# 2023-2024学年第2学期
# 《自然语言处理导论》
# 实验3：词嵌入和RNN

专业：计算机科学与技术
学号：210340170
姓名：薛皓天
教师：卢敏

## 一、实验目的

1、利用已有代码，能运行word2vec，训练模型，并为每个词(term)生成embedding(嵌入向量）。

2、调用PCA代码，为d维的嵌入向量映射到2维。

3、利用词嵌入向量，结合RNN（循环神经网络）做文本分类。

## 二、实验内容

### 1、word2vec

具体的word2vec模型可以分为两种，一种是跳字模型（skip-gram），具体思路可以表达为：基于某一个词来生成它在文本序列周围的词，生成距离不超过n的词的条件概率，并认为每个词相互独立，用连乘来估计最大似然函数，求解目标函数就是最大化似然函数。另一种是连续词袋模型（CBOW）和上述的模型正好是反过来的，用多个背景词来预测一个中心词，CBOW对小型数据库比较合适，而Skip-Gram在大型语料中表现更好。对于中文语料先使用jieba分词进行分词，然后带入模型进行词向量的求解，而对于英文语料直接通过空格即可进行分词。

实验word2vec使用一层嵌入层也就是隐藏层和一层全连接层就是输出层输出词向量的维度，如下Listing1即就是对应的实现，如图Figure1为word2vec的输出可视化结果。

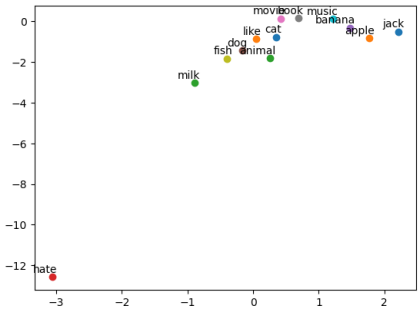


Figure 1: word2vec可视化结果

**Listing 1** 模型定义

```
class Word2Vec(nn.Module):

    def __init__(self):
        super(Word2Vec, self).__init__()
        # W: one-hot到词向量的hidden layer
        self.W = nn.Parameter(torch.randn(voc_size, embedding_size))
        # V: 输出层的参数
        self.V = nn.Parameter(torch.randn(embedding_size, voc_size))


    def forward(self, X):
        hidden_layer = torch.matmul(X, self.W)
        output_layer = torch.matmul(hidden_layer, self.V)
        return output_layer
```

## 2、PCA降维，显示结果

使用gensim提供的word2vec进行词向量的获取得到100维的一个词向量，然后使用sklearn提供的pca算法进行降维，将坐

标显示到坐标系中。如Listing2 所示维PCA的代码，如Listing3所示gensim的word2vec模型。如图Figure2为所示的PCA降维的结果。

{'布鲁是': 0, '饮毕': 1, '袭镇': 2, 'Middlebourne': 3, '克肖德': 4, '酷旱': 5, '小
示': 6, '圆桌会议': 7, 'DiDaDi': 8, '乔一琦': 9, '☆': 10, '命康义诚': 11, '44724':
12, '数来': 13, '金盈': 14, '余场': 15, '二维码': 16, 'Hilara': 17, 'MIYAKO': 18,
'朱士烈': 19, 'ErbB2': 20, '属泉类': 21, 'Maisie': 22, '牛金引': 23, '有齐': 24, '
宽法': 25, '125IN': 26, '拜帖': 27, '刘德旺': 28, '平知章': 29, '生阿努': 30,
'D10': 31, '经向外': 32, '白带': 33, 'MSIN': 34, '塑胶管': 35, '记为': 36, '三密':
37, '乐倍': 38, '遵外': 39, 'ElGamal': 40, '计而': 41, '序例': 42, '时和式': 43, '
施岗站': 44, '唇脂': 45, '郑桓公在': 46, '箴铭类': 47, '若以': 48}

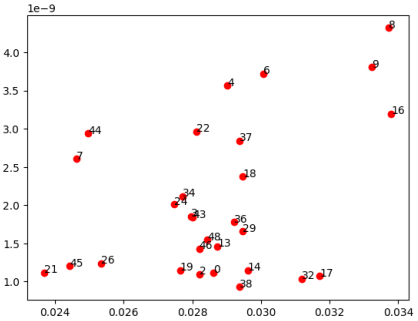


Figure 2: PCA结果

**Listing 2** PCA

```
from sklearn.decomposition import PCA
import numpy as np
pca = PCA(n_components=2)
d = {}
for idx , i in enumerate(dic):
    d[i] = pca.fit_transform(vector[i].reshape((2 , 50)))[0]
```

**Listing 3** gensim中word2vec模型

```
from gensim.models import Word2Vec
from gensim.models.word2vec import LineSentence

model = Word2Vec(
    LineSentence(open('./text/text.txt', 'r', encoding='utf8')),
    sg = 0, # sg=1是skip-gram算法，对低频词敏感；默认sg=0为CBOW算法
    # size = 100, # size是输出词向量的维数
    window = 3, # window是句子中当前词与目标词之间的最大距离
    min_count = 1 # min_count是对词进行过滤
)

# 词向量保存
model.wv.save_word2vec_format('data.vector', binary=False)

# 模型保存
model.save('./text/test.model')
```

## 3、RNN分类

循环神经网络 (RNN) 是一种使用序列数据或时序数据的人工神经网络。 这些深度学习算法常用于顺序或时间问题，如语言翻译、自然语言处理 (nlp)、语音识别、图像字幕等；它们包含在一些流行的应用中，比如 Siri、语音搜索和 Google Translate。 与前馈神经网络和卷积神经网络 (CNN) 一样，循环神经网络利用训练数据进行学习。 区别在于“记忆”，因为它从先前的输入中获取信息，以影响当前的输入和输出。 虽然传统的深度神经网络假设输入和输出相互独立的，但循环神经网络的输出依赖于序列中先前的元素。 尽管未来的活动也可能有助于确定特定序列的输出，但是单向循环神经网络无法在预测中说明这些事件。如下图Figure3，所示为RNN网络最基本的结构。如图Figure4，所示为

模型训练的准确率曲线，如图Figure5，所示为模型训练的损失曲线。

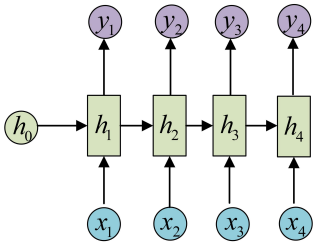


Figure 3: RNN


Figure 4: acc

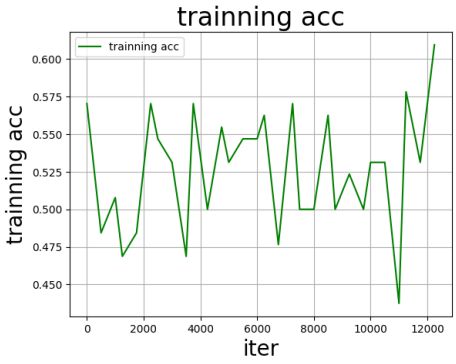


Figure 5: loss

## 三、实验结果和结论

通过使用Gensim库中的Word2Vec模型，我们成功地从文本数据中学习到了词嵌入向量。这些向量捕捉了单词之间的语义关系，使得相似的词在向量空间中位置相近。词嵌入的生成是自然语言处理任务中非常重要的一步，它为后续的任务提供了丰富的语义信息。为了直观地展示词嵌入的效果，我们利用PCA（主成分分析）技术将高维的词嵌入向量降维至二维空间，并使用matplotlib库进行可视化。通过可视化结果，我们可以观察到相似的词在二维空间中聚集在一起，这验证了Word2Vec模型的有效性。在文本分类任务中，我们利用生成的词嵌入向量作为RNN（循环神经网络）模型的输入。RNN模型能够捕捉文本中的序列信息，并结合词嵌入向量中的语义信息，对文本进行准确的分类。通过模型的训练和评估，我们验证了基于词嵌入和RNN的文本分类方法的有效性。

## 四、实验过程分析与建议

对数据进行深入探索，理解数据的分布、特性以及潜在的问题。确保数据的准确性和一致性对于后续的模型训练至关重要。在预处理阶段，除了基本的分词、去除停用词和转换为小写外，还可以考虑使用词干提取或词形还原来进一步减少词汇表的规模，并提升模型的泛化能力。在训练Word2Vec模型时，根据任务和数据集的特点，选择合适的参数配置，如窗口大小、最小词频、迭代次数等。这些参数会直接影响词嵌入的质量和模型的性能。监控模型的训练过程，包括损失函数的变化和词汇在嵌入空间中的分布，以便及时调整参数和优化模型。除了Word2Vec外，还可以探索其他词嵌入技术，如GloVe、FastText等。这些技术具有不同的特点和优势，可能更适合你的具体任务和数据集。在文本分类任务中，除了基本的RNN模型外，还可以尝试使用更复杂的模型架构，如LSTM、GRU、Transformer等。这些模型具有更

强的捕捉序列信息和长期依赖关系的能力。同时，可以探索使用深度学习中的集成学习、多任务学习等技术来进一步提升模型的性能。