《自然语言处理》课程报告

首尾截断策略与对抗微调结合的影评情感分类研究

2025 年 12 月 20 日

摘　要

随着预训练语言模型在自然语言处理领域的广泛应用，利用深度学习架构提升情感分类的精度与稳健性已成为研究重点。然而，在处理 IMDB 影评等复杂文本时，仍面临长文本信息遗失、语义极性转折剧烈以及模型泛化能力不足等核心挑战。针对上述问题，本报告提出了一种首尾截断策略与对抗微调结合的影评情感分类方法，主要包括：

(1) 深入挖掘影评文体“先铺垫、后总结”的结构特征，提出了启发式的“首尾截断（Head-Tail Truncation）”策略，通过对文本起始与末尾片段的双端信息留存，显著提升了模型对评论结论及极性转折的感知能力。

(2) 在微调阶段引入基于快速梯度方法（FGM）的对抗训练机制，通过在嵌入空间注入基于梯度方向的微小扰动 ϵ_{adv}，迫使模型学习更为平滑的决策边界，显著提升了系统在复杂语境下的鲁棒性。

实验结果表明，采用“128+382”首尾截断方案使 BERT-Base 模型准确率较基准提升了 1.31%，且对抗训练机制成功实现了判别性能与稳健性的协同增强。通过结合具备解耦注意力机制的 DeBERTa-v3-Large 架构与组合优化策略，最终在 IMDB 测试集上取得了 96.60% 的最优分类准确率。相关研究验证了先进预训练架构配合针对性微调策略在处理高难度情感倾向识别任务中的性能与工程应用价值。

关键词： 情感分析；BERT；首尾截断；对抗训练（FGM）；DeBERTa-v3；稳健性优化

目录

1 引言

1.1 研究背景与研究动机

文本情感分析作为自然语言处理领域的核心任务，旨在通过计算手段自动识别并定量化文本中的主观情感倾向。在电影工业体系中，针对 IMDB 等主流平台的影评分析具有显著的商业与学术价值：一方面，它能协助制片方获取实时的观众反馈以优化决策；另一方面，影评中蕴含的复杂语义特征是推荐系统实现精准分发的关键输入。本项目选取的 IMDB 数据集是该领域公认的基准测试集 [1]，包含了 50,000 条极具代表性的英文影评。该数据集的挑战性在于其文本长度波动剧烈，部分长篇评论远超常规模型的处理极限（512 Token），且文本中普遍存在反讽、隐晦转折等深度语义逻辑，这要求模型必须具备强大的长距离上下文捕捉能力。

1.2 预训练范式的演进与 BERT 架构

自然语言处理领域长期面临标注数据匮乏与模型泛化能力不足的矛盾。BERT (Bidirectional Encoder Representations from Transformers) 的提出，标志着 NLP 研究范式由传统的“针对特定任务手工设计架构”向“大规模自监督预训练 + 下游任务通用微调 (Pre-training + Fine-tuning)”的根本性转变 [2]。这一范式确立了“预训练模型作为语言基础设施”的地位，极大地降低了下游任务的训练门槛。

在架构层面，BERT 彻底摒弃了长短时记忆网络 (LSTM) 等循环神经网络结构，转而完全基于 Transformer 的编码器构建。得益于多头自注意力机制，BERT 突破了 RNN 必须按时序串行计算的限制，不仅实现了训练过程的高效并行化，更具备了直接捕捉序列中长距离依赖关系的能力。

与 GPT 等单向自回归模型不同，BERT 的核心创新在于其深层双向表征能力。通过引入掩码语言模型（Masked Language Model, MLM）任务，模型通过随机遮蔽输入序列中的部分 Token 并尝试利用上下文进行还原，这种“完形填空”式的训练迫使模型从双向维度融合语义信息，从而学习到全向语言表征。在此基础上，针对篇章级逻辑理解，BERT 还引入了下一句预测（Next Sentence Prediction, NSP）任务，通过判断两个句子序列的连续性，使模型具备了捕获句子间逻辑关联与结构信息的能力。

这种基于上下文的动态表征学习，有效解决了传统静态词向量（如 Word2Vec）无法处理多义词的固有缺陷。在 IMDB 情感分析等任务中，BERT 能够根据语境精确区分词汇在反讽、转折或特定搭配下的细微语义变化，并利用其篇章感知能力理解长篇影评中的逻辑流向，显著提升了模型对复杂句法结构和深层语义逻辑的理解深度。

1.3 进阶架构优化与 DeBERTa 机制

随着自然语言处理领域预训练技术的不断演进，DeBERTa (Decoding-enhanced BERT with disentangled attention) 通过对注意力机制的底层重构，代表了当前判别式预训练

模型的顶尖水平 [3]。相较于 BERT 和 RoBERTa 等前代模型，DeBERTa-v3 在架构设计上解决了传统位置编码策略中的信息耦合问题，显著提升了模型对长文本和复杂语义的建模能力。

传统 BERT 模型采用将内容嵌入与位置嵌入直接相加的方式来构建输入向量。这种做法虽然简洁，但在数学上导致了内容特征与位置特征在注意力计算前的过早融合，从而限制了注意力机制对相对位置关系的捕捉精度。针对这一瓶颈，DeBERTa 引入了创新的解耦注意力机制。该机制通过两组独立的向量分别表示 Token 的内容和相对位置，并在自注意力计算层将它们解耦。具体而言，注意力权重不再仅仅依赖于内容对内容的相似度，而是被分解为“内容-内容”、“内容-位置”以及“位置-内容”的多项交互得分之和。这种设计使得模型能够更敏锐地感知词汇之间的依赖距离和方向性，从而更精准地捕捉长序列中的位置偏差。

此外，DeBERTa 引入了增强型掩码解码器（Enhanced Mask Decoder, EMD）。考虑到相对位置编码虽然有利于捕捉局部依赖，但在预测被掩码的 Token 时，绝对位置信息依然至关重要。EMD 机制在 Softmax 预测层之前，显式地将绝对位置嵌入注入到解码过程中。这一改进强化了模型对细微语义差别的辨析能力，使其在处理如 IMDB 情感分析等富含逻辑转折、长距离依赖和反语修辞的任务时，展现出比标准 BERT 架构更强的鲁棒性与泛化上限。

1.4 主要工作与贡献

针对上述挑战，本报告提出了一种首尾截断策略与对抗微调结合的影评情感分类方法，主要贡献如下：

- **语义完整性保障**：针对长文本硬截断导致的信息缺失问题，设计了基于影评文体特征的“首尾截断（Head-Tail Truncation）”策略。该策略通过对文本起始与末尾片段的双端信息留存，显著提升了模型对评论结论及极性转折的感知能力。

- **决策稳健性增强**：引入了基于快速梯度方法（FGM）的对抗训练机制。通过在词嵌入层注入方向性微小扰动 ϵ_{adv}，迫使模型在微调过程中平滑决策边界，从而有效提升了系统面对非规范化文本与口语化噪声时的鲁棒性。

- **多架构集成验证**：在基础 BERT 以及具备解耦注意力机制的进阶 DeBERTa 模型上开展了详尽的对比实验。通过多组消融实验验证了所提策略的协同增益效应，并最终在 IMDB 基准任务上取得了 96.60% 的分类准确率。

上述工作的开展不仅在理论层面探讨了预训练语言模型在长序列情感表征上的潜力，也在工程实践层面为构建高稳健性的文本分析系统提供了可行的优化路径。后续章节将对上述各项技术策略的实现细节及实验表现展开深入论述。

2 任务背景与相关工作

2.1 情感分析任务定义与数据集特性

情感分析作为自然语言处理领域的核心研究方向，旨在通过计算手段自动识别、提取并定量化文本中的主观情感倾向。在电影工业体系中，针对 IMDB 等主流影评平台的文本分析具有重要的应用价值：一方面，它能协助制片方获取实时观众反馈；另一方面，情感极性作为文本的高层语义特征，是下游推荐系统实现精准分发的关键输入。IMDB 数据集作为该领域的基准，其独特性在于：文本长度分布极不均匀、反讽修辞广泛存在以及语义转折逻辑复杂。这不仅要求模型具备强大的局部特征提取能力，更需具备深层的全局语义感知能力。

2.2 国内外研究现状与演进

在深度学习发展的早期阶段，情感分类任务主要依赖于以循环神经网络（RNN）[4] 及其变体，如长短期记忆网络（LSTM）[5] 与门控循环单元（GRU）[6]，或卷积神经网络（CNN）[7] 为核心的架构。尽管这些方法在捕捉时序特征或局部关键词方面取得了一定进展，但在处理 IMDB 这种长达数百词且语义逻辑密集的序列时，由于梯度消失问题以及长距离依赖建模能力的先天不足，分类精度往往受限 [8]。这种建模局限性在处理包含深层转折的复杂影评时尤为显著。

近年来，以 Transformer 为基础的预训练语言模型（PLMs）彻底改写了 NLP 任务的基准纪录。自 BERT 问世以来，学术界的研究重点转向了模型架构的深度演进与长序列建模能力的提升。例如，Longformer [9] 与 Big Bird [10] 通过引入稀疏注意力机制，试图将 Transformer 处理文本的上限扩展至 4096 个 Token 以上，从而在根本上解决超长文档的截断问题。与此同时，DeBERTa 系列模型 [3] 通过解耦注意力机制和增强型掩码解码技术，进一步细化了模型对词汇内容与位置信息的理解深度，成为了当前判别式任务中的佼佼者。

此外，针对微调阶段模型稳健性不足及易过拟合的问题，对抗训练成为热门研究课题。除了本项目采用的 FGM 算法外，FreeLB [11] 等进阶算法通过在嵌入空间进行多次梯度迭代扰动，显著提升了模型在面对非规范化表达时的判别稳定性。综上所述，结合针对性的长文本表征策略与稳健的训练机制，已成为当前情感分类领域的研究共识。

2.3 BERT 及其原理

BERT 的出现标志着 NLP 领域进入了表征学习的新阶段。其核心架构与“预训练 + 微调”的二阶段工作流程如图 1 所示。该模型通过大规模语料的无监督预训练，构建了深层次的语义表示能力，其技术优势主要体现在以下两个方面：

- **基于 Transformer 的深度架构**：BERT 彻底弃用了传统的 RNN/CNN 结构，完全基于 Transformer 编码器。利用多头自注意力机制（Multi-Head Self-Attention），

模型能够计算序列中任意两个 Token 之间的关联权重，从而在 $O(1)$ 的路径长度内建模长距离依赖。

- **深度双向表征能力**：与单向语言模型不同，BERT 通过掩码语言模型（MLM）任务，强迫模型在表征当前词时必须同时参考双向上下文信息。这种“完形填空”式的训练方案使其生成的特征向量具备极强的语义判别性。

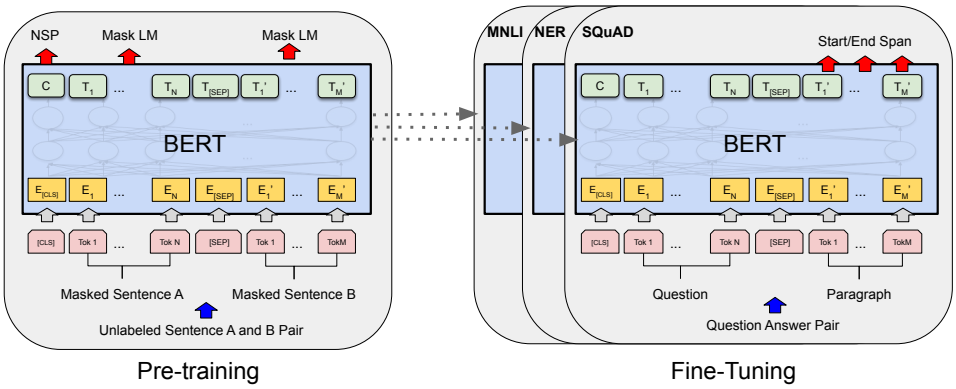

图 1: BERT 预训练与微调范式架构示意图

2.4 进阶变体架构：DeBERTa 机制

随着预训练技术的迭代，DeBERTa [3] 凭借其在 SuperGLUE 等基准测试上的卓越表现，成为当前 NLP 领域探索性能上限的首选模型。相较于 BERT 和 RoBERTa，DeBERTa 的核心贡献在于修正了自注意力机制中处理位置信息的归纳偏置，具体体现在以下两个维度：首先，DeBERTa 提出了解耦注意力机制。传统 BERT 模型将内容嵌入与位置嵌入直接加和（$H_i = C_i + P_i$），这使得后续的注意力计算无法区分语义相似度与位置邻近度。DeBERTa 摒弃了这种耦合策略，主张对内容和相对位置进行独立建模。在计算 Token 之间的注意力分数时，模型将其分解为三个独立的交互项：内容与内容的交互、内容与目标位置的交互、以及内容与源位置的交互。这种机制使得模型能够根据上下文动态调整对“语义”和“距离”的关注权重，从而在处理充满逻辑转折的复杂文本时，构建出更为精准的句法依存图。其次，为了解决相对位置编码导致的绝对位置信息丢失问题，DeBERTa 引入了增强型掩码解码器（EMD）。由于相对位置编码具有平移不变性，仅依赖它可能导致模型难以区分处于不同绝对位置的相同语义结构。EMD 策略通过在解码阶段（即 Softmax 层之前）重新引入绝对位置编码，成功融合了相对位置的局部感知能力与绝对位置的全局定位能力。实验表明，这一机制不仅加速了模型在下游任务微调时的收敛过程，更显著增强了模型在长文本分类与推理任务中的鲁棒性。

3 长文本分类中的双端语义表征策略研究

3.1 IMDB 数据集特性与长文本挑战

本项目采用的 IMDB 电影评论数据集包含总计 50,000 条带标签样本，其正负样本分布极度平衡。然而，通过对数据分布的深度调研发现，影评文本的长度差异显著，约有 10% 的样本长度突破了 BERT 模型原生支持的 512 个 Token 限制。

图 2: IMDB 长篇影评样本示例

针对这一挑战，先观察典型的影评样本，其结构特征如图 2 所示。可以看到，影评中往往包含大量的背景叙述（如示例中对剧情的陈述），而核心情感关键词（如标注出的否定词）常散落在文本的不同位置。更关键的是，作者通常在文章前部进行铺垫，而在末尾才给出最终的情感宣泄或总结性判别。这种“先抑后扬”或“先扬后抑”的结构特征意味着，若直接丢弃文本后半段，模型将丧失对核心情感转折的感知能力，从而导致预测偏差。

3.2 启发式首尾截断（Head-Tail Truncation）策略

针对上述痛点，研究中摒弃了通用的填充与硬截断策略，转而采用一种基于文体学特征的启发式“首尾截断（Head-Tail Truncation）”机制 [12]。该策略的核心逻辑在于通过“双端采样”来兼顾上下文背景与结论性信息。具体的序列拼接过程（Splicing Process）如图 3 所示，其逻辑构造可表示为：

$$T_{input} = [CLS] + \text{Token}_{1\ldots128} + \text{Token}_{-382\ldots-1} + [SEP] \tag{1}$$

在此配置下，文本起始处的 128 个 Token 被保留以抓取背景基调，同时末尾处的 382 个 Token 被提取以锁定情感归宿。如图 3 的拼接示意图所示，原始长文本中部的冗余信息被剔除，关键的双端语义片段被重新组合。总长度连同特殊标识位共计 512 个单位，精确适配了预训练模型的输入张量维度限制。

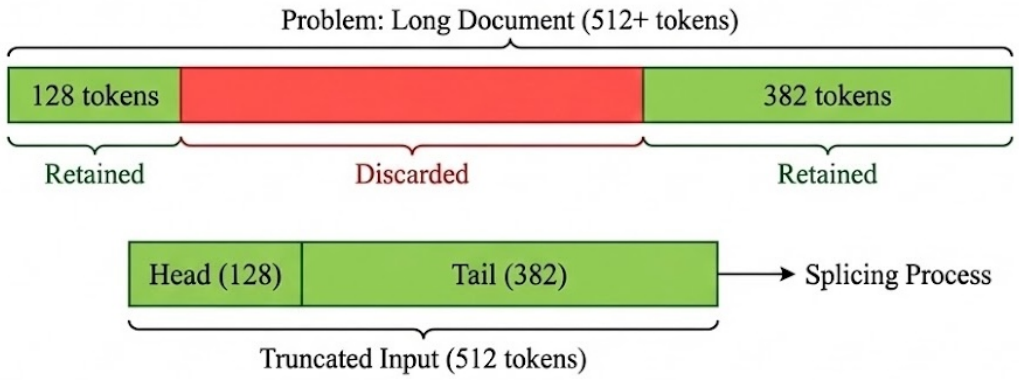

图 3: 首尾截断策略流程图：展示了 Head (128) 与 Tail (382) 信息的提取与拼接过程

通过这种双端语义表征策略，模型在处理超长文档时能够同时观测到文章的起因与结果，从而在分类决策中建立更具全局观的语义关联。实验数据表明，该策略对模型准确率的提升起到了关键的支撑作用。

4 嵌入空间扰动驱动的对抗微调机制研究

在上一部分中，研究通过首尾截断策略优化了长文本的语义表征，确保了输入数据在符合模型限制的前提下保留了核心情感信息。然而，在实际的微调过程中，仅靠优化数据输入尚不足以应对所有挑战。影评文本中广泛存在的口语化噪声、非规范表达以及复杂的语义逻辑，容易导致模型在有限的标注样本上出现过拟合，或使其决策边界变得异常脆弱。为了进一步提升模型在复杂语境下的判别稳健性，本章从训练机制入手，引入基于嵌入空间梯度扰动的对抗训练方案，旨在通过平滑模型决策边界来增强其泛化性能。

4.1 微调流程设计

在经过首尾截断处理后，输入序列被送入预训练的 Transformer 编码器中。按照 BERT 架构的通用范式，提取编码器顶层输出中首位 `[CLS]` 标记对应的隐藏层向量 $h_{cls} \in \mathbb{R}^d$ 作为整段文本的全局语义摘要。对于 BERT-Base 模型，$d = 768$；对于后续对比实验中的 Large 级别模型，$d = 1024$。

该向量随后被输入至下游线性分类层。为了缓解全参数微调过程中的过拟合风险，分类层前引入了 Dropout 随机失活机制：

$$Z = \text{Dropout}(h_{cls})W + b \tag{2}$$

其中 W 与 b 分别代表分类层的权重矩阵与偏置项。模型训练阶段采用交叉熵（Cross-Entropy）损失函数进行端到端的参数优化，使模型在保留通用语言表征能力的同时，适配 IMDB 任务的特定情感极性判别需求。

4.2 对抗训练：FGM 算法原理

为了进一步增强模型对于输入扰动的抗干扰能力，研究中引入了快速梯度方法（Fast Gradient Method, FGM）开展对抗训练 [13]。FGM 的核心逻辑在于模拟推理阶段可能出现的微小噪声，并在词嵌入层（Embedding Layer）寻找能够使损失函数 L 增加最快（即模型最易判错）的方向注入扰动，其原理示意如图 4 所示。

具体数学定义与扰动生成过程如下：

$$g = \nabla_x L(\theta, x, y), \quad \epsilon_{adv} = \epsilon \cdot \frac{g}{\|g\|} \tag{3}$$

其中，x 为输入的词嵌入向量，θ 为当前模型参数，ϵ 是控制扰动强度的超参数。通过这种方式，在训练的每一个步长中均实时生成针对当前模型薄弱环节的对抗样本。

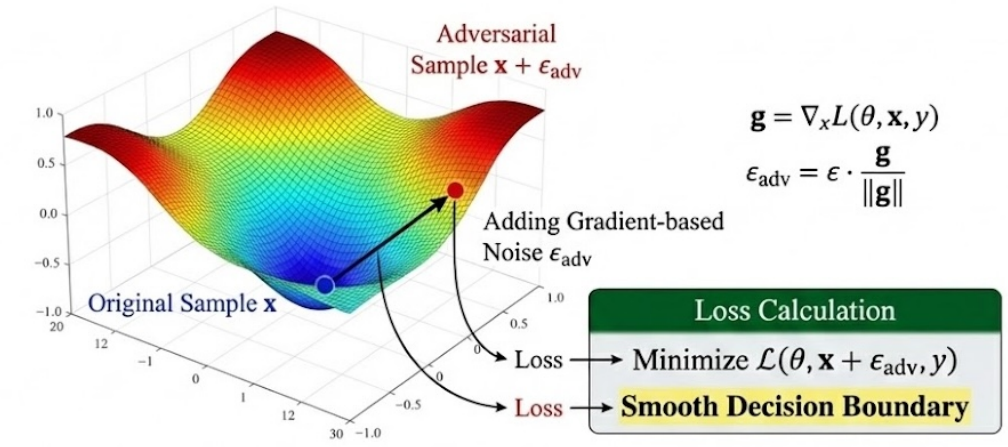

图 4: FGM 对抗训练原理：通过梯度扰动引导模型寻找更平滑的损失平面

4.3 决策边界平滑与稳健性分析

在引入 FGM 机制后，模型不仅需要最小化原始样本的损失，还需在输入扰动邻域内通过最小化对抗风险 $\mathcal{L}(\theta, x + \epsilon_{adv}, y)$ 来保持预测结果的一致性。

这种机制实质上是在嵌入空间中对模型实施了流形约束。如图 4 所示，原始样本沿梯度上升方向移动至对抗点。通过强制要求模型在这些点处仍输出正确标签，促使模型的决策边界向远离样本点的方向移动。这种训练方式有效平滑了模型在特征空间中的预测曲面，使其对于影评中常见的拼写变体或非规范口语化表达具备了更强的稳健性。实验结果显示，这一机制对提升模型在验证集上的最终准确率起到了显著的正向作用。

5 实验分析

5.1 实验基准模型

为了全面评估所提优化策略的有效性与泛化能力，研究构建了由浅入深的实验基准体系，涵盖了不同规模与架构的预训练模型。其架构演进逻辑如图 5 所示：

- **BERT (Base)**：包含 12 层 Transformer 编码器，采用掩码语言模型（MLM）与下一句预测（NSP）作为预训练任务。
- **BERT-Large**：扩展至 24 层深度架构，具备更强的特征表征空间。
- **DeBERTa-v3-Large**：引入了解耦注意力机制（Disentangled Attention），将内容与位置信息独立编码，并采用增强型掩码解码器与替换 Token 检测（RTD）任务进行预训练，代表了当前判别式任务的顶尖水平。

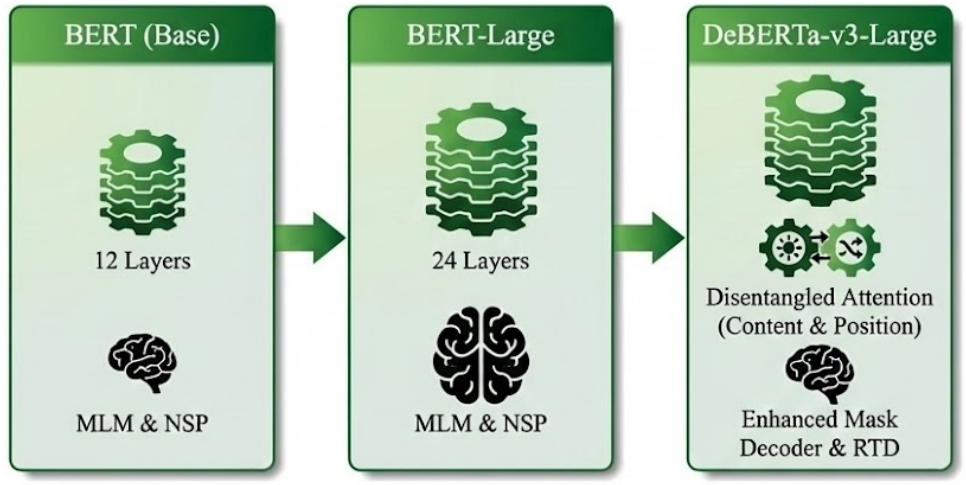

图 5: 实验基准模型架构演进

5.2 BERT (Base) 模型实验分析

针对基础 BERT 模型，本研究系统对比了不同优化策略对情感分类性能的影响。具体实验数据如表 1 所示，提出的综合方案在各项指标上均取得了提升。

表 1: 不同策略在 IMDB 情感分析任务（BERT-Base）上的性能对比

方法	**准确率 (%)**	**Macro F1 (%)**	**ROC AUC (%)**
Baseline (BERT)	93.04	93.04	98.04
+ Truncation	94.35	94.35	**98.64**
+ FGM	94.15	94.15	97.38
+ Combined (Ours)	**94.55**	**94.55**	97.78

数据分析表明，综合策略相比基准模型在准确率上实现了 1.51% 的绝对增益，峰值达到 94.55%。值得注意的是，首尾截断策略（Truncation）通过捕获长距离依赖有效提升了模型性能；而引入 FGM 对抗训练虽然进一步优化了分类准确率，但导致了 ROC AUC 的轻微下降。结合图 4 的损失平面分析可知，对抗训练在通过注入梯度噪声 ϵ_{adv} 强化决策边界平滑性的同时，可能在模型预测确定性与排序精度之间产生一定的性能权衡。

5.3 进阶模型性能分析

为了进一步探索模型性能上限，实验选取了更大规模的 BERT-Large 以及架构更先进的 DeBERTa-v3-Large 进行对比验证，详细性能指标记录于表 2。

表 2: 进阶模型（BERT-Large 与 DeBERTa-v3-Large）性能对比

方法	**准确率 (%)**	**Macro F1 (%)**	**ROC AUC (%)**
Model 1: BERT-Large			
Baseline	94.50	94.50	98.30
+ Truncation	94.67	94.67	98.34
+ FGM	94.88	94.88	98.51
+ Combined (Ours)	**95.02**	**95.02**	**98.58**
Model 2: DeBERTa-v3-Large			
Baseline	95.90	95.90	99.29
+ Truncation	96.46	96.46	**99.33**
+ FGM	96.30	96.30	99.11
+ Combined (Ours)	**96.60**	**96.60**	99.29

实验结果显示，DeBERTa-v3-Large 模型展现了卓越的基础判别性能。在集成双端信息留存与对抗微调策略后，DeBERTa 模型最终达到了 96.60% 的最高准确率。实验数据有力地证明了，随着模型参数规模的扩大及注意机制的优化，配合针对性的长文本处理方案，能够持续突破影评分类任务的性能瓶颈。

6 总结与展望

6.1 总结

本报告针对 IMDB 电影评论情感分析任务中存在的信息遗失与模型不稳定性问题，提出了一种首尾截断策略与对抗微调结合的影评情感分类方法。在数据表征层面，本报告通过深入分析影评文体“先铺垫后总结”的结构特征，实施了启发式的首尾截断策略。实验证明，该策略在保留评论背景的同时，精准锁定了位于篇末的核心情感极性，有效弥补了传统硬截断方案在长文本处理上的先天缺陷。在训练机制方面，本报告引入了基于嵌入空间梯度扰动的对抗训练（FGM）算法。通过在词嵌入层注入微小扰动 ϵ_{adv}，迫使模型在面对非规范化表达或口语化噪声时仍能保持决策的一致性。这一机制不仅显著增强了模型的鲁棒性，更通过平滑损失平面缓解了微调阶段常见的过拟合风险。最后，通过对 BERT-Base、BERT-Large 及 DeBERTa-v3-Large 等多尺度架构的对比验证，本报告揭示了先进预训练模型在集成本报告优化策略后的性能潜力。尤其是具备解耦注意力机制的 DeBERTa 模型，在最终测试中达到了 96.60% 的准确率，成功突破了该任务的性能瓶颈。

6.2 局限性与未来展望

尽管本项目在多个维度上取得了性能提升，但在实验深度与应用广度上仍存在进一步优化的空间。一方面，对于文本中蕴含的极度反讽、隐晦转折等深层语义特征，单一的模型架构仍面临一定的判别压力。未来计划探索引入外部知识图谱或多模态辅助信息，以增强模型对非直观极性的辨识深度。

另一方面，虽然“首尾截断”策略提升了长序列的信息质量，但这种启发式的筛选方法本质上仍会导致部分中间段落语义的断裂。在后续研究中，将考虑引入具备稀疏注意力机制的长文本专用模型（如 Longformer 或 Big Bird），实现对超长文本的全序列无损表征。此外，针对对抗训练带来的训练开销问题，计划尝试更多步迭代的进阶算法（如 FreeLB），旨在通过更细致的梯度寻优进一步平滑模型的决策曲面，从而构建出更具泛化性与鲁棒性的情感分析系统。

A 附录：实现代码

Listing 1: 基于 BERT 的文本分类与对抗训练核心代码

```python
import os
import sys
import numpy as np
import torch
import torch.nn as nn
from dataclasses import dataclass, field
from typing import Optional, Any, Dict, Union
from datasets import load_dataset
from sklearn.metrics import accuracy_score, roc_auc_score, f1_score
from transformers import (
    AutoTokenizer,
    AutoModelForSequenceClassification,
    TrainingArguments,
    Trainer,
    DataCollatorWithPadding,
    EarlyStoppingCallback
)

# --- 1. 配置类 (Train Config) ---
@dataclass
class TrainConfig:
    """
    统一管理训练超参和对抗训练策略
    """
    # 基础模型参数
    model_name: str = "bert-base-uncased"
    max_length: int = 512
    num_labels: int = 2

    # 训练超参
    output_dir: str = "results/bert_imdb"
    learning_rate: float = 2e-5
    batch_size: int = 32
    accu: int = 1
    epochs: int = 12

    # 文本处理与对抗训练配置
    token_first_last: bool = True      # 是否启用 Head+Tail 截断策略
    use_adv_training: bool = True       # 是否启用对抗训练
    adv_method: str = "fgm"             # 对抗方法: FGM
    adv_epsilon: float = 0.05
```

```
    adv_alpha: float = 0.9
    adv_start_epoch: float = 1.0        # 开启对抗训练的起始 Epoch

# 初始化配置与模型
cfg = TrainConfig()
model = AutoModelForSequenceClassification.from_pretrained(
    cfg.model_name,
    num_labels=cfg.num_labels,
    dtype=torch.bfloat16
).to("cuda")

# --- 2. 数据准备与预处理 ---
dataset = load_dataset("imdb")
tokenizer = AutoTokenizer.from_pretrained(cfg.model_name)

def preprocess_function_new(examples):
    """实现 Head-Tail 截断策略 (128 + 384)"""
    inputs = tokenizer(examples["text"], truncation=False, padding=False)
    labels = examples.get("label", None)

    max_len, head_len, tail_len = 512, 128, 384
    new_input_ids, new_attention_masks = [], []

    for input_ids, attention_mask in zip(inputs["input_ids"], inputs["
    attention_mask"]):
        if len(input_ids) > max_len:
            # 拼接头部 128 token 和尾部 384 token
            new_ids = input_ids[:head_len] + input_ids[-tail_len:]
            new_mask = attention_mask[:head_len] + attention_mask[-tail_len
    :]
        else:
            new_ids, new_mask = input_ids, attention_mask
        new_input_ids.append(new_ids)
        new_attention_masks.append(new_mask)

    return {
        "input_ids": new_input_ids,
        "attention_mask": new_attention_masks,
        **({"labels": labels} if labels is not None else {})
    }

tokenized_datasets = dataset.map(preprocess_function_new, batched=True,
    num_proc=8)
data_collator = DataCollatorWithPadding(tokenizer=tokenizer)

```

```python
# --- 3. 对抗工具类 (FGM) ---
class FGM:
    def __init__(self, model):
        self.model = model
        self.backup = {}

    def attack(self, epsilon=1.0, emb_name='word_embeddings'):
        for name, param in self.model.named_parameters():
            if param.requires_grad and emb_name in name:
                self.backup[name] = param.data.clone()
                norm = torch.norm(param.grad.float())
                if torch.isfinite(norm) and norm.item() > 0:
                    r_at = (epsilon * param.grad.float()) / (norm + 1e-12)
                    param.data.add_(r_at.to(param.data.dtype))

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

# --- 4. 自定义对抗训练器 ---
class AdversarialTrainer(Trainer):
    def __init__(self, train_config: TrainConfig, *args, **kwargs):
        super().__init__(*args, **kwargs)
        self.train_config = train_config
        self.fgm = FGM(self.model)

    def training_step(self, model, inputs, num_items_in_batch=None):
        model.train()
        inputs = self._prepare_inputs(inputs)
        loss = self.compute_loss(model, inputs)

        current_epoch = self.state.epoch if self.state.epoch is not None
    else 0
        should_do_adv = (self.train_config.use_adv_training and
                         current_epoch >= self.train_config.adv_start_epoch)

        if should_do_adv:
            # 1. 正常梯度回传
            self.accelerator.backward(loss * self.train_config.adv_alpha)
            # 2. 对抗攻击
            self.fgm.attack(epsilon=self.train_config.adv_epsilon)
            # 3. 对抗样本损失计算与回传
```

```
            loss_adv = self.compute_loss(model, inputs)
            self.accelerator.backward(loss_adv * (1 - self.train_config.adv_alpha))
            # 4. 恢复权重
            self.fgm.restore()
        else:
            self.accelerator.backward(loss)

        return loss.detach()

# --- 5. 评估指标与执行 ---
def compute_metrics(eval_pred):
    logits, labels = eval_pred
    predictions = np.argmax(logits, axis=-1)
    probs = torch.nn.functional.softmax(torch.tensor(logits), dim=-1).numpy()
    return {
        "accuracy": accuracy_score(labels, predictions),
        "roc_auc": roc_auc_score(labels, probs[:, 1]),
        "f1": f1_score(labels, predictions, average='macro'),
    }

args = TrainingArguments(
    output_dir=cfg.output_dir,
    eval_strategy="epoch",
    learning_rate=cfg.learning_rate,
    per_device_train_batch_size=cfg.batch_size,
    num_train_epochs=cfg.epochs,
    bf16=True,
    report_to="none"
)

trainer = AdversarialTrainer(
    train_config=cfg,
    model=model,
    args=args,
    train_dataset=tokenized_datasets["train"],
    eval_dataset=tokenized_datasets["test"],
    data_collator=data_collator,
    compute_metrics=compute_metrics,
)

trainer.train()
```

参考文献

[1] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 142–150, 2011.

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 4171–4186, 2019.

[3] P. He, X. Liu, J. Gao, and W. Chen, "DeBERTa: Decoding-enhanced BERT with disentangled attention," *arXiv preprint arXiv:2006.03654*, 2021.

[4] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[5] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[6] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

[7] Y. Kim, "Convolutional neural networks for sentence classification," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1746–1751, 2014.

[8] W. Li and et al., "Deep learning for sentiment analysis: A survey," *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 2022.

[9] I. Beltagy, M. E. Peters, and A. Cohan, "Longformer: The long-document transformer," *arXiv preprint arXiv:2004.05150*, 2020.

[10] M. Zaheer, G. Guruganesh, K. A. Dubey, *et al.*, "Big bird: Transformers for longer sequences," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[11] C. Zhu, Y. Cheng, Z. Gan, S. Sun, T. Goldstein, and J. Liu, "FreeLB: Stronger adversarial training for natural language understanding," in *International Conference on Learning Representations (ICLR)*, 2020.

[12] C. Sun, X. Qiu, X. Xu, and X. Huang, "How to fine-tune BERT for text classification?," in *Chinese Computational Linguistics*, pp. 194–206, 2019.

[13] T. Miyato, A. M. Dai, and I. Goodfellow, "Adversarial training methods for semi-supervised text classification," *arXiv preprint arXiv:1605.07725*, 2016.