哈尔滨工业大学（深圳）

大一年度项目结题报告

项目名称： 通知内容管理 App 的设计与实现

项目负责人： 王靳 **学号：** 220111012

联系电话： 15816870583 **电子邮箱：** 220111012@stu.hit.edu.cn

院系及专业： 计算机科学与技术学院

指导教师： 吴宇琳 **职称：** 助理教授

联系电话： 18126282493 **电子邮箱：** wuyulin@hit.edu.cn

院系及专业： 计算机科学与技术学院

填表日期： 2023 年 12 月 13 日

一、项目团队成员（包括项目负责人、按顺序）

姓名	性别	所在学院	学号	联系电话	本人签字
王靳	男	计算机科学与技术学院	220111012	15816870583	王靳
吴语诗	女	计算机科学与技术学院	220110928	15967167116	吴语诗
蔡德林	男	理学院	220810316	15919094899	蔡德林
邹悦	女	理学院	220810424	18820366233	邹悦

二、指导教师意见

团队工作认真，完成了项目所需相关工作，工作量充实，同意结题。

签　　名：吴宇琳

2023 年　12 月 18 日

三、项目专家组意见

组长签名：

年　　月　　日

四、项目成果

1、实现了 LayUI + Flask 的前端，LayUI 是一个 Web UI 包，提供了美观的前端界面设计。通过 LayUI 和 Flask 的结合，实现了用户友好的前端交互界面。

2、通过 SQLAlchemy 构架了 Notice 的 ORM 模型，实现了 app 与数据库（mysql/maridb）之间的交互。这样可以方便地对数据库进行操作，实现数据的存储和检索。

3、通过 Hugging Face 的 Transformers，进行了迁移学习，将一个二分类的模型修改输出层，使其支持 12 输出，实现了 12 分类。这样可以更加灵活地应用模型进行多类别的分类任务。

4、通过正则表达式，能够将通知中的日期进行正则匹配，并将其输出到 app 内置的日历中。这样可以方便用户查看通知中的时间信息，并进行日程安排。

5、通过 ajax 实现了网页与服务端的交互, 提升了用户体验, 使得前端界面更加流畅和便捷。这些为 app 的功能和性能提升提供了有力支持。

五、项目研究结题报告

1 课题背景

课题背景随着信息化发展，通知更多借助网络渠道。学校目前使用的飞书 App 仍存在重要通知被淹没、通知对象针对性不强、通知本身信息冗杂等问题。学长学姐开发的 App 不支持自动生成关于截止日期的提醒。大多数同学需要在群消息中反复寻找、查看同一条通知，时间利用效率低。项目计划设计并实现一个通知内容管理 App，实现学校通知精准分类、要点捕捉与简化、重要通知收藏与推荐、日程安排表个性化生成五项功能，希望服务于学院通知发布工作。

2 课题研究内容与方法

2.1 研究内容

本项目通过对“文本挖掘”的研究，利用相关算法将学院以大段文本形式呈现、信息糅合一体的通知抽象成一个个简单标签，可以实现对标签的分类，并且查询到相关的日期。此外，我们还实现了“文本摘要”，可以将大段冗余的通知总结成一段精简的通知。

2.2 实施方案

本次项目的实施方案分为三个阶段：基础知识的学习阶段、前端开发阶段和后端开发阶段。

2.2.1 第一阶段：基础学习

小组成员学习了相关知识：python 的基本用法，学习了基于 pytorch 的模型搭建与调试，学习了 transformers 的调试方法，学习了“迁移学习”的机器学习方法，学习了 css，javascript 语法，学习了 flask 框架、ajax 开发前端，学习了如何使用 SQLAlchemy。学习了 docker 部署等技巧，以及通过 git 进行版本的管理。

2.2.2 第二阶段：前端开发

在前端开发中，项目采用了非常简单易用的 LayUI 组件库。Layui 是一套免费的开源 WebUI 组件库，在 flask 中的 jinja2 模板中调用了 LayUI 的相关组件。通过 SQLAlchemy 创建了通知（notice）的 ORM 模型，这使得开发人员能够简单且安全地与本地数据库进行交互。下面是项目 server 的目录结构：

```
└──➤ tree
.
├── app.py
├── blueprints
│   └── notice.py
├── config.py
```

```
├── data_utils.py
├── exts.py
├── models.py
├── model
│   ├── config.json
│   ├── pytorch_model.bin
│   ├── special_tokens_map.json
│   ├── tokenizer_config.json
│   └── vocab.txt
├── statics
│   ├── image（文件夹，jinja2 模板中一些图片）
│   ├── js（文件夹，里面有 jQuery，被 layui、templates/*依赖）
│   └── layui（文件夹，web ui 组建库）
├── templates
│   ├── add.html
│   ├── base.html
│   ├── calendar.html
│   ├── search.html
│   └── categories.html
└── tokenizers_pegasus.py
```

下面是这个目录中的每个文件的介绍（根据拓扑顺序）：

- templates 文件夹里面存放的是 jinja2 模板，json 数据传递通过 ajax 实现
  - base.html 是其他 html 的底板，里面是整个网页的主题，可以在其他的 templates html 中看到有{% extends "base.html" %}这句话，也可以理解成面向对象的继承关系
  - add.html 是添加通知的前端页面，能够与我们的 server 进行交互
  - calender.html 用到了 layUI 的日历组件，能够展示日期与对应的通知
  - categories.html 是显示通知及其分类的页面
  - search.html 是搜索页面
- exts.py 只进行了一个简单的功能——打开数据库

- config.py 是与数据库交流的必要配置，帐号，密码，端口等。
- models.py 并不是人工智能的模型，而是与数据库交流的 ORM 模型
- data_utils.py 是 fengshen 模型库处理数据的辅助文件，被 tokenizer_pegasus.py 依赖。fengshen github 仓库的 README 中推荐我们将 data_utils.py 与 tokenizers_pegasus.py 这两个文件放到项目中。（fengshen 是 idea-ccnl 研究院针对中文训练的一系列大模型）
- tokenizers_pegasus.py 是 fengshen 模型库的 tokenizers。这个文件被 notice.py 依赖，在文本摘要的时候，用来将输入的文本编码。pegasus 就是一个专门用来处理文本摘要的模型。
- static 只是静态文件，简而言之就是——资源包。
- blueprints/notices.py 是用来处理通知的，里面有文本摘要、模糊搜索、添加通知、文本分类的方法。
- model 文件夹里存放着：字典 vocab.txt，超参数 tokenizer_config.json，special_tokens_map.json，config.json，以及训练好的模型 pytorch_model.bin。
- app.py 是 top_module，能路由网页、启动网页，调用了 config.py，exts.py，notice.py，templates 等文件。

2.2.3 第三阶段：后端开发

这就需要翻阅 server 之外的文件了。

```
(base) ┌──wangfiox@localhost ~/Documents/freshman_project/utils  <main*>
└──➤  tree
.
├── classification
│   ├── classification.ipynb
│   └── logs（文件夹，训练的日志）
├── data
│   ├── x_通知搜集.md （人工搜集的 400 条通知，有几个文件）
│   ├── combined_data.csv
│   ├── csv_into_db.ipynb
│   ├── dates_combined_data.csv
│   ├── dates_combined_data.ipynb
│   ├── dates_combined_data_summarized.csv
│   ├── extract_content_to_csv.ipynb
```

```
15. |   └──── test.csv
16. └──── summary
17.     ├──── data_utils.py
18.     ├──── summary.ipynb
19.     └──── tokenizers_pegasus.py
```

- summary 文件夹下，小组成员测试了“文本摘要”的效果
  - summary.ipynb 用于测试 IDEA-CCNL/Randeng-Pegasus-523M-Summary-Chinese-V1 模型的效果
  - tokenizers_pegasus.py 与 data_utils.py 在上文已经介绍过了
- data 文件夹中有：小组成员搜集的若干数据的原始文件，处理原始文件的脚本，将文本导入数据库的脚本
- classification 实现了“迁移学习”——将“二分类”转化为了“12 分类”。

2.3 具体功能的实现

为了明确将要操作的数据，让我们先来看看 ORM 模型：

```python
class NotificationModel(db.Model):
    __tablename__ = "notification_data"

    id = db.Column(db.Integer, primary_key=True, autoincrement=True)
    title = db.Column(db.String(200), nullable=False)
    content = db.Column(db.Text, nullable=False)
    # 提取日期
    date = db.Column(db.DateTime, nullable=True)
    # 摘要
    summary = db.Column(db.Text, nullable=True)
    created_date = db.Column(db.DateTime, default=datetime.now)
```

在接下来的介绍中，将以“自动识别”功能为核心，根据依赖关系递归地介绍具体功能的实现。

（1）“自动识别”（对应于前端的“自动识别”按钮）功能的实现：下面是该功能（方法）的调用关系图

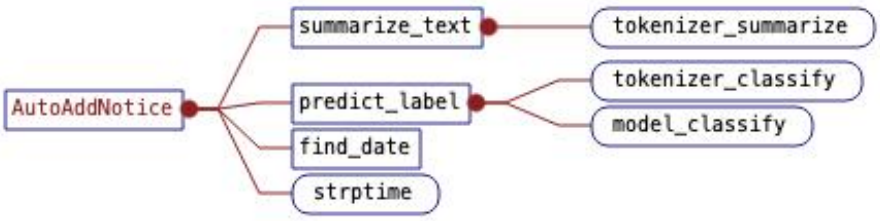

图 1“自动识别”的调用关系图

- summarize_text 方法用于“文本摘要”
- predict_label 方法用于“文本分类”
- find_date 方法用于正则的寻找日期
- strptime 方法用于将我们找到的日期，处理成“%Y 年%m 月%d 日”的格式

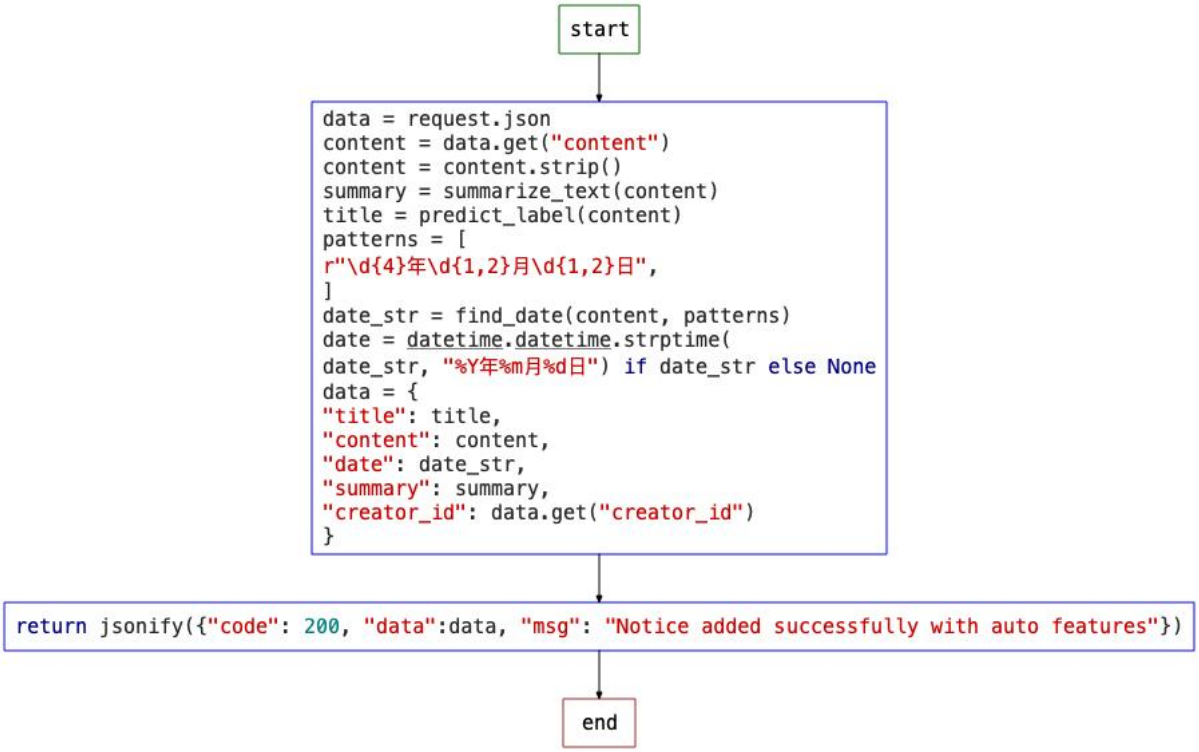

图 2“自动识别”的流程图

当点击“自动识别”后，AutoAddNotice 会调用 data.get（“content”）得到前端输入的文本。然后规范化文本-->文本摘要-->文本分类-->正则化匹配日期-->将数据封装成 json-->返回到前端。

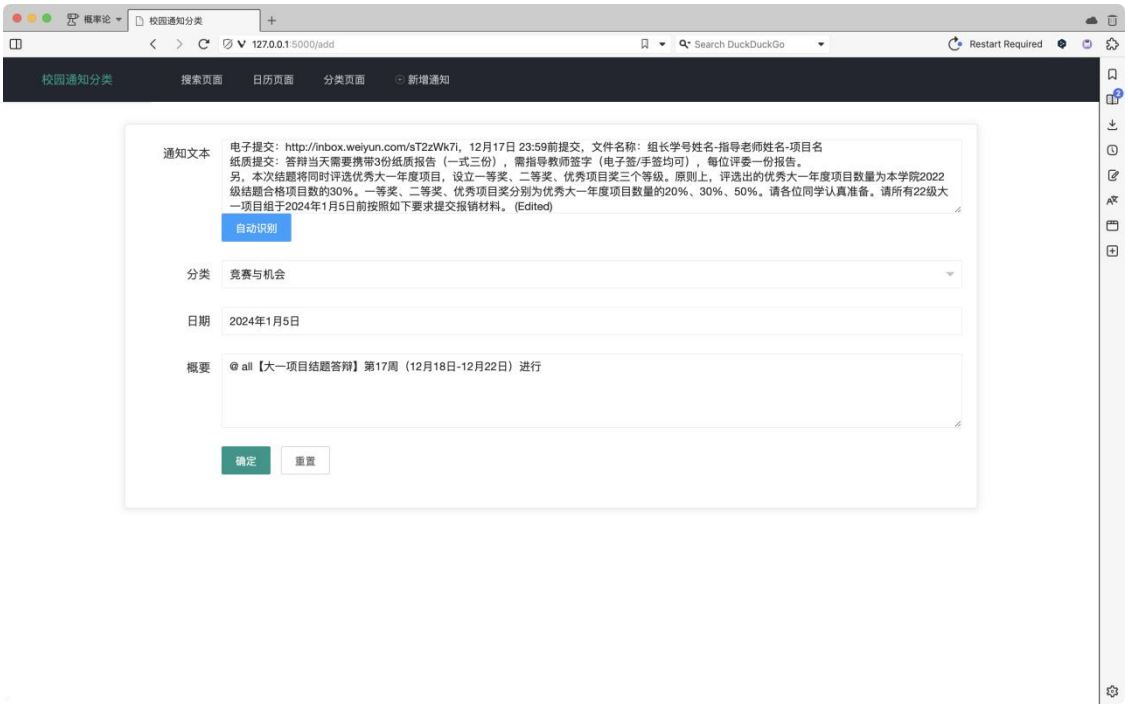

图 3“自动识别”效果展示

(这里的“分类”、“日期”、“摘要”是点击“自动识别”生成的)

(2) “文本摘要”的实现，下面是该功能的调用关系

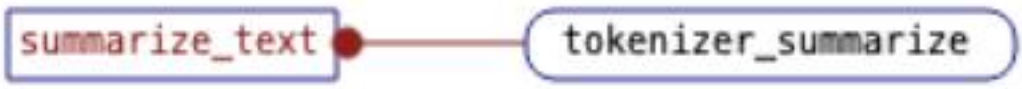

图 4“文本摘要”的调用关系

- tokenizers_summarize 方法由 IDEA-CCNL/Randeng-Pegasus-523M-Summary-Chinese-V1 模型的配套文件 tokenizers_pegasus.py 提供

图 5“文本摘要”的流程图

Pegasus 模型介绍：

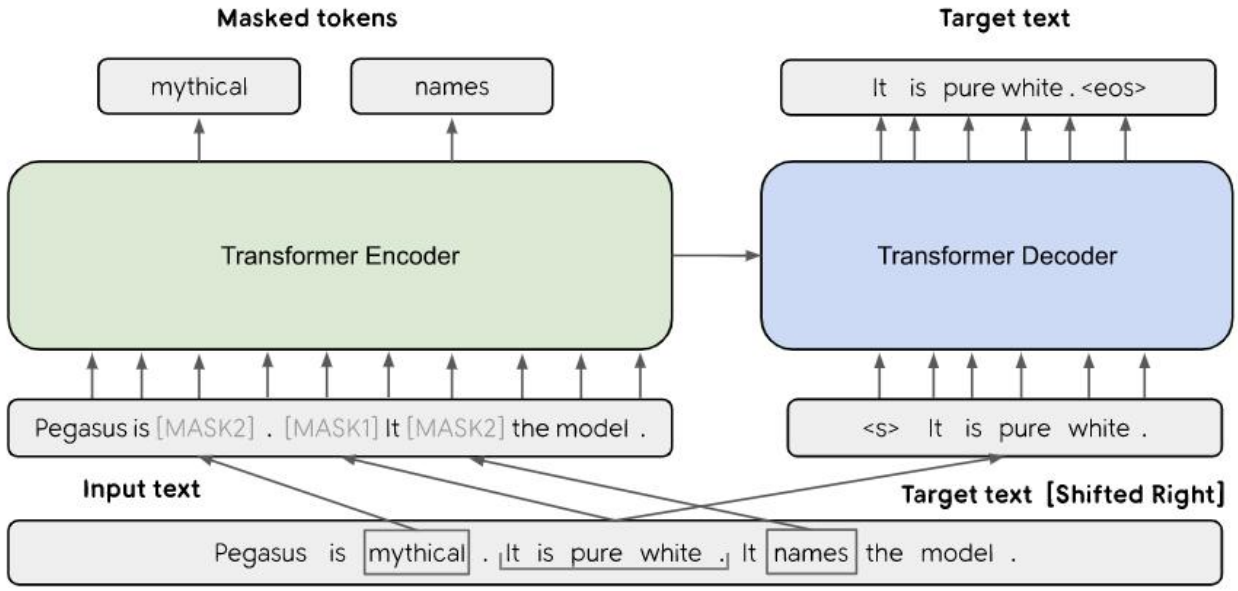

图 6 Pegasus 模型

Pegasus 模型是一种 seq2seq 模型，改进了 T5 与 BART 模型。T5 模型将不同掩码程度的文本进行训练。BART 在 T5 的基础上，引入了 denoising 的 autoencoder 层。Pegasus 在 BART 的基础上，不只是对词语进行掩码，还对整个句子剪切到 decoder 层的输入中。

（3）“文本分类”的实现，下面是该功能的调用关系

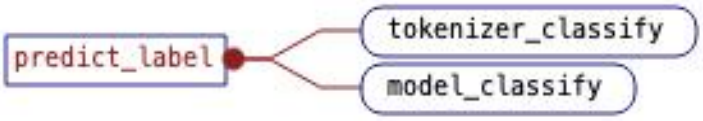

图 7“文本分类”的调用关系

- tokenizer_classify 用于将文本向量化，调用 model/vocab.txt 和 tokenizer_config.json
- model_classify 用于文本分类，调用我们训练好的模型

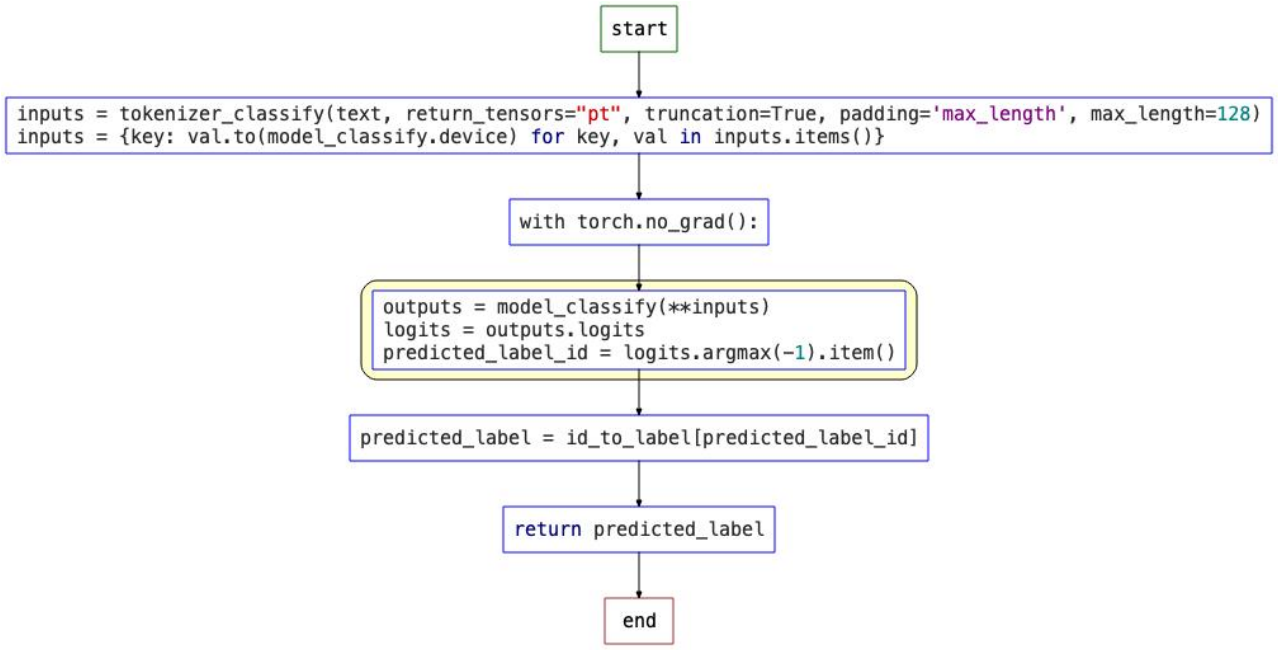

图 8 “文本分类”的流程图

首先对输入的文本进行分词处理，然后将向量转移到适配的设备上。如果 CUDA 可用，将数据转移到显存上，否则将在内存中进行处理。接下来，使用上下文管理器 with，在预测过程中关闭梯度计算。然后进行预测，并最终返回预测 ID 对应的文本标签。

(4) “文本分类”之“迁移学习”

“文本分类”利用了 RoBERTa 预训练模型。RoBERTa 模型是 Bert 模型的改进版：

1.动态掩码与静态掩码结合

2.使用了“没有 NSP loss 的 FULL-SENTENCES”的训练策略

3.Training with large batches

因为 RoBERTa 是对 BERT 模型的改进，下面介绍一下 BERT 模型。

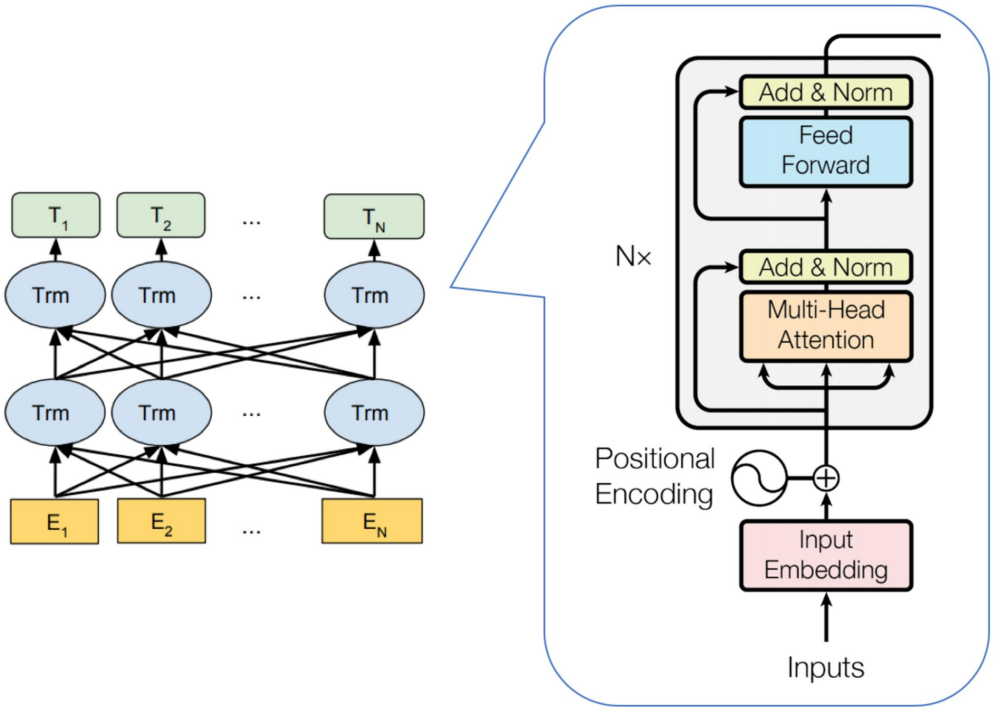

图 9 BERT 模型

我们可以看见，BERT 实际上是一个 encoder 层的叠加。最后如果我们想要输出一个分类结果，那么添加一个线性层和一个 softmax 激活层即可。本次项目的 RoBERTa 模型,层次结构如下：

```
1.  BertForSequenceClassification(
2.    (bert): BertModel(
3.      (embeddings): BertEmbeddings(
4.        (word_embeddings): Embedding(21128, 768, padding_idx=1)
5.        (position_embeddings): Embedding(512, 768)
6.        (token_type_embeddings): Embedding(2, 768)
7.        (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
8.        (dropout): Dropout(p=0.1, inplace=False)
9.      )
10.     (encoder): BertEncoder(
11.       (layer): ModuleList(
12.         (0-11): 12 x BertLayer(
13.           (attention): BertAttention(
14.             (self): BertSelfAttention(
15.               (query): Linear(in_features=768, out_features=768, bias=True)
16.               (key): Linear(in_features=768, out_features=768, bias=True)
17.               (value): Linear(in_features=768, out_features=768, bias=True)
18.               (dropout): Dropout(p=0.1, inplace=False)
```

```
            )
            (output): BertSelfOutput(
              (dense): Linear(in_features=768, out_features=768, bias=True)
              (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
          )
          (intermediate): BertIntermediate(
            (dense): Linear(in_features=768, out_features=3072, bias=True)
            (intermediate_act_fn): GELUActivation()
          )
          (output): BertOutput(
            (dense): Linear(in_features=3072, out_features=768, bias=True)
            (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
      )
    )
    (pooler): BertPooler(
      (dense): Linear(in_features=768, out_features=768, bias=True)
      (activation): Tanh()
    )
  )
  (dropout): Dropout(p=0.1, inplace=False)
  (classifier): Linear(in_features=768, out_features=12, bias=True)
)
```

(4.a) 测试预训练模型

```
# 使用预训练模型
model_name = 'IDEA-CCNL/Erlangshen-Roberta-110M-Sentiment'
tokenizer = BertTokenizer.from_pretrained(model_name)
model = BertForSequenceClassification.from_pretrained(model_name)  # 预训练模型
2.2.2.2、测试 二分类 结果
```

```
texta = '鲸鱼是哺乳动物，所有哺乳动物都是恒温动物'
textb = '鲸鱼也是恒温动物'
output = model(torch.tensor([tokenizer.encode(texta, textb)]))
print(torch.nn.functional.softmax(output.logits, dim=-1)) # 测试一下
```

OUTPUT：

```
tensor([[0.0645, 0.9355]], grad_fn=<SoftmaxBackward0>)
```

可以看到，第一句话模型认为的并不是很正确（但是实际上应该是对的）；第二句话模型认为是对的。

（4.b）数据预处理

加载数据，将文本 tokenize，将数据集划分为：训练集，测试集。因为预训练模型本身就比较大，就没有进行网格搜索，交叉验证，因此没有验证集。

```
from datasets import load_dataset, Features, Value

label_to_id = { # 分类
    "升学": 0,
    "志愿": 1,
    "教务": 2,
    "思政": 3,
    ... // 等等
}
# 将 label_to_id 进行反转
id_to_label = {value: key for key, value in label_to_id.items()}
print(id_to_label)

def preprocess_function(batch):
    # 对通知内容进行分词，并返回结果
    encoding = tokenizer(
        batch["通知内容"], truncation=True, padding="max_length", max_length=128
    ) # 分词，截断，填充
```

```
    encoding["labels"] = [
        label_to_id[label] for label in batch["类别"]
    ]  # 使用 label_to_id 将类别名转换为 ID
    return encoding

# 明确地定义 CSV 数据的特征描述
features = Features({"类别": Value("string"), "通知内容": Value("string")})

# 使用提供的特征描述加载数据集
dataset = load_dataset("csv", data_files="../data/combined_data.csv", features=features)

# 使用 map 函数进行预处理
encoded_dataset = (
    dataset["train"]
    .map(preprocess_function, batched=True)
    .train_test_split(test_size=0.05)
)

# 数据集的分割
train_dataset = encoded_dataset["train"]
test_dataset = encoded_dataset["test"]
```

(4.c) 修改输出层

```
# 修改模型输出
num_labels = len(label_to_id)
model = BertForSequenceClassification.from_pretrained(
    model_name, num_labels=num_labels, ignore_mismatched_sizes=True
)
```

```
# 打印模型的最后一层，验证是 12 分类
print(model.classifier)
```

OUTPUT：`Linear(in_features=768, out_features=12, bias=True)`

（4.d）激动人心的 trainer.train()

```
from transformers import Trainer, TrainingArguments

# 定义训练参数
training_args = TrainingArguments(
    output_dir="./results",
    evaluation_strategy="steps",
    eval_steps=10,
    per_device_train_batch_size=64,
    per_device_eval_batch_size=128,
    num_train_epochs=3,
    save_steps=50,
    logging_steps=20,
    learning_rate=2e-5,
    weight_decay=0.01,
    logging_dir="./logs",
    load_best_model_at_end=True,
)

# 创建 Trainer 对象
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=test_dataset,
```

```
    compute_metrics=None, # 如果你需要在验证时计算评估指标，请提供一个 compute_metrics 函数
)

# 启动！
trainer.train()
```

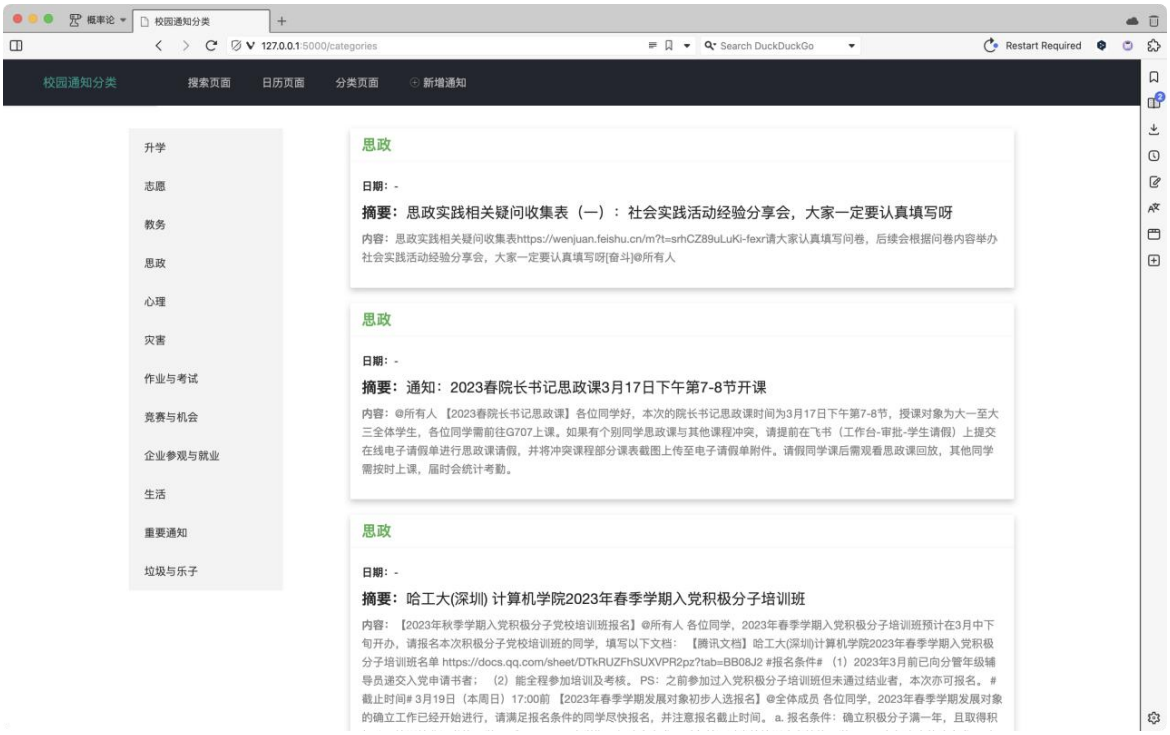

图 10“文本分类”效果展示

(5) 正则化搜索日期，下面是“正则化搜索日期”的流程图

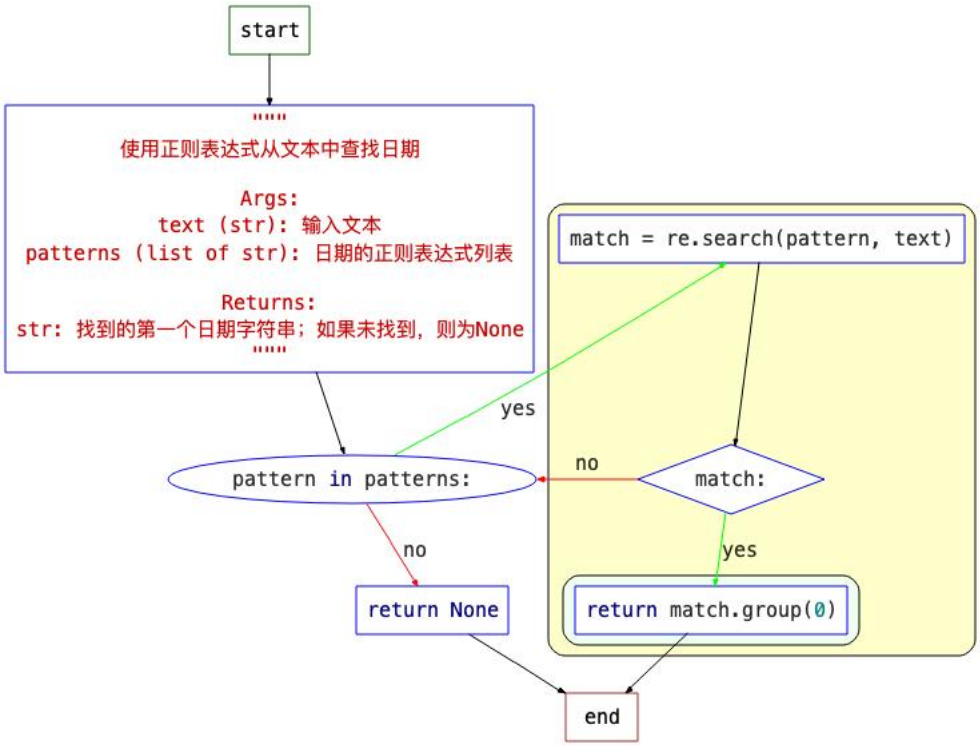

图 11“正则化搜索日期”的流程图

流程说明：find_date 函数接收 text 和 patterns 两个参数。patterns 是日期格式的列表，因为学院所推送的通知，其日期格式并非是一成不变的。在函数内部遍历所有的模式串，让其与传入的文本进行匹配。如果有匹配成功，返回第一个找到的日期；如果匹配失败，返回 NULL。

（6）模糊搜索，下面是“模糊搜素”的流程图

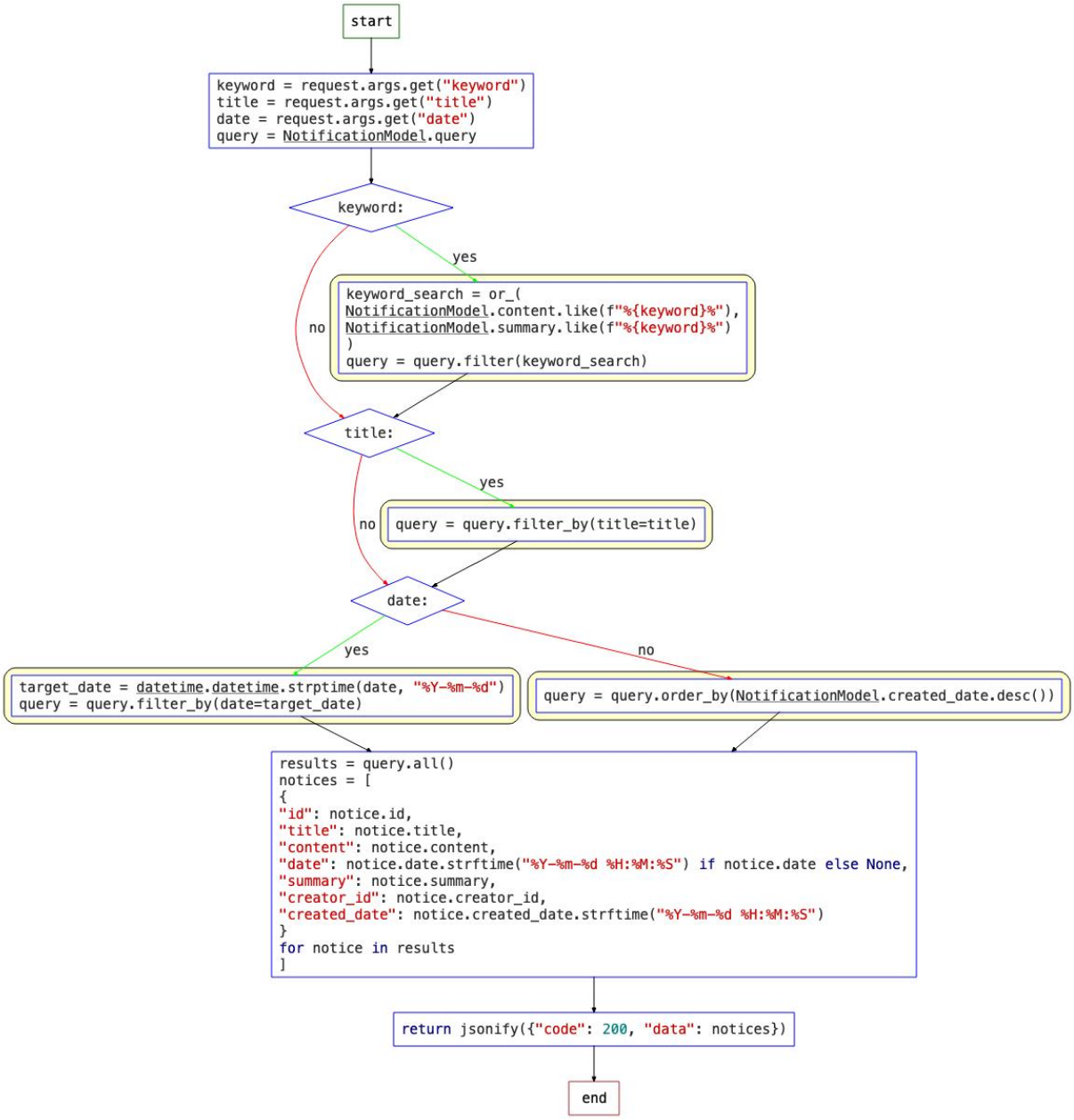

图 12 “模糊搜索” 的流程图

流程：先对输入的文本进行搜索，再是摘要的搜索，再是日期的搜索，这些搜索都是借助数据库完成。最后打包成 json，传到前端。

(7) 生成日程表的前后端代码

(7.a) 生成日程表的后端代码

```
1. @app.route('/calendar')
2. def calendar():
3.     # 使用模板插件，引入 index.html。此处会自动 Flask 模板文件目录寻找 index.html 文件。
```

```
4.      return render_template('calendar.html', name='calendar')
```

(7.b) 生成日程表的前端代码

```
1.  {% extends "base.html" %}
2.
3.  {% block content %}
4.  <div class="layui-container">
5.      <div class="layui-row">
6.          <!-- 左侧日期选择栏 -->
7.          <div class="layui-col-md3">
8.              <div>
9.                  <!-- Layui 日历组件 -->
10.                 <div class="layui-inline" id="date-picker"></div>
11.                 <br>
12.                 <br>
13.                 <!-- 无日期选项 -->
14.                 <div>
15.                     <button class="layui-btn layui-btn-primary" id="no-date">不限日期
    </button>
16.                 </div>
17.             </div>
18.         </div>
19.
20.         <!-- 通知卡片显示区域 -->
21.         <div class="layui-col-md9">
22.             <div id="notice-cards"></div>
23.         </div>
24.     </div>
25. </div>
26.
```

```
<script>
    layui.use(['jquery', 'element', 'laytpl', 'laydate'], function () {
        var $ = layui.jquery;
        var element = layui.element;
        var laytpl = layui.laytpl;
        var laydate = layui.laydate;

        // 初始化 Layui 日历组件
        laydate.render({
            elem: '#date-picker',
            position: 'static',
            done: function(value) {
                console.log(value);
                fetchNoticesByDate(value);
            }
        });


        // 无日期按钮点击事件
        $('#no-date').on('click', function () {
            fetchNoticesByDate(null);
        });

        function fetchNoticesByDate(date) {
            var url = date ? 'http://127.0.0.1:5000/Notice/Search?date=' + date : 'http://127.0.0.1:5000/Notice/Search';

            $.ajax({
                url: url,
                method: 'GET',
```

```
            success: function (data) {
                renderNotices(data.data); // 假设返回的数据中 notices 字段包含所有通知
            }
        });
    }

});
</script>

<style>

</style>

{% endblock %}
```

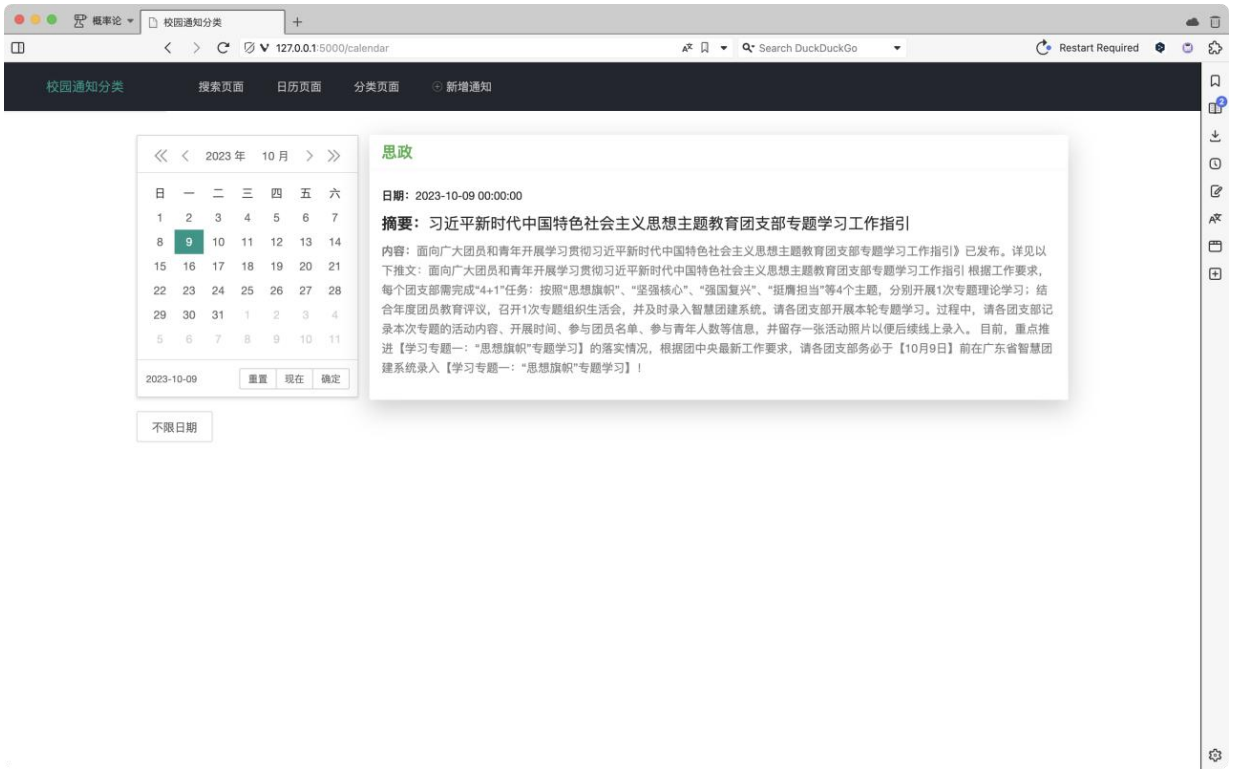

图 13 “日历”的效果展示

3 研究结果

小组成员通过深入学习和掌握基于 pytorch 的模型搭建与调试、flask 框架、ajax 开发前端等相关知识，成功实现通知内容管理 App，支持通知信息分类、信息精简、重点关注、模糊搜索、生成日程安排五项功能。

通知信息分类功能能够准确判别、分类通知，并打上相应的分类标签，使得用户可以更加有序地管理信息流。信息精简功能通过自动筛选和整理，精简提炼通知中重要的信息（如时间、地点）。为了满足用户的个性化需求，小组成员设计并成功实现了重点关注功能。用户可以根据个人需求设置重点关注标签，使得用户能够更为高效地获取相关信息。此外，模糊搜索功能使用户可以通过关键词模糊匹配的方式找到目标通知，从而更快捷地定位所需信息，极大地提高了信息检索的效率。最后，小组成员为 App 添加了生成日程安排的功能，用户可以根据通知信息生成日程安排，帮助用户更好地规划和安排自己的时间。

在实现 App 五项功能的基础上，开发过程注重前端设计的简洁美观，打造了一个用户友好且功能丰富的通知内容管理 App，为用户提供了一体化、高效率的信息管理解决方案。

4 创新点

通知内容管理 App 的信息分类、重点关注等功能，集成了飞书 App“pin 消息”功能（将消息固定置顶位置防止消息被淹没）和分类文档功能，在此基础上创新了信息精简、模糊搜索等功能，极大提升了对通知内有效信息等理解转化效率。

此外，学长学姐开发的《HITA 课表》、《HITsz 助手》等 App 仅支持查看每日课程安排等功能，通知内容管理 App 创新开发了生成日程安排的功能，解决了只能在日历上手动添加截止日期提醒的问题，提供了更为全面的信息管理服务，加强了它在生活中的实用性。

5 结束语

通过本次大一立项的研究与实践，小组成功地实现了通知内容管理 App 的设计与开发，为解决通知传达方面存在的问题提供了一种全新方案。在完成整个项目的过程中，我们深感信息化时代背景下对于高效通知管理的迫切需求，也认识到通过技术手段可以为用户提供更为便捷、个性化的服务。

作为计算机科学与技术学院和理学院的学生，小组通过深入学习和应用相关知识，成功实现了通知内容管理 App 的五项核心功能，即通知信息分类、信息精简、重点关注、模糊搜索和生成日程安排，为推动学院通知发布工作更为高效便利贡献了自己的力量。

此外，小组借鉴了飞书 App 的“pin 消息”和分类文档功能，引入信息精简、模糊搜索等创新功能，使得通知内容管理 App 在信息整理和检索方面表现更为出色。同时，小组也致力于解决其他校内 App 的局限性，创新性地添加生成日程安排的功能。这些创新结果为项目增色不少，也为今后进一步的优化和拓展提供了方向和思路。

在这一年的时间中，小组不仅学习掌握了技术知识，还培养了团队协作、问题解决和创新思维能力。在未来小组将不断学习和成长，继续关注并参与技术的发展，努力为社会提供更多创新且实用的解决方案。

感谢指导老师吴宇琳老师。她的耐心指导和深厚的学科知识提供了宝贵的学术引导，使小组在整个研究过程中受益匪浅。感谢开题答辩和中期答辩的老师们。老师们的严谨态度和指导性意见为项目的改进提供了有力的支持。之后小组将继续努力完善和优化此通知内容管理 App，希望能为广大师生提供更为便捷、高效的通知管理服务!

6 参考文献

[1] IDEA-CCNL. Randeng-Pegasus-523M-Summary-Chinese[CP/OL]. (2023-12-13). https://huggingface.co/IDEA-CCNL/Randeng-Pegasus-523M-Summary-Chinese

[2] IDEA-CCNL. Erlangshen-Roberta-110M-Sentiment[CP/OL]. (2023-12-13). https://huggingface.co/IDEA-CCNL/Erlangshen-Roberta-110M-Sentiment

[3] Flask[EB/OL]. (2023-12-13). https://flask.palletsprojects.com/en/3.0.x/

[4] Cornell University. PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization[EB/OL]. (2023-12-13). https://arxiv.org/abs/1912.08777

[5] Cornell University. RoBERTa: A Robustly Optimized BERT Pretraining Approach[EB/OL]. (2023-12-13). https://arxiv.org/abs/1907.11692

[6] IDEA-CCNL. Fengshenbang-LM[CP/OL]. (2023-12-13). https://github.com/IDEA-CCNL/Fengshenbang-LM

[7] WIKIPEDIA. Ajax_(programming). [EB/OL]. [2023-12-04] (2023-12-13). https://en.wikipedia.org/wiki/Ajax_(programming)

[8] Category: Ajax. [EB/OL]. (2023-12-13). https://api.jquery.com/category/ajax/

[9] revealjs[CP/OL]. (2023-12-13). https://revealjs.com

[10] The Python SQL Toolkit and Object Relational Mapper[CP/OL]. (2023-12-13). https://www.sqlalchemy.org

[11] Amazon.PyTorch[CP/OL]. (2023-12-13). https://pytorch.org

[12] LayUI[CP/OL]. (2023-12-13). https://layui.dev

[13] python[CP/OL]. (2023-12-13). https://www.python.org

[14] Cascading Style Sheets[CP/OL]. (2023-12-13). https://developer.mozilla.org/en-US/docs/Web/CSS

[15] javascript[CP/OL]. (2023-12-13). https://developer.mozilla.org/en-US/docs/Web/javascript

[16] node.js[CP/OL]. (2023-12-13). https://nodejs.org/en

[17] openSUSE[CP/OL]. (2023-12-13). https://www.opensuse.org

[18] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin. Attention Is All You Need[M/OL]. [2023-08-02] (2023-12-13). https://arxiv.org/abs/1706.03762

[19] Aston Zhang, Zack C. Lipton, Mu Li, Alex J. Smola. 动手学深度学习[M/OL]. (2023-12-13). https://zh-v2.d2l.ai

[20] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed,

Omer Levy, Ves Stoyanov, Luke Zettlemoyer. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension[EB/OL]. [2019-10-29] (2023-12-13). https://arxiv.org/abs/1910.13461

[21] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J. Liu.Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer[EB/OL]. [2023-09-19] (2023-12-13). https://arxiv.org/abs/1910.10683