基于深度学习的聊天机器人的研究与实现

集美大学诚毅学院信息工程系

计算机科学与技术 2022 届 朱晓璇 201841051050

摘要 随着深度学习技术在自然语言处理领域的快速发展，具有良好人机交互性的聊天机器人也逐渐进入市场。但目前市面上大多数聊天机器人都被限定在特定领域或特定环境中，比如客服机器人等。本课题独辟蹊径地创建了一个用户可以在日常生活中直接使用的聊天机器人。本文设计的聊天机器人由文本获取模块、模型创建模块、回复生成模块三个部分组成。其中，文本获取模块使用了贴近生活的第三方语料青云语料库，并且使用 jieba 工具包实现了中文分词的功能；在模型创建和回复生成模块内运用了深度学习技术中长短时记忆（LSTM）模型和基于注意力机制的序列到序列（Seq2seq）模型，并利用 Tkinter 库设计了聊天界面，该界面与聊天机器人模型后端相连接，实现了一问一答形式的模拟对话功能。为确保聊天机器人的回复功能正常，本文对聊天机器人进行了效果评估。评估结果表明，该聊天机器人既具合理性又具趣味性，是一款合格的深度学习模型下的陪聊产品。

关键词 深度学习； 聊天机器人； LSTM 模型； 注意力机制； Seq2seq 模型

Research and Implementation of Chatbot Based on Deep Learning

Zhu Xiaoxuan

2018410501050,Computer Science and Technology,2022

Information Engineering School,Chengyi University College,Jimei University

Abstract： With the rapid development of deep learning technology in the field of natural language processing, chatbots with good human-computer interaction are gradually entering the market. However, most chatbots on the market today are limited to specific domains or specific environments, such as customer service bots. This project is a unique approach to create a chatbot that users can use directly in their daily lives. The chatbot is composed of a text acquisition module, a model creation module, and a reply generation module. In the text acquisition module, we use a third-party corpus, the Qingyun corpus, and use the jieba toolkit to implement the Chinese word separation function. The interface is connected to the back-end of the chatbot model to realize the simulated dialogue function in the form of one-question-and-answer. In order to ensure the proper response function of the chatbot, the effectiveness of the chatbot is evaluated in this paper. The evaluation results show that the chatbot is both reasonable and interesting, and it is a qualified chatting product with deep learning model.

Key Words: Deep Learning; Chatbot; Long Short Term Memory Network Model; Attention Mechanism; Sequence-to-Sequence Model

目 录

第 1 章　绪论

1.1 研究背景及意义

由于最近深度学习技术在自动驾驶、视频、图形和语言等应用中的蓬勃发展，许多开发人员在自然语言领域实现深度学习技术方面取得了重大进展。设计聊天机器人的技术人员使用深度学习算法，因为它们为智能设备提供了卓越的分析和学习能力。一个好的聊天机器人软件结合了神经网络和自然语言处理技术，可以在不同的智能设备上工作。但是，目前市场上的大多数聊天机器人（比如客服机器人）都被限定在特定领域或特定环境中。一般的聊天机器人使用对话界面，以文字或语音的形式与用户交流并处理相应的请求，或为用户提供相应的服务。由于聊天机器人使用的技术具有极高价值潜力，各大企业都在积极开发和设计聊天机器人。但目前大多数聊天机器人在回复中往往会出现各种各样问题，比如遇到不知道答案的问题就回答“是的”、“我爱你”这种安全回复或与上文无关的回答，又或者无法完成连贯的多轮次的交谈[1]。聊天机器人可以代替我们完成的工作有很多，往小了说可以是陪我们聊天，或者播放一首歌曲，往大了说可以替我们完成紧急的工作，比如监控实时温度等。研究聊天机器人的目的是为我们做一些手头的琐碎事情，从而减少人类的工作量。

1.2 聊天机器人的分类

聊天机器人根据对话内容产生的方法可分成两个类型，一种是检索方式，一种是生成方式[1]。检索式聊天机器人的模型需要事先构建问答数据库，当用户输入信息时，聊天机器人通过对输入的信息进行检索和相似语句对比，再从已经建立好的数据库中选择合适的回答返回给用户。这个方法的主要弊端是在回复刻板，缺少新颖性。而和依赖于预先存在的数据库结构的传统的检索型机器人比较，生成型聊天机器人通过采用更深度的编码解码结构，并采用了神经机器翻译技术，而不是传统的自然语言处理过程[2]。生成式聊天机器人是目前研究的一个热点，它从大量的自然语言语料库中学习人类的说话方式，在训练了大量的语言模型样本后，可以使用训练好的模型来生成对用户输入的回复。答案是由聊天机器人自动生成的，可能是学习过的，也有可能是其自主创建的语料库中不存在的语句。

1.3 国内外研究现状

由于国外许多年前就开始对问答系统和聊天机器人进行研究，因此为用户制作了的系列聊天系统都相对成熟，比如 Apple Siri，Microsoft Cortana 等。这些跨平台的智能机器人，利用本公司在大数据、自然语言处理和机器学习方面的技术积累，形成了本家独有的语料库。在模型不断训练的过程中，聊天机器人通过理解聊天数据中的语义和上下文信息，实现超越简单的人机问题和答案的高级智能交互，为用户提供便利和愉悦。

许多海外著名高校和研究机构多年前就开始研究聊天机器人，并在这个领域取得了比较好的成绩[3]。由于国外问答系统的高速发展，近些年国内学者对聊天机器人的研究也越来越重视，发展势头火热。中国有实力的大学和研究机构在聊天机器人的研发方面取得了许多良好的成果，并在人工智能和自然语言处理的重要国际学术会议上提交了大量关于聊天机器人的研究报告。

和国外相比，我国对智能聊天领域的投入规模与研发水平差异相对较大，且进展并不明显，但仍有一些高校在这一领域取得了重大成果。不过值得注意的是，中国高校对于这个范畴的研究大多集中在对于自然语言处理工具的研究方面。在聊天机器人原型的设计中，由哈工大社会计算与信息检索研究中心设计的笨笨聊天机器人，已能够进行简单高效的开放域对话服务[1,3]。

1.4 论文组织结构

本文从绪论，相关技术原理，聊天机器人项目分析，聊天机器人项目设计，聊天机器人效果评估五个章节逐一展开。

第 1 章是绪论。从聊天机器人的研究背景、分类、国内外研究现状对比进行详细说明，并将本论文的组织结构与主要内容进行大致描述。

第 2 章是相关技术原理。介绍了本文为设计的聊天机器人技术所设计的有关技术与基础理论，并针对人工神经网络、循环神经网络、传统 Encoder-Decoder 模块结构、新引入 Attention 机制的 Seq 二 Seq 模型结构[4]等各自的基本原理作出了多方面阐述，为下文的设计分析与实际操作奠定理论基石。

第 3 章是聊天机器人项目分析。根据本文欲设计的聊天机器人自身特点，进行项目结构设计与分析，并对该项目应用的深度学习框架 Tensorflow[5]进行介绍。

第 4 章是聊天机器人项目设计。根据本文所设计的聊天机器人，从文本获取模块、

模型创建模块、回复生成模块三个部分的设计理念进行详细介绍。

第 5 章是聊天机器人效果评估。首先介绍了评估的目的和环境，再以 GUI 界面显示为主要方向，对本文设计的聊天机器人进行了效果测试。

第 2 章　相关技术和理论

2.1 深度学习

深度学习是机器学习课程领域的一个方向，它简化了机器学习的研究。深度学习简单而言，就是由多个层次（包括输入层、隐藏层和输出层）所构成的人工神经网络。深度学习技术实现的重大意义在于，让机器可以模拟人类的行为，如视觉，听力和意识等，克服了模式识别的诸多复杂挑战，从而使人工智能的相关科技的巨大的发展进展步[6]。深度学习理论的目标是创建一种模型，用来模拟大脑神经元间数据传输与信息处理的方式，以解释图像、声音和文本数据等[6]。经典的深度学习模型包括卷积神经网络、循环神经网络、贝叶斯概率生成模型等。本文设计的聊天机器人是由一种特殊的循环神经网络 LSTM 模型搭建的。

2.2 人工神经网络

从上文可以得知，深度学习系统是由多层次的人工神经网络形成，而人工神经网络的定义如下所述。人工神经网络(也称之为神经网络或类神经网络)使用仿生学的思想来创建模仿人脑神经系统的相应模型，以实现特定的功能。这种由许多相互连接的简单的神经元（cell）组成的，形成一个特殊的组织结构就称之为基本神经网络。这个网络依赖于信息系统的复杂性，并透过调节大规模的节点间的关联关系来管理信息[7]。其中最经典的 M-P 神经元模型如图 2.1 所示，它就是以人脑中神经元之间的信息传递过程抽象模拟的。

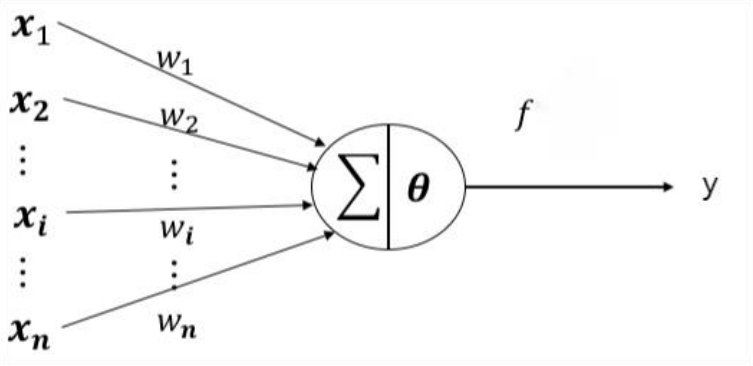

图 2.1 神经元模型

对于一个神经元来说，有i个输入，每一个输入都对应一个权重w，神经元具有一个偏置（阈值）θ，将所有的$i \cdot w$求和后减去阈值得到一个值，这个值就是激活函数的参数，激活函数将根据这个参数来判定这个神经元是否被激活。以上过程可以用公式 2.1 表示。

$$y = f(\sum_{i=1}^{n} w_i x_i - \theta) \tag{2.1}$$

其中：

f为激活函数；

w_i表示第i个神经元的连接权重；

x_i表示第i个神经元的输入；

θ表示阈值或者叫做偏置。

由上可知，神经元的主要功能就是运算输入矢量与权重矢量之间的内积，进而利用非线性传递函数接收输出。人工神经网络，就是由大量相互连接的神经元所构成的复杂功能模型。

2.3 循环神经网络

因为每一次完整的消息连接一次只能有一个输入，使得前一个输入可以完全独立于最后一个输入，但有些任务需要能更好地按次序处理消息，也就是说，前一个输入必须和最后一个输入有关。为克服这种类型所存在的若干问题，并可以更好地管理序列信息，RNN 应运而生[8]。简单地说，输入层x的内容传到隐藏层s之后，并没有直接传输到输出层y，而是通过隐藏层的w矩阵处理后再传给隐藏层的输入，此时再将隐藏层处理的结果传到输出层y，这样就形成了一个有向有环图，从而形成一个循环网络，如图 2.2 为最简单的循环神经网络结构。

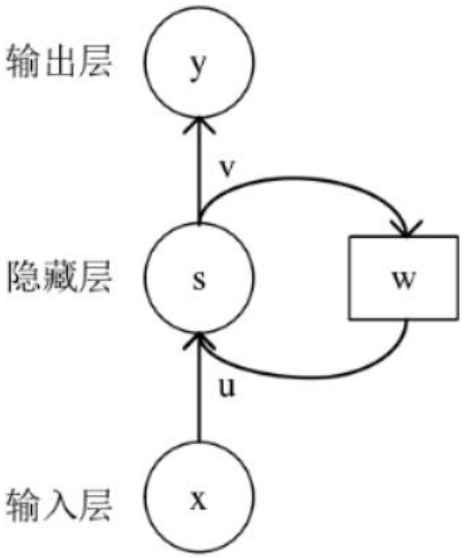

图 2.2 循环神经网络结构

图 2.3 是以时间序列展开的 RNN 网络。其中，$t-1$、t、$t+1$表示时间序列，而s_t代表t时刻隐藏层的状态，x_t表示在t时刻的输入，而o_t表示t时刻的输出值。t时刻的隐藏状态 S_t不仅仅取决于t时刻的输入x_t，也取决于$t-1$时刻的隐藏层状态s_{t-1}。

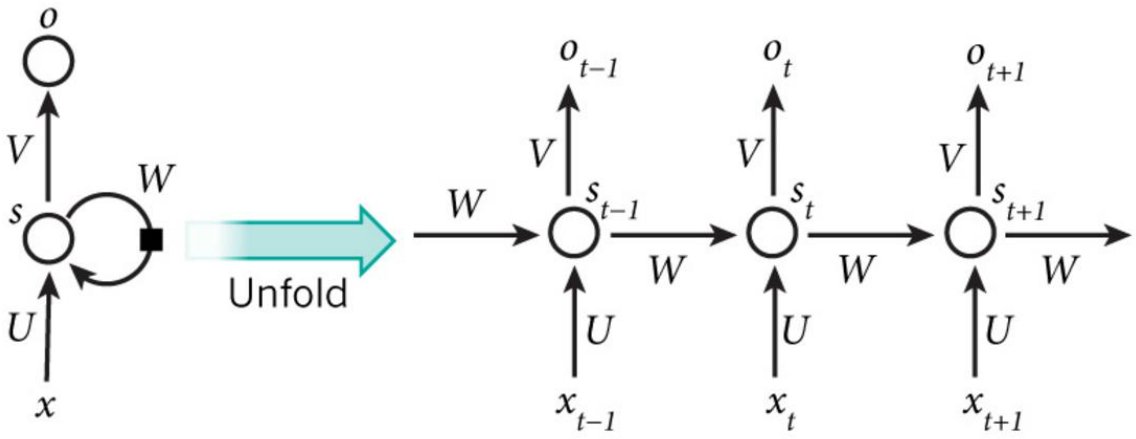

图 2.3 按时间序列展开的 RNN 网络

2.4 特殊的 RNN 模型 LSTM

长短期记忆人工神经网络(Long Short Term Memory Network,LSTM)是一类经过改进的循环神经网络，主要用于解决传统 RNN 系统没法处理长距离依赖时间的状况。它也适用于在处理或者预测时钟序列中间隔的时间延迟值相当长的情形。与传统的 RNN 相比，

LSTM 就是添加了三个门：遗忘门，输入门，输出门。

循环神经网络是有重复功能的链型网络，在经典的 RNN 中，这种重复模块只是一个 tanh 层，构造非常简单。如图 2.4 是一个标准 RNN 的结构。

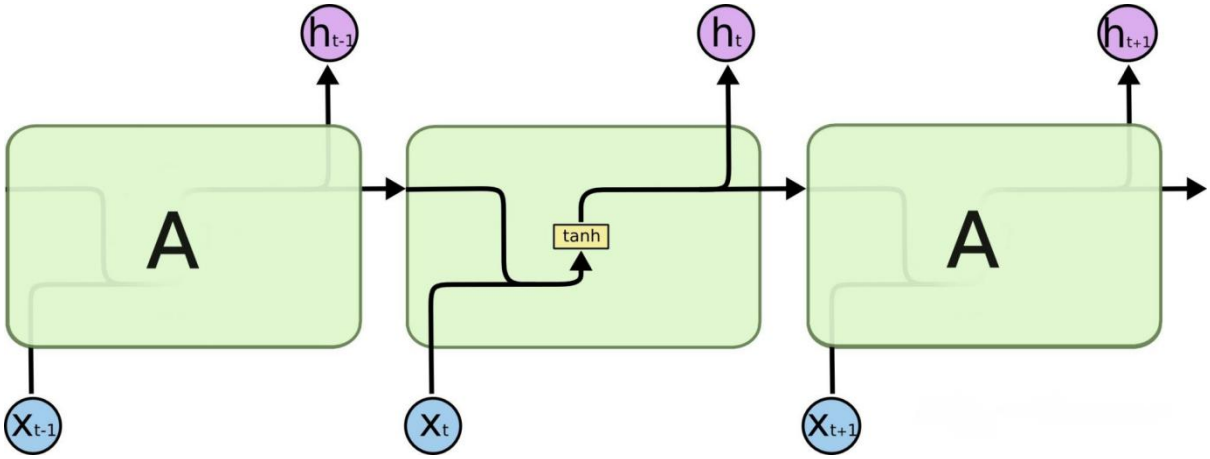

图 2.4 RNN 结构

LSTM 中也存在着这种链式构造。和传统的 RNN 不同，LSTM 的重复单元中存在着不同的构造。它不仅有一个层次的神经网络，而是有四个层次，这四个结构以特定的方式相互作用。如图 2.5 是一个完整的 LSTM 结构。

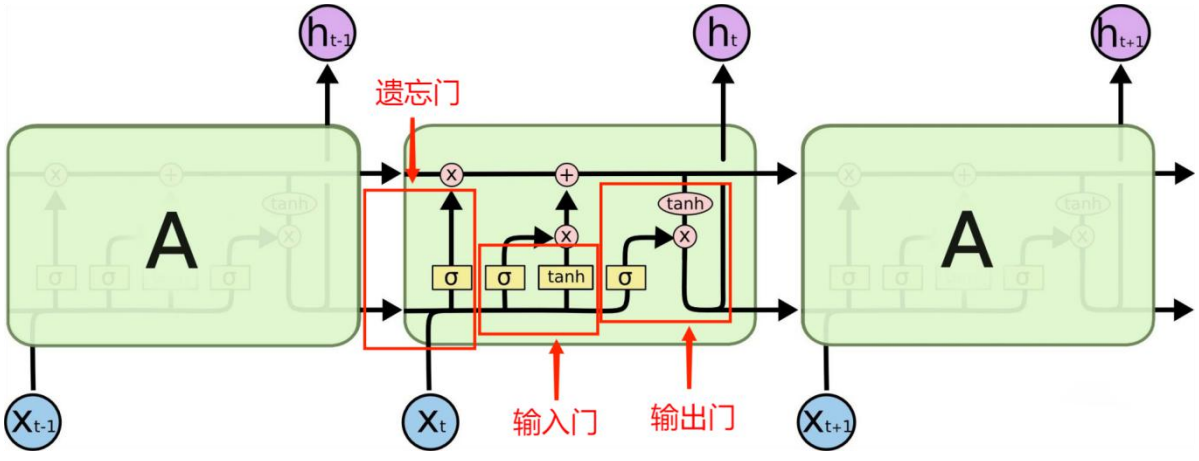

图 2.5 完整的 LSTM 结构

在上图中，由某个节点的输出到其他节点的输入，每条线传输了一组完整的矢量。黄色方框代表了神经网络层;粉色圆点代表逐点计算；合并线代表将两种不同的输入连接到一起，而分叉线代表将处理发送到其他地方。

LSTM 中每个重复模块又称为一个 cell（神经元），贯穿 cell 顶部的水平线代表每一个 cell 的状态，cell 状态的不同也就意味着模块的不同。每个 cell 的状态是沿着整

个链式结构传输的，在传输期间由于存在一些与建模相关的线性运算，所以可以保存时间序列较长的信息，使其保持不变而传到较远距离的位置去，神经元状态如图 2.6 所示。

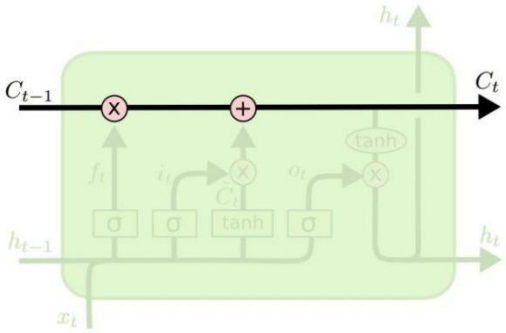

图 2.6 神经元状态

2.5 传统的 Encoder-Decoder 结构

Encoder-Decoder 模块主要由 Encoder 和 Decoder 两个部分所构成，Encoder 又称为编码器，通过编码方法把现实世界的信息都转换成模型中可以理解的内容，也就是说，它把人们所理解的所有输入信息(如文本、图像和声音)都转化为包括所有输入序列的信号的矢量。Decoder 又称为解码器，用来理解编码器所提供的信号，并提供人类理解的真实解决方案，即将先前产生的中间向量转化为文本、声音、图像等序列输出。Encoder-Decoder 架构，可谓是在自然语言文本信息处理研究领域中最常见的一种研究模型，使用限制较少、适用范围广，最先使用在机械编译研究领域，输入某一个序列，将其转换输出为另外一个序列[9]。如图 2.7 是文本处理领域里常用的 Encoder-Decoder 框架最常见的一种抽象的图像表示。

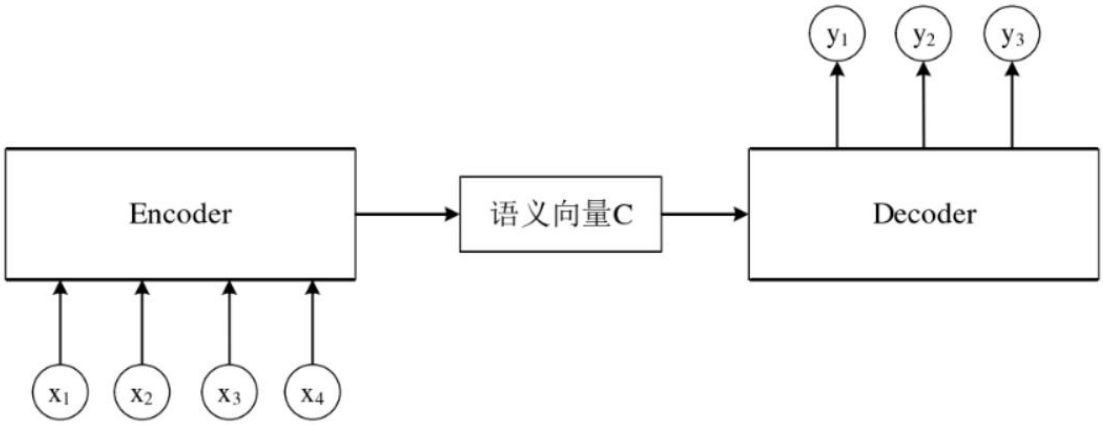

图 2.7 典型的 Encoder-Decoder 框架

聊天机器人最核心的功能就是按照用户的输入信号，得到对应的输出信号，因此可通过 Encoder-Decoder 模式实现搭建。对于给出的语句<X，Y>，X 代表用户输入的文本序列，Y 则代表聊天机器人所返回的文本序列。当用户将语言的 X 插入模型中时，由编码器按照其意思将 X 压缩成中间语义向量 C，而解码器再根据中间语义向量 C 生成 Y 响应。这就形成了一种实际的聊天机器人模式。这样就构建了一个实际的聊天机器人模型。

2.6 基于注意力机制的 Seq2seq 模型

序列到序列（Sequence to Sequence，Seq2seq)模型的本质思想是，先将序列转化为一条确定长度的矢量并注入到编码器中，进而将编码器上的所需进行解码的矢量，经由解码器转化为所需的顺序输出。从某种意义上说，Seq2seq 和 Encoder-Decoder 都是基于 LSTM 机制的模型，但 Seq2seq 更强调目的，不特指具体方法，满足有输入序列转换输出序列的结构都可以成为 Seq2seq 模型，常见应用包括但不仅限于聊天机器人、机器翻译等；Encoder-Decoder 更强调方法，常说的 Seq2seq 就属于 Encoder-Decoder 的大范畴。引入 Attention 机制主要是用来处理信息时间过长，信息容易丢失的问题。Attention 模型的优点是，编码器可以不再把整个输入顺序都解码为固定长度的中间矢量 C，而且把解码为单一矢量的序列[4,11]。引入了注意力机制的 Encoder-Decoder 模型如图 2.8 所示。Attention 机制的主要实现方法，就是先将 LSTM 编码器的中间输出结果 c 存储到输入序列上，而后不断训练模型并有选择的学习这种输入，同时在模型输出时把输入和输出序列与之加以联系。

换一种视角来看，在输出序列中的每一个项目的产生概率决定于在输入序列中选择了什么项目。没有注意力机制的 encoder-decoder 结构，一般将 encoder 的最后一种状态，当做整个 decoder 的输入(可能作为初始化，也可能作为每一时刻的输入)，不过由于 encoder 的内容毕竟是很有限的，保存不了太多的信息。对于 decoder 过程，每一个步骤都和之前的输入都没有关系了，只与这个传入的内容有关。注意力机制引入之后，decoder 根据时刻的不同，让每一时刻的输入都有所不同。

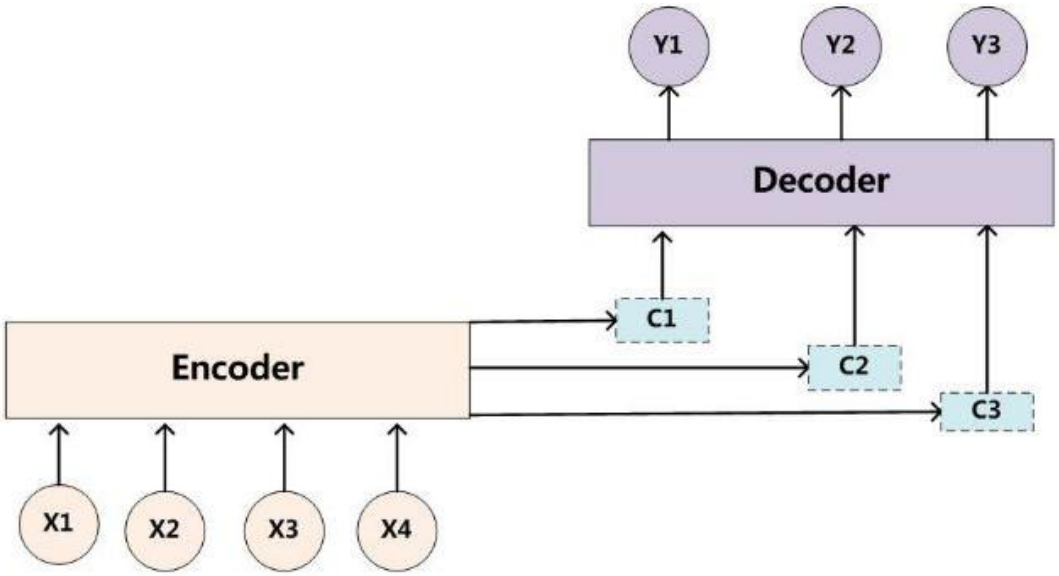

图 2.8 基于注意力的序列到序列（Sequence to Sequence)模型

第 3 章　聊天机器人项目分析

3.1 聊天机器人的需求分析

由于当前聊天机器人在市场上有极高的价值需求，因此，根据聊天机器人的分类[1]，本文设计的目标是在开放领域内生成式的闲聊型机器人，如图 3.1 所示。由于该聊天机器人是生成模式的聊天机器人，不是基于检索模式的，因此不会产生对数据库的依赖。根据这一特点，创建一个闲聊型的机器人首先要做的是构建一个生成式的模型，再对模型进行预训练，使聊天机器人学会日常生活中人类的表达方式。

而开放领域的聊天机器人，不像封闭领域那样需要把聊天的话题限定在某一范围内，比如计算机领域或医疗领域，且主要功能为与用户闲聊，因此需要选择贴近生活的语料进行训练，防止出现在对话过程中出现专业名词，导致用户无法理解聊天机器人回复的内容。

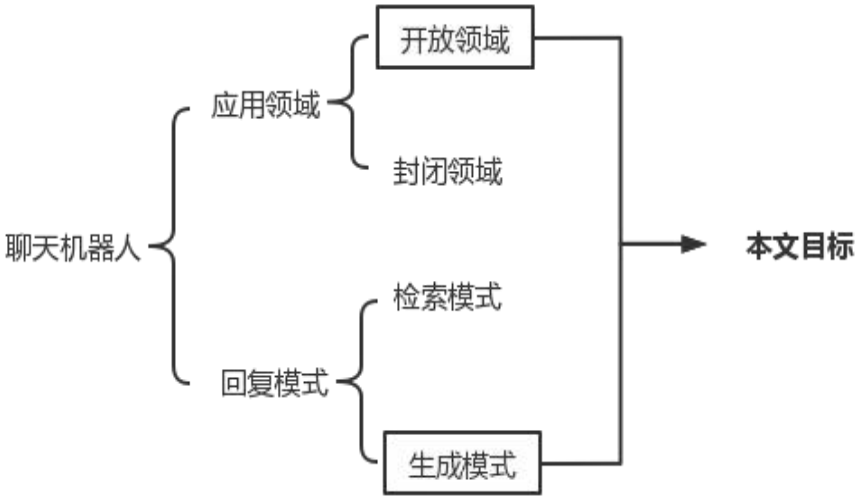

图 3.1 本文设计目标

3.2 聊天机器人的功能分析

本文设计的聊天机器人如图 3.2 所示，主要分为三个模块：第一个模块是文本获取模块，主要工作是将语料中问题和答案处理成机器可以理解的序列从而创建模型；第二部分是模型创建模块，主要工作是将处理好的 QA 序列，利用特殊的循环神经网络 LSTM 进行模型的训练[10]；第三部分是带 GUI 界面的回复生成模块，该模块的特点是引入了注

意力机制以提高回复的准确性。

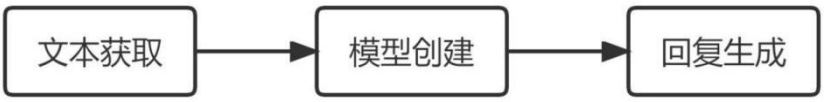

图 3.2　聊天机器人的总体框架

在文本获取模块中，本文设计的聊天机器人采用的文本内容是从网络中获取的青云语料库，该库与其他语料库相比较生活化，适用于日常闲聊。

在模型创建模块中，本文设计的聊天机器人的生成式模型是基于 TensorFlow+Keras 框架创建的。TensorFlow 是人工智能在 AI 应用领域的一种主要应用软件开发工具，它是由谷歌公司研发的开源应用软件[5]。人工智能技术主要包括三个层面，即基础层、技术层和应用层，TensorFlow 则是指技术层面上的学习框架。Keras 是一种由 Python 创建的开源人工神经网络库，经常用于 TensorFlow 的高级应用程序接口，主要用于深度学习模块的设计、测试、分析、应用。

在回复生成模块中，为聊天机器人配置了一个对话界面，该界面使用 Python 的标准库 Tkinter 实现。

第 4 章　聊天机器人项目设计

4.1 文本获取模块设计

收集更大规模的训练数据也是深度学习中需要处理的一个问题，训练样本的优劣直接影响模型的最后结论[9]。获得训练数据也就是语料的方式有很多种，比如在网上获得第三方语料库，或是使用一些软件自行制作语料。通常情况下，自主创建语料库适用于需要创建特定领域的语料，比如金融领域或医学领域的语料。由于本文设计的聊天机器人主要用于闲聊，因此本文使用第三方库青云语料，该语料适合创建小规模的闲聊机器人。

实现该模块的第一步是将下载的第三方语料进行处理，将其拆分为问题和答案两个部分，便于训练以及实现对话效果，并将文本进行分词。因为汉字不像英文一样天生就用空格分隔词汇，所以必须先要把汉语文字中的字符串切分为合理的词汇序列，然后再在此基础上为不同的词语添加索引[10]。本文使用第三方工具包 jieba 实现中文分词工作。

第二步是将文本矢量化，将分词后每一个不同的字符转换为不同的向量。

第三步是将对文本序列进行预处理，因为 keras 只允许相同的序列长度相等的输入，所以就必须把原序列转换为经过填充之后的另一个等长的新序列。

如下代码段为序列处理过程。本模块设计总框架如图 4.1 所示。

```
pad_question = sequence.pad_sequences(pad_question,
                                      maxlen=maxLen,
                                      dtype='int32',
                                      padding='post',
                                      truncating='post')
pad_answer = sequence.pad_sequences(pad_answer_a,
                                    maxlen=maxLen,
                                    dtype='int32',
                                    padding='post',
                                    truncating='post')
with open('pad_word_to_index.pkl', 'wb') as f:
    pickle.dump(word_to_index, f, pickle.HIGHEST_PROTOCOL)
with open('pad_index_to_word.pkl', 'wb') as f:
    pickle.dump(index_to_word, f, pickle.HIGHEST_PROTOCOL)
np.save('pad_question.npy', pad_question)
np.save('pad_answer.npy', pad_answer)
```

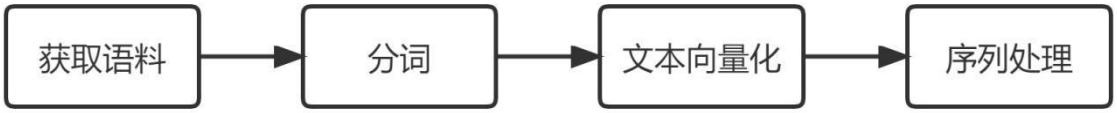

图 4.1 文本获取模块框架

4.2 模型创建模块设计

而在自然语言处理与深度学习领域中最重要的部分，是聊天模型的构建。把深度学习的技术融合到自然语言处理中，系统将会减少对问答数据库的依赖性，并利用对已处理好的大量数据进一步的训练以增强数据处理能力，而系统也将能够更加精确地和用户交流，这便是生成式的回复模型[7,10]。

在建立模型之时，若用直接用的 fit 函数方法来训练建模，则是必须传入全部的训练数据，而且一旦训练数据太过大将会极其浪费内存，所以利用 keras 提供的另一个方法 fit_generator，可以利用该函数方法分批地读取训练数据以节约内存，而唯一要做的事便是构建一个生成器[11]。生成器的核心是一个 yield 关键字，训练模型就是使用生成器逐批迭代生成不同的数据，代码如下所示。在本模块中使用到的激活函数为 Softmax，它常用于将一个数值向量归一化为一个概率之和为 1 的概率分布向量。Softmax 常常和交叉熵函数一起使用，在本文中亦是如此。

```
def generate_train(batch_size):
print('generate_train()')
    steps=0
    question_ = question
    answer_ = answer
    while True:
        batch_answer_o = answer_o[steps:steps+batch_size]
        batch_question = question_[steps:steps+batch_size]
        batch_answer = answer_[steps:steps+batch_size]
        outs = np.zeros([batch_size, maxLen, vocab_size], dtype='float32')
        for pos, i in enumerate(batch_answer_o):
            for pos_, j in enumerate(i):
                if pos_ > 20:
                    print(i)
                outs[pos, pos_, j] = 1
        yield [batch_question, batch_answer], outs
        steps += batch_size
        if steps == 100000:
            steps = 0
```

```
model.fit_generator(generate_train(batch_size=100),
                    steps_per_epoch=1000,
                    epochs=200,
                    verbose=1,
                    callbacks=callbacks_list,
                    class_weight=None,
                    max_queue_size=5,
                    workers=1,
                    use_multiprocessing=False,
                    shuffle=False,
                    initial_epoch=initial_epoch_
                    )
```

第二步通过模型来训练词向量——首先将嵌入矩阵作为模型中的一部分，再利用词和词之间的共现及上下文关联来调整模板参数，最后所得的矩阵便是词表中每个词的词向量[10]。本模块总体设计框架如图 4.2 所示。

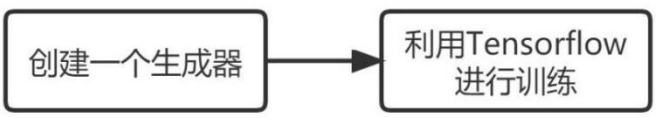

图 4.2 模型创建模块总体框架

4.3 回复生成模块设计

传统的 Encoder-Decoder 模式也是一个通用的模式，特别适合于处理从一个序列得到另一种序列的情形。在生成式聊天机器人系统中，输入方使用编码器把消息编码在中间语义向量 C 中，而后再使用解码器分析中间语义向量 C 中包含的消息，并将其转化为输出[12,14]。但由于一般的 Encoder-Decoder 模式不能很好的处理长文本序列，输入的靠后位的内容往往会把靠前面的信息掩盖，这也是人们通常所说的长距离依赖现象。具体而言就是指，在预测的位置上远离所依赖的相关数据时，就会很难学习到新的信息数据[12]。因此在 Seq2seq 模型的基础上引入注意力机制。Attention 机制的模型，实质上是在不同的输入输出信号间进行了适当的调整，从而动态地体现当前层的最终输入反馈到对网络的下一层或输出层[13,14]。

本文中所构建的聊天机器人利用的基本原理是对序列到序列模式之间加入注意力机制，使模型在产生序列时可以同时注意编码器中的所有信号，更准确的来说是重要的信息，而不仅仅是序列位置靠后的信息。本文所建立的生成式模型为，在编码器上将输入的消息转换为词向量并进行词矢量编码，以此完成回复语言的生成，语言的对话结构如图 4.3 所示。

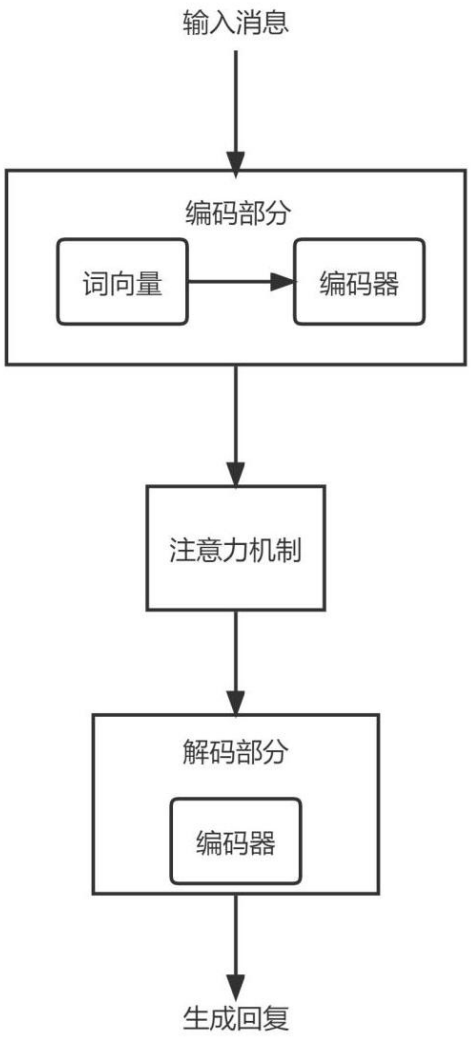

图 4.3 语言生成结构

第 5 章　聊天机器人效果评估

5.1 评估目的

前面几章从聊天机器人的总体设计要求出发，详尽介绍了在聊天机器人中所有模块的实现流程。本章将从聊天功能的角度评价聊天机器人系统的效果，以确保每个模块完整可用。本文的效果评估使用人工评估的方法。因为在对话情景中，每一百人就有一百种不同的回答，而很多千变万化的回答可能都是很好的，但无法用好与不好来当作衡量标准，所以并非说生成的对话必须和训练语料的内容越贴近才越好[15]。

5.2 评估环境

本文中设计的聊天机器人使用了 Python 编程语言进行系统的编程，并通过 TensorFlow+Keras 的框架进行深度学习模型的开发与使用，IDE 设计使用了 Anaconda Navigator，操作系统使用环境为 Windows 10。具体硬件配置如表 5.1：

表 5.1 硬件配置表

硬件名称	型号
处理器	Intel(R) Core(TM) i5-7300HQ CPU @ 2.50GHz
显卡	NVIDIA GeForce GTX 1050
内存	12.0 GB 2400 MHz DDR3
硬盘	128G SSD+1TB HDD

5.3 聊天机器人具体模块评估

由于本文设计的聊天机器人代码较复杂，以下评估内容只截取部分关键代码及效果，并不意味着其他部分不能实现。

5.3.1 文本数据获取模块评估

本文使用的第三方语料库是青云语料库，该库包含近 110，000 万的对话内容，在本模块第一步是需要导入该语料。导入部分源码如下所示。图 5.3.1 表示成功导入。

```
with open('qingyun.tsv', 'r', encoding='utf-8') as f:
    lines = f.read().split('\n')
    lines = lines[:-2]
question = []
answer = []
for pos, line in enumerate(lines):
    if '\t' not in line:
        print(line)
    line = line.split('\t')
    q = line[0].strip()
    a = line[1].strip()
    question.append(' '.join(jieba.lcut(Traditional2Simplified(q).strip(), cut_all=False)))
    answer.append(' '.join(jieba.lcut(Traditional2Simplified(a).strip(), cut_all=False)))
print(len(question))
print(answer[:10])
```

```
23 print(len(question))
24 print(answer[:10])

   105575
   ['在 这里 了', '我 吃 电 的 哦', '问 吧 ， 听 着 呢', '好话 不 分 轻重 ！ ', '
   早死 晚死 都 得 死 ！ ！ ！ ', '真是 不好意思', '噢 是 吗 ？ 那么 你 也 差不多 了
   ', '我 不 抠', '看 窗外 ~', '你 眼神 不好']
```

图 5.1 语料导入成功

利用 pickle 模块将列出的词汇索引转化二进制文件，以编译器能够理解的形式保存和读取，并列出词汇表大小。相关操作代码如下所示。

```
counts = {}
BE = ['BOS', 'EOS']
for word_list in question + answer + BE:
    for word in word_list.split(' '):
        counts[word] = counts.get(word, 0) + 1
word_to_index = {}
for pos, i in enumerate(counts.keys()):
    word_to_index[i] = pos
index_to_word = {}
for pos, i in enumerate(counts.keys()):
    index_to_word[pos] = i
vocab_bag =list(word_to_index.keys())
with open('word_to_index.pkl', 'wb') as f:
    pickle.dump(word_to_index, f)
with open('index_to_word.pkl', 'wb') as f:
    pickle.dump(index_to_word, f)
with open('vocab_bag.pkl', 'wb') as f:
    pickle.dump(vocab_bag, f)
```

5.3.2 模型创建模块评估

模型创建由于对电脑配置较高且需要大量的时间，本模型只是一个很小的训练结果，模型训练过程如图 5.2 所示。模型训练对计算机的硬件系统要求极高，表中的配置硬件使用情况如图 5.3 所示。

```
Epoch 1/200

**********************************generate_train()*********************************
1000/1000 [==============================] - 3946s 4s/step - loss: 4.5094

Epoch 00001: loss improved from inf to 4.50938, saving model to C:/Users/Zxx/Desktop/final/models/model512.h5
Epoch 2/200
1000/1000 [==============================] - 4058s 4s/step - loss: 3.1590

Epoch 00002: loss improved from 4.50938 to 3.15900, saving model to C:/Users/Zxx/Desktop/final/models/model512.h5
Epoch 3/200
```

图 5.2　模型训练

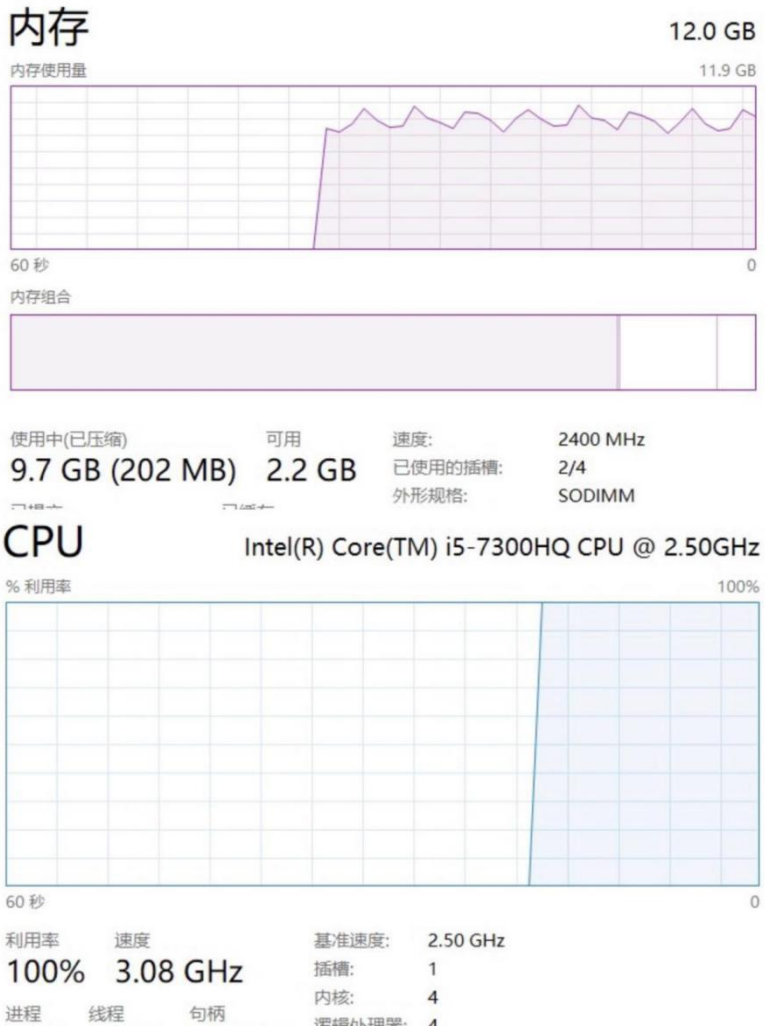

图 5.3 训练时的硬件使用情况

5.3.3 回复生成模块评估

对输入的语句进行分词并转化为长度相同的序列代码如下所示。

```
def input_question(seq):
    seq = jieba.lcut(seq.strip(), cut_all=False)
    sentence = seq
    try:
        seq = np.array([word_to_index[w] for w in seq])
    except KeyError:
        seq = np.array([36874, 165, 14625])
    seq = sequence.pad_sequences([seq], maxlen=maxLen,
                                 padding='post',
                                 truncating='post')
    print(seq)
    return seq, sentence
```

引入 attention 机制的输出序列预测代码如下所示。

```
def decode_greedy(seq, sentence):
    question = seq
    answer = np.zeros((1, 1))
    attention_plot = np.zeros((20, 20))
    answer[0, 0] = word_to_index['BOS']
    i=1
    answer_ = []
    flag = 0
    encoder_lstm_, question_h, question_c = question_model.predict(x=question,
verbose=1)
    while flag != 1:
        prediction, prediction_h, prediction_c, attention = answer_model.predict([
            answer, question_h, question_c, encoder_lstm_
        ])
        attention_weights = attention.reshape(-1, )
        attention_plot[i] = attention_weights
        word_arg = np.argmax(prediction[0, -1, :])#
        answer_.append(index_to_word[word_arg])
        if word_arg == word_to_index['EOS']    or i > 20:
            flag = 1
        answer = np.zeros((1, 1))
        answer[0, 0] = word_arg
        question_h = prediction_h
        question_c = prediction_c
        i += 1
    result = ' '.join(answer_)
    attention_plot = attention_plot[:len(result.split(' ')), :len(sentence)]
    plot_attention(attention_plot, sentence, result.split(' '))
    return ' '.join(answer_)
```

本文设计的聊天机器人界面以 Python 的内置库 Tkinter 实现，主要代码如下所示。

```
def show():
    strMsg = "  我:" + time.strftime("%Y-%m-%d %H:%M:%S", time.localtime()) + '\n '
    t1_Msg.insert("end", strMsg, 'green')
    seq = t2_sendMsg.get('0.0', 'end')
    t1_Msg.insert("end", seq)
    t1_Msg.tag_add('me_msg', 0.0, "end")
    print(' ' + strMsg + seq)
    seq, sentence = input_question(seq)
    answer = '      ' + decode_greedy(seq, sentence)
    answer = ''.join(answer.split(' ')[:-1])
    rplMsg = "水子:" + time.strftime("%Y-%m-%d %H:%M:%S", time.localtime()) + '\n'
    t1_Msg.configure(t1_Msg, state = 'normal')
    t1_Msg.insert("end", rplMsg + answer + "\n" ,'blue')
    t2_sendMsg.delete('0.0', "end")
```

界面效果展示如图 5.4 和图 5.5 所示。图 5.4 是一个弹窗，简单介绍了该聊天机器人的来源和用途。图 5.5 是打开的空界面，同时也是清屏后的界面。

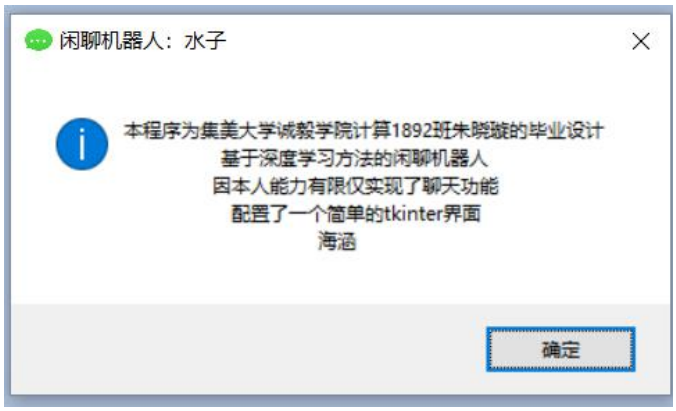

图 5.4 弹窗界面

图 5.5 聊天机器人界面

最后对聊天机器人进行回复测试，测试结果表明该聊天机器人回复功能正常，如图 5.6 所示。

图 5.6 聊天机器人回复功能测试

结 论

本文主要内容是将深度学习的技术应用于生成式聊天机器人中，并且取得了不错的成效。本文设计的聊天机器人由文本处理模块、模型构建模块、回复生成模块三部分组成，三个模块均使用深度学习的模型搭建。其中，Encoder-Decoder 模型是构建聊天机器人时的一个最常见的模型结构，而本文所使用的自然语言处理模块则通过引入 Attention 机制，改善了原有的 Encoder-Decoder 模型所存在的缺陷。主要工作包括：

1.对聊天机器人的技术方面做出详细介绍。文章主要从两个角度出发，分别研究了聊天机器人模型和深度学习中有关的技术与思想，最后把深度学习技术运用到了生成式的聊天机器人系统上。

2.建立生成式模型系统。本文主要分析和修正了传统 Encoder-Decoder 模式中出现的长距离依赖问题。利用 Attention 机制，为 Encoder-Decoder 模块提取多个语义编码，使得模块在产生序列的同时可以有选择性地关注编码设备中的相关数据。

3.设计实现聊天机器人。通过构建好的生成式模型，为聊天机器人设计一个聊天界面。该界面与聊天机器人的后端实时连接，产生一个类似于与人聊天的一问一答形式的对话界面。该界面以 Python 的标准库 Tkinter 实现。

当然，由于水平和精力有限，本文所设计的聊天机器人略有一些缺陷，可以改进的地方有很多，例如：

1.欠缺多模态信息。目前的聊天机器人功能主要集中在文字信息上，而现如今聊天的信息内容中已包含了文字、图片、视频等多种类型信息内容，因此聊天机器人也可根据更多模态信息内容来改善模型效果。

2.安全回复问题。在从序列到序列模式中，关于测试人员所提问的答案仍然有比较朦胧的不相应的"安全回答"，至于有些耐人寻味的提问，得出的回答也仍然差强人意，而且回答的品质也尚有待进一步提高。

3.功能较少。本文设计的聊天机器人仅具有闲聊的功能，不具备其他功能，例如查询天气，后台控制等，且无法移植到其他系统中使用，欠缺之处皆因本人能力有限。

聊天机器人技术在当前市场上尚需更深入发展，仍面临着技术瓶颈，语料数据库的建立也需进一步完善。

致 谢

时光匆匆，四年的本科生涯仿佛一眨眼就过去了。时间回到四年前，我独自一人跨越四千多公里，从新疆一个偏远的城市来到美丽的海滨城市厦门，在这里开启了我的求学生涯。俗话说“城在海上，海在城中”，厦门的的确确是一个非常美丽的城市，无论是生态环境还是人文环境，都是偏远地区无法比拟的。在这四年我对这个世界的认知也随着我的年龄不断增长，心智也越来越成熟。如今想到即将离开这个包含我四年青春回忆的地方，心中有百般不舍，但也有对未来生活的期待。当然，最多的还是对身边人的感激之情。

首先我要感谢的是我的父母。五年前高考失利，自己都想放弃自己了，是我的父母不断地给予我信心，坚定地鼓励我复读。如果不是他们，就不会有今天在这写论文的本科生，也不会有一个懂得感恩的女儿，更不会有一个懂得承担责任的大写的人。

其次我要感谢这四年教书育人给予我专业知识的老师们，尤其是我的指导老师颜庆茁老师。第一次见到她听她讲课是大三修的操作系统，当时我就暗下决心一定要找这个老师指导我的毕业设计。没想到梦想成真，在毕业设计和论文阶段都受到了颜老师的热情指点，同时在我考研路上也给予了我不少帮助。

我还要感谢我的室友。这四年来我不仅品尝到了全国各地的美食，也体会了全国各地尤其闽南地区的风土人情。最主要的是，我的室友们四年来对我的无比包容使我体会了如家一般的温馨，我们一起学习，一起玩耍，一起面对台风，一起对抗疫情，即便毕业后各奔东西，我也不会忘记那些年我们一起经历过的春夏秋冬。

也要感谢的是不断努力向上的自己。如果没有足够强大的心理承受能力，也不会选择在众多压力之下继续开启求学之路。

非常感谢我的母校集美大学诚毅学校带给了我珍贵的学习机会，使我在这个大学的平台上进一步锻炼心智，成就才华。“诚以待人，毅以处事”不仅是校训，更是我为人处世的准则，感谢诚毅！

参考文献

[1] 王艳秋,管浩言,张彤.聊天机器人的分类标准和评估标准综述[J].软件工程,2021,24(02):2-8.

[2] Ziegler Z M， Melas-Kyriazi L， Gehrmann S，et al. Encoder-Agnostic Adaptation for Conditional Language Generation[J]. 2019.

[3] 王浩畅，李斌.聊天机器人系统研究进展[J].计算机应用与软件,2018,第35卷(12): 1-6，89

[4] 廖文雄,曾碧,徐雅芸.结合一维扩展卷积与Attention机制的NLP模型[J].计算机工程与应用,2021,第57卷(4): 114-119.

[5] Pang, Bo;Nijkamp, Erik;Wu, Ying Nian.Deep Learning with TensorFlow: A Review[J].Journal of Educational and Behavioral Statistics,2020,Vol.45(2): 227-248.

[6] 陈先昌.基于卷积神经网络的深度学习算法与应用研究[D].浙江工商大学,2014.

[7] 张驰，郭媛，黎明.人工神经网络模型发展及应用综述[J].计算机工程与应用,2021,第57卷(11): 57-69.

[8] 王昭武. 基于深度学习的图像超分辨算法研究[D].西安电子科技大学,2021.DOI:10.27389/d.cnki.gxadu.2021.001586.基于神经网络的真空断路器状态诊断.

[9] 赵鸿阳.基于深度学习的智能聊天机器人的研究与实现[J].智能计算机与应用,2019,9(06):308-311.

[10] 李素建,李芸,纪鹭宁,徐睿峰. 词典匹配和串频统计相结合在自动主题分析中的应用[C]//.全国第八届计算语言学联合学术会议（JSCL-2005）论文集.,2005:684-686.

[11] Han Songhee,Lee Min Kyung. FAQ chatbot and inclusive learning in massive open online courses[J]. Computers & Education,2021(prepublish).

[12] 曹东岩. 基于强化学习的开放领域聊天机器人对话生成算法[D].哈尔滨工业大学,2017.

[13] 周震卿,韩立新.基于TextCNN情感预测器的情感监督聊天机器人[J].微型电脑应用,2019,35(05):104-106+110.

[14] 杨丰瑞,霍娜,张许红,韦巍.基于注意力机制的主题扩展情感对话生成[J].计算机应用,2021,41(04):1078-1083.

[15] 杨晔.基于深度学习的聊天机器人的研究[J].信息技术与信息化,2020(03):158-1593.