深度学习导论作业 2 报告:
基于 Attention 与 RNN 的垃圾邮件分类模型比较

PB22000150 刘行

2025 年 4 月

目录

1 实验任务与背景

本实验基于 Enron-Spam 数据集, 设计并实现基于自注意力机制 (Multi-head Attention) 与循环神经网络 (RNN) 的文本分类模型, 探究不同架构在垃圾邮件识别任务中的性能表现, 旨在:

- 熟悉文本分类任务中的数据清洗、词表构建及编码方法;
- 实现具备因果掩码 (Causal Mask) 的自回归多头注意力分类器;
- 搭建支持 RNN、LSTM、GRU 三种变体的循环神经网络分类器;
- 系统对比 Attention 与 RNN 两类模型在准确率、训练与推理效率上的优劣差异;
- 结合注意力权重可视化, 深入理解模型的决策依据与内部机制.

2 数据处理与词表构建

对原始文本数据执行以下预处理步骤:

- 将文本统一转换为小写 (`text.lower()`);
- 按空格进行分词;
- 基于词频统计, 构建词表 (最大词数 10 万, 最小词频 1);
- 引入特殊 Token (`<pad>`、`<unk>`), 并将文本长度统一至 200, 采用左填充策略;
- 删除缺失值, 并将 Subject 与 Message 列合并形成完整文本.

3 模型设计与训练设置

3.1 Attention 分类器

采用 Decoder-only 结构, 每个 token 仅能关注自身及历史 token. 主要组件包括:

- 嵌入层 (`nn.Embedding`);
- 手动实现的位置编码 (Positional Encoding);
- 多头自注意力层 (`nn.MultiheadAttention`), 结合因果掩码 (Causal Mask);
- 层归一化 (LayerNorm) 及 Dropout 正则化;
- 单值输出分类器, 损失函数为 `BCEWithLogitsLoss`.

3.2 RNN 分类器

RNN 分类器结构灵活, 支持以下三种循环单元:

- 简单 RNN (`nn.RNN`);
- 长短期记忆网络 LSTM (`nn.LSTM`);
- 门控循环单元 GRU (`nn.GRU`).

分类决策基于最后一个隐藏状态. 示例代码如下:

```
if rnn_type == 'rnn':
        self.rnn = nn.RNN(emb_dim, hidden_dim, num_layers, batch_first=True
            )
elif rnn_type == 'lstm':
        self.rnn = nn.LSTM(emb_dim, hidden_dim, num_layers, batch_first=
            True)
elif rnn_type == 'gru':
        self.rnn = nn.GRU(emb_dim, hidden_dim, num_layers, batch_first=True
            )
else:
        raise ValueError("rnn_type must be 'rnn', 'lstm', or 'gru'")
```

3.3 训练配置

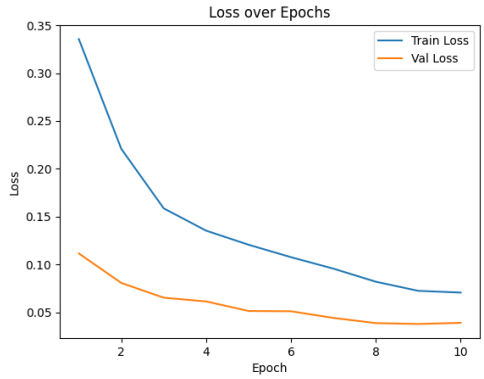
- 优化器: Adam, 学习率设为 $1e^{-2}$;
- 批量大小 (Batch Size): 64;
- 损失函数: `BCEWithLogitsLoss`;
- 早停策略 (Early Stopping): 若验证集性能连续多轮无提升, 则提前终止训练.

需要特别指出, 为绘制完整训练曲线, 实验中设置了较高的容忍度 (`patience=epochs+1`), 实际未触发早停. 但在实际应用中, 合理设置早停参数对于避免过拟合及提升训练效率至关重要.

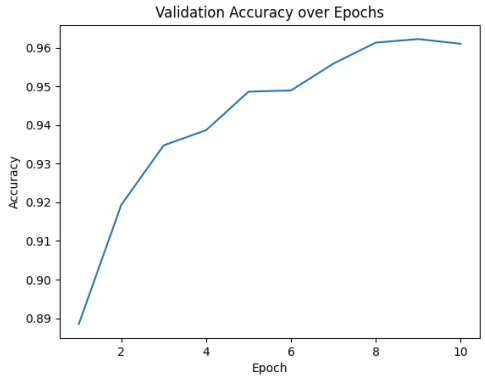
4 实验结果与分析

4.1 训练过程可视化

训练过程中, 记录了每个 epoch 的训练与验证集 Loss, Accuracy 等指标. 可视化结果如下:

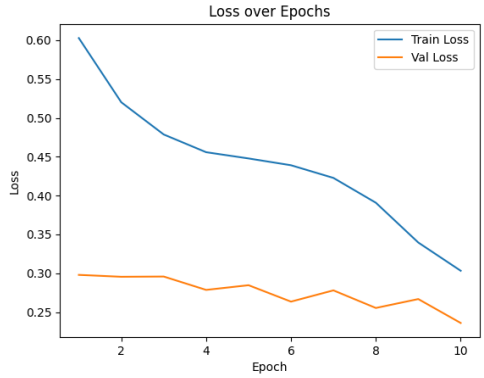


(a) 训练与验证集 Loss 变化曲线

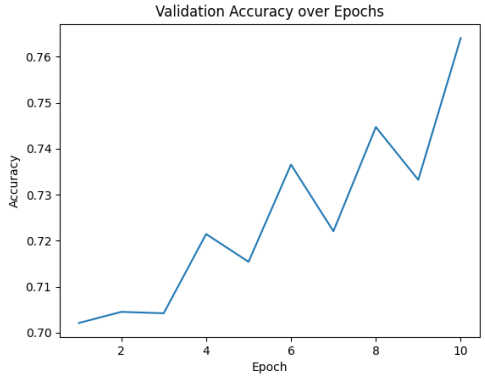


(b) 验证集 Accuracy 变化曲线

图 1: Attention 模型训练过程可视化



(a) 训练与验证集 Loss 变化曲线



(b) 验证集 Accuracy 变化曲线

图 2: RNN 模型训练过程可视化

4.2 最终测试集性能

模型	Accuracy	Precision	Recall	F1-score
Attention	0.945	0.950	0.941	0.945
RNN	0.766	0.792	0.731	0.760

表 1: Attention 与 RNN 模型在 Enron-Spam 测试集上的性能对比

4.3 注意力机制可视化实验

为加深理解, 使用 `attention_fig.py` 脚本, 构造短序列并绘制其 Self-Attention 权重热力图: 可以观察到, 每个 Token 仅关注自身及其历史位置, 符合自回归 Attention 的建模逻辑.

图 3: 示例序列的 Self-Attention 权重热力图

4.4 模型对比分析

4.4.1 模型性能对比

Attention 模型在准确率、Precision、Recall 及 F1-score 等所有指标上均优于 RNN, 主要归因于其更强的长距离依赖建模能力及丰富的表达特性.

4.4.2 训练与推理效率

尽管 RNN 单步计算开销较小, 但受限于时序依赖, 难以高效并行, 整体训练时间更长. 而 Attention 结构可充分利用并行计算资源, 训练速度明显更快, 推理阶段同样表现优异, 具备实际部署潜力.

5 代码亮点总结

- **灵活的 RNN 模块设计**: 支持 RNN/LSTM/GRU 三种结构切换, 便于全面评估循环单元性能;
- **位置编码与因果掩码手动实现**: 加深了对 Transformer 内部原理的理解;
- **早停机制完备**: 即便本次未启用, 仍具备良好的工程规范;
- **模块化设计**: 训练、评估、可视化流程清晰分离, 代码可维护性与扩展性良好;
- **丰富的可视化支持**: 包括 Loss、Accuracy、F1 变化曲线与 Attention 热力图, 为训练过程监控与模型分析提供重要支撑.

6 存在的局限性与改进建议

- **词表构建未引入预训练词向量**: 未来可结合 GloVe、FastText 等外部语义资源, 提升初始表示质量;
- **Attention 结构较浅**: 目前仅使用单层 Attention, 未来可探索堆叠多层、引入残差连接等策略;
- **Padding 位置未屏蔽**: 建议在 Attention 中对 `<pad>` Token 添加 mask, 减少无效注意力分配;
- **RNN 变体评估不够全面**: 后续可系统对比 RNN、LSTM、GRU 在此任务下的细粒度性能差异;
- **推理效率缺乏定量评估**: 可进一步测试吞吐率、延迟等指标, 丰富推理阶段分析.

7 结论与展望

本实验充分展示了基于自注意力机制的模型在文本分类任务中的强大表现. 未来研究方向可包括:

- 引入预训练 Embedding 或大规模预训练模型 (如 BERT) 增强文本表示;
- 尝试旋转位置编码 (RoPE) 等先进位置编码方法;
- 深化模型层数, 引入多头注意力、残差与正则化策略, 提升模型泛化能力;
- 加强可解释性研究, 结合 Attention 可视化与梯度敏感性分析, 提高模型透明度与可控性;
- 探索小样本迁移学习与增量学习场景下的应用潜力.

未来的真正挑战, 在于兼顾性能、效率与可解释性, 推动深度学习模型向更加可持续、可信赖的方向演进.