# 6. ELMO

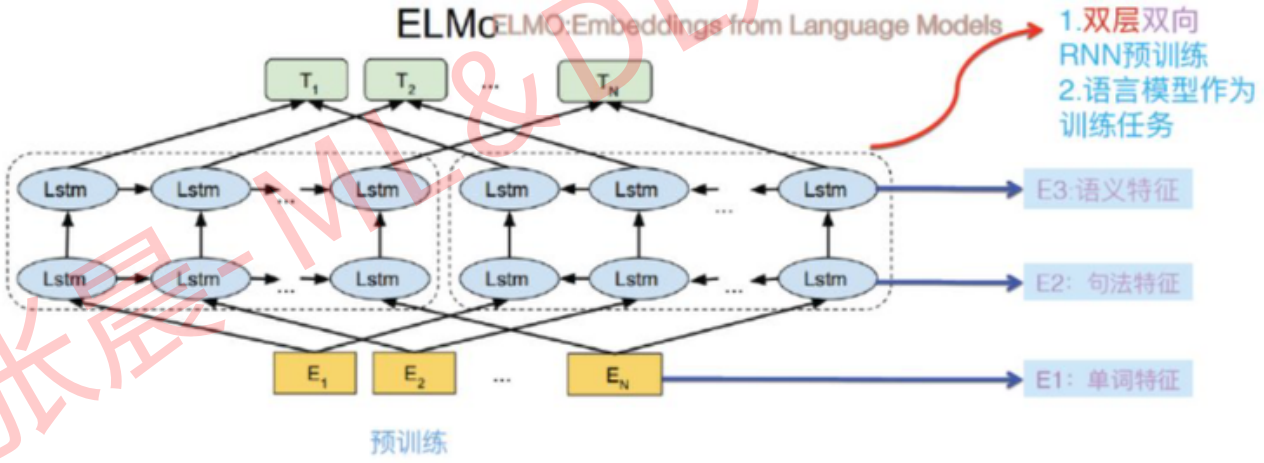
### 1．ELMo的基本原理是什么

ELMO采用了典型的两阶段过程，

「第一个阶段是利用语言模型进行预训练」；

「第二个阶段是在做下游任务时，从预训练网络中提取对应单词的网络各层的Word Embedding作为新特征补充到下游任务中。」

**第一阶段模型总览：**




上图展示的是其预训练过程，它的网络结构采用了双层双向LSTM，目前语言模型训练的任务目标是根据单词Wi的上下文去正确预测单词Wi，Wi之前的单词序列Context-before称为上文，之后的单词序列Context-after称为下文。使用这个网络结构利用大量语料做语言模型任务就能预先训练好这个网络，如果训练好这个网络后，输入一个新句子，句子中每个单词都能得到对应的三个Embedding：

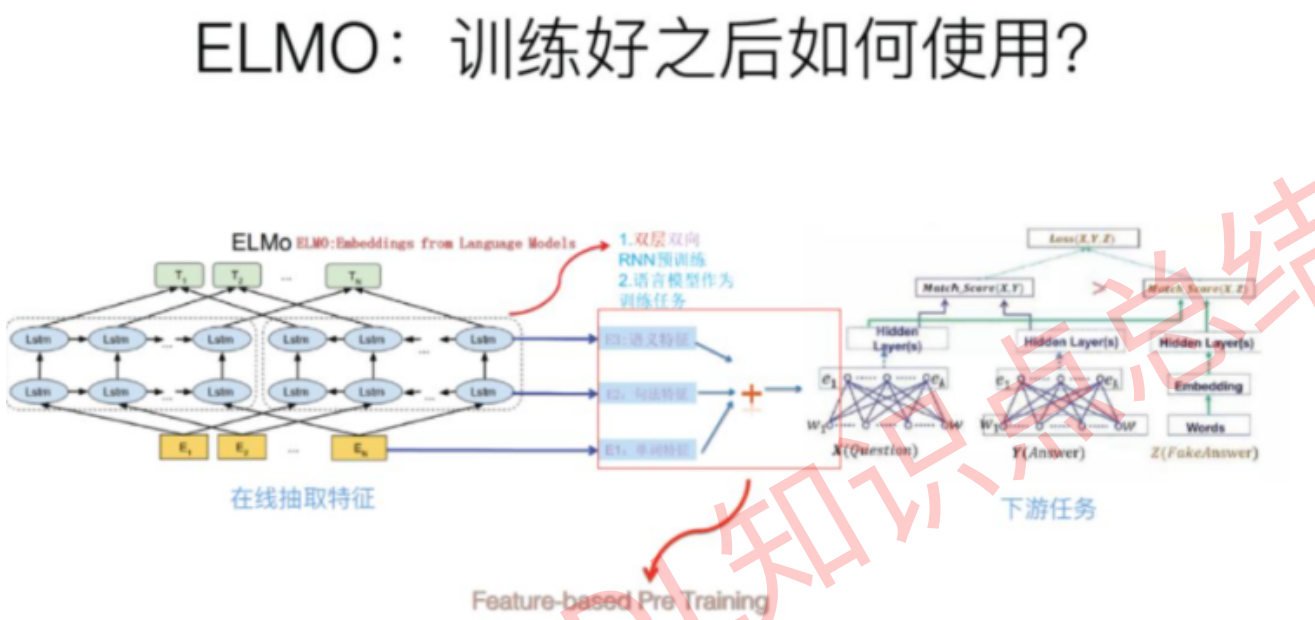
- 最底层是单词的Word Embedding
- 往上走是第一层双向LSTM中对应单词位置的Embedding，这层编码单词的

**句法信息**更多一些；

- 再往上走是第二层LSTM中对应单词位置的Embedding，这层编码单词的**语义信息**更多一些。

也就是说，**ELMO的预训练过程不仅仅学会单词的Word Embedding，还学会了一个双层双向的LSTM网络结构**，而这两者后面都有用。

**第二阶段模型总览：**




以QA问题为例，展示下游任务如何利用预训练好的embedding。

- 对于问句X，我们可以先将句子X作为预训练好的ELMO网络的输入，这样句子X中每个单词在ELMO网络中都能获得对应的三个Embedding；
- 之后给予这三个Embedding中的每一个Embedding一个权重a，这个权重可以学习得来，根据各自权重累加求和，将三个Embedding整合成一个；
- 然后将整合后的这个Embedding作为X句在自己任务的那个网络结构中对应单词的输入，以此作为补充的新特征给下游任务使用。

对于上图所示下游任务QA中的回答句子Y来说也是如此处理。**因为ELMO给下游提供的是每个单词的特征形式，所以这一类预训练的方法被称为"Feature-based Pre-Training"。**

## 2．ELMo的训练过程是什么样的？损失函数是什么？

ELMo的训练过程实际上指的是其第一阶段的预训练过程，第一阶段实际上是训练一个双向语言模型，假设给定一个序列，该序列含有 $N$ 个token $(t_1, t_2, \ldots, t_N)$，那么：

- 前向语言模型通过在给定上文 $(t_1, \ldots, t_{k-1})$(Context-before)的情况下对token $t_k$ 的概率建模来计算序列出现的概率：

$$p(t_1, t_2, \ldots, t_N) = \prod_{k=1}^{N} p(t_k | t_1, t_2, \ldots, t_{k-1})$$

- 后向语言模型与前向类似，但是它是“从后往前建模的”，通过在给定下文 $(t_{k+1}, \ldots, t_N)$(Context-after)的情况下对token $t_k$ 的概率建模来计算序列出现的概率：

$$p(t_1, t_2, \ldots, t_N) = \prod_{k=1}^{N} p(t_k | t_{k+1}, t_{k+2}, \ldots, t_N)$$

因此，由于ELMo结合了前后向语言模型，故其目标是同时最大化前后向语言模型的对数似然：

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

其中：

- $\Theta_x$为token表示的参数(前后向语言模型共享)
- $\Theta_s$为softmax分类的参数(前后向语言模型共享)
- $\overrightarrow{\Theta}_{LSTM}, \overleftarrow{\Theta}_{LSTM}$分别表示前后向语言模型LSTM层的参数

综上所述，ELMo的训练过程即为一个前后向语言模型的训练过程，通过上述描述则一目了然，而其损失函数即为简单的分类损失，取决于源码实现，不同源码中的实现可能略有不同。

### 3．ELMo训练好了之后如何使用？

ELMo的微调从严格意义上来说，不是真正的微调，预训练网络结果是fix的。 整体来说，是把句子输入到预训练网络的embedding，与下游任务word embedding做concat，concat的结果整体作为下游NLP任务的输入。

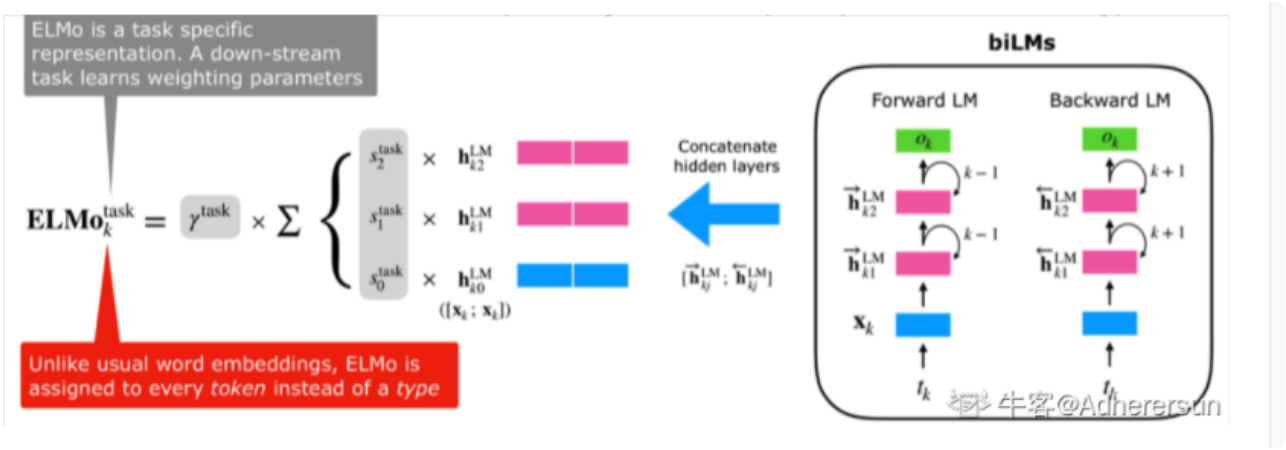
对于序列中的每个token，一个L层的双向语言模型就会得到其**2L+1个表示**，即为：

$$R_k = \left\{ \mathbf{x}_k^{LM}, \overrightarrow{\mathbf{h}}_{k,j}^{LM}, \overleftarrow{\mathbf{h}}_{k,j}^{LM} | j = 1, \ldots, L \right\} = \left\{ \mathbf{h}_{k,j}^{LM} | j = 0, \ldots, L \right\}$$

其中，$\mathbf{h}_{k,0}^{LM}$为token的表示(即$\mathbf{h}_{k,0}^{LM} = \mathbf{x}_k^{LM}$)，$\mathbf{h}_{k,j}^{LM} = [\overrightarrow{\mathbf{h}}_{k,j}^{LM}; \overleftarrow{\mathbf{h}}_{k,j}^{LM}]$为每个双向LSTM层得到的表示。

$$\mathbf{ELMo}_k^{task} = E\left(R_k; \Theta^{task}\right) = \gamma^{task} \sum_{j=0}^{L} s_j^{task} \mathbf{h}_{k,j}^{LM}, \quad s_j^{task} = e^{sj} / \sum_i^N e^{si}$$

其中，$s^{task}$是经过softmax归一化之后的权重，标量参数$\gamma^{task}$允许任务模型缩放整个ELMo向量。需要注意的是，$\gamma^{task}$是一个超参数，实际上这个参数是经验参数，一定程度上能够增强模型的灵活性。总结起来，整个为下游任务获取embedding的过程即为：




**4．ELMo的优点是什么？ELMo为什么有效?**

ELMo利用了深度上下文单词表征，该模型的优点：

- 引入双向语言模型，其实是 2 个单向语言模型（前向和后向）的集成；
- 通过保存预训练好的 2 层 biLSTM，通过特征集成或 finetune 应用于下游任务；

总结来说，通过上述结构，ELMo能够达到区分多义词的效果，每个单词(token)不再是只有一个上下文无关的embedding表示。

那么ELMo为什么有效呢？我认为主要原因有以下几点：

- 首先，ELMo的假设前提是一个词的词向量不应该是固定的，所以在多义词区分方面ELMo的效果必然比word2vec要好。
- 另外，ELMo通过语言模型生成的词向量是通过特定上下文的“传递”而来，再根据下游任务，对原本上下文无关的词向量以及上下文相关的词向量表示引入一个权重，这样既在原来的词向量中引入了上下文的信息，又能根据下游任务适时调整各部分的权重(权重是在网络中学习得来的)，因此这也是ELMo有效的一个原因。

**5．ELMo为什么能够达到区分多义词的效果?**

在ELMo第一阶段训练完成之后，将句子输入模型中在线提取各层embedding的时候，每个单词(token)对应两边LSTM网络的对应节点，那两个节点得到的embedding是动态改变的，会受到上下文单词的影响，周围单词的上下文不同应该会强化某种语义，弱化其它语义，这样就达到区分多义词的效果了。需要注意的是，第一个单词和最后一个单词也是有上下文的，譬如说第一个单词的上文是一个特殊的token <BOS>，下文是除第一个单词外的所有单词，最后一个单词的下文是一个特殊的token <EOS>，上文是除最后一个单词

外的所有单词。（总之，ELMO不仅训练好了词向量，也会得到一个模型，用于之后的NLP任务）

**6．ELMo把三种不同的向量叠加的意义是什么?**

- 之前很多方法都只用了最顶层LSTM的hidden state，但是通过实验验证，在很多任务中，将每一层hidden state融合在一起会取得更好的效果；
- 在上述实验中得到结论，每一层LSTM得到单词的embedding所蕴含的信息是不一样的，因此将所有信息融合起来，会让单词embedding的表达更丰富。

**7．elmo的网络结构是双向双层的lstm,如何实现双向的lstm的呢?**

与Bert在预训练目标中使用masked language model来实现双向不同（同时利用上下文信息），ELMo的双向概念实际是在网络结构中体现的（分开训练，最后再集成）。输入的embedding通过lstm的hidden state作为正向输出，embedding做reverse后的结果再通过lstm的hidden state反向输出，正向输出与反向输出做concat。最后输出实际是个lauguage model，基于前面的词计算下一个词的概率。

**8．elmo适合哪些下游NLP 任务?**

ELMo在短本任务上表现好。ELMo迁移到下游网络中， 一个是答案较短的数据集，提升有3-4个点， 一个答案较长的数据集，提升只有0.5 左右。在实验中，我们对比过词法分析和阅读理解任务，其在词法分析效果好于阅读理解。

**9．elmo的优缺点**

**优点:**

- ELMo 可以根据上下文语境来推断每个词对应的表示，比较好的解决一词多义问题。
- ELMo 建立语言模型的时候，可以运用类似的超大语料库去学习，学习好后，再应用到具体任务中去。

**缺点:**

- 在特征抽取器选择方面，ELMo 使用了 LSTM 而不是 Transformer，Transformer 提取特征的能力强于 LSTM。
- ELMo 采取双向拼接融合特征的能力可能比 Bert 一体化的融合特征方式弱。

- 实现的双向是两个单向的集成

- [ ] https://blog.csdn.net/abcdefg90876/article/details/104624401/
  <https://blog.csdn.net/abcdefg90876/article/details/104624401/>
- [ ] https://juejin.cn/post/6844903998361698311
  <https://juejin.cn/post/6844903998361698311>