# 自然语言处理实验报告

基于RNN与CNN的词向量训练与对比分析

姓名：张宗梓

学号：2023K8009991013

专业：人工智能

2025 年 12 月 22 日

# 目录

# 1　实验目的与任务

本实验旨在通过构建神经网络语言模型，利用《人民日报》语料库训练汉语词向量，并对比分析循环神经网络（RNN）与卷积神经网络（CNN）在特征提取和词向量生成上的差异。

主要任务包括：

1. 数据预处理：清洗语料，构建词表，将文本转换为索引序列。
2. 模型构建：分别搭建 RNN 和 CNN 模型。
3. 词向量提取：从 Embedding 层获取词向量。
4. 对比分析：计算并对比两个模型生成的词向量在语义相似度上的表现。

# 2　实验原理与设置

## 2.1　模型架构

### 2.1.1　RNN 模型

RNN 模型通过捕捉序列的时间依赖性来学习语义。本实验采用单向 RNN，结构如下：

- **Embedding 层**：将词索引映射为稠密向量。
- **RNN 层**：处理输入序列，保留上下文信息。
- **全连接层**：将最后一个时间步的隐状态映射为词表概率分布。

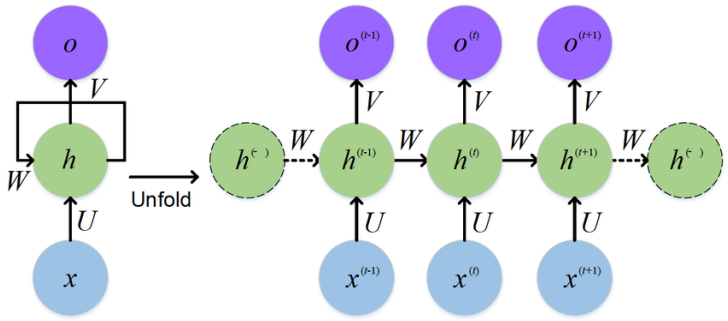



图 1: RNN 模型结构示意图

### 2.1.2 CNN 模型

CNN 模型通过滑动窗口提取局部特征。本实验结构如下：

- **Embedding 层**：同上。
- **Conv1d 层**：一维卷积，核大小为 2 。
- **Max Pooling 层**：全局最大池化，提取最显著特征。
- **全连接层**：输出预测概率。

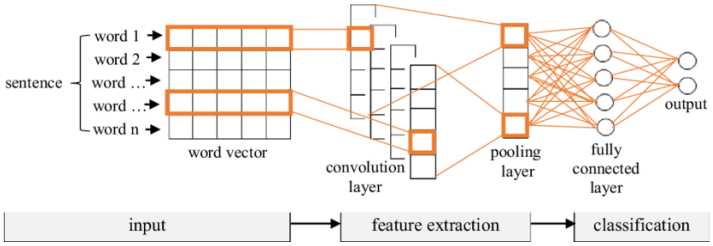



图 2: CNN 模型结构示意图

## 2.2 实验参数设置

表 1: 实验超参数设置

| 参数名 | 数值 |
|---|---|
| 词表大小 | 1000 |
| 词向量维度 | 15 |
| 输入序列长度 | 5 |
| 批大小 | 64 |
| 学习率 | 0.001 |
| 训练轮数 | 30 |

# 3 实验步骤与具体实现

## 3.1 数据预处理

利用 Python 对《人民日报》1998年1月语料进行清洗。

- **去噪**：剔除词性标记、数字、标点符号及非中文字符，仅保留纯汉字实词。
- **截断**：根据词频统计，保留前 1000 个高频词，其余标记为 `<UNK>`。
- **数据集构建**：采用滑动窗口法，取前 5 个词预测第 6 个词。

## 3.2 核心代码实现

下为 RNN 模型定义的关键代码片段：

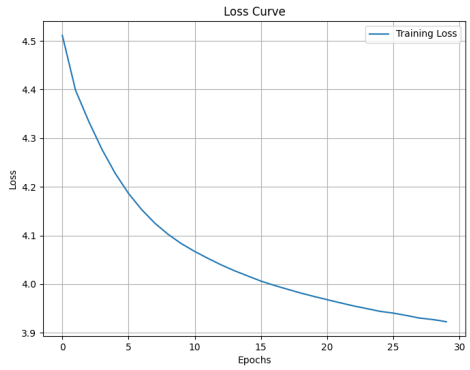
```
class RNNModel(nn.Module):
    def __init__(self, vocab_size, embed_dim, hidden_dim):
        super(RNNModel, self).__init__()
        self.embedding = nn.Embedding(vocab_size, embed_dim)
        self.rnn = nn.RNN(embed_dim, hidden_dim, batch_first=True)
        self.fc = nn.Linear(hidden_dim, vocab_size)

    def forward(self, x):
        embeds = self.embedding(x)
        out, _ = self.rnn(embeds)
        return self.fc(out[:, -1, :])
```

下为 CNN 模型定义的关键代码片段：

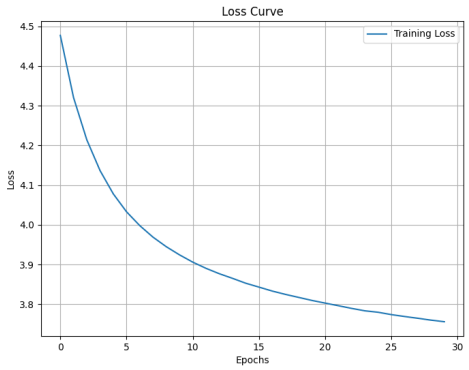
```
class CNNModel(nn.Module):
    def __init__(self, vocab_size, embed_dim, seq_len):
        super(CNNModel, self).__init__()
        self.embedding = nn.Embedding(vocab_size, embed_dim)
        self.conv1 = nn.Conv1d(in_channels=embed_dim, out_channels=32, kernel_size=2)
        self.pool = nn.AdaptiveMaxPool1d(1)
        self.fc = nn.Linear(32, vocab_size)
    def forward(self, x):
        embeds = self.embedding(x)
        embeds = embeds.permute(0, 2, 1)
        x = torch.relu(self.conv1(embeds))
        x = self.pool(x).squeeze(-1)
        return self.fc(x)
```

# 4　实验结果与分析

# 5　训练损失曲线




(a) CNN 训练损失曲线




(b) RNN 训练损失曲线

图 3: 训练损失曲线对比

在训练30批次的情况下，RNN的损失更低，故认定RNN的词向量效果更好。

## 5.1　词向量展示

表 2: CNN与RNN模型词向量示例（前8维）

| 词 | $d_1$ | $d_2$ | $d_3$ | $d_4$ | $d_5$ | $d_6$ | $d_7$ | $d_8$ |
|---|---|---|---|---|---|---|---|---|
| *CNN 模型* | | | | | | | | |
| 中国 | -1.02 | 0.74 | 0.56 | -0.32 | -0.55 | -1.07 | -0.86 | -0.67 |
| *RNN 模型* | | | | | | | | |
| 中国 | 0.17 | 1.34 | 0.77 | -2.23 | -0.17 | -2.03 | -0.28 | 0.93 |

## 5.2　词向量对比分析

我们选用例如“中国”、“经济”、“发展”等高频词进行对比分析。