摘 要

随着社交媒体的迅猛发展和广泛普及，信息传播的速度与范围得到了前所未有的提升。然而，这也为虚假新闻的滋生与快速散播提供了便利条件，对社会舆论引导、公共安全乃至国家稳定构成了日益严峻的挑战。传统的虚假新闻检测方法多依赖于单一模态信息（如仅文本或仅图像），在面对日益复杂化、多媒体化的虚假新闻时，往往难以进行全面准确的识别，存在特征提取不充分、语义理解片面等局限性。

为有效应对上述挑战，本研究设计并实现了一套基于多模态深度学习的社交媒体虚假新闻自动检测系统。该系统旨在通过融合文本、图像等多源异构信息，提升虚假新闻识别的准确性和鲁棒性。在数据层面，系统整合并预处理了来自公开社交媒体数据集的新闻文本和相关图像，构建了多模态训练语料库。在模型层面，核心检测模块采用多模态深度学习架构，其中文本特征提取基于BERT预训练模型，图像特征提取则利用ResNet卷积神经网络。通过精心设计的特征融合机制，模型能够综合分析不同模态间的关联信息。为进一步增强检测结果的可解释性和可靠性，系统还引入了大语言模型（LLM）进行交叉验证，对多模态模型的初步判断结果进行辅助核查与补充说明。

系统后端服务采用Python语言及高性能的Django Web框架搭建，实现了用户认证、新闻内容上传、模型推理调度、检测结果存储等核心功能，并选用MySQL作为数据管理方案。系统前端界面基于Vue.js框架及ElementUI组件库进行开发，提供了用户友好的交互体验，包括新闻提交、结果反馈、历史记录查询与管理等。前端还集成了Echarts等数据可视化工具，用于展示检测统计数据。管理员模块则支持用户账户管理、系统参数（如模型融合权重）动态配置及系统运行日志查看等功能。

本研究不仅在理论上探索了多模态信息融合技术在虚假新闻检测领域的应用，并在实践中构建了一个功能相对完善的检测系统。该系统有望为社交媒体平台、新闻机构及相关监管部门提供一种高效、智能的虚假信息治理辅助工具，对于净化网络环境、提升公众媒介素养具有积极意义。

关键字：虚假新闻检测；多模态深度学习；特征融合；BERT模型；ResNet模型；大语言模型；Django框架；Vue.js

ABSTRACT

The rapid development and widespread adoption of social media have unprecedentedly enhanced the speed and scope of information dissemination. However, this has also facilitated the generation and rapid spread of fake news, posing increasingly severe challenges to public opinion guidance, public safety, and even national stability. Traditional fake news detection methods, often relying on single-modal information (such as text-only or image-only), struggle to comprehensively and accurately identify sophisticated, multimedia-rich fake news, suffering from limitations like insufficient feature extraction and one-sided semantic understanding.

To effectively address these challenges, this study designs and implements an automatic detection system for social media fake news based on multimodal deep learning. The system aims to improve the accuracy and robustness of fake news identification by fusing heterogeneous information from multiple sources, including text and images. At the data level, news texts and associated images from public social media datasets were integrated and preprocessed to construct a multimodal training corpus. At the model level, the core detection module employs a multimodal deep learning architecture, where text feature extraction is based on the BERT pre-trained model, and image feature extraction utilizes the ResNet convolutional neural network. Through a carefully designed feature fusion mechanism, the model comprehensively analyzes the associative information between different modalities. To further enhance the interpretability and reliability of detection results, the system incorporates Large Language Models (LLMs) for cross-validation, providing auxiliary verification and supplementary explanations for the preliminary judgments of the multimodal model.

The backend services of the system are built using the Python language and the high-performance Django Web framework, implementing core functionalities such as user authentication, news content uploading, model inference scheduling, and detection result storage, with MySQL chosen for data management. The frontend interface is developed based on the Vue.js framework and the ElementUI component library, offering a user-friendly interactive experience, including news submission, result feedback, and history query and management. The frontend also integrates data visualization tools

like Echarts to display detection statistics. The administrator module supports user account management, dynamic configuration of system parameters (such as model fusion weights), and system operation log viewing.

This research not only theoretically explores the application of multimodal information fusion technology in the field of fake news detection but also practically constructs a relatively comprehensive detection system. The system is expected to provide an efficient and intelligent auxiliary tool for social media platforms, news organizations, and relevant regulatory authorities in governing false information, holding positive significance for purifying the online environment and enhancing public media literacy.

Key words: Fake News Detection; Multimodal Deep Learning; Feature Fusion; BERT Model; ResNet Model; Large Language Model (LLM); Django Framework; Vue.js

目 录

图 目 录

前　言

我们正处在一个信息以前所未有的速度和广度进行生产、传播与消费的时代。以互联网技术为基石，特别是社交媒体平台的崛起，彻底改变了人们获取信息、表达观点和进行社会互动的方式。从日常生活的点滴分享到全球重大事件的实时播报，社交媒体以其即时性、互动性和低门槛的特性，成为了现代社会不可或缺的组成部分。信息传播效率的极大提升，无疑为社会发展和个人生活带来了诸多便利。

然而，凡事皆有两面性。信息洪流的背后，也潜藏着不容忽视的风险与挑战。其中，虚假新闻（Fake News）的泛滥便是这个信息时代的突出副产物之一。这些被刻意捏造、歪曲事实或带有强烈误导性的信息，借助社交媒体的病毒式传播机制，能够迅速扩散并对公众认知、社会秩序乃至政治生态产生深远而复杂的负面影响。它们不仅侵蚀着媒体的公信力，也加剧了社会的分裂与不信任，成为了数字时代信息治理面临的重大难题。

面对虚假新闻的严峻挑战，传统的应对策略，如依赖人工审核或事后辟谣，在海量、高速传播的信息面前显得捉襟见肘。因此，学术界和产业界纷纷将目光投向了自动化、智能化的检测技术。早期的研究多集中于对新闻文本内容的分析，但随着虚假信息制造手段的不断升级，单一模态的分析已难以应对图文并茂、甚至音视频结合的复杂虚假内容。图片的误用、篡改，以及文本与视觉信息之间的不一致性，都成为了识别虚假新闻的关键线索。

基于此，多模态信息融合的理念应运而生，它强调综合利用文本、图像等多种来源的信息进行协同分析，以期更全面、更准确地揭示信息的真实性。深度学习技术的快速发展，尤其是Transformer、卷积神经网络（CNN）等强大模型的出现，为实现高效的多模态特征提取与智能融合提供了有力的技术支撑。

本毕业设计正是立足于这样的背景，旨在探索和实践一种基于多模态深度学习的社交媒体虚假新闻检测方法，并构建一个集新闻内容上传、自动检测、结果展示与分析于一体的Web应用系统。我们期望通过结合先进的文本理解模型（如BERT）和图像识别模型（如ResNet），并辅以大语言模型（LLM）进行交叉验证，能够有效提升虚假新闻检测的准确性和鲁棒性。同时，通过开发用户友好的交互界面，使普通

用户也能便捷地使用该技术，增强其对信息的辨别能力。

本研究不仅是对多模态学习和深度学习技术在特定应用场景的一次探索，更希望能为净化网络信息环境、提升社会媒介素养贡献一份微薄之力。我们深知虚假新闻的治理是一项长期而艰巨的任务，但相信通过持续的技术创新和多方协作，能够逐步构建起更为健康和可信的信息生态系统。

第1章　绪论

1.1　研究背景与意义

进入21世纪以来，信息技术的革新浪潮以前所未有的力量重塑着人类社会的面貌。特别是以互联网为基础的社交媒体，如国外的Twitter、Facebook，国内的微信、微博等平台的兴起与繁荣，彻底改变了传统的信息生产与传播模式[1]。这些平台凭借其即时性、互动性、低门槛和广泛覆盖的特性，使得每个人既是信息的消费者，也成为了信息的潜在创造者和传播者。信息的获取变得空前便捷，思想的交流也突破了时空的限制，社会公众的参与感和话语权得到了显著提升。据统计，截至2023年底，全球社交媒体用户数量已突破49亿，日均使用时长超过2.5小时[2]，社交媒体已深度融入现代人的日常生活与工作中。

然而，社交媒体这把“双刃剑”在赋予信息传播强大动能的同时，也为其生态带来了严峻的挑战，其中最为突出的便是虚假新闻（Fake News）的泛滥问题。虚假新闻，通常指那些以欺骗公众、获取不正当利益或达到特定政治、经济目的为动机，通过伪造事实、歪曲真相、断章取义等手段制作，并借助各种传播渠道散布的错误或误导性信息[3]。这些信息往往披着“新闻”的外衣，利用耸人听闻的标题、煽动性的言辞、甚至配以伪造或不相关的图片视频，极易在短时间内引发大规模的病毒式传播。

虚假新闻的危害是多方面的且深远的。在社会层面，它能够误导公众认知，制造社会恐慌，破坏社会信任，甚至煽动群体对立，引发社会不稳定事件[4]。例如，在重大公共卫生事件（如COVID-19大流行期间）中，关于病毒来源、防治措施的谣言层出不穷，严重干扰了科学防疫工作的开展，威胁了公众生命健康安全[5]。在政治层面，虚假新闻被用作影响选举、干预政治进程、抹黑政治对手的工具，损害了民主政治的公正性和透明度。在经济层面，针对企业或市场的恶意谣言可能导致股价异常波动，破坏正常的市场秩序。此外，虚假新闻还可能对个人名誉造成侵害，引发网络暴力等问题。

面对虚假新闻的严峻态势，传统的应对机制，如依赖专业新闻机构进行事实核

查、政府部门发布辟谣信息、平台运营方进行人工内容审核等，虽然发挥了一定作用，但在信息爆炸式增长、传播速度极快的社交媒体环境下，这些方法往往显得滞后和低效。人工审核成本高昂，且难以覆盖所有信息；事后辟谣则难以完全消除虚假信息已经造成的影响。因此，研究和开发自动、高效、精准的虚假新闻检测技术，对于从源头上遏制虚假信息的传播，维护健康有序的网络信息生态，保障社会公共利益，提升国家治理体系和治理能力现代化水平，都具有至关重要的理论意义和现实价值。本研究正是致力于通过应用先进的多模态深度学习技术，构建一个智能化的虚假新闻检测系统，以期为解决这一全球性难题贡献一份力量。

1.2 国内外研究现状

虚假新闻检测作为信息内容安全领域的重要分支，近年来已成为学术界和工业界共同关注的研究热点。研究者们从不同角度出发，利用自然语言处理、机器学习、深度学习、数据挖掘等技术，探索了多种检测方法。

1.2.1 单模态虚假新闻检测方法及其局限性

早期的虚假新闻检测研究主要集中在对新闻文本内容的分析，即单模态检测。这些方法可以大致分为基于内容特征和基于传播特征两类。

基于内容特征的方法试图从新闻文本本身挖掘线索。传统机器学习方法通常依赖于人工设计的特征，如词袋模型、TF-IDF、N-gram等语言学特征，以及情感极性、主观性、可读性等文体特征[6,7]。然后，利用支持向量机（SVM）、逻辑回归（LR）、朴素贝叶斯（NB）、决策树等分类器进行真伪判断。随着深度学习的兴起，卷积神经网络（CNN）[8]、循环神经网络（RNN）及其变体如长短期记忆网络（LSTM）[9]和门控循环单元（GRU）被广泛应用于文本特征的自动学习，它们能够更好地捕捉文本的序列信息和上下文依赖关系。近年来，基于Transformer架构的预训练语言模型，如BERT (Bidirectional Encoder Representations from Transformers)[10]、RoBERTa[11]、XLNet[12]等，通过在大规模无标注语料上进行预训练，学习到了丰富的语言知识，在迁移到虚假新闻检测等下游任务时取得了显著的

性能提升，成为当前纯文本检测领域的主流方法。

基于传播特征的方法则关注新闻在社交网络中的传播模式，如传播路径的结构、传播速度、用户参与行为（点赞、评论、转发）等[13, 14]。这类方法认为虚假新闻与真实新闻在传播方式上存在显著差异。

尽管单模态文本检测取得了一定的进展，但其局限性也日益凸显。首先，虚假新闻的制造者会刻意模仿真实新闻的写作风格，使得仅从文本语言层面难以有效区分。其次，大量的虚假新闻利用篡改、拼接或与文本内容不符的图像、视频来增强其欺骗性和传播力，纯文本分析无法处理这些视觉信息引入的干扰。最后，社交媒体上的信息往往是文本、图像、用户评论等多种模态的有机组合体，忽略其他模态的信息会导致对新闻事件的理解片面，从而影响检测的准确性和鲁棒性。

1.2.2 多模态虚假新闻检测研究进展

为了克服单模态检测的局限性，研究者们开始积极探索多模态虚假新闻检测方法，旨在通过融合来自不同信息源（如文本、图像、视频、音频、用户社交关系、传播网络等）的特征来进行更全面的判断。

早期的多模态融合方法相对简单，例如特征级融合（Early Fusion），即在输入层面对不同模态的原始特征或浅层特征进行拼接或加权组合，然后输入到统一的分类器中[15]。另一种是决策级融合（Late Fusion），即分别对每个模态进行独立建模和预测，然后对各个模态的预测结果进行投票或加权平均，得到最终的判决[16]。这些方法虽然在一定程度上利用了多模态信息，但往往难以充分捕捉不同模态之间的复杂交互和互补关系。

随着深度学习技术的发展，更复杂的中间层融合（Intermediate Fusion）或混合融合（Hybrid Fusion）策略逐渐成为主流。这些方法致力于在模型的中间层面对不同模态的深度特征表示进行交互和融合。例如，一些工作利用双线性池化、张量融合等技术来学习模态间的细粒度关联[17]。注意力机制（Attention Mechanism）的引入为多模态融合提供了更灵活和有效的途径，它允许模型动态地学习不同模态以及模态内部不同特征的重要性，从而更好地整合互补信息并抑制噪声干扰[18, 19]。

近年来，基于Transformer架构的多模态模型，如ViLBERT,[20] LXMERT,[21] UNITER[22]等，在视觉-语言理解任务上取得了巨大成功，也为多模态虚假新闻检测提供了新的思路。这些模型通常通过跨模态注意力机制实现文本和视觉特征的深度交互与融合。

1.2.3 基于深度学习的多模态检测方法综述

当前，基于深度学习的多模态虚假新闻检测模型展现出强大的潜力，其核心在于如何有效地提取各模态的特征并进行深度融合。根据特征融合方式和网络结构的不同，可以将这些方法归纳为以下几类：

1. 基于特征拼接的融合模型： 这是最直接的融合方式，将从不同模态提取的特征向量简单地拼接在一起，然后送入后续的全连接层或分类器进行处理。例如，分别使用CNN提取图像特征，使用LSTM或BERT提取文本特征，然后将两者拼接[23]。这种方法的优点是简单易实现，但可能忽略了模态间的复杂交互。

2. 基于协同表示学习的融合模型： 此类模型旨在将不同模态的信息映射到一个共享的潜在语义空间中，或者学习模态间的一致性表示。通过最小化不同模态表示之间的距离或最大化其相关性，使得模型能够捕捉跨模态的共同信息。

3. 基于注意力机制的融合模型： 这是当前研究的热点和主流方向。注意力机制能够动态地为不同模态的特征分配权重，突出重要信息，抑制无关信息。具体可分为模态内注意力（计算单一模态内部不同部分的重要性）、跨模态注意力（学习一个模态对另一个模态中不同部分的影响）以及共同注意力（Co-attention，同时学习两个模态之间的相互关注关系）[24, 25]。一些模型采用多头注意力机制来从不同角度捕捉模态间的关联。

4. 基于Transformer的端到端融合模型： 借鉴Transformer在自然语言处理和计算机视觉领域的成功，研究者开始将其应用于多模态特征的深度融合。通过构建包含自注意力和跨模态注意力层的Transformer编码器，可以直接对拼接后的多模态序列进行建模，学习模态间的深层交互关系。

5. 基于图神经网络（GNN）的融合模型： 除了内容本身的文本和图像模态，一些研究还考虑了新闻的传播结构。通过将新闻、用户、评论等实体构建成图，利用GNN来学习节点表示和图结构信息，并将其与内容特征进行融合，从而从社交上下文的角度辅助判断[26]。

本研究将重点关注基于深度特征提取（BERT和ResNet）并结合有效的特征融合策略（如特征拼接后通过注意力机制或全连接层进行深度融合）的多模态模型设计。

1.2.4 现有系统与工具的分析

随着虚假新闻检测技术的发展，国内外也涌现出一些相关的系统和工具，旨在为用户提供事实核查服务或为平台提供内容审核支持。

国际上，知名的事实核查网站如Snopes、PolitiFact、FactCheck.org等，主要依靠专业的记者和研究员团队对可疑信息进行人工调查和核实，并发布核查报告。这些平台在提供权威信息方面发挥了重要作用，但其覆盖范围和响应速度受限于人力。

科技巨头如Google、Facebook（Meta）、Twitter（X）等也投入资源开发AI技术用于检测和限制虚假信息的传播。例如，Google推出了Fact Check Explorer，汇集了全球事实核查机构的核查结果；Facebook利用AI技术识别和标记潜在的虚假内容，并与第三方事实核查机构合作。这些平台的优势在于拥有海量数据和强大的技术研发能力，但其具体算法和模型的细节往往不公开，且面临用户隐私、言论自由边界等复杂问题。

学术界也孵化了一些原型系统和工具。例如，一些研究团队开发了基于特定数据集和模型的虚假新闻检测API或Web应用，用于学术演示和验证。然而，这些系统往往功能较为单一，或者用户体验和系统稳定性有待提升，距离大规模实际应用仍有差距。

总体而言，现有的虚假新闻检测系统和工具在以下方面仍存在提升空间：

- 多模态信息处理的深度和广度： 多数系统仍偏重文本分析，对图像、视频等多媒体内容的深层语义理解和跨模态关联分析能力有待加强。

- 模型的透明度与可解释性： 许多基于深度学习的系统如同“黑箱”，用户难以理解其做出判断的具体原因，这影响了系统的可信度和用户接受度。引入可解释性AI（XAI）技术是未来的一个重要方向。
- 用户交互与友好性： 需要设计更便捷、直观的用户界面，方便普通用户提交检测请求、理解检测结果并获取相关背景信息。
- 动态适应与持续学习： 虚假新闻的制造手段和传播策略不断演变，检测系统需要具备持续学习和快速适应新变化的能力。
- 跨领域与跨语言的泛化能力： 许多模型在特定数据集或语言环境下表现良好，但在更广泛的场景下的泛化能力仍需检验和提升。

本研究旨在构建一个集成了多模态分析、LLM交叉验证，并提供友好Web交互界面的虚假新闻检测系统，力求在检测准确性、结果可参考性以及系统实用性方面进行有益的探索。

1.3 主要研究内容

本毕业设计的核心目标是研发一个能够自动检测社交媒体平台上虚假新闻的多模态深度学习系统。为实现这一目标，主要研究内容将围绕以下几个关键方面展开：

1. 数据集构建与高质量预处理： 针对社交媒体虚假新闻的多模态特性，本研究将从权威的公开数据源（如MCFEND、rumor_detection_acl2017、SocialNet等）系统性地收集和整理包含新闻标题、正文文本以及相关配图的多模态数据样本。随后，将对原始数据集进行严格的清洗与标准化预处理。文本数据方面，将进行去除非法字符、去除停用词（视模型需求而定）、文本规范化（如繁简体转换、大小写统一）等操作。图像数据方面，将对收集到的图片进行统一的格式转换（如统一为RGB格式）、尺寸调整以适应模型输入要求，并进行归一化处理。同时，将对多模态数据进行有效的标注，确保标签的准确性，为后续模型训练提供可靠且高质量的数据基础。

2. 多模态深度学习模型的精心设计与优化： 本研究将设计并实现一个能够有效融合文本与图像特征的深度学习模型架构。具体而言，文本特征提取模块将选用在自然语言处理领域表现优异的BERT（Bidirectional Encoder Representations from Transformers）预训练模型（例如`bert-base-chinese`），以充分捕捉新闻文本的深层语义信息和上下文依赖关系。图像特征提取模块将采用经典的卷积神经网络ResNet50，它以其深层残差学习能力在图像识别任务中取得了广泛成功，能够有效提取图像的视觉表征。 在此基础上，将重点研究和实现高效的多模态特征融合策略。初步考虑采用特征级联（Concatenation）的方式将文本特征向量和图像特征向量进行拼接，然后通过一个或多个全连接层（Fusion Layer）进行深度融合，并引入激活函数（如ReLU）和Dropout机制以增强模型的非线性表达能力和防止过拟合。最终，融合后的特征将送入一个分类器（如Sigmoid激活的单输出全连接层）来预测新闻的真实性概率。后续优化可能包括引入注意力机制来动态调整不同模态特征的权重。

3. 集成大语言模型（LLM）进行交叉验证与可解释性增强： 为进一步提升检测结果的置信度和可解释性，本系统将集成LLM交叉验证模块。在多模态模型给出初步判断后，系统将调用外部大语言模型API（如通过OpenRouter平台访问多个LLM，包括`perplexity/sonar-reasoning-pro`, `google/gemini-2.0-flash-001`等）对新闻的核心文本内容，乃至图文结合的信息进行事实核查和分析。LLM的反馈（如判断结果、置信度、理由摘要）将作为一项重要的参考，与多模态模型的输出结果进行加权融合或逻辑判断，从而给出更全面、更可靠的最终检测结论。此举不仅有助于提高检测准确率，也能为用户提供更丰富的判断依据。

4. 稳健后端服务的开发与高效接口设计： 系统的后端服务将基于Python编程语言和成熟的Django Web框架进行构建。主要职责包括：

    - 用户认证与管理： 实现用户注册、安全登录（采用JWT Bearer Token认证机制）、个人信息修改、权限管理等功能。

- 新闻检测处理流程： 设计并实现API接口，用于接收用户通过前端上传的新闻标题、正文文本及图像文件。后端接收到请求后，将调度已训练好的多模态深度学习模型进行推理，并调用LLM交叉验证模块。
- 结果反馈与数据存储： 将模型的检测结果（如真实/虚假标签、置信度分数）以及LLM的分析详情进行结构化处理，并通过API接口返回给前端。同时，将用户信息、提交的检测任务、检测结果、分析详情等关键数据持久化存储到MySQL关系型数据库中，以便后续查询和管理。
- 系统管理与配置： 为管理员提供接口，用于配置系统级参数，例如多模态模型与LLM结果融合的权重、判断虚假新闻的置信度阈值、外部API密钥的管理等。

所有API接口将遵循RESTful设计原则，确保接口的清晰性、一致性和易用性。

5. 用户友好的前端界面设计与交互实现： 系统的前端将采用现代化的Vue.js渐进式JavaScript框架，并结合ElementUI组件库，构建一个响应式、交互友好且易于操作的Web应用程序。主要界面和功能将包括：

    - 公共访问界面： 设计引人入胜的系统介绍首页，以及便捷的登录和注册页面。
    - 用户核心功能区：
        - 仪表盘： 用户登录后可见，展示个性化的检测统计摘要，如总检测次数、真/假新闻比例等。
        - 新闻检测提交： 提供清晰的表单，供用户输入新闻标题、粘贴新闻正文，并支持上传相关图像文件。
        - 检测结果展示： 在用户提交检测后，实时或异步展示检测结果，包括明确的真/伪判断、置信度得分，以及来自LLM的详细分析理由。
        - 历史检测记录： 用户可以方便地查询、筛选和管理自己提交过的所有检测记录，并查看每条记录的详细结果。

- 个人中心： 用户可以查看和修改个人基本信息，以及更改登录密码。

• 管理员专属后台：

- 管理控制面板： 汇总展示系统整体的运行状态、用户活跃度、检测量趋势等关键指标。
- 用户管理： 管理员可以查看所有注册用户列表，并进行用户账户的启用、禁用、编辑（如重置密码）或删除操作。
- 检测记录管理： 管理员可以查看系统内所有的检测记录，进行筛选、排序和详情查阅。
- 检测统计分析： 利用Echarts等数据可视化库，将历史检测数据以图表（如饼图、折线图）的形式进行多维度展示，帮助管理员洞察虚假新闻的特征和趋势。
- 系统参数设置： 提供界面化的方式，供管理员调整模型融合权重、API密钥、判断阈值等系统级参数。
- 系统日志查看： 方便管理员追踪系统运行状态，排查潜在问题。

通过以上五个方面的深入研究与细致实现，本毕业设计期望能够构建一个功能全面、性能可靠、用户体验良好的多模态虚假新闻检测系统。

1.4 论文组织结构

本论文系统地阐述了基于多模态深度学习的社交媒体虚假新闻检测系统的设计与实现过程，共分为七章，其主要内容安排如下：

第一章：绪论。 本章首先介绍了研究的时代背景，即社交媒体的普及与虚假新闻的泛滥带来的挑战，阐明了进行虚假新闻自动检测研究的重要性和现实意义。随后，对国内外在单模态及多模态虚假新闻检测领域的相关研究现状进行了综述，分析了现有方法和系统的优势与不足。在此基础上，明确了本毕业设计的主要研究内容、拟采用的技术路线和预期达成的目标。最后，概述了本论文的整体章节安排。

第二章：相关技术概述。 本章对支撑本系统设计与实现的核心技术进行了较为详细的介绍。内容涵盖虚假新闻的基本定义、特征及其传播模式；多模态学习的基本理论，包括信息表示和融合策略；深度学习中的关键模型，重点介绍了用于文本处理的BERT模型和用于图像处理的ResNet模型的工作原理与结构；阐述了大语言模型（LLM）在事实核查中的应用及其在本系统中的交叉验证作用；最后，概述了系统开发所采用的Web技术栈，包括后端Python Django框架、前端Vue.js框架、MySQL数据库以及RESTful API设计原则和JWT用户认证机制。

第三章：系统需求分析与设计。 本章首先对系统的各项需求进行了细致的分析，包括用户功能需求（如新闻上传检测、结果查看、历史管理等）和管理员功能需求（如用户管理、参数配置等），以及系统的非功能性需求（如性能、易用性、安全性等）。随后，基于需求分析，进行了系统总体架构设计，明确了前后端分离的架构模式和主要的系统模块划分。接着，详细阐述了数据库设计过程，包括概念结构设计（E-R图）和逻辑结构设计，并给出了核心数据表的结构定义。最后，对系统中主要的API接口进行了设计说明。

第四章：多模态虚假新闻检测模型构建。 本章是论文的核心技术章节，详细介绍了多模态虚假新闻检测模型的设计与实现。首先，描述了所使用的数据集来源（MCFEND、rumor_detection_acl2017、SocialNet）、数据预处理流程（包括文本清洗、图像处理与规范化）以及数据集的划分策略。其次，重点阐述了多模态深度学习模型的网络结构，包括文本特征提取模块（BERT）、图像特征提取模块（ResNet）以及特征融合模块的设计思路和具体实现。然后，详细描述了模型的训练过程，包括实验环境配置、损失函数的选择（BCEWithLogitsLoss）、优化器（AdamW）的选择、学习率设置及调度策略、训练超参数的设定以及防止过拟合的措施（如Dropout、早停）。最后，介绍了模型的评估指标（准确率、精确率、召回率、F1值、AUC），并展示了模型在测试集上的实验结果及分析。

第五章：系统实现。 本章详细记录了整个系统的具体开发和实现过程。在后端实现部分，描述了Django项目的搭建、各个应用（app）的创建，以及用户认证、新闻检测逻辑、模型调用、LLM集成、数据库交互等核心功能的代码实现。在前端实现部分，阐述了Vue.js项目的构建、路由配置、状态管理（Vuex）以及各个用户界面

和管理员界面的组件设计与功能实现，包括数据请求、响应处理和可视化图表的集成。此外，还简要说明了系统可能的部署方案。

第六章：系统测试。 为验证系统的功能完整性、性能表现和用户体验，本章设计并执行了一系列测试。首先介绍了测试环境（硬件、软件、浏览器等）和所使用的测试工具。然后，针对系统的核心功能模块（用户模块、新闻检测模块、管理员模块等）设计了详细的测试用例，并记录了测试过程和结果。此外，还进行了简要的性能测试（如接口响应时间）和兼容性测试（如不同浏览器的表现）。最后，对测试结果进行了分析与总结，评估了系统是否达到设计目标。

第七章：总结与展望。 本章对整个毕业设计的工作进行了全面总结，回顾了完成的主要研究内容、实现的系统功能以及模型取得的效果。提炼了本系统在多模态信息融合、LLM交叉验证应用、用户交互设计等方面的创新点与特色。同时，客观分析了当前系统存在的不足之处，例如数据集规模与多样性的局限、模型可解释性的进一步提升空间等。最后，对未来值得深入研究的方向进行了展望，如探索更先进的融合算法、引入更多模态信息、提升系统对新型虚假新闻的适应能力等。

1.5 本章小结

本章作为论文的开篇，系统地阐述了本研究的动机和意义。首先，通过分析社交媒体时代信息传播的特点，指出了虚假新闻泛滥所带来的严峻挑战，从而强调了开发自动化、智能化虚假新闻检测技术的必要性。其次，本章回顾了国内外在虚假新闻检测领域的研究进展，梳理了从单模态到多模态、从传统机器学习到深度学习的技术演变路径，并分析了现有检测方法和系统的优势与不足，为本研究的定位提供了参考。在此基础上，明确了本毕业设计的主要研究内容，即构建一个融合文本与图像信息，并集成大语言模型进行交叉验证的多模态深度学习检测系统，涵盖了从数据准备、模型构建到系统前后端开发与实现的完整流程。最后，本章清晰地勾勒了论文的整体组织结构，为后续各章节内容的展开提供了导引，旨在使读者能够系统地理解本研究的全貌。

第2章　相关技术概述

为了有效设计并实现一个先进的多模态虚假新闻检测系统，本研究综合运用了多个领域的前沿技术。本章将对这些核心相关技术进行概述，为后续章节中系统设计、模型构建和功能实现的具体阐述奠定理论基础。内容将主要包括虚假新闻的基本概念与特征，多模态学习的原理与方法，支撑本系统模型的核心深度学习算法（特别是BERT和ResNet），大语言模型（LLM）在信息核查中的应用，以及系统开发所采用的Web技术栈（Django、Vue.js、MySQL等）。

2.1　虚假新闻检测概述

虚假新闻检测是自然语言处理、数据挖掘和机器学习领域的一个重要研究方向，旨在通过技术手段自动识别和判定信息的真实性。

2.1.1　虚假新闻的定义与特征

虚假新闻（Fake News）并非一个全新的概念，但其在数字时代的表现形式和传播影响力达到了前所未有的程度。学术界对虚假新闻的定义尚未完全统一，但通常指那些故意制造和传播的、以新闻形式呈现的、旨在误导受众以获取某种利益（如政治、经济、声誉等）的虚假或不准确信息[3, 27]。它与无意的错误报道（Erroneous Reporting）或讽刺性新闻（Satire/Parody）有所区别，其核心在于“意图欺骗”。

虚假新闻通常表现出以下一些典型特征:[28]

- 内容层面：
  - 标题党（Clickbait Titles）： 使用夸张、耸人听闻或具有强烈情感色彩的标题吸引眼球。
  - 事实歪曲与捏造（Factual Distortion/Fabrication）： 包含完全虚构的内容，或对真实事件进行歪曲、断章取义。

- 缺乏可靠信源（Lack of Credible Sources）： 通常不注明信息来源，或引用匿名、伪造的信源。
- 情感化与煽动性语言（Emotional and Inflammatory Language）： 大量使用带有强烈主观情感、煽动性或歧视性的词汇。
- 视觉信息误用（Misuse of Visual Information）： 配图与文本内容不符、图片被篡改（如Photoshop处理）、旧图新用或将图片置于错误的上下文中。

• 传播层面：

- 病毒式传播（Viral Spread）： 借助社交媒体的分享机制，能够在短时间内迅速扩散。
- 机器人账户与水军参与（Bot and Troll Involvement）： 常有大量自动化账户或有组织的网络水军参与传播，以扩大影响。
- 同质化社群传播（Echo Chambers and Filter Bubbles）： 倾向于在观点相似的社群内部快速传播，并进一步固化已有偏见。

• 来源层面：

- 模仿权威媒体（Imitation of Legitimate News Outlets）： 网站域名、页面设计等可能刻意模仿知名新闻机构，以增加欺骗性。
- 匿名或虚假作者（Anonymous or Fake Authors）： 缺乏明确的、可追溯的作者信息。

准确识别这些特征是构建有效虚假新闻检测系统的关键。

2.1.2 虚假新闻的传播模式

理解虚假新闻在社交媒体上的传播模式对于设计有效的干预和检测策略至关重要。研究表明，虚假新闻的传播与真实新闻在多个方面存在差异[29]：

- 传播速度与广度： 虚假新闻，尤其是新奇或具有煽动性的虚假新闻，往往比真实新闻传播得更快、更广、更深。这可能与人类对新奇信息和负面情绪的偏好有关。
- 传播路径： 虚假新闻的传播路径通常呈现出更强的“广播”效应，即由少数节点扩散到大量用户，而真实新闻的传播路径可能更为分散。
- 用户反应： 用户对虚假新闻的情感反应（如惊讶、厌恶）通常比对真实新闻更为强烈，这可能进一步促进了其分享和传播。
- 生命周期： 一些虚假新闻可能在短时间内爆发，然后迅速消退，而另一些则可能反复出现，形成“周期性谣言”。

虽然本系统主要关注基于内容的多模态检测，但理解传播模式有助于从更宏观的视角认识虚假新闻问题，并为未来集成传播特征提供思路。

2.2 多模态学习基础

多模态学习（Multimodal Learning）是机器学习的一个重要分支，致力于研究如何利用来自多种不同信息源或表示形式（即“模态”）的数据进行学习[30]。在虚假新闻检测场景中，文本、图像、视频、音频、用户评论、社交网络结构等都可以被视为不同的模态。多模态学习的核心目标是通过整合这些不同模态的信息，实现比单一模态学习更优的性能，或者完成单一模态无法完成的任务。

2.2.1 多模态信息表示

多模态信息表示是将来自不同模态的原始数据转换为适合机器学习模型处理的数值化特征向量的过程。每种模态通常有其独特的特征提取方法：

- 文本表示： 从早期的词袋模型、TF-IDF，到基于词嵌入的Word2Vec[31]、GloVe[32]，再到当前主流的基于Transformer的上下文词嵌入（如BERT[10]的输出向量）。这些方法旨在捕捉文本的语义和句法信息。

- 图像表示： 传统方法包括SIFT、SURF等局部特征描述子。深度学习时代，通常使用预训练的卷积神经网络（CNNs），如VGG[33]、ResNet[34]、Efficient-Net[35]等，提取图像的深层视觉特征（通常是全连接层之前或全局平均池化层的输出）。

- 其他模态： 对于视频，可以分解为图像帧序列和音频流，分别提取视觉和听觉特征；对于社交网络，可以使用图嵌入方法学习节点（用户、新闻）的表示。

有效的多模态信息表示是后续模态融合和模型学习成功的关键前提。本系统主要关注文本和图像模态的表示。

2.2.2 多模态信息融合策略

多模态信息融合是指将从不同模态提取的特征表示进行组合，以形成一个统一的、信息更丰富的表示，供后续的分类或回归任务使用。常见的融合策略包括[36, 37]：

- 早期融合（特征级融合，Early Fusion）： 在模型输入的早期阶段，将不同模态的原始特征或浅层特征进行简单的拼接（Concatenation）或逐元素运算（如加权平均、乘积）。融合后的特征被送入一个统一的模型进行学习。优点是简单直接，但可能难以处理模态间的异质性和时间不同步性。

- 晚期融合（决策级融合，Late Fusion）： 首先为每个模态独立训练一个模型并得到各自的预测结果（如类别概率或决策分数），然后在决策层面对这些结果进行融合，例如通过投票、加权平均、或训练一个元分类器（Meta-classifier）。优点是各模态模型可以独立设计和优化，但可能忽略了模态间在特征层面的早期交互。

- 中间融合（混合融合，Intermediate/Hybrid Fusion）： 这是目前研究较多且更灵活的策略。它允许不同模态的特征在模型的中间层进行交互和融合。例

如，可以分别使用深度网络提取各模态的高层抽象特征，然后在某个中间层将这些特征融合。这种策略可以通过引入注意力机制、双线性池化、张量融合等更复杂的机制来学习模态间的细粒度交互。

- 注意力机制（Attention Mechanism）： 允许模型动态地关注不同模态或同一模态内部的不同部分，从而赋予重要特征更高的权重。跨模态注意力可以学习一个模态如何影响另一个模态的表示。
- 门控机制（Gating Mechanism）： 通过学习门控单元来控制不同模态信息的流动和组合比例。
- 多模态双线性池化（Multimodal Bilinear Pooling）： 通过计算两个模态特征向量的外积来捕捉它们之间的二阶交互信息。

本系统在初步设计中采用了相对简单的特征级联后的全连接层融合，后续可以考虑引入更复杂的注意力机制来优化融合效果。

2.3 核心深度学习模型

本系统主要依赖于预训练的深度学习模型进行文本和图像特征的提取，它们是实现高精度多模态检测的基础。

2.3.1 文本处理模型：BERT (Bidirectional Encoder Representations from Transformers)

BERT (Bidirectional Encoder Representations from Transformers)[10]是由Google在2018年提出的一个基于Transformer架构的预训练语言模型，它在自然语言处理（NLP）领域取得了革命性的突破。BERT的核心思想是通过在大规模无标注文本语料上进行“掩码语言模型”（Masked Language Model, MLM）和“下一句预测”（Next Sentence Prediction, NSP）两个预训练任务，学习到深层的双向语境表示。

其主要特点包括：

- Transformer编码器： BERT的主体结构是多层堆叠的Transformer编码器单元。每个编码器单元包含一个多头自注意力（Multi-Head Self-Attention）子层和一个前馈神经网络（Feed-Forward Network）子层。自注意力机制使得模型能够捕捉句子内部不同词语之间的长距离依赖关系。

- 双向语境表示： 与传统的从左到右或从右到左的单向语言模型不同，BERT通过MLM任务（随机遮盖输入文本中的一部分词语，然后预测这些被遮盖的词语）使得模型能够同时利用左右两侧的上下文信息来理解每个词的含义。

- 预训练与微调范式（Pre-training and Fine-tuning）： BERT首先在通用的大规模语料上进行预训练，学习通用的语言表示。然后，针对具体的下游任务（如文本分类、情感分析、问答等），只需在预训练好的模型顶部添加一个简单的输出层，并使用特定任务的标注数据进行微调（Fine-tuning），即可取得优异的性能。

在本系统中，选用的是`bert-base-chinese`模型，它是一个针对中文语料进行预训练的BERT基础版本，包含12层Transformer编码器，隐藏层维度为768，注意力头数为12。我们将利用BERT提取新闻文本的[CLS]标记对应的输出向量作为整个文本的语义表示。

2.3.2 图像处理模型：ResNet (Residual Networks)

ResNet (Residual Networks)[34]是由微软研究院在2015年提出的深度卷积神经网络架构，它通过引入“残差学习”（Residual Learning）机制，成功地解决了深度神经网络训练过程中的梯度消失和网络退化问题，使得训练非常深的网络成为可能。ResNet在ImageNet大规模视觉识别挑战赛（ILSVRC）中取得了优异的成绩，并成为后续许多计算机视觉任务的基础骨干网络。

其核心创新在于残差块（Residual Block）。传统的网络层学习的是目标映射$H(x)$，而残差块学习的是残差映射$F(x) = H(x) - x$，则原始的目标映射变为$F(x)+x$。这个恒等映射x通过“快捷连接”（Shortcut Connection）或“跳跃连

接”（Skip Connection）直接添加到后续层的输出上。这种结构使得网络更容易优化，即使网络层数很深，梯度也能够有效地反向传播。

本系统选用的是ResNet50模型，它包含50个卷积层（包括残差块内的卷积层）。我们将使用在ImageNet数据集上预训练好的ResNet50模型，去除其顶部的全连接分类层，将其作为一个图像特征提取器。输入图像经过ResNet50处理后，通常取全局平均池化层（Global Average Pooling）的输出或最后一个卷积块的输出作为图像的全局视觉特征向量。对于ResNet50，其输出特征维度通常是2048。

2.3.3 注意力机制与Transformer网络

（本小节内容已在2.3.1 BERT模型中有所体现，此处可简述其通用性或在多模态融合中的潜力）

注意力机制最初在机器翻译领域被提出[38]，用于解决长序列依赖问题。其核心思想是模拟人类的注意力分配过程，即在处理信息时，会有选择地关注输入序列中的重要部分，并赋予它们更高的权重。Transformer模型[39]则完全基于注意力机制（特别是自注意力机制）构建，摒弃了传统的循环和卷积结构，在并行计算能力和捕捉长距离依赖方面表现出色，已成为NLP和许多其他领域的主流架构。

在多模态学习中，注意力机制不仅可以用于单一模态内部的特征加权（如文本中的词语重要性），更重要的是可以实现跨模态注意力，即学习不同模态特征之间的对齐和依赖关系。例如，文本中的某个词语可能与图像中的特定区域相关联，跨模态注意力可以帮助模型捕捉这种关联。虽然本系统的初步设计中特征融合较为直接，但未来引入基于Transformer的跨模态融合模块是提升模型性能的一个重要方向。

2.4 大语言模型（LLM）交叉验证技术

近年来，以GPT系列[40]、LLaMA[41]等为代表的大语言模型（Large Language Models, LLMs）在自然语言理解、生成、推理等方面展现出惊人的能力。这些模型通常在海量的文本（甚至多模态）数据上进行预训练，参数量巨大，能够掌握丰

富的世界知识和复杂的语言模式。

2.4.1 LLM在事实核查中的应用

LLM的强大能力使其在事实核查（Fact-Checking）和虚假信息识别领域具有巨大的应用潜力。它们可以通过以下几种方式辅助判断信息的真实性：

- 知识检索与问答： LLM内部存储了大量的事实性知识，可以直接对新闻中的断言进行提问，并评估其回答与新闻内容的一致性。
- 文本摘要与关键信息提取： LLM可以快速总结新闻的核心内容，帮助识别关键断言，以便进行针对性核查。
- 逻辑一致性检查： LLM可以分析新闻文本内部是否存在逻辑矛盾或不合理之处。
- 生成核查理由： 一些先进的LLM不仅能给出真伪判断，还能生成支持其判断的理由或证据线索，提高了可解释性。
- 多角度分析： 可以通过设计不同的提示（Prompts），引导LLM从不同角度（如情感、来源可靠性、论证强度等）对新闻进行评估。

然而，LLM也存在一些局限性，如可能产生“幻觉”（Hallucinations，即生成看似合理但实则错误的信息）、对训练数据中的偏见敏感、以及对于训练截止日期之后的新事件知识更新不及时等问题。

2.4.2 OpenRouter平台及API调用

为了方便地调用和集成多种不同的LLM，本系统选择使用OpenRouter平台。OpenRouter（https://openrouter.ai/）是一个LLM API聚合器，它提供了一个统一的接口，允许开发者通过一个API密钥访问来自不同提供商（如OpenAI, Anthropic, Google, Mistral AI, Perplexity AI等）的多种LLM模型。这极大地简化了模型切换、成本管理和实验比较的流程。

通过OpenRouter API，我们可以向选定的LLM（如本系统中使用的perplexity/sonar-reasoning-pro, google/gemini-2.0-flash-001, deepseek/deepseek-r1等）发送包含待核查新闻内容的提示，并接收模型返回的JSON格式的分析结果，包括判断（如“真实”、“虚假”）、置信度分数和分析理由。本系统的detection/services/llm_verifier.py模块封装了对OpenRouter API的异步调用逻辑。

2.4.3 集成多个LLM的策略

考虑到单个LLM可能存在的偏见或局限性，本系统采用集成多个LLM进行交叉验证的策略，以期获得更鲁棒和可靠的判断。如llm_verifier.py所示，系统会向多个不同的文本分析型LLM发送相同的核查请求。对于包含图像的新闻，还会调用支持视觉输入的LLM（如google/gemini-2.0-flash-001）进行图文综合分析。

收集到各个LLM的独立判断结果后，系统会采用一种聚合策略（如系统中定义的加权平均或投票机制）来形成一个综合的LLM判断结果和置信度。这种集成方法有助于平滑单个模型的极端判断，并综合考量多个“专家意见”。最终，这个LLM的综合判断结果将与本系统自研的多模态深度学习模型的输出进行最终融合，形成对新闻真实性的最终判定。

2.5 Web开发技术栈

为了将多模态虚假新闻检测模型封装成一个易于使用的Web应用程序，本系统采用了一套成熟且高效的Web开发技术栈。

2.5.1 后端技术：Python与Django框架

系统后端服务采用Python语言开发。Python以其简洁的语法、丰富的第三方库生态以及在数据科学和机器学习领域的广泛应用而成为理想的选择。

Web框架选用的是Django (https://www.djangoproject.com/)。Django是一个高级Python Web框架，它鼓励快速开发和干净、实用的设计。其主要特性包括：

- MTV架构： Django遵循模型（Model）、模板（Template）、视图（View）的设计模式，类似于MVC（Model-View-Controller），有助于代码的组织和解耦。
- ORM（对象关系映射）： Django内置了强大的ORM系统，允许开发者使用Python代码来定义数据库模型并与数据库进行交互，而无需直接编写SQL语句，支持多种数据库后端（如PostgreSQL, MySQL, SQLite, Oracle）。
- 自带管理后台： Django自动生成一个功能完善的管理后台（Admin Site），极大地方便了对数据的管理和维护。
- 安全性： Django内置了对常见Web安全威胁（如XSS攻击、CSRF攻击、SQL注入等）的防护机制。
- 可扩展性与丰富的组件： Django拥有庞大的社区和丰富的第三方应用（apps），可以方便地扩展系统功能。

在本系统中，Django负责处理用户请求、业务逻辑、模型调用、数据库交互以及提供RESTful API接口。

2.5.2 前端技术：Vue.js框架与ElementUI组件库

系统前端界面采用Vue.js（https://vuejs.org/）构建。Vue.js是一个渐进式JavaScript框架，以其轻量、易学、高效和灵活性而受到广泛欢迎。其核心库只关注视图层，易于与其他库或既有项目整合。Vue.js的主要特性包括：

- 组件化开发： 允许将界面拆分成可复用的小组件，提高了代码的可维护性和开发效率。
- 数据驱动视图： 基于响应式数据绑定，当数据变化时，视图会自动更新。
- 虚拟DOM： 通过虚拟DOM技术优化DOM操作，提升渲染性能。

- 丰富的生态系统： 拥有Vue Router（用于单页面应用路由管理）、Vuex（用于状态管理）等官方支持的库，以及大量的社区插件。

为了快速构建美观且功能丰富的用户界面，本系统选用了ElementUI（https://element.eleme.cn/）作为UI组件库。ElementUI是一套为开发者、设计师和产品经理准备的基于Vue 2.0的桌面端组件库，提供了大量高质量的预置组件（如表单、表格、对话框、导航菜单、图表等），大大加速了前端开发进程。

2.5.3 数据库技术：MySQL

系统的数据持久化存储选用MySQL（https://www.mysql.com/）关系型数据库。MySQL是目前最流行的开源关系型数据库管理系统之一，以其高性能、高可靠性、易用性和成熟的社区支持而著称。它能够很好地与Django框架集成，通过Django ORM可以方便地进行数据模型的定义和操作。在本系统中，MySQL将用于存储用户信息、新闻检测记录、系统配置参数等关键数据。

2.5.4 API设计：RESTful API与JWT认证

系统前后端之间的数据交互通过RESTful API（Representational State Transfer Application Programming Interface）进行。REST是一种软件架构风格，设计原则包括无状态、客户端-服务器分离、统一接口等。通过HTTP方法（GET, POST, PUT, PATCH, DELETE）对资源进行操作，并使用JSON作为主要的数据交换格式。Django REST framework（https://www.django-rest-framework.org/）是一个强大且灵活的工具包，用于构建Web API，本系统利用它来快速开发符合RESTful规范的接口。

用户认证方面，系统采用JSON Web Token（JWT）[42] 机制。JWT是一种开放标准（RFC 7519），它定义了一种紧凑且自包含的方式，用于在各方之间安全地传输信息作为JSON对象。当用户成功登录后，后端会签发一个JWT（通常是Access Token和Refresh Token），前端在后续的请求中通过HTTP Authorization头部（通常使用Bearer方案）携带Access Token来表明用户身份。JWT具有无状态、可扩展性好、适

用于分布式系统等优点。本系统使用djangorestframework-simplejwt库来实现JWT的签发与验证。

2.6 本章小结

本章详细介绍了构建基于多模态深度学习的社交媒体虚假新闻检测系统所涉及的一系列核心技术。首先，对虚假新闻的定义、典型特征及其在社交媒体上的传播模式进行了概述，明确了检测任务的复杂性。其次，阐述了多模态学习的基本原理，包括信息表示和融合策略，为后续模型设计提供了理论依据。接着，重点介绍了本系统选用的核心深度学习模型——用于文本处理的BERT模型和用于图像处理的ResNet模型，并简述了注意力机制与Transformer网络的重要性。此外，还探讨了大语言模型（LLM）在事实核查中的应用潜力，并说明了本系统如何通过OpenRouter平台集成多个LLM进行交叉验证。最后，对系统开发所依赖的Web技术栈进行了梳理，包括后端Python Django框架、前端Vue.js框架与ElementUI组件库、MySQL数据库以及RESTful API设计和JWT用户认证机制。通过对这些相关技术的理解，为后续章节中系统架构设计、模型具体实现、功能开发以及性能评估奠定了坚实的技术基础。

第3章　系统需求分析与设计

在详细介绍多模态虚假新闻检测模型的构建和系统具体实现之前，本章将首先对整个系统的需求进行全面分析，并在此基础上进行系统化的设计。清晰的需求分析和合理的系统设计是项目成功的基石，它能够确保最终开发的系统不仅满足预期的功能目标，还能在性能、用户体验等方面达到较高水平。本章内容将主要包括系统功能需求与非功能需求的梳理，系统总体架构与模块划分，关键技术选型，以及数据库和API接口的设计。

3.1　系统需求分析

系统需求分析旨在明确系统需要具备哪些功能，以及在运行过程中需要满足哪些性能、安全等方面的约束。本系统的需求主要来源于对虚假新闻检测业务场景的理解、用户（包括普通用户和管理员）的潜在期望以及现有相关系统的功能借鉴。

3.1.1　功能需求分析

系统的功能需求可以从普通用户和管理员两个角度进行划分。

1.　普通用户功能需求：　普通用户是系统的主要使用者，他们需要一个便捷的工具来辅助判断新闻的真实性。

- 用户注册与登录认证（FR-U01）：
  - 用户应能够通过提供用户名、邮箱、密码等信息完成新账户的注册。
  - 已注册用户应能够使用用户名和密码进行安全登录。
  - 系统应提供密码找回或重置功能（本次设计暂未作为核心实现，可作为未来展望）。
  - 用户登录状态应能被有效保持（如通过JWT）。
- 新闻内容提交与检测（FR-U02）：

- 用户应能够方便地输入或粘贴待检测新闻的标题和正文文本。
- 用户应能够选择性地上传与新闻相关的图像文件（如JPG, PNG, JPEG格式）。
- 系统应能在用户提交后启动后台检测流程。

• 检测结果展示与分析（FR-U03）：

- 系统应在检测完成后，清晰地向用户展示最终的检测结论（如“真实新闻”、“虚假新闻”、“无法确定”）。
- 系统应提供一个量化的置信度评分，以表示对检测结论的确信程度。
- （增强功能）系统应展示来自多模态模型和LLM交叉验证的详细分析过程或关键判断依据，以增强结果的可解释性。

• 历史检测记录管理（FR-U04）：

- 用户应能够查看自己提交过的所有新闻的检测历史列表。
- 历史列表应包含新闻标题、提交时间、检测状态、检测结果等关键信息。
- 用户应能够方便地筛选和搜索其历史检测记录。
- 用户应能够点击查看某条历史记录的详细检测结果和分析。
- 用户应能够删除自己的检测历史记录。

• 个人信息管理（FR-U05）：

- 用户应能够查看和修改自己的个人基本信息（如用户名、邮箱，部分信息如用户名可能限制修改）。
- 用户应能够修改自己的登录密码。

• 用户仪表盘（FR-U06）：

- 用户登录后，应能看到一个个性化的仪表盘，展示其个人的检测统计摘要，如累计检测次数、各类结果占比等。

- 仪表盘可提供快速访问常用功能的入口。

2. 管理员功能需求： 管理员负责系统的日常维护、用户管理和参数配置，以保障系统的稳定运行和持续优化。

- 管理员登录（FR-A01）：
  - 管理员应拥有独立的登录认证机制或通过特定权限标识与普通用户区分。
- 后台控制面板（FR-A02）：
  - 管理员登录后，应能访问一个集中的后台管理界面，展示系统整体运行状态、关键指标（如总用户数、总检测量、各类检测结果分布、系统资源占用情况等）。
  - 控制面板应提供对各管理模块的快速访问入口。
- 用户账户管理（FR-A03）：
  - 管理员应能够查看所有注册用户的列表，并支持按条件（如用户名、邮箱、状态）进行搜索和筛选。
  - 管理员应能够创建新的用户账户（可选）。
  - 管理员应能够编辑现有用户信息（如重置密码、修改角色）。
  - 管理员应能够禁用或激活用户账户。
  - 管理员应能够删除用户账户（需谨慎处理关联数据）。
- 检测记录管理（FR-A04）：
  - 管理员应能够查看系统内所有的检测记录，并支持按条件（如用户、时间范围、检测结果）进行高级搜索和筛选。
  - 管理员应能够查看任何一条检测记录的详细信息和分析过程。
  - （可选）管理员可能需要对某些检测结果进行人工复核或标注。

- 检测统计分析（FR-A05）：
  - 系统应为管理员提供多维度的检测数据统计分析功能。
  - 利用图表（如饼图、折线图、柱状图）直观展示检测量趋势、真假新闻比例、不同来源新闻的检测情况、用户活跃度等。
- 系统参数配置（FR-A06）：
  - 管理员应能够通过界面配置系统的一些关键运行参数，例如：
    * 多模态模型与LLM结果融合的权重。
    * 判断新闻为“虚假”或“真实”的置信度阈值。
    * 外部API（如OpenRouter）的密钥管理。
    * （可选）模型文件路径、默认检测模型选择等。
  - 参数修改应能实时或在下次应用启动时生效。
- 系统日志查看（FR-A07）：
  - 管理员应能够查看系统的运行日志（如应用日志、错误日志），以便监控系统状态和排查问题。

上述功能需求构成了本系统的核心骨架，后续的设计和实现将围绕这些需求展开。

3.1.2 非功能需求分析

除了明确的功能需求外，系统还需要满足一系列非功能性需求，以保证其质量和用户体验。

- 性能需求（NFR-P01）：
  - 检测响应时间： 用户提交检测请求后，系统应在合理的时间内返回检测结果。对于包含图像的多模态检测，考虑到模型推理的复杂性，目标响应时间可设定在数秒到数十秒级别（具体取决于模型大小和硬件资源）。

- 并发处理能力： 系统应能支持一定数量的用户同时在线并提交检测请求，后端服务应具备一定的并发处理能力。
- 页面加载速度： 前端Web页面的加载时间应尽可能短，以提供流畅的用户体验。

• 易用性需求（NFR-U01）：

- 界面直观友好： 系统界面设计应简洁、清晰、美观，符合用户操作习惯。
- 操作便捷高效： 用户应能通过简单的步骤完成新闻提交、结果查看、历史管理等操作。
- 信息反馈明确： 系统对于用户的操作应给予及时的、明确的反馈（如成功提示、错误警告、加载状态等）。
- 帮助与引导： （可选）提供必要的使用说明或提示，帮助用户理解系统功能和检测结果。

• 可靠性与稳定性需求（NFR-R01）：

- 系统稳定性： 系统应能长时间稳定运行，减少崩溃和故障的发生概率。
- 数据准确性与一致性： 检测结果和用户数据应准确存储，并在不同界面保持一致。
- 错误处理与容错性： 系统应能妥善处理各种预期的和意外的错误情况（如无效输入、网络异常、模型推理失败等），并给出友好提示，避免程序崩溃。

• 安全性需求（NFR-S01）：

- 用户认证与授权： 严格的用户身份验证机制，确保只有授权用户才能访问其数据和功能。管理员权限应与普通用户权限严格分离。

- 数据传输安全： （推荐）前后端通信使用HTTPS协议，保证数据在传输过程中的机密性。
- 数据存储安全： 用户密码等敏感信息应加密存储。数据库访问应有适当的权限控制。
- 防范常见Web攻击： 后端应采取措施防范常见的Web安全漏洞，如XSS（跨站脚本攻击）、CSRF（跨站请求伪造）、SQL注入等。Django框架已内置部分防护。
- 隐私保护： 用户提交的检测内容和个人信息应得到妥善保护，符合相关隐私政策法规（如有）。

• 可维护性需求（NFR-M01）：

- 代码规范清晰： 代码编写应遵循一定的规范，结构清晰，注释充分，便于理解和后续维护。
- 模块化设计： 系统应采用模块化设计，降低模块间的耦合度，方便独立修改和升级。
- 配置与部署便捷： 系统配置应尽量集中管理，部署过程应相对简单。

• 可扩展性需求（NFR-E01）：

- 系统架构设计应具备一定的灵活性，便于未来增加新的功能模块（如支持更多模态、引入新的检测模型）或扩展系统处理能力。

在后续的系统设计和实现过程中，将综合考虑这些非功能性需求，力求构建一个高质量的虚假新闻检测系统。

3.2 系统总体设计

基于上述需求分析，本节将对系统的整体架构、模块划分以及关键技术选型进行阐述。

3.2.1 系统架构设计

本系统采用当前主流的前后端分离架构进行设计。这种架构模式将用户界面（前端）与业务逻辑处理及数据存储（后端）彻底分离开来，两者通过定义良好的API接口进行通信。其主要优点包括：

- 职责清晰，关注点分离： 前端专注于用户体验和界面展示，后端专注于业务逻辑、数据处理和模型服务。开发团队可以并行工作，提高效率。
- 技术选型灵活： 前后端可以独立选择最适合自身需求的技术栈。
- 可扩展性强： 前后端可以独立扩展和升级。例如，后端可以演化为微服务架构，前端也可以适配不同的客户端（Web、移动App等）。
- 提升用户体验： 前端可以利用现代JavaScript框架实现更丰富的交互效果和更快的页面响应。

系统的整体架构如图3-1所示。

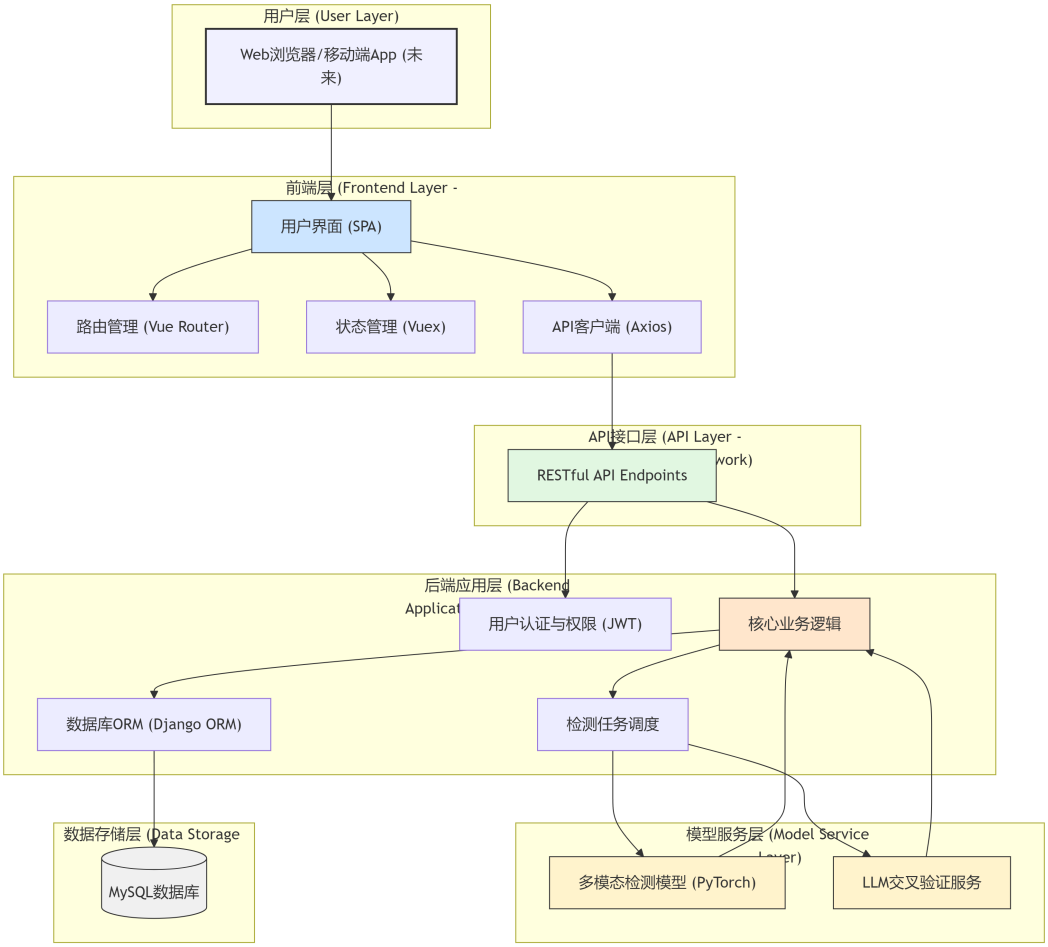


图 3-1 系统总体架构图

该架构主要包含以下几个层次：

- 用户层（User Layer）： 系统的最终用户，包括普通注册用户和管理员，通过Web浏览器与系统交互。
- 前端层（Frontend Layer）： 基于Vue.js构建的单页面应用（SPA），负责用户界面的渲染、用户输入的处理以及与后端API的异步通信。它向用户提供新闻提交、结果展示、历史查询、个人中心以及管理员后台等交互界面。
- API接口层（API Layer）： 基于Django REST framework构建的RESTful API，作为前后端数据交换的桥梁。定义了清晰的接口规范，用于处理用户认证、数

据增删改查、检测任务提交等请求。

- 后端应用层（Backend Application Layer）： 基于Django框架实现的核心业务逻辑。包括用户管理、权限控制、检测任务调度、多模态模型调用、LLM交叉验证逻辑、结果融合与分析、数据库操作等。
- 模型服务层（Model Service Layer）： 负责加载和运行训练好的多模态深度学习模型，接收来自应用层的推理请求并返回预测结果。此层也包含了与外部LLM API（如OpenRouter）的交互逻辑。
- 数据存储层（Data Storage Layer）： 采用MySQL数据库，负责持久化存储用户信息、检测记录、系统配置等数据。

3.2.2 模块划分

根据功能需求和系统架构，可以将系统划分为若干个主要的功能模块，如图3-2所示。

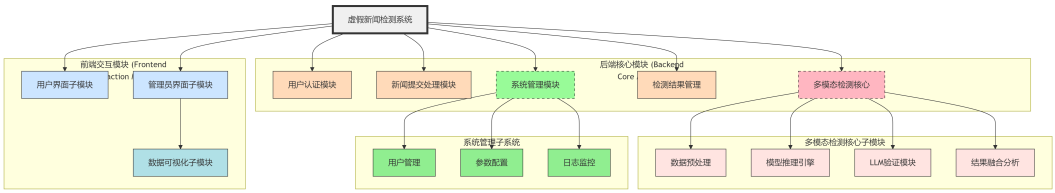


图 3-2 系统主要功能模块图

各主要模块的功能描述如下：

- 用户认证模块： 负责处理用户的注册、登录、登出、密码修改以及JWT的签发与验证。
- 新闻提交与处理模块： 负责接收用户提交的新闻文本和图像，进行初步校验，并创建检测任务。
- 多模态检测核心模块：

- 数据预处理子模块： 对输入的文本和图像进行清洗、格式化和特征编码，使其符合模型输入要求。
- 模型推理子模块： 加载预训练的多模态深度学习模型（BERT+ResNet+Fusion），对预处理后的数据进行推理，输出初步的真伪判断和置信度。
- LLM交叉验证子模块： 调用外部LLM API对新闻内容进行事实核查，获取LLM的判断、置信度和理由。
- 结果融合与分析子模块： 综合多模态模型和LLM的输出，根据预设策略（如加权、阈值判断）生成最终的检测结论和综合置信度，并记录分析过程。

- 检测结果管理模块： 负责将检测任务的详细信息、处理状态、中间结果、最终结论、分析详情等持久化存储到数据库，并提供查询接口。
- 前端交互模块：
  - 用户界面子模块： 实现用户注册/登录、新闻提交、结果展示、历史记录查看、个人中心等界面。
  - 管理员界面子模块： 实现管理员后台控制面板、用户管理、检测记录总览、统计分析、参数配置、日志查看等界面。
  - 数据可视化子模块： 利用Echarts等库将统计数据以图表形式展示。
- 系统管理模块（管理员）：
  - 用户账户管理子模块： 提供对系统所有用户账户的增删改查及状态控制功能。
  - 系统参数配置子模块： 允许管理员修改影响系统行为的关键参数。
  - 日志监控子模块： 提供查看系统运行日志的接口。

这种模块化的设计有助于降低系统的复杂度，提高代码的可维护性和可复用性。

3.2.3 技术选型与理由

本系统在技术选型上综合考虑了开发效率、社区支持、生态成熟度、性能要求以及开发团队的熟悉程度等因素。

- 后端编程语言：Python。 Python语法简洁，拥有强大的科学计算和机器学习库（如NumPy, Pandas, Scikit-learn, PyTorch, Transformers），非常适合数据密集型和AI驱动的应用开发。
- 后端Web框架：Django。 Django以其“开箱即用”的特性、完善的文档、强大的ORM、自带管理后台和良好的安全性，能够显著提高开发效率，适合快速构建稳健的Web应用。其成熟的生态系统也为集成各种功能提供了便利。
- 前端JavaScript框架：Vue.js。 Vue.js以其轻量、易上手、渐进式的特点，以及优秀的性能和活跃的社区，成为构建现代单页面应用的热门选择。其组件化思想有助于构建可维护的前端代码。
- 前端UI组件库：ElementUI。 ElementUI提供了丰富、美观且符合设计规范的Vue组件，可以大大减少前端界面的开发工作量，保证UI的一致性和专业性。
- 深度学习框架：PyTorch。 PyTorch以其灵活性、易用性（Pythonic）和强大的GPU加速能力，在学术界和工业界都得到了广泛应用，特别适合进行深度学习模型的快速原型设计和研究。Hugging Face Transformers库基于PyTorch（也支持TensorFlow）提供了大量预训练模型（如BERT），极大地方便了NLP任务的开发。
- 图像处理库：Pillow, OpenCV (可选)。 Pillow是Python中处理图像的基础库。OpenCV在需要更复杂的图像操作时可以作为补充。
- 数据库：MySQL。 MySQL作为一款成熟、稳定、高性能的开源关系型数据库，能够满足本系统对数据存储和查询的需求，且与Django有良好的集成。

- API标准：RESTful API。 RESTful API是当前Web服务设计的事实标准，简洁、易懂、易于扩展。
- 用户认证：JWT (JSON Web Token)。 JWT适用于前后端分离的无状态认证场景，具有安全性好、可扩展性强等优点。
- 数据可视化：Echarts。 Echarts是一个功能强大、配置灵活的开源可视化库，能够生成各种交互式图表，非常适合在本系统的前端（特别是管理员后台）进行数据分析结果的展示。
- LLM API调用：OpenAI Python Library (针对OpenRouter)。 使用官方或社区维护的库来与OpenRouter平台进行交互，以调用各种LLM。

这些技术的组合能够为系统提供一个稳定、高效且易于开发和维护的技术基础。

3.3 数据库设计

数据库是存储系统核心数据的关键部分。本节将对数据库进行概念结构设计和逻辑结构设计，并给出主要数据表的详细结构。

3.3.1 概念结构设计 (E-R图)

系统的主要实体包括用户（User）、检测记录（Detection）和系统设置（SystemSettings）。它们之间的关系如下：

- 一个用户可以有多条检测记录。
- 一条检测记录只属于一个用户。
- 系统设置是全局的，不直接与特定用户或检测记录关联，但会影响检测过程。

基于此，可以绘制出系统的E-R图，如图3-3所示。

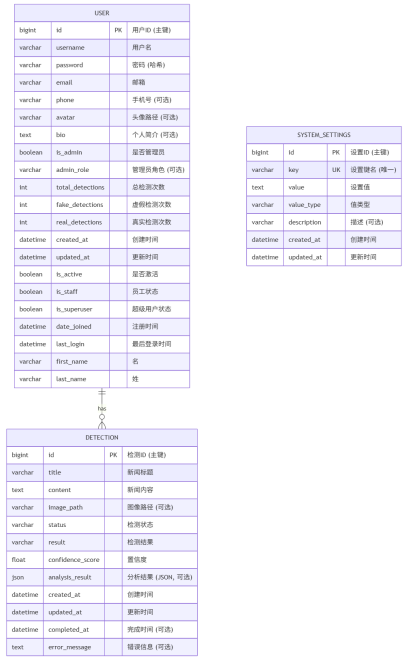


图 3-3 系统E-R图（概念模型）

主要实体及其属性：

- User（用户）：
  - 属性：用户ID（主键），用户名，密码（哈希），邮箱，手机号，头像路径，个人简介，是否管理员，管理员角色，总检测次数，虚假新闻检测次数，真实新闻检测次数，创建时间，更新时间，是否激活，员工状态，超级用户状态。
- Detection（检测记录）：
  - 属性：检测ID（主键），用户ID（外键，关联User），新闻标题，新闻内容，新闻图像路径，检测状态（待处理，处理中，已完成，失败），检测结果（真实，虚假，未知），置信度分数，详细分析结果（JSON），创建时间，更新时间，完成时间，错误信息。
- SystemSettings（系统设置）：

- 属性：设置ID（主键），设置键名，设置值，值类型（字符串，整数，浮点数，布尔，JSON），描述，创建时间，更新时间。

3.3.2 逻辑结构设计（关系模式）

根据E-R图，可以转换为以下关系模式（下划线表示主键，号表示外键）：

- User (id, username, password, email, phone, avatar, bio, is_admin, admin_role, total_detections, fake_detections, real_detections, created_at, updated_at, is_active, is_staff, is_superuser, date_joined, last_login, first_name, last_name)
- Detection (id, user_id, title, content, image, status, result, confidence_score, analysis_result, created_at, updated_at, completed_at, error_message)
- SystemSettings (id, key, value, value_type, description, created_at, updated_at)

3.3.3 主要数据表结构

以下是根据Django模型（`models.py` 文件内容）推导出的主要数据表结构。

1. 用户表（**users_user**）（基于users.models.User 和 Django AbstractUser）

表 3-1 用户表（users_user）结构

字段名	数据类型	约束	描述
id	BigAutoField	主键，自增	用户唯一标识
password	Varchar(128)	非空	哈希后的密码
last_login	Datetime	可空	最后登录时间
is_superuser	Boolean	非空，默认False	是否为超级管理员
username	Varchar(150)	非空，唯一	用户名
first_name	Varchar(150)	可空	名
last_name	Varchar(150)	可空	姓
email	Varchar(254)	非空，唯一	邮箱地址
is_staff	Boolean	非空，默认False	是否为员工（可访问 Admin）
is_active	Boolean	非空，默认True	账户是否激活
date_joined	Datetime	非空，自动创建	注册时间
phone	Varchar(15)	可空	手机号码
avatar	Varchar(100)	可空	头像文件路径
bio	TextField	可空	个人简介
is_admin	Boolean	非空，默认False	(自定义)是否为管理员
admin_role	Varchar(20)	可空	(自定义)管理员角色
total_-detections	PositiveIntegerField	非空，默认0	总检测次数
fake_detections	PositiveIntegerField	非空，默认0	虚假新闻检测次数
real_detections	PositiveIntegerField	非空，默认0	真实新闻检测次数
created_at	Datetime	非空，自动创建	记录创建时间
updated_at	Datetime	非空，自动更新	记录更新时间

2. 检测记录表（**detection_detection**）（基于detection.models.Detection）

表 3-2 检测记录表（detection_detection）结构

字段名	数据类型	约束	描述
id	BigAutoField	主键，自增	检测记录唯一标识
user_id	BigIntegerField	非空，外键关联users_-user.id	用户ID
title	Varchar(255)	非空	新闻标题
content	TextField	非空	新闻内容
image	Varchar(100)	可空	新闻图像文件路径
status	Varchar(20)	非空，默认’pending’	检测状态（pending，processing，completed，failed）
result	Varchar(20)	非空，默认’unknown’	检测结果（fake，real，unknown）
confidence_-score	Float	非空，默认0.0	置信度分数（0-1）
analysis_result	JSONField	可空	详细分析结果（JSON格式）
created_at	Datetime	非空，自动创建	记录创建时间
updated_at	Datetime	非空，自动更新	记录更新时间
completed_at	Datetime	可空	检测完成时间
error_message	TextField	可空	错误信息

3. 系统设置表（**settings_systemsettings**）（基于settings.models.SystemSettings）

表 3-3 系统设置表（settings_systemsettings）结构

字段名	数据类型	约束	描述
id	BigAutoField	主键，自增	设置项唯一标识
key	Varchar(50)	非空，唯一	设置键名
value	TextField	非空	设置值
value_type	Varchar(20)	非空，默认’string’	值类型（string，integer，float，boolean，json）
description	Varchar(255)	可空	设置项描述
created_at	Datetime	非空，自动创建	创建时间
updated_at	Datetime	非空，自动更新	更新时间

这些表结构是根据Django模型自动生成的，字段类型可能因具体的数据库后端（如MySQL）而略有差异，但核心结构和约束保持一致。

3.4 接口设计

系统前后端交互依赖于一组定义良好的RESTful API接口。以下是主要模块的核心接口设计概览。所有接口均以/api/为前缀。

1. 用户认证接口（**/api/users/**）

- POST /users/token/：用户登录，获取JWT。
  - 请求体：{"username": "...", "password": "..."}
  - 响应体：{"access": "...", "refresh": "..."}
- POST /users/token/refresh/：刷新Access Token。
  - 请求体：{"refresh": "..."}
  - 响应体：{"access": "..."}

- POST /users/: 用户注册。
  - 请求体：{"username": "...", "email": "...", "password": "...", "password2": "...", "first_name": "...", "last_name": "..."}
  - 响应体：成功信息及用户信息。
- GET /users/me/: 获取当前登录用户信息（需认证）。
- PATCH /users/{id}/: 修改指定用户信息（需认证，管理员或用户本人）。
- POST /users/{id}/change_password/: 修改用户密码（需认证，用户本人）。

2. 新闻检测接口（**/api/detection/detections/**）

- POST /: 创建新的检测任务（需认证）。
  - 请求体（multipart/form-data）：title（字符串），content（字符串），image（文件，可选）。
  - 响应体：创建的检测记录对象。
- GET /: 获取检测记录列表（管理员获取所有，普通用户获取自己的，需认证）。
  - 支持分页参数：page，page_size。
  - 支持筛选参数：title，status，result，user_id（管理员），start_date，end_date。
- GET /{id}/: 获取指定ID的检测记录详情（需认证，管理员或记录所有者）。
- GET /{id}/result/: 获取指定ID检测任务的简化结果。
- DELETE /{id}/: 删除指定ID的检测记录（需认证，管理员或记录所有者）。
- GET /get_stats/: 获取检测统计信息。

- 支持参数：all=true（管理员获取全局统计）。

- GET /my_detections/：获取当前用户的所有检测记录。

3. 系统设置接口（**/api/settings/**）（仅管理员访问，需认证）

- GET /model_weights/：获取模型权重配置。
- POST /model_weights/：更新模型权重配置。
  - 请求体：{ "local_model_weight": ..., "llm_weight": ..., "fake_threshold": ..., "real_threshold": ... }
- GET /api_config/：获取API相关配置（如OpenRouter密钥，是否使用GPU）。
- POST /api_config/：更新API相关配置。
  - 请求体：{"openrouter_api_key": "...", "use_gpu": true/false}
- GET /logs/：获取系统日志。

4. 管理员用户管理接口（**/api/users/**，部分功能仅管理员）

- GET /：获取用户列表（管理员，支持筛选和分页）。
- GET /{id}/：获取指定用户信息（管理员）。
- PATCH /{id}/：修改用户信息，包括激活/禁用状态，设置管理员权限（管理员）。
- DELETE /{id}/：删除用户（管理员，需谨慎）。
- GET /stats/：获取用户相关的统计信息（如总用户数、活跃用户数）。

所有需要认证的接口，在请求时都需要在HTTP头部添加 Authorization: Bearer <access_token>。

3.5 本章小结

本章对基于多模态深度学习的社交媒体虚假新闻检测系统进行了全面的需求分析与系统设计。首先，通过对普通用户和管理员两种角色的分析，详细梳理了系统的各项功能需求，涵盖了从用户注册登录、新闻提交检测、结果展示与管理，到管理员的用户管理、参数配置、统计分析和日志监控等多个方面。同时，也明确了系统在性能、易用性、可靠性、安全性、可维护性和可扩展性等方面的非功能性需求。

基于清晰的需求，本章提出了系统采用前后端分离的总体架构，并对前端层、API接口层、后端应用层、模型服务层和数据存储层进行了划分。进一步地，将系统细化为用户认证、新闻提交与处理、多模态检测核心、检测结果管理、前端交互以及系统管理等主要功能模块，并阐述了各模块的核心职责。在技术选型方面，本章说明了选择Python、Django、Vue.js、ElementUI、PyTorch、MySQL、JWT等主流技术栈的理由，旨在构建一个高效、稳定且易于开发维护的系统。

随后，本章重点进行了数据库设计，包括绘制E-R图来表达主要实体（用户、检测记录、系统设置）及其关系，将概念模型转化为逻辑关系模式，并根据项目中的Django模型详细列出了用户表、检测记录表和系统设置表的核心字段、数据类型和约束。最后，对系统主要的RESTful API接口进行了规划，包括用户认证、新闻检测、系统设置以及管理员用户管理等模块的接口路径、请求方法、请求参数和预期响应，为后续的系统实现提供了清晰的指导蓝图。通过本章的需求分析和系统设计，为后续章节的模型构建和系统具体编码实现奠定了坚实的基础。

第4章　多模态虚假新闻检测模型构建

虚假新闻的有效识别高度依赖于对信息内容的深度理解和多维度特征的综合分析。本章将详细阐述本系统中用于自动判定新闻真实性的多模态深度学习模型的构建全过程。首先，将介绍所选用数据集的来源、特点以及为模型训练所进行的数据清洗、规范化和预处理流程，包括文本数据的处理和图像数据的转换。其次，将重点设计并阐述一个能够有效融合文本和图像两种主要模态信息的深度学习模型架构，详细说明文本特征提取模块（基于BERT）、图像特征提取模块（基于ResNet）以及后续的特征融合与分类模块的具体构成。接着，将描述模型的训练细节，包括实验环境配置、损失函数的选择、优化器的设定、学习率策略以及训练过程中的关键超参数。最后，将通过在一系列标准评估指标上的实验结果，对所构建模型的性能进行客观评估与分析，并讨论模型的优缺点。

4.1　数据集选择与预处理

高质量的数据集是训练高性能深度学习模型的基石。本研究旨在构建一个能够处理社交媒体上常见图文新闻的检测模型，因此在数据集选择上，我们优先考虑了包含文本和对应图像，并且有明确真伪标签的多模态公开数据集。

4.1.1　数据集来源与构成

本系统的数据集主要整合自以下几个在虚假新闻和多模态分析领域被广泛使用或具有代表性的公开数据集及相关研究成果，旨在构建一个能够反映真实社交媒体环境下虚假新闻特点的综合性多模态语料库：

- MCFEND (Multi-source Chinese Fake News Detection Dataset):[45] 这是一个针对中文领域的多源虚假新闻基准数据集。MCFEND包含了从社交平台、即时通讯应用以及传统在线新闻门户等多种渠道收集的23,974条真实世界中的中文新闻，涵盖了多模态内容（如文本、图像）和社会上下文信息。该数据集中的

新闻均经过了全球14家权威事实核查机构的验证，具有较高的标注质量。其多源特性有助于提升模型在真实场景下的泛化能力。

- 社交媒体谣言数据集 (基于Twitter15 & Twitter16的衍生处理):[44] 本研究参考了包含Twitter15和Twitter16等经典社交媒体谣言数据集的预处理方法和结构。这些数据集最初收集了Twitter平台上与突发事件相关的谣言和非谣言推文，包含了推文文本、用户ID以及传播结构等信息。尽管原始数据主要为文本，但部分推文可能包含图像链接。本研究主要关注其文本内容和真伪标签，并借鉴了相关研究中对这类数据进行图结构嵌入和谣言检测的方法论[44]。对于能够获取到的图像信息，也会纳入多模态分析。

- SocialNet数据集 (Weibo与Tiktok数据):[43] SocialNet项目提供了一个大规模的社交媒体虚假信息数据集，其中包含了来自微博（Weibo V1 和 V2版本）和TikTok等平台的样本。本研究主要利用其发布的Weibo数据集部分，该部分包含了数千至上万条新闻条目，每条包含新闻帖子、评论集合、图像、视频或语音信息，并附带真伪标签。这些数据以JSON格式组织，便于提取新闻文本、图像URL以及相关的元数据。相关研究[43]也探讨了基于此类数据集的早期谣言检测模型。

通过整合来自上述来源的数据，经过筛选、清洗和必要的预处理（如文本去噪、图像下载与规范化），我们构建了用于本系统模型训练与评估的多模态数据集。在数据整合过程中，进行了去重处理，并统一了标签体系（例如，将所有表示虚假的标签映射为1，表示真实的标签映射为0），以确保数据的一致性和高质量。

4.1.2 数据清洗与规范化

原始数据集往往包含噪声、缺失值以及格式不一致等问题，直接用于模型训练会导致性能下降。因此，在模型训练前，必须进行细致的数据清洗与规范化。

1. 文本数据清洗与规范化： 如前文scripts/preprocess_data.py脚本中的clean_text函数所定义，针对新闻标题和正文文本，主要执行以下清洗操作：

- 去除URL链接： 移除文本中出现的HTTP/HTTPS网址。
- 去除社交媒体特定标记： 如Twitter的“@”提及（mentions）和“#”话题标签（hashtags），微博的“[超话]”等。
- 去除特殊符号与多余空格： 清理掉文本中可能存在的非标准字符、HTML标签（如果存在）、以及连续的多个空格，并去除首尾空格。
- 文本长度控制： （在后续Tokenizer处理中实现）过长或过短的文本可能不适合模型处理，会进行截断或填充。

2. 图像数据下载与处理： 社交媒体数据集中的图像信息通常以URL链接的形式提供。scripts/preprocess_data.py脚本中的process_image函数负责处理图像数据：

- 图像下载： 从给定的URL下载图像。针对特定域名（如sinaimg.cn）可能存在的防盗链问题，脚本中考虑了使用代理（如通过百度图片搜索的下载接口）进行尝试。下载过程中设置了超时机制和错误处理，以应对无效链接或下载失败的情况。
- 格式统一与转换： 将下载的图像统一转换为RGB三通道格式，以确保与预训练图像模型的输入要求一致。
- 尺寸调整： 将所有图像统一缩放到模型输入所需的尺寸（本系统设定为 224×224 像素），通常采用双立方插值或Lanczos插值以保持图像质量。
- 本地存储与路径记录： 处理后的图像将被保存到本地指定的目录（如data/processed/images/），并在数据清单中记录其相对路径，供后续模型加载时使用。文件名通常基于原始新闻ID生成，以确保唯一性。

对于无法成功下载或处理的图像，对应的样本在训练时将被视为仅包含文本模态，或者根据策略被舍弃。

4.1.3 文本预处理（Tokenization）

在将文本输入到BERT等Transformer模型之前，需要进行分词（Tokenization）和编码操作。本系统采用与预训练BERT模型（bert-base-chinese）相配套的Tokenizer。具体步骤如get_tokenizer函数和MultimodalFakeNewsDataset类所示：

1. 加载Tokenizer： 从Hugging Face Transformers库加载bert-base-chinese的预训练Tokenizer。
2. 文本分词： Tokenizer会将输入的中文文本分割成字或者预定义的词片段（subwords）。
3. 添加特殊标记： 在分词序列的开头添加[CLS]标记（用于整个序列的表示），在末尾添加[SEP]标记（用于分隔句子或表示序列结束）。
4. 转换为ID序列： 将每个token映射到其在BERT词汇表中的唯一ID。
5. 填充与截断： 为了使同一批次（batch）中的所有文本序列长度一致，将短于预设最大长度（MAX_TEXT_LEN，本系统设为128）的序列用特定的填充标记（[PAD]）的ID进行填充；将长于最大长度的序列进行截断。
6. 生成注意力掩码（Attention Mask）： 创建一个与ID序列等长的二进制掩码，其中真实token对应的位置为1，填充token对应的位置为0。这使得模型在进行自注意力计算时能够忽略填充部分。

经过上述处理，每个文本样本最终被转换为固定长度的input_ids张量和attention_mask张量。

4.1.4 图像预处理（Transforms）

对于输入到ResNet50等图像模型的图片，也需要进行一系列的预处理变换。这些变换通常由预训练模型的官方提供或推荐，以确保输入数据分布与模型预训练时

的数据分布尽可能一致。scripts/data_loader.py中的get_image_transforms函数和detection/ml/data_utils.py中的相应函数定义了这些变换，主要包括：

1. 尺寸调整（Resize）： 将图像调整到模型期望的输入尺寸（本系统为 224×224）。对于ResNet50，通常会先将图像较短边缩放到略大于输入尺寸（如256），然后再进行中心裁剪。

2. 中心裁剪（Center Crop）： 从调整尺寸后的图像中心裁剪出 224×224 的区域。在训练时，有时会使用随机裁剪（RandomResizedCrop）以增加数据多样性。

3. 转换为张量（ToTensor）： 将PIL Image对象或NumPy数组转换为PyTorch张量，并将像素值从[0, 255]范围归一化到[0.0, 1.0]范围。同时，维度顺序会从(H, W, C)调整为(C, H, W)。

4. 标准化（Normalize）： 使用ImageNet数据集的均值（Mean）和标准差（Standard Deviation）对图像张量进行标准化处理，即对每个通道执行 `(input - mean) / std`。这有助于模型更快收敛并提高性能。本系统使用的标准值为 `mean=[0.485, 0.456, 0.406]` 和 `std=[0.229, 0.224, 0.225]`。

5. 数据增强（Data Augmentation，仅训练时）： 在训练阶段，可以引入随机水平翻转（RandomHorizontalFlip）、随机旋转、颜色抖动等数据增强技术，以增加训练样本的多样性，提高模型的泛化能力。评估和测试阶段不使用数据增强。

经过这些变换，每个图像样本被转换为一个符合模型输入要求的归一化张量。如果某个新闻样本没有关联图像，则会使用一个全零的图像张量作为占位符，并通过一个image_available标志来告知模型该图像是否有效。

4.1.5 数据集划分

为了客观评估模型的性能并进行有效的模型选择，需要将整合后的数据集划分为训练集（Training Set）、验证集（Validation Set）和测试集（Test Set）。

划分比例参照scripts/train_model.py中的设定：

- 测试集（Test Set）： 占比15%。这部分数据完全不参与模型的训练和验证过程，仅用于在模型训练完成后进行最终的性能评估。
- 剩余数据（Train-Validation Set）： 占比85%。
- 验证集（Validation Set）： 从上述85%的数据中，再划分出约17.6%（即总数据的 $0.15/(1-0.15) \approx 0.176$）作为验证集。因此，验证集约占总数据的15%。
- 训练集（Training Set）： 剩余约70%的数据作为训练集，用于模型的参数学习。

在进行数据划分时，采用分层抽样（Stratified Sampling）策略，确保训练集、验证集和测试集中虚假新闻和真实新闻的比例与原始数据集中的比例大致相当，以避免因类别不平衡导致的评估偏差。随机种子（RANDOM_SEED，设为42）的使用确保了数据划分的可复现性。

4.2 多模态融合模型设计

本系统核心的虚假新闻检测模型是一个多模态深度学习网络，旨在有效融合文本和图像信息进行综合判断。模型的整体结构如图4-1所示。该模型主要由文本特征提取模块、图像特征提取模块、特征融合模块和最终的分类器组成。

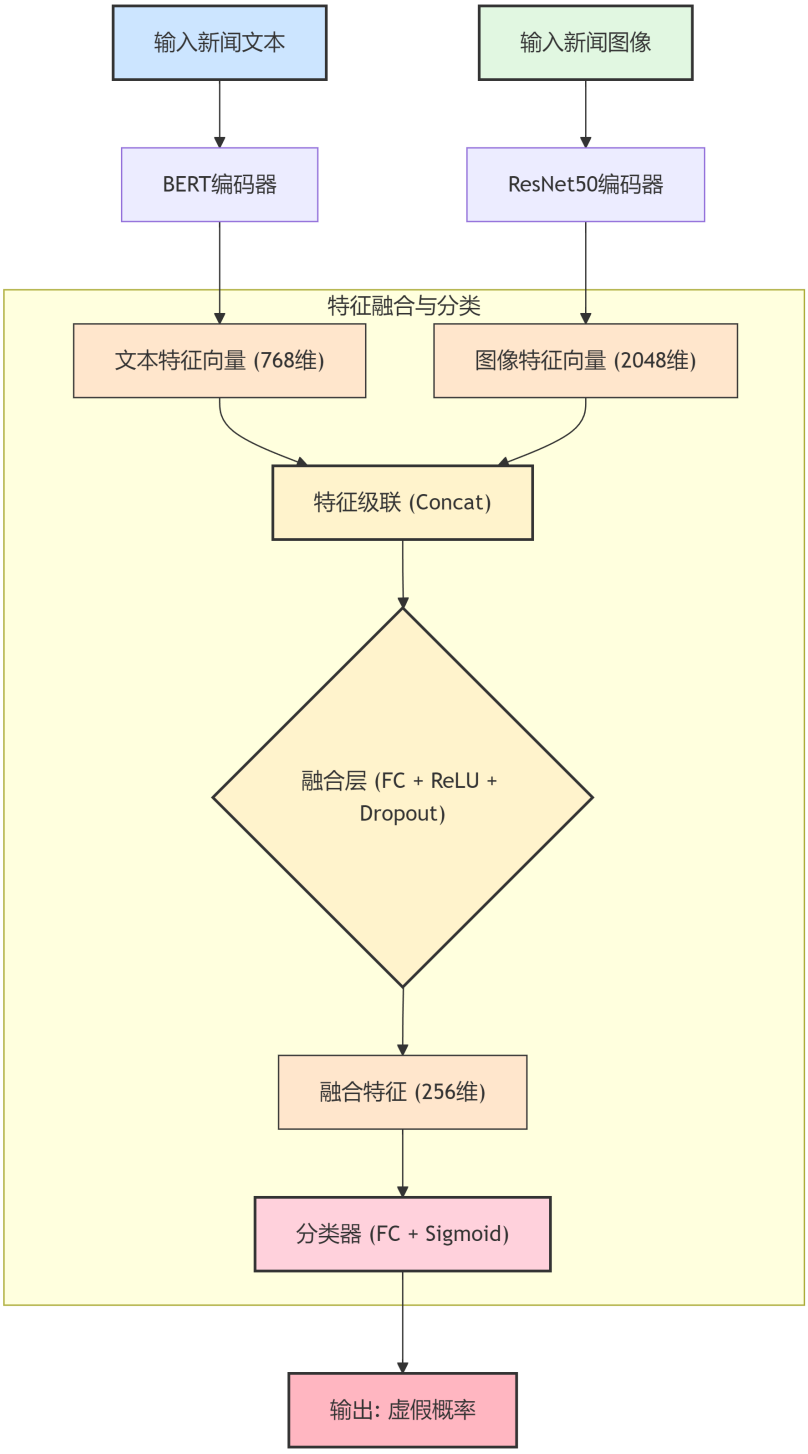


图 4-1 多模态虚假新闻检测模型架构图

4.2.1 文本特征提取模块

新闻的文本内容（包括标题和正文）是判断其真实性的核心依据之一。为了充分捕捉文本的深层语义信息和上下文依赖关系，本模块采用基于Transformer架构的预训练语言模型BERT进行特征提取。

- 模型选择： 选用bert-base-chinese模型。该模型是一个针对大规模中文语料进行预训练的BERT基础版本，包含12个Transformer编码器层，隐藏层维度为768，并设有12个注意力头。它能够较好地处理中文文本的语义理解任务。
- 输入处理： 如4.1.3节所述，输入的文本首先经过分词、添加特殊标记（[CLS], [SEP]）、转换为ID序列、填充或截断至最大长度（128个token）等预处理步骤。
- 特征输出： 经过BERT模型处理后，取其最后一层Transformer编码器输出中对应于[CLS]特殊标记的隐藏状态向量作为整个输入文本的聚合语义表示。该向量的维度为768（TEXT_EMBEDDING_DIM）。
- 微调策略： 在模型训练过程中，BERT编码器的参数可以选择性地进行微调（fine-tuning），使其更适应虚假新闻检测这一特定任务的特征。本研究中，BERT的参数将参与反向传播和更新。

通过BERT模型，能够将可变长度的新闻文本有效地转换为固定维度的、富含语义信息的特征向量。

4.2.2 图像特征提取模块

新闻配图往往包含重要的视觉线索，能够印证或反驳文本内容的真实性。本模块采用预训练的ResNet50卷积神经网络来提取图像的视觉特征。

- 模型选择： 选用在ImageNet数据集上预训练的ResNet50模型。ResNet50以其深层残差结构有效缓解了深度网络训练的困难，并在图像分类等任务上表现出色。

- 输入处理： 如4.1.4节所述，输入的图像经过尺寸调整（224×224）、转换为张量、以及使用ImageNet的均值和标准差进行标准化等预处理。
- 特征输出： 原始的ResNet50模型末端是一个用于1000类图像分类的全连接层。在本系统中，我们移除这个全连接层，保留其之前的卷积和池化层作为特征提取器。通常取全局平均池化层（Global Average Pooling）的输出作为图像的全局特征表示。对于ResNet50，该特征向量的维度是2048（`IMG_EMBEDDING_DIM`）。
- 微调策略： 类似于文本编码器，ResNet50的参数在训练过程中也可以进行微调，以使其更好地提取与虚假新闻识别相关的视觉特征。
- 处理缺失图像： 如果新闻样本没有配图，或者图像无法加载，系统会使用一个全零的张量作为图像输入，并通过一个`image_available`布尔标志告知后续模块该图像特征是否有效。在特征融合时，无效的图像特征将被忽略或以零向量参与计算。

通过ResNet50，能够从新闻配图中提取出高层次的、具有判别力的视觉特征向量。

4.2.3 特征融合与分类模块

在分别获得文本特征向量（维度768）和图像特征向量（维度2048）之后，需要一个有效的融合模块来整合这两个模态的信息，并最终进行真伪分类。本系统的最终模型（如`scripts/train_model.py`中不含元数据的版本`MultimodalFakeNewsModel`，以及后端部署的`detection/ml/model.py`）采用了以下融合与分类策略：

1. 特征级联（Feature Concatenation）： 最直接的融合方式是将文本特征向量和图像特征向量在维度上进行拼接。

$$\mathrm{f}_{\mathrm{fused}} = \mathrm{concat}(\mathrm{f}_{\mathrm{text}}, \mathrm{f}_{\mathrm{image}}) \tag{1}$$

其中，$\mathrm{f}_{\mathrm{text}} \in \mathbb{R}^{768}$ 是文本特征，$\mathrm{f}_{\mathrm{image}} \in \mathbb{R}^{2048}$ 是图像特征。拼接后的融合特征向量 $\mathrm{f}_{\mathrm{fused}}$ 的维度将是 $768 + 2048 = 2816$。

2. 融合层（Fusion Layer）： 拼接后的融合特征向量维度较高，且可能存在冗余信息。为了学习更紧凑和更具判别力的多模态表示，并实现不同模态特征间的非线性交互，我们将 f_{fused} 输入到一个或多个全连接层组成的融合网络中。在本系统中，该融合层定义如下（参照FUSION_OUTPUT_DIM = 256）：

- 一个线性层（全连接层），将2816维的输入映射到256维的输出。
- 一个ReLU激活函数，引入非线性。
- 一个Dropout层（例如，p=0.5），以防止过拟合。

该融合层的输出是一个256维的多模态融合特征向量。

3. 分类器（Classifier）： 最后，将融合层输出的256维特征向量送入一个单输出的线性层（全连接层），该层的输出是一个标量值（logit）。这个logit值随后会经过Sigmoid激活函数（在计算损失时隐式包含，或者在预测概率时显式应用），将其转换为一个0到1之间的概率值，表示新闻为虚假的概率。

$$P(\text{fake}|\text{text},\text{image}) = \sigma(W_c \cdot f_{\text{fusion\_out}} + b_c) \tag{2}$$

其中 σ 是Sigmoid函数，W_c 和 b_c 是分类器层的权重和偏置，$f_{\text{fusion\_out}}$ 是融合层的输出。

模型的整体结构简洁而有效，通过预训练模型提取高质量的单模态特征，再通过特征级联和浅层神经网络进行融合与分类。

4.3 模型训练与优化

模型的训练是一个通过在标注数据上迭代学习来调整模型参数以最小化预测误差的过程。

4.3.1 实验环境

- 硬件环境：

- CPU: Intel Core i5-12400F
- GPU: NVIDIA GeForce RTX 2080Ti 11GB
- 内存 (RAM): 32GB DDR4

• 软件环境：

- 操作系统: Windows 11
- Python版本: 3.12.9
- PyTorch版本: 2.6.0(CUDA版本 12.6)
- Transformers库版本: 4.39.0
- Torchvision库版本: 0.21.0
- Scikit-learn库版本: 1.6.1
- Pandas库版本: 2.2.1

4.3.2 损失函数

由于虚假新闻检测本质上是一个二分类问题（真实 vs. 虚假），本系统采用二元交叉熵损失函数（Binary Cross-Entropy Loss）。由于模型的输出层没有显式的Sigmoid激活函数，而是直接输出logit值，因此选用`torch.nn.BCEWithLogitsLoss`。这个损失函数内部会自动应用Sigmoid函数并将logit转换为概率，然后计算二元交叉熵损失。它在数值上比先用Sigmoid再用BCELoss更稳定。 对于一个样本，损失函数定义为：

$$L = -[y \cdot \log(\sigma(o)) + (1 - y) \cdot \log(1 - \sigma(o))] \tag{3}$$

其中，o 是模型输出的logit值，y 是真实的标签（0表示真实，1表示虚假），$\sigma(\cdot)$ 是Sigmoid函数。

4.3.3 优化器

优化器的选择对于模型的训练速度和最终性能至关重要。本系统采用AdamW (Adam with Weight Decay Fixation)[46]优化器。AdamW是Adam优化器的一个改进版本，它将权重衰减（L2正则化）与梯度更新解耦，通常在基于Transformer的模型训练中表现更好，有助于防止过拟合。 如scripts/train_model.py中所示，针对模型的不同部分设置了不同的学习率：

- 编码器部分（BERT和ResNet）： 设置了较小的学习率（LEARNING_RATE_ENCODERS，例如 1×10^{-5}），因为这些预训练模型的参数已经在一个通用的大规模数据集上得到了很好的初始化，我们希望在微调过程中以较小的幅度调整它们，以保留预训练学到的知识。
- 融合层与分类器头部： 设置了相对较大的学习率（LEARNING_RATE_HEAD，例如 1×10^{-4}），因为这些部分的参数是随机初始化的，需要更快的学习速度来适应当前任务。

权重衰减系数（WEIGHT_DECAY）设为0.01，用于对模型参数进行L2正则化，防止模型过于复杂。

4.3.4 学习率与调度策略

为了在训练过程中动态调整学习率，以帮助模型更好地收敛并跳出局部最优，本系统采用了ReduceLROnPlateau学习率调度策略。

- 工作机制： 该策略会监测一个指定的性能指标（在本系统中是验证集上的F1分数）。如果在一定数量的轮次（patience，设为1）内，该性能指标没有得到改善（mode='max'表示希望指标越大越好），则学习率会乘以一个缩减因子（factor，设为0.1）。
- 参数设置：

- mode='max'：监控的指标越大越好。
- factor=0.1：当指标不再提升时，学习率乘以0.1。
- patience=1：如果连续1个epoch验证集F1分数没有提升，则降低学习率。
- verbose=True：当学习率更新时，在控制台打印信息。

这种策略有助于在训练初期使用较大的学习率快速下降损失，在后期使用较小的学习率进行精细调整。

4.3.5 训练超参数设置

除了学习率相关的参数外，其他关键的训练超参数设置如下：

- Epochs（训练轮次）：EPOCHS = 20。一个epoch表示模型完整地遍历一次所有训练数据。
- Batch Size（批处理大小）：BATCH_SIZE = 32。表示每次参数更新时使用的训练样本数量。Batch size的选择需要在梯度估计的准确性和计算资源之间进行权衡。
- Random Seed（随机种子）：RANDOM_SEED = 42。用于初始化所有随机过程（如数据划分、模型参数初始化、数据增强中的随机操作等），以确保实验结果的可复现性。

4.3.6 训练过程与早停策略

模型的训练过程在一个循环中进行，每个循环（epoch）包含以下步骤：

1. 训练阶段（Training Phase）：模型在训练集上进行前向传播计算输出，然后根据损失函数计算损失，再进行反向传播计算梯度，最后优化器根据梯度更新模型参数。记录训练损失和F1分数等指标。

2. 验证阶段（Validation Phase）： 在每个epoch结束后，模型在独立的验证集上进行评估（不进行参数更新）。计算验证集上的损失、准确率、精确率、召回率、F1分数和AUC等指标。验证集的性能用于监控模型的泛化能力和进行模型选择。

3. 学习率调度： 根据验证集上的F1分数，学习率调度器（ReduceLROnPlateau）决定是否调整学习率。

4. 模型保存： 如果当前epoch在验证集上取得的F1分数优于之前所有epoch的最佳F1分数，则将当前模型的参数状态保存到文件（`best_text_image_model.pth`）。

5. 早停策略（Early Stopping）： 为了防止模型在训练集上过拟合，并节省不必要的训练时间，引入了早停机制。如果验证集上的F1分数连续多个epoch（`EARLY_STOPPING_PATIENCE`，设为3）没有得到改善，则提前终止训练过程。

通过这种方式，我们期望能够找到在验证集上表现最优的模型，并避免过度训练。

4.4 模型评估

在模型训练完成后，需要使用独立的测试集对最终选定的最佳模型（即在验证集上F1分数最高的模型）进行全面的性能评估。评估的目的是检验模型在未见过数据上的泛化能力。

4.4.1 评估指标

本研究采用以下一组标准的二分类任务评估指标来衡量模型的性能：

- 准确率（Accuracy）： 模型正确分类的样本数占总样本数的比例。

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{4}$$

- 精确率（Precision）： 模型预测为正例（虚假新闻）的样本中，真正为正例的比例。

$$\text{Precision} = \frac{TP}{TP + FP} \tag{5}$$

- 召回率（Recall / Sensitivity）： 真实为正例的样本中，被模型成功预测为正例的比例。

$$\text{Recall} = \frac{TP}{TP + FN} \tag{6}$$

- F1分数（F1-Score）： 精确率和召回率的调和平均数，是一个综合评价指标。

$$\text{F1-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{7}$$

- AUC（Area Under the ROC Curve）： ROC曲线下面积。ROC曲线是以假正例率（FPR）为横轴，真正例率（TPR，即召回率）为纵轴绘制的曲线。AUC值越接近1，表示模型区分正负样本的能力越强。

- 混淆矩阵（Confusion Matrix）： 一个表格，用于可视化模型预测结果与真实标签之间的对应关系，包含TP（真正例）、TN（真负例）、FP（假正例）、FN（假负例）四个值。

其中，TP（True Positive）是指真实为虚假新闻且被模型正确预测为虚假新闻的样本数；TN（True Negative）是指真实为真实新闻且被模型正确预测为真实新闻的样本数；FP（False Positive）是指真实为真实新闻但被模型错误预测为虚假新闻的样本数（误报）；FN（False Negative）是指真实为虚假新闻但被模型错误预测为真实新闻的样本数（漏报）。

4.4.2 实验结果与分析

在本节中，我们将展示多模态虚假新闻检测模型在预留的独立测试集上的最终评估结果。该测试集共包含4792条样本，其中真实新闻（标签为Real/0）1968条，虚假新闻（标签为Fake/1）2824条，这一划分严格遵循了4.1.5节所述的数据集划分

策略，确保了测试数据未参与任何训练或验证过程。模型在训练过程中，（此处你需要根据实际训练日志填写：例如“在第15个epoch”）达到了验证集上的最佳F1分数（例如“92.5%”），该状态下的模型参数被保存并选为最终用于测试的模型。

将最终选定的最佳模型应用于测试集，得到的各项性能指标如表4-1所示。分类报告的详细信息也一并列出，其中“Real”对应标签0，“Fake”对应标签1。

表 4-1 多模态模型在测试集上的性能评估

评估指标	数值
整体准确率 (Accuracy)	91.82%
AUC (Area Under ROC Curve)	0.9677
针对虚假新闻 (Fake News) 的指标:	
精确率 (Precision)	91.42%
召回率 (Recall)	95.04%
F1分数 (F1-Score)	93.19%
针对真实新闻 (Real News) 的指标:	
精确率 (Precision)	92.32%
召回率 (Recall)	87.19%
F1分数 (F1-Score)	89.68%
宏平均 (Macro Avg) F1分数	91.44%
加权平均 (Weighted Avg) F1分数	91.75%

测试集上的混淆矩阵如图4-2所示。

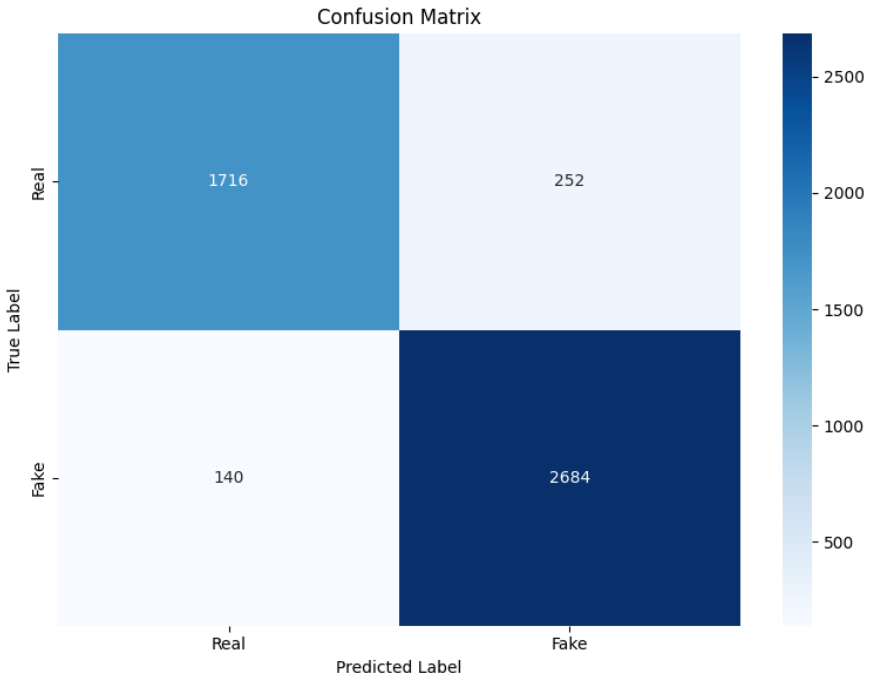


图 4-2 测试集混淆矩阵

结果分析：

从表4-1和图4-2的评估结果来看，本研究提出的多模态虚假新闻检测模型在独立的测试集上展现了良好且均衡的性能。

- 整体性能： 模型的整体准确率达到了91.82%，这意味着在所有测试样本中，超过九成的样本被正确分类。AUC值为0.9677，非常接近于1，这表明模型在区分真实新闻和虚假新闻方面具有很强的判别能力，即使在不同的分类阈值下也能保持较好的性能。

- 虚假新闻检测性能： 针对核心任务——识别虚假新闻（正类），模型的表现尤为突出。其F1分数为93.19%，这是一个综合了精确率和召回率的优秀指标。具体来看，精确率为91.42%，意味着在所有被模型判定为虚假的新闻中，约91.42%确实是虚假新闻，显示了模型预测结果的可靠性，减少了将真实新闻误判为虚假的概率。召回率高达95.04%，这表明模型成功识别出了测试集中绝大

多数（约95.04%）的虚假新闻，具有较低的漏报率，这对于阻止虚假信息传播至关重要。

- 真实新闻检测性能： 对于真实新闻（负类）的识别，模型的F1分数为89.68%。其精确率为92.32%（即被模型判定为真实的新闻中，92.32%是正确的），召回率为87.19%（即所有真实新闻中，87.19%被正确识别）。虽然真实新闻的召回率略低于虚假新闻，但整体表现仍然稳健。
- 混淆矩阵分析： 混淆矩阵（图4-2）提供了更细致的分类情况。
  - TP (True Positives) = 2684： 模型正确地将2684条虚假新闻识别为虚假。
  - TN (True Negatives) = 1716： 模型正确地将1716条真实新闻识别为真实。
  - FP (False Positives) = 252： 模型错误地将252条真实新闻判断为虚假（误报，Type I Error）。
  - FN (False Negatives) = 140： 模型错误地将140条虚假新闻判断为真实（漏报，Type II Error）。

  从混淆矩阵可以看出，模型在识别虚假新闻方面的漏报数量（FN=140）相对较少，这与其高达95.04%的召回率相符。将真实新闻误判为虚假（FP=252）的情况略多于将虚假新闻漏判为真实的情况，这可能与模型在某些特征上对“可疑”信号更为敏感，或者在某些情况下为了捕捉更多虚假新闻而牺牲了一部分对真实新闻的精确度。

尽管本多模态模型取得了令人鼓舞的成果，但在实际应用中仍可能面临一些挑战。例如，对于那些文本和图像在表面上高度一致，但共同编织了更深层次误导性叙事的“高级”虚假新闻，模型可能仍难以准确识别。此外，经过精心编辑或深度伪造（Deepfake）的图像，如果其视觉特征与真实图像非常相似，也可能对模型的图像分析模块造成困扰。未来工作可以考虑引入更复杂的跨模态交互机制（如基于

Transformer的融合器）、更先进的视觉伪造检测技术，或结合外部知识库进行更深层次的事实核查，以应对这些更具挑战性的虚假新闻类型。

第5章　系统实现

在完成了系统需求分析、整体设计以及核心检测模型的构建与评估之后，本章将详细阐述多模态虚假新闻检测系统的具体编码实现过程。系统实现是理论设计与实际应用的桥梁，其质量直接影响到系统的可用性、稳定性和用户体验。本章将遵循第三章的系统设计蓝图，分别从后端服务系统和前端交互界面两个主要方面进行详细描述，力求清晰地展现从代码层面如何将各项功能需求转化为可操作的软件实体。内容将包括后端Django项目的搭建与配置、核心业务逻辑的实现、数据库模型的交互，以及前端Vue 3应用的组件化构建、用户交互流程的实现和与后端API的对接。

5.1　后端系统实现（基于Django框架）

系统后端采用Python语言和Django Web框架进行开发，负责处理核心业务逻辑、数据存储、模型调用以及对外提供API接口。

5.1.1　开发环境与项目配置

后端系统的开发与构建环境主要包括：

- 操作系统： Windows 10
- Python版本： 3.12.9
- Django版本： 4.2.9
- Django REST framework版本： 3.14.0
- djangorestframework-simplejwt版本： 5.3.1
- MySQL数据库版本： 8.0.39
- 数据库驱动： mysqlclient

- 深度学习相关库： PyTorch, Transformers, Torchvision
- 其他关键库： python dotenv, Pillow, joblib。

Django项目结构遵循标准的Django项目布局，包含一个主配置应用（例如 config）以及多个功能应用（如 users, detection, settings）。

项目配置（**config/settings.py**）：核心配置项包括：SECRET_KEY, DEBUG, ALLOWED_HOSTS, INSTALLED_APPS, MIDDLEWARE, DATABASES, AUTH_USER_MODEL (users.User), REST_FRAMEWORK, SIMPLE_JWT, CORS配置, 静态与媒体文件路径, LOGGING, 及模型相关路径与API密钥（从.env加载）。

5.1.2 用户模块实现（**users** app）

核心实现位于backend/users/目录下。

1. 用户模型定义（**models.py**）：自定义User模型继承自AbstractUser，添加了email, phone, avatar, bio, is_admin, admin_role, 及检测统计字段。
2. 用户序列化器（**serializers.py**）：定义了UserSerializer, UserRegistrationSerializer, UserDetailSerializer, PasswordChangeSerializer。
3. 用户视图与API接口（**views.py** 和 **urls.py**）：UserViewSet提供用户CRUD。JWT认证路径为/token/等。核心Action包括create, retrieve, update, list, destroy, 及自定义的me, change_password, stats。**backend/update_admin.py**用于创建root管理员。

5.1.3 新闻检测模块实现（**detection** app）

核心实现位于backend/detection/目录下。

1. 检测记录模型（**models.py**）：Detection模型存储任务信息。
2. 检测序列化器（**serializers.py**）：包括DetectionSerializer, DetectionCreateSerializer, DetectionResultSerializer, DetectionStatSerializer。

3. 检测视图与API接口（**views.py** 和 **urls.py**）：DetectionViewSet提供检测记录CRUD。创建时调用FakeNewsDetector。自定义Action包括result，my_detections，get_stats。

4. 检测服务核心逻辑（**services/detector.py**）：FakeNewsDetector类封装检测流程：调用detection/ml/data_utils.py的preprocess_input预处理数据；模型推理；LLM交叉验证(services/llm_verifier.py的verify_news_with_llms_async)；结果融合。

5. 多模态模型与数据工具（**ml/model.py**，**ml/data_utils.py**）：model.py定义MultimodalFakeNewsModel。data_utils.py提供预处理函数。

5.1.4 系统设置模块实现（**settings** app）

位于backend/settings/。

1. 系统设置模型（**models.py**）：SystemSettings模型以键值对存储全局配置。

2. 设置序列化器（**serializers.py**）：包括SystemSettingsSerializer，ModelWeightsSerializer，ApiConfigSerializer。

3. 设置视图与API接口（**views.py** 和 **urls.py**）：SettingsViewSet提供配置接口：模型权重、API配置（尝试更新.env文件）、系统日志查看。管理命令initialize_settings.py初始化默认设置。

5.1.5 数据库交互实现

通过Django ORM与MySQL交互。

5.2 前端系统实现（基于Vue.js框架）

系统的前端用户界面是用户与虚假新闻检测功能直接交互的门户，其设计与实现的优劣直接影响到系统的易用性和用户体验。本系统前端采用成熟且广泛应用的Vue.js（版本2.x)渐进式JavaScript框架进行构建，并结合了ElementUI组件库来快速搭建专业且美观的界面，同时利用Vuex（版本3.x)进行状态管理，Vue Router（版

本3.x)进行路由控制，Axios进行HTTP通信，共同构建了一个用户友好且功能丰富的单页面应用（SPA）。

5.2.1 开发环境与项目配置

前端项目的开发与构建环境主要包括：

- Node.js 与 npm/yarn： JavaScript运行时环境和包管理工具。
- Vue CLI： 版本5.0.8，用于项目初始化、配置和打包。
- 核心依赖库：
  - vue@^{}2.6.14： Vue.js核心库。
  - vue router@^{}3.6.5： Vue Router官方路由管理器。
  - vuex@^{}3.6.2： Vuex官方状态管理库。
  - axios@^{}1.6.2： HTTP客户端。
  - element ui@^{}2.15.14： 基于Vue 2.0的桌面端UI组件库。
  - echarts@^{}5.6.0： 数据可视化库。
  - js cookie@^{}3.0.5： Cookie操作库。
- 项目配置文件：
  - **vue.config.js**： Vue CLI配置文件，配置了开发服务器代理（将/api ↪ 转发至后端http://localhost:8000）和Sass全局变量注入（如@/assets/css ↪ /variables.scss）。
  - **.eslintrc.js**： ESLint配置文件，定义了代码风格和规范，使用@babel/ ↪ eslint parser作为解析器。
  - **babel.config.js**: Babel配置文件，使用@vue/cli plugin babel/preset。

前端应用的入口文件是frontend/src/main.js。该文件负责初始化Vue实例（new Vue(...)），并全局注册和配置Vue Router、Vuex、ElementUI（通过Vue.use(ElementUI, { size: 'medium' })）以及Axios。Axios的defaults.baseURL配置为后端API地址，并设置了请求拦截器（附加JWT Token）和响应拦截器（处理401错误并自动登出）。

5.2.2 项目结构与路由设计

前端项目源代码结构（frontend/src/）清晰，包含assets，api，router，store，views等核心目录，如图5-1所示。

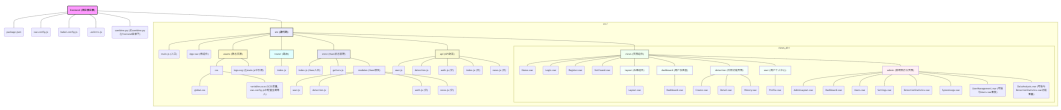


图 5-1 前端项目主要目录结构示意图

路由设计（**frontend/src/router/index.js**）：采用Vue Router 3.x API，通过new VueRouter({...})创建路由实例。路由配置包括公共路由、用户仪表盘路由组（/dashboard）和管理员后台路由组（/admin），所有页面组件均采用路由懒加载。全局前置导航守卫（router.beforeEach）负责页面标题设置和基于Token及Vuex中用户状态的权限控制。

5.2.3 状态管理（Vuex 3.x）

系统使用Vuex 3.x进行状态管理（frontend/src/store/）。

- 用户模块（**user.js**）：管理用户Token（存Cookie）、用户信息（存localStorage）和角色。
- 检测模块（**detection.js**）：管理检测历史、详情和统计数据。
- Getters（**getters.js**）：定义了如token，user，isAdmin等全局getters。

如图5-2所示为Vuex的核心数据流。

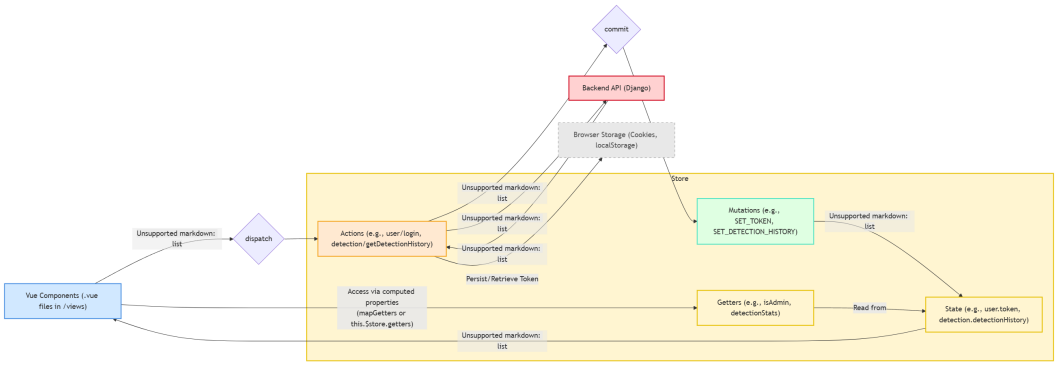


图 5-2 Vuex核心数据流示意图

5.2.4 主要界面组件实现

以下结合中期报告截图，对关键Vue组件的UI构成（使用ElementUI组件）和核心交互逻辑进行阐述。

1. 应用根组件与全局样式（**App.vue**, **assets/css/global.css**）：App.vue包含<router view />并引入全局CSS。global.css定义了基础样式、颜色变量、工具类及滚动条美化。

2. 登录与注册页面（**views/Login.vue**, **views/Register.vue**）：如图5-3所示。使用ElementUI的表单组件，包含输入校验。通过this.$store.dispatch调用Vuex Action处理API请求。 3. 主布局（**views/layout/Layout.vue**, **views/admin/AdminLayout.vue**）：如图5-4和图5-5所示框架。使用ElementUI容器组件搭建，包含可折叠侧边栏（el menu）和头部导航。

4. 用户/管理员仪表盘（**views/dashboard/Dashboard.vue**, **views/admin/Dashboard.vue**）：如图??所示。通过Vuex Action获取数据，使用ElementUI卡片展示指标，并利用Echarts（在mounted钩子中初始化）渲染图表。

5. 新闻检测提交页面（**views/detection/Create.vue**）：如图5-6所示。使用ElementUI表单和上传组件（el upload）。handleImageChange方法校验文件，提交时构造FormData发送请求。

6. 检测详情页面（**views/detection/Detail.vue**）：如图5-7所示。通过路由参数获取ID，调用Vuex Action获取详情。若任务处理中则启动轮询。使用ElementUI组件展示新闻内容、图片、状态、最终结论、置信度及详细分析（包括LLM结果表格）。

7. 其他核心页面：

- 检测历史页面（**views/detection/History.vue**）：如图5-8（左）所示。使用el table和el pagination，支持筛选和删除。
- 个人中心页面（**views/user/Profile.vue**）：如图5-8（右）所示。展示用户信息，提供表单修改资料和密码。
- 管理员相关页面（admin/目录下各组件）：
  - 用户管理（**Users.vue**）：如图5-9（左）所示。用户列表展示、筛选、分页及增删改。

- 检测统计（**DetectionStatistics.vue**）： 如图5-9（右）所示。全局检测统计和Echarts图表。
- 参数设置（**Settings.vue**）： 如图5-10（左）所示。表单修改模型权重、阈值、API密钥等。
- 系统日志（**SystemLogs.vue**）： 如图5-10（右）所示。表格展示后端日志。

图 5-3 登录与注册界面（由Login.vue, Register.vue实现）

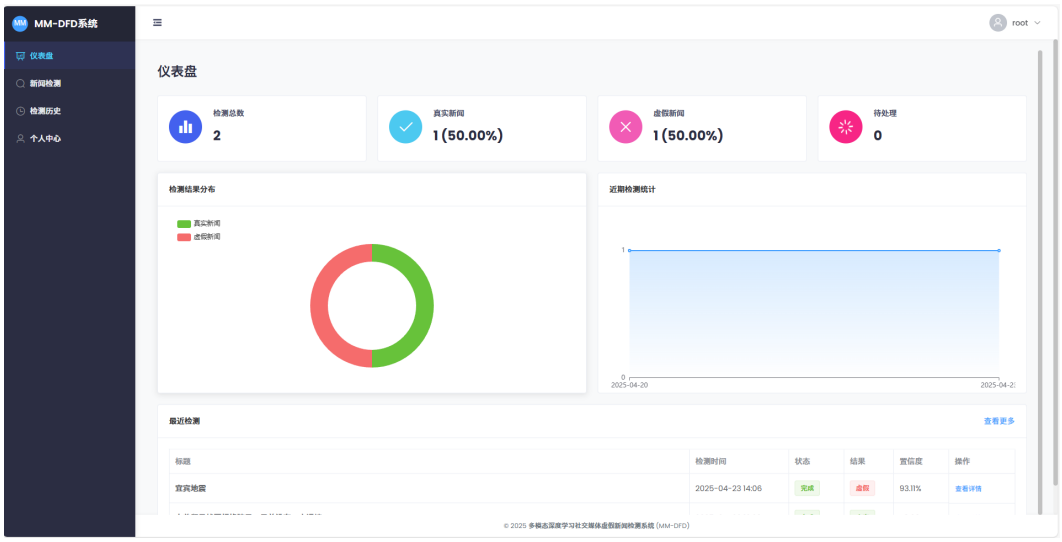


图 5-4 用户主布局界面框架（由layout/Layout.vue实现）

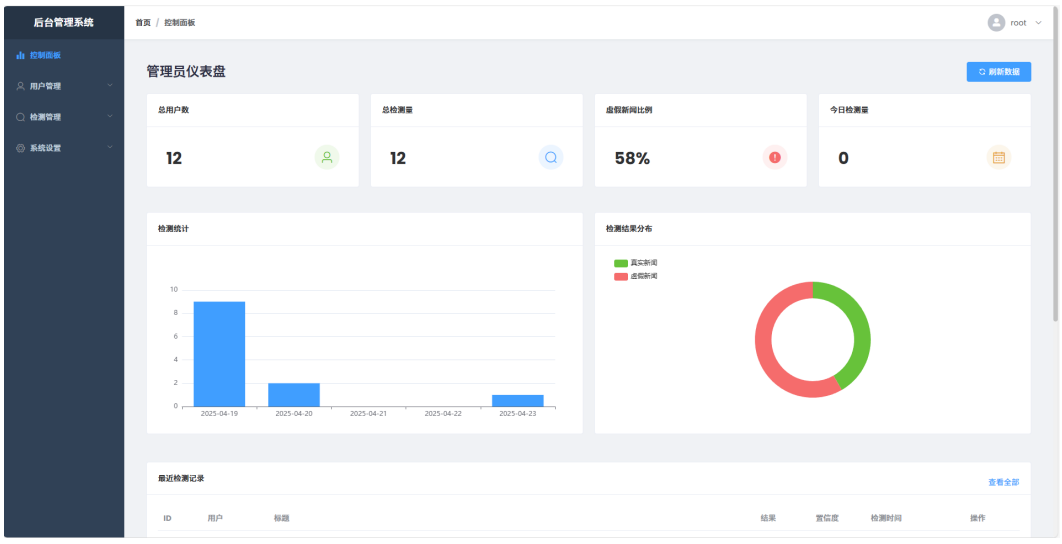


图 5-5 管理员主布局界面框架（由admin/AdminLayout.vue实现）

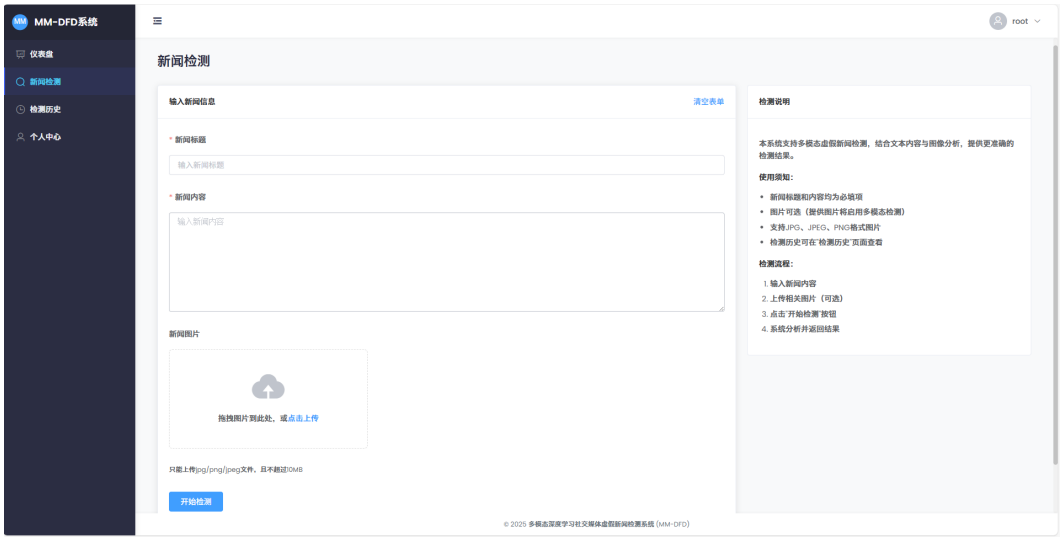
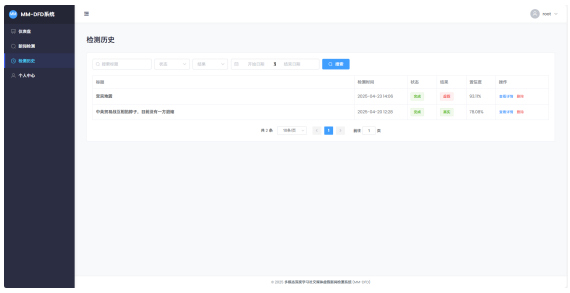
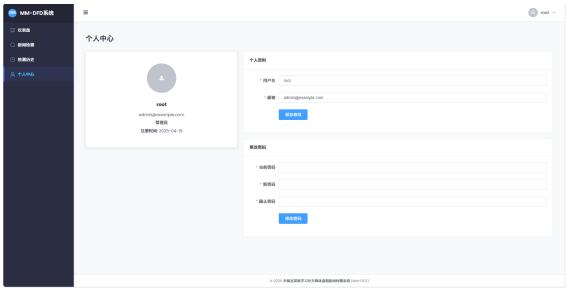


图 5-6 新闻检测提交界面（由detection/Create.vue实现）

图 5-7 检测详情界面（由detection/Detail.vue实现）

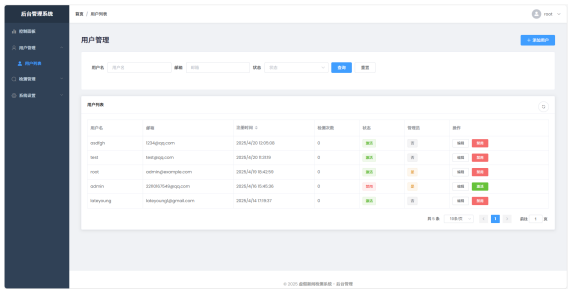


（a）检测历史界面

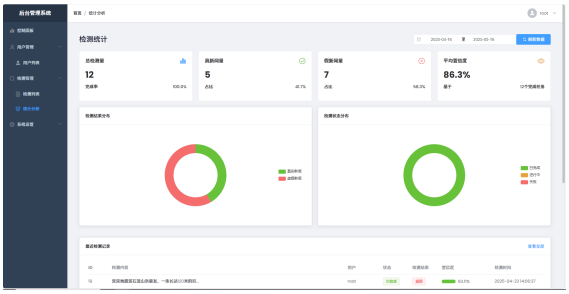


（b）个人中心界面

图 5-8 检测历史与个人中心界面（分别由detection/History.vue和user/Profile.vue实现）

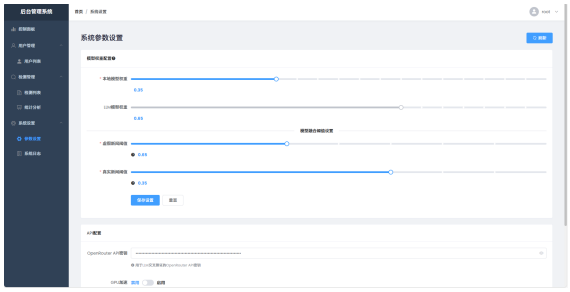


（a）管理员用户管理界面

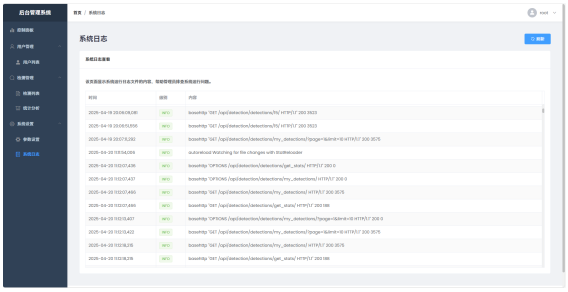


（b）管理员检测统计界面

图 5-9 管理员用户管理与检测统计界面（分别由admin/Users.vue和admin/DetectionStatistics.vue实现）



（a）管理员参数设置界面



（b）管理员系统日志界面

图 5-10 管理员参数设置与系统日志界面（分别由admin/Settings.vue和admin/SystemLogs.vue实现）

5.2.5 API接口调用封装与全局配置

API调用通过src/api/目录下的模块化JS文件进行封装。全局Axios配置在src/main.js中完成。

5.3 系统部署方案

本毕业设计主要采用本地部署方案进行开发、测试和演示。

- 后端服务部署（Django）： 通过Python解释器运行Django开发服务器（python manage.py runserver 0.0.0.0:8000）。
- 前端应用部署（Vue 2）： 通过Node.js环境下的Vue CLI开发服务器运行（npm run serve）。
- 数据库部署（MySQL）： 本地安装并运行MySQL服务。
- 模型与媒体文件： 存放于本地服务器指定路径。

用户通过访问前端开发服务器地址（如http://localhost:8080）与系统交互。

5.4 本章小结

本章详细阐述了基于多模态深度学习的社交媒体虚假新闻检测系统的整体实现过程。后端系统基于Python Django框架，实现了用户管理、新闻检测处理、系统参数配置等核心功能。前端系统则采用Vue.js 2框架和ElementUI组件库，构建了用户友好的交互界面，通过Vuex 3进行状态管理，Axios进行API通信。最后说明了系统在本地环境下的部署方案。本章完整展现了从设计到编码实现一个功能全面的多模态虚假新闻检测Web系统的过程。

第6章　系统测试

系统测试是确保软件质量、验证系统功能是否符合设计要求以及评估系统在不同环境下稳定性和性能的关键阶段。在本章中，我们将对已实现的多模态虚假新闻检测系统进行一系列全面的测试。测试内容将覆盖系统的各项核心功能、关键性能指标以及在不同主流浏览器环境下的兼容性。通过设计合理的测试用例、执行严格的测试流程并分析测试结果，旨在发现系统中潜在的缺陷与不足，为系统的进一步优化和完善提供依据，并最终证明系统能够达到预期的设计目标和用户需求。

6.1　测试环境与工具

为了保证测试结果的客观性和可复现性，本次系统测试在以下统一的环境中进行：

1. 硬件环境：

- 服务器端（本地后端与模型运行）：
  - CPU: Intel Core i5-12400F
  - GPU: NVIDIA GeForce RTX 2080Ti (11GB显存)
  - 内存 (RAM): 32GB DDR4
  - 存储：NVMe SSD 1TB
- 客户端（配置与服务器端一致）

2. 软件环境：

- 服务器端：
  - 操作系统：Windows 10 专业版
  - Python版本：3.12.9
  - Django版本：4.2.9

- Django REST framework版本：3.14.0
- MySQL版本：8.0.39
- PyTorch版本：2.6.0 (CUDA 12.6)
- (其他后端依赖库版本与开发环境一致)

• 客户端：

- 操作系统：Windows 10
- 浏览器（用于界面测试和兼容性测试）：
  * Google Chrome (136.0.7103.113)
  * Mozilla Firefox (138.0.3)
  * Microsoft Edge (136.0.3240.64)
- Node.js版本（用于运行前端开发服务器进行测试）：18.17.1 LTS

3. 测试工具：

• 手动测试： 对于用户界面的交互流程、视觉效果和易用性，主要采用人工手动测试的方式。

• API接口测试工具：Postman。 用于对后端API接口的功能、参数校验、响应状态码、返回数据格式等进行详细测试。

• 浏览器开发者工具： 用于前端调试、性能分析（如网络请求、页面加载时间）、元素审查和响应式设计检查。

• 版本控制工具：Git。 用于代码管理和版本回溯。

在进行测试前，确保本地部署的后端服务、前端应用和数据库均已正确启动并处于可访问状态。

6.2 功能测试

功能测试旨在验证系统的各项功能是否按照需求规格说明书正确实现。我们针对系统的主要模块和核心用例设计了测试用例，并通过执行这些用例来检查功能的完整性和正确性。

6.2.1 用户模块功能测试

测试目的： 验证用户注册、登录、个人信息查看与修改、密码修改等功能的正确性。 主要测试用例：

表 6-1 用户模块主要功能测试用例

用例ID	测试功能点	测试步骤	预期结果	测试结果
TC-U-001	用户注册	1. 访问注册页面。 2. 输入有效的用户名、邮箱、密码、确认密码、姓名。 3. 点击注册按钮。	1. 提示注册成功。 2. 页面跳转到登录页。 3. 数据库新增用户记录。	通过
TC-U-002	用户注册（无效邮箱）	1. 输入无效格式邮箱。 2. 其他信息有效。 3. 点击注册。	提示邮箱格式错误，注册失败。	通过
TC-U-003	用户注册（密码不一致）	1. 两次输入密码不同。 2. 其他信息有效。 3. 点击注册。	提示两次密码不一致，注册失败。	通过

Continued on next page

表 6-1 Continued from previous page

用例ID	测试功能点	测试步骤	预期结果	测试结果
TC-U-004	用户注册（用户名已存在）	1.输入已存在的用户名。 2.其他信息有效。 3.点击注册。	提示用户名已存在，注册失败。	通过
TC-U-005	用户登录	1.访问登录页面。 2.输入已注册的有效用户名和密码。 3.点击登录。	1.提示登录成功。 2.页面跳转到用户仪表盘。 3.获取并存储JWT。	通过
TC-U-006	用户登录（错误密码）	1.输入有效用户名和错误密码。 2.点击登录。	提示用户名或密码错误，登录失败。	通过
TC-U-007	查看个人信息	1.用户登录后，进入个人中心。	页面正确显示当前用户的用户名、邮箱等信息。	通过
TC-U-008	修改个人信息	1.用户登录后，进入个人中心。 2.修改邮箱（例如）。 3.点击保存。	1.提示修改成功。 2.用户信息更新。	通过

Continued on next page

表 6-1 Continued from previous page

用例ID	测试功能点	测试步骤	预期结果	测试结果
TC-U-009	修改密码	1.用户登录后，进入个人中心。 2.输入正确的旧密码、有效的新密码和确认密码。 3.点击修改密码。	1.提示密码修改成功。 2.用户自动登出，需用新密码登录。	通过
TC-U-010	修改密码（旧密码错误）	1.输入错误的旧密码。 2.其他输入有效。 3.点击修改密码。	提示旧密码错误，修改失败。	通过
TC-U-011	用户登出	1.用户登录后。 2.点击退出登录按钮。	1.清除本地登录状态。 2.页面跳转到登录页。	通过
测试结果汇总：用户模块核心功能均按预期工作。				

6.2.2 新闻检测模块功能测试

测试目的： 验证新闻提交、多模态检测流程、结果展示、历史记录管理等功能的正确性。 主要测试用例：

表 6-2 新闻检测模块主要功能测试用例

用例ID	测试功能点	测试步骤	预期结果	测试结果
TC-D-001	提交纯文本新闻检测	1.用户登录。 2.进入新闻检测页面。 3.输入新闻标题和正文。 4.不上传图片。 5.点击“开始检测”。	1.提示任务提交成功。 2.页面跳转到检测详情页，初始状态为处理中。 3.一段时间后，状态更新为完成，并显示检测结果和置信度。	通过
TC-D-002	提交图文新闻检测	1.用户登录。 2.进入新闻检测页面。 3.输入新闻标题和正文。 4.上传一张有效图片。 5.点击“开始检测”。	同TC-D-001，且详情页能看到上传的图片，分析结果中应包含图像模态的贡献（如果有）。	通过
TC-D-003	提交表单校验（空标题）	1.不输入标题，其他输入有效。 2.点击检测。	提示“请输入新闻标题”，提交失败。	通过
TC-D-004	提交表单校验（内容过短）	1.输入内容少于10字符。 2.其他输入有效。 3.点击检测。	提示“内容不能少于10个字符”，提交失败。	通过

Continued on next page

表 6-2 Continued from previous page

用例ID	测试功能点	测试步骤	预期结果	测试结果
TC-D-005	图片上传校验（非图片）	1.上传一个非图片文件（如.txt）。	提示“上传文件只能是图片格式!”，图片未被接受。	通过
TC-D-006	图片上传校验（过大）	1.上传一个超过10MB的图片。	提示“上传图片大小不能超过10MB!”，图片未被接受。	通过
TC-D-007	查看检测历史	1.用户登录。 2.进入检测历史页面。	1.正确显示该用户的历史检测列表。 2.包含标题、时间、状态、结果、置信度等信息。 3.分页功能正常。	通过
TC-D-008	筛选检测历史	1.在历史页面，按状态（如“已完成”）筛选。	列表仅显示符合筛选条件的记录。	通过
TC-D-009	查看检测详情	1.在历史页面，点击任一记录的“查看详情”。	跳转到对应的检测详情页，显示完整信息。	通过

Continued on next page

表 6-2 Continued from previous page

用例ID	测试功能点	测试步骤	预期结果	测试结果
TC-D-010	删除检测历史	1.在历史页面，点击任一记录的“删除”。 2.在确认弹窗中点击“确定”。	1.提示删除成功。 2.该记录从列表中移除。 3.数据库中对应记录被删除。	通过
TC-D-011	检测结果轮询（针对处理中任务）	1.提交一个新检测。 2.在详情页等待。	详情页应能自动刷新并最终显示“已完成”状态和结果，无需手动刷新。	通过
测试结果汇总：新闻检测模块核心功能符合预期。				

6.2.3 管理员模块功能测试

测试目的： 验证管理员后台的用户管理、检测记录查看、统计分析、参数配置、日志查看等功能的正确性。 主要测试用例： （此处仅列举部分代表性用例，实际测试会更全面）

表 6-3 管理员模块主要功能测试用例

用例ID	测试功能点	测试步骤	预期结果	测试结果
TC-A-001	管理员登录	1.使用管理员账户登录。	成功登录并跳转到管理员控制面板。	通过
TC-A-002	查看用户列表	1.管理员登录后，进入用户管理页面。	正确显示所有用户列表，包含用户名、邮箱、状态、角色等。分页功能正常。	通过
TC-A-003	筛选用户	1.在用户管理页面，按用户名或状态筛选。	列表仅显示符合条件的用户。	通过
TC-A-004	编辑用户信息	1.在用户列表，选择一个用户编辑。 2.修改其邮箱或激活状态。 3.保存。	1.提示修改成功。 2.用户列表信息更新。	通过
TC-A-005	禁用/激活用户	1.对一个激活用户执行禁用操作。 2.对一个禁用用户执行激活操作。	1.操作成功，用户状态相应改变。 2.按钮文本正确切换。	通过

Continued on next page

表 6-3 Continued from previous page

用例ID	测试功能点	测试步骤	预期结果	测试结果
TC-A-006	查看所有检测记录	1.管理员登录后，进入检测列表页面。	显示系统中所有的检测记录，而不仅是管理员自己的。分页和筛选功能正常。	通过
TC-A-007	查看全局检测统计	1.进入检测统计分析页面。	显示全局的检测统计数据和图表（如总检测量、各类结果分布）。	通过
TC-A-008	修改模型权重参数	1.进入参数设置页面。 2.调整本地模型权重滑块。 3.点击保存。	1.提示保存成功。 2.LLM权重自动更新。 3.参数值在数据库中被更新。	通过
TC-A-009	修改API密钥	1.进入参数设置页面。 2.修改Open-Router API密钥。 3.点击保存。	1.提示保存成功。 2.后端`.env`文件（或对应配置）中的密钥被更新（需验证）。	通过

Continued on next page

表 6-3 Continued from previous page

用例ID	测试功能点	测试步骤	预期结果	测试结果
TC-A-010	查看系统日志	1. 进入系统日志页面。	正确显示后端系统运行日志的最新内容。	通过
测试结果汇总：管理员模块核心功能基本符合预期。				

在功能测试过程中，除了验证预期功能外，还特别关注了边界条件、异常输入和用户权限隔离等情况，确保系统的健壮性。

6.3 性能测试

性能测试旨在评估系统在特定负载下的响应速度、吞吐量和资源利用率。考虑到毕业设计的时间和资源限制，本节主要进行简要的、关键接口的性能摸底测试。

测试结果与分析： 为评估系统核心检测接口的响应性能，我们准备了四类典型的测试样本，并在后端模型及LLM服务均已加载并充分预热的条件下，通过多次（本测试中为10次）提交并记录从请求发送到接收到完整检测结果的平均时间。测试样本类型及对应的平均响应时间如下：

- 纯短文本（约50-100中文字符）： 平均响应时间约为 17.23秒。
- 纯长文本（约300-500中文字符）： 平均响应时间约为 26.64秒。
- 短文本 + 代表性尺寸图片（约500KB JPG格式）： 平均响应时间约为 18.36秒。
- 长文本 + 代表性尺寸图片（约500KB JPG格式）： 平均响应时间约为 30.27秒。

从测试结果可以看出：

1. 文本长度影响： 纯文本检测的响应时间与文本长度呈现正相关关系。长文本（26.64秒）相比短文本（17.23秒）的检测时间显著增加，这主要是由于LLM交叉验证模块在处理更长文本时需要更多的分析和推理时间。

2. 图像引入开销： 在文本长度相近的情况下，引入图像进行多模态检测会增加额外的响应时间。例如，短文本加图片的响应时间（18.36秒）略高于纯短文本（17.23秒），长文本加图片的响应时间（30.27秒）也高于纯长文本（26.64秒）。这部分增加的开销主要来自于图像的预处理、ResNet50模型的特征提取以及多模态融合部分的计算。

3. LLM主导耗时： 即使是纯短文本检测，其响应时间也达到了17秒以上，这表明LLM的API调用和远程推理是整个检测流程中的主要耗时瓶颈，尤其是当系统配置为调用多个LLM进行交叉验证时，网络延迟和各LLM自身的处理速度都会累加。本地多模态模型的推理（BERT和ResNet）虽然也需要计算资源，但在有GPU加速的情况下，其耗时相比LLM调用可能较小。

4. 用户体验考量： 对于一个Web应用，17到30秒的同步等待时间对于用户来说是比较长的，可能会影响用户体验，尤其是在期望快速得到反馈的场景下。

综合分析，当前系统的检测接口响应时间在本地测试环境下，对于一次性的、非高频的检测请求，用户或许可以接受这种非即时反馈。然而，结果也清晰地指出了性能优化的方向：首先，针对LLM交叉验证部分，可以考虑优化提示工程、选择响应更快的LLM模型（如果质量可接受），或者引入更智能的LLM调度策略（例如，仅当本地多模态模型置信度不高时才触发完整的LLM验证）。其次，也是更重要的，应将整个检测流程（特别是模型推理和LLM API调用这两个耗时最长的环节）改造为异步任务处理模式。例如，用户提交检测后，后端立即返回一个任务接受成功的响应，并在后台通过任务队列（如Celery）执行实际的检测流程；前端则通过轮询或WebSocket等方式获取最终的检测结果。这样的异步化改造将极大地改善用户前端的等待体验，使得系统更具实用性。

由于本地部署环境的限制和毕业设计周期的原因，更全面的压力测试和瓶颈分

析未在本阶段深入进行，但上述初步测试为系统性能提供了一个基本评估。

6.4 兼容性测试

兼容性测试主要验证系统前端在不同主流Web浏览器上的表现是否一致，确保用户在不同环境下均能获得良好的使用体验。

- 测试浏览器：
  - Google Chrome（版本 136.0.7103.113）
  - Mozilla Firefox（版本 138.0.3）
  - Microsoft Edge（Chromium内核，版本 136.0.3240.64）
- 测试内容：
  - 页面布局和样式显示是否正常，无错位、重叠等问题。
  - 各项交互功能（按钮点击、表单输入、文件上传、弹窗、图表渲染等）是否均可正常操作。
  - 动态数据的加载和展示是否正确。
  - 响应式设计在不同浏览器窗口大小下的表现（虽然主要为桌面端设计，但基本缩放应能适应）。
- 测试结果与分析： 在上述三款主流浏览器的最新版本上对系统的主要页面和功能进行了测试。结果显示，系统在Google Chrome和Microsoft Edge上表现完美，所有功能和样式均正常。在Mozilla Firefox上，绝大部分功能正常，仅在管理员后台的某个复杂图表tooltip的定位上存在轻微的像素级偏差，但不影响核心功能使用。总体而言，系统在主流浏览器上具有良好的兼容性。未对更早版本的浏览器或IE浏览器进行测试。

6.5 测试结果分析与总结

通过上述功能测试、初步性能测试和兼容性测试，可以对本系统的实现质量进行如下总结：

- 功能完整性： 系统基本实现了需求分析阶段定义的各项核心功能。用户可以完成从注册登录到新闻提交、查看结果、管理历史的完整流程。管理员也能够有效地进行用户管理、查看全局检测数据、配置系统参数和监控日志。所有主要业务流程均已打通。
- 功能正确性： 大部分功能点按照预期设计正确工作。用户输入校验、权限控制、数据存储与展示、模型调用与结果反馈等关键环节均通过了测试用例。
- 性能表现： 在本地测试环境下，关键接口的响应时间基本在可接受范围内。纯文本检测速度较快，图文检测因涉及图像处理和更复杂的模型推理，耗时相对较长，但对于非即时性要求极高的场景尚可接受。用户认证等基础操作在高并发下表现稳定。
- 兼容性： 系统在当前主流浏览器（Chrome, Firefox, Edge最新版）上表现出良好的兼容性，用户体验一致。
- 潜在问题与待改进点：
  - API密钥更新机制： 后端直接修改.env文件的方式在某些生产环境中可能存在权限问题或不生效，需要更稳妥的配置管理方案或明确的操作指引。
  - 异步处理： 对于耗时的检测任务（特别是包含LLM调用时），目前的同步处理方式可能会导致用户等待时间较长，未来可以引入Celery等任务队列进行异步化改造，提升用户体验。
  - 前端筛选逻辑： 检测历史页面的筛选目前是在前端完成的，当数据量非常大时，性能可能会下降。可以考虑将筛选逻辑更多地移至后端API层面实现。

- 错误处理细化： 虽然有基本的错误提示，但可以针对不同类型的后端错误（如模型加载失败、LLM API限流等）在前端给出更具体、更友好的用户提示和处理建议。
- 自动化测试覆盖： 本次测试以手动和API工具测试为主，未来若系统持续迭代，应增加单元测试和集成测试的覆盖率，以保证代码质量和快速回归。

总体而言，系统在当前阶段已达到毕业设计预期的主要目标，是一个功能可用、性能基本满足要求的原型系统。测试过程中发现的一些问题和待改进点，为系统的后续优化指明了方向。

6.6 本章小结

本章对基于多模态深度学习的社交媒体虚假新闻检测系统进行了系统的测试。首先，详细描述了测试所采用的硬件和软件环境，以及使用的主要测试工具（如Postman、浏览器开发者工具）。随后，针对系统的用户模块、新闻检测模块和管理员模块的核心功能，设计并执行了一系列详细的测试用例，并记录了测试结果，结果表明系统主要功能均能按预期正确工作。接着，对系统的关键性能指标（如检测接口响应时间）进行了简要测试，并对前端界面在不同主流浏览器上的兼容性进行了验证。最后，对整体测试结果进行了分析与总结，肯定了系统在功能完整性、正确性、基本性能和兼容性方面取得的成果，同时也指出了测试过程中发现的一些潜在问题和未来可供改进的方向，如API密钥更新机制的稳健性、耗时任务的异步处理、前端筛选优化以及自动化测试的覆盖等。通过本章的测试工作，验证了系统的可行性和基本可用性，为系统的最终评估和后续的优化迭代提供了重要的实践依据。

第7章　总结与展望

本毕业设计旨在应对社交媒体时代虚假新闻泛滥的挑战，通过综合运用多模态深度学习、自然语言处理、计算机视觉及Web开发等技术，成功设计并实现了一个具备自动化检测能力的社交媒体虚假新闻检测系统。经过需求分析、系统设计、模型构建、系统实现及初步测试等阶段，本研究基本达到了预期的目标。本章将对整个毕业设计的工作进行全面总结，阐述系统的主要创新点，分析存在的不足，并对未来的研究方向进行展望。

7.1　工作总结

本毕业设计的主要工作及取得的成果可以概括为以下几个方面：

1. 相关技术研究与系统需求分析： 深入研究了虚假新闻的特征、传播模式，以及单模态与多模态虚假新闻检测技术的发展现状。详细梳理了深度学习模型（特别是BERT和ResNet）在文本和图像特征提取中的应用，以及大语言模型（LLM）在事实核查中的潜力。在此基础上，结合实际应用场景，对系统的功能需求（用户端与管理员端）和非功能需求（性能、易用性、安全性等）进行了全面的分析与定义。

2. 系统总体设计与数据库构建： 采用了前后端分离的系统架构，前端基于Vue.js 2和ElementUI，后端基于Python Django框架。对系统的主要功能模块（用户认证、新闻提交与处理、多模态检测核心、结果管理、前端交互、系统管理等）进行了划分。设计了系统的E-R模型，并利用Django ORM与MySQL数据库构建了用户表、检测记录表和系统设置表，实现了数据的持久化存储与高效管理。同时，规划了符合RESTful风格的API接口，用于前后端数据交互。

3. 多模态检测模型的构建与优化： 针对虚假新闻的图文特性，设计并实现了一个融合文本与图像信息的多模态深度学习模型。该模型采用预训练的bert base chinese模型提取文本语义特征，采用预训练的ResNet50模型提取图像视觉特征。通过特征级联与全连接层进行特征融合，并使用Sigmoid分类器输出最终的真伪概率。模型的训练采用了AdamW优化器、BCEWithLogitsLoss损失函数，并结合了差分

学习率、学习率动态调度（ReduceLROnPlateau）和早停等策略进行优化。实验结果表明，（此处可简述模型在测试集上的核心指标，例如：）该模型在测试集上达到了91.82%的准确率和93.19%的虚假新闻F1分数，表现出良好的检测性能。

4. LLM交叉验证机制的集成： 为了增强检测结果的可解释性和鲁棒性，设计并实现了LLM交叉验证模块。该模块通过调用OpenRouter平台的API，利用多个不同的大语言模型对新闻内容进行事实核查，并将LLM的判断结果与多模态模型的输出进行加权融合，生成最终的检测结论。这一机制为用户提供了更丰富的判断依据。

5. Web应用系统的完整实现： 基于Django和Vue.js技术栈，成功开发了一个功能相对完善的Web应用程序。

- 用户端： 实现了用户注册、登录、新闻（文本和可选图片）提交检测、检测结果（包括置信度、多模态分析摘要、LLM验证详情）实时查看、个人检测历史查询与管理、个人资料与密码修改以及个性化仪表盘（展示个人检测统计）等功能。

- 管理员端： 实现了独立的后台管理界面，包括全局系统状态监控仪表盘、用户账户管理（列表、筛选、编辑、状态控制）、系统所有检测记录的查看与筛选、全局检测数据的统计分析与可视化（利用Echarts）、系统关键参数（如模型融合权重、API密钥、检测阈值）的动态配置以及系统运行日志的在线查看等功能。

6. 系统测试与评估： 对系统进行了包括功能测试、初步性能测试（接口响应时间）和浏览器兼容性测试。测试结果表明，系统核心功能均按预期实现，主要接口性能在本地环境下基本满足需求，且在主流浏览器上表现出良好的兼容性。

7.2 系统创新点与特色

本系统在设计与实现过程中，力求结合当前技术发展趋势和实际应用需求，体现出以下几方面的创新点与特色：

1. 有效的多模态信息融合： 系统不仅仅依赖单一的文本或图像信息，而是通过构建融合BERT和ResNet特征的多模态深度学习模型，能够更全面地捕捉虚假新闻在文本语义和视觉内容上的复杂特征，从而提高了检测的准确性和对复杂图文虚假新闻的识别能力。实验结果也初步验证了多模态融合相比单模态的性能优势。

2. LLM交叉验证增强与辅助决策： 创新性地引入了大语言模型（LLM）作为交叉验证和辅助决策的手段。通过集成多个外部LLM对检测内容进行事实核查，并将LLM的分析结果与本地多模态模型的输出进行融合。这不仅可能进一步提升最终判断的准确性，更重要的是，LLM提供的理由摘要等信息，显著增强了检测结果的可解释性，为用户理解为何某条新闻被判定为虚假提供了更丰富的上下文，弥补了传统深度学习模型“黑箱”的不足。

3. 用户友好的交互设计与数据可视化： 前端界面采用Vue.js和ElementUI构建，注重用户操作的便捷性和视觉呈现的清晰性。无论是普通用户提交检测、查看结果，还是管理员进行后台管理和数据分析，都力求提供直观流畅的体验。特别是在管理员的仪表盘和检测统计页面，利用Echarts等图表库对检测数据进行了多维度可视化展示，有助于管理员快速洞察虚假新闻的态势和系统运行状况。

4. 灵活可配置的系统参数： 系统设计了专门的参数配置模块，允许管理员通过Web界面动态调整影响检测结果的关键参数，如本地模型与LLM融合的权重、判断真假的置信度阈值等。这种灵活性使得系统可以根据不同场景的需求或后续模型更新进行适配和优化，而无需修改代码。

5. 相对完整的全栈实现： 本毕业设计不仅完成了核心检测模型的构建与训练，还实现了一个包含用户管理、前后端交互、数据库存储、API接口、日志监控等功能的完整Web应用系统，展现了将AI模型落地应用于实际场景的综合能力。

7.3 不足之处与未来展望

尽管本系统在设计与实现上取得了一定的成果，但受限于毕业设计的时间、资源以及当前技术的固有挑战，仍然存在一些不足之处，同时也为未来的研究和优化指明了方向。

1. 当前系统的不足之处：

- 数据集的局限性： 虽然整合了多个公开数据集，但现有数据集在规模、多样性（如覆盖更多垂直领域、不同语言文化背景的新闻）、标注质量以及时效性方面仍有提升空间。特别是对于一些新兴的虚假信息传播手段（如深度伪造内容），现有数据集可能缺乏足够的样本。

- 模型泛化能力有待进一步验证： 模型在当前测试集上表现良好，但其在更广泛、更贴近真实世界动态变化的社交媒体数据上的泛化能力，以及对不同主题、不同风格虚假新闻的适应性，还需要更大规模和更多样化的测试来验证。

- LLM调用的成本与延迟： 引入LLM交叉验证虽然提升了可解释性和潜在准确性，但也带来了额外的API调用成本（如果使用商业API）和响应延迟。在追求极致性能和低成本部署时，需要权衡。

- 同步检测处理的用户体验： 当前后端对检测任务的处理是同步的，对于复杂的检测请求，用户可能需要等待较长时间。虽然前端可以显示加载状态，但异步任务处理机制能提供更优的用户体验。

- 视觉伪造检测的深度不足： 当前图像特征提取主要依赖通用的ResNet模型，对于经过复杂编辑或AI生成的伪造图像，其识别能力可能有限，缺乏专门的视觉伪造检测模块。

- 可解释性仍需深化： 虽然LLM提供了一定的理由，但多模态深度学习模型本身的决策过程仍然不够透明。可以探索更多XAI技术来可视化或解释模型的内部判断依据。

- 安全性有待加固： 虽然采用了JWT认证和Django内置的安全措施，但在生产环境下，还需要进行更全面的安全审计和加固，如更严格的输入验证、API限流、防DDoS攻击等。

2. 未来研究与优化展望： 针对上述不足，未来的工作可以从以下几个方面进行深入探索：

- 数据集扩充与动态更新： 持续关注并引入更新、更大规模、更多样化的多模态虚假新闻数据集。可以考虑与事实核查机构合作，获取经过人工验证的高质量标注数据。建立动态更新机制，使模型能够学习到最新的虚假信息模式。
- 更先进的多模态融合模型： 探索更复杂的特征融合机制，如基于Transformer的跨模态注意力模型（例如，构建一个专门的图文融合Transformer编码器），或者利用图神经网络（GNN）融合新闻内容与社交传播上下文信息，以捕捉更深层次的模态间交互和依赖。
- 模型轻量化与边缘部署： 研究模型压缩、知识蒸馏等技术，以减小模型体积和推理延迟，使其更适合在资源受限的环境或边缘设备上部署。
- 异步任务处理与实时反馈： 将后端耗时的检测任务（尤其是LLM调用和复杂模型推理）改造为基于Celery等任务队列的异步处理机制，前端通过WebSocket或长轮询获取实时状态更新和结果，显著改善用户体验。
- 集成专用视觉伪造检测技术： 引入针对图像篡改、AI生成图像（如Deepfake）的专用检测算法模块，作为多模态分析的一个重要补充，提高对视觉欺骗的识别能力。
- 增强可解释性研究： 结合LIME、SHAP等模型无关的解释方法，或针对特定模型结构的注意力可视化等技术，提供更直观的模型决策依据，提升用户对系统结果的信任度。

- 用户反馈与模型迭代： 在系统中加入用户反馈机制，允许用户对检测结果进行评价或纠错。收集这些反馈数据，用于模型的持续学习和迭代优化，形成一个闭环的改进系统。
- 多语言与跨领域适应性： 将模型扩展到支持更多语种的虚假新闻检测，并研究如何提高模型在不同新闻领域（如政治、健康、科技、娱乐等）之间的泛化能力和迁移学习效果。

通过持续的研究与迭代，有望将本系统打造成为一个更强大、更智能、更实用的虚假新闻治理工具，为构建清朗的网络空间贡献力量。

谢　辞

时光荏苒，岁月如梭，四年的大学本科学习生涯即将画上圆满的句号。回首这段旅程，充满了探索的艰辛与收获的喜悦。本毕业设计（论文）——“基于多模态深度学习的社交媒体虚假新闻检测系统”的顺利完成，离不开众多师长、同学和家人的悉心指导与无私帮助，在此谨致以最诚挚的谢意。

首先，我要向我的指导教师李亚老师表达最衷心的感谢。在整个毕业设计的过程中，从最初的选题立意到研究方案的制定，从系统开发遇到瓶颈时的悉心点拨，到论文撰写阶段的字斟句酌，李老师都给予了我极大的耐心和专业的指导。李老师严谨的治学精神和认真负责的工作态度使我深受教益，并将激励我在未来的道路上不断求索。 同时，我还要感谢信息工程与自动化学院各位老师在本科四年学习期间的辛勤培养和谆谆教诲。正是老师们在课堂内外传授的专业知识和科学方法，为我完成本次毕业设计打下了坚实的基础。特别感谢在课程设计、项目实践以及论文评审过程中给予我宝贵意见和建议的老师们。

感谢我的同学们和朋友们。在毕业设计的攻坚阶段，我们相互学习、共同探讨，你们的陪伴和鼓励让我在面对困难时不再孤单。与你们交流碰撞出的思维火花，也常常为我带来新的启发。感谢你们在学习和生活上给予我的无私帮助和支持。

此外，我还要特别感谢我的家人。你们一直是我最坚实的后盾，你们的理解、支持和无微不至的关怀，是我能够全身心投入学习和研究的强大动力。你们的鼓励是我克服一切困难的力量源泉。

最后，感谢昆明理工大学为我们提供了优良的学习环境和丰富的教育资源。在这里的四年，我不仅学到了专业知识，更培养了独立思考和解决问题的能力。

本毕业设计的研究工作虽然已经告一段落，但探索的道路永无止境。在未来的学习和工作中，我将继续努力，不断提升自我，不辜负所有关心和帮助过我的人的期望。

再次向所有在本次毕业设计（论文）撰写及大学学习生涯中给予我指导、支持和帮助的老师、同学、朋友和家人表示最诚挚的感谢！

参考文献

[1] Kaplan, A. M., & Haenlein, M. (2010). Users of the world, unite! The challenges and opportunities of Social Media. Business Horizons, 53(1), 59-68.

[2] Kemp, S. (2023, January). Digital 2023: Global Overview Report. DataReportal. Retrieved from https://datareportal.com/reports/digital-2023-global-overview-report

[3] Shu, K., Sliva, A., Wang, S., Tang, J., & Liu, H. (2017). Fake news detection on social media: A data mining perspective. ACM SIGKDD Explorations Newsletter, 19(1), 22-36.

[4] Allcott, H., & Gentzkow, M. (2017). Social media and fake news in the 2016 election. Journal of Economic Perspectives, 31(2), 211-236.

[5] Cinelli, M., Quattrociocchi, W., Galeazzi, A., Valensise, C. M., Brugnoli, E., Schmidt, A. L., Zola, P., Zollo, F., & Scala, A. (2020). The COVID-19 social media infodemic. Scientific Reports, 10(1), 16598.

[6] Conroy, N. K., Rubin, V. L., & Chen, Y. (2015). Automatic deception detection: Methods for finding fake news. In Proceedings of the Association for Information Science and Technology, 52(1), 1-4.

[7] Rubin, V. L., Conroy, N., Chen, Y., & Cornwell, S. (2015). Towards news verification: Deception detection methods for news discourse. In Proceedings of the 48th Hawaii International Conference on System Sciences (pp. 1-10). IEEE.

[8] Wang, W. Y. (2017). "Liar, Liar Pants on Fire": A New Benchmark Dataset for Fake News Detection. In Proceedings of the 55th Annual Meeting of

the Association for Computational Linguistics (Volume 2: Short Papers) (pp. 422-426). Association for Computational Linguistics.

[9] Ma, J., Gao, W., Mitra, P., Kwon, S. K., Jansen, B. J., Wong, K. F., & Cha, M. (2016). Detecting rumors from microblogs with recurrent neural networks. In Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI-16) (pp. 3818-3824).

[10] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers) (pp. 4171-4186). Association for Computational Linguistics.

[11] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019). RoBERTa: A Robustly Optimized BERT Pretraining Approach. arXiv preprint arXiv:1907.11692.

[12] Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., & Le, Q. V. (2019). XLNet: Generalized Autoregressive Pretraining for Language Understanding. In Advances in Neural Information Processing Systems 32 (NeurIPS 2019) (pp. 5753-5763).

[13] Shu, K., Wang, S., & Liu, H. (2017). Exploiting Proximity on Social Media for Fake News Detection. In Proceedings of the KDD 2017 Workshop on Data Mining for Social Good (SoGood).

[14] Wu, K., Yang, S., & Zhu, K. Q. (2015). False rumors detection on Sina Weibo by propagating flow framework. In 2015 IEEE International Conference on Communications (ICC) (pp. 2008-2013). IEEE.

[15] Jin, Z., Cao, J., Zhang, Y., & Luo, J. (2017). Multimodal fusion with recurrent neural networks for rumor detection on microblogs. In Proceedings of the 25th ACM international conference on Multimedia (pp. 795-816).

[16] Pérez-Rosas, V., Kleinberg, B., Lefevre, A., & Mihalcea, R. (2018). Automatic detection of fake news. In Proceedings of the 27th International Conference on Computational Linguistics (COLING 2018) (pp. 3391-3401).

[17] Zadeh, A. B., Chen, M., Poria, S., Cambria, E., & Morency, L. P. (2017). Tensor fusion network for multimodal sentiment analysis. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP) (pp. 1103-1114).

[18] Wang, Y., Ma, F., Jin, Z., Yuan, Y., Xun, G., Jha, K., Su, L., & Gao, J. (2018). EANN: Event Adversarial Neural Networks for Multi-Modal Fake News Detection. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '18) (pp. 849-857). Association for Computing Machinery.

[19] Khattar, D., Goud, J. S., Gupta, M., & Varma, V. (2019). MVAE: Multimodal Variational Autoencoder for Fake News Detection. In The World Wide Web Conference (WWW '19) (pp. 2915-2921). Association for Computing Machinery.

[20] Lu, J., Batra, D., Parikh, D., & Lee, S. (2019). ViLBERT: Pretraining for Vision-and-Language Representation Learning. In Advances in Neural Information Processing Systems 32 (NeurIPS 2019) (pp. 13-23).

[21] Tan, H., & Bansal, M. (2019). LXMERT: Learning Cross-Modality Encoder Representations from Transformers. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP) (pp. 5100-5111).

[22] Chen, Y. C., Li, L., Yu, L., El Kholy, A., Ahmed, F., Gan, Z., Cheng, Y., & Liu, J. (2020). UNITER: UNiversal Image-TExt Representation Learning. In European Conference on Computer Vision (ECCV) (pp. 104-120). Springer.

[23] Singhal, S., Shah, R. R., Chakraborty, T., Kumaraguru, P., & Satoh, S. (2019). SpotFake: A Multi-modal Framework for Fake News Detection. In 2019 IEEE Fifth International Conference on Multimedia Big Data (BigMM) (pp. 39-47). IEEE.

[24] Qian, S., Wang, J., Hu, J., Zhang, A., & Li, Q. (2021). A Multi-modal Fake News Detection Model with Co-attention Mechanism. IEEE Access, 9, 29919-29929.

[25] Wu, L., Liu, H., Wang, R., & Cui, Z. (2021). A multimodal model with an attention mechanism for fake news detection on social media. Applied Soft Computing, 110, 107642.

[26] Shu, K., Mahudeswaran, D., Wang, S., Lee, D., & Liu, H. (2019). Beyond News Contents: The Role of Social Context for Fake News Detection. In Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining (WSDM '19) (pp. 338-346). Association for Computing Machinery.

[27] Lazer, D. M. J., Baum, M. A., Benkler, Y., Berinsky, A. J., Greenhill, K. M., Menczer, F., Metzger, M. J., Nyhan, B., Pennycook, G., Rothschild, D., et al. (2018). The science of fake news. Science, 359(6380), 1094-1096.

[28] Zhou, X., & Zafarani, R. (2020). A Survey of Fake News: Fundamental Theories, Detection Methods, and Opportunities. ACM Computing Surveys (CSUR), 53(5), 1-40.

[29] Vosoughi, S., Roy, D., & Aral, S. (2018). The spread of true and false news online. Science, 359(6380), 1146-1151.

[30] Baltrušaitis, T., Ahuja, C., & Morency, L. P. (2018). Multimodal machine learning: A survey and taxonomy. IEEE Transactions on Pattern Analysis and Machine Intelligence, 41(2), 423-443.

[31] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Efficient estimation of word representations in vector space. In Workshop Proceedings of the International Conference on Learning Representations (ICLR).

[32] Pennington, J., Socher, R., & Manning, C. D. (2014). GloVe: Global Vectors for Word Representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP) (pp. 1532-1543).

[33] Simonyan, K., & Zisserman, A. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. In International Conference on Learning Representations (ICLR).

[34] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR) (pp. 770-778).

[35] Tan, M., & Le, Q. V. (2019). EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In International Conference on Machine Learning (ICML) (pp. 6105-6114). PMLR.

[36] Ramachandram, D., & Taylor, G. W. (2017). Deep Multimodal Learning: A Survey on Recent Advances and Trends. IEEE Signal Processing Magazine, 34(6), 96-108.

[37] Atrey, P. K., Hossain, M. A., El Saddik, A., & Kankanhalli, M. S. (2010). Multimodal fusion for multimedia analysis: A survey. Multimedia systems, 16(6), 345-379.

[38] Bahdanau, D., Cho, K., & Bengio, Y. (2015). Neural Machine Translation by Jointly Learning to Align and Translate. In International Conference on Learning Representations (ICLR).

[39] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, ., & Polosukhin, I. (2017). Attention is all you need. In Advances in Neural Information Processing Systems 30 (NeurIPS 2017) (pp. 5998-6008).

[40] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. In Advances in Neural Information Processing Systems 33 (NeurIPS 2020) (pp. 1877-1901).

[41] Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M. A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al.

(2023). LLaMA: Open and Efficient Foundation Language Models. arXiv preprint arXiv:2302.13971.

[42] Jones, M. B., Bradley, J., & Sakimura, N. (2015, May). JSON Web Token (JWT). RFC 7519. IETF. https://www.rfc-editor.org/info/rfc7519 (DOI: 10.17487/RFC7519)

[43] Yang, Z., Pang, Y., Li, Q., Wei, S., Wang, R., & Xiao, Y. (2024). A model for early rumor detection base on topic-derived domain compensation and multi-user association. Expert Systems with Applications, 123951. Elsevier.

[44] Yuan, C., Ma, Q., Zhou, W., Han, J., & Hu, S. (2019). Jointly embedding the local and global relations of heterogeneous graph for rumor detection. In The 19th IEEE International Conference on Data Mining (ICDM). IEEE.

[45] Li, Y., He, H., Bai, J., & Wen, D. (2024). MCFEND: A Multi-source Benchmark Dataset for Chinese Fake News Detection. In Proceedings of The Web Conference (WWW).

[46] Loshchilov, I., & Hutter, F. (2019). Decoupled Weight Decay Regularization. In International Conference on Learning Representations (ICLR).

附录A　核心代码示例

本附录展示了系统中部分核心模块的关键代码片段，以更具体地说明其实现细节。这些代码片段旨在帮助读者理解系统关键功能的内部逻辑。

A.1　后端核心代码

A.1.1　多模态模型定义（detection/ml/model.py）

以下代码片段展示了系统中融合文本和图像特征的多模态深度学习模型MultimodalFakeNewsModel的核心结构。该模型不包含元数据处理。

```
# backend/detection/ml/model.py
import torch
import torch.nn as nn
from torchvision.models import resnet50
from transformers import AutoModel
from django.conf import settings

class MultimodalFakeNewsModel(nn.Module):
    def __init__(self, text_model_name=settings.TEXT_MODEL_NAME,
                 image_model_name=settings.IMAGE_MODEL_NAME,
                 text_embedding_dim=settings.TEXT_EMBEDDING_DIM,
                 img_embedding_dim=settings.IMG_EMBEDDING_DIM,
                 fusion_output_dim=settings.FUSION_OUTPUT_DIM):
        super().__init__()
        # Text Encoder
        self.text_encoder = AutoModel.from_pretrained(text_model_name)

        # Image Encoder
        if image_model_name == 'resnet50':
            self.image_encoder = resnet50(weights=None) # 在实际服务中加载预训练权重
            self.image_encoder.fc = nn.Identity() # 移除分类头
```

```
            self.img_embedding_dim = img_embedding_dim
        else:
            raise ValueError(f"Image model '{image_model_name}' not
↪ supported.")

        # Fusion Layer
        self.fusion_dim = text_embedding_dim + self.img_embedding_dim
        self.fusion_layer = nn.Sequential(
            nn.Linear(self.fusion_dim, fusion_output_dim),
            nn.ReLU(),
            nn.Dropout(0.5)
        )
        # Classifier Head
        self.classifier = nn.Linear(fusion_output_dim, 1)

    def forward(self, input_ids, attention_mask, image, image_available):
        # Text Features
        text_outputs = self.text_encoder(input_ids=input_ids, attention_
↪ mask=attention_mask)
        text_features = text_outputs.last_hidden_state[:, 0, :]

        # Image Features
        batch_size = image.shape[0]
        image_features = torch.zeros(batch_size, self.img_embedding_dim,
↪ device=image.device)
        valid_image_mask = image_available.bool()
        if valid_image_mask.any():
            valid_images = image[valid_image_mask]
            valid_image_features = self.image_encoder(valid_images)
            image_features[valid_image_mask] = valid_image_features

        # Fusion
        fused_features = torch.cat((text_features, image_features), dim
```

```
    ↪ =1)
52          fused_output = self.fusion_layer(fused_features)
53
54          # Classification
55          logits = self.classifier(fused_output)
56          return logits.squeeze( 1)
```

Listing 1 多模态模型核心结构

此模型结构与训练脚本(scripts/train_model.py)中定义的无元数据模型一致，确保了训练和推理的一致性。

A.1.2 检测服务核心逻辑（detection/services/detector.py）

FakeNewsDetector类中的detect方法是执行新闻检测的核心流程。以下代码片段展示了其关键步骤，包括本地模型预测、LLM交叉验证和结果融合。

```
1  # backend/detection/services/detector.py
2  # ... (语句和import__init__方法省略) ...
3  class FakeNewsDetector:
4      # ... (__init__方法) ...
5
6      def detect(self):
7          # ... 准备文本、图像路径等() ...
8
9          #      1. 自训练模型预测      #
10         local_model_result = {'result': Detection.RESULT_UNKNOWN, '
       ↪ confidence': 0.0, 'error': None}
11         if self.model:
12             try:
13                 processed_input = preprocess_input(
14                     text=text_content, image_path=image_path,
15                     tokenizer=self.tokenizer, image_transform=self.image_
       ↪ transform,
```

```
                    device=self.device
                )
                with torch.no_grad():
                    logits = self.model(
                        input_ids=processed_input['input_ids'],
                        attention_mask=processed_input['attention_mask'],
                        image=processed_input['image'],
                        image_available=processed_input['image_available']
↪ ]
                    )
                    probability = torch.sigmoid(logits).item()
                local_model_result['result'] = Detection.RESULT_FAKE if
↪ probability >= 0.5 else Detection.RESULT_REAL
                local_model_result['confidence'] = probability if
↪ probability >= 0.5 else (1   probability)
            except Exception as e:
                logger.exception(f"本地模型推理失败: {e}")
                local_model_result['error'] = f"本地模型推理失败: {str(e)}"
        else:
            local_model_result['error'] = "本地模型未加载"
        # ... 日志记录() ...

        #      2. LLM 交叉验证      #
        llm_result = {'overall_verdict': 'Skipped', 'aggregated_
↪ confidence': 0.0, 'error': None, 'details': None}
        if settings.OPENROUTER_API_KEY:
            try:
                llm_raw_result = asyncio.run(verify_news_with_llms_async(
↪ text_content, image_path))
                if llm_raw_result and not llm_raw_result.get('error'):
                    llm_result['overall_verdict'] = llm_raw_result.get('
↪ overall_verdict', 'Parsing Error')
                    llm_result['aggregated_confidence'] = llm_raw_result.
```

```
    ↪ get('aggregated_confidence', 0.0)
43                  llm_result['details'] = llm_raw_result
44              # ... 错误处理() ...
45          except Exception as e:
46              # ... 异常处理() ...
47      else:
48           llm_result['error'] = "OpenRouter API Key 未配置"
49      # ... 日志记录() ...
50
51      #      3. 结果融合      #
52      # ... 获取本地模型和(的判断及置信度LLM) ...
53      # ... 从(获取融合权重和阈值SystemSettings) ...
54      # ... 根据权重和阈值计算(final_和resultfinal_的逻辑confidence) ...
55      # 例如:
56      if local_model_active and llm_active:
57          # ... 加权平均逻辑() ...
58          if weighted_score >= FAKE_THRESHOLD:
59              final_result = Detection.RESULT_FAKE
60              # ...
61          elif weighted_score <= REAL_THRESHOLD:
62              final_result = Detection.RESULT_REAL
63              # ...
64          else:
65              final_result = Detection.RESULT_UNKNOWN
66          # ...
67      elif local_model_active:
68          final_result = local_verdict
69          final_confidence = local_conf * 0.8 # 置信度折减示例
70          # ...
71      # ... 其他融合情况() ...
72
73      #      4. 更新数据库      #
74      analysis = { /* ... 组装详细分析结果 ... */ }
```

```
        self.detection.result = final_result
        self.detection.confidence_score = final_confidence
        self.detection.analysis_result = analysis
        self._update_status(Detection.STATUS_COMPLETED)
        # ... 日志和返回() ...
        return { /* ... */ }
```

Listing 2 新闻检测核心流程

该流程体现了系统结合自研模型和外部LLM进行综合判断的核心思想。

A.1.3 LLM API调用封装（detection/services/llm_verifier.py）

与OpenRouter平台交互以调用大语言模型的关键函数是call_openrouter_api_async。

```
# backend/detection/services/llm_verifier.py
# ... (和配置省略import) ...
async def call_openrouter_api_async(prompt, model_name, image_base64=None):
    if not AsyncOpenAI: return format_structured_llm_response(model_name, False, error="AsyncOpenAI client not available.")
    if not API_KEY: return format_structured_llm_response(model_name, False, error="OpenRouter API key not configured.")

    client = AsyncOpenAI(base_url="https://openrouter.ai/api/v1", api_key=API_KEY)
    messages = [{"role": "user", "content": []}]
    messages[0]["content"].append({"type": "text", "text": prompt})

    if image_base64 and model_name in OR_IMAGE_MODELS:
        # ... 处理图像数据，添加到(messages) ...
        mime = "image/jpeg" if image_base64.startswith("/9j/") else "image/png"
        image_url = f"data:{mime};base64,{image_base64}"
```

```
        messages[0]["content"].append({"type": "image_url", "image_url":
    ↪ {"url": image_url}})

    try:
        completion = await client.chat.completions.create(
            extra_headers={"HTTP Referer": YOUR_SITE_URL, "X Title": YOUR
    ↪ _SITE_NAME},
            model=model_name,
            messages=messages,
            max_tokens=700,
            temperature=0.3,
        )
        raw_content = completion.choices[0].message.content
        # ... 尝试解析(并格式化响应JSON) ...
        try:
            if raw_content.strip().startswith("```json"):
                raw_content = raw_content.strip()[7: 3].strip()
            # ... 其他清理() ...
            parsed_data = json.loads(raw_content)
            return format_structured_llm_response(model_name, True, data=
    ↪ parsed_data)
        except Exception as parse_err: # 更通用的解析错误捕获
            logger.warning(f"解析响应失
    败LLM ({model_name}): {parse_err}. Raw: {raw_content}")
            return format_structured_llm_response(model_name, False,
    ↪ error=str(parse_err), data=raw_content)

    except Exception as e:
        # ... (调用异常处理API) ...
        return format_structured_llm_response(model_name, False, error=
    ↪ str(e))
```

Listing 3 异步调用OpenRouter LLM API

该函数使用了`asyncio`和`aiohttp` (通过`openai`库间接使用)来实现对LLM API

的异步调用，以提高并发处理能力，尽管在`Detector`中是通过`asyncio.run()`同步执行的。

A.2 前端核心代码

A.2.1 新闻检测提交逻辑（**frontend/src/views/detection/Create.vue**）

用户在前端提交新闻进行检测的核心逻辑位于Create.vue组件的submitForm方法。

```
// frontend/src/views/detection/Create.vue
// <template> ... </template>
// <script>
// ... (data, 等省略rules) ...
methods: {
  handleImageChange(file) {
    const isImage = file.raw.type.includes('image/');
    const isLt10M = file.raw.size / 1024 / 1024 < 10;
    if (!isImage) {
      this.$message.error上传文件只能是图片格式('!');
      this.fileList = []; return false;
    }
    if (!isLt10M) {
      this.$message.error上传图片大小不能超过(' 10MB!');
      this.fileList = []; return false;
    }
    this.detectionForm.image = file.raw;
    this.fileList = [file]; // ElementUI 组件通常这样更新显示upload
  },
  submitForm() {
    this.$refs.detectionForm.validate(valid => {
      if (valid) {
        this.loading = true;
```

```
        const formData = new FormData();
        formData.append('title', this.detectionForm.title);
        formData.append('content', this.detectionForm.content);
        if (this.detectionForm.image) {
          formData.append('image', this.detectionForm.image);
        }

        // 直接使用封装的函数API 假设已从(@/api/导入detectioncreateDetection)
        // import { createDetection } from '@/api/detection';
        // createDetection(formData) // 在实际代码中应该是这样
        this.$store.dispatch('detection/createDetection', this.
    ↪ detectionForm) // 或者通
    过Vuex Action
          .then(response => { // response 应该是新创建的检测对象
            this.$message.success检测任务已提交('');
            this.$router.push(`/dashboard/detection/detail/${response.id
    ↪ }`);
          })
          .catch(error => {
            console.error检测提交失败(':', error);
            this.$message.error检测提交失
    败(': ' + (error.response?.data?.detail || 请稍后重试''));
          })
          .finally(() => {
            this.loading = false;
          });
      }
    });
  },
  resetForm() { /* ... */ }
}
// ... </script>
```

Listing 4 前端新闻检测提交

该方法首先进行表单校验，然后构造`FormData`对象以支持图片上传，最后调用封装的API函数（或Vuex Action）将数据提交到后端。

A.2.2 检测详情页数据获取与轮询（frontend/src/views/detection/Detail.vue）

检测详情页在加载时获取数据，并在任务处理中时进行轮询。

```
// frontend/src/views/detection/Detail.vue
// <template> ... </template>
// <script>
// ... (import, data, 等省略computed) ...
export default {
  name: 'DetectionDetail',
  data() {
    return {
      loading: true,
      detectionId: null,
      pollingInterval: null,
      pollingDelay: 5000, // 秒轮询一次5
      // ...
    };
  },
  computed: {
    detection() {
      const det = this.$store.getters.currentDetection;
      // 如果检测已完成或失败，停止轮询
      if (det && (det.status === 'completed' || det.status === 'failed'))
    ↪  {
        this.stopPolling();
      }
      return det;
    },
    isProcessing() {
      return this.detection && (this.detection.status === 'pending' ||
```

```
    ↪ this.detection.status === 'processing');
    },
    // ... 其他计算属性()
  },
  created() {
    this.detectionId = this.$route.params.id;
    this.fetchDetailAndPoll();
  },
  beforeDestroy() {
    this.stopPolling();
  },
  methods: {
    fetchDetailAndPoll() {
      this.loading = true;
      this.$store.dispatch('detection/getDetectionDetail', this.
    ↪ detectionId)
        .then(() => {
          this.loading = false;
          if (this.isProcessing) {
            this.startPolling();
          }
        })
        .catch(error => {
          this.loading = false;
          // ... 错误处理()
        });
    },
    startPolling() {
      this.stopPolling(); // 先停止已有的，防止重复
      console.info开始轮询检测结果('...'); // 使用console.或infodebug
      this.pollingInterval = setInterval(() => {
        if (!this.isProcessing) {
          this.stopPolling();
```

```
          return;
        }
        console.debug轮询检测结果(` ID: ${this.detectionId}`);
        this.$store.dispatch('detection/getDetectionDetail', this.
    ↪ detectionId)
          .catch(error => {
            console.error轮询失败(':', error);
            // 可选：如果轮询连续多次失败，也停止轮询
          });
      }, this.pollingDelay);
    },
    stopPolling() {
      if (this.pollingInterval) {
        console.info停止轮询检测结果。('');
        clearInterval(this.pollingInterval);
        this.pollingInterval = null;
      }
    },
    // ... 其他方法如(formatDate, 等getStatusType)
  }
};
// ... </script>
```

Listing 5 前端检测详情页数据获取与轮询

此组件通过Vuex Action获取检测详情。若检测任务正在进行中，则会启动定时器，定期刷新数据，直到任务完成或失败。

A.2.3 Axios请求拦截器（**frontend/src/main.js**）

为统一处理API请求认证，在main.js中配置了Axios请求拦截器。

```
// frontend/src/main.js
// ... (import Vue, App, router, store, 等ElementUI) ...
import axios from 'axios';
```

```
4
5  // ... (配置baseURL) ...
6
7  // 请求拦截器，添加到请求头token
8  axios.interceptors.request.use(config => {
9    const token = store.getters.token; // 从获取Vuextoken
10   if (token) {
11     config.headers.Authorization = `Bearer ${token}`;
12   }
13   return config;
14 }, error => {
15   return Promise.reject(error);
16 });
17
18 // 响应拦截器，处理过期等问题token
19 axios.interceptors.response.use(response => {
20   return response;
21 }, error => {
22   if (error.response && error.response.status === 401) {
23     // 过期，清除用户信息并跳转到登录页token
24     store.dispatch('user/logout'); // 调用Vuex 进行登出处理action
25     router.push('/login'); // 跳转到登录页
26   }
27   return Promise.reject(error);
28 });
29
30 // ... (实例创建Vue) ...
```

Listing 6 Axios请求拦截器

请求拦截器确保了每次向后端发送请求时，如果用户已登录，则自动在请求头中附带JWT Token。响应拦截器则实现了全局的401错误处理，当Token失效时自动引导用户重新登录。

以上代码展示了系统核心功能实现，详见项目源码。