2,3 表示使用第1,2,3作为训练的内容*

*-w，(布尔型)如果使用此参数代表词典中不去除停用词。如果使用，必须将停用词文件以stopwords.txt 命名，和训练文本放在同一路径下。默认情况下不使用此参数，即需将停用词文件stopwords.txt放在训练文本同一路径下*

*-A ,--tms\_param。即SVM训练的参数，完全兼容libtms，输入的格式为 –A “-s 0 –c 1.0 –g 0.25” 。为了避免和现在的参数相混淆，所以要加上双引号。*

*-d，--dic\_name。指定特征选择后词典的名称,默认为dic.key*

*-m, --model\_name,指定生成的分类模型的名称，默认为tms.model*

*-t, --train\_name,指定生成的分类模型的名称，默认为tms.train*

*-a, --param\_name,指定生成的分类模型的名称，默认为tms.param*

*-r，--ratio。指定特征选择保留词的比例。默认为0.4*

*-T，--tc\_splitTag。训练文本中各部分分割的符号，默认为”\t”*

*-S*,*--str\_splitTag.训练文本中分词的分割词，默认为”^”*

1. **Note:**

正常情况下，训练一个模型会经过特征选择，生成SVM输入格式，SVM参数选择，SVM模型训练这几个步骤。其中SVM参数选择将花费较长的时间(为了能训练出最好的模型，程序默认会进行两轮参数搜索，粗粒度和细粒度，共实验150对(c,g)，其中每对(c,g)都要经过4-flods的交叉验证。如果训练样本的个数在5000以下，整个模型的训练时间在十分钟左右，如果训练文本的数量级为万，则训练时间将达到几个小时。)。如果忍受不了这么长的时间。可以进行分步训练，或者是选择在晚上进行训练。

1. **调用示例**

**$ cat set.train**

1 ^台^军^人事^大幅^变动^ ^据^台湾^东森^新闻^报道 ^台^军^将领^将^有^大^调动^……

1 朝鲜^已经^准备^好^对^美国^进行^先发制人^的^打击 ^朝鲜^人民^武装^力量^部^部长^金一^哲^次帅^4月^8日^在^平壤^公开^表示……

1 ^伊朗^开始^大规模^投产^地对空导弹 报道^还^援引^伊朗^国防^部长^穆斯塔法^·^穆罕默德^·^纳贾尔^话说……

……

-1 五一^期间^外出^旅游^人数^仍然^保持^上升^趋势 记者^在^甘肃^、^宁夏^两^省^采访^时^发现^……

-1 承德^避暑山庄^永佑^寺^全新^的^面貌^与^游客^“^见面 ^记者^从^承德市^旅游局^获悉^，^目前^永佑^寺^内^“^陆^合^塔……

-1 强迫^购物^、^强迫^参加^自费^活动 许多^游客^游^完^泰国^回来^总会^表示^不满^。^“^去^之前^交^一次^团费……

……

训练文本总共有两类，每行代表一类的样本，总共有3个字段，第一个字段为类别，第二个类别为标题，第3个字段为内容。标号为1的类为军事类，-1的为旅游类。

## 文本SVM分类模型训练 tms\_train.py

1. 说明

此函数为文本SVM分类模型训练程序，与auto\_tms\_train.py不同的是，该函数可以使模型训练分步进行。

1. 调用示例：

*usage:%prog [options] filename*

*filename 在1),2),3)中代表输入训练文本，在4),5)中代表SVM的输入格式。*

其中im.train为训练文本的名称，其里面的内容如下，指定将第6,7,8,9,10位的内容一同放入模型训练。训练后的模型保存在源程序的上一层目录。

1. 自动进行模型训练

完整形式

python tms\_train.py -s 1 –p ../ -i 6,7,8,9,10 –w –d dic.key

–m im. model -t tms.train –a tms.param –r 0.4 –S ^ -T ^M ../im.train

精简形式：

python tms\_train.py –s 1 –p ../ -i 6,7,8,9,10 ../im.train

1. 特征选择

完整形式：

python tms\_train.py -s 2 –p ../ -i 6,7,8,9,10 –d dic.key

–r 0.4 –S ^ ../im.train

精简形式

python tms\_train.py -s 2 –p ../ -i 6,7,8,9,10 ../im.train

1. 生成SVM模型的输入格式

完整形式：

python tms\_train.py -s 3 –p ../ -i 6,7,8,9,10 –D ../dic.key

–S ^ ../im.train

简洁形式：

python tms\_train.py -s 3 –p ../ -i 6,7,8,9,10 –D ../dic.key ../im.train

1. SVM模型的参数选择

完整形式：

python tms\_train.py -s 4 –p ../ -P ../tms.train

其中tms.train 为第3步生成的SVM模型的输入格式

1. 模型训练

完整格式：

python tms\_train.py -s 5 –p ../ -P ../tms.train –A “-c 0 –t 2”

1. 输入格式

第一列为类别，如果为binary分类。最好Positive为1，Negative为-1。如果为多分类，正常样本为-1，其余的类可以选择1,2,3,4……

其余列为内容，可以有多列内容，然后在训练模型时指定训练的列数(

-i 3,4,5 即指定第4,5,6列训练—默认列数从0开始)。

1. 结果

模型结果会放在“model”文件夹中，里面有两个文件，默认情况下为dic.key 和 tms.model 。其中dic.key为特征选择后的词典；tms.model为训练好的SVM分类模型。

1. 参数说明：

*-s ,--step,即选择要进行的操作。1为自动训练模型，即auto\_tms\_train.py的功能。2为特征选择。3为根据训练样本生成SVM的输入格式。4为SVM模型参数选择；5为SVM训练*

*-p，--path，模型保存的路径，默认情况下为”../”*

*-i，--indexes,输入文本中训练的模型的部分（从0开始编号），默认为[1]，输入时可用 –i 1,2,3 表示使用第1,2,3作为训练的内容*

*-w，(布尔型)如果使用此参数代表词典中不去除停用词。如果使用，必须将停用词文件以stopwords.txt 命名，和训练文本放在同一路径下。默认情况下不使用此参数，即需将停用词文件stopwords.txt放在训练文本同一路径下*

*-A ,--tms\_param。即SVM训练的参数，完全兼容libtms，输入的格式为 –A* *“-s 0 –c 1.0 –g 0.25” 。为了避免和现在的参数相混淆，所以要加上双引号。默认为“-s 0 –c 1.0 –g 0.25”*

*-d，--dic\_name。指定特征选择后词典的名称,默认为dic.key*

*-D,--dic\_path。词典所在的路径及名称*

*-m, --model\_name,指定生成的分类模型的名称，默认为tms.model*

*-t, --train\_name,指定生成的分类模型的名称，默认为tms.train*

*-a, --param\_name,指定生成的分类模型的名称，默认为tms.param*

*-r，--ratio。指定特征选择保留词的比例。默认为0.4*

*-T，--tc\_splitTag。训练文本中各部分分割的符号，默认为”\t” -S*,*--str\_splitTag.训练文本中分词的分割词，默认为”^”*

1. Note:

## 模型预测程序

1. **说明**

此函数为文本SVM分类模型**预测**程序，给定测试文本及设置相应参数，即可为样本进行预测。

1. **调用示例：**

*usage:%prog [options] filename dic\_path model\_path*

完整形式：

python tms\_predict.py -f ../ sample.test -R ../ result/score.result -i 1,2

-D ../model/dic.key -M ../model/tms.model -r 0,1,2

其含义为对sample.test 中的第1,2列(列从0开始，1,2要融合在一起)进行预测，结果放在score.result文件中，其中第一列为分数，其余列为指定的第0，1,2列。指定词典以及训练好的模型。

1. **输入格式**

没有特定的输入格式。具体需要预测的内容可以通过 –i 指定。

1. **结果**

预测的结果会写入到指定的文件中。其中第一列为预测的分数。其余列为指定的需要同结果一同输出的内容。

1. **参数说明**：

*-i，--indexes,输入文本中训练的模型的部分（从0开始编号），默认为1，输入时可用 –i 1,2,3 表示使用第1,2,3作为训练的内容*

*-r，----result\_indexes。指定与预测分数一块输出的文本的指标项，其中预测分数放在第一列，其余的依次排列。默认为1，调用方式为 –r 1,2,3*

*-R,--result\_save 。结果保存的路径及文件名称。*

*-T，--tc\_splitTag。训练文本中各部分分割的符号，默认为”\t”*

*-S*,*--str\_splitTag.训练文本中分词的分割词，默认为”^”*

## 社区帖子监控模型预测程序

因为社区帖子需要为标题和标题+内容建立两个模型，所以在监控程序上需要做些修改，把两个模型都加入到检测上。其实调用两次[模型预测程序](#_模型预测程序)也可以实现，花费的时候也不会太多。

## LSA模型训练程序

1. **说明**

此函数为LSA模型**训练**程序，给定测试文本及训练好的SVM模型和词典的长度，即可为LSA模型进行训练。

LSA模型虽然在实际实验中没有得到预想的效果，但是在某些场景下应该会有用，所以LSA模型训练与预测的程序仍然保留。

1. **调用示例：**

*usage:%prog [options] filename svm\_model M*

完整形式：

python lsa\_tms\_train.py -f ../ sample.test -R ../ result/score.result -i 1,2

-D ../model/dic.key -M ../model/tms.model -r 0,1,2

其含义为对sample.test 中的第1,2列(列从0开始，1,2要融合在一起)进行预测，结果放在score.result文件中，其中第一列为分数，其余列为指定的第0，1,2列。指定词典以及训练好的模型。

1. **输入格式**
2. **结果**

**参数说明**：

*-p，--path，模型保存的路径，默认情况下为”../”*

*-e, --threshold 。LSA模型选取top n阈值。默认情况下为1.0*

*-K,--K 。选取的前k个特征根。*

*-f,--for\_lsa\_train 。SVM模型预测训练文本，并构造适合LSA模型的训练文本。默认为“for\_lsa.train”*

*-t,--train\_name；LSA模型做出的SVM训练文本格式,默认为“lsa.train”*

*-m,--model\_name；LSA模型的名称.默认为“lsa.model”*

*-A,--tms\_param；即SVM训练的参数，完全兼容libtms，输入的格式为 –A “-s 0 –c 1.0 –g 0.25” 。为了避免和现在的参数相混淆，所以要加上双引号。默认为“-s 0 –c 1.0 –g 0.25”*

*-a, --param\_name,指定生成的分类模型的名称，默认为tms.param*

## LSA模型预测程序

1. **说明**

此函数为文本LSA模型**预测**程序，给定测试文本及设置相应参数，即可为样本进行预测。

1. **调用示例：**

*usage:%prog [options] filename dic\_path model\_path sa\_path lsa\_model\_path*

完整形式：

**Python lsa\_predict\_py -i 6 -r 0 -R** **im\_lsa\_20.result lsa.test im.key im.model lsa lsa.model**

其含义为对**lsa.test** 中的第6列(列从0开始)进行预测，结果放在**im\_lsa\_20.result**文件中，并将原文件的第0列和结果一起输出。其中词典为im.key，SVM模型为im.model，LSA 矩阵前缀为lsa，LSA模型为lsa.model。

1. **输入格式**

没有特定的输入格式。具体需要预测的内容可以通过 –i 指定。

1. **结果**

预测的结果会写入到指定的文件中。其中第0列为SVM模型预测的分数，第1列为LSA模型预测分数。其余列为指定的需要同结果一同输出的内容。

1. **参数说明**：

*-i，--indexes,输入文本中训练的模型的部分（从0开始编号），默认为1，输入时可用 –i 1,2,3 表示使用第1,2,3作为训练的内容*

*-r，----result\_indexes。指定与预测分数一块输出的文本的指标项，其中预测分数放在第一列，其余的依次排列。默认为1，调用方式为 –r 1,2,3*

*-R,--result\_save 。结果保存的路径及文件名称。*

# 一些有用的工具

## 选择子集 subset.py

1. 说明

此函数为从大数据集总选择较小的子集。

1. 调用示例：

Usage: %s [options] dataset number [output1] [output2]

python subset.py –s 0 data.txt 3000 output1.txt output2.txt

即从data.txt中选择3000个样本，并将选择的子集输出到output1.txt，其余的部分输出到output2.txt中。其中参数-s 如果为0，则子集选择为分层抽样，即子集仍然会保持原数据集中各个类的比例。如果-s 为1，则从源数据集中随机选择，不会考虑各个类的比例。

1. 输入格式
2. 如果设置-s 0则，需要以下输入格式：

第一列为类别，如果为**binary**分类。最好Positive为1，Negative为-1。如果为**多**分类，正常样本为-1，其余的类可以选择1,2,3,4……

其余列为内容，可以有多列内容。

1. 如果设置-s 1,则输入格式没有限制
2. 结果

选择的子集输出到output1.txt，其余的部分输出到output2.txt中

1. 参数说明：

-s 子集选择的方法，默认为0

0 – 分层抽样

1 – 随机选择

output1 : 子集的输出(optional)

output2 : 剩余部分的输出 (optional)

## SVM参数选择 grid.py

1. 说明

此函数为从为SVM模型搜索最优的参数(c,g)。

1. 调用示例：

Usage: grid.py [-log2c begin,end,step] [-log2g begin,end,step] [-v fold]

[-tmstrain pathname] [-gnuplot pathname] [-out pathname] [-png pathname]

[additional parameters for tms-train] dataset

示例：python grid.py –log2c -5,15,2 –log2g -3,13,2 –v 5 tms.train

即对tms.train进行参数搜索，c从-5到15，步长为2，g从-3到13，步长为2。其中tms.train是libtms特定的输入格式：

<label> <index1>:<value1> <index2>:<value2> ...

<label>为类标签。<index1>整数，必须要按升序排列。可以通过调用[OLE\_LINK19](#OLE_LINK19) 来生成相应的输入格式。

1. 输入格式

<label> <index1>:<value1> <index2>:<value2> ...

.

.

<label>为一个整数值，代表着类标签。<index>:<value>指定对应特征上的值。<index>是从1开始的整数值，必须保持升序。如果第i个特征上的值为0，则可以省略不写。

1. 结果

该程序产生的结果文件较多，多为一些中间结果，最优的c与g可以在控制台查看。

1. 参数说明：

-log2c begin,end,step 设置参数c搜索的范围及步长

-log2g begin,end,step设置参数g搜索的范围及步长

-v fold 设置交叉验证的folds，默认情况下为5

-tmstrain pathname 设置tms-train程序的路径,默认为当前路径

-gnuplot pathname设置gnuplot程序的路径，默认为/usr/bin/gnuplot

1. Note

对于大数据集(样本>5000)，一般要进行两步：先选取子集[选择子集 subset.py](#_选择子集_subset.py)然后进行粗粒度搜索，再对全数据集进行细粒度搜索。

粗粒度搜索是指在一个较大的范围内加大步长。细粒度搜索是指在进行粗粒度搜索后，得到最优的c,g，然后在这个值周围选取局部区域，调小步长，再进行搜索。即可得到最优的参数

## 结果分析程序

Result\_anlaysis.py

如果需要对模型的参数进行调整，则需要对模型预测的指标进行计算。该模块主要读出程序的结果，然后计算相应的F值、召回率、正确率，以及做一些统计分析，并能根据相应的指标选择最优的阈值。

1. **说明**

可以对分类的结果进行统计分析。包括分类准确率、F值、召回率、准确率、宏观分类准确率、微观分类准确率、设定阈值的F值、召回率、准确率。

1. **调用示例：**

*usage:%prog [options] filename*

完整形式：

1. **输入格式**

没有特定的输入格式。具体需要预测的内容可以通过 –i 指定。

1. **结果**

预测的结果会写入到指定的文件中。其中第0列为SVM模型预测的分数，第1列为LSA模型预测分数。其余列为指定的需要同结果一同输出的内容。

1. **参数说明**：

默认情况下将会对分类准确率、F值、召回率、

*-s ，--step 。选择的步骤：1为多分类的微观分类准确率，宏观分类准确率，所有类的分类准确率。如果二分类中得到的结果是预测的类标签，也可以用该函数计算。2为多分类中各个类别的F值、召回率、准确率。果二分类中得到的结果是预测的类标签，也可以用该函数计算。3为计算二分类中的各个类别的F值、召回率、准确率，数据中无论是预测的类标签还是预测分数都适用。4为二分类中计算设定阈值后每个类别的F值、召回率、准确率。*

*-i，--indexes 。输入的数据文件字段。默认情况下位[0,1]，即第0列位预测分（类标签），第1列为实际的类标签。*

*-p,--predicted\_index 。**指定预测分（类标签）字段的位置，默认为第0列*

*-t","--true\_index。指定实际类标签字段的位置，默认为第1列*

*-e","--threshold"。在-s 3时设定的阈值.可以设定阈值，计算在改阈值下的各个类别的F值、召回率、准确率。默认情况下为0*

*-o","—output，指定分析结果输出位置，默认为标准屏幕输出*

*-m","—min，设定阈值的最小范围，在-s 4时，指定搜索阈值的最小范围*

*-M","—max，设定阈值的最大范围，在-s 4时，指定搜索阈值的最小范围*

*通常搜索会在[min,max）下取步长0.1进行搜索*

# 技术细节

TSC：基于libtms的文本挖掘系统 TML：Text Mining System based on Libtms

特征选择使用的Chi方法，因为在参考文献的结果，Chi的方法时最好的。

term/feature weight 默认使用的是TF,另外还可以选择TF\*IDF,TF\*Chi等方法。根据参考文献的，TF是最便捷、有效的特征权重，而文献中常用的TF\*TDF反而会降低分类的效果。

Libsvm 3.0

Liblinear 1.8

Numpy 1.6.1

Scipy 0.9

# 源码剖析

请原谅作者在有些地方没有按照OOP思想设计程序，函数式的流程仅仅是为了能过更好的表达流程，并尽可能的做到函数的复用。

# 数据集

数据集中选用的搜狗语料库。所有文本

|  |  |  |
| --- | --- | --- |
| 类别 | 类别名称 | 数量 |
| 8.00 | 财经 | 1990 |
| 10.00 | IT | 1990 |
| 13.00 | 健康 | 1990 |
| 14.00 | 体育 | 1990 |
| 16.00 | 旅游 | 1990 |
| 20.00 | 教育 | 1990 |
| 22.00 | 招聘 | 1990 |
| 23.00 | 文化 | 1990 |
| 24.00 | 军事 | 1990 |

# 关于我