

# 实验报告：词向量训练

Illusionna 2025XXXX04 人工智能学院

2025 年 10 月 25 日

## 1 模型结构

采用“*A neural probabilistic language model*”论文中的 NNLM 神经网络语言模型，一共分为三大层：特征层，隐藏层，输出层。

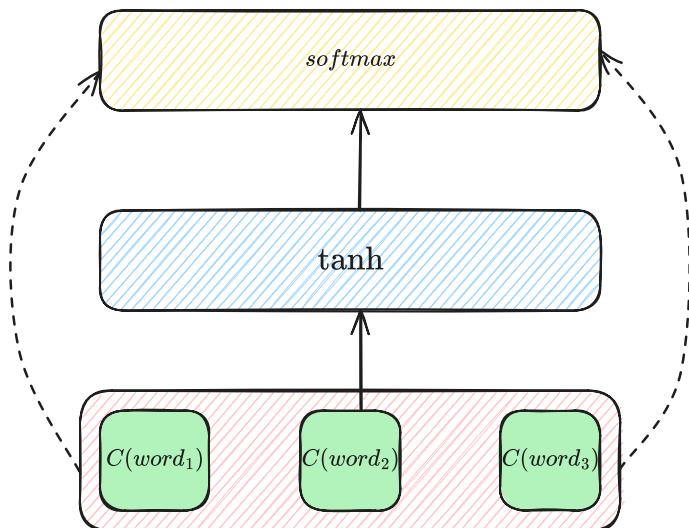


图 1: NNLM 模型架构

其中，参数空间  $\Theta$  如下：

$$\Theta = (\mathbf{b}, \mathbf{d}, \mathbf{W}, \mathbf{U}, \mathbf{H}, \mathbf{C})$$

$\mathbf{C}$  是词嵌入矩阵，样本量  $|V|$  是词汇表包含词的总个数，嵌入维度是常数  $m$  维，本次作业的目标即训练词向量矩阵。特征层  $\mathbf{W}$  是  $m(n - 1) \times |V|$  维度。隐藏层  $\mathbf{H}$  是  $m(n - 1) \times h$  维度。输出层  $\mathbf{U}$  是  $h \times |V|$  维度。已知前  $n - 1$  个单词，预测第  $n$  个单词。模型的输入输出满足论文核心公式：

$$\mathbf{y} = \mathbf{b} + \mathbf{Wx} + \mathbf{U} \tanh(\mathbf{d} + \mathbf{Hx})$$

- $h$ : 隐藏层神经元的个数.
- $m$ : 词向量长度.
- $\mathbf{b}$ : 输出层偏置,  $|V|$  个元素的向量.
- $\mathbf{d}$ : 隐藏层偏置, 向量元素个数与隐藏层神经元个数相同.

## 2 训练方式

### 计算设备

```
1 device = torch.device('mps' if platform.system() == 'Darwin' else  
                      'cuda' if torch.cuda.is_available() else 'cpu')
```

### 损失函数

```
1 criterion = torch.nn.CrossEntropyLoss()
```

### 优化器

```
1 optimizer = torch.optim.Adam(  
2     params = model.parameters(),  
3     lr = 1e-3,  
4     eps = 1e-8  
5 )
```

### 超参数

```
1 epoch = 1000  
2 m = 128  
3 h = 12  
4 n = 8
```

A screenshot of a macOS desktop environment. At the top, there's a menu bar with '访达' (Finder), '文件' (File), '编辑' (Edit), '显示' (View), '前往' (Go), '窗口' (Windows), and '帮助' (Help). The main window title is 'Homework (SSH: A6000)'. Below the title bar, there's a search bar and several small icons. The main content area contains a terminal window with Python code for training a neural network. The code includes imports for torch, defines a 'Train' class with a 'zh' method, and handles CSV file operations. It also includes a 'main' function that checks the platform and sets the device accordingly. Below the terminal, there's a file browser titled '资源管理器' (Finder) showing files like 'main.py', 'en\_embedding\_result.csv', and 'zh\_embedding\_result.csv'. The bottom of the screen features the Dock with icons for Finder, Mail, Safari, and other applications. On the right side, there are system status icons for battery level (7%), RAM usage (61%), signal strength, and the date/time (10月24日 周五 17:05).

图 2: 训练结果

### 3 遇到的问题与思考

## 问题一 Tokenizer

英文句子天然以空白分隔开来，因此很简单地就能将它们分成若干个单词，形成 LLM 输入的 token. 但中文句子字符串是连续的，给定的中文数据集是已经处理好，具备空白的句子，然而真实情况下是未经处理的连续文本字符串. GitHub 有一个 Jieba 项目，适用于中文句子分词，可以借鉴马尔可夫链 Viterbi 算法进行分词.

## 问题二    Serialization

一句话是具有一定 token 长度的, 倘若既定的 target 输出是第  $n$  个 token, 但超出了句子总长度, 这样是无法构成  $(\mathbf{x}, \mathbf{y}, \text{target})$  元组. 一种处理方式是过滤掉超过句子总长度太短的, 还有种方式是进行 padding 模式. 譬如, 句子头部增加标记符 |<BOS>|, 句子尾部增加标记符 |<EOS>|, 使得总长度达到  $n$ , 构成可训练的元组  $(\mathbf{x}, \mathbf{y}, \text{target})$ .