新冠肺炎疫情下网民情绪识别研究

摘要：新型冠状病毒肺炎疫情是影响世界的重大事件，大众时刻关注媒体对于疫情的报道，其中微博是重要的社会媒体，微博的内容和评论反映了网民对于疫情的态度和情绪，因而对微博内容进行情感分析对于管理网络舆情具有重要的作用。本文对于疫情期间的微博内容进行定量分析，采用 BERT 模型，将微博内容进行三分类，分别为积极、中性和消极，相比于二分类模型能更精准地把握民众的情绪，然后应用模型对未标注的数据进行预测，分析疫情期间网民的情绪变化趋势，经历潜伏、爆发、到稳定，不同的时期有不同的特征。同时，本文还基于情感分析技术，结合 LDA 模型分析型疫情关键时刻网民情绪的主题，进而帮助政府更准确地把握社会舆情，科学做好宣传防疫和引导舆情的工作，使社会达成共识，增强民族凝聚力。

关键词：网络舆情；情感分析；BERT；LDA 主题模型

0 引言

新冠肺炎疫情的防控，是一个具有世界影响力的事件，其中有着真相事实的碰撞，也有着各种诉求的碰撞、各种情绪的碰撞，如何疏导大众的情绪，形成共识，就需要深刻把握大众的情绪，在舆论上寻找同理心，引发共情，提出建设性的方向。据报告，截至 2020 年 3 月，我国网民达 9.04 亿人，互联网普及率为 64.5%[1]。而微博作为一个分享实时信息的网络平台，具有实时性，便捷性和传播性的特点，有着庞大的用户群体。截至 2019 年 12 月，微博月活跃用户达 5.16 亿[2]。这些用户既是信息的接收者也是传播者，他们在微博上发表自己关于新冠疫情的看法，其中不乏负面观点，而这有可能会对疫情防控造成负面的舆论影响，所以有关部门需要关注网民的情绪，引导舆论向积极的方向

发展，为疫情防控做好坚强的后盾。目前关于疫情舆情方面的研究较少，基于舆情文本的质性分析，邢鹏飞等[3]研究了舆情的形成机制和引导策略，他们认为，疫情舆情体系由舆情主体、舆情中介、舆情客体及政府力量组成，舆情经历潜伏、爆发、平稳及消散这四个阶段。本文通过定量建模的方法，讨论疫情期间民众情绪倾向变化和不同情绪的主题，能更精准地把握民众的情绪，科学地引导舆情的工作，增强对握舆情走向的把握力度，使舆论体系良性发展。

情感分析[4]是指将主观的文本情感内容进行分类，文本可以划分为积极和消极两类，或者积极、消极和中性的多类。本文主要讨论对微博内容进行情绪识别，将其分为积极、中性和消极三类，并分析不同情绪的主题，充分了解社会心态与社会情绪，相关部门可以在调研的基础上根据民意进行科学决策，推动公共决策与执行。

1 相关研究

情感分析是自然语言处理（natural language processing，NLP）的一项任务，自然语言处理专注于如何让计算机处理和分析人类的自然语言数据。情感分析的方法主要可以分为：基于词典的情感分析，基于机器学习的情感分析和基于深度学习的情感分析。基于词典的方法依赖构建的情感词典，对文中包含在词典中的单词按照规则进行计算，得出情感得分。基于机器学习的方法需要对文本进行特征构造，再结合机器学习模型进行分类。

近年来深度学习方法在自然语言处理方面取得极大的进展，尤其是词嵌入[5]和深度学习方法[6]的成功。Moraes 等[7]用人工神经网络（ANN）对文档进行情感分类，取得比支持向量机（SVM）更优的结果；Glorot 等[8]研究情感分类的领域适应问题，提出一种基于堆叠降噪自编码器的深度学习系统，可以使用已标记和未标记的数据进行无监督文本特征提取。Johnson 和 Zhang[9]提出一个名为 BoW-CNN 的卷积神经网络（CNN）变体，该变体在卷积层中采用了词袋转换，还设计一种称为 Seq-CNN 的新模型，通过连接多个单词的独热向量来获得单词的顺

序信息。Tang 等[10]提出了一种神经网络，首先用 CNN 或长短期记忆网络（LSTM）学习句子表示，然后用门控循环单元（GRU）对句子语义及固有关系进行自适应编码，以进行情感分类。Xu 等[11]提出了一种缓存的 LSTM 模型，以捕获长文本中的整体语义信息，模型中遗忘率低的内存组能够捕获全局语义特征，遗忘率高的组学习局部语义特征。Zhou 等[12]设计一种基于注意力的 LSTM 网络，用于跨语言情感分类，它可以有效地将情感信息从资源丰富的语言（英语）调整为资源贫乏的语言（中文），有助于提高情感分类性能。Li 等[13]提出了一种跨域情感分类的对抗记忆网络，来自源域和目标域的数据被一起建模。2018 年是自然语言处理的转折点，谷歌发布预训练模型 BERT[14]（Bidirectional Encoder Representations from Transformers），横扫 11 项 NLP 任务，使用多层 Transformer 结构[15]能够捕捉语句中的双向关系，理解上下文，本质是在海量语料上用自监督的方法为单词学习一个好的特征表示。

国内该领域的研究也比较活跃，张海涛等[16]应用卷积神经网络对微博舆情进行情感分析，发现比传统机器学习方法更优。张英[17]对微博短文本利用双向长短时记忆网络（BLSTM）进行情感分析，发现效果比 CNN 和 LSTM 更好。廖海涵等[18]应用 LDA 主题模型分析舆情传播周期中不同的主题结构和观点。任中杰等[19]对天津 812 事件用朴素贝叶斯进行情感分类，分析各阶段情感倾向演化。崔彦琛等[20]应用情感词典和时间序列分析对杭州保姆纵火案的进行实证分析，对事件不同阶段日均情感值进行计算。孙靖超等[21]针对舆情特点对 BERT 模型进行优化，使其能在小规模数据集上不过拟合。谌志群等[22]结合 BERT 和 BLSTM 模型对微博评论做倾向性分析，优于主流的模型。

通过对相关研究的分析发现，对舆情的研究结合感情分类模型与 LDA 主题模型的研究较少，大部分对舆情的情感分析是进行二分类，分为积极和消极，主要着眼于提升模型的准确性。本文考虑应用 BERT 模型对微博内容进行三分类，再利用 LDA 主题模型深入分析情感的主题，这样对于民众情绪可以有更准确的把握。

2 情感分类与主题挖掘模型

2.1 BERT 模型

许多深度学习方法需要词嵌入结果作为输入，词嵌入将一个维数为所有词数量的高维空间嵌入到一个低维的连续向量空间中，语义相似的词在空间中的距离也比较近，但不能共享上下文信息。word2vec、Glove 可以考虑上下文语境，但无法解决一词多义的问题。引入 Contextual Word Embedding，这种无监督学习考虑上下文，BERT 能学到一个词在不同上下文不同语义的表示方法。

BERT 的架构为 Transformer 中的 Encoder，Transformer 实际上是一种基于自注意力机制（Self-Attention mechanism）的 Seq2Seq 模型，自注意力机制解决了 RNN 不能并行运算的问题，RNN 需要按顺序通过所有的输入，才能得出输出，不容易平行化计算，而 self-attention 可以取代 RNN，对于输入x^i，$a^i = Wx^i$，$query: q^i = W^q a^i$，$key: k^i = W^k a^i$，$value: v^i = W^v a^i$，拿每个$query$和key做 attention，$a_{1,i} = q^1 \cdot k^i / \sqrt{d}$，$d$是$k$的维度，对其进行归一化$\hat{a}_{1,i} = \exp(a_{1,i}) / \sum_j \exp(a_{1,j})$。最终得到输出$b^1 = \sum_i \hat{a}_{1,i} v^i$，$b^i$以此类推。设$I = \{a^1, a^2, \dots, a^n\}$，$O = \{b^1, b^2, \dots, b^n\}$，以上的运算用矩阵表示为$Q = W^q I$，$K = W^k I$，$V = W^v I$，$\hat{A} = softmax(softmax(\frac{QK^T}{\sqrt{d_k}})V)$，$O = V\hat{A}$，可以并行计算。Encoder 由 N 个相同的层组成，每层由两个子层，多头自注意力（multi-head self-attention mechanism）和全连接前向神经网络（fully connected feed-forward network）组成，每个子层都加了入残差连接（residual connection）和层归一化（normalisation），因此可以将子层的输出表示为：$sub\_layer\_output = LayerNorm(x + (SubLayer(x)))$。multi-head attention 通过 h 个不同的线性变换对 Q，K，V 进行投影，最后将结果拼接：$MultiHead(Q, K, V) = Concat(head_1, \dots, head_h)W^O$，$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$，self-

attention 取 Q，K，V 相同。attention 的计算方式为$Attention(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$。

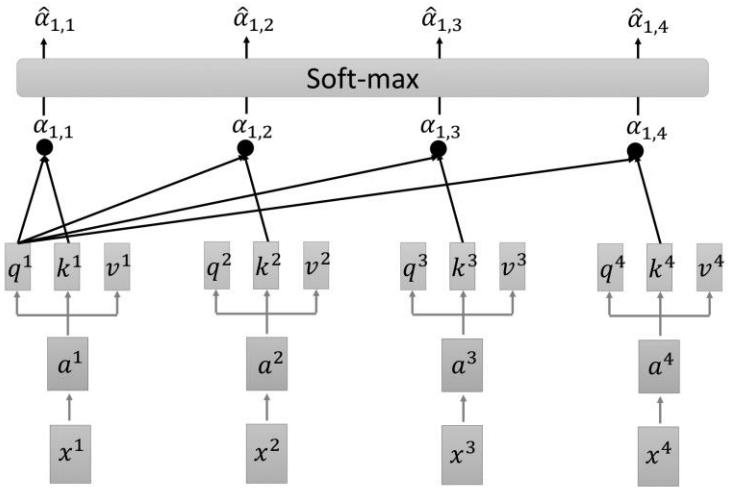

图 1　self-attention 示意图[1]

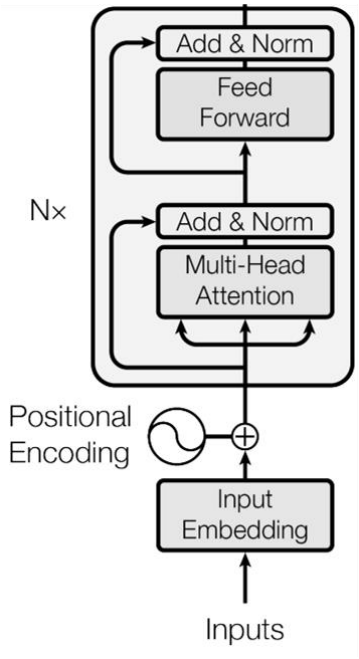

图 2　Transformer 的 Encoder[2]

[1] 图片来源：李宏毅. BERT 讲义. https://speech.ee.ntu.edu.tw/~tlkagk/courses/ML_2019/Lecture/BERT%20(v3).pdf. (2020-05-14).

[2] 图片来源：Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need. Advances in neural information processing systems. 2017: 5998-6008.

BERT 的输入，由词向量（Token Embeddings）、段向量（Segment Embeddings）和位置向量（Positional Embeddings）相加产生，Token Embeddings 第一个单词是 CLS 标志，可以用于之后的分类任务，标记[SEP]是用来分隔两个句子，段向量用来区别两种句子。

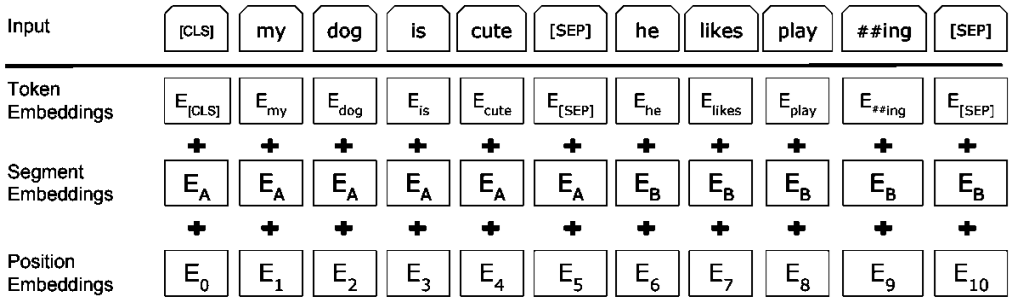

图 3　BERT 的输入[3]

预训练语言模型时，有两个任务：Masked Language Model，随机掩盖一些（15%）词汇，再去预测这些词汇，最终的损失函数只计算被掩盖的词汇；Next Sentence Prediction，二分类任务，输入 A、B 两个句子，判断 B 是否是 A 后面的句子，这样能提前学习到句子之间的关系。

在已经训练好的语言模型的基础上，对模型进行 fine-tuning，对于分类问题在模型基础上加一层 softmax 网络，然后在新的语料上重新训练。

2.2 LDA 主题模型

LDA[23] (Latent Dirichlet Allocation，隐含狄利克雷分布)是一种主题模型，假设每篇文章由数个主题组成，每个主题可以使用数个词来描述，相同的词可同时出现在不同的主题中。LDA 是一种无监督算法，仅需要文档集和指定的主题数 k。LDA 是典型的词袋模型，对于一篇文档，仅考虑一个词是否出现，不考虑出现的次序。

[3] 图片来源：Devlin J, Chang M W, Lee K, et al. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805, 2018.

LDA 假设人们在撰写文本时，有以下生成机制：（1）决定第i篇文本的用词数N_i；（2）决定第i篇文本在k个主题各自出现的概率$\boldsymbol{\theta}_i = [\theta_1, \dots, \theta_k]$；（3）撰写第$i$篇文本第$j$个用词$w_{ij}$时，先选一个对应的主题$z_{ij} = t$，根据主题$t$挑选一个可以描述该主题的词汇$w_{ij}$，对于第$t$个主题，词库中词汇出现的概率为$\boldsymbol{\phi}_t = [\phi_1^t, \dots, \phi_V^t]$。

有了文本生成机制，通过观察大量文本，估计文本生成的联合概率分布，使用先验分布来描述以上文本生成机制，见图 4，灰色圆圈表示可观察的文本，白色圆圈表示隐含变量(Latent Allocation)，文本i第j个用词在各主题出现次数z_{ij}服从多项式分布θ_i，多项式分布表示撰写一个用词，该用词是主题t的概率为θ_t。第i篇文本k个主题出现的概率θ_i服从狄利克雷分布α，狄利克雷分布是多项式分布的共轭先验概率分布，θ_i根据观察到的信息更新后，其后验分布仍然是狄利克雷分布。给定主题$z_{ij} = t$，第i篇文本第j个用词w_{ij}在词库中词汇出现的概率w_{ij}服从多项式分布$\boldsymbol{\phi}_t$，表示撰写一个用词，结果是词v的概率为$\boldsymbol{\phi}_v^t$。第t个主题每个用词出现的概率$\boldsymbol{\phi}_t$服从狄利克雷分布β。所有可见变量及隐藏变量的联合分布$p(w_i, z_i, \theta_i, \phi|\alpha, \beta) = \prod_{j=1}^{N} p(\theta_i|\alpha)p(z_{ij}|\theta_i)p(\phi|\beta)p(w_{ij}|\phi_{z_{ij}})$，一篇文档的词语分布的最大似然估计可以通过将上式的θ_i以及$\boldsymbol{\phi}$进行积分和对z_i进行求和得到$p(w_i|\alpha, \beta) = \int_{\theta_i} \int_{\Phi} \sum_{z_i} p\,(w_i, z_i, \theta_i, \phi|\alpha, \beta)$，可以通过吉布斯采样估计模型中的参数。

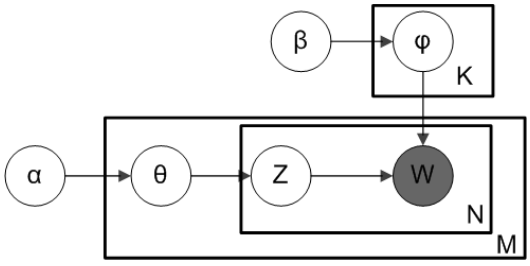

图 4　LDA 模型[4]

[4] 图片来源：Blei D M, Ng A Y, Jordan M I. Latent dirichlet allocation. Journal of machine Learning research, 2003, 3(1): 993-1022.

3 实验与结果分析

数据为 100 万条与疫情相关的微博[24]，从 2020 年 1 月 1 日到 2 月 20 日，其中 10 万条有标签，标签分为三类，分别为：1（积极），0（中性）和-1（消极），用这些有标签的数据建立 BERT 中文模型，训练集占 80%，测试集占 20%，对比 BERT-wwm[25]，基于全词遮罩（Whole Word Masking）的中文预训练模型，ERNIE[26]，百度提出知识增强的语义表示模型，以及 ERNIE-tiny[27]，通过模型结构压缩和蒸馏将 ERNIE 2.0 Base 模型进行压缩得到的轻量级预训练模型。模型评价采用加权精确率、加权召回率和加权 F1。

表 1 微博内容示例

时间	微博中文内容	情感倾向
2020.1.1	肺炎好可怕	-1
2020.1.1	不明原因肺炎患者转入传染病医院什么是病毒性肺炎？病毒性肺炎是由上呼吸道病毒感染，向下蔓延引起肺组织炎症…	0
2020.1.1	“新年快乐！2020 中国加油，香港加油！”今晨，一批爱国爱港人士齐聚香港金紫荆广场，参加 2020 年第一场升旗仪式，共同祝福祖国、祝福香港。	1

本文模型建立在 paddlepaddle 深度学习框架上。微博限制文本的长度为 140 个字，因此最大序列长度设为 140，学习率设为 5e-5，权重衰减设置为 0.01，训练预热的比例设置为 0.1。

评价指标考虑到样本不均衡的问题，各个类别的精确率（Precision）与 找回率（Recall）要乘该类在总样本中的比例再求和，$Precision_i = \frac{TP_i}{TP_i+FP_i}$，$Precision_{weighted} = \sum_{i=1}^{L} Precision_i \times w_i$，$Recall_i = \frac{TP_i}{TP_i+FN_i}$，$Recall_{weighted} = \sum_{i=1}^{L} Recall_i \times w_i$，$F1_i = \frac{2Precision_i Recall_i}{Precision_i+Recall_i}$，$F1_{weighted} = \sum_{i=1}^{L} F1_i \times w_i$ 。

表 2 情感分类模型的实验结果

模型	$Precision_{weighted}$	$Recall_{weighted}$	$F1_{weighted}$
BERT	0.7615	0.7627	0.7602
BERT-wwm	0.7401	0.7421	0.7347
ERNIE	0.7595	0.7575	0.7512
ERNIE-tiny	0.7544	0.7575	0.7528

表 3 BERT 模型测试集结果

实际\预测	-1	0	1
-1	2131	1158	99
0	931	9713	866
1	129	1559	3391

由表 2 可知 BERT 模型的精确率、召唤率和 F1 比其他三个模型高，再分析 BERT 模型在测试集上的结果，由表 3 可以看出模型容易将积极、消极的样本分类为中性，这可能与样本不均衡有关。总体而言，模型虽然对积极、消极样本有区分能力，但性能仍有提升空间。

将模型应用在未标注的数据上，然后观察随时间变化，不同情绪倾向的微博数量以及各情绪倾向所占的比例，见图 5 和图 6。结合图 7 新冠肺炎每日新增数，总体而言，中性的微博内容始终占大多数，占比在 60%左右，其次是积极的内容，占比基本在 20%以上，消极的内容数量占比最少。从 2020 年 1 月 1 日至 1 月 18 日，有关疫情的微博数量较少，消极情绪倾向占比较低，基本在 10%以下；1 月 19 日至 1 月 23 日，由于新冠肺炎每日新增患者从之前的 0 例达到 59 例，并且逐渐增加，相关微博数量激增，消极情绪数量在 23 日达到一个小高峰，将近 5000 条，占比超过 20%，这个阶段是唯一出现消极情绪占比比积极情绪占比更多的时刻，说明民众出现了明显的恐慌；之后消极情绪开始回落，但由于在 1 月 30 日 31 日每日新增患者在 2000 人左右，消极情绪在这两天有所升高，随后又回落；2 月 4 日 5 日新增人数是当时最多的两天，2 月 6 日的消极情绪占比是最高的，之后开始回落；由于改变新冠的诊断标准，2 月 16 日

新增患者数量将近 2 万，但发现消极情绪占比与之前相比没有变化，说明民众并没有出现大规模的恐慌；2 月 20 日的微博数量较少，出现消极比例高的情形。

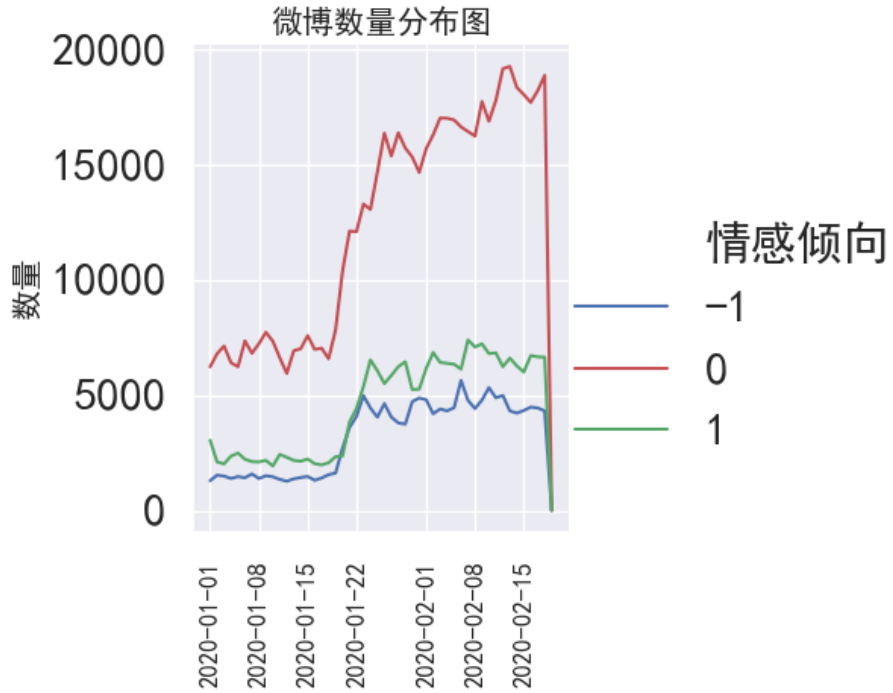

图 5 不同情感倾向的微博数量随时间的分布图

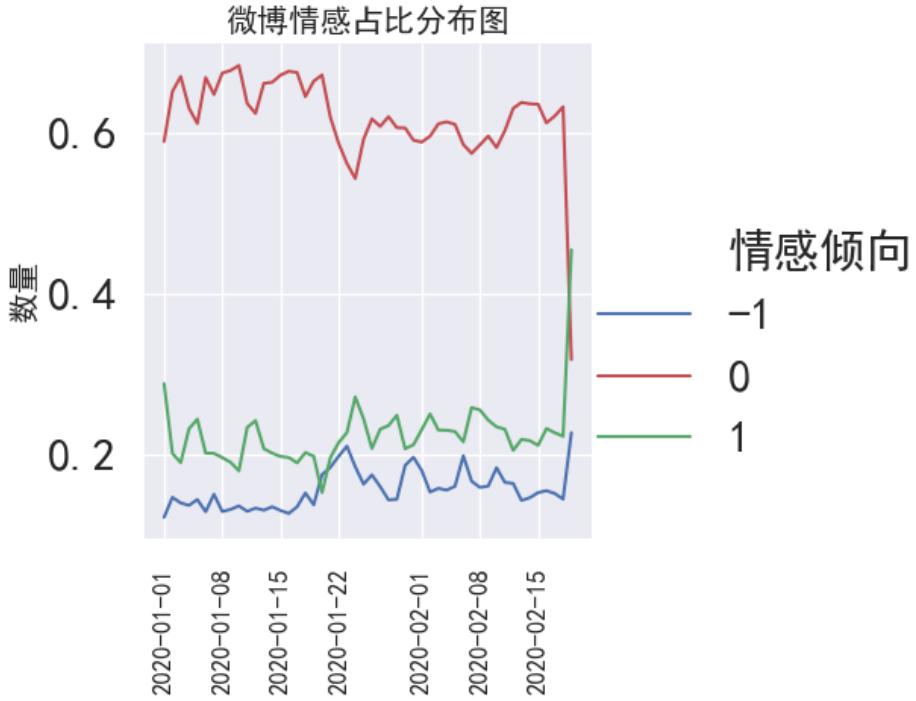

图 6 不同情感倾向微博所占的比例随时间的分布图

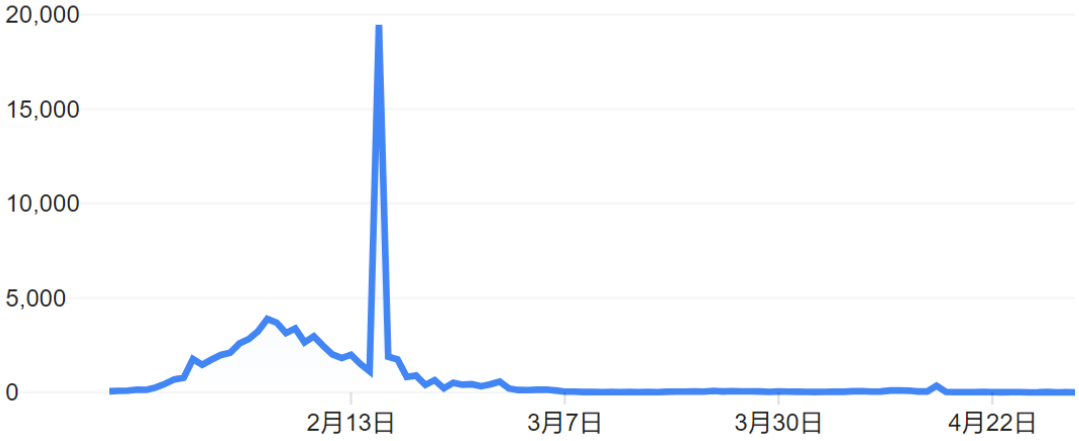

图 7 中国新冠肺炎每日新增数

对出现频率最高的 120 个词用 BERT 词向量表示，然后进行聚类，结果见图 8，上方是关于疫情的词，语义相近的词之间的距离也比较近。分别选取 1 月 1-2 日，1 月 17-18 日，1 月 23 -24 日，2 月 5-6 日，2 月 16-17 日，对不同情绪倾向的微博内容做主题分析，挖掘文本的主题，结果见表 4。

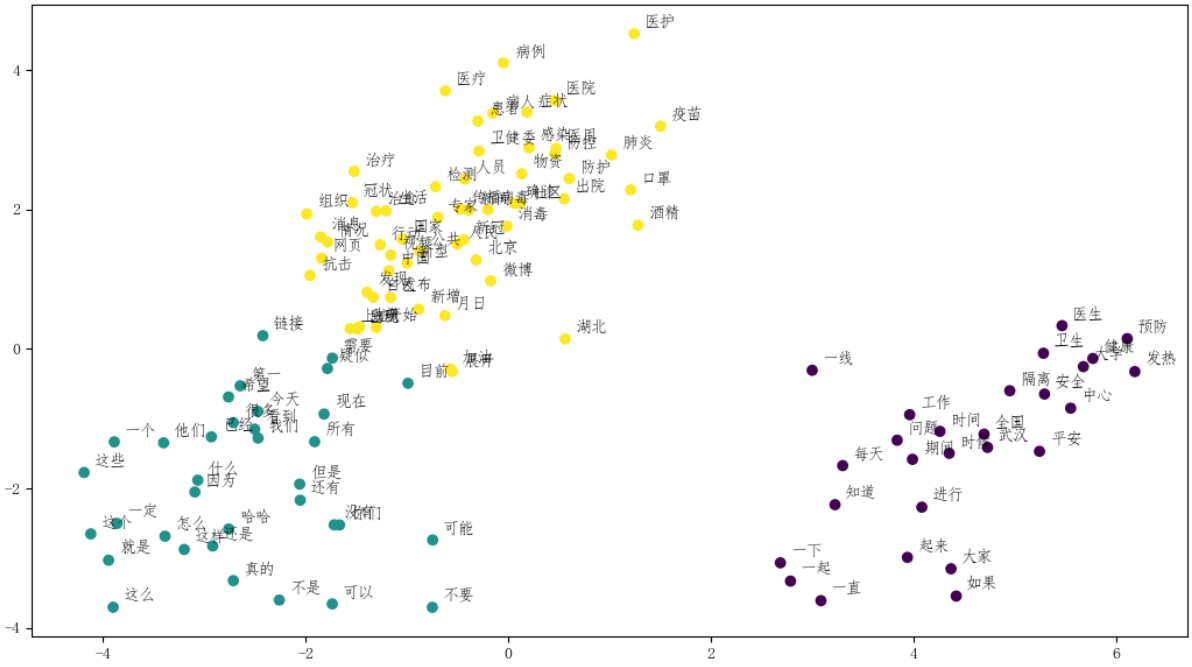

图 8　BERT 词向量聚类结果

表 4 不同时期各情感倾向的文本主题

日期	消极	中性	积极
1.1-1.2	医生 发烧 知道 第一医院	肺炎 武汉	新年 加油 哈哈 快乐 一起 中国 视频 心愿 希望 第一 生活 致敬
1.17-1.18	武汉 肺炎 新型 医院 口罩 病毒 冠状 就是 医生 知道	冠状 新型 武汉 口罩 疫情 病毒 日月 感染 肺炎 病例	希望 致敬 一定 武汉 平安 人员 加油 爱心 丝绸 发福 全世界 可爱 中国 哈哈 努力 一起 非典 注意 就是
1.23-1.24	医生 医院 野味 新型 他们 知道 病例 病毒 就是 武汉 确诊 口罩 肺炎 疫情	医生 医院 野味 新型 他们 知道 病例 病毒 就是 武汉 确诊 口罩 肺炎 疫情	致敬 人员 医护 新型 希望 你们 一起 一定 一线 健康 病毒 医院 感谢 医生 全国 努力 支援 快乐 平安 新年
2.5-2.6	医生 口罩 病毒 李文亮 医院 你们 去世 武汉 就是 感染 确诊 疫情 肺炎	疫情 肺炎 冠状 感染 武汉 医院 确诊 防控 月日 患者 工作 病毒 人员 口罩 视频 新型 病例	加油 致敬 一起 努力 抗击 希望 一线 哈哈 你们 一定 视频 奋战 勇战 肖战 患者 抗疫 辛苦 口罩 感谢 医护
2.16-2.17	病毒 口罩 医院 确诊 武汉 感染 疫情	疫情 病毒 肺炎 新冠 确诊 病例 患者 月日 中国 冠状 口罩 人员 视频 医院 感染 武汉 工作 防控 新型 治疗	加油 中国 一起 肺炎 医院 感谢 你们 湖北 平安 抗击 希望 俊凯 防控 保重 王俊凯 起来 白衣 消息 他们 医护

从表 4 中可以看出，消极和中性的主题词集中在疫情上，消极的主题词最少，只与疫情有关，中性的主题集中在疫情的报道与防控工作上，与积极的主题词区分非常明显，说明情感分类模型是有区分能力的。一月初时相关的主题词较少，主要是积极的新年祝福，中性的主题词只有武汉和肺炎；1 月 17-18 日，明确了病原体后，消极和中性的文本主题词出现了新型、冠状、病毒、口

罩，说明民众开始意识到威胁，有了防护措施；1 月 23-24 日，野味、确诊、疫情出现在消极和中性的主题中，疫情变得严峻，民众对于食野味行为感到愤怒，此时是新年，积极的主题为对同胞的祝福，对一线医护人员的致敬；2 月 5-6 日，消极的主题有李文亮，李文亮医生病危引发民众的关注，中性的主题出现防控，积极的主题出现抗击、抗疫，民众对于抗疫充满希望；2 月 16-17 日，消极的主题词汇减少，积极的主题为对医护人员的感谢与支持。

结合可以不同时间的情绪占比，可以看出 1 月 1 日至 1 月 18 日是舆论的潜伏期，相关微博数量较少，民众消极情绪占比低；1 月 19-1 月 23 日是爆发期，相关微博数量与消极情绪占比呈直线上升，民众出现大规模恐慌；1 月 24-2 月中旬，为平稳区，消极情绪占比逐渐下降，中性情绪占比上升，说明民众逐步回归理性。

4 结束语

微博是公众广泛使用的媒体，微博的内容对于公众有着巨大的影响力。本文对 2020 年 1 月 1 日至 2 月 19 日与疫情相关的微博内容，建立 BERT 模型进行情感分析，对文本内容进行三分类，相比于二分类，能更细致地反映公众的情绪，分析了疫情在爆发前与爆发中民众的情绪倾向变化，以及结合 LDA 主题模型深入分析关键时点情感倾向的主题。疫情引发了大量的社会情绪，不同阶段民众的情绪和主题存在差异，了解舆论情势的变迁，公众情绪、观念的变化对于抗击疫情有着重要作用，抗疫是举全国之力的战役，考验一个社会的凝聚力，了解民众的情绪，并且进行有效的舆论引导，汇聚情感，梳理共识，导向建设性方向。本文的研究存在一定的局限性，数据的时间跨度不够大，模型的准确性也有待提高，在今后的工作中将对这些问题做出改进，期望获得更准确全面的情感分析模型。

参考文献

[1] 中国互联网络信息中心, 第 45 次《中国互联网络发展状况统计报告》, http://www.cac.gov.cn/2020-04/27/c_1589535470378587.htm, (2020-05-14).

[2] 微博, 微博 2019 年报, http://ir.weibo.com/static-files/9537b1c6-da60-4bd4-8c66-9393d72ca7a3, (202-05-14).

[3] 邢鹏飞,李鑫鑫, 重大疫情防控中网络舆情形成机制及引导策略研究——基于新冠肺炎疫情期间网络舆情文本的质性分析, http://kns.cnki.net/kcms/detail/61.1167.G3.20200422.1511.002.html, (2020-05-26).

[4] 宗成庆, 统计自然语言处理, 北京:清华大学出版社, 2013:8.

[5] Mikolov T, Sutskever I, Chen K, et al, Distributed representations of words and phrases and their compositionality, Advances in neural information processing systems, 2013: 3111-3119, 2013.

[6] Socher R, Perelygin A, Wu J, et al, Recursive deep models for semantic compositionality over a sentiment treebank, Proceedings of the 2013 conference on empirical methods in natural language processing, 2013: 1631-1642, 2013.

[7] Moraes R, Valiati J O F, Neto W P G O, Document-level sentiment classification: An empirical comparison between SVM and ANN, Expert Systems with Applications, 40(2): 621-633, 2013.

[8] Glorot X, Bordes A, Bengio Y, Domain adaptation for large-scale sentiment classification: a deep learning approach, Proceedings of the 28th International Conference on International Conference on Machine Learning, 2011: 513-520, 2011.

[9] Johnson R, Zhang T, Effective use of word order for text categorization with convolutional neural networks, arXiv preprint, arXiv:1412.1058, 2014.

[10] Tang D, Qin B, Liu T, Document modeling with gated recurrent neural network for sentiment classification, Proceedings of the 2015 conference on empirical methods in natural language processing, 2015: 1422-1432, 2015.

[11] Xu J, Chen D, Qiu X, et al, Cached Long Short-Term Memory Neural Networks for Document-Level Sentiment Classification, Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, 2016: 1660-1669, 2016.

[12] Zhou X, Wan X, Xiao J, Attention-based LSTM network for cross-lingual sentiment classification, Proceedings of the 2016 conference on empirical methods in natural language processing, 2016: 247-256, 2016.

[13] Li Z, Zhang Y, Wei Y, et al, End-to-end adversarial memory network for cross-domain sentiment classification, Proceedings of the 26th International Joint Conference on Artificial Intelligence, 2017: 2237-2243, 2017.

[14] Devlin J, Chang M W, Lee K, et al, Bert: Pre-training of deep bidirectional transformers for language understanding, arXiv preprint, arXiv:1810.04805, 2018.

[15] Vaswani A, Shazeer N, Parmar N, et al, Attention is all you need, Advances in neural information processing systems, 2017: 5998-6008,c2017.

[16] 张海涛,王丹,徐海玲,孙思阳, 基于卷积神经网络的微博舆情情感分类研究, 情报学报, 37(07): 695-702, 2018.

[17] 张英, 基于深度神经网络的微博短文本情感分析研究, 中原工学院, 2017.

[18] 廖海涵,王曰芬,关鹏, 微博舆情传播周期中不同传播者的主题挖掘与观点识别, 图书情报工作, 62(19): 77-85, 2018.

[19] 任中杰,张鹏,李思成,兰月新,夏一雪,崔彦琛, 基于微博数据挖掘的突发事件情感态势演化分析——以天津 8·12 事故为例, 情报杂志, 38(02): 140-148, 2019.

[20] 崔彦琛,张鹏,兰月新,吴立志, 面向时间序列的微博突发事件衍生舆情情感分析研究——以“6.22”杭州保姆纵火案衍生舆情事件为例, 情报科学, 37(03): 119-126, 2019.

[21] 孙靖超, 基于优化深度双向自编码网络的舆情情感识别研究, http://kns.cnki.net/kcms/detail/61.1167.g3.20200323.1643.012.html, (2020-05-28).

[22] 谌志群,鞠婷, 基于 BERT 和双向 LSTM 的微博评论倾向性分析研究, http://kns.cnki.net/kcms/detail/11.1762.g3.20200411.2347.002.html, (2020-05-28).

[23] Blei D M, Ng A Y, Jordan M I, Latent dirichlet allocation. Journal of machine Learning research, 3(1): 993-1022, 2003.

[24] 与新冠肺炎相关的微博数据, https://www.datafountain.cn/competitions/423/datasets, (2020-05-14).

[25] Cui Y, Che W, Liu T, et al, Pre-training with whole word masking for chinese bert, arXiv preprint, arXiv:1906.08101, 2019.

[26] Sun Y, Wang S, Li Y, et al, Ernie: Enhanced representation through knowledge integration, arXiv preprint, arXiv:1904.09223, 2019.

[27] Sun Y, Wang S, Li Y, et al, Ernie 2.0: A continual pre-training framework for language understanding, arXiv preprint, arXiv:1907.12412, 2019.

附录

bert 模型

```
# 基于百度 AISTUDIO
import pandas as pd
train = pd.read_csv('data/data24278/train.csv', engine ='python')
unlablel = pd.read_csv('data/data24278/train_label.csv', engine ='python')
train.head()

print(train.shape)
print(train.columns)
print(unlablel.shape)
print(unlablel.columns)

# 标签分布
%matplotlib inline
train['情感倾向'].value_counts(normalize=True).plot(kind='bar');

# 微博长度
train['微博中文内容'].str.len().describe()

# 划分验证集，保存格式  text[\t]label
from sklearn.model_selection import train_test_split

train_labled = train[['微博中文内容', '情感倾向']]
train, valid = train_test_split(train_labled, test_size=0.2, random_state=2020)
train.to_csv('/home/aistudio/data/data24278/train.txt', index=False, header=False, sep='\t')
valid.to_csv('/home/aistudio/data/data24278/valid.txt', index=False, header=False, sep='\t')

# 自定义数据集
import os
import codecs
import csv

from paddlehub.dataset.base_nlp_dataset import BaseNLPDataset

class MyDataset(BaseNLPDataset):
    """DemoDataset"""
    def __init__(self):
        # 数据集存放位置
        self.dataset_dir = "/home/aistudio/data/data24278"
        super(MyDataset, self).__init__(
            base_path=self.dataset_dir,
            train_file="train.txt",
            dev_file="valid.txt",
            test_file="valid.txt",
            train_file_with_header=False,
            dev_file_with_header=False,
            test_file_with_header=False,
            # 数据集类别集合
            label_list=["-1", "0", "1"])
```

```
dataset = MyDataset()
for e in dataset.get_train_examples()[:3]:
    print("{}\t{}\t{}".format(e.guid, e.text_a, e.label))


# 加载模型
import paddlehub as hub
module = hub.Module(name='bert_chinese_L-12_H-768_A-12')

# 构建 Reader
reader = hub.reader.ClassifyReader(
    dataset=dataset,
    vocab_path=module.get_vocab_path(),
    sp_model_path=module.get_spm_path(),
    word_dict_path=module.get_word_dict_path(),
    max_seq_len=140)

# finetune 策略
strategy = hub.AdamWeightDecayStrategy(
    weight_decay=0.01,
    warmup_proportion=0.1,
    learning_rate=5e-5)

# finetune 策略
strategy = hub.AdamWeightDecayStrategy(
    weight_decay=0.01,
    warmup_proportion=0.1,
    learning_rate=5e-5)

# 运行配置
config = hub.RunConfig(
    use_cuda=True,
    num_epoch=1,
    checkpoint_dir="model",
    batch_size=32,
    eval_interval=100,
    strategy=strategy)

# Finetune Task
inputs, outputs, program = module.context(
    trainable=True, max_seq_len=140)

# Use "pooled_output" for classification tasks on an entire sentence.
pooled_output = outputs["pooled_output"]

feed_list = [
    inputs["input_ids"].name,
    inputs["position_ids"].name,
    inputs["segment_ids"].name,
    inputs["input_mask"].name,
]

cls_task = hub.TextClassifierTask(
            data_reader=reader,
```

```
            feature=pooled_output,
            feed_list=feed_list,
            num_classes=dataset.num_labels,
            config=config,
            metrics_choices=["f1"])

# finetune
run_states = cls_task.finetune_and_eval()

# test
import numpy as np
inv_label_map = {val: key for key, val in reader.label_map.items()}
# Data to be prdicted
data = valid[['微博中文内容']].fillna(' ').values.tolist()
run_states = cls_task.predict(data=data)
results = [run_state.run_results for run_state in run_states]
#  生成预测结果
proba = np.vstack([r[0] for r in results])
prediction = list(np.argmax(proba, axis=1))
prediction = [inv_label_map[p] for p in prediction]
print(len(prediction))
valid['y'] = prediction
print(valid.shape)
from sklearn.metrics import confusion_matrix
confusion_matrix(valid['情感倾向'].astype(str), valid['y'].astype(str))
from sklearn.metrics import f1_score, precision_score, recall_score
f1 = f1_score(valid['情感倾向'].astype(str), valid['y'].astype(str), average='weighted' )
p = precision_score(valid['情感倾向'].astype(str), valid['y'].astype(str), average='weighted')
r = recall_score(valid['情感倾向'].astype(str), valid['y'].astype(str), average='weighted')
print(p,r,f1)

#  生成预测结果
data = unlablel[['微博中文内容']].fillna(' ').values.tolist()
run_states = cls_task.predict(data=data)
results = [run_state.run_results for run_state in run_states]
proba = np.vstack([r[0] for r in results])
prediction = list(np.argmax(proba, axis=1))
prediction = [inv_label_map[p] for p in prediction]
unlablel['y'] = prediction
unlablel.head()
unlablel.to_csv('/home/aistudio/data/data24278/label.csv')
```

高频词聚类

```
#  词频统计
import pkuseg
import re
import time
from collections import Counter

#----------------------------------中文分词----------------------------------
# C-class.txt 为微博文本内容
```

```
cut_words = ""
all_words = ""
seg = pkuseg.pkuseg()
f = open('C-class-fenci.txt', 'w',encoding='utf-8')
for line in open('C-class.txt', encoding='utf-8'):
    line.strip('\n')
    line = re.sub('[0-9’!"#$%&\'()*+,-./:;<=>?@，。?★、…【】《》？“”‘’！[\\]^_`{|}~\s]+', "",
line)
    seg_list = seg.cut(line)
    # print(" ".join(seg_list))
    cut_words = (" ".join(seg_list))
    f.write(cut_words)
    all_words += cut_words
else:
    f.close()

# 输出结果
all_words = all_words.split()
print(len(all_words))

# 词频统计
c = Counter()
for x in all_words:
    if len(x)>1 and x != '\r\n':
        c[x] += 1

# 输出词频最高的前 10 个词
print('\n 词频统计结果：')
for (k,v) in c.most_common(10):
    print("%s:%d"%(k,v))

# 存储数据
name = time.strftime("%Y-%m-%d") + "-fc.csv"
fw = open(name, 'w', encoding='utf-8')
i = 1
for (k,v) in c.most_common(len(c)):
    fw.write(str(i)+','+str(k)+','+str(v)+'\n')
    i = i + 1
else:
    print("Over write file!")
    fw.close()

from bert_serving.client import BertClient
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
import pandas as pd
import matplotlib
bc = BertClient()
km = KMeans(n_clusters=3)
pca = PCA(n_components=2)
data = pd.read_csv(r"D:\workspace\ML\nlp\comment\2020-05-25-fc.csv")
corpus = data.iloc[0:120,1].tolist()
```

```
vectors = bc.encode(corpus)
vectors_ = pca.fit_transform(vectors)     #降维到二维
y_ = km.fit_predict(vectors_)             #聚类
plt.rcParams['font.sans-serif'] = ['FangSong']
matplotlib.rcParams['axes.unicode_minus']=False
plt.scatter(vectors_[:,0],vectors_[:, 1],c=y_)     #将点画在图上
for i in range(len(corpus)):       #给每个点进行标注
      plt.annotate(s=corpus[i], xy=(vectors_[:, 0][i], vectors_[:, 1][i]),xytext=(vectors_[:, 0][i] +
0.1, vectors_[:, 1][i] + 0.1))
plt.show()
```

作图、数据筛选

```
## import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
plt.style.use('seaborn')
sns.set(font_scale=2)

plt.rcParams['font.sans-serif']=['SimHei'] #用来正常显示中文标签
plt.rcParams['axes.unicode_minus']=False #用来正常显示负号

train = pd.read_csv('train.csv')
labeled = pd.read_csv('labeled.csv')

labeled.drop(['Unnamed: 0'],axis=1,inplace=True)
train.head()
# labeled.rename(columns={'y':'情感倾向'}, inplace = True)
# labeled.to_csv("labeled.csv")
labeled.head()
data = pd.concat([train,labeled],axis=0)
data.shape

#  标签总体分布
data['情感倾向'].value_counts()/data['情感倾向'].count()
(data['情感倾向'].value_counts()/data['情感倾向'].count()).plot.bar()
plt.show()
#  时间对于情感的影响
data['time'] = pd.to_datetime('2020 年' + data['微博发布时间'], format='%Y 年%m 月%d
日  %H:%M', errors='ignore')
data['date'] = data['time'].dt.date
date_influence = data.groupby(['date','情感倾向'],as_index=False).count()
sns.relplot(x="date", y="微博 id", kind="line", hue='情感倾向',palette=["b",
"r",'g'],data=date_influence)
plt.xticks(rotation=90,fontsize=12)
plt.xlabel('日期',fontsize=15)
plt.ylabel('数量',fontsize=15)
plt.title('微博数量分布图',fontsize=15)
```

```
plt.show()
# 时间对于情感的影响比例
date_influence = date_influence.merge(data.groupby('date',as_index=False)['情感倾向
'].count().rename(columns={'情感倾向':'weibo_count'}),how='left',on='date')
date_influence['weibo_rate'] = date_influence['微博 id']/date_influence['weibo_count']
sns.relplot(x="date", y="weibo_rate", kind="line", hue='情感倾向',palette=["b",
"r",'g'],data=date_influence)
plt.xticks(rotation=90,fontsize=12)
plt.xlabel('日期',fontsize=15)
plt.ylabel('数量',fontsize=15)
plt.title('微博情感占比分布图',fontsize=15)
plt.show()

# 进行数据筛选
data010102_0 = data[['微博中文内容']][((data['date']=='2020-01-01')|(data['date']=='2020-01-
02'))&(data['情感倾向']==-1)]
data010102_1 = data[['微博中文内容']][((data['date']=='2020-01-01')|(data['date']=='2020-01-
02'))&(data['情感倾向']==0)]
data010102_2 = data[['微博中文内容']][((data['date']=='2020-01-01')|(data['date']=='2020-01-
02'))&(data['情感倾向']==1)]
data010102_0.to_csv('data010102_0.txt', index=False, header=False, sep='\t')
data010102_1.to_csv('data010102_1.txt', index=False, header=False, sep='\t')
data010102_2.to_csv('data010102_2.txt', index=False, header=False, sep='\t')

data011920_0 = data[['微博中文内容']][((data['date']=='2020-01-19')|(data['date']=='2020-01-
20'))&(data['情感倾向']==-1)]
data011920_1 = data[['微博中文内容']][((data['date']=='2020-01-19')|(data['date']=='2020-01-
20'))&(data['情感倾向']==0)]
data011920_2 = data[['微博中文内容']][((data['date']=='2020-01-19')|(data['date']=='2020-01-
20'))&(data['情感倾向']==1)]
data011920_0.to_csv('data011920_0.txt', index=False, header=False, sep='\t')
data011920_1.to_csv('data011920_1.txt', index=False, header=False, sep='\t')
data011920_2.to_csv('data011920_2.txt', index=False, header=False, sep='\t')

data012324_0 = data[['微博中文内容']][((data['date']=='2020-01-23')|(data['date']=='2020-01-
24'))&(data['情感倾向']==-1)]
data012324_1 = data[['微博中文内容']][((data['date']=='2020-01-23')|(data['date']=='2020-01-
24'))&(data['情感倾向']==0)]
data012324_2 = data[['微博中文内容']][((data['date']=='2020-01-23')|(data['date']=='2020-01-
24'))&(data['情感倾向']==1)]
data012324_0.to_csv('data012324_0.txt', index=False, header=False, sep='\t')
data012324_1.to_csv('data012324_1.txt', index=False, header=False, sep='\t')
data012324_2.to_csv('data012324_2.txt', index=False, header=False, sep='\t')

data020506_0 = data[['微博中文内容']][((data['date']=='2020-02-05')|(data['date']=='2020-02-
06'))&(data['情感倾向']==-1)]
data020506_1 = data[['微博中文内容']][((data['date']=='2020-02-05')|(data['date']=='2020-02-
06'))&(data['情感倾向']==0)]
```

```
data020506_2 = data[['微博中文内容']][((data['date']=='2020-02-05')|(data['date']=='2020-02-06'))&(data['情感倾向']==1)]
data020506_0.to_csv('data020506_0.txt', index=False, header=False, sep='\t')
data020506_1.to_csv('data020506_1.txt', index=False, header=False, sep='\t')
data020506_2.to_csv('data020506_2.txt', index=False, header=False, sep='\t')

data021718_0 = data[['微博中文内容']][((data['date']=='2020-02-17')|(data['date']=='2020-02-18'))&(data['情感倾向']==-1)]
data021718_1 = data[['微博中文内容']][((data['date']=='2020-02-17')|(data['date']=='2020-02-18'))&(data['情感倾向']==0)]
data021718_2 = data[['微博中文内容']][((data['date']=='2020-02-17')|(data['date']=='2020-02-18'))&(data['情感倾向']==1)]
data021718_0.to_csv('data021718_0.txt', index=False, header=False, sep='\t')
data021718_1.to_csv('data021718_1.txt', index=False, header=False, sep='\t')
data021718_2.to_csv('data021718_2.txt', index=False, header=False, sep='\t')
```

LDA 主题模型

```
import os
import re
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import pkuseg
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer

txtname = 'data021718_2.txt'
#----------------------  第一步  读取数据  ----------------------
seg = pkuseg.pkuseg()
corpus = []
for line in open(txtname, encoding='utf-8'):
    line.strip('\n')
    line = re.sub('[0-9’!"#$%&\'()*+,-./:;<=>?@，。?★、…【】《》？“”‘’！[\\]^_`{|}~\s]+', "", line)
    seg_list = seg.cut(line)
    # print(" ".join(seg_list))
    corpus.extend(seg_list)

#-----------------------  第二步  计算 TF-IDF 值  -----------------------
# 设置特征数
n_features = 5000

tf_vectorizer = TfidfVectorizer(strip_accents = 'unicode',
                                max_features=n_features,
                                stop_words=['的','或','等','是','有','之','与','可以','还是','比较','这里',
                                            '一个','和','也','被','吗','于','中','最','但是','图片','大家',
                                            '一下','几天','200','还有','一看','300','50','哈哈哈哈',
```

```
                                                    '“','”','。','，','？','、','；','怎么','本来','发
现',
                                                    'and','in','of','the','我们','一直','真的','18','
一次',
                                                    '了','有些','已经','不是','这么','一一','一天','
这个','这种',
                                                    '一种','位于','之一','天空','没有','很多','有点
','什么','五个',
                                                    '特别','微博','链接','全文','展开','网页','自己
','今天','现在'],
                                        max_df = 0.99,
                                        min_df = 0.002) #去除文档内出现几率过大或过小的词
汇

tf = tf_vectorizer.fit_transform(corpus)

print(tf.shape)
print(tf)

#-------------------------   第三步 LDA 分析   ------------------------
from sklearn.decomposition import LatentDirichletAllocation

# 设置主题数
n_topics = 2

lda = LatentDirichletAllocation(n_components=n_topics,
                                        max_iter=40,
                                        learning_method='online',
                                        learning_offset=50,
                                        random_state=0)
lda.fit(tf)

# 显示主题数 model.topic_word_
print(lda.components_)
# 几个主题就是几行 多少个关键词就是几列
print(lda.components_.shape)

# 计算困惑度
print(u'困惑度：')
print(lda.perplexity(tf,sub_sampling = False))

# 主题-关键词分布
def print_top_words(model, tf_feature_names, n_top_words):
    for topic_idx,topic in enumerate(model.components_):   # lda.component 相当于
model.topic_word_
        print('Topic #%d:' % topic_idx)
        print(' '.join([tf_feature_names[i] for i in topic.argsort()[:-n_top_words-1:-1]]))
        print("")

# 定义好函数之后 暂定每个主题输出前 20 个关键词
n_top_words = 20
```

```
tf_feature_names = tf_vectorizer.get_feature_names()
# 调用函数
print_top_words(lda, tf_feature_names, n_top_words)


#------------------------  第四步 可视化分析  -------------------------
import pyLDAvis
import pyLDAvis.sklearn

#pyLDAvis.enable_notebook()

data = pyLDAvis.sklearn.prepare(lda,tf,tf_vectorizer)
print(data)

#显示图形
pyLDAvis.show(data, ip='127.0.0.1', port=8888, n_retries=50, local=True, open_browser=True,
http_server=None)
```