数据分析基础(Python)

副标题：大模型和 AI 的应用：基于 BERT 的聊天机器人、生物信息安全与 GUI 设计

日期: 2024-12-10

小组成员:

1. 杨金源+302023510166
2. 郑华舰+302023510105
3. 王鎏清+302023510172
4. 林晓德+302023503377
5. 徐佳坤+302023503027

任务分配:

王鎏清：程序开发和模型训练

郑华舰：报告撰写

杨金源：设计 GUI

林晓德：数据标注和答辩 ppt 制作

徐佳坤：数据清洗和模型评估

I. 课题背景介绍

随着人工智能技术的突飞猛进，大型预训练模型如 BERT（Bidirectional Encoder Representations from Transformers）已成为自然语言处理（NLP）领域的翘楚[1]。BERT 模型以其先进的双向编码能力，显著提升了语言理解任务的性能，为聊天机器人、情绪分析、文本分类等的实现提供了保障。在本项目中，我们采用了零一万物提供的 yi-large API 接口，以实现基础的聊天功能和文档分析能力[2]。特别地，我们利用 BERT 训练出来的模型对用户输入内容进行情感分析。此外，为了增强程序的安全性并提升用户体验，我们集成了生物信息解锁功能，并设计了一个用户友好的图形用户界面（GUI）。

II. 数据集描述

1. 测试集（Test Set）描述： 测试集中共有 4416 条评论，情绪分布如下：愤怒情绪的评论有 853 条，占比约为 19.3%；积极情绪的评论有 1616 条，占比约为 36.6%；消极情绪的评论有 1270 条，占比约为 28.8%；中性情绪的评论有 852 条，占比约为 19.3%。微博对特定愤怒情绪（如脏话）有一定程度的屏蔽，导致爬取的评论中愤怒情绪占比较低。可以预见的是，训练出来的模型对于脏话的情感分析能力并不出众，但是项目集成的脏话检测功能，将此影响减少到最低。

2. 训练集（Train Set）描述： 训练集中包含 361745 条评论，情绪分布如下：愤怒情绪的评论有 55267 条，占比约为 15.3%；积极情绪的评论有 199497 条，占比约为 55.1%；消极情绪的评论有 51714 条，占比约为 14.3%；中性情绪的评论有 55267 条，占比约为 15.3%，具体分布详见图 2。

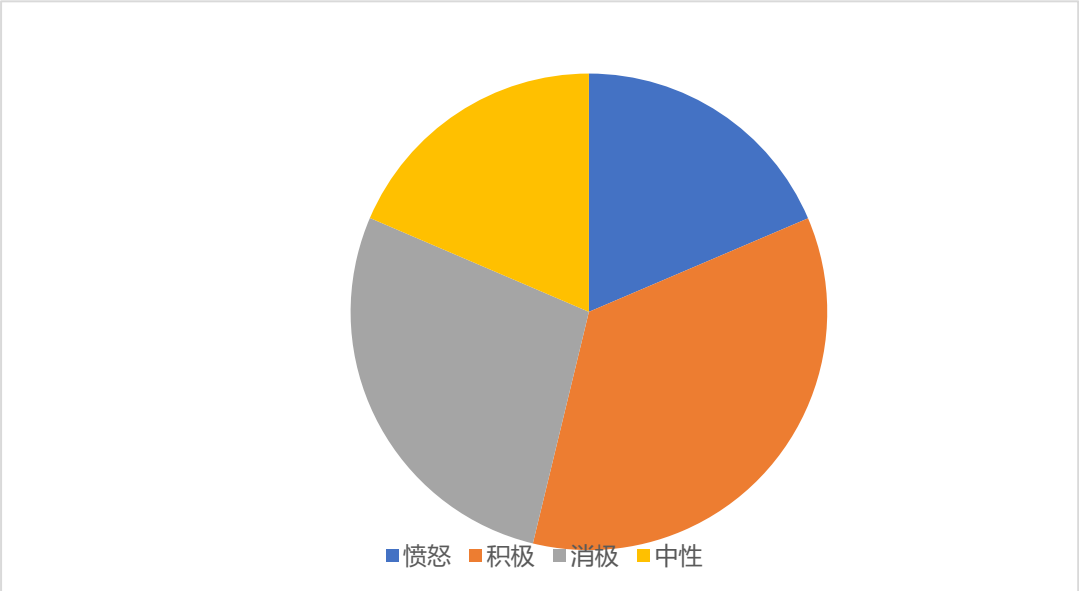

图 1 test 集数据分布图

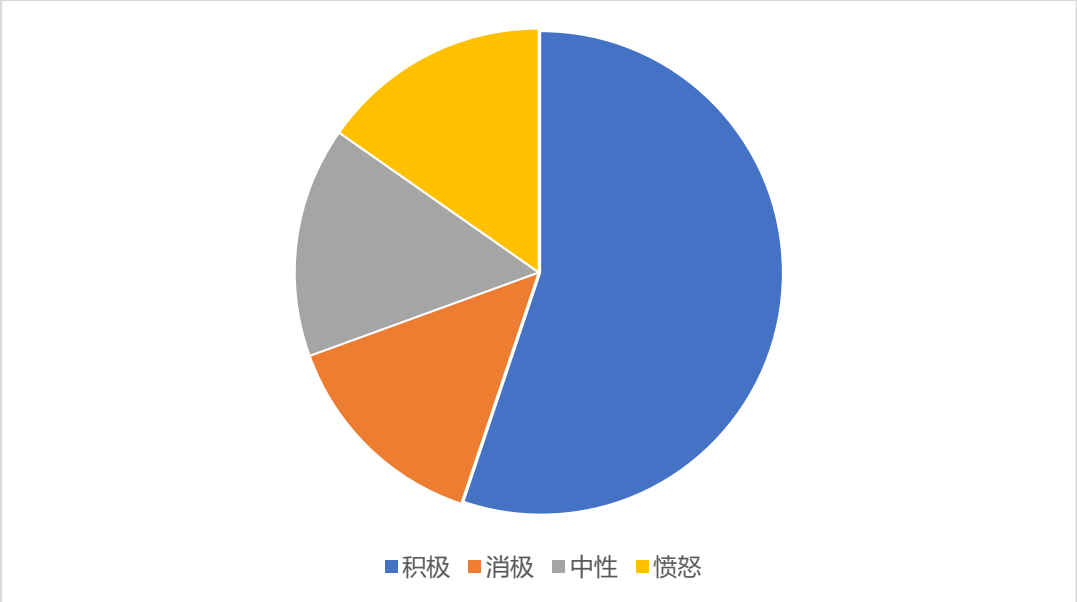

图 2 train 集数据分布图

III. 方法与结果

在开发过程中，我们最初使用了 SnowNLP 进行语义分析，但由于其准确率不理想，且公开的模型未能很好地满足我们的需求，因此决定自行训练。为了节省时间，训练时选择了公开的 simplifyweibo_4_moods 作为训练集。为了更好地评估模型质量，我们爬取了微博热搜评论数据，经过手工清洗和标注后，得到了额外的测试集。

1. 我们最开始采用了课堂上介绍的爬虫方法进行爬取，但在实际数据收集过程中我们遇到不少问题：

（1）程序缺少自动化请求机制，每次爬取时都需要手动修改请求（输入微博帖子 id），从而导致无法连续自动抓取，这极大影响了评论爬取的效率。（2）我们没有将评论封装为对象，这可能数据处理和维护的困难。（3）原始代码将数据直接写入 CSV 文件，这样的读取方式数据读写速度较低，延迟较高。（3）代码没有很好地模块化，这可能导致代码难以维护和扩展。(4)原始代码在写入 CSV 文件时，如果程序中断，可能会导致数据不完整或文件损坏。(5)爬取的数据良莠不齐，有些数据对模型训练无用，需要清洗。

经过迭代后，我们直接选择爬取微博热搜下的评论，通过调用 get_hot_list 方法，向微博热搜 API 发送 HTTP 请求（requests.get），获取返回的 JSON 数据，解析其中的热搜条目。每个热搜条目被封装为 Hot 对象，并加入到结果列表中返回。

针对每个热搜话题，系统进一步调用 get_weibo_list 方法，获取与该话题相关联的微博列表。具体实现上，系统通过替换热搜话题链接中的参数构建微博列表的 API 请求 URL，发送请求并解析返回的微博数据，包括微博 ID、用户信

息、发布时间和内容等。对于每条微博，系统会发送额外请求，获取完整的微博正文及发布时间。

此外，系统通过 get_comment_list 和 parse_comment 方法获取每条微博的评论列表。爬虫使用递归方式分页获取评论数据，通过微博 ID 和分页参数向微博评论 API 发送请求，随后系统解析返回的 JSON 数据，提取评论文本、时间戳和点赞数等信息，并将每条评论封装为 Comment 对象。

数据处理方面，我们通过对评论进行手工标注（由人工判定语句的情感，其中 0 为积极,1 为消极，2 为中性,3 为愤怒），使爬取到的数据可以用于模型的分析。由于数据的良莠不齐，系统通过 Eraser 从 dictionary 中读取要清洗的内容，这样就可以在评论爬取的过程中进行对数据的清洗，而使用 dictionary 则方便对在实际操作过程中遇到预期之外的需要清洗的内容进行添加。

数据存储方面，系统通过 RedisUtil.get_redis_connection 方法获取 Redis 连接，定义了针对热搜、微博和评论的键值存储方案。使用 Redis 的 hmset 和 rpush 命令将数据保存到 Redis 数据库，并为每个存储的键设置过期时间，从而实现自动清理过期数据[5]。

整个爬虫任务的启动由 run_crawler.py 文件中的 CrawlTask 实例完成，通过调用 run 方法启动爬取过程。此流程确保了数据的准确性和爬取的高效性，同时保证了存储系统的可靠性和数据的实时性。

2.训练方法：

我们选择了 BERT 模型[3]进行监督学习，是因为在有明确任务目标且具备标签数据的情况下，监督学习能够更高效地实现目标。与无监督学习相比，监督学习可以充分利用已有的标签信息，直接优化分类性能。

在传统的无监督学习中，由于没有标签，模型无法直接知道正确答案，因此通常需要依赖诸如聚类、降维等方法从数据中提取潜在模式。这种方法适用于没有标签的探索性分析,然而，对于情感分析等有明确任务目标的场景，监督学习显然更加合适,它能够通过标签信息来引导模型训练，从而高效地提高分类任务的准确性。

我们通过训练数据中的标签信息来指导模型学习。具体而言，模型的预测结果与真实标签之间的差异通过交叉熵损失函数量化，并利用反向传播 来更新模

型参数，从而优化分类性能。反向传播通过梯度下降法计算损失函数对模型参数的梯度，并调整模型参数，使得损失逐渐减小，模型的预测逐步接近真实标签。

在数据集中，我们注意到不同情感标签的样本数量可能存在不平衡问题，这可能导致训练时模型偏向样本较多的类别。为了解决这个问题，我们采用了加权损失函数。具体来说，我们根据每个类别的样本数量，计算出类别的权重，并在交叉熵损失函数中进行加权调整。这样，训练过程中，模型会对样本较少的类别给予更高的关注，从而提高模型在这些类别上的表现，具体加权的实现方式如下：

```
//定义加权损失函数
FUNCTION weighted_loss_function(logits,labels,class_weights_tensor):
    //将类别权重移动到与 logits 相同的设备上（例如：GPU 或 CPU）
    class_weights_tensor=class_weights_tensor.to(logits.device)
    //计算交叉熵损失
    loss=CROSS_ENTROPY_LOSS(logits, labels)
    //应用类别权重
    weighted_loss = loss * class_weights_tensor[labels]
    //返回加权损失
    RETURN weighted_loss
//在模型训练步骤中使用加权损失函数
FUNCTION training_step(model,inputs,class_weights_tensor):
    //将模型设置为训练模式
    model.train()
    //从输入中提取标签
    labels = inputs.pop("labels")
    //从输入中提取特征并准备模型输入
    features = inputs.to(model.device)
    //前向传播以获得模型输出
    outputs = model(features)
    logits = outputs.logits
    //计算加权损失
    loss = weighted_loss_function(logits, labels, class_weights_tensor)
```

```
//反向传播
loss.backward()
//返回损失
RETURN loss
```

3.聊天机器人实现方法：

（1）聊天交互：GptAssistant 类初始化一个客户端，用于与聊天模型进行交互。对话历史被维护在一个列表中，以便于生成上下文相关的回复。API 发送对话请求，并返回 AI 的回复。AI 的回复也被添加到对话历史中，以便于未来的交互能够保持上下文的连贯性。

（2）脏话识别：考虑到项目运行效率问题，我们选择了使用本地词典而不是已有的 api。程序接收用户输入的文本和本地词典作为参数，将文本转换为小写后，检查是否包含词典中的任何脏话。当检测到用户输入脏话，对话框将输出警告并且不会发送该消息。

（3）文档分析：我们使用 QFileDialog.getOpenFileName 方法打开一个文件选择对话框，让用户选择 Word 文档。这个方法返回一个元组，包含文件名和选中的过滤器，但这里只关心文件名，所以第二个返回值被忽略（用下划线_表示）。然后我们使用了 Doc_reading.read_docx_file(file_path)方法，从指定路径读取 Word 文档，并提取其中的文本内容。通过使用 python-docx 库，它遍历文档的所有段落，并将这些段落的文本合并成一个单一的字符串，从而实现文档内容的完整提取。然后我们使用了 ChatApp.analyze_document()方法对 Word 文档进行分析。首先，它弹出一个文件选择对话框，用户通过此对话框选择一个 Word 文档后，Doc_reading.read_docx_file 方法被调用来读取并提取文档内容。如果文档内容成功读取，该方法将内容发送至 AI 进行总结。最后，将 AI 生成的总结结果显示在聊天窗口中。

通过这些功能的实现，聊天机器人不仅能够与用户进行自然的对话交互，还能够理解和处理用户上传的文档，提供有价值的信息摘要。同时，通过脏话识别和情绪分析，聊天机器人能够维护一个健康和积极的对话环境。

4.人脸识别：

我们选择使用 OpenCv 进行面部识别功能，在图像处理方面我们使用了以下方法[4]：

（1）二值化：二值化是将之转换为是黑白图像。这有助于简化图像，突出人脸的轮廓和特征，减少背景噪声，使得人脸检测算法更容易识别出人脸区域。二值化的公式如下：

$$THRESH\_BINARY(x,y)=\begin{cases}\max val, & 当 src(x,y)>thresh \\ 0, others\end{cases} \tag{4-1}$$

（2）降噪：

采用高斯滤波降噪，这在可以去除暗光环境下产生的噪点。高斯滤波是一种常用的图像处理技术，用于对图像进行平滑处理，减少图像中的噪声[5]。高斯滤波的公式如下：

$$\sigma = 0.3\times(\frac{ksize-1}{2}-1)+0.8 \tag{4-2}$$

（σ：高斯分布的标准差。ksize：高斯核的大小。）

$$Gi=\alpha\cdot e^{-\frac{(i-\frac{ksize-1}{2})^2}{2\sigma^2}} \tag{4-3}$$

(Gi：高斯核中的第 i 个元素。α：归一化因子，确保高斯核中所有元素的和为。i：当前元素的索引，范围从 0 到 ksize−1）

在比较人脸方面我们使用了直方图比较，直方图比较是一种评估两幅图像相似度的方法，通过比较它们的直方图来实现。在人脸比较中，直方图比较可以用来评估捕获的人脸与注册人脸的相似度，从而判断是否为同一人。

结果：

我们训练出的文本分析模型在情感分析任务中达到了 85.24%的正确率，这表明模型在大多数情况下能够准确识别和分类情感。从 ROC 曲线看出，积极、消极、中性的 AUC（Area Under the Curve，曲线下面积）分别是 0.95、0.89、0.91，这表明模型在区分这些情感的时候表现极佳。而愤怒情绪的 ROC 曲线，其 AUC 值为 0.78。这是四个类别中 AUC 值最低的，从混淆矩阵中也能看出模型在区分愤怒情绪时的性能相对较弱。

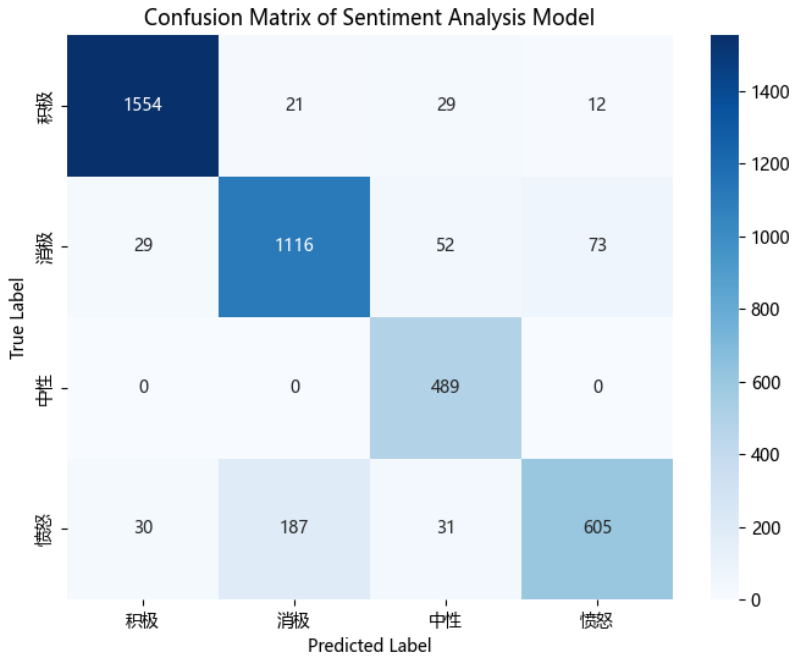

图 3 混淆矩阵

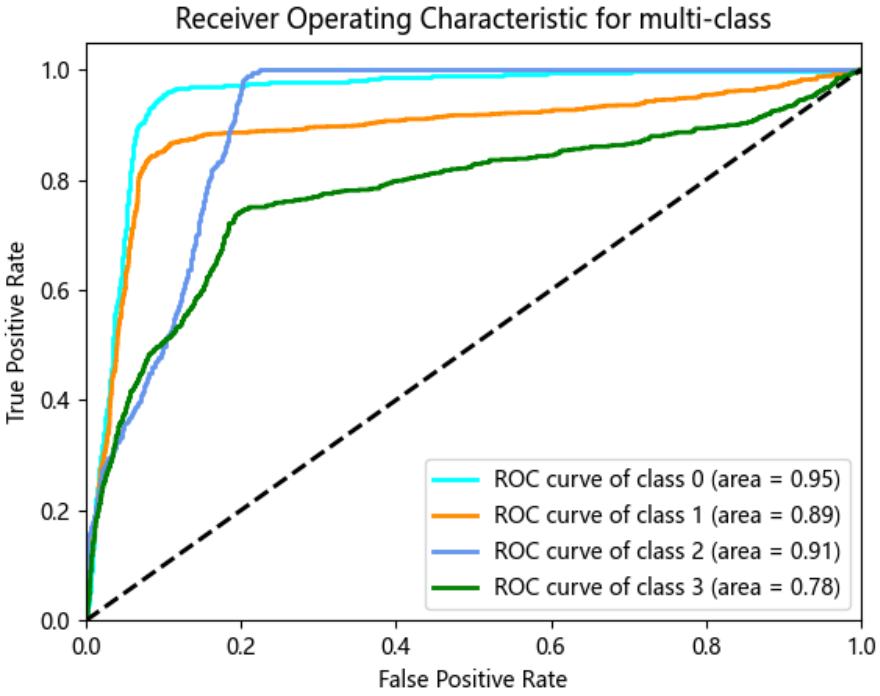

图 4 ROC 曲线

IV. GUI 设计

运行程序后会弹出一个智能聊天助手的对话框，可以在对话框下方的输入栏输入内容，如图 4。

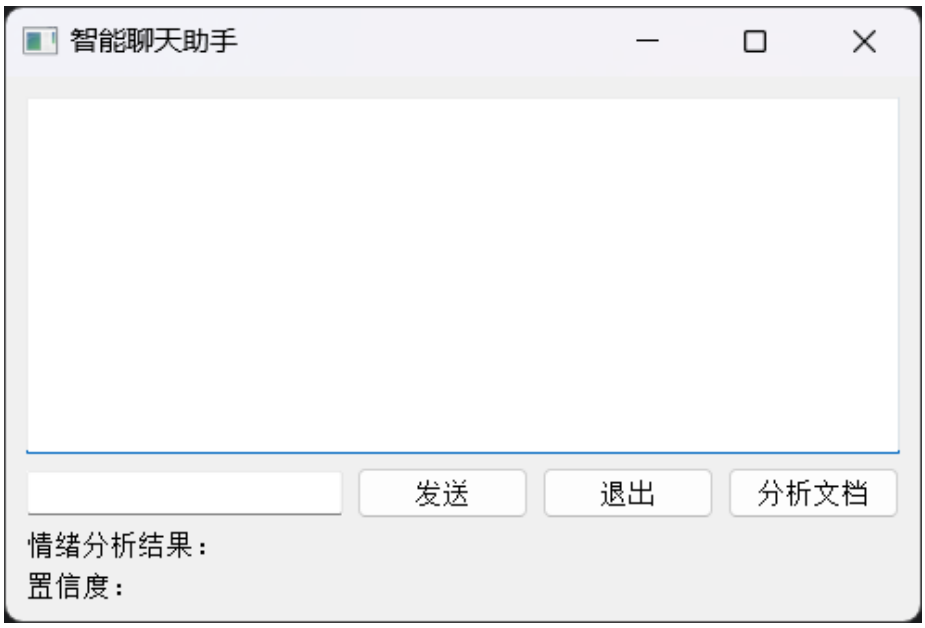

图 4 初始界面图

如果输入正常对话内容并点击发送，则可以看到对话框中 AI 给出的回复，如果输入脏话则会被提示无法输入，如图 7 所示。

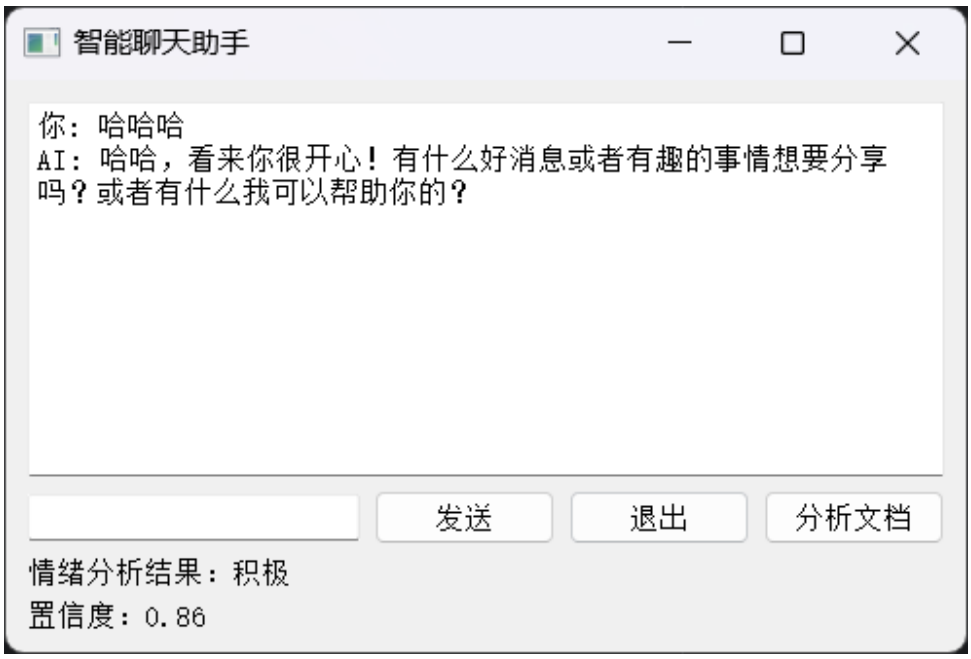

图 5 聊天示例图

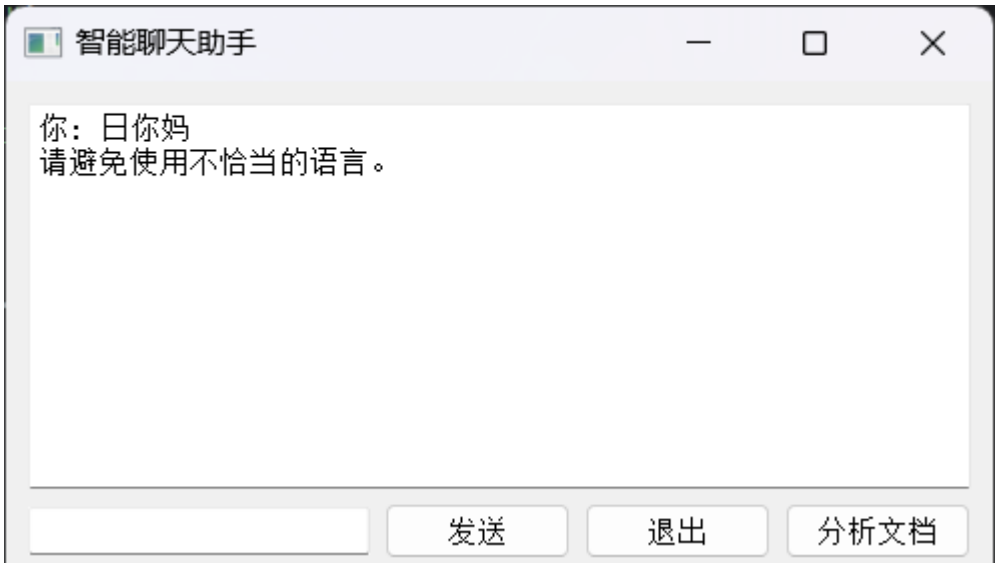

图 7 脏话输入示例图

如果输入一个文档并点击分析文档，则可以看到对话框中 AI 给出的文档分析和总结的内容，如图 6。

图 6 文档分析示例图

VI. 总结与讨论

总结：

在这次的项目实现过程中，我们学会了如何从微博高效率地爬取评论并进行处理作为训练模型的数据集，如何利用 BERT 训练出相对满足要求的模型，如何

用训练好的模型实现情感分析的功能。如何用 OpenCv 实现人脸识别来完成生物信息安全功能。在这些模块所实现的功能组合下，我们成功实现了聊天机器人的能在大部分语境下对用户的对话进行回复，对脏话的检测和提示，以及为用户对文档进行总结等的功能。

讨论：

项目还存在以下不足：

1. 模型在区分愤怒和消极情绪方面还存在一定的不准确性，这是由于模型的设计还存在不完善的、模型训练的程度不够充分等的原因，因此后续的改进方向之一便是通过增加训练数据的多样性和数量，特别是消极和愤怒情感的数据，有助于模型更好地学习和区分这些情感。

2. 人脸识别功能由于采用的是转成灰度图的方法，虽然这有利于提高识别效率，但也带来了如果光线条件变了可能会导致无法完成人脸识别功能等的问题，我们考虑的改进方案如下：（1）通过直方图均衡化，增强图像的对比度，使得在不同光线条件下的特征更加明显，以及伽马校正，调整图像的亮度，以减少光线变化的影响等的图像预处理的方法。（2）在不同尺度上分析图像，以识别在不同分辨率下都稳定的人脸特征。（3）使用颜色信息提供额外的线索来区分不同的人脸。（4）计算图像的光照分布并进行归一化处理，减少光照变化的影响。使用深度学习模型，如卷积神经网络（CNN），这些模型能够从大量数据中学习到在不同光照条件下的人脸识别特征。

3. 我们的代码还存在编写不规范，版本维护处理不重视等问题，比如新旧版本更替我们采用的是将旧的代码注释掉直接写入新的代码进行迭代，没有进行日志记录等操作，代码封装等方面的不规范导致了代码的可读性和可维护性等方面也不强，这些都是日后项目所需要改进以及我们在编写其他程序时所需要注意的方面。

VII. 参考文献

[1] Jacob Devlin, et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding[J]. arXiv preprint arXiv:1810.04805, 2018.

[2] 零一万物. 如何调用零一万物 API_大模型服务平台百炼(Model Studio)-阿里云帮助中心[EB/OL].(2024-06-13)[2024-12-13].https://help.aliyun.com/zh/model-studio/developer-reference/yi-large-api.

[3] 张泽源,张光妲,李佳雨,等.基于 BERT 的日常文本情感分析[J].齐齐哈尔大学学报(自然科

学版),2024,40(06):37-41.DOI:10.20171/j.cnki.23-1419/n.2024.06.005.
[4] 基于 OpenCV 的人脸识别算法研究与实现 https://doc.taixueshu.com/journal/20201473dnzsyjsxsb.html. 2020-06-11.
[5] 高斯滤波官方文档 https://docs.opencv.org/2.4/modules/imgproc/doc/filtering.html#Mat%20getGaussianKernel%28int%20ksize,%20double%20sigma,%20int%20ktype%29
[6] Antirez. Redis 官方文档[EB/OL]. (2024-06-13)[2024-12-13]. https://redis.io/documentation.

VIII. 代码

Python 版本：3.1.2，

所 用 包 ： Openai-1.56.2 python-docx-1.1.2 transformers-4.46.3 torch-2.5.1 tensorflow-2.18.0 opencv-python-4.10.0.84 redis-5.2.0 pandas-2.2.3

matplotlib-3.9.3 scikit-learn-1.5.2 datasets-3.1.0
