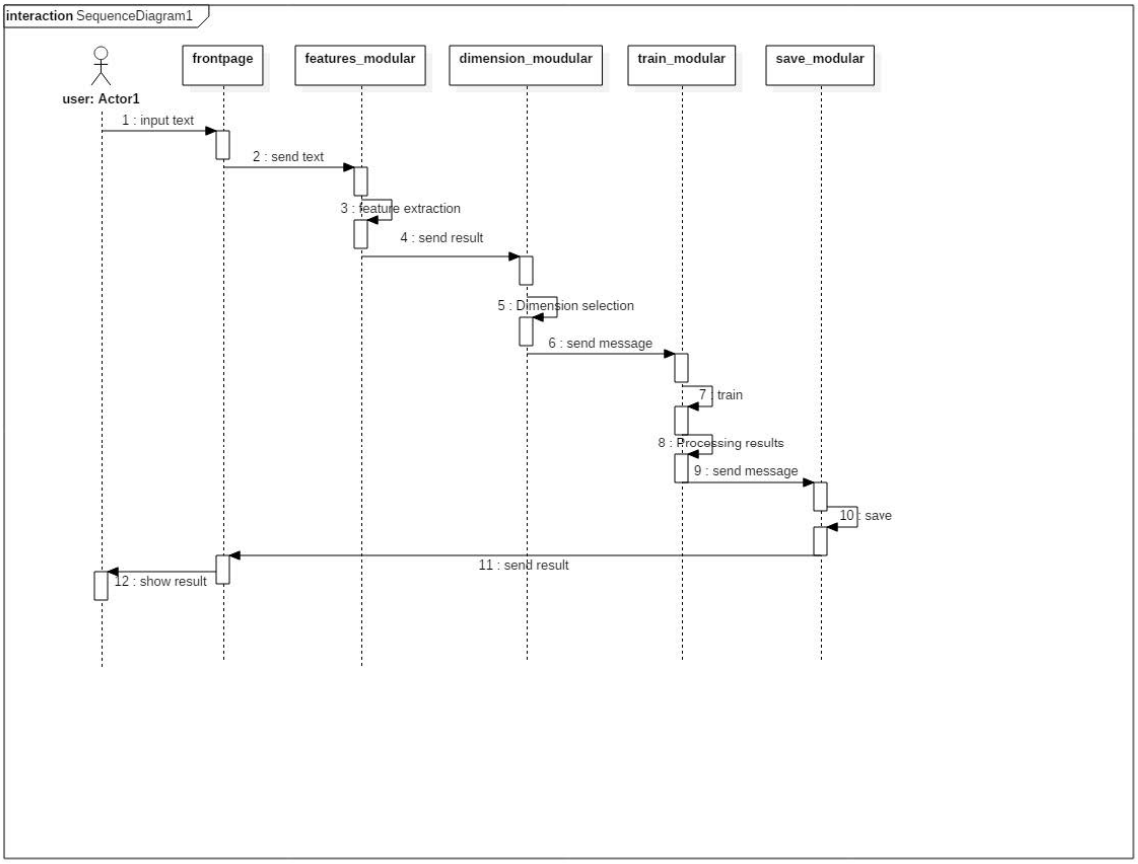
基于人民日报和微博等与疫情有关话题数据所作的两极情感分析---机器学习方法

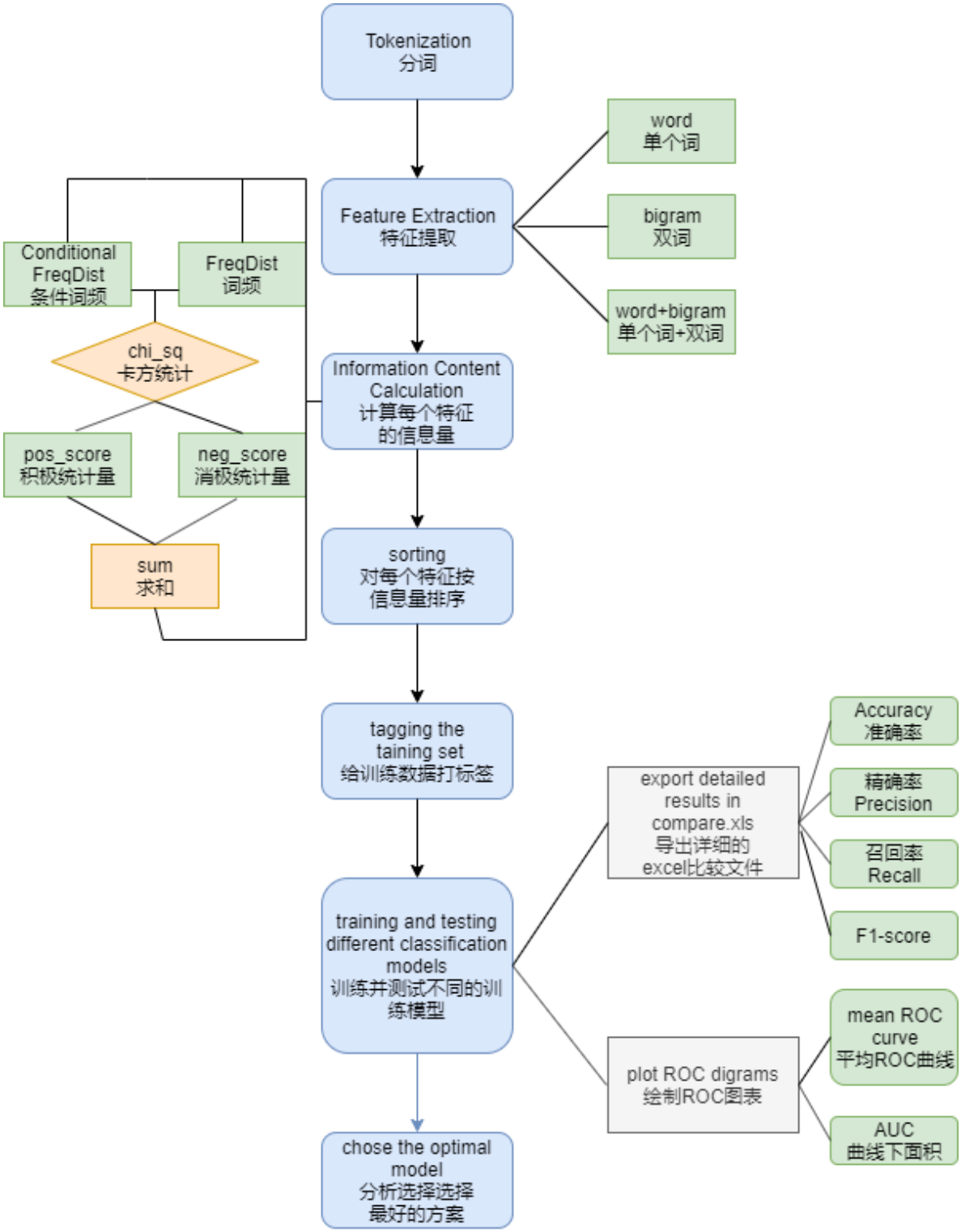
在成功获取来自微博和人民日宝的数据后，重点就是如何利用这些数据实现情感的极性分析。目前情感倾向分析的主流方法有两类，其一是基于情感词典，这需要应用到标注好的情感词典，英文语料的各类词典数量庞大，而现有的中文语料的词典却不是很多，主要有知网整理的情感词典Hownet和台湾大学整理发布的NTUSD两个情感词典，还有哈工大信息检索研究室开源的《同义词词林》可以用于情感词典的扩充。在实践过程中我们选取了大连理工大学的情感词汇本体库进行基于词典的情感分析.第二种方法则是基于机器学习，其需要大量人工标注的语料作为训练集，并通过提取文本特征，构建分类器模型来实现情感分类。

下图为使用机器学习方法的过程流：



相较于固定的情感词典，机器学习方法的优点就在于其精确度更高。首先，词典匹配会由于语义表达的丰富性而产生很大误差，而机器学习则无需深入到语法层面。其次，词典匹配的方法适用范围更加广泛，因为词典本身所包含的语料就十分丰富；相较而言，机器学习方法对训练集语料的质量依赖性较高，如果训练语料针对性不强，则会严重影响到模型的性能，相反优秀的训练语料会使训练所得模型在特定的预测方向拥有极高的精确度。

Python有很多优秀的适用于情感分类的模块，比如Python自然语言处理包，即Natural Language Toolkit，简称NLTK，本次实验中，还使用了jieba作为中文文本的分词工具。



中文分词 Chinese Word Segmentation

在NLP中，分词可以说是最基础的任务。分词需要正确地将句子，段落、文章之类的长文本分解为字词单位，方便后续的处理分析。分词的目的在于把原生数据转化成具有特定结构 的数据，这是模型提取特征的基础，使机器学习方法把原始问题转化成数学问题成为可能。英文天然可以通过空格分词，但有时候也要考虑将多个单词判断成一个词，比如“New York”，而且英文单词具有多种形态，为了应对这些复杂的词形转换，英文NLP相比中文存在一些独特的处理步骤，我们称为词形还原（Lemmatization）和词干提取（Stemming）；而对于中文来说，因为中文语句中没有空格，实际划分会比较困难。另外，中文分词还要考虑词语的粒度问题。“南京九乡河文理学院”就有多种分词方法：

- ”\南京九乡河文理学院\"
- "\南京\九乡河文理\学院\"
- "\南京\九乡河\文\理\学院\"

不同的分词粒度对应于不同的场景需要。

中文分词问题

分词中涉及到三个基本问题：分词规范、歧义切分、未登录词识别。

- 分词规范

  既上文提到的分词粒度的问题

- 歧义切分

  带有歧义的字段在汉语中普遍存在，而歧义字段是汉语切分的一个重要难点。梁南元最早对歧义字段进行了两种基本的定义：

  - 交集型切分歧义：汉字串AJB称作交集型切分歧义，如果满足AJ、JB同时为词（A、J、B分别为汉字串）。此时汉字串J称作交集串。如，大学生（大学/学生）、研究生物（研究生/生物）、结合成（结合/合成）.
  - 组合型切分歧义：汉字串AB称作多义组合型切分歧义，如果满足A、B、AB同时为词。如，起身（他|站|起|身|来/明天|起身|去北京）、学生会（我在|学生会|帮忙/我的|学生|会来|帮忙）

  可以看出，歧义字段给我们的分词问题带来了极大的困扰，所以想要正确的做出切分判断，一定要结合上下文语境，甚至韵律、语气、重音、停顿等。

- 未登录词识别

  未登录词，一种是指已有的词表中没有收录的词，另一种是指训练语料中未曾出现过的词。而后一种含义也可以被称作集外词，OOV（out of vocabulary），即训练集以外的词。通常情况下未登录词和OOV是一回事，这里不加以区分。
  未登录词大体可以分为如下几个类型：

  - 新出现的普通词汇，如网络用语当中层出不穷的新词，这在我们的分词系统这种也是一大挑战，一般对于大规模数据的分词系统，会专门集成一个新词发现模块，用于对新词进行挖掘发现，经过验证后加入到词典当中。
  - 专有名词，在分词系统中我们有一个专门的模块，命名体识别（NER name entity recognize），用于对人名、地名以及组织机构名等单独进行识别。
  - 专业名词和研究领域名称，这个在通用分词领域出现的情况比较少，如果出现特殊的新领域，专业，就会随之产生一批新的词汇。
  - 其他专用名词，包含其他新产生的产品名、电影、书籍等等。

  经过统计汉语分词出现问题更多是由于未登录词造成的，那么分词模型对于未登录词的处理将是衡量一个系统好坏的重要指标。

常用中文分词方法

从汉语自动分词的概念被提出来以后，人们提出了许多分词方法，早期的分词方法主要基于词典，属于字符串匹配算法，比较著名的有在上世纪80年代提出的正向最大匹配法（FMM),对文本呢从左至右切出最长的词；逆向最大匹配法（BMM),对文本从右到左切出最长的词；N-最短路径方法；双向匹配分词法，由左到右，由右到左作两次扫描。后来，随着统计方法的迅速发展，人们又提出来基于统计模型（HMM 和 n元语法）的分词方法，以及规则方法和统计方法相结合的分词技术。

- 基于词典

  基于词典的方法是经典的传统分词方法，这种方式很直观，即从大规模的训练语料中提取分词词库，并同时将词语的词频统计出来，再可以通过逆向最大匹配、N-最短路径等分词方法对句子进行切分。基于词典的分词方法非常直观，可以很容易的通过增减词典来调整最终的分词效果，比如当发现某个新出现的名词无法被正确切分的时候，可以直接在词典当中进行添加，以达到正确切分的目的；同样的，过于依赖于词典也导致这种方法对于未登录词的处理不是很好，并且当词典当中

的词出现公共子串的时候，就会出现歧义切分的问题，这就需要语料库足够的丰富，从而能够对每个词的频率有一个很好的设置。

- 基于模型的机器学习统计方法

  这类方法主要围绕特定的模型，背后有相应的数学理论作支撑。比较著名的有隐马尔科夫模型（HMM)、条件随机场模型（CRF）、最大熵模型（ME)、N元文法模型（$N-gram$)、支持向量机（SVM)等。其中还有基于深度学习的分词器，这里不一一介绍了。这类方法优缺点鲜明，缺点是训练集需要大量人工标注语料、还要整理统计特征；优点则是不仅考虑词频，还考虑上下文，可有效消除歧义、识别未登录词。

- 其他

  分词处理的方法还有：

  - 词向量转换/特征降维
  - $TF-IDF$

    TF表示某个词语在一个语料中出现的频次；DF 表示在全部语料中，共有多少个语料出现了这个词，IDF 是DF的倒数(取log)；$TF-IDF$ 越大，表示这个词越重要。这常用于提取关键词。
  - $TextRank$

    根据词语之间的邻近关系构建网络，通过$PageRank$迭代计算出词语的排名；也常用于关键词提取、自动摘要提取等。

开源免费的分词软件

按github上排名来看，我们选择了星标最多的jieba分词作为分词工具，其找出基于词频的最大切分组合，有中文分词、关键词提取、词性标注功能，支持自定义词典；采用HMM模型、 Viterbi算法；支持Java，C++，Python语言。

特征提取 Feature Extraction

> Finding collocations requires first calculating the frequencies of words and their appearance in the context of other words. Often the collection of words will then requiring filtering to only retain useful content terms. Each ngram of words may then be scored according to some association measure, in order to determine the relative likelihood of each ngram being a collocation.
>
> ——NLTK 3-5 documentation

在特征提取的过程中，我们分别尝试了直接把每一条文本的所有词作为特征、把文本变成双词搭配的形式，作筛选后作为特征和把所有词和双词搭配一起作为特征。把文本变成双词搭配的过程中需要调用$ntlk.collection$模块下的$BigramCollocationFinder$类，其中类方法`from_words`可以把文本变成双词搭配的形式。在双词筛选时，可以选择卡方统计、互信息等多种评分方法，`nbest`方法选择分数靠前的1000个双词作为特征。其实，在$ntlk.collection$模块下还有支持三元词的$TrigramCollocationFinder$类和支持四元词的$QuadgramCollocationFinder$类，会适用于更复杂的模型。

计算信息量 Calculate information content

接下来为了对特征做筛选，我们需要计算每一个特征（单词或双词）的信息量，保留信息量大的而剔除信息量过小的特征。这样做一来可以降维，减少计算复杂度，二来可以避免过拟合现象，充分提取训练集中的重要特征。具体思路为利用`ntlk.probability`模块下的`FreqDist`类和`ConditionalFreqDist`类来统计每个特征的词频和条件词频，再利用`nltk.metrics`模块下的

`BigramAssocMeasures`类中的`chi_sq`方法以卡方统计的方式计算得分，每个特征的总得分为积极消极得分之和。

卡方检验

特征筛选的目的就是去除无关特征，那么何为无关特征呢？对于分类问题，在过滤式方法中一般假设与标签独立的特征为无关特征，而卡方检验恰好可以进行**独立性检验**，所以其适用于特征选择。如果检验结果是某个特征与标签独立，则可以去除该特征。说到卡方检验自然会用到卡方分布，其定义如下：

> 设随机变量x_1,x_2,...x_n, $i.i.d$~N(0,1),即独立同分布于标准正态分布，那么这 n >个随机变量的平方和：
>
> $$X = \sum_{i=1}^{n} x_i^2 \qquad (1)$$
>
> 构成一个新的随机变量，其服从自由度为 n 的卡方分布(χ^2分布)，记为X~χ_n^2。

	positive	negative	word count
$word_1$	$cond\_word\_fd['pos'][word_1]$	$cond\_word\_fd['neg'][word_1]$	$word_1\_freq$
$word_2$	$cond\_word\_fd['pos'][word_2]$	$cond\_word\_fd['neg'][word2_]$	$word_2\_freq$
.....	.....	.....	.....
$word_n$	$cond\_word\_fd['pos'][word_n]$	$cond\_word\_fd['neg'][word_n]$	$word_n\_freq$
$p-n\_count$	pos_word_count	neg_word_count	$total\_word\_count$

对每一个词，我们做假设检验：

> 零假设 (H_0)：$word_i$与情感极性独立
>
> 非零假设（H_1)：$word_i$与情感极性不独立

因为零假设是两个变量独立，因此依独立性的定义：$P(A,B)=P(A)P(B)$,上表中每个格子的期望频数$cond_word_fd_expected['pos'/'neg'][word_i]$为$N \times P(A,B) = N \times P(A) \times P(B)$,其中$N$为总量$total_word_count$。于是我们又得到了一张新的总体期望频数表：

	positive	negative	word count
$word_1$	$cond\_word\_fd\_expected['pos'][word_1]$	$cond\_word\_fd\_expected['neg'][word_1]$	$word_1\_freq$
$word_2$	$cond\_word\_fd\_expected['pos'][word_2]$	$cond\_word\_fd\_expected['neg'][word_2]$	$word_2\_freq$
.....	.....	.....	.....
$word_n$	$cond\_word\_fd\_expected['pos'][word_n]$	$cond\_word\_fd\_expected['neg'][word_n]$	$word_n\_freq$
$p-n\_count$	pos_word_count	neg_word_count	$total\_word\_count$

在得到这两张表之后，就可以针对每个$word_i$计算检验统计量χ^2,χ^2越大，表示观测值和理论值相差越大，该词的信息量就越大。

我们调用了$nltk.BigramAssocMeasures$中的chi_sq评分方法，而chi_sq方法又是利用phi_sq方法实现的。这是基于2×2列联表的卡方统计值（χ^2），与Phi相关系数呈下述关系:

$$\phi^2 = \frac{\chi^2}{n} \qquad (2)$$

在一个2×2列联表中

```
        w1    ~w1
     ------ ------
 w2 | n_ii | n_oi | = n_xi
     ------ ------
~w2 | n_io | n_oo |
     ------ ------
     = n_ix        TOTAL = n_xx
```

可以得出 $w1$ 和 $w2$ 的 Phi相关系数如下：

$$\phi = \frac{n\_ii \cdot n\_oo - n\_io \cdot n\_oi}{\sqrt{n\_ii \cdot n\_oi \cdot n\_io \cdot n\_oo}} \qquad (3)$$

```python
@classmethod
def chi_sq(cls, n_ii, n_ix_xi_tuple, n_xx):
    """Scores bigrams using chi-square, i.e. phi-sq multiplied by the number
    of bigrams, as in Manning and Schutze 5.3.3.
    """
    (n_ix, n_xi) = n_ix_xi_tuple
    return n_xx * cls.phi_sq(n_ii, (n_ix, n_xi), n_xx)
```

```python
@classmethod
def phi_sq(cls, *marginals):
    """Scores bigrams using phi-square, the square of the Pearson correlation
    coefficient.
    """
    n_ii, n_io, n_oi, n_oo = cls._contingency(*marginals)

    return (n_ii * n_oo - n_io * n_oi) ** 2 / (
            (n_ii + n_io) * (n_ii + n_oi) * (n_io + n_oo) * (n_oi + n_oo)
    )
```

统计数所有积极和消极的得分后，将每个词的积极得分和消极得分相加，即得每个词最终的总信息量得分。

对每个特征按信息量排序 sorting

这一步需要对每个特征按照信息量的大小进行排序，取排名靠前的特征作为最终特征，完成筛选。

给训练数据打标签 tagging the training set

由于我们人工标注的数据只是被放进了“pos”、“neg”两个不同的文件夹，因此还需要处理属于数据，为训练集中的每条评论打上标签。

训练并测试不同的分类模型 training and testing different classification models

训练结果通过excel表格导出，命名为“compare.xls", 文件附在报告文件夹里。由于时间紧迫，我们人工只打了积极消极各212条标签。综合结果来看：

不同模型 different models

$sklearn.naive_bayes$模块下的$MultinomialNB$模型效果最优。该模型准确率最高，精确率也是最高的，但相应的召回率就比较低了；差准率和查全率的调和均值$fi-score$也比较高。

不同的n-gram选取方式 different n-gram selection modes

通过对比取所有单个词、只取双词、双词和单词结合的取词方法，发现将双词和单词结合方法最优，但是尽相对于只取单个词的方法在效果上有略微提升，而只取双词的方法则不够理想。值得注意的是，不同的模型对于不同的取词方法的反应也不同，比如$BernoulliNB$模型相对于只取单个词，在单双词都取后其准确率反而有所下降。受限于打好标签的样本数量匮乏，结果可能不够完善。

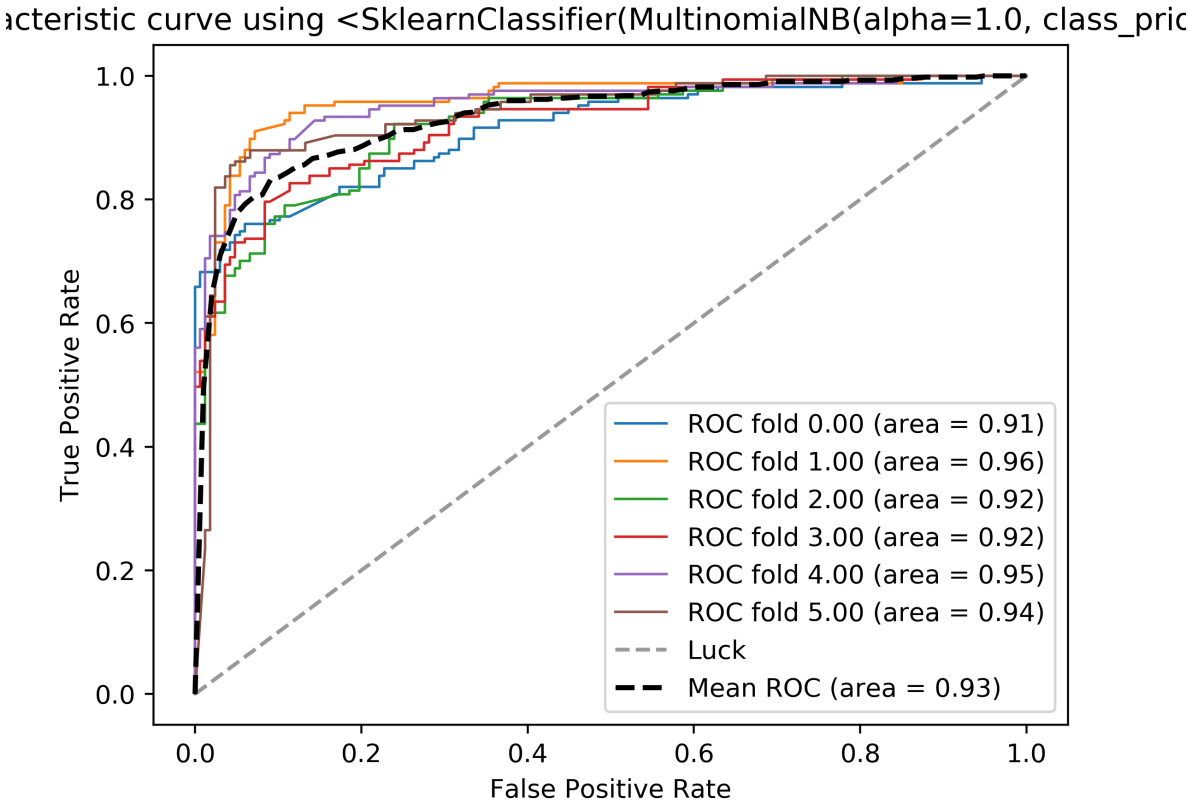
是否做特征提取apply feature extraction or not

截取信息量丰富的单词后，加以选择合适的维度，准确率相比直接取单词有所上升。例如$BernoulliNB$模型的准确率从0.605提高到了0.9；$MultinomiaNB$模型在本身已经达到0.925的情况下依然提高了1.5%；$LogisticRegression$模型和$NuSVC$模型的准确率也有很大提升。同样的，"直接将所有词和双词搭配"与"信息量丰富的所有词和双词搭配"两个方法做对比也会得到类似的结果。

维度截取 dimension interception

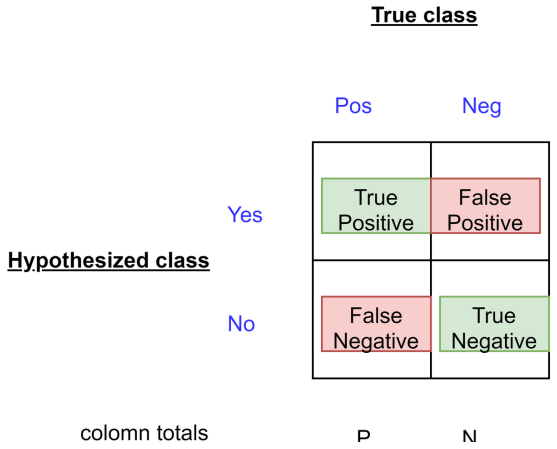
在选取信息量丰富的词时，我们直接截取前n维信息量最丰富的词。n的大小的选择是一个问题。在测试中我们测试了100维到3000维,发现模型的准确率随着截取维数而逐渐升高，但是又可能会随着维数的继续增加而有所下降。考虑到所给训练集的大小问题，大部分模型在截取过高维时的准确率都趋向稳定（数据太短没有更多维度了）。不同模型的最佳截取维度也不同，一方面要考虑准确率的高低，截取维度不能太低；另一方面要考虑过拟合和计算量问题，截取维度不能太高。

ROC 曲线 ROC-curves



上图是我们训练所得的最优模型$MultinomialNB$的ROC曲线图。6条不同的曲线是用$sklearn.model_selection$模块下的$StratifiedKFold$类的（$k=6$)交叉验证所得，黑色虚线是平均ROC曲线。area是AOC值。

ROC（Receiver Operating Characteristic）曲线，以及AUC（Area Under Curve），常用来评价一个二值分类器的优劣，ROC的横轴为$falsepositiverate$，**FPR**，“假正例率”，也就是误判为正确的比例；纵轴是$truepositiverate$，**TPR**，“真正例率”，也就是正确的判断为正确的比例。



$$FP\ rate = \frac{FP}{N} \qquad TP\ rate = \frac{TP}{P}$$

$$precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{P} \qquad accuracy = \frac{TP + TN}{P + N}$$

$$F1 - score = \frac{2}{1/precision + 1/recall}$$

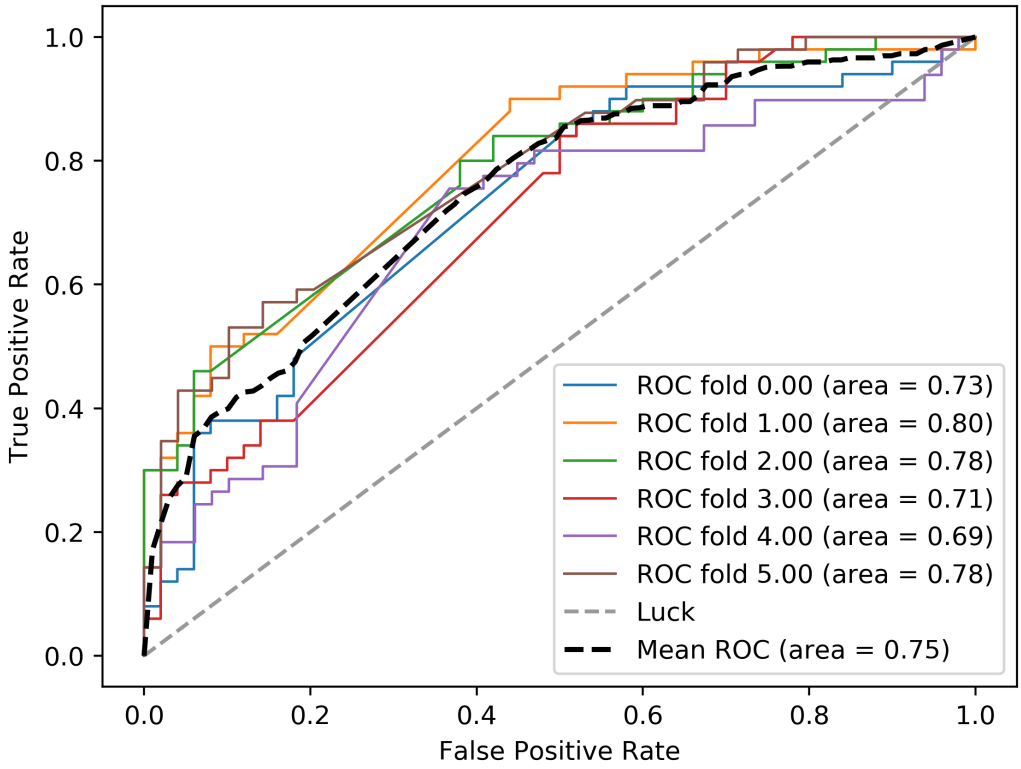
我们重点来看ROC曲线图中的四个点和一条线。第一个点，(0,1)，即FPR=0, TPR=1，这意味着FN（false negative）和FP（false positive）=0/这是一个理想状况下的完美的分类器，因为它将所有的样本都正确分类了，没有漏判也没有误判。第二个点，(1,0)，即FPR=1，TPR=0，与前者相反，这是一个最糟糕的分类器，因为它成功避开了所有的正确答案。第三个点，(0,0)，即FPR=TPR=0，即FP（false positive）= TP（true positive）= 0，可以发现该分类器预测所有的样本都为负样本（negative）。类似的，第四个点(1,1)，分类器实际上预测所有的样本都为正样本。综上，我们可以断言，ROC曲线越接近左上角，该分类器的性能越好。

AUC分数是曲线下的面积（Area under curve），越大意味着分类器效果越好。显然这个面积的数值不会大于1。又由于ROC曲线一般都处于$y = x$这条直线的上方，所以AUC的取值范围在0.5和1之间。使用AUC值作为评价标准是因为很多时候ROC曲线并不能清晰的说明哪个分类器的效果更好，而AUC作为一个数值，评价起来更直观：对应AUC更大的分类器效果更好。

ROC曲线作为关键评价标准之一的一个重要原因在于ROC曲线有个很好的特性：当测试集中的正负样本的分布变化的时候，ROC曲线能够保持不变。在实际的数据集中经常会出现类不平衡（class imbalance）现象，即负样本比正样本多很多（或者相反），而且测试数据中的正负样本的分布也可能随着时间变化。

作为反面参照，在我们的多角度分析下，也有一些性能不佳的分类器：

multi_class='warn', n_jobs=None, penalty='l2',
random_state=None, solver='warn', tol=0.0001, verbose=0,
warm_start=False))>



这些表现较差的分类器可以一眼通过ROC曲线的走向或者AUC的值分辨出来。

分析选择最好的方案 select the optimal model

最后一步就是选择最优模型并保存。这样就得到了一个经过训练集训练的模型，可以用于之后的测试。

参考文献 references

spaCy的分词步骤，https://spacy.io/usage/spacy-101#language-data

nltk 3.5 documentation http://www.nltk.org/index.html

jieba分词，https://github.com/fxsjy/jieba

HanLP, https://github.com/hankcs/HanLP

华天清，《中文分词方法和软件工具汇总笔记》，https://zhuanlan.zhihu.com/p/86322679

崔庆才，《中文分词原理及工具》，https://cuiqingcai.com/5844.html

$massquantity$, 特征选择： 卡方检验、F 检验和互信息,https://www.cnblogs.com/massquantity/p/10486904.html

基于酒店评论的情感分析，https://github.com/yirui-wang-0212/NLP-SentimentAnalysisForChineseText