

****

信 息 学 院

《社交网络技术与应用》

期末大作业 项目报告

**题 目 智能医疗问答系统**

**组 员 学号22920225705728 姓名 栗斌**

**学号 姓名**

**学号 姓名**

**学号 姓名**

## 实验步骤

### 数据爬取

##### 爬虫代码

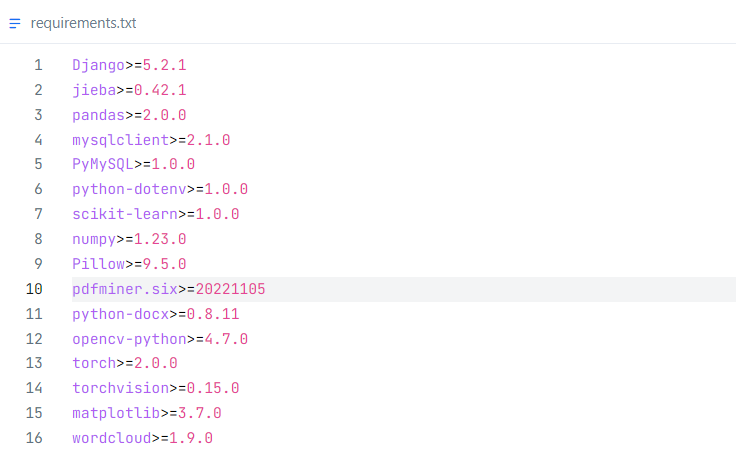
代码：

### 数据预处理与存储

##### 问答数据处理

* 在开始之前，需要安装项目所需的依赖库，通过 pip 进行安装：

|  |
| --- |
| pip install -r requirements.txt |

代码：

* 问答数据处理 - 数据预处理（中文分词、去停用词）与关键词提取  
  使用 jieba 库进行中文分词，并结合停用词表去除无意义的词汇，最后提取关键词

|  |
| --- |
| import pandas as pd  import jieba  import jieba.analyse  from .models import MedicalQA  from django.db import transaction  from pathlib import Path  from django.db.models import Q  from sklearn.feature\_extraction.text import TfidfVectorizer  from sklearn.metrics.pairwise import cosine\_similarity  import numpy as np  class DataProcessor:  def \_\_init\_\_(self):  self.data\_dir = Path(\_\_file\_\_).resolve().parent.parent / 'Data'  self.stopwords\_path = Path(\_\_file\_\_).resolve().parent.parent / 'static' / 'refs' / 'stopwords.txt'  self.departments = {  'Andriatria\_男科': '男科',  'IM\_内科': '内科',  'OAGD\_妇产科': '妇产科',  'Oncology\_肿瘤科': '肿瘤科',  'Pediatric\_儿科': '儿科',  'Surgical\_外科': '外科'  }  # 加载停用词  self.stopwords = self.load\_stopwords()  # 初始化TF-IDF向量化器  self.vectorizer = TfidfVectorizer()  # 缓存TF-IDF矩阵和问题列表  self.cached\_tfidf\_matrix = None  self.cached\_questions = None  self.cache\_size = 1000 # 缓存的问题数量限制  def load\_stopwords(self):  """加载停用词表"""  try:  with open(self.stopwords\_path, 'r', encoding='utf-8') as f:  return set([line.strip() for line in f])  except Exception as e:  print(f"加载停用词表时发生错误：{str(e)}")  return set()  def process\_text(self, text):  """文本预处理：分词、去停用词"""  if not isinstance(text, str):  return []  # 使用jieba分词  words = jieba.cut(text)  # 去除停用词  words = [word for word in words if word not in self.stopwords]  return words  def extract\_keywords(self, text, topK=10):  """提取关键词（去除停用词后）"""  # 使用jieba提取关键词，同时考虑停用词  keywords = jieba.analyse.extract\_tags(  text,  topK=topK,  withWeight=False,  allowPOS=('n', 'vn', 'v')  )  # 过滤停用词  keywords = [word for word in keywords if word not in self.stopwords]  return keywords |

* 问答数据处理 - 数据索引

提高后续搜索的效率，使用 TfidfVectorizer 对处理后的问题文本构建索引

|  |
| --- |
| def build\_index(self, questions):  """构建文本索引，使用缓存机制"""  # 如果问题列表与缓存的相同，直接返回缓存的矩阵  if (self.cached\_questions is not None and  len(questions) <= self.cache\_size and  questions == self.cached\_questions):  return self.cached\_tfidf\_matrix  # 将问题文本转换为TF-IDF向量  try:  # 限制问题数量  if len(questions) > self.cache\_size:  questions = questions[:self.cache\_size]    tfidf\_matrix = self.vectorizer.fit\_transform(questions)    # 更新缓存  self.cached\_tfidf\_matrix = tfidf\_matrix  self.cached\_questions = questions    return tfidf\_matrix  except Exception as e:  print(f"构建索引时发生错误：{str(e)}")  return None |

输出：



##### 数据存储

* 数据存储 - Mysql数据库的配置

|  |
| --- |
| DATABASES = {  'default': {  'ENGINE': 'django.db.backends.mysql',  'NAME': 'medintellect',  'USER': 'root',  'PASSWORD': 'root',  'HOST': 'localhost',  'PORT': '3306',  'OPTIONS': {  'connect\_timeout': 28800,  'read\_timeout': 28800,  'write\_timeout': 28800,  'init\_command': "SET sql\_mode='STRICT\_TRANS\_TABLES'",  'charset': 'utf8mb4',  }  }  } |

* 将处理后的数据存储进数据库设计好的表里

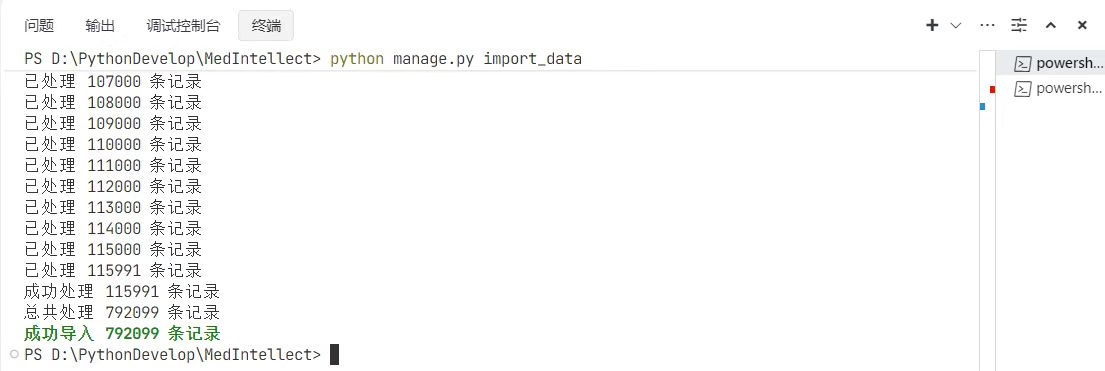
|  |
| --- |
| def process\_csv\_file(self, file\_path, department):  """处理单个CSV文件"""  try:  # 读取CSV文件  df = pd.read\_csv(file\_path, encoding='utf-8')    # 清理数据  df = df.fillna('')  for col in ['ask', 'answer', 'title']:  if col in df.columns:  df[col] = df[col].astype(str).apply(self.clean\_text)    # 确保数据框包含必要的列  required\_columns = ['department', 'title', 'ask', 'answer']  if not all(col in df.columns for col in required\_columns):  print(f"错误：{file\_path} 缺少必要的列（department, title, ask, answer）")  return 0  # 构建问题索引  questions = df['ask'].tolist()  tfidf\_matrix = self.build\_index(questions)  # 批量处理数据  batch\_size = 1000  total\_processed = 0  qa\_objects = []  for idx, row in df.iterrows():  try:  # 对问题进行分词和去停用词处理  processed\_text = self.process\_text(str(row['ask']))  # 提取关键词  keywords = self.extract\_keywords(str(row['ask']))  qa\_objects.append(MedicalQA(  title=str(row['title']) if 'title' in row else '',  question=str(row['ask']),  answer=str(row['answer']),  keywords=','.join(keywords),  department=department  ))  # 当达到批量大小时，执行批量插入  if len(qa\_objects) >= batch\_size:  with transaction.atomic():  MedicalQA.objects.bulk\_create(qa\_objects)  total\_processed += len(qa\_objects)  print(f'已处理 {total\_processed} 条记录')  qa\_objects = []  except Exception as e:  print(f'处理记录时发生错误：{str(e)}')  continue  # 保存剩余的记录  if qa\_objects:  try:  with transaction.atomic():  MedicalQA.objects.bulk\_create(qa\_objects)  total\_processed += len(qa\_objects)  print(f'已处理 {total\_processed} 条记录')  except Exception as e:  print(f'保存剩余记录时发生错误：{str(e)}')  return total\_processed  except Exception as e:  print(f"处理文件 {file\_path} 时发生错误：{str(e)}")  return 0  def process\_all\_data(self):  """处理所有数据文件"""  total\_processed = 0  for dept\_dir, dept\_name in self.departments.items():  dept\_path = self.data\_dir / dept\_dir  if not dept\_path.exists():  print(f"警告：目录 {dept\_path} 不存在")  continue  # 处理该科室下的所有CSV文件  for csv\_file in dept\_path.glob('\*.csv'):  print(f"正在处理 {csv\_file}...")  count = self.process\_csv\_file(csv\_file, dept\_name)  total\_processed += count  print(f"成功处理 {count} 条记录")  print(f"总共处理 {total\_processed} 条记录")  return total\_processed  @staticmethod  def clean\_text(text):  """清理文本数据"""  if pd.isna(text) or not isinstance(text, str):  return ""  # 移除特殊字符和多余的空格  text = text.strip()  # 移除零宽字符  text = text.replace('\u200b', '')  # 移除重复的空格  text = ' '.join(text.split())  return text |

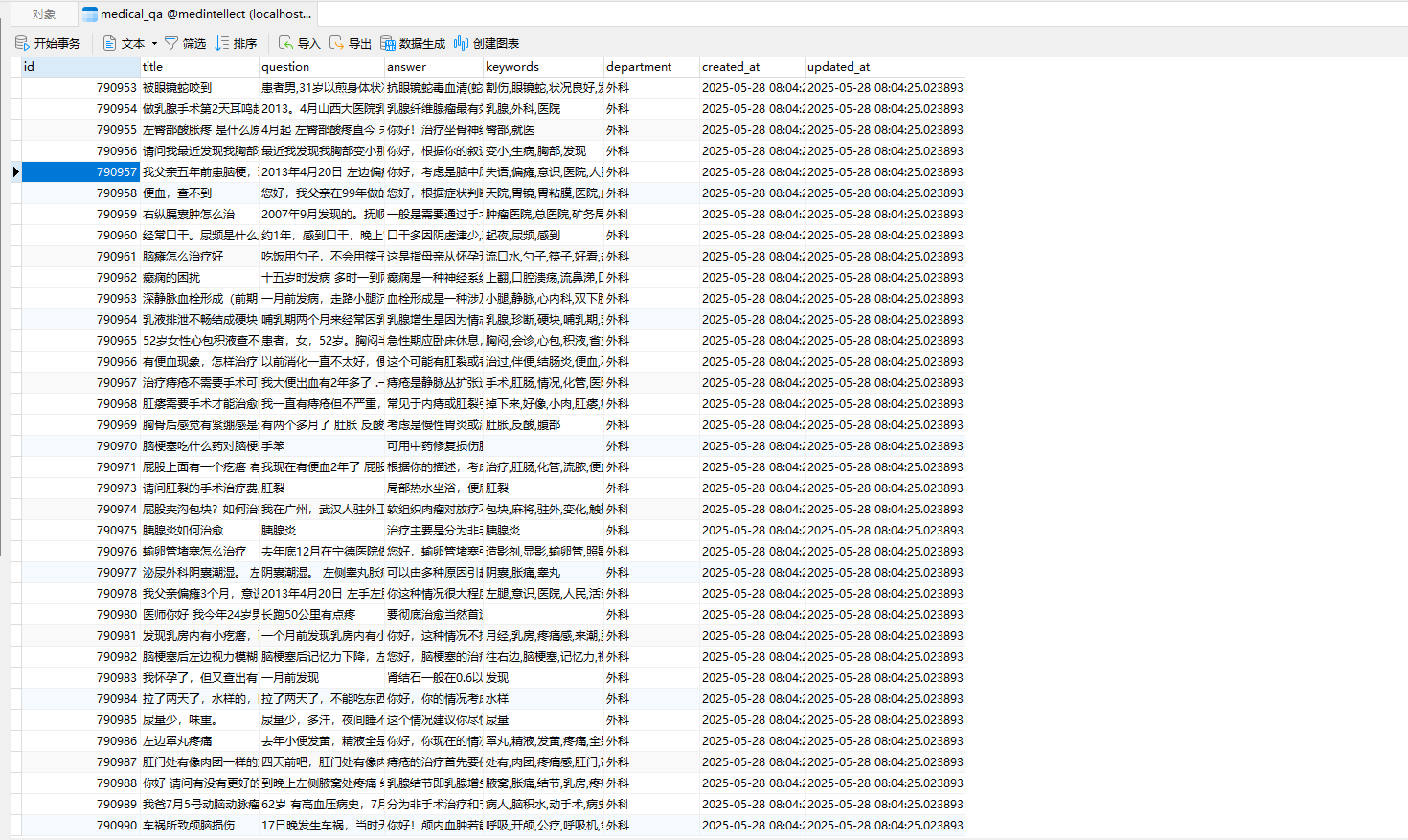
* 为了方便执行数据导入，创建一个管理命令 import\_data

|  |
| --- |
| from django.core.management.base import BaseCommand  from core.data\_processor import DataProcessor  class Command(BaseCommand):  help = '从CSV文件导入医疗问答数据到数据库'  def handle(self, \*args, \*\*options):  processor = DataProcessor()    self.stdout.write(self.style.SUCCESS('开始导入数据...'))    try:  total\_processed = processor.process\_all\_data()  self.stdout.write(  self.style.SUCCESS(f'成功导入 {total\_processed} 条记录')  )  except Exception as e:  self.stdout.write(  self.style.ERROR(f'导入数据时发生错误：{str(e)}')  ) |

输出：

* 通过执行 python manage.py import\_data 命令，程序会自动读取 Data 目录下的 CSV 文件，对数据进行处理并存储到数据库中





* 去除大部分无效数据  
  