基于双向 LSTM 的 sqe2seq 中文分词

——*NLP* 大作业报告

成员：汪子龙 1500011371 物理学院

刘浚哲 1500011370 物理学院

日期：2019 年 1 月 5 日

目 录

第 1 题

问题描述

我们利用双向 LSTM 模型完成了中文自动切词任务, 即把连续的中文文本切分成词序列. 在测试数据得到测试 Precision,Recall,F-score. 数据来自 Sighan-2004 中文切词国际比赛的标准数据. 训练数据为 86924 个句子, 测试数据为 3985 句子.

分词其实也是一种序列标注的问题, 即给句子中每一个字都打上相应的标注. 本文采用的是 4tag 标注法, 即 b,m,e,s, 分别代表: 开头, 中间, 结尾与单字. 给句中每一个字进行标注也相应地得到了分词的结果.

在网络模型的选择上, 我们使用的是双向 LSTM 模型, 相对于传统的 RNN 来说, LSTM 模型能够更好地学习序列中更长距离上的信息, 而双向 LSTM 模型则是从左到右和从右到左两个 LSTM 独立得到的结果进行叠加, 因此能够综合上下文的信息, 从而产生更好的效果.

我们利用训练好的 LSTM 模型进行序列上的预测: 对于一个给定的序列, 我们需要预测其最大概率的标注序列, 此时我们采用的就是生成模型, 而具体算法则是 Viterbi 算法. 通过维特比算法我们能够得到一个全局最优的解, 从而预测对应序列的最大概率标注.

第 2 题

模型与方法

§2.1 分词的字标注法

通过字标注法将分词问题变成序列标注问题, 对于输入序列, 输出大小相同、一一对应的标注序列.

能够实现分词的最简单标注方法是 2tag, 即通过 0、1 判断切分或不切分, 但通常情况下 2tag 表现不好, 因为两个标记很难体现语义规律. 在这里我们选择了更常用的 4tag 标记, 分为 b:begin, 多字词的开头; m: middle, 多字词的中间; e: end, 多字词的结尾; s: single, 单字词. 除此之外还使用了字符'x' 标记字符长度小于 max_length 时 padding 的部分. 4tag 法对一句话的标记示例如下:

```
他 “ 严格要求 自己 , 从 一个 科举 出身 的 进士 成为 一个 伟大 的 民主主义 者 ,
进而 成为 一 位 杰出 的 党外 共产主义 战士 , 献身 于 崇高 的 共产主义 事业 。

s bmme be s be be be s be be be be s bmme s be be s s be s be bmme be be s be s bmme be
```

§2.2 网络模型: LSTM

Long Short Term 网络——一般就叫做 LSTM ——是一种 RNN 特殊的类型，可以学习长期依赖信息. 所有 RNN 都具有一种重复神经网络模块的链式的形式。在标准的 RNN 中，这个重复的模块只有一个非常简单的结构，例如一个 tanh 层:

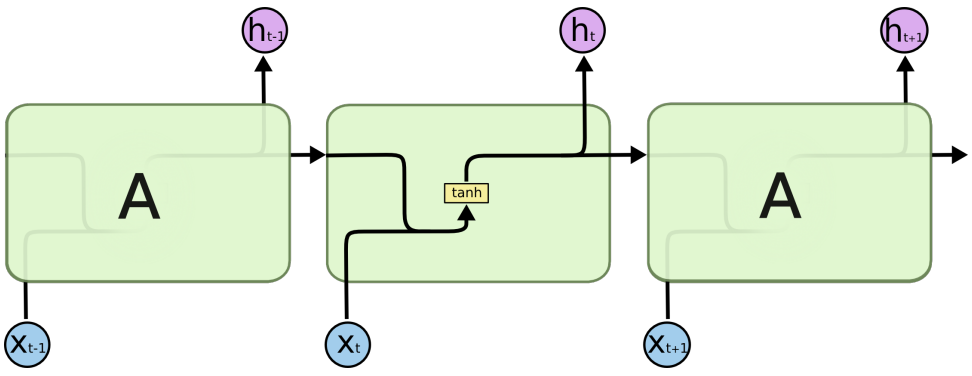


图 2.2-1 标准 RNN 中的重复模块包含单一的层

LSTM 同样是这样的结构，但是重复的模块拥有一个不同的结构。不同于单一神经网络层，这里是有四个，以一种非常特殊的方式进行交互。

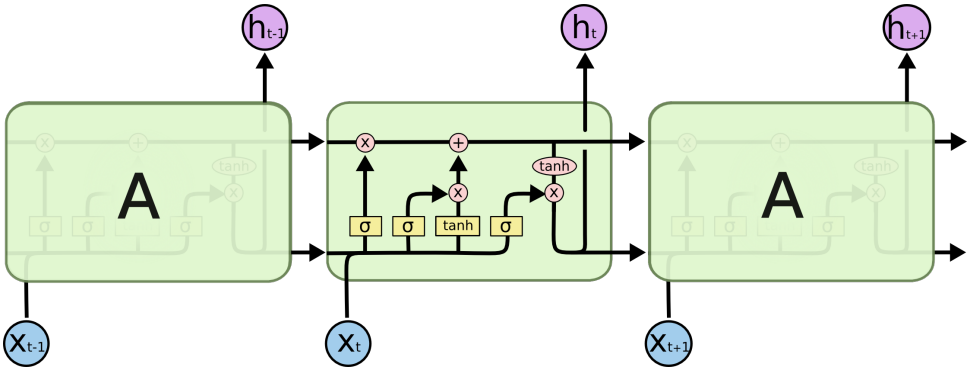


图 2.2-2 LSTM 中的重复模块包含四个交互的层

LSTM 的关键就是细胞状态，水平线在图上方贯穿运行。细胞状态类似于传送带。直接在整个链上运行，只有一些少量的线性交互。信息在上面流传保持不变会很容易。LSTM 拥有三个门，来保护和控制细胞状态。

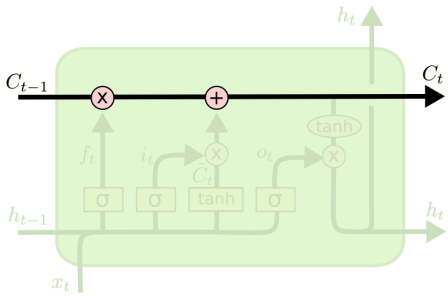
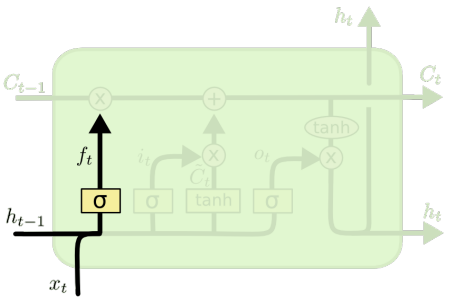


图 2.2-3 LSTM 中的细胞状态

在我们 LSTM 中的第一步是决定我们会从细胞状态中丢弃什么信息。这个决定通过一个称为忘记门层完成。该门会读取 h_{t-1} 和 x_t，输出一个在 0 到 1 之间的数值给每个在细胞状态 C_{t-1} 中的数字。1 表示“完全保留”，0 表示“完全舍弃”。

在语言模型的例子中来基于已经看到的预测下一个词，细胞状态可能包含当前主语的性别，因此正确的代词可以被选择出来。当我们看到新的主语，我们希望忘记旧的主语。



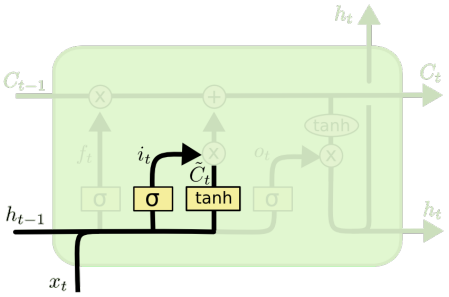
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

图 2.2-4 决定丢弃信息

下一步是确定什么样的新信息被存放在细胞状态中。这里包含两个部分。第一，sigmoid 层称“输入门层”决定什么值我们将要更新。然后，一个 tanh 层创建一个新的候选值向量，$\tilde{C}_t$，会被加入

到状态中。下一步，我们会讲这两个信息来产生对状态的更新。

在我们语言模型的例子中，我们希望增加新的主语的性别到细胞状态中，来替代旧的需要忘记的主语。



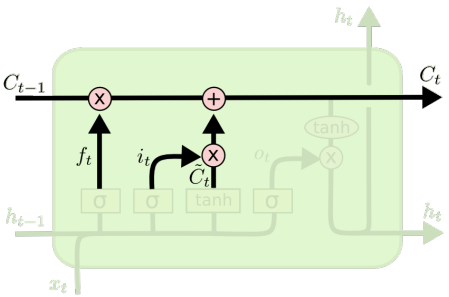
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

图 2.2-5 确定更新的信息

现在是更新旧细胞状态的时间了，C_{t-1} 更新为 C_t。前面的步骤已经决定了将会做什么，我们现在就是实际去完成。

我们把旧状态与 f_t 相乘，丢弃掉我们确定需要丢弃的信息。接着加上 $i_t * \tilde{C}_t$ 。这就是新的候选值，根据我们决定更新每个状态的程度进行变化。

在语言模型的例子中，这就是我们实际根据前面确定的目标，丢弃旧代词的性别信息并添加新的信息的地方。



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

图 2.2-6 更新细胞状态

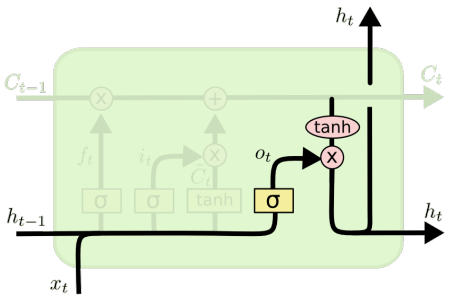
最终，我们需要确定输出什么值。这个输出将会基于我们的细胞状态，但是也是一个过滤后的版本。首先，我们运行一个 sigmoid 层来确定细胞状态的哪个部分将输出出去。接着，我们把细胞状态通过 tanh 进行处理（得到一个在 -1 到 1 之间的值）并将它和 sigmoid 门的输出相乘，最终我们仅仅会输出我们确定输出的那部分。

在语言模型的例子中，因为他就看到了一个代词，可能需要输出与一个动词相关的信息。例如，可能输出是否代词是单数还是负数，这样如果是动词的话，我们也知道动词需要进行的词形变化。

LSTM 对 RNN 做了改进，使得能够捕捉更长距离的信息。但是不管是 LSTM 还是 RNN，都有一个问题，它是从左往右推进的，因此后面的词会比前面的词更重要，但是对于分词这个任务来说是不妥的，因为句子各个字应该是平权的。因此出现了双向 LSTM，它从左到右做一次 LSTM，然后从右到左做一次 LSTM，然后把两次结果组合起来。

§2.3 Viterbi 算法

从第一个可能标签到最后一个可能标签的任何一条路径都可能产生一个新的序列，每条路径可能性不一样，我们需要做的是找出可能的路径。由于路径非常多，因此采用穷举是非常耗时的，因



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

图 2.2-7 输出信息

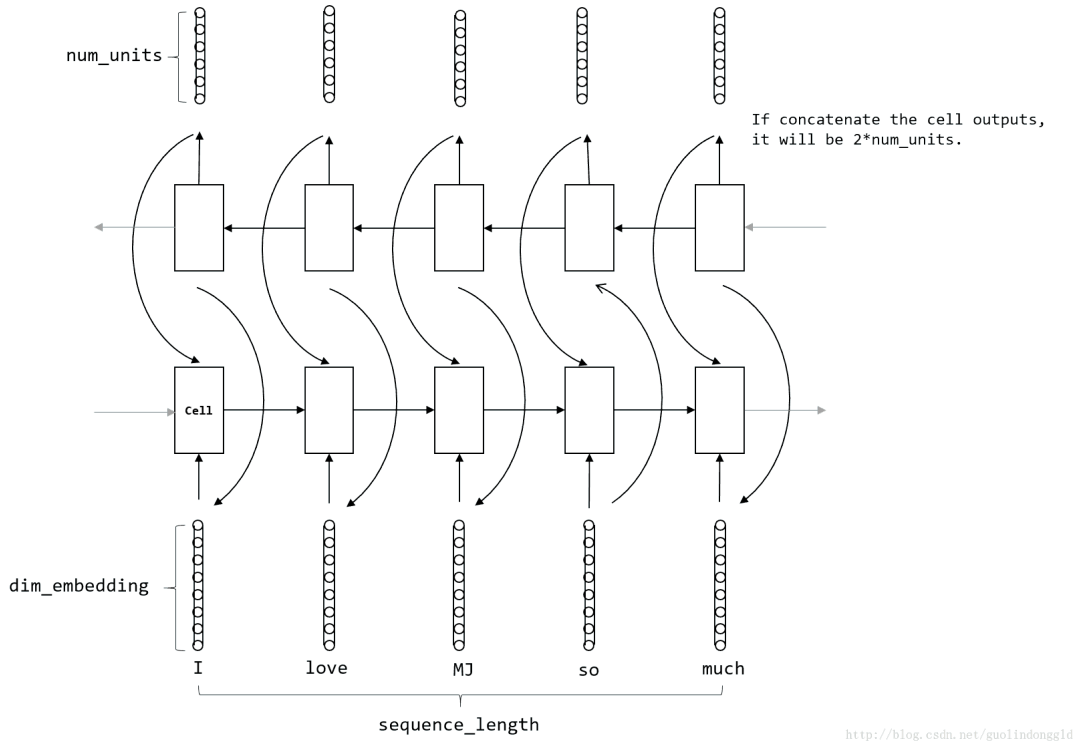


图 2.2-8 双向 LSTM

此引入 Viterbi 算法。

Viterbi 算法是一种动态规划算法。它用于寻找最有可能产生观测事件序列的维特比路径——隐含状态序列. 假设给定隐式马尔可夫模型（HMM）状态空间 S，共有 k 个状态，初始状态 i 的概率为 π_i ，从状态 i 到状态 j 的转移概率为 $a_{i,j}$。令观察到的输出为 $y_1, \ldots, y_T$。产生观察结果的最有可能的状态序列 $x_1, \ldots, x_T$ 由递推关系给出：

$$V_{1,k} = \mathrm{P}\big(y_1 \mid k\big) \cdot \pi_k$$
$$V_{t,k} = \max_{x \in S}\left(\mathrm{P}\big(y_t \mid k\big) \cdot a_{x,k} \cdot V_{t-1,x}\right)$$

此处 $V_{t,k}$ 是前 t 个最终状态为 k 的观测结果最有可能对应的状态序列的概率。通过保存向后指针记住在第二个等式中用到的状态 x 可以获得维特比路径。声明一个函数 $\mathrm{Ptr}(k,t)$ ，它返回若 $t > 1$ 时计算 $V_{t,k}$ 用到的 x 值或若 $t = 1$ 时的 k . 这样:

$$x_T = \arg\max_{x \in S}(V_{T,x})$$
$$x_{t-1} = \mathrm{Ptr}(x_t, t)$$

这里我们使用了 arg max 的标准定义算法复杂度为 $O(T \times |S|^2)$.

第 3 题

实验设置步骤

§3.1 数据预处理

§3.1.1 正则匹配标注与编码

首先我们需要将已经分好词的训练语料按照 4tag 标记转换为一系列标记. 这里我们采用了正则匹配的方法, 将所有的分隔符与标点符号先转换为 / 符号, 再对所有非斜杠的文字部分全部转换为 m 标记, 最后对 m 标记与斜杠 / 之间的相对位置关系再分别转换为 b,e,s 三种标记.

在这个过程中, 我们同时构建了一个包括数据集中出现过的所有字的字典, 在字典中, 每一个字都有唯一的一个序号. 假设一个句子包含 N 个字, 那么最后我们希望处理得到的形式为:

```
序号 1\quad 序号 2\quad\ldots 序号 N

标记 1\quad 标记 2\quad\ldots 标记 N
```

即序号唯一地标定了一个字, 而标记则确定了这个字在组成词中的成分.

§3.1.2 keras 简介

我们所需要用到的 embedding, LSTM 和双向网络在 keras 包里面已经实现, 我们只需要调用即可.Keras 是由纯 python 编写的基于 theano/tensorflow 的深度学习框架。Keras 是一个高层神经网络 API，支持快速实验. Keras 有两种类型的模型，序贯模型（Sequential）和函数式模型（Model），函数式模型应用更为广泛，序贯模型是函数式模型的一种特殊情况:

1. 序贯模型 (Sequential): 单输入单输出，一条路通到底，层与层之间只有相邻关系，没有跨层连接。这种模型编译速度快，操作也比较简单.
2. 函数式模型 (Model): 多输入多输出，层与层之间任意连接。

在 keras.layers 里面我们可以获取到 lstm 和双向网络的模型:

```
from keras.models import Model
from keras.layers import Input,Dense, Embedding, LSTM, Bidirectional, TimeDistributed
```

同时由于我们在预处理数据时候得到的标注序列无法被模型使用, 所以我们还需要将其转换为模型可以使用的数据形态, 也就是需要利用 keras 的 embedding 函数对标注序列进行进一步处理.

§3.2 网络模型

在得到了词向量 embedding 之后, 我们就可以将词向量输入 LSTM 进行处理. 模型有两个超参数: max_length, 与 word_size. 其中 max_length 指句子的长度, 我们初始设为 32, 即将句子长度超过 32 的进行截断, 只取前长为 32 的部分. 而 word_size 指词向量的长度, 初始设为 128, 它代表一个词向量的长度, 词向量长度不足 128 的部分我们将用 0 来进行填充. 网络模型具体参数如下:

```
Using TensorFlow backend.
[Reading Data...
Preparing data...
Saving the data...
Creating Network...
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 32)                0
_________________________________________________________________
embedding_1 (Embedding)      (None, 32, 128)           660352
_________________________________________________________________
bidirectional_1 (Bidirection (None, 32, 64)            98816
_________________________________________________________________
time_distributed_1 (TimeDist (None, 32, 5)             325
=================================================================
Total params: 759,493
Trainable params: 759,493
Non-trainable params: 0
_________________________________________________________________
None
Training...
Train on 78231 samples, validate on 8693 samples
```

图 3.2-1 网络参数

输入层接受一个词向量, embedding 层将其转化为一个长度为 word_size 的向量, 并输入到双向 LSTM 模型当中进行处理, 最后经过一层 softmax 层预测输出到 5 个标注的概率, 取其中最大概率的标注作为预测标注. 模型训练 20∼ 30 个 epoch, 每一个 epoch 当中, 将训练集按照 9:1 进行划分, 其中 9 份用于训练, 而剩下一份作为验证集, 训练时输出 loss 值与在验证集上的准确率 accuracy.

§3.3 序列标注预测

得到网络模型的参数之后, 我们进行保存, 并在测试集上进行分词预测. 在导入之前已经训练好的参数之后, 对于输入的一个词序列向量, 我们采用 Viterbi 算法来预测其对应的最大概率标注序列. 其中我们假设标注之间的转移概率的相等的:

```
zy = {'be': 0.5,
      'bm': 0.5,
      'eb': 0.5,
      'es': 0.5,
      'me': 0.5
```

```
    'mm': 0.5,
    'sb': 0.5,
    'ss': 0.5 }
```

得到分词序列标注后, 我们可以输出相应句子的分词结果, 得到最终的分词文档.

第 4 题

实验结果

§ 4.1 参数选择

§ 4.2 网络 loss, accuracy 变化

§ 4.3 最优模型的 F, recall, accuracy