“智能政务”中的文本挖掘：原理、实现与应用

Woodbird Zhuo

摘要：本文主要讨论文本挖掘中的文本分类、热门问题挖掘以及文本相关性、完整性和可读性评价的原理、算法与实现。文章采用了二元语法、词袋模型、卡方检验的方法，结合机器学习实现了文本自动分类；为了挖掘热门问题，文章采用 DBSCAN 的方法对留言进行聚类，并通过留言的点赞数、反对数和时间跨度挖掘出热门问题；为了给答复打分，文章结合词向量和关键句提取算法，度量了答复的相关性。再根据二元语法与字典匹配，度量了答复的局部整体性与可读性。最后，结合两者即可评价留言答复的质量。为了贴近工程实际，在每个问题最后，文章列举了某些底层的优化实现。

关键词：文本分类；机器学习；二元语法；词袋模型；文本聚类；词向量；

Text Mining in Intelligent Government Affairs Management: Principle, Implementation and Application

Abstract The present paper is mainly about the principle and implementation on texts classification, mining of hot spots as well as scoring of texts according their correlation, Integrality and readability. The paper adopts 2-gram, bag-of-words model and chi-square test to prepare the corpus. Then using machine learning algorithm, one could implement a classifier of texts. To deal with the hot-spots mining, DBSCAN clustering is deployed to cluster similar texts as a class. Then coming up a scoring model based on the number of agrees, disagree and time span to decide the hot-spots. Eventually, to calculate the relevance among texts, this article uses key-sentence extracting and word2vec method to calculate the similarity of texts. To scoring the integrality and readability, 2-gram and dictionary matching is adopted. Thus a scoring model based on both is proposed. In order to go in line with practice, this article will present some low-level implementation and optimization method.

key words: texts classification;machine learning; 2-gram; bag-of-words; texts clustering; word2vec

1 内容简介

从文本中挖掘有效信息，是自然语言处理（NLP）领域的重要问题。一般地，基于规则的挖掘算法已经在 19 世纪 50 年代遭遇挫折。因此，在文本挖掘问题中，业界通常采用基于统计方法的机器学习，和隶属机器学习、使用神经网络模型的深度学习。前者一般需要一个手工特征模板，对数据进行预处理。后者则不然，其类似于“黑盒子”，通过神经网络节点的训练，即可自动地提取出信息。使用文本挖掘的方法，亦可以减轻网络问政工作人员的负担。

在处理群众留言时，首先要对留言进行归类。这一点可以在已有的、分好类的留言详情语料库中，训练出一个能够自动分类的机器学习模型。为了实现这一点，本文在前人工作的基础上 [?]，使用效果极佳的二元语法以及卡方检验，从而将非结构的文本转换为结构化的特征向量。之后，文章从机器学习常用的模型之中，采用 K 折交叉验证、网格寻优的方法筛选出适合用于该问题的、常见的机器学习模型及其参数，如图??所示：

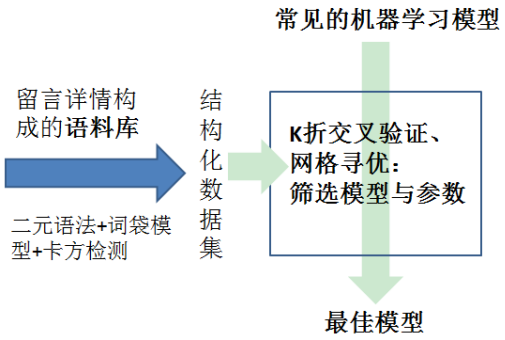

图 1. 问题一解题思路

同时，在附录中额外给出了神经网络的训练过程与模型效果。之后，以 F1 值评价训练好的模型，并得出适用于文本分类的模型有：贝叶斯分类器、SVC 和逻辑回归。同时，进一步反映了神经网络无法提升文本分类效果这一论断。

为了挖掘出热门问题，显然需要先对留言进行聚类，因此属于文本聚类的任务之一。考虑到二元语法得到的特征向量太过冗余，且聚类这种没有标签的无监督算法，无法采用卡方分布过滤特征。因此，本文采用条件随机场分词器，对留言详情构成的语料库进行分词。过滤掉停用词并在粗分的基础上进行合并后，再使用主成分分析法（PCA）对数据进行降维。之后，将预处理后的数据，使用自适应聚类算法——DBSCAN 进行聚类。从而将隶属与同一个问题的、相似的留言聚成一簇。之后，再根据留言的点赞数、反对数和时间跨度，建立一个问题的热度模型。最后，按照热度进行降序排序，即可找出 5 大热门问题。

考虑到留言详情的字数较多，且有可能同一热门问题包含许多条留言。因此，本文还结合 TextRank 算法和 BM25 算法，提取出文本中的关键句。再根据关键句，人工归纳出热门问题的问题描述。总体的做法见图??所示。

图 2. 问题二解题思路

为了评价留言答复的相关性，显然需要判断答复与留言详情之间的相似度，并以此为依据度量相关性。因此，本文采用了词向量的方法，结合两者的关键句，找出了答复与留言的相似度，从而评价相关性。为了衡量留言的完整性与可读性，文章采用了二元语法匹配的方法。扫描留言答复的同时，在字典中匹配当前两个字符，从而一定程度上度量了答复的局部完整性、可读性。最后，综合上述因子，即可建立一个评分模型，对留言答复进行评分。

文章在每一个问题最后，都或多或少地提及相应的底层实现和优化方法，使用它们将降低算法的复杂度和运算耗时。同时，在许多细节方面亦给出详细的处理方法。

最后，文章总结所做的工作，提出了笔者对 NLP 领域的一些浅薄的见解。同时分析了文章的不足之处，和 NLP 领域有待解决的问题。

2 文本分类问题

根据已经归类好的群众留言数据，对未知类别的留言进行分类显然属于一个文本分类问题。对于分类问题，使用机器学习的方法即可高效地解决。然而，要使用机器学习模型实现自动分类，首先需要将非结构化的文本数据转换成结构化的特征向量。考虑到留言语料库中留言详情所包含的信息量，远大于留言主题。且很明显，留言时间与文本的类别毫不相关[1]。因此，**本文中的语料库均指所有留言的留言详情**。本节将展示使用二元语法词袋模型，将非结构化的文本表示为向量。考虑到特征个数达到十万以上数量级，若直接投入机器学习模型的训练中，显然会造成“维度灾难”。为此，文章采用卡方检验的方法，从而过滤掉对分类结果影响不大的特征，从而进行特征降维。最后，文章将从多种常见的机器学习模型中，根据模型们在数据集中的表现，挑选出合适的模型及模型参数。

2.1 特征工程

将语料库进行处理，从而转换为可供机器学习使用的模式即为特征工程，或**数据预处理**。对于本例中的文本分类任务，这里将结合卡方检测，使用二元语法词袋模型，对语料库进行预处理。对于样本的类别，可

[1]笔者已将时间处理为精确到分钟有序序列，并通过单因素方差分析法，使用随机抽样（共 1000 个样本）证明了这一点

以直接将其转换为 **0 开始的有序整数**。

2.1.1 二元语法

如上所述，为了将非结构化的文本转换为结构化的特征向量，本文将采用二元语法词袋模型进行建模。所谓二元语法，即将连续的两个汉字（过滤掉标点符号、制表符、换行符等）视为一个特征。例如句子“第八届泰迪杯比赛。”，其二元语法为（“第八”，“八届”，“届泰”，...，“杯比”，“比赛”）。为了过滤掉标点符号，可以考虑使用**正向最长匹配**的方法[2]，将标点符号进行过滤。

值得一提的是，将文本转换为特征向量可以考虑进行**分词**。然而根据郭志芃等老师的开源工作 [**?**]，这种将文本中相邻两个字符作为特征，反而能够取得更好的成绩。

2.1.2 词袋模型

在许多外文文献中，也称词袋模型为 **BOW**。词袋模型将语料库（经过二元语法提取后）的所有特征，构成一个的向量，并作为每一句留言（文档）的**特征向量**。而文档中的特征向量的某个元素，其取值等于相应特征在文档中出现的频数。至此，就将语料库转换为一个稀疏的矩阵。

2.1.3 卡方检验

由于词袋模型得到的往往是一个稀疏矩阵，若直接供给机器学习模型训练，势必会出现“维度灾难”的问题。以示例数据为例，经过二元语法与词袋模型的处理后，语料库转换为 9210×396287 的畸形矩阵，即样本个体的特征个数近 40 万。但是，该矩阵中有绝大部分为 0 元素，换句话说，矩阵是稀疏的。

另一方面，由于许多常用的单词对分类决策的影响不大，比如停用词和表述词等。再者，许多单词在所有类别的样本中均频繁出现。因此，为了消除这些影响因素，这里考虑采用卡方检验的方法，过滤掉这些用处不大的特征。

类似于单因素方差分析，卡方检验通常由于判断两个随机事件是否相互独立。记语料库中的一个特征为 $f_n, n \in \{0, 1, \cdots, 396287\}$，事件“文档中存在 f_n”为 A_n，事件“文档属于类别 $c, c \in \{0, 1, \cdots, 6\}$”为 B_c，则卡方检验在于验证 $P(A_nB_c) = P(A_n)P(B_c)$ 是否成立。

记卡方检验的检验统计量为 γ^2，其计算公式如下:

$$\gamma^2(n, c) = \sum_{c} \sum_{n \in \{f_n, \bar{f}_n\}} \frac{(N_{nc} - E_{nc}^2)}{E_{nc}} \tag{1}$$

其中 N_{nc} 为特征 f_n 在属于类别 c 的文档中出现的频数。E_{nc} 为事件 A_n 和 B_n 同时出现的期望，可由如下式算出：

$$E_{nc} = N \times \frac{N_{nc} + N_{n\bar{c}}}{N} \times \frac{N_{nc} + N_{\bar{n}c}}{N} \tag{2}$$

其中 $\bar{n}, \bar{c}$ 表示逻辑非，N 为所有特征的频数, 即 $N = N_{nc} + N_{\bar{n}c} + N_{n\bar{c}} + N_{\bar{n}\bar{c}}$。由于 γ^2 服从卡方分布，根据所得值与卡方分布的表达式即可反推出概率 p

卡方检验的**原假设**为 $P(A_nB_c) = P(A_n)P(B_c)$ 成立，即待检验特征 f_n 对分类决策的帮助不大。取置信水平为 $\alpha = 0.001$, 也即检验犯一类错误的概率为 0.1%。于是，对语料库中的每一个特征，考虑将它们都进行卡方检验。若概率 $p < \alpha$, 则拒绝原假设，即认为该特征属于重要因子。反之，则接受原假设，此时即可将该特征 **删除**。

以附件 2 数据为例，其词袋模型的稀疏矩阵共有 396287 个特征。经过卡方检验的过滤后，降为 30291 个特征，**压缩**到原来的不到 10%。至此，数据预处理步骤结束 [3]。

2.2 文本分类模型的筛选

得到数据矩阵后，就可以通过机器学习的方法，根据数据集训练出一个文本分类模型了。由于汉语言处理文献较为缺乏，本文将从常见的机器学习模型中，筛选出最适合进行文本分类的模型。在附录 A 中，文章使用了 BP 神经网络进行文本分类，并发现其较之普通机器学习而言，效果反而更差。因此本文将着重采用机器学习的方法，解决文本分类的问题。

由于本文是根据 $F1$ 值筛选出模型和参数的，为了表述方便，在后文中均用**拟合优度均代指 $F1$ 值**。

[2] 具体细节详见**??**小节

[3] 预处理后的数据可见附件：data_q2_X_final_data.pkl

2.2.1 交叉验证与网格寻优筛选模型

笔者认为，机器学习是一门理论的科学，亦是一门实践的艺术。因此，在 NLP 特别是汉语言处理这门比较新的领域，任何模型都不能随意地认定其优劣。考虑到前人在这方面的研究较少，因此，本人将从逻辑回归[4]、支持向量分类器（以下称 SVC）、决策树、k 近邻算法（以下简称 kNN）、朴素贝叶斯分类器、随机森林和 AdaBoost 中，筛选最合适的模型以及模型参数。

在筛选模型之前，需要先筛选最佳的模型参数。上述模型中，带有参数的模型分别为 SVC、决策树、kNN、随机森林和 AdaBoost。如图**??**所示，K 折交叉验证常用来评价一个模型在指定数据集中的优劣。其将数据集复制成 K 份，记为 $D_i, i \in \{1, 2, \cdots, K\}$。同时将 D_i 按比例 $\alpha\%$ 拆分成训练集、测试集，$\alpha = 100/K$。之后对于某一个模型，通过 K 折训练集训练 K 个分模型，并分别计算它们在相应的测试集中的拟合优度，并构成拟合优度序列 $S_i, i \in \{1, 2, \cdots, K\}$。

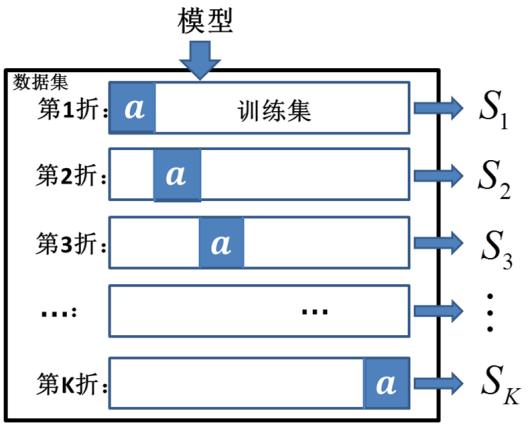

图 3. K 折交叉验证原理

根据序列 S_i 的均值 $\bar{S}$, 即可评价该模型在数据集中的**总体**拟合优度。对于不同模型，可以分别根据 $\bar{S}$ 最大，来筛选最优模型。对于同一模型的不同参数，同样可以将其视为不同模型，并根据上述方法筛选。为了筛选不同模型的最佳参数，可以通过遍历的方法遍历模型参数的所有取值可能，再使用交叉验证的方法筛选参数。然而，遍历法的代价实在太大。为了降低计算机的运算负荷，可以适当地加大步长，并参入人工因素选择。因此，本文采用网格寻优法，从**参数网格**中筛选最优参数。

由于网格寻优法从参数网格中寻找最佳参数，从这个意义上来说，网格寻优法可视为**大步长、动态步长、掺杂人工因素**的遍历法。

2.2.2 模型及其参数的筛选结果与 T 检验

承上所述，为了选择最好的模型，首先需要筛选模型们的参数。而需要选择参数的模型有 kNN、SVC、决策树、随机森林和 AdaBoost。本文使用网格寻优法，结合 5 折交叉验证，计算模型的 F1 值作为 S_i 筛选模型，最终的结果如表**??**所示[5]。

表 1. 各模型的参数网格与筛选结果

模型	参数网格	最佳结果
kNN	k:[1](3,5,7,9,11)	3
SVC	C:[2] (0,0.1,0.25,0.5,0.75,1,1.25,1.5,1.75,2,3,4,5,6,7,8,9)	C= 0.1, 核函数: 线性函数
	核函数:(线性函数、径向基函数、三次多项式函数)	
决策树	最大深度 d:(7,9,11,13,15,17,19,24,29,34,39,44,49,54,59,64,69,74,79,84,89)	$d = 79$, $\alpha_{cpp} = 0.0005$
	α_{cpp}:[3] (0.00025,0.0005,0.001,0.00125,0.015,0.01,0.05,0.1)	
随机森林	基模型个数：(15,25,35,45,50,65,75,85,95,100,150,200,250,300)	75
AdaBoost	基模型个数：(15,25,35,45,50,65,75,85,95,100,150,200,250,300)	15

[1] 这里不妨啰嗦一句，kNN 算法的 k 只能取奇数
[2] 即惩罚参数.
[3] 即最小代价复杂度剪枝处理的阀值

得到最佳参数后，再次使用 5 折交叉验证的方法，计算 $k = 3$ 的 kNN、$C = 0.1$ 核函数为线性函数的

[4] 正则化用于解决过拟合问题，然而考虑到这些模型的拟合优度均较低，因此不使用正则化
[5] 可以看到，参数网格由疏到密，这实际是渗入人工因素的结果，具体见**??**

SVC、$d=79, \alpha_{cpp}=0.0005$ 的决策树、基模型为 $d=5$ 的决策树、个数为 75 的随机森林、基模型为逻辑回归、个数为 15 的 AdaBoost、朴素贝叶斯分类器，以及逻辑回归，分别计算它们在数据集中的拟合优度序列 $S_i, i \in \{1,2,\cdots,5\}$。如表 **??**,**??**所示，各模型在数据集的拟合优度序列如下：

表 2. 各模型（最优参数下）的拟合优度序列

S_i	AdaBoost	决策树	kNN	逻辑回归
S_1	0.82	0.71	0.52	0.83
S_2	0.85	0.73	0.54	0.87
S_3	0.84	0.76	0.53	0.85
S_4	0.87	0.78	0.55	0.87
S_5	0.85	0.73	0.53	0.84
$\bar{S}$	0.85	0.74	0.53	0.85

表 3. 续上表

S_i	贝叶斯分类器	随机森林	SVC
S_1	0.85	0.45	0.82
S_2	0.86	0.45	0.85
S_3	0.83	0.46	0.83
S_4	0.87	0.49	0.86
S_5	0.89	0.46	0.84
$\bar{S}$	0.86	0.46	0.84

从各模型的拟合优度序列的均值 $\bar{S}$ 可以剔除决策树、随机森林和 kNN。剩下的模型差别均不大。但是，人们不能贸然地认为这些模型在效果上是 **等价**的。因此，为了判断这些模型是否等价，还需要采用 **T 检验**的方法。

类似于**??**小节所述的卡方检验，T 检验亦属于统计检验的方法。T 检验用于判断两个序列的均值，在置信水平下是否相等。篇幅所限，这里不再复述其原理。于是，本文考虑将 AdaBoost、逻辑回归、贝叶斯分类器和 SVC 的拟合优度序列，进行两两的 T 检验。设置置信水平为 $\alpha=0.05$，可得检验结果见表 **??**(见附录 A)。

由于所有模型两两 T 检验的概率均有 $p>\alpha$，故接受原假设，即认为各模型的效果两两等价。

2.3 文本分类模型的训练

本节将根据 T 检验法的结果，从中挑选出一个**适合的**模型，并训练它。

2.3.1 模型选择与分析

根据**??**小节的分析结果可知，AdaBoost、逻辑回归、贝叶斯分类器和 SVC 的效果是一样的。很明显，属于集成模型的 AdaBoost 所消耗的资源较多，没有必要选择它。而较之模型的训练时长而言，显然通过拙算法[6]训练的贝叶斯分类器，所需的训练时长最短。而需要迭代算法求解的逻辑回归和 SVC，在这方面略逊一筹。然而由于 SVC 需要求解的优化问题[7]较为复杂。但另一方面，较之逻辑回归，SVC 只需要训练**支持向量**。换句话说，在硬件实现上可以直接**剔除**非支持向量个体，因此在训练模型时，消耗的内存较低。

另外，由于贝叶斯分类器是通过拙算法训练的，需要存储数据的频率信息。因此，使用贝叶斯分类器所消耗的内存 (3MB) 更多。并且，分类决策所需要的时间亦长。再加上数据预处理所需要的内存资源，使得贝叶斯分类器无法用在嵌入式系统等场合。而逻辑回归与 SVC 则相反，它们只需要存储模型参数 (1MB 左右) 即可。

有的读者可能会认为 SVC 更具有稳定性（即每次训练时结果波动不大），这可能是由于支持向量机也叫**最大间隔模型**的原因。但不得不说，由于惩罚参数 $C=0.1$ 并且接近于 0，因此实际上该 SVC**软化**得很彻底的，所以其稳定性高的谬论不攻自破。

综上，在条件允许的情况（如个人电脑）下，可使用贝叶斯分类器。如果要求简单至上，轻装上阵，则可以选择逻辑回归和 SVC。

另外值得注意的是，SVC 的核函数为线性函数。也就是说，此时 SVC 与逻辑回归一样，属于**线性分类器**。并且，我们看到非线性分类器，除了贝叶斯分类器[8]以外，它们的效果无疑都很差。这是为什么呢？笔者认为，这是由于特征过多，导致的数据集线性可

[6]通过存储数据的频率信息

[7]即模型训练过程中，使得代价函数最小的问题

[8]AdaBoost 属于 Boost 集成，线性模型的 Boost 集成还是线性的，这点笔者已经在之前的研究中验证过

分的缘故。

2.3.2 模型训练

在得出模型之后，还需要将数据集拆分成训练集、测试集筛选数据。可能有读者认为这是多此一举，因为在筛选模型的时候已经反复训练了。但并非如此，因为测试集的意义在于测试模型的拟合优度，人们总是期望在陌生的数据中测试。如果测试集的信息在除测试以外的阶段“泄露”了，那么将会失去测试的意义。

所以说，如果直接拿交叉验证时训练的模型投入使用，那么等于直接拿未经测试的模型投入使用。无论是工业界还是学术界，这都是不可取的。因为无法评价模型的泛化能力，并判断其是否过拟合。

因此，本文将数据集按 7:3 拆分成训练集、测试集，在训练集中分别训练逻辑回归、SVC 和决策树模型。并计算模型们在训练集、测试集中的 $F1$ 值。结果如表 **??**所示。

表 4. 模型在测试集、训练集中的拟合优度值

F1	贝叶斯分类器	逻辑回归	SVC
训练集	0.93	1	1
测试集	0.88	0.87	0.86

读者可以读取附件文件 nb_model.pkl，lg_model.pkl 和 svc_model.pkl 使用这些模型。

2.4 算法的底层实现与操作简介

笔者的计算机配置为：Inter(R) Core(TM)i5-5200U CPU2.20GHz，内存 8GB，Win7 系统。在编程过程中，受硬件限制，遇到了很多不可避免的问题。另外，上一小节所介绍的模型、以及数据预处理时潜在许多问题与相应的解决办法，下面将一一介绍这些底层算法实现。

2.4.1 稀疏矩阵的存取

在**??**小节中，使用词袋模型处理数据将会得到一个 9210×396287 的矩阵。矩阵绝大部分为 0 元素，若直接生成，则需要 27GB 左右的内存空间。因此，本文在将二元语法转换为词袋模型时，只保存非零元素的索引和值。这样可将数据压缩到 4MB 左右，同时节省了操作系统释放、存取内存[9]的时间。

另外，由于特征的取值为频数，其值为整数且大多很小，因此可以将其转换为无符号短整型（即一个字节），从而节省存储开支。

2.4.2 多分类与类别不均衡

对于逻辑回归和 SVC 模型来说，由于其只能输出正负两个结果，故不可以直接用于多分类任务。所以笔者用它们进行文本分类时，将不同类别的样本“分而治之”，从而将多分类任务转换为多个二分类任务。

这里采用 **OvR**的分而治之策略，即在划分某一类时，将不属于该类的样本视为负样本，从而转换为二分类问题。这么做比起 **OvO**[10] 而言，其算法复杂度更低[11] 。然而，这么做会使得正负样本数量不均衡，从而影响模型的拟合优度。举个简单的例子，若正负样本比例为 $1:99$。那么一个只会点头的模型可能达到 99% 的精度，这显然不是人们愿意看到的。

为了解决这种类别不均衡问题，笔者采用了**边界 SMOTE**的过采样方法。该方法旨在通过少数类样本，使用插值法产生新的样本。其中边界样本产生更多的新样本，从而降低简单复制粘贴数据导致模型过拟合的风险。其具体算法见参考文献 [**?**]，这里不再赘述。

2.4.3 网格寻优的具体操作

前面提到，网格寻优法能够加入人工因素，从而避免盲目地遍历参数。笔者在筛选参数的时候，先使用**大步长**遍历大范围参数。并根据结果的左右边界，逐渐减小步长，并缩小参数范围，从而更加精确地筛选模型参数。

2.4.4 贝叶斯分类器的某些设置

贝叶斯分类器根据特征的连续与否，可以分为**多项式分布、伯努利分布和正态分布**三种方式。伯努利分布一般用在二值特征之中，因此不采用。而数据的词袋模型由频数构成，其天然具有离散特征的性质。所以，本文使用的贝叶斯分类器属于**多项式分布**类型。

[9]由于 8GB 远远不够用，因此计算机需要将数据缓存到硬盘

[10]即 one vs one 的缩写，是另一种分而治之的策略。同样地，OvR 为 one vs rest

[11]不难证明，OvR 的复杂度为 $O(n)$，而 OvO 为 $O(n^2)$

当然，也可以将这种“自然数”类型的离散特征视为连续型，从而采用正态分布[12]。

由于这里采用了多项式分布类型的贝叶斯分类器，因此为了提高模型的泛化程度，需要给模型进行平滑处理。本文采用的是一种 +1 平滑策略[13]，具体实现请参阅文献 [?]，这里不再赘述。

2.4.5 参数寻优算法

除了 kNN、贝叶斯分类器以外，其余的所有模型都需要寻找某个参数，使得某个 **代价函数**最小，从而得到模型的参数。换句话说，模型的训练（参数求解过程）是一个优化问题。在本文中，笔者使用优化算法 **LBFGS**来求解优化问题。LBPGS 类似于拟牛顿法的随机优化算法，它使用 mini-batch 来降低计算量。较之拟牛顿法，该方法节省内存，且采用 mini-batch 的它能够降低海赛矩阵的计算时间[14]。本文使用的 LBPGS 的步长为 0.01，mini-batch 为 100 个样本。LBFGS 算法具体细节可参阅参考文献 [?]，这里不过多复述。

3 文本聚类与热度算法

要从每一个群众的留言中，收集某一时间段内群众集中反映的问题，显然属于一个文本聚类的问题。如果将群众相似的留言聚成一簇，即可将簇视为某个集中问题。根据该簇包含的留言条数、支持和反对的总数，并考虑其热度随时间的衰减，即可估计该问题的热度。

为了提取出聚类簇中留言的问题描述，以及地点和人群。本文考虑使用关键语句提取算法，从而自动生成关键句，再从关键句中人工提取出地点和人群。籍此就可以降低直接从留言详情中，人工提取问题概述的工作量。

3.1 特征工程

不同于**??**小节，文本聚类属于无监督问题，其不能根据卡方检测来筛选特征。因此如果仍然采用二元语法对语料库建模，将会导致特征个数非常多。因此，为了降低特征个数，本文考虑使用汉语分词器将句子拆分成一个个的单词。同时，采用正向最长匹配算法，过滤掉停用词、常见词。并采用类似的方法，将地点、人名、机构名等在粗分的结果下合并。经过上述处理之后，再使用与**??**小节同样的方法，使用词袋模型将语料库转换为稀疏矩阵。为了进行特征降维，文章还将采用主成分分析的方法，压缩数据的维度。同样，这里的语料库亦均指代**留言详情**。

3.1.1 条件随机场分词器

如上所述，为了将留言拆分成一个个单词，首先需要使用机器学习的方法训练一个分词器。同样地，为了训练分词器，就需要一个**事先拆分好**的语料库作为训练集。

一种获取语料库的方法是，在附件二的基础上手工分词，但这么做的代价着实太大，得不偿失。因此，本文考虑采用开源的语料库，如 SIGHAN05[15]提供的 PKU 和 MSR 预料库。考虑到 MSR 在标注一致性上要优于 PKU，这一点可以用历史报告佐证。并且 MSR 的拆分颗粒度较大，一些地名 MSR 不予拆分，因此适合用在本场合中。MSR 语料库分为训练、测试语料库，其部分展示如下：

“ 人们 常 说 生活 是 一 部 教科书 ，
而 血 与 火 的 战争 更 是 不可多
得 的 教科书 ， 她 确实 是 名副其
实 的 ‘ 我 的 大学 ’ 。
“ 心 静 渐 知 春 似 海 ，
花 深 每 觉 影 生 香 。
“ 吃 屎 的 东西 ，
连 一 捆 麦 也 铡 不 动 呀 ？...

由于汉语的分词问题实际上是一种序列标注的问题，定义标注集为 $\{B, M, E, S\}$，其中 B, M, E, S 分别代表开头、中间、结尾和单个词。于是例句 我爱第八届泰迪杯挑战赛拆分可得：

[我/S，爱/S，第/B八/M届/E，泰/B
迪/E，杯/S，挑/B战/E，赛/S]

因此对于每一个汉字，都有一个状态与之对应。很显然这是一个分类问题，其亦可以使用机器学习解决。

[12]不建议读者这么做，根据笔者的许多研究和实践，发现正态分布类型的准确度等指标往往较低，无论在特征连续与否都是如此。笔者认为这是正态分布参数难以训练的结果

[13]即拉普拉斯修正系数为 1 的平滑策略

[14]相应的收敛会减缓，但不影响收敛

[15]第二届国际中文分词评测，可免费用于研究目的

考虑到一个汉字的状态，与前面一个汉字的状态有关。因此，这里结构判别模型——**条件随机场模型**解决。当然，笔者始终秉持着机器学习是一门实践的艺术这一理念，在考虑条件随机场时，亦通过多种模型筛选的方式，发现其效果最优，才采用该模型的，具体做法详见附录 B。

条件随机场类似于隐马尔可夫模型，如图??所示。其中特征 $\boldsymbol{x}_t$ 由 n 个连续的汉字 $x_i, i \in \{1, 2, \cdots, n\}$ 组成，这里取 $n = 5$。方块可以理解为一个特征函数 $f_k(y_{t-1}, y_t, \boldsymbol{x}_t)$，而 $\boldsymbol{y}_t = (y_1, y_2, \cdots, y_5), y \in \{B, M, E, S\}$ 属于标签向量。

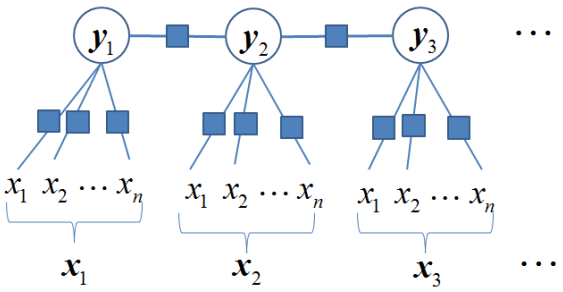

图 4. 条件随机场原理

于是，条件随机场的定义如下：

$$p(\boldsymbol{y}|\boldsymbol{x}) = \frac{1}{Z(\boldsymbol{x})} \prod_{t=1}^{T} \exp\{\sum_{k=1}^{K} \boldsymbol{w}_k f_k(y_{t-1}, y_t, \boldsymbol{x}_t)\} \quad (3)$$

其中 $\boldsymbol{w}_k$ 为待训练参数，而 $Z(\boldsymbol{x})$ 为归一化系数，其值为：

$$Z(\boldsymbol{x}) = \sum_{\boldsymbol{y}} \prod_{t=1}^{T} \exp\{\sum_{k=1}^{K} \boldsymbol{w}_k f_k(y_{t-1}, y_t, \boldsymbol{x}_t)\}$$

条件随机场的训练比较冗长，篇幅所限，这里不再详细介绍，具体内容可参阅参考文献 [?]。使用维特比算法 [?] 训练模型[16]后，即可用模型给语料库进行序列标注，并根据标注结果分词即可。

3.1.2 停用词过滤与命名实体合并

在汉语中，有些词语如的、啊、呢、换句话说、总而言之等对句子的信息影响不大。并且，**标点符号**和制表符等**特殊符号**亦不影响语义。因此需要在粗拆分的基础上过滤掉这些**停用词**。另外，一些人民、地名和机构名，以及数字等在粗拆分的基础上，需要将其再度合并。

这些都可以用**正向最长匹配**算法实现。该算法需要一个词典，以停用词过滤为例，这里使用的是 HanLP 开源词典[17]。正向最长匹配从某个汉字开始，从前往后的扫描每个汉字。若途中构成的词存在于词典中，而与下一个汉字组合却不存在（即最长），则将其过滤。

对于人民、地名和机构名也是一样，结合某部字典，通过**正向最长匹配**，在粗分的基础上进行再合并。

通过上述条件随机场、停用词过滤等处理后，以附件三第二条留言为例，其分词结果如下所示。其中词“10年”就是粗分后合并的结果。

[A市，A，6区，道路，命名，规划，已经，初步，成果，公示，文件，转化，成为，正式，成果，希望，加快，完成，路名，规范，道路，安装，路，名牌，变更，路，名牌，及时，更换，A，6区，农村，门牌，10年，未曾，更换，会，统一，更换，现在，找，地方，只能，说，路口，没有，充分，发挥，路名，地名，作用，A，6区，行政区划，已经，调整，完毕，门牌，更新，应该，同步，开展]

3.1.3 词袋模型与 PCA 降维

为了将分词后的文档转换为结构化的特征向量，本文将再次采用??小节所述的词袋模型，从而将语料库转换为 4326×42754 的稀释矩阵。同样，为了节省内存，这里仍旧采用??小节的方法存取稀疏数据。

值得注意的是，将语料库转换为特征向量的方法还有许多，例如 TF-IDF。但是，若采用 TF-IDF，由于其实现是基于词袋模型的，如果采用这种方法，会导致稀疏特性遭到破坏，增大内存与 CPU 的负荷。其二，TF-IDF 将语料库中出现的高频词赋为低值，这导致了诸如地名等重要特征无形中被忽视。对于聚类

[16] 训练好的模型大小 156MB，恕不上传。

[17] 见文件 stopwords.txt

[18] 实际上在少主题的聚类中使用 TF-IDF 方法会更好，比如图像的 BOW 聚类。对于这种多主题的聚类，TF-IDF 实在乏善可陈。这一点笔者已经在之前的研究中验证过

问题而言，这样反而弄巧成拙[18]。

由于转换后数据的特征仍然居高不下，因此这里考虑对数据进行主成分分析（后文将简称为 PCA）。考虑到特征个数远大于样本容量的情况下，样本通常线性可分。因此，这里采用的 PCA**不使用核函数**。如图**??**所示，存在一个两个特征的数据集，PCA 首先找出其**特征值和特征向量**。根据数据所在两个方向的稀疏程度，将数据投影到数据分布较稀的方向中。这是因为稀疏意味着相同量纲的情况下，信息量较大。因此在用 PCA 降维时，往往将其投影到 e_1 上。

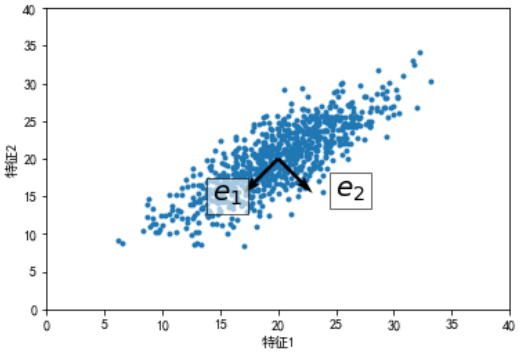

图 5. PCA 原理简介

评价 PCA 效果可用**解释方差占比**，其可由单个变量与总方差的方差之比算出，能够一定程度说明该变量在总样本中的信息量。如图**??**所示，使用 PCA 将其样本的特征个数从 42754 个降低至 1000 个[19]，并且其累计的解释方差占比已经超过 90%，可见其信息下降不会太大，降维是合理的。

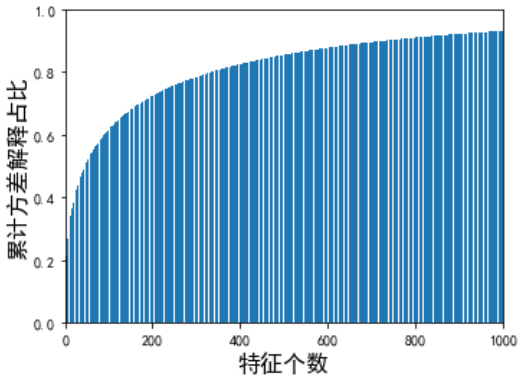

图 6. 特征个数与累计解释方差占比

至此，本文已将非结构的文本处理成结构化的数据集。

[19]选择 1000 个特征的理由见

3.2 文本聚类

得到结构化的数据集后，就可以采用聚类方法，从而实现文本的聚类。常用的聚类算法有 K-means 聚类，但其聚类簇数 k 需要人工择定，不太现实。因此本章将采用自适应的聚类算法——DBSCAN 算法。

3.2.1 DBSCAN 原理

设数据集为 $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_m\}$，$m$ 为样本容量。记 DBSCAN 的参数 ε 和 m' 为预设的阀值参数，定义相关概念如下：

- **邻域**：类似于 kNN 算法 [?] 的近邻的概念，定义样本个体 $\boldsymbol{x}_i$ 的 ε 邻域为距 $\boldsymbol{x}_i$ 的欧氏距离小于预设阀值 ε 的个体：$N(\boldsymbol{x}_i) = \{\boldsymbol{x}_j | d(\boldsymbol{x}_i, \boldsymbol{x}_j) \leq \varepsilon\}$；同时定义 $|N(\boldsymbol{x}_i)|$ 为个体 $\boldsymbol{x}_i$ 的 ε 邻域包含的个体数。

- **核心个体**：若个体满足 $|N(\boldsymbol{x}_i)| \geq m'$，则称为 $\boldsymbol{x}_i$ 核心个体。

- **密度可达**：若个体 $\boldsymbol{x}_j$ 在核心个体 $\boldsymbol{x}_i$ 的 ε 邻域内，则称 $\boldsymbol{x}_i$ 可达 $\boldsymbol{x}_j$。密度可达具有传递性：若 $\boldsymbol{x}_i, \boldsymbol{x}_j$ 均为核心个体，且 $\boldsymbol{x}_i$ 可达 $\boldsymbol{x}_j$，$\boldsymbol{x}_j$ 可达 $\boldsymbol{x}_k$，则 $\boldsymbol{x}_i$ 可达 $\boldsymbol{x}_k$。密度可达不具有对称性，即 $\boldsymbol{x}_i$ 可达 $\boldsymbol{x}_j$ 不一定有 $\boldsymbol{x}_j$ 可达 $\boldsymbol{x}_i$，除非两者皆为核心个体。

- **密度相连**：核心个体 $\boldsymbol{x}_i$ 可达 $\boldsymbol{x}_j, \boldsymbol{x}_k$，则称 $\boldsymbol{x}_j, \boldsymbol{x}_k$ 密度相连。

如图**??**所示，其中棱形个体为核心个体，且 $m' = 3$；ε 邻域为图中的圆圈。直线相互连接的个体为密度可达的核心个体串，核心个体串的 ε 邻域内的所有个体密度相连。更加详细的描述可参阅参看文献 [?]。

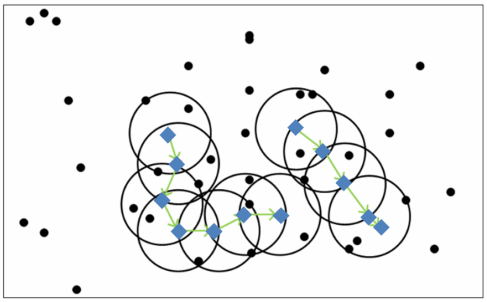

图 7. 核心个体,ε 邻域、密度可达与密度相连

因此，如果能找到如图**??**所示的核心个体串，将这些串的 ε 邻域构成的子集作为一个聚类簇，即可实现对样本的聚类。从图中亦可以看出，参数 ε, m' 均影响聚类结果。

为了实现聚类，DBSCAN 通过任意地选择某个核心个体作为种子，并根据该核心个体的所有密度可达的个体构成一个簇。显然，该簇的所有个体密度相连，且有可能包含不止一个核心个体（即该种子能生成核心个体串）。之后，再次选择一个不属于任何已知簇的核心个体作为种子，将其所有密度可达个体构成一个簇，如此循环直到再无未归类的核心个体为止。显然，经过上述循环后，可能仍具有许多的非核心个体未被聚类。在 DBSCAN 中，一般将其标记为噪声点。另外，DBSCAN 对个体进行聚类时一般采取先到先得的原则。即对于那些被多个核心个体可达、且核心个体不属于同一簇的个体，它的所属簇等于其第一次被划分的簇。

3.2.2 文本聚类与热度排行

对于文档的聚类而言，显然经过 DBSCAN 后，其噪声点或离群留言应该较多。换句话说，必须找到合适的参数，使得噪声点较多的情况下，尽可能产生较多的、留言个数大于 2 的簇。

因此，文章使用 $\varepsilon = 3, m' = 2$ 的 DBSCAN 聚类算法[20]，将每一句留言进行聚类，或者说将相近的留言归为一簇。于是，便可在簇的基础上，分析出热门问题。聚类结果如下所示，其中 -1 表示仅包含一条留言的簇。展示格式为“簇：留言数”：

$$\{-1:3924, 0:3, 1:2, 2:2, 3:3, 4:2, 5:2, 6:2, 7:2, 8:3, 9:2, 10:2, 11:2, 12:3, 13:4, 14:2, 176:2, 177:2, 178:2\}$$

可见，留言数大于 1 的簇共 179 个，其余均为单条留言或离群留言。鉴于总留言数共 4326 条，而离群留言共 3924 条，可见聚类效果还是比较可观的。

为了提取出热门问题，首先需要斟酌每一条留言的热度。设一条留言的**关注度**为 $y_i, i \in \{1, 2, \cdots, 4326\}$。由于发起一条留言所需要的代价远比点赞要大，因此设置每条留言的**起始关注度**为 10。留言每被**点赞**一次，增加 1 点关注度。考虑到热度的定义包含群众对问题的参与度，因此实际上留言的**反对数**亦一定程度上反应了留言的热度。但是，由于反对终究是对留言的否定，因此，文章考虑留言每被反对一次，则**倒扣** 0.5 点关注度。

另一方面，考虑到留言的点赞数、反对数会随着时间逐渐积累，但相反问题会被解决。因此，必须考虑进时间因素，即历时较长的留言的关注度应适当降低。不妨记留言 i 由于时间因素，导致的**关注度降低量**为 Δy_i。分析语料库中的时间，可以发现留言最近时间为:T=`2020-1-26 19:47:11`，而最远时间为:`2017-6-8 17:31:20`，即最远的时间跨度为 962 天。

由于 Δy_i 随着时间跨度 Δt，即与当前时间和 T 的间隔天数[21]应呈指数关系。换句话说，当前关注度衰减在近几个月的衰减不明显，之后则显著增大。因此，可以设 Δt 与 $-\Delta y$ 的表达式为：

$$-\Delta y = a \exp(b\Delta t) - c \tag{4}$$

为了求解出参数，文章统计出关注度最大的留言为 2107。因此，这里考虑经过 962 天后，关注度下跌 2107。经过 962/2 天后，关注度下跌 2107/16。于是令时间跨度以天数为单位，可列出方程组为：

$$\begin{cases} a\exp(0) - c = 0, \\ a\exp(\frac{962}{2}b) - c = 2107/16, \\ a\exp(962b) - c = 2107 \end{cases} \tag{5}$$

[20] 其参数选择方法见底层实现的**??**小节

[21] 本文在求时间的天数差时精确到秒

最后求得关注度降低函数为:

$$-\Delta y = 262.125\exp(0.0023\Delta t) - 262.125 \quad (6)$$

有关关注度下跌函数的更多细节，请参阅附录 B。

由于留言的关注度不能为负数，因此结合时间衰减，留言 i 的关注度为：

$$\begin{cases} y_i = 10 + Y - 0.5N + \Delta y_i & y_i > 0 \\ 0 & y_i \leq 0 \end{cases} \quad (7)$$

其中 Y, N 分别为点赞数和反对数，Δy_i 则根据(??)式与留言 i 的时间跨度算出。

于是，把聚类簇中的所有留言的关注度累加，即可得到该簇的热度。由于一个聚类簇中的留言是相近的，故一个簇可以视为一个问题。之后，根据每一个簇的热度降序排行，即可找出前 5 个热门问题。

3.3 内容归纳与关键句提取

虽然根据??小节提取出来的 5 大热点问题仅有一条留言。但考虑到留言的字数较多，并且实际应用中，某个问题可能包含多条留言。这时为了概括问题描述，可以将簇内的所有留言合并成一个**合成文档**，再使用某些算法进行关键句提取。本节将展示关键句提取算法——TextRank 算法，并结合示例问题，展示其使用方法。

3.3.1 PageRank 与 TextRank

PageRank 是 Google 开发的一种用于网页排序的算法。其将互联网视为有向图，网页视为图的节点，两个网页之间的连接为节点的有向边。每个网页的得分取决于外链的数量，和所链接节点的得分。设节点 V_i, V_j 之间存在有向边，则其权重的可由式算出：

$$S(V_i) = (1-d) + d \times \sum_{V_j \in \mathrm{In(V_i)}} \frac{1}{|\mathrm{Out(V_j)}|} S(V_j) \quad (8)$$

其中 $S(V)$ 为节点 V 的得分，其初始化分数为 1。$d \in (0, 1]$ 是一个常数因子，模拟用户点击网页 V_i 中的链接的概率。

将 PageRank 推广到关键句中，以句子为节点[22]，设每条句子两两之间都存在链接。并记 d 为当前句子与其余所有句子之间的“间隔”均值，即处在中心位置的句子 d 的取值越大。

由于所有句子都存在链接，因此直接使用式(??)是不可取的。这是因为 $\mathrm{Out(V_j)}$ 等于链接个数，因此无法区分句子是否“关键”。因此，这里定义两条句子的相似度 BM25 如下：

$$BM25(V_i, V_j) = \sum_{k=1}^{n} IDF(term_k) \frac{TF(term_k, V_i)(\alpha+1)}{TF(term_k, V_i) + \alpha(1 - \beta + \beta\frac{|V_i|}{DL})} \quad (9)$$

其中 α, β 为待定常数，其值分别决定了词频、句子长度为 BM25 的贡献。α 越大，则词频对权重的正面影响越大。若 β 越大，则句子长度对权重的负面影响越大。这里无差别对待，即取 $\alpha, \beta = 1$。$|V_i|$ 为句子包含的单词量，因此计算时需要用??小节的**条件随机场模型**分词。DL 为**合成文档**中，每一条句子的平均词量。以句子为单位，TF, IDF 分别为单词在文档中的词频和逆文本频率。其中 $TF(term_k, V_i)$ 等于单词 $term_k$ 在句子 V_i 中的词频。而 IDF 的计算公式如下：

$$IDF(term_k) = \log(\frac{S}{DF+1}) \quad (10)$$

其中 S 为合成文档中句子的总数，DF 为包含单词 $term_k$ 的句子个数。

于是，仿照式(??)，即可找出每一条句子的“得分”，具体算式如下：

$$S(V_i) = (1-d) + d \times \sum_{V_j \in \mathrm{In(V_i)}} \frac{BM25(V_i, V_j)}{\sum_{V_k \in \mathrm{Out(V_j)}} BM25(V_i, V_k)} S(V_j) \quad (11)$$

设每条句子的初始化分数为 1，经过多次迭代后，即可得出每条句子的最终得分。之后，根据得分最高原则，即可找到合成文档中的关键句。算法详情可参阅参考文献 [?] 的第 9 章。

之后，从该关键句人工提取出对应簇的问题描述即可。至于地区或人群，可以在关键句相应的留言中提取。

[22]这一点可以用正向匹配，搜索逗号、句号、分号等，从而将句子从文本中提取出来

3.3.2 热点问题归纳

经过聚类后，计算聚类簇的热度指数，并降序排行后即可找到前 5 个热点问题。若热点问题只包含**一条**留言，那么可以**直接**人工概括，或**提取多条关键句后再**概括 (本文采用此种办法)。若某问题包含多条留言，可取留言的时间跨度作为问题的时间范围。根据**??**小节提取出关键句后，**根据关键句人工概括**问题描述，并从关键句所在的留言中，提取其地区范围和人群。

考虑到藉由上述热度排行算法得出的前 5 个热点问题，它们包含的留言只有一条。为了展示关键句提取算法的使用，笔者总结了前 10 个热点问题。

以热度排行第 7 的问题为例，其包含 9 条留言。为了得到问题的总体描述，可以将这 9 条留言合并成一个文档，再进行 TextRank 排行，从而提取出 3 条关键句，如下所示：

[请问A7县文体中心乒羽中心何时能正式对外开放，请问北横线还修吗，请问A4区植基路为什么一直是断头路]

结合三条关键句及其所在留言，可以断定该热点问题为咨询和反馈 A 市城市规划有关问题。同理，可以归纳出前 10 个热点问题，它们的详细描述详见附件的 Excel 表格。

3.4 底层实现

为了使文中所描述的方法投入工业应用之中，需要适当地改进一些算法。另外，本节还将解释 DBSCAN 的参数来源，以及调参根据和方法。

3.4.1 关于匹配算法的查询优化

在进行停用词过滤（见小节）等过程中，需要使用到预定义的 字典。并且每一次扫描的过程中，都需要查询一次字典，已确定是否有匹配的单词。如果使用顺序的存储结果，则每次查询的复杂度为 $O(n)$，显然是非常巨大的。

虽然将字典存储成**哈希表**能够使搜索的复杂度降至 1[23]，但是在词典较大的情况下，还是不推荐使用的。考虑到停用词字典的大小才 7KB 左右，因此本文在实现的使用哈希表匹配字典。

3.4.2 截断奇异值分解降维

由于普通的 PCA 降维需要进行归一化处理，并且需要求解数据的**协方差矩阵**的特征值、特征向量。然而，由于笔者在处理数据时，将稀疏数据用特殊方式保存。因此归一化、求解协方差矩阵等操作，势必会破坏矩阵的稀疏性，或生成新的、等尺寸的稠密矩阵。所以说，直接进行 PCA 是不可取的。

因此，本章在实现 PCA 降维时，采用了截断奇异值分解降维的方法。该方法的原理类似于 PCA，即考虑将数据投影到某个特征向量所在的方向中。但其计算特征向量时，直接在稀疏数据中算出，因此可以防止稀疏性被破坏。

3.4.3 DBSCAN 调参过程

本文采用 $\varepsilon = 3, m' = 2$ 的 DBSCAN 进行聚类，其效果经过是筛选与斟酌的。笔者发现，当 $m' > 2, \forall \varepsilon \in (0, 10)$ 时，样本个数大于 1 的聚类簇数显著减少，从原本的近 200 个降低至不到 50 个，而每一个聚类簇包含的样本数仍旧是个位数。因此有理由认为，$m' > 2$ 会使得聚类的区分度过高。

当 $\forall m' \in (0, 2], \varepsilon > 4$ 时，虽然样本个数大于 1 的聚类簇数并没有太大的变化，但是某个聚类簇却包含了大于 100 个样本。所以，有理由认为 $\varepsilon > 3.5$，会导致区分度过分下降。

当 $\forall m' \in (0, 2], \varepsilon \leq 3$ 时, 很少有样本数大于 10 的簇出现。综上当 $\forall m' \in (0, 2], \forall \varepsilon \in (3, 4]$ 时效果最佳。

经过热度排行的实际效果考量，文章选择 $\varepsilon = 4, m' = 2$ 作为 DBSCAN 的参数。

4 答复相关性与完整性分析

为了评价留言答复的相关性，本文将考虑采用**词向量**，即深度学习的方法，从而比较留言详情与留言答复之间的相似度。并以两个文本的相似度，作为答复的相关性的评价指标。考虑到留言和答复的字数都比较长，且词向量的运算也是非常耗时的。因此，在计算文本相似度之前，首先采用小节所述的关键句提取方法，从留言和答复中提取出若干关键句，再比较关键句之间的相似度，从而作为答复的相关性指标。

顺带一提，区别于文本分析、文本聚类，两文本相似度的评价问题较为复杂，这是依赖手工提取特征的机

[23]内存运地址运算包含于芯片的指令集中，不消耗 CPU 周期

器学习方法所处理不了的。因此，为了评价两个文本的相似度，往往需要神经网络这种“黑盒子”模型，从而自动文本的特征信息。

为了评价答复的完整性与可理解性，本文将采用二元语法字典，以相邻两个汉字为单位，逐一扫描留言答复。并匹配当前扫面是否存在于二元语法字典中，从而判断留言的局部完整性核可理解性。

4.1 答复相关性评价

如上所述，为了评价答复的相关性，本文将采用词向量的方法。运用关键句提取算法，分别从留言和答复中提取出其关键句。计算它们的词向量后，只要根据词向量之间夹角的余弦值，即可度量它们的相似度。

4.1.1 词向量与 one-hot 编码法

语言学家 J.R.Firth 认为，意义相似的词语，其上下文也是相似的。根据这一点，每个词语都存在一个上下文**词向量**与之对应，使得统计模型能够根据该单词的词向量预测到该单词。换句话说，词向量表达了该单词。

但是，如何找到每一个单词的词向量呢？显然，使用人工分析的方法犹如天方夜谭，而根据手工特征为底子的机器学习亦难于胜任。因此，在这个问题上，能够自动提取特征的深度学习，即神经网络的方法有着天然的优势。

为了训练出每个单词的词向量，显然需要一个分词过的语料库。这里仍旧采用 SHANG05 的开源语料库 MSR 作为训练集。把单词所在位置的前**后相邻的词**作为训练特征，称为**上下文向量**。为了将单词转换为训练特征，这里采用处理无序字符串常用的编码法——**one-hot**编码法，对单词的上下文进行编码。

例如假设训练语料库为“第八届 泰迪杯 建模 挑战 赛”，则单词“建模”的上下文向量为[第八届，泰迪杯，挑战，赛]。使用 one-hot 编码法，可转换为 $\boldsymbol{x} = [[1,0,0,0,0]^T, [0,1,0,0,0]^T, [0,0,0,1,0]^T, [0,0,0,0,1]^T]$，并与单词 “建模”的 one-hot 编码 $y = [0,0,1,0,0]^T$ 对应。

显然，这是一个极其稀疏的矩阵，因此在存取方面，亦将其按**??**小节的方法处理。当然，语料库包含的词量远远不止这些。此时可以定义一个窗口，以 1 为步长从前往后第扫描语料库。并以使用窗口中心和其前后单词，通过 one-hot 编码法生成多个上下文向量。注意，此时一个词对应的 one-hot 向量长度应等于语料库中，词的总数。

4.1.2 词向量与 CBOW 模型

上一小节我们知道，语料库每一个单词[24]的 one-hot 向量 $\boldsymbol{y}_i$，都对应一个或多个上下文向量 $\boldsymbol{x}_i$。因此，若以 $\boldsymbol{x}_i$ 为输入，以 $\boldsymbol{y}_i$ 为输出，就可以训练一个深度学习模型了，其中比较著名的就有 CBOW 模型。CBOW 模型实际上是一个三层的神经网络模型，如图**??**所示。

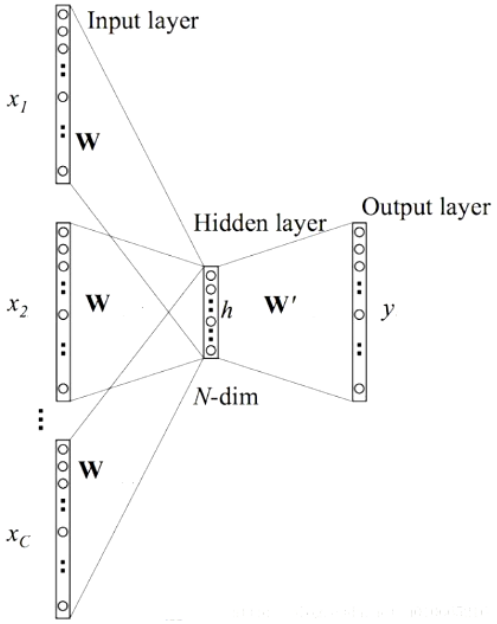

图 8. CBOW 模型概述[1]

其中 $x_i, i \in 1,2,\cdots,C$ 为单词 one-hot 编码后的列向量，C 取决于窗口的大小。设窗口大小为 m, 则 $C = 2m$。矩阵 $\boldsymbol{W} \in \mathbb{R}^{n\times|V|}$ 为 CBOW 网络的输入层到隐藏层的权重矩阵，$\boldsymbol{W'} \in \mathbb{R}^{|V|\times n}$ 为隐藏层到输出层的权重矩阵。其中 n 为隐藏层的节点数[25]，需要人工择定；$|V|$ 为语料库中单词“种”数。$\boldsymbol{W}$ 的第 i 个列向量为语料库中第 i 个单词的**输入词向量**，记为

[24]除去开头的，前窗口半径个单词外

[1]本图改自 https://docs.google.com/file/d/0B7XkCwpI5KDYRWRnd1RzWXQ2TWc/edit

[25]本文采用的 CBOW 模型的隐藏层节点个数为 100 个。

$\boldsymbol{w}_i$；类似地，$\boldsymbol{W}'$ 中第 j 个行向量为第 j 个单词的**输出词向量**，记为 $\boldsymbol{w}'_j$。由于输入词向量离输入层近，所以在应用中，一般**抛弃**输出词向量，将**输入词向量**作为单词的词向量模块使用。

输入层输入经过隐藏层节点后，转换为向量：$\boldsymbol{v}_1 = \boldsymbol{W}\boldsymbol{x}_1, \boldsymbol{v}_2 = \boldsymbol{W}\boldsymbol{x}_2, \cdots, \boldsymbol{v}_C = \boldsymbol{W}\boldsymbol{x}_C$。将其求取平均后得到 $\bar{v}$ 供给输出层，从而得到净输出 $\boldsymbol{u} = \boldsymbol{W}'\bar{\boldsymbol{v}} \in \mathbb{R}^{|V|}$。

最后通过 softmax 激活函数，得到 $\boldsymbol{y}_i$ 的预测值 $\bar{\boldsymbol{y}}_i = \text{softmax}(\boldsymbol{u})$。如同所有机器学习模型一样，为了训练出词向量矩阵 $\boldsymbol{W}$，需要定义损失函数如下：

$$H(\bar{\boldsymbol{y}}, \boldsymbol{y}) = -\sum_{j=1}^{|V|} \boldsymbol{y}_j \log(\bar{\boldsymbol{y}}) \tag{12}$$

因此，求取词向量的过程就等价为让式(??)最小的优化问题。同样地，该优化问题可以使用随机搜索算法，如 LBFGS、Adam 等算法求解。有关 CBOW 模型，可以参阅 [?] 获取详细解释。

4.1.3 答复相关度计算

通过 MSR 语料库，求出 CBOW 模型的参数后，即可通过输入权重矩阵 $\boldsymbol{W}$ 得出某个词的词向量。于是，对于某一条句子，找出一条句子中每一个单词的词向量。之后，将这些词向量取平均，得到一个稠密向量 $\boldsymbol{s}$。于是，对于任意的两条句子，都可以通过 $\boldsymbol{s}_i, \boldsymbol{s}_j$ 夹角的余弦值，作为句子的相似度，如式(??)所示：

$$\cos\theta = \frac{\boldsymbol{s}_i \cdot \boldsymbol{s}_j}{|\boldsymbol{s}_i||\boldsymbol{s}_j|} \tag{13}$$

于是，对于留言以及留言答复，可以使用??小节的方法，分别找出留言和答复的 5 条关键句。分别计算它们的词向量 $\boldsymbol{s}_i, \bar{\boldsymbol{s}}_j$，再找出它们之间的最大值作为留言的相关性即可，如式(??)所示：

$$S = \max_{\boldsymbol{s}, \bar{\boldsymbol{s}}} \cos(<\boldsymbol{s}, \bar{\boldsymbol{s}}>) = \frac{\boldsymbol{s} \cdot \bar{\boldsymbol{s}}}{|\boldsymbol{s}|\bar{\boldsymbol{s}}|} \tag{14}$$

由于 $S \in [-1, 1]$，若 $S < 0$，则证明答复与留言风马牛不相及。

4.2 答复的完整度与可理解性

虽然答复的完整性可以使用**语法分析**，来判断一个句子是否通顺。但是要进行语法分析，首先要对单词进行**词性标注**。如同分词一般，词性标注亦属于一个序列标注问题，需要训练一个机器学习模型从而自动标注。标注之后，还需要训练又一个模型，从而进行语法分析。虽然可以通过一些开源的语料库解决以上两个问题，但是为了根据语法判断句子是否通顺，或计算其通顺程度，还需要再训练一个模型。

实际上，词汇标注和语法分析在业界都已实现，笔者之前业已实现过，但最后一步却是最难的。首先，根据语法计算句子的通顺程度，是一个回归问题。另外，最重要的一点是语料库的匮乏和不统一。如何根据语法判断是否完整和通顺，就已经众说纷纭了，更何况给其打分。因此，要使用机器学习的方法完成，目前似乎没有可行的办法[26]。

于是，文章姑且采用二元语法与字典匹配，来判断句子的**局部整体性**。如??小节所述，二元语法即通过正向地扫描句子，将相邻的两个汉字视为一个词的方法。之后，在某部收录了所有二元语法的字典中[27]，查找当前词是否存在字典中。这么做能够在找出错别字的同时，判断句子的局部连贯性。

于是句子的完整性等表征指标 C，可以根据式(??)计算：

$$C = 1 - N_{no}/L \tag{15}$$

其中 N_{no} 为字典中无法匹配的、答复的二元语法“词”的总数，L 为留言答复的二元语法“词”的总数。

4.3 答复评价模型

综上所述，某条留言答复的得分为：

$$G = \frac{S + C}{2} \tag{16}$$

其中，S 表征留言答复的相关性，其可由式(??)算出。C 表征了留言答复的完整度、可理解性，可根据式(??)算出。已示例问题为例，答复的评分可见附件的 Excel 表格。

[26]但英文的 NLP 却有相关的实现库

[27]该字典必须采集自“通顺”的语料库中，这里采用何晗老师收集的开源核心字典

4.4 底层优化算法

为了提高模型的训练效率，本文在训练 CBOW 模型时主要采用了并行多线程的方法。另外，在计算答复的得分指标 C 时，由于搜索字典需要大量的时间，因此也需要进行一定的优化。本节即向读者们展示这些底层的优化算法。

4.4.1 双数组字典树

如上所示，在计算 C 时，由于搜索花费了大量的时间，需要首先进行优化。?? 小节所述的哈希表的方法是不可行的。这是因为二元语法字典太大，直接将其加载到内存中，并分配相应的哈希散列，显得不太现实。

因此，对于这种情况，文章采用**双数组字典树**的方式加载字典到内存中。首先，字典树是一中树型存储结构，其并非直接在节点上存储词，而是将词存储在节点之间的连线上。如图??所示。由于扫描并非之前的正向最常匹配，因此字典树的根节点引伸出去的叶节点较多。因此，为了提高速率，这里采用首字哈希算法，即给根节点用一个哈希表映射到其叶节点。这样，就能够以少量的存储资源，换取搜索的速度。如图??所示：

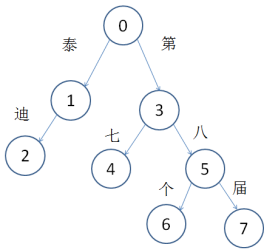

图 9. 字典树原理

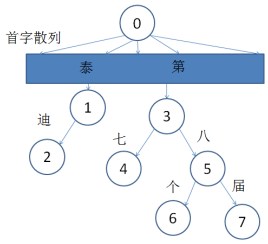

图 10. 首字散列的字典树

但一个不好的地方就是，由于其余节点都用二分法查找，其复杂度为 $O(\log(n))$。当子节点的个数 n 很大时，查询依旧很慢。因此，为了提高单次查询的，这里采用双数组字典树进行改进。

双数组字典树大致原理使用两个数组表示字典树中的节点，节点的转移可用数组 base 和 check 的元素和下标表示。当节点 a 接收到字符 $char$ 时，将会使用如下查询：

```
b = base[a]+char
if  check[b] == base[a]:
    accept
```

于是，如果满足条件，则证明有相应的字符与之匹配。这样，就可以将原本 $O(log(n))$ 降低至常数 $O(1)$。这里引用参考文献 [?] 的结果，使用双数组树后，与顺序存储相比，其速度提高了近两个数量级。

4.4.2 搜索算法的并行实现

对于搜索算法（包括 LBFGS）而言，其参数迭代均可由式(??)表示：

$$\boldsymbol{\omega}_{k+1} = \boldsymbol{\omega}_k + \alpha \boldsymbol{d}_k \tag{17}$$

在串行实现中，似乎只有求出了 $\boldsymbol{\omega}_k$ 才“有资格”求出 $\boldsymbol{\omega}_{k+1}$。但是，由于随机搜索算法使用 mini-batch 求出梯度或海塞矩阵 $\boldsymbol{d}_k$，其值原本就并非精确的。所以，实际上 $\boldsymbol{\omega}_{k+1}$ 可从 $\boldsymbol{\omega}_{k-1}, \boldsymbol{\omega}_{k-2}, \cdots, \boldsymbol{\omega}_{k-n}$ 算出。从这一点出发，就可以用并行的方式训练模型的参数。

这里采用的并行模式是 Master-Worker 模式。Master 负责接收和分配任务，接收 Worker 中算出来的 $\boldsymbol{d}$，从而计算 $\boldsymbol{\omega}$，并反过来将其发配给 Worker。Worker 从 Master 那里接收最新的 $\boldsymbol{\omega}$，并运算 $\boldsymbol{d}$ 传输给 Master。

由于每个 Worker 之间的行为是异步的，因此在更新 $\boldsymbol{\omega}_k$ 时，某些 Worker 提供给 Master 的 $\boldsymbol{d}_k$ 存在延迟。换句话说，$\boldsymbol{d}_k$ 可能不是基于 $\boldsymbol{\omega}_{k-1}$ 得到的。

通过上述方法，即可将模型采用多线程并行训练。于是，训练一个 100 节点的 CBOW 网络仅耗时不到 3 分钟。实际上，这种并行的方法，也可以用于训练 AdaBoost(见??小节) 这样的集成模型。

5 结束语

机器学习是一门理论的科学，亦是一门实践的艺术。虽然汉语言处理这一领域已经走了相当长的一段时间，但仍有许多问题是有待解决的。比如很多开源的库都是学术性的，很多底层实现上都不太注重效率，一些具体的细节也经常被省略过去。另外，在某些问题，如根据语法评价句子的通顺、完整性这一块，仍旧是有待开辟的处女地。

5.1 笔者的话

在许多前人的文本分类问题研究中，都不太注重模型的筛选。笔者看到的许多论文中，都是援引外语 NLP 的结果，即使用贝叶斯分类器分类。虽然贝叶斯分类器的确效果优良，这一点亦在文中有所体现。但是，逻辑回归、支持向量机等模型的效果亦不可无视。因此，本文的贡献在于整合机器学习的，几乎所有常见的分类模型（不包括结构化预测模型，如条件随机场等），探究出合适于汉语文本分类的模型——贝叶斯分类器、SVC 和逻辑回归。希望本人的工作，能够对汉语文本分类这一块，共享一点点绵薄的力量。

不得不说，许多学者们把神经网络视为“三板斧”，这实际上是不可取的。这在文本分类领域更是如此，这一点本文业已说明。另外，许多直接从其他领域跨越到 NLP 的研究者，在使用模型之前，往往没有经过严格、科学的筛选，比如文中的 K 折交叉验证、网格寻优。甚至直接使用几个样本测试模型，而没有拆分训练集，就判断模型优劣与否。这些现象在 CV 领域更如此，这种乱象也从侧面反映了当前汉 NLP 的不成熟。希望本文的这种比较科学的模型筛选方法，能够帮助人们重新认识机器学习的整体流程。

5.2 本文的缺点与不足

本文最大的败笔，是使用了分词、词袋模型和 DBSCAN 算法进行文本聚类。这一点笔者在整理聚类结果时，发现有很多讨论同一话题的留言并没有被聚为一簇。实际上，笔者犯了一个错误，就是任何无监督聚类算法，在采用词袋模型时，都是依靠表面的“单词”是否完全一样来聚类。而要真正的将其留言聚类在一起，应该是将留言所要传达的“意思”相近的聚在一起，而非明面上的“单词”相同与否。因此，实际上要解决这一点，可以采用**??**小节所述的词向量，作为特征进行聚类。因此词向量表达了单词的上下文，或单词应该“所处”的位置。从某个方面来讲，词向量表达了单词暗面的“意思”。所以，如果使用**??**和**??**小节所述的方法，使用词向量和关键句度量留言的相似性，根据相似性聚类实际上会更好。另外，也可以使用词向量作为单词的特征，再用 DBSCAN 来聚类，虽然这么做计算机的负荷比较大。本文之所以不修改，是因为本文采取的文本聚类方法，好处之一运算负荷较低，且只需要存储 DBSCAN 的两个参数即可。这是训练 CBOW 模型、存储词向量矩阵（MSR 语料库的词向量约 50MB），并且需要语料库所不能比的。另外，在嵌入式 AI 开发中，研究人员很少采用深度学习，导致所谓的“惯性思维”。因此，这里也给同行们树立一个前车之鉴，以便后续的研究人员铺平道路。

其二是筛选模型是没有完全考虑集成模型。实际上 AdaBoost 是一个能够比肩深度学习的模型之一，但笔者只考虑其子模型取逻辑回归的情况。另外，由于训练算法设置的不合理，在训练过程中，出现 AdaBoost 无法有效收敛的情况，因此只能手工暂停训练。另外，很多边缘的、却有用的集成模型，比如 Stack 集成，本文没有考虑。实际上，Stack 集成的非同质集成，可能更加适合于汉文本分类任务。

其三，热度计算、留言评价算法的设置完全是根据个人的主观意识设计的。实际上，这些热度计算方法、留言评价算法的设计，至今还没有统一的方法。机器学习学者和语言学者的意见亦不统一，而笔者也缺乏对这方面的知识。另外，在设计关注度衰减函数时，亦没有科学的根据，这实在是相当遗憾的事情。

5.3 后续工作与展望

在**??**小节中，笔者曾经简略的提到，之所以线性分类模型的效果很好，是因为特征个数过高，导致样本线性可分的缘故。实际上，根据笔者的研究经验，这可能暗示了样本不够这一事实。文中的样本有 9000 多条，用来训练的样本接近 7000 条，而特征个数达到 3000 多。显而易见，样本是不够的。

另外，如前所述，本文并没有完全发挥出集成学习的潜力。希望在后续的工作中，能够以此为目标，挖掘出 Boost 集成、Stack 集成，在汉文本分类任务中的效果。

另一方面，在**??**小节中，笔者曾经谈到句子通顺性很难使用机器学习的方法。这是因为语料库匮乏的原因，或者说几乎没有。因此，这里也希望有关机构能够团结起来，进行语法通顺语料库的标注工作。例如可以在目前的依存语法的基础上，标出某句子的依存语法，进而评判句子的通顺度。希望这些绵薄的意见，能够起到抛砖引玉的作用。

最后，机器学习是一门理论的科学，也是一门实践的艺术，采用统计方法的 NLP 也是如此。希望在接下来的工作中，能够与君共勉。

A 附录 A

A.1 T 检验表

表 5. T 检验表

p 值	Adaboost	逻辑回归	贝叶斯分类器	SVC
Adaboost	1.00	0.60	0.23	0.59
逻辑回归	0.60	1.00	0.43	0.28
贝叶斯分类器	0.23	0.23	1.00	0.10
SVC	0.59	0.43	0.28	1.00

A.2 BP 神经网络解决分类问题

将数据通过??节的处理后，由于样本的因变量（即类别）为 0 ~ 6 的有序整数。因此，可见分类问题视为回归问题。于是，就可以训练一个用于回归问题的神经网络模型。模型的输出可以四舍五入取整，从而转换为整数，进而得到样本的类别

笔者尝试过直接使用 1 个节点，将激活函数设置为 softmax，但发现模型的精确度只有 20%。当然，笔者可尝试过用 one-hot 编码法，将输出转换为 7 个二值变量，从而转换为多标签问题，再用 7 个节点对应一个变量。但这么做很难用 F1 等指标评价模型，并且凭空多出了许多待训练的节点。

本文采用的神经网络的结构如表??所示。

表 6. 神经网络结构

	节点数	激活函数
隐藏层 1	500	RELU
隐藏层 2	250	RELU
隐藏层 3	100	RELU
隐藏层 4	50	RELU
输出层	1	线性函数

设置训练步数为 100，训练算法为步长 0.01 的 Adam，mini-batch 为 100。将数据集按 7 : 3 拆分为训练集、测试集。以**最小均方误差 (MSE)**作为代价函数，在训练集中训练模型的参数。模型在训练集、测试集中的 MSE 随迭代步长的变化如图??所示。

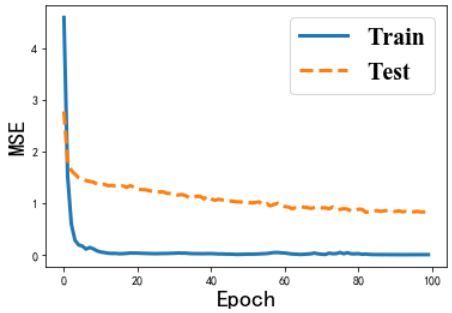

图 11. 未进行正则化的 BP 神经网络训练效果

从图中可以看出，模型存在过拟合的倾向。为了缓解过拟合，可考虑在模型训练过程中，采用 Dropout 正则化，从而介绍过拟合。本文采用的 Dropout 正则化是对节点的大颗粒正则化，即在每一次迭代时，随机地令某些节点（这里采用该层节点数的 10%）不参与训练。有关正则化的内容可以参阅参考文献 [?] 的第 65 页，这里不再复述。

如图??所示，采用 Dropout 正则化后，模型训练过程出现“毛刺”，但是模型的过拟合趋势有所缓解。

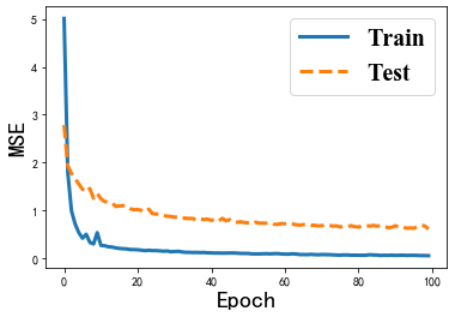

图 12. 正则化后的 BP 神经网络训练效果

计算模型在训练集、测试集中的 F1 值分别为：0.87 和 0.66，比之表??中的机器学习模型尚有不足。且模型所需的存储资源相当大（175MB），因此在 NLP 领域，采用深度学习显得非常的费力不讨好。

B 附录 B

B.1 各分词器在 MSR 语料库中的结果对比

由于分词问题实际上是序列标注问题，即分类问题，但某汉字的标注与上一个汉字有关，因此采用逻辑回归、SVC 等非结构化的模型效果势必较低。因此，这里采用结构化的分类模型，并从中选择最好的模型投入使用。

这里采用 MSR 语料库的训练集训练模型，并在测试集中评价其拟合优度。这里处理计算 F1 值之外，还要考虑未登录词的召回率，即分类器拆分完全陌生单词的能力。其结果如表??所示。因此，虽然训练代价最大，耗时最长[28]，但文章仍旧使用效果最佳的条件随机场模型。

表 7. 各结构化分词器在 MSR 语料库的表现

模型	F1 值	未登录词的召回率
一元隐马尔可夫模型	79.44	41.02
二元隐马尔可夫模型	79.2	42.11
结构化感知器	94.62	70.54
条件随机场	96.78	71.52

B.2 关注度下跌函数细节

根据关注度下跌函数的表达式(??)，绘制出图像如??所示：

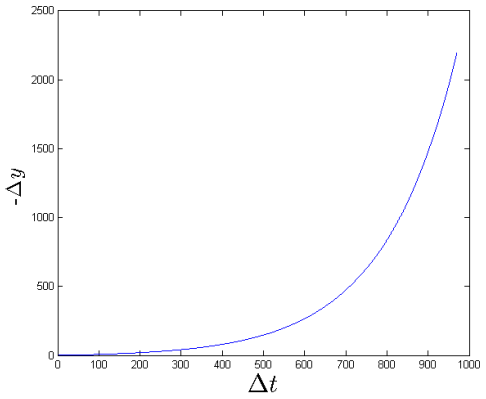

图 13. 关注度下跌函数图像

可以看出，随着时间跨度的增大，留言的关注度下降越迅猛。有意思的是，当时间跨度 $\Delta t = 30$，即一个月时，留言的关注度才衰减不到 1.7。其他数据见表??所示。

表 8. 常见时间跨度的关注度下降

时间跨度	衰减
1 个月	1.69
3 个月	6.14
6 个月	16.35
1 年	61.5
18 个月	185.8
2 年	528.4

[28]大约 8 小时左右，2.2GHz 的 CPU，无 GPU 加速