【NLP实战】手把手带你HAN文本分类

AI小白入门 3月4日

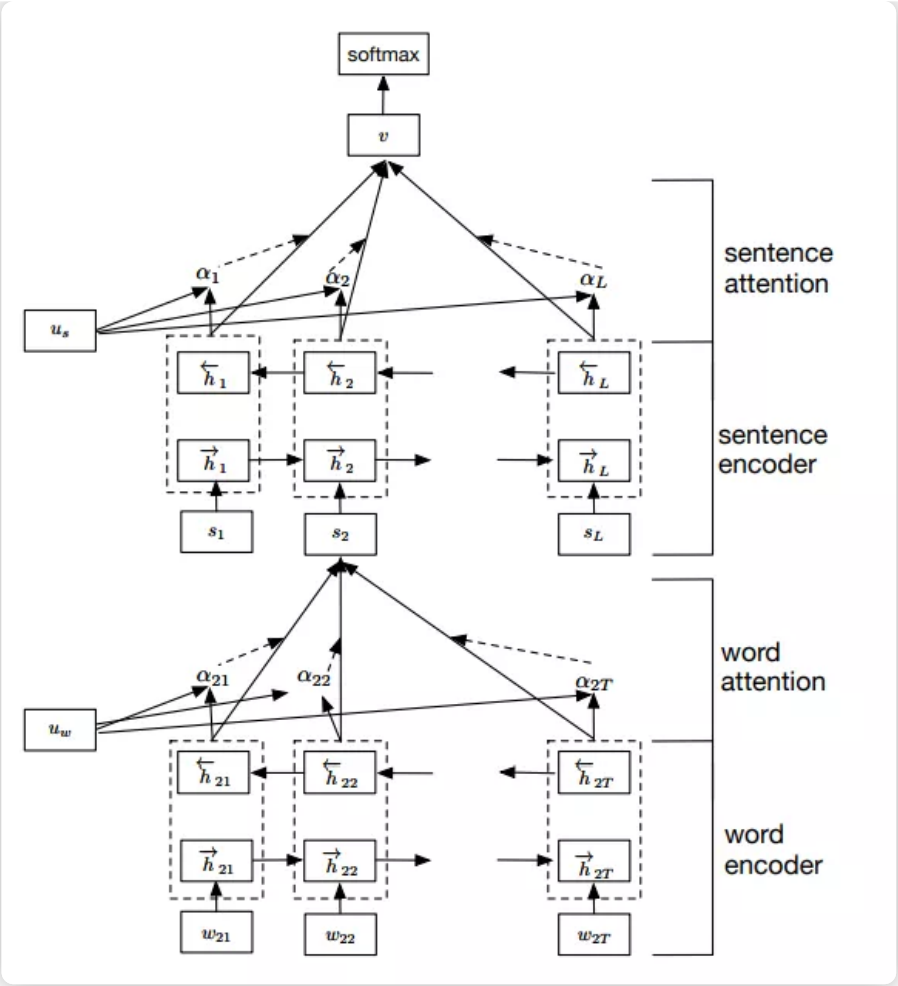
以下文章来源于NewBeeNLP ，作者kaiyuan

NewBeeNLP
一个自然语言处理&人工智能的原创杂货铺子，希望能找到你喜欢的小玩意儿

今天来看看网红Attention的效果，来自ACL的论文Hierarchical Attention Networks for Document Classification

论文概述

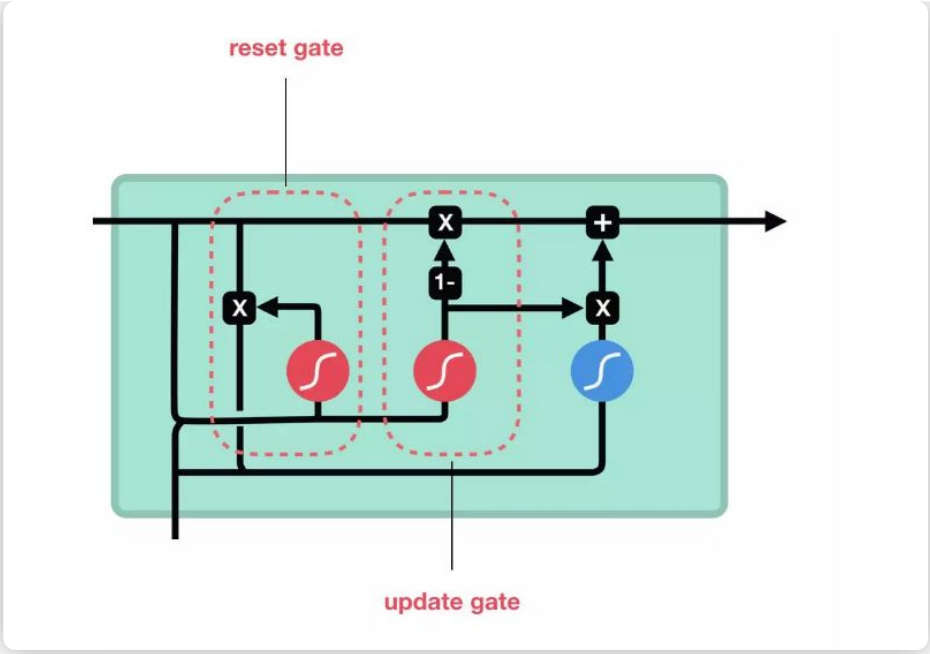
近年来，在NLP领域，好像最流行的就是RNN、LSTM、GRU、Attention等及其变体的组合框架。这篇论文里作者就对文本的结构进行分析，使用了双向GRU的结构，并且对Attention进行调整：考虑了word层面的attention和sentence层面的attention，分别对单词在句子中和句子在文档中的重要性进行了建模。仔细一想确实是挺合理的，一篇文档就是由无数句子构成的，而一个句子又是由无数单词构成的，充分考虑了文档的内部结构。



上图就是论文中文本分类模型的整体框架，可以看出主要分为四个部分：

- word encoder （BiGRU layer）
- word attention （Attention layer）
- sentence encoder （BiGRU layer）
- sentence attention （Attention layer）

首先回顾一下GRU的原理：



GRU是RNN的一个变种，使用门机制来记录当前序列的状态。在GRU中有两种类型的门（gate）:reset gate和update gate。这两个门一起控制来决定当前状态有多少信息要更新。

reset gate是用于决定多少过去的信息被用于生成候选状态，如果Rt为0，表明忘记之前的所有状态：

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r)$$

根据reset gate的分数可以计算出候选状态：

$$\tilde{h}_t = \tanh(W_h x_t + r_t \odot (U_h h_{t-1}) + b_h)$$

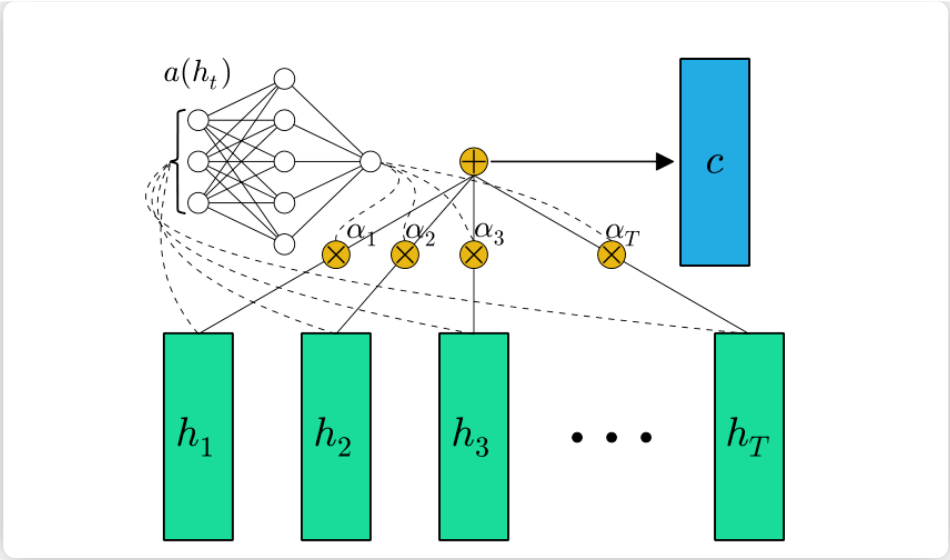
update gate是用来决定有多少过去的信息被保留，以及多少新信息被加进来：

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z)$$

最后，隐藏层状态的计算公式，有update gate、候选状态和之前的状态共同决定：

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t$$

接着来回顾一下Attention原理：



好啦，下面具体来看看论文里的模型：

1、word encoder layer

首先，将每个句子中的单词做embedding转换成词向量，然后，输入到双向GRU网络中，结合上下文的信息，获得该单词对应的隐藏状态输出$h_{it} = [\overrightarrow{h_{it}}, \tilde{h_{it}}]$

$$x_{it} = W_e w_{it}, t \in [1, T]$$
$$\overrightarrow{h_{it}} = G\vec{R}U(x_{it})$$
$$\dot{h}_{it} = GRU(x_{it})$$

2、word attention layer

attention机制的目的就是要把一个句子中，对句子表达最重要的单词找出来，赋予一个更大的比重。

首先将word encoder那一步的输出得到的h_{it}输入到一个单层的感知机中得到结果u_{it}作为其隐含表示

$$u_{it} = \tanh(W_w h_{it} + b_w)$$

接着为了衡量单词的重要性，定义了一个随机初始化的单词层面上下文向量$\mathcal{U}_w$，计算其与句子中每个单词的相似度，然后经过一个softmax操作获得了一个归一化的attention权重矩阵α_{it}，代表句子i中第t个单词的权重：

$$\alpha_{it} = \frac{\exp\left(u_{it}^{\top} u_w\right)}{\sum_t \exp\left(u_{it}^{\top} u_w\right)}$$

于是，句子的向量Si就可以看做是句子中单词的向量的加权求和。这里的单词层面上下文向量是随机初始化并且可以在训练的过程中学习得到的，我们可以把它看成是一种query的高级表示：“句子中哪些词含有比较重要的信息？”

$$s_i = \sum_t \alpha_{it} h_{it}$$

3、sentence encoder

通过上述步骤我们得到了每个句子的向量表示，然后可以用相似的方法得到文档向量 $h_i = [\vec{h_i}, \overleftarrow{h_i}]$

$$\vec{h}_i = \overrightarrow{\mathrm{GRU}}(s_i), i \in [1, L]$$
$$\overleftarrow{h}_i = \overleftarrow{\mathrm{GRU}}(s_i), t \in [L, 1]$$

4、sentence attention

和词级别的attention类似，作者提出了一个句子级别的上下文向量,来衡量一个句子在整篇文本的重要性。

$$u_i = \tanh(W_s h_i + b_s)$$
$$\alpha_i = \frac{\exp\left(u_i^\top u_s\right)}{\sum_i \exp\left(u_i^\top u_s\right)}$$
$$v = \sum_i \alpha_i h_i$$

5、softmax

上面的v向量就是我们得到的最后文档表示，然后输入一个全连接的softmax层进行分类就ok了。

6、模型效果

	Methods	Yelp'13	Yelp'14	Yelp'15	IMDB	Yahoo Answer	Amazon
Zhang et al., 2015	BoW	-	-	58.0	-	68.9	54.4
	BoW TFIDF	-	-	59.9	-	71.0	55.3
	ngrams	-	-	56.3	-	68.5	54.3
	ngrams TFIDF	-	-	54.8	-	68.5	52.4
	Bag-of-means	-	-	52.5	-	60.5	44.1
Tang et al., 2015	Majority	35.6	36.1	36.9	17.9	-	-
	SVM + Unigrams	58.9	60.0	61.1	39.9	-	-
	SVM + Bigrams	57.6	61.6	62.4	40.9	-	-
	SVM + TextFeatures	59.8	61.8	62.4	40.5	-	-
	SVM + AverageSG	54.3	55.7	56.8	31.9	-	-
	SVM + SSWE	53.5	54.3	55.4	26.2	-	-
Zhang et al., 2015	LSTM	-	-	58.2	-	70.8	59.4
	CNN-char	-	-	62.0	-	71.2	59.6
	CNN-word	-	-	60.5	-	71.2	57.6
Tang et al., 2015	Paragraph Vector	57.7	59.2	60.5	34.1	-	-
	CNN-word	59.7	61.0	61.5	37.6	-	-
	Conv-GRNN	63.7	65.5	66.0	42.5	-	-
	LSTM-GRNN	65.1	67.1	67.6	45.3	-	-
This paper	HN-AVE	67.0	69.3	69.9	47.8	75.2	62.9
	HN-MAX	66.9	69.3	70.1	48.2	75.2	62.9
	HN-ATT	**68.2**	**70.5**	**71.0**	**49.4**	**75.8**	**63.6**

Table 2: Document Classification, in percentage

代码实现

定义模型

```
class HAN(object):
    def __init__(self, max_sentence_num, max_sentence_length, num_classes,
                 embedding_size, learning_rate, decay_steps, decay_rate,
                 hidden_size, l2_lambda, grad_clip, is_training=False,
                 initializer=tf.random_normal_initializer(stddev=0.1)):
        self.vocab_size = vocab_size
        self.max_sentence_num = max_sentence_num
        self.max_sentence_length = max_sentence_length
        self.num_classes = num_classes
        self.embedding_size = embedding_size
        self.hidden_size = hidden_size
        self.learning_rate = learning_rate
        self.decay_rate = decay_rate
        self.decay_steps = decay_steps
        self.l2_lambda = l2_lambda
        self.grad_clip = grad_clip
        self.initializer = initializer

        self.global_step = tf.Variable(0, trainable=False, name='global_ste

        # placeholder
        self.input_x = tf.placeholder(tf.int32, [None, max_sentence_num, ma
```

```
        self.input_y = tf.placeholder(tf.int32, [None, num_classes], name='
        self.dropout_keep_prob = tf.placeholder(tf.float32, name='dropout_k

        ifnot is_training:
            return

        word_embedding = self.word2vec()
        sen_vec = self.sen2vec(word_embedding)
        doc_vec = self.doc2vec(sen_vec)

        self.logits = self.inference(doc_vec)
        self.loss_val = self.loss(self.input_y, self.logits)
        self.train_op = self.train()
        self.prediction = tf.argmax(self.logits, axis=1, name='prediction')
        self.pred_min = tf.reduce_min(self.prediction)
        self.pred_max = tf.reduce_max(self.prediction)
        self.pred_cnt = tf.bincount(tf.cast(self.prediction, dtype=tf.int32
        self.label_cnt = tf.bincount(tf.cast(tf.argmax(self.input_y, axis=1
        self.accuracy = self.accuracy(self.logits, self.input_y)
```

- END -

The End

方便交流学习，备注：**昵称-学校or公司-方向**，进入DL&NLP交流群。

记得备注呦

【推荐阅读】

初学者|NLP相关任务简介

【科研】自然语言理解难在哪儿?

自然语言处理中注意力机制综述

新年送福气|您有一份NLP大礼包待领取

“达观杯”文本智能处理挑战赛，季军带你飞

【机器学习】一文读懂线性回归、岭回归和Lasso回归