

Tmsvm

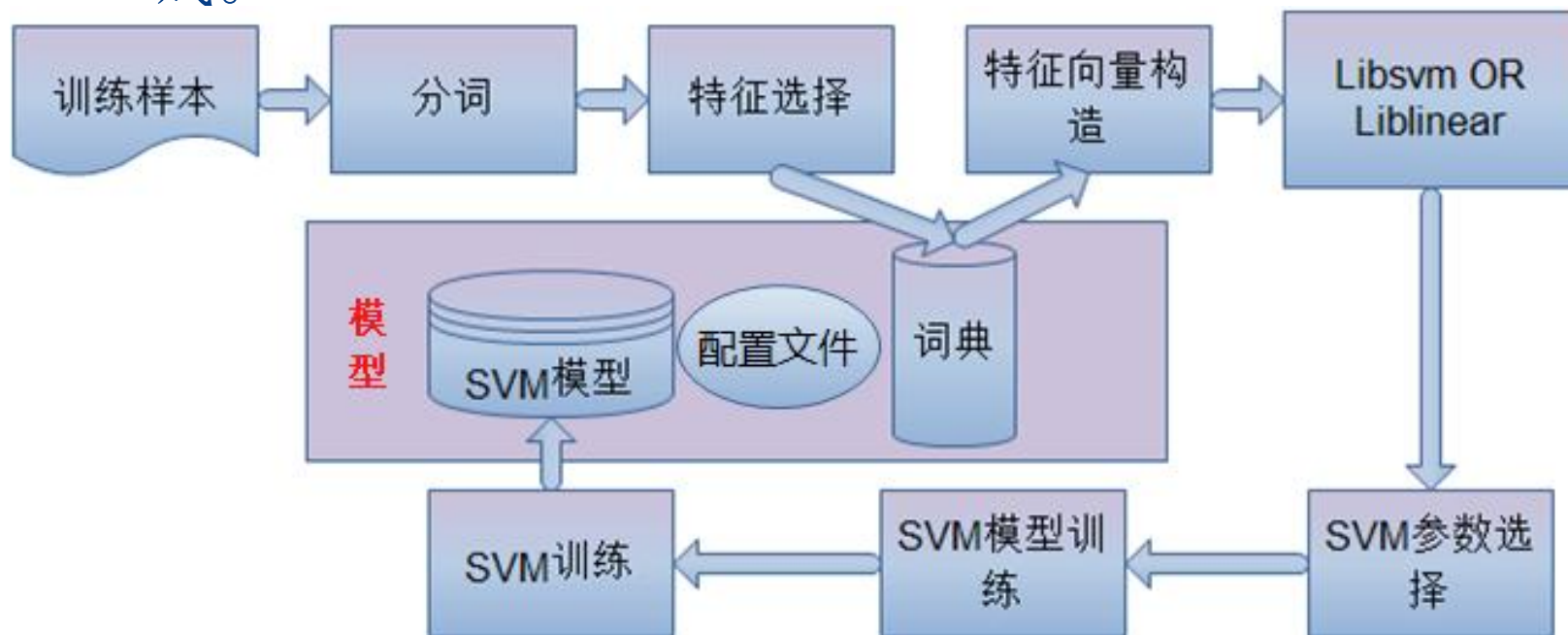
Text Mining System based SVM

张知临

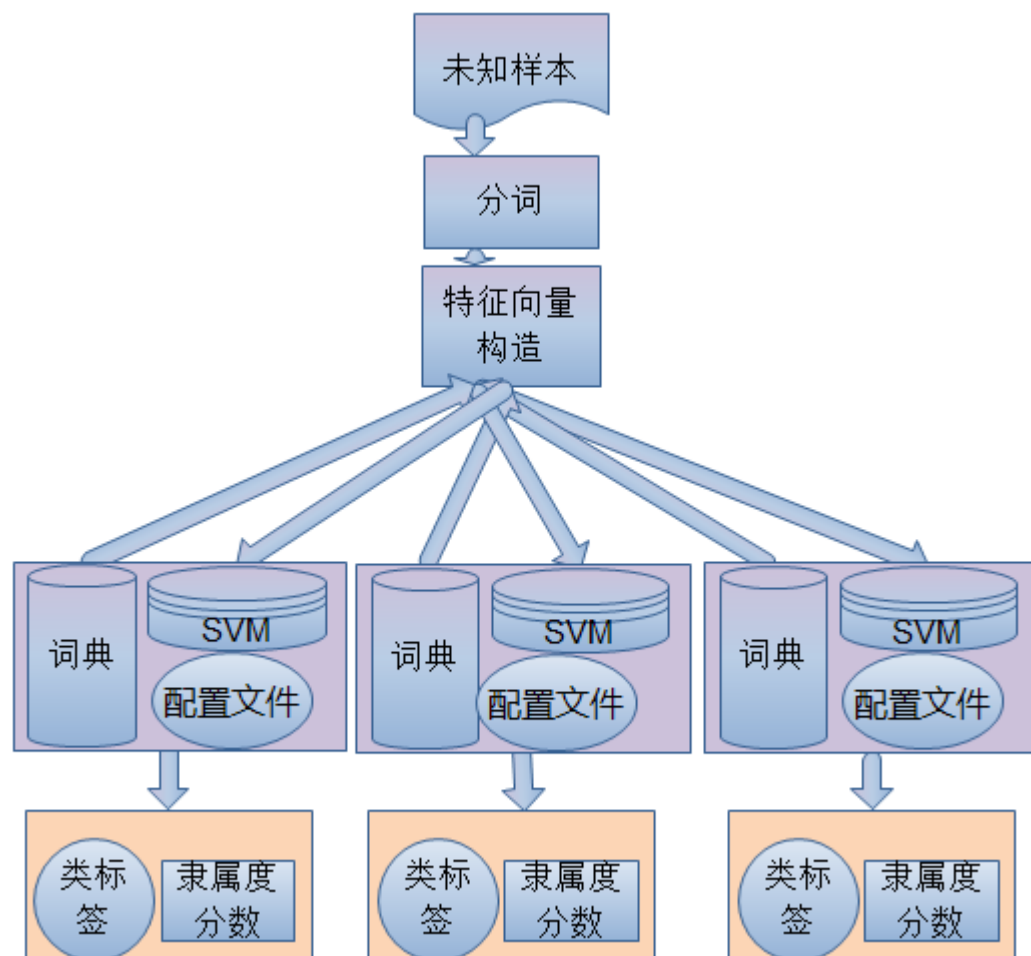
zhzhl202@163.com

该系统可以做什么

- * 对文本自动做SVM模型的训练。包括Libsvm、Liblinear包的选择，分词，词典生成，特征选择，SVM参数的选优，SVM模型的训练等都可以一步完成。



- * 利用生成的模型对未知文本做预测。可自动识别 libsvm 和 liblinear 的模型。



* 自动分析预测结果

- * 计算多分类以及二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。
- * 计算多分类以及二分类中所有类别的F值、召回率、准确率。
- * 计算多分类以及二分类中对指定的类别，对特定阈值下的F值、召回率、准确率。
- * 多分类以及二分类中计算所有类别的在阈值区间中的每个阈值每个类别的F值、召回率、准确率，旨在为用户分析出每个类别最好的阈值

- * **分词**。对文本利用**mmseg算法**对文本进行分词。
- * **特征选择**。对文本进行特征选择，选择最具代表性的词。
- * **SVM参数的选择**。利用**交叉验证**方法对SVM模型的参数进行识别，可以指定搜索范围，大于大数据，会自动选择子集做粗粒度的搜索，然后再用全量数据做细粒度的搜索，直到找到最优的参数。对libsvm会选择c,g(gamma)，对与liblinear会选择c。
- * 对文本直接生成libsvm、liblinear的**输入格式**。libsvm、liblinear以及其他诸如weka等数据挖掘软件都要求数据是具有向量格式，使用该系统可以生成这种格式：label index:value
- * **SVM模型训练**。利用libsvm、liblinear对模型进行训练。
- * 利用**LSA**对进行**Feature Extraction**，从而提高分类效果

该系统的特色

- * 封装libsvm、liblinear

- * Libsvm实现了多种核函数，在解决非线性SVM效果显著。liblinear用来实现了线性SVM多种算法，且可以应对百万级别的数据。
- * 文本分类一般线性可分，但有时映射到高维空间会提高分类效果。因此将这两者结合起来对在不同情况下的文本分类意义重大。
- * 两者虽然类似，但是在模型组成、参数选择等还是存在差异。系统将这两者进行包装，屏蔽其中的差异。

多种特征权重

支持Binary,Tf,log(tf),Tf*Idf,tf*rf,tf*chi等多种特征权重

特征权重一般公式: $a'_{ij} = L(i, j) * G(i)$

$Tf : L(i, j) = a_{i,j}; G(i) = 1$ $Tf * Idf : L(i, j) = \log(a_{i,j} + 1); G(i) = \log(\frac{N}{n})$

消息^快^评^深度^报告^权威^内参^来自^^证券^通^

词	id	G
深度	1	0.25
国家	2	0.32
股市	3	0.48
权威	4	0.36
消息	5	0.85
行情	6	0.23
证券	7	0.12
客服	8	0.21

ID	1	2	3	4	5	6	7	8	模
词	深度	国家	股市	权威	消息	行情	证券	客服	
局部因子	1	0	0	1	1	0	1	0	
全局因子	0.25	0.32	0.48	0.36	0.85	0.23	0.12	0.21	
特征权重	0.25	0	0.48	0	0.85	0	0.12	0	1.79
归一化	0.14	0	0.27	0	0.43	0	0.07	0	

基于LSA的Feature Extraction

$$\mathbf{t}_i^T \rightarrow \begin{bmatrix} x_{1,1} & \dots & x_{1,n} \\ \vdots & \ddots & \vdots \\ x_{m,1} & \dots & x_{m,n} \end{bmatrix} \quad \mathbf{d}_j = \begin{bmatrix} x_{1,j} \\ \vdots \\ x_{m,j} \end{bmatrix}$$

\mathbf{d}_j
↓

$$X = U \Sigma V^T$$

$$\begin{array}{ccccccc} & X & & U & & \Sigma & & V^T \\ & (\mathbf{d}_j) & & & & & & (\hat{\mathbf{d}}_j) \\ & \downarrow & & & & & & \downarrow \\ (\mathbf{t}_i^T) \rightarrow \begin{bmatrix} x_{1,1} & \dots & x_{1,n} \\ \vdots & \ddots & \vdots \\ x_{m,1} & \dots & x_{m,n} \end{bmatrix} & = & (\hat{\mathbf{t}}_i^T) \rightarrow \left[\begin{bmatrix} \mathbf{u}_1 \end{bmatrix} \dots \begin{bmatrix} \mathbf{u}_l \end{bmatrix} \right] & \cdot & \begin{bmatrix} \sigma_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma_l \end{bmatrix} & \cdot & \left[\begin{bmatrix} \mathbf{v}_1 \\ \vdots \\ \mathbf{v}_l \end{bmatrix} \right] \end{array}$$

$$X_k = U_k \Sigma_k V_k^T$$

原文档：m维 $d_m = X_{.j}$

经过LSA之后的文档：K维 $d_k = (\Sigma V^T)_{.j}$

新文档 $d_k = d^T U_k$

LSA应用于文本分类

- * LSA应用于文本分类

- * Global SVD

- * 没有考虑类别信息，过滤掉特征值较小但类别区分度大的特征

- * Local SVD

- * Based on class label

- * Local Relevancy Weighted LSI (LRW-LSI)

[Tao Liu](http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1410280) etc, Improving text classification using local latent semantic indexing
http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1410280

LRW-LSI

* 模型训练


- ① 训练初始分类器 C_0 。
- ② 对训练样本预测，生成初始分值 $f(rs_i) = \frac{1}{1 + e^{-a(rs_i+b)}}$
- ③ 文档特征向量变换 $\vec{d}_i' = \vec{d}_i * f(rs_i)$
- ④ 设定阈值，选择top n文档作为局部LSA区域
- ⑤ 对局部词/文档矩阵做SVD分解。得到U、S、V矩阵
- ⑥ 将其他的训练样本映射到U空间中
- ⑦ 对所有经过变换后的训练样本进行训练，得到LSA分类器

* 模型预测

- ① 利用 C_0 预测得到其初始分值
- ② 文档特征向量变换
- ③ 映射到U空间
- ④ 利用LSA模型进行预测得分

结果返回类别隶属度

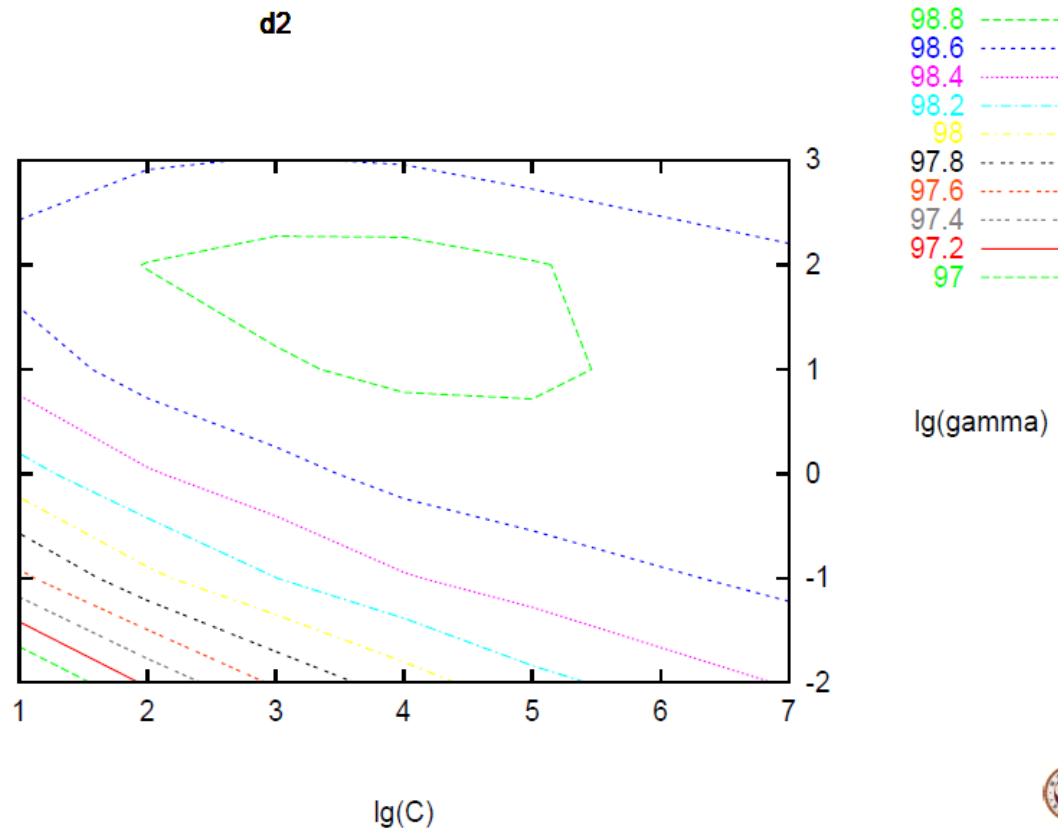
- * 对未知样本预测返回的结果有两个：
 - * Label: 即该样本所属的类别。
 - * Score: 该样本属于该类的隶属度，分值越大，代表属于该类的置信度越大。
 - * Score计算公式为 $score = \frac{\sum s_i}{2 * k} + \frac{k}{2 * n}$ ，其中k为所有支持判别类得个数，n为所有类别个数，si为所有支持判别类得分数
 - * 返回score的好处是对与information filtering问题，因为训练样本的unbalance和randomly sampling问题，依据判别的标签得到的结果准确率较低，因此需要通过阈值控制



SVM s	1:2	1:3	1:4	2:3	2:4	3:4	sum		
value s	0.95	1.1	0.5	0.6	-0.7	1.0	votes		value s
1	0.95	1.1	0.5	0	0	0	3	2.55	0.8
2	0	0	0	0.6	0	0	1	0.6	
3	0	0	0	0	0	1.0	1	1.0	
4	0	0	0	0	0.7	0	1	0.7	

SVM Model Selection

- * SVM模型的**好坏**在与**C**与**gamma**的选择
 - * C是惩罚系数，即对误差的宽容度。c越高，说明越不能容忍出现误差。C过大或过小，**泛化能力变差**
 - * gamma是选择**RBF**函数作为kernel后，该函数自带的一个参数。隐含地决定了数据映射到新的特征空间后的分布，gamma越大，支持向量越少，gamma值越小，支持向量越多。支持向量的个数影响**训练与预测的速度**。
- * **最优**的C与gamma与训练集**有关**
- * Grid Search
 - * 可以得到**全局最优**
 - * (C,gamma)相互独立，便于**并行化**进行。



The good region of parameters is **quite large**
SVM is sensitive to parameters, but not that sensitive
Sometimes default parameters work
but it's good to select them if time is allowed

Large Train Set SVM Model Selection

- * Problem:

- * 每对(c,g)都要进行n-flolds Cross Validation。如果在全量数据进行，需要花费较长的时间。

- * Solution

- * 粗粒度搜索

- * 选择较大的步长，找到一个最优的(c,g)

- * 细粒度搜索

- * 选择局部区域，较小的步长，找到全局最优的(c,g)

- * 对大数据集，先选择一个较小的子集做粗粒度搜索，然后再局部区域中，做细粒度搜索，找到全局最优的(c,g)

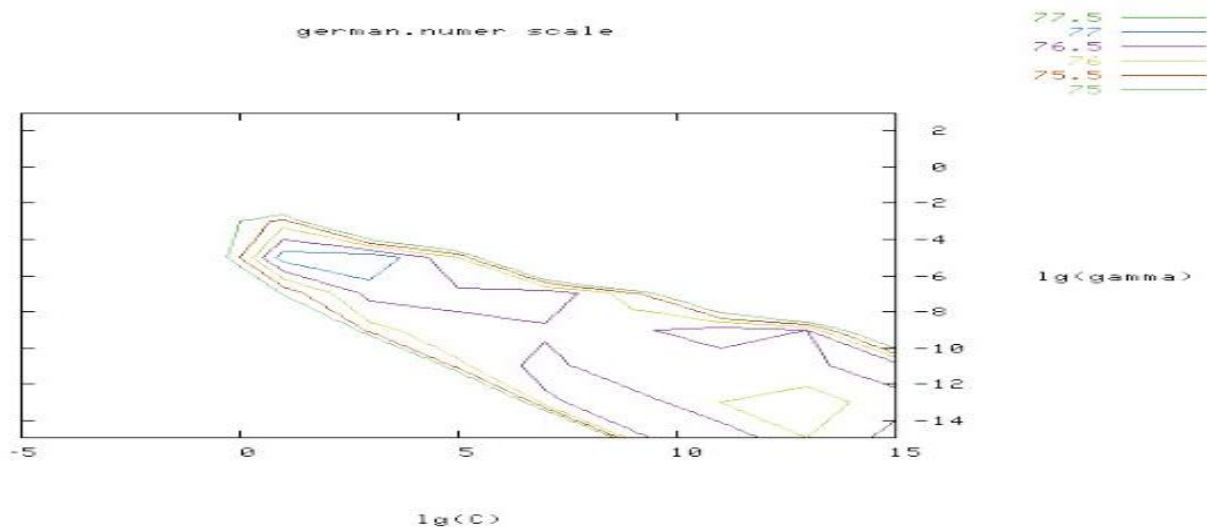


Figure 2: Loose grid search on $C = 2^{-5}, 2^{-3}, \dots, 2^{15}$ and $\gamma = 2^{-15}, 2^{-13}, \dots, 2^3$.

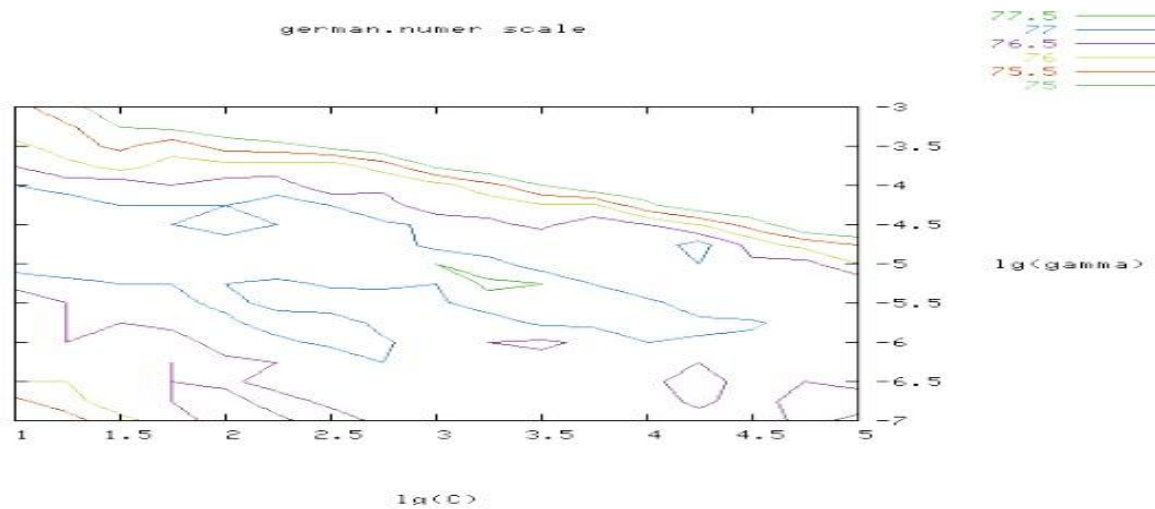


Figure 3: Fine grid-search on $C = 2^1, 2^{1.25}, \dots, 2^5$ and $\gamma = 2^{-7}, 2^{-6.75}, \dots, 2^{-3}$.

多种评价指标

- * 支持macro-average、micro-average、F-measure、Recall、Precision、Accuracy等多种评价指标

召回率 (Recall): $r = \frac{\text{correct assignments by the system}}{\text{total number of correct assignments}}$

正确率 (Precision): $p = \frac{\text{correct assignments by the system}}{\text{total number of the system's assignments}}$

F₁ 测度由 van Rijsbergen 提出, 是关于召回率和正确率的函数:

$$F_1(r, p) = \frac{2rp}{r + p}$$

宏平均是每一个类的性能指标的算术平均值,

微平均是每一个实例 (文档) 的性能指标的算术平均

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}}$$

其他特点

- * 跨平台、多语言
 - * 该系统在linux、windows下都可以方便的使用
 - * 支持32、64机器
 - * 程序有Python和Java两种版本
- * 支持多个SVM模型同时进行模型预测
- * 采用python的csc_matrix支持存储大稀疏矩阵。
- * 引入第三方分词工具自动进行分词
- * 将文本直接转化为libsvm、liblinear所支持的格式。



谢谢