

天津科技大学



本科毕业设计

学院名称：人工智能学院

学科专业：计算机科学与技术（信息处理）

题 目：基于机器学习的智能医疗对话 APP 的设计

研究方向：机器学习

年 级：2020 学号：20201220

学生姓名：陈治宇

二〇二四年五月

学校代码 10057

密级 公开

基于机器学习的智能医疗对话 APP 的设计
**Design of an Intelligent Healthcare Dialogue
App Based on Machine Learning**

作 者	<u>陈治宇</u>	指导老师	<u>王燕</u>
申请学位	<u>工学学士</u>	培养单位	<u>人工智能学院</u>
学科专业	<u>计算机科学与技术（信息处理）</u>	研究方向	<u>机器学习</u>

二〇二四 年 五 月

天津科技大学

学位论文原创性声明

本人郑重声明：所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的作品成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全知晓本声明的法律后果由本人承担。

学位论文作者签名：陈治宇

日期：2024 年 5 月 30 日

天津科技大学

学位论文使用授权书

本人同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。

本学位论文属于：

☒ 公开论文

☐ 内部论文，保密 ☐ 1 年 / ☐ 2 年 / ☐ 3 年，过保密期后适用本授权书。

☐ 秘密论文，保密 ____ 年（不超过 10 年），过保密期后适用本授权书。

☐ 机密论文，保密 ____ 年（不超过 20 年），过保密期后适用本授权书。

（请在以上方框内选择打“√”）

学位论文作者签名：陈治宇

日期：2024 年 5 月 30 日

指导教师签名：王燕

日期：2024 年 5 月 30 日

天津科技大学本科毕业设计（论文）任务书

人工智能学院 计算机科学与技术（信息处理）专业

学生学号：20201220 学生姓名：陈治宇 指导教师姓名：王燕

完成期限：2023 年 11 月 20 日至 2024 年 5 月 30 日

一、题目名称：基于机器学习的智能医疗对话 APP 的设计

二、设计(论文)内容及要求：

设计内容：

该设计旨在开发一款基于机器学习的智能医疗对话应用程序，以提供便捷的医疗咨询和建议。该应用程序将结合自然语言处理和机器学习算法，使用户能够通过对话的方式与应用程序进行交互，获取医疗方面的信息和建议。

功能模块：

（一）用户注册与登录：

用户将能够创建个人账户并进行登录，以便记录他们的健康信息和对话历史。注册过程中，用户需要提供基本的个人信息，如姓名、性别、年龄等。

（二）智能对话功能：

1.使用自然语言处理技术，使用户能够通过文字与应用程序进行对话。

2.用户可以向应用程序提出健康问题、寻求医疗建议或获取相关医疗知识。

3.基于预先训练的机器学习模型分析用户的意图，并检索知识图谱给出用户相关答案。

（三）健康数据管理：

1.用户可以将个人的健康数据与应用进行同步，如体重、血压、血糖等。

2.记录用户的健康数据，并提供数据可视化和趋势分析功能，以帮助用户更好地管理自己的健康状况。

（四）医疗资讯和知识库：

1.提供医疗资讯和知识库，为用户提供有关常见疾病、药物信息、医疗检查等方面的知识。

2.推荐系统：根据用户的问题，行为，身体健康情况，推荐用户相关的医疗文章。

设计要求：

1.开发环境：PyCharm Community Edition 2023.2，Android Studio 2023.3.1。

2.开发语言：机器学习：Python；Android 应用：Java；数据库：MySQL，Neo4j；机器学习框架：Tensorflow1.15.0。

3. 设计美观大方，简洁实用。

4. 系统需要建立数据库，数据库使用 MySQL。

5. 系统界面设计友好，功能完全，易用性高。

6. 完成毕业实习，实习后需书写 2500 字左右的实习报告。

7. 翻译一篇英文文献，译文字数 5000 字左右。

8. 查阅大量相关文献，开题报告中，需对国内外的研究现状进行对比说明。

9. 毕业论文字数 2 万字左右，文中需对系统进行功能分析、设计、实现、测试做详细描述。

参考文献：

[1].Vaswani, Ashish, et al. Attention Is All You Need. Advances in Neural Information Processing Systems, vol. 30, 2017.

[2].Clark, Kevin, et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." arXiv preprint arXiv:1905.05950 (2019).

[3].Huang, Zhiheng, Wei Xu, and Kai Yu. Bidirectional LSTM-CRF Models for Sequence Tagging. arXiv preprint arXiv:1508.01991 (2015).

[4].He, Xiangnan, et al. Neural Collaborative Filtering. Proceedings of the 26th International Conference on World Wide Web, 2017, pp. 173-182.

[5].Bordes, Antoine, et al. Translating embeddings for modeling multi-relational data. Advances in Neural Information Processing Systems, vol. 26, 2013, pp. 2787-2795.

[6].王雪梅.在线医疗问答文本命名实体识别的研究.西南交通大学希望学院,移动信息.2021 年第 5 期 0133-0135,0142

[7].周可,方强龙,张弘强.智慧移动医疗 APP 的设计与实现,安徽信息工程学院管理工程学院,安徽信息工程学院大数据与人工智能学院.物联网技术[J]2023 年第 2 期 92-94

指导教师签字： 王燕

填写日期： 2023 年 12 月 7 日

注：本任务书发给学生，毕业设计（论文）完成后装入毕业设计（论文）档案。

摘 要

目前，医疗资源匮乏、不均的问题仍然存在。因为医疗知识结构复杂，而专业医师数量和时间有限，无法为每个人提供医疗服务。在这样的背景下，如何为用户提供一种专业医疗问诊、个性化医疗服务的系统成为一个研究热点。

本文设计与实现了一种基于机器学习的医疗对话问诊、个性化医疗推荐和预测的系统。在系统设计中，移动端使用 Android Studio 开发，后端采用 Spring Boot 框架。分别使用 MySQL 和 SQLite 来存储云端和本地数据。对话模块使用自然语言处理技术实现，其中 BERT 模型和 Text-CNN 层实现分类，并使用知识图谱检索医疗问诊资料实现医疗问答。医疗推荐模块使用 NCF 模型，利用用户的行为计算用户对物品的感兴趣评分，通过 NCF 模型计算出用户对物品的感兴趣程度。医疗预测模块使用多层感知机模型，通过用户的情况预测用户患病的概率。

在实验部分，本系统对问诊、推荐和医疗预测模型进行参数调整并一一进行验证。结果表明，该系统具有较高的准确性和可靠性，为用户提供了个性化的医疗推荐、问诊和预测服务，能实现全天候的为用户服务，缓解了医疗资源匮乏、不均的问题，具有一定的实际价值和推广意义。

关键词：机器学习；医疗对话；医疗推荐；医疗预测；自然语言处理

ABSTRACT

At present, the problem of scarce and unequal distribution of medical resources still exists. Due to the complex structure of medical knowledge and the limited number of professional physicians and time, it is impossible to provide medical services to everyone. In such a context, how to provide users with a system that offers professional medical consultations and personalized medical services has become a research hot spot.

This paper designs and implements a machine learning-based system for medical dialogue consultation, personalized medical recommendations, and predictions. In the system design, the mobile end is developed using Android Studio, and the back-end adopts the Spring Boot framework. MySQL and SQLite are used to store cloud and local data, respectively. The dialogue module utilizes natural language processing techniques, with the BERT model and Text-CNN layer used for classification, and a knowledge graph is employed to retrieve medical consultation information for medical Q&A. The medical recommendation module utilizes the NCF model to calculate the user's interest rating for items based on user behavior and determines the user's level of interest in the items. The medical prediction module employs a multilayer perceptron model to predict the probability of a user being affected by a disease based on the user's condition.

In the experimental part, this system tunes and validates the parameters of the consultation, recommendation, and medical prediction modules. The results show that the system has high accuracy and reliability, providing users with personalized medical recommendations, consultations, and predictions, and can provide services to users around the clock. It alleviates the problem of scarce and unequal distribution of medical resources, and has practical value and promotional significance.

KeyWords:Machine learning, medical dialogue, medical recommendations, medical prediction, natural language processing

目 录

第 1 章	绪论	1
1.1	课题的研究背景与意义	1
1.2	研究发展现状	1
第 2 章	相关技术知识	3
2.1	文本分类概述	3
2.2	推荐算法概述	6
2.3	机器学习相关知识	11
第 3 章	移动应用与后端设计	14
3.1	系统设计概述	14
3.2	数据库分析与设计	15
3.3	软件分层与设计模式	21
3.4	构建知识图谱	22
3.5	登录与注册模块设计	23
3.6	推荐模块设计	25
3.7	对话模块设计	27
3.8	健康模块设计	28
3.9	后台管理模块设计	29
第 4 章	实验及分析	31
4.1	BERT 模型实验与分析	31
4.2	NCF 模型实验与分析	35
4.3	医疗预测模型实验与分析	40
4.4	模型在系统中的应用测试	42
第 5 章	结论与展望	47
5.1	系统不足之处	47
5.2	展望	47
参考文献	48
致 谢	50

第 1 章 绪论

1.1 课题的研究背景与意义

随着我国的快速发展，城市医疗水平已经大幅提升。平均医疗水平却是不尽人意，导致居民在获取医疗咨询和治疗方面存在困难和不便。传统的医疗咨询和推荐往往需要人工操作和专业知识，存在信息不对称和人力资源短缺的问题，这限制了医疗服务的覆盖范围和效率。据《2022 年我国卫生健康事业发展统计公报》^[1]，到 2022 年年底，三级医院医师日均担负诊疗人次为 6.7，高于一级医院的日均担负诊疗 4.9 人次。公立与私立医院也存在同样的问题。同时乡村医师日均担负诊疗人次还上涨到 9.1，比城镇医院医师日均担负诊疗 6.2 人次更高。存在严重的医疗资源不足问题。

传统医生与患者沟通往往受时间和空间的限制，导致患者因在需要医疗建议和信息时无法及时获取而耽误治疗。

另外，部分地区的医疗科普工作也存在不足。由于医疗资源不均衡，医疗科普的开展相对有限，导致部分地区居民对健康知识的了解和认知程度较低。许多居民可能无法准确判断自身的健康状况，甚至可能不知道自己是否生病了。

在医疗资源有限且不均的情况下，如何让更多用户能够及时获得医疗知识和医疗问诊服务成为一大难题。

在当前的“人工智能+”和“互联网+”时代，人工智能可以为用户提供全天候的服务。借助人工智能技术，用户可以随时与 AI 进行在线咨询和交流。这对于解决医疗资源不足尤为重要。人工智能还可以通过分析用户的症状描述来诊断病情，可以快速推荐适合用户的医疗建议和治疗方案。这种个性化的服务可以更好地满足居民的需求，提供针对性的医疗支持。

APP 能将上述技术融合，并且具有易推广、性能优的特性。开发一款能够帮助居民及时获得医疗服务和健康指导的 APP，这将为改善医疗条件、提高居民的健康水平起到积极的推动作用。而本系统旨在通过自然语言处理技术为用户提供全天候的医疗问诊服务和医疗养身资讯推荐，从而缓解医疗资源不足问题。

1.2 研究发展现状

随着自然语言处理(NLP, Natural Language Processing)和神经网络算法的发展，NLP 浪潮几乎席卷了所有行业。其中智能聊天机器人系统是机器人系统和自然语言处理等应用领域的热门话题。聊天机器人系统可用于对话系统，用于医疗、建筑、教育等领域。

在国内，智能医疗对话获得了显著成就。国内的医疗对话模型分为：GPT(Generative Pre-trained Transformer)模型，LLM(Large Language Model)模型和 BERT(Bidirectional

Encoder Representation from Transformers)模型。其中大部分模型主要是基于 OpenAI GPT 的 API。在对话生成任务中, LLM 模型被用于编码对话历史, 并通过使用潜在变量生成多样化且富有创造性的回复, 可以提高对话系统的表现。LLM 模型可以用于对话生成任务, 通过编码对话历史和使用潜在变量来生成多样化和有创造性的回复。LLM 使用海量文本数据进行训练, 可以适应不同的任务和领域, 而不需要针对每个任务进行专门的训练和调整。但是在临床医学领域, LLM 模型如 LLaMa, ChatGLM 模型因缺乏充分的医学专业知识语料, 性能受到限制表现不佳。哈尔滨工业大学的研究团队训练出中文医学大模型“华驼”^[2]就是基于 LLaMA-7B 基座模型, 进行有监督的微调, 构建了“本草”中文医学大模型。BERT 模型是一种基于 Transformer 架构的预训练语言模型, 广泛应用于自然语言处理任务。通过双向上下文建模来捕捉输入序列的语义信息, 适用于对话中的问答、文本分类等任务。在对话生成任务中, 可以将 BERT 模型应用于编码输入对话历史和生成回复的过程。GPT 模型也是基于 Transformer 的预训练语言模型, 但与 BERT 不同, GPT 模型是单向的, 只使用上文信息生成下一个词。GPT 模型在生成任务中表现出色, 可以用于对话生成任务, 通过对对话历史进行建模, 生成连贯的回复。但是相比 BERT 的双向上下文建模, GPT 的联系上下文能力相对差一些。深圳市大数据研究院&香港中文大学(深圳)推出华佗 GPT^[3], 是通过融合 ChatGPT 生成的“蒸馏数据”和真实世界医生回复的数据, 训练并开源了一个新的医疗大模型。

在国外, 智能医疗对话也取得了显著的进展。许多国外的研究机构和企业致力于开发智能医疗助手, 以提供更好的医疗体验和个性化的健康管理。其中谷歌医疗大模型(Med-PaLM2)^[4]成效最为突出, 回答的评分高达 92.6%, 与现实中人类临床医生的水平(92.9%)相当。此模型是基于 GPT4, 利用大规模数据进行无监督训练而成的模型。相比国内的模型, 在数据和参数上有很大的优势。相比国内, 国外使用英语训练更加具有优势^[5]。英语作为全球最通用的语言, 拥有庞大的训练数据。当前开发的 NLP 语言模型主要处理英文文本, 特别是数字化医疗记录。然而, 全球范围内对于 NLP 应用的广泛采用可能导致在处理非英语临床文本时出现语言不平等和文化偏见的问题。

第 2 章 相关技术知识

本系统包含的对话模块，推荐模块和医疗预测模块需要使用到机器学习。本章主要介绍机器学习以及各个模块的相关技术知识。

2.1 文本分类概述

实现对话模块的方法可以分为：使用自然语言生成或自然语言理解+槽填充。与自然语言理解+槽填充技术相比，自然语言生成需要使用大量的算力，并且不仅在训练模型的时候需要使用 GPU 计算，而且在生成的时候同样需要使用 GPU 算力。

要使用槽填充进行回答问题就必须明确的知道搜索方式和搜索实体。而文本分类就是将自然语言转化为 CQL(Cypher Query Language)语言的关键技术。

2.1.1 自然语言处理中文本分类方法的发展

文本分类是自然语言处理领域中一项重要的任务，目标是将给定的文本内容自动归类到预定义类别中。

早期的文本分类模型主要基于循环神经网络(RNN, Recurrent Neural Network)^[6]模型。RNN 是一种用于处理序列数据的神经网络。在输入层中，会将文本输入转换为词嵌入(word embedding)向量序列，每个词映射为一个固定长度的向量。在隐藏层中，一系列 RNN 单元会根据当前输入词向量和前一时刻的隐藏状态，计算出当前时刻的新隐藏状态。在输出层中，RNN 模型会利用最终的隐藏状态进行文本分类预测。通常将隐藏状态经过一个全连接层和 Softmax 激活函数，输出各个类别的概率分布。

然而，经典的 RNN 模型也存在一些局限性，例如难以处理长序列文本，容易出现梯度消失或爆炸问题。为了克服 RNN 模型的缺陷，长短期记忆(LSTM, Long Short Term Memory)^[7]模型在 RNN 的基础上引入了细胞状态和门控机制。细胞状态作为额外变量贯穿整个序列处理过程，能够更好的保存长期的记忆信息。相比 RNN 的梯度通过隐藏层传递，LSTM 的细胞状态 C 可以直接传递到下一个时间步，不会受到隐藏状态 h 的影响。这样就大大减少了梯度消失或爆炸的风险。同时，LSTM 的门控机制(遗忘门、输入门、输出门)能够有选择性地更新细胞状态 C，进一步增强了 LSTM 对长期依赖关系的建模能力。

然而，LSTM 作为基于序列建模的循环神经网络模型需要逐步处理输入序列，在处理长序列时计算速度相对较慢，只关注当前时刻的局部信息，很难全面地捕捉输入序列的全局特征。

为了克服 LSTM 的上述问题，2017 年由 Ashish Vaswani 等人提出的 Transformer^[8]引入自注意力机制，可以并行处理输入序列，大幅提高了计算效率。Transformer 采用编码器-解码器的架构，能够对输入的序列进行全局建模，捕捉更丰富的特征信息。

Transformer 由自注意力机制(self-Attention)和前馈神经网络(Feed Forward Neural Network)组成。相比 RNN 或 LSTM 只能从文本序列的一端到另一端依次计算, Transformer 可以用其自注意力机制进行并行计算。

Transformer 首先对句子进行标记化(tokenization), 将句子转化为一个 token 序列。通过线性变换为每个 token 计算出它的 Query 矩阵、Key 矩阵和 Value 矩阵, 其中 Query 矩阵代表当前需要关注的目标 token, Key 矩阵代表序列中所有 token 的特征表示, Value 矩阵代表序列中所有 token 的语义信息。然后进行注意力计算, 注意力机制公式如公式(2-1)所示:

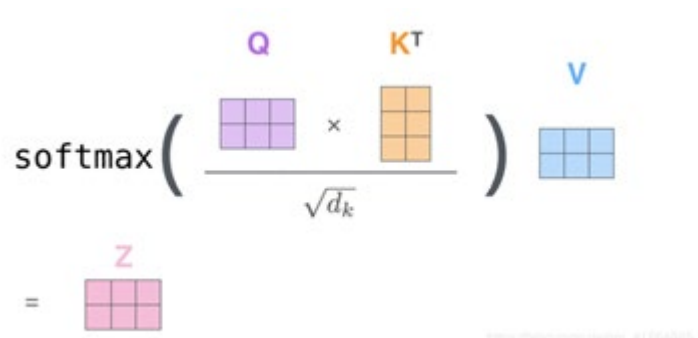
$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad \text{式(2-1)}$$


图 2-1 Transformer Self-Attention 计算

公式图解如图 2-1 所示。首先将 Q 和 K 矩阵的转置 K^T 相乘。然后除以 $\sqrt{d_k}$ 进行点积注意力的缩放(Scaled Dot-Product Attention)防止点积值导致 Softmax 函数计算时出现数值溢出或者饱和问题。然后 $\text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)$ 与 V 矩阵相乘得到每个 token 的上下文表示。然后通过前馈神经网络和全连接层, 最后输出各类文本的分类概率。Transformer 的注意力机制计算是矩阵相乘, 相比 RNN, LSTM 的处理长序列时计算较快。并且用 Positional Encoding 给输入序列的 token 添加位置信息, 让 Transformer 更全面的捕获全局的特征信息。然而 Transformer 还存在一些问题, Transformer 是一个基于 Encoder-Decoder 架构的模型, Encoder 和 Decoder 是分开的, 无法很好地捕捉双向的语义关系。在此基础上, Google 团队推出了 BERT 模型。

2.1.2 BERT 模型概述

BERT^[9]是基于 Transformer 的双向编码表示, 可以更好的捕获双向语义关系。BERT 模型是一个预训练模型, 训练时的两个任务是预测句子中被掩盖的词以及判断输入的两个句子是不是上下句。在预训练好的 BERT 模型后面根据特定任务加上相应的网络, 可以完成 NLP 的下游任务, 比如文本分类、机器翻译等。

如图 2-2 所示, 当一段句子输入 BERT 模型时, 首先会进入 Embedding 层, 由三种 Embedding 求和而成, 包括 Token Embeddings, Segment Embeddings 和 Position Embeddings。其中 Token Embeddings 是词嵌入, 是将输入的词语转换成对应的词向量表示。Segment Embeddings 用于区分输入中的两个句子, 是为了保持输入格式的一致性, 方便 BERT 模型进行后续的处理和学习。Position Embeddings 将句子做标记, 是为了让 BERT 模型学习到输入序列中每个 token 的位置信息。

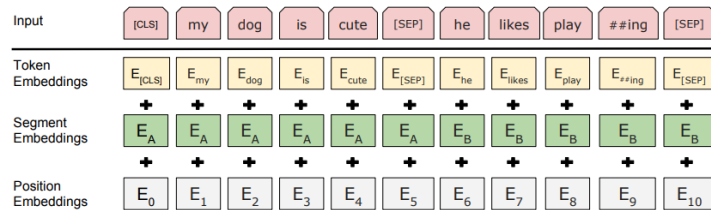


图 2-2 BERT 的 Embedding 层

经过 Embedding 层处理的 token 序列, 会进入 BERT 的 Transformer Encoder 层部分。与 Transformer 不同, BERT 使用的是双向编码。处理后每个 token 都会输出一个具有丰富上下文语义的向量表示。这些 token 表示可以用于下游的各种 NLP 任务, 如文本分类、问答、命名实体识别等。

2.1.3 Text-CNN

要想将 BERT 模型输出的 token 序列转化为具体的文本分类就需要对文本进行处理, 常规的处理办法可以通过连接一个全连接层来实现文本分类。但是全连接层容易忽略掉 token 序列中的一些细节和上下文语义。而 Text-CNN 通过使用卷积神经网络的结构, 可以捕捉到 token 序列中的局部特征和上下文语义, 可以通过将不同大小的卷积核应用于输入序列, 从而提取出不同层次的特征, 更好的实现文本分类任务。

Text-CNN(Text-Convolutional Neural Networks)^[10]是一种用于文本分类的卷积神经网络模型, 由卷积层、池化层和全连接层组成, 网络结构如图 2-3 所示。核心思想是利用卷积操作捕获文本中的局部 n-gram 特征。

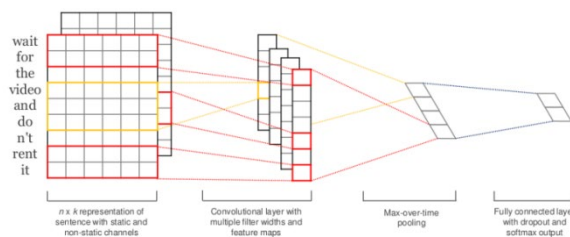


Figure 1: Model architecture with two channels for an example sentence. <https://arxiv.org/pdf/1609.03126v1.pdf>

图 2-3 Text-CNN 的网络结构

在 Text-CNN 卷积层中，卷积窗口会选择输入词向量矩阵中的部分区域与卷积核做点积运算，点积结果就是该位置的新特征值，组成了一个新的特征图。接下来卷积窗口会在输入序列上逐个滑动，直到得到完整的特征图。卷积操作会生成大量的特征图，如果直接将它们连接起来作为最终特征则会产生很高维度的特征向量。为了减少冗余信息并浓缩特征，文本处理中采用了池化操作。其作用是降低文本特征的维度，并增强模型对输入文本变化的不变性和感知范围。在 Text-CNN 的池化层中对每个特征图使用最大池化(max-pooling)，将特征图压缩为一个固定长度的向量。在保留最显著的特征的同时，大幅降低了特征维度。然后将不同大小卷积核产生的特征向量进行拼接，交给全连接层，进一步做文本分类工作。

BERT 预训练模型的训练任务是预测句子中被掩盖的词。要进行文本分类还需要对其 Transformer Encoder 层的输出做处理。可以添加一个 Text-CNN 层处理 BERT 输出的向量，从而达到文本分类的效果。

2.2 推荐算法概述

推荐系统是互联网时代中不可或缺的技术。根据用户的兴趣、爱好和行为，为其提供相关的物品、内容或服务。高效的推荐算法是推荐系统的核心，直接影响到推荐的准确性和用户体验。近年来，随着大数据和机器学习技术的快速发展，各种先进的推荐算法不断涌现。从最早的基于物品或用户的协同过滤，到后来的隐语义模型，再到基于深度学习的神经网络模型，推荐算法的理论和实践不断推进，满足了用户日益多样化的需求。

2.2.1 基于图路径的物品相似度推荐算法

物品相似度算法是利用两个物品之间的相似度来进行推荐。基本思想是，如果用户喜欢一个物品 a，那么该用户可能也喜欢与 a 相似度高的物品其他物品。而知识图谱又可以系统的存有物品之间的关系，所以一般物品相似度是基于知识图谱进行实体计算。其方法包括基于图路径的物品相似度计算，基于语义的物品相关性计算等。其中，对于专业性系统如医疗领域，基于语义的相关性计算只能通过文本计算实体的相似度，并无法捕获专业医疗体系中的实体关系，例如疾病与药物之间的治疗关系或症状与疾病之间的关联。而基于图路径相似度计算可以更好地考虑实体间的结构关系和路径信息，能够更准确地捕捉到医疗领域中的实体相似性和关联度。因此，在医疗领域的专业性系统中，基于图路径的物品相似度计算方法更具优势。

2.2.2 协同过滤算法概述

协同过滤^[11]算法是推荐系统中最为经典和广泛应用的算法之一。基本思想是，如果一

个用户喜欢某些物品，那么与该用户有相似偏好的其他用户也可能喜欢这些物品。协同过滤算法通过分析用户的历史行为数据，找出用户之间的相似性，从而为目标用户推荐感兴趣的物品。

协同过滤算法主要分为两类：基于用户的协同过滤算法(UserCF, User Collaboration Filter)和基于物品的协同过滤算法(ItemCF, Item Collaboration Filter)^[12]。

基于用户的协同过滤算法关注的是用户之间的相似性。算法首先计算出每两个用户之间的相似度，然后为目标用户推荐与其最相似的其他用户所喜欢的物品。这种方法能够捕捉用户之间的潜在偏好关系，但是当用户量很大时计算开销会很大。

相比之下，基于物品的协同过滤算法则更关注物品之间的相似性。算法首先计算出每两件物品之间的相似度，然后为目标用户推荐与其已经喜欢的物品最相似的其他物品。这种方法计算复杂度较低，但可能无法发现用户之间的隐藏偏好关系。

三、隐语义模型算法

UserCF 和 ItemCF 算法都是基于“用户-物品”的显式评分数据来计算用户或物品的相似度，从而进行推荐。然而，在真实的推荐系统中 User 的数量和 Item 的数量往往都是很大的数字，用户并不能对每个物品做出显式评分，导致用户评分矩阵数据往往稀疏，这使得相似度计算变得困难。其次，这些算法只能利用用户对物品的显式评分，如果用户对某物品没有评价并不代表用户不喜欢该物品，有可能是用户根本没有发现该物品。这种情况下，使用显式评分无法充分利用用户的潜在喜好。因此，单纯依靠用户对物品的显式评分进行协同过滤推荐，存在一定局限性。为了解决这些问题，隐语义模型(LFM, Latent Factor Model)^[13]应运而生。

为了解决基于“用户-物品”矩阵的稀疏以及无法获得潜在喜好问题，隐语义模型使用矩阵分解(MF, Matrix Factorization)技术。在推荐系统中，初始数据是用户对物品的评分矩阵稀疏 R_i ，目标是获得完整的“用户-物品”评价矩阵 R_r 。可以用矩阵分解的方法将 R_i 分解为用户对物品类别的感兴趣程度矩阵 P 和每个物品对每个类别的符合程度矩阵 Q 。如公式(2-2)所示。

$$PQ^T = R_r \quad \text{式(2-2)}$$

其中一种矩阵方法是奇异值分解^[14](SVD, Singular value decomposition),但是由于 SVD 需要的是完整的矩阵，而推荐数据的稀疏矩阵中的未评分数据也会被 SVD 当作已知信息来参与分解过程，这些未评分的元素实际上没有任何意义，但会对分解结果产生干扰。而 LFM 算法使用随机初始化填充稀疏矩阵，然后使用梯度下降算法来优化随机初始化的矩阵，使其逐渐逼近真实矩阵。

$$p(u, i) = p_u^T q_i = \sum_{k=1}^K p_{u,k} q_{i,k} \quad \text{式(2-3)}$$

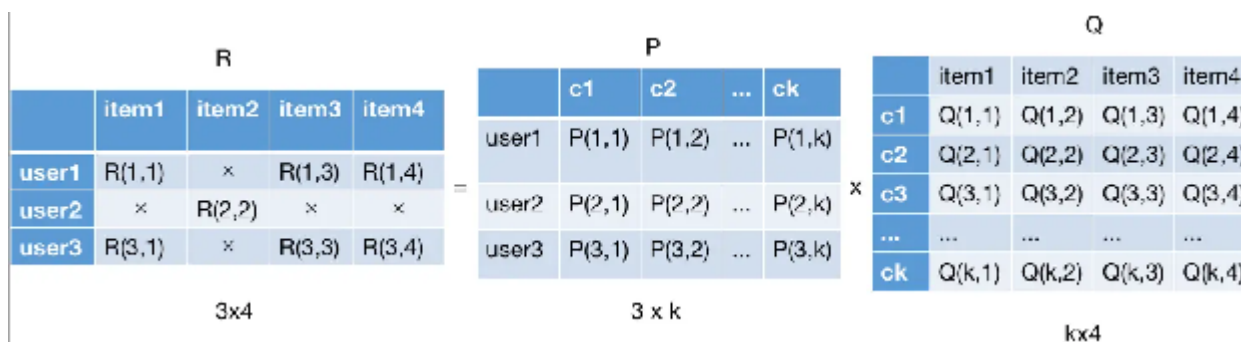


图 2-4 LMF 对稀疏矩阵进行矩阵分解

由公式(2-3)LFM 矩阵分解公式和图 2-4 所示，其中 $p(u, i)$ 表示用户 u 对物品 i 的感兴趣程度， $p_u^T q_i$ 是用户矩阵的转置与物品矩阵相乘。 $p_{u,k}$ 是用户 u 对第 k 类物品的感兴趣程度， $q_{i,k}$ 是物品 i 对第 k 类物品的符合程度。 K 是分类数量，是个超参数，并不是表示物品真实的分类，在 LFM 算法中 K 是可以调整的。在 LFM 算法解决了 SVD 没法处理的稀疏矩阵，不需要设置真实分类，只需要设置 LFM 的分类数 K ， K 越大，则粒度就越细，模型越精准。

在 LFM 的梯度下降算法中使用均方差作为目标函数，之后对这个目标函数求导，得到梯度，然后使用随机梯度下降^[15](SDG, Stochastic Gradient Descent)法更新模型参数，实现模型的反向传播训练。LFM 模型最终输出用户对物品类别的感兴趣程度矩阵 P 和每个物品对每个类别的符合程度矩阵 Q 。

2.2.3 NCF 算法

尽管基于矩阵分解的协同过滤方法在推荐系统中广泛应用，但其性能受限于采用简单的内积操作来捕捉用户和物品之间的交互关系。内积仅是将用户和物品的潜在特征进行线性组合，无法充分挖掘复杂的“用户-物品”交互模式。

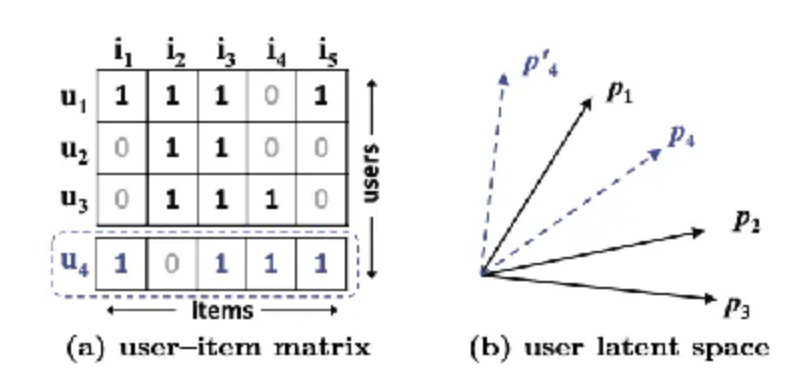


图 2-5 内积操作束缚矩阵分解的表达效果

图 2-5 中(a)代表的是 user-item 矩阵，(b)代表的是用户相似度在隐式空间中的分布。利用 Jaccard 系数 S_{AB} 可以计算出 $S_{23}(0.66) > S_{12}(0.5) > S_{13}(0.4)$ ，可得出结论 p_4 与 p_3 相比

p4 与 p2 更接近。但是根据图(b)可知，在隐式空间中 p4 与 p2 相比 p4 与 p3 更接近，这是仅由内积进行线性变换导致的排名误差。虽然可以通过增大 LFM 的潜在因子 K 来改善这一问题，但是增大隐式空间的维度又容易造成模型过拟合。

为了克服这一限制，研究者提出了神经协同过滤^[16](NCF, Neural Collaborative Filtering)框架。NCF 利用神经网络结构对用户和物品的潜在特征进行非线性建模，从而更好地捕捉“用户-物品”之间的复杂交互关系。与传统的矩阵分解方法相比，NCF 能够更灵活地学习用户偏好和物品属性之间的复杂映射，从而提高推荐系统的性能。

NCF 模型的结构如图 2-6 所示：

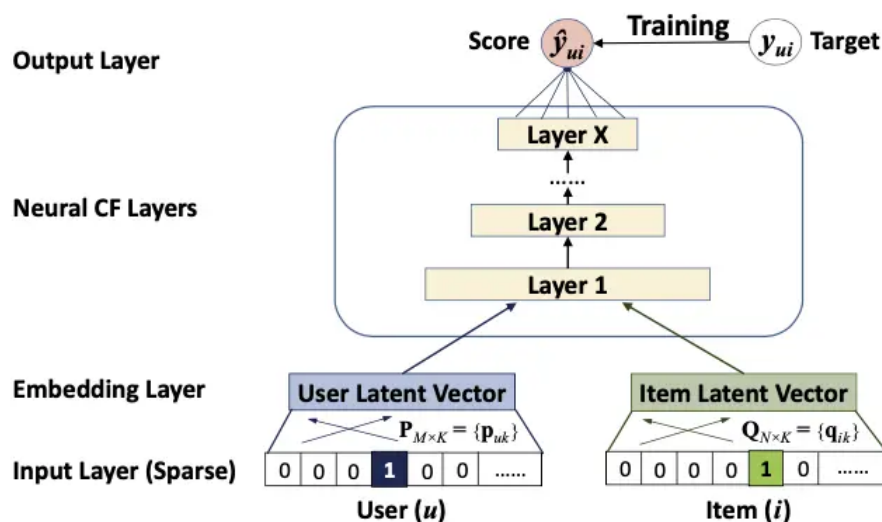


图 2-6 NCF 框架图

最下面的输入层接受用户 id 和物品 id，然后通过输入层将两个 id 转化为 one-hot 编码的二值化稀疏向量。one-hot 编码可以将离散的用户 id 和物品 id 转化为向量表示，保持不同 id 之间特征的独立性。然后用户和物品的二值化稀疏向量会进入 Embedding 层，该层会将稀疏的二值化向量转化为低维的稠密向量。然后这些向量会通过广义矩阵分解(GMF, Generalized Matrix Factorization)或者多层感知机(MLP, Multi-Layer perceptron)网络层进行前向传播计算，然后将两个层输出的张量进行拼接输入 NeuMF 层(Neural Matrix Factorization)，再次进行前向传播计算。这些网络层会进一步提取和组合用户和物品特征之间的复杂交互关系。在训练过程中，模型会根据某种目标函数计算损失，并使用 SDG 梯度下降法对模型参数进行优化更新，包括 Embedding 层的权重参数也会在反向传播过程中更新，使 Embedding 能够更好地表示用户和物品之间的潜在联系。

其实 GMF 层就是求特征向量的乘积，定义层的映射函数如公式(2-4)所示：

$$\Phi_1(p_u, q_i) = p_u \odot q_i \quad \text{式(2-4)}$$

其中 p_u 和 q_i 分别代表用户和物品的潜在特征向量。 \odot 表示将向量元素逐一相乘，可得到输

出层，如公式(2-5)所示：

$$\hat{y}_{ui} = a_{out}(h^T(p_u \odot q_i)) \quad \text{式(2-5)}$$

其中 h 是输出层权重，是在模型在训练过程中进行梯度下降和反向传播的得到的。 a_{out} 是激活函数，采用 sigmoid 函数，用于将模型对用户和物品的推荐程度控制在(0,1)区间内。由此可以得到 GMF 模型的框架图如图 2-7 所示：

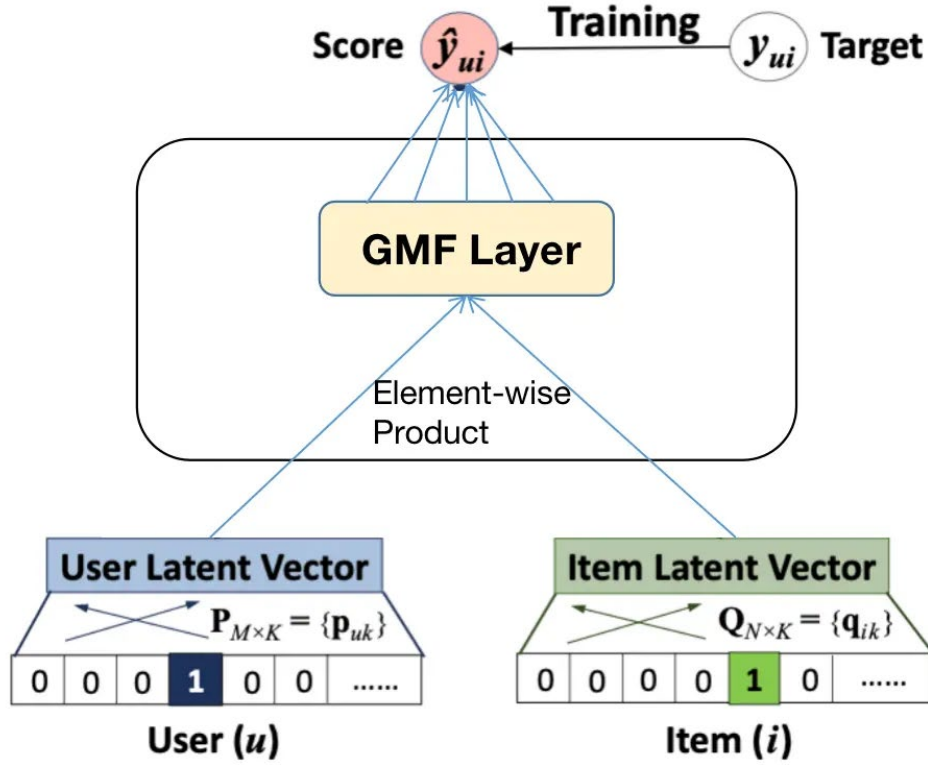


图 2-7 GMF 框架图

在 NCF 模型中使用了两种方法对用户和物品进行建模。只依靠 GMF 学习用户和物品之间的线性关系是不足够的，还需要用 MLP(多层感知机)来捕捉更加复杂的非线性交互模式，增强模型的表达能力。

NCF 中的 MLP 层的数学形式如公式(2-6)所示：

$$\begin{aligned} z_1 &= \Phi_1(p_u, q_i), \\ \Phi_2(z_1) &= a_2(W_2^T z_1 + b_2), \\ &\dots \dots \\ \Phi_L(z_{L-1}) &= a_L(W_L^T z_{L-1} + b_L), \\ \hat{y}_{ui} &= \sigma(h^T \Phi_L(z_{L-1})) \end{aligned} \quad \text{式(2-6)}$$

其中 W_L 表示权重矩阵， b_L 表示偏置矩阵， a_L 表示激活函数。对于感知机层的激活函数一般使用 ReLU(Rectified Linear Unit)^[17]。相比于一些饱和性激活函数(如 sigmoid 和 tanh)，ReLU 函数在正区间上不会饱和，能够更好地缓解梯度消失问题，不会导致过拟合。MLP 的结构

如图 2-8 所示:

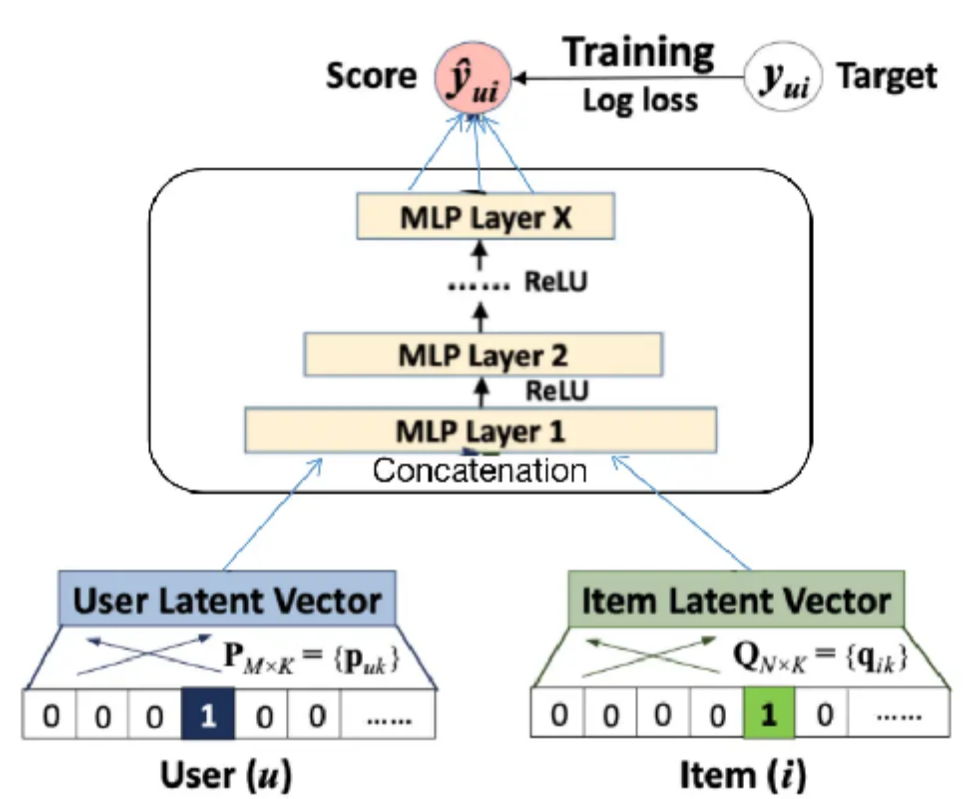


图 2-8 MLP 框架图

NeuMF 层是将 CMF 和 MLP 的张量进行拼接操作，由 GMF 和 MLP 可得出 NCF 的数学公式如公式(2-7)所示：

$$\begin{aligned}\Phi^{GMF} &= p_u \odot q_i, \\ \Phi^{MLP} &= a_L(W_L^T z_{L-1} + b_L), \\ \hat{y}_{ui} &= \sigma\left(h^T \begin{bmatrix} \Phi^{GMF} \\ \Phi^{MLP} \end{bmatrix}\right)\end{aligned}\quad \text{式(2-7)}$$

其中 Φ^{GMF} 是 GMF 层的矩阵分解输出， Φ^{MLP} 是 MLP 层的多层感知机输出。NeuMF 层将两层的输出张量进行拼接之后通过 sigmoid 函数输出物品 i 对用户 u 的推荐值 \hat{y}_{ui} 。

2.3 机器学习相关知识

前面两个小节都是介绍选用模型及其功能，本小节将重点讲述机器学习训练过程中的相关知识。

模型的训练需要定义损失函数和超参数。损失函数是表示一个模型的预测与真实的距离。一个好的损失函数能使模型收敛的更快，如果损失函数使用的不当会导致模型收敛过慢甚至无法收敛。超参数的定义往往会决定模型的性能。比如，之前提到的 LMF 模型的 K (隐语义)就是个超参数，如果 K 设置的过小会导致模型无法捕获数据中的全部特征。然后当 K 设置过大会导致出现过拟合现象，即训练集上准确率很高，但是测试集上的准确率

变化不大，甚至变低，这就是过拟合导致的。相当于学生死记硬背了所有的模拟考题，然后将答案写在正式考试上一样。同时，学习率的设置也很重要，如果学习率过小可能导致梯度弥散，即模型始终无法收敛。而学习率过大可能导致梯度震荡，即始终在最优解周围徘徊而无法真正的去到最优解。

2.3.1 损失函数的选择

真实世界的的数据大致分为离散和连续两类，这两种不同的数据类型大致可以分为：分类问题和回归问题。对于这两类不同的问题需要使用两种不同的损失函数求预测值和真实值之间的差距。

对于分类问题，一般使用交叉熵^[18](CE, Cross Entropy)作为损失函数。熵的作用是判断一个系统的不确定性或者“无序性”。熵越大，则意味着系统越“无序”或者不确定性越大。信息熵是熵在信息论中的应用。信息熵越大，意味着信息源越“不确定”或者随机性越强，每个符号携带的信息量也就越大。在分类问题中，标签通常是 one-hot 独热编码的向量，用于表示真实值。而分类模型预测最后一层是 softmax 函数，将结果转化为和为 1 的概率分布。如果 softmax 输出的值中，各个分类的概率越平均，则说明输出的越“无序”，即熵越大。此时的交叉熵损失值也就越大。在训练过程中，模型会通过梯度下降和反向传播不断使更新权重，使得输出层中 softmax 函数的输的概率分布更加趋近于 one-hot 独热编码的标签。交叉熵的数学公式如公式(2-8)所示：

$$Loss = - \sum_{i=0}^{C-1} y_i \log(p_i) \quad \text{式(2-8)}$$

其中 $p = [p_0, \dots, p_{C-1}]$ 是一个 softmax 函数输出的概率分布， $y = [y_0, \dots, y_{C-1}]$ 是样本真实标签的 one-hot 形式。

对于回归问题，一般使用均方误差^[19](MSE, Mean Squared Error)作为损失函数。回归问题中，真实值和预测值可能是两个数，也有可能是两个向量或者矩阵。对于真实值和预测值的损失函数如公式(2-9)所示：

$$loss = \sum_i^C (y_i - \hat{y}_i)^2 \quad \text{式(2-9)}$$

其中 C 是所有数据的数量， y_i 是真实值， \hat{y}_i 是预测值。如果预测值与真实值的偏差越多，则损失越大。模型想要降低 loss 只能使得 \hat{y}_i 最接近 y_i 。

2.3.2 超参数的设置

超参数的设置往往能直接影响模型的性能，需要经过多次实验调整出来最优的超参数。对于学习率有梯度弥散和梯度震荡的问题，Adam 学习率优化器可以动态的调整学习率，从而解决这一问题。

Adam 是自适应学习率^[20]，相比传统地将学习率设为固定值，Adam 算法能够自适应

地调整每个参数的学习率。如果初始学习率过大，会导致模型在最后无法收敛，梯度下降的过程中反复震荡而无法得到最优解；而初始学习率过小会导致模型收敛过慢甚至无法下降到最优解。

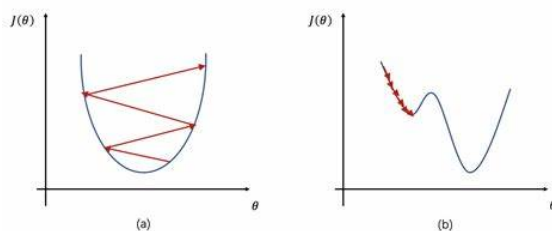


图 2-9 学习率设置过大和过小对梯度下降的影响

如图 2-9 所示，学习率过大会导致梯度反复震荡，而过小会导致梯度弥散，可能会停在一个局部最优解而不是全局最优解。相比之下 Adam 具有动量机制，通过累积历史梯度信息来平滑更新方向。相当于物理第一定律：惯性。人在滑雪过程中，滑到第一个低谷的时候不是立刻停下来，而是随着惯性继续往前滑行一段距离，这样就能找到第二个可能存在的低谷。Adam 保留历史梯度，在下降到最低点的时候会利用历史梯度继续向前，以便进一步探索前方是否存在更优解。

模型的层维度过小可能导致无法充分捕获特征，而层维度过大有可能导致过拟合现象。为了解决这一现象通常有两种解决办法。一种是丢弃(Dropout)在训练过程中会丢弃一些神经元的权重，用于在训练过程中减少神经网络中神经元之间的依赖关系，以防发生过拟合现象。Dropout 的参数设置是丢弃百分比。

对于过拟合现象还有另一种解决办法是使用 L2 正则化项^[21]，也称为权重衰减(weight decay)。该方法通过在损失函数中添加一个正则化项，以限制模型的权重参数的大小。

比如对回归问题使用 L2 正则化项，如公式(2-10)所示：

$$loss = \sum_i^C (y_i - \hat{y}_i)^2 + \lambda |y_i|^2 \quad \text{式(2-10)}$$

L2 正则化项的作用是通过惩罚较大的权重值，促使模型倾向于使用较小的权重，从而防止模型过度拟合训练数据。在公式(2-10)中，当模型的权重变得较大时，正则化项的值也会增加，即 $\lambda |y_i|^2$ 增加，实现了防止过拟合的效果，从而在总体损失中引入一个惩罚项。通过调整正则化系数 λ 的大小，可以控制正则化项的权重，以平衡正则化和训练数据拟合之间的关系。

第3章 移动应用与后端设计

3.1 系统设计概述

3.1.1 系统概述

该系统旨在利用自然语言处理技术实现医疗对话，利用机器学习算法分析用户的个人健康数据、病历资料等信息，为用户提供个性化的医疗服务和健康管理方案。主要功能包括：用户健康数据分析、病症诊断与养生推荐、用药提醒、医疗问诊等。

3.1.2 系统功能模块

主要模块分为：首页推荐模块、AI 对话问诊模块、个人健康预测模块、用药提醒模块和用户个人信息模块。

首页推荐模块：利用 NCF、知识图谱图路径相似度等推荐算法，根据用户的历史行为、健康数据和偏好，为用户推荐相关的医疗资讯、健康文章、医生咨询等内容。

AI 对话问诊模块：利用 Google 预训练的 BERT-Base 模型和 Text-CNN 模型识别用户意图，并通过知识图谱搜索相应答案，为用户提供准确的医疗咨询和诊断建议。

个人健康预测模块：基于用户的个人健康数据和医学统计数据，利用 Tensorflow 训练线性回归预测模型，提供个人健康状态的预测和评估。

用药提醒模块：通过个人用药数据管理和安卓定时广播提醒功能，帮助用户按照医嘱定时用药，避免耽误医疗疗程。

个人信息模块：通过安卓 SQLite 数据库和 Spring Boot 框架实现对用户的本地数据和云端数据的管理。

后台管理模块：通过 HTML，CSS，JavaScript 实现后台管理系统，帮助管理员对数据库进行管理。

系统功能框架图如图 3-1 所示：

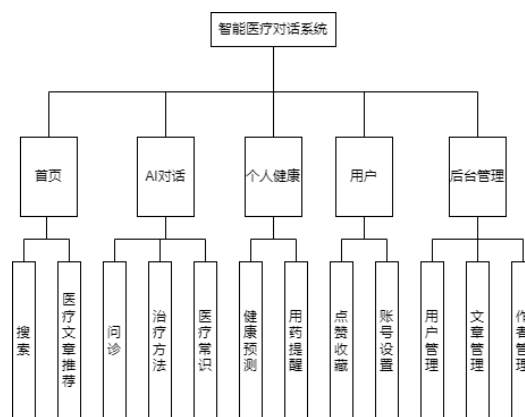


图 3-1 系统功能架构图

3.2 数据库分析与设计

3.2.1 数据库分析

根据上一小节对系统功能的描述，可以分析出各个模块需要存储的数据实体有以下几个：

1. 用户登录模块和个人信息模块需要保存用户的个人信息。
2. 首页推荐模块需要展示文章的标题，内容，图片等数据。
3. AI 对话问诊模块需要保存用户与 AI 的聊天记录。
4. 个人健康预测模块需要保存用户的个人健康信息。
5. 用药提醒模块需要保存提醒用药的药品，时间等信息。
6. 用户信息模块需要保存用户收藏的文章信息。
7. 用于推荐系统中文章某一关键词信息。

其中：用户账号登录信息要分别保存在云端和本地。文章数据，用户收藏文章数据信息只需要保存在云端。AI 问诊产生的聊天记录，用药提醒的数据只需要保存在本地。

根据对上述功能的思考与分析，需要建立云端 MySQL 数据库和本地 SQLite 数据库。在确定好数据库的实体之后，需要设计数据库的结构。

3.2.2 数据库概念设计

为了将现实世界中的实体转化为数据库的结构，需要进行概念设计。经过分析，现实中的实体有：用户，文章，作者，健康数据，聊天记录和用药提醒数据。为了实现推荐算法还需要将用于表设计文章特征和作为用户感兴趣程度的文章关键词为一个实体。云端数据库的实体可分为：用户，文章，作者，关键词，健康。本地数据库的实体可以分为：健康数据，聊天记录，用药提醒。

E-R 图^[22](实体-联系)是一种常用的数据库设计工具，用于可视化和描述实体之间的关系。为了规范化数据库以及减少数据冗余和提高数据一致性需要绘制数据库的 E-R 图。云端 MySQL 数据库 E-R 图如图 3-2 所示：

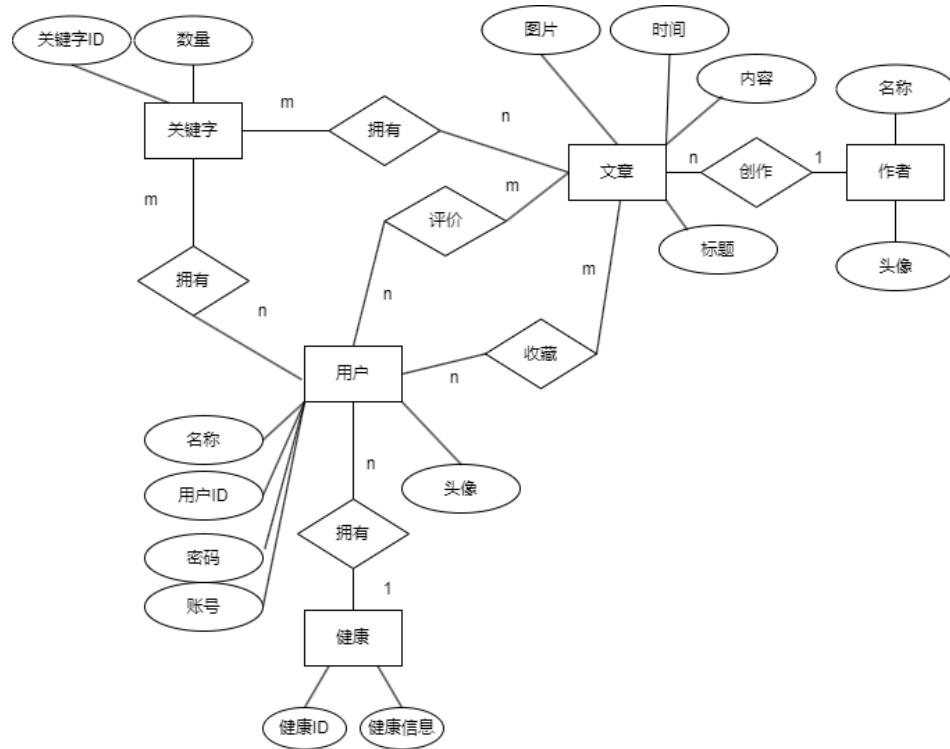


图 3-2 MySQL 云端数据库 E-R 图

APP 本地 SQLite 数据库 E-R 图如图 3-3 所示：

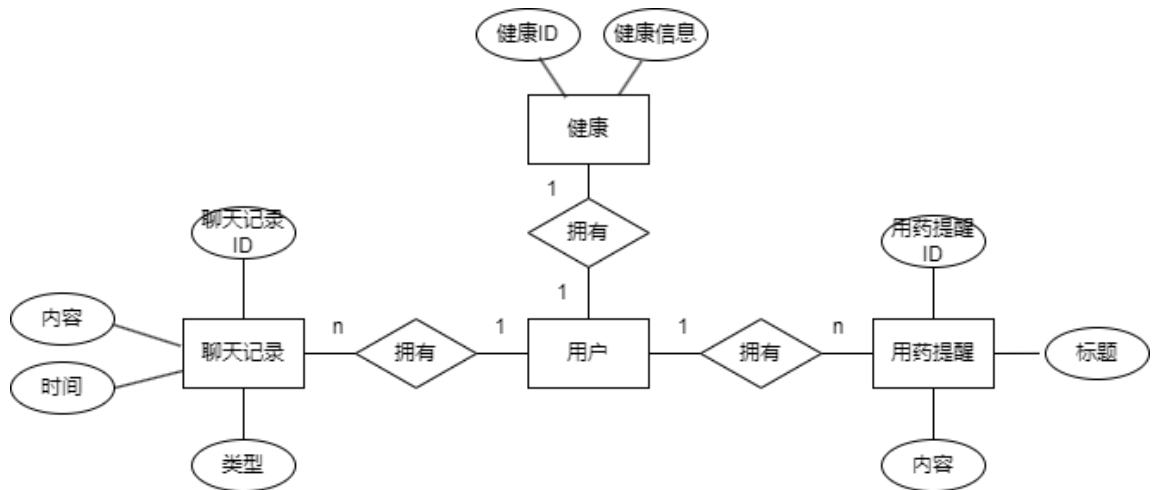


图 3-3 SQLite 本地数据库 E-R 图

3.2.3 数据库逻辑设计

在进行了概念设计之后需要对数据库进行逻辑设计。逻辑设计的目的是将概念模型转换为数据库管理系统(DBMS, Database Management System)可以理解和实现的逻辑模型。根据上一小节中 E-R 图中实体之间的数量关系可以发现：

1. 实体用户和文章之间存在关系收藏和评价，对应的数量对应关系都是 $m:n$ 。
2. 实体用户和关键词之间的数量关系是 $m:n$ 。
3. 实体文章和关键词之间的数量关系是 $m:n$ 。

根据数据库的减少数据冗余和提高数据一致性原则，需要额外设计关联表：用户-文章收藏，用户-文章评价，用户-关键词和文章-关键词这 4 个表。通过引入关联表，可以避免在原始实体表中存储冗余的数据，并且可以更好地管理和查询多对多关系。

为了确定数据的存储结构，还需要分析数据的字段类型。以下分析部分关键实体的字段设计。

对于 article 实体，为了进行基本的数据操作，需要设计 ID 对 article 实体进行管理。设置非空，并选用 MySQL 数据库的 ID 自增。文章标题长度不等，需要设计为存储字符串的 varchar 类型，最大长度限制设为 255，并且不能为空。对于文章内容，需要能接受长字符，所以需要将字段类型设置为 text。发布时间是文章发布的关键数据，故设为非空，并且使用 datetime 类型存储。作者 ID 是选用 author 表中的 ID 作为外键，非空，在作者更新的时候文章的作者也需要更新，作者删除的时候，文章的所有内容也需要删除，故设为操作式进行 CASCADE 操作。图片是用来存储文章的封面，大概为 100KB~1MB 不等，在 MySQL 数据库中，存储图片的类型有 TINYBLOB、BLOB、MEDIUMBLOB 和 LONGBLOB，其中 MEDIUMBLOB 最大存储容量为 16,777,215 字节，兼容了存储功能和存储效益，最适合存储目标图片数据，故选择用 MEDIUMBLOB 类型作为图片的字段类型。

3.2.4 数据库物理设计

在将数据存储到数据库之前需要进行物理设计，物理设计阶段是将逻辑模型映射到具体的存储结构和实现方式的过程。主要目标是设计合理的表结构，文件组织方式等。根据上一小节分析，可以设计出以下表：

对于云端数据库，数据表主要有下述几个：

article 表结构如表 3-1 所示，在用户打开推荐界面的时候，会根据推荐算法获取需要推荐的文章 ID 列表，后端会根据 article_id 来 article 表中调取相应的文章。

表 3-1 article 表结构

表名	article				
备注	字段名	字段类型	可否为空	主键	默认
文章编号	id	int(11)	否	是	无
文章标题	title	varchar(255)	否	否	null
文章内容	content	text	是	否	null
发布时间	time	datetime	否	否	null
作者 ID	author_id	int(11)	否	否	null
文章图片	picture	mediumblob	是	否	null

user 表结构如表 3-2 所示，在用户在注册的时候会将用户信息发送到后端，后端会根据信息创建用户实例然后将数据插入到 **user** 表中。用户在登录时，后端会调取 **user** 表中的数据与用户发来的数据进行比对，实现登录功能。

表 3-2 user 表结构

表名	user				
备注	字段名	字段类型	可否为空	主键	默认
用户编号	id	int(11)	否	是	无
用户名称	name	varchar(255)	是	否	null
用户账号	sign_account	varchar(255)	否	否	null
用户密码	password	varchar(255)	否	否	null
用户图片	picture	mediumblob	是	否	null

author 表结构如表 3-3 所示，在系统中，作者和文章是一对多的关系，用户可以通过某一作者 ID 找到他的全部文章信息。

表 3-3 author 表结构

表名	author				
备注	字段名	字段类型	可否为空	主键	默认
作者编号	id	int(11)	否	是	无
作者名称	name	varchar(255)	否	否	null
作者图片	face_picture	mediumblob	是	否	null

entity 表结构如表 3-4 所示，在系统中，文章中都包含数个知识图谱中的关键词，这些关键词会作为推荐算法中的图路径相似度计算实体，用来推荐给用户相应的文章。

表 3-4 entity 表结构

表名	entity				
备注	字段名	字段类型	可否为空	主键	默认
关键词编号	id	int(11)	否	是	无
关键词名称	name	varchar(255)	是	否	null
关键词数量	number	int(11)	是	否	null

health 表结构如表 3-5 所示，在系统中，**health** 表用来保存用户的健康信息，其中 **user_id** 是外键，引用的是 **user** 表的 ID。

表 3-5 health 表结构

表名	health				
备注	字段名	字段类型	可否为空	主键	默认
健康信息编号	id	int(11)	否	是	无
用户 ID	user_id	int(11)	否	否	null
健康信息	health	varchar(255)	否	否	null

article_entity 表结构如表 3-6 所示，此表用来记录某一文章中的关键词信息，以便用于推荐系统。其中 article_id 是外键，引用的是 article 表的 ID。entity_id 是外键，引用的是 entity 表的 ID。

表 3-6 article_entity 表结构

表名	article_entity				
备注	字段名	字段类型	可否为空	主键	默认
编号 ID	id	int(11)	否	是	无
文章 ID	article_id	int(11)	否	否	null
关键词 ID	entity_id	int(11)	否	否	null

user_article_collection 表结构如表 3-7 所示，此表用来记录某用户的收藏的文章。其中 article_id 是外键，引用的是 article 表的 ID。user_id 是外键，引用的是 user 表的 ID。

表 3-7 user_article_collection 表结构

表名	user_article_collection				
备注	字段名	字段类型	可否为空	主键	默认
编号 ID	id	int(11)	否	是	无
用户 ID	user_id	int(11)	否	否	null
文章 ID	article_id	int(11)	否	否	null

user_article_evaluate 表结构如表 3-8 所示，此表用来记录用户对某一文章的评分。其中 article_id 是 article 表的外键，user_id 是外键，引用的是 user 表的 ID。

表 3-8 user_article_evaluate 表结构

表名	user_article_evaluate				
备注	字段名	字段类型	可否为空	主键	默认
编号 ID	id	int(11)	否	是	无
评分	score	int(11)	是	否	null

续表 3-8 user_article_evaluate 表结构

备注	字段名	字段类型	可否为空	主键	默认
编号 ID	id	int(11)	否	是	无
评分	score	int(11)	是	否	null

user_entity 表结构如表 3-9 所示，此表用来记录用户对某一关键词的感兴趣程度。user_id 是外键，引用的是 user 表的 ID。entity_id 是外键，引用的是 entity 表的 ID。

表 3-9 user_entity 表结构

表名	user_entity				
备注	字段名	字段类型	可否为空	主键	默认
编号 ID	id	int(11)	否	是	无
用户 ID	user_id	int(11)	否	否	null
关键词 ID	entity_id	int(11)	否	否	null
感兴趣程度	level	int(11)	是	否	null

对于本地 SQLite 数据库中表设计如下所示：

user 表结构如表 3-10 所示，为了使本地用户 ID 与云端用户 ID 一一对应，SQLite 的用户表不采用 id 作为用户 ID。表中的 id 是本表的表编号，user_id 是用户编号。save_account 和 save_password 用来记录用户是否选择保存账号在本地。

表 3-10 user 表结构

表名	user				
备注	字段名	字段类型	可否为空	主键	默认
表编号	id	INTEGER	否	是	无
用户编号	user_id	INTEGER	否	否	null
用户名称	name	TEXT	是	否	null
用户账号	account	TEXT	否	否	null
用户密码	password	TEXT	否	否	null
用户图片	picture	blob	是	否	null
保存账号	save_account	INTEGER	否	否	0
保存密码	save_password	INTEGER	否	否	0

3.3 软件分层与设计模式

在完成功能设计和数据库设计之后需要对软件模块进行分工与设计，根据上面小节对系统功能的描述，可以分析出系统主要分为三大模块：APP 前端模块，Spring Boot 后端数据业务模块，Flask 模型调用模块。

为了使得系统更易于扩展、调试和重构，同时提高代码的可读性和可维护性。需要遵循“高内聚，低耦合”原则。高内聚(High Cohesion)是指一个模块内的元素紧密相关共同完成任务；低耦合(Low Coupling)是指一个模块间相互依赖关系弱，耦合度较低。

为了实现上述目标，需要对系统中的各个模块规定设计模式。

前端使用的设计模式大致为：模型-视图-视图模型(MVC，Model-View-Controller)，模型-视图-表示器模型 (MVP，Model-View-Presenter)和模型-视图-视图模型(MVVM，Model-View-ViewModel)^[23]。MVC 主要分为：模型、视图和控制器。每个部分负责不同的职责，可以实现模块之间的相对独立性。但是在 MVC 模式中，需要视图与模型之间的直接交流，这可能导致视图与模型之间的紧耦合。这种紧耦合可能会导致视图对模型的过度依赖，使得视图难以独立变化和重用。结构如图 3-4 所示：

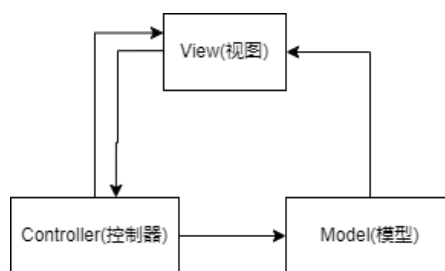


图 3-4 MVC 设计模式

为了解决这一问题提出了 MVP。通过引入 Presenter 作为中介者，视图与模型之间的交互通过 Presenter 进行。这种分离使得视图和模型可以独立变化，从而提高了模块的内聚性和低耦合性。结构如图 3-5 所示：

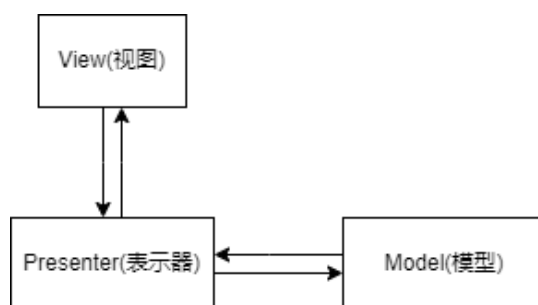


图 3-5 MVP 设计模式

但是 MVP 中视图跟模型的交流全部需要经过中间层，这使得表示层承担了过多的责任，可能会变得过于臃肿。为了解决这一问题提出了 MVVM 模式。

在 MVVM 中采用了数据绑定机制，用于在视图和视图模型之间建立双向绑定关系，这一自动更新机制减少了视图和视图模型之间的交互代码，从而减轻了中间层的臃肿。结

构如图 3-6 所示:

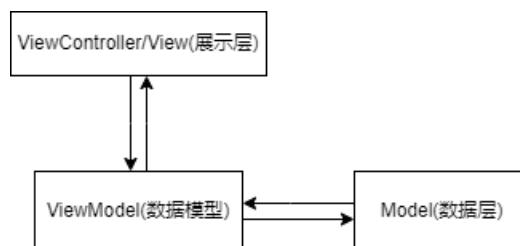


图 3-6 MVVM 设计模式

故本系统采用 MVVM 设计模式, 并通过 Google 提供的视图绑定实现各层的设计:

表示层包括: 首页推荐模块的文章自定义 View, 对话模块的消息自定义 View 等界面。

数据模型层包括: 用户点击事件设计, 交互数据等业务逻辑。

数据层包括: 用户健康数据, 用户聊天记录等数据库数据。

后端的设计模式基本为领域驱动设计(DDD, Domain-Driven Design)^[24], 大致分为基础层, 包括数据库等; 领域层, 包括 Domain 实体类和用于对实体类进行操作的 Dao 数据接口; 应用层, 包括 Service, 是对一个或多个实体的业务逻辑操作; 用户接口层, 包括用户 Controller 网络请求等。

对于后端数据业务模块和模型调用模块都需要用到此设计模式。根据上一节对数据库的分析, 在后端数据业务模块中, 需要设计出:

文章, 作者, 用户, 关键词, 文章-关键词, 用户-文章-收藏, 用户-文章-评价和用户-关键词的 domain 实体, dao 操作函数, service 服务以及 controller 应用接口。

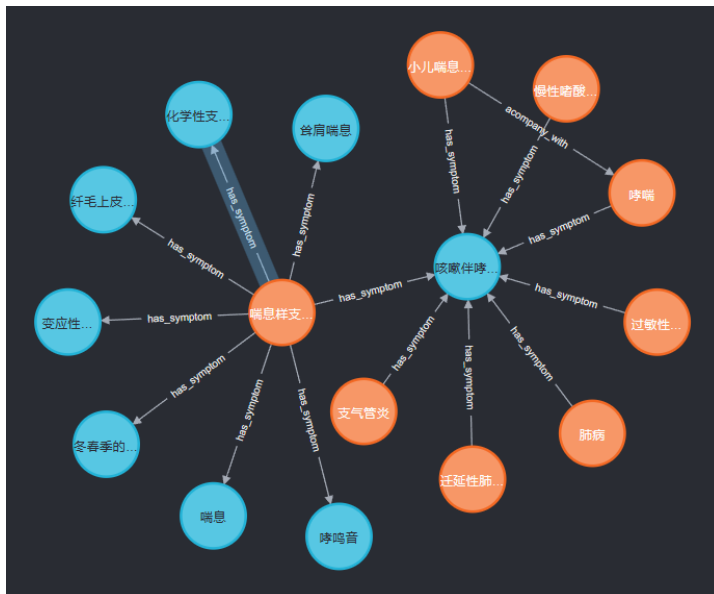
对于模型调用模块只需要使用使用用户-文章-评价和用户-关键词数据进行模型的 domain, dao 进行获取数据用于模型训练。

3.4 构建知识图谱

本系统涉及推荐系统和对话系统, 推荐系统中涉及实体图路径相似度计算, 并利用相似度作为权值来推荐。为了构建实体的关系图以及实现计算实体在关系图中的距离, 需要进行知识图谱^[25]的构建。其次本系统的对话系统涉及大量的关系查询, 需要构建知识图谱。在如 MySQL 的关系型数据库中, 数据库局限于二维表格, 难以捕捉复杂的实体联系。如果要查询一个实体的所有属性, MySQL 数据库则需要执行大量的 Join 操作, 数据库在处理大规模数据时性能会下降。相比之下, 知识图谱这种数据库可以更自然的表达实体间的关系, 并且在查询实体属性的时候可以通过基于图遍历的方式来查询, 比 SQL 的大量 Join 操作性能更好。

在医疗领域, 涉及大量复杂的实体(如疾病、症状、药物等)及其复杂的相互关系, 需要更直观的观察实体间的关系。在这样的前提下相比使用关系型数据库, 使用知识图谱这种图形数据库是更好的选择。

构建知识图谱的数据集是采用 Github 的开源数据, 其中包括症状、检查、药品等 8 个



3.5 登录与注册模块设计

12:58

🔊 📶 🔒

智医

Login to continue

Email

Password

☐ 记住账号

☐ 记住密码

登录


创建新的账户

游客模式

12:58

🔊 📶 🔒

Create New Account



Name

Email

Password

Confirm Password

注册

去登录

23

如图 3-8 所示, 在进入 APP 之后, 用户可以选择登录进入或者游客访问, 已经注册的用户可以通过输入账号和密码直接登录或者游客模式进入。但是游客模式无法提供数据云端存储和个性化的推荐。而未进行注册的用户可以通过点击创建新账号进行注册, 在注册的时候也可以选择自己喜欢的头像。

在用户输入完账号信息之后, APP 会通过文本框读取用户的账号和密码等数据, 通过 SQLite 记录账号密码在本地。当用户点击登录按钮之后前端会将账号密码信息转化为 JSON 格式以 POST 请求方式发送到后端。如果后端 MySQL 数据库存在此账号并且账号密码一一对应, 就返回 flag 为 True 的 JSON 格式相应数据, 若不存在或不对应则返回 False。

登录时的前端请求代码如下:

```
private void CheckSignIn(){
    String account = this.binding.SignInInputEmail.getText().toString();
    String password = this.binding.SignInInputPassword.getText().toString();
    if(account.equals("") || password.equals("")){
        return;
    }
    UserRequestType userRequestType = new UserRequestType(account,password);
    userData = userRequestType;
    R_dataType rDataType = new R_dataType(userRequestType);
    String jsonData = R_Util.R_JsonUtils.toJson(rDataType);
    RequestUtils post_requestUtils = new RequestUtils(5000, requestClass, "POST",
    UriUtil.GetSignInUrl(), jsonData);
    post_requestUtils.callback = this;
    post_requestUtils.StartThread();
}
@Override
public void onSuccess(String callbackClass, R_dataType rData) {
    if(callbackClass.equals(requestClass)){
        UserBackType userBackType = new UserBackType();
        userBackType.getByRData(rData);
        if(userBackType.imageBytes != null){
            User.User_image = userBackType.imageBytes;
        }
        User.Name = userBackType.name;
        User.user_id = userBackType.id;
        SignInAccount(userBackType.id);
    }
}
```

前端的数据会通过异步请求发送到后端, 当获取到数据之后, 会通过回调函数来对结果进行处理。

后端处理请求代码如下:

```
@Autowired
UserDao userDao;

@Autowired
```



```

UserService userService;

@PostMapping("/signIn")
public R_dataType SignIn(@RequestBody R_dataType rDataType) {
    boolean flag = rDataType.getFlag();
    if(!flag){
        return new R_dataType(false);
    }
    Object data = rDataType.getData();
    UserRequestType userRequest = R_Util.R_JsonUtils.parseData(data,UserRequestType.class);
    UserBackType userBackType;
    if(userDao.signInNum(userRequest.account,userRequest.password) > 0){
        int user_id = userDao.signInID(userRequest.account,userRequest.password);
        User selectUser = userService.getById(user_id);
        byte[] image = selectUser.getPicture();
        if(image != null){
            userBackType = new UserBackType(user_id,image,selectUser.getName());
        }
        else {
            userBackType = new UserBackType(user_id,null,selectUser.getName());
        }
    }
    else {
        userBackType = new UserBackType(0,null,"null");
    }
    return new R_dataType(true,userBackType);
}

```

后端代码通过定义@RequestMapping("/sign")和@PostMapping("/signIn")来定义处理方法的路由，并通过 UserDao 和 UserService 定义的对象来分别查询和获取 user 对象，最后将数据返回前端。

3.6 推荐模块设计

如图 3-9 和图 3-10 所示，系统为了提供每个用户个性化的推荐，在推荐页面中会根据不同用户的喜好来推荐不同的文章。同时用户也可以通过搜索来寻找自己喜欢的文章。



图 3-9 推荐界面



图 3-10 文章界面

在启动后端服务器之后，用户推荐 Service 会自动执行@Scheduled 设置的定时方法。后端会自动检测当前的用户在线情况，优先为在线用户获取需要推荐的文章列表。

当用户以游客身份进入，会请求后端的 ResponseNotLogged()方法，将当前的热门文章推荐给用户。

当用户登录账号进入时，后端会根据用户 ID 获取推荐缓存队列中的推荐文章列表并将文章实体列表返回给前端。

热门文章列表和推荐列表需要 Spring Boot 服务器跨域请求负责推荐算法的 Flask 服务器，其中跨域问题通过 CORS 来解决。推荐算法采用的是 NCF 模型和知识图谱实体距离查询。其中知识图谱实体距离查询是利用知识图谱的 CQL 语句查询实体间的距离，代码如下：

```
def calculate_path_length(start_entity, end_entity):
    if start_entity == end_entity:
        return 0
    query = f'MATCH path=shortestPath((start)-[*]-(end)) WHERE start.name='{start_entity}' AND end.name='{end_entity}' RETURN length(path) AS path_length'
    result = graph.run(query).evaluate()
    return result if result is not None else float('inf')
```

如果实体是自己则设置距离为 0，如果没有实体则设置距离为无穷大。距离越短则实体相关性越高，推荐权重越高。

NCF 模型训练需要初始“用户-物品”矩阵。系统中预先设定的用户及其感兴趣的实体，使用知识图谱实体距离查询生成用户对其他相关实体的评价表。然后通过 AC 自动机导出文章中存在的实体并存入数据库。并通过用户对实体的评价表生成对实体对应的文章的评价表，最后将评价表导入数据库形成了初始“用户-物品”矩阵。然后 NCF 模型就利用初始的“用户-物品”矩阵进行训练，最后得到可以输出某一用户对某一物品感兴趣程度的推荐模型。

在用户注册之后，用户是没有任何行为的，直接使用 NCF 模型对用户喜好进行预测是

存在空矩阵问题的。为了解决推荐系统的冷启动问题，在用户登录之后，用户推荐 Service 会先获取热门文章列表并计算用户的冷因子。热门文章的获取方式为：将所有用户的感兴趣文章评分叠加起来，然后获取前 k 个文章。

冷因子公式如公式(3-1)所示：

$$coldFactor = \sigma \left(\sum_{k=1}^K L_{ue} - Cold_{constant} \right) \quad \text{式(3-1)}$$

其中 L_{ue} 是用户 u 对第 e 个实体的感兴趣程度， $Cold_{constant}$ 是冷因子常数，是由初始“用户-物品”矩阵中用户对实体的。K 是用户 u 感兴趣的实体数量。 σ 是 sigmoid 函数，负责将数据控制在(0,1)范围内，以便表示当前系统对用户的了解程度，代码如下：

```
public final int ColdnessConstant = 19;
public double Calculate_userColdnessFactor(int user_id, User_EntityDao userEntityDao){
    Integer allEntitiesLevel = userEntityDao.sumUserEntityLevelsById(user_id);
    if (allEntitiesLevel == null){
        allEntitiesLevel = 0;
    }
    int basicColdnessVariable = allEntitiesLevel - this.ColdnessConstant;
    return sigmoid(basicColdnessVariable);
}
public static double sigmoid(double x) {
    return 1 / (1 + Math.exp(-x));
}
```

在得到了用户冷因子之后，系统会根据冷因子计算使用热门文章和个性化推荐文章的比例。其中对于新用户有 $coldFactor=0$ ，所以不进行个性化推荐，只推荐热门文章，通过对热门文章的行为来记录用户对文章的评分以及对实体的喜好程度。

用户点击文章之后会跳转到文章 Activity 界面然后记录点击进入的时间。当用户点击了喜欢、收藏或者是不喜欢这些显性评分的时候，用户的这些评分数据会直接传入后端进行保存。当用户没有显性行为时，在返回推荐界面的时候会再次记录返回时。然后计算用户的阅读时间。通过阅读时间来判断用户对文章的感兴趣程度。并将隐性评分数据发送到后端进行保存。之后在 NCF 模型训练的时候会将之前保存在数据库的数据导出用来训练。

在用户下次登录的时候，系统会根据用户之前的行为数据进行用户冷因子计算，NCF 模型推荐计算以及知识图谱实体距离推荐计算。然后根据用户冷因子的比例将热门文章和个性化文章推荐给用户。

3.7 对话模块设计

当用户想要咨询自身疾病相关的问题的时候，可以与医疗 AI 对话来咨询相关问题。

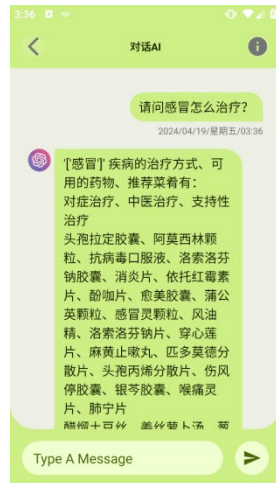


图 3-11 询问治疗方法

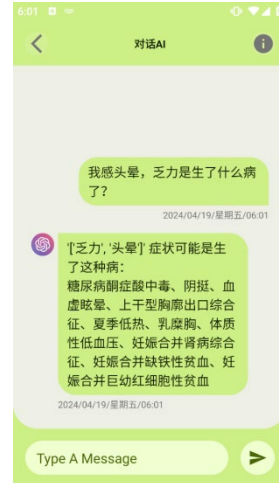


图 3-12 问诊是何种疾病

如图 3-9 所示, 主页界面右下角有一个对话按钮, 点开可以进入到与 AI 对话的界面。如图 3-11 图 3-12 所示, 在主页点开 AI 对话之后, 用户可以询问 AI 医疗相关的问题, 例如, 询问某个疾病的病因是什么? 某疾病如何治疗? 某个症状是生了某种疾病? 等等。

用户输入的文本会通过 EditText 读取数据, 然后封装成 JSON 通过 POST 请求发送到 Flask 后端。在 Flask 后端中, 数据首先会进入 BertIntentModel 进行意图识别, 意图分为定义、病因、预防、临床表现等 15 类。之后将文本中的实体通过 AC 自动机进行提取。然后选择意图相对应的 CQL 语句模板并将识别出来的实体填入。最后将 CQL 语句查询结果返回到前端并通过自定义 view 的 RecyclerView 显示出来。用户重新打开界面时, 前端会自动从 SQLite 中读取聊天记录数据并展示在 View 中。

3.8 健康模块设计

用户能够在健康模块管理自己的个人健康信息、进行医疗预测以及设置用药提醒。

在用户点击健康信息填写之后会出现一个弹窗让用户填写健康信息数据, 如图 3-13 所示。



图 3-13 健康信息填写



图 3-14 健康信息查看

之后用户在弹窗填写信息会被记录到本地数据库中。如图 3-14 所示用户可以在下方查看自己的健康信息。在点击医疗预测之后，用户的数据会被发送到医疗预测模型后端，经过模型预测可以得到用户患有糖尿病和心脏病的概率。

APP 使用静态注册广播，可以实现 APP 在后台的时候检查 SQLite 本地数据库中的用药提醒数据。并通过后台读取时间来检查是否应该提醒用药了。用户可以通过点击用药提醒来设置需要提醒的药物、时间和内容，如图 3-15 所示：



图 3-15 用药提醒填写



图 3-16 用药通知提醒

在系统检测到了提醒的时间之后，会通过通知的形式将需要提醒的数据发送给用户，如图 3-16 所示。

3.9 后台管理模块设计

一、模块设计

本系统可以通过网页来管理后台数据,包括:文章、用户和作者信息。网页通过 HTML、CSS 和 JavaScript 实现。可以在登录界面登录管理换账号来对数据进行管理。

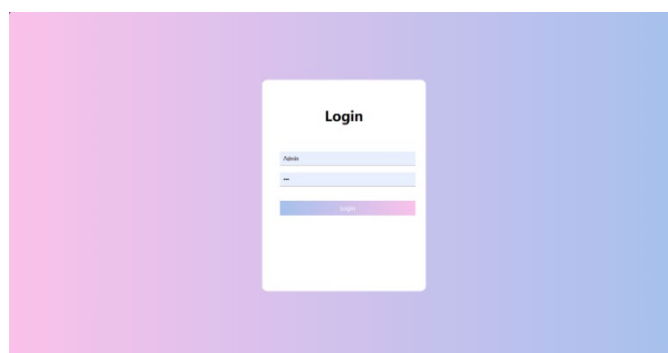


图 3-17 后台管理登录界面

如图 3-17 所示，登录之后会进入到管理界面主页。

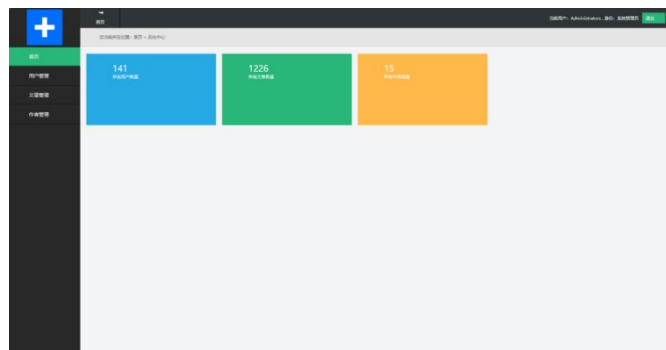


图 3-18 后台 home 界面

如图 3-18 所示，主页侧边框可以对包括用户、文章和作者三种数据进行管理。



图 3-19 用户管理界面

如图 3-19 所示，点开列表中一个数据管理之后，JavaScript 会通过 GET 方法向后端请求数据。标注跨域权限@CrossOrigin(origins = "**")的后端方法会通过 Dao 查询相关数据并返回给前端。前端接收到数据之后会将数据 userList 列表组件中并展示出来。



图 3-20 对数据的增加，删除，修改和查询功能

如图 3-20 所示，管理员可以通过点击更新、删除、查询和添加按钮实现向后端请求相应的方法从而实现对数据的增加，删除，修改和查询功能。

第4章 实验及分析

在第二章中详细介绍了模型设计的理论基础，第三章中详细介绍了本系统所设计的模型用途。作为对该模型的进一步验证，本章将重点探讨模型的结构设计、训练过程以及对实验数据的深入分析，评估所提方法的性能。

4.1 BERT 模型实验与分析

在介绍了系统的对话系统之后，本小节介绍如何实现对话模型，包括构建，训练和分析。

4.1.1 BERT 模型文本分类实验总体流程

本实验的实验过程为：收集医疗问题相关数据集，对数据进行预处理，定义模型参数以及相关系数，构建 BERT 模型，训练模型，最后选择最佳训练结果以及将训练过程可视化。

4.1.2 数据预处理

数据集使用 Github 上的开源数据集。首先利用 pandas 将 ChineseBLUE 的 cvs 文件中的数据读取到 dataset 中。然后通过 bert4keras 的 Tokenizer 将文本数据变为 BERT 模型需要的输入格式。

```
tokenizer = Tokenizer(dict_path)
token_ids, segment_ids = tokenizer.encode(text, maxlen=maxlen)
```

其中 Tokenizer 读取一个词表，并将其作为把文本转化为数字的映射关系。maxlen 限制了文本最大长度，对超过最大长度的文本做裁剪，而对不足最大长度的文本在末尾补上多个[PAD]做 padding 填充操作。保证输入模型的张量形状相同。token_ids 是文本转化为数字表示 id 的序列，segment_ids 是一个由 0 和 1 组成的列表，表示输入文本的分段信息。随后会将这些数据分批传入 BERT 模型。

4.1.3 定义相关参数

在进行训练之前需要定义适当的参数以便训练出最好的结果。BERT-Base 模型的部分参数如下：

```
"directionality": "bidi",
"hidden_size": 768,
"max_position_embeddings": 512,
"num_attention_heads": 12,
"num_hidden_layers": 3,
```

```

"pooler_fc_size": 768,
"pooler_num_attention_heads": 12,
"pooler_num_fc_layers": 3,
"pooler_size_per_head": 128,
"vocab_size": 21128

```

其中 `directionality` 定义为：“`bidi`”，表示模型的方向为双向，即同时考虑左右上下文信息。`hidden_size` 表示隐藏层的维度大小，`num_attention_heads` 是多头注意力数量，`pooler_size_per_head` 是每个注意力头的大小，这些数量过大都可能会导致过拟合，而数量过小可能会导致模型无法捕获到文本特征。

4.1.4 模型构建

在定义完模型参数之后，需要定义模型结构。BERT 模型会将文本转化为 `token` 向量序列，而要将向量序列转化成最终的文本输出需要连接一个输出层对序列进行处理。本系统 BERT 模型使用 Text-CNN 层作为模型的输出层，代码如下：

```

def text_cnn(inputs, kernel_initializer, dropout = 0.2):
    cnn1 = keras.layers.Conv1D(
        256,
        3,
        strides=1,
        padding='same',
        activation='relu',
        kernel_initializer=kernel_initializer
    )(inputs) # shape=[batch_size,maxlen-2,256]
    cnn1 = keras.layers.GlobalMaxPooling1D()(cnn1) # shape=[batch_size,256]
    cnn2 = keras.layers.Conv1D(
        256,
        4,
        strides=1,
        padding='same',
        activation='relu',
        kernel_initializer=kernel_initializer
    )(inputs)
    cnn2 = keras.layers.GlobalMaxPooling1D()(cnn2)
    cnn3 = keras.layers.Conv1D(
        256,
        5,
        strides=1,
        padding='same',
        kernel_initializer=kernel_initializer
    )(inputs)
    cnn3 = keras.layers.GlobalMaxPooling1D()(cnn3)
    output = keras.layers.concatenate(
        [cnn1, cnn2, cnn3],
        axis=-1)
    output = keras.layers.Dropout(dropout)(output)
    return output

```


其中层的输入是 BERT 的 token 序列，形状是(batch_size, maxlen-2768)，batch_size 表示每个训练批次，maxlen 表示输入数据序列的最大长度，因为 BERT 会对首位做 CLS 标记和 SEP 标记，所以需要-2。而 768 则是 BERT 模型的隐藏状态的维度。Text-CNN 层的 kernel_initializer 设置为 bert.initializer 表示使用 BERT 模型的初始化器作为全连接层的权重初始化方法。该 Text-CNN 层使用了三个卷积窗口宽度为 3、4 和 5 的积层。以便于通过不同大小的卷积窗口来捕捉不同尺度的文本特征。在每次卷积之后使用了最大池化层用来对卷积层的输出进行降维和特征提取。最后使用了一个 Dropout 防止过拟合。

BERT 模型则利用 Text-CNN 层来实现文本的分类，代码如下：

```
def build_bert_model(config_path,checkpoint_path,class_nums):
    bert = build_transformer_model(
        config_path=config_path,
        checkpoint_path=checkpoint_path,
        model='bert',
        return_keras_model=False)
    cls_features = keras.layers.Lambda(
        lambda x: x[:, 0],
        name='cls-token'
    )(bert.model.output)
    all_token_embedding = keras.layers.Lambda(
        lambda x: x[:, 1:-1],
        name='all-token'
    )(bert.model.output) # shape=[batch_size,maxlen-2,768]
    cnn_features = text_cnn(
        all_token_embedding,bert.initializer
    )
    concat_features = keras.layers.concatenate(
        [cls_features, cnn_features],
        axis=-1)
    dense = keras.layers.Dense(
        units=512,
        activation='relu',
        kernel_initializer=bert.initializer
    )(concat_features)
    output = keras.layers.Dense(
        units=class_nums,
        activation='softmax',
        kernel_initializer=bert.initializer
    )(dense)
    model = keras.models.Model(bert.model.input, output)
    return model
```

首先配置文件和检查点路径构建 BERT 模型。接下来，通过使用 Lambda 层，提取 BERT 模型的句子的分类标记 CLS。然后，使用 Lambda 层提取 BERT 模型除了 CLS 和 SEP 的所有标记的嵌入表示。这些嵌入表示将作为输入传递给文本卷积操作。使用 Text-CNN 层对所有标记的嵌入表示进行文本卷积操作。通过卷积操作捕获输入文本中的局部特征，然后

将 CLS 特征和文本卷积特征在最后一个维度上进行拼接，得到拼接后的特征。然后，通过一个全连接层将拼接后的特征映射到一个较低维度的特征空间，并使用 ReLU 激活函数进行非线性变换。最后，通过第二个全连接层将上层的输出映射到最终的类别数量，并使用 softmax 激活函数进行概率归一化得到各个类别的概率分布。最终，使用 `keras.models.Model` 将输入和输出定义为模型的输入和输出，构建整个 BERT 模型。

五、训练模型以及结果

在定义好模型结构之后，需要设置初始的超参数如学习率、训练批次大小、训练回合数、损失函数等。模型的训练部分代码如下：

```
learning_rate = 5e-6
batch_size = 16
maxlen = 60
new_class_nums = 15
Loss='sparse_categorical_crossentropy'
model = build_bert_model(config_path,bert_model_path,new_class_nums)
model.compile(
    loss=Loss,
    optimizer=Adam(learning_rate),
    metrics=['accuracy'],
)
model.fit_generator(
    train_generator.forfit(),
    steps_per_epoch=len(train_generator),
    epochs=10,
    validation_data=test_generator.forfit(),
    validation_steps=len(test_generator),
    shuffle=True,
    callbacks=[early_stop, checkpoint]
)
```

其中损失函数是系数分类交叉熵。根据第二章中的介绍，对于分类问题更适合使用交叉熵作为损失函数。对于文本分类问题 BERT 模型使用稀疏交叉熵作为损失函数能更好的达到预期效果。其中模型使用的是 Adam 学习率优化器，可以使得 BERT 模型快速收敛。

最后在训练代码中使用回调函数记录，并使用 plt 绘图：

```
callbacks=[
    early_stop,
    checkpoint,
    LossHistory(train_loss, val_loss, train_acc, val_acc)
]
plt.figure(figsize=(20, 7))
plt.plot(train_loss, label='Training Loss')
plt.plot(val_loss, label='Validation Loss')
plt.xlabel('Iteration')
plt.ylabel('Loss')
plt.title('Training and Validation Loss')
plt.legend()
```

```
plt.grid(True)
plt.savefig('loss_curve.png')
plt.show()
```

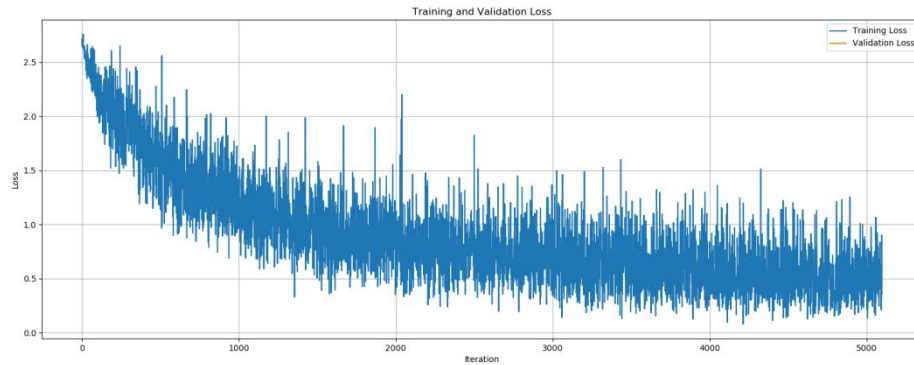


图 4-1 BERT 模型训练 loss 图

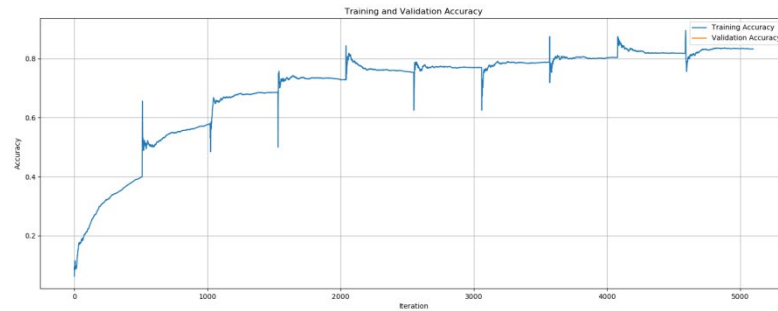


图 4-2 BERT 模型训练 accuracy 图

由图 4-1 可得出在最开始训练时，模型的损失接近 3.0，经过 2000 次训练之后，loss 下降到 0.75 左右。之后模型又进行了 3000 次训练，loss 保持在 0.5 左右。最后由于设置的 early_stop，在连续 10 个回合没有下降之后触发了训练停止。早停代码如下：

```
early_stop = keras.callbacks.EarlyStopping(
    monitor='val_loss',
    patience=3,
    verbose=2,
    mode='min'
)
```

由图 4-2 可得出在训练最开始准确率在 0.1 左右，随着训练轮次增加，在准确率快速提升。之后准确率保持在 0.8 左右直到结束。

4.2 NCF 模型实验与分析

在 BERT 模型实验之后，本小节继续讲解如何实现推荐算法的模型以及训练和分析。

4.2.1 NCF 模型推荐实验总体流程

本实验的实验过程为：收集医疗相关文章，数据预处理，构建 NCF 模型，训练模型，最后选择最佳训练结果以及将训练过程可视化。

4.2.2 数据预处理

数据使用 python 的 selenium 和 lxml 库来爬取网页文章数据，然后设定预先的用户-文章感兴趣矩阵。通过 pandas 将导出的用户-文章感兴趣矩阵导入数据集。

```
def read_base() -> List[Tuple[int, int, int]]:
    data = []
    with open(dataset_path, 'r', newline='') as file:
        reader = csv.reader(file)
        header = next(reader)
        for row in reader:
            user_id, article_id, score = map(int, row)
            data.append((user_id, article_id, score))
    return data
```

上述代码将数据打包成(用户 id，文章 id，评分)三元组的列表返回，用于将文件内的费结构化数据转化为可以用于 NCF 模型训练的结构化模型。

4.2.3 模型结构设计及参数调整

在完成数据的处理之后，需要构建模型。在 NCF 模型中，通过输入用户的 id 和文章的 id 返回两者的相似度。根据《Neural Collaborative Filtering》论文所描述，NCF 模型的结构应该为：Embedding 层，GMF 和 MLP 层，NeuMF 层。论文中并没有明确定义各层参数、目标函数以及各层具体层结构。本系统对 NCF 模型结构的设计如下：

```
def NeuMF_model_test(n_user: int, n_item: int, dim=3, layers=[6,3]) \
    -> Tuple[tf.keras.Model, tf.keras.Model, tf.keras.Model]:
    user_id = tf.keras.Input(shape=(), name='user_id', dtype=tf.int32)
    item_id = tf.keras.Input(shape=(), name='item_id', dtype=tf.int32)
    u = tf.keras.layers.Embedding(n_user, dim)(user_id)
    i = tf.keras.layers.Embedding(n_item, dim)(item_id)
    gmf = u * i
    gmf_out = tf.keras.layers.Dense(1, activation='sigmoid', name='gmf_out')(gmf)
    mlp = tf.concat([u, i], axis=1)
    for n in layers:
        mlp = tf.keras.layers.Dense(n, activation='relu')(mlp)
    mlp_out = tf.keras.layers.Dense(1, activation='sigmoid', name='mlp_out')(mlp)
```

```
x = tf.concat([gmf, mlp], axis=1)
```

```
out = tf.keras.layers.Dense(1, activation='sigmoid', name='out')(x)
```

其中 **dim** 是 Embedding 层的嵌入维度。维度越大，模型捕获特征的能力越强，但是可能导致过拟合。**Layer** 是多层感知机的层数以及全连接层的神经元个数，神经元个数越多，捕获特征能力越强，神经元过多同样会导致过拟合问题。**GMF** 层中通过用户 **id** 特征向量 **u** 与文章特征向量 **i** 相乘实现广义矩阵分解。**MLP** 层通过多个全连接层的叠加实现多层感知机，全连接层使用计算开销小的 **ReLU** 激活函数。最终结果经过一个神经元为 1 的全连接层，进行用户 **u** 对 **i** 的感兴趣程度预测，并使用 **sigmoid** 函数进行激活，将数据控制在(0,1)内。然后将 **GMF** 和 **MLP** 输出的张量进行拼接输入 **NeuMF** 层。**NeuMF** 层再次做一个简单的全连接进行输出。

图形结构如图 4-3 所示：

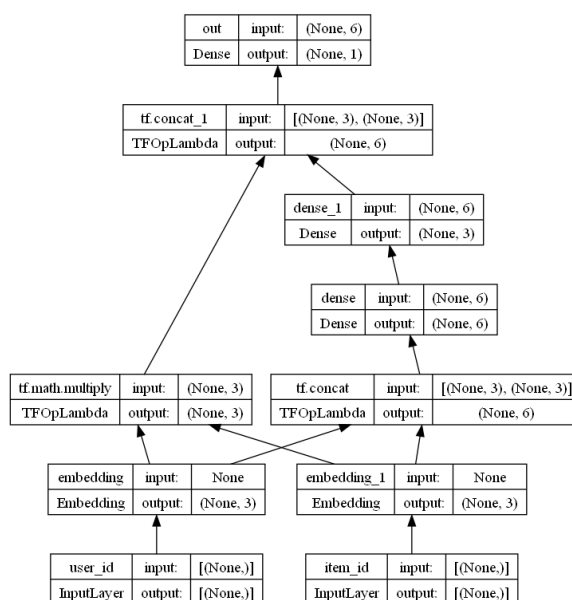


图 4-3 NCF 模型结构图

在初始 NCF 模型构建完成之后对其进行训练。将模型模拟为分类问题，以及推荐给用户和不推荐。训练使用的二元交叉熵损失函数，分为推荐和不推荐两种情况。学习率以及梯度下降算法使用的是学习率优化器 **Adam**。训练结果如下：

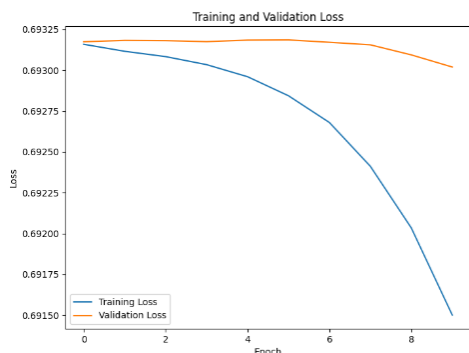


图 4-4 GMF 的 Loss 曲线



图 4-5 GMF 的 Accuracy 曲线

由图 4-4 可以看出在训练过程中训练集的 loss 在下降，但是测试集的 loss 几乎不变；同时，由图 4-5 可以看出 GMF 训练集的 accuracy 在上升，而测试集的 accuracy 几乎不变，说明 GMF 模型明显出现过拟合问题。图中在 8 个 epochs 后 loss 下降仅下降了 0.002，说明模型的捕获特征能力不足，综合来看是 loss 目标函数存在问题。虽然推荐可以作为二分类问题，即推荐与不推荐。但是二元交叉熵损失函数难以捕获到预测值和真实值之间的距离关系，难以进行优化。

于是对从新模型结构进行调整。将损失函数从二元交叉熵损失函数改为均方误差损失函数 MSE，同时为了提高 GMF 和 MLP 的特征捕获能力，对 Embedding 层的嵌入维度和 MLP 的层数做了修改：

为各层增加了 L2 正则化项，以防止过拟合。最后修改 NCF 的层的结构，具体修改如下：

```
gmf_dim, mlp_dim, layers, l2 = 8, 16, [32, 16, 8], 1e-4
```

根据实验，将参数设置上述所示时可以在获得最大特征提取能力的同时防止了过拟合。并对 NeuMF 层做了修改：

```
x = tf.keras.layers.Dense(16, activation='relu', kernel_regularizer=l2)(x)
x = tf.keras.layers.Attention()([x, x])
out = tf.keras.layers.Dense(1, activation='sigmoid', kernel_regularizer=l2, name='out')(x)
```

其中新添加了 16 个神经元的全连接层和一个注意力机制层，通过注意力机制，模型可以专注于对预测最为重要的特征，从而提高在新数据上的泛化能力。修改后的模型结构如图 4-6 所示：

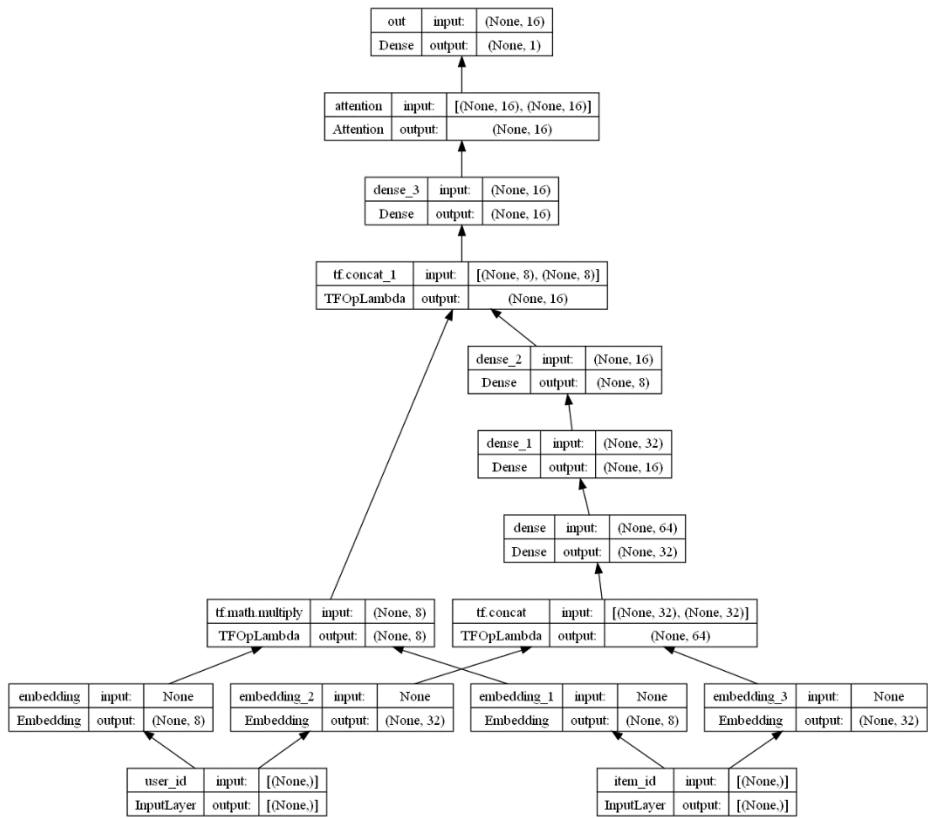


图 4-6 修改后 NCF 模型结构图

经过修改，模型的训练情况如下：

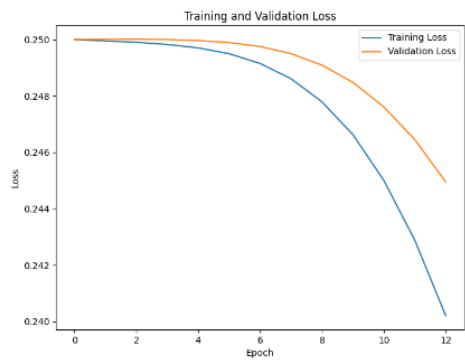


图 4-7 改良后 GMF 的 Loss 曲线

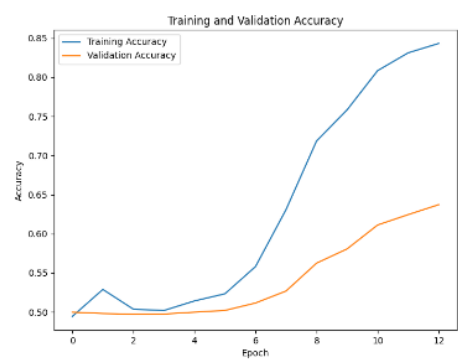


图 4-8 改良后 GMF 的 Accuracy 曲线

经过修改，可以观察到测试集和训练集的 loss 下降基本保持一致。在 12 个 epochs 之后下降了大约 0.01，并且 loss 值 0.24 要远低于修改之前的 0.69。同时，相比之前的测试集准确率，修改之后的准确率少有提升，如图 4-7 和 4-8 所示。

修改前后的模型 loss 和准确率对比：

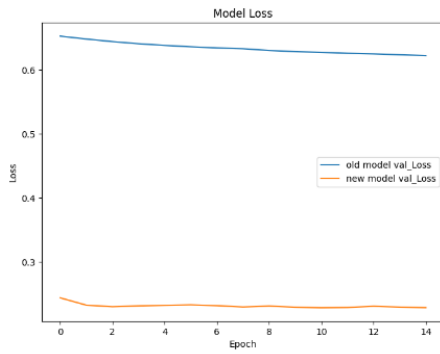


图 4-9 模型间的 Loss 曲线对比

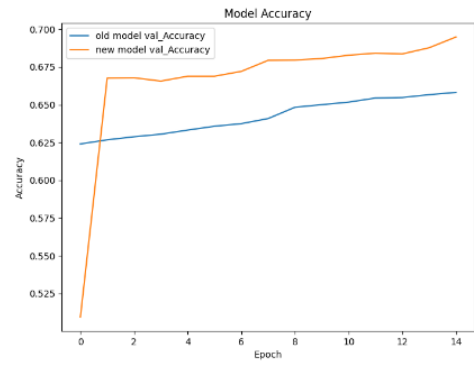


图 4-10 模型间的 Accuracy 曲线对比

根据图 4-9 和图 4-10 可知，修改后的模型在降低损失值和提高准确率方面均有所提升。

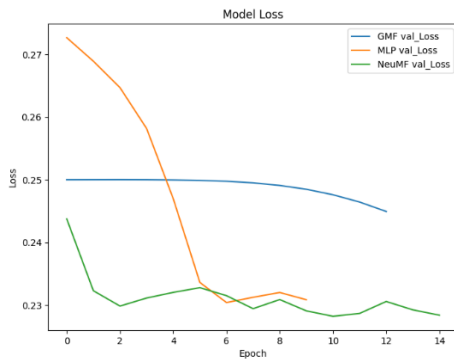


图 4-11 模型各层的 Loss 曲线对比

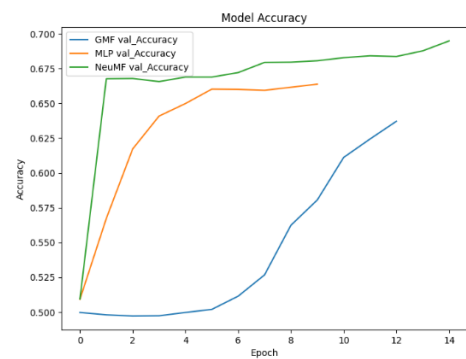


图 4-12 模型各层 Accuracy 曲线对比

根据图 4-11 和 4-12 可知，实验结果符合《Neural Collaborative Filtering》论文所描述，即 GMF 无法充分挖掘复杂的“用户-物品”交互模式，并且在增加维度 dim 的时候会导致过拟合，而 MLP 能更好地捕捉“用户-物品”之间的复杂交互关系却不能有效地建模简单的线性相关关系，所以将两者的输出结合通过 NeuMF 层处理可以充分融合两者的特点并获得效果更好的模型。

4.3 医疗预测模型实验与分析

在 NCF 模型实验之后，本小节继续讲解如何实现推荐算法的模型以及训练和分析。

4.3.1 医疗预测模型实验流程

医疗预测模型的本质是通过学习病例数据，构建预测函数进行预测。由于预测函数是未知的数学模型，所以只能通过这些病例数据进行反向建模，确定预测函数的参数和形式。

在构建模型之前需要进行病例数据收集，数据预处理。之后将进行线性回归模型构建训练模型，最后得到训练的模型。

4.3.2 数据预处理

数据选用的 Kaggle 上的 Disease Predictions 开源数据集。在进行训练之前首先要对数据进行预处理。由于收集到的数据大多数为未发生疾病的样本，导致了严重的样本不平衡问题。训练后的模型虽然准确率高达 90%，但实际上是模型每次都预测为“不患病”，这并不能反映模型对实际病例的预测能力。因此需要对数据进行平衡处理。处理代码如下：

```
def balance_data(data, target_column):  
    class_0 = data[data[target_column] == 0]  
    class_1 = data[data[target_column] == 1]  
    minority_count = min(len(class_0), len(class_1))  
    class_0_downsampled = resample(class_0, replace=False, n_samples=minority_count,  
random_state=42)  
    balanced_data = pd.concat([class_0_downsampled, class_1])  
    return balanced_data
```

代码首先将数据分为两个类别。然后获取数量较少的类别的数据数量并对数量较多的类别进行下采样，即随机选择 minority_count 个样本。最后组成数量平衡的样本并返回。

4.3.3 模型构建与模型选择

处理好数据之后需要构建模型用来训练，构建模型代码如下：

```
model = tf.keras.Sequential([  
    tf.keras.layers.Dense(64, activation='relu', input_shape=(input_shape_1,)),  
    tf.keras.layers.Dense(32, activation='relu'),  
    tf.keras.layers.Dense(16, activation='relu'),  
    tf.keras.layers.Dense(1, activation='sigmoid')  
)  
optimizer = tf.keras.optimizers.Adam(learning_rate=lr)  
model.compile(optimizer=optimizer, loss='binary_crossentropy', metrics=['accuracy'])
```

模型由多层感知机构成，每一层使用 ReLU 函数激活，并在最后一层使用 sigmoid 函数激活并输出二分类结果。模型使用二元交叉熵(BCE, Binary Cross Entropy)作为损失函数。

对于多属性输入预测单一标签的问题有很多种方法，其中包括全连接层，多层感知机，注意力机制，特征交互等。损失函数也可以选择二元稀疏交叉熵损失函数或者均方误差损失函数。实验数据如下：

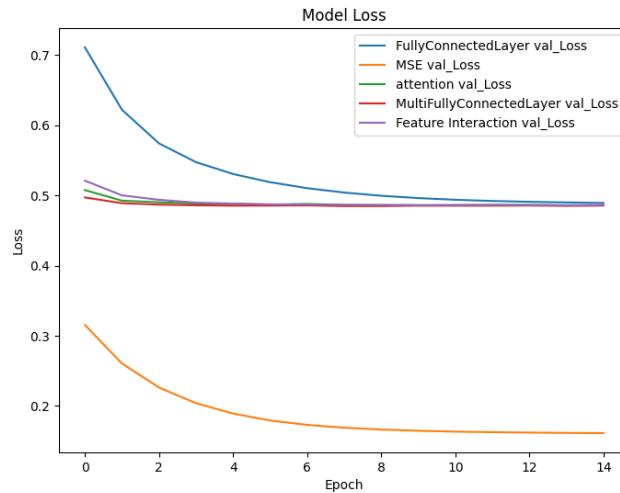


图 4-13 各模型的 Loss 曲线

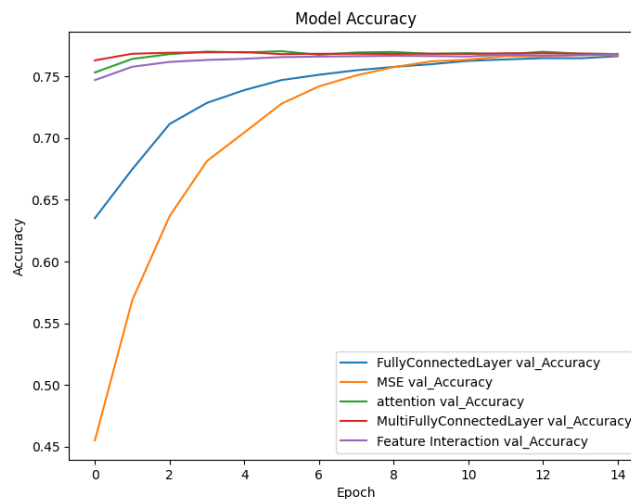


图 4-14 各模型的 Accuracy 曲线

由图 4-13 可知，单一的全连接层是收敛速度最慢的，而多个全连接层组成的多层感知机是收敛最快的。但是不同模型的 loss 在经过 15 轮次训练之后都趋近于同一水平。而由图 4-14 可知，均方误差 MSE 作为目标函数收敛最慢，在多属性输入做二分类的问题上，并不适合使用均方误差。分类问题使用二元稀疏交叉熵更加合适，而均方误差更适做回归问题。收敛最快的模型结构还是多层感知机，综上所述，在本系统中医疗预模型最适用的模型结构是多层感知机，而最适用的损失函数是二元稀疏交叉熵。

4.4 模型在系统中的应用测试

在进行了模型设计以及实验之后，需要把模型实际应用到系统中。本小节主要讲解模型在系统中的应用实现情况。

4.4.1 医疗问答测试

问答系统中需要测试其对不同意图以及实体的识别。对“询问疾病治疗时长”意图识别进行测试：询问“感冒的治疗时间是多久？”，测试结果如下：

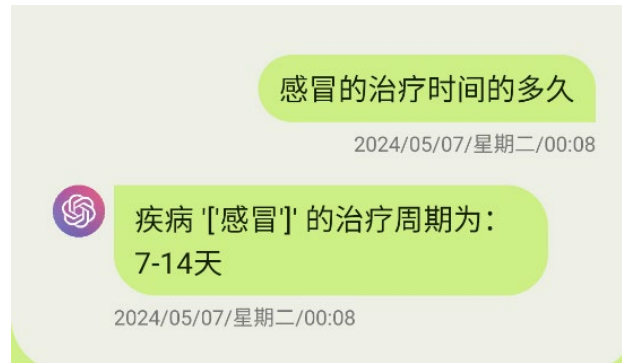


图 4-15 询问治疗时长的测试结果

由图 4-15 可知，模型能成功识别文本中“询问疾病治疗时长”的意图，并且能成功识别实体：“感冒”。对“询问疾病病因”意图识别进行测试：询问“月经不调的原因是什么？”，测试结果如下：

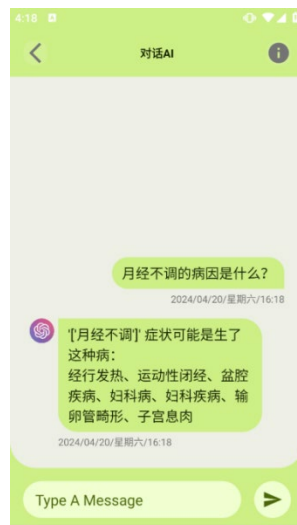


图 4-16 通过病症询问疾病测试结果

由图 4-16 可知，模型能够成功识别文本中“询问疾病病因”的意图。并且能够识别出实体：“月经不调”。

对“疾病定义”意图识别进行测试：询问“感冒的定义是什么？”，测试结果如下：

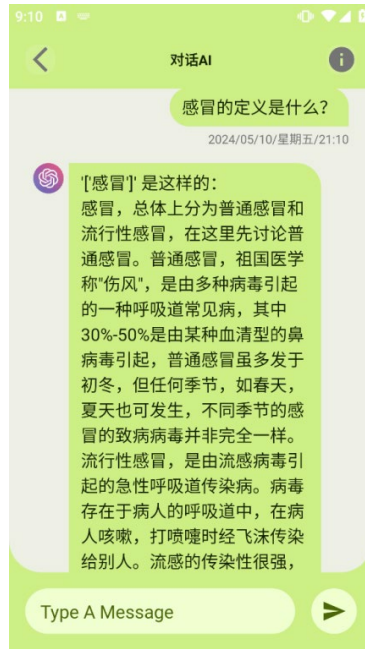


图 4-17 询问感冒的定义测试结果

由图 4-17 可知，模型能成功识别文本中“询问疾病定义”的意图。对“询问疾病所属科室”意图识别进行测试：询问“老年痴呆的所属科室是？”，测试结果如下：

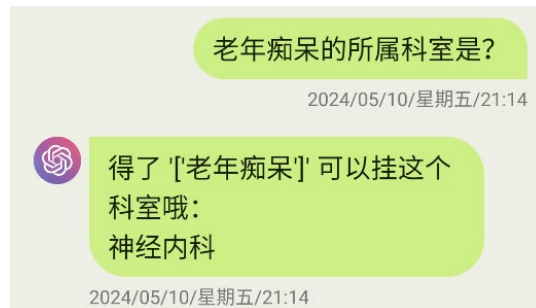


图 4-18 询问老年痴呆所属科室测试结果

由图 4-18 可知，模型能够成功识别文本中“询问疾病所属科室”的意图。并且能够识别出实体：“老年痴呆”。对“询问疾病传染性”意图识别进行测试：询问“感冒的传染性是多少？”，测试结果如下：

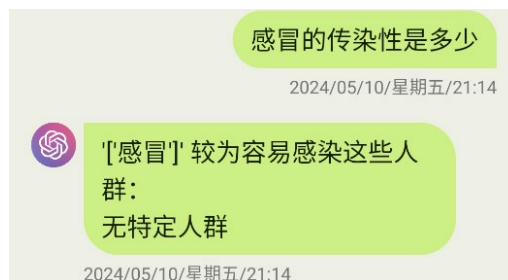


图 4-19 询问感冒的传染性测试结果

由图 4-19 可知，模型能够成功识别文本中“询问疾病传染性”的意图。

4.4.2 用户个性化推荐测试

推荐系统需要测试系统是否会根据每个用户感兴趣的实体推荐相应的文章。测试账号 139 号用户的初始实体是：“醉酒，鸡蛋，肾虚”。根据包含初始实体的所有文章，随机生成了 139 号喜欢的文章。这些数据都参与了 NCF 模型的训练。测试预期是获得相关性较高的个性化推荐数据。测试结果如下：



图 4-20 通过推荐算法推荐的文章

根据图 4-20 可知，推荐的文章几乎跟初始矩阵实体强相关。系统测试符合预期结果。

4.4.3 医疗预测测试

医疗预测系统需要根据用户的健康状况数据预测用户患病概率。数据选取两条数据集中测试集的数据作为模型的输入测试，其中具体数据如表 4-1 所示，前 19 项是测试集中用户的个人健康数据，后 2 项是测试集中用户的患病标签。

数据输入之后，得到预测情况：



图 4-21 医疗预测数据以及结果(1)

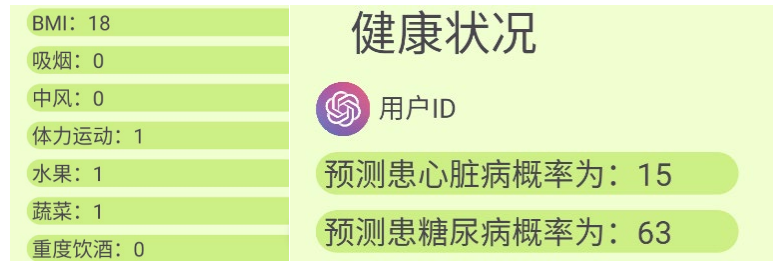


图 4-22 医疗预测数据以及结果(2)

表 4-1 用户健康状况输入数据

健康情况\用户健康数据	用户 1	用户 2
是否患有高血压	是	否
是否患有高胆固醇	是	否
身体质量指数 BMI	45	18
是否吸烟	是	否
是否中风	是	否
是否有体力运动	否	是
是否吃水果	否	是
是否吃蔬菜	否	是
是否重度饮酒	是	否
是否有医疗保健	否	是
没有医疗保险	是	否
健康状况评分	5	30
心理健康评分	6	25
身体健康评分	10	10
是否行走困难	是	否
性别	男	男
年龄	45	23
教育水平	2	6
收入水平	3	5
患有心脏病标签	患病	不患病
患有糖尿病标签	患病	患病

根据图 4-21 和 4-22 的预测结果来看，模型对第一条数据预测患病的概率都是 90%以上，跟真实患病结果一致。模型对第二条数据的预测为患有心脏病的概率 15%，患有糖尿病的概率为 63%，也大体跟真实患病结果一致。

经过测试，系统中各模型均达到预测效果。

第5章 结论与展望

5.1 系统不足之处

本次设置的智能对话和推荐系统顺利通过了测试。事实也证明系统是比较完善的。不过与大型系统相比还有很大的进步空间。在本系统中，基于知识图谱的实体距离查询速度较慢，训练的模型也为实现多线程并发运行。导致计算结果较慢。由于模型不能实现并发运行，所以无法实现多个用户的高并发使用。需要分别将每个用户的推荐列表提前计算出来，这大大耗费了时间。此外，本系统中的对话模块并没有使用实体命名识别模型，无法识别出实体的歧义。在本系统设计最初是使用到了 Bi-LSTM-CRF(Bidirectional Long Short-Term Memory - Conditional Random Field)来实现实体命名识别任务。但是由于用于训练的数据集数量少、模型结构简单导致效果并不理想，经常把一个分类中不是实体的关键词视为该分类中的实体。相比匹配到的词语存在语义冲突问题，该模型的实体识别错误对系统的后续处理存在更严重的影响。所以在最后本系统取消了使用实体命名识别而是采用 AC 自动机树来识别实体。在最开始本系统想使用 Hugging Face 的 GPT2 模型实现自然语言生成，但是由于计算机性能限制，本系统只能使用 BERT-Base 预训练模型来进行意图识别。

5.2 展望

虽然 BERT 模型在自然语言处理中已经广泛应用，能够处理本分类、命名实体识别和句子关系判断等任务。但随着 GPT 的出现，自然语言生成迎来质的飞跃，在问答系统中不再是简单的讲意图识别然后根据模板回复了，而是可以真正的执行文本生成、文本补全等任务。虽然 GPT 在自然语言生成上已经取代了 BERT，但是它是基于无监督训练的生成模型，在医疗等需要严格准确专业知识领域的回答并不能做到 100%准确，还需要继续磨练。随着技术的更新和进步，相信未来能出现一个真正的接近人类顶尖水平的智能医疗对话 AI。

参考文献

- [1] 中华人民共和国人民政府网. 2021 年我国卫生健康事业发展统计公报 [EB/OL].https://www.gov.cn/lianbo/bumen/202310/content_6908685.htm,2023-10-12.
- [2] Wang H, Liu C, Xi N, et al. Huatuo: Tuning llama model with chinese medical knowledge[J]. arXiv preprint arXiv:2304.06975, 2023.
- [3] Zhang H, Chen J, Jiang F, et al. Huatuogpt, towards taming language model to be a doctor[J]. arXiv preprint arXiv:2305.15075, 2023.
- [4] Qian J, Jin Z, Zhang Q, et al. A Liver Cancer Question-Answering System Based on Next-Generation Intelligence and the Large Model Med-PaLM 2[J]. International Journal of Computer Science and Information Technology, 2024, 2(1): 28-35.
- [5] 王琼. 人工智能自然语言处理临床应用的伦理研究[J]. 医学与哲学, 2023, 44(21): 12-16. doi: 10.12014/j.issn.1002-0772.2023.21.03.
- [6] Hopfield J J. Neural networks and physical systems with emergent collective computational abilities[J]. Proceedings of the national academy of sciences, 1982, 79(8): 2554-2558.
- [7] Hochreiter S, Schmidhuber J. Long short-term memory[J]. Neural computation, 1997, 9(8): 1735-1780.
- [8] Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need[J]. Advances in neural information processing systems, 2017, 30.
- [9] Devlin J, Chang M W, Lee K, et al. Bert: Pre-training of deep bidirectional transformers for language understanding[J]. arXiv preprint arXiv:1810.04805, 2018.
- [10] Kim, Yoon. “Convolutional Neural Networks for Sentence Classification.” Conference on Empirical Methods in Natural Language Processing (2014).
- [11] Goldberg D, Nichols D, Oki B M, et al. Using collaborative filtering to weave an information tapestry[J]. Communications of the ACM, 1992, 35(12): 61-70.
- [12] Sarwar B, Karypis G, Konstan J, et al. Item-based collaborative filtering recommendation algorithms[C]//Proceedings of the 10th international conference on World Wide Web. 2001: 285-295.
- [13] Koren Y, Bell R, Volinsky C. Matrix factorization techniques for recommender systems[J]. Computer, 2009, 42(8): 30-37.
- [14] Zhang Z. The singular value decomposition, applications and beyond[J]. arXiv preprint arXiv:1510.08532, 2015.
- [15] Ruder S. An overview of gradient descent optimization algorithms[J]. arXiv preprint arXiv:1609.04747, 2016.
- [16] He X, Liao L, Zhang H, et al. Neural collaborative filtering[C]//Proceedings of the 26th international conference on world wide web. 2017: 173-182.
- [17] Ramachandran P, Zoph B, Le Q V. Searching for activation functions[J]. arXiv preprint arXiv:1710.05941, 2017.
- [18] Goodfellow I, Bengio Y, Courville A. Deep learning[M]. MIT press, 2016: 171-185.
- [19] Goodfellow I, Bengio Y, Courville A. Deep learning[M]. MIT press, 2016: 128-132.
- [20] Kingma D P, Ba J. Adam: A method for stochastic optimization[J]. arXiv preprint arXiv:1412.6980, 2014.

- [21]Phaisangittisagul E. An analysis of the regularization between L2 and dropout in single hidden layer neural network[C]//2016 7th International Conference on Intelligent Systems, Modelling and Simulation (ISMS). IEEE, 2016: 174-179.
- [22]邵超, 万春红. ER 图的精细设计[J]. 计算机教育, 2015 (8): 78-81.
- [23]陈涛. MVVM 设计模式及其应用研究[J]. 计算机与数字工程, 2014, 42(10): 1982-1985.
- [24]Evans E. Domain-driven design: tackling complexity in the heart of software[M]. Addison-Wesley Professional, 2004.
- [25]Silvescu A, Caragea D, Atramentov A. Graph databases[J]. Artificial Intelligence Research Laboratory Department of Computer Science, Iowa State University, 2012.

致 谢

四年的读书生活在这个季节即将划上一个句号，而于我的人生却只是一个逗号，我将面对又一次征程的开始。四年的求学生涯在师长、亲友的大力支持下，走得辛苦却也收获满囊。

对于这次毕业设计，最需要感谢的是我的导师王燕老师，她在整个过程中都给予了我充分的帮助和支持。老师不仅耐心地为我的毕业设计，并改进其中的不足之处，为我提出宝贵的建议，而且还在我遇到困难时尽心地进行指点与解答。在此借毕业设计完成之际，表示由衷的感谢和敬意。

此外，感谢大学期间所有教导过我的和其他帮助我的老师，你们对我的教导，是我大学四年最宝贵的财富。

最后，感谢学院为我提供良好的做毕业设计的环境。最后，向论文评阅人和答辩委员会的所有专家、教授致以最衷心的感谢！