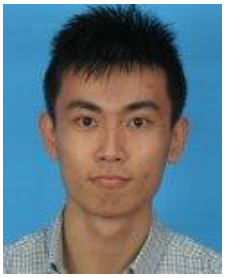
Neural Relation Extraction with Selective Attention over Instances



Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, Maosong Sun

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 2124–2133, Berlin, Germany, August 7-12, 2016.

说到两个实体之间的关系，比如 Microsoft 和 Bill Gates 这俩词吧，他们之间关系就是 Bill Gates 是 Microsoft 的 founder

Microsoft ←(founder)— Bill Gates

⇩ 以往我们总是从一个句子中来判断他们之间的关系，比如下面这个句子中

Bill Gates is the founder Sentence of Microsoft.
(founder)

⇩ 就明确的说明了他们之间的关系。但是现在有很多个句子也包含了 Microsoft 和 Bill Gates 这俩单词，是不是从多个句子的集合来判断这俩单词的关系，会比从一个单一的句子来判断来的更准确一点呢？

⇩ 那就试试？好那就试试。现在有很多句子的集合，都有这俩词 Microsoft 和 Bill Gates

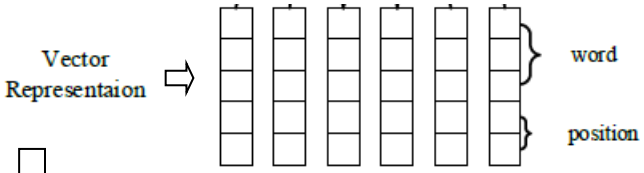
$$\{x_1, x_2, \cdots, x_n\}$$

⇩ 这些句子中的 Microsoft 和 Bill Gates 有很多种可能的关系，那么如何判断前他们是其中某种关系 relation r 的概率呢？为了算这个概率，要先表达这些句子。首先我们考虑其中每个句子，x，其中有 m 个单词

$$x = \{w_1, w_2, \cdots, w_m\}$$

⇩ 为了表达这个句子，我们首先把每个单词的 word embedding 矢量给找出来。然后把每个单词跟 Microsoft 和 Bill Gates 的相对位置的 embedding 也找出来。这些 embedding 就组合在一起形成所有这些单词的 embedding 矢量

$$\mathbf{w} = \{\mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_m\}$$



⇩ 然后就用一个 window 吧每个单词前面的 l 个单词给框出来，做成这个单词新的特征矢量

$$\mathbf{q}_i = \mathbf{w}_{i-l+1:i}$$

⇩ 然后就送入 convolutional neural network 去处理了。第一层就是先搞几个 fitler，对于第 i 个 filter，就是对所有的 window 搞个线性变换：

$$\mathbf{p}_i = [\mathbf{W}\mathbf{q} + \mathbf{b}]_i$$

⇩ 搞完了以后呢，以 Microsoft 和 Bill Gates 为界（这俩词叫做 head and tail entities），吧整个句子分成三部分，同事把这些 filter 的结果也按照 head and tail entities 为界分成三部分

$$(\mathbf{p}_{i1}, \mathbf{p}_{i2}, \mathbf{p}_{i3})$$

⇩ 分然后对于每一部分做 max-pooling 选其中最大的

$$[\mathbf{x}]_{ij} = \max(\mathbf{p}_{ij})$$

⇩ 如此这般，对于每个像（Microsoft, Bill Gates）这样的$(head, tail)$的组合，我们把所有包含他们的句子$S = \{x_1, x_2, \cdots, x_n\}$都用 CNN 变成矢量

$$\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n.$$

⇩ 然后为了表达整个的包含$(head, tail)$的句子的集合$\{x_1, x_2, \cdots, x_n\}$，我们把这些矢量加权平均

$$\mathbf{s} = \sum_i \alpha_i \mathbf{x}_i$$

关键就是如何算这些句子的权重α_i，其实就是到底挑选哪些句子来预测$(head, tail)$之间的关系比较靠谱的问题了。

为了看清一个包含$(head, tail)$的句子$\mathbf{x}_i$在预测$(head, tail)$是不是有 relation r 关系的问题上的贡献，我们首先把一个 relation r 表示成一个矢量$\mathbf{r}$，

relation r ⇨ $\mathbf{r}$,

⇩ 然后就匹配一下$\mathbf{r}$和这个句子的矢量$\mathbf{x}_i$

$$e_i = \mathbf{x}_i \mathbf{A} \mathbf{r}$$

↑ $\mathbf{A}$ is a weighted diagonal matrix 是匹配的参数

⇩ 就得到了这个句子用来预测$(head, tail)$的 relation r 的靠谱分数。这时我们还要从众多句子中得到这个句子的概率来当做α_i，就用 softmax function 就可以了

$$\alpha_i = \frac{\exp(e_i)}{\sum_k \exp(e_k)}$$

那有了$(head, tail)$的句子的集合 S 的矢量以后，就可以看看每个可能的 relation r 的分数了，我们还是用一个线性的函数来做

$$\mathbf{o} = \mathbf{M}\mathbf{s} + \mathbf{d}$$

⇩ 这样就得到了一共n_r个可能的 relation r 的分数。为了算出每个 relation r 的概率，我们还是用一个 softmax 函数

$$p(r|S,\theta) = \frac{\exp(o_r)}{\sum_{k=1}^{n_r}\exp(o_k)},$$

⇩ 为了学习其中所有模型的参数 θ，我们所有的训练的句子集合和对应正确的 relation r 作为训练集合(S_i,r_i)， $i=1$ …s,套进这个概率模型，再加一个 log 函数

$\log p(r_i|S_i,\theta)$， $i=1$ …s

⇩ 既然是正确的 relation r，那自然是这些概率的 log 加起来越大越好

$$\text{Max}_{\theta} \sum_{i=1}^{s} \log p(r_i|S_i,\theta)$$

这个问题用 stochastic gradient descent (SGD)解就行了。

好的，打完收工！

启发：

1. 之前都是用一个句子作为一个单位处理，这里别出心裁用一个句子集合来做单位，很有新意
2. 每个句子的权重用了一个简单的 relation-sentence match 的方法来算。能不能作为一个变量？加上一些约束来算？
3. Relation Extraction 是一个很有趣的话题，可以多关注一下。