舆情监测与情感倾向分析建模

摘要

随着网络科技的迅猛发展，网络传播带来许多舆论问题。本文主要通过舆情监测与情感倾向分析建模，建立了舆情信息筛选模型、舆情数据抓取模型、舆情传播控制模型和舆情等级处理模型 4 个模型来解决相关的舆情分析问题。

针对问题一，本文提出一种针对特定主题的舆情筛选方法。首先，进行数据预处理，将完整文本转化为有关键词组成的集合，再利用 LDA 聚类将附件 1 数据分成三类，再计算用户的文字评论的 $TF-IDF$ 特征值，求解发现特征值排名前三的关键词也同样通过 LDA 聚类分成了三类，从而验证了模型的准确性。再以 $TF-IDF$ 特征值作为 SVM 分类器模型的输入变量，通过对原始的特征空间进行核函数映射，实现新建特征空间的线性可分。为了验证模型的实用性，随机选取搜狗新闻数据中 70% 的评论内容作为训练集，总计 70266 个评论，使用 $TF-IDF$ 特征训练分类器模型，然后将剩下的 30% 内容作为测试集，利用训练的好的超平面进行分类。结果显示分类器结果准确率达到了 92.10%，模型实用性较强。

针对问题二，本文提供一个数据抓取方法，包含诸如发表时间、评论人数、关注人数及具体内容等具有深层次分析价值的数据。我们使用 *Python* 爬虫，设计数据抓取算法。以手机端微博为例并沿用 $TFIDF$ 模型进行深入分析。

针对问题三，本文建立舆情传播过程中的情绪检测和控制模型，提供一种能够合理引导网民们情感倾向逐步转向对政府或企业有利的干预方法。首先通过模型一对文本数据进行分主题处理，然后为了判断舆情的情感导向，本文建立情感字典。主要基于 *Hownet* 基础情感字典、互联网网络情感字典表情符号情感字典、程度副词情感字典和否定词情感字典 5 类。然后采用情感倾向文字分析数学问题，即倾向性分析。通过建立词组情感计算公式对每组文本呢的关键词进行情感打分，从而确立文本情感倾向。对于正向性评论，本文考虑建立共鸣文本库，随机抽取相应文本赋于该评论下方；对于中性评论则不做处理；而对于负向性评论，本文通过相似度公式计算词组间的相似度，运用搜索算法在该主题正向性评论内找到相似度最高的文本赋于负向性评论下方。

针对问题四，本文建立舆情等级处理模型，考虑到当前舆论情况对政府和社会的影响程度判断和后续处理手段和力度的选择，对当前热点话题进行了舆论等级划分。结合舆论的基本影响指标和其细化后的影响指标，使用层次分析法对其当前对政府和社会影响程度和未来对舆论发展状况重要性进行评价，得出所有细化指标的权重。使用 *Dephi* 法客观准确的对其每个影响指标现状进行赋分，利用权重得出该舆论的总得分，按照评分体系的制定可按分数等级划分：[0,25) 为较小危险程度 [25,50) 为一般危险程度 [50,75) 为危险程度 [75,100] 为严重危险程度。

关键字： SVM　$TF-IDF$ 特征训练　*Python* 爬虫　倾向性分析

一、 问题重述

1.1 问题背景

如今多媒体发展迅速，信息的传播渠道多种多样，而在恶性公共事件发生时，相关信息将会在短时间内快速传播，引发广大群众的注意。在这一过程中，相关负向报道或者群众主观片面的失实评判往往在一定程度上激发群众的危机感，严重的则会影响到国家政府的公信力，影响到企业的口碑形象，因此舆情对政府和企业都至关重要。同时，通过舆情的情感倾向预测，有助于企业能够了解媒体或网民对相关事件或者品牌的舆情情感倾向分布和情感倾向趋势，同时能快速识别负向情感倾向的文章或评论，及时对口碑进行维护。

1.2 问题描述

问题一：针对媒体或网民评论的数据库，提供某一主题的舆情筛选方法。

问题二：为了从互联网上自动获取大量舆情数据，且提取出包含诸如发表时间、评论人数、关注人数及具体内容等具有深层次分析价值的数据，需要构建一个全新的数据的抓取方法。

问题三：网络舆论作为社会舆论的网络反映，是社会舆论最主要的构成成分之一。正确引导网络玉林，阻止不良态势的进一步蔓延，将给政府和企业带来巨大的帮助。采用删除评论等模式处理舆情，不仅不能解决问题，网民们可能还会以另外一种形式继续传播舆情。因此请提供一种能够合理引导网民们情感倾向逐步转向对政府或企业有利的干预方法。

问题四：网络舆情传播速度的差异性大，且管理部门在舆情管理的资源有限。为了在资源有限的情况下对不同的阶段的舆情进行不同等级的干预，需要度量网络舆情事件的重要性，且设计一个全面考虑舆情传播时间、规模及网民情感倾向的舆情处理等级的划分方法。

二、 问题分析

2.1 问题一的分析

问题一要求我们针对特定的主题，提供舆情筛选方法。根据题目中所给出的附件 1 一些通过技术手段抓取了部分媒体或网民评论的数据，我们随机选取 80% 的内容作为我们模型的数据集，将其余 20% 部分作为我们模型实验结果的验证集。选用数据后进行 RBF 核函数类型的模型进行分析。具体步骤为首先对选用的数据集进行文本处理。将文本拆分为单词，使用这些单词作为训练集输入来训练 SVM，得到几个大类。然后选用数据集的数据进行分类并使用 $TF-IDF$ 作为特征进行舆情筛选。

2.2 问题二的分析

问题二要求我们提供一个数据抓取方法，包含诸如发表时间、评论人数、关注人数及具体内容等具有深层次分析价值的数据。我们使用 *Python* 爬虫，设计数据抓取算法。首先通过浏览器审查，在搜索框搜索指定内容，在网络界面捕获 *XHR* 格式的 *URL*，作为抓取数据的来源地址。然后根据 *Python* 第三方库 *requests*，设置头部，模拟浏览器内核 *Chrome*/76.0.3809.100 内核解析该源地址的 *json* 文件内容并将数据储存为 *CSV* 文件。最后，以手机端微博为例并继续沿用问题 1 的基于支持向量机的舆情筛选模型挖掘有价值的数据进行深层次分析。

2.3 问题三的分析

针对问题三，提供一种能够合理引导网民们情感倾向逐步转向对政府或企业有利的干预方法。为了判断舆情的情感导向，我们首先建立情感字典。主要基于 *Hownet* 基础情感字典、互联网网络情感字典表情符号情感字典、程度副词情感字典和否定词情感字典 5 类。在此基础上进行相关的情感计算与分析，得到各文本的情感倾向，在此基础上利用搜索算法对各个负面性评论进行回复。

2.4 问题四的分析

问题四研究的是针对舆情的处理等级的划分方法，好的等级划分方法可以将等级较高的舆情进行及时的干预，并且避免对等级较低的舆情进行无必要的干预，节约了人力、物力、财力。问题四属于层次分析问题，对于此类问题可以利用层次分析将舆情传播时间、规模及网民情感倾向作为判定的因素进行综合分析。对于本题，在干预措施已经完备的情况下，由于可以影响划分等级的因素较多，在忽略其他条件下，以舆情传播时间、规模及网民情感倾向为判定标准，进行综合分析。问题四所求为一种划分方法，如何判定某一个舆情的等级成为了本题关键。由于以上原因，我们可以首先建立层次分析模型，以舆情传播时间、规模及网民情感倾向为判定标准，在对其基本标准进行细化，同样是使用层次分析法确定其细化标准对总评分的权值，通过客观标准分数的确定计算等级，得出最终的等级划分方法。

三、 模型假设

1. 假设附件 1 中抓取了部分媒体或网民评论的数据真实可靠
2. 假设舆情的发展仅受网民和政府干预的影响，不受其它外界因素的控制与影响。
3. 假设每条评论都能真实的反映网民的内心想法，不存在评论与内心想法不一致的情况。
4. 假设为了解决问题三中的情感干预问题，本文假设已经提供了充足的具有正负情感的语料库。
5. 假设当网民看到与自己情感倾向相反的干预文本时，其情感会向干预文本的导向发生转变。

四、 符号说明

符号	意义
TF_w	词频
$count_w$	词条 w 在某一类主题里出现的次数
$Count$	该主题下所有词条总数
$document_w$	包含词条 w 的文档数
$Similarity$	余弦相似度
W_i	情感词
S_w	情感值

五、 模型一建立与求解

5.1 问题一数据预处理

1. 对附件一舆情数据按 7：3 的比例分为训练集和测试集。
2. 由于本文主要是通过对关键词的分析来判别文本内容，因此为提高文本识别度，需进行分词处理。具体操作是在通过 Python 的第三方库 jieba 分词器，将汉字文本进行二次分词，使得完整语句变为多个中文词组 (如图 1)。

近日 哪吒 汽车 旗下 哪吒 的 两款 新车型 上市 补贴 后 的 售价 分别 为 万元 与 万元 在 外观 与 内饰 方面 两款 新车 均 与 现款 车型 保持 了 一致 仅 在 配置 方面 进行 了 小幅 调整 具体 来看 新增 车型 的 车身 尺寸 为 轴距 为 车头 处 采用 了 封闭式 的 黑色 前格 栅 两侧 大灯 造型 上扬 车侧 底部 有着 银色 的 饰条 设计 车尾 则 采用 了 横向 贯穿 的 镀铬 饰 条车 内 部分 横向 贯穿 的 银色 饰条 为 内饰 增添 了 一定 的 层次感 英寸 的 中控 屏幕 应用 功能 较为 丰富 后排 座椅 支持 折叠 放到 从而 获得 更 多 空间 配置 方面 两款 新增 车型 较 在 售 的 车型 减少 了 电 折叠 后视镜 中央 扶手 电动 座椅 以及 全 液晶 仪表盘 此外 车型 还 加入 了 抬头 显示 配置 动力 方面 两款 新增 车型 的 电机 功率 为 千瓦 最大 扭矩 为 牛米 此外 新增 车型 还 搭载 有 千瓦时 容量 的 电池组 能够 实现 千米 的 续航 里程

图 1

3. 需考虑剔除文本中的停止词来提高特征向量质量，即删除文本中一些代词、连词。例如：“是”、“不是”、“得”、“和”、“或”、“为”、“你” 等。具体实现方法可采用 *Python* 的第三方库 *Sklearn* 的机器学习库，使用其内置的特征处理方法即可。

5.2 模型一的建立

问题一要求我们建立争对特定主题的舆论筛选方法，即通过主题关键词得到相关的文本段落。为此，本文考虑通过核函数映射处理得到与线性支持向量机相似的非线性支持向量机，利用 LDA 聚类对文本进行初步分类，再通过 $TF-IDF$ 参数建立 SVM 分类器模型，具体流程图介绍见图 2。

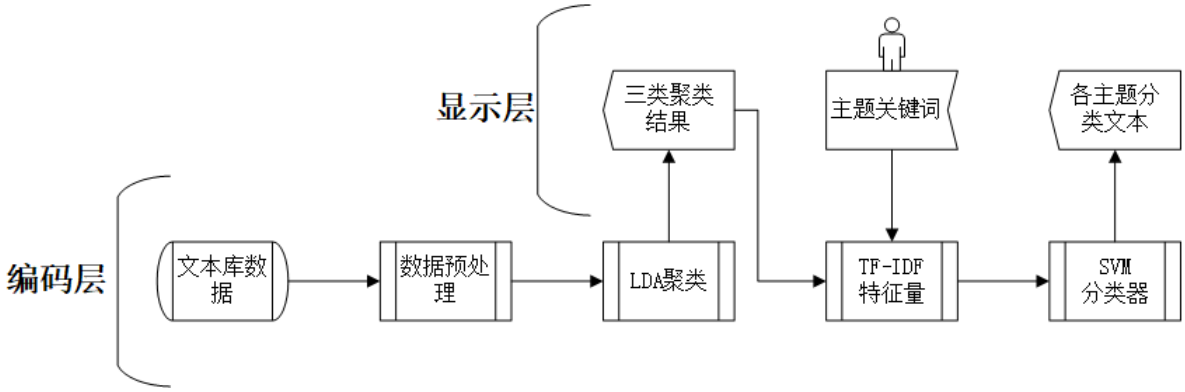

图 2　模型一流程图

在提取了网民评论的特征向量时，建立特征向量空间的训练集。由于训练集是我们根据题目所给的附件 1 随机选取得到的，可以先通过 LDA 聚类将训练集分成三类，再将这三类文本数据的特征集通过一个高效的分类器对其进行训练，从而实现对正负面两种舆论导向的分类鉴别。本文选择 SVM 作为分类模型，是因为其具有根据有限样本找到最优解的能力，能够避免神经网络中的局部极值问题而得到全局最优点和高维特征处理能力。具体模型介绍如下：

5.2.1 LDA 聚类

为提高 SVM 分类的精度，在数据预处理后，本文先通过 LDA 聚类进行初步分类，得到三组(本题人为定义) 训练数据，本分类依据是样本间相似程度，同一组内样本性质彼此接近，而组间的相似距离则较大。具体做法如下：

对于每篇文档 ω：

(1) 选择服从 $Dirichlet$ 分布的 θ 和服从 $Poisson$ 的参数 N 。

(2) 对于 N 个词的每篇文档中的每个词 ω_n 进行如下处理：

a. 选择服从多项式分布的主题 z_n。

b. 从主题 $_n$ 中以 $p(\omega_n,z_n)$ 的概率选择一个服从多项式分布的词 ω_n。

则联合概率为：

$$p(\theta,|\alpha,\beta)=p(\theta,\alpha)\pi_{n=1}^{N}p(z_n,\theta)p(w_n|z_n,\beta) \tag{1}$$

给 $\alpha\beta$ 赋值后，不断重复上述过程，最终收敛到的结果就是 LDA 的输出结果。通过 LDA 聚类模型，可以通过训练将语料聚为 N 个不同的主题簇，同时对于每个簇输出相应的 M 个候选关

键词，如下式：

$$LDA = \{K_1, K_2, \ldots, K_n\} \tag{2}$$

其中 K 表示第 i 个主题簇对应的关键词集合：

$$K(\text{语料}) = \{\text{关键词 1}, \text{关键词 2}, \ldots, \text{关键词 m}\} \tag{3}$$

利用 LDA 对语料进行聚类，可以初步将规模庞大的语料库划分为几个规模稍小的语料簇，同时不同的语料簇拥有各自的关键词集合，该关键词集合即可作为该簇的主题关键词，供进一步的舆情信息筛选使用。而本文在之后的求解中发现人为设定分为三类时，得到的分类结果相对准确且分类组数较少，便于后期 SVM 分类器运算。

5.2.2 非线性支持向量机

构建 SVM 模型的基础是构建在特征空间内线性可分或非线性可分的训练集，而由于文本数据在原始特征空间中不可分，即无法对训练集进行平面和超平面划分，因此本文考虑通过一个映射空间使数据在新的特征空间内线性可分。由于需要用到核函数和线性支持向量机内容，在此对它们进行解释：

1、核函数映射

核技巧

由于被映射到高维的特征向量总是以成对内积的形式存在，即 $\phi(x_i)^T\phi(x_i)$ ，此时是先计算特征在空间 R_d 中的映射，在计算内积，其复杂度是 O'_d 。而当特征被映射到更高维的空间。当其接近无穷时，将变成非常大的存储和计算量。为了解决以上问题，便出现了核技巧的概念，核技巧旨在将特征映射和内积两步运算压缩成一步，使复杂度由 O'_d 下降为 O_d ，计算量大大降低。即，核技巧是构造一个核函数 $k(x_i, x_j)$：

$$k(x_i, x_j) = \phi(x_i)^T\phi(x_i) \tag{4}$$

对于某一映射：

$$\phi : x \to e^{(-x^2)}\left[1, \sqrt{\frac{2}{1}}x, \sqrt{\frac{2^2}{2!}}x, \ldots,\right]^T \tag{5}$$

其对应的核函数：

$$k(x_i, x_j) = e^{-(x_i - x_j)^2} \tag{6}$$

核函数选择

通过高维空间的核技巧映射，我们便可以高效地解决样本非线性可分问题。但同时又会出现一个新的问题：很难去确定应该具体向什么样的高维空间映射 (即选择怎样的核函数)。核函数选取的适合与否直接决定分类器的性能和效果。本文在比较了三类核函数的特点后最终决定选取 RBF 函数，函数表达式为 $exp(- \parallel (x_i - x_j)^2 \parallel /2\sigma^2)$。

2、线性支持向量机

支持向量机的线性分类的目标是希望在特征空间中找到一个划分超平面，将拥有不同标记的样本分开，并且该划分超平面距离各样本最远，即需要找到一组合适的参数 (a,b) ，使得：

$$\max_{\omega,b}\min_{i}\frac{2}{\|\omega\|}|\omega^T x_i+b| \tag{7}$$
$$s.t.\quad y_i h(x_i)=1,\quad i=1,2,\ldots,m$$

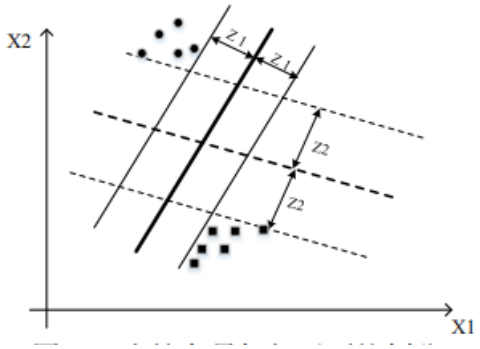

图 3　支持向量机超平面的划分

图 3 描述了线性向量机的分类 (图中虚线和实线)，图中虚线对应的分类器，其间隔为 $2z_2$ ，实线对应的分类器间隔为 $2z_1$ ，由于 $2z_2>2z_1$ ，故最终选择虚线对应的分类器。

3、非线性支持向量机构建

已知文本训练集在特征空间 R_d 中不是线性可分的，结合上文内容，本文构建非线性支持向量机，本文构建非线性支持向量机，使得通过映射 $\phi:R_d\to R_{\tilde{d}}$ 后，训练数据在新的特征空间 $R_{\tilde{d}}$ 内线性可分。则得到非线性支持向量机表达式如下：

$$\max_{\omega,b}\min_{i}\frac{2}{\|\omega\|}|\omega^T\phi(x_i)+b| \tag{8}$$
$$s.t.\quad y_i h(x_i)=1,\quad i=1,2,\ldots,m$$

5.2.3 $TF-IDF$ 特征量

通过构建非线性支持向量机，得到了 SVM 分类模型的基本框架。在此本文考虑引入 TF-IDF 特征量作为 SVM 分类模型的特征参数，使得模型分类结果更加准确。下面 TF-IDF 特征量的基本内涵。

TF 特征量

本文定义词组在一段文本中的重要程度与其在该文本的出现的次数成正比，与其在所有文本的出现的总次数成反比，因此可构建 TF_w 表达式如下：

$$TF_w = \frac{count_w}{Count} \tag{9}$$

TF_w 为词频，$count_w$ 为在某一类中词条 w 出现的次数，$Count$ 为该类中所有的词条数目。

IDF 特征量

而考虑到部分关键词在文本内出现频次较低，但是其本身所表达的意义重大。因此还需要引入 *IDF* 指标作为权重，则权重越大，对文本的影响能力越显著。具体构建函数表达书如下：

$$IDF = log(\frac{Document}{document_w + 1}) \tag{10}$$

这里 $Document$ 为语料库的文档总数。而 $document_w$ 为包含词条 w 的文档数。

综上，在模型求解时，可以引入 $TF-IDF$ 作为特征输入，式 (11) 是它的基本表达式。$TF-IDF$ 法则具有过滤高频词汇，突出重点词汇的特点。

$$TF - IDF = TF \times IDF \tag{11}$$

5.3 模型结果展示

LDA 主题聚类需要预先判定文本的主题类数。为了提高分析效率，本文选用附件 1 的前 5000 条数据作为本题的分析文本。首先对所有分析文本进行 *TFIDF* 特征提取，获取所有文本中的高词频词语的同时过滤掉通用词语，得到文本的初步关键词信息。其部分结果如下表所示。

表 1　*LDA* 聚类结果

排名	词语	$TF-IDF$ 特征值	排名	词语	$TF-IDF$ 特征值
1	新能源	2085.4	6	企业	1325.6
2	公司	1865.8	7	品牌	1111.5
3	中国	1746.6	8	市场	1089.5
4	产品	1535.9	9	车型	958.3
5	托运	1358.3	10	技术	895.6

根据上表数据，考虑到单条评论中可能同时含有多个关键词，本文选取 *LDA* 聚类的主题类数为 3，对分析文本进行聚类，得到结果如表 (2) 所示。分析发现上表中排名前三的高频词各为一

类，说明选取聚类数为 3 是合理的，分析数据可以大致分为三类。根据关键词，可以猜测第一类的主题与汽车公司的技术产业有关，第二类的主题与疫情期间的新能源汽车市场有关，第三类的主题与中国的新能源汽车品牌和车型有关。

表 2　*LDA* 聚类结果

第 i 类	关键词 1	关键词 2	关键词 3	关键词 4	关键词 5
第一类	汽车	公司	托运	产业	技术
第二类	汽车	新能源	疫情	市场	企业
第三类	汽车	中国	品牌	车型	新能源

为了能够更加直观的分析文本的词组，我们使用 *Python* 的第三方库 *WordCloud* 来进行词云展示，如下图所示为分析文本的关键词词云：

图 4　词云图

根据 *LDA* 主题聚类所给的分析数据的分类情况，得到一个长度为 5000 的分类结果向量，表示这 5000 条分析文本分别属于哪个类。但考虑到对于新加入的文本，单纯地依靠向量距离来进行分类存在一定的误差。因此本文设计了基于支持向量机的主题分类模型，将上述向量和各段文字所包含的词语的 *TFIDF* 特征值所转化的向量一同送入 *SVM* 模型中进行训练，得到了适用于本题的支持向量机模型。再将各段文字的 *TFIDF* 特征向量送入模型中进行验证，得到该 *SVM* 模型的正确率可以达到 97.2%。将特征向量进行缩减，保留关键词“新能源”和“公司”的特征值，送入 *SVM* 模型中进行分类，得到各个特征向量的散点图和 *SVM* 的分类依据图，如下图 (5) 所示。

为了保证模型的泛化能力，本文选取附件 1 的第 5001-10000 条数据作为测试数据，首先整体根据 *LDA* 分类得到其类别，再分别送入上述 *SVM* 模型中进行分类。将两个分类结果进行对比，发现 *SVM* 模型正确率仍可达 90.6%。说明该 *SVM* 模型的泛华能力较强，可以运用于新加数据的主题筛选。

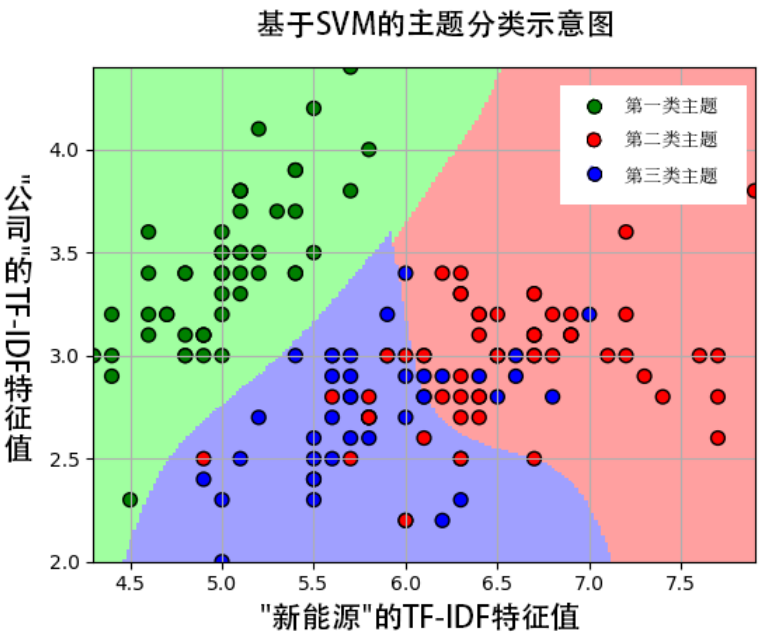

图 5　*SVM* 主题分类示意图

六、 问题二模型建立与求解

6.1 舆论抓取筛选模型建立

本文所建立的舆情数据抓取模型基于其爬虫技术。当在垂直领域获得舆情数据或有明确的舆情导向需求时，需要滤掉无用的数据并挖掘相关舆论信息。如图 (6) 所示，首先对要爬取数据界面，获取网页的源代码，考虑采用正则表达式提取信息。并将其保存为 *CSV* 格式文件。具体实例我们选取了微博作为对象，针对微博热门话题、微博热门评论和微博热门用户 3 部分进行舆情信息抓取，其中针对微博热门话题，设计抓取了用户 *ID*、发表时间、来源设备；针对微博热评论，设计抓取了评论时间、用户 *ID*、用户名、评论内容、用户年龄、用户性别、用户所在地。

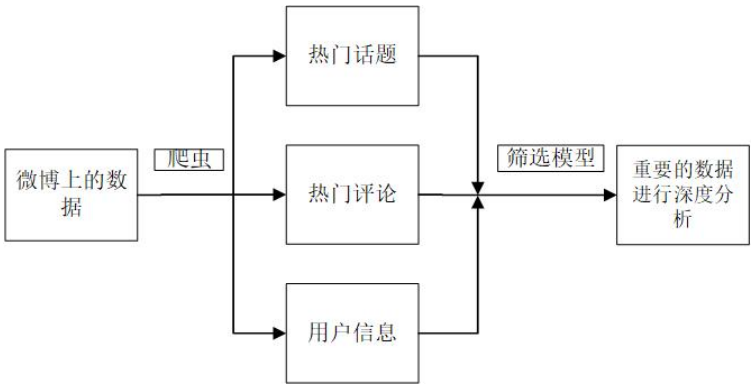

图 6　模型二流程图

6.1.1 微博热门话题抓取

针对热门话题，首先需要在手机端官方微博界面，查找相应话题，并在话题名前后加井字符号。例如，本文针对抗疫日记相关热门话题进行抓取。在搜索框搜索上述内容，然后按下 *F*12，打开浏览器审查元素，在网络界面捕获 *XHR* 格式的 *URL*，作为抓取数据的来源地址。

然后解析该源地址的 *json* 文件内容。数据储存在 *data* 下，*cards* 表示标签，一页有 10 个文件，每个文件中 *mblog* 为评论的详细话题内容，*mblog.user.id* 为用户的 *ID*，*mblog.user.name* 为用户的昵称，$mblog.reposts_count$ 为该话题转发的数量，$mblog.comments_count$ 为该话题评论的数量，$mblog.attitudes_count$ 为点赞数量 *mblog.text* 为话题的文本，$mblog.created_at$ 为发表时间，*mlbog.source* 为来源设备。考虑使用 *Python* 第三方库 *requests* 来进行数据抓取，将抓取后的文本保存至 *CSV* 格式文件。首先设置头部，模拟浏览器的基本内核，这里采用 *Chrome*/76.0.3809.100 内核，其次设置 *Cookie*，防止被微博系统检测到爬虫，而禁止抓取。使用 *Python* 第三方库 *requests* 来进行数据抓取，将抓取后的文本保存至 *CSV* 格式文件。

6.1.2 微博热门评论抓取

针对热门评论，首先需要在手机端官方微博界面，找到一个话题点击进去。同样按下 *F*12，打开浏览器审查元素，在网络界面捕获 *XHR* 格式的 *URL*，作为抓取数据的来源地址。同样地，解析该源地址的 *json* 文件内容。数据储存在 *data* 下，表示标签，一页有 10 个文件，每个文件中 *mblog* 为评论的详细话题内容，*id* 为用户的 *ID*，*mblog.user.name* 为用户的昵称，$mblog.reposts_count$ 为该话题转发的数量，$mblog.comments_count$ 为该话题评论的数量，$mblog.attitudes_count$ 为点赞数量 *mblog.text* 为话题的文本，$mblog.created_at$ 为发表时间，*mlbog.source* 为来源设备。最后设置头部和 *Cookie*，防止被微博系统检测到爬虫。使用 *Python* 第三方库 *requests* 来进行数据抓取，将抓取后的文本保存至 *CSV* 格式文件。

6.1.3 微博用户信息抓取

为了进一步针对用户的其它信息抓取，考虑设计了新的用户数据抓取算法。首先通过我发现用户数据网址格式为 $http://weibo.cn/*/info$，其中 $*$ 为每个用户的 *ID*，在前两个数据抓取算法中已经获取到了用户的 *ID*，只需遍历这些 *ID* 的信息页，来获取信息内容即可。首先设置头部和 *Cookie* 信息，防反爬虫。然后使用第三方库 *Request* 中的 *requests.get*() 函数来进行网页访问，以获取用户信息。由于这里的网页为静态网页，此处文本处理使用正则表达式对网页源代码分析即可，其中 $<divclass="c">$ 昵称:$(.*?)
$ 中 * 代表用户昵称，$
$ 性别:$(.*?)
$ 中 $*$ 代表用户性别，$
$ 地区:$(.*?)
$ 中 $*$ 代表用户地区，$
:(.*?)
$ 中 $*$ 代表用户生日，通过生日可进一步计算出年龄。

6.1.4 $TF-IDF$ 模型进行筛选

结合第一问建立的 $TF-IDF$ 模型，首先对于无监督模型中关键词的提取使用 $TF-IDF$ 模型：用于反映一个词对于某篇文档的重要性。考虑到点赞数，评论数每篇文章或评论都会有，可能会具有较高的词频，因此先对文章的常用词进行分析。

对于特定词的提取，如发表时间，评论人数，具体内容等，可以使用序列标注模型中的词性标注，即根据词性对词进行标注，如发表时间如 $2020-3$ 即为 $2020(())/m)3(())/m)$，只需要提取特定的词性组合，就可以识别出时间，比如 $2020-3$ 的词性组合为 m，m。对于微博等发表具有特定的时间标注，可以较好的识别出来。

考虑到文本本身的复杂性，如上下文等，但是往往也会有识别其它实体的需求，比如时间、点赞人数，具体内容等。如果是对所需内容进行识别的话，可以首先自建平台将序列进行初步分类，自建的序列标注平台需要大量的自主性的对特定所需内容进行标注。即输入特定的文本，会对文本进行标注，其中特定的文本会被用 BIO 规范标注。

达成标注要求后，标注完的文本可以作为训练样本，再利用 $bert$ 等框架对模型进行训练。完成训练后的模型即可对任意输入的语料标注出所需的特定内容。但是此种方法所需时间太大，需要对大量数据进行标注训练才能达到较好的效果。

6.2 问题二结果展示

以微博热门话题中“抗疫日记”为例，代入数据到舆情数据抓取模型中，对所有发表用户进行信息获取。针对该问题的我国因地区位置差异而对该问题的关注程度会有所不同，使用上述方法获取到用户位置信息后，我们按照颜色深浅绘制用户地区关注该热点话题程度的图像：

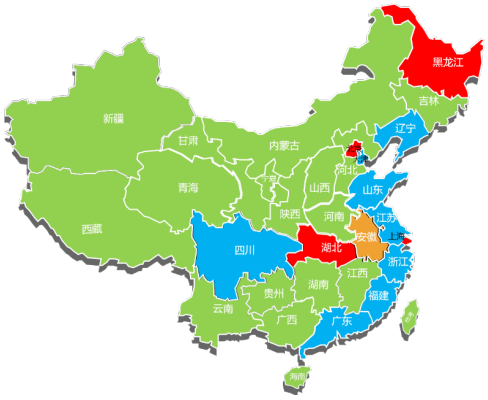

图 7　全国话题热度分布表

从图 (7) 中可看出，“抗疫日记”话题讨论中集中在我国东部和南部沿海的地区，该地区多以运输贸易为主且人流量也较大，疫情的出现可能导致其贸易的中断和经济发展受限，故当地对疫情的关注程度也较大，其中黑龙江因为境外的人员较多，故全省对其疫情的关注程度也较大。同样，我们对抓取到的性别信息进行统计，可得到下图：

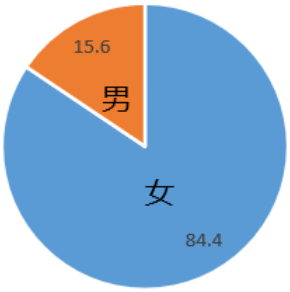

图 8　性别比例表

由下图 (8) 可看出，“抗疫日记”话题女士关注度远远超过男性。通过用户性别信息我们可挖掘到不同话题所受众的性别比例，并提取有效的分析。

我们还可以对 *Python* 爬虫后抓取到数据文件沿用所建立的舆情信息筛选模型，将捕获到的话题内容，通过舆情信息筛选模型进行文本分词处理。由话题词云图可看出“抗疫日记”话题重点舆情信息为” 确诊”、” 病例”、” 输入”、” 境外”、” 累计”。得到这些信息资料的意义是更有利于分析和发现疫情对全国地区和人民生活的影响，对地域的分析可知：境外流入，经济影响和感染人数是关注的重心点，这三方面决定该地区人民对疫情的关注程度，主要体现在湖北，东南沿海，黑龙江等地。另外男女之间对此类公共卫生事件的关注程度也是有很大差异，女生很明显比男生在平台上对该类事件的关注程度高很多，对于话题的搜索和评论点击的人数都远远超出男性，是网络该话题核心人群。

七、 问题三模型建立与求解

7.1 问题三模型的建立

如图 (9) 所示，*part*1 是对语料主题的处理，将获取的文本信息按照主题关键词进行处理，得到同一主题下的所有文本信息。*part*2 是对词语的情感分析，需要考虑建立词语情感库和相关情感分数计算准则。*part*3 则是从政府角度建立情感自主引导程序。

7.1.1 语料主题处理

本题的语料主题处理模型主要采用模型一的 *SVM* 模型，在此不再赘述。

7.1.2 词语情感分析

词语情感库

在判断舆情的情感导向时，一般来说，情感为人对客观事物是否满足自己的需要而产生的态度体验。其核心部分由一系列情感词和情感短语以及它们的情感极性和强度组成。然而，现有的情感词典并不适用于最新的情感分析。用户经常使用非正式的新词，如“好飒”，“666”等词汇。这

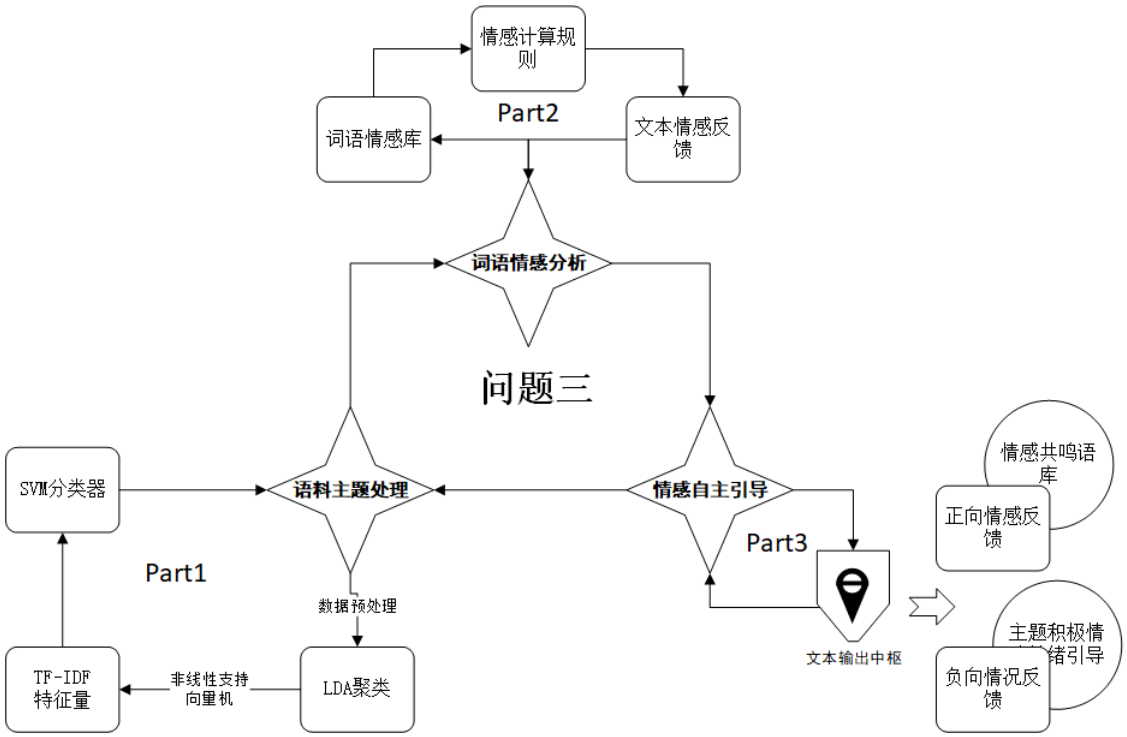

图 9　问题三流程图

些用于传达了丰富的情感信息，对情感分析尤为重要。因此我们首先针对目前主流的网络讨论平台建立特定的情感词典，主要可以分为正向性、中性、负向性三类。避免了人工检测和注释等方法的成本高，耗时长的弊端。我们所建立的情感成本字典主要分为 Hownet 基础情感字典、互联网网络情感字典表情符号情感字典、程度副词情感字典和否定词情感字典 5 种。

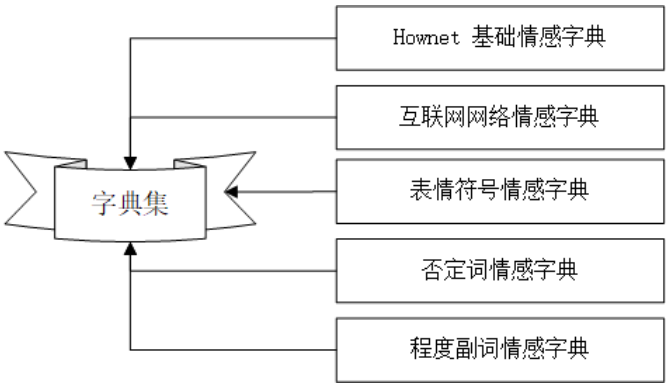

图 10　情感词典库

情感分数计算

先将文本分解成句子集合 S，即 $D = S_1, S_2, \ldots, S_n$。在计算得出每个 s_i 的情感值 $F(s_i)$ 后，对所有 s_i 情感值进行整合得到文本 D 的情感值 $F(S)$。如式 (12)、(13) 所示：

$$F(S_i) = \sum S_{w_i} \tag{12}$$

$$F(S) = \sum F(S_i) \tag{13}$$

其中，S_{w_i} 为句中情感词 W_i 的情感值。

若 $F(S) > 0$，则判定文本为正向情感；若 $F(S) < 0$，则判定文本为负向情感；若 $F(S) = 0$，则判定文本为中性情感。S_{w_i} 表示如下：

$$S_{w_i} = P_{w_i} - N_{w_i} \tag{14}$$

在公式 (14) 中，P_{w_i} 和 N_{w_i} 由公式 (15)，(16) 计算得出：

$$P_{w_i} = \frac{fp_{w_i}}{(fp_{w_i} + fn_{w_i})} \neq \frac{N_p}{(N_p + N_n)} \tag{15}$$

$$N_{w_i} = \frac{fn_{w_i}}{(fp_{w_i} + fn_{w_i})} \neq \frac{N_p}{(N_p + N_n)} \tag{16}$$

其中，N_p 表示情感词典中正向词汇的数量，N_n 为负向词汇的数量；fp_{w_i} 为情感词汇 W_i ，与正向情感词汇数量的比例 fn_{w_i}。与之相反。通过公式 (17) 计算情感值 S_{w_i} ，

若 $S_{w_i} > 0$，则 w_i 为正向情感词；若 $S_{w_i} < 0$，则 w_i 为负向情感词；若 $S_{w_i} = 0$，则 w_i 为中性情感词。

此外，为保证情感值的准确性，考虑加入程度副词为情感参数，公式如下：

$$S_{w_i} = (P_{w_i} - N_{w_i}) * (1 \pm \sigma) * N_e \tag{17}$$

其中 N_e 为否定系数，

若情感词 w_i 紧邻否定词 (不、没、非等)，情感将发生反转，故将否定系数 N_e 设置成-1，σ 为调节系数；若情感 w_i 紧邻“非常，极其”等程度副词时，则情感得分为 $S_{w_i} = (P_{w_i} - N_{w_i}) * (1 + \sigma) * N_e$；若情感 w_i 紧邻“稍微，一般”等程度副词时，则情感得分为 $S_{w_i} = (P_{w_i} - N_{w_i}) * (1 - \sigma) * N_e$。

7.1.3 情绪自主引导模型

对于政府对舆论的合理干预，本文考虑建立评论自主回复模型，通过 $Part1$ 的 SVM 主题分类和 $Part2$ 的情感分析，可以得到各主题的正向情感评论和负向情感评论。

对于正向情感评论，可以建立情感共鸣语库，语库内有包括“你说的很有道路、赞、不错不错”等积极的简短评论。当出现正向情感评论时，可以自动调用情感共鸣语库，回复一条积极性的评论。

而对于负向情感评论，本文考虑建立关键词搜索算法，记录负向情感评论的所有关键词 (关键词不包括有情感色彩的词)，并根据这些关键词搜索正向评论中的相关语句。找到同一主题下正向文本里与负向关键词相似度最高的语句，将其回复在该负向评论下。相似度计算公式如下：

$$(x, y) = \frac{\sum_{i=i}^{k}(x_i * y_i)}{\sqrt{\sum_{i=i}^{k}(x_i)^2} * \sqrt{\sum_{i=i}^{k}(y_i)^2}} \tag{18}$$

其中 k 为关键词 x 与关键词 y 经 SVM 模型输出的词向量维度，本文中 $k = 300$。通过计算该负向文本关键词与同一主题内多个正向文本关键词的相似度均值，将平均相似度最大的正向文本的原评论回复在该负向评论下方，即：

$$V^* = argmax_{V_i}(V_i, \tilde{V}) \tag{19}$$

其中 $\tilde{V}$ 为负向文本关键词词向量集合， 为第 i 个同主题正向文本的关键词向量集合，V^* 即选择的目标回复内容。

7.2 问题三模型求解

获取表 3 所示的爬虫数据，进行 $part1$ 主题分类和 $part2$ 情感分析，得到下图 (11) 所示结果，此事件整体虽以正向情感为主，也有负向情感多次显现。

表 3　爬虫数据

舆论事件名称	评论获取事件跨度	语料数量 (条)
福州赵宇案	2019 年 1 月 1 日-2019 年 3 月 10 日	83158
北大学生弒母案	2019 年 4 月 25 日-2019 年 5 月 15 日	58441
香港暴动事件	2019 年 7 月 1 日-2019 年 12 月 15 日	213556

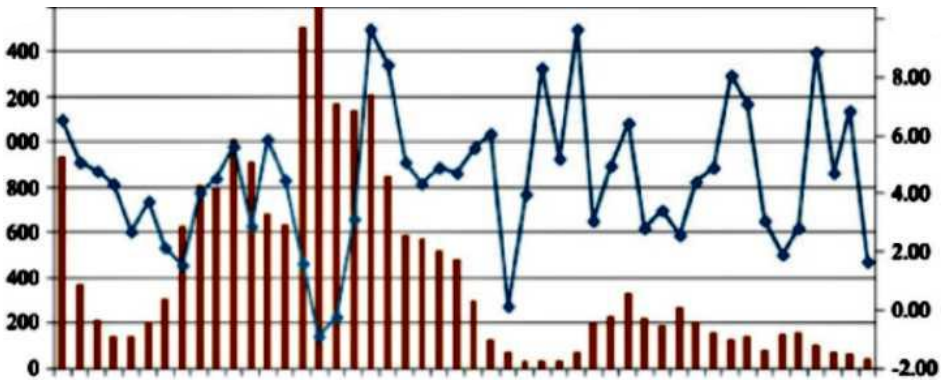

图 11　特定时期内某事件舆论网民平均情感倾向时序图

将数据进一步带入 part3，得到以下文本引导结果：

表 4　情绪引导文本展示

情绪状态	文本引导内容
福州赵宇案正向情绪	给你点赞哟！
北大学生弒母案负向情绪	我不赞同你的看法哦！
香港暴动事件负向情绪	国人要热爱香港！保护香港！

八、 问题四模型建立与求解

8.1 模型四的建立

对于特定的舆论事件对政府和社会的影响危险等级的划分对控制舆论的发展又重要的意义，本题建立层次分析模型对舆论影响程度基本影响指标进行权重计算，再考虑利用 Dephi 法对每个指标给出客观合理的评分，计算该舆论事件的总得分，按照评分体系建立等级划分。其基本流程图如下：

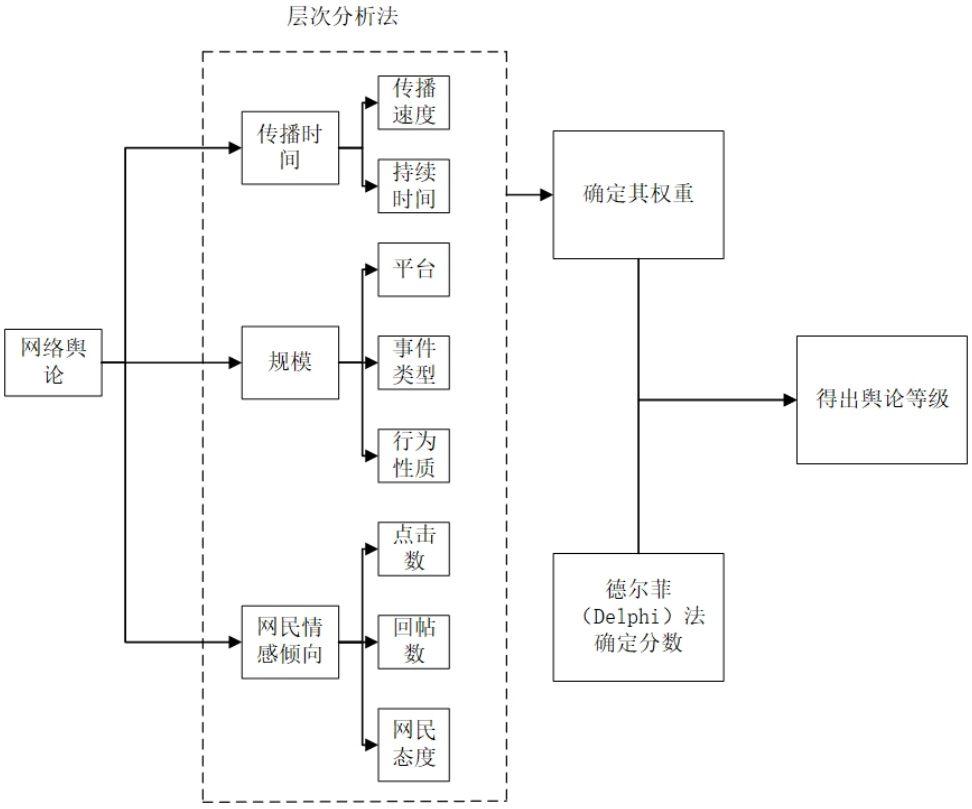

图 12　模型四流程图

8.1.1 层次分析法

通过对实际情况的分析，可以将舆论传播时间、规模及网民情感倾向进行细化分类，舆论传播时间细化为传播速度和持续时间；规模细化为平台的种类，事件的类别和行为的性质；网民情感倾向可以细化到点击数，回帖数和网民的态度比例。平台的种类包括微博，微信，今日头条等各大新闻传播媒介；事件的类别包括娱乐，教育，政治等话题；行为的性质包括道德层面，法律层面等。如上的评价指标的权重具有一定的主观性，考虑使用层次分析法来确定。

此下为层次分析法的具体做法：

step1: 确定影响舆论的若干因素；并将其重要程度数量化，本题考虑 1-8 的数量化等级

step2: 构造判别矩阵

$$A = (a_{ij})_{n*n} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} \tag{20}$$

step3: 定义一致性指标

$$CI = \frac{\lambda - n}{n-1} \tag{21}$$

step4：进行一致性检验

$$CR = \frac{CI}{RI} \tag{22}$$

当 CR 小于 0.1 时，可以认为其矩阵满足一致性要求

先对舆论传播时间、规模及网民情感倾向这三个一级评价指标进行层次分析法，得到其三者的基本权值矩阵 (A,B,C) 且 $A + B + C = 1$

若可以对每个细化指标进行分数量化处理，舆论传播时间细化为传播速度和持续时间且同样是使用层次分析法得出其细化评价指标权重为 (a_1, a_2)，规模细化为平台的种类，事件的类别和行为的性质的指标权重为 (b_1, b_2, b_3)；网民情感倾向可以细化到点击数，回帖数和网民的态度比例指标权重为 (c_1, c_2, c_3)。得到每个细化的指标权重计算可知每个评价指标对于最终分数的权重，即 8 个指标对整体的权重为 $\omega = (\omega_1, \omega_2, \dots, \omega_8)$，计算的方法为：

$$(\omega_1, \omega_2) = A(a_1, a_2) \tag{23}$$

$$(\omega_3, \omega_4, \omega_5) = B(b_1, b_2, b_3) \tag{24}$$

$$(\omega_6, \omega_7, \omega_8) = C(c_1, c_2, c_3) \tag{25}$$

每个舆论进行分析后都会有一个评分，该评分代表着该舆论对目前政府或企业的影响程度，分数越高，其造成的社会影响也会越大且处理起来的困难也会越大。对于分数的给出，要求是客观且具有预判思想的合理有效分数。对此，我们考虑使用 *Dephi* 法，此方法的评分需要专业的人员对过去事件发展状况趋势的判断经验，然后对舆论现状进行整体的评价，给出客观统一的分数，且专家的意见进行反复修改最终达到统一的状态，这可以有效避免每个专家因为思考的角度不同，基于之前的经验不同导致的判断不同所给出的评分不同。

8.1.2 *Dephi* 法

Dephi 法的具体过程为调查部门利用前述的模型找出一个热门话题的相关信息，将这些相关信息作为判断材料给 10 个在舆论等级判断方面研究的资深专家，让其就我们上述的 8 个细化的角度进行评分并给出自己的判断依据，规定分数从 0 分到 100 分，0 分代表该事件对政府或企业毫无影响，100 分代表该事件对政府或企业的影响很大，基本到达无法干预且对政府或企业有极大舆论破坏的程度。如果这些专家之间的评分不一致，则需要对评分进行修正。考虑请专家重审争论，对上下四分点外的对立意见作出评价，给出自己新的评价分数，以此反复直到其说有的专家的意见完全一致，该各项分数就是最终得到的各项分数记为:$G=(g_1,g_2,\ldots,g_8)$.

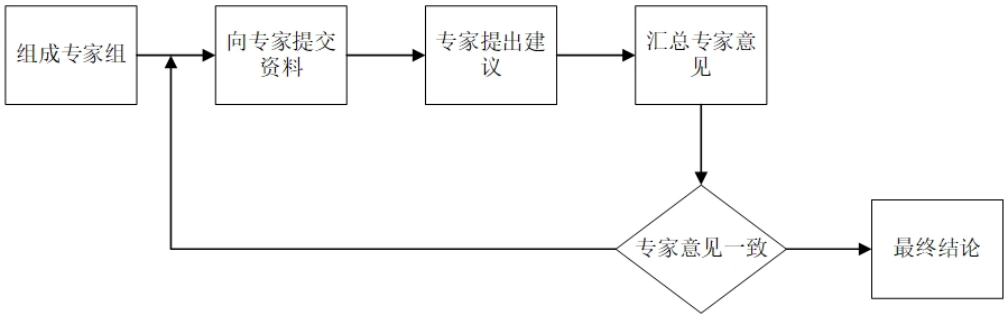

图 13　*Dephi* 具体流程

综上所述，使用层次分析法得出各细化指标的权重时和利用 *Dephi* 法得出各项指标在目前舆论前的具体分数，该舆论事件的总评分 T 为：

$$T=\omega G^T \tag{26}$$

用本题所建模型的评价体系可知，如果当所有的细化评价指标的所有分数值都为 100 时，即该舆论事件的总评分就为 100，即 $g_1=g_2=\cdots=g_8=100\Rightarrow T_{max}=100$ ，即该舆论事件已经达到了严重危险的极端程度，同样此为舆论分数的极限值，故对从 0 分到 100 分进行舆论等级划分，考虑对整体的分数进行四等分，每等分得出的分数区间代表着一种相同的舆论危险程度，分别为较小，一般，危险，严重危险的危险等级且此危险等级均以 25 分为等分分数跨度。

表 5　等级划分表

分数区间	[0,25]	[25,50]	[50,75]	[75,100]
危险等级	较小	一般	危险	严重危险

8.2 问题四模型求解

可知特征值法误差最小，所以将表中数据带入上述模型用 *MATALAB* 计算可知表 (6) 中的一致性比例 $CR=0.0516$ ，$\lambda_{max}=3.0536$ 。可以知道 $CR=0.0516<0.1$ ，表明本文的判断矩阵满

表 6　网络舆情判断矩阵与权重表

网络舆情	传播时间	传播规模	网民情感倾向	ω_i
传播时间	1	0.5	0.5	0.1958
传播规模	2	1	2	0.4934
网民情感倾向	2	0.5	1	0.3108

足一致性，并且得到了传播时间、传播规模和网民情感倾向的权重值。同理可得其他判断矩阵的权重值为：$\omega = (\omega_1, \omega_2, \ldots, \omega_8)$ 结果为：

表 7　所有细化指标的权重

网络舆情	传播速度	持续时间	平台的种类	事件类别	行为性质	点击数	回帖数	网民态度
权重	0.048	0.147	0.3084	0.1177	0.0673	0.1332	0.1332	0.0452

利用层次分析法得出权重后，利用上述的评分体系方法可以计算出该舆论的总体对政府和社会的影响分数，依据给定的分数评分等级区间，合理且高效对其进行相应的措施。

九、 模型评价与推广

9.1 模型优点

1. 模型一所使用的 SVM 支持向量机具有较好的适应性强和全局优化的优点。
2. 模型三在建立情感字典时充分考虑了互联网环节、副词、否定词对词语情感评判的影响，使词语情感评判更加客观。
3. 模型四通过层次分析法将情感分析这一定性分析转化为数值上的定量分析，使模型得到了简化，方法易于使用。

9.2 模型缺点

对于该空间中的每个高维空间映射 F，如何确定 F 也是一个核函数，目前尚无合适的方法，因此对于一般问题，SVM 只是化解了高维空间复杂性的难题进入内核功能困难。即使确定了内核函数，在求解问题分类时，也需要对求解函数进行二次编程，而这则需要大量存储空间。

9.3 模型推广

本文在情感分析过程中，发现情感词典词典是最重要的资源。可以通常对结果和相应的分析产生决定性的影响。但是很难构建一个适合所有领域的通用情绪字典，因为情感词通常只适用于它所适用的领域。因此可以针对不同领域进行情感词典的推广，不仅仅是常见的情感词，同样也可以是属于该专业的情感词汇，这样的处理可以扩充其情感词汇库。这样当在不同的情况下使用时，修改模型的舆情情感词，可以使情感词可以有相反的表达，从而适应不同的话题领域。

参考文献

[1] 彭晓平. 基于 SVM 的政务舆情分析研究 [C]. 中国城市规划学会城市规划新技术应用学术委员会. 智慧规划 • 生态人居 • 品质空间——2019 年中国城市规划信息化年会论文集. 中国城市规划学会城市规划新技术应用学术委员会:《规划师》杂志社,2019:403-408.

[2] 郭江民, 王一然, 祝彬, 关晓红. 基于支持向量机的网络舆情预测 [J]. 网络新媒体技术,2017,6(05):29-35.

[3] 汪涛, 樊孝忠. 主题爬虫的设计与实现 [J]. 计算机应用, 2004, 024(0z1):270-272.

附录 A 代码

第一题

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from gensim import corpora, mods
import jieba.posseg as jp
from wordcloud import WordCloud as wc
import matplotlib as mpl
from sklearn import svm
from matplotlib import colors

# 环境准备
plt.style.use('seaborn')
sns.set(font_scale=2)
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False
stopwords = ('根据', '图片', 'Moment', '是', '不', '也', '到', '将', '有', '都', 'G', '更', '就', '还',
    '会', '人', '要', '没有',
'可能', '需要', '就是', '包括', '已经', '还有', '甚至', '作为', '非常', '进行') # 停词


# 数据读取
def loadcsv():
f = open(r'附件1.csv', encoding='utf-8"')
df = pd.read_csv(f, delimiter='\t', sep='\t', error_bad_lines=False)
df['char_length'] = df['正文'].astype(str).apply(len)
return df


# jieba分词
def fenci(t):
print('开始分词')
flags = ('n', 'nr', 'ns', 'nt', 'eng', 'v', 'd') # 词性
# 分词
words_ls = []
onesen = ""
for text in t:
words = [w.word for w in jp.cut(text) if w.flag in flags and w.word not in stopwords]
onesen += " ".join(words) + " "
words_ls.append(words)
print('分词完毕')
return words_ls, onesen
```

```
# lda 模型
def lda(words_ls):
# 构造词典
dictionary = corpora.Dictionary(words_ls)
print('词典构建完毕')
# 对每段文本依据词典生成向量
cor = [dictionary.doc2bow(words) for words in words_ls]
print('lda模型开始训练')
lda = mods.ldamod.LdaMod(cor=cor, id2word=dictionary, num_topics=3)
# 打印所有主题, 每个主题显示 5 个词
for topic in lda.print_topics(num_words=10):
print(topic)
# 主题推断
print(lda.inference(cor))
print('lda模型已完成')


# 生成词云
def cloud(sen):
print('开始制作词云')
mywc = wc(font_path="C:/Windows/Fonts/simkai.ttf", background_color="white", max_font_s=130,
max_words=200, stopwords=set(stopwords)).generate(sen)
image = mywc.to_image()
mywc.to_file('ciyun.png')
image.show()
print('词云制作结束')


# 支持向量机主题分类
def svm(x, y):
clf = svm.SVC(C=0.5, kernel='rbf', gamma=1, decision_function_shape='ovr')
clf.fit(x, y.ravel())
x1_min, x1_max = x[:, 0].min(), x[:, 0].max()
x2_min, x2_max = x[:, 1].min(), x[:, 1].max()
x1, x2 = np.mg[x1_min:x1_max:200j, x2_min:x2_max:200j]
g_t = np.stack((x1.flat, x2.flat), axis=1)
g_h = clf.predict(g_t)
g_h = g_h.reshape(x1.shape)

cm_l = mpl.colors.ListedColormap(['#A0FFA0', '#FFA0A0', '#A0A0FF'])
cm_d = mpl.colors.ListedColormap(['g', 'b', 'r'])

plt.pcolormesh(x1, x2, g_h, cmap=cm_l)
plt.scatter(x[:, 0], x[:, 1], c=np.squeeze(y), edgecolor='k', s=50, cmap=cm_d) # 样本点
plt.xlabel('关键词1的词频')
```

```
plt.ylabel('关键词2的词频')
plt.xlim(x1_min, x1_max)
plt.ylim(x2_min, x2_max)
plt.title('基于svm的主题分类示意图')
plt.grid()
plt.show()


df = loadcsv()
t = df['正文'].tolist()[0:5000]
wlist, sen = fenci(t)
lda(wlist)
cloud(sen)
svm(x,y)


第二题
import requests
import pandas as pd
import time


# 爬虫
def crawler():
re = []
header = {
'Content-Type': 'application/json; charset=utf-8',
'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
    Chrome/76.0.3809.100 Safari/537.36'
}
Cookie = {'Cookie': ''}
for ii in range(19):
# 抗疫日记
url_base = 'https://m.weibo.cn/api/container/getIndex?containerid=100103type%3D60%26q%3' \
'D%23%E6%8A%97%E7%96%AB%E6%97%A5%E8%AE%B0%23%26t%3D0&page_type=searchall'
url = url_base + str(ii + 1)
html = requests.get(url, headers=header, cookies=Cookie)
try:
for jj in range(len(html.json()['data']['cards'])):
if html.json()['data']['cards'][jj]['mblog']['isLongText'] == False:
text=html.json()['data']['cards'][jj]['mblog']['text']
else:
text=html.json()['data']['cards'][jj]['mblog']['longText']['longTextContent']
fin=""
flag=False
for i in text:
if i=='<':
```

```
flag=True
if i=='>':
flag=False
if flag:
continue
fin+=i
data1 = [(html.json()['data']['cards'][jj]['mblog']['user']['id'], # 用户 ID
html.json()['data']['cards'][jj]['mblog']['user']['screen_name'], # 用户名
html.json()['data']['cards'][jj]['mblog']['reposts_count'], # 转发数
html.json()['data']['cards'][jj]['mblog']['comments_count'], # 评论数
html.json()['data']['cards'][jj]['mblog']['attitudes_count'], # 点赞数量
fin,
html.json()['data']['cards'][jj]['mblog']['cd_at'], # 发表时间
html.json()['data']['cards'][jj]['mblog']['source'])] # 来源设备
data2 = pd.DataFrame(data1)
data2.to_csv('weibo_content-yq.csv', header=False, index=False, mode='a+',encoding='GBK')
re.extend(data1)
except:
print("抓取失败")
print('page ' + str(ii + 1) + ' has done')
time.sleep(3)
return re


comments = crawler()
```

第三题

```
import numpy as np,tensorflow as tf ,cPickle, random
from tensorflow.python.ops import tensor_array_ops, control_flow_ops
from dl import Gen_D_l, Dis_dl
from ge import Ge
from dis import Dis
from r import R
from tar_lstm import TAR_LSTM

class Ge(object):
def __init__(self, n_e, b_s, e_d, hid_d,
seq_len, s_tok,
l_rate=0.01, re_g=0.95):
self.n_e = n_e
self.b_s = b_s
```

```
self.e_d = e_d
self.hid_d = hid_d
self.seq_len = seq_len
self.s_tok = tf.constant([s_tok] * self.b_s, dtype=tf.int32)
self.l_rate = tf.Variable(float(l_rate), tra=False)
self.re_g = re_g
self.g_pa = []
self.d_pa = []
self.t = 1.0
self.grad_clip = 5.0

self.exp_re = tf.Variable(tf.zeros([self.seq_len]))

with tf.variable_scope('ge'):
self.g_eeddings = tf.Variable(self.init_matrix([self.n_e, self.e_d]))
self.g_pa.append(self.g_eeddings)
self.g_re = self.c_re(self.g_pa)
self.g_o = self.c_o(self.g_pa)

self.x = tf.placeholder(tf.int32, shape=[self.b_s,
self.seq_len])
self.res = tf.placeholder(tf.float32, shape=[self.b_s,
self.seq_len])

with tf.device("/cpu:0"):
self.pr_x = tf.transpose(tf.nn.eedding_lookup(self.g_eeddings, self.x),
perm=[1, 0, 2])
self.h0 = tf.zeros([self.b_s, self.hid_d])
self.h0 = tf.stack([self.h0, self.h0])

gen_o = tensor_array_ops.TensorArray(dtype=tf.float32, s=self.seq_len,
d_s=False, infer_shape=True)
gen_x = tensor_array_ops.TensorArray(dtype=tf.int32, s=self.seq_len,
d_s=False, infer_shape=True)

def _g_re(i, x_t, h_tm1, gen_o, gen_x):
h_t = self.g_re(x_t, h_tm1)
o_t = self.g_o(h_t)
log_prob = tf.log(tf.nn.softmax(o_t))
n_tok = tf.cast(tf.reshape(tf.multinomial(log_prob, 1), [self.b_s]), tf.int32)
x_tp1 = tf.nn.eedding_lookup(self.g_eeddings, n_tok) # b x e_d
gen_o = gen_o.write(i, tf.reduce_sum(tf.multiply(tf.one_hot(n_tok, self.n_e, 1.0, 0.0),
tf.nn.softmax(o_t)), 1))
gen_x = gen_x.write(i, n_tok)
return i + 1, x_tp1, h_t, gen_o, gen_x

_, _, _, self.gen_o, self.gen_x = control_flow_ops.while_loop(
```

```
cond=lambda i, _1, _2, _3, _4: i < self.seq_len,
body=_g_re,
loop_vars=(tf.constant(0, dtype=tf.int32),
tf.nn.eedding_lookup(self.g_eeddings, self.s_tok), self.h0, gen_o, gen_x))

self.gen_x = self.gen_x.stack()
self.gen_x = tf.transpose(self.gen_x, perm=[1, 0])

g_pre = tensor_array_ops.TensorArray(
dtype=tf.float32, s=self.seq_len,
d_s=False, infer_shape=True)

ta_e_x = tensor_array_ops.TensorArray(
dtype=tf.float32, s=self.seq_len)
ta_e_x = ta_e_x.unstack(self.pr_x)

def _pre_re(i, x_t, h_tm1, g_pre):
h_t = self.g_re(x_t, h_tm1)
o_t = self.g_o(h_t)
g_pre = g_pre.write(i, tf.nn.softmax(o_t))
x_tp1 = ta_e_x.read(i)
return i + 1, x_tp1, h_t, g_pre

_, _, _, self.g_pre = control_flow_ops.while_loop(
cond=lambda i, _1, _2, _3: i < self.seq_len,
body=_pre_re,
loop_vars=(tf.constant(0, dtype=tf.int32),
tf.nn.eedding_lookup(self.g_eeddings, self.s_tok),
self.h0, g_pre))

self.g_pre = tf.transpose(self.g_pre.stack(),
perm=[1, 0, 2])
self.pre_l = -tf.reduce_sum(
tf.one_hot(tf.to_int32(tf.reshape(self.x, [-1])), self.n_e, 1.0, 0.0) * tf.log(
tf.clip_by_value(tf.reshape(self.g_pre, [-1, self.n_e]), 1e-20, 1.0)
)
) / (self.seq_len * self.b_s)

pre_opt = self.g_optimizer(self.l_rate)

self.pre_grad, _ = tf.clip_by_global_norm(tf.gradients(self.pre_l, self.g_pa), self.grad_clip)
self.pre_updates = pre_opt.apply_gradients(zip(self.pre_grad, self.g_pa))



self.g_l = -tf.reduce_sum(
tf.reduce_sum(
tf.one_hot(tf.to_int32(tf.reshape(self.x, [-1])), self.n_e, 1.0, 0.0) * tf.log(
```

```
tf.clip_by_value(tf.reshape(self.g_pre, [-1, self.n_e]), 1e-20, 1.0)
), 1) * tf.reshape(self.res, [-1])
)

g_opt = self.g_optimizer(self.l_rate)

self.g_grad, _ = tf.clip_by_global_norm(tf.gradients(self.g_l, self.g_pa), self.grad_clip)
self.g_updates = g_opt.apply_gradients(zip(self.g_grad, self.g_pa))

def ge(self, sess):
os = sess.run(self.gen_x)
return os

def pre_step(self, sess, x):
os = sess.run([self.pre_updates, self.pre_l], f_dict={self.x: x})
return os

def init_matrix(self, shape):
return tf.random_normal(shape, stddev=0.1)

def init_vector(self, shape):
return tf.zeros(shape)

def c_re(self, pa):
self.Wi = tf.Variable(self.init_matrix([self.e_d, self.hid_d]))
self.Ui = tf.Variable(self.init_matrix([self.hid_d, self.hid_d]))
self.bi = tf.Variable(self.init_matrix([self.hid_d]))

self.Wf = tf.Variable(self.init_matrix([self.e_d, self.hid_d]))
self.Uf = tf.Variable(self.init_matrix([self.hid_d, self.hid_d]))
self.bf = tf.Variable(self.init_matrix([self.hid_d]))

self.Wog = tf.Variable(self.init_matrix([self.e_d, self.hid_d]))
self.Uog = tf.Variable(self.init_matrix([self.hid_d, self.hid_d]))
self.bog = tf.Variable(self.init_matrix([self.hid_d]))

self.Wc = tf.Variable(self.init_matrix([self.e_d, self.hid_d]))
self.Uc = tf.Variable(self.init_matrix([self.hid_d, self.hid_d]))
self.bc = tf.Variable(self.init_matrix([self.hid_d]))
pa.extend([
self.Wi, self.Ui, self.bi,
self.Wf, self.Uf, self.bf,
self.Wog, self.Uog, self.bog,
self.Wc, self.Uc, self.bc])

def unit(x, hid_memory_tm1):
pre_hid_sta, c_prev = tf.unstack(hid_memory_tm1)
```

```
i = tf.sigmoid(
tf.matmul(x, self.Wi) +
tf.matmul(pre_hid_sta, self.Ui) + self.bi
)

f = tf.sigmoid(
tf.matmul(x, self.Wf) +
tf.matmul(pre_hid_sta, self.Uf) + self.bf
)

o = tf.sigmoid(
tf.matmul(x, self.Wog) +
tf.matmul(pre_hid_sta, self.Uog) + self.bog
)

c_ = tf.nn.tanh(
tf.matmul(x, self.Wc) +
tf.matmul(pre_hid_sta, self.Uc) + self.bc
)

c = f * c_prev + i * c_

current_hid_sta = o * tf.nn.tanh(c)

return tf.stack([current_hid_sta, c])

return unit

def c_o(self, pa):
self.Wo = tf.Variable(self.init_matrix([self.hid_d, self.n_e]))
self.bo = tf.Variable(self.init_matrix([self.n_e]))
pa.extend([self.Wo, self.bo])

def unit(hid_memory_tuple):
hid_sta, c_prev = tf.unstack(hid_memory_tuple)
log = tf.matmul(hid_sta, self.Wo) + self.bo
return log

return unit

def g_optimizer(self, *args, **kwargs):
return tf.train.AdamOptimizer(*args, **kwargs)



E_D = 32
HID_D = 32
```

```
SEQ_LEN = 20
S_TOK = 0
PRE_W_N = 120
SEED = 88
B_S = 64

dis_eedding_d = 64
dis_f_ss = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20]
dis_n_fs = [100, 200, 200, 200, 200, 100, 100, 100, 100, 100, 160, 160]
dis_do_keep_prob = 0.75
dis_l2_reg_lambda = 0.2
dis_b_s = 64

TOTAL_B = 200
ged_n = 10000



def ge_s(sess, tra_mod, b_s, ged_n, o_file):
ged_s = []
for _ in range(int(ged_n / b_s)):
ged_s.extend(tra_mod.ge(sess))

def tar_l(sess, tar_lstm, d_l):
nll = []
d_l.re_po()

for it in xrange(d_l.n_b):
b = d_l.n_b()
g_l = sess.run(tar_lstm.pre_l, {tar_lstm.x: b})
nll.append(g_l)

return np.mean(nll)



def pre_train_w(sess, tra_mod, d_l):
sup_g_les = []
d_l.re_po()

for it in xrange(d_l.n_b):
b = d_l.n_b()
_, g_l = tra_mod.pre_step(sess, b)
sup_g_les.append(g_l)

return np.mean(sup_g_les)



def main():
```

```
random.seed(SEED)
np.random.seed(SEED)
assert S_TOK == 0

gen_d_l = Gen_D_l(B_S)
like_d_l = Gen_D_l(B_S)
vocab_s = 5000
dis_d_l = Dis_dl(B_S)

ge = Ge(vocab_s, B_S, E_D, HID_D, SEQ_LEN, S_TOK)
tar_pa = cPickle.load(open('s/tar_pa.pkl'))
tar_lstm = TAR_LSTM(vocab_s, B_S, E_D, HID_D, SEQ_LEN, S_TOK,
tar_pa)
dis = Dis(seq_len=20, n_classes=2, vocab_s=vocab_s,
eedding_s=dis_eedding_d,
f_ss=dis_f_ss, n_fs=dis_n_fs,
l2_reg_lambda=dis_l2_reg_lambda)

con = tf.ConProto()
con.gpu_o.allow_growth = True
sess = tf.Session(con=con)
sess.run(tf.global_va_initializer())

ge_s(sess, tar_lstm, B_S, ged_n, positive_file)
gen_d_l.c_bes(positive_file)
for w in xrange(PRE_W_N):
l = pre_train_w(sess, ge, gen_d_l)
if w % 5 == 0:
ge_s(sess, ge, B_S, ged_n, eval_file)
like_d_l.c_bes(eval_file)
t_l = tar_l(sess, tar_lstm, like_d_l)

for _ in range(50):
ge_s(sess, ge, B_S, ged_n, negative_file)
dis_d_l.load_train_d(positive_file, negative_file)
for _ in range(3):
dis_d_l.re_po()
for it in xrange(dis_d_l.n_b):
x_b, y_b = dis_d_l.n_b()
f = {
dis.i_x: x_b,
dis.i_y: y_b,
dis.do_keep_prob: dis_do_keep_prob
}
_ = sess.run(dis.train_op, f)

r = R(ge, 0.8)
```

```
for total_b in range(TOTAL_B):
for it in range(1):
s = ge.generate(sess)
res = r.get_re(sess, s, 16, dis)
f = {ge.x: s, ge.res: res}
_ = sess.run(ge.g_updates, f_dict=f)

if total_b % 5 == 0 or total_b == TOTAL_B - 1:
ge_s(sess, ge, B_S, ged_n, eval_file)
like_d_l.c_bes(eval_file)
t_l = tar_l(sess, tar_lstm, like_d_l)

r.update_pa()

for _ in range(5):
ge_s(sess, ge, B_S, ged_n, negative_file)
dis_d_l.load_train_d(positive_file, negative_file)

for _ in range(3):
dis_d_l.re_po()
for it in xrange(dis_d_l.n_b):
x_b, y_b = dis_d_l.n_b()
f = {
dis.i_x: x_b,
dis.i_y: y_b,
dis.do_keep_prob: dis_do_keep_prob
}
_ = sess.run(dis.train_op, f)


main()
```