

# 华中科技大学

## 本科生毕业设计

### 基于深度学习的情绪感知对话回复生成 关键技术研究

院 系 计算机科学与技术

专业班级 CS1703

姓 名 范唯

学 号 U201714670

指导教师 魏巍

2021 年 6 月 1 日

## 学位论文原创性声明

本人郑重声明：所呈交的论文是本人在导师的指导下独立进行研究所取得的研究成果。除了文中特别加以标注引用的内容外，本论文不包括任何其他个人或集体已经发表或撰写的成果作品。本人完全意识到本声明的法律后果由本人承担。

作者签名：                                年    月    日

## 学位论文版权使用授权书

本学位论文作者完全了解学校有关保障、使用学位论文的规定，同意学校保留并向有关学位论文管理部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权省级优秀学士论文评选机构将本学位论文的全部或部分内容编入有关数据进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

本学位论文属于 1、保密 ☐，在    年解密后适用本授权书。

2、不保密 ☐。

(请在以上相应方框内打“√”)

作者签名：                                年    月    日

导师签名：                                年    月    日

## 摘 要

目前，人工智能以及大数据从方方面面不断地影响着人们的生活，比如目前比较流行的智能聊天机器人。许多真实世界的智能语音机器人已经能够准确完成很多基本的任务型对话，足以证明目前人工技术的发展已经取得长足进步。但是，在很多闲聊性对话中，智能聊天机器人目前对话结果不太理想，比如个性一致性、风格一致性、情感交互方面等，尤其是情感交互方面，人与人的日常交流主要依赖于情感交互，因此在目前已有的人机交互中加入情感因素，使得智能聊天机器人能够感知情感，并实现带有情感的回复，能够极大的提升人机交互体验，这也是工业界和学术界共同关注的热门研究问题之一。

因此，本文重点研究基于深度学习的情绪感知对话回复生成技术，其在已有序列到序列的对话回复生成模型基础上，考虑引入情绪自动选择模块，实现端到端的自动化情绪感知回复生成系统。具体地说，本文主要工作是在传统的基于序列到序列的文本生成模型基础上，考虑对文本中的情绪进行预测，并进行情感编码，通过情绪嵌入的方式与文本原本语义编码进行融合，并利用注意力机制进行加强以实现基于自动化情绪感知的文本对话生成模型，其能够有效感知对话中的情绪并选择合适情绪融入生成的对话回复中。另外，在前端展示界面则考虑采用 Vue.js 设计实现，同时后端服务器端考虑使用 Django 框架，上述系统不仅能直观的展现整个历史对话过程，还能够直观的提升人机交互的舒适性和流畅性，极大提升用户体验。

通过真实实验显示，提出的模型不仅在回复多样性及相关性上较已有系统有大幅提升，且生成的回复在情感上较以往系统更丰富，较好的解决现有人机对话系统中生成回复单一、通用等问题。

**关键词：**情感嵌入；语义嵌入；情感对话生成；情感对话机器人；注意力机制；序列到序列模型；

## Abstract

At present, artificial intelligence and big data continue to affect people's lives in all aspects, such as the currently popular intelligent chat robots. Many real-world intelligent chat robots have been able to accurately complete many basic task-based dialogues, which is enough to prove that artificial technology has made great progress. However, in many small chat conversations, the current dialogue results of intelligent chat robots are not ideal, such as personality consistency, style consistency, emotional interaction, etc, especially in emotional interaction. Daily communication between people mainly depends on emotional interaction. Therefore, adding emotional factors to the existing human-computer interaction so that intelligent chat robots can perceive emotions and realize emotional responses can greatly enhance the human-computer interaction experience. Therefore, it is also the common concern of industry and hot academia research.

Consequently, this article focuses on the deep learning-based emotion-aware dialogue response generation technology. Based on the existing sequence-to-sequence dialogue response generation model, it considers the introduction of an automatic emotion selection module to build an end-to-end automatic emotion-aware response generation system. Specifically, the main work of this paper is based on the traditional sequence-to-sequence text generation model, considering the prediction of emotions in the text and encoding the emotions by means of emotion embedding and fusing the original semantic coding of the text. And dialogue generation model based on automatic emotion perception, which can effectively perceive the emotions in the dialogue and select appropriate emotions to incorporate into the generated dialogue replies, is strengthened by using attention mechanism. In addition, the front-end display interface is designed and implemented by Vue.js, and the back-end server is intended to use the Django framework. The above-mentioned system can not only intuitively display the entire historical dialogue process, but also enhance the comfort and smoothness of human-computer interaction, which greatly improves the user experience.

Real experiments show that the proposed model not only greatly improves the diversity and relevance of responses than existing systems, but also generates more

emotional responses than previous systems, which is a better solution to the existing human-machine dialogue system generating single, universal responses and other issues.

**Key Words:** emotional embedding, semantic embedding, emotional dialogue generation, emotional chatbot, Attention Mechanism, Sequence to Sequence

## 目 录

摘要 . . . . .	I
Abstract . . . . .	II
1 绪论 . . . . .	1
1.1 课题背景 . . . . .	1
1.2 国内外研究现状 . . . . .	1
1.3 研究目的和主要内容 . . . . .	2
1.4 论文结构 . . . . .	3
2 技术背景概述 . . . . .	5
2.1 词嵌入 . . . . .	5
2.2 循环神经网络 . . . . .	8
2.3 序列到序列模型 . . . . .	9
2.4 注意力机制 . . . . .	11
2.5 本章小结 . . . . .	12
3 基于注意力机制 (Attention) 的情感对话生成系统设计 . . . . .	14
3.1 基于融合预测网络的情感选择模块 . . . . .	15
3.2 基于内部外部记忆的回复生成模块 . . . . .	22
3.3 损失函数 . . . . .	25
3.4 设计中考虑的制约因素 . . . . .	26
3.5 成本估算 . . . . .	27
3.6 本章小结 . . . . .	28
4 基于注意力机制 (Attention) 的情感对话生成系统实现 . . . . .	30
4.1 系统主体架构实现 . . . . .	30
4.2 基于融合预测网络的情感选择器 . . . . .	34
4.3 基于内部外部记忆的回复生成器 . . . . .	35
4.4 本章小结 . . . . .	38
5 性能评估与分析 . . . . .	39
5.1 测试环境与方案 . . . . .	39

5.2	自动评估	40
5.3	人工评估	42
5.4	系统界面	43
5.5	本章小结	46
<b>6</b>	<b>总结与展望</b>	<b>47</b>
6.1	个人总结	47
6.2	未来展望	47
	致谢	49
	参考文献	50

## 1 绪论

### 1.1 课题背景

在自动对话系统日益完善的今天，人机对话已经在日常生活中非常普及了，比如 Siri、小度、小爱同学、天猫精灵等等，但是业界中普及的一般是传统的问答模式，用户能很明显的感受到这是来自机器人的回复，简单直接且富有理性。但是，研究发现在对话中加入情感因素，会大幅度提高整个对话的流畅程度和用户满意度。

目前现存的情感对话生成系统，大多是基于朱小燕、黄民烈老师团队开发的 Emotion Chat Machine<sup>[1]</sup> 的一些变体，虽然能生成富有情感的回复，但是存在的问题是非常依赖于手动选择情感，在用户使用体验上较差。所以该类系统需要改进的地方是（1）生成的回复富有情感且完全不依赖于手动标注；（2）回复具有多样性并与主题相关；（3）避免统一而通用的回复。基于此，对传统 ECM 模型进行优化改进有着重要的意义。

### 1.2 国内外研究现状

现有的对话生成模型主要是基于序列到序列架构 (Seq2seq)，这种架构可以有效的解决序列映射问题，且被广泛用于各大领域，其中主要包括机器翻译和图像标题提取。

首先在原生 Seq2seq 架构的研究工作中，Bahdanau<sup>[2]</sup> 等人使用双向长短期记忆网络 (Bi-LSTM) 的神经注意力机制去捕获句子中的重要语义信息，并且该架构能自动搜索上下文中的相关部分。Luong<sup>[3]</sup> 等人则非常详尽地评估了具有不同注意力机制的序列到序列架构的性能。Jean<sup>[4]</sup> 等人提出了使用采样分类器的方法来解决巨量词汇引发的解码复杂度过高的问题。这些工作显著地提升了 Seq2seq 模型的生成性能，并且加快了编/解码过程，为此后基于此架构的研究工作奠定了十分坚实的基础。

在近几年的研究中，科研人员还致力于研究如何将 Seq2seq 模型应用于对话系统，并且提出了不同的变体来解决不同特定领域的特定问题，例如主题感知模



型、最大信息交互模型、层级循环模型、增强型定向搜索模型等等，来产生更多元且更有信息含量的回复。并且随着研究的深入，逐渐将对话模型分为闲聊机器人与特定领域服务机器人。

在回复内容多样性方面，彭<sup>[5]</sup>等人提出了 VAE-ECG 模型，其主要架构应用了变分自编码，该架构利用了变分自编码的优点来更好更准确的对语句的非显式语义进行模型搭建，使得生成的回复内容具有更好的拓展性。刘<sup>[6]</sup>等人则基于对话风格迁移的对话神经网络，提出了融合检索结果机制的生成对话模型，来避免重复单一的通用性回复。而诸多研究表面情感因素对于成功建立起拟人对话生成模型具有重要意义，Ghosh<sup>[7]</sup>的团队提出并构建了一个情感语言模型，该模型可以按照指定的情感类型和情感强度来生成对应的结构化语言。Zhou 等人<sup>[1]</sup>所提出的基于条件和强化学习的自动编码框架，同样也是在给定的情感下生成最为合适的回复，刘<sup>[8]</sup>等人提出了基于 PAD 的对话生成框架，PAD 简单直接的将情感纬度从三个维度划分，分别为激活、愉悦与优势。但是上述的研究一直致力于提升生成回复的流利度和情感丰富度，但都需要人来手动介入去选择注入最佳的情感类别，导致了对话生成过程的性能和效率大大降低。所以本课题的主要研究方向则是在 Zhou<sup>[1]</sup>等人的经典 ECM 框架下进行升级，加入自动情感选择模块，使情感对话系统能自动选择最佳的回复情感并生成多样性的回复。

## 1.3 研究目的和主要内容

迄今为止，对话生成模型大多采用了序列到序列模型，序列到序列 (Seq2seq) 注意力模型通常是基于深度 RNN 的编码/解码器的体系结构。但是该模型在情感对话上目前有着下面几个关键性问题：

(1) 回答缺乏多样性，在很多生成结果中单一而通用，主要是因为闲聊数据集的很多 response 都比较单一。

(2) 缺少情感的编码和解码，即模型不考虑回答和问题之间的情感关系，只关心逻辑关系。

(3) 需要手动选择最佳的情感类别，没有情感自动选择模块。

基于以上三个主要问题，本课题中在模型设计和实现过程中将会主要聚焦于以下三点：

(1) 语义嵌入 (Semantic Embedding), 基于注意力机制针对文本中的主题相关词进行建模, 并在回复中围绕主题相关词语进行相似性扩展。

(2) 情感嵌入 (Sentiment Embedding), 将情感理解成一种特殊的语义来进行编码, 将其作为等价于语义和主题的一种维度来进行嵌入, 使生成的回复具有与主题联系更密切的情感倾向。

(3) 最佳情感选择器<sup>[9]</sup> (Optimal Emotion Selector), 因为实际对话过程中不会有人实时来为回复进行情感选择, 所以自动情感选择模块显得尤为重要。但值得注意的是情感选择并不只是单纯的将 *post* 中的情感提取出来, 然后在 *response* 中复用。因为不会有人想在生气的阐述观点时, 听到对方也愤怒的回复他。

而且在之前的情感对话生成中, 回复中所嵌入的情感都是单一的情感, 但是人类对于情感的感知十分微妙, 在言语中所包含的情感也不是只有一个种类, 所以在本课题中提出了情感向量空间<sup>[10]</sup> 并且假定情感概率的分布在整个向量空间上, 从而使得生成的回复不会被单一的情绪所约束。此外, 同样也希望能为聊天机器人注入自我角色和情感偏好, 也就是独有的性格, 能让它在对话生成过程中有自己独立的个人身份和语言风格<sup>[11]</sup> (比如积极、悲观、易怒、可爱甜美等), 而不是完全取决于用户的情感输入来产生对应的情绪, 但这并不是此次课题的主要研究内容, 可作为未来展望。

## 1.4 论文结构

本课题论文的主要结构如下所示:

第一章中简明扼要地概述了本课题的相关知识背景, 国内外目前对于情感对话生成模型的研究现状, 以及原有的模型架构存在的诸多问题, 并提出了大致的优化改进方案。

第二章详细归纳与总结了对话生成系统相关的技术核心知识, 其中包含了词嵌入原理、序列到序列模型、注意力机制等算法模型的底层逻辑。

第三章介绍系统的整体结果设计, 主要由情感选择模块与回复生成模块两大部分构成, 并剖析了两大模块内部子模块的设计思路。

第四章基于前面的设计部分进行具体实现, 主要包括系统主体架构实现, 以及基于 TensorFlow 开发的的情感选择器和回复生成器。

第五章中描述了测试环境与基本方案，并横向对比了相似模型架构的测试结果，其中评估的主要内容包括语义流畅度与情感准确度，以及基于 Vue.js 与 Django 的交互展示界面的实现。

第六章是对整个研究的个人工作的总结与未来前景的展望，总结了本系统相较于前人研究工作的优势与研究过程中的心得体会，并对未来能够升级迭代的模块进行了充分的展望。

## 2 技术背景概述

### 2.1 词嵌入

在进入神经网络训练之前，需要先进行预训练，其中主要包括 word embedding，也就是词嵌入，词嵌入的作用是把一个高维空间，维度数量为语料库数据集中的词汇全部数量，并嵌入到一个向量空间中，因为神经网络是无法识别直接的语言文字的，所以只能将其转换为向量空间来进行训练。这个向量空间维数低得多并且是连续的而非离散的，并且全部的 word 或者 phrase 都会被一一对应映射到实数域上，展开为向量。

而最简单的映射编码方式是 one-hot encoding，譬如如果一个字典和字库里一共有  $N$  个单词，比如 boy, girl, man, woman 这四个单词，它们的 one-hot 编码应该是：

$$boy = [1, 0, 0, 0]$$

$$girl = [0, 1, 0, 0]$$

$$man = [0, 0, 1, 0]$$

$$woman = [0, 0, 0, 1]$$

但是显然这种编码方式无法判断出相似词语的关系，因为在这个向量空间中，任何两个词语的距离都是一样的，而理想的情况应该是  $distance(boy, man) = distance(girl, woman)$ ，所以需要用到其他词嵌入算法，其中最为经典的应该是 Word2Vec，以下为本课题中所用的词嵌入技术详解。

Word2Vec 是一种非常典型的无监督学习，可以从海量的语料库数据集中学习到单词多维特征。而它的范畴中目前主要包括两种类型：Skip-Gram<sup>[12]</sup> 和 CBOW，从本质区别上来说，Skip-gram 算法预测上下文是通过给定的中间部分的词语。相反的是，CBOW 预测中间的 word，是通过前后出现的词语的。如图2-1所示

而本课题中用到的词嵌入算法就是 Skip-gram，Skip-gram 模型是输入一个单词，例如  $x_t$  输出是  $\{x_{t-1}, x_{t-2}, x_{t+1}, x_{t+2}\}$ ，上下文窗口大小为 4。举例来说的话，

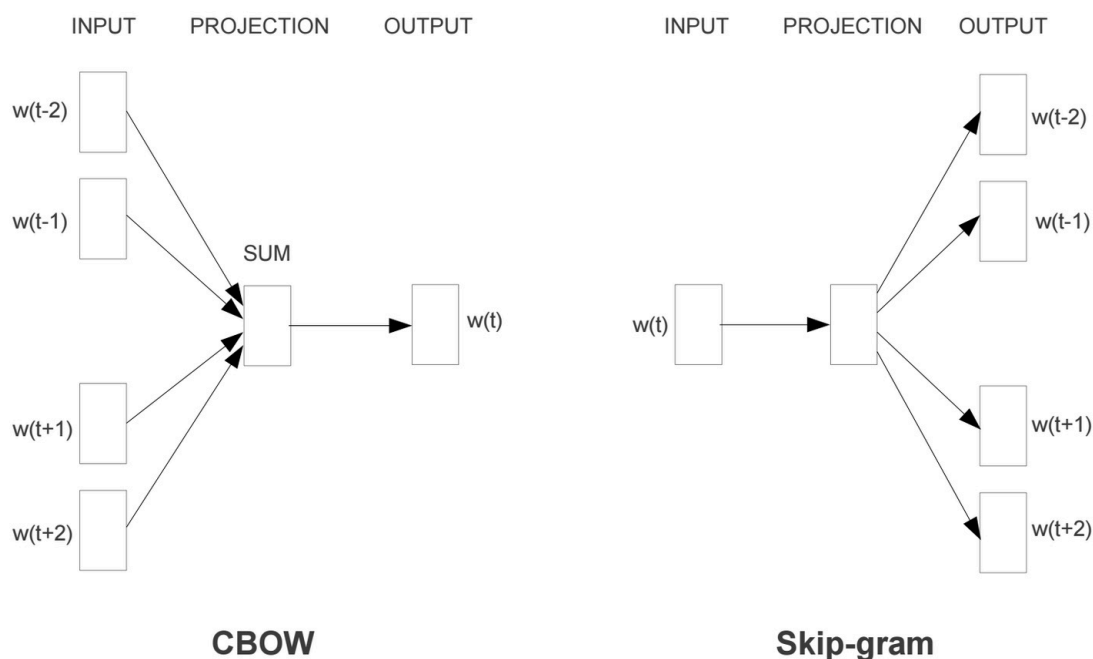


图 2-1 CBOW vs Skip-gram 对比原理图

比如数据集中有个 post 为 “China is a good country”。如果将 “a” 作为训练的时候输入神经网络中的数据，则输出为单词组 {”China”, ”is”, ”good”, ”country”}。

而如何得到词向量，仍然以 “China is a good country” 为例。

第一步输入向量，其为 one-hot 编码的向量，如果整个语料库数据集中单词总数为 10000，则集合中每个单词维度为词总数乘上一维，所以可以使用一个 one-hot 矩阵来表示”a”，然后隐藏层定 300 个神经元即可。如果规定为 100 个神经元，那么最后生成的权重矩阵的一个维度则为 100。

然后在 Input 和 Hide 层之间，会存在一个维度为  $(1 * 10^4) * (3 * 10^2)$  权重矩阵，也可以称之为全部单词的词向量矩阵  $W$ 。每个 Input 层的 one-hot 单词会在通过  $W$  之后变成一个大小为  $1*300$  的矩阵，这也是该单词的词向量  $V_c$ 。这个词向量会和  $W^T$ ，也就是全部单词向量矩阵的转置矩阵进行相乘，之后会得到大小为  $1 * (10^4)$  的矩阵，这个结果直观上理解就是输入单词和所有单词的相似度，也可以理解为此单词出现时其他单词出现概率的合集。此时 Output 层输出的加权总数为 10000 个，然后通过 softmax 层将这 10000 个输出加权转换为等量的概率，显然这 10000 个输出概率相加起来的和等于 1，计算过程如图2-2所示。

上述为 skip-gram 中全部的网络层示例。以”a”为例，如果将窗口大小 window-size 设置为 2，即选用输入词语的前面两个单词和后面两个单词，则一共会组四

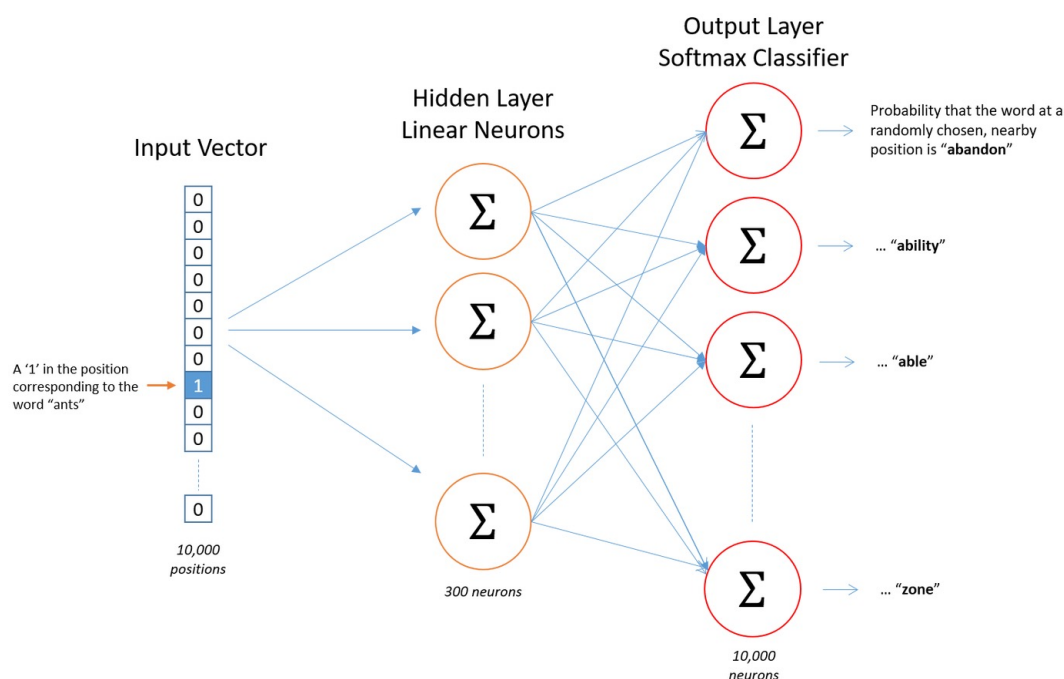


图 2-2 Skip-gram 原理图

组训练数据，分别是 (a, China)、(a, is)、(a, good)、(a, country)。

下面进行权重优化，拿 (a, country) 来说，”a” 被展开为大小为  $1 \times (10^4)$  的 one-hot 向量矩阵，而经过 Input 层的预处理之后变为 300 维的词向量  $V_a$ ，经过隐藏层与 softmax 层之后输出为 10000 个概率分布，并且在后续反向传播中通过修正隐藏层的权重分布使得最后 Output 时  $V_a * V_{country}^T$  的大小提升。代价函数 (loss function) 可以在训练过程中不断缩小两个矩阵的交叉熵，同样也可以实时计算出两个矩阵的损失值。通俗来说就是使得这两个单词在向量空间的距离尽可能的近。

再通过不断重复的反向传播和迭代后，输入层和隐藏层在训练结束后会形成一个权重矩阵，矩阵的大小为  $10000 \times 300$ ，而最后求得的权重矩阵将会通过 softmax 层映射到输入层中的单词向量，大小同样也为  $10000 \times 300$ 。

而业界也有许多开源的预训练词向量项目，例如<sup>[13]</sup>，也是本课题中所使用到词向量集，因为本课题主要是设计一款闲聊机器人，所以使用到的为微博用户的发言数据，维度大小为 300。



## 2.2 循环神经网络

循环神经网络，简称 RNN<sup>[14]</sup>，其基本思想是通过序列信息进行学习。在传统的神经网络里，Input 和 Output 层是没有直接联系的，然而如果需要预测句子中的一个单词是什么，则需要知道之前的 word 都有哪些。

如图2-3为 RNN 单元展开图，之所以被称为循环网络，是因为它们对于句子中的每个元素会执行同样的任务，且输出会依赖于上一步的计算。从另外一个层面来理解的话，RNN 可以使用记忆来获得之前所有计算过的信息。在按照时间步的序列展开 RNN 之后，可以在展开之后得到全部的序列网络，如果句中单词的总个数为 10，神经网络就会被展开为 10 层。而 RNN 中每个 Cell 中的计算过程如下：

- $x_t$ : 时刻  $t$  的输入，例如  $x_2$  则句子中的三号词的 one-hot 编码。
- $s_t$ : 时刻  $t$  的隐藏状态，也可以称为  $h_t$ ，是网络的 memory 结构， $s_t$  的计算表达式为： $s_t = f(Ux_t + Ws_{t-1})$
- $o_t$ : 时刻  $t$  的输出，也是所有的单词在下一个时刻出现概率的向量合集矩阵，表达式为  $o_t = softmax(Vs_t)$

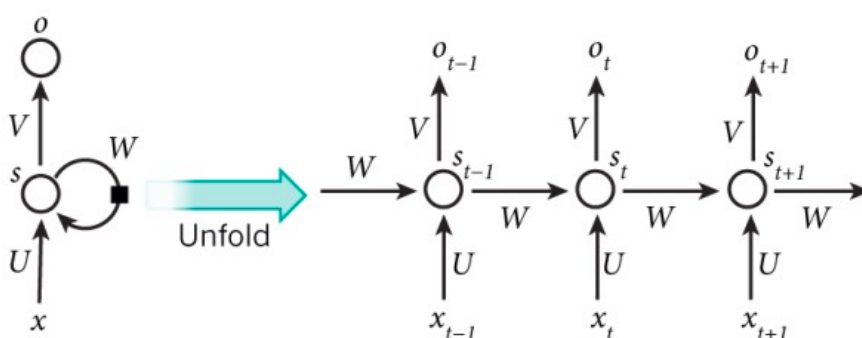


图 2-3 RNN 展开原理图

此外需要注意的是：

- 可以将隐藏状态  $s_t$  视为网络的存储器，获得先前时间中发生的信息，并且仅使用时间  $t$  处的存储器来计算输出。如前所述，这种情况稍微复杂一些，因为实际上只能获得一定时期的信息。
- 传统的神经网络在每个层中使用不同的参数，而 RNN 在所有步骤中都使用

公共参数  $(U, V, W)$ ，这意味着我们在每个步骤中执行相同的任务，但只有输入是不同的。这将减少需要学习的参数数量

- RNN 最重要的就是隐藏层，它可以间接的用于获取句子的信息。图2-3是不是绝对必要的，观察可以发现图中每个步骤都在输出，这需要针对不同的任务进行设计。例如，如果需要得到一个句子情感的预测结果，仅仅需要关注最终的输出而不需要关心每个单词的情感值。同样也不必每次都输入信息。

而 RNN 也有许多不同的种类来应对不同的需求。

- one-to-one: 就是最普通的神经网络，几乎不能算是 RNN。
- one-to-many: 常用于 music generation、image captioning。
- many-to-one: 常用于 sentiment classification。
- many-to-many (输入输出等长): 常用于 video classification on frame level。
- many-to-many (输入输出不等长): 常用于 machine translation，前一部分的作用是 encoder，后一部分作用是 decoder。

## 2.3 序列到序列模型

序列到序列 (Sequence to Sequence) 模型由两部分组成: Encoder 和 Decoder, 每一个部分都是由一个 RNN Cell 结构, 在本课题中选择了 GRU 来作为 RNN Cell 子结构。传统的序列到序列模型, 是通过 Encoder 把一个 Sequence 编码成为定长上下文向量  $c$ , 然后 Decoder 再将其解码为  $y$ , 也就是输出的序列。而该模型, 便是上一小节中提到的 many-to-many 可变长度结构的 RNN 模型, 输入和输出序列是不等长的。这类框架由 Cho<sup>[15]</sup> 等人提出, 同时 Cho 的团队也提出了 GRU 的原型。如图2-4所示, 在 Encoder 中, 主要关注的是隐藏状态  $h_t$  的生成,  $h_t$  的生成过程如下公式所示:

$$h_t = \tanh(W[h_{t-1}, x_t] + b) \quad (2.1)$$

最终 Encoder 输出上下文语义向量为:

$$c = \tanh(Uh_T) \quad (2.2)$$



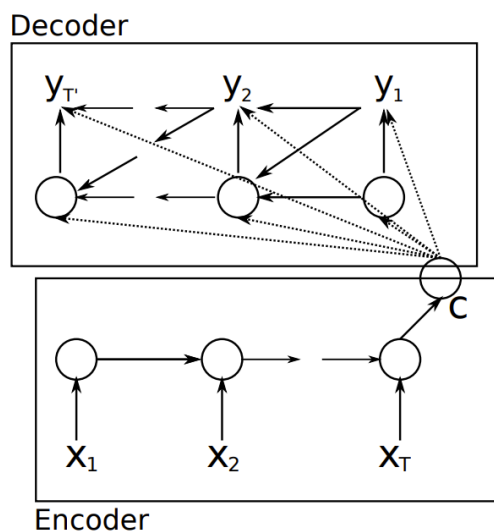


图 2-4 序列到序列模型原理图

在 Decoder 中对  $h_t$  和  $o_t$  进行迭代计算：

$$h_t = \tanh(W[h_{t-1}, y_{t-1}, c] + b) \quad (2.3)$$

$$o_t = \text{softmax}(Vh_t + c) \quad (2.4)$$

而 Decoder 接收到 Encoder 传入的语义向量  $c$  之后，首先会输入  $\langle START \rangle$  信号，和初始化  $h_0$  隐藏向量，然后按照公式2.3和2.4来进行循环计算。

$$h_1 = \tanh(W[h_0, y_0, c] + b)$$

$$o_1 = \text{softmax}(Vh_1 + c)$$

$$h_2 = \tanh(W[h_1, y_1, c] + b)$$

$$o_2 = \text{softmax}(Vh_2 + c)$$

...

$$h_t = \tanh(W[h_{t-1}, y_{t-1}, c] + b)$$

$$o_t = \text{softmax}(Vh_t + c)$$

其中  $o_t$  是一个向量，是每个时间点的输出，向量的维度为单词表的长度。向量中每个值是每个单词的概率。而这个过程将一直持续到预测值  $\langle END \rangle$  的概率为最大时，预测就会结束。预测下一个单词出现概率的公式为：

$$P(y_t | y_{t-1}, y_{t-2}, y_{t-3}, \dots, y_1, c) = g(h_t, y_{t-1}, c) \quad (2.5)$$

从公式2.5来说，其实不难发现所谓的对话生成模型，其实从根源上来说还是计算句子出现的概率，即当 A 句子出现时，B 句子出现的概率为多大。而最终生成出来的句子其实是出现概率最大的句子。

## 2.4 注意力机制

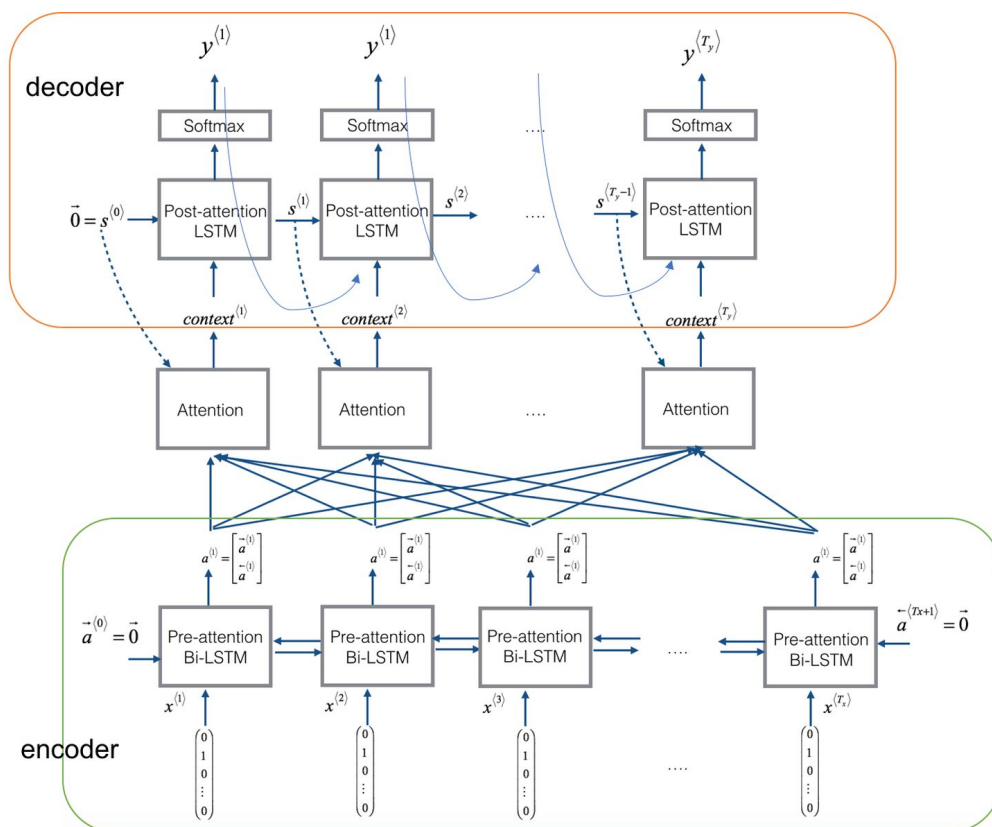


图 2-5 Seq2seq with Attention 原理图

深入研究和剖析了序列到序列模型的机制和方法后会发现，在许多场景下传统 Seq2seq 方法会有很多局限性。例如翻译一些很长的句子时，人工翻译时不会看完真个句子之后再进行翻译，而是先翻译一部分，然后再翻译进行下一部分，一直往复循环到最后。因为记忆整个句子是不现实的，所以人工在翻译某个单词时，实际上只是把原来句子的局部范围最为上下文。所以 attention 机制就是在针对每个所生成的  $y_t$  时，都会给出特定的 context 上下文信息  $c_t$ ，这是非常符合人的直观感受的，这样原文中每个不同的部分对于翻译当前的这个单词的影响权重是不一样的，这样效果就会大幅度提升，尤其是在解决翻译长难句的问题时。

如图2-5所示, 和传统 Seq2seq 模型不同的是, Seq2seq with Attention 实际上是在 decoder 和 encoder 之间加入了一个 attention 层。用于计算每一步 decode 时所需要的上下文向量  $context_t$

decoder 部分针对每种  $y_t$  进行计算的时候, 包括三个部分的输入:

1.  $context_t$ : 上下文信息。
2.  $h_{t-1}$ : 也可以写作  $s_{t-1}$ , 为隐藏层状态。
3.  $y_{t-1}$ : 上个时刻中 decoder 的输出。

在 attention 模块部分, 使用  $A_{<t,t^*>}$  来表示在翻译  $y_t$  的时候应该花在  $a_{t^*}$  上的注意力的数量, 所以计算  $context_t$ :

$$context_t = \sum_{t^*=1}^{T_x} A_{<t,t^*>} a_{t^*} \quad (2.6)$$

在 attention 进行计算时, 输入包括两部分:

1. decoder 模块中上一个时刻的隐藏状态  $s_{t-1}$ 。
2. encoder 模块所有时刻的输出  $a_{t^*}$ 。

如图2-5所示, attention 模块其实本质上也是一个神经网络, 然后使用 softmax 对输出进行归一化处理。

$$\sum_{t^*=1}^{T_x} A_{<t,t^*>} = 1 \quad (2.7)$$

## 2.5 本章小结

本个章节主要围绕本课题在设计实现过程中所需要用到的背景知识和技术来进行展开介绍, 主要包括词嵌入、Seq2seq model 以及模块 Attention mechanism, 但是和机器翻译不同的地方在于, 本课题需要将情感纳入考量范围, 所以情感也作为一种特殊的语义参与输入和输出。首先本章节单独详细讲解了一下词嵌入的机制, 因为词嵌入不仅是对语义的一种优化, 也可以将其抽象思想应用于情感因素, 而具象出情感嵌入机制。接着是 RNN 模型的底层逻辑, RNN 的诞生很好的解决了序列化信息处理难题, 其中最有名的应用则是 Seq2seq 模型, 而 Seq2seq 模型也被称之为条件语言模型, 因为在生成过程中, 与其说是生成了一

个句子，不如说是计算这个句子出现的概率是否是最大的。而 attention 机制则是优化概率计算过程中权重分配的一个类神经网络层。而纵观本小节整个发展思路，其实就是在不断地模拟真人的思维方式，尤其是 attention 机制，不仅可以让机器关注于翻译时的上下文，更可以聚焦于那些对情感有明显作用的词句，这也是本课题的中心思想。

### 3 基于注意力机制（Attention）的情感对话生成系统设计

本章节将围绕整个原型系统的设计来展开，在 Zhou<sup>[1]</sup> 等人的课题基础上，加入了情感选择模块，并对基于内外部记忆的回复生成模块进行了优化改进，不再依赖于手动操作。

本课题主要围绕情感对话生成来开展工作，所以其主要任务是能主动获取 post 中的情感和语义，并预测出 response 中所使用的最佳情感并生成对应的回复。

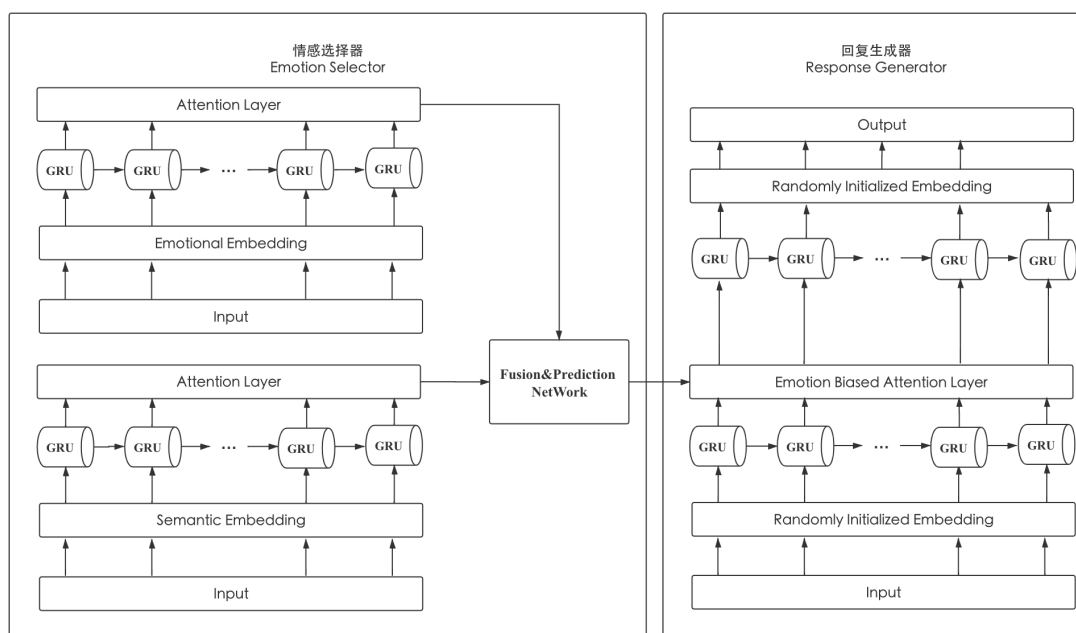


图 3-1 Emotional Chat Machine Pro 原理图

综合需求可将系统的总体架构划分为情感选择模块和回复生成模块两个部分。系统命名为 Emotion Chat Machine Pro，其在 Zhou 等人<sup>[1]</sup> 的基础上进行优化和功能升级，系统整体设计架构如图3-1所示。

情感选择模块可以拆分为情感序列编码、语义序列编码和融合预测，基于融合预测网络情感选择模块<sup>[9]</sup>（Emotion Selector）使用情感编码模块，获取 post 中的感情倾向，而语义编码模块则用于获取 post 中的语义向量，然后在融合预测网络中进行交叉计算。而情感编码中使用情感嵌入、序列到序列（Seq2seq）模

型以及注意力 (Attention) 机制, 语义编码中同样也使用到了语义嵌入和基于注意力机制的序列到序列模型, 所以很有意思的一点是, 可以把情感也理解成为语义的一个维度。

而回复生成模块 (Response Generator) 则主要沿用 ECM<sup>[1]</sup> 模型中的生成模块, 使用 many-to-many 不定长度的 sequence to sequence 模型来生成回复, 但是会在情感注意力层将情感选择模块所生成的向量纳入考量, 其主要思想仍然是使用内部外部记忆来使得所选择的情感能在所生成的回复中得到完全的释放。

以下两个章节为两个子模块的详细设计。

## 3.1 基于融合预测网络的情感选择模块

情感选择模块总共分为以下三个部分:

1. 语义编码模块
2. 情感编码模块
3. 融合预测网络

下面三个小节将分别描述这三个部分的功能和作用, 以及训练时使用到的损失函数。其中融合预测网络会将语义编码模块和情感编码模块中所产生的语义和情感隐藏向量中进行权重重组融合, 并得到最佳情感。

### 3.1.1 语义编码

针对于语义编码问题, 输入为微博中用户对话的汉语文本, 但模型只能接受数值型的数据输入, 所以首先需要将输入句子进行语义嵌入。这里可以参考<sup>[16]</sup>以及<sup>[13]</sup>中词嵌入的详细论述, 实现词嵌入目前的主流技术为 Word2Vec, 而本课题中采用了 Skip-gram 算法来从 *post* 中挖掘有效语义信息, 并以向量的形式输入到 GRU 单元中进行计算。从历史经验来看, 使用 300 维来表示一个词语是最优解, 且业界也已经有了很多比较成熟的实现方案和预训练好的结果。如果想用 300 个特征来定义一个单词 (意思是每个单词可以被展开为一个 300 维的向量)。那么一个大小为 100 万的词集最后在 Word2Vec 后的结果应该是一个  $1000000 \times 300$  的矩阵。而最终的目标就是获得这个矩阵。从直觉上说, 如果有两

个不同的单词有着非常相近的上下文，例如：

- 我爱中国
- 我爱美国

那么通过模型的训练，这两个单词（中国，美国）在词嵌入之后的向量应该非常相似。Word2Vec 的创始人提出了几个非常重要的优化方案，例如：

- 将非常常用的 word 当成单个单词进行处理。
- 随机抽样出现次数非常高频的单词，这样的操作可以减少样本训练个数。
- 对于所需要优化的目标，使用 negative sampling 来降低算力负担和提高训练词向量的质量。

在处理好输入之后，语义编码模块的神经网络层由基于注意力机制的 Seq2seq 模型所构成，而在本课题中，所采用的子单元（RNN Cell）由 GRU 构成而并没有使用 LSTM。

LSTM<sup>[17]</sup>（long short term memory）能从输入的词句数据集中习得长期的依赖关系。LSTM 主要由四层构成，第一层是遗忘层，它决定了 cell 中遗忘了哪些信息。第二层是记忆层，它决定了 cell 需要记忆哪些新的信息。第三层是更新层，它将 cell 遗忘和记忆的内容拼接起来来更新 cell 的状态。第四层是输出层，决定了最后该输出什么，输出的值跟 cell 状态有关。此外 LSTM 网络还有一个巨大的优势是攻克了梯度爆炸和消失的难题。

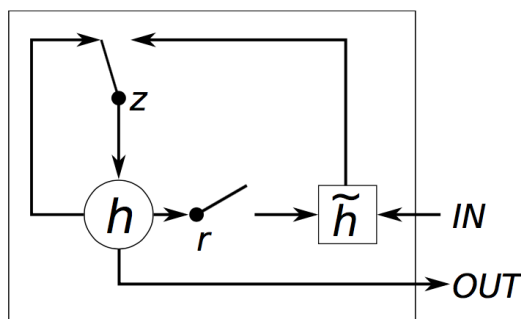


图 3-2 GRU 原理图

GRU<sup>[18]</sup>（Gated Recurrent Unit）结构上比 LSTM 简单，更重要的是它所消

耗的计算资源要小的多。因为 GRU 在实现过程中将记忆和遗忘合并到了同一个步骤中，GRU 也是 LSTM 目前接受度和承认度最高的变体。LSTM 一共有三个门，分别是 Input 门、Forget 和 Output 门。GRU 如图3-2所示，一共有 Reset 门和 Update 门两个门，不难发现 GRU 并不会保留控制单元内部所产生的记忆并且摒弃了 LSTM 中的 Output 门。

梯度消失是 RNN 中一个非常棘手的问题，相较于 LSTM 的多个门来说，Reset 门和 Update 门是 GRU 简化思想的经典之处。这两个门可以决定哪些信息是可以作为最终的输出，它们可以保存跨度非常长的序列中的有效信息，并且这种信息不会随着时间的推移而清除或者被遗忘掉。所以在本课题中采用了 GRU 而非 LSTM 来作为最小的 RNN 单元。

通过 GRU 网络，可以从  $post$  序列  $x = (x_1, x_2, \dots, x_T)$  中抽取关键信息，并且将其映射到隐藏层，计算公式为：

$$h_s^t = GRU(h_s^{t-1}, x_t) \quad (3.1)$$

其中  $h_s^t$  表示在时间  $t$  时隐藏层的状态。

但是为了增强隐藏层的信息传递有效性，可以使用注意力机制来使编码器更关注于和主题密切相关的词句。并且在最后获得最终的语义向量  $\tilde{h}_s$ ，其计算公式如下：

$$\tilde{h}_s = \sum_{i=1}^T a_i h_s^i \quad (3.2)$$

如公式3.2中所述， $a_i$  是隐藏层状态向量  $h_s^i$  的权重，因为如注意力机制所描述的算法来说， $h_e^t$  不再仅仅是依赖于  $h_e^{t-1}$ ，而是会依赖于每个 GRU 节点的隐藏层状态。而  $a_i$  则是训练过程中需要不断地去学习和重新分配的权重。

而  $a_i$  主要依赖于 encoder 的隐藏层状态  $h_s^i$ ，将  $h_s^i$  输入一个多层感知机，然后再通过一个 softmax 层来得到  $a_i$  权重大小，并且可以确保的是所有的  $a_i$  相加起来的总和为 1。 $a_i$  的计算公式如下：

$$a_i = softmax(V_a \tanh(W_a (h_s^i)^T)) \quad (3.3)$$

将输出全部映射到从 0 到 1 的区间内，softmax 从本质上可以视为一种概率分布映射算法，从概率向量空间中的分布来多分类。它是常用于多分类的一个算法，



如果有一堆需要多分类的数，那么第  $i$  个元素的 softmax 的值就是：

$$S_i = \frac{e_i}{\sum_j e_j} \quad (3.4)$$

而  $\tanh$  则是一个激活函数，激活函数在训练过程中是非常有必要的。因为如果使用线性的激活函数，那神经网络则失去了意义，仅仅是将输入经过线性组合变换之后再输出，在这类情况下，多层隐藏层的神经网络和单个隐藏层的神经网络没有任何的区别，常用的激活函数一般有 sigmoid、 $\tanh$  和  $\text{relu}$ 。在这里使用  $\tanh$  作为激活函数的原因是其值域为  $(-1, 1)$ ，而 sigmoid 为  $(0, 1)$ ， $\tanh$  的公式如下：

$$a = g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \quad (3.5)$$

但是 sigmoid 和  $\tanh$  作为激活函数都有一些共同的缺点，比如  $z$  在很大或者很小的时候，梯度基本为 0。所以在优化梯度下降算法来更新神经网络的时候使用  $\tanh$  性能比较差，但是在本课题中影响不大。因为  $z$  的分布一般位于中间值域。

综上在语义嵌入以及 GRU 网络编码之后，通过上述一系列转换，最终可以通过公式 3.2 来获得最后的  $\tilde{h}_s$  来作为语义编码向量。

### 3.1.2 情感编码

和语义编码类似的是，情感从抽象角度也是作为一种存在于语句中的信息，和主题一样通过可以被编码为向量。同样输入为汉语文本，可以参考 SSWE<sup>[19]</sup> (sentiment specific word embedding) 中的对推特的评论回复进行情感编码的实现过程来实现。而本课题中可以模拟 SSWE<sup>[19]</sup> 的算法来对中国微博中的用户评论回复来进行训练和编码。

情感作为一种信息，可以大致分类为 (*like, disgust, sad, angry, happy, none*) 6 种情感。本课题中没有单一的采用一种情感来作为分类指标，而是将其将感情视作一种多维数据，例如一句话的情感可以同时具有 *like* 和 *happy*，或者 *sad* 和 *angry*，如果的确只有单一的情感 *like*，则为 *like* 和 *none* 的结合。而情感编码的本质其实就是计算不同词语对不同情感的影响程度。如下面的例子，如果整个数据集只有这两句话：

- 我爱中国
- 她爱美国

如果这两句话都被标记为 like，那么在机器学习过程中，将判定“爱”对于 like 的影响程度是高于 [“她”，“吃”，“雪糕”] 以及 [“我”，“中国”] 的权重。

同样在处理好输入 sentiment embedding 的相关问题后，将其输入 GRU 网络中进行训练。和语义编码不太类似的地方在于，情感编码在 decode 阶段是需要分类的，有具体而准确的结果，所在本小节最后会根据此来设计损失函数。

同样的，情感编码也是通过 GRU 网络，从 post 序列  $x = (x_1, x_2, \dots, x_T)$  中抽取情感相关信息，然后映射到隐藏层，计算式如下：

$$h_e^t = GRU(h_e^{t-1}, x_t) \quad (3.6)$$

其中  $h_e^t$  表示，隐藏层在时间 t 时候的状态。如图3-3所示， $h_e^t$  会在经过 MLP 和 Softmax 层之后得到一个权重  $a_t$ ，其中采用到了多层感知机（MLP），也可以称作人工神经网络，多层感知机最大的特点就是层与层之间都是全连接的，而传统的 MLP 可以分为输入层，隐藏层，输出层三层。当然，MLP 的隐藏层可以分为很多层，这里视需求来进行选择。

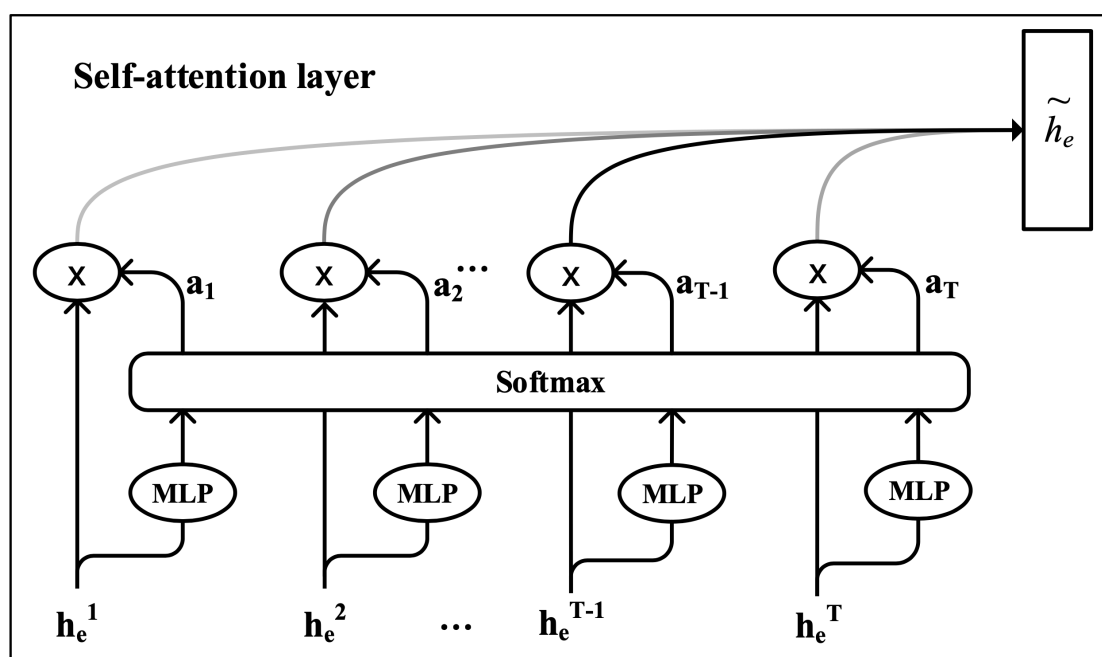


图 3-3 attention layer 原理图

为了增强隐藏层对于情感信息的感知能力，如图3-3，同样使用注意力机制来聚焦于情感词句，并在最后生成情感编码向量  $\tilde{h}_e$ ，可知计算公式如下：

$$\tilde{h}_e = \sum_{i=1}^T a_i h_e^i \quad (3.7)$$

整个网络为了使 RNN 网络其更侧重于情感相关信息，设计了一个交叉熵损失函数。之所以这样做是因为，可以通过获取的  $\tilde{h}_e$  来进行计算，通过一个线性层和 sigmoid 层来将其映射情感空间中，来得到最后编码出来情感解码出来的结果的损失有多大。如下公式：

$$\hat{e}_p = \sigma(W_e \tilde{h}_e + b) \quad (3.8)$$

$$\mathcal{L}_p = -e_p \log(\hat{e}_p) \quad (3.9)$$

通过损失函数的结果反向传播，然后经过上述转换后，最后可通过公式3.7来得到最后的  $\tilde{h}_e$  来作为情感编码向量。

### 3.1.3 融合预测网络

在经过情感编码和语义编码之后，需要将得到的两个向量进行融合并进行对 response 情感的预测，这也是本课题的重中之重。为了构建  $e_r$  的分类结果，这里使用融合预测网络来融合均衡两个不同编码向量对于情感的贡献。并在之后部署一个预测网络来选择 response 的最佳情感分类。如图3-4 所示，在将  $\tilde{h}_s$  和  $\tilde{h}_e$  输入一个 sigmoid 网络后可以获得一个权重大小，计算过程如下所示：

通过式3.10来获取权重分布系数  $w$ ，其实这个系数和 GRU 中用于记忆和遗忘的系数  $z$  有异曲同工之妙。

$$w = \sigma([\tilde{h}_s; \tilde{h}_e]) \quad (3.10)$$

$$\tilde{h}'_e = \tanh(\tilde{h}_e) \quad (3.11)$$

$$\tilde{h}'_s = \tanh(\tilde{h}_s) \quad (3.12)$$

之所以选用  $\tanh$  作为激活函数的原因也很简单，衡量一个好的激活函数的方法是看其是否会在迭代训练中引发梯度消失或爆炸的问题，反向传播时会对其求导，并与每一个 layer 进行相乘来计算误差。然而如果激活函数的导数比 1

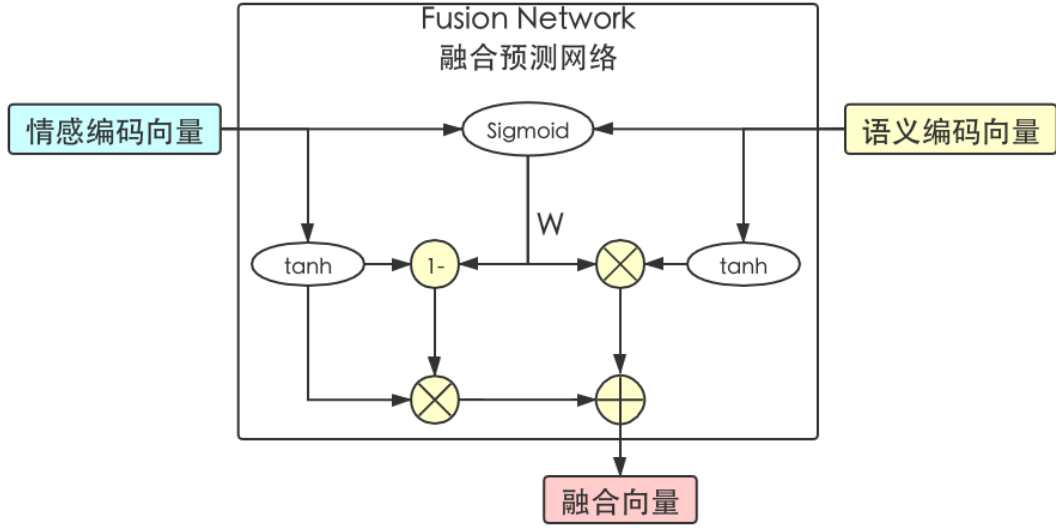


图 3-4 Fusion Network 原理图

要小，注意是绝对值，在非常多的迭代相乘之后，误差的大小会无限接近于 0。在 Activation Function 的导数的大于 1 时，同样也是绝对值，那么在非常多次的迭代相乘之后最后的结果会变得非常巨大，也就是与梯度消失相对应的难题——梯度爆炸。

在得到这个最后被转换过的  $\tilde{h}_s$  和  $\tilde{h}_e$  之后，再通过如下公式进行融合求得最终权重：

$$\tilde{h}_{es} = w \otimes \tilde{h}_s' + (1 - w) \otimes \tilde{h}_e' \quad (3.13)$$

而拿到了  $\tilde{h}_{es}$  之后，需要将其输入预测网络来产生一个情感向量，也就是  $\hat{e}_r$ 。这里同样需要用到多层感知器来进行一个映射。

$$\hat{e}_r = \sigma(W_r \tilde{h}_{es} + b) \quad (3.14)$$

通过上述公式可以直观的得到一个情感分类的概率分布。并且依据此来设计一个和公式3.9类似的损失函数。

$$\mathcal{L}_r = -e_r \log(\hat{e}_r) \quad (3.15)$$

这里的  $e_r$  是代表的每种情感是否会被选到，取值为 0 或者 1，而  $\hat{e}_r$  则是选取到该情感的概率。最后情感选择器会将概率最高的 emotion 的类别输入到回复生成模块用于生成拥有最佳情感的回复。

### 3.2 基于内部外部记忆的回复生成模块

回复生成模块在 ECM<sup>[1]</sup> 中已经有了比较详细的设计，而在本课题中，为了能够融合语义和情感在生成的回复中，在 ECM 的基础进行改进和升级，以便能够接收在情感选择模块中所产生的  $e_r$ ，在其生成过程的主要思想如图3-5依旧沿用了 ECM 中的内部外部记忆。而在本项目中继续沿用 ECM 中内外记忆的设计

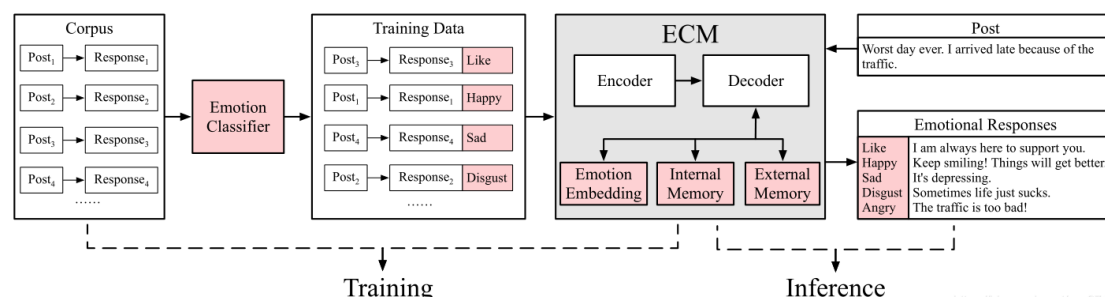


图 3-5 ECM 原理图

理念，因为内外情感的统一可以最大化的不损失语义的连贯性。

首先如 Affect-Im<sup>[7]</sup> 文中所述情感嵌入在生成的过程中是不会被改变的，但是这会极大程度的损失语义的连续性。从文中得到的启示是情感是快速波动且状态短暂的，在生成情感回复时的情感变化过程也有着相同的特征，所以 ECM 动态捕捉每一步解码过程中的情感，并构建了一个基于内部记忆的模块来实现这个功能。

举个简单的例子来说明在表达情感时的整个过程，decode 开始之前会存在一个情感状态存在于内在，并且在生成过程中的每一个 step 都会衰减，所以在 decode 过程完成的时候情感状态已经为 0，这也说明了该情感状态在生成回复时被非常充分的表达了出来。

ECM 在最初的 Seq2seq 模型中加入了 emotion embedding，这是一种静态方法，动态方法包括内部记忆情感状态网络以及外部记忆情感词语机制。这种改进能让 ECM 更准确的将输入的情感类别转换为情感信息和情感状态并改写无情感的回复。

而下面将会详细的介绍基于内部记忆的情感状态网络模块的设计思路，以及基于外部记忆的情感词语选择器的基础理论。

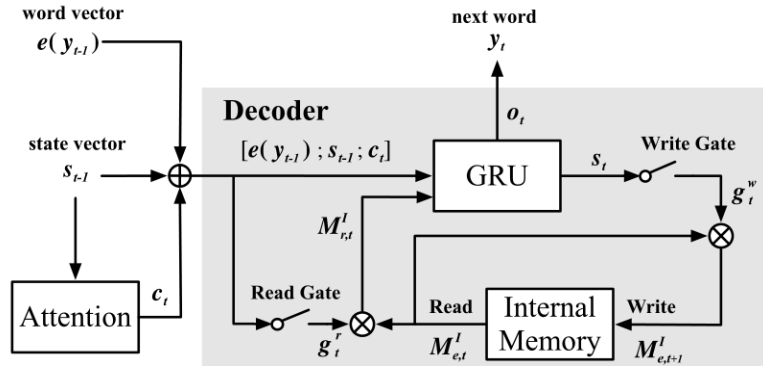


图 3-6 内部记忆模块原理图

如图3-6所示，在内部记忆的解码模块中，内部情感状态变化和选词的关联不是显式的，所以并不能被直接的观察得到。在每个时间步的计算过程中，都会计算到 Read Gate  $g_t^r$  和 Write Gate  $g_t^w$ ，如下述公式所示：

$$g_t^r = \sigma(W_g^r[e(y_{t-1}); s_{t-1}; c_t]) \quad g_t^w = \sigma(W_g^w[e(y_{t-1}); s_{t-1}; c_t]) \quad (3.16)$$

$g_t^r$  和  $g_t^w$  都是用于对内部记忆模块来读写的，在每个时间步中， $g_t^w$  都会被消除一部分，而在最后一个时间步中情感状态会衰减至 0，此时情感被完全表达。

$$M_{r,t}^I = g_t^r \otimes M_{e,t}^I \quad M_{r,t+1}^I = g_t^w \otimes M_{e,t}^I \quad (3.17)$$

而隐藏层的状态也会根据内部情感状态来进行改变

$$s_t = GRU(s_{t-1}, [c_t; e(y_{t-1}; M_{e,t}^I)]) \quad (3.18)$$

情感表达和情感词并不是一个意思，比如可爱和令人仰慕，与一般的非情感词语相比，例如“中国”和“美国”，情感词承载了丰富的情感，ECM 通过为分别分配不同的生成权重给富含情感的词汇和普通非情感词，非隐式的构建回复生成中的情感表达的模型。基于此提出了外部记忆模块，根据这一模型，生成过程中会从富有情感的词汇或普通的通用词汇中来选择生成词。

如图3-7, 针对于不同词汇表达的不同情感，external memory 模块会为情感词汇和普通无情感词汇分别赋上生成概率，来决策下一个单词如何生成。此模块主要的思想是在每一个生成 step 中实时计算 Emotion/Generic Word 的权重，并且与 internal memory 模块强耦合。

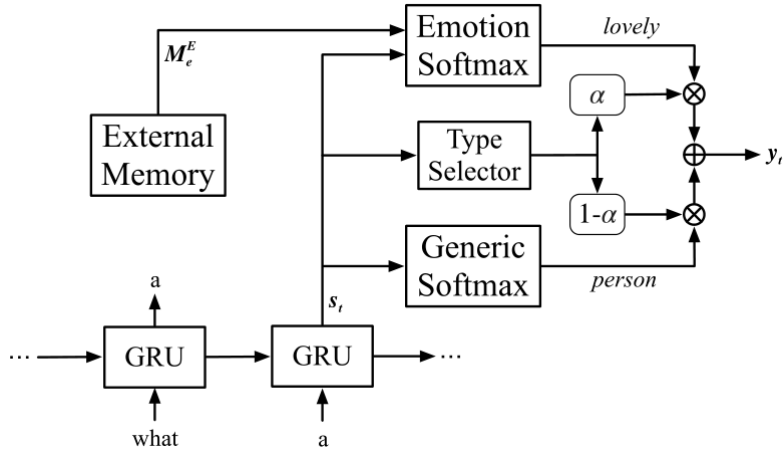


图 3-7 外部记忆模块原理图

$$a_t = \sigma(v_u^T s_t) \quad (3.19)$$

$$P_g(y_t = w_g) = softmax(W_g^0 s_t) \quad (3.20)$$

$$P_e(y_t = w_e) = softmax(W_e^0 s_t) \quad (3.21)$$

$$y_t o_t = P(y_t) = a_t P_e(y_t = w_e) + (1 - a_t) P_y(y_t = w_g) \quad (3.22)$$

以上描述的是在已知 response 的情感之后，生成对应回复的过程和方法，而现在需要设计 response generator 和 emotion selector 的衔接。在获得了 Emotion Selector 给出的最佳回复情感之后，为了生成回复的情感嵌入向量，需要将  $\hat{e}_r$  的维度放大到和隐藏层向量一样大。所以需要将  $\hat{e}_r$  乘上一个随机向量  $W_e$ 。计算方法如下：

$$V_e = W_e \hat{e}_r \quad (3.23)$$

引用 Plutchik<sup>[20]</sup> 对情感的抽象本质的描述来说的话，现实生活中所描述的情感只是一维的，例如：like，sad 等等，但是实际上人类在成长过程中，会不断地学习这些情感所对应的场景以及本课题所提到的——词句。因此在这里随机输入的一个随机向量  $V_e$  就是相当于新生儿初次来到这个世界时，对于情感的认知，它一定是不准确且需要不断地修正的，然后将其输入到 RNN 中，通过不断的反向传播来修正  $V_e$  的权重。

在本课题中整个数据集中包括大量无感情，单一情感和双重情感的 post-



response 对。在此前的 ECM 中，使用的是基于内在和外在记忆的模式，通过编码输入序列到 RNN 网络中来得到隐藏层向量  $h = (h_1, h_2, \dots, h_T)$ ，然后生成出一个上下文向量  $c_t$  用于 decode 这一步的隐藏层向量  $s_t$ 。本课题将通过注意力机制重写这一过程，主要是优化了权重的分配这一步骤。此前的计算方式如下：

$$u_i^t = v^T \tanh(W_1 h_i + W_2 s_t) \quad (3.24)$$

$$a_i^t = \text{softmax}(u_i^t) \quad (3.25)$$

$$c_t = \sum_{i=1}^T a_i^t h_i \quad (3.26)$$

在每个时间  $t$  中，上下文向量  $c_t$  都会根据注意力机制重新编码，然后使得在长句子生成中拥有更好的性能。但是以上的公式显然是忽略了刚刚所说到的  $V_e$  情感向量。现在，将其纳入训练算法中并依此重写公式3.24, 如下所示

$$u_i^t = v^T \tanh(W_1 h_i + W_2 s_t + W_3 V_e) \quad (3.27)$$

然后通过公式3.25和3.26来生成全新的  $c_t$ 。并且通过  $c_t$  和原先的解码器  $s_t$  融合产生新的隐藏层状态：

$$s'_t = W_4[s_t; c_t] \quad (3.28)$$

在获取到最新的隐藏层向量后，可以重构公式3.1为：

$$s_t = GRU(s'_{t-1}, [y_{t-1}; V_e]) \quad (3.29)$$

上述计算式表示解码器在每次更新状态时，都会将上一轮的输出、隐藏层向量以及情感向量作为输入，这样可以最大化保持语义连贯和情感的饱满。

### 3.3 损失函数

在本课题中损失函数主要由两个部分构成，一个是情感损失，一个是语义的损失。可以表示为：

$$\mathcal{L}_{ECMP}(m) = a\mathcal{L}_e + (1 - a)\mathcal{L}_{seq2seq} \quad (3.30)$$

其中  $a$  是一个用来平衡情感损失和语义损失权重的因子，也是需要调参的一个地方。



而  $\mathcal{L}_e$  则是代表情感损失，计算过程如3.31：

$$\mathcal{L}_e = \mathcal{L}_p + \mathcal{L}_r \quad (3.31)$$

其中  $\mathcal{L}_p$  表示 post 中的情感损失，而  $\mathcal{L}_r$  表示 response 中的情感损失。

## 3.4 设计中考虑的制约因素

ECMP 设计中的制约因素主要需要通过系统可用性、未来可扩展性、使用便捷性、算力资源预算四个角度来考察。

### 3.4.1 系统可用性

在整个原型系统的设计过程中，模块与模块之间的结耦合程度会极大的影响到整个系统的可用性，因为在理论上情感选择模块和回复生成模块是完全区分开来的两个模块，但是实际的实现过程中，因为都是基于序列到序列模型来进行生成的，所以会出现大量的共用的函数，但是许多定制化的需求就会受到影响，并且在设计过程中需要着重考虑的就是 debug 的成本，一个模块的异常会直接影响到另外一个模块，而且难以发觉是哪个步骤出现了问题。在设计实现完整个系统后，也要不断的考虑异常情况，以免用户的非常规输入直接导致了系统的报错。

所以在不同的系统上和不同 TensorFlow 版本的兼容问题是设计过程中需要考虑的头号需求。

### 3.4.2 模块可扩展性

设计过程中比较重要的一点是该系统的可扩展性，因为本课题是基于 ECM<sup>[1]</sup> 的项目基础上进行进一步的扩展，而为了使得后来的研究人员可以继续在本项目基础上做持续性的修改，模块之间的可扩展性也是非常重要的，因为可能后期有人需要在其基础上添加知识图谱、风格迁移等等新的功能，所以在项目编写过程中不能把模型写的过于死板，并且一定要注意把注释写得更为详细以免后来者无法快速上手。但是可扩展性高也意味着系统的高度模块化，这也会使得开发过程的效率会有所下降。

### 3.4.3 使用便捷性

因为本系统是基于 TensorFlow 编写的，所以在编写过程中需要考虑到的一点是，用户不可能直接通过命令行来与原型系统进行交互，并且原型系统在能正常运行之前，需要配置大量的环境变量和安装包，这显然是非常反人类和不利于推广的。所以在后面的章节中也会提到，本课题基于 Vue.js 和 Django 框架设计了一套完整的交互界面供用户体验，这套交互系统应该符合大众审美，且功能按钮齐全，交互界面简单明了，能让每个用户快速上手使用起来。

### 3.4.4 算力资源预算约束

因为本课题涉及到神经网络的训练，所以对 CPU 和 GPU 的配置都有一定的要求，在算力资源方面，因为最近的矿潮导致了 GPU 的全面涨价，所以在购买或者租赁算力时，需要考虑训练过程中所需要使用到的算力资源消耗，以免超过预算。

## 3.5 成本估算

### 3.5.1 基于 Putnam 模型的时间成本估算

从开始设计项目到完成项目，以及前后调研和撰写代码的总时长来看，本次课题的时间成本大约为 3 个月左右。代码设计的实际实际约为 4 周，而代码实现的实际时间大约为 4 周，其余时间主要用在学习相关理论知识和对接课题组成员，本节采用 Putnam 模型<sup>[21]</sup>来估算整个软件开发的成本。

而 Putnam 提出的动态多变量模型，计算过程如下

$$L = Ck * K^{1/3} * td^{4/3}$$

如下表3-1所示，本项目 Ck 值位于优秀队列，作为一个规模较小的深度学习项目，该项目的实现成本并不高，所以可移植性和可拓展性均较强。而项目的主要人力资源都用在了前期的相关知识学习和系统架构实现部分。

表 3-1 Putnam 系数

类型	开销大小
L (源代码行数)	1573 (行)
K (整个开发过程所花费时间)	0.75 (年)
Td (开发持续时间)	0.25 (年)
Ck (技术状态常数)	11000

## 3.5.2 算力资源与硬件消耗的资金成本

因为此次课题是有关于深度学习的项目，所以主要的开销在电脑硬件和电力消耗上。因为硬件是之前已经采购过的，所以真实的硬件开销应该只需要计

表 3-2 成本开销

类型	GPU	CPU	其他硬件
电脑硬件	1350¥	2700¥	17000¥
电力消耗	20¥	5¥	10¥

算折旧的部分，大约为硬件开销的 5% 左右。

## 3.6 本章小结

本章的主要内容是围绕情感对话系统来进行展开，所以主要分为情感选择器和回复生成器的设计。

在情感选择器部分中着重需要注意的是情感嵌入 (sentiment embedding) 以及 attention 机制的引入，这两个部分也是主要工作量的来源，从本质上说情感嵌入也是语义嵌入的一个维度，只不过在本课题中将其单独进行抽象和具象编码，而情感选择器中的语义嵌入则更类似于传统机器翻译模型中编码器的结构，除此之外，融合网络将情感编码  $\hat{h}_e$  和语义编码  $\hat{h}_s$  后的结构进行权重分配，这里的设计其实类似于 GRU Cell 的内部记忆/遗忘门的实现，并将分配之后得到的向量映射到情感空间来得到 response 的最佳情感，这里分配重组后向量的损失函数是反向传播时考虑的重要因素，也会在训练时不断更新。

对于回复生成器的设计则更多的参考了 ECM<sup>[1]</sup> 的设计，在解码阶段采用内部情感状态衰减和外部情感词语选择两个模块，内部情感状态会不断地衰减以

防对语义连贯性产生过大的影响，而外部情感词语选择则在每一步生成词的过程中，选择情感丰富的词或者非情感相关普通词。这样生成出来的 response 在语义流畅度上有着更好的表现，且完全不需要再依赖于人来手动进行选择回复的最佳情感。

## 4 基于注意力机制（Attention）的情感对话生成系统实现

在本课题的实现过程中，主要采用 python 作为编码语言，因为 python 本质上是一种解释型语言，而且面向对象，最重要的是其生态链中有着大量实用的第三方库。现如今在深度学习方面，TensorFlow 和 PyTorch 已经是最为流行的两个用于深度学习的库了。而在本课题中采用的则是 TensorFlow 来作为项目基础框架。

而因为要使用到 GPU 的原因，所以使用的训练环境如下：

- 系统环境：Windows 10
- 编码平台：PyCharm
- GPU 配置：GTX 1050Ti
- TensorFlow Version: tensorflow-gpu==0.12.\*
- Python Version: 3.5

### 4.1 系统主体架构实现

系统的主体虽然是分为情感选择和回复生成两个部分，但是从传统的机器学习项目上来说，总体结构依然是分为数据处理、训练、测试三个主要部分的。

系统在 Train 和 Test 阶段中，都会经历数据准备（prepare data）和模型建立（create model）两个步骤，而数据准备阶段的主要实现过程包括读取数据、数据序列化以及特定格式嵌入，模型部署则是对基于 attention 机制的 Seq2seq 的引用，除了数据嵌入层不同以外，其他大同小异。如图4-1所示。其中值得注意的是，为了更好的进行情感向量的嵌入，实验过程中将改写多种 GRU Cell 变体来适应新的需求，比如内部情感状态记忆功能或者外部情感词权重分配能力。而 Response Generator 的实现则沿用了 ECM 的实现过程，在本章中将会对其算法实现过程进行详细的剖析。

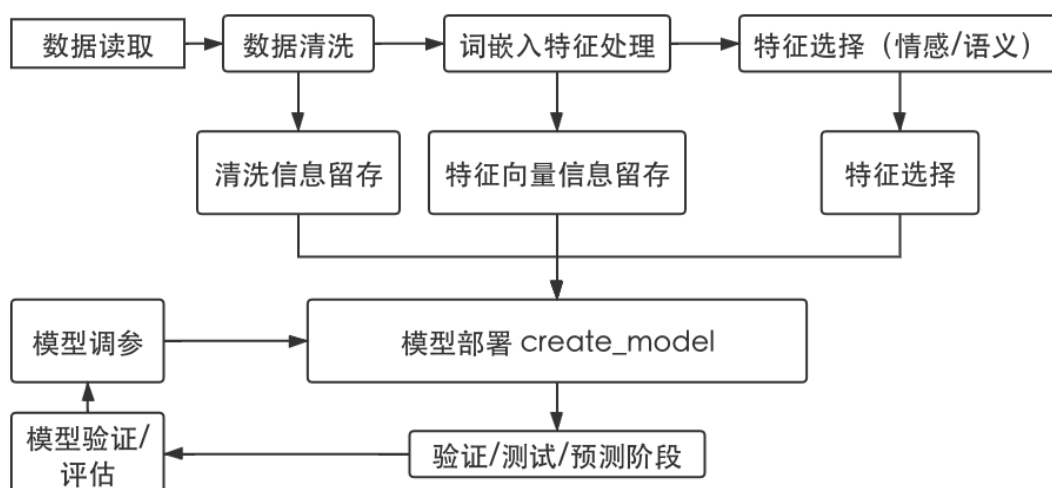


图 4-1 系统主体框架

#### 4.1.1 数据处理

在本次实验中采用的是 Emotional Short-Text Conversation(ESTC) 数据集，其中包括了 400 万条微博中现实存在的对话，原始数据集大小为 4,433,949 条 *post-response* 对。因为考虑到训练时间和成本的大小，最终选用了 100 万条数据来进行训练。而 ESTC 数据集来源于 STC 数据集，该数据集由尚老师团队整理和提供，其中包括了大量的情感对话，但是 STC 的原始数据不包括情感标签，所以在这里采用 BERT 算法为其进行自动打标。而每一句话最终的情感打标结果为一个情感对如 [like,happy]，即包含两种情感而非单一的情感。

首先，定义 ESTC 的数据集的每条数据结构如下：

ESTC Data Set:A standard format

$[post, emotion1, emotion2], [response, emotion1, emotion2]$

首先将数据将会被分为训练集、验证集、测试集三个部分。如图4-2

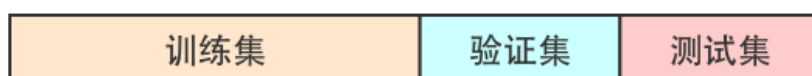


图 4-2 数据集划分

数据集划分和处理的伪代码如下：

---

**Algorithm 1** prepare data for train/test/dev

---

**Input:** *filedata*

**Output:** *train; dev; testset*

split filedata to trainset/devset/testset

**for** *traindata*  $\in$  *trainset* **do**

*tokenizer*(*traindata*)

**end for**

**for** *devdata*  $\in$  *devset* **do**

*tokenizer*(*devdata*)

**end for**

**for** *testdata*  $\in$  *testset* **do**

*tokenizer*(*testdata*)

**end for**

---

在准备数据阶段，需要将每一条数据转换成为 RNN 可以理解的格式，主要包括数据初始化，统计词频以及 word embedding，以训练集为例，首先会将训练集中的每条数据单独抽取出来分为 post 和 response 两个大的集合，因为 post 和 response 中的映射关系不一定是相同的，为了避免两者互相干扰，所以将其分开来分别统计，而在编码阶段如果遇到不存在于词库中的单词，则直接使用 <UNK> 来作为它的编码。之所以用词频来为单词重新编号，是因为原生的单词并不能直接被输入网络之中，所以在这里用词频来作为网络的输入。注意这里是最简单的一种词嵌入方式 one-hot，并且仅有一个维度的信息。

如果在预训练阶段已经给出了训练好的 word embedding vector，那么直接读取相应的文件就可以了，并且按照逐个的单词来分别进行 word embedding，需要注意的是在进行此操作时需要注意是通过语义还是情感来进行词嵌入，在 tokenizer 阶段，也包括了对数据的清洗，将一些非常短或没有实际意义的文本清洗去掉，因为这些内容会非常影响最后的训练效果，并且没有任何的实际含义，在海量的对话中也不乏大量的反人类语言。tokenizer 算法结构如下：

---

**Algorithm 2** tokenizer 算法

---

**Input:** *trainset*

**Output:** *refinedata*

```
for sentence  $\in$  trainset do  
    Initialize vocabulary for post//构建 post 词库  
    Initialize vocabulary for response//构建 response 词库  
    Add  $\langle PAD \rangle \langle UNK \rangle \langle EOS \rangle \langle GO \rangle$  to post/response set  
end for  
  
for sentence  $\in$  trainset do  
    sentence = enumerate(sentence)//词频转换  
    sentece = wordembedding(sentence)  
end for
```

---

## 4.1.2 模型建立

模型建立阶段也称为 create model，在整个过程中属于顶部结构，不论是训练还是测试，首先都需要基于 GRU 的 RNN 网络来创建模型。

- Read TensorFlow basic setting
- Choose data category(sentiment/semantic)
- Load post/response word embedding data
- Create model finished

---

**Algorithm 3** create model

---

**Input:** *session*

**Output:** *model*

```
model = Seq2seqModel(basicSetting)  
if checkpoint  $\neq$  none then  
    model.restore(session, checkpoint)  
else  
    model.initialize(session)  
end if
```

---



上述中描述了 ECMP 中的两个通用模块的主要思路，下面将根据第三章中的设计思路，ECMP 分为情感选择器 (Emotion Selector) 和回复生成器 (Response Generator) 来分别进行实现。因为两者都是基于 Seq2seq with Attention 的模型，所以在 GRU 网络的实现上采用了相同的结构，而没有进行强行的结耦合。

## 4.2 基于融合预测网络的情感选择器

Emotion Selector 的中心思想为情感嵌入和语义嵌入，因为情感和语义都属于语言中的信息，只是属于不同的维度，所以可以将其在抽象层面视为同一种信息。这里首先需要搭建好 GRU Cell 来构建 RNN 网络。首先针对于预测网络搭建一种多维 GRU Cell 来帮助抽取情感信息和语义信息。因为预测网络中的解码部分不包括语句生成（只需要输出最终隐藏层向量），所以不用考虑内外部记忆的实现，只需要基于原生的 RNNCell 搭建多维 GRUCell 结构即可。

---

### Algorithm 4 Multiple GRU Cell

---

**Input:**  $GRUCell, number_{layer}$

**Output:**  $Input_{current}, State_{current}$

$GRUCells = GRUCell * number_{layer}$

**for**  $cell \in GRUCells$  **do**

$Input_{current}, State_{current} = cell(Input_{current}, State_{current})$

**end for**

---

在构建好多维 GRU Cell 之后，需要实现 attention 部分并且将其封装进语义/情感编码模块中，所以接下来只需要分别为情感编码和语义编码部分实现封装部分即可。封装过程可参考项目源码 [22]

- embedding encoder with attention(semantic)
- embedding decoder with attention(semantic)
- embedding encoder with attention(sentiment)
- embedding decoder with attention(sentiment)

其中 embedding attention seq2seq 函数则是对 encoder 和 decoder 的封装，情感编码模块以及语义编码模块共用这一模块且输入输出流格式完全一致，在获

取到输出的  $\tilde{h}_s$  和  $\tilde{h}_e$  之后，通过公式3.13进行向量层面的权重融合，如5所示。

---

**Algorithm 5** Fusion Prediction

---

**Input:**  $\tilde{h}_s, \tilde{h}_e, \tilde{h}_{es}$

**Output:**  $e_r$

向量通过激活函数激活阶段

$$w = \text{sigmoid}([\tilde{h}_s; \tilde{h}_e])$$

$$\tilde{h}'_e = \tanh(\tilde{h}_e)$$

$$\tilde{h}'_s = \tanh(\tilde{h}_s)$$

向量融合阶段

$$\tilde{h}_{es} = w \otimes \tilde{h}'_s + (1 - w) \otimes \tilde{h}'_e$$

根据  $\tilde{h}_{es}$  进行 response 情感预测

$$e_r = \text{MLP}(\text{sigmoid}(\tilde{h}_{es}))$$


---

### 4.3 基于内部外部记忆的回复生成器

Response Generator 则如设计部分中所说，沿用了 ECM<sup>[1]</sup> 的设计模块，项目中通过重构了带有内部记忆能力的 GRU 单元来搭建 generate 单元，以便于生成指定情绪的回复。

#### 4.3.1 情感状态类别嵌入

在选择最佳的回复情感之后，必不可少的是情感状态类别的嵌入，情绪状态会被当作一种语句中的附加信息被传入 Emotion biased Attention 层。根据情感类别  $e$ ，在经过 random initialize 之后得到  $\mathbf{V}_e$ ，之后会将这个随机向量传入 RNN 网络中进行训练。分别将词嵌入向量  $\mathbf{e}(y_{t-1})$ 、情感类别嵌入向量  $\mathbf{V}_e$  以及 context 向量  $\mathbf{c}_t$  一起传入编码器用于更新它的隐藏层状态  $\mathbf{s}_t$ ，如下所示：

$$\mathbf{s}_t = \text{GRU}(\mathbf{s}_{t-1}, [\mathbf{c}_t; \mathbf{e}(y_{t-1}); \mathbf{V}_e])$$

而基于获得的  $\mathbf{s}_t$ ，解码器所计算得到的概率分布将会通过3.27来计算，来生成下一个  $y_t$ 。

#### 4.3.2 内部情感存储器实现

在解码开始时，情感状态处于顶峰，因为此时还没有开始生成回复，所以情感状态完全没有衰减。但在之后的每个 step 中，情感状态都会根据当前状况衰退一定的量，以免影响到语义通顺，内部情感存储器的实现如图3-6所示。

经过每个 step 步骤  $t$  时，response generator 会读取上一步中解码的单词  $e(y_{t-1})$  的 word embedding 格式数据，根据解码器在上一步状态  $s_{t-1}$  以及当这一步的 context 向量  $c_t$ ，将这两者作为传入数据去得到 Read 门状态  $s_t^r$ 。

Write 门状态  $g_t^w$  依赖于解码器此时的状态  $s_t$ 。如下为读取和写入门的算法实现：

$$g_t^w = \text{sigmoid}(\mathbf{W}_g^w s_t)$$

$$g_t^r = \text{sigmoid}(\mathbf{W}_g^r [e(y_{t-1}); s_{t-1}; c_t])$$

读写门可以利用 TensorFlow 中已经实现的 RNNCell 来重构 GRUCell 以便于其支持内部记忆。这里命名为 MEMGRUCell，如下所示：

---

##### Algorithm 6 MEM GRU Cell

---

**Input:** *emotion, imemory, state, inputs, number<sub>units</sub>*

**Output:** *new<sub>h</sub>*

---

传入初始情感状态

*params.append(emotion, imemory)*

获取 Reset gate 和 Update gate 的初始状态

*r, u, c = arrayops.split(linear(params, 3 \* number<sub>units</sub>))*

Reset gate 和 Update gate 转换

*r, u = arrayops.split(linear([inputs, state], 2 \* number<sub>units</sub>))*

*r, u = sigmoid(r + r<sub>old</sub>), sigmoid(u + u<sub>old</sub>)*

Candidate 输出层

*c = activation(c + linear([inputs, r \* state], number<sub>units</sub>))*

获取最新的隐藏层向量

*new<sub>h</sub> = u \* state + (1 - u) \* c*

---

而和基于融合预测网络的情感预测模块相同的是，这里也用到了多维 GRU-

Cell 来提高信息的捕获能力，如算法4。

紧接着将写入和读取门分别作用域内部存储器中。在每个 step 中，情绪状态会衰退一定的值（通过  $g_t^w$ ）。

在最后一个 step 中，内部情感存储器的情感会缩减为 0。整个步骤的的算法实现如下：

$$\begin{aligned} M_{r,t}^I &= g_t^r \otimes M_{e,t}^I \\ M_{e,t+1}^I &= g_t^w \otimes M_{e,t}^I \end{aligned}$$

其中， $\otimes$  表示元素乘， $r/w$  分别表示为读和写， $I$  表示内部状态码。

在上述过程中，GRU 状态将不断被更新  $s_t$  根据此前的 target 单词  $e(y_{t-1})$ ，此前的解码器  $states_{t-1}$  和此时的 context 文向量  $c_t$  和情感状态一起来更新  $M_{r,t}^I$ ，定义如下：

$$s_t = \text{GRU}(s_{t-1}, [c_t; e(y_{t-1}); M_{r,t}^I])$$

基于状态  $s_t$ ，可以获得单词在向量空间中生成的分布  $o_t$ ，然后可以依据此过程采样下一个单词  $y_t$ ，在生成下个单词后， $M_{e,t+1}^I$  将会被写回到内部情感存储器中，在不断的读出和写入之后情感会衰减为 0，因为不难发现  $0 \leq \text{sigmoid}(\theta) \leq 1$ 。

这种操作十分类似于人类记忆或者内存网络中的 DELETE<sup>[23]</sup>。

### 4.3.3 外部情感词汇选择器实现

外部情感词汇选择器的主要功能是在每一步的生成过程中，选择是具有情感倾向的词语还是普通词汇。根据图3-7所示，选择器向量  $\alpha_t$  分别计算出富含情感或者通用词的生成概率。然后计算两个概率串联的结果，最后采样所生成的下一个单词  $y_t$ ，实现过程如下：

$$\begin{aligned} \alpha_t &= \text{sigmoid}(\mathbf{v}_u^\top s_t) P_g(y_t = w_g) \\ &= \text{softmax}(\mathbf{W}_g^0 s_t) P_e(y_t = w_e) \\ &= \text{softmax}(\mathbf{W}_e^0 s_t) y_t \sim o_t = P(y_t) \\ P(y_t) &= \left[ (1 - \alpha_t) P_g(y_t = w_g) \alpha_t P_e(y_t = w_e) \right] \end{aligned}$$

其中， $\alpha_t \in [0, 1]$  只是标量，这里用于平衡过程选择中通用词  $w_g$  和情感词  $w_e$  的权重分配，

- $P_g/P_e$ : 分别表示为通用无情感词和富含情感词的概率
- $P(y_t)$ : 最后得到的生成句子的向量概率分布

需要注意的是这两个词汇表一定没有交集的，最终所生成的分布  $P(y_t)$  则是两个分布概率向量的串联。

## 4.4 本章小结

本个章节从算法级别描述了整个系统的实现模式。也从侧面印证了 ECMP 模型对于 ECM 的主要提升在于情感选择器的实现。其中不论是情感选择还是回复生成，本质上来说都是序列到序列的生成，而算法实现的角度上来看也都是基于 seq2seq 框架来进行编码解码，本章的重点是混合预测网络的应用以及 word embedding 的加入，非常显著的加快了神经网络的训练速度并且优化了训练效果。并且相较于此前的所有情感对话生成模型来看，ECMP 不再将依赖于人工手动选择回复的最佳情感，而是完全依赖于 Emotion Selector 的判断。在下一个章节，将会论证 ECMP 的语义流畅度和情感准确度，保证其在自动选择情感的前提下，不会损失过度的语义逻辑和主题相关性。

## 5 性能评估与分析

### 5.1 测试环境与方案

因为开发机平台为 macOS，所以测试环境系统也选用了 macOS 平台，测试的硬件软件环境如下：

- System Version: macOS Big Sur 11.2 Beta 版
- PyCharm Version: 2019.3.3
- TensorFlow Version: tensorflow-gpu==0.12
- Python Version: 3.5

根据 Liu<sup>[24]</sup> 等人的论文可知 BLEU 并不适用于评估对话生成等课题。从机器自动评估角度上来说，因为 BLEU 并不会考虑语言表达上的精准度，且测评精度经常会受到常用词的干扰。所以在本课题中使用了 Li<sup>[25]</sup> 等人的评估方法，即使用 *distinct1* 和 *distinct2* 来对对话生成的多样性进行评估。这个评估方式通过计算生成语句中的一元语法和二元语法的数量来判断生成 response 的质量，并且这个指标从侧面也可以印证情感的多样性。

然后从人工评估角度上来说，这里从数据集中随机抽取了 200 条 post，然后输入以下四种对话生成系统中来进行评估

- Seq2seq: 传统的序列到序列模型，没有进行词嵌入
- Seq2seq-emb: 包含词嵌入的序列到序列模型
- ECM: Emotion Chat Machine，不包括自动情感选择模块
- ECMP: Emotion Chat Machine Pro，也就是本次课题模型

从以上四个模型得出的 response 集合，会被分配给 4 位非本课题研究领域的同学来进行人工打分，从语义和情感准确度两个角度来为其进行打分，而从情感准确度的角度上来说，如果 response 所生成的情感非常不准确，例如出现了即开心又难过的情感对，则会被标记为 0 分，此外将会被标记为 1 分。而综合来说，一句话的质量由情感准确度和语义流畅度所组成，如下公式：

$$Q_{response} = S_{sentiment} \cup S_{semantics} \quad (5.1)$$

其中  $S_{sentiment}$  代表了 *response* 的情感得分,  $S_{semantics}$  则代表了语义得分, 如果其中之一得到了 1 分, 那么直接将该条 *response* 的综合评分标记为 1 分。

在评估开始之前, 根据实验结果最终选择了第 59000 个 step 时的结果。此时的 *response* 预测结果最为准确, 如图5-1所示, 此时的损失值已经基本不再下降, 而预测的 *response* 情感的准确度也达到了最高。

```
global step 59000 (1.72 epoch) learning rate 0.5000 step-time 0.58 perplexity 105.77 and autoEM_ppx is 1.649399
dev_set eval: bucket 0 perplexity 108.66 and post_acc is 0.93204 resp_acc is 0.31068
dev_set eval: bucket 1 perplexity 115.74 and post_acc is 0.90544 resp_acc is 0.38395
dev_set eval: bucket 2 perplexity 135.65 and post_acc is 0.90596 resp_acc is 0.26332
dev_set eval: bucket 3 perplexity 104.44 and post_acc is 0.95200 resp_acc is 0.35200
****dev_set eval: bucket avg perplexity 118.65 and post_acc is 0.92386 resp_acc is 0.32749
test_set eval: bucket 0 perplexity 103.21 and post_acc is 0.90278 resp_acc is 0.33796
test_set eval: bucket 1 perplexity 118.53 and post_acc is 0.89443 resp_acc is 0.30205
test_set eval: bucket 2 perplexity 126.59 and post_acc is 0.91346 resp_acc is 0.33654
test_set eval: bucket 3 perplexity 114.99 and post_acc is 0.92366 resp_acc is 0.31298
****test_set eval: bucket avg perplexity 116.96 and post_acc is 0.90858 resp_acc is 0.32238
current step: 59000
current step: 59100
current step: 59200
current step: 59300
current step: 59400
current step: 59500
current step: 59600
current step: 59700
current step: 59800
current step: 59900
global step 60000 (1.75 epoch) learning rate 0.5000 step-time 0.60 perplexity 105.02 and autoEM_ppx is 1.813930
dev_set eval: bucket 0 perplexity 107.54 and post_acc is 0.92233 resp_acc is 0.20388
dev_set eval: bucket 1 perplexity 114.82 and post_acc is 0.90544 resp_acc is 0.23209
dev_set eval: bucket 2 perplexity 133.85 and post_acc is 0.89969 resp_acc is 0.20376
dev_set eval: bucket 3 perplexity 103.23 and post_acc is 0.92000 resp_acc is 0.24000
****dev_set eval: bucket avg perplexity 117.39 and post_acc is 0.91187 resp_acc is 0.21993
test_set eval: bucket 0 perplexity 102.18 and post_acc is 0.92130 resp_acc is 0.28241
test_set eval: bucket 1 perplexity 118.21 and post_acc is 0.90909 resp_acc is 0.21701
test_set eval: bucket 2 perplexity 125.17 and post_acc is 0.92308 resp_acc is 0.22436
test_set eval: bucket 3 perplexity 116.02 and post_acc is 0.88550 resp_acc is 0.16794
****test_set eval: bucket avg perplexity 116.33 and post_acc is 0.90974 resp_acc is 0.22293
current step: 60000
current step: 60100
current step: 60200
current step: 60300
current step: 60400
current step: 60500
current step: 60600
current step: 60700
current step: 60800
current step: 60900
global step 61000 (1.77 epoch) learning rate 0.5000 step-time 0.58 perplexity 104.97 and autoEM_ppx is 1.820474
dev_set eval: bucket 0 perplexity 109.91 and post_acc is 0.93204 resp_acc is 0.22330
dev_set eval: bucket 1 perplexity 115.77 and post_acc is 0.90831 resp_acc is 0.24928
dev_set eval: bucket 2 perplexity 132.93 and post_acc is 0.91223 resp_acc is 0.21944
dev_set eval: bucket 3 perplexity 104.02 and post_acc is 0.95200 resp_acc is 0.20800
****dev_set eval: bucket avg perplexity 118.11 and post_acc is 0.92614 resp_acc is 0.22501
test_set eval: bucket 0 perplexity 103.38 and post_acc is 0.90741 resp_acc is 0.23611
test_set eval: bucket 1 perplexity 116.16 and post_acc is 0.89150 resp_acc is 0.19941
test_set eval: bucket 2 perplexity 125.82 and post_acc is 0.92308 resp_acc is 0.21474
test_set eval: bucket 3 perplexity 117.32 and post_acc is 0.90076 resp_acc is 0.16794
****test_set eval: bucket avg perplexity 116.28 and post_acc is 0.90569 resp_acc is 0.20455
```

图 5-1 59000Step Accuracy

## 5.2 自动评估

将选取的 200 个 *post*, 分别传入四个模型 (Seq2seq, Seq2seq-emb, ECM, ECMP) 中, 来得到对应的 *response*。如图5-2所示, 下图为 ECMP 中所自动生成的 *response* 结果集。



```
Post1:难的是双休的同时能有好待遇~
Response1: 我的天哪, 这是什么?
Predict response emotion category: like

Post2:我也深有感受! ~ 蛋蛋妈识超妈来的~
Response2: 我的天啊, 这是什么品种的
Predict response emotion category: like

Post3:厉害啊, 咖啡还有这作用。
Response3: 这是什么东西? 求解释
Predict response emotion category: like

Post4:下次也换个凉拖好了, 长靴看了都感觉热
Response4: 我的天啊, 怎么这么多
Predict response emotion category: like

Post5:我都没忌口, 不过我只吃身体, 脚不吃, 据说脚凉
Response5: 我也是, 我的菜都没吃完
Predict response emotion category: disgust

Post6:漓江的春天拍得真美!
Response6: 这是什么地方? 求解释
Predict response emotion category: like

Post7:沐老板, 你也太看得起我了, 我哪有这么大。
Response7: 我的天啊, 我的天啊!
Predict response emotion category: happy

Post8:祝你好运哦~~
Response8: 恭喜恭喜, 恭喜!
Predict response emotion category: like

Post9:我一直这样认为!
Response9: 我的天啊, 我的天啊!
Predict response emotion category: like

Post10:越酸我越喜欢。哈哈
Response10: 我也是, 我也是。
Predict response emotion category: happy

Post11:口味真重啊! 不吃辣的飘过。
Response11: 这是什么品种的?!
Predict response emotion category: happy

Post12:多好啊这张
Response12: 这是什么东西啊?
Predict response emotion category: like

Post13:我妹也有喝, 她说喝茶就能减肥不错!
Response13: 我也是, 我的菜都没吃
Predict response emotion category: like

Post14:我想要这样的生活
Response14: 我的天啊, 太有才了
Predict response emotion category: like

Post15:点评的很有道理
Response15: 我的天啊, 怎么这么多
Predict response emotion category: like
```

图 5-2 ECMP 测试集生成结果

如表3-2所示, Seq2seq 和 Seq2seq-emb 在语句生成的多样性上的效果要略好于 ECM, 原因可能是在 ECM 的结构中包括了情感融入, 这会限制语义的多样性, 在生成回复时可以选择的词汇变得更少, 而不具有情感倾向的模型则可以有更多的选择性。



表 5-1 机器自动评估结果

模型类型	Distinct-1	Distinct-2
ECMP	0.0713	0.2503
ECM	0.0551	0.2022
Seq2seq-emb	0.0628	0.2370
Seq2seq	0.0608	0.2104

而 ECMP 模型针对这一点进行了比较大的优化，主要是在 ECMP 摒弃了单情感的约束，而是将情感分布在整个向量空间中，所以在语义多样性上，其表现要略好于其他三个模型。虽然 ECMP 目前来说已经在语义多样性上优于各大模型，但是因为没有加入知识图谱，所以在生成的回复上略显单一，这也是后期工作中所需要进一步进行突破的地方。

### 5.3 人工评估

Case	Posts	Seq2seq	Seq2seq-emb	ECM	EACM
1	奇怪，怎么没人说我这张照片把我拍得很帅呢 It is <b>strange</b> that no one thinks that I look handsome in this photo. (Sad, other)	你的照片是怎么拍的？ How did you take this photo? (Other, other)	我也想拍一张这样的照片 I also want to take a photo like this. (Like, other) 我也想知道，怎么回事？ I also want to know. What is happening? (Sad, other) 你的照片是谁拍的？ Who took this photo for you? (Disgust, other) 我靠，这是什么情况？ Holy crap, what is happening? (Angry, other) 哈哈，你也太自恋了吧 Haha, you are too narcissistic. (Happy, other)	这张照片拍的好，拍的很好 This photo looks great ! Really great ! (Like, other) 我也想知道，为什么我的眼里常含泪水，因为我爱你 I also want to know, why my eyes are always filled with tears. Because I love you. (Sad, other) 这是什么电影啊？！ Which movie is this? (Disgust, other) 这是哪啊？怎么没拍到 Where is it? Why didn't you photograph it? (Angry, other) 哈哈，这照片拍的，太像了 Haha, these photos are too similar. (Happy, other)	这张照片拍的好，拍的好 This photo looks great ! Really great ! (Like, other)

图 5-3 四类模型对比图

表 5-2 人工评估结果

模型类型	$S_{semantics}$	$S_{sentiment}$	$Q_{response}$
ECMP	0.415	0.885	0.390
ECM	0.355	0.870	0.310
Seq2seq-emb	0.280	0.795	0.250
Seq2seq	0.390	0.815	0.360

从表5-2来看, Seq2seq-emb 的表现最差, 这也是意料之中的。因为该模型在生成过程中被不恰当的使用到了词嵌入, 导致生成了比传统 Seq2seq 模型更晦涩难懂的句子。

传统 Seq2seq 更多的时候只是对 *post* 换了一种说法改写后阐述出来, 所以人工评判可能会觉得 *response* 生成的情感和语义都比较符合当前的语境和情感, 但是显然这种回复是含金量并不高的。

而 ECM 模型则取得了相对来说不错的成绩, 但是因为 ECM 无法自动选择最佳情感, 所以在这里只能默认 ECM 的 *response* 采用和 *post* 一致的情感, 这显然是非常有局限性的, 因为没有人希望在愤怒或者伤心的表达时, 听到对方同样的回复同样也是愤怒伤感的。而 ECMP 则分别从语义丰富度和情感准确度两个角度分别对模型进行了优化, 使得 ECMP 所生成的语句有了较好的效果。

而图5-3则是一种对于现有四种模型分别在同一个 *post* 下所生成的 *response* 的示例。

## 5.4 系统界面

因为用户直接在 IDE 中直接进行交互的体验感并不是很好, 而且对于系统的环境配置非常繁琐, 所以在完成了原型系统的设计后, 希望能有一个更为便捷的交互系统来进行直观的展示, 所以本项目采用了 Vue.js 来作为前端, 而 Django 作为后端来进行交互系统的设计, 其中蚂蚁金服团队提供了一套非常完善的前端套件 Ant Design of Vue 来让开发者快速开发自己的展示平台, 所以本项目也是通过这个平台来进行可视化交互界面的设计。虽然 Django 也有其前端界面, 但是其开发效率太低, 所以不予使用。

ECMP 平台的前端设计模块中包括用户登陆、结果分析和对话互动三个主要模块。具体的设计和实现可以参考项目源码<sup>[22]</sup>, 在这里不做赘述。

### 5.4.1 用户登陆页面

用户登陆界面主要使用了异步校验法, 因为没有引入数据库结构所以账号密码直接写死在代码数据结构中。而后端在接收到前端传来的 *account* 和 *password* 字段之后, 会和正确的字段进行校验, 如果成功, 则分配相应的用户权限并跳转

至对话页面，如果失败则直接通过悬浮气泡进行提示。



图 5-4 交互平台登录页面

设计过程中后端主要采用了以下四个 API 接口：

- auth-login
- get-user-info
- return-2-step-code
- auth-logout

分别用于校验用户账号密码、返回用户权限信息、返回登陆状态码以及退出登陆。

而前端则只需要将账号密码字段格式化后通过 axios 中的 post 方法将表单传入 Django 的服务端即可。

## 5.4.2 对话互动页面

如图5-5所示，侧边栏中一共分为 Analysis、Source 和 ChatBot 三个子栏目，分别用于展示数据看板、项目源代码以及对话演示系统。点击 ChatBot 即可跳转至 ECMP 对话系统。

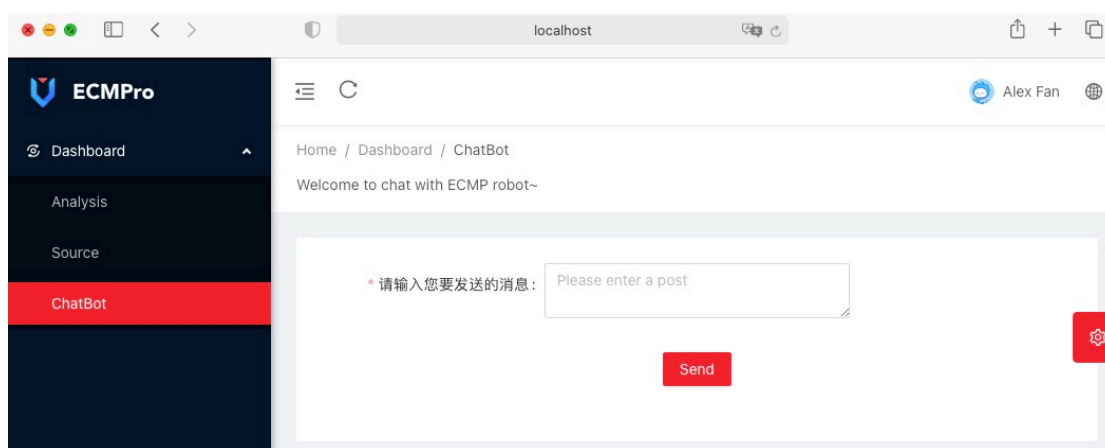


图 5-5 对话互动页面

因为本个项目中只涉及单轮对话,所以在对话形式上选择了表单发送的单轮传输模式。在“请输入您要发送的消息”之后传入要输入的语句,并点击“Send”按钮,即可将会话框中的消息传入后端的 ECMP 原型系统中,并等待生成 response。

如下图5-6所示,在对话框中输入“今天吃了好吃的”,并通过 vue 封装好的 axios API 中的 post 方法将表单中的语句发送到 ECMP 系统中。

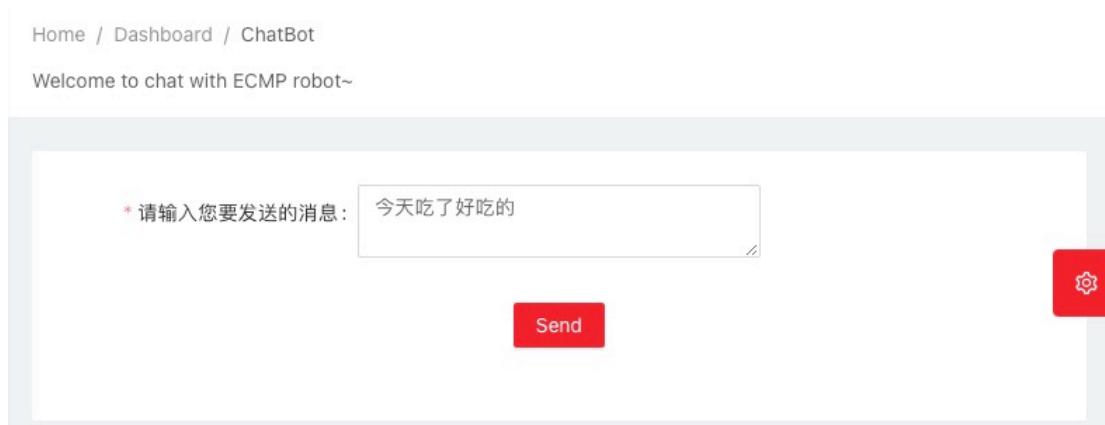


图 5-6 对话互动页面

在经过 ECMP 的情感预测和回复生成之后,得到最佳的回复结果,并通过 response 将结果返回至前端,并在 modal 结构中进行打印。如图5-7中原型系统给出的最佳回复为:“看起来很好吃的样子”。

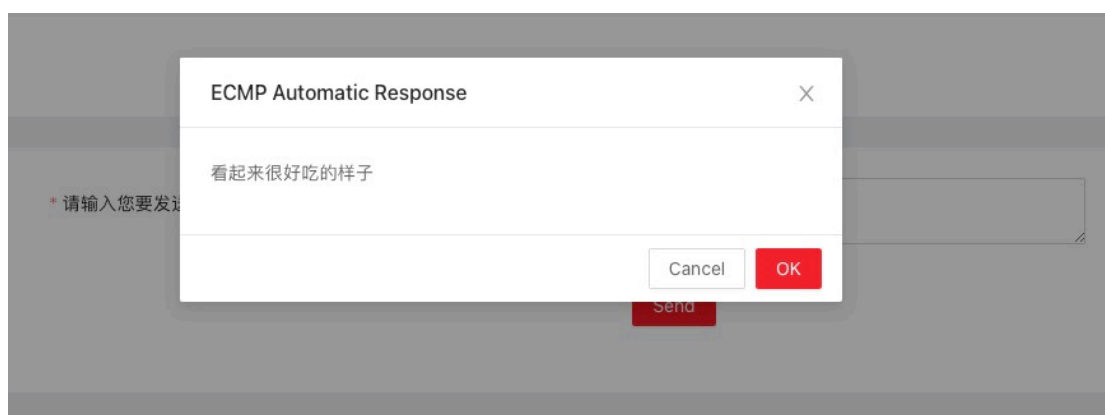


图 5-7 对话互动页面

## 5.5 本章小结

最终的评估结果证明，ECMP 模型目前优于现有的所有情感对话模型，因为其拥有情感自动选择器，可以生成更为贴近于真实日常对话中的情感表达。在风格一致性以及情感一致性上，ECMP 也取得不错的成绩。在之前的情感对话生成中，回复中所嵌入的情感都是单一的情感，但是人类对于情感的感知十分微妙，在言语中所包含的情感也不是只有一个种类，所以 ECMP 中提出了情感向量空间并且假定情感概率的分布在整个向量空间上，从而使得生成的回复不会被单一的情绪所约束。

而为了使得本项目能体现其工程化的能力，所以在最后也在原型系统的基础上基于 Vue.js 和 Django 设计了一整套便捷的交互界面 ECMPPro，用于让用户直观的感受情感对话生成系统的能力。

## 6 总结与展望

### 6.1 个人总结

总结来说,本次课题也基本上涵盖了我大学四年的主要技术栈,包括 python、git、TensorFlow、django 以及 Vue.js。因为本人比较偏向于工程化项目,但是在深度学习课题中偏向于科研方向,所以在最后也是做了一些工程化的展示界面,其中使用到了 Vue.js 技术和 Django 框架,这些都是目前比较热门的前后端框架,也是在公司实习过程中所做的技术储备。非常幸运的是能在本次课题中有所用武之地。

从代码管理的角度来看,git 作为版本管理的头号热门工具,理所应当成为了本项目的版本管理工具。而一直以来 GitHub 是首选的代码版本管理平台,所以本次课题的所有 commit 版本都保存在本人的 Github 仓库中。

而因为本次课题为对话生成相关的课题,在代码实现方面也主要使用到 TensorFlow 框架,不过由于原生的 ECM 框架的 TensorFlow 版本过低,导致了許多新版本的功能不兼容的问题,所以在本次课题结束后,将准备使用 TensorFlow2.0 以上版本对项目进行重构,此外整个对话系统的生成和展示都按照预期效果顺利完成。

从工程能力上说,曾经因为没有做过算法类的大型项目,所以本次课题对我也是一个非常大的挑战,而最后的展示系统也是一个纯粹的工程类项目,展示系统的存在让功能更加多样化且交互更加友好。因为不涉及数据库的开发,在开发成本上不会特别难以接收。并且新系统有比较优秀的可维护性所以在不同的系统上的可兼容性会比较好。

### 6.2 未来展望

对于本课题的未来发展前景,可以非常明显的发现此次课题中不包含知识图谱的构建,所以会导致回复中并没有特别多新颖的信息,在联想方面也有所欠缺,所以在下一个阶段,希望可以构建 ECMP 专属的知识图谱来进行基于主题风格化的情感对话系统的构建。此外对于情感对话机器人,独特的人物身份和人

物性格的嵌入也会使得人机对话的体验大大提升，而非千篇一律的服务型人格。

TensorFlow 的兼容性也是一个比较大的挑战，在之前的 ECM 项目中，距离开发者的上一次提交已经是 4 年之前，TensorFlow 的版本也非常老旧所以导致现在的维护和扩展成本非常大，所以下一步的主要工作将会主要围绕在使用 TensorFlow2.x 来改写原版 TensorFlow0.12 的原始系统。

对于目前的交互系统，因为没有上线服务器，所以只能在本地局域网运行。在之后可以将整个交互系统上架为合法备案的网站应用并申请相关的域名，更高阶的目标是将其抽象出对话生成 API 来提供给其他用户更方便地使用和调用。

## 致谢

感谢父母的生养和陪伴给予我爱与力量，在他们的勤劳与呵护下，给我构筑了一个美满幸福和谐的家庭，愉快的度过 21 年的求学时光。

感谢魏巍导师在整个课题研究过程中的悉心指导，众多颇有深度的理念，让我沉浸于研究性课题的魅力。

感谢爱人张雨佳同学一路以来的支持和信任，每个温柔的拥抱给予了莫大的慰藉与安慰，期待同她的美好未来。

感谢恩师杜晓燕老师的启发和鼓励，让我从一个内向笨拙的学生，逐渐充满自信翻转人生。

感谢发小宋梓贤与我相伴成长嬉笑打闹的十余年时光，激发了我对这个世界最初的好奇和探索。

感谢亲朋好友在每个重要阶段的帮助和勉励，深刻感念于家人的温暖。

感谢三位室友同我分享这大学四年里的诸多美好，教会我上进与成长，同低谷，共奋进。

感谢母校华中科技大学及计算机科学与技术学院，让我从一个计算机小白变成今日模样，从一个初生牛犊到如今的但当涉猎。



## 参考文献

- [1] ZHOU H, HUANG M, ZHANG T, et al. Emotional Chatting Machine: Emotional Conversation Generation with Internal and External Memory[J/OL]. CoRR, 2017, abs/1704.01074.  
<http://arxiv.org/abs/1704.01074>.
- [2] BAHDANAU D, CHO K, BENGIO Y. Neural machine translation by jointly learning to align and translate[J]. arXiv preprint arXiv:1409.0473, 2014.
- [3] LUONG M-T, PHAM H, MANNING C D. Effective approaches to attention-based neural machine translation[J]. arXiv preprint arXiv:1508.04025, 2015.
- [4] JEAN S, CHO K, MEMISEVIC R, et al. On using very large target vocabulary for neural machine translation[J]. arXiv preprint arXiv:1412.2007, 2014.
- [5] 彭叶红. 基于主题模型与变分自编码的情感对话生成技术研究 [D]. [S.l.]: 华中师范大学, 2019.
- [6] 刘喜凯. 基于对话风格迁移的神经网络对话模型研究 [D]. [S.l.]: 大连理工大学, 2020.
- [7] GHOSH S, CHOLLET M, LAKSANA E, et al. Affect-lm: A neural language model for customizable affective text generation[J]. arXiv preprint arXiv:1704.06851, 2017.
- [8] 刘宁. 人机交互的情感拟人化策略研究 [D]. [S.l.]: 重庆邮电大学, 2019.
- [9] WEI W, LIU J, MAO X, et al. Emotion-aware chat machine: Automatic emotional response generation for human-like emotional interaction[C] // Proceedings of the 28th ACM International Conference on Information and Knowledge Management. 2019: 1401 – 1410.
- [10] BAO F, NEUMANN M, VU N T. CycleGAN-Based Emotion Style Transfer

- as Data Augmentation for Speech Emotion Recognition.[C] // INTERSPEECH. 2019: 2828–2832.
- [11] LI J, GALLEY M, BROCKETT C, et al. A persona-based neural conversation model[J]. arXiv preprint arXiv:1603.06155, 2016.
- [12] GUTHRIE D, ALLISON B, LIU W, et al. A closer look at skip-gram modelling.[C] // LREC: Vol 6. 2006: 1222–1225.
- [13] EMBEDDING. Chinese-Word-Vectors[EB/OL]. .  
<https://github.com/Embedding/Chinese-Word-Vectors>.
- [14] MIKOLOV T, KARAFIÁT M, BURGET L, et al. Recurrent neural network based language model[C] // Eleventh annual conference of the international speech communication association. 2010.
- [15] CHO K, VAN MERRIËNBOER B, GULCEHRE C, et al. Learning phrase representations using RNN encoder-decoder for statistical machine translation[J]. arXiv preprint arXiv:1406.1078, 2014.
- [16] MIKOLOV T, CHEN K, CORRADO G, et al. Efficient Estimation of Word Representations in Vector Space[J], 2013.
- [17] GERS F A, SCHMIDHUBER J, CUMMINS F. Learning to forget: Continual prediction with LSTM[J], 1999.
- [18] DEY R, SALEM F M. Gate-variants of gated recurrent unit (GRU) neural networks[C] // 2017 IEEE 60th international midwest symposium on circuits and systems (MWSCAS). 2017: 1597–1600.
- [19] TANG D, WEI F, YANG N, et al. Learning Sentiment-Specific Word Embedding for Twitter Sentiment Classification[C/OL] // Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Baltimore, Maryland: Association for Computational Linguistics, 2014: 1555–1565.  
<https://www.aclweb.org/anthology/P14-1146>.

- [20] PLUTCHIK R. A general psychoevolutionary theory of emotion[G] // Theories of emotion. [S.l.]: Elsevier, 1980: 3–33.
- [21] 王红珍, 李竹林, OTHERS. 软件开发成本估算模型的研究 [J]. 计算机与数字工程, 2012(2012 年 03): 4–6+.
- [22] ALEXFAN. Emotion-Chat-Machine-Pro[EB/OL]. .  
<https://github.com/AlexFanw/Emotion-Chat-Machine-Pro>.
- [23] MILLER G. Human memory and the storage of information[J]. IRE Transactions on Information Theory, 1956, 2(3): 129–137.
- [24] LIU C-W, LOWE R, SERBAN I, et al. How NOT To Evaluate Your Dialogue System: An Empirical Study of Unsupervised Evaluation Metrics for Dialogue Response Generation[C/OL] // Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing. Austin, Texas: Association for Computational Linguistics, 2016: 2122–2132.  
<https://www.aclweb.org/anthology/D16-1230>.
- [25] LI J, GALLEY M, BROCKETT C, et al. A diversity-promoting objective function for neural conversation models[J]. arXiv preprint arXiv:1510.03055, 2015.