# NLP系列之词向量-Elmo（十三）

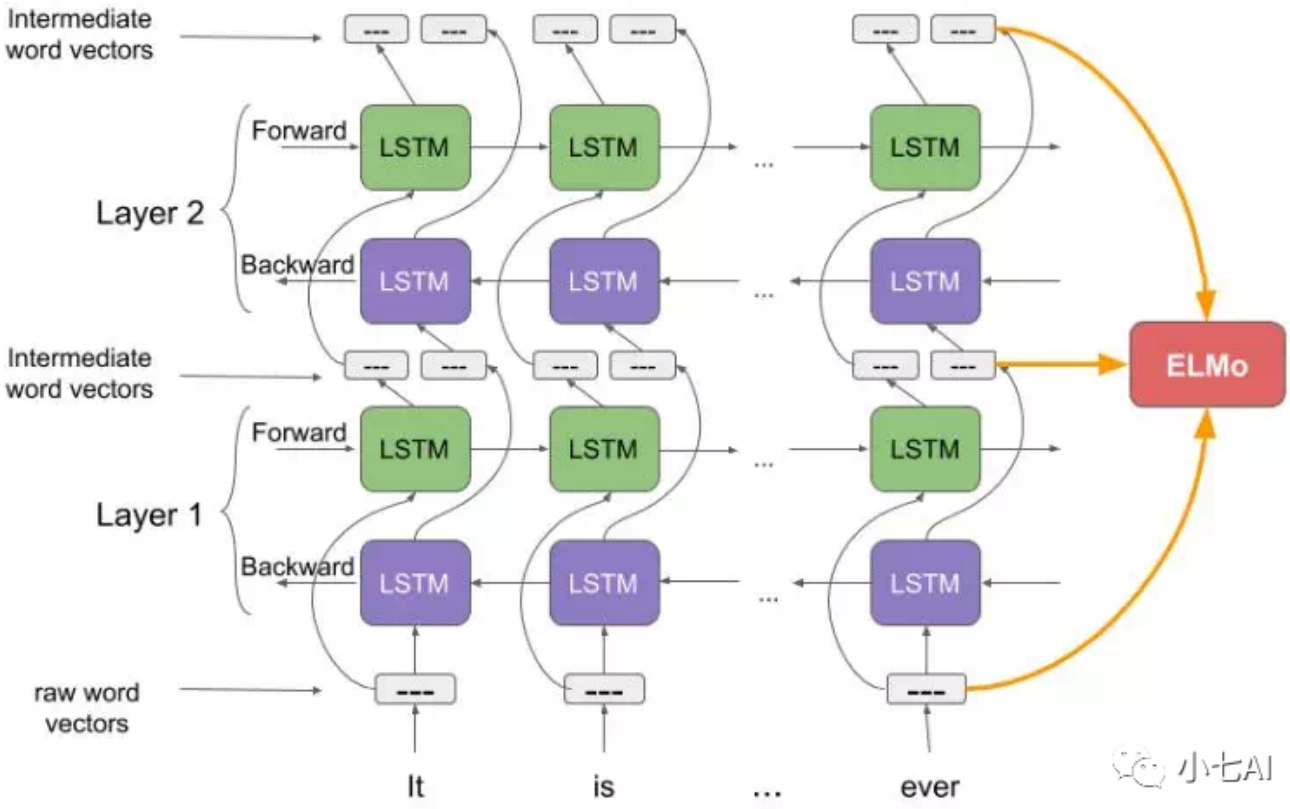
原创 小七AI 小七AI 2020-10-22

上一期我们了解了一种词向量模型——FastText，它将整篇文档的词及n-gram向量叠加平均得到文档向量，然后使用文档向量做softmax多分类，运行很快。

今天我们要讲的是另一种词向量模型——**Elmo（嵌入语言模型）**。它全称叫：Embeddings from Language Models，是AllenNLP研发的一种在词向量（vector）或词嵌入（embedding）中表示词汇的新方法。

**一、Elmo的工作原理**

Elmo的词向量是在双层双向语言模型（two-layer bidirectional language model，biLM）上计算的。这种模型由两层叠在一起，每层都有前向（forward pass）和后向（backward pass）两种迭代。



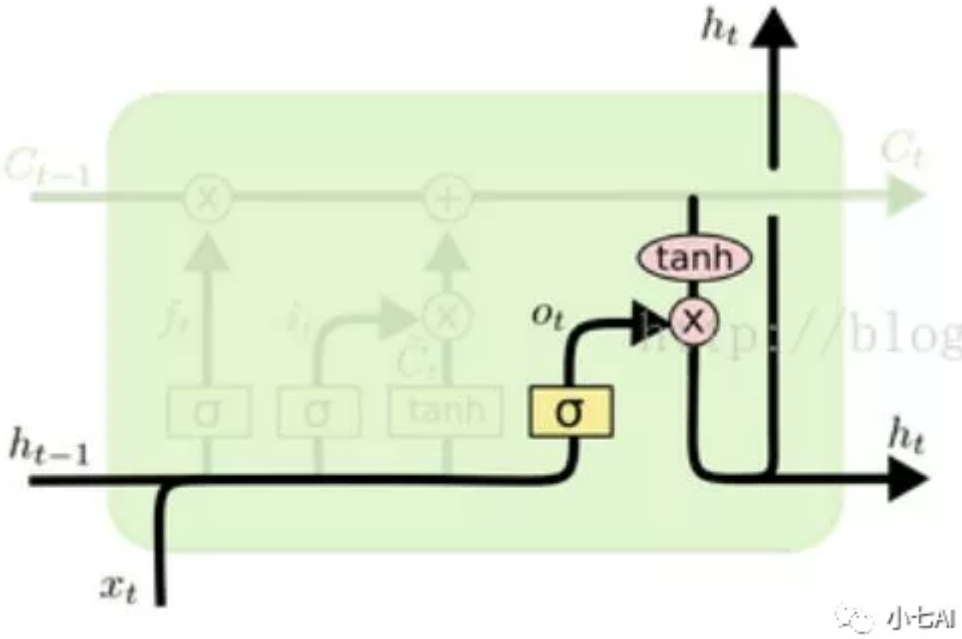

- 上图中的结构使用字符级卷积神经网络（convolutional neural network, CNN）来将文本中的词转换成原始词向量（raw word vector）
- 将这些原始词向量输入双向语言模型中第一层
- 前向迭代中包含了该词以及该词之前的一些词汇或语境的信息
- 后向迭代中包含了该词之后的信息
- 这两种迭代的信息组成了中间词向量（intermediate word vector）

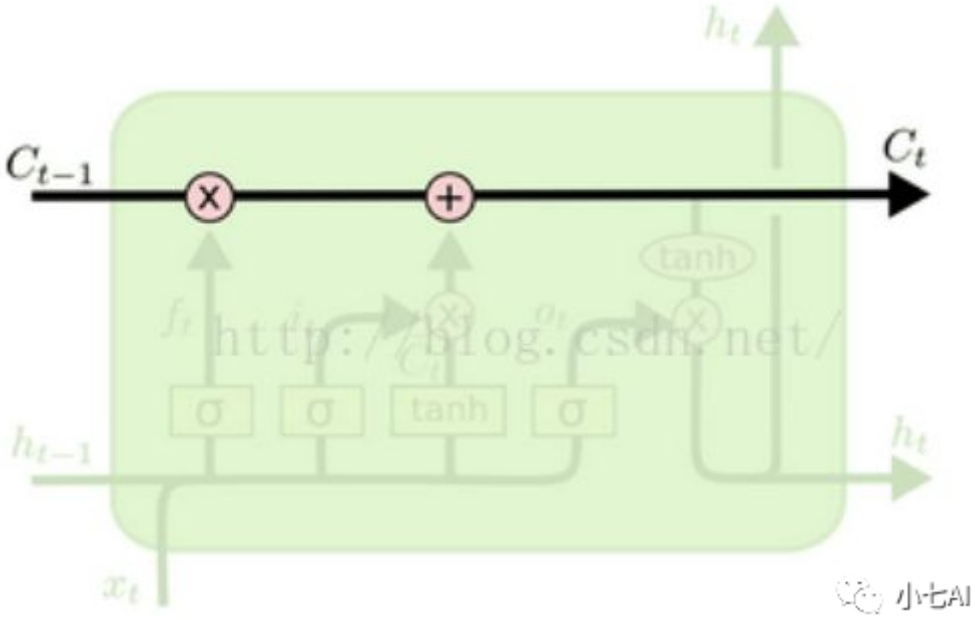
- 这些中间词向量被输入到模型的下一层
- 最终表示（ELMo）就是原始词向量和两个中间词向量的加权和.

因为双向语言模型的输入度量是字符而不是词汇，该模型能捕捉词的内部结构信息。比如beauty和beautiful，即使不了解这两个词的上下文，双向语言模型也能够识别出它们的一定程度上的相关性。

## 二、Elmo训练过程

首先明确LSTM 每一个时间点都会有一个输出，不要把这个输出和向下一时间点传送的记忆内容搞混，两个是有区别的，LSTM向下一个时间点传送的东西有两个，一个是该时间点的输出也就是隐状态H，同时传给了下一个时间点，另一个是记忆单元C，每个时间点的输出只有H（如图）。



- 理解了LSTM的输出后，再来看一层双向LSTM就会有两个输出，一个来自正向，一个来自负向，这两个一个代表前面信息对该词的影响的输出，一个代表后面信息对该词影响的输出，都能拿来当作词向量用，所以一层就会有两个词向量，两层就有4个，再加上最开始的输入，raw word vectors，就有4+1，也就有了2L+1个词向量！！！
- 再来理解一下RNN的提出，RNN的提出本身就是为了解决序列信息的问题（在NLP中），普通神经网络在操作时，后面的输入和前面的输入都是无关的，所以RNN本身就可以拿来做语言模型，我们也做过用LSTM/GRU或普通RNN来直接做语言模型的。
- 理解了RNN语言模型，那Elmo就很简单了，就是个多层的LSTM来用作训练语言模型。
- 具体公式如图，目标就是最小化损失函数，反向传播更新就完事了：

给定一个长度为N的句子，假设为$t_1, t_2, \ldots, t_N$，语言模型会计算给定$t_1, \ldots, t_{k-1}$的条件下出现$t_k$的概率：

$$p(t_1, \ldots, t_N) = \prod_{i=1}^{k} p(t_k | t_1, \ldots, t_{k-1})$$

传统的N-gram语言模型不能考虑很长的历史，因此现在的主流是使用多层双向的RNN(LSTM/GRU)来实现语言模型。在每个时刻k，RNN的第j层会输出一个隐状态$\overrightarrow{h}_{kj}^{LM}$，其中$j = 1, 2, \ldots, L$，L是RNN的层数。最上层是$\overrightarrow{h}_{kL}^{LM}$，对它进行softmax之后就可以预测输出词的概率。类似的，我们可以用一个反向的RNN来计算概率：

$$p(t_1, \ldots, t_N) = \prod_{i=1}^{k} p(t_k | t_{k+1}, \ldots, t_N)$$

通过这个RNN，我们可以得到$\overleftarrow{h}_{kj}^{LM}$。我们把这两个方向的RNN合并起来就得到Bi-LSTM。我们优化的损失函数是两个LSTM的交叉熵加起来是最小的：

$$Loss = \sum_{k=1}^{N} (logp(t_k | t_1, \ldots, t_{k-1}; \Theta_x, \overrightarrow{\Theta}_{LSTM}, \Theta_s) + logp(t_k | t_{k+1}, \ldots, t_N; \Theta_x, \overleftarrow{\Theta}_{LSTM}, \Theta_s)$$

这两个LSTM有各自的参数$\overrightarrow{\Theta}_{LSTM}$和$\overleftarrow{\Theta}_{LSTM}$，但是word embedding参数$\Theta_x$和softmax参数$\Theta_s$是共享的。

## 三、Elmo使用过程

这么多隐藏状态任君选择，爱怎么玩怎么玩，加权求和。

ELMo会根据不同的任务，把上面得到的双向的LSTM的不同层的隐状态组合起来。对于输入的词$t_k$，我们可以得到2L+1个向量，分别是$\{x_k^{LM}, \overrightarrow{h}_{kj}^{LM}, \overleftarrow{h}_{kj}^{LM}, j=1,2,\ldots,L\}$，我们把它记作$R_k=\{h_{kj}^{LM}, j=0,1,\ldots,L\}$。其中$h_{k0}^{LM}$是词的Embedding，它与上下文无关，而其它的$h_{kj}^{LM}=[\overrightarrow{h}_{kj}^{LM}; \overleftarrow{h}_{kj}^{LM}], j>0$是把双向的LSTM的输出拼接起来的，它们与上下文相关的。为了用于下游(downstream)的特定任务，我们会把不同层的隐状态组合起来，组合的参数是根据特定任务学习出来的，公式如下：

$$ELMo_k^{task}=E(R_k;\Theta_{task})=\gamma^{task}\sum_{j=0}^{L}s_j^{task}h_{kj}^{LM}$$

这里的$\gamma^{task}$是一个缩放因子，而$s_j^{task}$用于把不同层的输出加权组合出来。在实际的任务中，RNN的参数$h_{kj}^{LM}$都是固定的，可以调的参数只是$\gamma^{task}$和$s_j^{task}$。当然这里ELMo只是一个特征提取，实际任务会再加上一些其它的网络结构，那么那些参数也是一起调整的。



**四、Elmo训练时的输入**

输入是一个句子，先分成token，每一个token再通过里面的字符进行编码，比如说‘english’这个单词，e有一个编码，n有一个，。。。，h有一个，将他们拼接，再加上起始和结尾符，再加上为了统一单词长度的padding，最后CNN一操作，再一pooling，就每一个词得到一个词向量，具体参考（https://www.infoq.cn/article/B8-BMA1BUfuh5MxQ687T）

**五、Elmo的特点**

- 首先Elmo是一个被预训练好的多层双向LSTM语言模型，意思就是里面的参数已经经过大量的语料库调好了，不是预训练好的词向量
- 它的词向量是在真实下游任务中产生的，所以根据输入不同，任务不同，同一个词获取的词向量是不同的
- 可以看作是特征提取的过程，在实际任务中，对于输入的句子，使用Elmo这个语言模型处理他，得到输出的向量，拿来做词向量。

**六、Elmo与其他词嵌入的区别**

与word2vec或GLoVe等传统词嵌入不同，ELMo中每个词对应的向量实际上是一个包含该词的整个句子的函数。因此，同一个词在不同的上下文中会有不同的词向量，即**能解决多义词的问题。**

欢迎大家关注小七AI公众号：