北京航空航天大學
BEIHANG UNIVERSITY

深度学习与自然语言处理（NLP）第三次课后作业

院（系）名称	自动化科学与电气工程学院
专业名称	自动化
学号	ZY2303808
学生姓名	牛晨然

2024 年 05 月

1 内容介绍

利用给定语料库，利用 1～2 种神经语言模型（如：基于 Word2Vec，LSTM， GloVe 等模型）来训练词向量，通过计算词向量之间的语意距离、某一类词语的聚类、某些段落直接的语意关联、或者其他方法来验证词向量的有效性。

2 实验原理

Word2vec 原理：

Word2vec 是一种用于生成词向量的模型，它能够将词语映射到一个连续的向量空间中，使得语义相近的词语在向量空间中的距离也相近。词向量是自然语言处理中的一种重要技术，它能够捕捉词语之间的语义和语法关系，为文本分析、情感分析、文本分类等任务提供有力支持。

Word2vec 模型的核心思想是通过词语的上下文信息来学习词语的向量表示。具体来说，Word2vec 模型通过训练一个神经网络模型，使得给定一个词语的上下文时，能够预测该词语本身（CBOW 模型），或者给定一个词语时，能够预测其上下文（Skip-gram 模型）。

Word2Vec 主要包括 CBOW 模型（连续词袋模型）和 Skip-gram 模型（跳字模型），CBOW 适合于数据集较小的情况，而 Skip-Gram 在大型语料中表现更好。其中 CBOW 如下图 1 左部分所示，使用围绕目标单词的其他单词（语境）作为输入，在映射层做加权处理后输出目标单词。与 CBOW 根据语境预测目标单词不同，Skip-gram 根据当前单词预测语境，如下图 1 右部分所示。假如我们有一个句子“There is an apple on the table”作为训练数据（训练集是一个句子，用上下单词来预测当前单词，与目标单词对比，从而训练网络），CBOW 的输入为（is,an,on,the），输出为 apple。而 Skip-gram 的输入为 apple，输出为（is,an,on,the）。本实验使用的模型为 CBOW 模型。

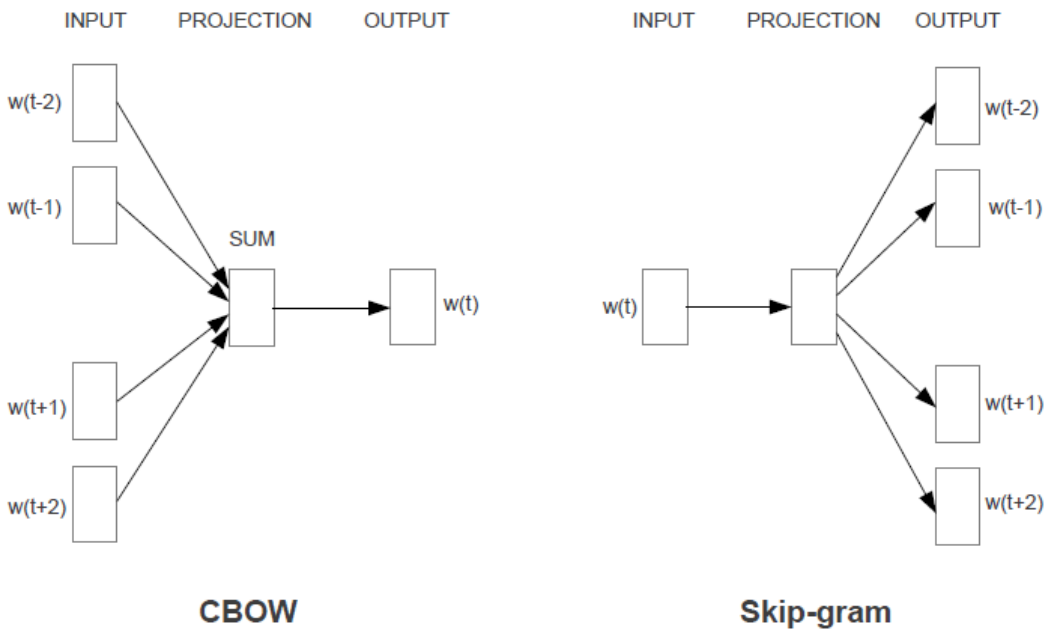


图 1 Word2Vec 模型

CBOW 模型：

其结构模型如下图 2 所示。

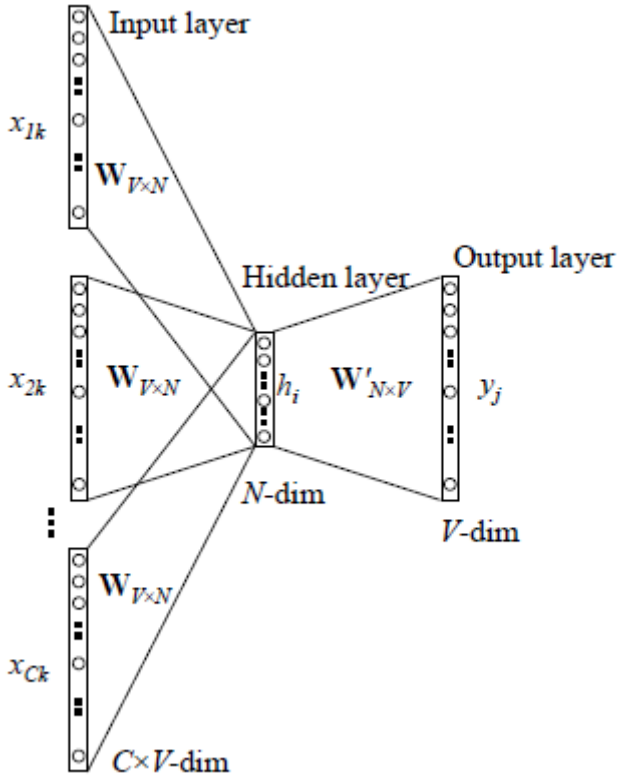


图 2 CBOW 模型

其步骤如下：

1、输入层：上下文单词的 One-Hot 编码词向量，V 为词汇表单词个数，N 为自定义的词向量的维度，C 为上下文单词个数。

2、初始化一个权重矩阵W_{V*N}，然后用所有输入的 One-Hot 编码词向量左乘该矩阵,得到维数为 N 的向量$\omega_1, \omega_2, \omega_3, \omega_c$，这里的 N 由自己根据任务需要设置。

3、将所得的向量$\omega_1, \omega_2, \omega_3, \omega_c$相加求平均作为隐藏层向量 h

4、初始化另一个权重矩阵W'_{V*N}，用隐藏层向量 h 左乘这个矩阵，再经激活函数处理得到 V 维的向量 y，y 的每一个元素代表相对应的每个单词的概率分布。

5、y 中概率最大的元素所指示的单词为预测出的中间词（target word）与 true label 的 One-Hot 编码词向量做比较，误差越小越好（根据误差更新两个权重矩阵）

在训练前需要定义好损失函数（一般为交叉熵代价函数），采用梯度下降算法更新 W 和 W'。训练完毕后，输入层的每个单词与矩阵 W 相乘得到的向量的就是我们想要的 Distributed Representation 表示的词向量，也叫做 word embedding。因为 One-Hot 编码词向量中只有一个元素为 1，其他都为 0，所以第 i 个词向量乘以矩阵 W 得到的就是矩阵的第 i 行，所以这个矩阵也叫做 look up table，有了 look up table 就可以免去训练过程，直接查表得到单词的词向量了。

Skip-gram 模型：

其结构模型如下图 3 所示。

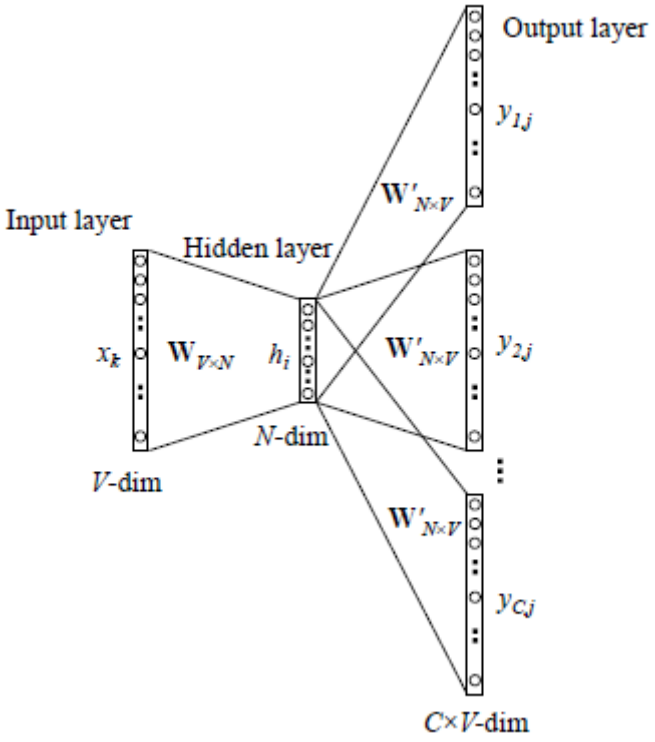


图 3 Skip-gram 模型

以句子“There is an apple on the table”为例，其步骤如下：

1、首先我们选句子中间的一个词作为我们的输入词，例如我们选取“apple”作为 input word；

2、有了 input word 以后，我们再定义一个叫做 skip_window 的参数，它代表着我们从当前 input word 的一侧（左边或右边）选取词的数量。如果我们设置 skip_window=2，那么我们最终获得窗口中的词（包括 input word 在内）就是[‘is’,’an’,’apple’,’on’,’the’]。skip_window=2 代表着选取左 input word 左侧 2 个词和右侧 2 个词进入我们的窗口，所以整个窗口大小 span=2x2=4。另一个参数叫 num_skips，它代表着我们从整个窗口中选取多少个不同的词作为我们的 output word，当 skip_window=2，num_skips=2 时，我们将会得到两组 (input word, output word) 形式的训练数据，即 ('apple', 'an')，('apple', 'one')。

3、神经网络基于这些训练数据中每对单词出现的次数习得统计结果，并输出一个概率分布，这个概率分布代表着到我们词典中每个词有多大可能性跟 input word 同时出现。举个例子，如果我们向神经网络模型中输入一个单词 “中

国”，那么最终模型的输出概率中，像“英国”， “俄罗斯”这种相关词的概率将远高于像“苹果”，“蝈蝈”非相关词的概率。因为“英国”，“俄罗斯”在文本中更大可能在“中国”的窗口中出现。我们将通过给神经网络输入文本中成对的单词来训练它完成上面所说的概率计算。

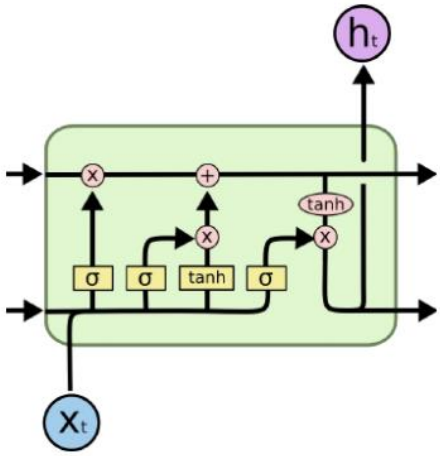
4、通过梯度下降和反向传播更新矩阵 W

5、W 中的行向量即为每个单词的 Word embedding 表示。

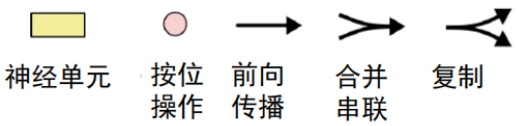
LSTM 原理：

LSTM（Long Short Term Memory）是一种特殊的递归神经网络（RNN），它能够有效地捕捉和保留长期依赖信息,克服了传统 RNN 在处理长序列数据时容易遗忘早期信息的缺点。LSTM 通过引入门控机制,能够在序列数据处理中更好地控制信息的流动和记忆的保存。

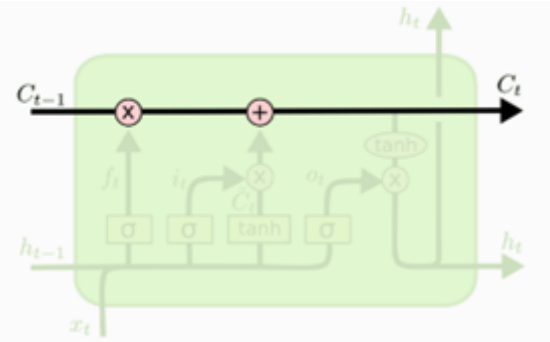
LSTM 的单元（cell）组成如下图所示



上图中使用的各个元素的图标的含义如下图所示:

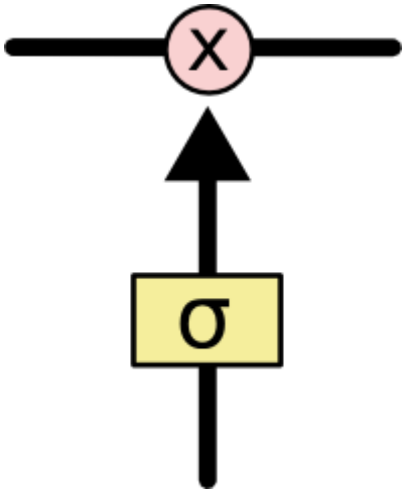


LSTM 单元的状态如下所示:



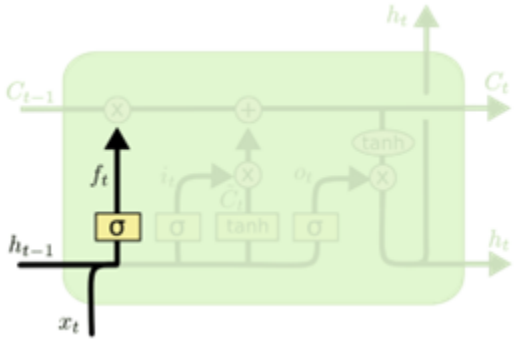
穿过图的顶部的长横线，长直线称之为节点状态（Cell State），决定什么样的信息会被保留，什么样的信息会被遗忘，记为C_t。

LSTM 网络能通过一种被称为门的结构对节点状态进行删除或者添加信息。门能够有选择性的决定让哪些信息通过。门的结构为一个 sigmoid 层和一个点乘操作的组合, sigmoid 层输出 0 到 1 之间的数, 描述每个部分有多少量可以通过, 0 代表不允许任何量通过，1 表示允许任何量通过，结构如下图所示:



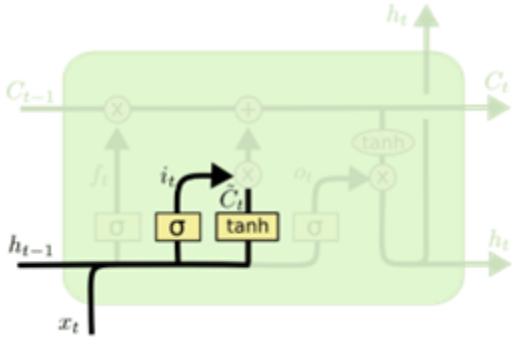
LSTM 实现了三个门计算，即遗忘门、输入门和输出门，用来保护和控制节点状态。

遗忘门负责决定保留多少上一时刻的单元状态到当前时刻的单元状态, 即决定从节点状态中丢弃什么信息。该门读取h_{t-1}和x_t，然后经过 sigmoid 层后，输出一个 0-1 之间的数f_t给每个在细胞状态C_{t-1}中的数字逐点相乘。f_t的值为 0 表示完全丢弃，1 表示完全保留。遗忘门结构如下图所示:



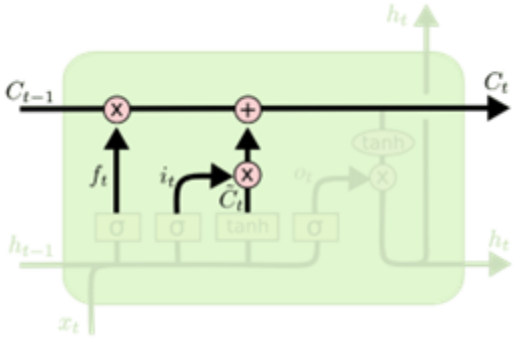
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

输入门负责决定保留多少当前时刻的输入到当前时刻的单元状态，包含两个部分，第一部分为 sigmoid 层，该层决定要更新什么值，第二部分为 tanh 层，该层把需要更新的信息更新到细胞状态里。tanh 层创建一个新的细胞状态值向量$\widetilde{C_t}$，$\widetilde{C_t}$会被加入到状态中。输入门结构如下图：



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

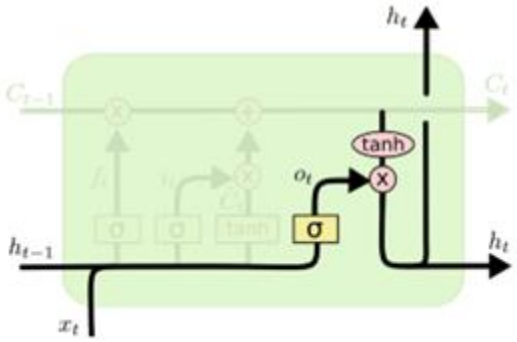
然后就到了更新旧细胞状态的时间了，将C_{t-1}更新为C_t，把旧状态与f_t相乘，丢弃确定需要丢弃的信息，再加上 $i_t * \widetilde{C_t}$，这样就完成了细胞状态的更新，结构如下图所示：



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

输出门负责决定当前时刻的单元状态有多少输出，通过一个 sigmoid 层来确定细胞状态的哪个部分将输出出去。把细胞状态通过 tanh 进行处理，得到一个-

1 到 1 之间的值，并将它和 sigmoid 门的输出相乘，最终仅仅输出确定输出的部分。输出门结构如下图所示：



$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

LSTM 的参数训练算法，依然是反向传播算法。主要有如下三个步骤：

第一步：前向计算每个神经元的输出值。对于 LSTM 而言，依据前面介绍的算法，分别进行计算。

第二步：确定优化目标函数。在训练早期，输出值和预期值会不一致，于是计算每个神经元的误差项值，构造出损失函数。

第三步：根据损失函数的梯度指引，更新网络权值参数。与传统 RNN 类似，LSTM 误差项的反向传播包括两个层面：一个是空间上层面的，将误差项向网络的上一层传播。另一个是时间层面上的，沿时间反向传播，即从当前 t 时刻开始，计算每个时刻的误差。

然后跳转第一步，重复做第一、二和三步，直至网络误差小于给定值。

3 实验结果与分析

选择《倚天屠龙记》、《天龙八部》、《射雕英雄传》、《神雕侠侣》、《笑傲江湖》作为样本，分别使用 Word2Vec、LSTM 对其中的主角进行了聚类分析，选取关联度最高的五个名词。训练结果如下所示：

表 1 倚天屠龙记

Word2Vec	LSTM(epochs=10)	LSTM(epochs=20)

张无忌	关联度	张无忌	关联度	张无忌	关联度
周芷若	0.46898	张翠山	0.8182477	张翠山	0.75191104
赵敏	0.42187	周芷若	0.7379257	谢逊	0.72421616
小昭	0.34814	谢逊	0.7039327	周芷若	0.6702966
蛛儿	0.34319	殷素素	0.68889856	殷素素	0.6513906
殷离	0.30412	易三娘	0.67564106	朱长龄	0.6478996

表 2 天龙八部

Word2Vec		LSTM(epochs=10)		LSTM(epochs=20)	
段誉	关联度	段誉	关联度	段誉	关联度
王语嫣	0.47879	虚竹	0.7721436	王语嫣	0.6066237
木婉清	0.41397	木婉清	0.74719924	虚竹	0.57555115
钟灵	0.34539	鸠摩智	0.7175795	鸠摩智	0.55559474
刀白凤	0.31817	萧峰	0.70776737	萧峰	0.5539005
阿紫	0.31132	慕容复	0.6985424	木婉清	0.5338779

表 3 射雕英雄传

Word2Vec		LSTM(epochs=10)		LSTM(epochs=20)	
郭靖	关联度	郭靖	关联度	郭靖	关联度
黄蓉	0.49151	黄蓉	0.6434272	周伯通	0.55350757
欧阳锋	0.35977	周伯通	0.6251872	彭连虎	0.54680884
洪七公	0.33123	黄药师	0.6189608	洪七公	0.52766824
欧阳克	0.31986	洪七公	0.5972002	欧阳克	0.5240533
傻姑	0.30538	穆念慈	0.5939337	完颜康	0.51947856

表 4 神雕侠侣

Word2Vec		LSTM(epochs=10)		LSTM(epochs=20)	
杨过	关联度	杨过	关联度	杨过	关联度
小龙女	0.58471	小龙女	0.797016	小龙女	0.68315786
杨过笑	0.378163	法王	0.718907	裘千尺	0.6382822
姑姑	0.37467	武修文	0.7166222	李莫愁	0.63818145
公孙谷主	0.36098	绿萼	0.7109548	法王	0.63697165

李莫愁	0.33371	周伯通	0.70717555	周伯通	0.63379896

表 5 笑傲江湖

Word2Vec		LSTM(epochs=10)		LSTM(epochs=20)	
令狐冲	关联度	令狐冲	关联度	令狐冲	关联度
岳不群	0.46246	林平之	0.77235067	盈盈	0.6996944
盈盈	0.41324	岳不群	0.7719485	岳不群	0.68788004
岳灵珊	0.37638	盈盈	0.74918467	林平之	0.6719217
田伯光	0.3304	岳灵珊	0.74723315	向问天	0.6308986
仪琳	0.3214	任行	0.7409293	陆大有	0.6206068

由图表可以看出，LSTM 迭代次数越多，得到的准确率越高，但是当 LSTM 迭代次数为 20 时，郭靖关联度最高的前五个人名中没有黄蓉，分析原因可能是因为模型可能开始过拟合训练数据。这意味着它捕捉到了更多特定的模式，这些模式可能不具有良好的泛化能力。因此，与“郭靖”经常共同出现的其他人物或词语可能变得更加突出，从而掩盖了“黄蓉”。

相较于 LSTM，Word2Vec 的训练速度更快，但是使用 LSTM 生成的干扰词更少，准确度更高，选择哪种方法需要根据具体任务的需求来决定。

4 参考文献

[1] https://blog.csdn.net/weixin_44966965/article/details/124732760
[2] https://www.cnblogs.com/lfri/p/15032919.html
[3] https://blog.csdn.net/lyc_yongcai/article/details/73201446