自然语言处理

自然语言和单词的分布式表示

- 语料库的准备
-

```
test = 'You say goodbye and I say hello.'
test = test.lower()
test = test.replace('.', ' .')

#分词
words = test.split(' ')

word_to_id = {}
id_to_word = {}

for word in words:
    if word not in word_to_id:
        new_id = len(word_to_id)
        word_to_id[word] = new_id
        id_to_word[new_id] = word

print(id_to_word) # {0: 'you', 1: 'say', 2: 'goodbye', 3: 'and', 4: 'i', 5:
'hello', 6: '.'}
print(word_to_id) # {'you': 0, 'say': 1, 'goodbye': 2, 'and': 3, 'i': 4, 'hello':
5, '.': 6}

#将单词转换为单词ID列表
import numpy as np
corpus = [word_to_id[w] for w in words]
corpus = np.array(corpus)
print(corpus) # [0 1 2 3 4 1 5 6]
```

- 共现矩阵
- 利用共现矩阵求余弦相似度（分母加上 eps=10^{-8}防止为0）

- 点互信息（PMI），正的点互信息（PPMI）
- PPMI矩阵-SVD降维
- PTB语料库
- 单词的分布式表示（每个单词表示为固定长度的密集向量）：用语料库 - 计算上下文中单词数量，创建共现矩阵 - 转换为PPMI矩阵 - 基于SVD降维 - 得到单词向量

Word2vec

- 是一种基于推理的方法-预测
- CBOW模型-根据上下文预测目标词的神经网络
- skip-gram 模型：从目标词推断上下文
- Embedding层
  - 从权重参数抽取 '单词id对应行向量'
- 负采样-用二分类拟合多分类
- 基于向量的加减法来解决类推问题
- 迁移学习能力o(￣▽￣)ｄ

RNN

- 前馈神经网络不能很好的处理时间序列的数据
- 语言模型给出了单词序列发生的概率
- RNN有两个权重：
  - 将输入x转为输出h的权重w_x
  - 将自己上一个时刻的输出转化为当前时刻输出的权重w_h
- BPTT 基于时间的反向传播
- 处理长数据：Truncated BPTT（在时间序列过长处截断反向链接）
- Truncated BPTT的mini-batch学习
- 基于RNN的模型：RNNLM

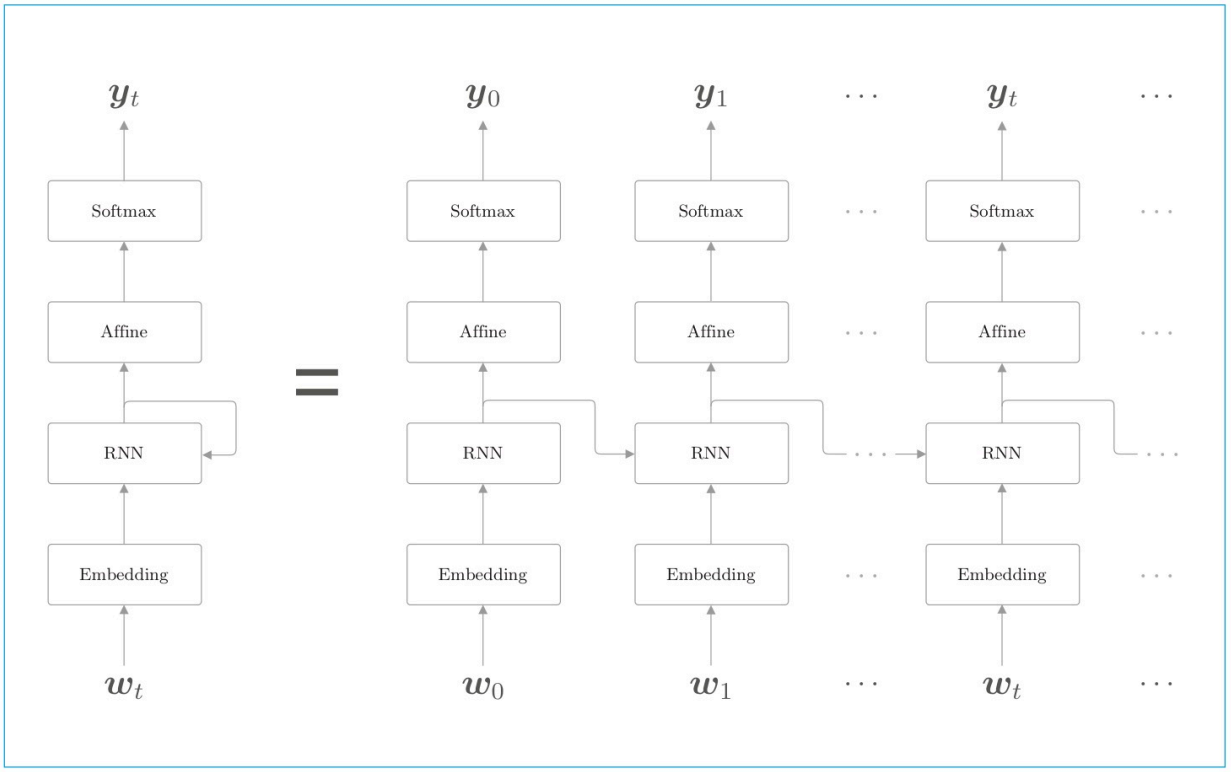

图5-25　RNNLM的网络图（左图是展开前，右图是展开后）

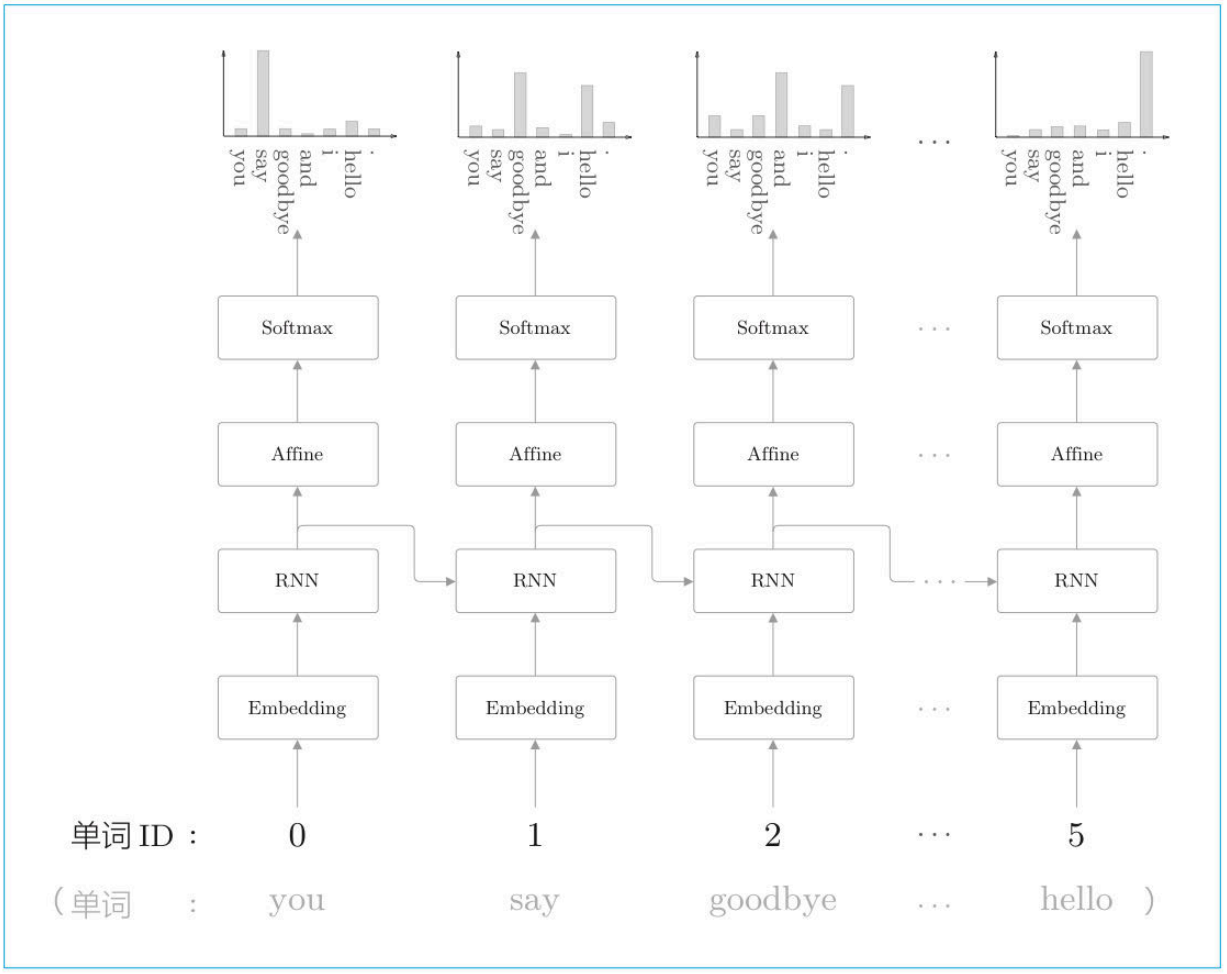

图5-26　处理样本语料库“you say goodbye and i say hello.”的RNNLM的例子

- 评价性能的指标：困惑度-正确单词预测概率的倒数

Gated RNN

- 门：可以学习到时序数据的长期依赖关系
- 普通RNN：梯度消失||梯度爆炸
- 梯度爆炸解决：梯度裁剪
- LSTM：

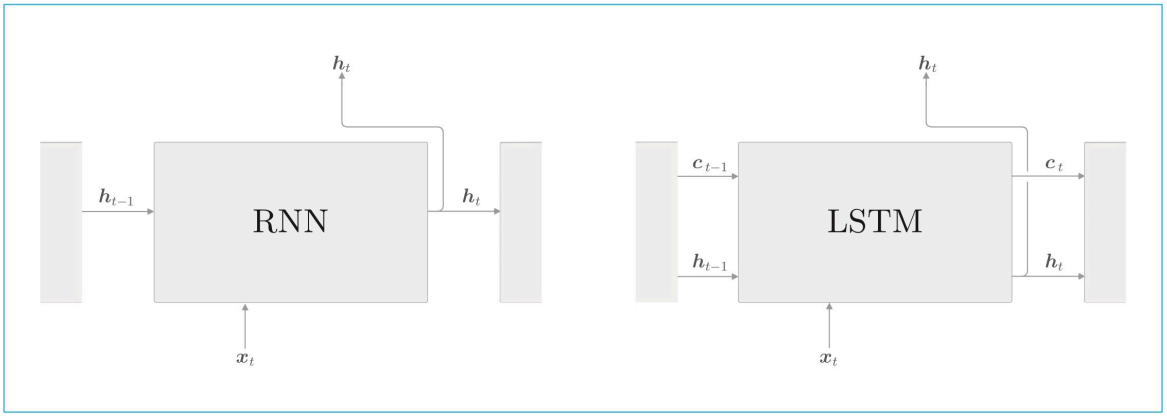

图6-11　RNN层与LSTM层的比较

  - 记忆单元在LSTM内部工作，不向其他层输出
  - 输出门：管理下一个隐藏状态h_t的输出
    - 开合程度由输入x_t和上一个状态h_{t-1}求得
  - 遗忘门：计算公式同输出门
    - 记忆单元：c_t=f⊙c_{t-1}
  - 输入门：计算公式同输出门，判断新增信息价值
  - 不会发生梯度消失：反向传播进行的是对应元素乘积计算，遗忘门认为要忘记记忆单元元素的梯度变小，遗忘门觉得不能忘记的梯度不会退化
  - 改进
    - 多层LSTM
    - LSTM容易过拟合-正则化
    - 权重共享

基于RNN生成文本

- seq2seq模型，也叫Encorder-Decorder模型
  - 编码器对输入数据编码，解码器对被编码的数据解码
- 编码就是将任意长度的文本转化为一个固定长度的向量

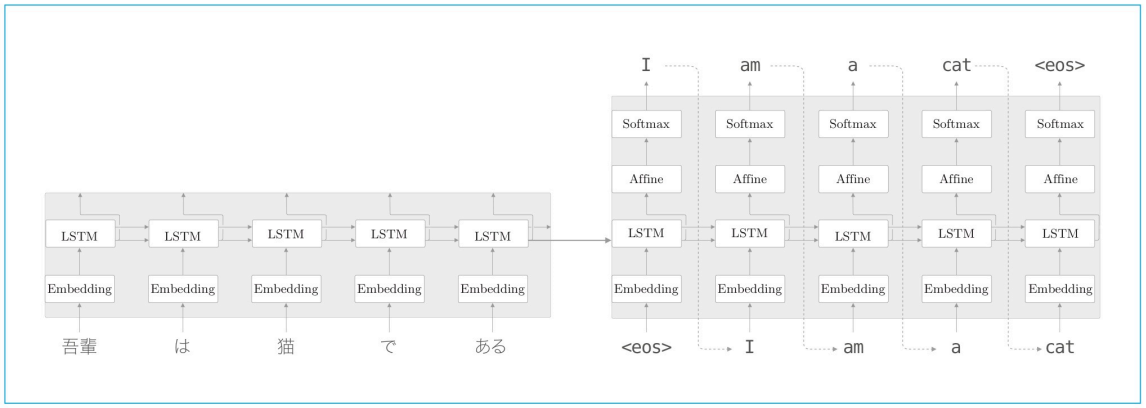

图7-9　seq2seq的整体的层结构

- 如上，解码器LSTM层隐藏状态是编码器和解码器的桥梁
- 改进seq2seq：
  - 反转输入数据-Reverse 12+56->65+21
  - peeky:将编码好的信息分配给解码器其他层

Attention

- 强大而优美
- seq2seq：输出的是固定长度向量
  - 文本过长的话有用的信息会从向量中溢出
- 改进
  - 编码器：输出长度根据文本长度相应改变
  - 解码器：
    - 学习“输入和输出中的哪些单词和哪些单词有关”，仅关注必要信息，根据该信息进行时序转换

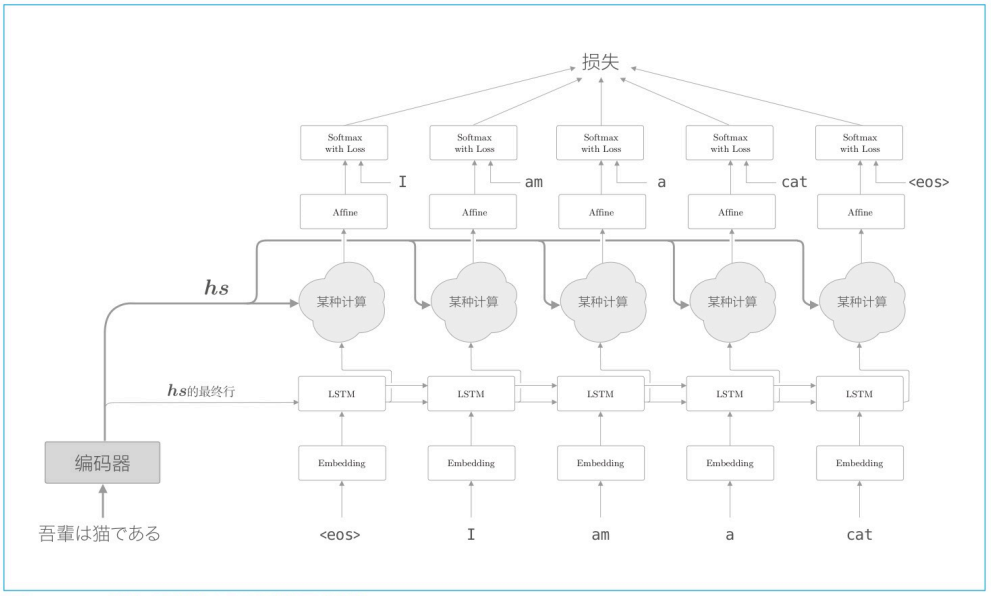

图 8-6　改进后的解码器的层结构

- 计算表示各个单词重要度的权重

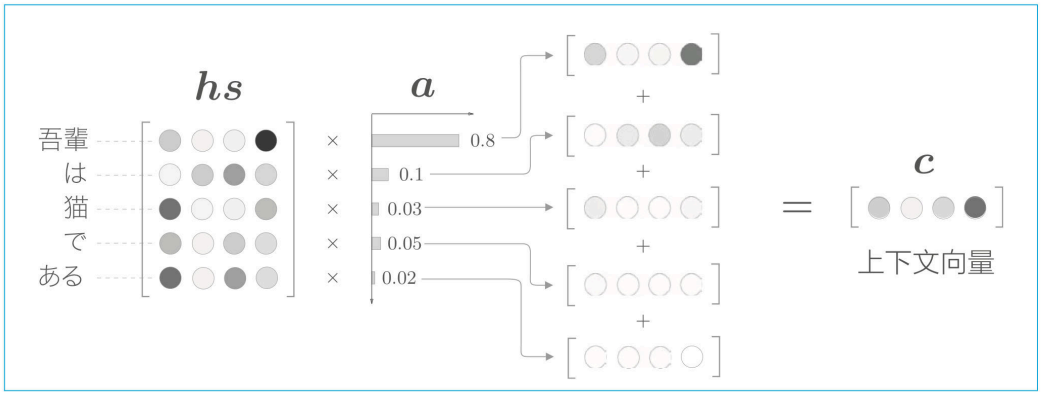

图 8-8　通过计算加权和，可以得到上下文向量

  - 求a：用内积算相似度
- 带Attention的seq2seq：LSTM和Affine之间加一个Attention层
- Attention的可视化
- 双向LSTM：一个 -> 一个 <- 然后拼接输出
- 残差链接
- transformer-self attention
- NTM