学号__20174507__ 密级_____

东北大学本科毕业论文

基于片段抽取的
机器阅读理解系统设计与实现

学 院 名 称 ：计算机科学与工程学院

专 业 名 称 ：物联网工程

学 生 姓 名 ：王德君

指 导 教 师 ：王会珍讲师、王厚峰教授

二〇二一年六月

郑 重 声 明

本人呈交的学位论文，是在导师的指导下，独立进行研究工作所取得的成果，所有数据、图片资料真实可靠。尽我所知，除文中已经注明引用的内容外，本学位论文的研究成果不包含他人享有著作权的内容。对本论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确的方式标明。本学位论文的知识产权归属于培养单位。

本人签名： 日期：

摘 要

随着信息技术的不断进步，用户获得信息的途径日益丰富，互联网上的信息量也在迅速增加。人们往往会因为获取的信息量太大而无法筛选出对自己有价值的信息，从而造成信息过载。作为自然语言处理领域的重要技术，机器阅读理解（MRC）可以从较长的文章中检索出指定问题的答案，在很大程度上可以缓解上述问题。本文以 Albert 模型为基础，设计实现了多种基于片段抽取的 MRC 系统，并开发了阅读理解应用软件 QA App。

本文的主要工作是数据获取、MRC 系统的设计、实验以及应用实现。

数据获取：通过下载 SQuAD2.0 等开源数据集，以及对数据集文件进行拆分，最终获得了 140,000 多个问答对作为训练集和验证集。

MRC 系统的设计与实验：除复现原论文中的 Albert 模型作为基线外，本文还自行设计了 2 种基于 Albert 的 MRC 模型，分别是基线与 Pointer Network 的融合和 4 折 Albert 集成模型。本文通过实验对 3 种模型进行微调训练、对比测试和参数调优，发现集成模型效果最好。

MRC 系统的应用：实现了演示系统 QA App。该系统使用了 C/S 架构，是将实验中调优完全的 4 折 Albert 集成模型封装成 API 部署在服务端，对客户端的请求数据进行答案预测，即时结果返回并在客户端界面展示。客户端使用 PyQt5 开发，提供实时准确率显示功能，有助于用户随时了解模型的运行效果。

本文设计实现的片段抽取式机器阅读理解系统能够大幅加快人们带着问题阅读文章的速度，提高信息检索的效率，并为以后有关机器阅读理解的学习与研究工作提供参考。本系统将在答案形式的拓展、更多 SOTA 模型的尝试以及语言支持等方面，进一步完善。

关键词：机器阅读理解（MRC）；Albert；Pointer Network；4 折；C/S 架构

ABSTRACT

With the continuous progress of information technology, there are more and more ways for users to obtain information, and the amount of information on the Internet is also increasing rapidly. People often get too much information to screen out valuable information, which leads to information overload. As an important technology in the field of natural language processing, machine reading comprehension (MRC) can retrieve the answers to specific questions from long articles, which can alleviate the above problems to a great extent. Based on Albert model, this thesis designs and implements a variety of MRC systems based on span extraction, and develops a reading comprehension application called QA App.

The main work of this thesis is data acquisition, and design, experiment as well as application of MRC system.

Data acquisition: by downloading open source datasets such as SQuAD2.0 and splitting the dataset files, more than 140,000 Q & A pairs were obtained as training sets and validation sets.

Design and experiment of MRC system: in addition to reproducing the Albert model in the original paper as the baseline, this thesis also designs two kinds of MRC models based on Albert, which are the fusion of baseline and Pointer Network and the 4-fold Albert ensemble model. In this thesis, three types of models are trained, tested and optimized by experiments, and the results show that the ensemble model is the best.

Application of MRC system: QA App is implemented. The system uses the C/S architecture, which encapsulates the completely optimized 4-fold Albert ensemble model in the experiment into API deployed on the server side, answers the client's request data, instantly returns the results and displays them in the client GUI. The client is developed with PyQt5, and it also provides real-time accuracy display, which helps users understand the runtime performance of the model at any time.

The machine reading comprehension system designed and implemented in this thesis can greatly speed up reading articles with questions, improve the efficiency of information retrieval, and provide references for future study and research about machine reading comprehension. The system will be further improved in the expansion of answer forms,

more attempts on state-of-the-art models and language support.

Key words: Machine Reading Comprehension; Albert; Pointer Network; 4 Folds; C/S Architecture

目 录

1 绪论

1.1 研究背景及意义

随着信息技术的不断进步，用户获取信息的方式越来越便捷。同时，互联网应用的发展和普及使得网络信息量迅速增加，用户往往沉迷在信息的汪洋大海中无法自拔，无法快速得到对自己有用的信息，从而造成了信息过载（Information Overload）问题。

搜索引擎（如 Bing、Google 等）的出现一定程度地缓解了信息过载问题。搜索引擎通过在互联网中采集信息，并对信息进行排序，为用户提供检索服务。但搜索引擎的不方便之处也很明显：搜索词大都采用关键词而非问句形式，难以准确表达用户意图；此外，搜索产生的结果通常是网页列表，用户需要选择一个网页进入，再阅读其内容才能筛选出想要的问题答案。人工的选择和阅读一定程度上降低了用户在搜索引擎上的检索效率和使用体验。

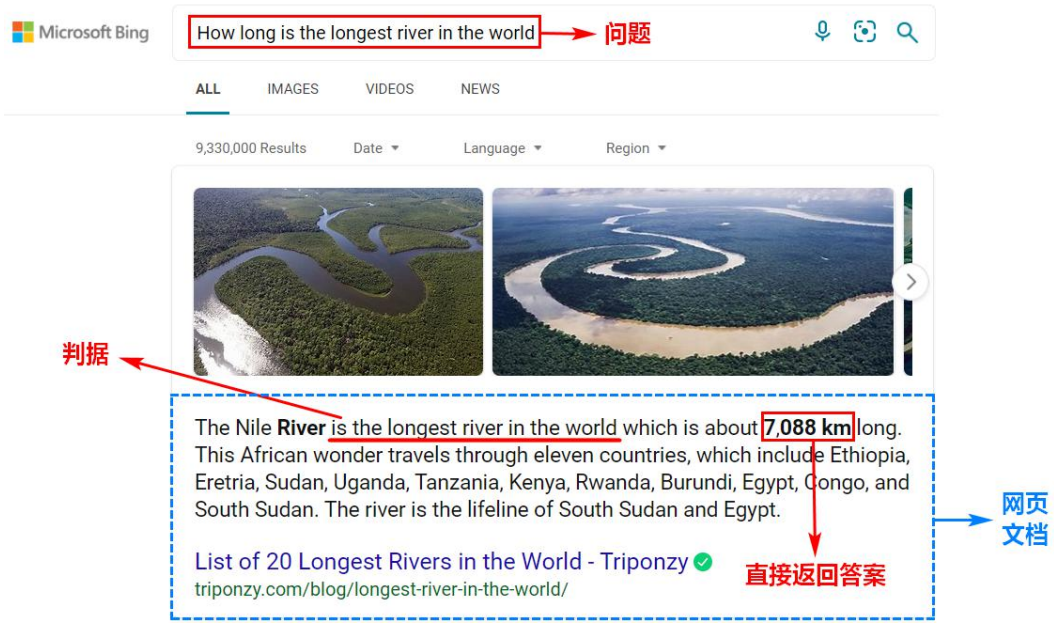

图 1.1 一个应用 MRC 技术的搜索引擎实例 Bing（https://www.bing.com）

机器阅读理解（Machine Reading Comprehension，简称 MRC）技术在很大的程度上可以缓解上述问题。相对于原始搜索引擎，机器阅读理解可以通过网页排序和文本-问题匹配得到问题答案并将结果直接呈现给用户。如图 1.1 所示，使用 MRC 技术的搜索引擎能够根据用户提出的问题检索信息并直接呈现答案，而不再是相关的 web

页面列表。此外，配备 MRC 系统的智能助手可以阅读帮助文档，为用户提供高质量的咨询服务。

机器阅读理解是指让计算机理解给定的自然语言文本内容，从而对与文本有关的特定问题作出解答，反映的是 AI 对一门语言语义的掌握、理解和挖掘水平。未来，该技术将在问答、搜索、智能客服等领域大放异彩，可能会颠覆人与机器之间原有的交互方式。

1.2 国内外研究现状

1.2.1 国外研究现状

MRC 系统最早可以追溯到 1977 年 Lehnert 在其博士毕业论文[4]中提出的 QUALM 系统。然而，由于其规模小、适用范围受限等问题，该系统未得到广泛的应用。直到 1999 年，Lynette 等人[5]发布了一个 MRC 数据集，其中包含了小学三至六年级材料中的故事以及五种“wh”（what、where、when、why 和 who）问题，这才再次将研究注意力引向了 MRC 任务。当时，解决 MRC 任务的方法主要是基于规则或机器学习的方法：Lynette 等人[5]提出了一种词袋技术，将带有问题和上下文的句子表示为一组词，并选择出现在问题和上下文中的词作为答案。Riloff 等人[6]设计了一个基于规则的 MRC 系统 Quarc，它包含了不同的启发式规则和形态分析，例如词性标注、语义类标注和实体识别，用于不同类型的 wh 问题。Poon 等人[7]使用了一种结合自举、马尔可夫逻辑和自监督学习的机器学习方法来进行机器阅读。但是，这些方法存在以下一些限制：

（1）它们主要基于需要大量人力的手工规则或特性；

（2）这些系统不具备泛化能力，并且它们的性能可能会因为大量的文章类型的数据集而降低；

（3）一些传统的方法不仅忽略了长程依赖，而且无法提取上下文信息。

因为机器学习方法和基于规则方法自身存在短板，并且学术界缺乏大规模数据集，在很长一段时间内 MRC 系统效果一直都不好，无法在实际生产生活中得到应用，MRC 的研究进展很缓慢。

2015 年以来，随着深度学习方法在 NLP 领域中的兴起，MRC 系统的表现能力才有了大的提升，这可以归因于两种驱动力：一方面，基于深度学习的 MRC（即神经机器阅读理解）在获取语境信息方面显示出了其优越性，大大优于传统的阅读理解系

统；另一方面，问答数据集的制作成本降低，各种大规模基准数据集纷纷提出，如CNN/Daily Mail[15]、斯坦福问答数据集（SQuAD）[11]和微软发布的 MS MARCO[16]等，使得利用深层神经网络解决 MRC 任务成为可能，并为评估 MRC 系统性能提供了试验平台。

神经 MRC 的主要结构包括嵌入模块（Embedding）、特征提取模块、上下文-问题交互模块、答案预测模块。其中嵌入模块用于将输入的文本按词（或字）切分，并转成向量表征，模型对语言的理解程度差异主要就体现在这一模块。Dhingra 等人[8]指出，在词汇表达上的细微选择会导致系统最终表现的巨大差异。模型结构创新也多发生在这一层。

2018 年，Jacob Devlin 等人基于 Transformers[2]的 Encoder 框架提出了 Bert[3]预训练语言模型，该模型能够很好地利用自注意力机制学习到上下文相关的词嵌入（Contextual Word Embedding），在 SQuAD、MNLI[19]等十几个下游任务上取得了里程碑式的提高，逐渐成为学界处理 MRC 任务的优先选择。Bert 所代表的预训练语言模型往往分成两阶段进行模型训练：第一阶段（即预训练阶段）选用维基百科等大规模语料库作为预训练数据，MLM（Masked Language Model）、NSP（Next Sentence Prediction） 等作为预训练任务，目的是尽可能提高模型对自然语言的理解和泛化能力，从而获得效果更好的 Embedding；第二阶段（即微调阶段）则针对具体的下游任务，在具体的任务数据集上进行进一步训练。近几年新提出的高效 MRC 模型大多都是对 Bert 系列的改进。

在片段抽取的机器阅读理解和开放式问答领域，比较热门的英文数据集有 SQuAD2.0[12]、TriviaQA[13]等，中文的有百度发布的开放领域问答数据集 DuReader[14]等。在这些数据集上表现出色的模型有 Albert[1]、XLNet[21]等。其中，Albert 是对 Bert 的一种“瘦身”，通过对词嵌入矩阵的参数分解和所有 Encoder 层的层间参数共享等手段减少了 Bert 的参数量，大幅节约了模型所占用的内存空间，但同时在 SQuAD2.0 等 QA 数据集上仍取得了不输于 Bert 的效果。而 XLNet 则将 Bert 与广义自回归语言模型相结合，也取得了不错的效果。

1.2.2 国内研究现状

国内对 MRC 系统的研究发展势头迅猛，近年来不少学者进行了阅读理解相关的研究，大量的成果相继涌现。

阅读理解领域一个主要的改进方向就是预训练语言模型预训练任务的选择，在这

一方向的改进可以让所有下游 NLP 任务获得提高。2019 年，百度公司的 Sun 等人[22]认为原生 Bert 基于 token 级别的预训练任务 MLM 不能够很好地学习到中文词与词之间的语义关系，故提出了 Ernie1.0。该模型与 Bert 的区别主要体现在预训练阶段新加入实体级别和短语级别的 MLM 任务，以及新提出的对话语言模型 DLM 任务。Ernie1.0 模型在情感分析、命名实体识别等 5 项中文自然语言处理任务榜单上登顶。同年，该团队又提出了效果更强的 Ernie2.0[23]，该模型在预训练阶段引入词法层、语法结构层、语义层共 3 层 7 个任务，从而在预训练阶段实现多轮多任务持续学习，极大地增强了模型的通用语义表示能力。Ernie 2.0 在中文和英文共十几个 NLP 任务上的表现几乎全部超过 XLNet 和 Bert，并在包括 GLUE 在内的多个榜单上获得了第一。

此外，在问答系统领域，上海交通大学的赵海等人在 Bert 的基础上提出了回顾式阅读理解器 Retro-Reader[24]，在 SQuAD2.0 的榜单上获得了第一的 EM 和 F1 成绩（2020 年 1 月）。该模型创新性地提出了略读模块和精读模块。略读模块由编码层、交互层和外部前置验证器（E-FV）构成。编码层用于将输入文本编码，进而输入到多层 Transformer 交互层构建篇章和问题间的向量空间关系，得到的隐层向量表示在验证器得到可回答性的初步预测。精读模块使用与略读模块同样的编码和交互。在得到隐层向量表示后，在输入线性层得到用于可回答问题的起止位置概率的同时，并行输入到内部前置判别器（I-FV），得到不可回答的概率。最后，E-FV 和 I-FV 的不可回答概率在后置判别器（RV）中融合得到最终的回答决策。

1.3 可行性分析

这样的机器阅读理解系统，在信息检索效率和准确率方面带来提升，且界面友好。此外，该项研究在社会、经济和环境等方面拥有更深远的影响。

在社会上，人类日常生活中有不少需要阅读理解的场景，通过 MRC 系统，可以逐渐让机器取代人类根据问题从大段文本中查找信息，从而加快信息检索的效率。节约出的时间可以进行更多创造性活动，推动社会发展。

在经济上，应用 MRC 系统可以带动一系列科技产业，比如智能化搜索引擎和智能聊天机器人等，促进经济繁荣。

在环境上，MRC 系统可以加快人们查阅信息的速度，从而了解更多环境保护方面的知识，激发大家的环保意识。

1.4 本文主要工作

蓝振忠等人于 2019 年在论文[1]中提出了 Albert 模型。本文基于该模型，对模型进行复现并使用 3 种方式加以改进，以此提高模型性能。同时，本文深入地探究了影响模型性能的因素并得出了相应结论。最后，实现了英文机器阅读理解演示系统 QA App。

（1）模型复现

Albert 模型同时在训练速度、参数量和模型效果上取得巨大成功。另外，该模型是基于 Bert 实现，较为简单直观。所以，本文首先选择复现该模型。最终本文复现了该模型并使用原论文提供的超参进行实验，EM 值和 F1 分数分别为 86.7%和 89.6%，接近于原文中结果。

（2）模型改进

本文首先使用 Pointer Network 对模型的输出层进行改进。该网络是 R-Net 的输出层，与基线模型使用 TAV 判别预测答案不同，Ptr-Net 使用 seq2seq 的方式生成预测结果，可以较好地解决输出字典长度不固定问题。本文将 Albert Encoder 的输出拆分为问题 Q 和上下文 P 的表征，作为输入衔接到 Ptr-Net 上。随后对该模型的效果进行了实验测试，该方法对模型性能有一定下降。

然后本文尝试了基于多折的 Albert 集成模型。经过参数调优和模型组合实验后发现，该模型显著提高了准确率，获得了本文中所有模型里的最好结果：EM 值 88.4。本文打算用其作为 QA App 的后端模型。考虑应用后期部署在服务器上的负载不能太高，本文设置的集成折数 N=4。

（3）实验分析

在改进模型性能的同时，本文分别对以下三方面进行探究。

第一，本文探究了模型超参对模型性能的影响。在使用原论文提供的超参进行实验后，本文进一步探究了模型超参对实验结果的影响。本文分别调整最大序列长度、初始学习率、批大小以及有答案阈值进行实验。发现最大序列长度、初始学习率、有答案阈值对实验结果影响巨大，而批大小对实验结果影响较小。其中最大序列长度越大越好，初始学习率设为 1e-5 效果最佳，有答案阈值则需要进行二分测试才能确定。

第二，本文探究了什么样的改进对模型效果提升是有益的。一方面，对于 Bert 系列预训练语言模型，比较容易上手的提升方法主要集中在精细调参（如 RoBerta[20]）

和模型集成上。另一方面，由于基线模型准确率已经很高（将近 90%），其输出词嵌入中包含的语义信息已经非常丰富，使用更老的准确率较低模型中的结构对 Bert 系模型进行优化效果都不太理想。此外，使用额外数据进行预训练可能会使模型效果有一定提升（+1%左右）。

（4）开发英文机器阅读理解系统

最后，本文还实现了英文机器阅读理解系统 QA App。该系统使用 C/S 架构实现，缓解了客户端运行深度学习模型的压力。客户端界面使用 PyQt5 开发，支持从本地文件导入数据或者用户手动输入数据。客户端从服务端 API 获得预测结果后，还可以显示系统的实时准确率，便于用户更好地测试和评估模型。

1.5 本文组织结构

本文共分为七章，组织结构如下：

第一章介绍了机器阅读理解领域的研究背景。对已提出的 MRC 解决方法及其发展做了总结，在对 2018 年提出的 Bert 模型分析的基础上，阐述了基于此模型做出的改进。

第二章对 MRC 任务分类及其评估标准做了概述。分别从该任务的种类划分方式，四类 MRC 任务的定义、特点及相互之间的横向对比，以及片段抽取式 MRC 任务评估指标等几个方面进行了介绍。

第三章复现了一种基于 Albert 的基线 MRC 模型。本章先简单介绍了 Albert 模型结构，及其与它的前身 Bert 模型间的区别，其次具体介绍了基线模型的设计细节，以及实验数据选取及预处理流程。随后进行了实验，最后对实验结果进行分析并对超参进行调优以探究不同超参对模型性能的影响。

第四章提出一种基于 Albert+Pointer-Network 的 MRC 模型，使用 Ptr-Net 替代原基线模型的输出层和 TAV 判别过程，效果上较基线模型有所下降。本章先介绍了类似地使用 Pointer-Network 作输出层的 R-Net 模型，然后提出参照 R-Net 替换基线模型的输出层并进行实验。最后深入探究了实验效果下降的原因，提出了改进思路。

第五章提出一种基于 N 折 Albert 集成的 MRC 模型，使用多个 Albert 模型集成替代原先的单个模型。本章先介绍了集成模型的设计思路，然后进行了参数调优实验，得到了较好的实验效果。最后分析了产生该结果的原因。

第六章介绍了英文机器阅读理解演示系统 QA App。其中详细介绍了软件实现中

的关键技术，如 C/S 架构和 PyQt5 界面类等。

第七章是总结与展望。概括了本文所做的工作和笔者毕设以来的心得，分析了本文设计实现的 MRC 系统存在的不足，并讨论了提供更多语言支持、尝试新模型等后续可能进行的扩展和补充工作。

2 相关理论和技术

2.1 机器阅读理解任务定义及分类

机器阅读理解（MRC）是文本问答系统（QA）的一种典型形式。每个问题都与某个文本相关，从中推断出答案。MRC 的目标是从给定的上下文中提取正确的答案，甚至根据上下文生成更复杂的答案。MRC 有望弥合人类和机器之间理解自然语言的鸿沟。MRC 的形式定义如表 2.1 所示。

表 2.1 机器阅读理解的定义

机器阅读理解	
定义	给定文本 C 和问题 Q，机器阅读理解任务要求模型去学习一个函数 F，由函数预测出问题 Q 的正确答案 A，使得 A=F(C, Q)。

在这一部分中，本文根据 Chen 的博士学位论文[9]，将 MRC 基于答案形式划分为四个任务，分别是：完形填空、多项选择、范围抽取和自由回答。为了更好的理解，表 2.2~表 2.5 给出了一些代表性数据集的例子。在下一部分中，本文用形式化的定义描述了每个任务，并从不同的维度对它们进行了比较。

2.1.1 完形填空

表 2.2 完形填空的定义及示例

完形填空		
定义	给定上下文 C，其中的一个单词或实体 A（A∈C）被去掉，完形填空任务要求模型通过学习函数 F 填入正确的单词或实体 A，使得 A＝F(C−{A})。	
示例（CNN/Daily Mail[15]）	上下文	the ent381 producer allegedly struck by ent212 will not press charges against the “ent153” host, his lawyer said Friday. ent212, who hosted one of the most-watched television shows in the world, was dropped by the ent381 Wednesday after an internal investigation by the ent180 broadcaster found he had subjected producer ent193 “to an unprovoked physical and verbal attack.”
	问题	Producer ____ will not press charges against ent212, his lawyer says.
	答案	ent193

完形填空（Cloze Tests），又称填空测试，是评价学生语言水平的常用考试方法。受此启发，这个任务被用来衡量机器理解自然语言的能力。在完形填空测试中，问题

是通过从文章中删除一些单词或实体而产生的。回答问题时，要求在一个空白处填空。完形填空给阅读增加了障碍，既要求理解语境，又要求使用词汇，对机器阅读理解具有挑战性。

完形填空的特点是：（1）答案 A 是给定语境 C 中的一个词或实体；（2）问题 Q 是通过从给定的上下文 C 中删除一个单词或实体而产生的，因而有 Q=C−A。根据这些特点，完形填空测试的形式化定义及示例如表 2.2 所示。

2.1.2 多项选择

多项选择（Multiple Choice）是另一个受语言能力考试启发而产生的机器阅读理解任务，它要求机器根据提供的上下文在多个候选答案里选出正确的那一个。多项选择题的答案可以不是上下文中的实体和单词，因此答案形式相比完形填空更为灵活，但这项任务需要预先给出包含正确答案的 4 个候选项。

多项选择的特点是给出一个候选答案列表 $A=\{A_1, A_2, \cdots, A_n\}$，作为预测答案的辅助信息。多项选择可形式化如表 2.3 所示。

表 2.3 多项选择的定义及示例

多项选择		
定义	给定上下文 C、问题 Q 和候选答案 $A=\{A_1, A_2, \cdots, A_n\}$，多项选择要求模型通过学习函数 F 从 A 中选出正确答案 A_i（$A_i \in A$），使得 Ai=F(C, Q, A)。	
示例（RACE[17]）	上下文	If you have a cold or flu, you must always deal with used tissues carefully. Do not leave dirty tissues on your desk or on the floor. Someone else must pick these up and viruses could be passed on.
	问题	Dealing with used tissues properly is important because ____.
	选项	A. it helps keep your classroom tidy B. people hate picking up dirty tissues C. it prevents the spread of colds and flu D. picking up lots of tissues is hard work
	答案	C

2.1.3 片段抽取

虽然多项选择和完形填空能够在某种程度上衡量模型对自然语言的理解能力，但在现实环境下是没有候选项可供选择的，而且用实体和词语回答问题往往并不充分，还要再补充一些句子才行。因而，人们提出了片段抽取任务（Span Exaction）来克服前两个任务的这些缺点。在已知上下文和问题的条件下，此任务要求机器从相应上下

文中提取一段文本作为答案。

片段抽取的特点是要求答案 A 是给定上下文 C 的连续子序列。其形式化定义及示例如表 2.4 所示。

表 2.4 片段抽取的定义及示例

片段抽取		
定义	给定由 n 个标记组成的上下文 C，即 $C=\{t_1, t_2, \cdots, t_n\}$，以及问题 Q。片段抽取任务需要模型通过学习函数 F，从上下文 C 中提取连续子序列 $A=\{t_i, t_{i+1}, \cdots, t_{i+k}\}$（$1 \leqslant i \leqslant i+k \leqslant n$）作为问题 Q 的正确答案，使得 A=F(C, Q)。	
示例（SQuAD2.0[12]）	上下文	New Zealand (Māori: Aotearoa) is a sovereign island country in the southwestern Pacific Ocean. It has a total land area of 268,000 square kilometers (103,500 sq mi), and a population of 4.9 million. New Zealand's capital city is Wellington, and its most populous city is Auckland.
	问题	What's the largest city?
	答案	Auckland

2.1.4 自由回答

表 2.5 自由回答的定义及示例

自由回答		
定义	给定上下文 C 和问题 Q，自由回答任务的答案 A 不一定是来自原文 C 的子串，也可以有部分词汇来自开放域词表，即 $A \subseteq C$ 或 $A \not\subset C$。这项任务要求模型通过学习函数 F 来预测正确答案 A，使得 A=F(C, Q)。	
示例（MS MARCO[16]）	上下文 1	Rachel Carson’s essay on The Obligation to Endure, is a very convincing argument about the harmful uses of chemical, pesticides, herbicides and fertilizers on the environment.
	上下文 2	Carson believes that as man tries to eliminate unwanted insects and weeds; however he is actually causing more problems by polluting the environment with, for example, DDT and harming living things.
	上下文 3	Carson subtly defers her writing in just the right writing for it to not be subject to an induction run rampant style which grabs the readers interest without biasing the whole article.
	问题	Why did Rachel Carson write an obligation to endure?
	答案	Rachel Carson writes The Obligation to Endure because believes that as man tries to eliminate unwanted insects and weeds; however he is actually causing more problems by polluting the environment.

与完形填空和多项选择相比，片段抽取任务在允许机器给出更灵活的答案方面取得了长足的进步，但这还不够，因为给出限制在一定范围内的答案仍然是不现实的。为了回答这些问题，机器需要对文本的多个部分进行推理并总结证据，这也就是自由回答任务（Free Answering）。在这四项任务中，自由回答无疑是最为复杂的，因为它的答案形式没有限制，更适合实际应用场景。

与其他任务相比，自由回答减少了一些限制，更注重使用自由形式的自然语言来更好地回答问题。其定义及示例如表 2.5 所示。

2.1.5 四种任务间的区别

为了比较这 4 种 MRC 任务的贡献和局限性，本文设置了 5 个维度的评估指标：构造难度、理解能力、灵活性、评估难度和应用价值。在每个维度中，对任务进行比较并根据相对排名对其进行评分。由于有 4 种任务要纳入考虑范围，得 4 分表示相应的任务在该维度上的表现最好，而得 1 分表示表现最差。此外，两个任务的得分在同一维度相同意味着很难判断哪一个在该维度上更好。

各指标的定义及评分依据如下：

（1）构造难度：这个维度度量为任务构造数据集是否容易。越简单，分数就越高。

（2）理解能力：这个维度用于评估一个任务测试模型理解能力的程度。如果一项任务需要模型进行更多的理解和推理，那么分数会更高。

（3）灵活性：答案形式的灵活性可以衡量任务的质量。答案越灵活，灵活性得分越高。

（4）评估难度：评估是检验模型在数据集上表现效果的手段。任务评估的难易程度与任务的自身价值息息相关，太难评估的任务往往没有实用价值。如果一项任务容易评估，那么在这个维度上将得到高分。

（5）应用价值：一个任务与真实世界应用的相似程度。因此，如果一项任务可以很容易地应用到现实世界中，那么这个维度的得分就很高。

如图 2.1 所示，这五个维度的分数因不同的任务而异。更具体地说，构建数据集和评估完形填空测试是最容易的。然而，由于完形填空仅限于原始上下文中的单个词或名称实体，因此完形填空不能很好地测试机器理解能力，也不符合实际应用。多项选择任务为每个问题提供候选答案，这样即使答案不局限于原始上下文，也可以很容易地进行评估。且为这项任务出数据集没什么难度，因为语言考试中的多项选择题就

可以直接拿来使用，不需要额外人工合成数据。然而，候选答案并不现实，与现实条件差异较大。相比之下，片段抽取式 MRC 任务能够很方便地构建和评估数据集，是适中之选。此外，该任务还可以测试机器对文本的理解。所有这些优点都有助于对片段抽取任务进行大量的研究。片段抽取的缺点是，答案必须是上下文的子序列，这一要求与现实条件还有一定差异。自由回答任务在理解性、灵活性、应用性等方面表现出优势，最接近实际应用。然而，每个硬币都有两面。由于答案形式的灵活性，该任务下数据集的构建和模型性能的评价仍然较为困难。

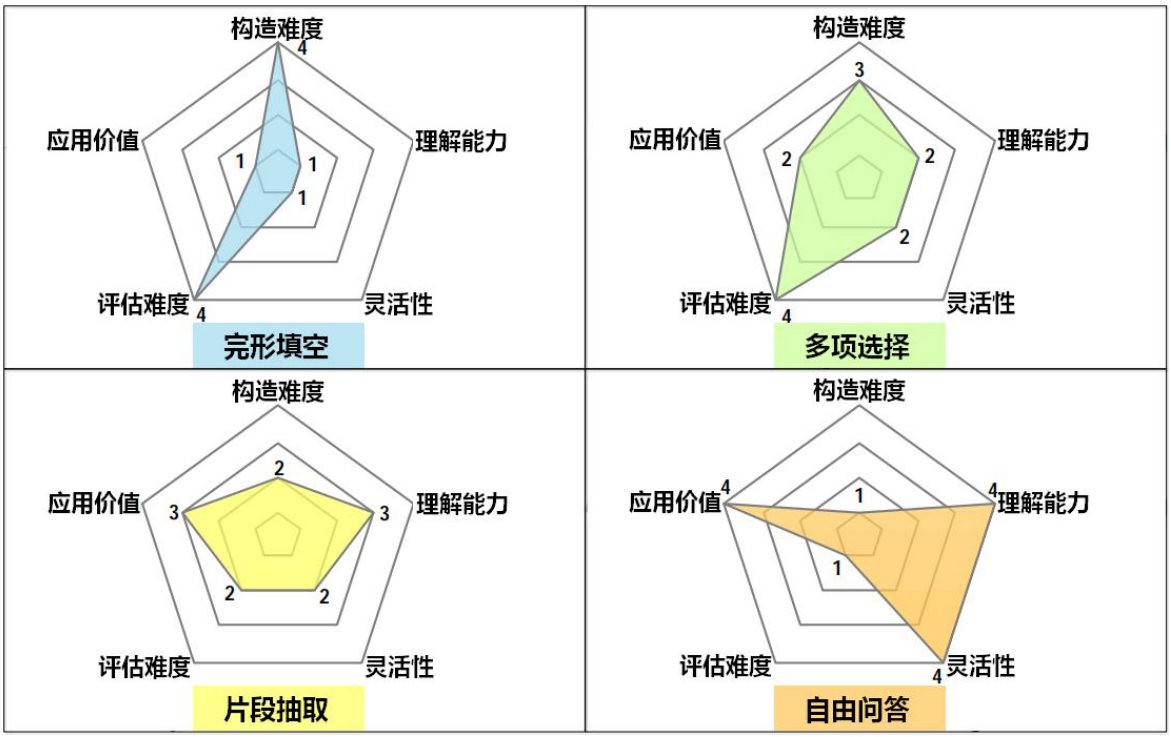

图 2.1 四种 MRC 任务在五个维度上的评分结果[10]

2.2 片段抽取式 MRC 评估标准

常见的片段抽取式 MRC 数据集，如 SQuAD2.0[12]、NewsQA[18]等，往往使用两种不同的指标来评估模型的准确性。这两个指标的计算均忽略标点符号和冠词（即 a、an、the）。有关两个指标的定义及计算方法如下：

（1）完全匹配值（Exact Match，简称 EM）。由于一个问题的标准答案可能有多个，只要预测结果与这多个标准答案中的任意一个相匹配（即在忽略标点和冠词的情况下两个字符串相同），则称该答案完全匹配。整个数据集上的 EM 值是完全匹配的样本数占总样本数的百分比。

（2）F1 分数（F1 Score，简称 F1）。这个指标衡量的是预测结果和标准答案在

token 级别的平均重叠率。计算时将预测出的结果和标准答案均视为一组 token，并计算它们的 F1。对每一个问题取预测结果和所有标准答案最大的 F1，然后对所有问题取平均值，即可得到在整个数据集上的 F1 分数。

3 基于 Albert 的基线 MRC 模型

3.1 Albert 模型介绍

Bert 预训练语言模型由 Jacob 等人[3]于 2018 年提出，模型结构由多个 Transformer 的 Encoder 层构成。其上游任务有：（1）下一句预测（Next Sentence Prediction，简称 NSP），内容是判断两个句子是否连贯；（2）遮蔽语言模型（Masked Language Model，简称 MLM），内容是预测被掩码遮住的单词。二者均为无监督任务。

随着 Bert 论文的发表，该模型很快大幅刷新了多个 NLP 任务的历史纪录，在排行榜上登顶。但是由于模型过大，导致了如内存超限、模型退化等一些问题。

为了解决上述诸多问题，Albert 应运而生。

Albert 由 Google 的蓝振忠等人[1]在 2019 年提出。作为 Bert 的一种改进，Albert 通过层参数共享、替换预训练任务以及嵌入矩阵分解三种手段大幅削减了参数量，同时又保证了模型性能，使得模型用起来更加便捷。

（1） 层参数共享

12 层的 Bert-base 模型有 1.1 亿参数，而 Bert-large 层数上只多一倍，参数量却有前者的 3 倍多。由此可见，Bert 模型参数量随着层数增加而呈指数增长。两种 Bert 模型的各项数据对比如表 3.1。

表 3.1 Bert-large 与 Bert-base 参数数量对比表

模型版本	隐藏单元数量 H	层数 L	注意力头数 A	参数量 P
Bert-base	768	12	12	110M
Bert-large	1024	24	16	340M

为了解决这个问题，Albert 使用了共享层参数这一手段。比如在训练 24 层的 Bert-large 模型时，只初始化第一层的参数，并在运行每一个样本时将该层的参数梯度优化 24 遍，而不是将 24 个独立的参数层每个优化一遍。

Albert 模型支持自定义参数共享的位置，可以直接共享整层的所有参数，也可以选择只共享 Attention 参数或者 FFN 层的参数。论文选择共享了整层的所有参数。与参数量达 110M 的 Bert-base 相比，Albert-base 同样只有 12 层和 768 个隐藏单元，却只有 31M 参数。

论文还研究了共享参数的位置和量对模型精度的影响。发现嵌入大小越大，参数

共享引起的模型精度下降越显著。其中 FFN 层参数共享对精度下降贡献最大，远大于共享 Attention 参数的贡献。

（2）句子顺序预测（SOP）

Bert 中的 NSP 任务是为提高模型对句子间关系判断能力而设计的预训练任务，其一般过程为：（1）正样本由同一文档中两个连续语句组成，负样本则由两篇文档中各抽出的一句话构成；（2）让模型学会判断样本中的两句话语义上是否连续。

然而，RoBerta[20]、XLNet[21]等论文指出，NSP 并不能提高模型在下游任务上的性能。多个下游任务表现都随着 NSP 的取消获得了提升。

因此，Albert 提出了句子顺序预测（Sentence Order Prediction，简称 SOP）任务，用于替换原来的 NSP。其一般过程为：（1）正样本由同一文档中两个连续语句组成，负样本则由正样本中两个语句交换顺序得到；（2）让模型学会判断样本中的两句话语义上是否连续。

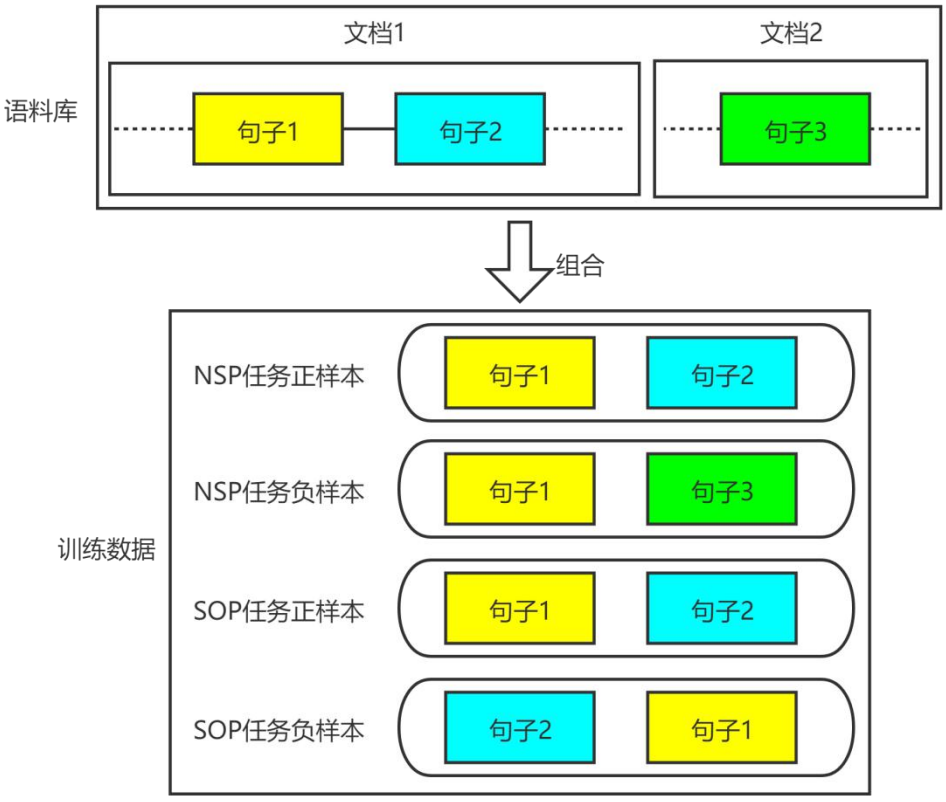

图 3.1 NSP 与 SOP 预训练任务对比

NSP 任务和 SOP 任务内容上的对比见图 3.1。

Albert 认为 NSP 包含了连贯性预测和主题预测，其中主题预测与 MLM 任务有重叠，故模型在未学过连贯性预测时仍能在 NSP 上取得很好的效果。且 NSP 比 MLM

任务似乎更简单，故 NSP 是无效的。

而 SOP 任务能训练模型学习到语句一致性更细微的差异。使用 SOP 后，模型在 SQuAD v1.1、SQuAD v2.0、MNLI 等多个下游任务上的表现得到提升。

表 3.2 使用不同形式附加 loss 的模型在上下游任务上的对照实验结果[1]

	上游任务			下游任务					
附加句间 loss 形式	MLM	NSP	SOP	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg
None	54.9	52.4	53.3	88.6/81.5	78.1/75.3	81.5	89.9	61.7	79.0
NSP	54.5	90.5	52.0	88.4/81.5	77.2/74.6	81.6	**91.1**	62.3	79.2
SOP	54.0	78.9	86.5	**89.3/82.3**	**80.0/77.1**	**82.0**	90.3	**64.0**	**80.1**

由表 3.2 可知，SOP 是比 NSP 更强、更有效的预训练任务，可以使用前者替换后者。表现为：（1）将 SOP loss 用在 NSP 上的表现要比将 NSP loss 用在 SOP 上的表现强得多（78.9>52.0），说明 SOP 在训练内容上包含一部分 NSP；（2）用 SOP loss 预训练出的模型在下游任务上效果几乎全面优于使用 NSP loss 的结果（80.1>79.2）。

注：这里的 None 为对照组，代表 RoBerta 和 XLNet；NSP loss 代表 Bert；SOP loss 代表 Albert。

（3）嵌入参数分解

在 Bert 中，有隐藏单元数量 H=词嵌入大小 E。假设词表长度 30K，word-piece embedding 大小 E=768，隐藏单元数量 H=768。词嵌入需要随着 Encoder 层隐藏单元数量的增加而不断添加维度，这会导致参数量显著增加。

通过用两个小矩阵代替分解前的大嵌入矩阵，Albert 成功将词表嵌入大小与隐藏单元数量解耦，增加隐藏单元数量不再会导致词表嵌入参数量显著增加。

例如，将 one-hot 编码的输入数据先映射至嵌入空间 E=100，再映射至 H=768 的隐层空间。如图 2.10，可将参数量从 30K×768 减少到(30K+768)×100，约减少到原来的 1/7。示意图见图 3.2。

经过上述 3 个改变后，Albert 的参数量减小为 Bert-large 模型的 1/18，训练速度提高 70%，在 GLUE、RACE、SQuAD 等数据集上均得到了当时领先的结果，是 NLP 预训练语言模型领域的又一大进步。[28]

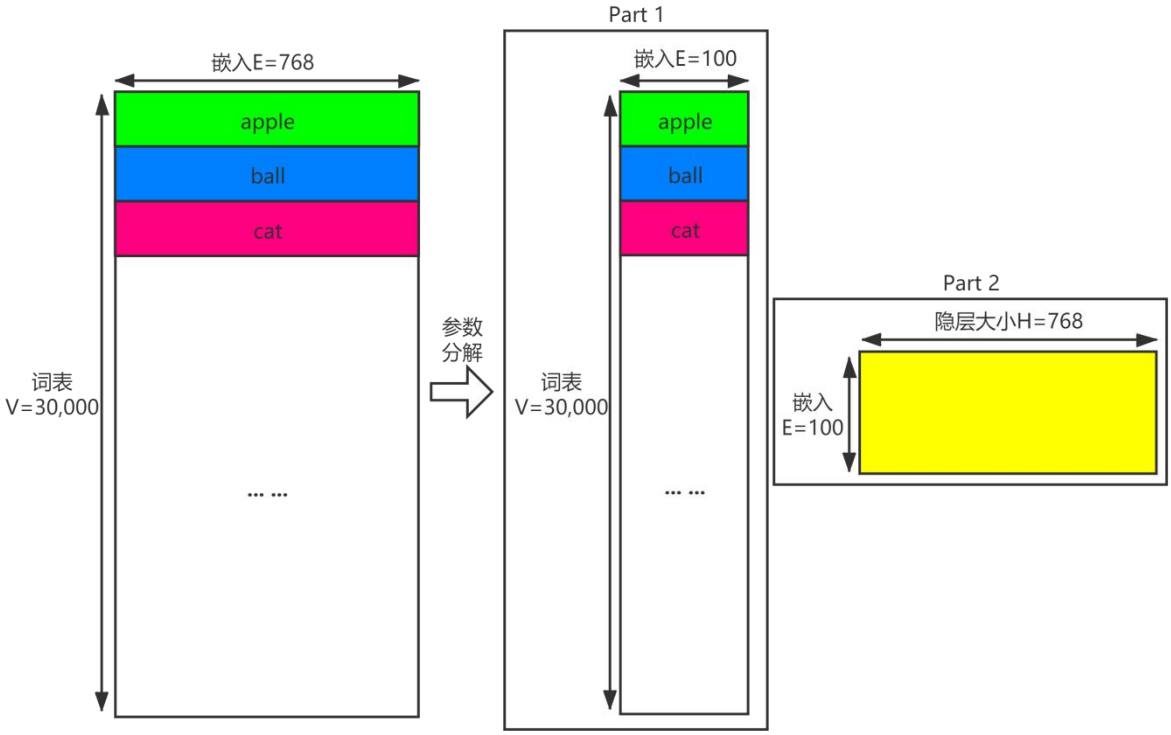

图 3.2 嵌入矩阵参数分解前后对比

3.2 模型设计

3.2.1 TAV 判别

基于阈值的可回答性判别（Threshold-based Answerable Verification，简称 TAV）由上海交通大学的张倬胜等人[24]提出，这是一种根据模型预测出的答案开始和结束 logits 来判断问题是否可回答的启发式策略。

给定模型输出层（Output Layer）得到的开始和结束 logits s 和 e，可计算出有答案得分 $score_{has}$ 和无答案得分 $score_{null}$，并将 $score_{null}$ 和 $score_{has}$ 二者间差值作为最终的无答案分数 $score_{diff}$。过程如公式（3.1）~（3.3）所示。

$$score_{has} = \max(s_k + e_l), 1 < k \le l \le n \tag{3.1}$$

$$score_{null} = s_1 + e_1 \tag{3.2}$$

$$score_{diff} = score_{null} - score_{has} \tag{3.3}$$

其中可回答阈值δ为超参数，将通过在验证集上测试得出。内容是通过区间逼近法依次测试δ位于[-6,0]范围内各点时模型在测试集上的准确率，取 0.1 为δ最小精度，取结果准确率最高时的δ值作为模型部署时使用的参数。一个典型的阈值实验结果如第 5 章图 5.2 所示。

如果最终得分 $score_{diff}$ 不高于阈值δ，则模型将得出有答案得分 $score_{has}$ 的上下文片段 context[k:l+1]作为预测结果，否则返回为空字符串“”。如公式（3.4）。

$$answer = \begin{cases} context[k:l+1], score_{diff} \le \delta, \\ "", score_{diff} > \delta. \end{cases} \tag{3.4}$$

3.2.2 基于 Albert 的基线模型

本文实现的 Albert 模型代码使用 PyTorch 框架，基于 HuggingFace 的 transformers 库（库项目链接：https://github.com/huggingface/transformers）开发，预训练权重使用的是 Albert-xxlarge-v2 版本的权重（权重地址：https://huggingface.co/albert-xxlarge-v2）。

Albert 基线模型的网络结构图如图 3.3，网络参数名称及数量表如表 3.3。该模型的可训练参数共有约 206M。开发该基线模型的目的在于复现 Albert 论文[1]中提出的模型并将其应用到 MRC 领域，同时为后面几章开发更复杂的 MRC 模型提供对照和参考依据。

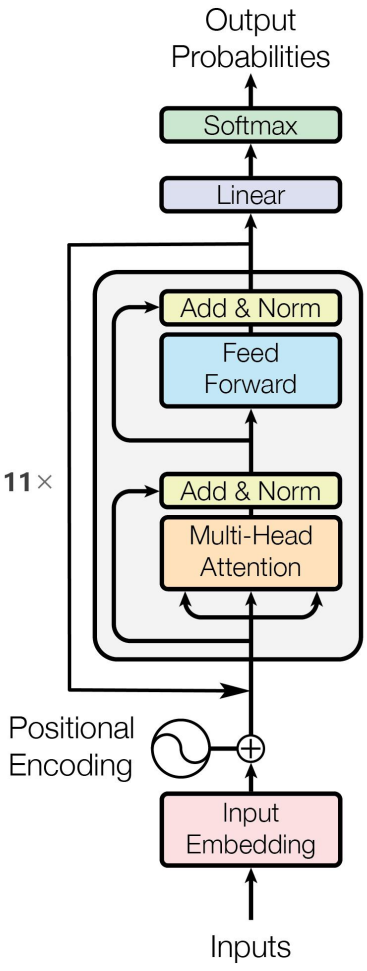

图 3.3 Albert 基线模型网络结构图[2]

表 3.3 Albert 基线模型网络参数表

隶属模块	参数名称	参数形状	参数量
Input Embedding	albert.embeddings.word_embeddings.weight	(30000,128)	3840000
	albert.embeddings.position_embeddings.weight	(512,128)	65536
	albert.embeddings.token_type_embeddings.weight	(2,128)	256
	albert.embeddings.LayerNorm.weight	(128)	128
	albert.embeddings.LayerNorm.bias	(128)	128
	albert.encoder.embedding_hidden_mapping_in.weight	(4096,128)	524288
	albert.encoder.embedding_hidden_mapping_in.bias	(4096)	4096
LayerNorm (In Loop)	albert.encoder.albert_layers.0.full_layer_layer_norm.weight	(4096)	4096
	albert.encoder.albert_layers.0.full_layer_layer_norm.bias	(4096)	4096
Multi-Head Attention (In Loop)	albert.encoder.albert_layers.0.attention.query.weight	(4096,4096)	16777216
	albert.encoder.albert_layers.0.attention.query.bias	(4096)	4096
	albert.encoder.albert_layers.0.attention.key.weight	(4096,4096)	16777216
	albert.encoder.albert_layers.0.attention.key.bias	(4096)	4096
	albert.encoder.albert_layers.0.attention.value.weight	(4096,4096)	16777216
	albert.encoder.albert_layers.0.attention.value.bias	(4096)	4096
	albert.encoder.albert_layers.0.attention.dense.weight	(4096,4096)	16777216
	albert.encoder.albert_layers.0.attention.dense.bias	(4096)	4096
LayerNorm (In Loop)	albert.encoder.albert_layers.0.attention.LayerNorm.weight	(4096)	4096
	albert.encoder.albert_layers.0.attention.LayerNorm.bias	(4096)	4096
Feed Forward (In Loop)	albert.encoder.albert_layers.0.ffn.weight	(16384,4096)	67108864
	albert.encoder.albert_layers.0.ffn.bias	(16384)	16384
	albert.encoder.albert_layers.0.ffn_output.weight	(4096,16384)	67108864
	albert.encoder.albert_layers.0.ffn_output.bias	(4096)	4096
Linear	qa_outputs.weight	(2,4096)	8192
	qa_outputs.bias	(2)	2
-	总计	-	205822466

在本文设计实现的所有模型中，除特殊说明外，TAV 都默认被用作可回答性决策的最后一步，在后序表格中将表示为“（+TAV）”。

使用 TAV 判别的 Albert 基线模型的程序流程图如图 3.4 所示。

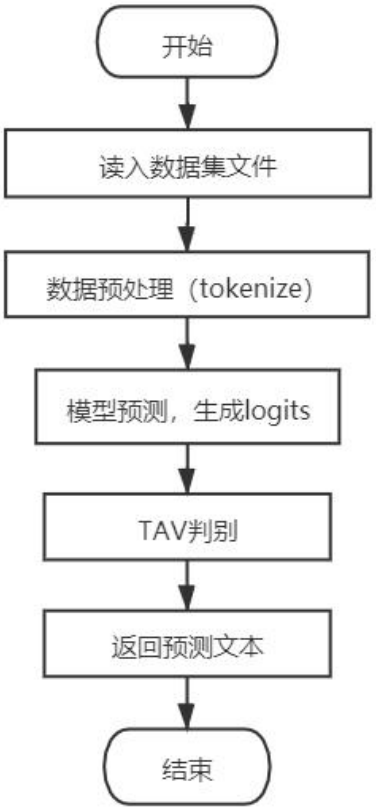

图 3.4 Albert 基线模型预测阶段程序流程图

3.3 数据选取及预处理

3.3.1 数据选取

由于本次毕业设计的目的是设计和实现片段抽取式的 MRC，故数据集上也应选择配套的抽取式问答数据集以便模型的训练和评估。

目前常见的抽取式数据集有 SQuAD1.1、SQuAD2.0、NewsQA、TriviaQA 等。SQuAD2.0 版与 1.1 版相比，其中有一半的问题是不可回答问题，而 1.1 版中是没有这种问题类型的。另一方面，对如何处理该种“不可回答问题”的讨论，正是当前学界的热点。为了保证研究的即时性和挑战性，本文首选 SQuAD2.0 作为本次毕设的主数据集。

为了分析方法的适用性，本文还选择了 NewsQA 作为数据补充。NewsQA 也有占比 1/3 左右的“不可回答问题”，而且 NewsQA 的上下文篇幅更长，能更好地评测

模型的长文本连接和理解能力。

综上所述，本次毕设训练集使用 SQuAD2.0-train 和 NewsQA-train，验证集统一使用 SQuAD2.0-dev。

3.3.2 数据构成

```
"""
dataset architecture:
{
    "data": [
        {
            "title": "Super_Bowl_50",
            "paragraphs": [
                {
                    "context": " numerals 50.",
                    "qas": [
                        {
                            "answers": [
                                {
                                    "answer_start": 177,
                                    "text": "Denver Broncos",
                                },
                            ],
                            "question": "Which NFL team represented the AFC at Super Bowl 50?",
                            "id": "56be4db0acb8001400a502ec"
                            "is_impossible": False
                        }
                    ]
                }
            ]
        }
    ],
    "version": "2.0"
}
"""
```

图 3.5 SQuAD2.0 数据集文件结构

数据构成部分以 SQuAD2.0 数据集为例进行说明。该数据集文件为 json 格式，包含一个训练集文件“train-v2.0.json”和一个验证集文件“dev-v2.0.json”。其中训练集文件结构如图 3.5 所示。

“data”字段下是一个文章构成的列表，每篇文章在“paragraphs”字段下又可拆为多个片段，每个片段的上下文文本在“context”字段中给出，而“qas”字段中给出的是对应该文本的问题列表。每个问题结构包括问题文本“question”、问题答案“answers”、问题编号“id”以及问题是否可回答的标志符“is_impossible”。问题答案可能有多个，内容上包括答案的开始位置“answer_start”以及答案文本“text”。

3.3.3 数据预处理

笔者本次毕设使用 Transformers.AlbertTokenizer 来预处理数据。首先将数据集文件处理成为 SquadExample 对象，然后使用 AlbertTokenizer 对对象中的文本数据进行编码、连接、填充/截断并转为张量，返回值包括 input_ids、attention_mask、token_type_ids 三个。由于后两个返回值的生成较为容易，这里主要对 input_ids 的生

成进行讲解。

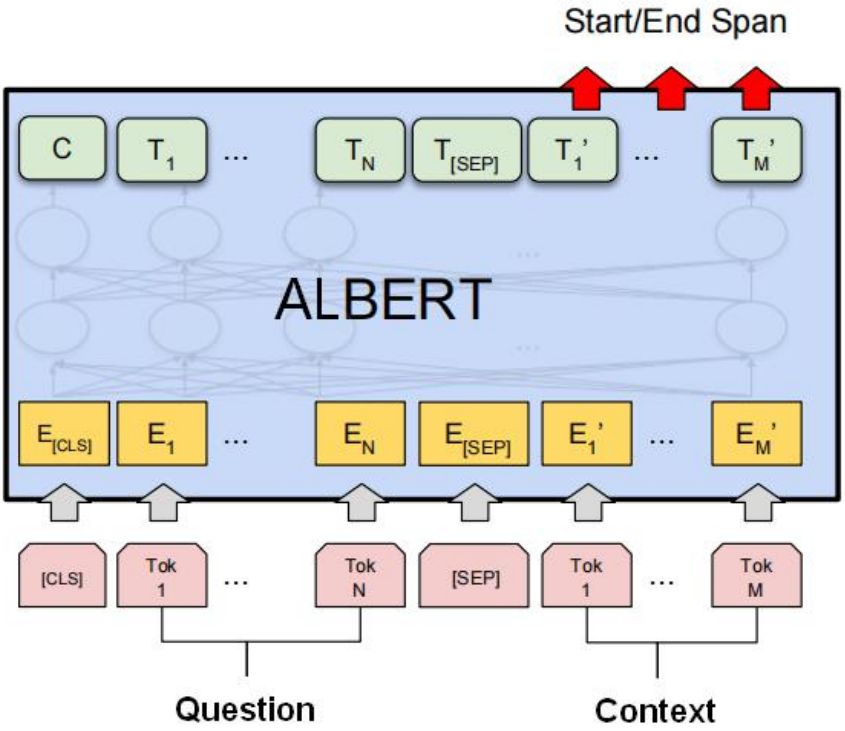

图 3.6 Albert 使用[CLS]和[SEP]连接 Question 和 Context 作为输入[3]

SquadExample 本质上只是把数据集文件中的信息重新用对象存储了一遍，内容包括 context_text、question_text、answer_text、answers 等。为了便于后期模型输入，需将问题 Q 和上下文 C 先使用 sentencepiece 编码为 token 序列，然后使用“[CLS]”和“[SEP]”连接为“[CLS] Q [SEP] C [SEP]”。当该 token 串长度超出（或达不到）预设的超参数 max_seq_length 时，会将 C 截断为若干段并与“[CLS]”、Q、“[SEP]”重新连接（或使用[PAD]填充），例如截断为 3 串并填充作为 input_ids：“[CLS] Q [SEP] C1 [SEP], [CLS] Q [SEP] C2 [SEP], [CLS] Q [SEP] C3 [SEP] [PAD]...[PAD]”。结果保证每串长度均为 max_seq_length。示意图见图 3.6。

AlbertTokenizer 会将上述生成 input_ids、attention_mask、token_type_ids 三者的工作一并自动完成，之后便是将 token 数据转为张量，存储，并输入到模型了。

3.4 实验

3.4.1 实验环境

本文实验所基于的本地和服务器环境配置如表 3.4 所示。

3.4.2 实验结果与分析

本次实验仅在 SQuAD2.0 数据集上完成，不引入额外数据（即 NewsQA）。其中训练数据使用 SQuAD2.0-train，测试数据使用 SQuAD2.0-dev。

表 3.4 实验的软硬件环境

	本地环境	服务器环境
CPU	Intel Core i7-8750H CPU @ 2.20GHz	Intel Xeon Gold 6240 CPU @ 2.60GHz
GPU	GTX 1060 Mobile 6GB ×1	Titan RTX 24G ×8
内存	32GB	512GB
硬盘	512G SSD	1T HDD
操作系统	Windows 10	CentOS 7
开发语言	Python 3.6.8	Python 3.6.8
开发框架	PyTorch 1.4.0	PyTorch 1.4.0
开发工具	Anaconda3 2021.05、PyCharm 2020.2.2 Professional、Xshell 6	Anaconda3 2021.05

在未特别说明的情况下，本章及后序几章所有实验均默认使用 AdamWeightDecay 优化器，学习率调整策略均为 transformers.get_linear_schedule_with_warmup，TAV 判别中的有答案阈值δ均默认设为 0。

为了记录方便，本文将 Albert 原论文[1]实验中涉及的 Albert 单模型编号记作 SM（Single Model），将其中的 Albert 集成模型编号记作 EM（Ensemble Model）。

在不同超参数下的 Albert 基线模型对比实验结果见表 3.5。

表 3.5 Albert 基线模型超参数配置对比实验结果

	模型编号	M3.1	M3.1	M3.2	M3.2	M3.3	M3.3	SM[1]
SQuAD2.0-train 上训练	最大序列长度	384	384	384	384	384	384	**512**
	初始学习率	3e-5	3e-5	1e-5	**1e-5**	5e-6	5e-6	-
	批大小	4	4	4	4	4	4	-
	训练轮数	2	2	2	2	2	2	-
SQuAD2.0-dev 上测试	有答案阈值δ	0	-5(最优)	0	**-3.9**(最优)	0	-4(最优)	-
	最大 EM 值（%）	85.34	86.02	86.14	**86.72**	85.98	86.49	**87.4**

在对模型的初始学习率和有答案阈值δ这两个超参数分别进行调参对比实验后，可得出模型的学习率设置为 1e-5、有答案阈值设置为-3.9 时，基线模型的效果最好，此时模型 M3.2 可在 SQuAD2.0-dev 上达到 86.72%的 EM 值。学习率太高或者太低都不利于模型的表达。

值得一提的是，尽管如此，上述本地实验得出的最强基线模型 M3.2 效果仍达不到原论文给出的 Albert 单模型准确率 87.4%，不过原论文并未给全训练模型所使用的参数。但有理由推测，准确率上的偏差应该和笔者最大序列长度 max_seq_length 和批大小 batch_size 因算力限制而设得较小有关。

3.5 本章小结

本章首先对 Albert 模型及其网络结构进行了概述，然后基于 Albert 论文复现了 MRC 基线模型。最后探究了超参的不同设置对模型性能的影响，发现初始学习率和有答案阈值的变化均会影响模型性能，同时反思可能是最大序列长度设得较小而导致复现模型的准确率与原论文中结果仍有一些差距。

4 基于 Albert+Ptr-Net 的 MRC 模型

4.1 R-Net 模型介绍

斯坦福问答数据集 SQuAD v1.1 是由 Rajpurkar 等人[11]于 2016 年提出的一个片段抽取类型英文问答数据集，包含大约 10 万个样例，每个样例均由一个三元组构成：文章 P、相应问题 Q 和对应答案 A。该数据集发布后，学术界很快掀起了一股对抽取式阅读理解模型的探索热潮。由微软亚洲研究院提出的 R-Net[25]是便是首个在 SQuAD 数据集某些评测指标上接近人类水平的深度学习模型。

R-Net 使用了 Matching-LSTM[26]和 Pointer Network[27]（也记作 Ptr-Net）的思想，在结构上包括 4 个部分（层），即 Question and Passage Encoder、Gated Attention-based RNN、Self-Matching Attention 以及 Output Layer。模型结构总览如图 4.1。

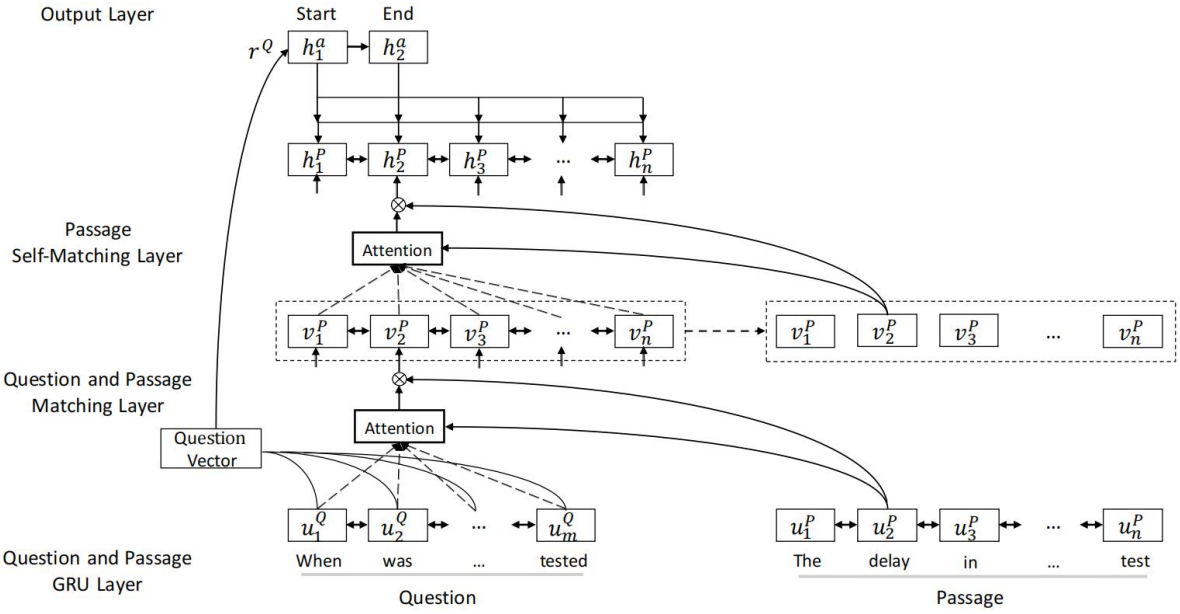

图 4.1 R-Net 模型结构总览[25]

（1）Question and Passage Encoder

第一层是编码层，也就是将输入的字词转化为词向量。R-Net 的编码方式如公式（4.1）所示，其中等式右边的 glove_emb 是把问题 Q 和文章 P 使用 Glove 词向量编码后得到的 $\{e_t^P\}_{t=1}^n$ 、$\{e_t^Q\}_{t=1}^m$，char_emb 是使用字符集卷积神经网络 Char-CNN 得出的 $\{c_t^P\}_{t=1}^n$，$\{c_t^Q\}_{t=1}^m$。两种嵌入连接到一起后再把它们的联合表征依次通过 BiRNN。

$$input = glove\_emb + char\_emb \tag{4.1}$$

$$u_t^P = BiRNN_P(u_{t-1}^P, [e_t^P, c_t^P]) \tag{4.2}$$

$$u_t^Q = BiRNN_Q(u_{t-1}^Q, [e_t^Q, c_t^Q]) \tag{4.3}$$

（2）Gated Attention-based RNN

问题 Q 关于文章 P 的语义理解v_t^P，由第二层网络对第一层网络输出的u_t^Q和u_t^P进行 attention 加权融合得出。即有：

$$v_t^P = RNN(v_{t-1}^P, c_t) \tag{4.4}$$

$$c_t = att(u^Q, [u_t^P, v_{t-1}^P]) \tag{4.5}$$

其中公式（4.5）表示对问题 Q 的注意力加权。其中c_t的注意力权重取决于问题 Q 下前一单词的理解v_{t-1}^P，以及文章 P 中单词的表征u_t^P。Attention-pooling 的过程如下：

$$s_j^t = v^T \tanh(W_u^Q u_j^Q + W_u^P u_t^Q + W_v^P v_{t-1}^P) \tag{4.6}$$

$$a_i^t = \frac{\exp(s_i^t)}{\sum_{j=1}^{m} \exp(s_j^t)} \tag{4.7}$$

$$c_t = \sum_{i=1}^{m} a_i^t u_i^Q \tag{4.8}$$

此外，该层网络还有 2 个改进，一是将c_t、u_t^P连接成$[c_t, u_t^P]$通过 RNN，即：

$$v_t^P = RNN(v_{t-1}^P, [c_t, u_t^P]) \tag{4.9}$$

二是将门控机制应用在$[c_t, u_t^P]$上，便于提取关于文章的问题信息，即：

$$[c_t, u_t^P]^* = sigmoid(W_g[c_t, u_t^P]) \cdot [c_t, u_t^P] \tag{4.10}$$

（3）Self-Matching Attention

该层充分运用了自注意力机制[2]，大幅提升了模型性能，且方便实现。如公式（4.11）和（4.12）所示，当前单词下的篇章语义c_t便是由 attention-pooling 得到的：

$$h_t^P = BiRNN(h_{t-1}^P, [c_t, v_t^P]) \tag{4.11}$$

$$c_t = att(v^P, v_t^P) \tag{4.12}$$

（4）Output Layer

作为模型输出层，Output Layer 受到 Ptr-Net 的启发，使用 seq2seq 方式分 2 次返

回文章中答案的起始位置分布 p^1 和终止位置分布 p^2。

首先，该层使用 attention-pooling 对问题表征 u_t^Q 加权，得到 r^Q，并将其作为整个 seq2seq 过程的初始状态。随后，如公式（4.16）所示，模型算出答案的开始位置分布 p^1，再输入 p^1 对全文表征的加权结果，便可得出答案的结束位置分布 p^2。

$$r^Q = att(u^Q, v_r^Q) \tag{4.13}$$

和 Self-Matching Attention 层相同，该层中 c_t 仍是对文章表征 h^P 注意力加权的结果：

$$h_t^a = RNN(h_{t-1}^a, c_t) \tag{4.14}$$

$$c_t = att(h^P, h_{t-1}^a) \tag{4.15}$$

分别令公式（4.14）和（4.15）中的 t=1、t=2，可让 GRU 和 attention-pooling 的过程循环两次，从而得到两个 logits 分布，最终通过取最大维获得答案起止位置：[29]

$$p^t = \arg\max\{a_1^t, ..., a_n^t\} \tag{4.16}$$

4.2 模型设计

受到 R-Net 的启发，本文对 Albert 模型进行了改造，将第 3 章表 3.3 中 Linear 层（即 qa_outputs 层）之后的内容换成 R-Net 中的 Output Layer（即 Pointer Network），使用 seq2seq 的方式生成答案的开始位置预测 p^1 和终止位置预测 p^2。

将原先 Albert 中段最后一个 Encoder 层输出的形状为(max_seq_length, hidden_dims)的嵌入矩阵 albert_output 用下标从小到大第一个出现的“[SEP]”切割为两部分，第一部分为问题 Q 的嵌入表征 u^Q，第二部分为上下文 P 的嵌入表征 h^P。为便于统一处理，在各维度相对原嵌入 albert_output 位置不变的情况下，将 u^Q 第一维长度截断或填充成 max_query_length，h^P 第一维长度填充成 max_seq_length。

然后对 u^Q 使用加法式 attention-pooling，可得到 Output Layer 中 GRU 的初始状态 h_0^a，如公式（4.17）~（4.19）所示。注：（1）t=1 或 2；（2）i 和 j 均表示嵌入矩阵第一维的 token 下标；（3）v、W、b 均为可训练参数。

$$s_j = v^T tanh(W_u^Q u_j^Q) + b \tag{4.17}$$

$$a_i = \frac{exp(s_i)}{\sum_{j=1}^{m} exp(s_i)} \tag{4.18}$$

$$h_0^a = r^Q = \sum_{i=1}^{m} a_i u_i^Q \tag{4.19}$$

最后，依次对 h^P 和 h_0^a、h^P 和 h_1^a 使用 1 次加法式 attention-pooling，即可得到位置预测结果 p^1 和 p^2。公式如（4.20）~（4.24）所示。

$$s_j^t = v^T tanh(W_h^P h_j^P + W_h^a h_{t-1}^a) \tag{4.20}$$

$$a_i^t = \frac{exp(s_i^t)}{\sum_{j=1}^{n} exp(s_j^t)} \tag{4.21}$$

$$p^t = argmax(a_1^t, ..., a_n^t) \tag{4.22}$$

$$c_t = \sum_{i=1}^{n} a_i^t h_i^P \tag{4.23}$$

$$h_t^a = GRU(h_{t-1}^a, c_t) \tag{4.24}$$

基于 Albert+Pointer-Network 的 MRC 模型程序流程图如图 4.2 所示。

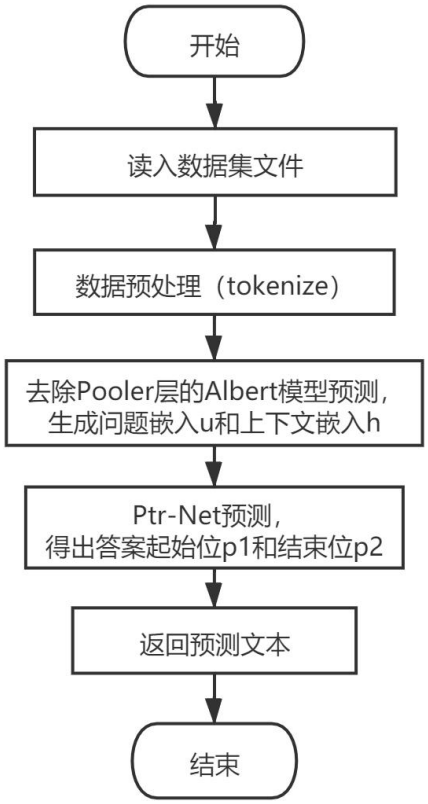

图 4.2 基于 Albert+Pointer-Network 的 MRC 模型预测阶段程序流程图

4.3 实验结果

本次实验在服务器上进行。

实验结果见表 4.1。

表 4.1 基于 Albert+Pointer-Network 的 MRC 模型超参数配置对比实验结果

	模型编号	M4.1	M4.2	M4.3	M3.2(基线)	SM[1]
SQuAD2.0-train 上训练	最大序列长度	384	384	384	384	512
	初始学习率	3e-5	**1e-5**	5e-6	**1e-5**	-
	批大小	2	2	2	4	-
	训练轮数	2	2	2	2	-
SQuAD2.0-dev 上测试	最大 EM 值（%）	69.3	**71.5**	70.4	**86.7**	87.4

由表可得，本章实验实现的 3 个模型中 M4.2 效果最好，在验证集上准确率为 71.5%，但与第 3 章的基线 M3.2 有 15%左右的差距。这个差距属实过大，本文将在下一节中分析可能造成这一准确率滑坡的原因。

4.4 模型分析与可行解

用 Ptr-Net 替换 Albert 基线二阶段原有的 Linear 层和 TAV 判别导致准确率下降了大约 15%，主要原因有：

（1）改进后的 Output Layer 对不可回答问题判断能力较差。改进后的输出层本质上是 Enc-Dec Attention 和 GRU 的结合，这种类型的网络本身比 Bert 系列理解能力就差，对不可回答问题的判断准确度也注定不如改进前的 Linear 层及 TAV 判别。

（2）Albert Encoder 层对 Pointer Network 的输入 u^Q 对结果预测有干扰。Pointer Network 在 R-Net 中接收的输入 u^Q 中不含上下文 P 的信息，而改进后 Albert 传入的 u^Q 中却兼有 P 和 Q 的信息，这不可避免地导致 u^Q 丢失了一部分自带的 Q 信息，从而对末端 seq2seq 预测造成了一些干扰。

由于 R-Net 模型提出较早，使用的 Attention 机制为 Enc-Dec Attention，且当时 SQuAD2.0 还未诞生，因而在存在“不可回答问题”的问答数据集上表现都不太好。作为解决方案，应使用一些近几年的新思路对模型进行改进，比如自注意力机制（Self Attention）。

4.5 本章小结

本章首先介绍了 R-Net 模型及其对 Pointer Network 的使用思路，接着设计并实现了将 Pointer-Network 应用于 Albert 末端的 MRC 模型。然后，在服务器上对设计的模型进行了训练和验证，发现与改进前相比模型的准确率不升反降。最后，进一步分析了可能导致这种现象的原因，并提出了可行的解决方案。

5 基于 N 折 Albert 的 MRC 模型

5.1 模型设计

N 折模型（N≥2）与普通的单个模型（即 1 折）的区别主要表现在：

（1）训练阶段：N 折模型训练数据只有单折模型的(N-1)/N；

（2）预测阶段：N 个模型计算出的 logits 需要进行集成后才能使用 TAV 判别得出预测答案。

下面分别从这两点出发，说明 N 折集成的过程。

5.1.1 N 折模型的训练数据构成

N 折模型的训练数据获得方法为先将原始训练集数据均匀切割为 N 份，再将第 k 份数据去除，剩下（N-1）份数据即为第 k 折模型所用的训练数据。

如此一来，N 折模型中的每一折使用的训练数据均有一大部分是相同的，但又不完全一样，这就使得这些训练出的模型在权重上略有差异但准确率基本相同。将它们的预测结果进行融合可以得到非常好的泛化效果，往往比单个模型有 2%~3%的准确率提升。

5.1.2 N 折模型的 logits 集成

logits 数据本质上是 SquadResult 对象，内部包含的数据元素有 unique_id、start_logits、end_logits，其中后两者是 logits 集成所需要操作的对象。start_logits 和 end_logits 分别表示对应长度为 Max_Seq_Length 的 token 序列上每个 token 为预测答案起始位置和终止位置的分布，因而这两个张量的 shape 均为[Max_Seq_Length]。

集成前后的 start_logits s、end_logits e 的关系表达式如公式（5.1）和（5.2）所示。（即集成 logits 是多个单模型 logits 的均值）

$$s = \frac{1}{N}\sum_{i=1}^{N} s_i \tag{5.1}$$

$$e = \frac{1}{N}\sum_{i=1}^{N} e_i \tag{5.2}$$

N 折 Albert 集成模型在预测阶段的程序流程图如图 5.1 所示。

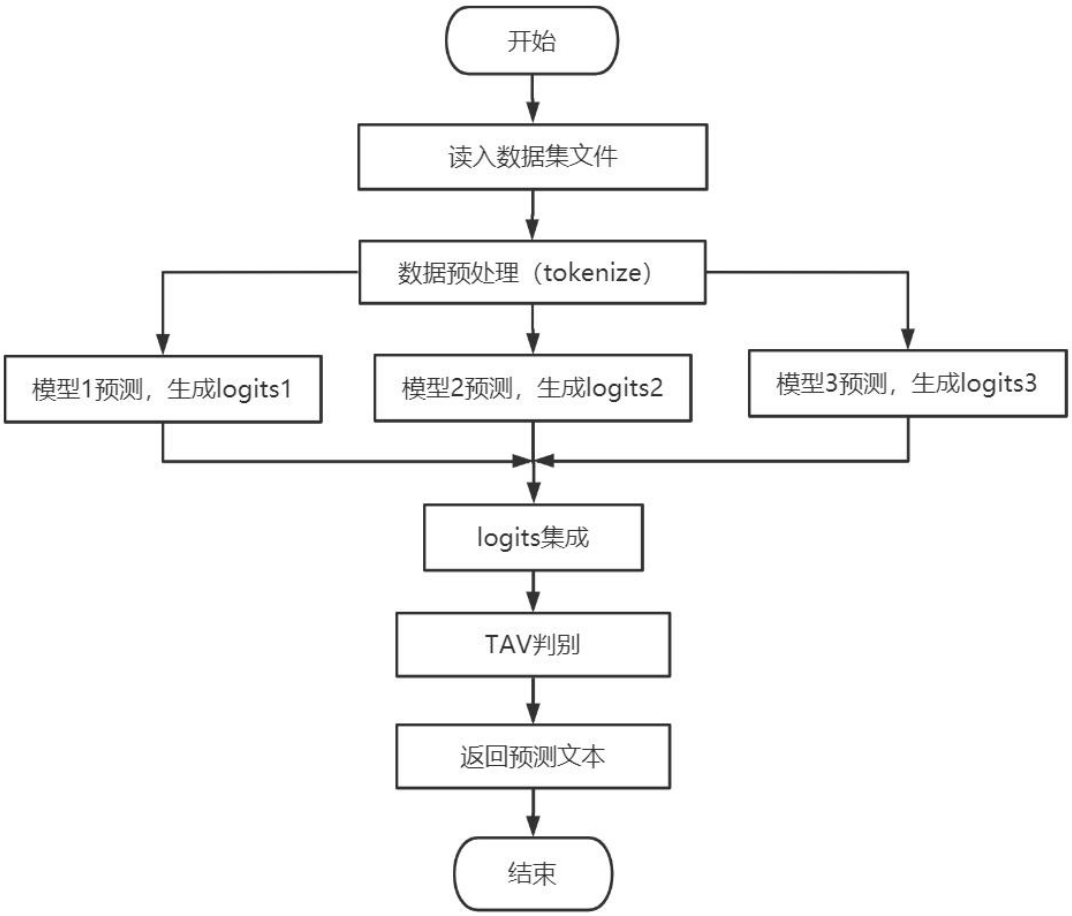

图 5.1 N 折 Albert 集成模型预测阶段程序流程图

5.2 实验结果与分析

本次实验在服务器上进行，实验环境与第三章相同。数据选取方面，在第 3 章 SQuAD2.0 的基础上增加了 NewsQA 数据集作为额外数据。考虑集成模型折数太多会导致后期在服务端运行时负载和响应延迟太高，折数太少又会使训练数据利用率和单个模型准确率下降，这里取折数 N=4 进行模型集成实验。

本次实验内容包括：

（1）Albert-xxlarge-v2 模型在 NewsQA-train 上的训练、在 NewsQA-dev 上的测试和在 SQuAD2.0-dev 上的测试；

（2）Albert-xxlarge-v2 模型在 SQuAD2.0-train 上的 4 折训练和在 SQuAD2.0-dev 上的 4 次测试。

对于任务（1），实验参数配置及结果见表 5.1。

对于任务（2），实验参数配置及结果见表 5.2。

表 5.1 Albert-NewsQA 实验超参数配置及实验结果（模型 M5.1）

	模型编号	M5.1
在 NewsQA-train 上训练	最大序列长度	384
	初始学习率	1e-5
	批大小	18
	训练轮数	3
在 NewsQA-dev 上测试	有答案阈值δ	0
	最大 EM 值（%）	58.91
在 SQuAD2.0-dev 上测试	有答案阈值δ	0
	最大 EM 值（%）	63.18

表 5.2 4 折 Albert 实验超参数配置及实验结果（模型 M5.2~M5.5）

	模型编号	M5.2	M5.3	M5.4	M5.5	EM[1]
在 SQuAD2.0-train 上训练	折数编号	1	2	**3**	4	-
	最大序列长度	384	384	384	384	**512**
	初始学习率	5e-6	1e-5	**1e-5**	1e-5	-
	批大小	12	12	**16**	20	-
	训练轮数	3	2	2	2	-
在 SQuAD2.0-dev 上测试	有答案阈值δ	0	0	0	0	-
	最大 EM 值（%）	80.39	86.09	**87.06**	86.35	**88.9**

显然，模型 M5.1 和模型 M5.2 的最高准确率与模型 M5.3~M5.5 相差较多，故只采用模型 M5.3~M5.5 进行集成。

如图 5.2，经过 TAV 测试可知，当有答案阈值δ取-3.1 时，由模型 M5.3、M5.4、M5.5 构成的集成模型在 SQuAD2.0-dev 上准确率最高，EM 值可达 88.36%。这是所有实验方案中的最优解。

上述最优集成参数配置将作为第 7 章 QA App 开发服务端最终使用的 MRC 模型配置。

对比模型 M5.2~M5.5 可知，不同折数的模型由于训练数据的差异所适用的最优参数配置也不同。第 1 折模型的效果显著低于后 3 折模型，而第 3 折效果最好，这说明训练数据质量上有：第 1 份>>第 2 份>第 4 份>第 3 份。

尽管如此，上述本地实验得出的最强模型 M5.4 在验证集上的最高准确率 88.4%仍达不到原论文给出的集成模型准确率 88.9%。显然，与第 3 章的问题类似，这一偏差主要是最大序列长度 max_seq_length 设得较小导致的。

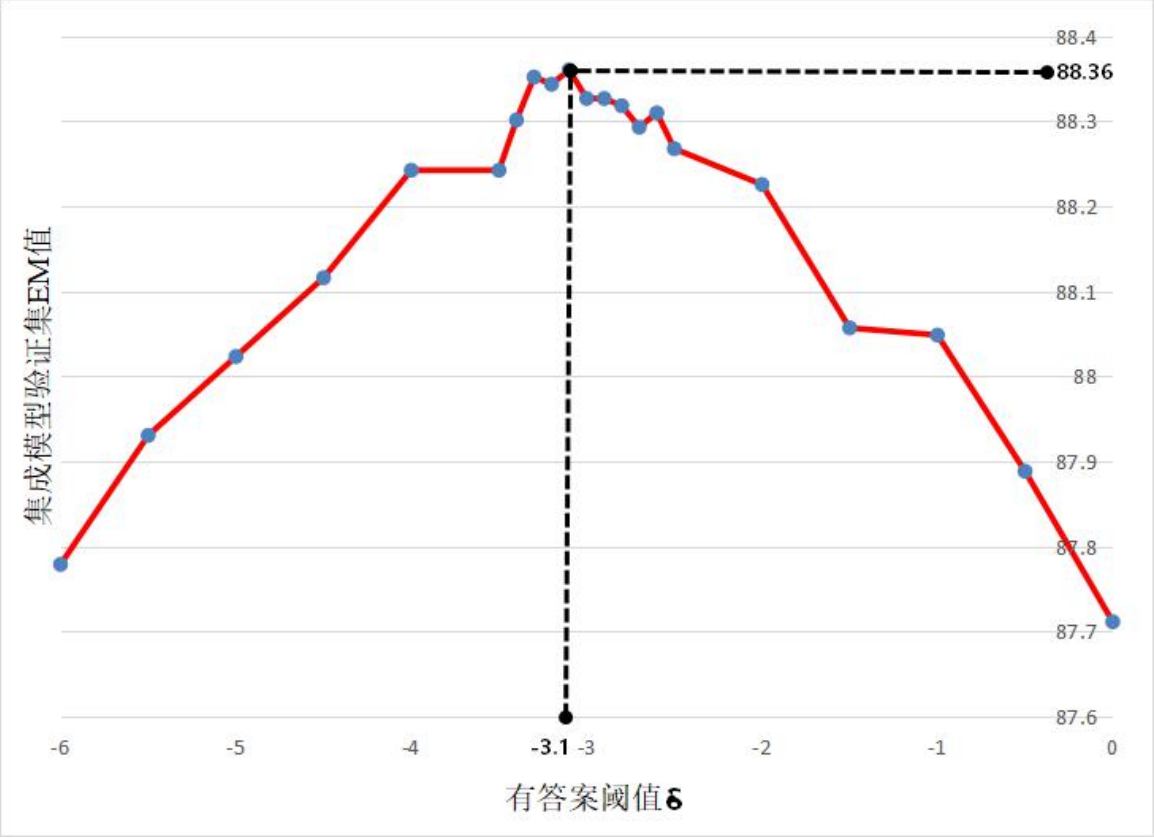

图 5.2 集成模型组合 M5.3~M5.5 的验证集 EM 值关于有答案阈值δ的变化曲线

5.3 本章小结

本章首先对基于 N 折 Albert 的 MRC 模型的设计思路进行了阐述，包括其在训练数据及 logits 集成两方面的特点。然后对该模型进行了实验测试和参数调优，得到了在 SQuAD2.0-dev 数据集上本文所有模型里最好的效果。最后对实验结果进行了分析，同时反思是最大序列长度设得较小而导致集成模型的准确率与原论文中对应结果仍有差距。

6 QA App 的设计与实现

6.1 基础技术介绍

6.1.1 C/S 架构

C/S 架构，即客户机-服务器结构，是一种软件部署模式，采用服务器和客户端两层来实现。使用 C/S 架构实现的程序在物理上一分为二：一是部署在服务器上的服务端程序，负责计算、管理、维护存储在服务器上的应用核心数据；二是部署在部署在用户终端设备（如手机、PC）上的客户端程序，一般带有图形用户界面（GUI），负责与用户进行交互，为用户直接提供服务。

客户端程序和服务端程序间的数据交互通过局域网实现。用户打开客户端软件后，客户端会通过网络向服务端验证用户信息并请求数据。服务端收到后，会在云端运行数据库查询、模型推理等服务，完成后将结果传回客户端。客户端收到后再将结果显示在图形界面上。

此外，服务端程序是多用户共享的，可运行共享数据库等后台服务；而客户端程序是每个用户专用的，可以很方便地实现帮助文档、提示错误等前台功能。

C/S 结构的优点是访问安全、便于实现负载均衡和批量处理数据。但是该结构通用性差，系统数据维护、管理和扩展较为困难，故 C/S 结构主要只适用于规模较小的局域网应用下。[30]

本文中的 App 后端使用多个深度学习模型集成实现，负载较高，为了减小应用对本地显卡配置依赖，充分利用云端资源，同时降低客户端运行负载，QA App 使用 C/S 架构实现。

6.1.2 PyQt5

PyQt5 是一款使用 Python 模块实现的跨平台界面开发工具包，有 6600 多个函数和类，其图形界面接口使用 Qt5 标准。PyQt5 最初由 Riverbank Computing 公司开发，可以兼容包括 Windows、Unix 和 Mac OS 的所有主流操作系统。

本文选用 PyQt5 界面框架开发 QA App 的原因如下：

（1）Python 语言对深度学习的相关支持很友好，后端的深度学习模型最适合使用 Python 编写。为了节约开发成本并保证前后端兼容，本文前端也将使用 Python3 的 GUI 框架进行开发。

（2）PyQt5 库功能丰富，使用便捷。

（3）笔者先前有过 PyQt5 项目开发经验，上手较为容易。

6.2 QA App 系统设计

问答系统 App 使用 Python3 开发，C/S 架构实现。考虑到 MRC 服务本身需要 GPU 资源的支持，对客户端电脑的处理器及显卡配置要求很高。为了降低本地负载，同时为用户提供高效便捷的服务体验，我们决定使用 C/S 架构来实现 APP。即将客户端的 GUI 界面显示与后端的模型计算采用分布式的方法进行实现，从而达到负载均衡的效果。

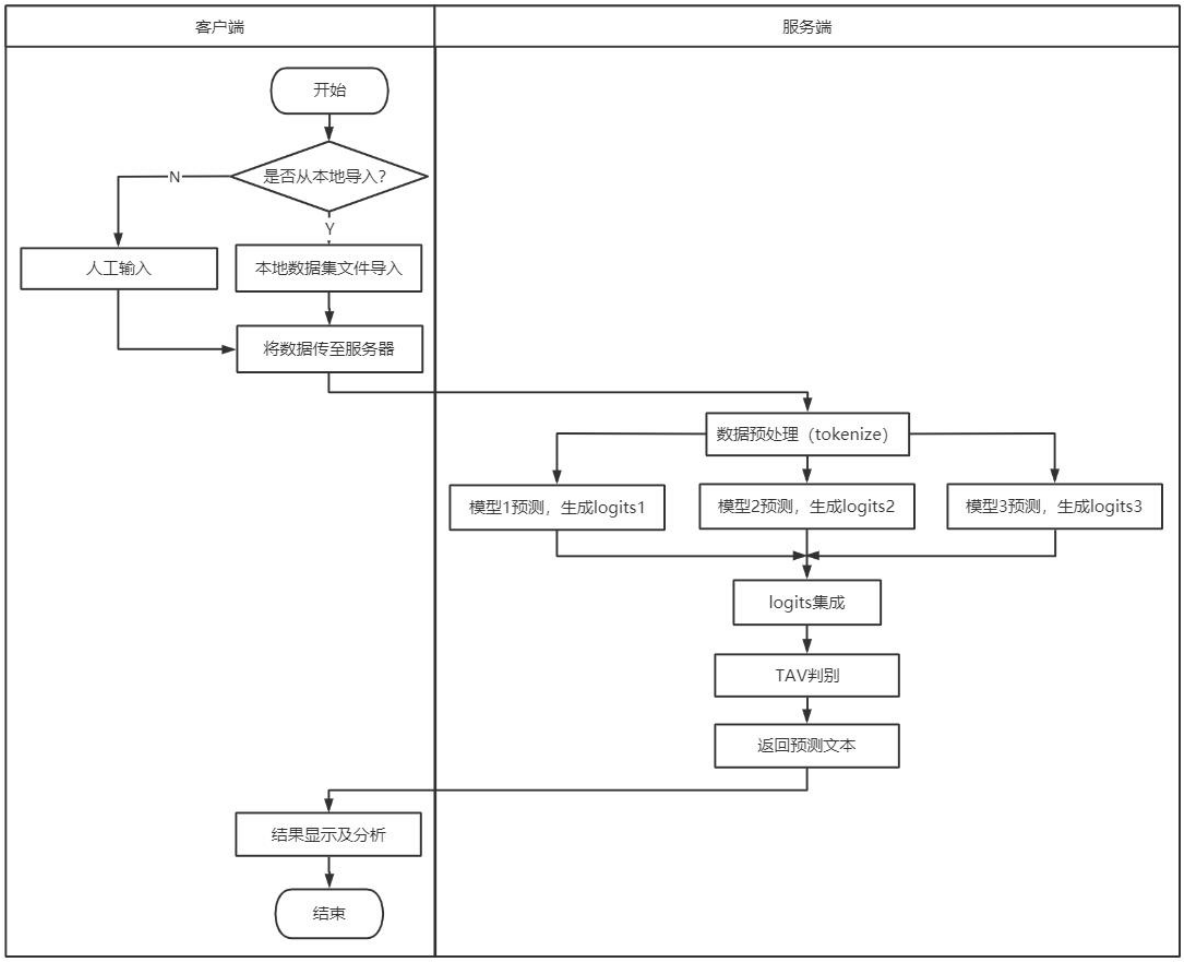

图 6.1 QA App 泳道图

客户端的功能包括界面显示、数据获取和结果分析，从 PyQt5 开发而来，由界面类 Ui_QADemoGui 负责实现。数据获取的方式有两种：（1）从本地数据集文件中随机导入，每一个文件的内容包括上下文、问题及标准答案；（2）由用户手动输入上下文、问题及标准答案。

获得数据后，App 后端会自动将上下文和问题传到 API 服务器上进行模型计算，由云端返回预测结果并显示在客户端 GUI 界面上，同时软件会自动将预测结果与标准答案进行比对，进行结果分析并更新显示出的准确率等指标。

服务端 API 使用的 MRC 模型是本文第 5 章提出的 N 折 Albert 集成模型。经 5.2 节实验表明，该模型在第 3 章~第 5 章提出的所有 3 种选型中效果最好，在 SQuAD2.0-dev 上的准确率可达 88.4%。其中模型折数 N=4，但只取第 2、3、4 折用于集成。

QA App 的泳道图如图 6.1。该系统目前只支持英文。

6.3 客户端界面类 Ui_QADemoGui 实现

作为 QA App 软件中唯一的界面类，Ui_QADemoGui 承担了所有的客户端界面显示功能，包括各种按钮图标显示、事件响应、文本展示、服务器连接及数据统计等。该类派生自 PyQt5.QtWidgets.QMainWindow。其主要控件名称及功能见表 6.1。

表 6.1 Ui_QADemoGui 类中的主要控件功能列表

控件名称	控件类型	控件功能
load_data_and_predict_btn	按钮	点击该按钮即可实现一个完整的从本地导入数据并预测的过程： （1）从本地指定验证集目录（./squad_v2/devset/）下随机选择一个 json 文件加载，并将 Context、Question、Golden Answer 等信息显示到界面上； （2）向 API 服务器传输信息并将返回结果显示在界面的 predicted_answer_label 上； （3）更新界面上已预测样例数、准确率等统计信息。
context_label	文本框	用于显示上下文文本。支持从本地文件导入或者手动输入。
question_label	文本框	用于显示问题文本。支持从本地文件导入或者手动输入。

续表 6.1 Ui_QADemoGui 类中的主要控件功能列表

控件名称	控件类型	控件功能
predict_from_manual_input_btn	按钮	点击该按钮即可实现基于用户手动输入的数据进行预测的过程： （1）从 App 界面上的 context_label、question_label、golden_answer_label 三个文本框中分别获得 Context、Question、Golden Answer 三个信息； （2）向 API 服务器传输信息并将返回结果显示在界面的 predicted_answer_label 上； （3）更新界面上已预测样例数、准确率等统计信息。
predicted_answer_label	文本框	用于显示服务器返回的预测答案。无答案时显示“No Answer”。
time_cost_label	文本框	用于显示服务器的服务响应时延，单位为 ms。该时间包括服务器与客户端之间的通信开销以及服务器在该样例上的计算开销。
answer_check_label	文本框	用于显示预测答案和标准答案的比对结果。结果正确则显示“Exact Match”，否则使用红色字体显示“Not Exact Match”。
predicted_samples_label	文本框	用于显示目前已预测的样本数。每一次重启后该值将自动清零。
accuracy_label	文本框	用于显示自本次客户端启动以后的样例平均准确率，每一次重启后该值将自动清零。准确率字段以“%”结尾，数值上保留两位小数。
golden_answer_label	文本框	用于显示标准答案。标准答案可能有多个，每个答案占一行，且数据保证答案不重复。无答案时显示“No Answer”。支持从本地文件导入或者手动输入。

客户端界面运行效果如图 6.2 和图 6.3 所示。

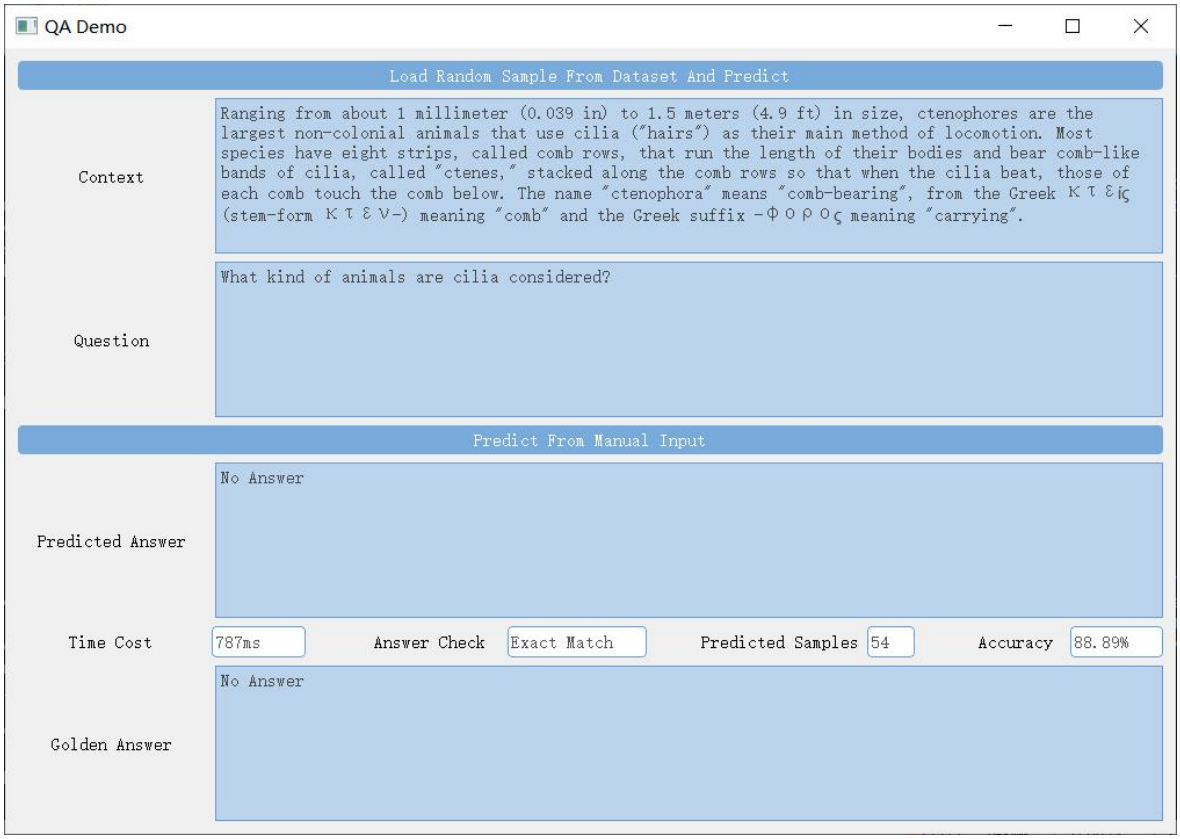

图 6.2 QA App 客户端界面运行效果 1（Exact Match）

图 6.3 QA App 客户端界面运行效果 2（Not Exact Match）

6.4 C/S 连接实现

笔者的服务器地址为 http://60.19.58.86，API 端口号为 20000，连接时将二者拼接成为套接字“http://60.19.58.86:20000”作为 API 地址。

连接方式使用 HTTP 协议的 POST 请求，客户端代码使用 python 的 requests 库实现，服务端代码使用 TestHTTPHandle 类实现，该类是通过派生 http 库的 BaseHTTPRequestHandler 基类并重写 do_GET 和 do_POST 方法得到的。

客户端数据使用 json 格式传输，编码方式为 utf-8，其中发送数据和接收数据使用的关键字列表见表 6.2 和表 6.3。

表 6.2 客户端发送数据键值对列表

关键字字段名	数据类型	值内容	值说明
token	str	qa_en	固定值
context	str	New Zealand (Māori: Aotearoa) is a sovereign island country in the southwestern Pacific Ocean. It has a total land area of 268,000 square kilometers (103,500 sq mi), and a population of 4.9 million. New Zealand's capital city is Wellington, and its most populous city is Auckland.	示例
question	str	What's the largest city?	示例

表 6.3 客户端接收数据键值对列表

关键字字段名	数据类型	值内容	值说明
answer	str	Auckland	示例

其中 token 字段为固定值“qa_en”，其余字段值均只是示例。

客户端使用 post 请求将含有 token、context 和 question 字段的 json 数据传递到服务端后，服务端会使用 utf-8 和 Base64 对字节序列进行解码，还原成传输前的 json 格式。如泳道图 6.1 所示，然后将相关数据经过 N 个模型串行运算、集成后得到 logits，使用 TAV 判别后，再返回预测的答案文本到客户端。

当服务器 API 连接失败或者 API 未开启时，程序会弹出“API 连接失败”的提示，如图 6.4 所示。

6.5 服务端 API 实现

服务端的代码文件主要集成在 api_server.py 中，在命令行界面下使用 python 运

行该文件即可启动 QA API。API 在默认情况下挂载在一个名为“qa-api”的 screen 下，使用快捷键 Ctrl+A-D 将服务挂起，使用 Ctrl+C 终止服务。

关于 API 后端 MRC 模型的设计和实现细节，详情请见第 5 章。

XShell 工具下服务端 API 运行时的 log 界面效果如图 6.5。

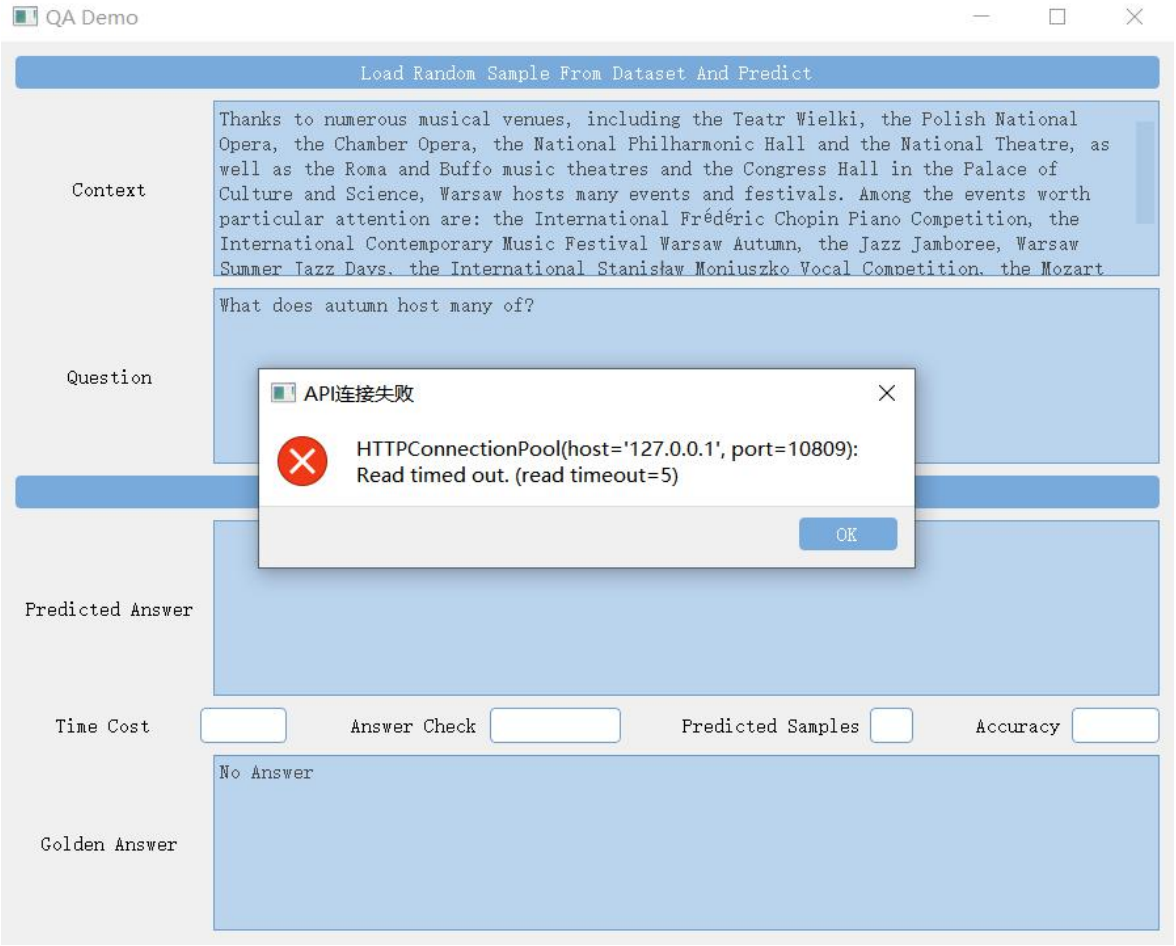

图 6.4 QA App 客户端 API 连接失败时的效果

6.6 系统测试

经过程序集成测试和效果验证，笔者开发的 QA App 使用便捷，响应流畅。在共计 500 轮的连续和间断、文件输入和人工输入测试下，App 的预测平均响应延迟为 734ms，准确率 89.1%，高于毕设开题时制定的“响应延迟不高于 1s，准确率不低于 86%”的软件开发目标。

程序通过了系统测试。

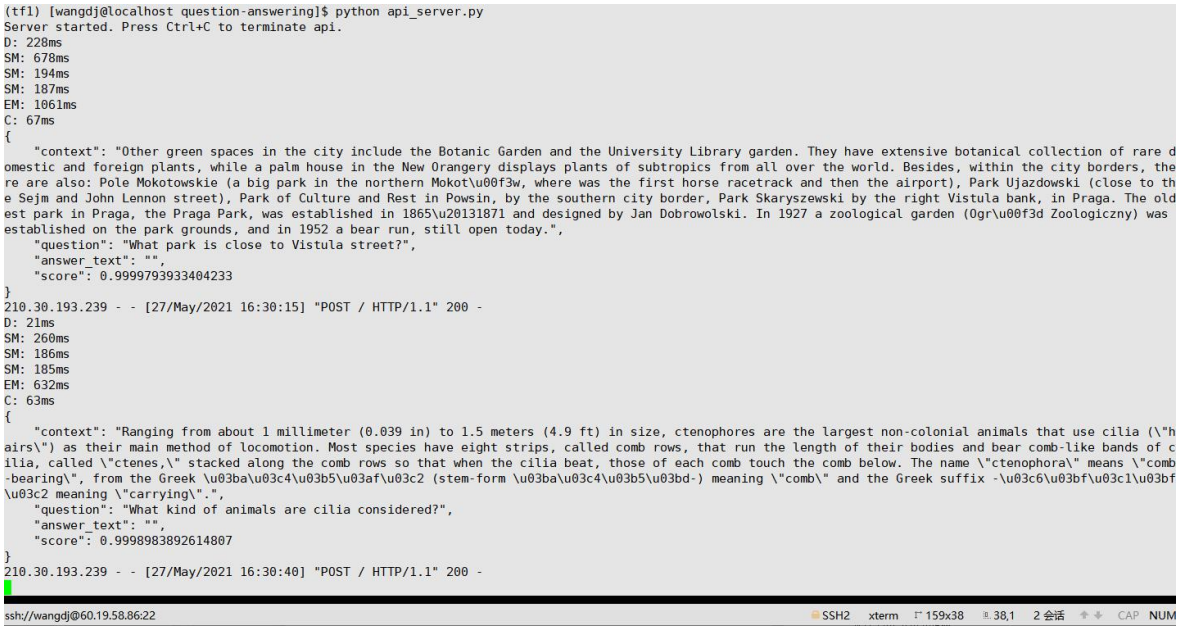

图 6.5 QA App 服务端 API 界面运行效果（Xshell）

6.7 本章小结

本章开发了机器阅读理解系统 QA App。首先，本章对该系统所使用的主要技术 C/S 架构和 PyQt5 作了介绍；其次，介绍了该系统的设计思路、工作流程及实现细节，并进行了客户端和服务端的运行效果展示；最后，针对应用的服务响应速度和准确率对系统进行了综合测试。

7 总结与展望

7.1 总结

通过近半年的学习，基于片段抽取的机器阅读理解系统的设计与实现已经顺利完成，不仅完成了任务书中的要求，而且在系统测试的结果也较为成功，令人满意，尤其是响应速度和准确率上也符合预期目标。本文的重点在于 N 折 Albert 模型的搭建、模型在 SQuAD2.0-train 上的训练、服务端 API 以及客户端 GUI 界面的开发。为便于称呼，下面将本文最终实现的系统称为“本系统”。

对这段时间的学习和工作总结如下：

（1）在工作前期，收集国内外相关研究现状，并进行分析，了解本课题的研究难点和重点，综合考量可行性和先进性之后选择了 Albert 模型作为 baseline，为接下去的研究进展作铺垫。

（2）本文从经典的问答数据集 SQuAD2.0 上获取数据，训练集大小 130319，验证集大小 11873，保证了充足的数据用于模型训练和评估。另一方面，该数据集中有一半左右的“不可回答问题”，该类型问题的出现使得本系统的实用性更高，同时也对模型理解能力提出了更高的要求。

（3）本文共训练了数十个基于神经网络的 MRC 系统，其中效果较好的有：在 SQuAD2.0 上的 3 个 Albert+Ptr-Net 模型、在 NewsQA 上训练的 1 个 Albert 模型、和在 SQuAD2.0 上使用 4 折的方式训练出的 4 个 Albert 模型。从准确率的角度考虑，由于基于 NewsQA 的模型在 SQuAD2.0-dev 上的准确率只有 63%，而基于 SQuAD2.0-train 的 Albert+Ptr-Net 和第 1 折 Albert 准确率均未过 81%，与另外 3 个准确率在 87%附近的模型差距较远，故只取后 3 折模型进行集成。

（4）为满足系统演示的便捷性，同时支持用户手动输入数据，本系统支持从本地导入数据集文件和用户手动在文本框中输入两种模式下的数据读取。不同的输入模式供用户自己选择，满足人们对于个性化的追求。

（5）本系统能够实时展示模型的预测准确率，便于软件使用者了解模型的效果，提升了用户的使用体验。

（6）本系统使用 C/S 架构开发实现，引入了云计算的思想，把模型推理的部分放在了云端 GPU 服务器上实现，并使用 API 接口与客户端进行数据交互。成功地减

轻了本地客户端的负载，同时提高了模型的计算效率。

（7）笔者在今年开始做毕业设计之前对 NLP 领域了解不多，但通过这一学期对 MRC 系统的大量探索和实践，笔者学习了大量 NLP 方面的网络模型和调参技术，也在编程实践中找到了做 AI 学术研究的乐趣，这更加坚定了笔者研究生期间要做 NLP 方向研究的志向。

7.2 展望

本文完成了机器阅读理解系统这一核心功能，并添加了基于 C/S 架构的软件实现。总结全文，本文还有一些可以进一步研究的工作：

（1）扩大 MRC 系统的答案形式。本系统最终实现的是片段抽取式的 MRC 系统，该种获得答案的方式与实际生活中仍有一些差距，可以考虑进一步扩大答案的来源形式到完全的自由格式问答。

（2）增加对其他模型的尝试。本文最终使用的是 N 折 Albert 模型，生成答案回复的效果符合预期，但在训练过程中需要预设较多参数。参数调节的好坏对 MRC 模型有着较大的影响。因此，预设的参数事先需要进行大量对照实验从而确定。还可以增加其他模型的尝试，从而提高模型的准确度。

（3）提供对其他语言的支持。本文开发的 MRC 系统目前仅支持英文文本的输入和处理，后期可以使用一些其他语言（如中文）的数据集训练 MRC 模型，以添加对其它主流语言的阅读理解功能支持。

参考文献

[1] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations[C]. arXiv preprint arXiv:1909.11942, 2019.

[2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention Is All You Need[C]. In NeurIPS, 2017: 6000-6010.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding[C]. arXiv preprint arXiv:1810.04805, 2018.

[4] Wendy G Lehnert. The Process of Question Answering[D]. PhD thesis, Yale University, 1977.

[5] Lynette Hirschman, Marc Light, Eric Breck, and John D Burger. Deep read: A reading comprehension system[C]. In ACL, 1999.

[6] Ellen Riloff and Michael Thelen. A rule-based question answering system for reading comprehension tests[C]. In NAACL, 2000.

[7] Hoifung Poon, Janara Christensen, Pedro Domingos, Oren Etzioni, Raphael Hoffmann, Chloe Kiddon, Thomas Lin, Xiao Ling, Alan Ritter, Stefan Schoenmackers, et al. Machine reading at the university of washington[C]. In NAACL, 2010.

[8] Bhuwan Dhingra, Hanxiao Liu, Ruslan Salakhutdinov, and William W Cohen. A comparative study of word embeddings for reading comprehension[C]. arXiv preprint arXiv:1703.00993, 2017.

[9] Danqi Chen. Neural Reading Comprehension and Beyond[D]. PhD thesis, Stanford University, 2018.

[10] Shanshan Liu, Xin Zhang, Sheng Zhang, Hui Wang, and Weiming Zhang. Neural Machine Reading Comprehension: Methods and Trends[J]. Applied Sciences, 2019, 9(18):3698.

[11] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text[C]. In EMNLP, 2016: 2383-2392.

[12]Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don’ t know: Unanswerable questions for squad[C]. arXiv preprint arXiv:1806.03822, 2018.

[13]Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension[C]. arXiv preprint arXiv:1705.03551, 2017.

[14]Wei He, Kai Liu, Yajuan Lyu, Shiqi Zhao, Xinyan Xiao, Yuan Liu, Yizhong Wang, Hua Wu, Qiaoqiao She, Xuan Liu, et al. Dureader: a chinese machine reading comprehension dataset from realworld applications[C]. arXiv preprint arXiv:1711.05073, 2017.

[15]Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend[C]. In NeurIPS, 2015: 1693-1701.

[16]Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. Ms marco: A human generated machine reading comprehension dataset[C]. arXiv preprint arXiv:1611.09268, 2016.

[17]Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. Race: Large-scale reading comprehension dataset from examinations[C]. In EMNLP, 2017: 785–794.

[18]Trischler, A.; Wang, T.; Yuan, X.; Harris, J.; Sordoni, A.; Bachman, P.; and Suleman, K. NewsQA: A Machine Comprehension Dataset[C]. In Proceedings of the 2nd Workshop on Representation Learning for NLP, 2017: 191–200.

[19]Adina Williams, Nikita Nangia, and Samuel R Bowman. A broad-coverage challenge corpus for sentence understanding through inference[C]. In NAACL, 2018.

[20]Yinhan Liu, Myle Ott, Naman Goyal et al. RoBERTa: A Robustly Optimized BERT Pretraining Approach[C]. In ICLR, 2020.

[21]Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. XLNet: Generalized autoregressive pretraining for language understanding[C]. arXiv preprint arXiv:1906.08237, 2019.

[22]Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. ERNIE: Enhanced language representation with informative entities[C]. In ACL, 2019:

1441-1451.

[23]Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. Ernie 2.0: A continual pre-training framework for language understanding[C]. arXiv preprint arXiv:1907.12412, 2019.

[24]Zhuosheng Zhang, Junjie Yang, and Hai Zhao. Retrospective reader for machine reading comprehension[C]. arXiv preprint arXiv:2001.09694, 2020.

[25]Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. Gated self-matching networks for reading comprehension and question answering[C]. In ACL, 2017.

[26]Shuohang Wang and Jing Jiang. Machine comprehension using match-lstm and answer pointer[C]. arXiv preprint arXiv:1608.07905, 2016.

[27]Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks[C]. In NeurIPS, 2015: 2692–2700.

[28]amitness.BERT 的 youxiu 变体： ALBERT 论文图解介绍[EB/OL].CSDN 博客,https://blog.csdn.net/u011984148/article/details/106205143/．2020-05-18.

[29]jyh764790374.R-NET 机器阅读理解（原理解析）[EB/OL].CSDN 博客,https://blog.csdn.net/jyh764790374/article/details/80247204．2018-05-08.

[30]张连明．服务器-客户机[EB/OL]. 百度百科 ,https://baike.baidu.com/item/%E6%9C%8D%E5%8A%A1%E5%99%A8-%E5%AE%A2%E6%88%B7%E6%9C%BA/5937024．2016-12-01.

致谢

大学本科的四年时光，匆匆即逝。恍惚之间，我已从2017年迈入东北大学的那个懵懂无知、对校园里的一切都充满着憧憬的“学弟”，成长为一个即将从母校光荣毕业、读研的“学长”。回首在东大的这段时光，有过得知自己获得优异成绩的喜悦，有过竞赛失利的悲痛，获得过不少成就，也遇到过一些挫折。但总体而言，我生活得很快乐充实。这其中少不了老师、家人和同学对我的帮助和鼓励。在此，我向他们表示衷心的感谢。

首先，我非常感谢我的指导老师肖桐老师、王会珍老师、王厚峰老师等对我毕业设计的指导。在整个过程中，给予我很大帮助与支持。在一开始对论文题目感到迷茫，把握不定方向时，是老师给予我建议，给我指定方向；在我对接下去的工作无从下手时，是老师肯定我的方向，并扩宽我的思路；在我对毕业设计工作感到懈怠时，是老师每周一次的工作指导，让我坚定继续工作。

感谢实验室提供的学习氛围和实验环境，让我能在疫情这样特殊的时期，坚持学习，努力提升自我，并完成本次毕业设计。

感谢黄潋哲等学长学姐们，他们为我提供了很多思路，引导我自己去尝试查找资料、实验和反思，锻炼了我独立思考和解决问题的能力。

感谢一起做毕业设计的小伙伴，他们给予我很多帮助，并相互督促，共同进步。

感谢父母为我提供了温馨的港湾，让我能够丰衣足食，在学校没有后顾之忧。

最后，感谢四年来的寝室室友，感谢他们长期以来的关怀与鼓舞、陪伴与支持，让我的生活多姿多彩，令人难忘。也让我对未来的每一天充满期待。

谢谢大家！