初识预训练模型： elmo

Ⅰ. 什么是ELMO

ELMo(Embeddingsfrom Language Models) 是语义表示模型之一。以LSTM为基本单元，在**预训练阶段**通过最大化前后向**语言模型**的对数似然，得到通用的语义表示；在**下游任务**中，将通用的语义表示作为Feature使用。

Ⅱ. 为什么需要ELMO

NLP的作用，就是帮助计算机理解并处理自然语言。但计算机并不能直接处理自然语言，这就需要我们把自然语言转为**计算机可以“理解”的表示**；如第一章节第一课所讲，词典ID映射、one-hot、word2vec，包括这节的ELMO等都是这样的作用。

有同学会问了，为什么有了word2vec还需要ELMO呢？我们思考一个问题，word2vec可以解决**一词多义的问题吗**？答案是NO，因为同一个词经过训练后的word2vec得到的词向量是固定的，不管它的上下文是什么。而NLP的一个核心问题就是**如何学习不同语境下的语义表示，**所以ELMO就应运而生了。

```
#包袱
脱口秀大赛里处处都是包袱，太逗了
她手里抱着一个小红包袱，步履轻易的朝我走来

#Apple
One Apple A Day Keep Doctor Away.
Mr. Lei said Xiaomi went to the same companies that made the metallic-framed Apple iPhone to see what they could do for him.
```

Ⅲ. ELMO的原理

Ⅲ.Ⅰ LSTM

详见前置课程

Ⅲ.Ⅱ 预训练阶段

预训练是为了从大量的无监督数据中获取通用的语义信息，为下游任务(模型)提供了一个效果更好的初始化参数并加速收敛，以避免在小数据集上过拟合。

和大家熟知的bert一样，elmo也属于预训练模型家族的一员，只不过根据使用方式一般把elmo归为“基于特征的Pre-Training”，而把bert归为“基于Fine-tuning的Pre-Training”。

图2展示了预训练过程，elmo采用双层双向LSTM，其中E1采用的是token embedding或者char-base的卷积，在Ⅳ小节中我们会同时实现这两种表示并进行融合；图中左侧的前向双层

LSTM代表正方向编码器，输入的是从左到右顺序的除了预测单词W外的上文Context-before，图中右端的逆向双层LSTM代表反方向编码器，输入的是从右到左的逆序的句子下文Context-after；其中层与层之间还有残差进行连接。

使用图2的网络结构并利用大量语料做无监督语言模型任务就可以预训练好模型并得到相应权重：

- E1：单词embedding
- E2：第一层LSTM中对应单词的embedding，粒度比E1粗，句法信息
- E3：第二层LSTM中对应单词的embedding，粒度比E2粗，语义信息

是不是感觉和cv中不同层之间的信息表示有异曲同工之妙呢?

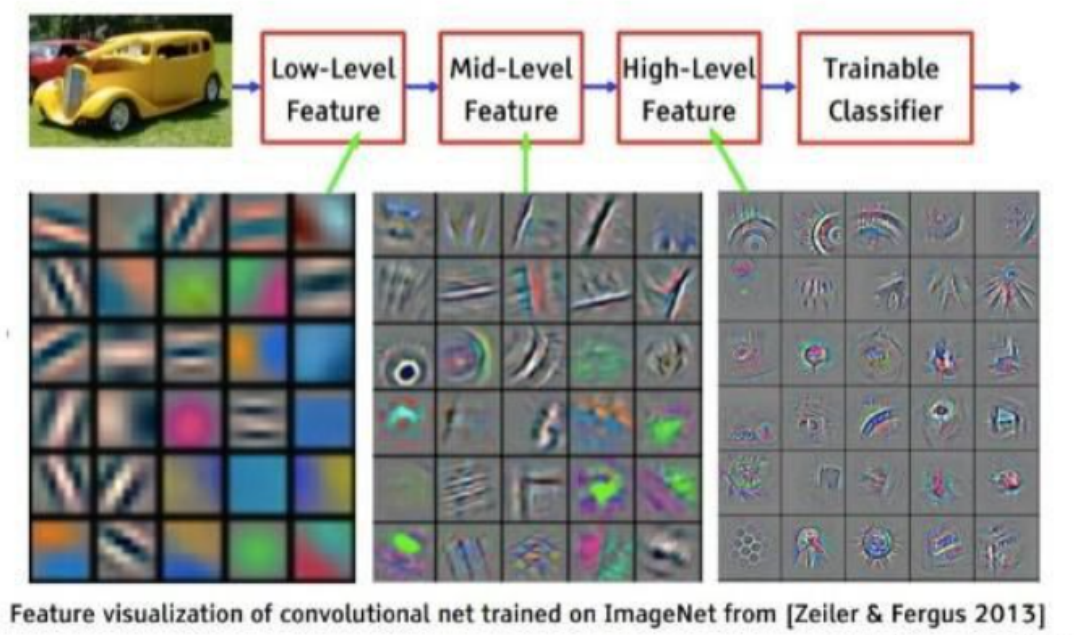

图1. cv中的层次表示

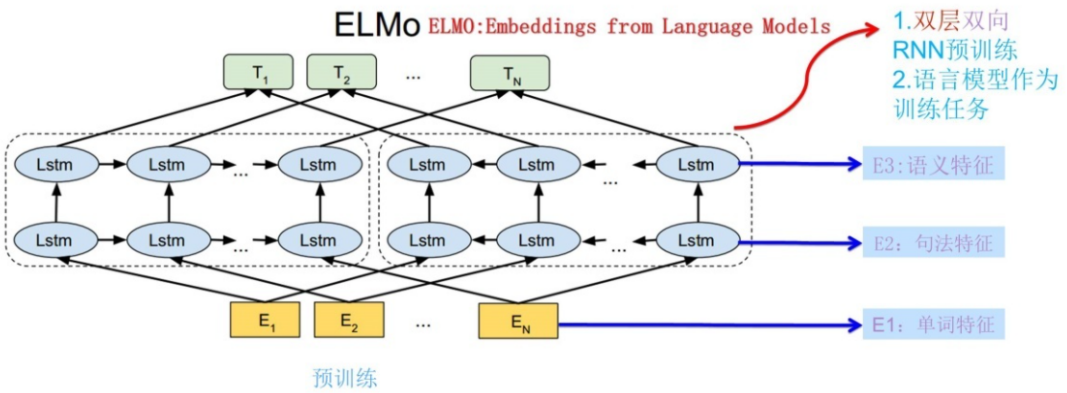

图2. 预训练阶段[1]

III.III 语言模型

通俗来讲，语言模型就是用来判断一句话从语法上是否通顺，核心思想就是根据句子中单词W前面的一系列单词预测后面跟哪个单词的概率最大。

那具体是怎么做的呢？为了弄明白具体过程，我们先用n-gram语言模型举例说明，然后再推广到NNLM，先简单回顾以下几个概念：

- 马尔科夫假设：简化条件概率
- 贝叶斯公式、链式法则：联合概率 -> 条件概率

贝叶斯公式：

$$\begin{aligned} P(A \cdot B) &= P(A|B) \cdot P(B) \\ &= P(B|A) \cdot P(A) \end{aligned}$$

Chain Rule：

$$\begin{aligned} P(A,B,C,D) &= P(A) \cdot P(B|A) \cdot P(C|A,B) \cdot P(D|A,B,C) \\ &= P(A,B) \cdot P(C|A,B) \cdot P(D|A,B,C) \\ &= P(A,B,C) \cdot P(D|A,B,C) \\ &= P(A,B,C,D) \end{aligned}$$

$$P(w_1, w_2, w_3, w_4, w_5, \ldots, w_n) = P(w_1) \cdot P(w_2|w_1) \cdot P(w_3|w_1, w_2) \ldots P(w_n|w_1, w_2, \ldots, w_{n-1})$$

一阶马尔科夫假设：

$$P(\text{休息} \mid \text{今天}, \text{是}, \text{春节}, \text{我们}, \text{都}) \approx P(\text{休息} \mid \text{都})$$

二阶马尔科夫假设：

$$P(\text{休息} \mid \text{今天}, \text{是}, \text{春节}, \text{我们}, \text{都}) \approx P(\text{休息} \mid \text{我们}, \text{都})$$

三阶马尔科夫假设：

$$P(\text{休息} \mid \text{今天}, \text{是}, \text{春节}, \text{我们}, \text{都}) \approx P(\text{休息} \mid \text{春节}, \text{我们}, \text{都})$$

现在假设我们知道下列单词的概率（从语料中统计），那怎么比较句子A和句子B哪个更通顺呢?

```
P(是|今天) = 0.01
P(今天) = 0.002
P(周日|是) = 0.001
P(周日|今天) = 0.0001
P(周日) = 0.02
P(是|周日) = 0.0002

句子A：今天是周日
句子B：今天周日是
```

根据上面的贝叶斯公式、链式法则及马尔科夫假设我们可知P(A) > P(B)：

$$P(\text{今天是周日}) = P(\text{今天}) \cdot P(\text{是} \mid \text{今天}) \cdot P(\text{周日} \mid \text{是}) = 0.002 \cdot 0.01 \cdot 0.001 = 2 * 10^{-8}$$
$$P(\text{今天周日是}) = P(\text{今天}) \cdot P(\text{周日} \mid \text{今天}) \cdot P(\text{是} \mid \text{周日}) = 0.002 \cdot 0.0001 \cdot 0.0002 = 4 * 10^{-10}$$

其实NNLM和上面的N-gram语言模型类似，只不过理论上当前词可以看到之前的所有词，而N-gram只能看到有限个。

III.IV 下游任务使用

经过预训练之后，对于输入L层双向语言模型的句子来说，其中的每个token都可以得到2L+1个表示，如图3所示。

我们以L=2举例子；当j为0时，代表的就是III.II小节中的E1，j为1时即为E2，以此类推及j为2时代表的就是E3。

$$\begin{aligned} R_k &= \{\mathbf{x}_k^{LM}, \overrightarrow{\mathbf{h}}_{k,j}^{LM}, \overleftarrow{\mathbf{h}}_{k,j}^{LM} \mid j = 1, \ldots, L\} \\ &= \{\mathbf{h}_{k,j}^{LM} \mid j = 0, \ldots, L\}, \end{aligned}$$

图3. token表示

下游任务会将这些表示都利用起来，并分配不同的权重，如图4所示。其中S-task是经过softmask归一化后的权重；γ-task是超参数，可以增强模型的灵活性。看到这里应该不难想明白为什么ELMO可以解决一词多义的问题了，因为对于同词不同句来说，词语表示中的E2和E3会受到不同上下文的影响，从而达到了区分的目的。

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

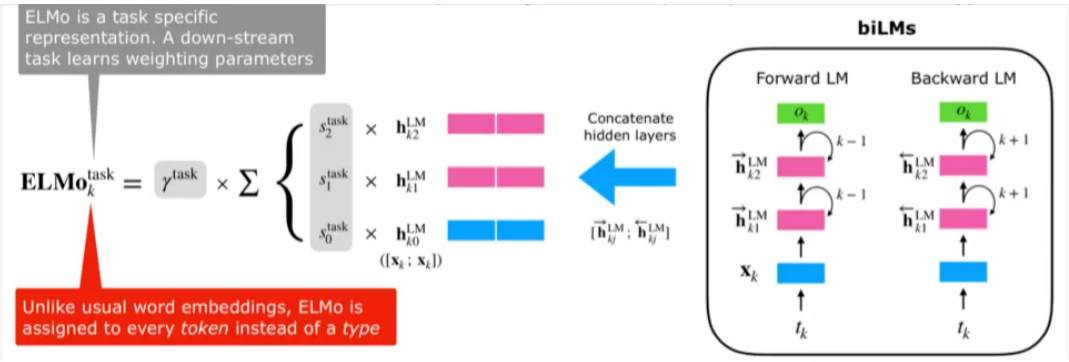

图4. 下游任务获取feature

IV. 实现

本小节展示从语料处理直到模型训练的过程。

ELMo.ipynb
20.1KB
预览

IV. 引用

[1] https://zhuanlan.zhihu.com/p/49271699
[2] https://mp.weixin.qq.com/s/i7EJSNzDsNNbK2YA_YNu8g
[3] https://zhuanlan.zhihu.com/p/82602015