Report for NLP-Beginner task 3

基于注意力机制的文本匹配

叶栩冰

Report for NLP-Beginner task 3

1 Task
2 Environment & Data
2.1 Environment
2.2 Data
3 Method & Model
3.1 Embedding
3.2 ESIM
3.2.1《Enhanced LSTM for Natural Language Inference》
3.2.2 模型架构
3.2.2.1 Input Encoding
3.2.2.2 Local Inference Modeling
3.2.2.3 Inference Composition
3.3 Model
4 Train & Result
4.1 Train
4.2 Result
4.3 Analysis

---

1 Task

输入两个句子判断，判断它们之间的关系。参考ESIM（可以只用LSTM，忽略Tree-LSTM），用双向的注意力机制实现。

1. 参考
    1. 《神经网络与深度学习》 第7章
    2. Reasoning about Entailment with Neural Attention https://arxiv.org/pdf/1509.06664v1.pdf
    3. Enhanced LSTM for Natural Language Inference https://arxiv.org/pdf/1609.06038v3.pdf
2. 数据集：https://nlp.stanford.edu/projects/snli/
3. 实现要求：Pytorch
4. 知识点：
    1. 注意力机制
    2. token2token attention
5. 时间：两周

2 Environment & Data

2.1 Environment

> Python ~= 3.6
>
> IDE : Jetbrains Pycharm
>
> torch ~= 1.10.0
>
> pandas ~= 0.19.2
>
> pickle ~= 0.12.0
>
> time ~= 1.0

2.2 Data

The Stanford Natural Language Inference (SNLI) Corpus. The Stanford Natural Language Inference (SNLI) corpus (version 1.0) is a collection of 570k human-written English sentence pairs manually labeled for balanced classification with the labels *entailment*, *contradiction*, and *neutral*. We aim for it to serve both as a benchmark for evaluating representational systems for text, especially including those induced by representation-learning methods, as well as a resource for developing NLP models of any kind.

训练集共有55万余项，语言为英文，匹配关系共有四种：蕴含（Entailment），矛盾（Contradiction），中立/不冲突（Neutral），未知（样本量较少）。

Example：

> **Input**： A man inspects the uniform of a figure in some East Asian country.
> **hypothesis**： The man is sleeping.
> **Output**： 矛盾（C）

> **Input**： An older and younger man smiling.
> **hypothesis**： Two men are smiling and laughing at the cats playing on the floor.
> **Output**： 中立（N）

> **Input**： A black race car starts up in front of a crowd of people.
> **hypothesis**： A man is driving down a lonely road.
> **Output**： 矛盾（C）

> **Input**： A soccer game with multiple males playing.
> **hypothesis**： Some men are playing a sport.
> **Output**： 蕴含（E）

3 Method & Model

3.1 Embedding

同task2

3.2 ESIM

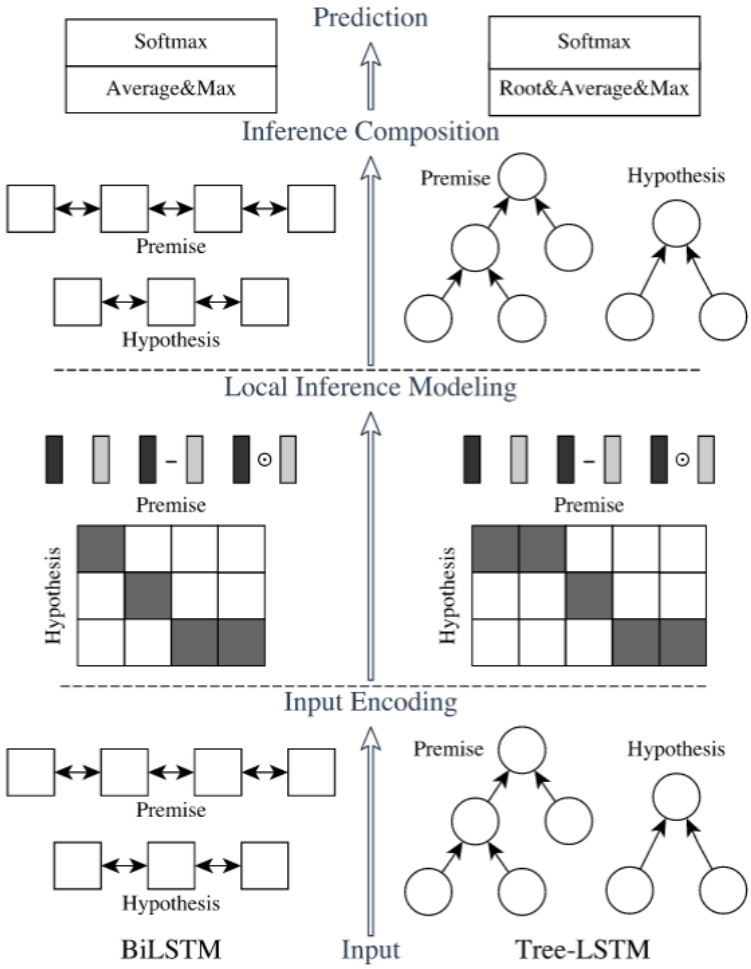
3.2.1《Enhanced LSTM for Natural Language Inference》

本文主要做的事情就是自然语言推断（Natural Language Inference，NLI），即给定前提premise和假设hypothesis，要求判断两者的关系（1.不相干neural；2.冲突contradiction，即有矛盾，3.蕴含entailment，即能从p推断h或者两者表达相同的意思），与本任务所给数据集匹配。

这是该文章的重点部分，引入了句子间的注意力机制（intra-sentence attention），来实现局部的推断，进而实现全局的推断。

- 基于链式LSTMs精心设计了序列推断模型 (carefully designing sequential inference models based on chain LSTMs)；
- 考虑局部推断和推断组合（in both local inference modeling and inference composition）

3.2.2 模型架构



上图是ESIM的核心框架，主要有三部分：Input Encoding、Local Inference Modeling 和Inference Composition。左半部分就是我们要讲的ESIM，右半部分的区别在于使用了一种叫Tree-LSTM的变种LSTM结构，可用于做句子的语法分析。

3.2.2.1 Input Encoding

很简单，就是输入的两句话分别接embedding 和BiLSTM（作者使用的应该是静态词向量，通过双向LSTM来表达句子局部信息），在论文中，作者也解释了为什么不使用BiGRU，因为实验效果没有BiLSTM好。

$$\bar{a}_i = BILSTM(a, i), \forall i \in [1, \ldots, l_a]$$

$$\bar{b}_j = BiLSTM(b, j), \forall j \in [1, \dots, l_b]$$

使用 BiLSTM 可以学习如何表示一句话中的 word 和它上下文的关系，我们也可以理解成这是 在 word embedding 之后，在当前的语境下重新编码，得到新的 embeding 向量。

3.2.2.2 Local Inference Modeling

在做Local Inference之前需要对上面得到的 $\bar{a}_i$ 和 $\bar{b}_j$ 进行alignment（有点attention的感觉）。

首先，计算相似度矩阵：

$$e_{ij} = \bar{a}_i^T \bar{b}_j$$

在进行两句话的Local Inference，结合 $\bar{a}_i$ 、 $\bar{b}_j$ 和 e_{ij} ，生成相似性加权后的向量：

$$\tilde{a}_i = \sum_{j=1}^{l_b} \frac{exp(e_{ij})}{\sum_{k=1}^{l_b} exp(e_{ik})} \bar{b}_j, \forall i \in [1, \dots, l_a]$$

$$\tilde{b}_j = \sum_{j=1}^{l_a} \frac{exp(e_{ij})}{\sum_{k=1}^{l_a} exp(e_{kj})} \bar{a}_i, \forall j \in [1, \dots, l_b]$$

在Local Inference之后，进行Enhancement of local inference information。使用的方法就是，计算差和点击，来体现一种差异性（也算是一种构造的特征）。

$$m_a = [\bar{a}; \tilde{a}; \bar{a} - \tilde{a}; \bar{a} \odot \tilde{a}]$$

$$m_b = [\bar{b}; \tilde{b}; \bar{b} - \tilde{b}; \bar{b} \odot \tilde{b}]$$

3.2.2.3 Inference Composition

最后一步就比较简单了，对 m_a 和 m_b 再使用BiLSTM来提取信息。但是这样得到的两个句子矩阵的维度是和句子本身长度有关系的，所以不一定维度一致，作者通过使用MaxPooling和AvgPooling的池化操作，使维度能够保持一直（一定程度上还可以理解为简化计算且不失重要特征），最后连接一个全连接层，进行预测。

$v_{a,i} = BiLSTM(m_a)$ →AvgPooling: $v_{a,ave} = \sum_{i=1}^{l_a} \frac{v_{a,i}}{l_a}$ ； MaxPooling:

$v_{a,max} = \max_{i=1}^{l_a} v_{a,i}$

$v_{b,j} = BiLSTM(m_b)$ →AvePooling: $v_{b,ave} = \sum_{j=1}^{l_b} \frac{v_{b,j}}{l_b}$ ； MaxPooling:

$v_{b,max} = \max_{j=1}^{l_b} v_{b,j}$

$v = [v_{a,ave}; v_{a,max}; v_{b,ave}; v_{b,max}]$ →MLP→softmax（Output）

3.3 Model

```
self._word_embedding = nn.Embedding(self.vocab_size,
                                    self.embedding_dim,
                                    padding_idx=padding_idx,
                                    _weight=embeddings)
        if self.dropout:
            self._rnn_dropout = Dropout(p=self.dropout)
            # self._rnn_dropout = nn.Dropout(p=self.dropout)
        self._encoding = Seq2SeqEncoder(nn.LSTM,
                                        self.embedding_dim,
                                        self.hidden_size,
                                        bidirectional=True)
        self._attention = SoftmaxAttention()
        self._projection = nn.Sequential(nn.Linear(4*2*self.hidden_size,
self.hidden_size),
                                         nn.ReLU())
        self._composition = Seq2SeqEncoder(nn.LSTM,
                                           self.hidden_size,
                                           self.hidden_size,
                                           bidirectional=True)
        self._classification = nn.Sequential(nn.Dropout(p=self.dropout),
                                             nn.Linear(2*4*self.hidden_size,
self.hidden_size),
                                             nn.Tanh(),
                                             nn.Dropout(p=self.dropout),
                                             nn.Linear(self.hidden_size,
self.num_classes))
        self.apply(_init_esim_weights)
```

4 Train & Result

4.1 Train

由于数据量较大，因此借助colab完成。本作业在训练前对样本进行了按句子长短排序的处理，为防止长句子的存在使短句子需要padding过长的现象发生。

具体训练参数如下：

- batch_size = 512
- patience = 5
- hidden_size = 50
- dropout = 0.5
- num_classes = 3
- lr = 0.0004
- epochs = 1
- max_grad_norm = 10.0

4.2 Result

本作业对GLove/Random、排序/未排序的组合进行了 训练与和比较，结果如下。

Model	Accuracy	Recall
ESIM + Random + Sorted	0.784514	0.802746
ESIM + Glove + Sorted	0.792022	0.814470
ESIM + Random + Unsorted	0.820554	0.832511
ESIM + Glove + Unsorted	0.819726	0.829747

4.3 Analysis

1. Glove & Random

   从结果可分析GloVe初始化要比随机初始化的结果表现要显著更好。随机初始化是不遵循任何规律进行初始化，因此使用GloVe更具有优势。

2. Sorted & Unsorted

   本作业在训练前对样本进行了按句子长短排序的处理，为防止长句子的存在使短句子需要padding过长的现象发生。但是不对文本句子进行排序，只保证同一个batch内句子padding到同一个长度的实验反而测试集准确率有提升。

3. 对《Enhanced LSTM for Natural Language Inference》论文的一些想法。

   - LSTM抽取上下文信息。Tree-LSTM的尝试也为信息抽取带来启发。
   - 多种交互形式的特征进行concat，多种信息抽取方式进行组合。
   - attention的使用，其实有出处A Decomposable Attention Model for Natural Language Inference，思想其实是两者一一对比得到交互矩阵，利用该交互矩阵构造类似softmax的权重，为各自的关键信息进行加权，重点提取。transformer的出现更是将attention推向高潮。
   - 信息的对比来自于可以来自减和乘，减直接计算两者的差距，类似欧氏距离，乘的使用则来源于余弦距离，既然要对比特征，那就把这两个用到极致。
   - 其实我在此之前没接触过ree-lstm，我们平时比较难用到，原因是这个树不好构建，需要依赖依存句法。