项 目 结 题 报 告

项目名称：BERT 预训练模型对问答机器人的效果提升

委托方（甲方）：马上消费金融股份有限公司

受托方（乙方）：重庆师范大学

项目负责人：吴至友

手 机：15923585049

通讯地址：重庆师范大学大学城校区数学科学学院

邮政编码：400030

时 间：2020 年 8 月

目录

一、 BERT 模型简介

BERT(Bidirectional Encoder Representations from Transformers)[1]是由谷歌提出的自然语言处理领域的预训练模型，自提出以来，已大幅刷新了 NLP 领域 11 项任务的精度，也是 NLP 领域的一大技术突破。随着 BERT 模型的问世，也不断出现了基于 Transformer[15]的其他模型，如 XLNet[2]、Roberta[3]、Albert[4]、Electra[5]等预训练模型。而在下游应用中，基于预训练模型的 fine tune 也逐渐成为了一种较为高效的方式，本项目的主要工作也是在意图识别任务中，针对 BERT 系列模型进行 fine tune。

二、 项目要求

（1）小数据集：针对于公司提供的小数据集（660 条训练数据，220 条验证数据），搭建基于 BERT 系列模型的 fine tune 模型，并能够达到结题要求。

（2）公开数据集：针对于公开的意图识别评测数据集，基于 BERT 系列模型搭建 fine tune 模型，能够达到指定要求。

（3）大数据集：针对公司提供的大数据集，搭建基于 BERT 系列模型的 fine tune 模型，并能够达到结题要求。同时，模型的单条数据预测时间要小于 200 毫秒。

经马上金融方与重师方多次沟通、商定后达成了一致的结题要求：重师方的模型在小数据集、公开数据集、大数据集三个任务上的效果优于马上金融的模型在对应任务上的效果，如果在某一任务上，重师方模型的准确率较马山金融模型在该任务中的准确率低于 1%以内，若重师方的解决方案与经验能为马上金融的相关研究提供一些建议和启发也可视为达标，予以结题。

三、 主要工作

根据项目要求，分别针对公司提供的小数据集，大数据集，以及公开意图识别评测集，搭建 BERT fine tune 模型。以下将从小数据集 fine tune 模型，公开评测集 fine tune 模型以及大数据集 fine tune 模型三个任务，介绍重庆师范大学数学科学学院 NLP 小组在本项目中的主要工作。

（一）小数据集 fine tune 模型

本章节将从方案设计，实验分析以及结论三个方面加以具体阐述。

1. 方案设计

在小数据集场景下，随着项目研究开展的逐步深入，为解决任务中模型，数据等方面的问题，共制定三套解决方案，方案一：单模型 fine tune、方案二：数据增强以及方案三：MT-DNN 模型，以下将具体阐述。

1）方案一：单模型 fine tune

① 背景和意义

如图 3.1.1 所示，文本分类中 BERT 将最后一层的[CLS]作为输出，来表达句子含义，并通过 Softmax 对句子的意图进行分类[1]。

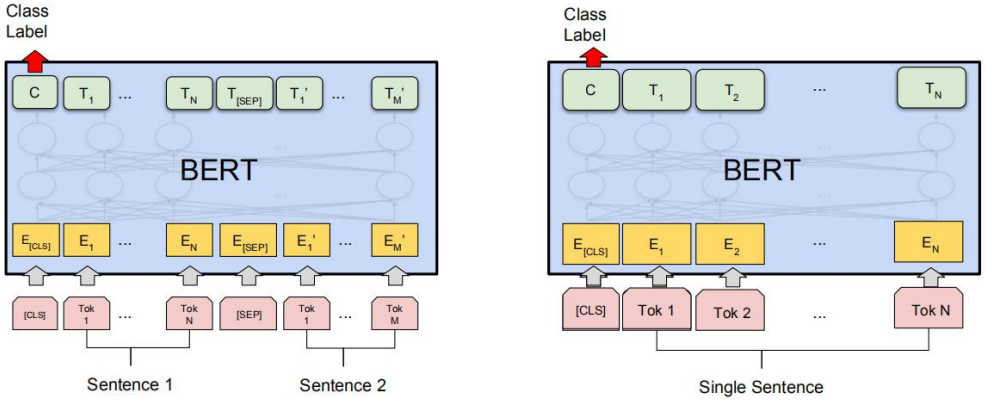

图 3.1.1 BERT fine tune 示例

表 3.1.1 Sun 等人论文实验结果

Method	IMDB	Sougou
head-only	5.63	2.58
tail-only	5.44	3.17
haed+tail	**5.42**	**2.43**
hier.mean	5.89	2.83
hier.max	5.71	2.47
hier.self-attention	5.49	2.61

为进一步提升 BERT 在文本分类中的性能，Chi Sun[6]等人提出多种 fine tune 方法，表 3.1.1 展示了该工作中的不同 fine tune 方法在中文

和英文数据集上进行文本分类的性能变化。其中 head-only 为使用[CLS]进行文本分类，tail 则使用 BERT 中最后一层的最后一个向量进行分类，而 hier 则是针对除头尾以外的其他向量进行分类。根据实验结果可知，通过拼接 head 和 tail 组成新向量后可以提升文本分类的效果。

受到以上工作的启发，搭建了得到 BERT 不同输出的三种 fine tune 模型。

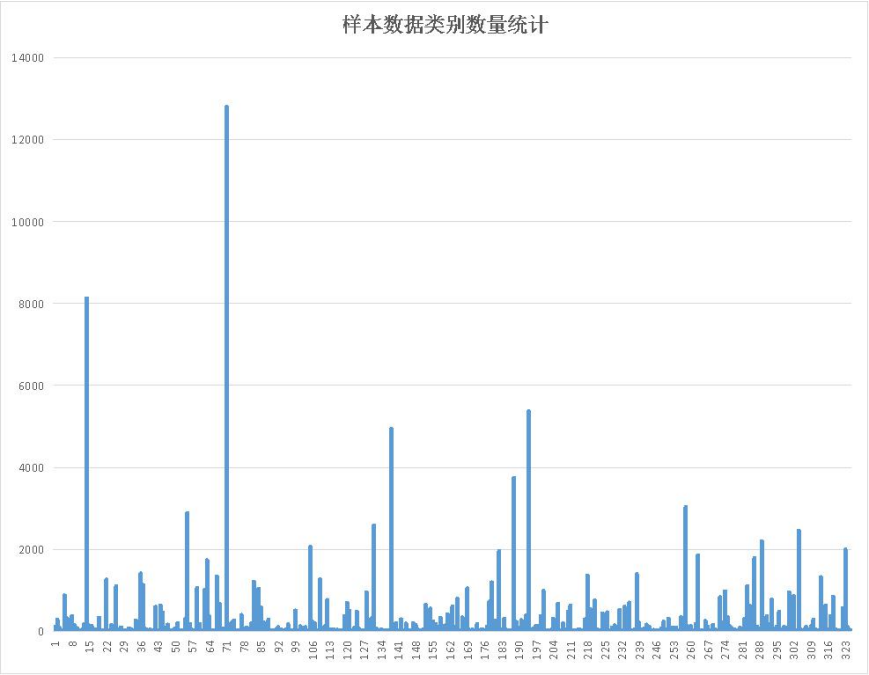

图 3.1.2 样本数据各类别数据量统计

在真实场景中(图 3.1.2 中为大数据集任务),训练数据可能存在类别不均衡的问题，如图 3.1.2 所示，第 13，第 71 等类数据量较大,而第 92，第 240 等类别数据量非常少。为一定程度克服这些问题，本次方案拟采用 LabelSmoothing[7]、 FocalLoss[8]作为损失函数。LabelSmoothing 将标签“光滑化”，避免预测标签的概率两极化，从而一定程度上增强模型的泛化能力，也一定程度上克服数据类别不均衡等问题。FocalLoss[8]由何凯明等人提出,采用调制系数的函数去度量难分类和易分类样本对总的损失的贡献，最后将两者结合，既能调整正负样本的权重，同时控制难易分类样本的权重，从而一定程度上克服数据的类别不均衡的问题。

② 方案介绍

参考现有工作的方法，搭建 BERT fine tune 的 Bert_base，Bert_pad 以及 Bert_sep 模型，具体介绍如下。

a. **Bert_base 模型：**该结构如图 3.1.3 所示，将 Bert 模型最后一层[CLS]的输出，作为 BERT 的输出，输入到下游分类任务中。

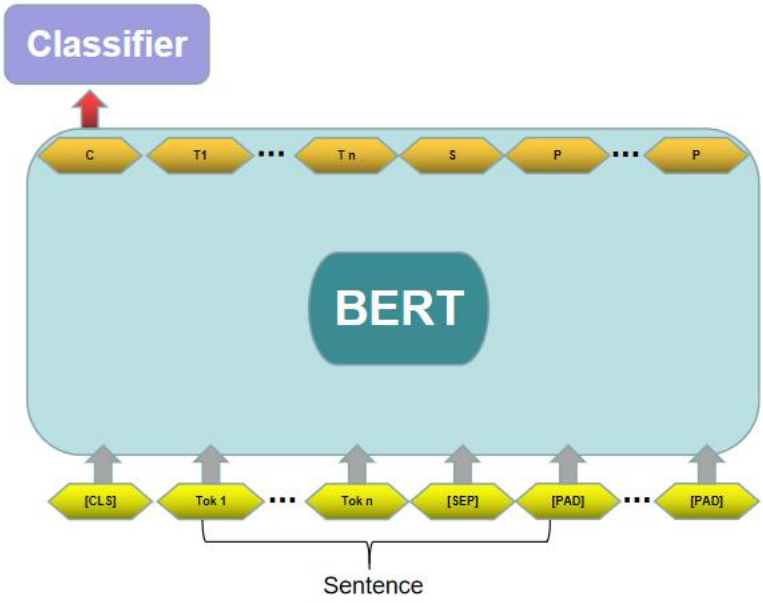

图 3.1.3 Bert_base 模型示意

b. **Bert_pad 模型：**该结构如图 3.1.4 所示，将 Bert 模型最后一层的[CLS]的输出与最后一个[PAD]的输出，拼接后作为 BERT 输出，输入到下游分类任务中。

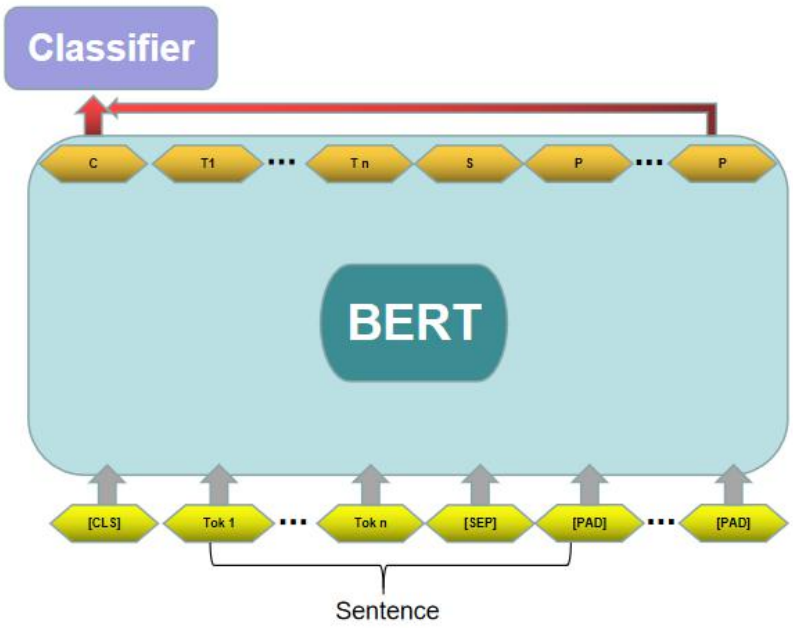

图 3.1.4 BERT_pad 模型示意

c. **Bert_sep 模型：**该结构如图 3.1.5 所示，将 Bert 模型最后一层的[CLS]的输出与[SEP]的输出，拼接后作为 BERT 输出，输入到下游分类任务中。

在以上三个模型的基础上，选取不同的损失函数：CrossEntropyLoss、LabelSmoothing[7]和 FocalLoss[8]，来进一步验证模型以及损失函数对该任务效果的影响。

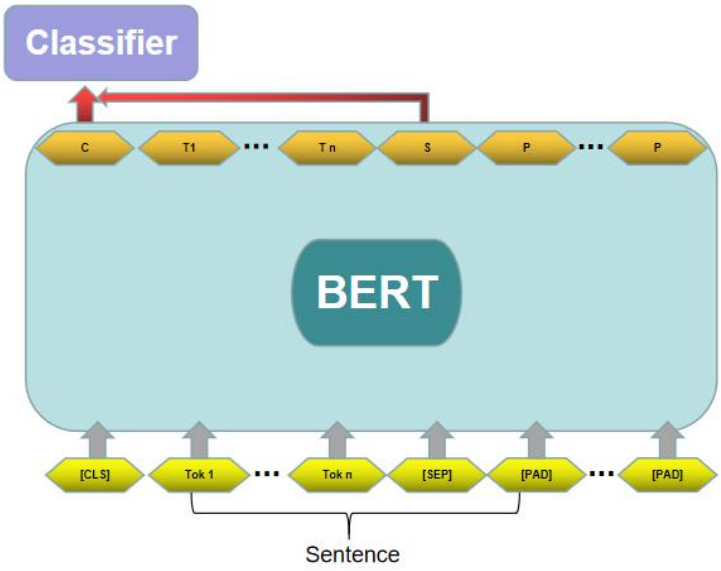

图 3.1.5 BERT_sep 模型示意

2）方案二：数据增强

① 背景和意义

在小数据集任务中可能存在两个问题影响模型在分类时的性能，第一个问题是公司提供的训练数据仅有 660 条，可能存在训练数据量不足的问题；第二个问题是若验证集中存在单词没有在训练集中，即 OOV（Out Of Vacabulary）问题，则可能会影响模型预测效果[20]。为解决可能存在的以上两个问题，考虑尝试数据增强。

② 方案介绍

为解决第一个问题，方案二拟采用数据增强的办法增加训练数据。具体方式如图 3.1.6 所示，该方法首先随机选择一个训练数据，其次在该训练数据中随机选择一个名词或者动词利用预训练好的 word2vec 得到该词的近义词，随后利用近义词替换该名词或者动词进而得到新的训练数据。最后将新的训练数据添加到训练集中扩大训练集。

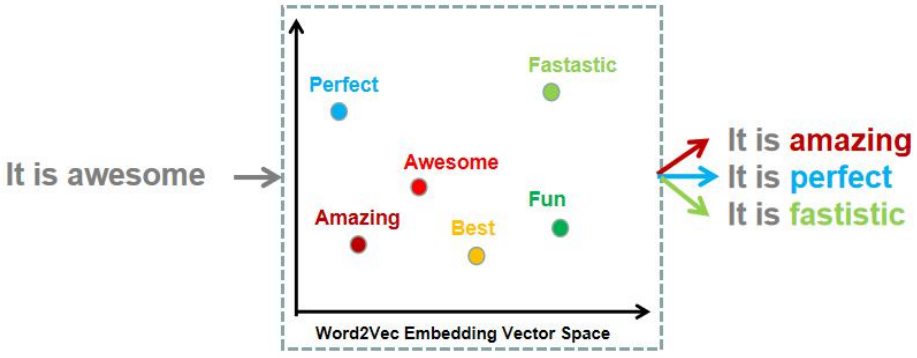

图 3.1.6 word2vec 数据增强示意

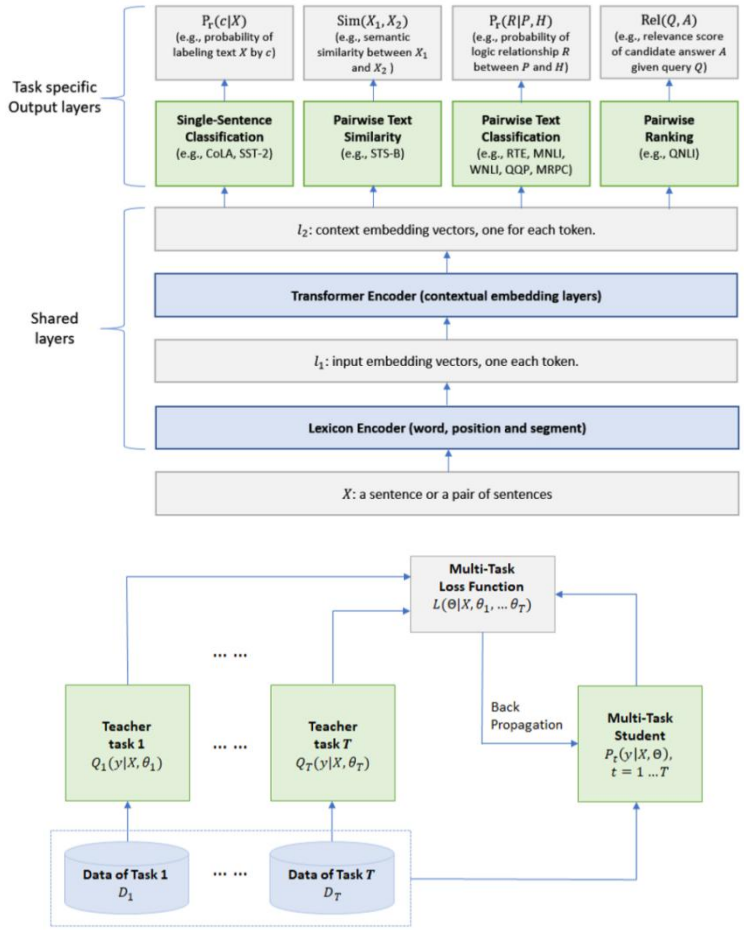

图 3.1.7 MT-DNN 模型示意

3）方案三：MT-DNN 模型

① 背景和意义

MTDNN[10]是一个多任务对 bert 进行 finetune 的框架，从实验结果中可知将多任务的 teacher net 学到的知识注入到 student net 中，可以有效的提高 bert 在各个任务上的性能，故考虑将 MT-DNN 方法引入到本任务中。

② 方案介绍

MT-DNN 算法，如图 3.1.7 所示，MT-DNN 采用 Teacher Net 与 Student Net 的形式，将所学知识存在互补性的模型作为 Teacher Net，同时 Teacher Net 和 Student Net 共享同一个 Bert 模型的参数，通过模型的学习，将不同模型的知识注入 Student Net 中。本次方案中，Teacher Net 拟采用 TextCNN[11]、TextRNN[12]、RCNN[13]、Bert_base 四个模型；

Student Net 为 Bert_pad 模型,选择这些模型的原因会在第三章第一节方案三的实验中加以说明。

2. 实验与分析

1）方案一：单模型 fine tune 实验

① 数据集

数据为公司提供的训练数据 660 条，验证数据 220 条，共包含 22 个类。该实验将 660 条的训练数据与 220 条验证数据合并后，采用五折交叉验证，如图 3.1.8，将数据分为五份，其中四份做训练，一份做测试，构成交叉验证数据集。

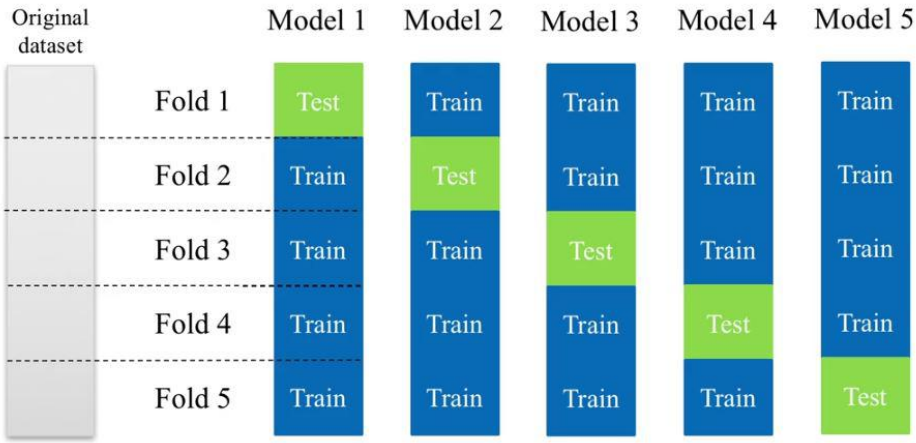

图 3.1.8 交叉验证数据集示意

② 实验设计

针对该方案中的三个 fine tune 模型：Bert_base、Bert_pad、Bert_sep，分别在五份交叉验证数据集上进行验证。

在以上实验的基础上，进一步验证损失函数对分类效果的影响，故针对以上三个模型 Bert_base、Bert_pad 和 Bert_sep，在各自模型参数等配置不变的前提下，采用不同的损失函数分别训练三个模型，来验证损失函数对模型效果的影响。

表 3.1.2 硬件及模型配置

配置	参数
CPU	E5-2630 V4
GPU	Titan XP 12G
优化器	Torch.AdamW
损失函数	CrossEntropyLoss

③ 参数设置

实验中除模型外，采用 huggingface[21]的 Pytorch 代码，具体配置如下，实验中的变量为模型与其对应参数。实验中的模型超参数包括：Epoch、Lr(Learning rate)、Batchsize、Max_length，实验的硬件配置如表 3.1.2 所示，同时参数设置如表 3.1.3 所示。

表 3.1.3 实验参数设置

Parameters	Range
Epoch	{20，21，40，42，**45**，50}
Lr	{**5e-5**，6e-5}
Batchsize	{32}
Max_length	{100}

④ 实验结果与分析

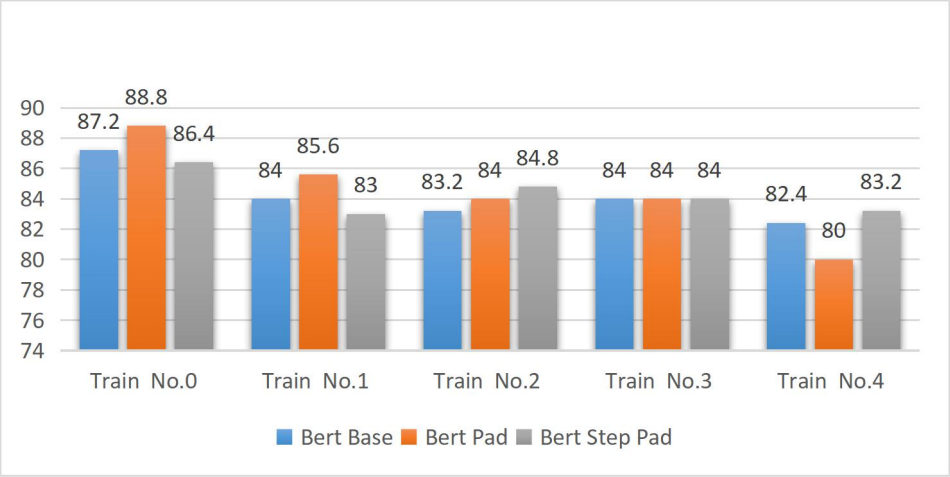

图 3.1.9 交叉验证准确率对比

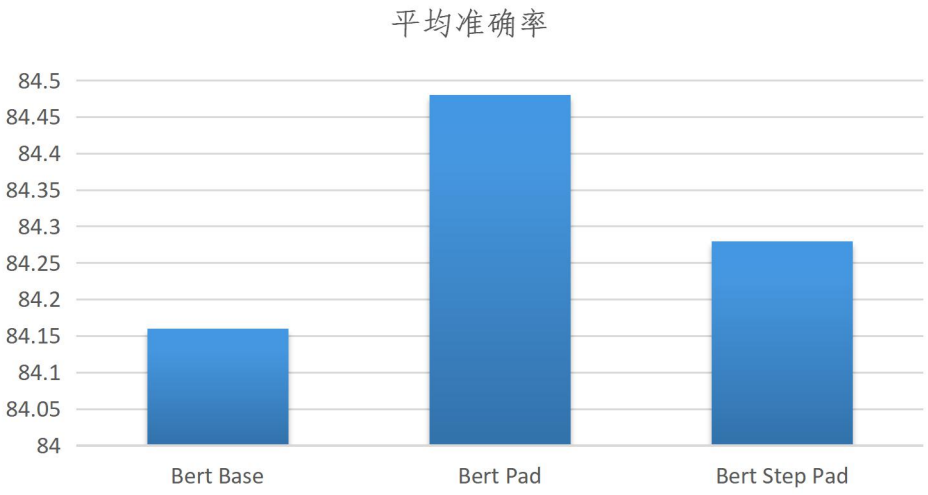

图 3.1.10 不同模型平均准确率对比

结果 1：采用交叉熵损失函数，针对不同的 fine tune 模型，在五个交叉验证数据集上的准确率对比如图 3.1.9 所示。三个模型在五个交叉验证数据集上的平均准确率如图 3.1.10 所示。

结论 1：Bert_pad 模型较另外两个模型的效果要稍好一些，但距离项目要求仍差距较大。

结果 2：针对不同的损失函数，三个 fine tune 模型在交叉验证数据集上的效果对比如图 3.1.11 所示。

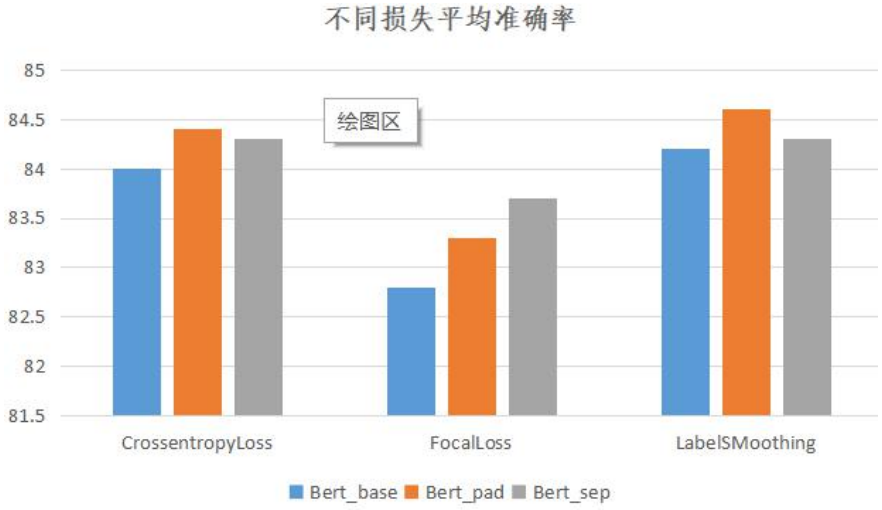

图 3.1.11 不同损失平均准确率对比

表 3.1.4 标签相关联的数据示例

预测错误原句	预测标签	真实标签
您好!我想了解一下，一年可以去多少次港澳，旅游团签和个签去港澳的次数一样吗？	旅游	逗留有效期及频次
请问成都户口在办理港澳旅游签注的时候能一次办理多次往返的签证吗？	逗留有效期及频次	旅游
我 10 月 28 日-11 月 2 日六天时间在香港澳门旅游，机票是订的港进港出，中间想去澳门旅游 2 天，请问需要几次香港签注	逗留有效期及频次	旅游

结论 2：由图 3.1.11 可知，LabelSmoothing 较另外两种损失函数的效果要好，FocalLoss 的效果最差，LabelSmoothing 的效果最佳，也说明了测试数据存在不同标签数据之间有相互关联的问题，例如表 3.1.4 所示，“旅游”与“逗留有效期及频次”标签的数据之间存在很大的相似性，这样的数据既可以属于“旅游”也可以属于“逗留有效期及频次”（预测错误数据共 20 多条）。

2) 方案二：数据增强实验

① 数据集

该实验数据集与方案一：单模型 fine tune 实验相同，依然为五折交叉验证数据集。

② 实验设计

表 3.1.5 硬件及模型配置

配置	参数
CPU	E5-2630 V4
GPU	Titan XP 12G
优化器	Torch.AdamW
损失函数	CrossEntropyLoss

针对方案一中的交叉验证数据集，采用 word2vec 进行数据增强后得到的数据集，采用 Bert_base 模型（因为该模型为 BERT 微调的基础模型）进行训练，该实验较上一实验仅数据集变为增强后的数据集，其他因素保持不变。实验中的硬件等配置如表 3.1.5 所示。

③ 参数设置

Bert_base 模型超参数包括：Epoch、Lr(Learning rate)、Batchsize、Max_length，具体参数设置如表 3.1.6 所示。

表 3.1.6 实验参数设置

Parameters	Range
Epoch	{40，42，50}
Lr	{6e-5}
Batchsize	{32}
Max_length	{100}

④ 实验结果与分析

结果 1：首先将交叉验证数据集的 OOV 比例与 Bert_base 在各个交叉验证数据集上效果进行对比。如图 3.1.12 所示，左图为各个交叉验证数据集 OOV 的比例对比表，右图为 Bert_base 模型在各个交叉验证数据集上的准确率对比。

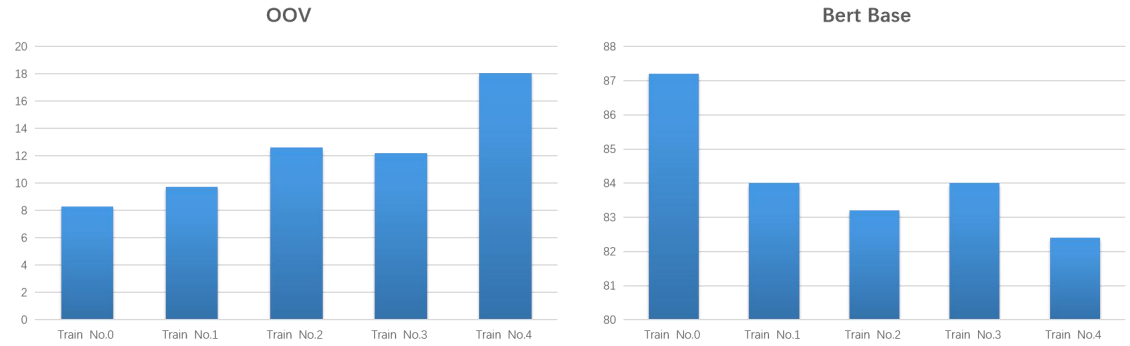

图 3.1.12 OOV 与 Bert_base 准确率对比

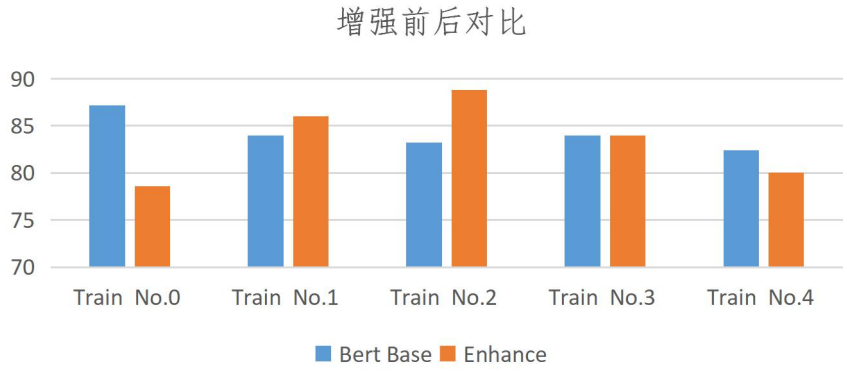

图 3.1.13 数据增强前后效果对比

结果 2：Bert_base 模型采用增强后的数据集训练后，在各个数据集上的效果对比如图 3.1.13 所示。

结论 1：模型的性能和 oov 的比例呈负相关，即 oov 比例越大，模型的性能越差，反之 oov 的比例越小，模型的性能越好。

表 3.1.7 数据增强单词统计

训练集序号	原数据集字数	增强数据集字数	在增强不在原	在测试不在原	在增强不在测试	增强后与原	比例
Train0	660	876	216	20	19	1	0.46%
Train1	647	861	214	30	22	8	3.74%
Train2	680	850	210	36	26	10	4.76%
Train3	662	925	263	40	30	10	3.80%
Train4	607	870	263	69	41	28	10.65%

结论 2：数据增强后，在有的数据集中准确率可以提升，但在有的数据集上准确率却大幅下降，是因对比原数据集，增强数据集会减少测试集不在训练集中的字，但训练集中增添了很多不在原训练集与测试集中的

字，属于“冗余信息”，统计如表 3.1.7，从而导致了图 3.1.12 中的结果，反之，若尽可能减少测试集不在训练集中的字的同时引入更少的“冗余信息”，则可以提升模型的效果。

结果 3：针对以上数据增强存在的问题，设计如下实验，（1）在数据增强过程中，将训练集中增加的字进行减少；（2）将测试集中不在训练集中的字进行扩充。

表 3.1.8 数据增强分析结果

数据集处理	原数据集	train,test 去字母	train 换 20	train 换 60	train 换 100
NO.train0	0.784	0.84	0.856	0.872	0.864
NO.train1	0.86	0.816	0.816	0.824	0.808
NO.train2	0.88	0.856	0.864	0.864	0.864
NO.train3	0.84	0.848	0.848	0.832	0.84
NO.train4	0.792	0.776	0.8	0.808	0.784

表 3.1.9 数据增强分析结果

数据集处理	train 换 20 去语气助词	train 换 60 去语气助词	train 换 100 去语气助词	train 换 20 test 全换去语气助词	train 换 60 test 全换去语气助词	train 换 100 test 全换去语气助词
NO.train0	0.864	0.872	0.872	0.856	0.88	0.864
NO.train1	0.824	0.8	0.816	0.832	0.808	0.816
NO.train2	0.872	0.848	0.856	0.872	0.848	0.856
NO.train3	0.832	0.84	0.848	0.84	0.832	0.856
NO.train4	0.784	0.816	0.816	0.8	0.784	0.808

具体方案如下：

（1）增强数据集对比原数据集，将增强数据集中多出的英文字母去掉（在不影响原句语义的前提下）；

（2）初步分析训练集中的标点去掉后，效果可能会上升，故制作一份训练集去掉标点的数据集；

（3）针对训练集增加的字数，先减少二十个，测试效果；减少五十个字数，测试效果；减少一百个字数，测试效果；

（4）在（3）的前提下，分别去掉以上三个数据集中的语气助词；

（5）在（4）的前提下，将测试集中没有的字添加到训练集中，分别对应以上三个数据集。

最终的结果对比如下表 3.1.8 和表 3.1.9 所示。

结论 3：在五个交叉验证数据集中，三个数据集上，减少训练集中增加的单词，同时减少语气助词等可以提升模型效果。

方案三：MT-DNN 模型实验

① 数据集

该实验数据集与方案一：单模型 fine tune 实验相同，依然为五折交叉验证数据集。

② 实验设计

在交叉验证数据集上，采用 MT-DNN 算法训练，本实验与方案一实验的不同因素为训练的算法不同，并且选用的优化器为 Lookahead，加入 EMA、AverageMeter 优化技巧，具体相关配置见表 3.1.10。

表 3.1.10 硬件及模型配置

配置	参数
CPU	E5-2630 V4
GPU	Titan XP 12G
优化器	Lookahead
损失函数	CrossEntropyLoss

③ 参数设置

模型超参数包括：Epoch、Lr(Learning rate)、Batchsize、sMax_length，具体参数设置如表 3.1.11：

表 3.1.11 实验参数设置表

Parameters	Range
Epoch	{55, 65}
Lr	{5e-5, 6e-5}
Batchsize	{16}
Max_length	{70}

④ 实验结果与分析

结果 1：之前的实验效果不太理想，为便于寻找不同模型所学特征之间的关联关系，故对不同的模型的 bad case 进行分析，为方便分析，借助如图 3.1.14 所示的模型。

在交叉验证数据集 Train No.4 上，采用 TextCNN 和 TextRNN 对 Bert_base 的 bad case 进行分析，如图 3.1.15，标红数据表示 TextCNN 和 TextRNN 的 bad case 存在差异的数据。s

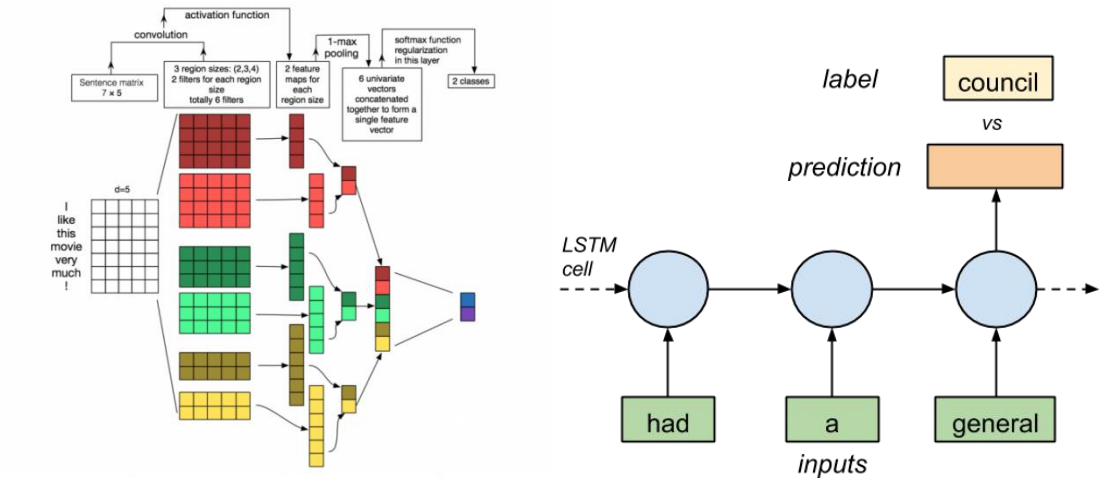

图 3.1.14 数据分析模型结构

Text CNN

二次赴港办理港澳签证需要多少天，工本费是多少，需要带哪些资料，如果在四川办理的港澳通行
没有满十六岁　10　3
上个月在成都市出入境中心办理港澳通行证，现在让人带领，除了给他回执单，还需要其他证明吗
没有钱怎么办？　17　7
现金支付应该也是可以的吧，毕竟拒收现金是违法的鸭<ahref='/n/重庆公安出入境'>@重
今天去日月光广场分理处办理港澳通行证，太便捷高效啦，为我大重庆的出入境管理点赞????　1
之前在武侯区出入境中心办理了护照和港澳台通行证，但是半个月后只收到了港澳台通行证可以领
请问成都居民持旅游港澳通行证，可以办理自助通关吗？　19　10
警官您好！我想请问一下港澳通行证的受理号码怎么查询呢？　0　10
你好?我是宜宾户口?需要续签港澳通行证?请问成都可以办理吗?需要什么手续?具体位置在哪?谢谢
我的港澳通行证7月到期，请问现在还可以签吗　4　11
能否网上预约办理？　1　11
成都首次办理港澳通行证可以在自助办理机上办理吗？　1　11
你好，旧通行证办过签注，换的新港澳通行证未办签注，可以网上申请签注或在签注自助机办理吗
你们哪个直接上班呀　0　12
没问题，可以　8　13
本人目前在广州，无法回户口所在地成都办理澳门签注。已有港澳通行证，是否可以委托他人帮我
已有港澳通行证，请问办理港澳签注需要哪些资料，在哪里办理，谢谢。　1　15
港澳通行证即将于3月到期，现在想申请换领，请问是否需要携带原通行证？　10　15
请问我都江堰的户口，用临时身份证能否在成都办理港澳台通行证　11　17

Text RNN

没有满十六岁　10　3
你好，我10.16号办的通行证。22号我查询，怎么还是在审核中呢！给我的回执单上面写的是25号之前就可以寄送
没有钱怎么办？　16　7
今天去日月光广场分理处办理港澳通行证，太便捷高效啦，为我大重庆的出入境管理点赞????　10　8
之前在武侯区出入境中心办理了护照和港澳台通行证，但是半个月后只收到了港澳台通行证可以领取的通知短信
请问成都居民持旅游港澳通行证，可以办理自助通关吗？　19　10
我预约3月20日办普通护照和港澳签证，不知道是否成功？再预约不起了。　4　10
警官您好！我想请问一下港澳通行证的受理号码怎么查询呢？　6　10
如何查询？谢谢。　8　10
你好?我是宜宾户口?需要续签港澳通行证?请问成都可以办理吗?需要什么手续?具体位置在哪?谢谢　1　11
我的港澳通行证7月到期，请问现在还可以签吗　10　11
能否网上预约办理？　18　11
成都首次办理港澳通行证可以在自助办理机上办理吗？　1　11
你好，旧通行证办过签注，换的新港澳通行证未办签注，可以网上申请签注或在签注自助机办理吗　19　11
你好，我是德阳地区的户口，港澳通行证是在德阳办的，现在有成都的居住证，请问可以用原来的通行证在成都办
你们哪个直接上班呀　10　12
没问题，可以　20　13
本人目前在广州，无法回户口所在地成都办理澳门签注。已有港澳通行证，是否可以委托他人帮我办理澳门签注。
港澳通行证即将于3月到期，现在想申请换领，请问是否需要携带原通行证？　10　15
我要去香港开心生病需要我们指定　18　16
请问我都江堰的户口，用临时身份证能否在成都办理港澳台通行证　11　17

图 3.1.15 TextCNN 与 TextRnn bad case 对比

结论 1：对比两个模型的 bad case，发现 TextCNN 和 TextRNN 所学知识存在互补性。

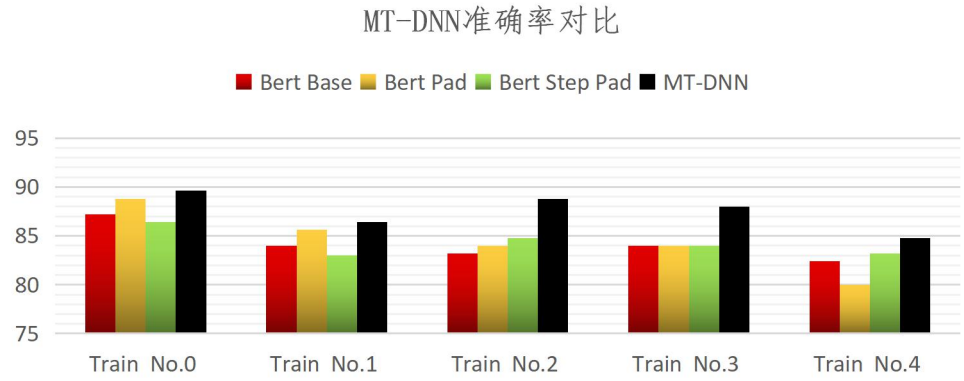

图 3.1.16 MT-DNN 准确率对比

结果 2：根据结论 1，故选取特征存在互补的模型作为 teacher net 来指导 MT-DNN 中 student net 的学习(具体模型配置见方案设计)。MT-DNN 与方案一中的三个模型 Bert_base、Bert_pad、Bert_sep 的效果对比，如图 3.1.16 所示。

MT-DNN 与三个模型在交叉验证数据集上的平均准确率对比如图 3.1.17 所示。

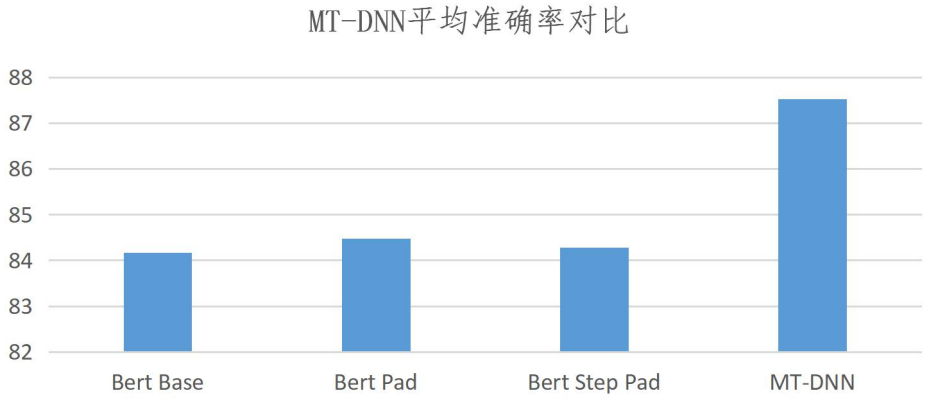

图 3.1.17 MT-DNN 平均准确率对比

结论 1：在交叉验证数据集上，采用 MT-DNN 后的模型效果，较之前的模型效果有较大提升，将所学特征存在互补性的模型作为 teacher net 可以实现指导 student net 的学习，将知识注入 student net 中。

3. 数据问题

在方案的实施过程中，对比模型的 bad case 发现，原训练数据（660 条训练数据）与测试数据（220 条测试数据）存在标注不一致的情况，表

3.1.12 为类别与标签的对应。

表 3.1.12 类别标签及其 ID 列表

标签	ID	标签 ID	ID	标签	ID
XX 什么意思	0	感谢语过渡语	8	探亲和逗留	16
办理地址	1	领证方式咨询	9	是否可以用其他证件	17
办理费用	2	其他意图	10	委托办理	18
办理年龄要求	3	如何办理	11	旅游	19
办理时长	4	上班时间	12	否	20
代领签证咨询	5	是	13	邮寄签证费用	21
逗留有效期及频次	6	是否可以加急	14	委托办理	18
费用缴纳方式	7	首次非首次提供办理资料	15		

模型的预测与训练数据的对比，第 8 类的数据如图 3.1.18 所示：

图 3.1.18 第 8 类测试数据、预测错误数据与训练数据对比

第十类数据对比如图 3.1.19 所示：

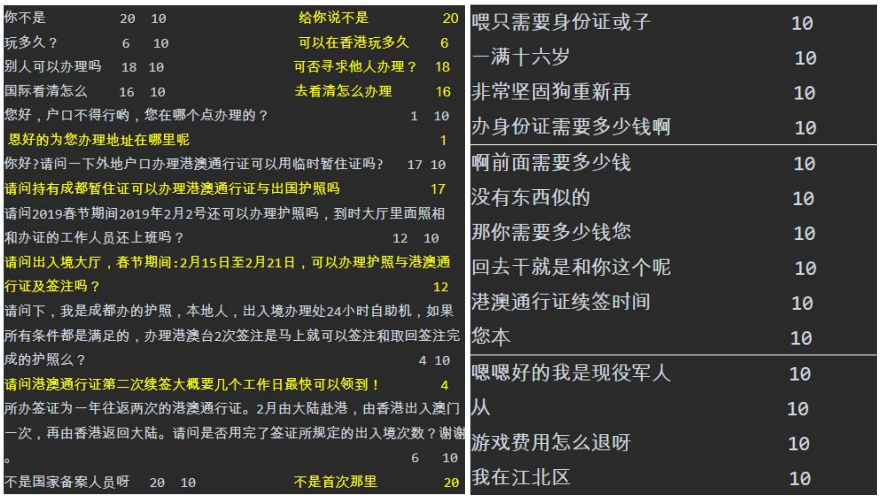

图 3.1.19 第 10 类测试数据、预测错误数据与训练数据对比

结论：训练和测试数据中，第 8 类和第 10 类数据存在标注不一致的问题，去掉两类数据前后的模型效果对比如表 3.1.13 所示。

表 3.1.13 标注错误数据删除前后效果对比

模型		去掉前	去掉后
Student net:	Bert_pad	0.849	0.9076
Teacher net:	RCNN	0.868	0.9239
	Bert_base	0.849	0.9130
	TextCNNbert	0.854	0.9022
	TextRNNbert	0.849	0.9076

4. 结论

MT-DNN 在小数据集场景下，可以取得非常好的效果，将 MTDNN 与公司模型的效果对比，如表 3.1.14：

表 3.1.14 马上模型与重师小数据集交付模型效果对比

场景	测试集	数据量	马上模型	重师模型			
				Bert_pad	Bert_base	TextCNNbert	TextRNNbert
小样本场景	测试集 A（样本均衡）	197	0.8205	0.8769	0.8769	0.8718	0.8769
	测试集 B（样本不均衡）	7476	0.7871	0.7912	0.7923	0.7945	0.7821

由表可知，MT-DNN 在测试集 A 和 B 上都超过了公司的模型。

（二）公开数据集 fine tune 模型

本章节将从方案设计，实验与分析以及结论三个方面加以具体阐述。

1. 方案设计

在小数据集任务中，MT-DNN 算法已经被验证是非常有效的方法，故在公开数据集 fine tune 任务中，方案依然采用小数据集中的基础的 Bert_base 模型以及 MT-DNN 算法。

2. 实验与分析

① 数据集

该任务中，数据集采用公开的意图识别评测数据集 ATIS，该数据集共包含 26 个类，训练数据 4478 条，验证数据 500 条，测试数据 893 条。

② 实验设计

将 MD-DNN 算法与基础的 Bert_base 模型的效果进行比较，实验中的变量因素为模型的不同，具体硬件设置如下表，模型的参数参照小数据集中同模型的参数设置，具体配置信息如表 3.2.1：

表 3.2.1 硬件及其模型配置

配置	参数
CPU	E5-2630 V4
GPU	Titan XP 12G
优化器	Lookahead
损失函数	CrossEntropyLoss

③ 实验结果与分析

结果 1:MT-DNN 和 Bert_base 在公开数据集 ATIS 上的效果如表 3.2.2：

表 3.2.2 MT-DNN 与 Bert_base 在 ATIS 上效果对比

Model		Accuracy
Bert_base		0.981
MT-DNN	Student:Bert_pad	0.984
	RCNN	0.981
	Bert_base	0.983
	TextCNNbert	0.983
	TextRNNbert	0.984

3. 结论

在公开数据集 ATIS 上，MT-DNN 效果较 Bert_base 有所提升，也已达到预期要求。

（三）大数据集 fine tune 模型

本章节将从方案设计，实验与分析以及结论三个方面加以具体阐述。

1. 方案设计

在大数据集场景下，为提升模型精度和模型响应时间等问题，共制定五套解决方案，方案一：单模型 Fine Tune、方案二：MT-DNN 模型、方案三：apex 训练加速以及方案四：两阶段 fine tune 与模型集成，以下将具体阐述。

1）方案一：单模型 fine tune

① 背景和意义

虽然在小数据集与公开数据集中，已经验证了诸多模型的效果，但在大数据集中，依然需要选取更加适合大数据集任务，效果较好的基础模型，故对比 Bert，Albert 等模型 fine tune 的效果。

分类任务(v1版本,正式版)

模型	Score	参数	AFQMC	TNEWS'	IFLYTEK'	CMNLI	WSC	CSL
BERT-base	68.77%	108M	73.70%	56.58%	60.29%	79.69%	62.0%	80.36%
BERT-wwm-ext	70.47%	108M	74.07%	56.84%	59.43%	80.42%	61.1%	80.63%
ERNIE-base	70.55%	108M	73.83%	58.33%	58.96%	80.29%	60.8%	79.1%
RoBERTa-large	72.63%	334M	74.02%	57.86%	62.55%	81.70%	72.7%	81.36%
XLNet-mid	68.65%	200M	70.50%	56.24%	57.85%	81.25%	64.4%	81.26%
ALBERT-xxlarge	71.04%	235M	75.6%	**59.46%**	62.89%	**83.14%**	61.54%	**83.63%**
ALBERT-xlarge	68.91%	60M	69.96%	57.36%	59.50%	81.13%	64.34%	81.20%
ALBERT-large	67.91%	18M	74%	55.16%	57.00%	78.77%	62.24%	80.30%
ALBERT-base	67.44%	12M	72.55%	55.06%	56.58%	77.58%	64.34%	78.5%
ALBERT-tiny	61.92%	**4M**	69.92%	53.35%	48.71%	70.61%	58.5%	74.56%
RoBERTa-wwm-ext	71.72%	108M	74.04%	56.94%	60.31%	80.51%	67.8%	81.0%
RoBERTa-wwm-large	**73.45%**	330M	**76.55%**	58.61%	**62.98%**	82.12%	**74.6%**	82.13%

Model	URL	Score	CoLA	SST-2	MRPC	STS-B	QQP	MNLI-m	MNLI-mm	QNLI	RTE	WNLI	AX
ALBERT + DAAF + NAS		90.6	73.5	97.2	94.0/92.0	93.0/92.4	76.1/91.0	91.6	91.3	97.5	91.7	94.5	51.2
ERNIE		90.4	74.4	97.5	93.5/91.4	93.0/92.6	75.2/90.9	91.4	91.0	96.6	90.9	94.5	51.7
StructBERT		90.3	75.3	97.1	93.9/91.9	93.0/92.5	74.8/91.0	90.9	90.7	96.4	90.2	94.5	49.1
T5		90.3	71.6	97.5	92.8/90.4	93.1/92.8	75.1/90.6	92.2	91.9	96.9	92.8	94.5	53.1
MT-DNN-SMART		89.9	69.5	97.5	93.7/91.6	92.9/92.5	73.9/90.2	91.0	90.8	99.2	89.7	94.5	50.2
Funnel-Transformer (Ensemble B10-10-10H1024)		89.7	70.5	97.5	93.4/91.2	92.6/92.3	75.4/90.7	91.4	91.1	95.8	90.0	94.5	51.6
ELECTRA-Large + Standard Tricks		89.4	71.7	97.1	93.1/90.7	92.9/92.5	75.6/90.8	91.3	90.8	95.8	89.8	91.8	50.7
NEZHA-Large		89.1	69.9	97.3	93.3/91.0	92.4/91.9	74.2/90.6	91.0	90.7	95.7	88.7	93.2	47.9
FreeLB-RoBERTa (ensemble)		88.4	68.0	96.8	93.1/90.8	92.3/92.1	74.8/90.3	91.1	90.7	95.6	88.7	89.0	50.1
HIRE-RoBERTa		88.3	68.6	97.1	93.0/90.7	92.4/92.0	74.3/90.2	90.7	90.4	95.5	87.9	89.0	49.3
RoBERTa		88.1	67.8	96.7	92.3/89.8	92.2/91.9	74.3/90.2	90.8	90.2	95.4	88.2	89.0	48.7
MT-DNN-ensemble		87.6	68.4	96.5	92.7/90.3	91.1/90.7	73.7/89.9	87.9	87.4	96.0	86.3	89.0	42.8
GLUE Human Baselines		87.1	66.4	97.8	86.3/80.8	92.7/92.6	59.5/80.4	92.0	92.8	91.2	93.6	95.9	-
Snorkel MeTaL		83.2	63.8	96.2	91.5/88.5	90.1/89.7	73.1/89.9	87.6	87.2	93.9	80.9	65.1	39.9
XLM (English only)		83.1	62.9	95.6	90.7/87.1	88.8/88.2	73.2/89.8	89.1	88.5	94.0	76.0	71.9	44.7
SemBERT		82.9	62.3	94.6	91.2/88.3	87.8/86.7	72.8/89.8	87.6	86.3	94.6	84.5	65.1	42.4

图 3.3.1 GLUE 与 CLUE 模型测评排行

BERT 被提出后不久，便陆续推出了各种 Bert 模型的变种模型，如 Albert，Electra，Roberta 等。为找到较 Bert 更加适合大数据集任务的预训练模型，如图 3.3.1 所示，参考中文任务基准评测(CLUE)的排行榜[17]，以及通用语言理解评估(GLUE)排行[18]。

综合选取性能优于 Bert 的多个模型，包括 Bert_base_chinese、Bert_base_chinese-wwm、Ernie_base、Electra-base、Albert_xxlarge、XLNet_base_chinese 和 Roberta_base-chinese，在大数据集任务中验证模型的效果，并进行选择。

② 方案介绍

本方案中，采用的 Bert_base 即为小数据集任务中构建的模型。对于不同模型的 fine tune 方式，以 XLNet 为例，则只需将 Bert_base 模型中的 BERT 预训练模型替换为 XLNet 预训练模型即可，记为 XLNet_base。Albert 等其他模型，与 XLNet 思路类似，将 Bert_base 中的 BERT 预训练模型，用 Albert 系列的预训练模型进行替换即可。

2）方案二：MT-DNN 模型

① 方案介绍

该任务中的 MT-DNN 算法也与小数据集任务中的 MT-DNN 相同，但考虑到大数据集因数据量远比小数据集大，而硬件条件有限，故在该场景下，将 Teacher Net 设置为 TextCNN、TextRNN 两个模型；Student Net 依然为 Bert_pad 模型。

如图 3.3.2，将 MT-DNN 训练得到的三个模型 TextCNNbert、TextRNNbert 和 Bert_base 采用“投票”的策略，将三者进行集成，当三个模型的标签预测各不相同时，则最终更加信任 Bert_base 的预测，因为在多次的实验中，Bert_base 作为 Student Net 的效果要优于其他两个模型。

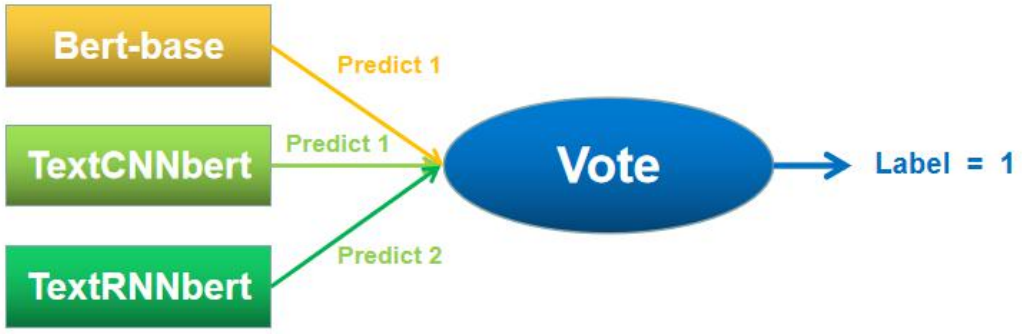

图 3.3.2 模型集成“投票”策略

3）方案三：apex 训练加速

① 背景和意义

训练大数据集时存在训练模型效率较低的问题，该问题一方面是由于训练模型的数据规模较大，而另一方面则是由于一些模型的参数规模大于 BERT。同时参数规模过大的模型会导致模型的显存占用率变高，只能选择

更小的 batch size 训练，从而进一步导致梯度不稳定影响性能。为解决该问题，本方案采用 apex[14]对模型的训练进行加速。

apex 是由 NVIDIA 提出的可以完美支持 Pytorch，将模型的大部分操作改用 Float16 执行，Float16 可以有效的减少模型的显存占用率并提高 GPU 的推理速度，进而提升训练模型的效率。

② 方案介绍

由于 apex 采用了 Float16 的方案，其对模型推理的精度影响未知，因此本方案拟对各个 BERT 模型在使用 apex 前后的性能、显存占用率，训练效率进行对比。

4）方案四：两阶段 fine tune 与模型集成

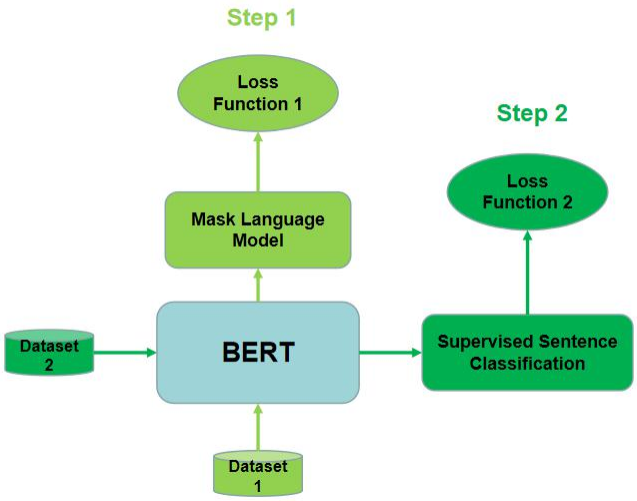

图 3.3.3 两阶段 fine tune 模型示意

① 背景和意义

由于 BERT 是使用维基百科的语料训练的，其数据分布与意图识别任务的数据分布可能并不一致。为了使 BERT 的数据分布更接近下游任务，本方案拟采用两阶段 fine tune 的方法对 bert 在意图识别上进行优化两阶段 fine tune，其中第一阶段采用 Mask Language Model，Mask Language Model 通过随机遮盖一些词（替换为统一标记），然后预测这些被遮盖的词，来使每个词的表示包含其上下文信息，Mask Language Model 对 BERT 模型进行 fine tune，在将 fine tune 后的模型作为下游分类（这里为分类任务）的输入，再次 fine tune。由于 mask language model 是一个自学习模型，在实验中我们采用训练语料来做第一阶段 fine tune。

② 方案介绍

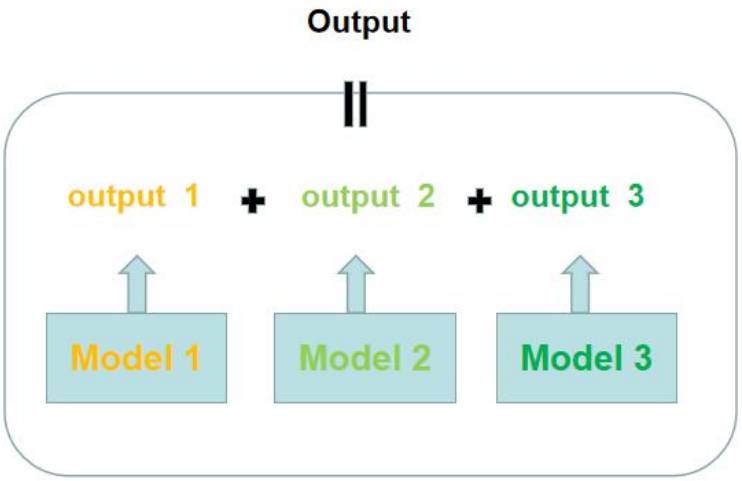

图 3.3.4 模型集成策略示意

两阶段的 fine tune 的架构图如图 3.3.3 所示，第一阶段是采用 Mask Language Model 对 bert-base-chinese 预训练模型进行 fine tune；第二阶段是将第一阶段 fine tune 后的模型作为分类模型的输入，再次 fine tune，从而提升模型的效果。

为了提升意图分类的效果，两阶段 fine tune 后的 BERT 模型会和其他 BERT 的变种模型进行集成，集成方式如图 3.3.4 所示。在集成模型中将每个模型的输出相加，最终选择得分最高的模型作为最后的预测结果。

2. 实验与分析

1）方案一：单模型 fine tune 实验

① 数据集

将公司提供的 132595 条，325 个类别的数据作为训练数据，公司提供的 1000 条测试数据作为验证数据。

② 实验设计

在以上数据集上，采用 Bert 等不同模型进行训练，比较模型效果，变量因素为模型及模型的参数，实验具体配置如表 3.3.1 所示。

表 3.3.1 硬件及模型配置

配置	参数
CPU	Intel Core i9-9820X
GPU	2080Ti 12G
优化器	Torch.AdamW
损失函数	CrossEntropyLoss

③ 参数设置

模型超参数包括：Epoch、Lr(Learning rate)、Batchsize、Max_length，具体参数设置如表 3.3.2：

表 3.3.2 实验参数设置

Parameters	Range
Epoch	{10，20，25，30}
Lr	{2e-4，2e-5，2e-6}
Batchsize	{64，128，256}
Max_length	{30}

④ 实验结果

表 3.3.3 Bert_base 与 XLNet 等模型效果对比

Model	Accuracy
Bert_base_chinese	0.872
Bert_base_chinese_wwm	0.867
Ernie_base_chinese	0.867
Electra-base_chinese	0.863
Albert_xxlarge	0.861
XLNet_base_chinese	0.858
Roberta_base-chinese	0.852

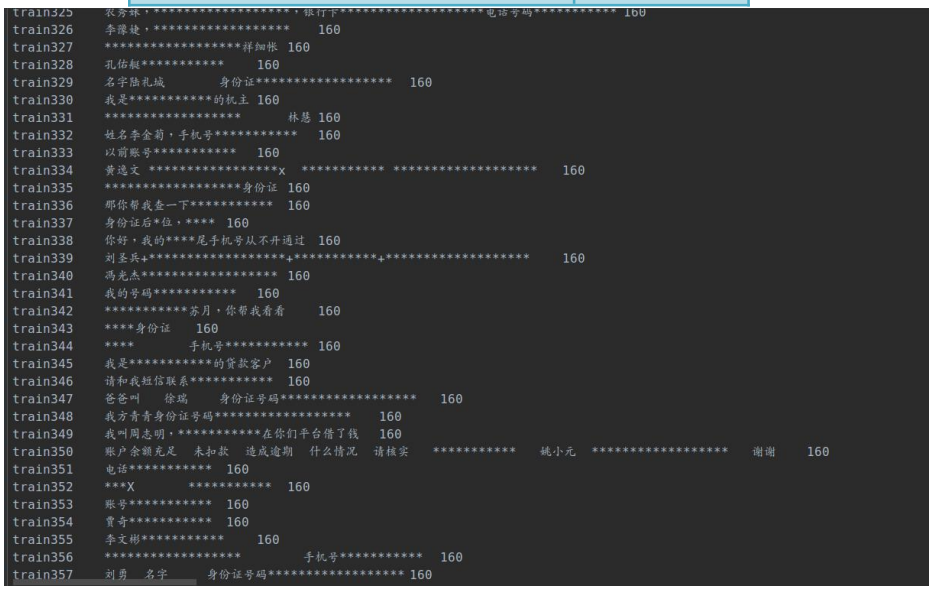

图 3.3.5 带“*”号数据示例

结果 1：针对以上抽样训练集和 1000 条的验证集，验证 Bert、XLNet、Albert 等模型的效果，效果对比如表 3.3.3 所示该结果得分析见第三章第三节。

结果 2：观察数据集发现，训练数据和测试错误数据中，包含大量带“*”与制表符等符号的数据，如图 3.3.5 所示。

“*”号和制表符等符号可能影响了模型的学习，为验证该问题，将数据集中的“*”号去掉，构成新的数据集，重新训练，得到模型效果如表 3.3.4，可知，去掉“*”号后，反而会导致模型效果的下降，因为星号具有其语义(人名，证件号，手机号等)，去掉星号后会导致句子的语义不完整，进而使得模型性能下降。

表 3.3.4 带“*”号数据集与去“*”号数据集效果对比

Model	Accuracy	
	未去“*”号	去“*”号
Bert	0.7929	0.7917

2）方案二：MT-DNN 模型实验

① 数据集

训练集是根据公司提供的 132595 条，325 个类别的训练数据，按每个类别的 80%进行随机抽样而得到，验证集包含两个，第一个为训练数据的剩余数据作为验证集.第二个为公司提供的 1000 条测试样例数据，将公司提供的 1000 条测试样例数据作为验证集。

② 实验设计

在同一份数据集下，将 MT-DNN 用于大数据集任务，并将结果与 Bert_base 进行对比，实验的唯一变量因素为模型及其参数的不同。具体配置如下表 3.3.5：

表 3.3.5 硬件及模型配置

配置	参数
CPU	Intel Core i9-9820X
GPU	2080Ti 12G
优化器	Lookahead
损失函数	CrossEntropyLoss

③ 参数设置

模型超参数包括：Epoch、Lr(Learning rate)、Batchsize、Max_length，具体参数设置如表 3.3.6 所示。

表 3.3.6 实验参数设置

Parameters	Range
Epoch	{5, 24}
Lr	{5e-5, 6e-5}
Batchsize	{16, 32}
Max_length	{70}

④ 实验结果

结果 1：在八二分得到的验证集中，模型的实验效果对比如表 3.3.7：

表 3.3.7 八二分验证集效果

Model	Bert_base	MTDNN
		Student net:Bert_pad
Accuracy	0.7929	0.836

结果 2:在 1000 条测试样例验证集中,模型的实验效果对比如表 3.3.8 所示：

表 3.3.8 1000 条测试样例的模型效果

Model	Bert_base	MTDNN		
		Bert_pad	TextCNNbert	TextRNNbert
Accuracy	0.840	0.854	0.831	0.849

在以上八二分数据集上，MTDNN 较 Bert_base_chinese 效果都有所提升，但在公司的真实测试数据上，效果基本没有提升，如表 3.3.9 所示，具体原因分析见第三章第 3 节。

表 3.3.9 MT-DNN 与马上模型对比

场景	数据量	马上模型		重师模型(80%)			
		马上(13 万)	马上(80%)	Bert_base	Bert_pad	TextCNNbert	TextCNNbert
大样本场景	审核测试集:1w	0.88	0.8645	0.8478	0.8489	0.8187	0.84
	原始测试集:1.4w	0.844	0.8315	0.8123	0.8178	0.7906	0.8078
	验证集:3.3w	0.808	0.7926	0.7862	0.7844	0.7355	0.7616

表 3.3.10 MT-DNN 模型集成效果对比

Model	Bert_base	MTDNN			MT-DNN 集成
		Bert_pad	TextCNNbert	TextRNNbert	Triple model
Accuracy	0.840	0.854	0.831	0.849	0.865

结果 2：MT-DNN 模型集成后，在 1000 条测试样例验证集中与之前模型效果对比如表 3.3.10 所示。

结论：在该场景下，对 MT-DNN 训练得到的模型进行集成，可以提升模型效果，而在公司测试集上效果依然不佳，具体原因分析见第三章第 3 节。

3）方案三：apex 训练加速实验

① 数据集

为节省实验验证时间，将公司提供的 132595 条，325 个类别的训练数

据，按每个类别的 10%数据抽样得到训练数据集，同样将公司提供的 1000 条测试数据作为验证数据集。

注：前期交付的模型在 1000 条上的测试效果与在公司测试集上的测试效果保持一致性，故后续实验依然以 1000 条数据集上的测试效果作为参考标准。

② 实验设计

该组实验分为两部分，第一部分将 Albert 系列模型与其他模型进行对比，验证效果，该部分实验中唯一的变量因素为模型及其训练参数的不同。

第二部分实验将 Albert 系列模型与 Bert_base 采用 apex 训练，并将模型效果与未使用 apex 的效果进行对比，该部分实验的唯一变量因素为是否使用 apex。

③ 参数设置

模型超参数包括：Epoch、Lr(Learning rate)、Batchsize、Max_length，具体参数设置如表 3.3.11：

表 3.3.11 实验参数设置

Parameters	Range
Epoch	{10, 20, 25}
Lr	{2e-4, 2e-5, 2e-6}
Batchsize	{64, 128, 256}
Max_length	{30}

④ 实验结果

表 3.3.12 apex 优化训练前后训练时间与所占显存对比

Model	Time(s)		Cuda(M)		时间节省	显存节省
	used	unused	used	unused		
Albert-base	12	22	1917	2343	46%	19%
Bert-base-chinese	19	27	3803	3867	30%	2%

结果 1：采用 apex 前后的训练测试时间，及所占用 GPU 内存对比如表 3.3.12 所示。

表 3.3.13 apex 优化训练前后准确率对比

Model	Accuracy	
	used	unused
Albert-xxlarge	0.759	0.758
Albert-base	0.731	0.608
Bert-base-chinese	0.756	0.749

结果 2：采用 apex 前后的模型准确率对比如表 3.3.13 所示。

结论 1：(1)apex 可以节省运算时间，同时节省 GPU 内存;(2)apex 可以进一步提升模型的准确率。

结果 3：在 132595 条 325 个类别的大数据集作为训练集，公司提供的 1000 条作为测试数据的数据集上，将 Albert-xxlarge + apex 与之前得到的最好的模型比较，结果如下表 3.3.14 所示。

结论 2：虽然在 1000 条抽样测试样例数据上，模型效果提升较大，但在公司真实测试数据上，提升不明显，具体原因分析见第三章第 3 节。

表 3.3.14 Albert-xxlarge + apex 准确率对比

Times	Model	Accuracy
本次	Albert-xxlarge + apex	0.861
之前 MT-DNN	Bert_pad	0.854
	TextCNNbert	0.849
	TextRNNbert	0.831
之前 Bert_base	Bert_base_chinese	0.852

4）方案四：两阶段 fine tune 与模型集成实验

① 数据集

将公司提供的 132595 条，325 个类别的数据作为训练数据，公司提供的 1000 条测试数据作为验证数据。

② 实验设计

该组实验分为三部分，三部分实验都采用 apex 来优化模型的训练，第一部分因模型集成需不同的模型进行集成，故先训练好多个不同的模型，该实验中唯一的变量因素为模型。

第二部分为两阶段的 fine tune,该部分实验是对 Bert_base_chinese 进行两阶段的 fine tune，并将之前的 bert_base_chinese 的效果与经过两阶段 fine tune 的 bert_base_chinese 效果进行对比，该实验中唯一变量因素为是否经过两阶段的 fine tune。

表 3.3.15 硬件及模型配置

配置	参数
CPU	Intel Core i9-9820X
GPU	2080Ti 12G
优化器	Lookahead
损失函数	CrossEntropyLoss

第三部分为将训练好的模型，进行集成，该组实验中唯一的变量因素为集成模型采用的子模型不同，具体硬件及模型配置如下表 3.3.15 所示。

③ 参数设置

模型超参数包括：Epoch、Lr(Learning rate)、Batchsize、Max_length，具体参数设置如表 3.3.16：

表 3.3.16 实验参数设置

Parameters	Range
Epoch	{10，20，25}
Lr	{2e-4，2e-5，2e-6}
Batchsize	{64}
Max_length	{30}

④ 实验结果

结果 1：两阶段的 fine tune 前后，效果对比如表 3.3.17 所示。

结论 1：采用两阶段，第一阶段为 Mask Language Model，可以让模型更好的拟合训练数据的分布，故第二阶段的 Fine Tune 可以进一步提升 BERT 模型在 1000 条样例测试数据上的预测准确率。

表 3.3.17 两阶段 fine tune 效果对比

Model	Accuracy
Bert_base_chinese	0.8719
MLM-Bert-base-chinese	0.8739

结果 2：将三个模型的输出，直接相加后的输出，作为最终的预测。采用不同模型进行集成，具体结果如表 3.3.18 所示。

表 3.3.18 不同模型集成效果对比

Model 集成方式	Accuracy
Bert_wwm + Electra + MLM-Bert	0.8859
Bert_wwm + Electra + Albert	0.8789
Bert_wwm + Electra + Bert_base	0.8879
1.3 * Bert_wwm + Electra + Albert	0.8819
Bert_base + Electra + Albert	0.8759
Bert_base + Electra + MLM-Bert	0.8839
1.3 * Bert_base + Electra + Albert	0.8789
Bert_wwm + Albert + MLM-Bert	0.8758
Bert_wwm + Ernie + MLM-Bert	0.8719
XLNet + Electra + MLM-Bert	0.8668

结果 3：对以上模型的最优组合 Bert_wwm + Electra + MLM-Bert 计算该集成模型的响应时间，具体如下：

（1）Bert_wwm + Electra + MLM-Bert 在 GPU 上单条数据的响应时间为 **30 毫秒**；

（2）Bert_wwm + Electra + MLM-Bert 在 CPU 上单条数据的响应时间为 **142 毫秒**。

所以，将模型的输出直接相加作为模型最终的预测，进行集成，可以进一步提升模型的效果；两阶段的 fine tune 模型可以进一步提升集成模型的效果；虽然 Electra 模型本身效果一般，但该模型所学特征与其他模型存在差异，故将其进行集成，可有效提升效果，若替换掉该模型，则效果会下降；采用 apex 后，模型的效率得到有效提升，apex 会提高模型推理的效率。

3. 结论

表 3.3.19 大数据集场景下马上模型与重师模型效果对比

场景	数据量	马上模型		重师模型					
		集成模型	单模型	Bert_base	Bert_base_wwm	Electra	MLMbert	Albert	集成模型
大样本场景	审核测试集:1w	0.8804	0.867	0.8697	0.8666	0.8634	0.868	0.8628	0.8781
	原始测试集:1.4w	0.844	-	0.8353	0.8347	0.8296	0.8326	0.8312	0.8431
	验证集:3.3w	0.808	-	0.7961	0.7978	0.7906	0.7953	0.7889	0.8111

交付模型与公司模型效果对比如表 3.3.19 所示。

重师模型在审核测试集 1w 和原始测试集 1.4w 上表现稍差于马上模型，在验证集 3.3w 上略优于马上模型，同时重师模型的 CPU 响应时间为 142 毫秒，响应时间较快。

二维码是什么？	13	扫二维码是什么啊	86
今天到期.	13	今天到时间了	43
身份证过期了可以吗	13	身份证过期了可以申请贷款吗？	100
我改了日期	13	我改了还款日期，	71
积分签到	13	app积分在哪签到	190
安逸花没开通是什么意思	13	提示安逸花没开通 60 提示安逸花没开通	60
****元*个月	13	****元*个月多少	40
还款未.	13	还款未显示 还款未出账单	15
逾期一天有什么后果，会上征信吗？	58	请问。逾期一天不会上报征信吧	118
我的借款都还清了，怎么还差十元？	45	为什么我的借款都还清了，过了一阵他提示还要我还十元	16
哦，我提现了还没到呢	13	我提现了，我的可用额度没有变	160
为什么不能提现了？	0	现在不能提现了？	70
经我司评估，暂时不支持提现和消费	6	经我司综合评估，您暂不支持提现和消费是怎么回事儿	0
我的借款都没通过，怎么就要还款？	60	请问我的借款都已经提前结清，怎么还会有账单呢	16
如何提现才能到账	13	如何提现成功！	79
马上金融APP里面 有职享花可以申请吗	24	职享花可以申请吗？	100
这样查看我还有多少钱没换	134	请问我还有多少钱还没还。	121

图 3.3.6 训练噪声数据示例

关于重师在审核测试集 1w 和原始测试集 1.4w 上表现不好，因只能根据 1000 条测试样例数据进行模型调优（第一次模型测试采用八二分数据，效果不好，后考虑将 1000 条样例数据作为验证集），最终原因分析如下：

(1) 训练数据存在噪声。根据模型的分类结果，对训练数据进行分析发现训练数据中存在噪声，如图 3.3.6 所示为一些噪声数据的示例，左侧为训练数据集中的句子，右侧为找到的类似表达且标签不同的句子。

(2) 训练数据与真实测试数据存在分布不一致的问题。将方案一的模型在 1000 条测试样例上测试，在 167 条预测错误数据中，存在较多训练数据与 1000 条样例数据不一致的情况，部分错误数据示例如表 3.3.20 所示：

表 3.3.20 训练与测试样例数据分布不一致示例

验证集原句	验证集原句标签	训练集原句标签	训练集原句
我买的东西什么时候发货	13	16	购买的东西什么时候发货
购物不发货	13	16	一直没有发货
怎么我的进不去	13	185	怎么进不去?
不提供借款	13	229	不能借款嘛?
借不	13	229	借款借不?
哪里有分期门店	13	318	在哪里分期
现在借款怎么借不出来?	21	229	为什么借款现在不行了
货款千不通过,明天来	21	229	货款为什么老是不通过?
为什么老借不到	21	229	为啥总借不出来钱
电话	222	121	电话
微支付付款有利息吗	147	164	用微支付用不用利息
微支付利息是多少	147	164	微支付怎么计息
微支付的利息怎么算	147	164	微支付消费怎么算

表 3.3.21 单模型准确率与模型大小对比

Model	Accuracy	模型参数量(M)
Bert_base_chinese	0.872	108
Bert_base_chinese_wwm	0.867	108
Ernie_base_chinese	0.867	108
Electra-base_chinese	0.863	102
Albert_xxlarge	0.861	235
XLNet_base_chinese	0.858	200
Roberta_base-chinese	0.852	110

在 1000 条样例数据中，存在训练与样例数据不一致的情况，测试样例数据为真实测试集抽样所得，故真实的测试数据也存在此类数据分布不一致的问题。

(3) 因数据存在以上噪音和分布不均的问题，故对方案一中的模型进行分析，存在如表 3.3.21 所示的规律，当模型越大，模型效果反而下降。这是因为当模型越大时，模型会很容易学到数据的分布，当数据包含大量噪声时，则模型会优先学得数据的噪声，论文[19]也说明了这个现象。

(4) 由方案二 MT-DNN 的实验可知，如表 3.3.22 所示，在八二分验证集上，MT-DNN 较 BERT 可以取得 4 个百分点的提升，而在真实测试场景中，MT-DNN 几乎没有获得提升。MT-DNN 由两个 teacher net 与一个 student net 组成，因为每个模型都学到了不同的数据分布，而训练数据与真实测试数据存在分布不一致，故导致了 MT-DNN 在真实测试数据中提升不明显。

表 3.3.22 不同场景 MT-DNN 与 Bert_base 效果对比

场景	Model	Accuracy
八二分验证集	Bert_base	0.7929
	MT-DNN(Student net):Bert_pad	0.836
1000 条测试样例	Bert_base	0.840
	MT-DNN(Student net):Bert_pad	0.854
真实测试数据(1w)	Bert_base	0.8478
	MT-DNN(Student net):Bert_pad	0.8489

(5) 在方案四中，采用两阶段的 fine tune，在 13 万的训练集，1000 条测试样例数据中，两阶段 fine tune 较 BERT 可以有 0.2%的提升，如表 3.3.23 所示，而在真实的三个测试集中，两阶段 fine tune 后的效果反而要差。

表 3.3.23 不同场景集成模型及其子模型效果对比

场景	Model	Accuracy
1000 条测试样例	Bert_base_chinese	0.8719
	MLM-Bert-base-chinese	0.8739
	Bert_base_chinese_wwm	0.867
	Electra-base_chinese	0.863
	集成模型(Bert_wwm + Electra + MLM-Bert)	0.8859
真实测试数据(1w)	Bert_base_chinese	0.8697
	MLM-Bert-base-chinese	0.868
	Bert_base_chinese_wwm	0.8666
	Electra-base_chinese	0.8634
	集成模型(Bert_wwm + Electra + MLM-Bert)	0.8781

表 3.3.24 集成模型中的子模型在测试样例与 1w 测试集上效果对比

Model	1000 条测试样例	真实测试数据(1w)	误差
Bert_base_chinese	0.872	0.870	0.002
MLM-Bert-base-chinese	0.874	0.868	0.006
Bert_base_chinese_wwm	0.867	0.866	0.001
Electra-base_chinese	0.863	0.863	0

如表 3.3.24 所示，在真实测试集中，集成模型中的各个子模型的效果与其在 1000 条测试样例结果误差如表所示,而 MLM-Bert-base-chinese 出现了较大幅度的下降，从而导致了集成模型整体效果的下降。也是因为第一阶段的 fine tune 使得模型所学数据分布更加符合训练数据的分布，而真实测试数据存在分布不一致，从而导致的效果变差。

四、 项目交付情况

目前,本项目已交付内容的清单如下:

序号	交付内容
1	小数据集模型，并完成测试
2	公开意图识别评测集模型，并完成测试
3	大数据集模型，并完成测试
4	小数据集，公开意图识别数据集任务工程源码
5	大数据集任务工程源码
6	小，公开意图识别数据集任务实验环境 Docker
7	大数据集任务实验环境 Docker
8	项目进展报告两份

五、 关于结题时间的说明

按合同要求，本项目原计划于 2020 年 2 月 9 日完成结题，但因疫情影响(不可抗力),参与该项目的师生于 2020 年 6 月 20 日后才继续返校工作，经双方(甲方:马上金融，乙方:重师方)商定，将结题时间顺延至今，在此，进行说明。

参考文献

[1] Devlin J, Chang M W, Lee K, et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding[C]. 2019.

[2] Yang Z, Dai Z, Yang Y, et al. XLNet: Generalized Autoregressive Pretraining for Language Understanding[C]. 2019.

[3] Liu Y, Ott M, Goyal N , et al. RoBERTa: A Robustly Optimized BERT Pretraining Approach[J].2019.

[4] Lan Z, Chen M, Goodman S, et al. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations[C]. 2020.

[5] Clark K, Luong M T, Le Q V, et al. ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators[C]. 2020.

[6] Sun C, Qiu X, Xu Y, et al. How to Fine-Tune BERT for Text Classification?[C]. 2019.

[7] He T, Zhang Z, Zhang H, et al. Bag of Tricks for Image Classification with Convolutional Neural Networks[C]. 2019.

[8] Lin T Y, Goyal P, Girshick R, et al. Focal Loss for Dense Object Detection[J]. 2017.

[9] Mikolov T, Chen K, Corrado G,et al. Efficient Estimation of Word Representations in Vector Space[C]. 2013.

[10] Xiaodong L,Yu W,Jianshu J,et al.The Microsoft Toolkit of Multi-Task Deep Neural Networks for Natural Language Understanding[C]. 2020.

[11] Kim Y. Convolutional Neural Networks for Sentence Classification[C]. 2014.

[12] Liu P, Qiu X, Huang X. Recurrent Neural Network for Text Classification with Multi-Task Learning[C]. 2016.

[13] Girshick R, Donahue J, Darrell T, et al.Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation[C]. 2014.

[14] https://github.com/NVIDIA/apex.

[15] Vaswani A , Shazeer N , Parmar N , et al. Attention Is All You Need[C]. 2017.

[16] https://www.cluebenchmarks.com/classification.html.

[17] https://github.com/CLUEbenchmark/CLUE.

[18] https://gluebenchmark.com/leaderboard.

[19] Han B, Yao Q, Yu X , et al. Co-teaching: Robust Training of Deep Neural Networks with Extremely Noisy Labels[C]. 2018.

[20] Li X , Meng Y , Sun X , et al. Is Word Segmentation Necessary for Deep Learning of Chinese Representations?[C]. 2019.

[21] https://github.com/huggingface/transformers