## E9\_22336216



人工智能实验

# 中山大学计算机学院 人工智能 本科生实验报告

(2023学年春季学期)

课程名称: Artificial Intelligence

教学班级	DCS315	专业(方向)	计算机科学与技术(系统结 构)
学号	22336216	姓名	陶宇卓

## 1 实验题目

实验九: 自然语言推理 (Natural Language Inference)

#### 一、算法原理

• 自然语言推理(Natural Language Inference)

自然语言推理(Natural Language Inference,简称NLI),也称为文本蕴涵(Textual Entailment),是一项自然语言处理(NLP)任务,旨在判断给定的两个文本片段之间的推理关系。 具体来说,给定一个前提(premise)和一个假设(hypothesis),NLI任务的目标是确定以下三种关系之一:

- 1. **蕴涵(Entailment)**: 如果前提为真,那么假设也为真。例如:
  - 前提: 所有的鸟都会飞。
  - 假设: 麻雀会飞。 这里, 假设从前提中可以推断出来, 因此是蕴涵关系。
- 2. **不蕴涵(Contradiction)**: 如果前提为真,那么假设为假。例如:
  - 前提: 所有的鸟都会飞。
  - 假设: 企鹅不会飞。 这里, 假设和前提是相互矛盾的。
- 3. 中立 (Neutral) (三分类特有): 前提和假设之间既不支持也不矛盾。例如:
  - 前提: 所有的鸟都会飞。
  - 假设: 这只鸟是红色的。 这里,假设与前提无直接关系,既不能从前提推断出假设,也不会与前提矛盾。

## NLI的应用

NLI在许多NLP应用中都起到了重要作用,包括但不限于:

- 问答系统:通过推理用户问题与候选答案之间的关系,判断答案的正确性。
- **文本摘要**:通过推理摘要和原文之间的关系,验证摘要是否涵盖了原文的关键内容。
- 信息检索:通过推理查询和文档之间的关系、筛选出与查询最相关的文档。

## NLI模型

近年来,基于深度学习的NLI模型取得了显著进展,尤其是基于预训练语言模型(如BERT、RoBERTa、GPT等)的模型。这些模型通常通过以下步骤来解决NLI任务:

- 1. 编码器: 使用预训练语言模型将前提和假设编码为高维向量表示。
- 2. 拼接和交互:将前提和假设的向量表示拼接或进行其他交互操作,以捕捉两者之间的关系。
- 3. 分类器: 使用全连接层和softmax激活函数,输出蕴涵、矛盾和中立的概率。
- Bi-LSTM

LSTM是一种改进的循环神经网络(RNN),旨在解决RNN中的长期依赖问题。LSTM通过引入门控机制(输入门、遗忘门、输出门)来控制信息的流动,从而能够更好地捕捉序列中的长时依赖关系。 LSTM单元的核心组件:

• 遗忘门(Forget Gate): 控制前一时间步的记忆内容是否保留。

• 输入门(Input Gate): 控制当前时间步的新信息是否写入记忆。

• 输出门(Output Gate): 控制当前时间步的记忆内容如何输出。

#### 双向LSTM结构

双向LSTM由两个独立的LSTM网络组成,一个处理输入序列的正向(从前到后),另一个处理反向(从后到前)。通过结合这两个方向的信息,双向LSTM能够更全面地理解序列数据的上下文信息。 双向LSTM的结构示意图:

输入序列: x1, x2, ..., xn

正向LSTM: h1\_forward, h2\_forward, ..., hn\_forward

反向LSTM: h1\_backward, h2\_backward, ..., hn\_backward

双向LSTM输出: [h1\_forward, h1\_backward], [h2\_forward, h2\_backward], ...,
[hn\_forward, hn\_backward]

#### 算法原理

#### 1. 输入序列:

○ 输入序列 (x1,x2,...,xn)(x\_1, x\_2, ..., x\_n)(x1,x2,...,xn) 同时送入正向和反向LSTM网络。

#### 2. 正向传播:

○ 正向LSTM按照时间顺序(从 x1x\_1x1 到 xnx\_nxn)处理输入序列,计算每个时间步的隐藏状态。

#### 3. 反向传播:

○ 反向LSTM按照时间逆序(从 xnx\_nxn 到 x1x\_1x1)处理输入序列,计算每个时间步的隐藏状态。

#### 4. 隐藏状态拼接:

○ 将正向LSTM和反向LSTM在每个时间步的隐藏状态拼接,形成双向LSTM的输出。

#### 5. 输出层:

双向LSTM的输出可以连接到后续的全连接层、输出层或其他网络层,用于特定任务(如分类、预测等)。

## 优点

### • 更全面的上下文信息:

○ 双向LSTM能够同时考虑每个时间步的前后文信息,适用于需要捕捉全局依赖的任务。

#### • 改进的性能:

○ 在许多自然语言处理任务(如文本分类、机器翻译等)中,双向LSTM通常比单向LSTM表现更好。

## • 具体实现

#### 1. 数据读取

数据读取是NLI任务的第一步,需要读取包含前提(premise)和假设(hypothesis)对的标注数据集。在QNLI任务中,前提通常是句子,而假设是一个问题。目标是判断前提和假设之间的关系是否存在。

● 数据集格式: QNLI数据集包含三列: ID、句子(Sentence)、问题(Question)和标签(Label)。标签指示问题和句子是否匹配(即句子是否回答了问题)。

#### 2. 数据预处理

在数据预处理阶段,数据需要进行清洗和格式化,以便后续处理。

- **文本清洗**:去除特殊字符等噪音数据。
- **标记化(Tokenization)**: 将文本划分为单词或子词单元。使用标准的NLP库如NLTK或SpaCy 进行标记化。
- 映射标签: 将标签 (如"entailment"和"not entailment") 映射到数值,如0和1。

#### 3. 词嵌入 (Embedding)

词嵌入是将文本中的单词映射到高维向量空间中,以捕捉词汇的语义信息。

- 加载GloVe: 从预训练的GloVe嵌入文件(如glove.6B.300d.txt)中加载词嵌入。每个单词将映射到一个固定大小的向量(如300维)。
- **创建嵌入矩阵**:构建一个嵌入矩阵,矩阵的每一行对应一个词的GloVe向量。如果数据集中某些词在GloVe中不存在,可以随机初始化这些词的嵌入向量,或者直接初始化向量为零向量。

### 4. 数据对齐 (Padding)

由于不同的句子长度不一致,需要对齐(padding)以便批处理。

• **固定长度**:设定最大序列长度,超过该长度的句子进行截断,不足的句子进行填充 (padding),通常填充零。

#### 5. 使用分类模型进行训练

使用深度学习分类模型对处理后的数据进行训练,以预测前提和假设之间的关系。

#### • 模型架构:

○ 输入层:接受前提和假设的嵌入向量。

○ 嵌入层: 使用预加载的GloVe嵌入矩阵,将输入单词索引映射到词向量。

- **编码层**:使用双向LSTM对前提和假设的嵌入进行编码。LSTM的隐藏状态可以捕捉序列中的上下文信息。
- 拼接层:将前提和假设的编码表示拼接起来。
- **全连接层**:将拼接后的表示传递给全连接层,进行分类。
- **输出层**:使用softmax激活函数输出两类的概率。

#### • 训练步骤:

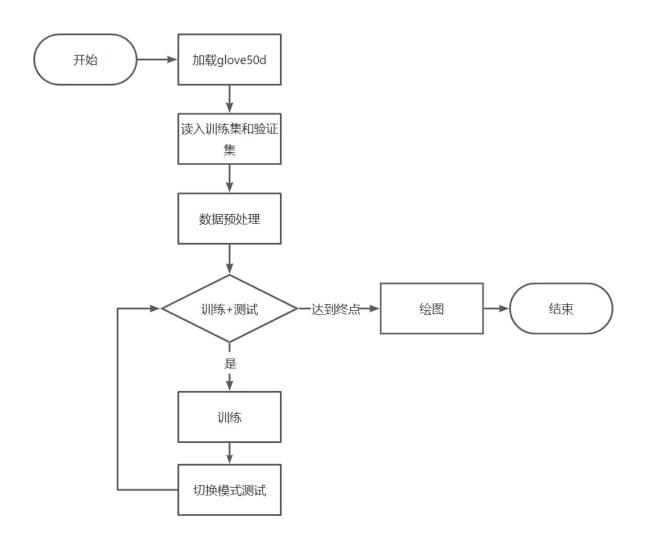
- **输入编码**:将前提和假设输入编码为向量表示。
- **前向传播**:将输入向量传递给模型,计算输出。
- 损失函数: 使用交叉熵损失函数计算预测结果与真实标签之间的误差。
- **反向传播**: 计算梯度并更新模型参数,以最小化损失函数。
- 超参数调整: 调整学习率、批量大小等超参数,以优化模型性能。

#### 6. 输出分析

训练完成后,对模型进行评估和分析,以了解其性能和改进方向。

• **评估指标**: 使用准确率、loss值等指标评估模型性能。

#### • 流程图



## 二、伪代码

Python

1	定义 preprocess_data 函数:
2	读取并处理数据
3	参数:
4	文件路径
5	句子最大长度
6	词嵌入模型
7	步骤:
8	初始化句子和标签列表
9	读取文件中的数据
10	获取英语停用词列表
11	遍历数据中的每一行:
12	对问题和句子进行分词
13	合并并处理分词结果,排除停用词,限制句子长度
14	将单词转换为词向量,若单词不在词嵌入模型中则用零向量代替
15	对齐句子长度,填充零向量
16	将处理后的句子和标签添加到对应列表中
17	返回值:
18	处理后的句子和标签列表
19	
20	# LSTM模型类
21	定义 LSTMModel 类:
22	初始化函数:
23	构建双向LSTM层,指定输入维度、隐藏层维度、层数
24	构建全连接层,指定输出维度
25	构建Dropout层
26	前向传播函数:
27	初始化LSTM层的隐藏状态和细胞状态
28	进行LSTM前向传播
29	获取最后时间步的输出
30	通过全连接层并进行Dropout
31	返回输出
32	
33	# 训练函数
34	定义 train 函数:
35	模型切换到训练模式
36	初始化损失值、正确预测数和总数
37	遍历数据加载器中的每个批次 <b>:</b>
38	获取输入和标签
39	梯度清零
40	进行前向传播,计算输出
41	计算损失

# 清極个epoch的平均损失和准确率	42	反向传播,更新模型参数
45       返回平均损失和准确率         46       # 评估函数         47       # 评估函数         48       定义 evaluate 函数:         49       模型切换到评估模式         50       被用格度计算:         51       禁用梯度计算:         52       油历数据加载器中的每个批次:         53       获取输入和标签         54       进行前向传播、计算输出         55       计算损失         56       影加损失值和正确预测数         57       计算每个epoch的平均损失和准确率         58       "训练文和准确率         60       # 训练文件         64       验证文件         65       句子最大长度         10       一方强大长度         11       创建训练和验证文件         66       北次大小         67       词嵌入模型         68       地次大小         69       步骤:         69       步骤:         70       调用 preprocess_data 函数处理训练和验证数据         60       地次大小         61       创建训练和验证技术         72       间域         73       打印訓练和选证样本数量         74       初始化長型参数:         75       输入程度         66       院域         77       原数         78	43	累加损失值和正确预测数
# 评估函数	44	计算每个epoch的平均损失和准确率
# 评估函数  定义 evaluate 函数:     模型切换到评估模式     初始化损失值、正确预测数和总数     禁用梯度计算:     遍历数据加载器中的每个批次:     获取输入和标签     进行前向传播,计算输出     计算极失值和正确预测数     计算每个epoch的平均损失和准确率     返回平均损失和准确率     滤回平均损失和准确率      维 训练ONLT模型函数     定义 train_qnli_model 函数:     参数:	45	返回平均损失和准确率
48       定义 evaluate 函数:         49       模型的换到评估模式         50       初始化损失值、正确预测数和总数         51       禁用梯度计算:         52       適历数据加载器中的每个批次:         53       获取输入和标签         54       进行前向传播, 计算输出         55       财场集份         56       累加损失值和正确预测数         57       计算每个epoch的平均损失和准确率         58       返回平均损失和准确率         60       # 训练ONLI模型函数         62       参数:         63       训练文件         64       验证文件         65       每分表大长度         66       批次大小         67       间嵌入模型         68       训练轮缴         69       步骤:         70       调解 preprocess_data 函数处理训练和验证数据         71       间域和数据加载证据         72       初始化数据加载证券         73       打印训练和验证样本数量         74       初始化模型参数:         75       输入维度         6       隐藏层维度         76       隐藏层线度         77       原数         78       输出维度         79       初始化STM模型         20       定义损失函数为交叉         30       加给化STM模型 <t< td=""><td>46</td><td></td></t<>	46	
49       模型切换到评估模式         50       初始化损失值、正确预测数和总数         51       禁用梯度计算:         52       遍历数据加载器中的每个批次:         53       获取输入和标签         54       进行前向传播,计算输出         55       计算级失         56       累加损失值和正确预测数         57       计算每个epoch的平均损失和准确率         58       返回平均损失和准确率         59       # 训练QNLT模型函数         62       参数:         63       训练文件         64       验证文件         65       可分最大长度         66       批次大小         67       河接入模型         68       步骤:         69       步骤:         60       步骤:         61       批次大小         62       加纳给税数据加载器, 指定批次大小         63       动始化数据加载器, 指定批次大小         73       打印训练和验证样本数量         74       初始化费型         75       输入维度         6       隐藏层维度         76       隐藏层维度         77       原数         78       输出建度         79       初始化STM模型         80       定义优化器为Adam         30       加给化STM模型         31	47	# 评估函数
50 初始化损失值、正确预测数和总数	48	定义 evaluate 函数:
51       禁用梯度计算:         52       遍历数据加载器中的每个批次:         53       获取输入和标签         54       进行前向传播, 计算输出         55       计算块生         56       累加损失和准确率         57       计算每个epoch的平均损失和准确率         58       返回平均损失和准确率         59       # 训练ONLI模型函数         61       定义 train_qnli_model 函数:         62       参数:         63       训练文件         64       验证文件         65       句子最大长度         66       批次大小         67       词嵌入模型         68       训练轮数         69       步骤:         70       调用 preprocess_data 函数处理训练和验证数据         61       创建训练和验证数据集         72       初始化数据加载器、指定批次大小         73       打印训练和验证样本数量         74       初始化模型参数:         75       输入维度         0       震動医生物、有限         76       隐藏层维度         77       原数         78       输出维度         79       初始化LSTM模型         79       初始化多数         79       初始化多数         70       市域         70       市域 <t< td=""><td>49</td><td>模型切换到评估模式</td></t<>	49	模型切换到评估模式
52	50	初始化损失值、正确预测数和总数
53       获取输入和标签         54       进行前向传播, 计算输出         55       累加损失值和正确预测数         57       计算每个epoch的平均损失和准确率         58       返回平均损失和准确率         59       # 训练QNLI模型函数         60       # 训练QNLI模型函数         62       参数:         63       训练文件         64       验证文件         65       句子最大长度         66       批次大小         67       词嵌入模型         68       训练轮数         69       步骤:         70       调用 preprocess_data 函数处理训练和验证数据         71       创建训练和验证数据集         72       初始化数据加载器, 指定批次大小         73       打印训练和验证样本数量         74       初始化模型参数:         75       输入维度         6       隐藏层维度         76       隐藏层维度         77       层数         78       输出维度         79       初始化LSTM模型         80       定义损失函数为交叉熵损失         81       定义优化器为Adam         初始化线为关和准确率列表         83       训练多个epoch:         84       记录开始时间	51	禁用梯度计算:
54       进行前向传播,计算输出         55       计算损失         56       累加损失值和正确预测数         57       计算每个epoch的平均损失和准确率         58       返回平均损失和准确率         59       # 训练QNLI模型函数         61       定义 train_qnli_model 函数:         62       参数:         63       训练文件         64       验证文件         65       句子最大长度         66       批次大小         67       词嵌入模型         68       训练轮数         69       步骤:         70       间用 preprocess_data 函数处理训练和验证数据         10       创建训练和验证数据集         72       初始化数据加载器, 指定批次大小         73       打印训练和验证样本数量         74       初始化模型参数:         75       输入维度         76       隐藏层维度         67       层数         78       输出维度         79       初始化长了M模型         80       定义损失函数为交叉熵损失         81       定义优化器为Adam         82       初始化线数Adam         30       训练多个epoch:         83       训练多个epoch:         84       记录开始时间	52	遍历数据加载器中的每个批次:
55       计算损失         56       累加损失值和正确预测数         57       计算每个epoch的平均损失和准确率         58       返回平均损失和准确率         59       # 训练ONLI模型函数         61       定义 train_qnli_model 函数:         62       参数:         63       训练文件         64       验证文件         65       句子最大长度         66       批次大小         67       词嵌入模型         68       训练轮数         69       步骤:         70       调用 preprocess_data 函数处理训练和验证数据         71       创始化数据加载器,指定批次大小         73       打印训练和验证样本数量         74       初始化模型参数:         75       输入维度         76       隐藏层维度         77       层数         78       输出生度         79       初始化LSTM模型         80       定义损失函数为交叉熵损失         81       定义损失函数为交叉熵损失         82       初始化透光和准确率列表         83       训练多个epoch:         84       记录开始时间	53	获取输入和标签
56       累加损失值和正确预测数         57       计算每个epoch的平均损失和准确率         58       返回平均损失和准确率         59       # 训练QNLI模型函数         61       定义 train_qnli_model 函数:         62       参数:         63       训练文件         64       验证文件         65       句子最大长度         66       批次大小         67       词嵌入模型         68       训练轮数         69       步骤:         70       调用 preprocess_data 函数处理训练和验证数据         71       创建训练和验证数据集         72       初始化数据加载器, 指定批次大小         73       打印训练和验证样本数量         74       初始化模型参数:         75       输入维度         76       隐藏层维度         77       层数         78       输出维度         79       初始化LSTM模型         80       定义损失函数为交叉熵损失         81       定义优化器为Adam         82       初始化损失和准确率列表         83       训练多个epoch:         84       记录开始时间		
57       计算每个epoch的平均损失和准确率         58       返回平均损失和准确率         59       # 训练QNLI模型函数         61       定义 train_qnli_model 函数:         62       参数:         63       训练文件         64       验证文件         65       句子最大长度         66       批次大小         67       词嵌入模型         68       训练轮数         69       步骤:         70       调用 preprocess_data 函数处理训练和验证数据         71       创建训练和验证数据集         72       初始化数据加载器, 指定批次大小         73       打印训练和验证样本数量         74       初始化模型参数:         75       输入维度         6       隐藏层维度         76       隐藏层维度         77       层数         78       输出维度         79       初始化LSTM模型         80       定义损失函数为交叉熵损失         81       定义损失函数为交叉熵损失         82       初始化损失和准确率列表         83       训练多个epoch:         84       记录开始时间		
58       返回平均损失和准确率         59       # 训练QNLI模型函数         61       定义 train_qnli_model 函数:         62       参数:         63       训练文件         64       验证文件         65       句子最大长度         66       批次大小         67       词嵌入模型         68       训练轮数         69       步骤:         70       调用 preprocess_data 函数处理训练和验证数据         71       创建训练和验证数据集         72       初始化数据加载器, 指定批次大小         73       打印训练和验证样本数量         74       初始化模型参数:         75       输入维度         76       隐藏层维度         77       层数         78       输出维度         79       初始化LSTM模型         80       定义损失函数为交叉熵损失         81       定义优化器为Adam         82       初始化损失和准确率列表         83       训练多个epoch:         84       记录开始时间		· · · · · · · · · · · · · · · · · · ·
59 60 # 训练QNLI模型函数 61 定义 train_qnli_model 函数: 62 参数: 63 训练文件 64 验证文件 65 句子最大长度 66 批次大小 67 词嵌入模型 3 训练轮数 69 步骤: 70 调用 preprocess_data 函数处理训练和验证数据 71 创建训练和验证数据集 72 初始化数据加载器,指定批次大小 73 打印训练和验证样本数量 74 初始化模型参数: 75 输入维度 76 隐藏层维度 77 层数 78 输出维度 79 初始化LSTM模型 80 定义损失函数为交叉熵损失 81 定义优化器为Adam 82 初始化损失和准确率列表 83 训练多个epoch: 84		·
60 # 训练QNLI模型函数 61 定义 train_qnli_model 函数: 62 参数: 63 训练文件 64 验证文件 65 句子最大长度 66 批次大小 67 词嵌入模型 68 训练轮数 69 步骤: 70 调用 preprocess_data 函数处理训练和验证数据 71 创建训练和验证数据集 72 初始化数据加载器,指定批次大小 73 打印训练和验证样本数量 74 初始化模型参数: 75 输入维度 76 隐藏层维度 77 层数 78 输出维度 79 初始化LSTM模型 80 定义损失函数为交叉熵损失 81 定义优化器为Adam 82 初始化摄失和准确率列表 83 训练多个epoch: 84 记录开始时间		返回平均损失和准确率
61 定义 train_qnli_model 函数: 62 参数: 63 训练文件 64 验证文件 65 句子最大长度 66 批次大小 67 词嵌入模型 68 训练轮数 69 步骤: 70 调用 preprocess_data 函数处理训练和验证数据 61 创建训练和验证数据集 72 初始化数据加载证器,指定批次大小 73 打印训练和验证样本数量 74 初始化模型参数: 75 输入维度 隐藏层维度 76 隐藏层维度 77 层数 输出维度 79 初始化LSTM模型 80 定义损失函数为交叉熵损失 81 定义优化器为Adam 82 初始化损失和准确率列表 33 训练多个epoch: 84 记录开始时间		
62 参数: 63 训练文件 64 验证文件 65 句子最大长度 66 批次大小 67 词嵌入模型 68 训练轮数 69 步骤: 70 调用 preprocess_data 函数处理训练和验证数据 71 创建训练和验证数据集 72 初始化数据加载器,指定批次大小 73 打印训练和验证样本数量 74 初始化模型参数: 75 输入维度 76 隐藏层维度 77 层数 78 输出维度 79 初始化LSTM模型 80 定义损失函数为交叉熵损失 81 定义优化器为Adam 82 初始化损失和准确率列表 83 训练多个epoch: 84 记录开始时间		
63 训练文件 64 验证文件 65 句子最大长度 66 批次大小 67 词嵌入模型 68 训练轮数 69 步骤: 70 调用 preprocess_data 函数处理训练和验证数据 71 创建训练和验证数据集 72 初始化数据加载器,指定批次大小 73 打印训练和验证样本数量 74 初始化模型参数: 75 输入维度 76 隐藏层维度 77 层数 78 输出维度 79 初始化LSTM模型 80 定义损失函数为交叉熵损失 81 定义优化器为Adam 82 初始化损失和准确率列表 83 训练多个epoch: 84 记录开始时间		-· -
64 验证文件 65 句子最大长度 66 批次大小 67 词嵌入模型 108 训练轮数 69 步骤: 70 调用 preprocess_data 函数处理训练和验证数据 71 创建训练和验证数据集 72 初始化数据加载器,指定批次大小 73 打印训练和验证样本数量 74 初始化模型参数: 75 输入维度 76 隐藏层维度 77 层数 78 输出维度 79 初始化LSTM模型 80 定义损失函数为交叉熵损失 81 定义优化器为Adam 82 初始化损失和准确率列表 83 训练多个epoch: 84 记录开始时间		
65 句子最大长度 66 批次大小 67 词嵌入模型 68 训练轮数 69 步骤: 70 调用 preprocess_data 函数处理训练和验证数据 71 创建训练和验证数据集 72 初始化数据加载器,指定批次大小 73 打印训练和验证样本数量 74 初始化模型参数: 75 输入维度 76 隐藏层维度 77 层数 78 输出维度 79 初始化LSTM模型 80 定义损失函数为交叉熵损失 81 定义优化器为Adam 82 初始化损失和准确率列表 83 训练多个epoch: 84 记录开始时间		
66		
67		
68		
69       步骤:         70       调用 preprocess_data 函数处理训练和验证数据         71       创建训练和验证数据集         72       初始化数据加载器,指定批次大小         73       打印训练和验证样本数量         74       初始化模型参数:         75       输入维度         76       隐藏层维度         77       层数         78       输出维度         79       初始化LSTM模型         80       定义损失函数为交叉熵损失         81       定义优化器为Adam         82       初始化损失和准确率列表         83       训练多个epoch:         84       记录开始时间		
70调用 preprocess_data 函数处理训练和验证数据71创建训练和验证数据集72初始化数据加载器,指定批次大小73打印训练和验证样本数量74初始化模型参数:75输入维度76隐藏层维度77层数78输出维度79初始化LSTM模型80定义损失函数为交叉熵损失81定义优化器为Adam82初始化损失和准确率列表83训练多个epoch:84记录开始时间		
71       创建训练和验证数据集         72       初始化数据加载器,指定批次大小         73       打印训练和验证样本数量         74       初始化模型参数:         75       输入维度         76       隐藏层维度         77       层数         78       输出维度         79       初始化LSTM模型         80       定义损失函数为交叉熵损失         81       定义优化器为Adam         82       初始化损失和准确率列表         83       训练多个epoch:         84       记录开始时间		
72 初始化数据加载器,指定批次大小 73 打印训练和验证样本数量 74 初始化模型参数: 75 输入维度 76 隐藏层维度 77 层数 78 输出维度 79 初始化LSTM模型 80 定义损失函数为交叉熵损失 81 定义优化器为Adam 82 初始化损失和准确率列表 83 训练多个epoch: 84 记录开始时间		
73 打印训练和验证样本数量 74 初始化模型参数: 75 输入维度 76 隐藏层维度 77 层数 78 输出维度 79 初始化LSTM模型 80 定义损失函数为交叉熵损失 81 定义优化器为Adam 82 初始化损失和准确率列表 83 训练多个epoch: 84 记录开始时间		
74       初始化模型参数:         75       输入维度         76       隐藏层维度         77       层数         78       输出维度         79       初始化LSTM模型         80       定义损失函数为交叉熵损失         81       定义优化器为Adam         82       初始化损失和准确率列表         83       训练多个epoch:         84       记录开始时间		
75 输入维度 76 隐藏层维度 77 层数 78 输出维度 79 初始化LSTM模型 80 定义损失函数为交叉熵损失 81 定义优化器为Adam 82 初始化损失和准确率列表 83 训练多个epoch: 84 记录开始时间		
76 隐藏层维度 77 层数 78 输出维度 79 初始化LSTM模型 80 定义损失函数为交叉熵损失 81 定义优化器为Adam 82 初始化损失和准确率列表 83 训练多个epoch: 64 记录开始时间		
77       层数         78       输出维度         79       初始化LSTM模型         80       定义损失函数为交叉熵损失         81       定义优化器为Adam         82       初始化损失和准确率列表         83       训练多个epoch:         84       记录开始时间		
78       输出维度         79       初始化LSTM模型         80       定义损失函数为交叉熵损失         81       定义优化器为Adam         82       初始化损失和准确率列表         83       训练多个epoch:         84       记录开始时间		
79       初始化LSTM模型         80       定义损失函数为交叉熵损失         81       定义优化器为Adam         82       初始化损失和准确率列表         83       训练多个epoch:         84       记录开始时间		
80       定义损失函数为交叉熵损失         81       定义优化器为Adam         82       初始化损失和准确率列表         83       训练多个epoch:         84       记录开始时间		
81       定义优化器为Adam         82       初始化损失和准确率列表         83       训练多个epoch:         84       记录开始时间		***************************************
82       初始化损失和准确率列表         83       训练多个epoch:         84       记录开始时间		
83   训练多个epoch:     84   记录开始时间	82	
84 记录开始时间	83	
85 调田 train 函数进行训练 萃取训练提生和准确家	84	记录开始时间
9万 いはは 四女人に 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	85	调用 train 函数进行训练,获取训练损失和准确率

86	调用 evaluate 函数进行评估,获取验证损失和准确率	
87	记录结束时间	
88	打印当前epoch的训练和验证损失、准确率及时间	
89	记录损失和准确率	
OW	海同训练和感证的提生 准确家	

## 三、关键代码展示

完整代码见../code

• 数据预处理函数preprocess\_data

1 \* def preprocess\_data(file\_path, max\_length, embedding\_model): 2 3 函数功能:一体化数据处理。读取数据,对数据进行预处理,词嵌入和数据对齐 :param file\_path: 文件路径 4 5 :param max\_length: 句子最大长度 --> 对于整个数据集, 我们将所有句子都填充 或截断到相同的长度60,再大可能会使数组过大而报错 :param embedding model: 词嵌入模型 6 ..... 7 8 sentences = []labels = []9 10 embedding dim = len(embedding model[next(iter(embedding model.key s()))]) 11 12 df = pd.read\_csv(file\_path, sep='\t', header=0, on\_bad\_lines='ski p') 13 14 stop words = set(stopwords.words('english')) # 获取英语停用词列表 15 for , row in df.iterrows(): question = nltk.word tokenize(row['question']) 16 17 hypothesis = nltk.word\_tokenize(row['sentence']) 18 sentence = question + hypothesis # 不做任何处理 # sentence = [word.lower() for word in question + hypothesi 19 # 转小写 sl 20 # sentence = [word for word in question + hypothesis if word. isalpha()] # 去掉标点符号 21 # sentence = [word for word in question + hypothesis if word. isalpha() and word not in stop words] # sentence = [word.lower() for word in question + hypothesis 22 if word not in stop words] # 排除停用词1 23 sentence = sentence[:max length] 24 25 sentence\_vectors = [embedding\_model[word] if word in embeddin g\_model else np.zeros(embedding\_dim) for word in sentence] 26 27 while len(sentence vectors) < max length:</pre> 28 sentence\_vectors.append(np.zeros\_like(sentence\_vectors [0])) 29 sentences.append(np.array(sentence\_vectors, dtype='float32')) 30 labels.append(int(row['label'] == 'entailment')) 31 32 return sentences, labels

#### • 定义Bi-LSTM

```
1 * class LSTMModel(nn.Module):
         def __init__(self, input_dim, hidden_dim, output_dim, n_layers):
 2 =
             super(LSTMModel, self).__init__()
 3
 4
             self.lstm = nn.LSTM(input_dim, hidden_dim, num_layers=n_layer
    s, bidirectional=True, batch_first=True)
             self.fc = nn.Linear(hidden dim * 2, output dim)
 5
             self.dropout = nn.Dropout(0.5)
 6
 7
         def forward(self, x):
 8 =
 9
             h0 = torch.zeros(self.lstm.num_layers * 2, x.size(0), self.ls
    tm.hidden_size).to(x.device)
             c0 = torch.zeros(self.lstm.num_layers * 2, x.size(0), self.ls
10
    tm.hidden_size).to(x.device)
             h_{stm}, _ = self.lstm(x, (h0, c0))
11
12
             h_{stm} = h_{stm}[:, -1, :]
             out = self.fc(h_lstm)
13
             out = self.dropout(out)
14
15
             return out
16
```

训练函数

Python

```
1 • def train_qnli_model(train_file, valid_file, max_length, batch_size,
    embedding_model, num_epochs):
         embedding_dim = len(embedding_model[next(iter(embedding_model.key
 2
    s()))])
 3
 4
        train_sentences, train_labels = preprocess_data(train_file, max_l
    ength, embedding model)
        valid_sentences, valid_labels = preprocess_data(valid_file, max_l
 5
    ength, embedding model)
 6
 7
         train dataset = list(zip(train sentences, train labels))
        valid_dataset = list(zip(valid_sentences, valid_labels))
 8
 9
         train loader = DataLoader(train dataset, batch size=batch size, s
10
    huffle=True)
        valid loader = DataLoader(valid dataset, batch size=batch size, s
11
    huffle=False)
12
13
         print('Number of training samples:', len(train_dataset))
        print('Number of validation samples:', len(valid_dataset))
14
15
         input_dim = embedding_dim
16
17
        hidden_dim = 128
18
        n_{ayers} = 2
19
         output dim = 2
20
21
        model = LSTMModel(input_dim, hidden_dim, output_dim, n_layers)
22
         criterion = nn.CrossEntropyLoss()
23
         optimizer = optim.Adam(model.parameters(), lr=0.0001)
24
25
         loss = []
26
         accuracy = []
27
28 -
         for epoch in range(num epochs):
29
             start time = time.time()
30
             train_loss, train_acc = train(model, train_loader, criterio
    n, optimizer)
             valid_loss, valid_acc = evaluate(model, valid_loader, criteri
31
    on)
32
             end time = time.time()
33
```

```
34
            print('Epoch:', epoch + 1)
            print('Training Loss:', train_loss, 'Training Accuracy:', tra
35
    in_acc)
            print('Validation Loss:', valid_loss, 'Validation Accurac
36
    y:', valid_acc)
            print('Time per epoch:', end_time - start_time, 'seconds')
37
38
             loss.append((train_loss, valid_loss))
39
             accuracy.append((train acc, valid acc))
40
41
42
         return loss accuracy
```

#### • 主函数

```
1 - def main():
2
        # 定义训练集和验证集文件路径
3
        train file = '../train 40.tsv'
        valid file = '../dev 40.tsv'
4
5
6
        # 定义句子最大长度和批次大小
7
        max_length = 50
        batch_size = 32
8
9
        num_epochs = 10
10
        # 加载GloVe词向量模型
11
        embedding model = {}
12
        with open('E:\glove.6B\glove.6B.50d.txt', 'r', encoding='utf-8')
13 -
    as f:
            for line in f:
14 -
                values = line.strip().split()
15
16
                word = values[0]
17
                vector = np.array(values[1:], dtype='float32')
                embedding_model[word] = vector
18
19
        embedding dim = len(embedding model[next(iter(embedding model.key
    s()))])
20
21
        loss, accuracy = train_qnli_model(train_file, valid_file, max_len
    qth, batch size, embedding model, num epochs)
22
        train loss, valid loss = zip(*loss)
23
        train acc, valid acc = zip(*accuracy)
24
        plot_curves(train_loss, train_acc, valid_loss, valid_acc)
25
```

#### 四、创新点&优化

### • 数据预处理的综合性:

代码中预处理数据函数preprocess\_data,对数据进行了分词、去停用词、词向量化以及对齐等一系列操作。这种综合性的预处理能提高数据的质量,进而提升模型的性能。

同时,与其他需要多个函数进行数据预处理的代码形式相比,一个函数封装更有代码可读性, 更规范。

#### • 双向LSTM网络结构:

使用双向LSTM网络(self.lstm = nn.LSTM(..., bidirectional=True)),能从两个方向捕捉序列信息,有助于提高模型对文本的理解和预测能力。

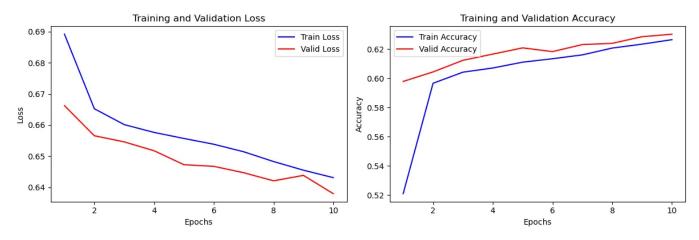
## 3 实验结果及分析

1. 实验结果展示, 评测指标分析

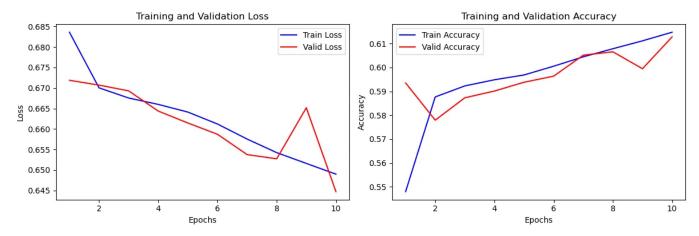
因为本人的电脑没有N卡,所以一轮训练和测试就要大约4-6分钟。因此epoch没有太大,定为10。

### 对以下预处理方式进行了多次测试,结果如下:

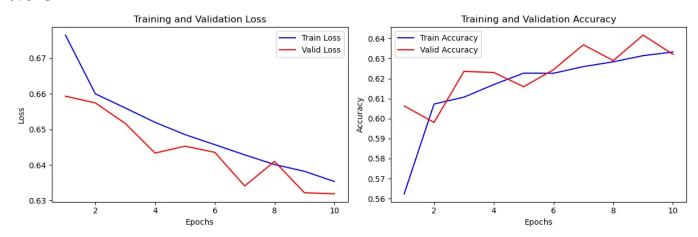
#### 不做处理:



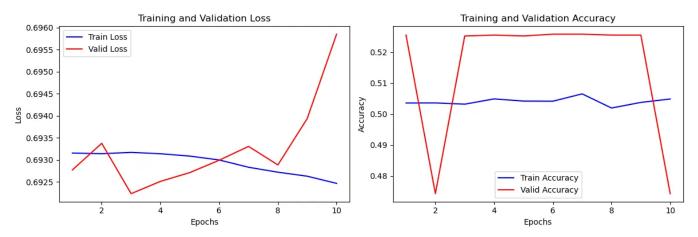
## 去掉停用词:



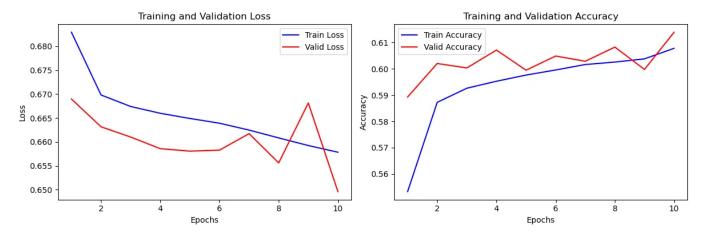
### 转小写:



## 去掉标点和停用词:



#### 去标点:



#### 汇总如下:

类型	不做处理	去掉停用词	转小写	去掉标点和停 用词	去掉标点
最大准确率	62.5%	61.3%	64.2%	52.7%	61.4%
波动情况	稳定	略微波动	略微波动	明显波动	略微波动
loss是否收敛	是	是	是	否	是

需要说明的是,因为运行一次程序所需时间对于我来说过长,所以我没有进行很多次的测试取最好结果。所以此结果只能说非常粗略。

但是可以确定的是,去掉标点符号的效果一定不算很好,因为glove词汇表里是有诸如逗号引号这样的标点的,去掉了反而不利于体现句子整体语义,去掉停用词导致效果变差的原因如下:

- **语义信息丢失**: 虽然停用词(如 "the", "is", "at")在很多情况下不携带重要的语义信息,但在某些上下文中,它们可能对理解句子的含义至关重要。同样,标点符号也可以携带语义信息,例如,问号可能表示一个句子是一个问题。
- **语境信息丢失:** 在某些情况下,即使单个词可能不携带很多信息,但是词与词之间的关系可能非常重要。例如,在否定句中,"not" 后面的词可能会改变句子的整体含义。

另外,在我的实验结果里,词汇转小写的最大准确率要更好,而不做任何处理所训练出来得到的结果没有过拟合,也没有明显波动。原因可能如下:

减少词汇表大小:如果不进行大小写转换,那么同一个词语的大小写形式会被视为两个不同的词语。这会导致词汇表的大小增加,从而增加模型的复杂性和训练难度。通过转化为小写,我们可以有效地减少词汇表的大小。

- 提高数据稀疏性:在自然语言处理中,数据稀疏性是一个常见的问题。如果我们将大小写视为不同的词语,那么一些词语可能只在某些特定的情况下(例如句首)以大写形式出现,这会导致这些词语的出现次数变少,从而增加数据的稀疏性。通过转化为小写可以有效地减少数据的稀疏性。
- **消除噪声:** 在一些情况下,词语的大小写形式可能会受到打字错误或其他非语言因素的影响。将所有的词语转化为小写可以消除这些噪声,从而提高模型的准确性。

## 时空复杂度分析:

• 时间复杂度:

假设数据集有 N 行、每行的最大长度为 L、词嵌入的维度为 D。

- 读取和处理数据:需要遍历每一行,处理每个单词。时间复杂度为 O(N×L)。
- 分词和去停用词: 假设分词和去停用词操作的时间复杂度为 O(L), 那么总的时间复杂度也是 O(N×L)。
- 词向量转换:对每个单词查找词向量,时间复杂度为 O(L×D),总的时间复杂度为 O(N×L×D)。

综上,数据预处理的总时间复杂度为 O(N×L×D)。

- 空间复杂度:
- 存储数据:需要存储处理后的数据,每个句子的长度为 L,词嵌入的维度为 D,总空间复杂度为 O(N×L×D)。

总体空间复杂度为 O(N×L×D)。

## 4 思考题

无

## 5 参考资料

实验课PPT

●基于pytorch构建双向LSTM(Bi-LSTM)文本情感分类实例(使用glove词向量)\_pytorch实现基于 双向lstm模型完成文本分类任务-CSDN博客 ●glove使用以及词表文件详解——小白必看,通俗易懂\_glove.6b-CSDN博客