RoBERTa: A Robustly Optimized BERT Pretraining Approach

摘要

RoBERTa 针对 BERT 预训练不足的问题进行改进，通过更长时间的优化训练和动态 Masking 等策略提高模型的性能，在下游性能上获得了一定的提升。本文聚焦于医疗诊疗对话意图识别任务，对比 RoBERTa 和 BERT 两种主流预训练语言模型在下游任务的能力，同时提出了滑动窗口机制和 PGD 对抗训练两项改进。在天池医疗诊疗对话意图识别挑战赛数据集上的实验表明，这两项改进显著提升了模型的意图识别性能，其中 BERT 在引入改进后达到最高分数 0.8475。虽然 RoBERTa 在基础性能上优于 BERT，但其在实际业务场景中的适配性提升有限。

关键词： RoBERTa；BERT；意图识别

1 引言

近年来，预训练语言模型在自然语言处理任务中取得了显著进展，特别是 BERT（Bidirectional Encoder Representations from Transformers）的提出，大幅提高了多种下游任务的性能 [3]。然而，随着模型的不断发展，其优化策略和结构设计成为了进一步提升模型性能的关键。RoBERTa（A Robustly Optimized BERT Pretraining Approach）作为对 BERT 的改进模型，通过更大规模数据、更长的训练时间以及优化的训练策略，实现了对 BERT 性能的全面超越 [6]。

在医疗领域，诊疗对话意图识别是一项极具挑战性且具有重要实际意义的任务。随着大语言模型（LLM，Large Language Model）被广泛应用于智能问诊、医疗助理等领域，构建能够准确理解患者意图的系统显得尤为重要 [7]。诊疗对话中，用户表达的意图可能包括询问病因、请求建议、预约就诊等，这些意图的识别直接关系到医疗服务的响应质量和用户体验。

目前，许多基于 LLM 的系统通过引入 agent 进行业务逻辑处理。这些 agents 在医疗场景中通常需要根据患者的意图动态调整生成的内容或调用专业模型。例如：精准选择生成策略：识别患者是否需要简单的病因说明、医疗建议，或更复杂的诊疗方案，如果能够准确识别意图，系统就可以选择合适的 prompt，从而提升生成内容的专业性和针对性；专业模型调用：当患者的需求涉及复杂病理或多学科联合诊疗时，系统可以根据意图识别结果调用专门训练的细分领域模型，例如针对影像诊断的模型或药物推荐的模型；减少误解与风险：在医疗场景中，错误的意图识别可能导致严重后果，例如将病情紧急的用户意图错误分类为普通咨询，因此，提高意图识别的准确性对于增强系统的安全性和可靠性至关重要。

然而，医疗对话中存在大量专业术语、多变的语言表达，以及长文本上下文依赖等问题，使得传统的文本分类方法难以在意图识别任务中取得满意的效果。基于此情况，本研究以天池医疗诊疗对话意图识别挑战赛[1]为评测方式，选择 RoBERTa 和 BERT 两种主流语言模型进行性能对比，并通过滑动窗口捕获上文信息和 PGD（Projected Gradient Descent）对抗训练两种改进措施提升模型在该中文医疗问诊意图识别任务上的表现。

本文的意义在于：探索 RoBERTa 和 BERT 在中文医疗意图识别任务中的性能差异，为实际应用提供参考依据。通过改进措施，验证增强模型效果的可能性，丰富领域内的技术实践。为后续研究提供方法借鉴，推动自然语言处理技术在医疗领域的深入应用。

2 相关工作

2.1 BERT

BERT 是 Google 于 2018 年提出的一种预训练语言模型。它基于 Transformer 架构 [8]，通过引入双向编码器，突破了传统单向语言模型在语境理解上的局限。BERT 的核心思想是充分利用上下文信息，即在预测当前单词时同时参考其左右两侧的词语，而不是像传统的自回归语言模型那样，仅利用单向的上下文。这一设计使得 BERT 能够更好地捕捉句子级别和段落级别的语义关系，极大地提升了模型在多种下游任务中的表现。

在技术实现上，BERT 采用了两种预训练任务：Masked Language Model（MLM）和 Next Sentence Prediction（NSP）。MLM 是 BERT 的关键创新之一，在训练过程中，模型会随机遮盖输入句子中的一部分单词，然后通过上下文预测这些单词的真实值。这一方法不仅能够有效避免传统语言模型中的信息泄露问题，还能够训练模型更好地理解文本中的深层语义结构。NSP 的设计初衷是为了增强模型对句间关系的理解能力，通过引入句对分类任务，判断两段句子是否相邻，从而帮助模型更好地处理诸如问答匹配和文本推理等需要跨句分析的任务。然而，后续研究发现 NSP 的实际贡献有限，在某些场景下甚至可能带来负面影响，这也为后续改进模型如 RoBERTa 的设计提供了启发。

BERT 的提出对自然语言处理领域的影响是革命性的。在发布之初，BERT 在多个自然语言处理基准测试（例如 GLUE 和 SQuAD）上刷新了当时的最佳成绩，这得益于其强大的语义建模能力和广泛的适用性。BERT 的成功不仅推动了预训练语言模型的研究热潮，也为众多下游应用提供了全新的解决方案，如文本分类、机器翻译、阅读理解和信息检索等。

2.2 RoBERTa

RoBERTa 是 Facebook AI 提出的一种对 BERT 的改进模型。RoBERTa 的目标是在 BERT 的基础上，通过优化训练策略和数据使用方式进一步挖掘模型的潜力。尽管 BERT 已经展现了卓越的性能，但其训练方法中的一些设计选择被认为可能限制了模型的上限表现，例如训练时间较短、数据集规模较小、引入了可能无效的任务，以及掩码生成策略的局限性。RoBERTa 针对这些问题进行了系统性改进，最终在多个自然语言处理任务上超越了 BERT，并成为预训练语言模型领域的重要里程碑之一。

[1] https://tianchi.aliyun.com/competition/entrance/532044

2.3 PGD 对抗训练

PGD 对抗训练是一种常用的方法，用于提升深度学习模型的鲁棒性和泛化能力。在自然语言处理任务中，输入文本通常包含噪声和不确定性，例如拼写错误、同义词替换、词序颠倒等，这些因素可能显著影响模型的预测性能。PGD 对抗训练通过在训练过程中加入精心设计的对抗样本，迫使模型对微小扰动具有更强的适应性，从而提高其在真实应用场景中的稳健性。PGD 的核心思想是基于梯度生成对抗样本并对模型进行更新。具体来说，在每次训练迭代中，PGD 会沿着输入样本的梯度方向生成扰动，使输入朝着最能迷惑模型的方向偏移。

在 NLP 任务中，PGD 的应用相较于计算机视觉领域更具挑战性，因为文本是离散的，直接对输入进行微小数值扰动可能会破坏语义或语法。因此，PGD 通常被扩展为在词向量空间中操作，而非直接修改离散的文本。例如，在 BERT 或 RoBERTa 模型中，PGD 会对词嵌入（word embeddings）施加扰动，而不改变实际的输入单词。这种方法既能够保留文本的语义一致性，又能够有效生成具有攻击性的对抗样本。

在本研究中，PGD 被用于提升模型在医疗诊疗对话意图识别任务上的鲁棒性。这一任务中，患者的表达可能存在同义词替换或轻微的语法错误，传统模型在这种情况下容易出现误判。通过引入 PGD 对抗训练，模型能够在面对这类干扰时仍然准确识别患者意图。例如，在训练过程中加入针对高频词或关键语义词的对抗扰动，迫使模型学习到更加鲁棒的语义表示。

3 本文方法

RoBERTa 对 BERT 的主要改进体现在四个方面：首先，移除了 NSP 任务，简化了训练目标并提升了下游任务的性能；其次，引入动态 Masking 策略，在每次训练中动态生成掩码，增加数据多样性；第三，扩展了预训练数据集规模，从 BERT 的 16GB 增加至 160GB，并显著延长了训练时间，使模型能够充分学习更大规模语料的语义特征；最后，采用更大的训练批次和全长度输入序列，从预训练过程的效率和效果两个方面进一步优化了模型性能。

RoBERTa 移除了 BERT 中的 NSP 任务，尽管 NSP 的初衷是帮助模型理解句间关系，但后续研究表明，其实际对大多数下游任务的提升作用有限，甚至可能因为增加无关任务的训练目标而对模型的语义建模能力产生干扰。通过取消 NSP，RoBERTa 将注意力集中在更关键的语言建模任务上，从而提升了整体性能。RoBERTa 引入了动态 Masking 策略以取代 BERT 中的静态 Masking。在 BERT 的训练过程中，掩码在数据预处理中已经固定生成，这意味着同一个句子在所有训练轮次中都会使用相同的遮盖模式，从而限制了数据多样性。而 RoBERTa 则通过在每个训练批次中动态生成掩码，使得相同的句子可以在不同的训练轮次中拥有多种遮盖组合，这一策略有效提高了模型对上下文信息的学习能力，并增强了其泛化性能。

数据集规模和训练时间的扩展是 RoBERTa 相较 BERT 的另一大优势。BERT 的预训练数据集包括 BOOKCORPUS 和 WIKIPEDIA，总计约 16GB，而 RoBERTa 在此基础上引入了多个大规模开放语料，如 CC-News、OpenWebText 和 Stories，总数据规模扩大至 160GB。此外，RoBERTa 在训练时间上也显著延长，相较于 BERT 的 100 万步，RoBERTa 的训练步数可以扩展至 50 万至 300 万步不等。这种“更多数据、更长时间”的策略极大地挖掘了模型潜力，使得 RoBERTa 在多个任务中显著超越 BERT。

RoBERTa 还针对输入长度的优化进行了尝试。在 BERT 的训练中，通常会在初期使用

较短的输入序列，随后逐步增加至 512 个 Token。而 RoBERTa 则从一开始就使用完整长度的输入序列，这一变化不仅加速了模型对长距离依赖关系的学习，还避免了切换序列长度可能带来的额外调整成本。

RoBERTa 在保持 BERT 原有架构和 MLM 任务的基础上，通过移除 NSP、动态 Masking、更大数据集和更长训练时间等优化策略，充分展现了预训练语言模型的性能上限。在多个任务的基准测试中，RoBERTa 的表现均优于 BERT，证明了这些改进的有效性。RoBERTa 的成功不仅为后续研究提供了优化方向，也进一步巩固了预训练语言模型在自然语言处理领域的核心地位。

4 复现细节

4.1 评测标准

为公平探索 RoBERTa 在中文具体应用场景下的表现，本文采用阿里云天池的医疗诊疗对话意图识别挑战赛作为评测标准，识别医患对话的意图在在线问诊中发挥着重要的作用，可以帮助达到更好的诊疗效果。对话意图识别任务共定义了 16 类对话意图，标注方式采用句子级标注，对话意图的预定义类别定义如表 1。评测开放训练集数据 1,824 条，验证集数据 616 条，测试集数据 612 条，每条数据包含多条问答句子 $[1, 2, 4, 5, 9]$。将模型充分训练后，对测试集的问题进行推理并提交到平台上得到对应分数，满分为 1.0000[2]。

表 1. 对话意图类别分类表

中文类别	**English Category**
提问-症状	Request-Symptom
告知-症状	Inform-Symptom
提问-病因	Request-Etiology
告知-病因	Inform-Etiology
提问-基本信息	Request-Basic_Information
告知-基本信息	Inform-Basic_Information
提问-已有检查和治疗	Request-Existing_Examination_and_Treatment
告知-已有检查和治疗	Inform-Existing_Examination_and_Treatment
提问-用药建议	Request-Drug_Recommendation
告知-用药建议	Inform-Drug_Recommendation
提问-就医建议	Request-Medical_Advice
告知-就医建议	Inform-Medical_Advice
提问-注意事项	Request-Precautions
告知-注意事项	Inform-Precautions
诊断	Diagnose
其他	Other

[2]https://tianchi.aliyun.com/competition/entrance/532044/information

4.2 与已有开源代码对比

本文基于开源代码实现基础的 BERT 意图识别功能，如图 1，主要是将 BERT 最后的全连接层的维度跟实际任务进行匹配；在此基础上，额外加入开源 PGD 对抗训练代码，如图 2，具体应用训练的时候会进行扰动，让模型迷惑并逐渐适应，从而提高模型的鲁棒性。经过对基础 BERT 在验证集的错误答案具体分析，有相当一部分的对话是因为单句话本身具有不确定性，需要结合上文的问题进行分析，因此，通过自行创新的滑动窗口机制增加上文信息进行训练和推理，如图 3，同时，加入的特殊字符 [SEP] 作为对话拼接分隔符，实验表明该方法极大的提升了对话意图识别的准确率。由于采用的医疗诊断意图识别为中文题目，而原始 BERT 对中文的支持性不好，本文使用 HuggingFace 提供的 bert-base-chinese 和 chinese-roberta-wwm-ext 预训练模型，HF 格式的预训练模型能轻松使用 pytorch 和 transformers 库进行调用训练。

```
34  class Model(torch.nn.Module):
35      def __init__(self):
36          super().__init__()
37          self.bert = BertModel.from_pretrained(model_path)
38          for param in self.bert.parameters():
39              param.requires_grad = True
40          hidden_size = self.bert.config.hidden_size
41          self.linear = torch.nn.Linear(hidden_size, len(act2id))  # 根据目标类别调整输出层维度
42
43      def forward(self, input_ids, attention_mask, token_type_ids):
44          out = self.bert(
45              input_ids=input_ids,
46              attention_mask=attention_mask,
47              token_type_ids=token_type_ids
48          )
49          out = self.linear(out.last_hidden_state[:, 0])  # 获取CLS向量
50          return out
```

图 1. 模型全连接层的修改

```python
class PGD(object):

    def __init__(self, model, emb_name, epsilon=1., alpha=0.3):
        # emb_name这个参数要换成你模型中embedding的参数名
        self.model = model
        self.emb_name = emb_name
        self.epsilon = epsilon
        self.alpha = alpha
        self.emb_backup = {}
        self.grad_backup = {}

    def attack(self, is_first_attack=False):
        for name, param in self.model.named_parameters():
            if param.requires_grad and self.emb_name in name:
                if is_first_attack:
                    self.emb_backup[name] = param.data.clone()
                norm = torch.norm(param.grad)
                if norm != 0:
                    r_at = self.alpha * param.grad / norm
                    param.data.add_(r_at)
                    param.data = self.project(name, param.data, self.epsilon)

    def restore(self):
        for name, param in self.model.named_parameters():
            if param.requires_grad and self.emb_name in name:
                assert name in self.emb_backup
                param.data = self.emb_backup[name]
        self.emb_backup = {}

    def project(self, param_name, param_data, epsilon):
        r = param_data - self.emb_backup[param_name]
        if torch.norm(r) > epsilon:
            r = epsilon * r / torch.norm(r)
        return self.emb_backup[param_name] + r

    def backup_grad(self):
        for name, param in self.model.named_parameters():
            if param.requires_grad and param.grad is not None:
                self.grad_backup[name] = param.grad.clone()

    def restore_grad(self):
        for name, param in self.model.named_parameters():
            if param.requires_grad and param.grad is not None:
                param.grad = self.grad_backup[name]
```

图 2. PGD 对抗训练

```
        self.samples = []
        for dialogue_id, sentences in data.items():
            context = []
            for sentence in sentences:
                speaker = sentence["speaker"]
                text = f"{speaker}: {sentence['sentence']}"  # 当前句子
                label = act2id[sentence["dialogue_act"]]

                # 拼接上下文
                input_text = "[SEP] ".join(context[-context_window:] + [text])
                self.samples.append((input_text, label))

                # 更新上下文
                context.append(text)
```

图 3. 滑动窗口

4.3 实验环境搭建

实验环境如表 2所示，推荐使用 Anaconda 进行环境配置，用户可以根据自己的服务器配置调整批量大小和训练轮次等参数。用户直接运行训练文件即可，其中包含数据预处理、网络参数、保存模型权重等功能。实验参数设置：训练轮数大小为 5、句子最大长度为 256、批次大小为 64、滑动窗口大小为 3，其中取验证集效果最好的轮次作为最佳权重进行评测。

表 2. 实验环境配置表

类型	具体配置
CPU	Intel(R) Xeon(R) Gold 6330 CPU @ 2.00GHz
GPU	NVIDIA GeForce RTX 4090
Python 版本	3.10.14
CUDA 版本	12.4
Pytorch 版本	2.2.0

4.4 创新点

本报告的创新点主要有两个：

1. 为应对实际医疗问诊的复杂语境情况，使用滑动窗口捕获上文信息辅助进行意图识别。
2. 由于实际对话有较多的口语化词汇，这使得样本相较于许多预训练数据的复杂度都相对较高，为了提高模型的鲁棒性，加入了 PGD 对抗训练，实现更好的效果。

5 实验结果分析

实验结果如表 3所示，通过表格分析可知，RoBERTa 对于 BERT 在基础性能上确实更好，而创新的滑动窗口通过引入上文信息辅助判断对话意图无论是对 BERT 还是 RoBERTa 都有比较大的提升，这也印证了数据是影响模型性能的最重要因素之一；引入 PGD 对抗训练

进一步增强了滑动窗口的效果，但两个创新点的改进在最终效果上 RoBERTa 都不如 BERT，这说明 RoBERTa 在实际业务的适用性相较于 BERT 并没有很好的提升。另外，截止本文撰写时，在已报名的 1594 名队伍中，本文的方法排名第 10，如图 4所示。

表 3. 实验结果

版本	分数
BERT	0.8165
BERT + context window	0.8422
BERT + context window + PGD	**0.8475**
RoBERTa	0.8182
RoBERTa + context window	0.8410
RoBERTa + context window + PGD	0.8434

图 4. 比赛排名

6 总结与展望

本文围绕中文医疗诊疗对话意图识别任务，探索了 RoBERTa 和 BERT 两种预训练语言模型的性能差异，并提出了两项关键改进：滑动窗口机制和 PGD 对抗训练。滑动窗口有效捕获了长文本中的上下文信息，显著提升了模型的意图识别能力；PGD 对抗训练通过引入对抗扰动，增强了模型在噪声数据中的鲁棒性。在实验中，BERT 和 RoBERTa 在引入改进措施后均取得了较大的性能提升，验证了这些方法的有效性。最终结果表明，RoBERTa 在基础性能上略优于 BERT，但在实际业务场景中的适用性提升并不显著，可能与任务数据的特点和预训练模型的特定设计有关。

尽管本文的研究取得了一定进展，但仍存在一些不足：数据依赖性强：医疗诊疗对话数据的标注质量和规模直接影响模型性能，而实际数据可能存在噪声或偏差，限制了改进方法的适用范围。RoBERTa 的潜力未充分挖掘：尽管 RoBERTa 在理论上具备更强的预训练性能，但其在本文实验中的表现未能显著优于 BERT，可能需要进一步优化模型参数或调整任务设计。计算资源消耗较高：滑动窗口和对抗训练的方法在一定程度上增加了计算成本，未来在资源有限的场景下需要平衡性能与效率的矛盾。

针对上述问题，未来可以从以下几个方向展开进一步研究：多模态数据的引入：结合患者的医疗记录、影像数据或穿戴设备监测信息，与对话文本共同构成多模态数据，提升模型的决策能力。模型优化：针对 RoBERTa 和 BERT 的实际表现，可以尝试调整超参数设置，或引入更大规模的中文医疗语料进行专属预训练，以增强模型的适配性。高效算法设计：进一步优化滑动窗口和对抗训练的实现方式，减少计算资源消耗，例如使用更高效的对抗样本生成方法或自适应的窗口切分策略。增强领域知识的融入：通过知识蒸馏或外部医学知识库的引入，丰富模型对医疗领域专业术语和隐含语义的理解能力，从而提高意图识别的准确性。

参考文献

[1] Wei Chen, Zhiwei Li, Hongyi Fang, Qianyuan Yao, Cheng Zhong, Jianye Hao, Qi Zhang, Xuanjing Huang, Zhongyu Wei, et al. A benchmark for automatic medical consultation system: Frameworks, tasks and datasets. *arXiv preprint arXiv:2204.08997*, 2022.

[2] Wei Chen, Cheng Zhong, Jiajie Peng, and Zhongyu Wei. Dxformer: A decoupled automatic diagnostic system based on decoder-encoder transformer with dense symptom representations. *arXiv preprint arXiv:2205.03755*, 2022.

[3] Jacob Devlin. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[4] K. Liao, Q. Liu, Z. Wei, B. Peng, Q. Chen, W. Sun, and X. Huang. Task-oriented dialogue system for automatic disease diagnosis via hierarchical reinforcement learning. *arXiv preprint arXiv:2004.14254*, 2020.

[5] X Lin, X He, Q. Chen, H. Tou, and T. Chen. Enhancing dialogue symptom diagnosis with global attention and symptom graph. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019.

[6] Yinhan Liu. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 364, 2019.

[7] Dimitrios P Panagoulias, Maria Virvou, and George A Tsihrintzis. Evaluating llm–generated multimodal diagnosis from medical images and symptom analysis. *arXiv preprint arXiv:2402.01730*, 2024.

[8] A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

[9] Z. Wei, Q. Liu, B. Peng, H. Tou, and X Dai. Task-oriented dialogue system for automatic diagnosis. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 2018.