# 目　　录

# 第一部分：任务规划

## 1 个人知识背景

本人此前已经根据网上教程，自学过 python 的语法知识，并且写过一些小脚本，如重命名文件、爬取图片等，基本可以做到熟练应用。

## 2 任务规划

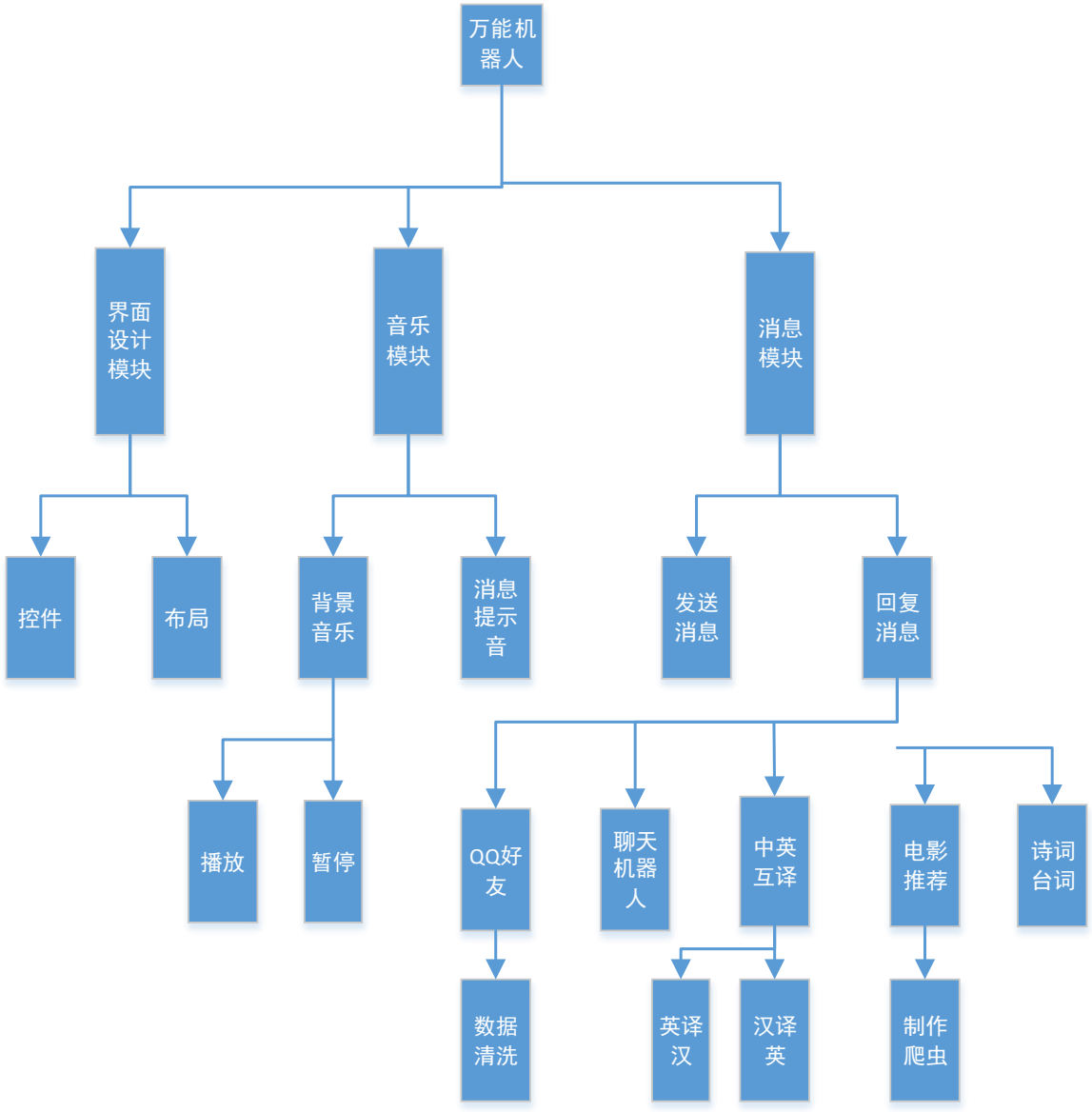
（1） 周一：完成 python 语法的复习，并确定主要完成的项目内容。
（2） 周二：确定技术难点，确定将使用的第三方库。开始进行开发，熟悉相关的第三方库。
（3） 周三：进行代码重构，进一步完善功能。
（4） 周四：进阶性地了解机器学习，熟悉相关的第三方库以及框架，进行示例的学习，如 tensorflow、sklearn 等。
（5） 周五：撰写文档，总结一周的课程学习成果。

# 第二部分：完成项目

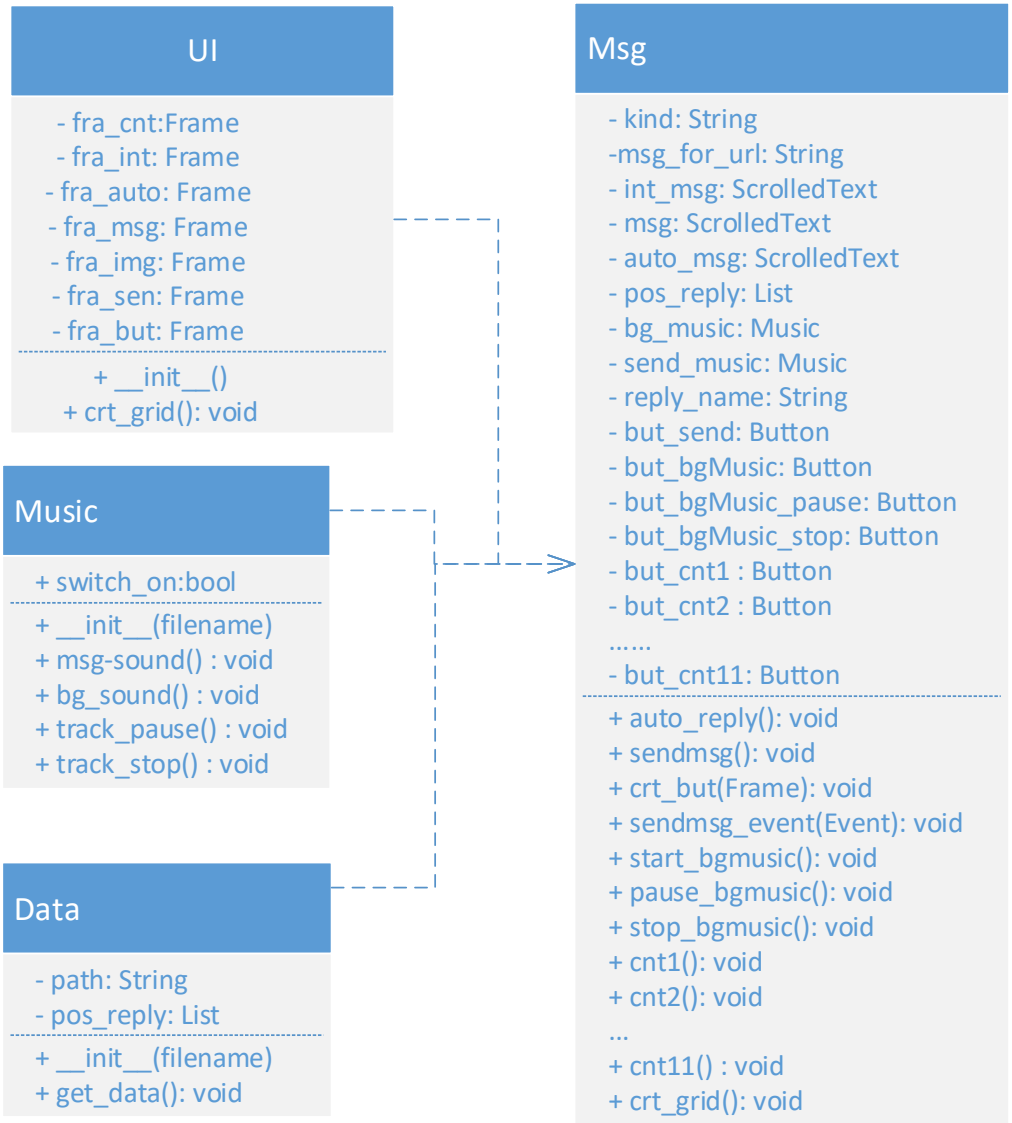
## 1 项目简介

本软件为一个多功能机器人，类型为工具包智能软件（含界面），根据用户的输入，进行相应的回复，可以完成的功能有模拟与 QQ 好友的聊天、与聊天机器人聊天、在线进行中英文互译、推荐电影、播放音乐等。

# 2 功能模块划分


万能机器人
界面设计模块
音乐模块
消息模块
控件
布局
背景音乐
消息提示音
发送消息
回复消息
播放
暂停
QQ好友
聊天机器人
中英互译
电影推荐
诗词台词
数据清洗
英译汉
汉译英
制作爬虫

# 3 类的设计


UI
- fra_cnt:Frame
- fra_int: Frame
- fra_auto: Frame
- fra_msg: Frame
- fra_img: Frame
- fra_sen: Frame
- fra_but: Frame
+ __init__()
+ crt_grid(): void
Music
+ switch_on:bool
+ __init__(filename)
+ msg-sound() : void
+ bg_sound() : void
+ track_pause() : void
+ track_stop() : void
Data
- path: String
- pos_reply: List
+ __init__(filename)
+ get_data(): void
Msg
- kind: String
-msg_for_url: String
- int_msg: ScrolledText
- msg: ScrolledText
- auto_msg: ScrolledText
- pos_reply: List
- bg_music: Music
- send_music: Music
- reply_name: String
- but_send: Button
- but_bgMusic: Button
- but_bgMusic_pause: Button
- but_bgMusic_stop: Button
- but_cnt1 : Button
- but_cnt2 : Button
……
- but_cnt11: Button
+ auto_reply(): void
+ sendmsg(): void
+ crt_but(Frame): void
+ sendmsg_event(Event): void
+ start_bgmusic(): void
+ pause_bgmusic(): void
+ stop_bgmusic(): void
+ cnt1(): void
+ cnt2(): void
...
+ cnt11() : void
+ crt_grid(): void

# 4 详细设计与实现

## 4.1 整体思路

（1） 使用的第三方库：

```
from tkinter import *
from tkinter import scrolledtext
from PIL import ImageTk
import PIL.Image
import pygame
import time
import random
import requests
import hashlib
import time
import json
```

（2） UI 类负责根据网格布局，对整个界面进行设定，创建 Frame 类的多个对象，便于后续的填充完善。

（3） Music 类调用了 pygame 库，对音乐的播放、暂停等功能进行了实现。背景音乐和消息提示音可以作为两个对象进行操作。

（4） Data 类进行数据的获取，存储了可能的智能回复列表。内部有文件操作。针对 QQ 聊天记录，进行了数据清洗。针对电影数据，制作爬虫，从豆瓣电影上爬取数据。针对翻译以及聊天机器人，调用对应官网的 API，进行 json 格式的解析以及获取数据。

（5） Msg 类是核心类，创建了不同的按钮，并且根据用户在左侧通信对象的选择，以及用户的输入消息，进行智能的回复。对输入框、自我消息框、接受消息框进行实时更新。

## 4.2 技术难点

（1） 对 tkinter 图形界面库的熟悉以及布局色彩方面的知识的掌握。

（2） 爬虫的制作。对正则表达式、json、网络请求方面的理解。进一步了解了 requests、bs4 库。

（3） 官网 API 的调用。查看官方文档，解析 json 数据。

# 5 核心代码

## 5.1 整体布局

```
class UI:

    def __init__(self):
        self.fra_cnt = Frame(width=150,height=409,bg="gray")
        self.fra_int = Frame(width=250,height=300,bg="white")
        self.fra_auto = Frame(width=250,height=300,bg="white")
        self.fra_msg = Frame(width=500,height=100,bg="white")
        self.fra_img = Frame(width=100,height=409,bg="white")
        self.fra_sen = Frame(width=655,height=20,bg="#00ff88")
        self.fra_but = Frame(width=100,height=20,bg="gray")
        # 创建几个区域划分，指定每块的大小，颜色等属性
        # 以后：加进头像，音乐，动画

    # 规划以及显示总体布局
    def crt_grid(self):

        self.fra_cnt.grid(row=0,rowspan=5,column=0,padx=6,pady=6)
        self.fra_int.grid(row=0,rowspan=4,column=1,pady=6)
        self.fra_auto.grid(row=0,rowspan=4,column=2,pady=6)
        self.fra_msg.grid(row=4,column=1,columnspan=2,padx=3)
        self.fra_img.grid(row=0,rowspan=5,column=3,padx=3)
        self.fra_sen.grid(row=5,column=0,columnspan=3,padx=2,pady=6)
        self.fra_but.grid(row=5,column=3)

        self.fra_cnt.grid_propagate(0)
        self.fra_int.grid_propagate(0)
        self.fra_auto.grid_propagate(0)
        self.fra_msg.grid_propagate(0)
        self.fra_img.grid_propagate(0)
        self.fra_sen.grid_propagate(0)
        self.fra_but.grid_propagate(0)
```

## 5.2 智能回复

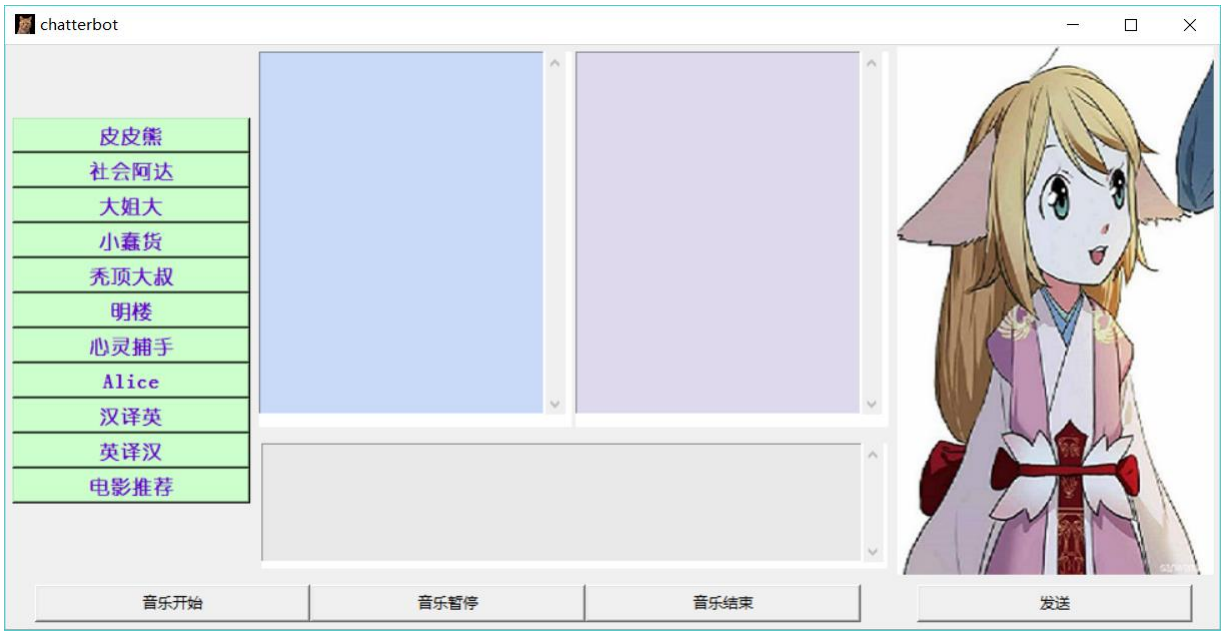
```
def auto_reply(self):
    if self.kind == 'bolt':
        key = "86a516ab67a64ed29f28bb683e77e1f8"
        url = "http://www.tuling123.com/openapi/api?key=" + key + "&info=" + self.msg_for_url
        html = requests.get(url)
        fin_msg = html.json()['text']+'\n'
    elif self.kind == '汉译英':
        appKey = '397d5478d0f3fa46'
        secretKey = 'H3c6MnnBgL4Itq0uR5ajz6Muca4XxJIO'
        q = self.msg_for_url
        salt = str(time.time())[:10]
        sign = appKey+q+salt+secretKey
        sign = hashlib.md5(sign.encode("utf-8")).hexdigest()
        url = "http://openapi.youdao.com/api"
        params = {
                    'q':q.encode('utf-8'),
                    'from':'zh',
                    'to':'en',
                    'appKey':appKey,
                    'salt': salt,
                    'sign':sign
                    }
        html = requests.get(url,params=params)
        html.encoding = html.apparent_encoding
        html = html.text
        res = json.loads(html)
        fin_msg = res['translation'][0]+'\n'
    elif self.kind == '英译汉':
        appKey = '397d5478d0f3fa46'
        secretKey = 'H3c6MnnBgL4Itq0uR5ajz6Muca4XxJIO'
        q = self.msg_for_url
        salt = str(time.time())[:10]
        sign = appKey+q+salt+secretKey
        sign = hashlib.md5(sign.encode("utf-8")).hexdigest()
        url = "http://openapi.youdao.com/api"
        params = {
                    'q':q.encode('utf-8'),
                    'from':'en',
                    'to':'zh',
                    'appKey':appKey,
                    'salt': salt,
                    'sign':sign
                    }
        html = requests.get(url,params=params)
        html.encoding = html.apparent_encoding
        html = html.text
        res = json.loads(html)
        fin_msg = res['translation'][0]+'\n'
    else:
        while self.i==self.j:
            self.i = random.randint(0, len(self.pos_reply))
        self.j = self.i
        fin_msg = self.pos_reply[self.i]+'\n'
    self.auto_msg.insert(END, self.reply_name+": ", 'begin')
    self.auto_msg.insert(END, fin_msg, 'automsg')
```

## 5.3 数据爬取

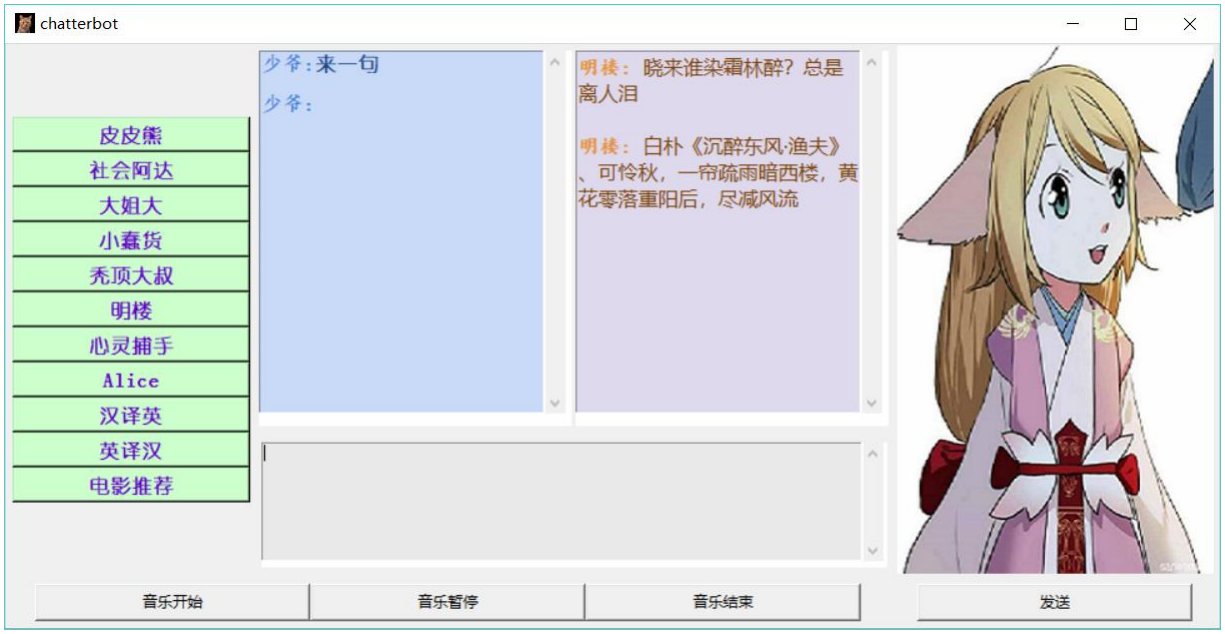
```
def get_html(url):
    r = requests.get(url)
    r.encoding = r.apparent_encoding
    return r
def make_soup(html,n):

    movies = []
    s = "{0:{3}^5}{1:{3}^5}{2:{3}^5}"
    soup = BeautifulSoup(html.text,'html.parser')
    orign = soup.find('ol',class_="grid_view")
    a = orign.find_all('li')


    for i in a:
        data = []
        if i.find('span',class_='title'):
            data.append(i.find('span',class_='title').string)
        else:
            data.append("无")
        if i.find('span',class_='rating_num'):
            data.append(i.find('span',class_='rating_num').string)
        else:
            data.append("无")
        if i.find('span',class_='inq'):
            data.append(i.find('span',class_='inq').string)
        else:
            data.append("无")
        movies.append(data)

    with open("D:\\ChatRobot\\ChatRobotV2\\text\\db_movie.txt",'a',encoding='utf-8') as f:
        # f.write(s.format("电影名称","评分","短评",chr(12288))+'\n')
        an = 0
        for i in movies:
            # print(i)
            f.write(s.format(i[0],i[1],i[2],chr(12288))+'\n')

def main():

    for i in range(0,10):
        url = "https://movie.douban.com/top250"+"?start="+str(25*i)
        html = get_html(url)
        make_soup(html,25*i)
```

# 6 核心界面截图

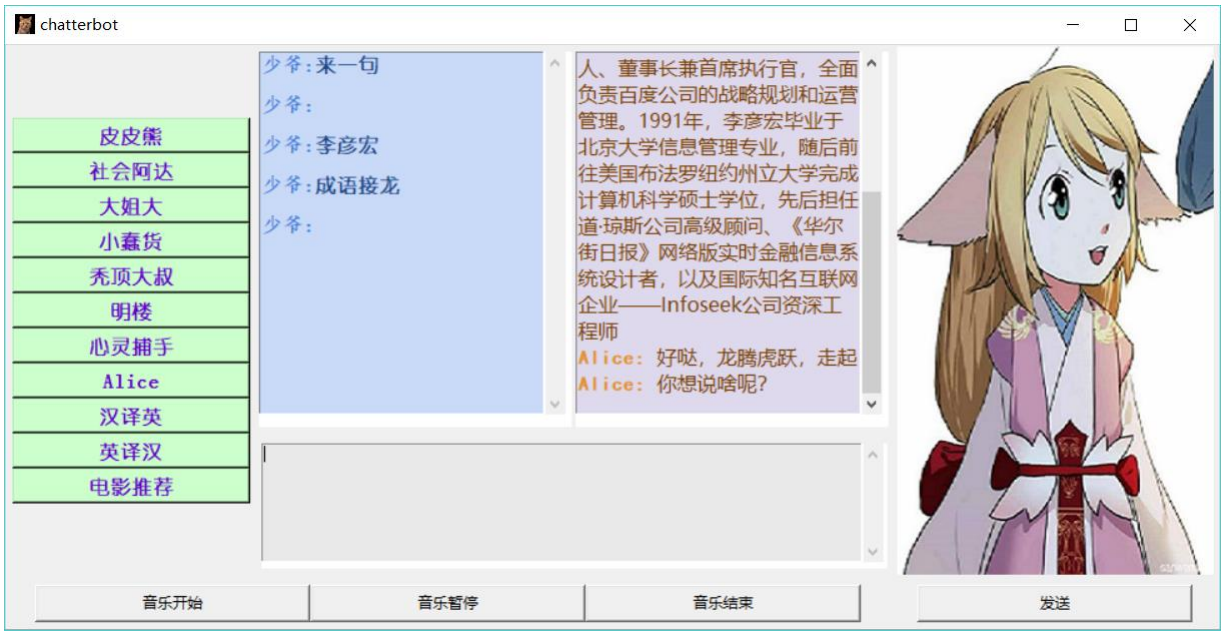
## 6.1 主界面




## . 6.2 诗词回复



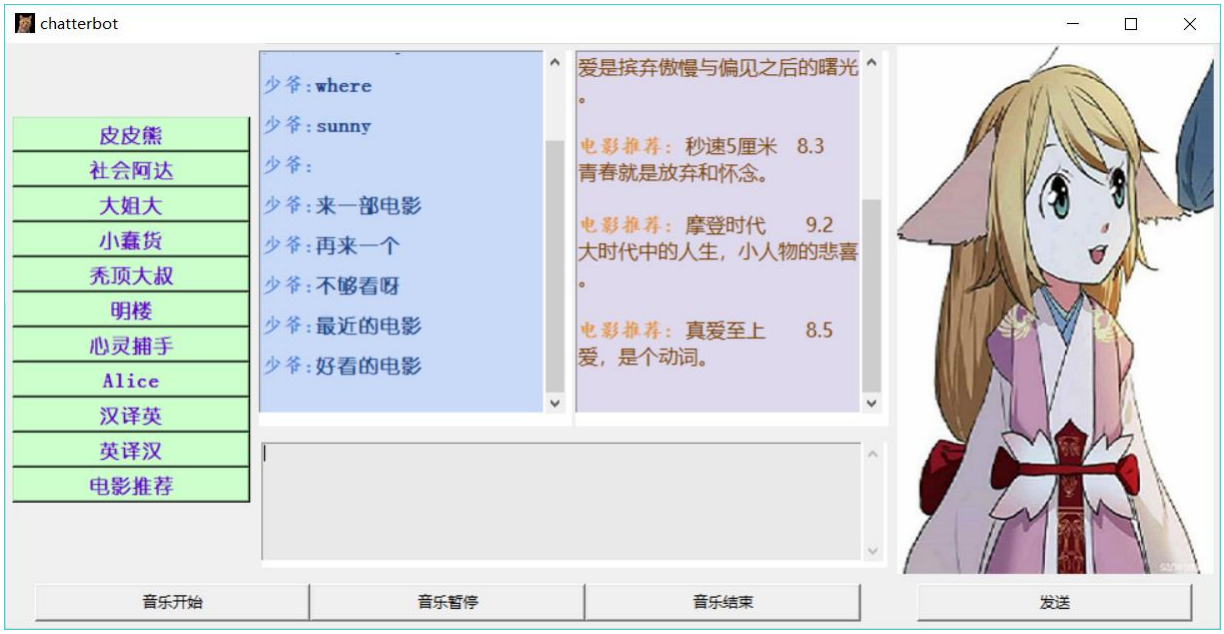
## 6.3 聊天机器人回复




## 6.4 汉译英

## 6.5 英译汉


## 6.5 电影推荐



# 第三部分：机器学习入门

## 1 整体把握

（1） 完成 tensorflow 环境的配置，并且对 cuda、gpu 进行了相应的了解。

（2） 对 tensorflow 框架的基本概念有了把握，如张量、数据流、损失函数、学习率、最大梯度下降算法、激励函数等。

（3） 对用 tensorflow 进行基本的神经网络操作流程有了基本掌握。

（4） 熟悉了 matplotlib、numpy 等数据分析和可视化的工具。

## 2 实例练习

### 2.1 解方程

根据 CSDN 博客上的示例进行学习理解。对简单的方程 Y=Wx+b 进行有标签的学习，根据已知的 x 和 y 数据集，构建神经网络，利用最大梯度下降算法，推演出 w 和 b.

```
import numpy as np  # 这是Python的一种开源的数值计算扩展，非常强大
import tensorflow as tf  # 导入tensorflow
##构造数据##
x_data = np.random.rand(100).astype(np.float32)  # 随机生成100个类型为float32的值
y_data = x_data * 0.1 + 0.3  # 定义方程式y=x_data*A+B
##-------##
##建立TensorFlow神经计算结构##
weight = tf.Variable(tf.random_uniform([1], -1.0, 1.0))
biases = tf.Variable(tf.zeros([1]))
y = weight * x_data + biases
##-------##
loss = tf.reduce_mean(tf.square(y - y_data))  # 判断与正确值的差距
optimizer = tf.train.GradientDescentOptimizer(0.1)  # 根据差距进行反向传播修正参数
train = optimizer.minimize(loss)  # 建立训练器
init = tf.initialize_all_variables()  # 初始化TensorFlow训练结构
sess = tf.Session()  # 建立TensorFlow训练会话
sess.run(init)  # 将训练结构装载到会话中
for step in range(400):  # 循环训练400次
    sess.run(train)  # 使用训练器根据训练结构进行训练
    if step % 20 == 0:  # 每20次打印一次训练结果
        print(step, sess.run(weight), sess.run(biases))  # 训练次数，A值，B值

```

```
0 [ 0.09407941] [ 0.06358099]
20 [ 0.17139012] [ 0.26260602]
40 [ 0.15629344] [ 0.27111843]
60 [ 0.14419717] [ 0.27732643]
80 [ 0.13469948] [ 0.28219885]
100 [ 0.1272428] [ 0.28602418]
120 [ 0.12138849] [ 0.28902751]
140 [ 0.11679224] [ 0.29138544]
160 [ 0.11318369] [ 0.29323664]
180 [ 0.1103506] [ 0.29469004]
200 [ 0.10812633] [ 0.29583111]
220 [ 0.10638005] [ 0.296727]
240 [ 0.10500904] [ 0.29743031]
260 [ 0.10393263] [ 0.29798254]
280 [ 0.10308753] [ 0.29841611]
300 [ 0.10242405] [ 0.29875645]
320 [ 0.10190312] [ 0.29902372]
340 [ 0.10149413] [ 0.29923353]
360 [ 0.10117306] [ 0.29939824]
380 [ 0.10092098] [ 0.29952756]
[Finished in 6.6s]
```

## 2.2 MNIST 数据集

根据 CSDN 博客上的示例进行学习理解，主要是理论上的理解。

```python
import tensorflow as tf
import tensorflow.examples.tutorials.mnist.input_data as input_data
mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)
# mnist = read_data_sets("MNIST_data/", one_hot=True)
x = tf.placeholder(tf.float32, [None, 784])  # 图像输入向量
W = tf.Variable(tf.zeros([784, 10]))  # 权重，初始化值为全零
b = tf.Variable(tf.zeros([10]))  # 偏置，初始化值为全零
# 进行模型计算，y是预测，y_ 是实际
y = tf.nn.softmax(tf.matmul(x, W) + b)

y_ = tf.placeholder("float", [None, 10])

# 计算交叉熵
cross_entropy = -tf.reduce_sum(y_ * tf.log(y))
# 接下来使用BP算法来进行微调,以0.01的学习速率
train_step = tf.train.GradientDescentOptimizer(0.01).minimize(cross_entropy)

# 上面设置好了模型，添加初始化创建变量的操作
init = tf.initialize_all_variables()
# 启动创建的模型，并初始化变量
sess = tf.Session()
sess.run(init)
# 开始训练模型，循环训练1000次
for i in range(1000):
    # 随机抓取训练数据中的100个批处理数据点
    batch_xs, batch_ys = mnist.train.next_batch(100)
    sess.run(train_step, feed_dict={x: batch_xs, y_: batch_ys})
''''' 进行模型评估 '''
# 判断预测标签和实际标签是否匹配
correct_prediction = tf.equal(tf.argmax(y, 1), tf.argmax(y_, 1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, "float"))
# 计算所学习到的模型在测试数据集上面的正确率
print(sess.run(accuracy, feed_dict={
    x: mnist.test.images, y_: mnist.test.labels}))

```

```
instructions that this TensorFlow binary was not compiled to use: AVX AVX2
2018-07-26 14:46:30.192084: I C:\tf_jenkins\workspace\rel-win\M\windows-gpu\PY\35\te
properties:
name: GeForce GTX 950M major: 5 minor: 0 memoryClockRate(GHz): 1.124
pciBusID: 0000:01:00.0
totalMemory: 4.00GiB freeMemory: 3.35GiB
2018-07-26 14:46:30.192903: I C:\tf_jenkins\workspace\rel-win\M\windows-gpu\PY\35\te
device (/device:GPU:0) -> (device: 0, name: GeForce GTX 950M, pci bus id: 0000:01:00
0.9031
[Finished in 11.7s]
```

# 第四部分：学习总结与体会

## 1 心得体会

经过本次认识实习课程，我在自己原有的 python 基础上，对原有的知识进行了巩固，对语法知识以及一些技巧有了更加深入的了解。

通过做一个实际的项目，从最初的面向过程开发，进行代码重构，过渡到面向对象，使代码的易读性和扩展性得到了提高。除了工程学方面的进步，我还对 python 的图形库 tkinter 和 pygame 有了深入的了解，学会了布局管理。当然，还有网络方面的知识，例如制作一个爬虫，调用网络 API, 这都是一些很实用的技巧，或许也是 python 强大的原因。我也学会了看官方文档，而不仅仅是看网上的现有教程去学习。

在机器学习的入门中，我体会到了 python 作为高级语言的优势，有着丰富的第三方库以及框架，Tensorflow 更是优秀的一个，我只需要了解一些基本的概念，算法，然后可以将这个框架当做一个黑箱子，去进行训练，并最终得到结果。

## 2 不足与反思

（1）在整个项目界面的设计中，我明显感觉到了自己 UI 设计方面灵感以及经验的匮乏，色彩的搭配以及图片的选取，都是可以改善的地方。

（2）在本门课程中，虽然我利用了爬虫技术，但是并没有使用 scrapy 库，没有真正发挥多线程爬虫的威力。

（3）虽然我对机器学习有了大概的了解，但是很多理论上的东西，比如算法、思想等，仍然不够清楚，日后还要投入大量的时间去钻研，才能真正体会到机器学习的美丽，做出真正有趣的、令人惊叹的应用。