“小开同学”问答系统

计算机科学与技术专业 1711436 皮春莹

2019 年 12 月 14 日

目录

1 实验要求

1) 数据来源：南开校内网页

2) 智能问答：模型不限、开发平台及语言不限

3) 问答示例：

- 你：小开同学，南开大学成立于哪一年？
- 小开同学：1919 年
- 你：小开同学，魔小龙是哪个部门的？
- 小开同学：计算机科学与技术系

2 实验环境

1) 语言：Python 3.6

2) 编译器：JetBrains PyCharm Community Edition 2019.1.3

3) 爬虫系统主要用到的插件：

(a) beautifulSoup：用来解析 HTML 文本，可以获取爬到的 HTML 文本中的各个项的属性。

(b) selenium：可以模拟浏览器获取异步数据或者执行点击操作。有一些网站的部分内容是用 JQuery 或者 Ajax 异步加载的，用简单的 Get 方式并不能获取到所有想要抓到的内容。并且爬取过程中需要通过搜索递归的方式，需要实现模拟浏览器的点击操作。

4) 索引和检索主要用到的模块和插件：

(a) sklearn：Scikit-learn(sklearn) 是机器学习中常用的第三方模块，对常用的机器学习方法进行了封装，包括回归 (Regression)、降维 (Dimensionality Reduction)、分类 (Classfication)、聚类 (Clustering) 等方法。有关特征的提取，scikit-learn 给出了很多方法，具体分成了图片特征提取和文本特征提取。文本特征提取的接口是 sklearn.feature_extraction.text。实验中中用到了文本特征提取中提供的 CountVectorizer、TfidfTransformer 和 TfidfVectorizer：

- CountVectorizer 的功能是将文本文档集合转换为计数的稀疏矩阵。内部的实现方法为调用 scipy.sparse.csr_matrix 模块。并且，如果在调用 CountVectorizer() 时不提供先验词典并且不使用执行某种特征选择的分析器，则特征词的数量将等于通过该方法直接分析数据找到的词汇量。
- TfidfTransformer 的功能是功能是将计数矩阵，转换为标准化的 tf 或 tf-idf 表示。Tf 表示术语频率，而 tf-idf 表示术语频率乘以逆文档频率。这是信息检索中常用的术语加权方案，在文档分类中也有很好的用途。用于计算项的 tf-idf 的公式是 tf-idf（d，t）= tf（t）* idf（d，t）。
- TfidfVectorizer：相当于两者的结合使用，先后调用 CountVectorizer 和 TfidfTransformer 两种方法（简化了代码，但运算思想还是不变），并且参数的使用基本一致。

(b) jieba：中文分词包，建立索引的过程中需要对中文进行分词。

5) 可视化及语音主要用到的插件：

(a) tkinter：Tkinter 模块 (Tk 接口) 是 Python 的标准 Tk GUI 工具包的接口。

(b) AipSpeech：百度语音 API 接口，在本次实验中用于合成语音。

(c) playsound：是 Windows 用于播放音乐的 API 函数，在本次实验中用于播放搜索结果合成的音频。

3 实现思路

3.1 数据爬取

本次实验要求的是爬取南开内网数据，我计划让小开同学可以回答多个学院的教师相关信息，以及实验室概况，因此，在网页抓取子系统中，我爬取了多个南开各单位教师网页，收集教师信息，主要包括：姓名、职称、邮箱、电话、专业、研究方向、学院、个人简介等，还收集了多个学院的概况。由 scrapyData.py 实现。

使用 selenium 中的 webdriver 模拟火狐浏览器执行点击操作，根据指定的 URL 获取页面内容，存放在 browser 变量中，使用 BeautifulSoup 解析 HTML 文本，剖析器为 html.parser，根据网页的结构，找到需要信息的所在位置，调用 select() 函数取出信息。也可以使用 browser.find_element_by_class_name() 方法，取出信息。在爬取信息时有几个需要特殊处理的地方：

1) 不同的学院老师的信息类别不同，有的信息可能并不展示，所以需要使用 try-catch 结构，将没有找到的项置为空。

2) 由于个人简介的信息太长，并且不是关键信息，没必要全部存储，所以只取了前 200 个字。

将爬虫得到的数据以 json 格式分块存入了磁盘。爬取到的数据主要包括：TeacherInfo 文件夹下按照学院存储的教师信息，以 TeacherInfo/computer.json 为例，它记录了计算机学院和网络空间安全学院的教师信息，如图 1所示，整个文件是一个 list 数据结构，list 中每一项是一个 dict，key 为类别，value 为具体内容。name、sex、department、position、title、degree、major、telephone、email、direction、introduction 分别表示教师的姓名、性别、所属部门、行政职务、职称、学历、所学专业、办公电话、电子邮件、研究方向和个人简介。

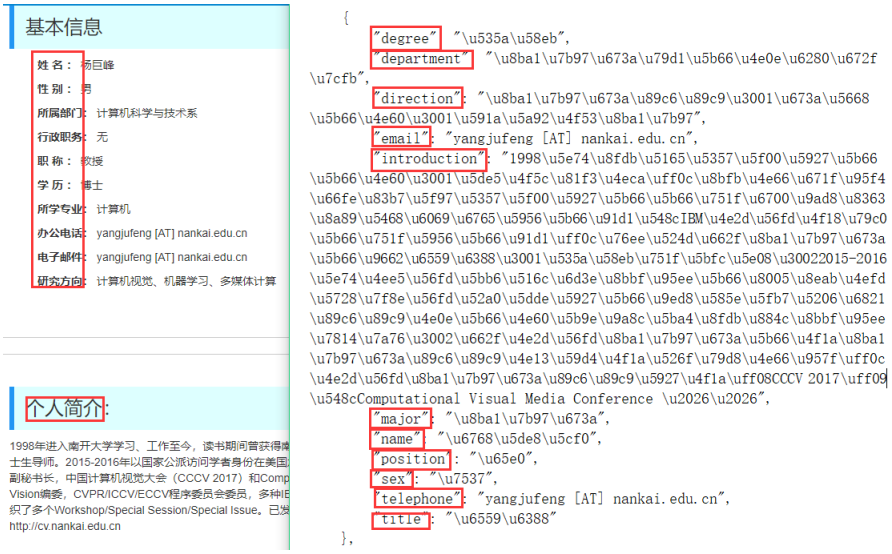

图 1: TeacherInfo/computer.json

3.2 索引准备

本次实验中，我主要是基于 sklearn 库，搭建一个比较简单的问答系统。在内容检索之前，需要先准备好语料库为了能让小开同学从用户输入的内容提取出关键信息，需要先准备一些问题的模板，用于匹配输入串。比如用户输入“杨巨峰的研究方向是什么?”或“杨巨峰是做什么方向的?”或“杨巨峰做什么方面研究的?”，小开同学应该知道当前都是需要去查“name = 杨巨峰”的 direction 信息。

提前准备好的问题关键词与对应的索引信息，如图 2所示。图片左侧是 Qest.txt 的截图，用户输入的问题会与其中一行匹配，匹配的方法稍后解释。图片右侧是 Ans.txt 的截

图，每一行分别对应着 Qest.txt 的每一样。在上面的例子中，三个问题都会匹配到“研究方向，做什么方面”这一行，也就对应着“direction”这一行，同时程序从问题中提取出前面的人名“杨巨峰”，从而理解用户想要查询的信息。

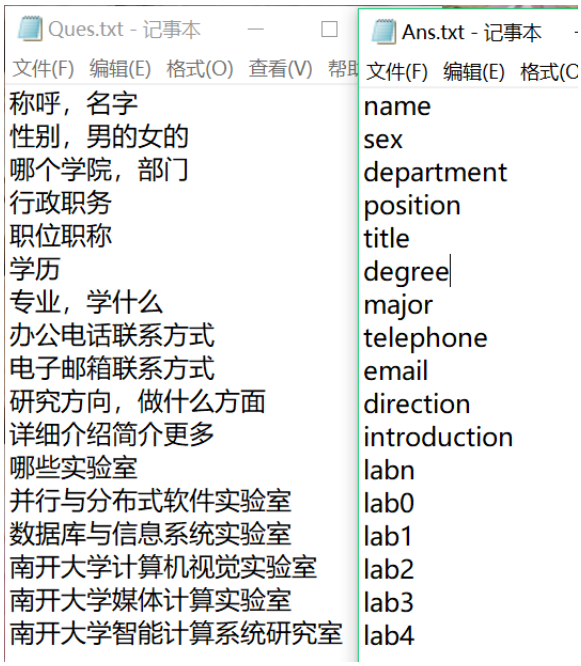

图 2: TeacherInfo/computer.json

3.3 内容检索

(1) 首先需要去读取给定的文件，并把文件里的内容读取到 list 里面。用到了文件 IO 操作方面的基本知识。

(2) 处理已有的字符串数据，并把它们转换成词袋向量。这部分内容涉及到一些简单的字符串预处理技术（比如过滤掉一些没用的字符、分词等），还有就是基于 sklearn 的把字符串转换向量的过程。用到了字符串操作、分词、词袋模型相关的基础知识。

词袋模型能够把一个句子转化为向量表示，是比较简单直白的一种方法，它不考虑句子中单词的顺序，只考虑词表（vocabulary）中单词在这个句子中的出现次数。scikit-learn 中的 CountVectorizer() 函数实现了 BOW 模型，可以直接调用。

BOW 模型有很多缺点，首先它没有考虑单词之间的顺序，其次它无法反应出一个句子的关键词，而 TF-IDF 则可以解决这个问题。TF（Term Frequency，词频）的公

式为：

$$TF(w) = \frac{num}{NUM}$$

num 是单词 w 在文章中出现的次数，NUM 是文章的单词总数。

而 IDF（inverse document frequency，逆文本频率）的公式为：

$$IDF(w) = log(\frac{D}{d+1})$$

D 是语料库中文档的总数，d 是包含词 w 的文档数。

TF-IDF 的公式为：

$$TF{-}IDF(w) = TF(w) * IDF(w)$$

sklearn 中封装了 TF-IDF 方法 TfidfVectorizer()，所以直接调函数就可以了。这一部分主要由 feature_extractors.py 程序实现, 定义及使用到的函数如图 3所示。

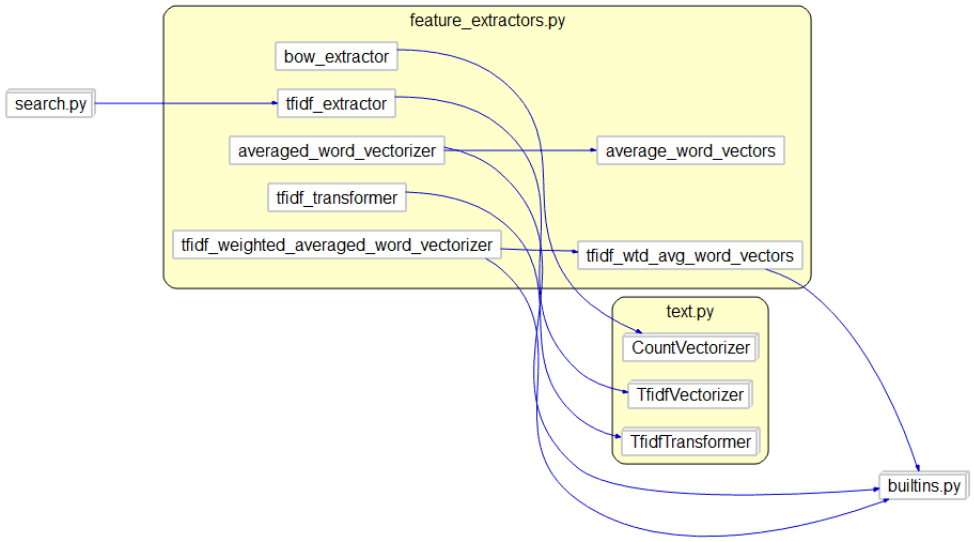

图 3: ClusterCallButterfly-feature_extractors-py

(3) 对于用户的新输入，返回答案。这是最后一部分，也就是等我们创建完词袋向量，用 TF-IDF 算法可以自动提取关键词之后，我们就可以输入一些新的问题，然后从库中找出最合适的答案。这部分的任务涉及到余弦相似度、简单搜索排序等方面基础知识。

余弦相似度是通过计算两个向量的夹角余弦值来评估他们的相似度。在 idx_for_largest_cosine_sim(input, questions) 函数中实现，用到了 numpy 包。

由于小开同学能回答的问题有限，所以在这里需要做 try-catch 处理，如果没有该项信息，要使程序正确返回提示信息。

另外，还对查询过程做了一些优化，比如教师的数据只在第一次查询时从磁盘读取到内存，后续的查询时，直接从内存中读数据，速度会快一些。并且使用百度的语音合成接口，将查到的答案合成 MP3 文件，并使用 Windows 的系统函数 playsound() 播放语音。

这一部分主要由 feature_extractors.py 程序实现, 定义及使用到的函数如图 4所示。

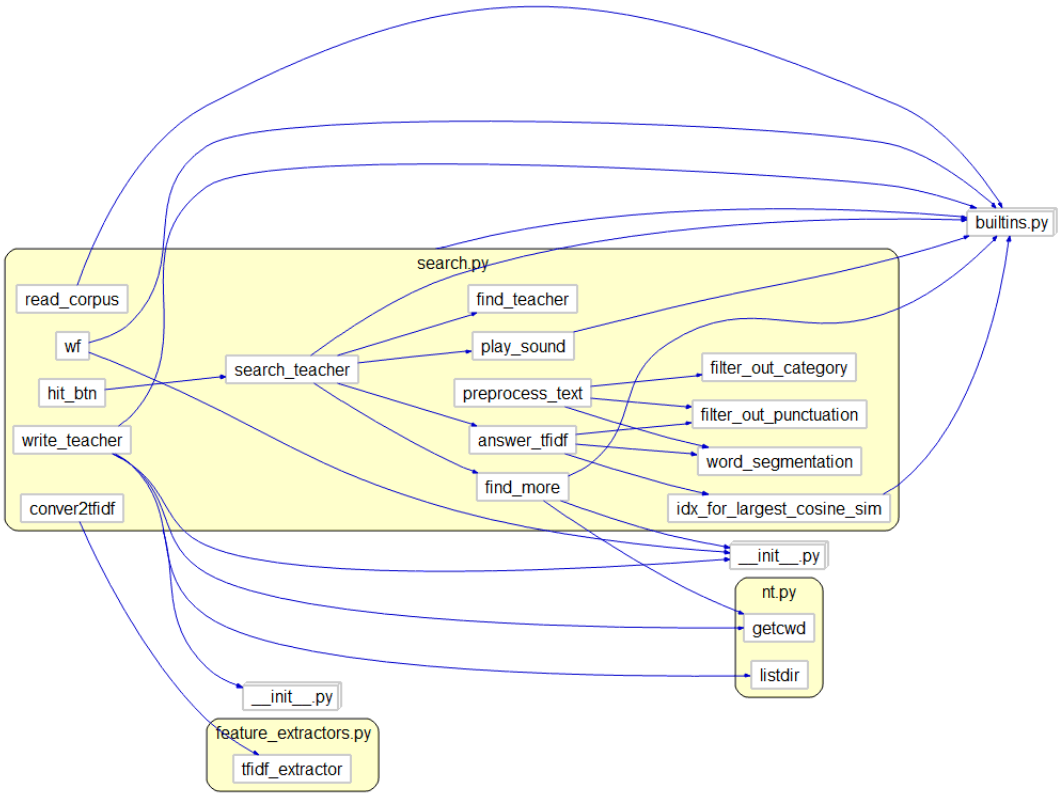

图 4: Cluster-search-py

3.4 界面设计

在设计小开同学的查询界面时，使用了南开的特有标志——阳阳和亮亮，使用南开紫作为背景色，上方是查询输入框，用户完成输入后，点击下方的“OK”按钮，就会触发search_teacher() 函数，搜索的结果会显示在下方的文本区中，如图 5所示。

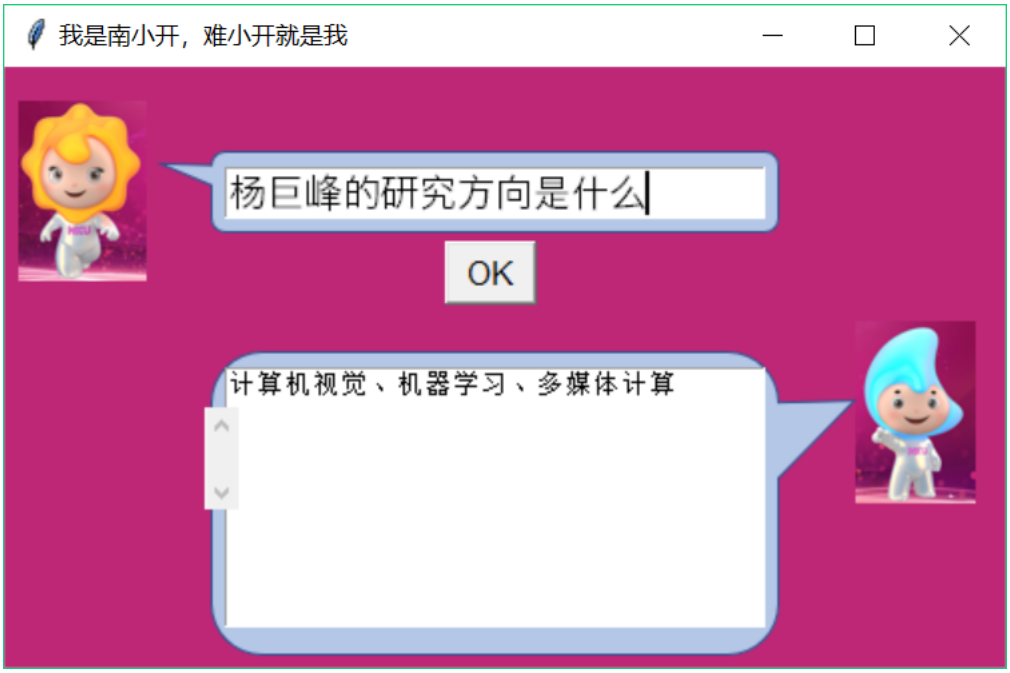

图 5: 我是南小开

4 代码实现

4.1 爬虫代码

在网页抓取子系统中，使用了 selenium 和 BeautifulSoup 工具包，每各学院网站的结构都不尽相同，具体的代码也会稍有不同，所以我分学院写了爬虫的函数，这些函数的原理是相同的。这里只粘贴了爬取金融学院教师信息时用到的两个主要函数。首先在 computercol() 函数中使用 selenium 中的 webdriver 模拟火狐浏览器执行点击操作，根据指定的 URL 获取页面内容，存放在 browser 变量中。然后调用get_cc_data(t_info, url) 函数，使用 BeautifulSoup 解析 HTML 文本，剖析器为html.parser，根据网页的结构，找到需要信息的所在位置，对于计算机学院的“师资队伍”

页面，用 F12 开发者工具可以查看到我们需要的信息，调用 select() 函数取出信息。可以使用 browser.find_element_by_class_name('teacherDetail').text，取出相关文字。另外，由于个人简介的信息太长，并且不是关键信息，没必要全部存储，所以只取了前 200 个字。

```
def get_cc_data(t_info, url):
    urlset = set() # 用于判断是否重复
    try:
        # 使用剖析器为html.parser
        soup = BeautifulSoup(browser.page_source, 'html.parser')
        allList = soup.select('tr')
        Len = len(allList)
        cc = ['name', 'sex', 'department', 'position', 'title', 'degree', 'major',
            'telephone', 'email', 'direction',
             'introduction']
        for i in range(1, Len):
            a = allList[i].select('td')[0].select('a')
            try: # 如果抛出异常就代表为空
                href = a[0]['href']
                if not href.startswith('http'):
                    href = r'http://cc.nankai.edu.cn' + href
                browser.get(href)
                if href in urlset:
                    continue
                else:
                    urlset.add(href)
                    intro = browser.find_element_by_class_name('text').text
                    info = {}
                    for each in range(1, 11):
                        x_path =
                            r'/html/body/div[2]/div[1]/div/div[1]/div[1]/p[%d]/span[2]' %
                            each
                        info[cc[each - 1]] = browser.find_element_by_xpath(x_path).text
                    print(info)
                    info[cc[10]] = (intro[0:min(len(intro), 200)] + ' ……')
                    sleep(1)
                    t_info.append(info)
            except:
                href = ''
    except Exception as e:
```

```
        print(e)


def computercol():
    info = []
    urls = [r'http://cc.nankai.edu.cn/jswyjy/list.htm',
        r'http://cc.nankai.edu.cn/fjswfyjy/list.htm',
          r'http://cc.nankai.edu.cn/js/list.htm',
              r'http://cc.nankai.edu.cn/syjxdw/list.htm']
    for url in urls:
        browser.get(url)
        get_cc_data(info, url)
    # 写入
    wf('TeacherInfo/computer.json', info)
```

4.2 内容检索相关代码

索引的建立与查找主要由 search.py 和 feature_extractors.py 两个文件实现。首先是 feature_extractors.py 中的关键代码：

```
from sklearn.feature_extraction.text import CountVectorizer


def bow_extractor(corpus, ngram_range=(1, 1)):
    vectorizer = CountVectorizer(min_df=1, ngram_range=ngram_range)
    features = vectorizer.fit_transform(corpus)
    return vectorizer, features


from sklearn.feature_extraction.text import TfidfTransformer


def tfidf_transformer(bow_matrix):
    transformer = TfidfTransformer(norm='l2',
                                   smooth_idf=True,
                                   use_idf=True)
    tfidf_matrix = transformer.fit_transform(bow_matrix)
    return transformer, tfidf_matrix
```

```
from sklearn.feature_extraction.text import TfidfVectorizer


def tfidf_extractor(corpus, ngram_range=(1, 1)):
    vectorizer = TfidfVectorizer(min_df=1,
                                 norm='l2',
                                 smooth_idf=True,
                                 use_idf=True,
                                 ngram_range=ngram_range)
    features = vectorizer.fit_transform(corpus)
    return vectorizer, features


import numpy as np


def average_word_vectors(words, model, vocabulary, num_features):
    feature_vector = np.zeros((num_features,), dtype="float64")
    nwords = 0.

    for word in words:
        if word in vocabulary:
            nwords = nwords + 1.
            feature_vector = np.add(feature_vector, model[word])

    if nwords:
        feature_vector = np.divide(feature_vector, nwords)

    return feature_vector


def averaged_word_vectorizer(corpus, model, num_features):
    vocabulary = set(model.index2word)
    features = [average_word_vectors(tokenized_sentence, model, vocabulary,
        num_features)
               for tokenized_sentence in corpus]
    return np.array(features)
```

```
def tfidf_wtd_avg_word_vectors(words, tfidf_vector, tfidf_vocabulary, model,
    num_features):
    word_tfidfs = [tfidf_vector[0, tfidf_vocabulary.get(word)]
                if tfidf_vocabulary.get(word)
                else 0 for word in words]
    word_tfidf_map = {word: tfidf_val for word, tfidf_val in zip(words, word_tfidfs)}

    feature_vector = np.zeros((num_features,), dtype="float64")
    vocabulary = set(model.index2word)
    wts = 0.
    for word in words:
        if word in vocabulary:
            word_vector = model[word]
            weighted_word_vector = word_tfidf_map[word] * word_vector
            wts = wts + word_tfidf_map[word]
            feature_vector = np.add(feature_vector, weighted_word_vector)
    if wts:
        feature_vector = np.divide(feature_vector, wts)

    return feature_vector


def tfidf_weighted_averaged_word_vectorizer(corpus, tfidf_vectors,
                                        tfidf_vocabulary, model, num_features):
    docs_tfidfs = [(doc, doc_tfidf)
                for doc, doc_tfidf
                in zip(corpus, tfidf_vectors)]
    features = [tfidf_wtd_avg_word_vectors(tokenized_sentence, tfidf, tfidf_vocabulary,
                                       model, num_features)
            for tokenized_sentence, tfidf in docs_tfidfs]
    return np.array(features)
```

下面是 search.py 中的关键代码：

首先需要去读取给定的文件，并把文件里的内容读取到 list 里面:

```
def read_corpus(file):
    with open(file, 'r', encoding='utf8', errors='ignore') as f:
        list = []
```

```
        lines = f.readlines()
        for i in lines:
            list.append(i)
    return list


questions = read_corpus('./Ques.txt')
answers = read_corpus('./Ans.txt')
```

处理已有的字符串数据，并把它们转换成词袋向量。这部分内容涉及到一些简单的字符串预处理技术（比如过滤掉一些没用的字符、分词等），还有就是基于 sklearn 的把字符串转换向量的过程:

```
def filter_out_category(input):
    new_input = re.sub('[\u4e00-\u9fa5]{2,5}\\/', '', input)
    return new_input


def filter_out_punctuation(input):
    new_input = re.sub('([a-zA-Z0-9])', '', input)
    new_input = ''.join(e for e in new_input if e.isalnum())
    return new_input


def word_segmentation(input):
    new_input = ','.join(jieba.cut(input))
    return new_input


def preprocess_text(data):
    new_data = []
    for q in data:
        q = filter_out_category(q)
        q = filter_out_punctuation(q)
        q = word_segmentation(q)
        new_data.append(q)
    return new_data
```

```
qlist = preprocess_text(questions) # 更新后的


def conver2tfidf(data):
    new_data = []
    for q in data:
        new_data.append(q)
    tfidf_vectorizer, tfidf_X = tfidf_extractor(new_data)
    return tfidf_vectorizer, tfidf_X
```

对爬取到的教师信息进行整理：

```
def write_teacher():
    filedir = os.getcwd() + '\index'
    filenames = os.listdir(filedir)

    for filename in filenames:
        filepath = filedir + '\\' + filename
        with open(filepath, encoding='utf-8')as f:
            cp_list = json.load(f)
        count = 0
        for cp_dict in cp_list:
            name = cp_dict["name"]
            print(name)
            namelist.append(name)
            teacher_dict[name] = [count, filename]
            count = count + 1
    wf('teacher_dict.json', teacher_dict)
    wf('namelist.json', namelist)
```

对于用户的新输入，返回答案。这是最后一部分，也就是等我们创建完词袋向量之后，我们就可以输入一些新的问题，然后从库中找出最合适的答案:

```
tfidf_vectorizer, tfidf_X = conver2tfidf(qlist)


def idx_for_largest_cosine_sim(input, questions):
    list = []
    input = (input.toarray())[0]
    for question in questions:
```

```
        question = question.toarray()
        num = float(np.matmul(question, input))
        denom = np.linalg.norm(question) * np.linalg.norm(input)

        if denom == 0:
            cos = 0.0
        else:
            cos = num / denom

        list.append(cos)

    best_idx = list.index(max(list))
    return best_idx


def answer_tfidf(input):
    input = filter_out_punctuation(input)
    input = word_segmentation(input)
    bow = tfidf_vectorizer.transform([input])
    best_idx = idx_for_largest_cosine_sim(bow, tfidf_X)
    return answers[best_idx]


teacher_dict = dict()




def find_teacher(name):
    list = teacher_dict[name]
    return list


def find_more(list, line):
    num, url = list
    filedir = os.getcwd() + r'\index'
    filepath = filedir + '\\' + url
    with open(filepath, encoding='utf-8')as f:
        cp_list = json.load(f)
```

```
    now_list = cp_list[num]
    try:
        find = now_list[line]
    except:
        # 没有该项信息
        find = '抱歉，这个问题我暂时不知道呢'
    return find


def search_teacher(str):
    s1 = str[0:3]
    n1 = str[0:2]
    s2 = str[3:]
    if s1 in namelist or n1 in namelist:
        # 教师相关
        if s1 in namelist:
            name = s1
        else:
            name = n1
        find_line = answer_tfidf(s2)
        line = find_line.strip('\n')
        finder = find_teacher(name)
        final_answer = find_more(finder, line)
        print(name, final_answer)
        play_sound(final_answer)
        return final_answer
    else:
        find_line = answer_tfidf(s2)
        line = find_line.strip('\n')
        if line[3] == 'n':
            final_answer =
                '并行与分布式软件实验室，数据库与信息系统实验室，南开大学计算机视觉实验室，南开大学媒体计算实验室，
            print(final_answer)
            play_sound(final_answer)
            return final_answer

        else:
            n = int(line[3])
            with open('lab.json', encoding='utf-8')as fl:
```

```
            lab = json.load(fl)
            final_answer = lab[n]
            print(final_answer)
            play_sound(final_answer)
            return final_answer
```

用于播放音频的函数：

```
def play_sound(answer):
    from aip import AipSpeech

    """ 你的 APPID AK SK """
    APP_ID = '18013674'
    API_KEY = 'vy91NRWxtpYq0XPdRIGXEopp'
    SECRET_KEY = 'A1BBTZI159XVFs5rubMH517fhcP53nhW'
    client = AipSpeech(APP_ID, API_KEY, SECRET_KEY)

    result = client.synthesis(answer, 'zh', 1, {
        'vol': 5,
    })
    # 识别正确返回语音二进制 错误则返回dict 参照下面错误码
    if not isinstance(result, dict):
        global tmp
        tmp += 1
        file = r'audio/' + str(tmp) + '.mp3'
        with open(file, 'wb') as f:
            f.write(result)

        playsound(file)
```

主函数，界面设计部分：

```
if __name__ == '__main__':
    # write_teacher()
    with open('teacher_dict.json', encoding='utf-8')as f:
        teacher_dict = json.load(f)
    with open('namelist.json', encoding='utf-8')as f:
        namelist = json.load(f)
    # 实例化object，建立窗口window
    window = tk.Tk()
```

```
# 给窗口的可视化起名字
window.title('我是南小开，难小开就是我')
# 设定窗口的大小(长 * 宽)
window.geometry('500x300')
# 增加背景图片
photo = tk.PhotoImage(file="bg.png")
theLabel = tk.Label(window, image=photo, compound=tk.CENTER, fg="white") # 前景色
theLabel.pack()

var_from = tk.StringVar()
entry_from = tk.Entry(window, textvariable=var_from, font=('Arial', 14))
entry_from.place(width=270, x=110, y=50)

t = tk.Text(window, height=7)
scroll = Scrollbar()
scroll.place(x=98, y=173)
scroll.config(command=t.yview)
t.config(yscrollcommand=scroll.set)
t.place(width=270, height=130, x=110, y=150)


def hit_btn():
    # 定义一个函数功能供点击Button按键时调用，调用命令参数command=函数名
    e_from = var_from.get()
    text = search_teacher(e_from)
    t.delete(0.0, END)
    t.insert(INSERT, text)


btn_search = tk.Button(window, text='OK', font=('Arial', 12), width=4, height=1,
    command=hit_btn)
btn_search.place(x=220, y=87)

# 主窗口循环显示，让window不断的刷新
window.mainloop()
```

5 效果展示

这里没法展示语音回答的效果，所以只展示搜索到的文字。

(1) 对于没有找到的情况（图 6）

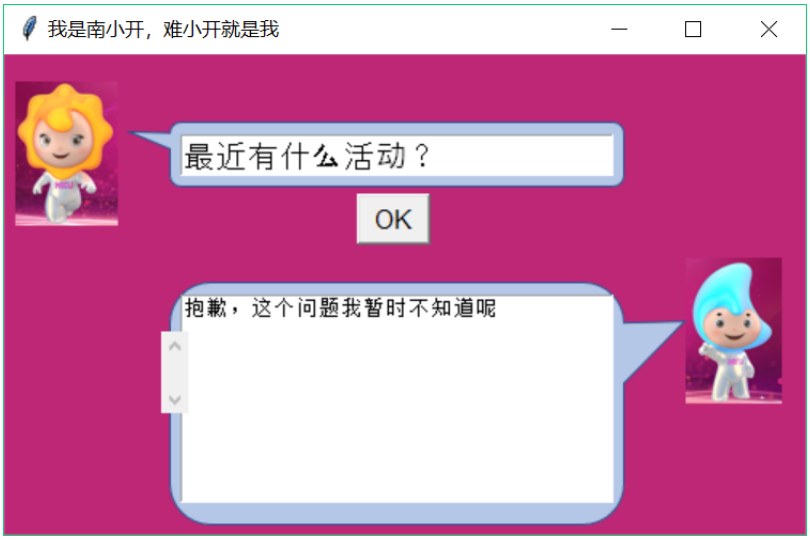

图 6: noInfo

(2) 杨巨峰老师在哪个部门？（图 7）

(3) 袁晓洁老师的简介（图 8）

(4) 有哪些实验室？（图 9）

(5) 并行与分布式软件实验室的简介（图 10）

(6) 程明明老师的研究方向是什么？（图 11）

(7) 程仁洪老师的电子邮箱是多少？（图 12）

6 用户评价

(1) “还不错，如果能查询更多的问题就更好了”

(2) “不到一周时间做成这样，还是不错的，如果时间多一点，占分多一点，肯定能做得更好”

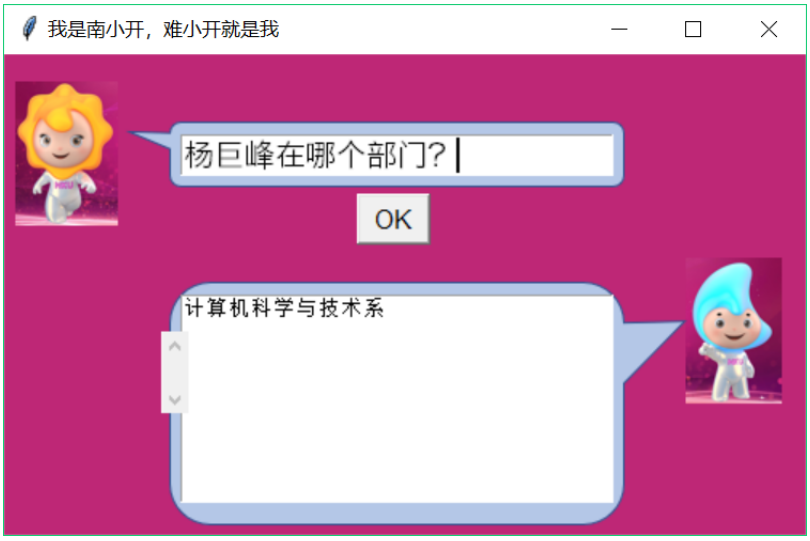

图 7: department

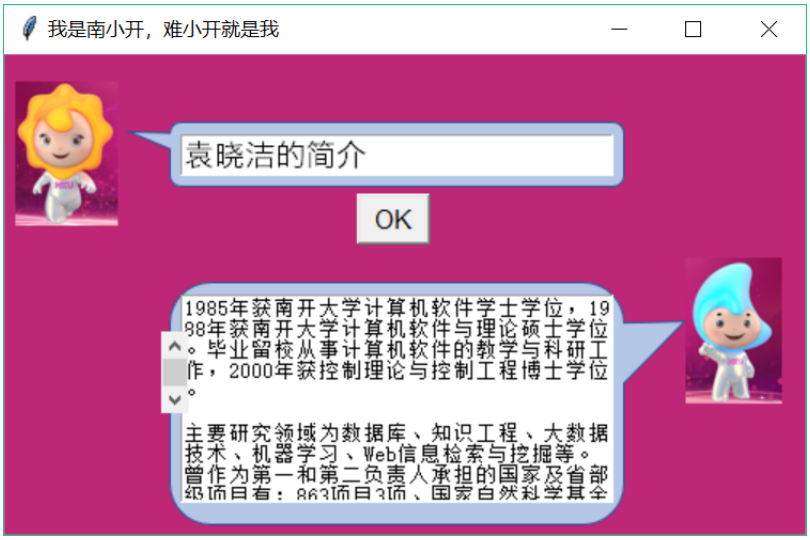

图 8: introduction

(3) “这大概是个“伪智能”问答系统吧。”

(4) “能问的问题有点受限制，不过还是很棒的呀，加油加油”

(5) “整体框架可以的。可能是时间问题，数据量不是很大，只能回答一下部分问题。有

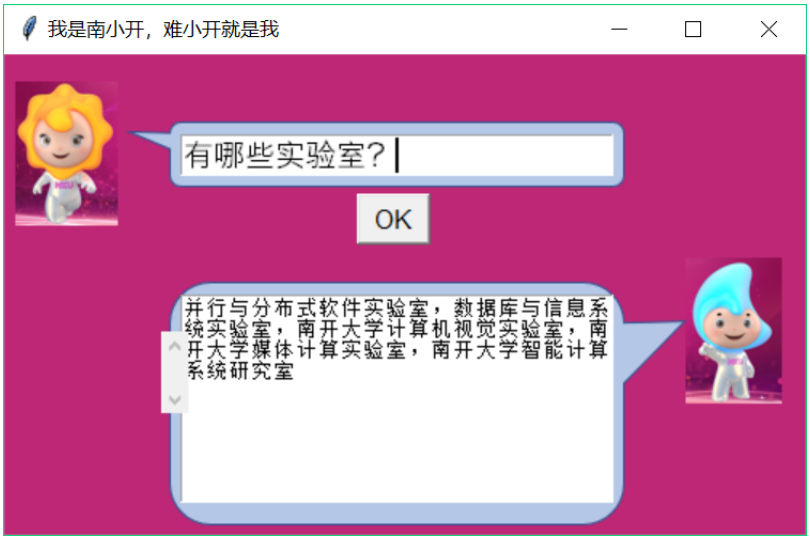

图 9: labInfo

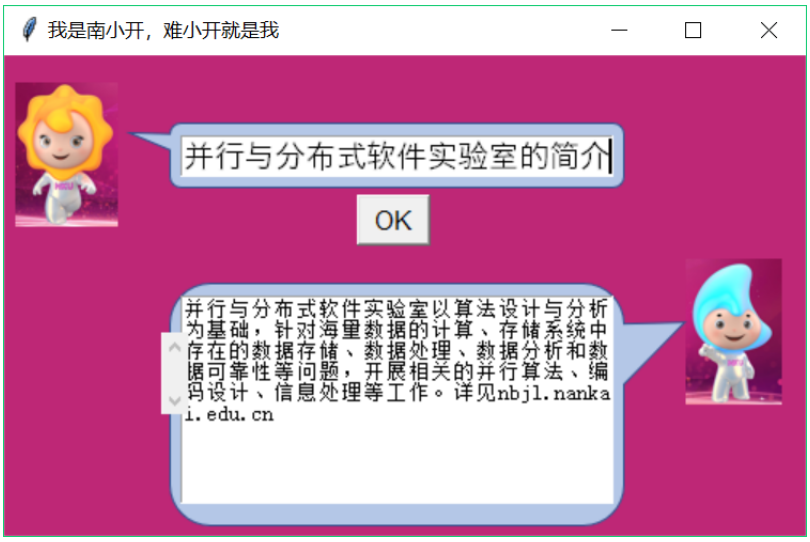

图 10: lab1

机会的话可以试试比较高级的包，加油 ”

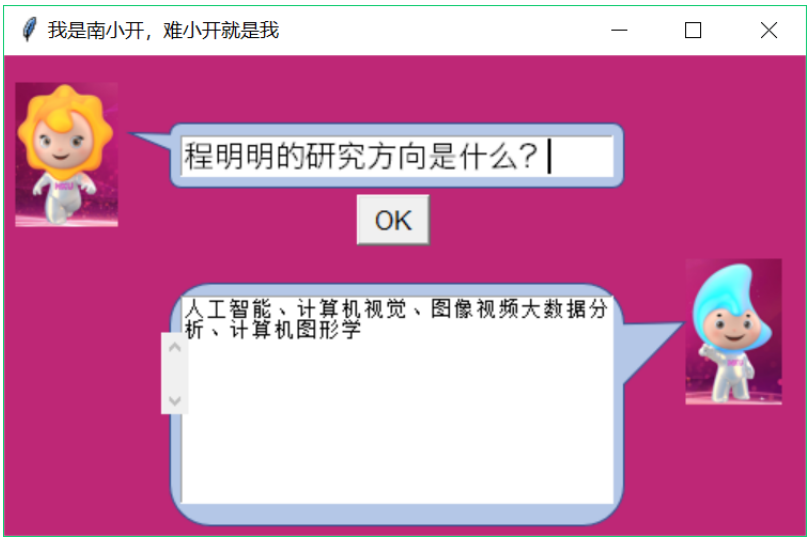

图 11: direction

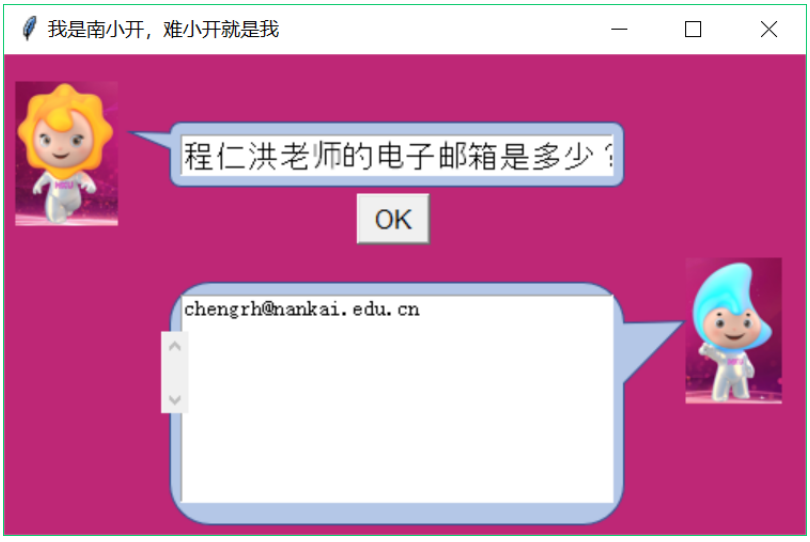

图 12: email