2020 年第五届“数维杯”大学生
数学建模竞赛论文

题　　目　**突发舆情事件分析及应对方法**

摘　要

网络舆情作为社会舆情的网络反映，成为社会舆情的最主要的构成之一。如何对舆情的情感倾向分析，并正确引导舆情，给政府和企业带来了前所未有的挑战。

针对问题 1，本文首先对附件 1 中数据进行预处理，得到可供模型使用的语料库。然后利用 LDA 主题聚类模型对附件语料进行初步聚类，得到每一簇语料的簇关键词。同时，利用附件语料训练得到 Word2Vec 模型，将筛选主题的关键词和聚类得到的簇关键词输入到 Word2Vec 模型得到主题关键词以及簇关键词的表征向量，进而求得这些向量之间的余弦相似度，从而将筛选范围限制在相似度最高的一簇语料内，达到舆情信息初步筛选的目的。最后通过计算输入关键词与该簇中语料的相似度，实现最终的舆情信息筛选。

针对问题 2，本文首先设计了用于数据采集的网络爬虫，以实现对网页快速地采集。然后设计了基于网页布局相似度及模块面积大小的加权算法，完整地从网页中抽取了包括标题、正文、时间等与主题内容密切相关的信息。

针对问题 3，本文提出了基于对抗神经网络的文本生成干预方法，采用 GAN 模型，并加入主体和情感记忆向量，通过设置多个分类器进行多任务博弈训练，从而生成关于特定主体且融合特定情感的文本，并以发帖或者回帖的方式对网民进行认知干预。

针对问题 4，采用了建立在主曲线排序理论基础上的无监督排序学习算法。首先构建了一个科学系统的舆情事件演化趋势评价指标体系，将舆情危机设为五个等级。然后采用无监督排序学习算法，得到舆情事件的安全态势评分，评估事件危机等级。

关键词　LDA 主题聚类；Word2Vec；对抗神经网络；排序学习算法

目 录

一、问题重述

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：网络舆情作为社会舆情的网络反映，成为社会舆情的最主要的构成之一。如何正确引导网络舆情，避免不良态势的蔓延，这些给政府和企业带来了前所未有的挑战。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 1 的分析

问题 1 要求提供一个针对某一主题的舆情筛选方法，即针对某一主题下的一个或多个关键词进行舆情信息的筛选。本文提出了一种基于 Word2Vec 以及 LDA 主题聚类的舆情信息筛选模型，首先利用 LDA 模型对附件语料进行初步聚类，得到每一簇语料的簇关键词。同时，利用附件语料训练得到 Word2Vec 模型，通过该模型可以得到输入主题关键词以及簇关键词的表征向量，进而求得向量之间的余弦相似度，从而将筛选范围限制在某一簇内，达到舆情信息初步筛选的目的。最后通过计算输入关键词与该簇中语料的相似度，实现最终的舆情信息筛选。

2.2 问题 2 的分析

问题 2 要求设计一个全新数据的抓取方法。舆情数据的抓取包含数据的网页采集和网页信息的抽取两个部分。

舆情数据采集属于 Web 信息采集的范畴。数据采集的主要方法就是通过网络爬虫来自动在互联网上爬取网页，爬虫程序利用网页之间的链接关系，从一些种子链接出发扩展到整个互联网，将满足条件的网页收集起来，通过在线（网页被下载后）或离线（网页被保存后）的方式，集中进行进一步的分析处理。

舆情数据采集得到的网页含有大量噪声，真正有用的舆情数据往往淹没在噪声中，因此需要通过数据抽取来获取舆情数据。本文采用元数据来描述舆情信息，包含标题、作者、内容、时间、点击量、评论数、评论内容等数据，这些数据是舆情数据抽取中主要的对象。针对此，采用无监督的 Web 信息抽取，从网页的布局角度出发，利用影响网页的布局结构的 HTML 标签，通过查找网页内部相

似的布局结构及板块的面积来确定各个主题信息块，再从主题信息块中提取出元数据，有效提高抽取的效率和准确率。

2.3 问题 3 的分析

问题 3 希望我们能提出一种对政府和企业有利的干预方法，来合理引导网民的感情倾向。删除评论的方式通常具有权限受限、工作量大等缺陷，因此本文采用一种疏导的方法——文本引导，即在有负面情绪的评论下面跟正面情绪的帖，引导网民走向正面的情感倾向，以此改善网络负面氛围。本文建立了融合情感的干预文本生成模型，采用 GAN 模型，并加入主体和情感记忆向量，通过设置多个分类器进行多任务博弈训练，从而生成关于特定主体且融合特定情感的文本。

2.4 问题 4 的分析

问题 4 要求提供一个舆情处理等级的划分方法。为了实现对舆情事件的及时反应和正确应对，首先要构建科学系统的舆情事件演化趋势评价指标体系，采用合理的指标体系，将不同性质、不同发展阶段的舆情事件进行量化，并对舆情事件的演化趋势进行准确判断。在此基础上，将舆情演化趋势评估任务转化为排序学习进行研究，即通过比较得出舆情事件的演化趋势重要性排序，从而定量地评判网络舆情的安全态势，建立预警模型。

三、模型假设

本文对题目中的舆情情感倾向分析问题提出以下假设：

假设 1：附件中所提供的文本都是有效的，即不存在无语义、不通顺的样本

假设 2：问题中所针对的舆情信息来源网站是可爬取的，即不存在由于反爬措施导致的字段缺省情况。

假设 3：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。

假设 4：为了解决问题 3 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。

四、定义与符号说明

符号定义	符号说明
θ	概率
λ	加权因子
V^*	最优主题簇
ε	语义距离阈值
L	熵
余弦相似度 Similarity	通过计算两个向量的夹角余弦值来评估他们的相似度
布局相似度 *SoL*	0 到 1 之间的值，度量节点在布局结构上的相似度

五、模型的建立与求解

5.1 问题 1 的模型建立与求解

5.1.1 模型的建立

本文提出了一种基于 Word2Vec 以及 LDA 主题聚类的舆情信息筛选模型，如图 5-1。首先对附件 1 中的数据进行预处理，然后利用 LDA 模型对附件语料进行初步聚类，得到每一簇语料的簇关键词。同时，利用附件语料训练得到 Word2Vec 模型，通过该模型可以得到输入关键词以及簇关键词的表征向量，进而求得向量之间的余弦相似度，从而将筛选范围限制在某一簇内，达到舆情信息初步筛选的目的。最后通过计算输入关键词与该簇中语料的相似度，实现最终的舆情信息筛选。

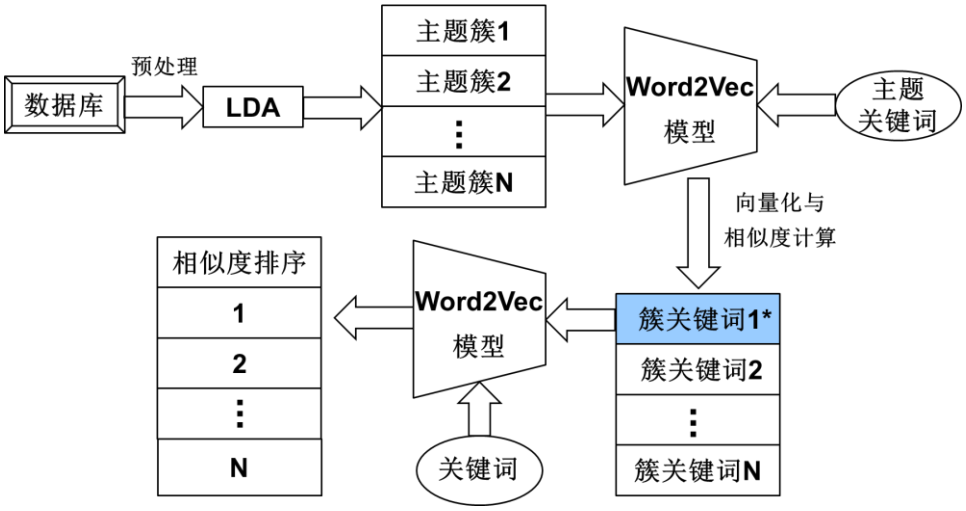

图 5-1 舆情信息筛选模型

5.1.2 模型的求解

（一）数据预处理

附件 1 中的数据需要经过分词、剔除停用词等预处理才能成为供后续模型、使用的语料库。本文使用 *jieba* 提供的函数实现停用词的提出以及进一步的分词处理。

（二）LDA 主题聚类

聚类是数据挖掘的重要工具，通过对无标记的大型数据进行分析，根据样本之间的相似情况进行分组，使组内的数据之间彼此性质互相靠近，而组间数据相似距离较大。由于这种自动分类方法所具有的"无监督"性，从而使聚类方法在海

量信息处理、基因数据分析、模式识别等领域中得到极大的应用。该方法通过引入文本主题分布大大降低了数据的维度，同时模型的参数空间规模是固定的，与文本集自身规模无关，因此更适用于大规模文本集。本文即采用 LDA 模型实现文本主题聚类以及关键词挖掘。

LDA 对文档集 D 中的每篇文档 w 进行以下的处理过程：

（1）选择一个N ~ Possion(ξ)，N 服从泊松分布。

（2）选择θ~Dir (α)，θ服从 Dirichlet 分布。

（3）对于具有 N 个词的每篇文档中的每个词w_n。

a. 选择一个主题 z_n~ Multinomial(θ)，z 服从多项式分布。

b. 从主题z_n中以p$(w_n - z_n)$的概率选择一个词w_n，服从多项式分布。

LDA 的联合概率为：

$$\mathrm{p}(\theta, z|\alpha, \beta) = \mathrm{p}(\theta, \alpha) \prod_{n=1}^{N} \mathrm{p}(z_n|\theta)\mathrm{p}(w_n|z_n, \beta) \quad (1)$$

算法开始时，先随机地给α和β：赋值。然后上述过程不断重复，最终收敛到的结果就是 LDA 的输出。

利用 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：

$$LDA(\text{语料}) = \{\mathcal{K}_1, \mathcal{K}_2, \mathcal{K}_3 \ldots \mathcal{K}_N\} \quad (2)$$

其中$\mathcal{K}_i$表示第 i 个主题簇对应的关键词集合：

$$\mathcal{K}_1 = \{\text{关键词 1}, \text{关键词 2}, \text{关键词 3} \ldots \text{关键词 M}\} \quad (3)$$

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，如图 5-2。

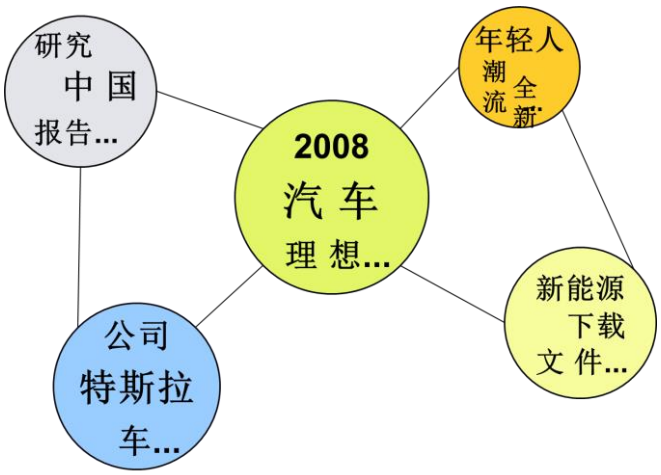

图 5-2　LDA 模型聚类结果

（三）Word2Vec 模型训练

Word2Vec 工具摒弃了传统方法的 one-hot 编码方式，实质上是一个两层神经网络，将一个词语对应一个多维向量，通过该多维向量允许我们用变化较小的数字来表征词语，通过欧氏距离或者余弦相似度就可以把相近意思的词语放在相近的位置，而且一般用的是实数向量.通过将所有的词向量化,词与词之间就可以

定量的度量他们之间的关系。Word2Vec 包括 CBOW 模型和 skip-gram 模型，前者是根据上下文预测当前词语概率，后者则通过当前词预测上下文概率。本文选取 Word2Vec 里的 skip-gram 词向量模型将文本转换为词向量。

给定针对某一主题的关键词集合$\widetilde{\mathcal{K}}$后，通过寻找与主题关键词语义最相近的簇关键词集合$\mathcal{K}^*$，并将该簇作为初步舆情信息筛选的结果。为了能够计算这两者之间的相似度，首先需要利用 2.1 中划分好的语料库训练 Word2Vec 模型，通过训练好的 Word2Vec 模型，分别得到两者的低维表征向量，进而计算相似度。该部分模型如下图 5-3 所示。

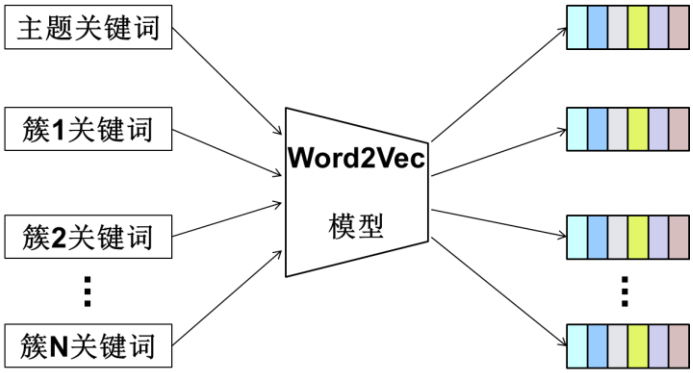

图 5-3 基于 Word2Vec 的训练模型

（四）关键词相似度计算

本文使用余弦相似度作为度量标准：

$$Similarity(\mathrm{x},\mathrm{y}) = \frac{\sum_{i=1}^{k}(x_i \times y_i)}{\sqrt{\sum_{i=1}^{k}(x_i)^2} \times \sqrt{\sum_{i=1}^{k}(y_i)^2}} \tag{4}$$

其中k为关键词x与关键词y经 Word2Vec 模型输出的词向量维度，本文中$k = 300$。通过计算主题关键词与每个主题簇中关键词的相似度均值，将平均相似度最大的主题簇中的文本作为初步的舆情信息筛选结果，即：

$$V^* = \underset{V_i}{\operatorname{argmax}}\, Similarity\left(V_i, \tilde{V}\right) \tag{5}$$

其中$\tilde{V}$为主题关键词词向量集合，V_i为第 i 个簇的关键词向量集合，主题簇V^*中的文本即为初步筛选得到的结果。

（五）基于相似度排序的文本筛选

将 2.4 中确定的主题簇作为初步筛选的文本后，利用 2.3 中训练好的 Word2Vec 将给定的主题关键词以及该主题簇中的文本转化为低维表征向量，计算关键词与文本之间的相似度，并将相似度从高到低进行排序，而经过排序得到的文本，其与主题关键词之间的相关度也从上到下逐渐递减。该模块实现效果如图 5-4 所示。

文本	得分
早在2018年就有消息称福特福克斯ActiveWagon（跨界旅行版）有望国产，如今一年多的	0.7333932134423544
其实很多人都口水GTI,领克03+,福克斯ST这样的性能小钢炮的,这部分人不会因为省油而去	0.7299471810281009
欢迎来到评中评探店，本周编辑孟斌在福特4S店，为您带来2020款福克斯ST-Line两厢版2	0.7102744231145771
十万块买车是大多数家庭的预算，因为这个级别的车型适合小康家庭使用，不管是从经济	0.7092449160013359
对于一些消费者来说，他们买车只是普通的追求代步，不追求豪华，不少的消费者本身购	0.7030172317009029
锐际作为长安福特全新推出的一款紧凑型suv，不少人对这个名字还有点陌生，但对福特比	0.6973141325205178
近年来，在SUV热潮以及豪华品牌下探的冲击之下，各大普通品牌B级车的市场份额受到	0.6971358477742414
15万预算，19年值得买的几款车，第二款省油，第三款配置丰富！15万的价格对于我们普	0.6942029353584565
作者：陈曦 大要是1968年的时候，福特在欧洲推出了一款吨吨拉力赛场的车型，这即是E	0.6916305026125917
10年前买车，大家往往更愿意选择合资车，一方面是那时候的国产车长得有些难看，另一	0.6915959419488609
国产汽车现在因为质量好，配置高，已经被大众认可了，并且这位朋友原本就是国产汽车	0.6905353169258491
资深车迷点评心中最美女神，不到20万起即可拥有她--说起福特，我想起了我大学时期的	0.6900509469558704
如今在汽车市场中，虽然向多元化发展，SUV变得原来越火爆。但是主导中国市场的依旧	0.6896513091796097
对于一些消费者来说，他们买车只是普通的追求代步，不追求豪华，不少的消费者本身购	0.6894316795621904

图 5-4 基于相似度排序的文本筛选结果

5.1.3 模型的验证与结果

为了验证舆情筛选方法的有效性，本文采用基于先验知识的模型验证方法。首先根据输入的主题关键词，构建与其主题契合的语料库$\mathcal{P}$，再构建与其主题背离的语料库$\mathcal{N}$。构建内容如下：

语料库 $\mathcal{P}$	奥迪汽车性价比太低
	年轻人有经济能力买车吗
	美国疫情太严重了
语料库 $\mathcal{N}$	吃菜有益身体健康
	蝗虫是害虫
	门口有个垃圾桶

在 2.5 中得到的排序结果中，选取相关性最高的文本作为参考文本，再分别利用 Word2Vec 产生的低维表征计算参考文本与$\mathcal{P}$、$\mathcal{N}$中语料的相似度，进而证明通过本文提出的方法，可以针对某一主题实现舆情信息筛选。最终的相似度结果如表 5-1：

表 5-1 参考文本与$\mathcal{P}$、$\mathcal{N}$中语料的相似度

关键词	对比的句子		相关度结果
汽车，性价比	句 1	奥迪汽车性价比太低	64.96%
	句 2	吃菜有益身体健康	35.53%
年轻人，经济	句 1	年轻人有经济能力买车吗	45.4%
	句 2	蝗虫是害虫	0%
美国，疫情	句 1	美国疫情太严重了	51.84%
	句 2	门口有个垃圾桶	27.14%

以第一组数据为例，当在程序中输入关键词“汽车、性价比”时，模型从附件 1 中输出了以相关度排序的一系列语料。为了验证模型的正确性，我们挑选出与

关键词相关度最高的语料文本 a，然后分别输入句 1(包含关键词)：“奥迪汽车性价比太低”、句 2（不包含关键词）“吃菜有益身体健康”两个句子，输出句 1 与文本 a、句 2 与文本 a 的相关度结果。很明显句 1 与文本 a 的相关度应该高于句 2 与文本 a 的相关度，而实验结果显示句 1 与文本 a 的相关度为 64.96%，明显高于句 2 与文本 a 的相关度 35.53%，与认知相符合，说明我们的模型具有可行性。

5.2 问题 2 的模型建立与求解

5.2.1 模型的建立

对于问题 2，首先设计了数据采集的网络爬虫，以实现对网络舆情数据精确、快速地采集。然后采用无监督的 Web 信息抽取，设计了基于网页布局相似度及模块面积大小的加权算法，完整地提取出包括标题、正文、作者等与主题内容密切相关的信息。整个模型如图 5-5。

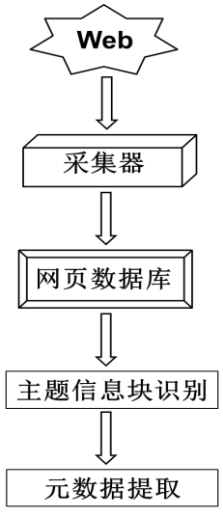

图 5-5 数据抓取模型

5.2.2 模型的求解

（一）舆情数据采集

网络爬虫的基本原理很简单，从给定的种子链接出发，向 Web 服务器发送 HTTP 请求（本文只讨论 HTTP 协议），获取网页内容，并从中分析出所有链接，依据一定的算法从中选取某些链接添加到下载队列中，重复这个过程直到满足停止条件。其工作的基本流程如图 5-6 所示。

（二）舆情数据抽取

在进行舆情数据抽取时，我们主要关心发表时间、评论人数、关注人数及具体内容等具有深层次分析价值的元数据，包含这些内容的区域称为网页的主题信

息块。为了快速，准确地从网页中提取出元数据，我们采用分级处理的方式，主要包括两个步骤：

（1）网页级处理，滤除整体噪声，识别出主题信息块；

（2）区域级处理，滤除局部噪声，从主题信息块中提取出元数据。

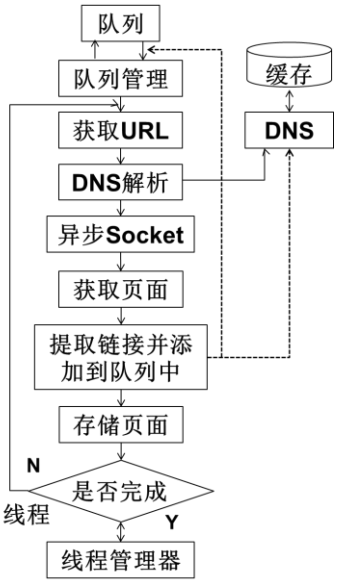

图 5-6　网络爬虫工作流程图

A、主题信息块识别

网页一般被划分成不同内容的块状区域且都有固定的位置，比如主题块在网页中间，导航区位于页面顶端，广告一般在两侧，版权信息在网页底部。这种布局实现是由 HTML 语言的块状标签节点来控制的，这些节点将网页分割成各个相对独立的区域。通常来说，网页的主题信息块含有大量（通常都有数十个）布局结构相似的区域，如图 5-7。然而，通常网页中存在不止一个具有布局结构相似的区域，如导航栏、广告栏等，也会具有与主题信息块一样的特性。

图 5-7 网页示意图

为了防止以上情况带来的爬取误差，进一步准确识别主题信息块，我们首先

找到网页中含有大量相似结构的区域，将这些区域称为候选主题信息块。同时，根据主题信息块通常占据页面面积最大的特性，本文提出一种基于网页布局相似度及元素面积大小的加权算法，来综合确定主题信息块。算法的伪代码如下：

Algorithm 1 基于网页布局相似度及元素面积大小的加权算法.

输入页面中的所有 DOM 节点、当前页面 URL、权重 λ.
设相似节点数量集合 $\mathcal{C}=[\,]$ 与 DOM 节点面积集合 $\mathcal{S}=[\,]$
for 页面中的每个 DOM 节点 DOM_i **do**
    Count = F(DOM_i)
    if Count >= K
        $\mathcal{C}$.append(Count)
end for
for 候选主题信息节点 DOM_j in $\mathcal{Y}$ **do**
    $\mathcal{S}$.append(Selenium(URL,DOM_j))
end for
$\tilde{\mathcal{C}}=\frac{\mathcal{C}}{\sum_{i-1}^{m}\mathcal{C}_i}$
$\tilde{\mathcal{S}}=\frac{\mathcal{S}}{\sum_{i-1}^{m}\mathcal{S}_i}$
$\mathcal{Q}=\lambda*\tilde{\mathcal{C}}+(1-\lambda)*\tilde{\mathcal{S}}$
输出加权和 $\mathcal{Q}$

Algorithm 2 候选主题信息模块筛选算法 F.

输入 DOM 节点、DOM 节点子节点数量 L、相似度阈值 R、相似节点数量阈值 K.
for DOM 节点中的每一个子节点 i (i<L-K) **do**
    Count = 0;
    for DOM 节点中的每一个子节点 j (i<j<L) **do**
        计算节点 i 与节点 j 之间的布局相似度 **SoL(x,y)**;
        if **SoL(x,y)>R** Count++;
    end for
    if **Count>=K**
        将 DOM 节点加入到候选主题信息模块集合 $\mathcal{Y}$ 中, 返回 Count;
end for
if **Count<K**
for 当前 DOM 节点中的每个子节点 childNode **do**
    F(childNode)
end for
返回 Count

候选主题信息模块筛选算法将返回输入 DOM 节点中的相似子节点数量 Count，同时将符合条件的 DOM 节点添加到候选主题信息模块集合$\mathcal{Y}$中。我们定义节点的布局相似度 *SoL*（Similarity of Layout）来度量两个节点在布局结构上的相似程度。*SoL* 是 0 到 1 之间的值，越接近 1 表示两个节点的布局结构越相似。设有两个节点 x, y 则它们的 *SoL* 定义为：

$$SoL(x,y)=\sum_{i=1}^{N}\omega_i\sum_{j=1}^{M_i}\frac{1}{M_i}S_{ij} \tag{6}$$

其中 N 表示比较的深度，即只比较到第 N 层节点，N 一般取 2 或 3；M_i表示第 i 层子节点的个数；为第 i 层子节点对整体结构布局的贡献系数，其值满足公式（7）。

$$\sum_{i=0}^{N-1}\omega_i=1,\omega_i>\omega_j,0\le i<j\le N \tag{7}$$

一般认为越深层次的节点对宏观布局的影响越小，因而它们对应的ω_i值就越

小，其反映的更多的是节点间细节上的差异。S_{ij}表示进行比较的两个节点的第 i 层，第 j 个节点是否为同种类型的块状节点，其值取 0 或 1。首先判断两个节点是否为使用了同样的 HTML 标签，比如是否同为 TR 或 TD 标签，若不同，则为 0；若相同，则继续比较两节点属性是否相同，如 width，style，align 等能反映节点布局结构的属性，若这些属性值也相同，则S_{ij}为 1。

算法 1 通过遍历页面中所有 DOM 节点，将所有符合条件的 DOM 节点放入候选主题信息模块集合$\mathcal{Y}$中，同时将各个候选模块的相似子节点数量加入集合$\mathcal{C}$中。根据前文讨论可知，仅凭区域内相似结构的数量判断是否为主题信息块是片面的，因此本文将区域相似子节点大于 K 的信息块称为候选主题信息块，K 为设定的阈值。针对候选主题信息块集合中的每一个 DOM 节点，通过 Selenium 爬虫框架追踪 URL，并提取当前 DOM 节点的长宽尺寸，进而计算得到其面积，最终产生一个候选主题信息块面积集合$\mathcal{S}$。

最后，分别对集合$\mathcal{C}$与集合$\mathcal{S}$进行归一化处理，并进行加权求和，λ为加权因子，得到各个候选主题信息块的最终评分，评分越高说明越有可能是主题信息块，即：

$$Q = \lambda * \frac{C}{\sum_{i-1}^{m} c_i} + (1-\lambda) * \frac{S}{\sum_{i-1}^{m} s_i} \tag{8}$$

B、元数据提取

元数据在主题信息块中的位置是相对固定的而且数据本身各有特点，如时间一般有固定的格式；正文通常含有较多文本且其中很少出现超链接；标题链接的锚文本长度一般要大于其它链接的锚文本长度，这些信息都有助于我们正确提取数据。但有些情况不能忽视，比如有些回帖的内容很短；正文中出现的数字，时间；表示作者的链接锚文本可能长于标题的链接锚文本。另外，主题信息块中仍有少量噪声，比如一些广告链接和功能性链接等，这些情况都将严重影响抽取的准确性。

由于主题信息块之间是相似的，相同的内容都有相同的表现形式，会表现出一定的统计规律性，比如若某一位置所有的主题信息块中都出现时间则可认为是时间，从而区别于个别正文中出现的时间。因此，我们考虑所有的主题信息块。将主题信息块表示成具有明显语义信息的节点的集合，比如文本节点、超链接、图片等，其它节点不予考虑。第 i 个主题信息块B_i表示为：$B_i = \{n_1, n_2, n_3 \dots n_k\}$，$n$ 代表各语义节点。

采用深度优先的方式遍历主题信息块中的所有节点，按照下面的步骤得到B_i：

（1）获取下一个要处理的节点，若为空，则结束；否则，转至（2）。

（2）若当前节点的子节点只含有文本节点或链接节点，则将其添加到B_i中，转至（1）。

由上述方法将所有主题信息块表示成语义节点的集合。先对B_i中的节点进一步过滤，若所有的B_i中节点n_i都相同，则认为n_i是噪声节点，再应用以下规则从B_i中抽取代表元数据的节点：

R_1：对所有B_i中对应非锚文本节点求出其长度的平均值，最大者为正文；

R_2：对所有B_i中对应链接节点求出其锚文本长度的平均值，最大者为标题；

R_3：所有B_i 中对应某节点其文本中均含有数字则为查看回复数；

R_4：所有B_i中对应某节点其文本中均含有一定格式的时间字符串则为时间；

5.2.3 模型的验证与结果

对于具有相同布局的网站簇，如百度贴吧下的各个贴吧子站点，本文直接编写合适的爬虫进行所需信息的爬取，如发布日期、回复人数等。这类网站通常具有稳定的结构特征，因此采用常规的爬虫手段进行信息爬取即可。以“考试”为主题关键词，针对这种情况本方法在贴吧类的站点中爬取结果如图 5-8。可以看到该情况下本方法设计的爬虫能够将所需的舆情信息完整的爬取下来，具有极高的参考价值。

图 5-8 贴吧类子站点信息抓取结果

针对不同布局的网站，根据本文提出的基于网页布局相似度及模块面积大小的加权算法，本文实现了从不同布局的网站中爬取相同字段的爬虫，选取贴吧、百度搜一搜作为爬取站点验证算法有效性，爬取结果如图 5-9。通过图 5-9 可以观察到，根据本文方法编写的爬虫可以同时满足百度搜索和百度贴吧的信息爬取，内容包括 url、标题、正文、发布时间以及评论数。证明了本文爬取方法的有效性。

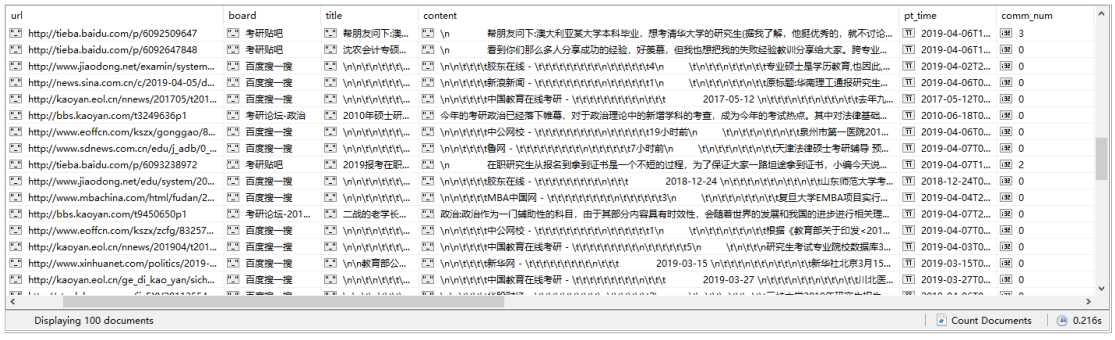

图 5-9 不同布局网站信息抓取结果

5.3 问题 3 的模型建立与求解

5.3.1 模型的建立

针对问题 3，本文提出了文本引导模型，如图 5-10。收集社交网络中热点事件的全部博文作为源语料集，根据舆情事件管理需要，通过融合情感的干预文本生成模块生成与网络负面文本对抗的观点，然后发表在评论下，实现对网民的情

感倾向的引导。比如甲评论：“我很讨厌 xxx”，则在评论下面评论：“xxx 很好”。从 web 抽取源语料已经在问题 2 中得到很好的解决，因此我们接下来主要介绍干预文本生成模块的求解。

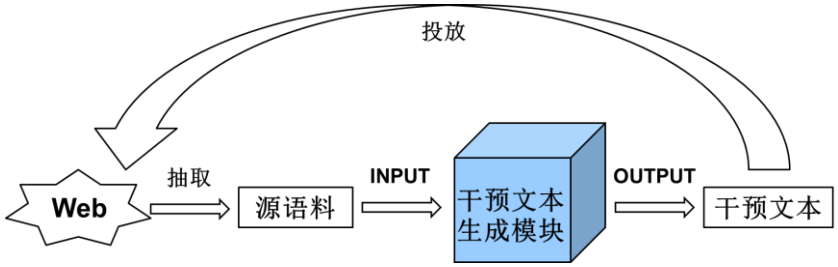

图 5-10 文本引导模型

5.3.2 模型的求解

干预文本生成模块旨在生成针对目标主体的特定情感倾向的文本，从而更具针对性和贴合性地对目标主体进行干预。干预文本生成模块采用 GAN 模型的多任务训练方式，分为构造分类器（判别器）和生成器判别器博弈训练两部分，如图 5-11。

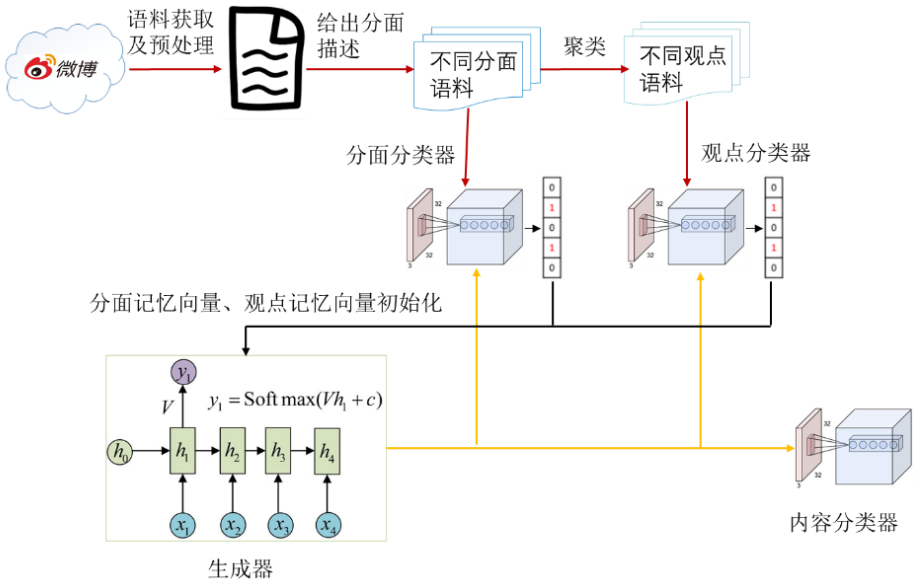

图 5-11 融合情感的干预文本生成架构图

（一）分类器构造

首先收集微博上某舆情事件的全部博文作为源语料集，对源语料集进行清洗后，以标点符号为依据将博文分割为多个子句。根据专家知识或舆情事件管理需要，确定该舆情事件的重要分面，对于每个分面，给出关键词或短语描述，即分面描述。利用源语料集训练 Word2Vec 词向量模型，从而获得语料句子和分面描述的词嵌入向量表示，例如，语料句子表示为$Y = \{y_1, y_2, y_3 \dots\dots y_n\}$，其中$y_i, i = 1,2,3 \dots\dots$表示每个词的词嵌入向量。同理，设定每个分面描述的句子为$A = \{a_1, a_2, a_3 \dots\dots a_n\}$。

利用源语料集训练编码-解码模型，这里的编码器、解码器均为长短期记忆网络 LSTM，具体编码过程为:

$$h_{final} = LSTM1(h_0, Y) \tag{9}$$

$LSTM1$ 为编码器，h_0为初始隐层向量，h_{final}为最终输出隐层向量。解码过程为：

$$Y' = LSTM2(h_{final}) \tag{10}$$

$LSTM2$为解码器，Y′为解码器生成的句子。优化目标为使Y和Y′的交叉熵 L 最小。

$$L = -\frac{1}{k}\sum_0^k Y\, log(Y') \tag{11}$$

k 为语料中子句的总数。训练完成后将分面描述和源语料集中每个句子输入训练好的编码器：

$$h = LSTM1(Y) \tag{12}$$

$$h' = LSTM1(A) \tag{13}$$

利用得到的语料句子的隐含语义向量 h 和分面描述隐含语义向量h'计算句子和每个分面描述的语义距离，例如以h和h'的余弦相似度作为其语义距离：

$$r = \frac{h*h'}{|h|*|h'|} \tag{14}$$

并设定语义距离阈值ε，当 r>ε则认为语料句子属于该分面，从而实现源语料集中不同分面的子句提取。给不同分面的语料子句打上不同的标签，如 001,010,100，用带有标签的语料子句训练分面分类器，使其可以区分子句所属分面。这里的分面分类器为卷积神经网络 CNN，训练过程为：

$$logits=conv1(Y) \tag{15}$$

其中 conv1 为卷积函数，logits 为卷积网络输出。

$$output=softmax(w*logits+b) \tag{16}$$

将卷积结果 logits 输入全连接层，其中 w 和 b 分别为可学习参数，全连接层的最后进行 softmax 操作，得到预测的分类结果 output。优化目标为使分类器对语料句子所属分面的判别与其真实所属分面相同，即计算结果与标签的交叉熵 L 最小。

$$L = -\frac{1}{k}\sum_0^k label\, log(output) \tag{17}$$

其中 label 为每个子句真实所属分面的标签。

针对每个分面的语料，利用基于观点的文本聚类方法，将该分面中的表达不同观点的子句分在不同的簇，表达相同观点的子句分在相同的簇，最后给不同簇中的子句打上不同的标签，同理给其赋标签为 001,010,100……，表示其表达不同的观点。

利用得到的带有标签的语料集子句训练观点分类器，使其能够区分同一分面内表达不同观点的语料子句。其中观点分类器同样为卷积神经网络，训练过程与分面分类器同理。

（二）生成模型训练与推理

文本生成模型包括一个生成器和三个对抗训练的判别器，生成器生成和语料句子分面相同、观点对抗的文本，三个判别器区分生成器生成的句子和源语料句子，如此反复对抗训练，直到三个判别器均无法区分生成器生成的句子和源语料句子。

A、文本生成模型训练阶段

对于每一条源语料句子，利用训练好的分面判别器识别其所属分面，利用训练好的观点判别器识别其所属观点：

$$logits_1=\text{conv1(Y)} \tag{18}$$

$$label_1=\text{softmax}(w_1*logits_1+b_1) \tag{19}$$

其中 conv1 为分面判别器卷积函数，$logits_1$为分面判别器输出，$label_1$为语料句子所属分面类别。

$$logits_2= conv2\ \text{(Y)} \tag{20}$$

$$label_2=\text{softmax}(w_2*logits_2+b_2) \tag{21}$$

其中$conv2$为观点判别器卷积函数，$logits_2$为观点判别器输出，$label_2$为语料子句表达观点类别。

为了使生成器模型更容易训练，用语料句子的分面判别器输出$logits_1$初始化分面记忆向量，其对抗观点判别器输出$logits_2$初始化观点记忆向量，因为分面判别器的输出$logits_1$本身包含了句子的隐含分面信息，其对抗观点判别器的输出$logits_2$同理。

如图 5-12 生成器实现细节图所示，在编码器每一层，将相应的观点记忆向量和分面记忆向量与上一层输出的隐层向量拼接，并输入到编码器的下一层。即：

$$y'_{t+1}, h_{t+1} = LSTM(y'_t, [h_t; m; v]) \tag{22}$$

其中，y'_t为生成器 t 时刻的输出，h_t为 t 时刻的隐层向量，m为分面记忆向量，v为观点记忆向量。y'_{t+1}为生成器 t+1 时刻的输出，h_{t+1}为 t+1 时刻的隐层向量。

生成器的优化目标为使生成文本$Y' = \{y'_1, y'_2, y'_3 \dots\dots y'_n\}$和原语料句子$Y = \{y_1, y_2, y_3 \dots\dots y_n\}$的交叉熵最小：

$$\min_G\ L_G = -\frac{1}{k}\sum_0^k Y\ log(Y') \tag{23}$$

生成器最终的输出$Y' = \{y'_1, y'_2, y'_3 \dots\dots y'_n\}$和原语料句子的嵌入向量表示$Y = \{y_1, y_2, y_3 \dots\dots y_n\}$作为分面判别器、内容判别器和观点判别器的输入。

判别器优化目标由三部分组成，第一部分是分面判别器D_1对生成器输出向量Y'和原语料句子嵌入向量Y的判别结果交叉熵L_1最大：

$$\max_{D_1}\ L_1 = -\frac{1}{k}\sum_0^k D_1(Y)\ log(D_1(Y')) \tag{24}$$

第二部分为让观点判别器D_2对生成器输出向量Y'和原语料句子嵌入向量Y的判别结果交叉熵最小。

$$\min_{D_2}\ L_2 = -\frac{1}{k}\sum_0^k D_2(Y)\ log(D_2(Y')) \tag{25}$$

第三部分为让内容判别器D_3对生成器输出向量Y'和原语料句子嵌入向量Y的判别结果交叉熵最大。

$$\max_{D_3}\ L_3 = -\frac{1}{k}\sum_0^k D_3(Y)\ log(D_3(Y')) \tag{26}$$

判别器部分的总优化目标L_D为：

$$\max_{D_1,D_2,D_3} L_D = L_1 + (1 - L_2) + L_3 \tag{27}$$

该优化目标保证了生成器生成的句子既在内容上符合该事件的自然语言规律又表达出要求的该分面的该观点。

最后，文本生成模型的总优化目标L为：

$$\min_{G,D_1,D_2,D_3} L = L_G + (1 - L_D) \tag{28}$$

生成器生成和语料句子分面相同、观点对抗的文本，三个判别器区分生成器生成的句子和源语料句子，如此反复对抗训练，直到三个判别器均无法判别出生成器生成的句子和源语料句子。

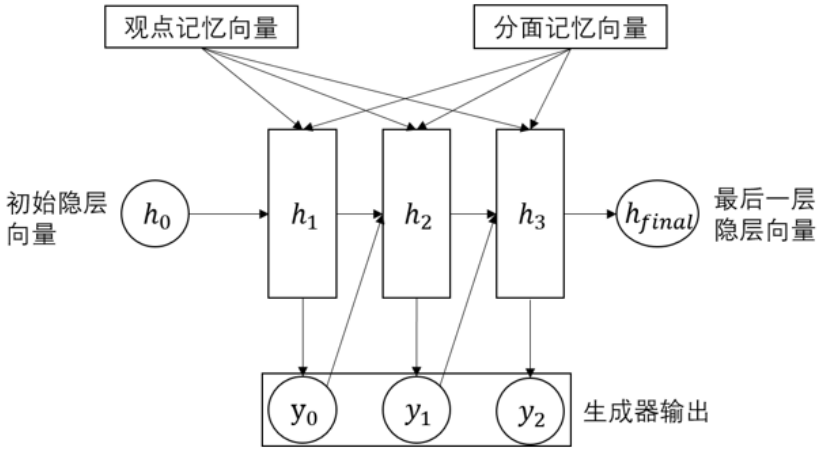

图 5-12 模型生成器细节图

B、文本生成模型推理过程

对于每一条源语料句子，利用分面判别器识别其所属分面，利用观点判别器识别其所属观点。

调用相应的已经训练好的分面记忆向量 m 和与其形成对抗的观点记忆向量 v。分面记忆向量 m、观点记忆向量 v 与初始随机向量拼接输入生成器，生成器则生成与给定语料集子句具有相同分面但观点相反且符合该舆情事件语言规律的文本。

5.3.3 模型的验证与结果

（一）数据集

模型训练数据集由微博、论坛、新闻网站等社交媒体上爬取的 2019 年热点舆情事件的网络发帖组成，分别有不同的舆情事件语料，部分见表 5-2：

表 5-2 舆情事件语料数据

舆情事件名称	事件跨度	语料数量（条）
男子穿“和服”进武大赏樱遭殴打	2019 年 3 月 24 日-2019 年 3 月 26 日	83168
陕西奔驰女车主维权	2019 年 4 月 11 日-2019 年 4 月 17 日	202374
保时捷女怒扇男司机耳光	2019 年 7 月 30 日-2019 年 7 月 31 日	92122
昆明理工大学学生李心草溺亡	2019 年 10 月 12 日-2019 年 10 月 16 日	164521

（二）测试结果分析

以 2019 年 3 月 24 日至 3 月 26 日发生的‘男子穿“和服”进武大赏樱遭殴打’事件为例进行融合情感的干预文本生成方法测试。

在本事件中，主要有‘穿和服男子’以及‘武汉大学’两个分面主体，分别存在的观点有：

观点 1: 主体为穿和服男子，情感倾向为负向，认为该男子穿和服严重冒犯了我国的民族情结和勿忘国耻的爱国情怀以及

观点 2： 主体为武汉大学，情感倾向为负向，武汉大学作为中国甚至世界一流大学，包容性不足，表现了狭隘的民族主义倾向。

观点 3: 主体为武汉大学，情感倾向为正向，肯定和支持武汉大学的做法，认为需要有自己的原则。

对该事件生成融合情感的认知干预的文本，主要包括生成上述两个主体的正负情感倾向的观点，生成的部分干预文本见表 5-3。从测试结果分析可得，融合情感的干预文本基本满足特定主体和情感倾向要求，也基本符合正确的语言语法规律。

表 5-3 男子穿“和服”进武大赏樱遭殴打’事件融合情感的干预文本生成结果

主体为穿和服男子，情感倾向为负向	1. 呵呵，中国人不要那样的不肖子孙。
	2. 这两人如果是中国人，他们有故意挑衅的嫌疑！
	3. 看到中国人在里面穿和服心里也会不爽。
主体为武汉大学，情感倾向为正向	1. 强烈支持武大，做人不能忘本。
	2. 学校有权利选来参观的人，没毛病啊。
	3. 大学里的花，是他的自由，制定个赏花规则没毛病。
主体为武汉大学，情感倾向为负向	1.武大这种管理水平算是一流大学？可笑。
	2. 武汉大学干出这么低级的事？穿和服到底影响什么了？
	3.国内知名大学能做出这样的通知，也真是耻辱！狭隘、无知！

5.4 问题 4 的模型建立与求解

5.4.1 模型的建立

针对问题四，首先构建一个科学系统的舆情事件演化趋势评价指标体系，然后采用无监督排序学习算法，得到舆情的安全态势。如图 5-13 所示，可以看出是一个典型的“输入—处理—输出”的过程。输入是指标体系的实际值，输出是要得到的结果，即舆情事件的演化安全态势评分，中间的处理部分是采用排序学习算法。

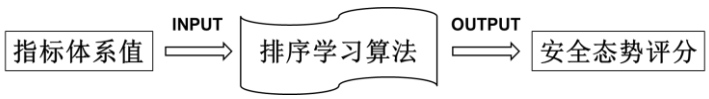

图 5-13 模型示意图

5.4.2 模型的求解

（一）评价指标体系的建立

在查阅文献的基础上，以“因人”“因事”“因势”为中心构建舆情事件指标体系，包括基本指标和进阶指标。基本指标是指可以由舆情数据集直接计算得到的指标，例如参与人数量，舆情事件传播速度（以单位时间内新增帖子数量来表征）等；进阶指标是指需要引入外部信息辅助计算才能得到的指标，例如舆情事件的敏感度，事件的倾向性等。体系如图 5-14 所示。

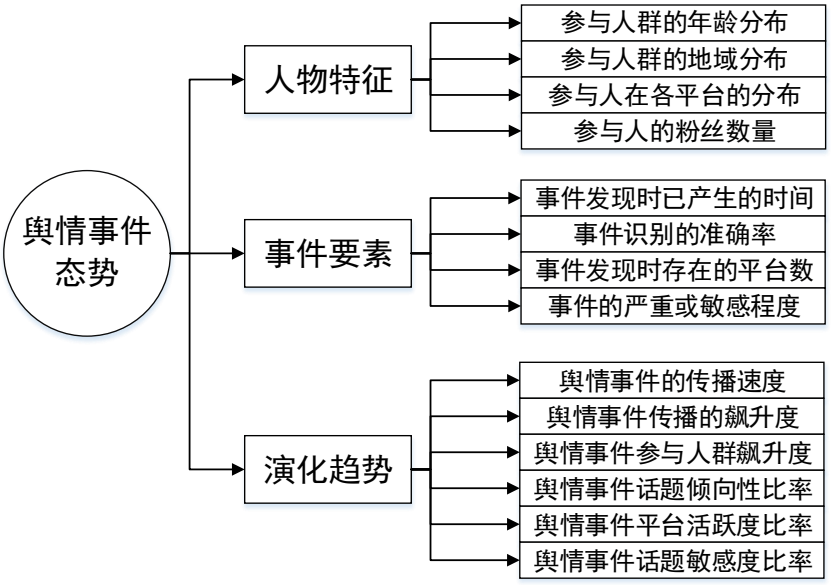

图 5-14 舆情事件演化趋势评估指标体系

本文所构建的舆情事件评价指标体系为：对舆情事件的总体评价p为一级指标；以a人物特征，b事件要素，c演化趋势构建二级指标；由二级指标的具体内容构建三级指标，其中，a人物特征所辖三级指标为（a_1参与人群的年龄分布，a_2参与人群的地域分布，a_3参与人在各平台的分布，a_4参与人的粉丝数量），b事件构成要素所辖三级指标为（b_1事件发现时已产生的时间，b_2事件识别的准确率，b_3事件发现时存在的平台数，b_4事件的严重或敏感程度），c事件演化趋势所辖三级指标为（c_1舆情事件的传播速度，c_2舆情事件传播的飙升度，c_3舆情事件参与人群飙升度，c_4舆情事件话题倾向性比率，c_5舆情事件平台活跃度比率，c_6舆情事件话题敏感度比率）。各三级指标值的计算方式和形成的指标评价对象向量如下：

$a_1 = [a_{11}, a_{12}, \dots, a_{1n}]$，$n$ 为设定时间点数，n 不小于 1；a_{1n}表示第 n 个设定时间点处获取的参与人群的年龄分布，用AG表示此时参与人群年龄信息的平均值，AG_s表示此时参与人群年龄信息的标准差，AG_m表示此时参与人群年龄中最大最小年龄之差，则$a_{1n} = \frac{100AG}{70} - \left(\frac{100AG}{70} MOD 10\right) + \frac{10AG_s}{AG_m}$；

$a_2 = [a_{21}, a_{22}, \dots, a_{2n}]$，$n$ 为设定时间点数，n 不小于 1；a_{2n}表示第 n 个设定时间点处获取的参与人群的地域分布，即为由每个地域参与人数量与总参与人数量之比得到的每个地域参与人数量比例数据集合的标准差；

$a_3 = [a_{31}, a_{32}, \dots, a_{3n}]$，$n$ 为设定时间点数，n 不小于 1；a_{3n}表示第 n 个设定时间点处获取的参与人在各平台的分布，即为由每个平台参与人数量与总参与人数量之比得到的每个平台参与人数量比例数据集合的标准差；

$a_4 = [a_{41}, a_{42}, \dots, a_{4n}]$，$n$ 为设定时间点数，n 不小于 1；a_{4n}表示第 n 个设定时间点处获取的参与人的粉丝数量，即为所有参与人粉丝数量的平均值；

$b_1 = [b_{11}, b_{12}, \dots, b_{1n}]$，$n$ 为设定时间点数，n 不小于 1；b_{1n}表示第 n 个设定时间点处获取的事件发现时已产生的时间，即从事件发生到爬取事件信息时所经历的时间$\mathrm{t}_n - \mathrm{t}_0$；

$b_2 = [b_{21}, b_{22}, \dots, b_{2n}]$，$n$ 为设定时间点数，n 不小于 1；b_{2n}表示第 n 个设定时间点处获取的事件识别的准确率，即爬取的帖子中符合舆情事件主题的帖子数量占总帖子数量的百分比$\mathrm{TP}/\mathrm{TP}_0$；

$b_3 = [b_{31}, b_{32}, \dots, b_{3n}]$，$n$ 为设定时间点数，n 不小于 1；b_{3n}表示第 n 个设定时间点处获取的事件发现时存在的平台数，即爬取的数据所属的站点数量 NS；

$b_4 = [b_{41}, b_{42}, \dots, b_{4n}]$，$n$ 为设定时间点数，n 不小于 1；b_{4n}表示第 n 个设定时间点处事件的严重或敏感程度评级；

$c_1 = [c_{11}, c_{12}, \dots, c_{1n}]$，$n$ 为设定时间点数，c_{11}=0，n 不小于 2；c_{1n}表示第 n 个设定时间点处获取的舆情事件的传播速度，即符合目标主题的帖子数量在单位时间间隔内的新增量与单位时间间隔之比$(TP_n - TP_{n-1})/T$，TP_n表示第 n 个设定时间点爬取的符合目标主题的帖子数量。

$c_2 = [c_{21}, c_{22}, \dots, c_{2n}]$，$n$ 为设定时间点数，$c_{21} = 0$，$c_{22} = (TP_2 - TP_1)/T$，n 不小于 3；c_{2n}表示第 n 个设定时间点处获取的舆情事件传播的飙升度，即符合目标主题的帖子数量在第$n-1$个时间间隔内的新增量和在前一个时间间隔内的新增量之差与单位时间间隔之比$(TP_n - TP_{n-1}) - (TP_{n-1} - TP_{n-2}))/T$；

$c_3 = [c_{31}, c_{32}, \dots, c_{3n}]$，$n$ 为设定时间点数，$c_{31} = 0$，$c_{22} = (H_2 - H_1)/T$，n 不小于 3；c_{3n}表示第 n 个设定时间点处获取的舆情事件参与人群飙升度，即参与人数量在第$n-1$个时间间隔内的新增量和在前一个时间间隔内的新增量之差与单位时间间隔之比$((H_n - H_{n-1}) - (H_{n-1} - H_{n-2}))/\mathrm{T}$，其中$H_n$代表第 n 个设定时间点处的参与人数量；

$c_4 = [c_{41}, c_{42}, \dots, c_{4n}]$，$n$ 为设定时间点数，n 不小于 1；c_{4n}表示第 n 个设定时间点处获取的舆情事件话题倾向性比率，即爬取的帖子中负向帖子数量占符合目标主题的总帖子数量的比率TP_{ne}/TP；

$c_5 = [c_{51}, c_{52}, \dots, c_{5n}]$，$n$ 为设定时间点数，n 不小于 1；c_{5n}表示第 n 个设定时间点处获取的舆情事件平台活跃度比率，即爬取到符合目标主题的帖子数量超过设定值的平台的数量NE_0与总平台数量NE之比，为NE_0/NE；

$c_6 = [c_{61}, c_{62}, \dots, c_{6n}]$，$n$ 为设定时间点数，n 不小于 1；c_{6n}表示第 n 个设定时间点处获取的舆情事件话题敏感度比率，即为符合目标主题的帖子中的敏感帖子的数量占符合目标主题的总帖子数量之比TP_{se}/TP。

将指标体系应用于舆情数据集可以计算得到舆情特征集为防止大数据“吃掉”小数据，对数据集进行标准化，至此得到舆情数据的特征向量集合。

（二）排序学习算法的建立

根据监督形式的主曲线排序理论，利用贝塞尔曲线得到排序学习算法，排序学习算法的伪代码如下所示：

Input:

X：数据特征向量集合

ζ：很小的正数

Output：

P：排序主曲线的控制点

S：所有数据对象的相关性评分值

1．把数据特征向量的每个分量都归一化到$[0,1]$上

2．设定数据集的单调标识α

3．初始化控制点$P^{(0)}$

4．**while** $\Delta C > \zeta$ **do**

5．　　利用牛顿法由$P^{(t)}$确定相关性评分矩阵$s^{(t)}$

6．　　将$s^{(t)}$代入优化函数计算$P^{(t+1)}$

7．　　调整控制点，使其在超立方体内$(0,1)^d$

8．　　**if** $\Delta C < 0$ **then**

9．　　停止迭代

10．　　**end if**

11．**end while**

（三） 舆情事件的安全态势评分

本文根据网络舆情危机的严重程度、紧急程度、可控性和影响范围等因素，将预警等级设为五个等级：安全、轻警、中警、重警、巨警；分别用绿色、蓝色、黄色、橙色、红色信号表示。

根据模型的输出安全态势分数划定事件的等级：1—20 分为安全，此时要居安思危，防范于未然，实时监控网络信息；21—40 为轻警，分析引起危机的缘由。官方机构要实事求是地调查事情真相，并在第一时间向民众告知真相，联合各大新闻网站进行舆情疏导；41—60 为中警，此时严密监控网民情绪，并大幅度、频繁地进行新闻发布、媒体联动,不断地消除网民的疑惑、负面声音，掌握舆论并进行舆论引导；61—80 为重警，表明网络上的关于危机主体的观点大都是负面观点，此时相关部门要高度重视，采取一定的手段解除此时的危机状态，安抚民众此时高涨的情绪，避免转化成行为舆论如游行示威、罢工群斗、动乱等；81—100 为巨警，此时状态已达危机最高警戒，一不小心就面临崩溃，极有可能产生行为舆论、不可抑制的后果。此时相关的所有部门要联合起来，有效指挥，尽力挽回损失。最好的方法还是开诚布公、信息透明，让民众了解到事情真相，并把民众的情绪引导到关注好的另一方面，相关部门要彻查此事件，出台相关政策，彻实贯彻以人为本的方针，寻根溯源，标本兼治。

六、模型的评价及优化

针对问题 1，通过对附件 1 数据库文本聚类以及词向量训练，提供了一种针

对某个主题舆情信息的提取方法，可以筛选出相关性最强的文本信息。但是该算法存在运行速度较慢，模型训练时间长的问题，另外 Word2Vec 是一种静态的方式，虽然通用性强，但是无法针对特定任务做动态优化。针对训练时间较长，可以考虑添加负采样来减少模型的训练时间。

针对问题 2，通过本文的模型，在 web 上抓取了一系列数据，并提取了发表时间、评论人数、作者姓名及具体内容等具有深层分析价值的数据。不足之处是不同类型的网页之间差异性很大，抓取过程中会存在些许误差，所以在有条件的情况下，针对不同类型的网页时，最好能针对性地使用不同的方法会比较精确。

针对问题 3，提供了一种通过基于风格迁移的文本生成技术手段来引导网民情感倾向的方法，对处理公共危机舆情能发挥一定功能。然而该方法依赖于充足的语料才能生成有效的引导文本，因此从社会法律等多层面来同时引导舆情发展会更有效果。

针对问题 4，本文提供了一个舆情处理等级的划分方法，使用此模型可以定量地评判网络舆情的安全态势，建立预警模型。但是由于时间紧急，代码尚未完善好，只给出了伪代码，实现时必能提供一个强有力的舆情处理预警方法。

参考文献

[1] Mikolov T , Chen K , Corrado G , et al. Efficient Estimation of Word Representations in Vector Space[J]. Computer ence, 2013.

[2] Goodfellow I J , Pouget-Abadie J , Mirza M , et al. Generative Adversarial Networks[J]. Advances in neural information processing systems, 2014, 3:2672-2680.

[3] Blei D M , Ng A Y , Jordan M I , et al. Latent Dirichlet Allocation[J]. Journal of Machine Learning Research, 2003, 3:993-1022.

[4] 孙艳, 周学广, 付伟. 基于主题情感混合模型的无监督文本情感分析[J]. 北京大学学报(自然科学版), 2013, 49(1):102-108.

[5] 郝洁. 基于主题模型的文本情感分析研究[D].

[6] 崔雪莲, 那日萨, 刘晓君. 基于主题相似性的在线评论情感分析[J]. 系统管理学报, 2018, 27(05):24-30.

[7] 陈慧敏. 基于对抗训练的文本情感分析研究[D]. 2019.

[8] 冷永才. 基于深度学习的短文本情感分析算法研究[D]. 2018.

[9] 孙立伟, 何国辉, 吴礼发. 网络爬虫技术的研究%Research on the Web Crawler[J]. 电脑知识与技术, 2010, 006(015):4112-4115.

[10]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

Evaluate.py

```
from gensim.models.word2vec import Word2Vec
import math
import json
import numpy as np
import time
from read_file import *
import os
import jieba
import jieba.posseg as jp
from gensim import corpora, models
from read_file import *
from word2vec_2 import doc_vec
from similarity_1 import *
import csv
import codecs
# Global Dictionary


# ------------构建语料库--------------#
path = 'data/data_A.csv'
yuliao_path_train = 'data/yuliao_train.txt'
yuliao_path_test = 'data/yuliao_test.txt'
documents =    read_file(path).tolist()
new_words = ['奥预赛', '折叠屏']   # 新词
stopword_file_path = 'stopword.txt'
stopwords = stopword_file(stopword_file_path)
synonyms = {'韩国': '南朝鲜', '传言': '流言'}   # 同义词
stopwords.append('\n')
topK = 20

def remove_stopwords(ls):    # 去除停用词
      return [word for word in ls if word not in stopwords]


def replace_synonyms(ls):    # 替换同义词
      return [synonyms[i] if i in synonyms else i for i in ls]

path = 'data/data_A.csv'
documents =    read_file(path).tolist()
isExists = os.path.exists(yuliao_path_train)
if not isExists:
```

```
    words_ls = []
    print('开始构建语料库...')
    # with open(yuliao_path_train,'w') as f:
    memory_0 = {}
    for i,text in enumerate(documents):
        words = replace_synonyms(remove_stopwords([w.word for w in jp.cut(text)]))
        words_ls.append(words)
        memory_0[text] = words
        if i % 50 == 0:
            print('#:' + str(i) + '/' + str(len(documents)))
    write_file(yuliao_path_train,words_ls)
    memory_1,memory_2 = doc_vec()
    memory_3 = cal_similarity()
    sen2score = {}
    for key in memory_0.keys():
        sen2score[key] = memory_3[memory_1[memory_0[key]]]

else:
    n_dim=300 #向量维度

    print('语料库已经存在')
    memory_0 = {}
    memory_00 = {}
    with open(yuliao_path_train,'r') as f: #读数据
        train_data_list=f.read().strip().split('\n')
    print('构建 memory_0...')
    for i,sen in enumerate(documents):
        memory_0[train_data_list[i]] = sen
        memory_00[sen] = train_data_list[i]
        if i % 10000 == 0:
            print('#:' + str(i) + '/' + str(len(documents)))
    print('memory_0 构建完毕')

    memory_1,memory_2 = doc_vec(yuliao_path_train,'data/senvec300.json')
    memory_3 = cal_similarity([' 汽车',' 性价比','年轻人'])
    rank_senvec = {}
    sen2score = []
    print('构建 sen2score...')
    for key in memory_3.keys():
        sen2score.append([memory_0[key],memory_3[key]])
    print('sen2score 构建完毕')


    sen2score = sorted(sen2score,key=lambda x:x[1],reverse=True)
```

```
for i,sen in enumerate(sen2score[:topK]):
    tmp1 = memory_00[sen[0]]
    rank_senvec[sen[0]] = memory_2[tmp1]

sen_test = ['福克斯汽车的性价比太低了，不如选择购买奥迪，奥迪才是年轻人应该开的品牌','多吃蔬菜有益身体健康']
#把 rank_sentence 和 sen_test 分别分词

words_ls = []
print('开始对测试集分词...')
# with open(yuliao_path_train,'w') as f:
memory_0 = {}
for i,text in enumerate(sen_test):
    words = replace_synonyms(remove_stopwords([w.word for w in jp.cut(text)]))
    words_ls.append(words)
    # memory_0[words[0]] = text
    if i % 50 == 0:
        print('#:' + str(i) + '/' + str(len(sen_test)))
print(words_ls)
w2v_model = Word2Vec.load('model/w2v_model_300.pkl')  # 加载训练好的 Word2Vec 模型
vector_dt=[] #句子：句子向量
print('开始构建文本对应的向量...')
for i,index in enumerate(words_ls):
    senvec=build_sentence_vector_weight(words_ls[i], n_dim, w2v_model)
    vector_dt.append(senvec.tolist())
    if i % 1000 == 0:
        print('#:' + str(i) + '/' + str(len(words_ls)))
print('文本向量构造完毕')
# print(vector_dt)
dist = []
for vec in vector_dt:
    dist.append(np.sum(cosine_similarity(vec, list(rank_senvec.values())[0])))
print(dist)


Word2vec.py
from gensim.models.word2vec import Word2Vec
import math
import json
import numpy as np
import time
import os
```

```
    # 构建 word2vec 模型，词向量的训练与生成
    def get_dataset_vec(dataset,n_dim):
        w2v_model = Word2Vec(dataset, sg=1, size=n_dim, min_count=0, hs=1,iter=10)  # 初始化模型并训练
        w2v_model.save('model/w2v_model_300.pkl')  # 保存训练结果

    # 对每个句子的所有词向量取均值，来生成一个句子的 vector
    def build_sentence_vector_weight(sentence, size, w2v_model):

        sen_vec = np.zeros(size).reshape((1, size))
        count = 0

        for word in sentence:
            try:
                sen_vec += w2v_model[word].reshape((1, size))
                count += 1
            except KeyError:
                continue
        if count != 0:
            sen_vec /= count
        return sen_vec


    # 将文本数据转换为文本向量
    def doc_vec(path_yuliao,path_senvec):
        memory = {}
        n_dim=300 #向量维度

        with open(path_yuliao,'r') as f: #读数据
            train_data_list=f.read().strip().split('\n')

        #将数据转换为[[词，词，词],[词，词，词],[词，词，词]]形式
        train_data=[]
        print('构造 train_data_list...')
        for i,sen in enumerate(train_data_list):
            memory[str(sen.split(','))] = sen
            train_data.append(sen.split(','))
            if i % 1000 == 0:
                print('#:' + str(i) + '/' + str(len(train_data_list)))
        print('train_data_list 构造完毕')

        # print (train_data)
        isExists = os.path.exists('model/w2v_model_300.pkl')
        if not isExists:
```

```
            get_dataset_vec(train_data,n_dim) #训练模型
            print('模型训练完毕')
        else:
            print('加载已有模型')

        w2v_model = Word2Vec.load('model/w2v_model_300.pkl')    # 加载训练好的Word2Vec 模型
        #print(model.wv.index2word())

        isExists = os.path.exists(path_senvec)
        if not isExists:
            vector_dt={} #句子：句子向量
            print('开始构建文本对应的向量...')
            for i,index in enumerate(train_data):
                # print(index)

                senvec=build_sentence_vector_weight(train_data[i], n_dim, w2v_model)
                vector_dt[train_data_list[i]]=senvec.tolist()
                if i % 1000 == 0:
                    print('#:' + str(i) + '/' + str(len(train_data)))
            print('文本向量构造完毕')

            #将句子及其向量以字典的形式存入 json 文件
            with open(path_senvec,'w') as f:
                json.dump(vector_dt, f)
        else:
            print('加载已有词向量')
            with open(path_senvec, 'r') as f:
                vector_dt = json.load(f) #将所有句子及其向量下载下来
        return memory,vector_dt
```

Similarity.py

```
#!/usr/bin/env python
# -*- coding:utf-8 -*-
from gensim.models.word2vec import Word2Vec
import json
from sklearn.metrics.pairwise import cosine_similarity
import numpy as np
from gensim.models.doc2vec import Doc2Vec, TaggedDocument
# import jieba
#  加载训练好的模型
def build_sentence_vector_weight(sentence, size, w2v_model):
    sen_vec = np.zeros(size).reshape((1, size))
    count = 0
```

```
    for word in sentence:
        try:
            sen_vec += w2v_model[word].reshape((1, size))
            count += 1
        except KeyError:
            continue
    if count != 0:
        sen_vec /= count
    return sen_vec

def cal_similarity(keyword):
    w2v_model = Word2Vec.load('model/w2v_model_300.pkl')  # 加载训练好的 Word2Vec 模型
    with open('data/senvec300.json', 'r') as f:
        vector = json.load(f) #将所有句子及其向量下载下来
    print(len(vector))
    keyword=build_sentence_vector_weight(keyword,300,w2v_model) #获得每个关键词描述的向量
    threshold=0.6 #语义距离阈值
    dt={} #保存与关键词语义距离符合要求的句子及对应的距离，并存入 json 文件
    idx = 0
    print('开始计主题算关键词与文本之间的相似度...')
    for key in vector.keys():
        dist = np.sum(cosine_similarity(keyword, vector[key]))
        if dist>threshold:
            dt[key]=dist
        idx += 1
        if idx % 100 == 0:
            print('#:' + str(idx) + '/' + str(len(vector)))
    print('相似度计算完毕')
    f=open("result/result.json", mode="a", encoding="utf-8")
    json.dump(dt,f)
    return dt

read_file.py
import pandas as pd
import numpy as np
from gensim.corpora import WikiCorpus
# import zhconv
# import jieba

def read_file(path):
    data_A = pd.read_csv(path,delimiter="\t")
    data_A = data_A.values
```

```
    data_B = np.unique(data_A)
    return data_B
def write_file(path,data):
    with open(path,'w') as f:
        for ele in data:
            f.write(str(ele).replace('[','').replace(']','').replace('\'','')+'\n')

def stopword_file(path):
    list_ = set()
    f = open(path)
    data = f.read().splitlines()
    f.close()
    return data
```