项目详细方案

智能型教育机器人——童乐

目录

一、设计需求分析

1. 1 项目背景

1.1.1 基于人工智能社会背景下的巨大需求

随着人们生活节奏的加快，现在很多的父母因为工作或者其他的一些原因，能够陪伴孩子或者与孩子沟通的时间也变得越来越少，对于孩子的学习与教育更是显得力不从心，久而久之，对于孩子的发展和成长是非常不利的。在这种大的社会需求下，智能教育机器人就应运而生了，在一定程度上缓解了家长在育儿方面的痛点。

最近几年，基于人工智能这样一个大的社会背景下，机器人行业迎来了史上前所未有的发展，做为智能机器人中的教育机器人也成为了众多家长和孩子们的新宠。大多数智能教育机器人普遍都具有亲子沟通、英语辅导、教材同步和海量的云端资源，无论对于孩子个人习惯的培养还是学习方面都有着很大的帮助。

1.1.2 国家给予大力支持推动产业发展

从目前市场形式来看，教育机器人的发展可以说是如日中天，据最新的 2018 产业发展报告数据显示，2018 年我国机器人市场规模预计达到 87.4 亿美元，2013－2018 年均增长率达到 29．7%，市场发展形势一片大好。与此同时，国家发布了多项机器人补贴政策，国内机器人企业加速布局，企业自主研发技术专利不断革新，资本市场青睐度与日俱增。

1.1.3 机器人技术成为大趋势

机器人技术已然成为人工智能时代的一个大的趋势，是科技发展的必然产物，当然，目前人工智能技术还处于初级阶段，所以基于人工智能技术而研发生产的各种机器人也无法做到百分之百的完美，所以，本次我们的项目研究致力于打造智能陪伴学习机器人，帮助婴幼儿童学习知识，同时提供帮助大脑开发的娱乐游戏，实现互动陪护的目标。

1.2 项目需求

1.2.1 需求类型

根据中国机器人协会发布的数据显示，目前，我国教育机器人市场中，以学习型机器人为主导，其中学习型机器人和比赛型机器人占据了90%以上的市场份额。尽管随着机器人通用性的普及，未来学习型和比赛型机器人的概念或将有所弱化，通用型和专用型机器人概念将得到重视。**但短期内，学习型机器人占主导的市场格局仍将保持不变，将是未来行业竞争的热点领域。**

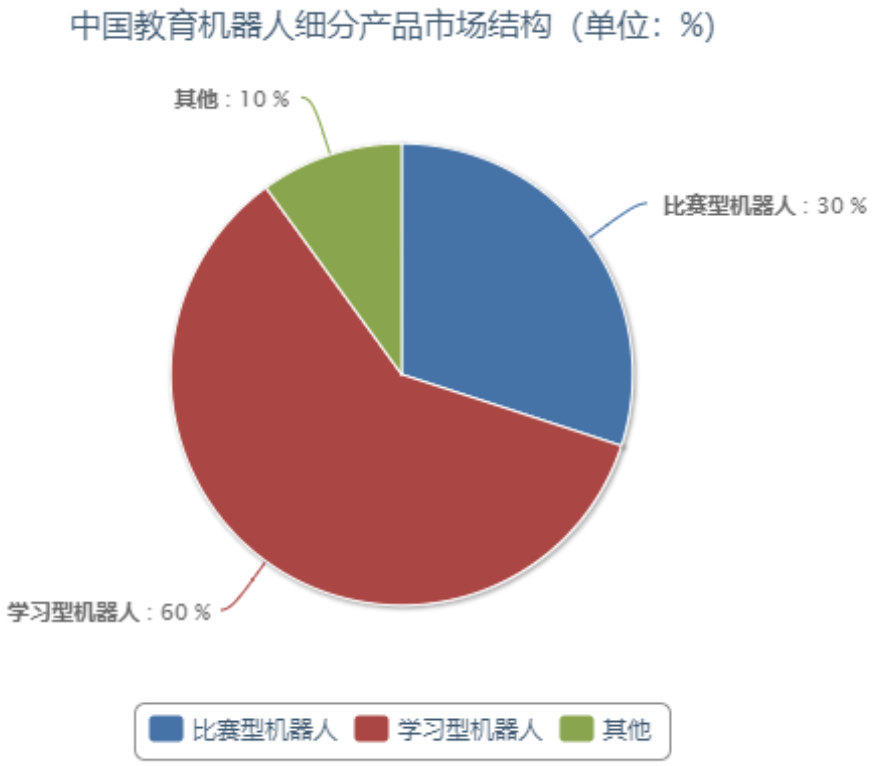


图 1-1 中国教育机器人细分产品市场结构

1.2.2 需求人群

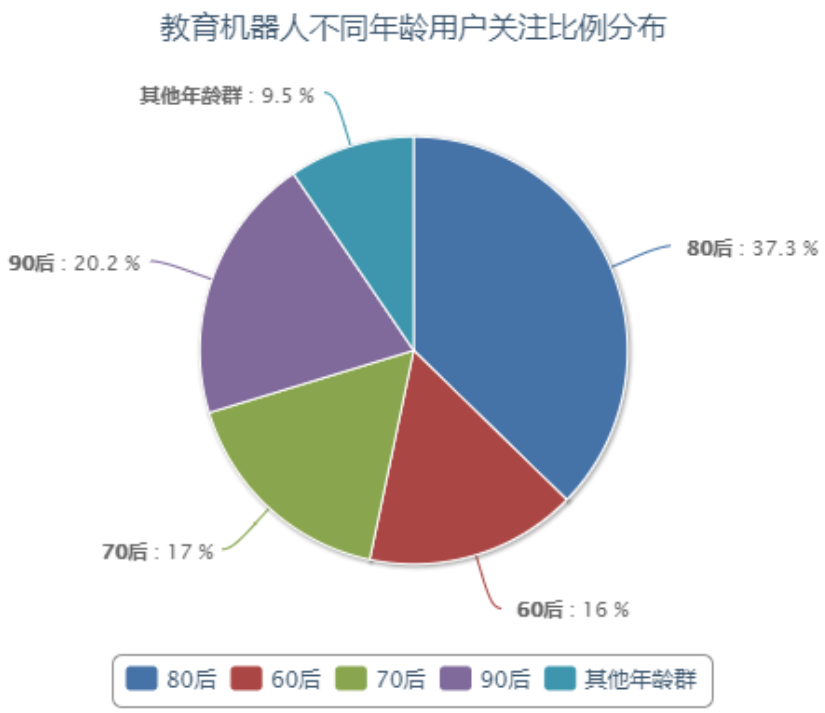


图 1-2 教育机器人不同年龄用户关注比例分布

根据 2010 年中国第 6 次人口普查结果显示，**我国 0～14 岁儿童超过 2.2 亿**，这批人中的绝大部分就是我们常说的零零后。二孩政策放开后，每年新生婴儿可达到 2000 万以上，许多父母也越来越愿意为孩子的成长和教育花钱。

从调查数据中我们可以看到，在用户层面最受关心的是教育陪护机器人，**使用陪护机器人用户中绝大部分是零零后群体，虽然他们本身不具备购买力，但他们背后的家长恰恰是以 80 后为主的职场主力军**。80 后对下一代的教育有两个鲜明的特点，其一是婴幼儿智能启蒙教育、其二是场景式陪伴，而教育陪护机器人正好具有这些功能。

1.3 项目概述

1.3.1 项目概述

智能型教育机器人——童乐是集学习、娱乐及陪伴为一体的儿童机器人，它不仅可以进行人机对话，还助力孩子在线学习。我们针对 0~12 岁的儿童进行设计，针对该阶段孩子成

长特征和关键期需求，内置故事、儿歌、唐诗、语言等早教资源。保证每个内容都是经典传世、寓教于乐的杰作，并且是适合给该阶段孩子听的。

在教育方面：童乐机器人内置学习模块，家长可通过用户端远程让孩子学习语文、数学、英语等知识，为孩子定制学习计划。同时，在孩子与童乐机器人进行对话的过程中，将会智能地回答孩子所提出的相关问题，例如：“一加一等于几”、“李白是谁”等知识的解答。

在娱乐方面：童乐机器人提供许多儿童歌谣、儿童故事等丰富孩子的课外知识，扩展孩子的知识库。设置了一些益智小游戏，如“成语接龙”、“接古诗词”等有趣的小游戏，陪伴孩子度过休闲时光。

在陪护方面：童乐机器人有着智能的人机交互，智能地识别语义，与孩子进行交流，解答孩子的奇思妙想，成为孩子的玩伴。家长还可使用手机用户端通过机器人与孩子进行微聊，实现远程陪伴，实时了解孩子的情况。

1.3.2 功能概述

本项目使用 Web app 作为载体，具有跨平台的优势，支持多种渠道，嵌入更加方便。与原生 APP 相比，Web app 在开发、功能、应用安装使用等方面都更有优势,也更加适用于本项目用户端的需求。

根据需求分析，在手机用户端我们开发了以下几点功能：

- 登录后可在手机端的菜单中选择想听的内容
- 定制学习计划，到点播放
- 联网微聊，与孩子实时聊天

智能机器人是直接与孩子进行对话的本体，通过模拟与分析孩子的日常行为及语言习惯，我们为机器人童乐开发了以下功能：

- 语音指令唤醒机器人
- 进行智能问答交流
- 语音指令进行功能模块选择
- 与手机端进行连接实现远程控制

1.3.3 产品设计总体流程

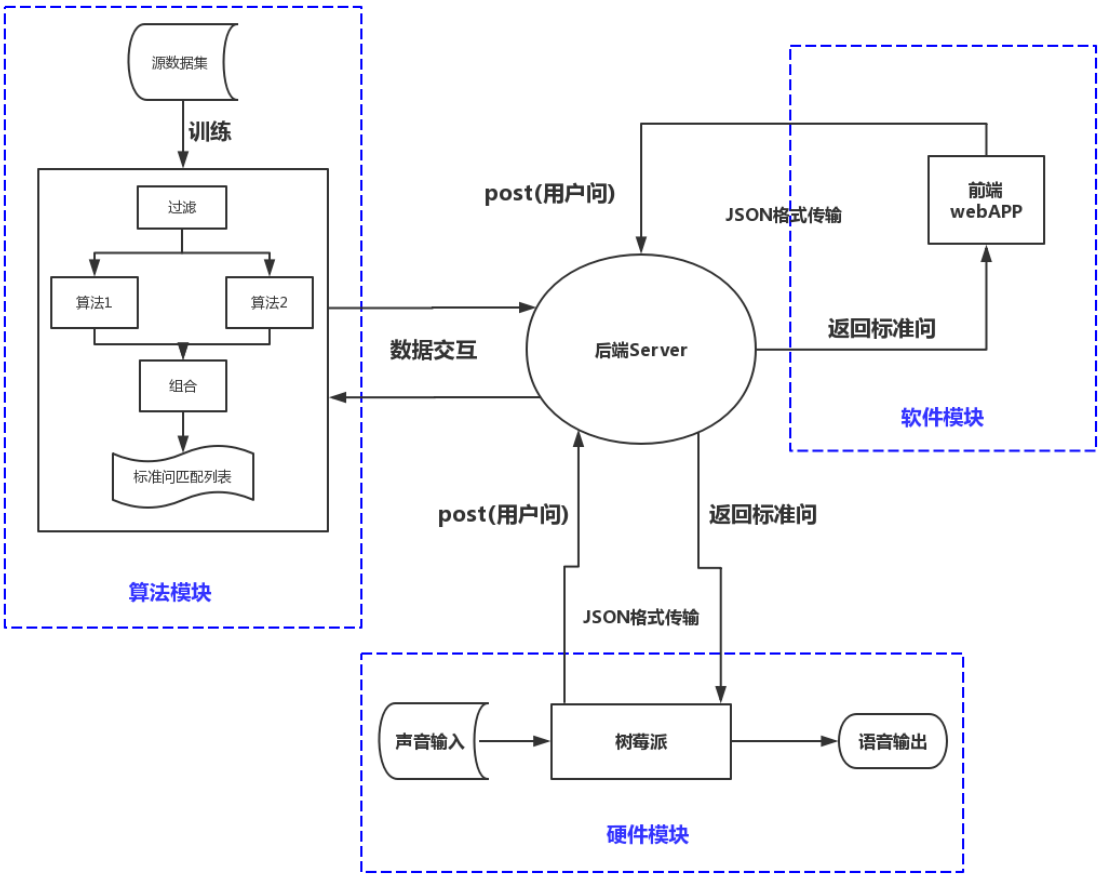


图 1-3 产品设计总体流程图

在软件系统的总体设计上，我们采用了前端后分离架构。主要分为算法模块、软件模块、硬件模块三大模块。各个模块之间通过统一数据格式进行网络通信数据交互。

二、特色与创新

2.1 针对儿童教育领域，设计更合理

通过对 0~12 岁年龄层的孩子进行调查分析，我们更加了解他们的常问问题及教育需求，从而为他们设计了多样化、多领域的资源库。童乐机器人贴合孩子的年龄层，对他们的早教

启蒙起到了帮助，同时具有强大的查询能力，解答孩子们的奇思妙想。语音互动，提高孩子的学习兴趣。

2.2 多种算法结合，智能问答更准确

为了使机器人能够准确识别孩子的语义，我们在底层自构建了算法模型，通过不断地测试训练，设计了一套问答匹配准确率最高的算法模型，结合了自然语言处理（NLP）技术的多种主流算法，克服了整个人机对话过程中难点。在语音识别、意图理解、信息抽取、信息检索、信息匹配的部分都更加准确，从而使得机器人达到智能的要求。

2.3 用户端界面美观，功能丰富

使用 Web app 实现一个界面美观，交互友好的用户端。简洁的界面设计使得用户更加直观地了解整个前端结构，能够快速地找到各种功能的入口。功能丰富，除了能选择学习的课程，还能为孩子定制学习计划，通过微聊，实时了解孩子动态。

2.4 资源库丰富，更新及时

我们还提供了后台管理系统，工作人员可以及时更新资源库，保证孩子可以了解到更多知识，拓宽孩子们的学习领域，跟上时事变化。资源库内容丰富健康，保证了孩子们接受正面教育，实现健康引导成长的目标。

三、算法构建

3.1 实现技术与过程

表 3-1 算法实现环境说明表

环境	说明
编程语言	Python3.6.2
操作系统	Windows10
主要框架	Web.py

表 3-2 算法核心模块说明表

模块名称	模块说明
filter_by_IR	基于信息检索角度过滤部分用户问
get_standard_question_by_cosSim	基于余弦相似度获取标准问
get_standard_question_by_NB	基于朴素贝叶斯获取标准问
get_standard_question_bt_compose	基于组合模型获取最终的标准问

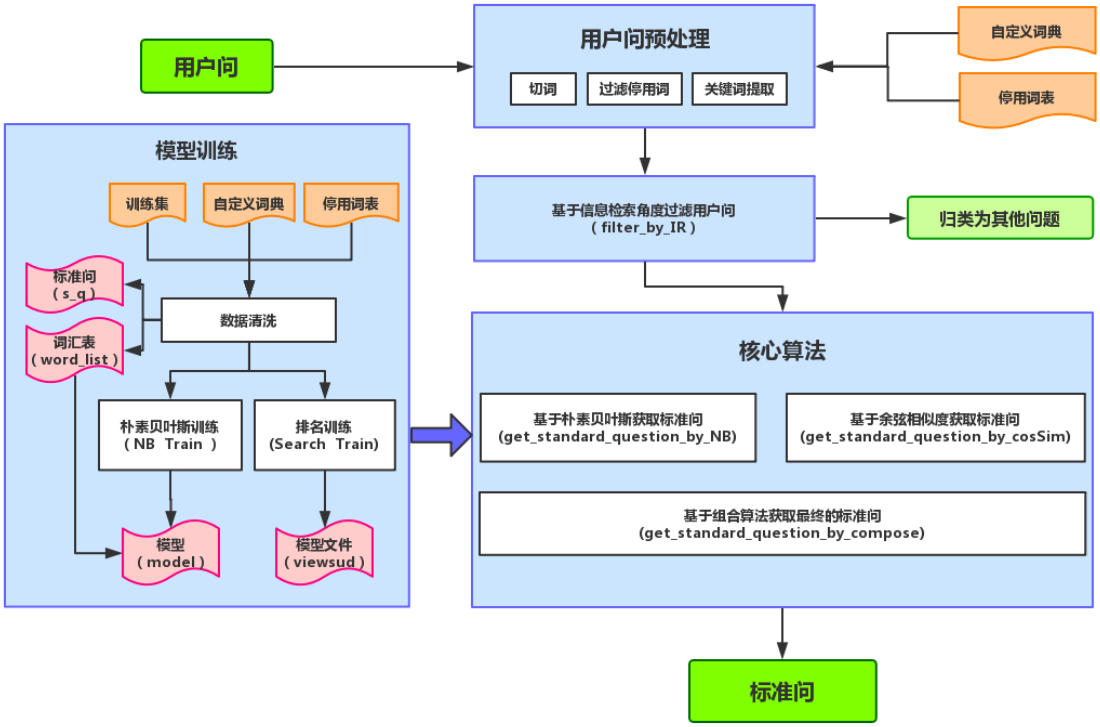


图 3-1 算法框架图

如图 3-1 所示，本团队针对该赛题所设计实现的核心算法主要分为以下四个部分：

1) 用户问预处理。主要是对接收到的用户问进行分词、过滤停用词、提取用户问中的语义词。
2) 用户问过滤。主要是过滤掉部分 “闲聊”式的提问，进一步提升匹配标准问的准确性。
3) 模型训练。主要是展现了核心算法的大致的训练过程。
4) 核心算法。主要是通过核心算法对过滤后的用户问匹配相应的标准问。

3.2 实现技术与过程

3.2.1 数据集分析

为了更好的对模型进行构建，突出模型构建和训练的整体过程，我们团队采用了一份基于网络爬虫及人工打标签的儿童常见提问 QA 数据集以及一份基于金融证券类 QA 数据集。首先利用金融证券类的数据集进行模型的训练，实验验证核心算法的理论可行后替换儿童常见问题数据集语料进行二次训练，最终实现整个模型的构建。介于儿童常见问题数据集打标签的数量不多，为了更好的突出核心算法的整个分析构建过程，本详细设计文档基于金融证券类的数据集展示了核心算法的分析过程以及整体构建过程。

首先我们团队从“数据集结构特征”、“数据内容”、“数据分布”等角度对其进行了数据可视化分析。

1	用户提问	知识库标准问
2	3加1等于几呀?	加法算术问题?
3	1加1等于几呀?	加法算术问题?
4	2减1等于多少呀?	减法算术问题?
5	可以讲个故事吗?	讲故事问题?
6	我想听个故事可以吗?	讲故事问题?

1	用户提问	知识库标准问
2	自助开户的第三方存管要怎么补办?	自助开户后如何补办三方存管?
3	自助开户三方存管	自助开户后如何补办三方存管?
4	手机App里开户后，如何绑定第三方存管	自助开户后如何补办三方存管?
5	自助开户后如何补办三方存管?	自助开户后如何补办三方存管?
6	我的帐户怎么不能进行和银行间的转账了?	转账失败的原因?

图 3-2 数据集结构图

如图 3-2 所示，通过分析数据集的结构特征，我们发现该数据集主要以“用户问-标准问”的形式存在，并且夹杂着中文、大小写英文、标点、特殊符号、重复等。在此分析基础上，我们首先进行了数据清洗：过滤掉部分标点符号、常用停用词，保留自定义词典中的词语，去重等操作。从而去除掉数据集中的部分噪声，为后续算法构建及模型训练提供帮助。

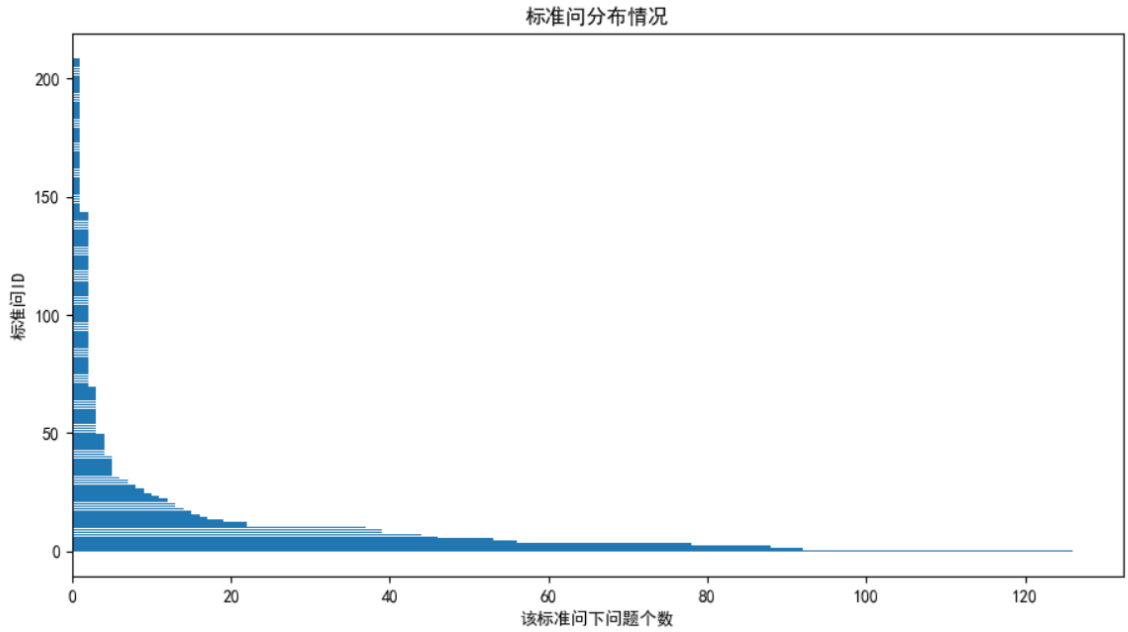


图 3-3 标准问分布图

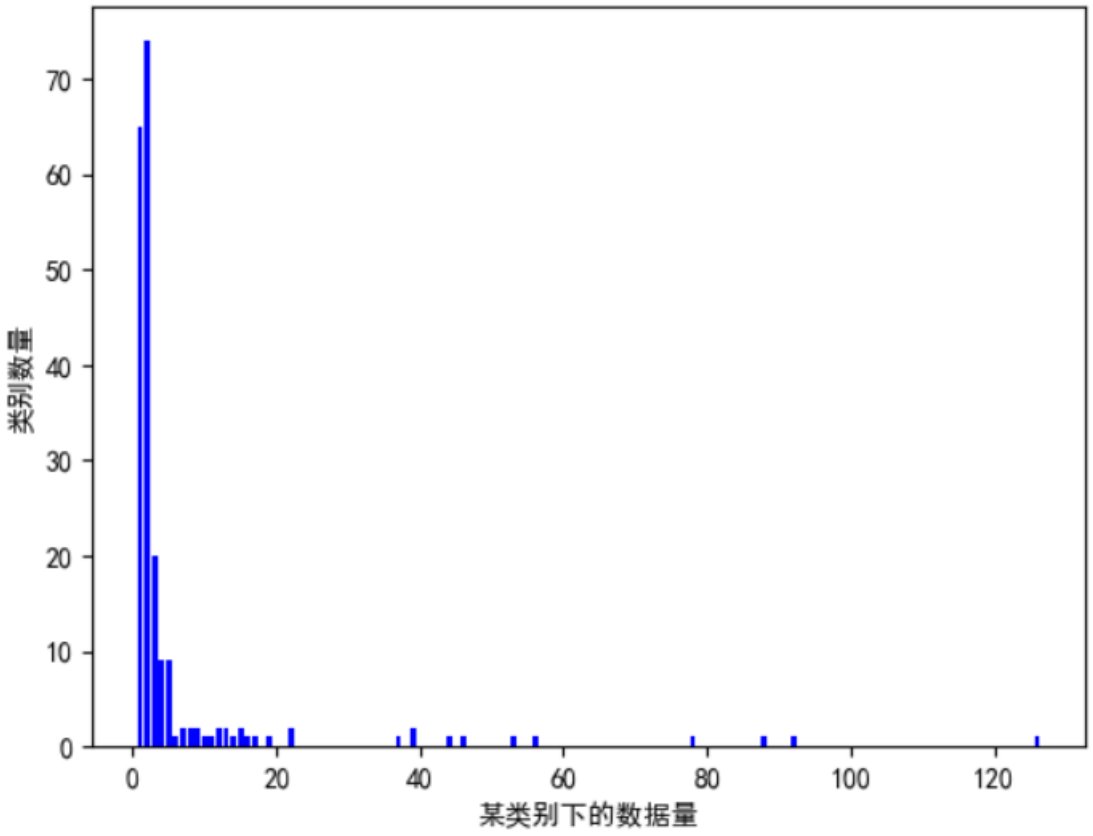


图 3-4 数据量分布图

如图 3-3 和图 3-4 所示，在分析数据集中标准问的分布情况和改标准问对应的用户问数量后，我们发现该数据集存在严重的类不平衡问题。其中，大部分标准问对应的用户问数量小于等 5，极少个标准问对应的用户问数量则达到了上百条。对此，我们从“数据增强”和“采样方式”两个角度进行了处理。

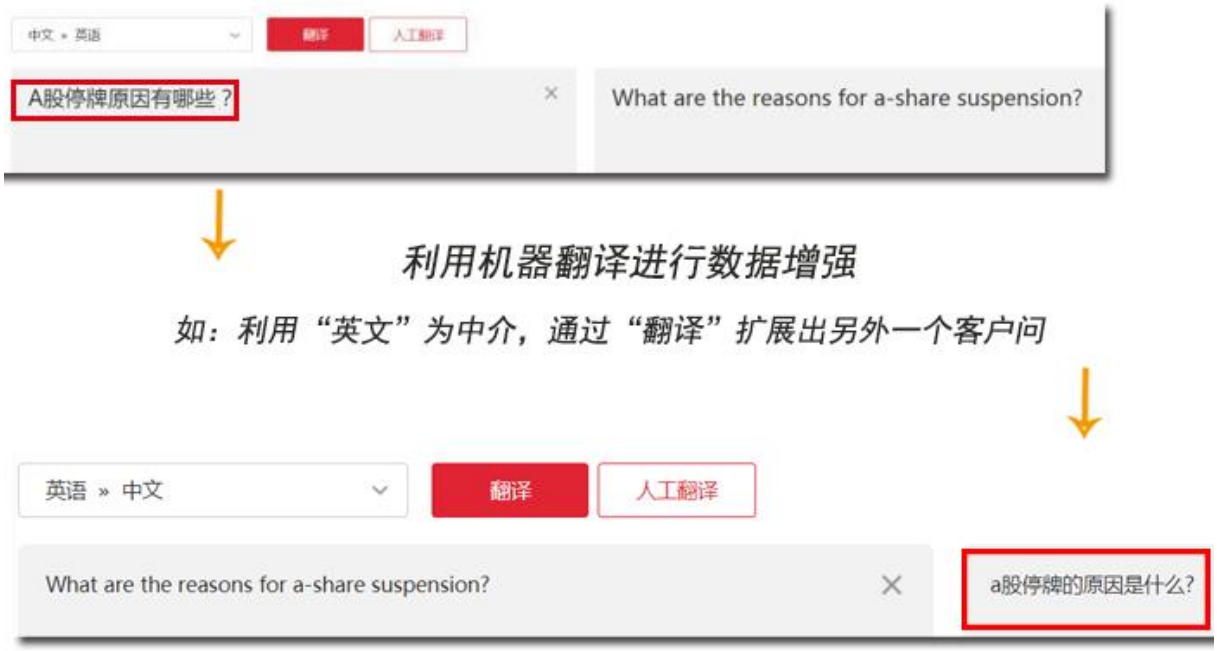


图 3-5 使用“机器翻译”进行数据增强原理图

首先，我们使用“机器翻译”的方式对数据进行了增强。从图 3-5 中可看出，标准问“A 股停牌原因有哪些？”经过中文机器翻译成英文，然后再从英文机器翻译回中的方式，原标准问则变成了“a 股停牌的原因是什么？”这个描述从语义上可近似相等，因而可扩展成为该标准问的一个用户问。此外，除了英文，我们还加入日文、韩文等语言充当“中介”语言，进而通过机器翻译这种方式所带来的语言之间的差异性增强了原有数据。

其次，通过分析数据量分布情况，在进行了数据增强的基础上，我们对于用户问数量较多的标准问采用了“欠采样”的方式进行采样处理；对于用户问数量较少的标准问采用了“过采样”的方式进行采样处理。

最后，通过“数据增强”与“过/欠采样”两种方式对原数据集中类不平衡问题进行了缓解。

3.2.2 问题分析与算法构建

经过讨论分析，我们的系统主要目的是要用户问尽可能准确的匹配到知识库中的标准问。该问题属于 NLP 领域的问题，传统的做法是利用句子间的相似度计算从而进行匹配。主要的构建整体思路如下图所示：

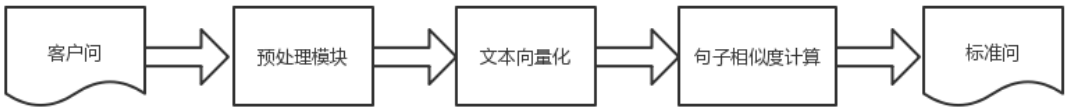


图 3-6 传统算法构建整体思路图

根据传统的解决方案思路，我们设计了基于余弦相似度标准问匹配算法。在文本向量化这一模块中我们采用了 TF-IDF 算法进行用户问与标准问的文本向量化。其中，余弦相似度以及 TF-IDF 的计算公式如下：

$$similarity = \cos(\theta) = \frac{A \bullet B}{\| A \| \| B \|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}} \quad (1)$$

$$\begin{aligned} TF-IDF(w_i) &= tf(w_i) \times idf(w_i) \\ &= tf_j(w_i) \times \log(N / df(w_i)) \end{aligned} \quad (2)$$

给定两个属性向量 A 和 B，其余弦相似性 θ 则可用式（1）表示，其中 $A_i B_i$ 分别代表向量 A 和向量 B 的各分量。

在式（2）中 $tf_j(w_i)$ 的含义为当前关键词 w_i 在文本 j 中出现的频率，N 则表示总文本数，$df(w_i)$ 则表示的是当前关键词 w_i 出现在了多少个文本中。

其中基于余弦相似度获取标准问算法（get_standard_question_by_cosSim）的核心步骤如下：

Step1：注入当前用户问以及知识库中的标准问

Step2：数据预处理

Step3：使用 TF-IDF 分别对用户问及标准问进行文本向量化

Step4：使用余弦相似度计算当前用户问与标准问的相似度

Step5：排序输出与当前用户问最相似的 TOP5 个标准问列表

除了传统的解决方案外，经过分析和讨论，我们认为该问题可以转化为机器学习中的多分类问题。于是变从该角度入手，运用机器学习中的常用分类算法针对本系统进行建模分析，最终采用了朴素贝叶斯算法的模型并加以优化，最后构建了整体算法框架中另外一个核心标准问匹配算法——get_standard_question_by_NB。该算法的训练过程伪代码如下：

表 3-3　get_standard_question_by_NB 伪代码符号说明表

符号	说明
get_standard_question_by_NB	基于朴素贝叶斯获取标准问
train_set	训练集
model	模型文件
load_data(self)	加载数据
get_question_txt()	得到模型文件（question）
prepare(self)	到模型文件（wordlist）
cal_probability(self)	计算先验/后验概率
save_model(self)	保存模型文件（model）

算法名称：get_standard_question_by_NB

输入：train_set

输出：model

（1） def load_data(self):

（2） def get_question_txt():

（3） def prepare(self):

（4） def cal_probability(self):

（5） def save_model(self):

3.3 算法优化

3.3.1 get_standard_question_by_NB 算法优化

该算法核心部分由朴素贝叶斯分类器组成，在给定目标值时假定属性之间相互条件独立的情况下，则该分类器可由以下式子表示：

$$P(\text{Category}|\text{Document}) = \frac{P(Document|Category)*P(Category)}{P(Document)} \quad (3)$$

在算法实现过程中，出现了没有出现过的词概率为 0 的情况以及在概率求积中遇到浮点数溢出的情况。针对该两种情况，我们讨论分析并在查找相关资料后作出了以下处理：

1） 加入拉普拉斯平滑参数。

2） 对贝叶斯公式中的分子取对数。

3.3.2 get_standard_question_bt_compose 算法优化

我们采用控制变量法对基于 NB 的标准问获取算法和基于余弦相似度的标准问获取算法进行并行实验，最后在测试集中我们发现两种单模型在 F1 指标值上分别只有 0.8 和 0.7 左右的表现。为分析模型误判的原因，我将误判的数据集提取出来进行可视化分析：

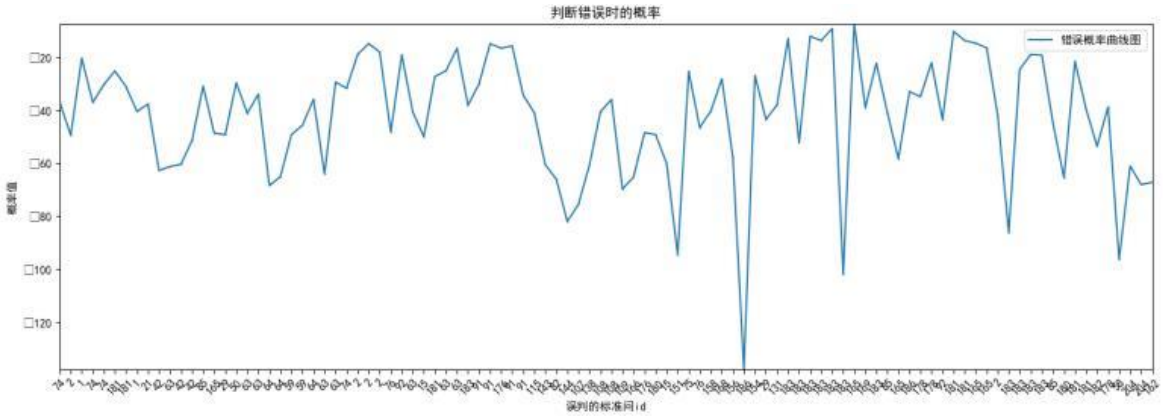


图 3-7 误判数据概率分布图

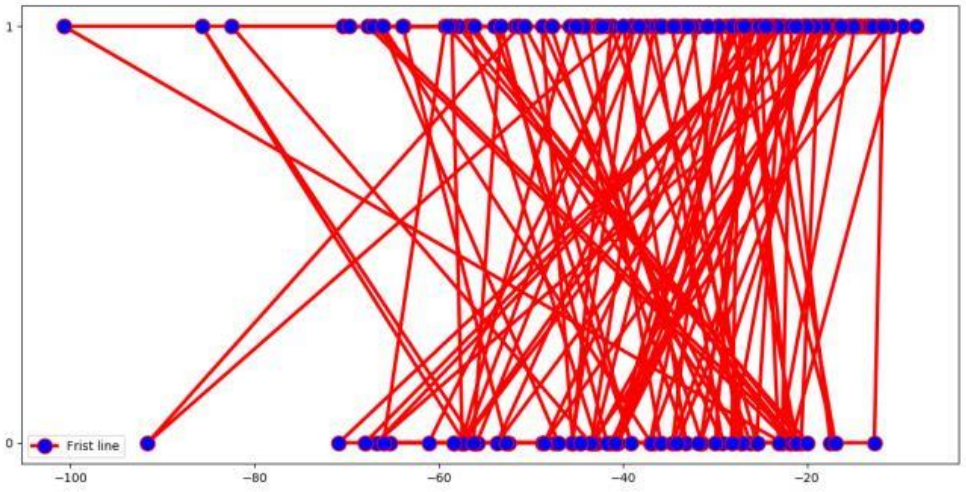


图 3-8 测试数据 0 1 分布图

通过分析图 3-7，我们发现在基于朴素贝叶斯的标准问获取算法下被误判的标准问所对应的概率值大部分集中在一定的范围值中。从图 3-8 的分析中，我们发现判断正确（标记为 1）的标准问和判断错误（标记为 0）的标准问所处的概率值范围也相近。综合图 3-7 和图 3-8，未能得出较好的结论。

但是，在通过控制用户问不变的情况下，两种获取标准问算法的表现在某些用户问下的表现却截然不同，如下图所示：

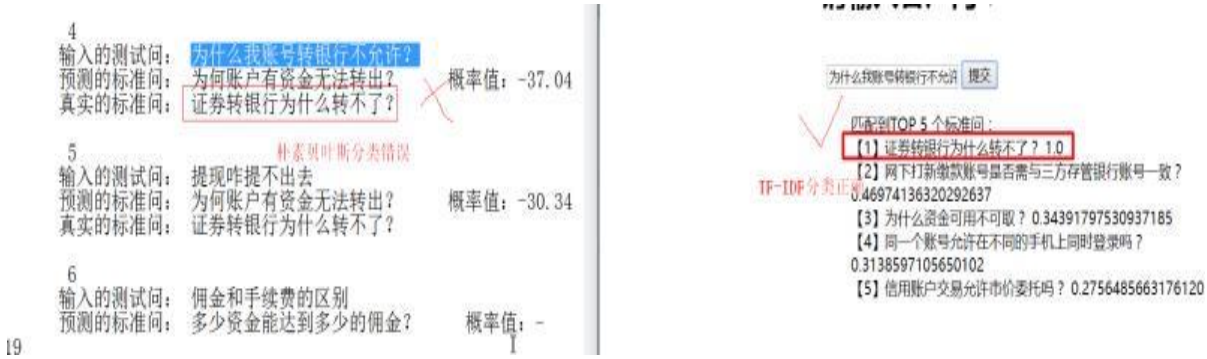


图 3-9 单模型算法测试用户问对比图

由图 3-9 可看出，虽然在整体 F1 值的表现中基于余弦相似获取标准问算法略逊一筹，但是在例如“为什么我账号转银行不允许？”这个用户问下，基于余弦相似度的获取标准问算法比基于 NB 的获取标准问算法判断的更加正确。经过多次试验，在两个算法输出的标准问 TOP5 列表中，我们发现其匹配相同的标准问但是在列表的排列顺序上却有所差异。如果用户问中存在与标准问过多的关键词，则基于余弦相似度的算法更能匹配出正确的标准问。

对此，我们设计了基于组合模型获取最终标准问列表算法，伪代码如下：

表 3-4 get_standard_question_bt_compose 伪代码符号说明表

符号	说明
get_standard_question_bt_compose	基于组合模型获取最终的标准问
sq_list_by_NB	基于朴素贝叶斯匹配到的 TOP5 标准问列表
sq_list_by_cosSim	基于余弦相似度匹配到的 TOP5 标准问列表
top5_list	最终匹配到的 TOP5 标准问列表
get_list(self)	获取模型产物
get_same_item(self)	获取相同的标准问
ranking(self)	重新排名
get_top5_list	获得最终匹配到的 TOP5 标准问列表

算法名称：get_standard_question_bt_compose

输入：sq_list_by_NB, sq_list_by_cosSim

输出：top5_list

（6） def get_list(self):

（7） def get_same_item(self):

（8） def ranking(self):

（9） def get_top5_list(self):

3.3.3 filter_by_IR 算法优化

在进行算法的组合后，我们模拟真实用户对整个问答系统进行人工自检测试。在此过程中，我们发现如果用户提出了一些“闲聊”式的提问，如“天气不错鸭”在之前版本的系统中则仍会匹配到一个标准问列表出来，而实际上此处应该转化为“闲聊”型机器人对话模式。为进一步提高用户体验，我们受到信息检索角度领域技术的启发，设计了基于信息检索角度过滤部分用户问。例如，当用户输入“天气不错鸭”时，我们提示将其归为“其他类型问题”并输出指定友好的语句，使得主要业务场景得以保留并提高了用户体验。

其中，filter_by_IR 算法核心步骤如下：

Step1：将当前标准问下的所有用户问“合并”成一篇“文章”。

Step2：对用户问进行数据清洗，提取出关键词。

Step3：计算用户问中所有关键词出现在所有“文章”中的 TF-IDF 的值。

Step4：若对应的 TF-IDF 值为 0 则判定为“闲聊”式提问直接归为其他类型问题。

四、后端部署

4. 1 模块功能要求

1) 与前端进行及时有效的数据传输

2) 与硬件进行及时有效的交互

3) 与算法模块进行数据间的交互

4) 算法模块部署在后端服务器上且能流畅的运行，在接收到用户问后能根据该用户问返回 TOP5 匹配的标准问列表或者返回其他提示信息。

4.2 后台管理系统模块

为了方便数据管理，使得整个系统更具扩展性和便于管理。我们设计并实现了一个简易的智能问答后台管理系统。其中后台管理系统部分功能模块如下：

图 4-1 后台管理系统登录页面

图 4-2 后台管理系统首页

图 4-3 数据管理页面

图 4-4 数据信息编辑页面

4.3 实现技术与部署流程

4.3.1 数据交互

软件整体设计前端后分离，所有的数据交互皆通过统一的 JSON 数据格式进行传输。

例如，前端 webAPP 通过 JQuery 发起 POST 请求，后端处理后得到 JSON 格式的数据结果。

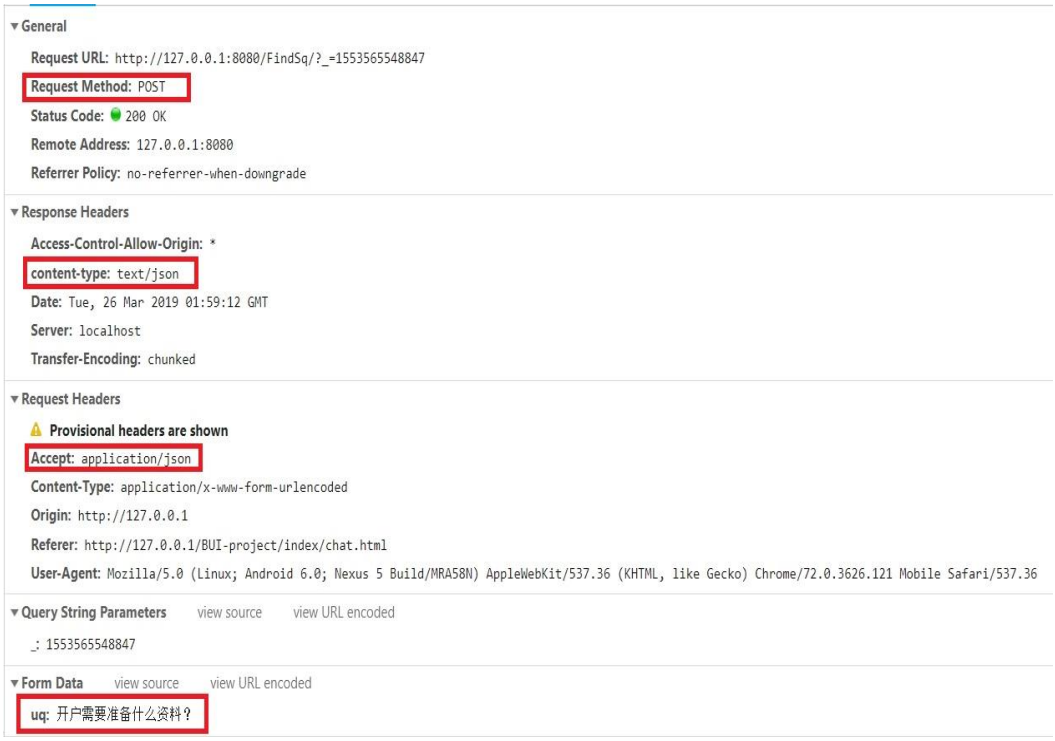


图 4-5 前端发起请求（本地调试）图

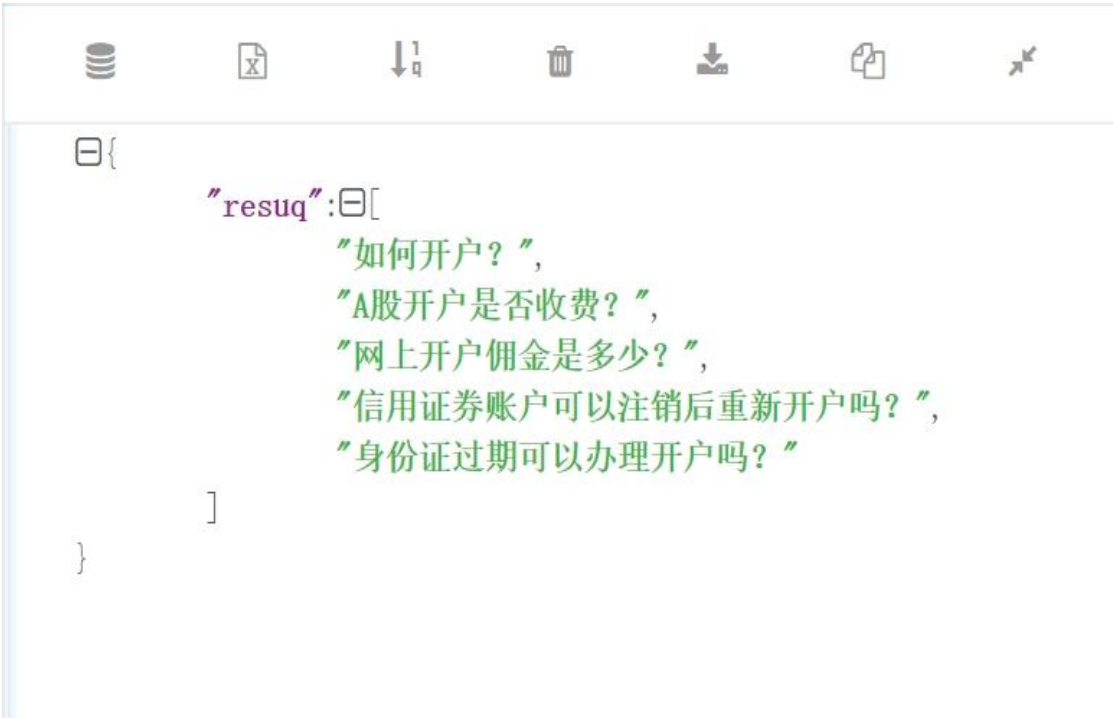


图 4-6 后端返回数据格式图

4.3.2 web 框架

在搭建后端服务时，我们采用了 web.py 这个框架。web.py 是一个轻量级 Python web 框架，它简单而且功能强大。web.py 是一个开源项目。该框架由已故美国作家、Reddit 联合创始人、RSS 规格合作创造者、著名计算机黑客 Aaron Swartz 开发。web.py 目前已被很多家大型网站所使用。当运行主文件时，该框架会自动调起服务，默认端口为 8080，监听所有发送过来的请求。

```
#URL映射
urls = (
    '/(.*)', 'FindSq'
)
app = web.application(urls, globals())

class FindSq:

    def POST(self, uq):
        # 跨域
        web.header("Access-Control-Allow-Origin", "*")
        web.header('content-type', 'text/json')
        data = web.input()
        user_question = data['uq']
        sq_list_by_cosSim = tfidf_cosin_main.main_get_top_n(user_question)
        nbp = N.NavieBayesPredict('./model.txt')
        sq_list_by_NB = nbp.predict(user_question)
        get_top5_list = get_standard_question_bt_compose(sq_list_by_cosSim, sq_list_by_NB)
        res_dic = {'resuq': get_top5_list}

        return json.dumps(res_dic)

if __name__ == "__main__":
    app.run()
```

图 4-7 web.py 处理请求部分代码截图

4.3.3 部署流程

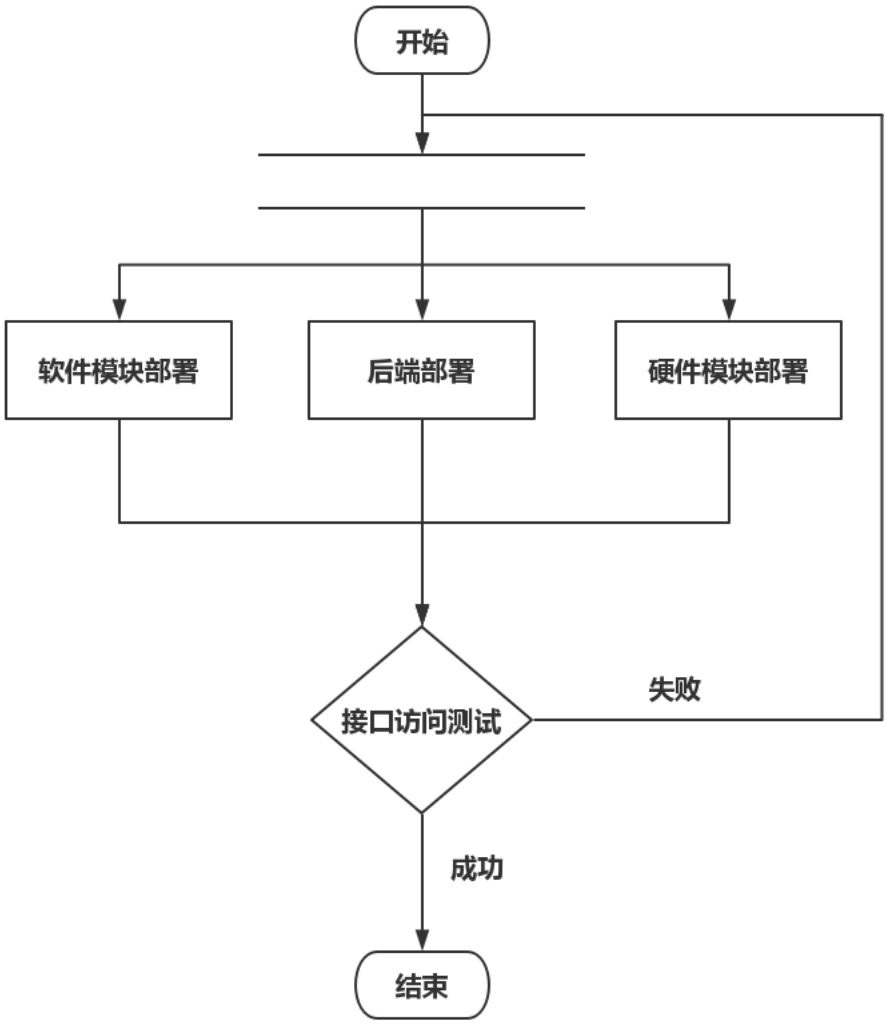


图 4-8 系统部署流程图

如图 4-8 所示，系统部署主要流程分为四大部分。

第一部分为软件模块部署，将 webAPP project 部署到 Apache 服务器上，并开启服务器，默认端口为 80 。

第二部分为后端部署，将核心算法模块与 web.py 搭建起来，并运行 web.py 主文件，默认端口为 8080，并使得该主文件一直运行着，不断的监听发送过来

的请求。

第三部分为硬件模块部署，将树莓派已经语音等模块进行组建并与服务器进行连调。

第四部分为测试接口，打开前端 webAPP 在客服页面输入用户问，然后点击“发送”测试后端服务是否正常运行。若存在异常，则根据异常信息重新部署；若能接收到后端返回的匹配到的标准问列表则表示部署成功。

五、前端设计

5.1 模块功能要求

- 登录后可在手机端的菜单中选择想听的内容
- 定制学习计划，到点播放
- 联网微聊，与孩子实时聊天

5.2 开发环境

表 5-1 开发环境表

操作系统	Windows10
开发工具	Sublime Text 3
开发模式	Web app
使用的框架	BUI（Build In User Interface）
使用的服务器	Apache HTTP Server

5.3 实现技术与流程

我们使用 Web app 为本次用户端的开发模式,与原生 APP 相比，其开发成本低，开发速度快，具有跨平台、统一性高的优势。开发使用了 BUI 框架，提供了丰富的 DPL 含有强大的控件库，灵活地开发了用户端使用的界面，且界面简洁美观，交互良好。

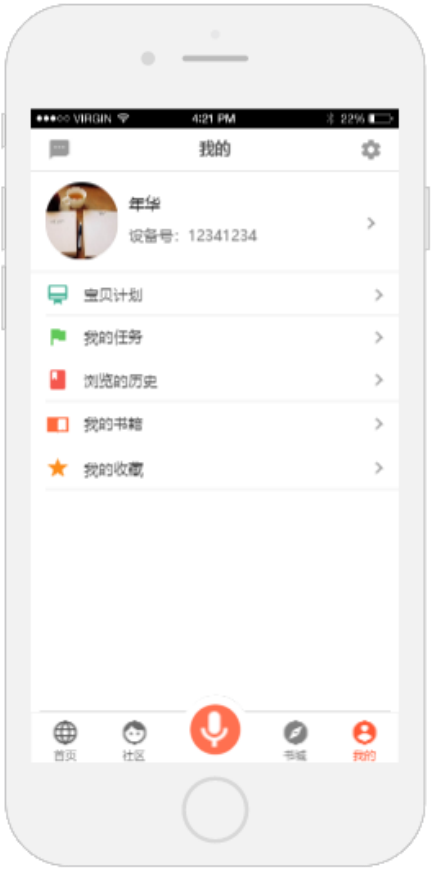
5.4 界面设计

本次界面设计，整体以蓝调为主，以简洁美观为主要特点，融合了扁平化设计，使得整体界面的观赏性较好。以下是部分前端界面的截图及介绍：

- 书城页面：在本页面用户可以为孩子挑选书籍进行播放，页面有分类推荐、热门推荐、精品推荐等，种类繁多、资源丰富。

- 计划页面：在本页面用户可以为孩子定制学习计划，为孩子添加每日学习行程，设置提醒时间，机器人将会准时提醒孩子并进行学习。

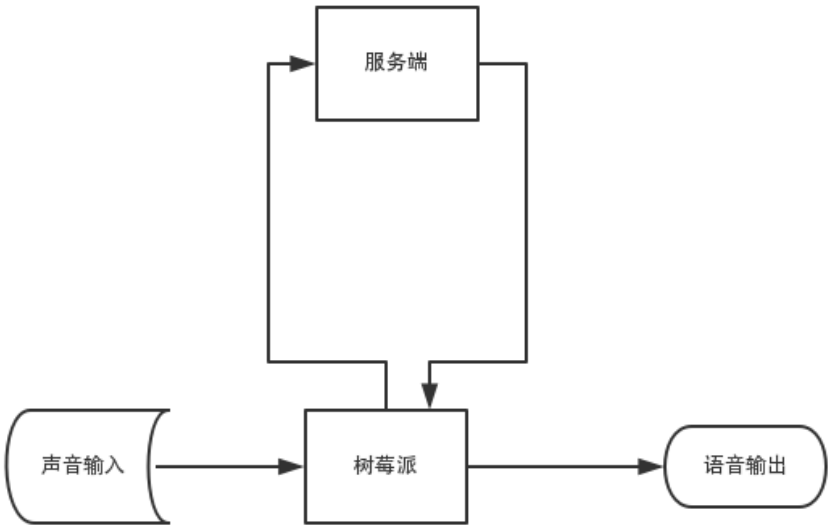
- 我的页面：在本页面用户可以为查看修改自己的个人信息，包括宝贝计划、我的任务、我的书籍、我的收藏等模块。



- 微聊页面：在本页面用户在联网的状态下，可通过机器人实时与孩子进行聊天，了解掌握孩子的动态。

六、硬件设计

6.1 硬件框架



6.2 硬件简介

1) 树莓派

Raspberry Pi(中文名为“树莓派”,简写为 RPi,(或者 RasPi / RPI) 是为学习计算机编程教育而设计),只有信用卡大小的微型电脑,其系统基于 Linux。随着 Windows 10 IoT 的发布，我们也将可以用上运行 Windows 的树莓派。

自问世以来，受众多计算机发烧友和创客的追捧，曾经一“派”难求。别看其外表“娇小”，内“心”却很强大，视频、音频等功能通通皆有，可谓是“麻雀虽小，五脏俱全”。

2) 硬件优势

它是一款基于 ARM 的微型电脑主板，以 SD/MicroSD 卡为内存硬盘，卡片主板周围有 1/2/4 个 USB 接口和一个 10/100 以太网接口（A 型没有网口），可连接键盘、鼠标和网线，同时拥有视频模拟信号的电视输出接口和 HDMI 高清视频输出接口，以上部件全部整合在一张仅比信用卡稍大的主板上，具备所有 PC 的基本功能只需接通电视机和键盘，就能执行如电子表格、文字处理、玩游戏、播放高清视频等诸多功能。 Raspberry Pi B 款只提供电脑板，无内存、电源、键盘、机箱或连线。在普通的单片机的比较下，具有更强的处理能力，以及更好的环境

兼容。

6.3 硬件需求

- 实现语音识别
- 实现与服务端交流
- 实现语音合成

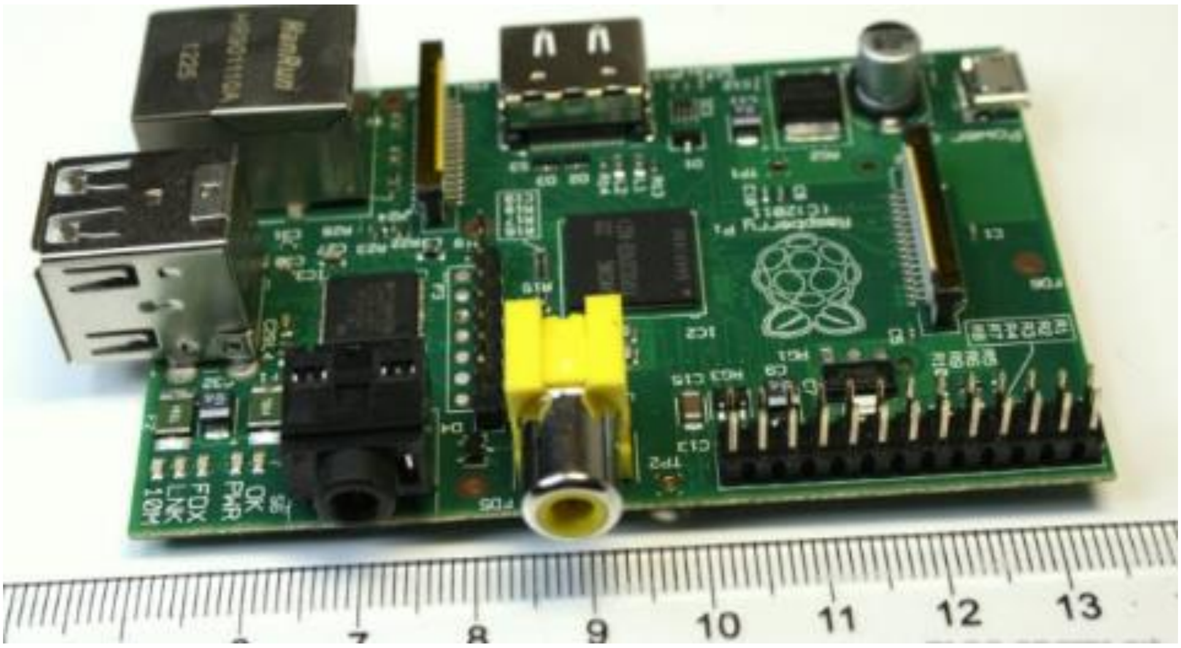
6.4 硬件开发环境

操作系统	linux
开发工具	python
开发模式	树莓派开发

6.5 实现技术与流程

在用户进行按键触发后进行语音录入，通过连接百度 ai 进行语音识别，将文本信息发至服务端进行处理。对服务端回调的文字信息进行语音合成，并且进行语音播放。

硬件核心模块展示：



树莓派展示

项目	Raspberry Pi B	Raspberry Pi B+	Raspberry Pi A+	Raspberry Pi 2 Model B	Raspberry Pi Zero	Raspberry Pi 3 Model B
发布时间	2011-12	2014-07-14	2014-11-11	2015-02-02	2015-11-26	2016-02-29
SoC	BCM2835	BCM2835	BCM2835	BCM2836	BCM2835	BCM2837
CPU	ARM1176JZF-S核心 700MHz 单核	ARM1176JZF-S核心 700MHz 单核	ARM1176JZF-S核心 700MHz 单核	ARM Cortex-A7 900MHz 四核	ARM1176JZF-S核心 700MHz 单核	ARM Cortex-A53 1.2GHz 四核
GPU	Broadcom VideoCore IV, OpenGL ES 2.0, 1080p 30 h.264/MPEG-4 AVC 高清解码器（本表格由树莓派实验室绘制 http://shumeipai.nxez.com 若有疏漏请联系我们更正）					
RAM	512MB	512MB	256MB	1GB	512MB	1GB
USB接口	USB2.0 × 2	USB2.0 × 4	USB2.0 × 1	USB2.0 × 4	Micro USB2.0 × 1	USB2.0 × 4
视频接口	RCA视频接口输出，支持PAL和NTSC制式，支持HDMI (1.3和1.4)，分辨率为640 x 350 至 1920 x 1200 支持PAL 和 NTSC制式。	支持PAL和NTSC制式，支持HDMI (1.3和1.4)，分辨率为640 x 350 至 1920 x 1200 支持PAL 和 NTSC制式。	支持PAL和NTSC制式，支持HDMI (1.3和1.4)，分辨率为640 x 350 至 1920 x 1200 支持PAL 和 NTSC制式。	支持PAL和NTSC制式，支持HDMI (1.3和1.4)，分辨率为640 x 350 至 1920 x 1200 支持PAL 和 NTSC制式。	支持PAL和NTSC制式，支持HDMI (1.3和1.4)，分辨率为640 x 350 至 1920 x 1200 支持PAL 和 NTSC制式。	支持PAL和NTSC制式，支持HDMI (1.3和1.4)，分辨率为640 x 350 至 1920 x 1200 支持PAL 和 NTSC制式。
音频接口	3.5mm 插孔，HDMI（高清晰度多音频/视频接口）	3.5mm 插孔，HDMI（高清晰度多音频/视频接口）	3.5mm 插孔，HDMI（高清晰度多音频/视频接口）	3.5mm 插孔，HDMI（高清晰度多音频/视频接口）	HDMI（高清晰度多音频/视频接口）	3.5mm 插孔，HDMI（高清晰度多音频/视频接口）
SD卡接口	标准SD卡接口	Micro SD卡接口	Micro SD卡接口	Micro SD卡接口	Micro SD卡接口	Micro SD卡接口
网络接口	10/100 以太网接口（RJ45接口）	10/100 以太网接口（RJ45接口）	无	10/100 以太网接口（RJ45接口）	无	10/100 以太网接口（RJ45接口），内置WiFi、蓝牙。
GPIO接口	26PIN	40PIN	40PIN	40PIN	40PIN	40PIN
额定功率	700毫安(为3.5W)	600毫安(为3.0W)	未知，但更低	1000毫安(为5.0W)	未知，但更低	未知，但更高
电源接口	MicroUSB 5V	MicroUSB 5V	MicroUSB 5V	MicroUSB 5V	MicroUSB 5V	MicroUSB 5V
尺寸	85.60 × 53.98 mm	85 x 56 x 17 mm	65 × 56 mm	85 x 56 x 17 mm	65 × 30 x 5 mm	85 x 56 x 17 mm
官方定价	35美元	35美元	20美元	35美元	5美元	35美元

参数对照表

6.6 部分源码

```
#相关库导入
from aip import AipSpeech
import pyaudio
import wave
import os
import sys
import time
import pygame
```

```
#init api
APP_ID='16035778'
API_KEY='Ezw6NH1GdbZvGiX6VQWtDTOn'
SECRET_KEY='GoFGEtNjXOW9BdnldUDgn6DtBFNzOwLe'
client = AipSpeech(APP_ID, API_KEY, SECRET_KEY)
```

```
#录音
def getvoice:
    CHUNK = 512
    FORMAT = pyaudio.paInt16
    CHANNELS = 1
    RATE = 44100
    RECORD_SECONDS = 5
    WAVE_OUTPUT_FILENAME = "output.wav"

    p = pyaudio.PyAudio()

    stream = p.open(format=FORMAT,
                    channels=CHANNELS,
                    rate=RATE,
                    input=True,
                    frames_per_buffer=CHUNK)

    print("recording...")

    frames = []

    for i in range(0, int(RATE / CHUNK * RECORD_SECONDS)):
        data = stream.read(CHUNK)
        frames.append(data)

    print("done")

    stream.stop_stream()
    stream.close()
    p.terminate()

    wf = wave.open(WAVE_OUTPUT_FILENAME, 'wb')
    wf.setnchannels(CHANNELS)
    wf.setsampwidth(p.get_sample_size(FORMAT))
    wf.setframerate(RATE)
    wf.writeframes(b''.join(frames))
    wf.close()
    os.close(sys.stderr.fileno())
```

```
# 语音识别
def get_file_content(filePath):
    with open(filePath, 'rb') as fp:
        return fp.read()

# 识别本地文件
client.asr(get_file_content('output.wav'), 'wav', 16000, {
    'dev_pid': 1536,
})
#语音合成
result  =client.synthesis('x', 'zh', 1, {'vol': 5,})

if not isinstance(result, dict):
    with open('auido.mp3','wb') as f:
        f.write(result)
#语音播放
def musice()
    pygame.init()
    pygame.mixer.init()
    file='Desktop\\auido.mp3'
    pygame.mixer.music.load(file)
    pygame.mixer.music.play()
```