**常州工学院**

**计算机信息工程学院**

**《数据结构》课程设计报告**

**题****目**  神经网络模型

**年　　级** 2022级软件工程

**班　　级** 22软件三

**组长姓名学号** 杨熙承 (22030531)

**组员姓名学号** 王语桐 (22030527)

**组员姓名学号**  林 越 (22030519)

**组员姓名学号**  徐姜旸 (22030528)

**指导教师** 叶　鸿

2024 **年** 1 **月** 12 **日**

常州工学院计算机信息工程学院

《数据结构》课程设计

任 务 书

设计名称： 神经网络模型

指导教师： 叶　鸿 下达时间： 20231225

学生姓名： 杨熙承 学 号：  22030531

学生姓名： 王语桐 学 号： 22030527

学生姓名： 林 越 学 号： 22030519

学生姓名： 徐姜旸 学 号： 22030528

专 业： 软件工程

一、课程设计的基本要求

编写一个应用程序，实现一个简单的神经网络模型。收集一定数量的诗歌数据集，对收集到的诗歌数据进行预处理，使用Python构建循环神经网络模型，使用训练数据对循环神经网络模型进行训练，利用测试数据对训练好的模型进行评估。使用训练好的模型，输入一个给定的藏头诗，通过模型生成一首完整的诗歌。

二、课程设计的主要内容（包含分工）

通过算法实现了一个神经网络模型。

小组成员具体分工如下：

杨熙承（组长）：主要代码编写，RNN模型搭建，前后端搭建

王语桐：对代码进行重构；搜集相关RNN的资料和公式；对大报告进行文案撰写以及排版和插图；公式和数据的可视化

徐姜旸：解释翻译题目并进行需求分析，对代码进行测试；搜集古诗数据并且做预处理；使用latex对大报告公式的主要撰写

林越：代码反应公式整理；对代码进行RNN与LSTM的神经网络模型可视化图的设计；并做大报告的主要文字撰写，文案设计

三、课程设计的进程安排

|  |  |
| --- | --- |
| **时间** | **安排** |
| 2024年1月1日之前 | 确定选题和组建团队 |
| 2023年1月1日–1月2日 | 实施课程设计：完成功能需求设计和开发环境 |
| 2023年1月3日–1月11日 | 编码实现 |
| 2023年1月12日 | 答辩验收 |

**2024年1月12日**

《数据结构》课程设计报告

1. 目的

此处概述课程设计的选题名称、工作内容和工作目标。

选题名称即报告标题。

工作内容即本次课程设计打算完成的功能、模块或打算实现的性能指标。

工作目标即预期得到一个什么样的结果，是实现一种算法，还是完成一个应用程序？

选题名称：

工作内容：

数据收集：首先，我们需要收集大量的藏头诗样本。这些样本将用于训练和验证循环神经网络。

数据预处理：对收集到的数据进行清洗和格式化，以便于输入到神经网络中。这可能包括去除无关字符、统一格式等步骤。

构建模型：使用循环神经网络构建模型，该模型能够学习藏头诗的内在结构和模式。

训练模型：使用处理过的数据对模型进行训练，使其能够自动生成藏头诗。

性能评估：通过对比生成的藏头诗与原始样本，评估模型的性能。

工作目标：

预期的结果是构建一个能够自动生成高质量藏头诗的模型。该模型不仅应遵循特定的格式，还应体现出一定的文学风格和意义。此外，我们也希望模型能够处理不同的主题，以展示其泛化能力。

二、需求分析

针对本次课程设计，逐一分析要实现的功能点。功能点是分解需求后得到的具体工作。

1. 读取诗歌数据：从CSV文件中读取包含诗歌内容的列，例如名为 content 的列。

2. 清理诗歌数据：去除诗歌文本中的非字母数字字符。

3. 构建词汇集：生成诗歌数据的词汇集，并建立字符到索引的映射关系。

4. 生成训练数据：将诗歌文本转换为神经网络的输入和输出数据。

5. 初始化神经网络参数：随机初始化神经网络的权重和偏置。

6. 前向传播和反向传播：实现神经网络的前向传播和反向传播，计算损失和梯度。

7. 训练神经网络：通过多次迭代，在给定训练数据上更新神经网络参数。

8. 生成藏头诗：根据用户输入的藏头字生成一首新的诗歌。

9. 用户交互：提供用户界面或命令行界面，接收用户输入，展示生成的诗歌。

10. 训练过程监控：在训练过程中实时监控损失值的变化，方便用户了解训练过程。

11. 超参数调整：允许用户调整神经网络的超参数，例如学习率、隐藏层大小等。

12. 测试条件设置：设置测试时的条件，包括数据集和神经网络参数。

13. 测试结果导出：将测试结果导出为文件，以便进行结果分析和比较。

三、概要设计

此处概述整个课程设计采用的方法和工具，并说明项目的模块划分及其依赖关系。例如，对于一个迷宫生成器而言，从功能点可以归纳出的模块有（仅供参考）：

概要设计：神经网络诗歌生成系统

方法和工具：

编程语言： Python

神经网络框架： Numpy（手动实现神经网络）

数据处理： CSV 模块用于读取诗歌数据

图形展示： Matplotlib 用于可视化训练过程和结果

用户交互： 命令行交互

模块划分及依赖关系：

1. 数据处理模块：

负责从CSV文件中读取诗歌数据。

依赖：CSV 模块

2. 文本清理模块：

清理诗歌文本，去除非字母数字字符。

依赖：数据处理模块

3. 词汇集生成模块：

生成诗歌数据的词汇集，并建立字符到索引的映射关系。

依赖：文本清理模块

4. 训练数据生成模块：

将诗歌文本转换为神经网络的输入和输出数据。

依赖：词汇集生成模块

5. 神经网络模块：

包含神经网络的前向传播和反向传播功能，以及参数的初始化。

依赖：训练数据生成模块

6. 训练模块：

负责多次迭代，在给定训练数据上更新神经网络参数。

实时监控损失值的变化，提供训练过程监控功能。

依赖：神经网络模块

7. 生成模块：

根据用户输入的藏头字生成一首新的诗歌。

依赖：神经网络模块

8. 用户交互模块：

提供用户界面或命令行界面，接收用户输入，展示生成的诗歌。

允许用户调整神经网络的超参数。

依赖：生成模块，训练模块

9. 测试模块：

设置测试时的条件，包括数据集和神经网络参数。

导出测试结果为文件，方便结果分析和比较。

依赖：训练模块

模块关系：

数据处理模块、文本清理模块、词汇集生成模块、训练数据生成模块在数据预处理阶段依次调用。

神经网络模块包含前向传播、反向传播功能，与训练模块协同工作。

训练模块在多次迭代中调用神经网络模块，同时提供实时监控功能。

生成模块负责根据用户输入生成诗歌，依赖神经网络模块。

用户交互模块与生成模块和训练模块交互，允许用户调整神经网络的超参数。

测试模块与训练模块协同工作，设置测试条件和导出测试结果。

设计原则：

模块独立性： 每个模块负责一个明确定义的功能，尽量减少模块之间的耦合。

可扩展性： 模块之间的依赖关系设计灵活，便于未来对系统功能的扩展和修改。

可维护性： 代码结构清晰，注释充分，便于团队协作和维护。

用户友好性： 用户交互模块提供友好的界面，允许用户通过命令行操作系统。

四、详细设计

此处详细说明课程设计各模块所用到的数据结构、算法思想，以及核心代码示例。

建议按功能模块分小节描述。例如，对于一个迷宫生成器而言，小节可以是以下示例。

1. 数据处理模块

1.1 数据类型

数据处理模块主要涉及文件读取，使用CSV模块读取数据，无需额外的自定义数据类型。

1.2 算法思想

无特定算法思想，使用CSV模块提供的函数进行文件读取。

1.3 核心代码示例

import csv

def read\_poems\_from\_csv(file\_path):

poems = []

with open(file\_path, 'r', encoding='utf-8') as file:

csv\_reader = csv.DictReader(file)

for row in csv\_reader:

poem\_content = row['content']

poems.append(poem\_content)

return poems

2. 文本清理模块 (text\_cleaning)

2.1 数据类型

文本清理模块主要涉及字符串处理，无需额外的自定义数据类型。

2.2 算法思想

使用正则表达式去除非字母数字字符。

2.3 核心代码示例

import re

def clean\_poems(poems):

cleaned\_poems = [re.sub(r'[^\w\s]', '', poem) for poem in poems]

return cleaned\_poems

3. 词汇集生成模块

3.1 数据类型

`vocab`: 集合，存储所有清理后的诗歌数据的字符。

`char\_to\_index`: 字典，将字符映射到索引。

`index\_to\_char`: 字典，将索引映射到字符。

3.2 算法思想

遍历所有清理后的诗歌数据，构建字符集合，并为每个字符分配唯一的索引。

3.3 核心代码示例

def generate\_vocabulary(cleaned\_poems):

vocab = set("".join(cleaned\_poems))

vocab\_size = len(vocab)

char\_to\_index = {char: i for i, char in enumerate(vocab)}

index\_to\_char = {i: char for i, char in enumerate(vocab)}

return vocab, char\_to\_index, index\_to\_char

4. 训练数据生成模块

4.1 数据类型

`X`: NumPy数组，存储训练数据的输入。

`y`: NumPy数组，存储训练数据的输出。

4.2 算法思想

遍历所有清理后的诗歌数据，将每个字符映射为其在词汇集中的索引，构建训练数据。

4.3 核心代码示例

import numpy as np

def generate\_training\_data(cleaned\_poems, char\_to\_index):

X = []

y = []

for poem in cleaned\_poems:

indices = [char\_to\_index[char] for char in poem]

for i in range(len(indices) 1):

if(i==0): continue

X.append(indices[i])

y.append(indices[i + 1])

X = np.array(X)

y = np.array(y)

return X, y

5. 神经网络模块

5.1 数据类型

`Wxh`, `Whh`, `Why`: NumPy数组，权重矩阵。

`bh`, `by`: NumPy数组，偏置向量。

`hs`, `ys`, `ps`: 字典，存储中间结果。

`loss`: 损失值。

5.2 算法思想

使用反向传播算法更新权重和偏置。

使用tanh作为激活函数。

5.3 核心代码示例

def forward\_backward(inputs, targets, hprev, Wxh, Whh, Why, bh, by):

#省略前向传播和反向传播的具体实现

#返回损失值和更新后的参数

return loss, dWxh, dWhh, dWhy, dbh, dby, hs[len(inputs)-1]

6. 训练模块

6.1 数据类型

`losses`: 列表，存储每次迭代的损失值。

`num\_losses`: 列表，存储每100次迭代的损失值。

6.2 算法思想

多次迭代，在给定训练数据上更新神经网络参数。

实时监控损失值的变化。

6.3 核心代码示例

def train(data, iter\_num, Wxh, Whh, Why, bh, by, char\_to\_index):

#省略训练过程的具体实现

#更新权重和偏置，监控损失值

print(f"Epoch {n}, Loss: {loss}")

7. 生成模块 (poem\_generation)

7.1 数据类型

`start\_char`: 字符，用户输入的藏头字。

`length`: 整数，生成诗歌的长度。

7.2 算法思想

根据用户输入的藏头字生成一首新的诗歌。

7.3 核心代码示例

def generate\_line(start\_char, length, Wxh, Whh, Why, bh, by, char\_to\_index, index\_to\_char):

省略生成诗歌的具体实现

return start\_char + generated\_poem[1:]

8. 用户交互模块 (user\_interaction)

8.1 数据类型

`input\_chars`: 字符串列表，用户输入的藏头字。

8.2 算法思想

提供用户界面或命令行界面，接收用户输入，展示生成的诗歌。

允许用户调整神经网络的超参数。

8.3 核心代码示例

def user\_interaction():

省略用户交互的具体实现

接收用户输入，展示

生成的诗歌，允许用户调整超参数

for char in input\_chars:

generated\_line = generate\_line(char, length=5)

print(generated\_line)

9. 测试模块 (testing)

9.1 数据类型

`test\_conditions`: 字典，存储测试时的条件，包括数据集和神经网络参数。

9.2 算法思想

设置测试时的条件，包括数据集和神经网络参数。

导出测试结果为文件，方便结果分析和比较。

9.3 核心代码示例

def run\_tests(test\_conditions, Wxh, Whh, Why, bh, by, char\_to\_index):

省略测试过程的具体实现

设置测试条件，导出测试结果

以上为每个模块的详细设计，具体实现时可根据需求和实际情况进行调整。以下略过。根据实际情况组织报告的内容。

五、调试分析

此处详细说明课程设计各模块的调试方法和调试过程。在调试过程中遇到的问题，也可以在本节阐述。

六、测试结果

此处详细说明课程设计各模块、各界面的测试过程和测试结果。如果使用自动化测试，则说明自动化脚本编写过程。

七、用户使用说明

此处从用户的角度，详细说明所编写的课程设计应该如何使用。可以按照需求用例撰写说明书。

八、总结

此处回顾本次课程设计流程，总结从课程设计习得的经验教训。可以展开描述本次实现的作品有什么不足，以及未来讲如何改进。

在开始阶段，我们明确了藏头诗创作的任务目标，并对其背景进行了深入的研究。藏头诗是一种具有特定格式和要求的文学形式，要求每行诗的第一个字或词合起来，能形成一句话或者传达某种特定的信息。

数据收集与预处理中，我们重点进行了藏头诗数据的收集，并进行了必要的预处理工作，如数据清洗、格式统一等，为后续的模型训练做准备。

基于对任务需求和数据特点的分析，我们选择了循环神经网络作为基础架构。利用预处理后的数据，我们对模型进行了训练。在训练过程中，我们不断调整超参数、优化损失函数，并采用有效的学习策略，以使模型性能达到最优。

在模型训练完成后，我们对生成的结果进行了评估。通过与真实数据集进行对比，我们评估了模型的准确率、召回率等指标，并根据评估结果对模型进行了进一步的优化。

最后，我们将训练好的模型集成到一个应用程序中，用户可以通过输入主题，自动生成藏头诗。这个应用程序不仅是一个学术研究的展示，也可以作为娱乐工具，供人们欣赏和创作藏头诗。

附录

附全套源代码，或项目托管仓库（如GitHub）的访问链接。

源代码用等宽字体（例如Consolas）表示，必要的关键功能点应作注释。