**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：** wangfiox@hotmail.com

**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](mailto: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 |  |
| 吴语诗 | 女 | 计算机科学与技术学院 | 220110928 | 15967167116 |  |
| 蔡德林 | 男 | 理学院 | 220810316 | 15919094899 |  |
| 邹悦 | 女 | 理学院 | 220810424 | 18820366233 |  |
|  |  |  |  |  |  |

**二、指导教师意见**

|  |
| --- |
| 签 名：  年 月 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**（字数应在3000字左右）

**（一）主要内容：**

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

（我们是先有了后端，再有的前端的，但是报告顺序以及答辩顺序，我们认为可以先展示前端）

##### 2.2.1、第一阶段：基础学习

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。我们还学习了组员的协作，通过 git 进行版本的管理。

##### 2.2.3、第三阶段：前端开发

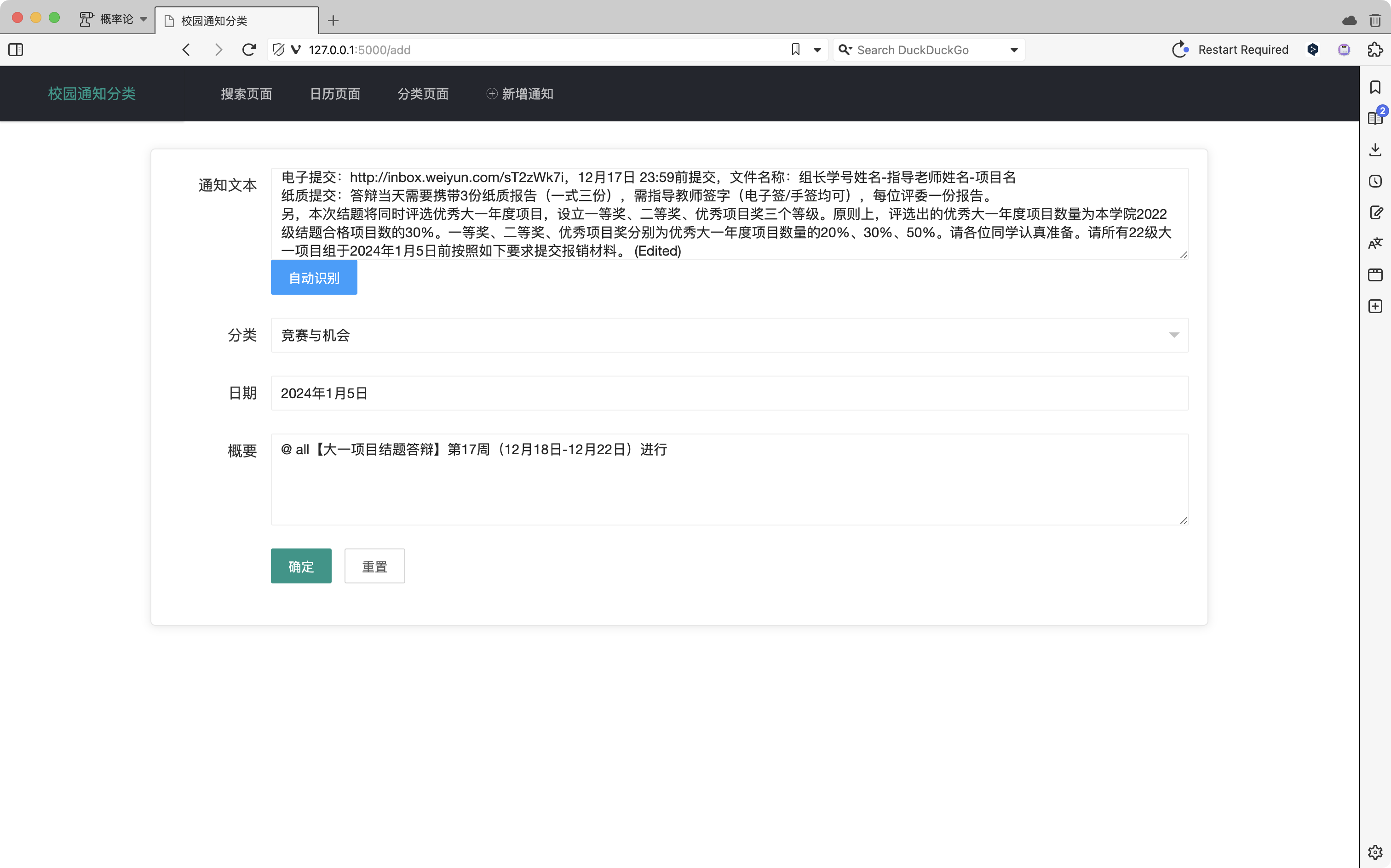
前端开发：我们使用了非常简单易用的 LayUI。Layui 是一套免费的开源 Web UI 组件库，我们在 flask 中的 jinja2 模板中调用了 LayUI。我们通过 SQLAlchemy，创建了通知（notice）的 ORM 模型，这能让我们简单且安全的与本地数据库进行交互。

下面是 server，也就是我们这个项目的主体的目录结构：

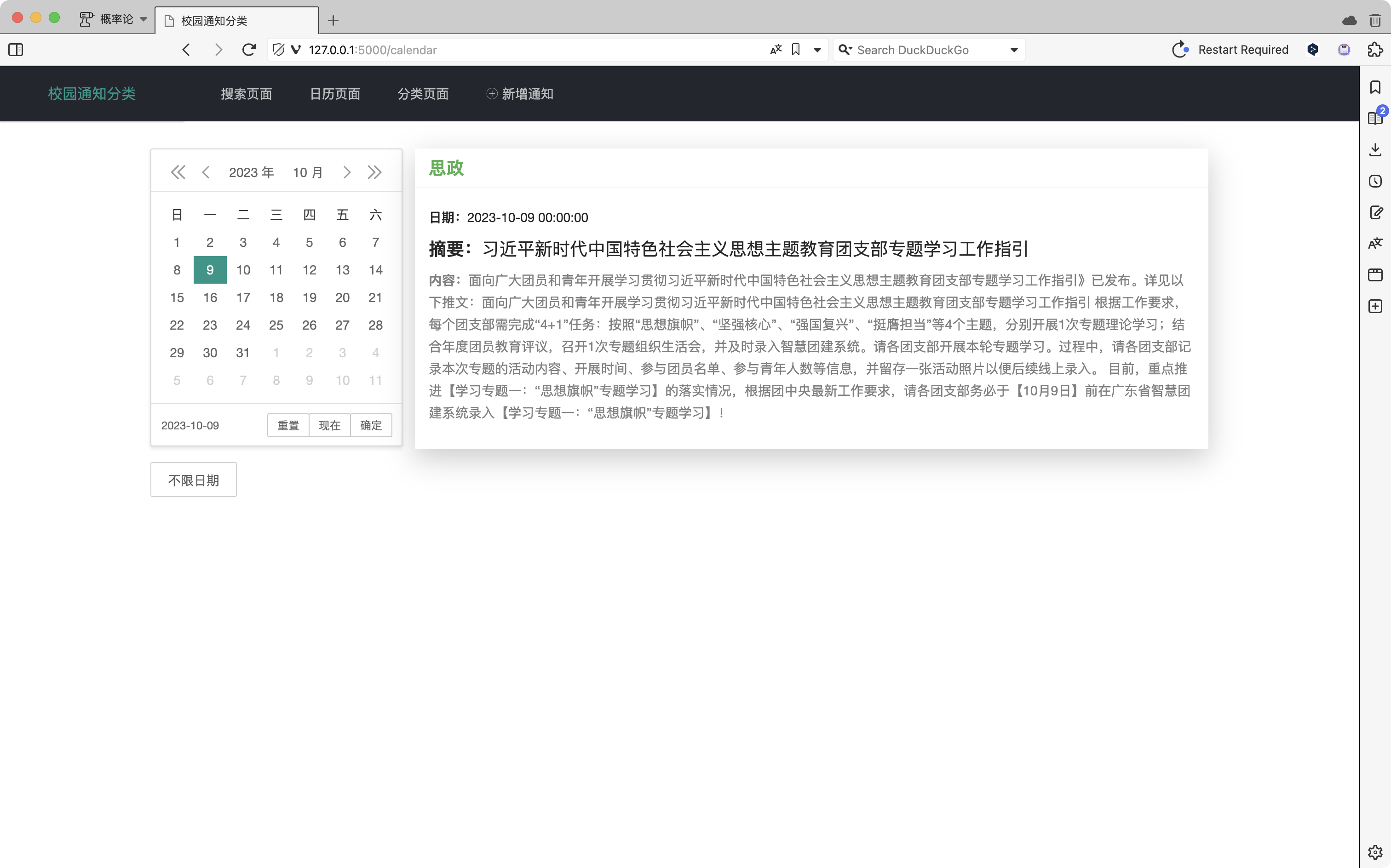
1. (base) ╭─wangfiox@localhost ~/Documents/freshman\_project  ‹main\*›
2. ╰─➤  tree
3. .
4. ├── app.py
5. ├── blueprints
6. │   └── notice.py
7. ├── config.py
8. ├── data\_utils.py
9. ├── exts.py
10. ├── models.py
11. ├── model
12. │   ├── config.json
13. │   ├── pytorch\_model.bin
14. │   ├── special\_tokens\_map.json
15. │   ├── tokenizer\_config.json
16. │   └── vocab.txt
17. ├── statics
18. │   ├── image（文件夹，jinja2 模板中一些图片）
19. │   ├── js（文件夹，里面有jQuery，被layui、templates/\*依赖）
20. │   └── layui（文件夹，web ui 组建库）
21. ├── templates
22. │   ├── add.html
23. │   ├── base.html
24. │   ├── calendar.html
25. │   ├── search.html
26. │   └── categories.html
27. ├── tokenizers\_pegasus.py

下面我将对这个目录中的每个文件进行介绍（根据拓扑顺序）：

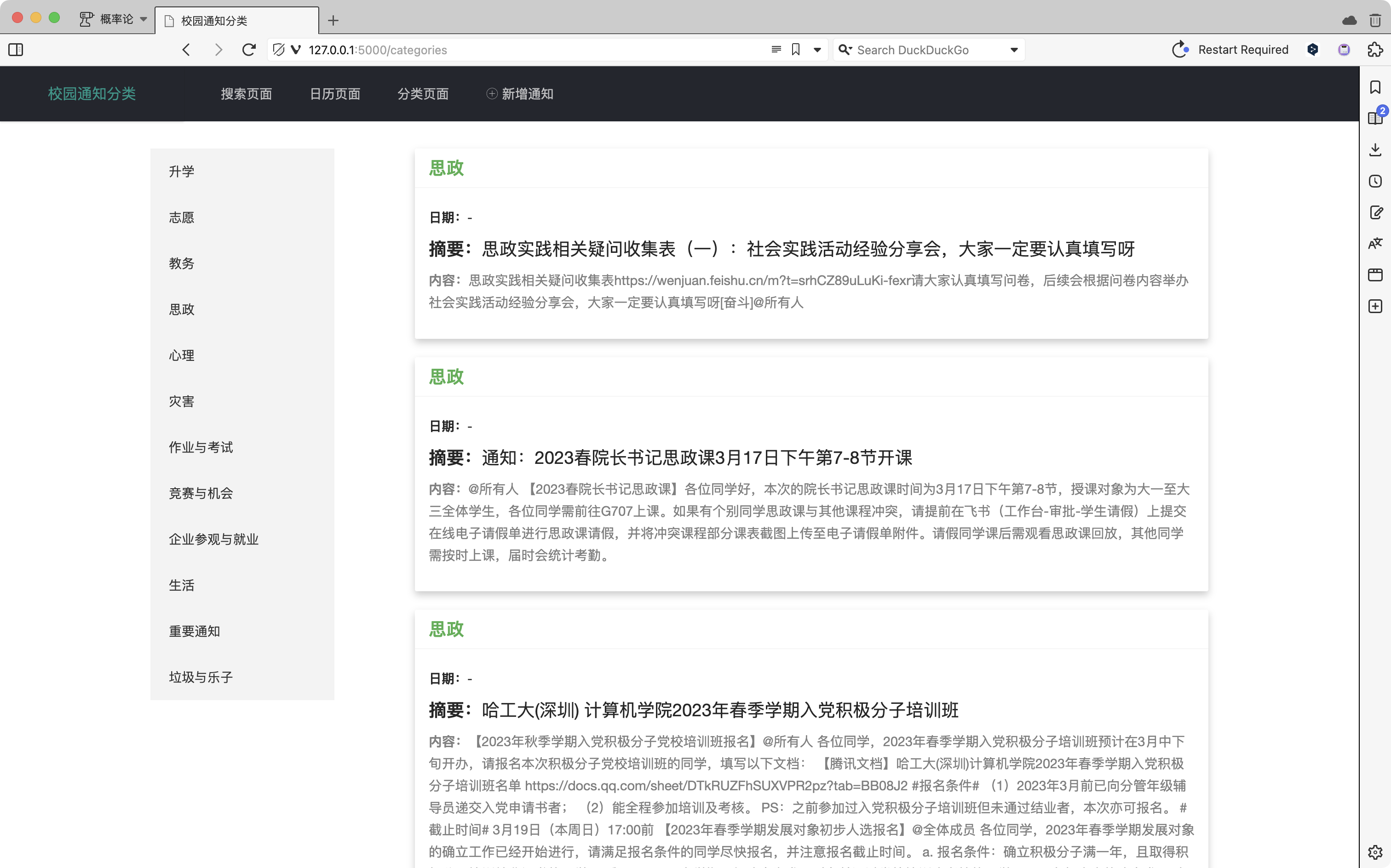
* templates 文件夹里面存放的就是我们的 jinja2 模板，json 数据传递通过 ajax 实现
  + base.html相当于是我们其他 html 的底板，里面是整个网页的主题，可以在其他的 templates html 中看到有{% extends "base.html" %}这句话，也可以理解成是继承的关系
  + add.html就是添加通知，并且能够与我们的 server 进行交互
  + calender.html就是用到了 layUI 的日历组件，日期与对应的 通知
  + categories.html就是我们可以看到分类与分类标签的地方（上图！）
  + search.html就是搜索页面
* 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,这里直接放到了项目中。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这个就是用来处理通知的，里面有文本摘要，模糊搜索，添加通知，文本分类的函数。
* model这个文件夹里面就存放着：字典vocab.txt，超参数tokenizer\_config.json，special\_tokens\_map.json，config.json，以及训练好的模型。
* app.py是top\_module，能路由网页等，启动网页，依赖config.py，exts.py，notice.py，templates



[add.html]



[calender.html]



[categories.html]

看了上面的介绍，可以看出，我们的最核心的文件是notice.py，下面一览notice.py的全貌

###### 2.2.3.1、输入文本，输出标签

1. def predict\_label(text):
2. *# 对文本进行编码*
3. inputs = tokenizer\_classify(text, return\_tensors="pt", truncation=True, padding='max\_length', max\_length=128)
4. *# 将输入移到模型所在的设备上*
5. inputs = {key: val.to(model\_classify.device) for key, val in inputs.items()}
6. *# 使用模型进行预测，这里不会进行训练*
7. with torch.no\_grad():
8. outputs = model\_classify(\*\*inputs)
9. logits = outputs.logits
10. predicted\_label\_id = logits.argmax(-1).item()
11. *# 获取预测的类别名*
12. predicted\_label = id\_to\_label[predicted\_label\_id]
13. return predicted\_label

###### 2.2.3.2、文本摘要

1. def summarize\_text(text, max\_length=128, num\_beams=4, length\_penalty=2.0, max\_length\_output=150):
2. """
3. 用输入的模型和分词器生成输入文本的摘要。
4. 参数：
5. - text (str): 要生成摘要的输入文本。
6. """
7. inputs = tokenizer\_summarize(text, max\_length=max\_length,
8. truncation=True, return\_tensors="pt")
9. *# beam search 生成摘要*
10. summary\_ids = model\_summarize.generate(
11. inputs["input\_ids"],
12. num\_beams=num\_beams,
13. length\_penalty=length\_penalty,  *# 惩罚系数是2.0*
14. max\_length=max\_length\_output,
15. no\_repeat\_ngram\_size=3  *# 生成文本时避免重复的n元组的大小*
16. )
17. return tokenizer\_summarize.batch\_decode(summary\_ids, skip\_special\_tokens=True, clean\_up\_tokenization\_spaces=False)[0]

###### 2.2.3.3、正则搜索日期

1. def find\_date(text, patterns):
2. """
3. 使用正则表达式从文本中查找日期
4. Args:
5. text (str): 输入文本
6. patterns (list of str): 日期的正则表达式列表
7. Returns:
8. str: 找到的第一个日期字符串；如果未找到，则为None
9. """
10. for pattern in patterns:
11. *# 通过正则表达式找日期*
12. match = re.search(pattern, text)
13. if match:
14. return match.group(0)
15. return None

###### 2.2.3.4、添加新通知（手动指定）

1. *# 添加新通知*
2. @bp.route("/Add", methods=["POST"])
3. def AddNotice():
4. data = request.json
5. *# 分类*
6. new\_notice = NotificationModel(
7. title=data.get("title"),
8. content=data.get("content"),
9. date=datetime.datetime.strptime(
10. data.get("date"), "%Y年%m月%d日") if data.get("date") else None,
11. summary=data.get("summary"),
12. creator\_id=data.get("creator\_id")
13. )
14. db.session.add(new\_notice)
15. db.session.commit()
16. data = {
17. "title": data.get("title"),
18. "content": data.get("content"),
19. "date": new\_notice.date,
20. "summary": data.get("summary"),
21. "creator\_id": data.get("creator\_id")
22. }
23. return jsonify({"code": 200, "data":data , "msg": "Notice added successfully"})

###### 2.2.3.5、模糊搜索

1. @bp.route("/Search", methods=["GET"])
2. def SearchNotice():
3. keyword = request.args.get("keyword")
4. title = request.args.get("title")
5. date = request.args.get("date")
6. query = NotificationModel.query
7. *# 模糊查询*
8. if keyword:
9. keyword\_search = or\_(
10. NotificationModel.content.like(f"%{keyword}%"),
11. NotificationModel.summary.like(f"%{keyword}%")
12. )
13. query = query.filter(keyword\_search)
14. *# 根据类别查询*
15. if title:
16. query = query.filter\_by(title=title)
17. *# 根据日期查询*
18. if date:
19. target\_date = datetime.datetime.strptime(date, "%Y-%m-%d")
20. query = query.filter\_by(date=target\_date)
21. else:
22. *# 按created\_date降序排列*
23. query = query.order\_by(NotificationModel.created\_date.desc())
24. results = query.all()
25. *# 搜索到的结果*
26. notices = [
27. {
28. "id": notice.id,
29. "title": notice.title,
30. "content": notice.content,
31. "date": notice.date.strftime("%Y-%m-%d %H:%M:%S") if notice.date else None,
32. "summary": notice.summary,
33. "creator\_id": notice.creator\_id,
34. "created\_date": notice.created\_date.strftime("%Y-%m-%d %H:%M:%S")
35. }
36. for notice in results
37. ]
38. return jsonify({"code": 200, "data": notices})

###### 2.2.3.6、自动生成标签，日期，摘要

1. @bp.route("/AutoAdd", methods=["POST"])
2. def AutoAddNotice():
3. data = request.json
4. content = data.get("content")
5. *# 去除前后空格*
6. content = content.strip()
7. *# 自动生成摘要*
8. summary = summarize\_text(content)
9. *# 自动生成类别标签*
10. title = predict\_label(content)
11. *# 自动提取日期*
12. patterns = [
13. r"\d{4}年\d{1,2}月\d{1,2}日",  *# YYYY年MM月DD日*
14. ]
15. date\_str = find\_date(content, patterns)
16. date = datetime.datetime.strptime(
17. date\_str, "%Y年%m月%d日") if date\_str else None
18. *# new\_notice = NotificationModel(*
19. *#     title=title,*
20. *#     content=content,*
21. *#     date=date,*
22. *#     summary=summary,*
23. *#     creator\_id=data.get("creator\_id")*
24. *# )*
25. data = {
26. "title": title,
27. "content": content,
28. "date": date\_str,
29. "summary": summary,
30. "creator\_id": data.get("creator\_id")
31. }
32. *# db.session.add(new\_notice)*
33. *# db.session.commit()*
34. return jsonify({"code": 200, "data":data, "msg": "Notice added successfully with auto features"})

##### 2.2.2、第二阶段：后端开发

这就要看我们server之外的文件们了：

1. (base) ╭─wangfiox@localhost ~/Documents/freshman\_project/utils  ‹main\*›
2. ╰─➤  tree
3. .
4. ├── classification
5. │   ├── classification.ipynb
6. │   └── logs（文件夹，训练的日志）
7. ├── data
8. │   ├── x\_通知搜集.md （人工搜集的400条通知，有几个文件）
9. │   ├── combined\_data.csv
10. │   ├── csv\_into\_db.ipynb
11. │   ├── dates\_combined\_data.csv
12. │   ├── dates\_combined\_data.ipynb
13. │   ├── dates\_combined\_data\_summarized.csv
14. │   ├── extract\_content\_to\_csv.ipynb
15. │   └── test.csv
16. └── summary
17. ├── data\_utils.py
18. ├── summary.ipynb
19. └── tokenizers\_pegasus.py

* summary，我们这里是直接 “拿来主义”，直接调用了IDEA-CCNL/Randeng-Pegasus-523M-Summary-Chinese-V1，并在 jupyter 中测试了效果
  + summary.ipynb，测试 “拿来主义” 的模型的效果
  + tokenizers\_pegasus.py 与 data\_utils.py 在上文中介绍了令人尴尬的效果（
* data里面有一些处理的脚本，我们是搜集了几个文件，一个是将文件合并，一个是将文本总结，一个是将文件导入数据库
* classification才是重头戏！！！下面介绍！！！

###### 2.2.2.1、使用预训练模型

1. *# 使用预训练模型*
2. model\_name = 'IDEA-CCNL/Erlangshen-Roberta-110M-Sentiment'
3. tokenizer = BertTokenizer.from\_pretrained(model\_name)
4. model = BertForSequenceClassification.from\_pretrained(model\_name)  *# 预训练模型*
5. 2.2.2.2、测试 二分类 结果
6. texta = '鲸鱼是哺乳动物，所有哺乳动物都是恒温动物'
7. textb = '鲸鱼也是恒温动物'
8. output = model(torch.tensor([tokenizer.encode(texta, textb)]))
9. print(torch.nn.functional.softmax(output.logits, dim=-1))  *# 测试一下*

OUTPUT:

1. 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。

###### 2.2.2.3、准备数据

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。

1. from datasets import load\_dataset, Features, Value
2. label\_to\_id = {  *# 分类*
3. "升学": 0,
4. "志愿": 1,
5. "教务": 2,
6. "思政": 3,
7. "心理": 4,
8. "灾害": 5,
9. "作业与考试": 6,
10. "竞赛与机会": 7,
11. "企业参观与就业": 8,
12. "生活": 9,
13. "重要通知": 10,
14. "垃圾与乐子": 11,
15. }
16. *# 将label\_to\_id进行反转*
17. id\_to\_label = {value: key for key, value in label\_to\_id.items()}
18. print(id\_to\_label)  *# 测试一下*
19. *# 明确地定义CSV数据的特征描述*
20. features = Features({
21. '类别': Value('string'),
22. '通知内容': Value('string')
23. })
24. *# 使用提供的特征描述加载数据集*
25. dataset = load\_dataset('csv', data\_files='../data/combined\_data.csv', features=features)
26. print(dataset)  *# 预览数据集*
27. *# 数据处理*
28. def preprocess\_function(batch):
29. *# 对通知内容进行分词，并返回结果*
30. encoding = tokenizer(batch['通知内容'], truncation=True, padding='max\_length', max\_length=128)  *# 分词，截断，填充*
31. 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*
32. return encoding
33. *# 使用map函数进行预处理*
34. encoded\_dataset = dataset['train'].map(preprocess\_function, batched=True).train\_test\_split(test\_size=0.05)
35. train\_dataset = encoded\_dataset['train']
36. test\_dataset = encoded\_dataset['test']
37. *# 输出训练集和测试集的大小*
38. print(len(train\_dataset))
39. print(len(test\_dataset))
40. *# 打印第一个样本的内容，带换行符*
41. print(train\_dataset[0])
42. *# 输出：*
43. *# {'类别': '生活', '通知内容': '各位同学@所有人 今天晚上收到多名同学反馈在教学楼、活动中心和宿舍楼附近发现卖笔的人员，请大家不要轻信和购买，保护好自身财产安全', 'input\_ids': [101, 1392, 855, 1398, 2110, 137, 2792, 3300, 782, 791, 1921, 3241, 677, 3119, 1168, 1914, 1399, 1398, 2110, 1353, 7668, 1762, 3136, 2110, 3517, 510, 3833, 1220, 704, 2552, 1469, 2162, 5650, 3517, 7353, 6818, 1355, 4385, 1297, 5011, 4638, 782, 1447, 8024, 6435, 1920, 2157, 679, 6206, 6768, 928, 1469, 6579, 743, 8024, 924, 2844, 1962, 5632, 6716, 6568, 772, 2128, 1059, 102, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], 'token\_type\_ids': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], 'attention\_mask': [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], 'labels': 9}*

###### 2.2.2.4、将 二分类 转化为 12 分类

1. *# 修改模型输出*
2. num\_labels = len(label\_to\_id)
3. model = BertForSequenceClassification.from\_pretrained(model\_name, num\_labels=num\_labels, ignore\_mismatched\_sizes=True)
4. *# 打印模型的最后一层，验证是12分类*
5. print(model.classifier)

OUTPUT:

1. Linear(in\_features=768, out\_features=12, bias=True)

###### 2.2.2.5、激动人心的 trainer.train()

1. from transformers import Trainer, TrainingArguments
2. *# 定义训练参数*
3. training\_args = TrainingArguments(
4. output\_dir='./results',
5. evaluation\_strategy="steps",
6. eval\_steps=10,
7. per\_device\_train\_batch\_size=64,
8. per\_device\_eval\_batch\_size=128,
9. num\_train\_epochs=3,
10. save\_steps=50,
11. logging\_steps=20,
12. learning\_rate=2e-5,
13. weight\_decay=0.01,
14. logging\_dir='./logs',
15. load\_best\_model\_at\_end=True,
16. )
17. *# 创建Trainer对象*
18. trainer = Trainer(
19. model=model,
20. args=training\_args,
21. train\_dataset=train\_dataset,
22. eval\_dataset=test\_dataset,
23. 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*
24. )
25. *# 开始训练*
26. trainer.train()

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] https://huggingface.co/IDEA-CCNL/Randeng-Pegasus-523M-Summary-Chinese 文本摘要 （pegasus）

[2] https://huggingface.co/IDEA-CCNL/Erlangshen-Roberta-110M-Sentiment 文本分类（二分类）

[3] https://flask.palletsprojects.com/en/3.0.x/ flask 文档

[4] https://arxiv.org/abs/1912.08777 PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization（pegasus 文本摘要预训练模型）

[5] https://arxiv.org/abs/1907.11692 RoBERTa: A Robustly Optimized BERT Pretraining Approach （预训练的 BERT 模型，用于分类）

[6] https://github.com/IDEA-CCNL/Fengshenbang-LM 封神榜大模型

[7] https://en.wikipedia.org/wiki/Ajax\_(programming) ajax wiki

[8] https://api.jquery.com/category/ajax/ ajax jquery

[9] https://revealjs.com reveal.js 通过html做ppt的node.js库

[10] https://www.sqlalchemy.org sqlalchemy

[11] https://pytorch.org pytorch

[12] https://layui.dev LayUI

[13] https://www.python.org python

[14] https://developer.mozilla.org/en-US/docs/Web/CSS css

[15] https://developer.mozilla.org/en-US/docs/Web/javascript javascript

[16] https://nodejs.org/en/download node.js（npm）

[17] https://www.opensuse.org opensuse（本项目在 opensuse 上训练的）

[18] https://arxiv.org/abs/1706.03762 attention is all you need

[19] https://zh-v2.d2l.ai 动手学深度学习