TextAssessor: 基于BERT的文本评估模型

# 1. 概述

## 1.1 项目背景

TextAssessor是一个基于BERT预训练模型的文本评估模型，专门用于伙伴部队能力评估和上帝抵抗军威胁分析的自动化文本处理。该模型是对RAND公司研究报告[《Leveraging Machine Learning for Operation Assessment》](https://www.rand.org/pubs/research_reports/RR4196.html)在深度学习上的技术拓展和实现。

该模型能够同时处理两类评估任务：

PF (Partner Force) 评估：伙伴部队能力评估，包括PF\_score、PF\_US、PF\_neg三个维度。

LRA (Lord’s Resistance Army) 评估：上帝抵抗军威胁评估，包括Threat\_up、Threat\_down、Citizen\_impact三个维度。

## 1.2 技术特点

(1) 双任务学习架构：同时处理PF（伙伴部队）和LRA（上帝抵抗军）两类评估任务

(2) 注意力过滤机制：通过可学习的权重过滤机制提升特定任务的特征表示

(3) 分阶段训练策略：采用五+F阶段渐进式训练，逐步解冻更多的预训练层

(4) 不平衡数据处理：使用正样本权重调整处理标签不平衡问题

(5) 完整工具链：集成数据清洗、关键词标注、API调用、权重计算等核心工具函数

(6) 自动化标注：基于DeepSeek API的智能文本标注

# 2. 理论基础

## 2.1 BERT预训练模型

BERT (Bidirectional Encoder Representations from Transformers) 是一个基于Transformer的双向编码器模型。其核心特点是：

Self-Attention机制：



其中，为查询矩阵(Query)，为键矩阵(Key)，为值矩阵 (Value)，为键向量的维度。

多头注意力：



其中：



## 2.2 双任务学习理论

双任务学习通过共享底层表示学习，利用任务间的相关性提高整体性能。

设两个任务的损失函数分别为和，总损失为：



其中，和是任务权重参数。

## 2.3 注意力过滤机制

本模型引入可学习的过滤器，对句子级表示进行加权平均：

句子嵌入计算：



其中，为第个句子的第个token的隐状态，为注意力掩码，是防止除零的小常数。

任务特定过滤：



加权聚合：



# 3. 模型架构

## 3.1 整体架构



图 1 整体架构

## 3.2 组件详解

### 3.2.1 文本编码器 (Text Encoder)

* 模型：BERT-base-uncased
* 维度：768
* 功能：将输入文本转换为上下文相关的向量表示

### 3.2.2 过滤器 (Filter)

* PF过滤器：用于识别与伙伴部队相关的句子
* LRA过滤器：用于识别与上帝抵抗军相关的句子



图 2 过滤器结构

### 3.2.3 评估器 (Assessor)

* PF评估器：输出3维预测 (PF\_score, PF\_US, PF\_neg)
* LRA评估器：输出3维预测 (Threat\_up, Threat\_down, Citizen\_impact)



图 3 评估器结构

## 3.3 数学表示

设输入文本序列为，模型的前向传播过程为：

* BERT编码：



* 句子嵌入：



* 任务过滤：



* 加权聚合：



* 最终预测：



# 4. 算法原理

## 4.1 损失函数

采用加权二元交叉熵损失处理不平衡数据：



其中，为正样本权重，是正样本数量，是负样本数量。

总损失函数：



## 4.2 优化策略

采用AdamW优化器，具有权重衰减正则化：



其中，是学习率，、是动量估计，是权重衰减系数。

## 4.3 评估指标

准确率计算：



其中，为样本数量，是预测维度数量，是指示函数

# 5. 数据预处理

## 5.1 原始数据处理流程

### 5.1.1 数据来源

原始数据文件为 Lexis-Nexis\_LRA.csv，包含以下字段：

表 1原始数据文件字段

|  |  |
| --- | --- |
| 字段 | 描述 |
| Title | 报告标题 |
| Source | 报告来源 |
| Time | 报告时间 |
| Year | 报告年份 |
| Month | 报告月份 |
| Day | 报告日期 |
| Text | 报告正文 |

### 5.1.2 数据清洗步骤

* 添加ID列：为每条记录分配唯一标识符
* 文本预处理：
  + 去除 Text 列两端空格
  + 删除 Year、Month、Day 列
  + 进行编码与格式检查
  + 去除重复文本
* 数据验证：使用 is\_legal\_text() 函数验证文本合法性

清洗后数据保存为 cleaned.csv，包含字段：

表 2清洗后数据字段

|  |  |
| --- | --- |
| 字段 | 描述 |
| ID | 原数据ID |
| Title | 报告标题 |
| Source | 报告来源 |
| Time | 报告时间 |
| Text | 报告正文（列表格式） |

## 5.2 目标文本标注

### 5.2.1 PF（伙伴部队）目标标注

筛选包含以下关键词的文本片段，这些地区是伙伴部队活动的主要区域：

- “Uganda”（乌干达）

- “Sudan”（苏丹）

- “Central African Republic”, “CAR”（中非共和国）

- “Democratic Republic of the Congo”, “DRC”（刚果民主共和国）

使用 abstract\_text() 函数进行关键词匹配和上下文提取。

### 5.2.2 LRA（上帝抵抗军）目标标注

筛选包含以下关键词的文本片段：

- “Lord’s Resistance Army”, “LRA”（上帝抵抗军）

使用 abstract\_text() 函数进行关键词匹配和上下文提取。

## 5.3 训练数据构建

* 数据筛选：从同时包含PF和LRA目标的文本中选择300个样本
* 人工标注流程：使用DeepSeek API进行六个维度的0-1标注
* 最终数据集：经过标注和整合，最终训练数据集包含以下字段：

表 3 训练数据字段

|  |  |
| --- | --- |
| 字段 | 描述 |
| Text | 文本列表 (list) |
| PF\_TARGET | 伙伴部队相关句子标记 (list) |
| LRA\_TARGET | 上帝抵抗军相关句子标记 (list) |
| PF\_score | 报告提到伙伴部队的成功 (binary) |
| PF\_US | 报告提到美国与伙伴部队合作 (binaty) |
| PF\_neg | 报告提到美国伙伴部队的负面情况 (binary) |
| Threat\_up | 报告提到上帝抵抗军的威胁增加 (binary) |
| Threat\_down | 报告提到上帝抵抗军的威胁减少 (binary) |
| Citizen\_impact | 报告提到公民受到上帝抵抗军暴力的影响 (binary) |

## 5.4 文本预处理

* 分词和编码
* 数据类型转换

## 5.5 正样本权重计算

计算各任务维度的正样本权重，用于处理类别不平衡问题。权重用于加权二元交叉熵损失函数，较高的权重给予少数类（正样本）更多关注，在实际使用时会对权重取平方根以避免过度加权。

## 5.6 数据处理工具函数

### 5.6.1 文本合法性检查

检查文本是否为合法的ASCII字符串且可被ast.literal\_eval解析。使用正则表达式检查文本是否包含非ASCII字符，如包含则返回False；尝试使用ast.literal\_eval解析文本，成功则返回True；解析失败则返回False。

### 5.6.2 关键词匹配和上下文提取

在文本列表中匹配目标关键词，并返回包含关键词及其上下文的布尔掩码。遍历文本列表，检查每个句子是否包含目标关键词；使用滑动窗口策略，将匹配范围扩展到相邻句子；通过zip操作合并当前句子、前一句子、后一句子的匹配状态；返回扩展后的匹配掩码，用于提取相关上下文。

### 5.6.3 DeepSeek API标注接口

调用DeepSeek API进行文本标注。使用OpenAI客户端兼容接口调用DeepSeek API；系统提示设置为通用助手角色；采用非流式模式获取完整响应。

### 5.6.4 正样本权重计算

计算各评估维度的正样本权重，用于处理类别不平衡问题。使用lambda函数简化计算：f = lambda x: x[0] / x[1]；通过value\_counts()[[0, 1]]获取负样本和正样本数量；字典推导式批量计算所有目标的权重。

# 6. 训练策略

## 6.1 五+F阶段渐进式训练策略

基于实际实验设计，模型采用五+F阶段渐进式训练策略，逐步解冻BERT的不同层次，实现从任务特定组件到深层特征提取的渐进式微调。

### 第一阶段 (70 epochs)

* 冻结层：BERT编码器所有参数
* 训练层：PF过滤器、LRA过滤器、PF评估器、LRA评估器
* 学习率：1e-6 (所有训练层)
* 权重衰减：1e-3
* 批次大小：6
* 目的：训练任务特定的上层组件，学习任务相关的表示

### 第二阶段 (50 epochs)

* 解冻层：BERT pooler层
* 学习率：
  + 过滤器/评估器：1e-6
  + BERT pooler：8e-7
* 权重衰减：1e-3
* 批次大小：6
* 目的：微调句子级表示，优化序列级特征提取

### 第三阶段 (70 epochs)

* 解冻层：BERT最后一层编码器 (layer[-1:])
* 学习率：
  + 过滤器/评估器：1e-6
  + BERT pooler：8e-7
  + BERT最后一层：4e-7
* 权重衰减：1e-3
* 批次大小：6
* 目的：精细调整高层特征提取

### 第四阶段 (70 epochs)

* 解冻层：BERT倒数第2-3层编码器 (layer[-3:-1])
* 学习率：
  + 过滤器/评估器：1e-6
  + BERT pooler：1e-6
  + BERT最后一层：8e-7
  + BERT倒数2-3层：4e-7
* 权重衰减：1e-3
* 批次大小：6
* 目的：扩展微调范围，优化中高层特征

### 第五阶段 (90 epochs)

* 解冻层：BERT倒数第4-7层编码器 (layer[-7:-3])
* 学习率：
  + 过滤器/评估器：1e-6
  + BERT pooler：1e-6
  + BERT最后一层：1e-6
  + BERT倒数2-3层：8e-7
  + BERT倒数4-7层：4e-7
* 权重衰减：1e-3
* 批次大小：6
* 目的：进一步扩展微调范围

### 最终阶段 (140 epochs)

* 解冻层：所有BERT参数（编码器、嵌入层）
* 学习率：
  + 过滤器/评估器：1e-6
  + BERT pooler：1e-6
  + BERT最后一层：1e-6
  + BERT倒数2-3层：1e-6
  + BERT倒数4-7层：8e-7
  + BERT其余编码器层：4e-7
  + BERT嵌入层：2e-7
* 权重衰减：1e-4 (降低正则化强度)
* 批次大小：1 (精细化训练)
* 目的：全模型微调，达到最优性能

## 6.2 渐进式学习率调度策略

采用层次化学习率策略，遵循以下原则：

* 任务特定层：保持较高学习率 (1e-6)，快速适应任务
* 高层特征：中等学习率 (8e-7 - 1e-6)，适度调整
* 中层特征：较低学习率 (4e-7 - 8e-7)，谨慎微调
* 底层特征：最低学习率 (2e-7 - 4e-7)，保持预训练知识

## 6.3 训练策略优势

* 渐进式解冻：避免预训练知识的灾难性遗忘
* 层次化学习率：不同层采用不同的学习速度
* 动态批次调整：最终阶段使用更小批次进行精细调整
* 自适应正则化：根据训练阶段调整权重衰减强度

# 7. 实验设计

## 7.1 数据集配置

### 7.1.1 原始数据统计

* 原始文件：Lexis-Nexis\_LRA.csv
* 清洗后数据：cleaned.csv
* 训练样本选择：300个同时包含PF和LRA目标的文本
* 最终训练集：300个标注完成的文档

### 7.1.2 数据分割策略

* 训练集：80% (240个样本)
* 测试集：20% (60个样本)
* 分割方式：随机分割，固定随机种子20040508
* 数据来源：train\_target\_df.csv

### 7.1.3 数据标注质量

* 标注方式：基于DeepSeek API的自动标注
* 标注标准：六个明确的二分类标准
* 质量控制：通过正则表达式提取和验证标注结果
* 标注文档：完整记录保存在target\_df.md

## 7.2 实验环境

* 硬件：GPU (CUDA)
* 深度学习框架：PyTorch + Transformers
* 预训练模型：BERT-base-uncased (本地路径: ./model/bert-base-uncased)
* 最大序列长度：256 tokens
* 分词器：BertTokenizer

## 7.3 超参数配置

### 7.3.1 基础参数

表 4 基础参数

| 参数 | 值 |
| --- | --- |
| 随机种子 | 20040508 |
| Dropout率 | 0.7 |
| 优化器 | AdamW |
| 最大序列长度 | 256 |

### 7.3.2 各阶段训练参数

表 5 各阶段训练参数

| 阶段 | 训练轮数 | 批次大小 | 权重衰减 | 主要学习率范围 |
| --- | --- | --- | --- | --- |
| 第一阶段 | 70 | 6 | 1e-3 | 1e-6 |
| 第二阶段 | 50 | 6 | 1e-3 | 1e-6 ~ 8e-7 |
| 第三阶段 | 70 | 6 | 1e-3 | 1e-6 ~ 4e-7 |
| 第四阶段 | 70 | 6 | 1e-3 | 1e-6 ~ 4e-7 |
| 第五阶段 | 90 | 6 | 1e-3 | 1e-6 ~ 4e-7 |
| 最终阶段 | 140 | 1 | 1e-4 | 1e-6 ~ 2e-7 |

### 7.3.3 正样本权重

通过 get\_pos\_weights() 函数计算各任务的类别权重：

- PF\_score: 负样本数 / 正样本数

- PF\_US: 负样本数 / 正样本数

- PF\_neg: 负样本数 / 正样本数

- Threat\_up: 负样本数 / 正样本数

- Threat\_down: 负样本数 / 正样本数

- Citizen\_impact: 负样本数 / 正样本数

## 7.4 实验流程

### 7.4.1 数据预处理完整流程

* 原始数据加载：从Lexis-Nexis\_LRA.csv读取新闻数据
* 数据清洗：
  + 添加ID列进行唯一标识
  + 文本格式标准化和去重
  + 删除无效和重复记录
* 关键词标注：
  + PF目标：标注包含Uganda、Sudan、CAR、DRC的句子
  + LRA目标：标注包含Lord’s Resistance Army的句子
* 样本筛选：选择同时包含PF和LRA目标的300个文本
* 人工标注：使用DeepSeek API进行六维度标注
* 数据整合：合并原始文本和标注结果生成train\_target\_df.csv
* 训练测试分割：使用sklearn.train\_test\_split进行数据分割

### 7.4.2 模型训练流程



图 4 训练流程

### 7.4.3 模型检查点

* 阶段性保存：每个主要阶段结束后保存模型
* 关键检查点：
  + text\_assessor\_A(260).pth: 260 epochs后
  + text\_assessor\_B(350).pth: 350 epochs后
  + text\_assessor\_C(490).pth: 490 epochs后
* 损失(&准确率)图表：
  + loss-plot\_A(260).svg
  + loss-plot\_B(350).svg
  + loss-plot\_C(490).svg

## 7.5 评估策略

### 7.5.1 训练时评估

* 每个epoch评估：在训练和验证集上计算损失和准确率
* 实时监控：使用tqdm显示训练进度
* 定期可视化：每10个epoch生成损失和准确率曲线

### 7.5.2 评估指标

* 损失函数：加权二元交叉熵损失
* 准确率：二分类准确率的平均值
* 收敛性：训练和验证损失的变化趋势

### 7.5.3 过拟合监控

通过比较训练损失和验证损失的差异来监控过拟合：

* 正常学习：训练损失和验证损失同步下降
* 轻微过拟合：验证损失停止下降或轻微上升
* 严重过拟合：验证损失显著上升

# 8. 性能评估

## 8.1 评估指标

### 8.1.1 二分类准确率

对每个任务维度计算：



### 8.1.2 整体准确率



## 8.2 实验结果分析

### 8.2.1 训练过程监控

通过三个检查点的损失曲线图进行分析：

* 260 epochs后 (loss-plot\_A(260).svg)：
  + 观察初期训练效果
  + 验证渐进式解冻策略的有效性
  + 分析前四个训练阶段的收敛情况
* 350 epochs后 (loss-plot\_B(350).svg)：
  + 中期训练效果评估
  + 检查是否出现过拟合迹象
  + 验证第五阶段训练的必要性
* 490 epochs后 (loss-plot\_C(490).svg)：
  + 最终训练效果
  + 全模型微调的收敛性分析
  + 训练和验证性能的最终对比

### 8.2.2 性能指标分析

* 收敛性：
  + 训练损失和验证损失是否稳定下降
  + 各阶段的收敛速度对比
  + 学习率调整对收敛的影响
* 过拟合检测：
  + 训练损失与验证损失的差距变化
  + 批次大小调整（6→1）对过拟合的影响
  + 权重衰减调整（1e-3→1e-4）的效果
* 学习效果：
  + 准确率在各阶段的提升趋势
  + 六个任务维度的平衡学习情况
  + 最终达到的性能水平

### 8.2.3 阶段性评估

* 第一阶段：任务特定组件的学习效果
* 第二阶段：句子级表示的优化程度
* 第三-五阶段：渐进式特征提取的改进
* 最终阶段：全模型微调的性能提升

## 8.3 模型应用和评估

### 模型推理流程

模型推理包含以下步骤：



图 5 模型应用

### 8.3.2 预测结果格式

模型输出包含六个维度的预测值：

表 6 预测结果字段

| 维度 | 类型 | 描述 |
| --- | --- | --- |
| pred.values.PF\_score | float | 未归一化的PF\_score预测值 |
| pred.values.PF\_US | float | 未归一化的PF\_US预测值 |
| pred.values.PF\_neg | float | 未归一化的PF\_neg预测值 |
| pred.values.Threat\_up | float | 未归一化的Threat\_up预测值 |
| pred.values.Threat\_down | float | 未归一化的Threat\_down预测值 |
| pred.values.Citizen\_impact | float | 未归一化的Citizen\_impact预测值 |

### 8.3.3 大规模数据处理

* 数据来源：cleaned.csv（完整清洗后数据）
* 处理策略：逐条处理，避免内存溢出
* 异常处理：对空文本列表进行特殊处理
* 结果保存：预测结果合并到result\_df.csv

### 8.3.4 模型检查点选择

实验使用了text\_assessor\_B(350).pth作为推理模型，该检查点代表：

* 训练进度：350个epochs后的模型状态
* 性能平衡：在训练充分性和过拟合之间的平衡点
* 实用性：适合大规模推理应用的稳定模型

# 9. 算法伪代码

### 9.1 前向传播算法

|  |
| --- |
| 算法1: 前向传播算法 |
| 输入: 文本序列列表 X = [x1, x2, ..., xN]  输出: 预测结果 logits = [logit1, logit2, ..., logitN]  1: logits ← []  2: for xi in X:  3: 获取句子嵌入input\_ids, attention\_mask ← tokenize(xi)  4: H ← BERT\_encoder(input\_ids, attention\_mask)  5: 加权平均池化mask ← attention\_mask.unsqueeze(-1).expand(H.size())  6: S ← sum(H \* mask, dim=1) / (sum(mask, dim=1) + ε) // [n, hidden\_size]  7: 任务特定过滤P\_PF ← PF\_filter(S), P\_LRA ← LRA\_filter(S)  8: 加权聚合C\_PF ← sum(S \* P\_PF.unsqueeze(-1), dim=0) / (sum(P\_PF) + ε)  9: C\_LRA ← sum(S \* P\_LRA.unsqueeze(-1), dim=0) / (sum(P\_LRA) + ε)  10: 最终预测y\_PF ← PF\_assessor(C\_PF), y\_LRA ← LRA\_assessor(C\_LRA)  11: logits.append(concat([y\_PF, y\_LRA]))  12: return stack(logits, dim=0) |

### 9.2 损失计算算法

|  |
| --- |
| 算法2: 加权二元交叉熵损失计算 |
| 输入: 预测值 y\_pred, 真实标签 y\_true, 正样本权重 pos\_weights  输出: 总损失 L\_total  1: L\_total ← 0  2: targets ← ['PF\_score', 'PF\_US', 'PF\_neg', 'Threat\_up', 'Threat\_down', 'Citizen\_impact']  3: for i in range(6):  wi ← sqrt(pos\_weights[targets[i]])  Li ← binary\_cross\_entropy\_with\_logits(y\_pred[i], y\_true[i], pos\_weight=wi)  L\_total ← L\_total + Li  4: return L\_total |

### 9.3 评估算法

|  |
| --- |
| 算法3: 模型评估 |
| 输入: 模型 M, 数据加载器 dataloader, 正样本权重 pos\_weights  输出: 平均损失 avg\_loss, 平均准确率 avg\_accuracy  1: M.eval()  2: total\_loss ← 0  3: total\_correct ← 0  4: for batch in dataloader:  outputs ← M(batch)  loss ← compute\_loss(outputs, batch, pos\_weights)  total\_loss ← total\_loss + loss.item()  // 计算准确率  for output, target in zip(outputs, batch):  predictions ← (sigmoid(output) > 0.5).float()  correct ← count\_correct\_predictions(predictions, target)  total\_correct ← total\_correct + correct  5: avg\_loss ← total\_loss / len(dataloader)  6: avg\_accuracy ← total\_correct / len(dataloader)  7: return avg\_loss, avg\_accuracy |

# 10. 结论与展望

## 10.1 模型优势

* 双任务协同学习：通过共享底层BERT编码器，PF和LRA任务能够相互促进
* 注意力过滤机制：有效识别任务相关的关键句子，提高模型解释性
* 分阶段训练策略：渐进式微调避免了梯度消失和过拟合问题
* 不平衡数据处理：加权损失函数有效处理标签不平衡问题

## 10.2 技术创新点

* 句子级注意力过滤：引入可学习的过滤器对句子进行任务特定的重要性评估
* 五+F阶段渐进式微调策略：从任务特定组件到全模型的渐进式训练
* 层次化学习率调度：不同层采用不同学习率，保持预训练知识的同时优化任务性能
* 动态批次调整：最终阶段采用更小批次进行精细化训练
* 多任务损失平衡：等权重组合多个二分类损失，处理类别不平衡问题

## 10.3 性能期望

基于当前五+F阶段训练架构和490个epochs的训练策略，预期模型能够达到：

* 整体准确率：> 80%
* 收敛稳定性：各阶段均能稳定收敛，总训练时间约490个epochs
* 泛化能力：通过渐进式微调和动态批次调整，在未见过的文本上保持较好性能
* 训练效率：通过分阶段训练和层次化学习率，在保证性能的同时提高训练效率

## 10.4 完整工作流程

### 10.4.1 数据处理流程

* 原始数据清洗：从Lexis-Nexis\_LRA.csv到cleaned.csv
* 目标句子标注：PF和LRA关键词匹配
* 样本筛选：选择包含两类目标的300个文本
* 人工标注：使用DeepSeek API进行六维标注
* 训练数据生成：合并生成train\_target\_df.csv

### 10.4.2 模型训练流程

* 五+F阶段渐进式训练：从任务层到全模型的逐步解冻
* 检查点管理：关键阶段保存模型状态
* 性能监控：实时追踪损失和准确率变化
* 可视化输出：生成训练过程的损失曲线图

### 10.4.3 模型应用流程

* 模型选择：使用350 epochs的检查点进行推理
* 大规模预测：对全部清洗后数据进行预测
* 结果整合：合并预测结果和原始数据
* 结果保存：输出完整的result\_df.csv

### 10.4.4 技术亮点总结

* 端到端流程：从原始数据到最终应用的完整链路
* 自动化标注：使用DeepSeek API减少人工标注成本
* 渐进式训练：科学的训练策略确保模型收敛
* 生产就绪：完整的推理和批量处理能力

# 附录

## 附录A：数据文件说明

| 文件名 | 描述 | 用途 |
| --- | --- | --- |
| Lexis-Nexis\_LRA.csv | 原始新闻数据 | 数据来源 |
| cleaned.csv | 清洗后数据 | 预处理结果 |
| target\_df.csv | 标注数据 | 人工标注结果 |
| target\_df.md | 标注文档 | 标注过程记录 |
| train\_target\_df.csv | 训练数据 | 模型训练输入 |
| result\_df.csv | 预测结果 | 模型应用输出 |

## 附录B：模型文件说明

| 文件名 | 描述 | 训练轮数 |
| --- | --- | --- |
| text\_assessor.pth | 最终模型 | 490 epochs |
| text\_assessor\_A(260).pth | 早期检查点 | 260 epochs |
| text\_assessor\_B(350).pth | 中期检查点 | 350 epochs |
| text\_assessor\_C(490).pth | 最终检查点 | 490 epochs |

## 附录C：可视化文件说明

| 文件名 | 描述 | 内容 |
| --- | --- | --- |
| loss-plot\_A(260).svg | 早期训练曲线 | 前260轮损失和准确率 |
| loss-plot\_B(350).svg | 中期训练曲线 | 前350轮损失和准确率 |
| loss-plot\_C(490).svg | 完整训练曲线 | 全部490轮损失和准确率 |