《人工智能导论》大作业

任务名称：谣言检测与识别

完成组号：5

小组人员：刘晟祺 李子曦 钟文博

完成时间：2025.6.1

一、任务目标

完成一个基于谣言检测数据集的二分类模型，尽可能保证对推文进行高准确率的谣言检测与识别，同时具有泛化能力。模型输出结果，0 表示非谣言，1 表示谣言。

二、具体内容

（一）实施方案

模型最关键的两个脚本，train_bigru_advance.py 用于模型训练，保存性能最优模型；classify.py 用于加载保存的模型，并对文本进行推理。

train_bigru_advance.py 的 main 函数用于训练模型，读入、预处理、增强样本，提取标签计算 pos_weight，构建 vocab 并保存，创建 Dataset（train & val）和 DataLoader，初始化模型、优化器、损失函数、学习率调度器，设置早停机制与准确率记录列表。在每个训练循环中，计算损失、在训练集和测试集上的准确率，并检查是否需要早停。

classify.py 加载出训练阶段保存的 checkpoint，构造 BiGRU 模型并加载权重。Classify 通过调用模型，正向推理得到标量 logit，使用 torch.sigmod 将标量 logit 转化为概率，输出分类结果。

（二）核心代码分析

2.1 train_bigru_advance.py

```
def synonym_replace(text, prob=0.2):
```

定义了同义词替换函数以实现数据增强。利用 WordNet 可以搜索同义词，得到一系列同种语义下的近义词组。每个词组包含一系列 lemmas，即具体的词（格式为单词基本形式如动词原形、名词单数）。我们以一定概率选用第一个 lemma 去替换。虽然可能存在词性或者词形不对的问题，但仍然可以作为训练泛化能力更强模型的手段之一。

```
def augment_data(df):
```

我们采用 Pandas DataFrame 的数据结构，label 字段标识是否属于谣言，而 test 字段是信息内容，在用 df_aug 的 text 替换 df 后，将两个表格合并，并用 ignore_index=True 使得合并后的 DataFrame 有连续的整数索引。

```
def preprocess_text(text):
```

预处理函数将输入文本转为小写，并使用正则表达式 [^a-zA-Z0-9!?\s] 移除除字母、数字、和特殊标点外的字符，规范化文本以便模型处理。

```
def tokenize(text):
```

分词器主要通过 findall 使用正则表达式 \w+|"!?"] 将输入文本分词为单词列表，匹配连续的字母、数字、下划线或特殊标点， + 表示连续多次（大于等于 1），r 表示保留原始字符串。

```
def encode(text, vocab):
```

在编码函数将输入文本分词后映射为词汇表中的索引序列，未知词映射为 <UNK>，长度不足 MAX_LEN 时用 <PAD> 填充，超出时截断，确保输出固定长度的序列供模型使用。

```
class RumorDataset(Dataset):
```

对数据集做了封装，getitem 函数使用 encode()函数，将第 idx 条文本基于词汇表 vocab 转换成数字 ID 序列，随后转换为 PyTorch 张量，.clone().detach() 的作用是创建一个新的张量副本，避免梯度计算污染，确保安全操作。

```
class BiGRU(nn.Module):
```

该类定义了一个双向 GRU 模型。输入 x 是一个形状为 [batch_size, MAX_LEN] 的二维张量，表示批量词索引。嵌入层将词索引转换为 [batch_size, MAX_LEN, embedding_dim] 的词向量，经过第一个 Dropout 层防止过拟合。双向 GRU 层处理词向量序列，输出正向和反向的最后隐藏状态，拼接为 [batch_size, 2*hidden_dim] 的向量。再次应用 Dropout 层后，通过全连接层和 ReLU 激活降维，最后在全连接层输出单值 logits [batch_size]，用于二分类预测。

2.2 classify.py

```
def __init__(self):
```

使用 torch.load()加载保存的模型文件 bigru_model.pt，使用 pickle.load()加载词汇表 vocab.pkl，设置最大序列长度为 64，从模型检查点中恢复嵌入维度、隐藏层维度以及 dropout 率，实例化 BiGRU 模型，将模型权重加载进模型，设置模型为评估模式。

```
def classify(self, text: str) -> int:
```

将已经转换为 PyTorch 张量的词编码送入模型，模型输出是一个 logit，使用 sigmoid 函数将其转化为概率值，若概率大于 0.5，则判定为谣言（1），否则为非谣言（0）。

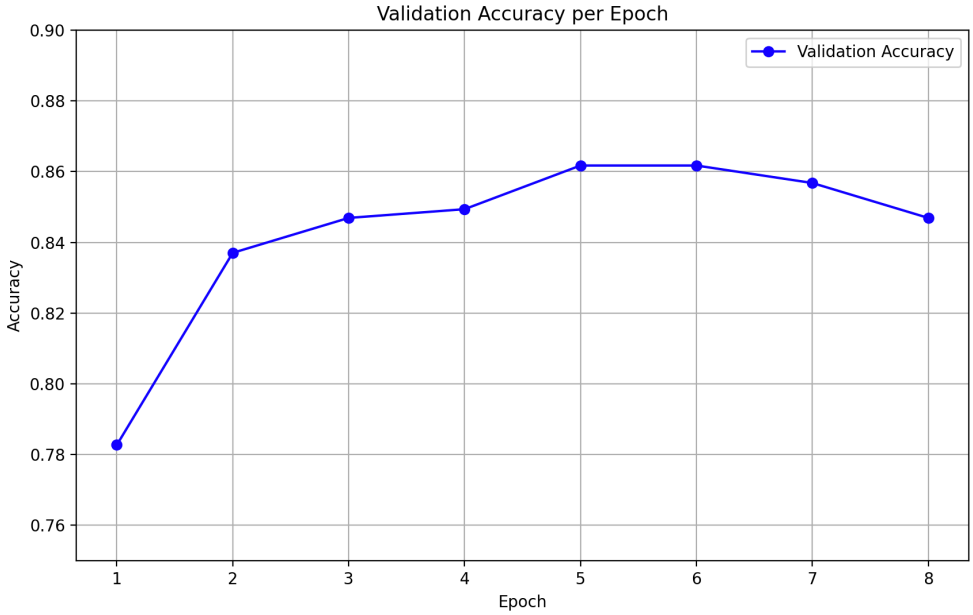

图 1 模型在验证集上的准确率变化

```
[nltk_data] Downloading package wordnet to
[nltk_data]     /Users/sachielliu/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
Epoch 1, Train Loss: 0.6052, Train Acc: 0.8573, Val Acc: 0.7827
Epoch 2, Train Loss: 0.3851, Train Acc: 0.9299, Val Acc: 0.8370
Epoch 3, Train Loss: 0.2394, Train Acc: 0.9749, Val Acc: 0.8469
Epoch 4, Train Loss: 0.1634, Train Acc: 0.9902, Val Acc: 0.8494
Epoch 5, Train Loss: 0.0937, Train Acc: 0.9953, Val Acc: 0.8617
Epoch 6, Train Loss: 0.0849, Train Acc: 0.9981, Val Acc: 0.8617
Epoch 7, Train Loss: 0.0575, Train Acc: 0.9974, Val Acc: 0.8568
Epoch 8, Train Loss: 0.0490, Train Acc: 0.9990, Val Acc: 0.8469
Early stopping at epoch 8
Best Val Acc: 0.8617
Model saved as bigru_model.pt
```

图 2 模型训练过程

在训练过程中，损失函数逐渐下降，在训练集上的准确率逐渐升高，趋于饱和（接近 100%），在验证集上的准确率先升后降，但早停机制帮助我们保留了最优模型。

三、工作总结

（一）收获、心得

通过实验，我们对 AI 模型的构建、训练和评估等关键环节建立了系统的认知。通过数据预处理和增强技术的应用，显著提升了模型的泛化性能。BiGRU 模型的结构也深化了我们对深度学习架构的理解。完善的模型保存与加载机制保障了训练与推理过程的一致性。本次实践让我们直观体会到模型细节对最终性能的影响，同时也认识到泛化

能力继续提高的可能性。总的来说，这是一次卓有成效的探索。

（二）遇到问题及解决思路

最初完成代码后，在提供的训练集训练后，模型在同源测试集上的表现还不错，但当测试集主题不同时，模型的表现比较糟糕，将大量的谣言判断为非谣言，泛化能力欠佳。为了提高模型的泛化能力，我们采取了以下方式进行优化：

1. 同义词替换：在数据预处理时进行同义词替换，使模型能够更容易地识别出带有感情色彩的语句；
2. 正则过滤：在文本预处理时，删除除了字母、数字和标点符号之外的其他字符；
3. 参数调优：不断修改训练脚本的各个超参数，最终留下性能最好的超参数组合。

经过模型优化后，在不同主题不同源测试集上的表现得到提升，但在同源测试集上的表现略有下降。考虑到模型的实用性，我们选择在显著提升模型泛化性能时，略微降低对模型同源测试集准确率的要求。

四、课程建议

人工智能导论这门课让我们建立起人工智能知识体系的基本框架，有了实践经验，对每个学生的专业发展起到重要影响。唯一的小建议是，我们觉得本门课程可以通过一些措施鼓励学生课下的自主探索和实践，增加独立思考和学习时间，能够将理论真正与实践挂钩，避免出现纸上谈兵的情况。