Task6 miniWatson Final Report

何青蓉 2018202185
组员：徐轶琦

1. 项目分工

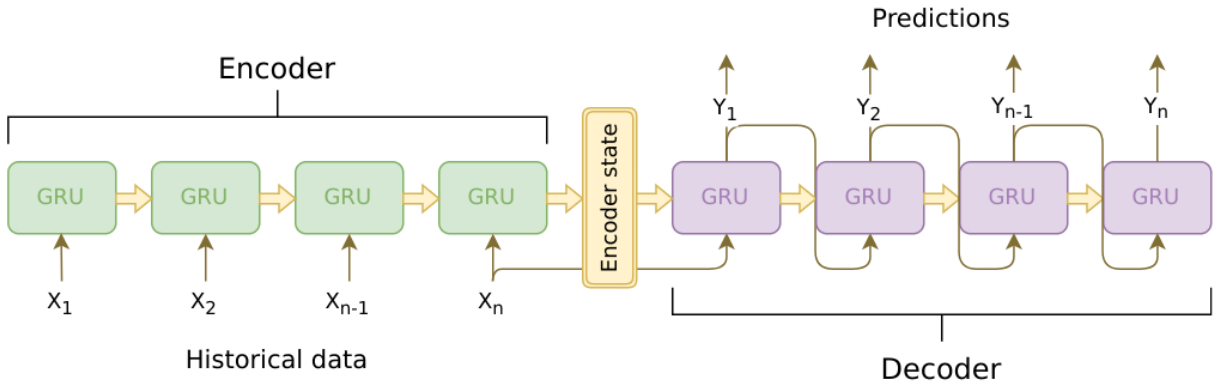
- 1.1 前期准备
  - 文献搜集：徐轶琦
  - 相关代码、实现框架的搜集和复现：何青蓉
- 1.2 数据预处理
  - 分词、构建词典、去除低频词和过长的句子：徐轶琦
- 1.3 模型的建立
  - encoder、decoder模型的建立：何青蓉
- 1.4 模型的训练
  - 解码算法的定义、loss函数的定义：徐轶琦
- 1.5 模型的优化
  - attention机制：何青蓉
  - teaching force：徐轶琦
- 1.6 交互界面
  - 交互界面设计：何青蓉

2. 我完成的部分

- 2.1 前期准备：相关代码、实现框架的搜集和复现

  本项目实现miniWatson采用的是生成式模型——基于RNN的Seq2Seq序列到序列模型。Seq2Seq属于encoder-decoder结构的一种，利用两个RNN，一个RNN作为encoder，负责将输入的序列压缩成指定长

度的向量，这个向量表征了序列的语义；另一个RNN作为decoder，负责根据语义向量生成指定的序列。

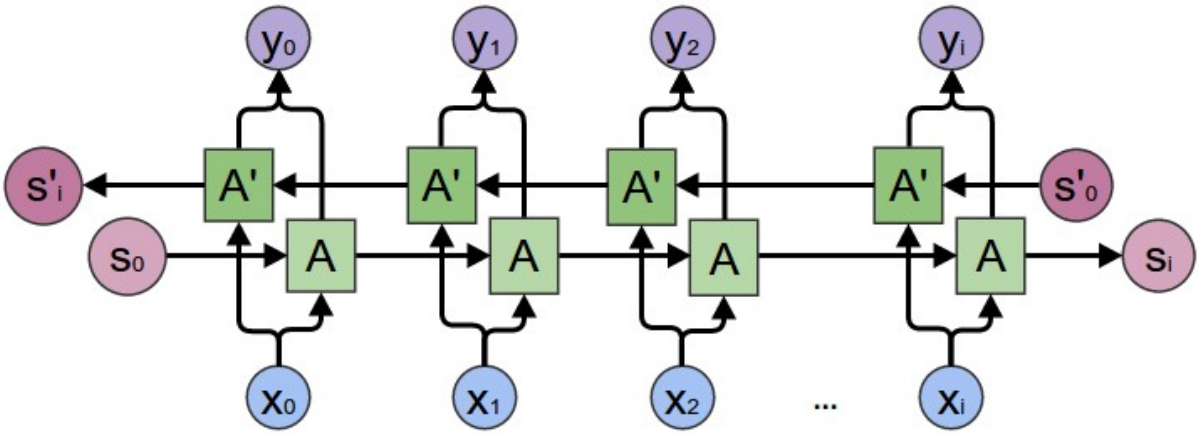


- 2.2 模型的建立：encoder、decoder

- **encoder**

使用多层的Gated Recurrent Unit（GRU）作为Encoder，遍历每个词（Token），每个时刻的输入是上一个时刻的隐状态和输入，产生一个输出和新的隐状态，作为下一个时刻的输入隐状态。把最后一个时刻的隐状态作为Decoder的初始隐状态。最终返回所有时刻的输出和最后时刻的隐状态。

（1）把词的ID通过embedding层变成向量
（2）把padding后的数据进行pack
（3）传入GRU进行forward计算
（4）unpack计算结果
（5）把双向GRU的结果向量加起来
（6）返回所有时刻的输出和最后时刻的隐状态




```
class EncoderRNN(nn.Module):
    def __init__(self, hidden_size, embedding, n_layers=1, dropout=0):
        super(EncoderRNN, self).__init__()
        self.n_layers = n_layers
        self.hidden_size = hidden_size
        self.embedding = embedding
        self.gru = nn.GRU(hidden_size, hidden_size, n_layers, dropout=(0 if
n_layers == 1 else dropout), bidirectional=True)

    def forward(self, input_seq, input_lengths, hidden=None):
```

```
        embedded = self.embedding(input_seq)
        packed = torch.nn.utils.rnn.pack_padded_sequence(embedded, input_lengths)
        outputs, hidden = self.gru(packed, hidden)
        outputs, _ = torch.nn.utils.rnn.pad_packed_sequence(outputs)
        outputs = outputs[:, :, :self.hidden_size] + outputs[:, :
,self.hidden_size:]
        return outputs, hidden
```

- **decoder**

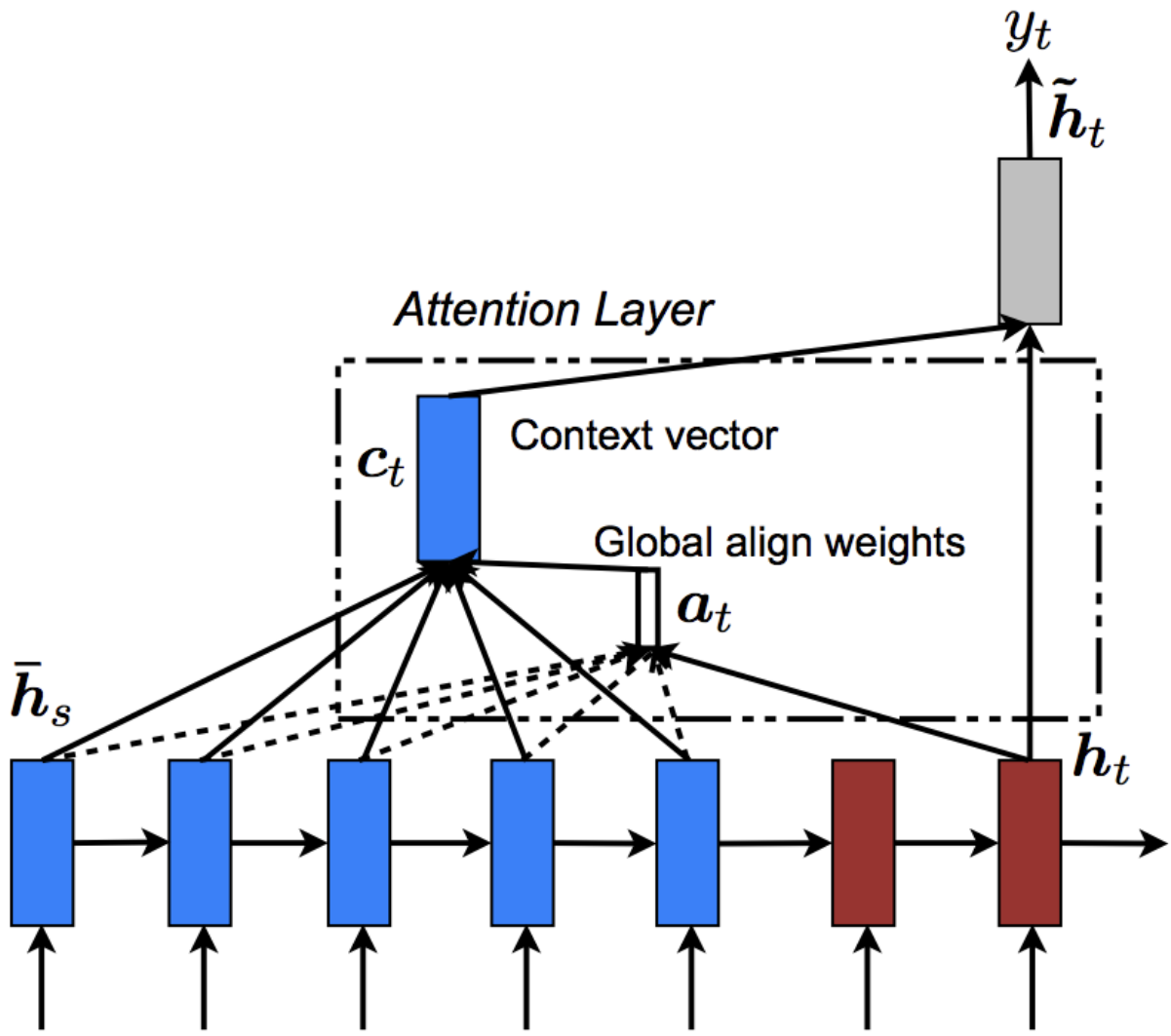
  每个时刻的输入是上一个时刻的隐状态和上一个时刻的输出。初始隐状态是Encoder最后时刻的隐状态。利用RNN计算新的隐状态和输出的第一个词，接着用新的新状态和第一个词计算第二个词，以此类推直到遇到结束符。

  （1）把词ID输入embedding层
  （2）使用单向的GRU继续Forward进行一个时刻的计算
  （3）使用新的隐状态计算注意力权重
  （4）用注意力权重得到context向量
  （5）context向量和GRU的输出拼接起来，经过一个全连接网络，使得输出大小仍然是hidden_size
  （6）使用一个投影矩阵把输出从hidden_size变成词典大小，然后用softmax变成概率
  （7）返回输出和新的隐状态

```
class LuongAttnDecoderRNN(nn.Module):
    def __init__(self, attn_model, embedding, hidden_size, output_size,
n_layers=1, dropout=0.1):
        super(LuongAttnDecoderRNN, self).__init__()
        self.attn_model = attn_model
        self.hidden_size = hidden_size
        self.output_size = output_size
        self.n_layers = n_layers
        self.dropout = dropout

        self.embedding = embedding
        self.embedding_dropout = nn.Dropout(dropout)
        self.gru = nn.GRU(hidden_size, hidden_size, n_layers, dropout=(0 if
n_layers == 1 else dropout))
        self.concat = nn.Linear(hidden_size * 2, hidden_size)
        self.out = nn.Linear(hidden_size, output_size)

        self.attn = Attn(attn_model, hidden_size)

    def forward(self, input_step, last_hidden, encoder_outputs):
        embedded = self.embedding(input_step)
        embedded = self.embedding_dropout(embedded)
        rnn_output, hidden = self.gru(embedded, last_hidden)
        attn_weights = self.attn(rnn_output, encoder_outputs)
        context = attn_weights.bmm(encoder_outputs.transpose(0, 1))
        rnn_output = rnn_output.squeeze(0)
        context = context.squeeze(1)
        concat_input = torch.cat((rnn_output, context), 1)
        concat_output = torch.tanh(self.concat(concat_input))
```

```
        output = self.out(concat_output)
        output = F.softmax(output, dim=1)
        return output, hidden
```

- 2.3 模型的优化：attention机制



用当前时刻的GRU计算出的新的隐状态来计算注意力得分，首先它用一个score函数计算这个隐状态和Encoder的输出的相似度得分，得分越大，说明越应该注意这个词。然后再用softmax函数把score变成概率。

score函数计算方法：

$$\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) = \begin{cases} \boldsymbol{h}_t^\top \bar{\boldsymbol{h}}_s & \textit{dot} \\ \boldsymbol{h}_t^\top \boldsymbol{W}_a \bar{\boldsymbol{h}}_s & \textit{general} \\ \boldsymbol{v}_a^\top \tanh\left(\boldsymbol{W}_a [\boldsymbol{h}_t; \bar{\boldsymbol{h}}_s]\right) & \textit{concat} \end{cases}$$

```
class Attn(torch.nn.Module):
    def __init__(self, method, hidden_size):
        super(Attn, self).__init__()
        self.method = method
        if self.method not in ['dot', 'general', 'concat']:
            raise ValueError(self.method, "is not an appropriate attention
method.")
        self.hidden_size = hidden_size
        if self.method == 'general':
            self.attn = torch.nn.Linear(self.hidden_size, hidden_size)
        elif self.method == 'concat':
            self.attn = torch.nn.Linear(self.hidden_size * 2, hidden_size)
            self.v = torch.nn.Parameter(torch.FloatTensor(hidden_size))

    def dot_score(self, hidden, encoder_output):
        return torch.sum(hidden * encoder_output, dim=2)

    def general_score(self, hidden, encoder_output):
        energy = self.attn(encoder_output)
        return torch.sum(hidden * energy, dim=2)

    def concat_score(self, hidden, encoder_output):
        energy = self.attn(torch.cat((hidden.expand(encoder_output.size(0), -1,
-1), encoder_output), 2)).tanh()
        return torch.sum(self.v * energy, dim=2)

    def forward(self, hidden, encoder_outputs):
        if self.method == 'general':
            attn_energies = self.general_score(hidden, encoder_outputs)
        elif self.method == 'concat':
            attn_energies = self.concat_score(hidden, encoder_outputs)
        elif self.method == 'dot':
            attn_energies = self.dot_score(hidden, encoder_outputs)
        attn_energies = attn_energies.t()
        return F.softmax(attn_energies, dim=1).unsqueeze(1)
```

- 2.4 交互界面

借助django实现前后端交互。

- 前端
  - css、js渲染html
  - 用户输入message，js响应，ajax把message送到后端并接收后端传回的reply。
- 后端
  - 载入词典
  - 载入训练好的encoder、decoder模型，代入解码器searcher
  - 接收前端传回的message
  - 将得分score最大的词ID集合还原为词序列，如果遇到没有收入词典的message则报错处理，将reply返回前端

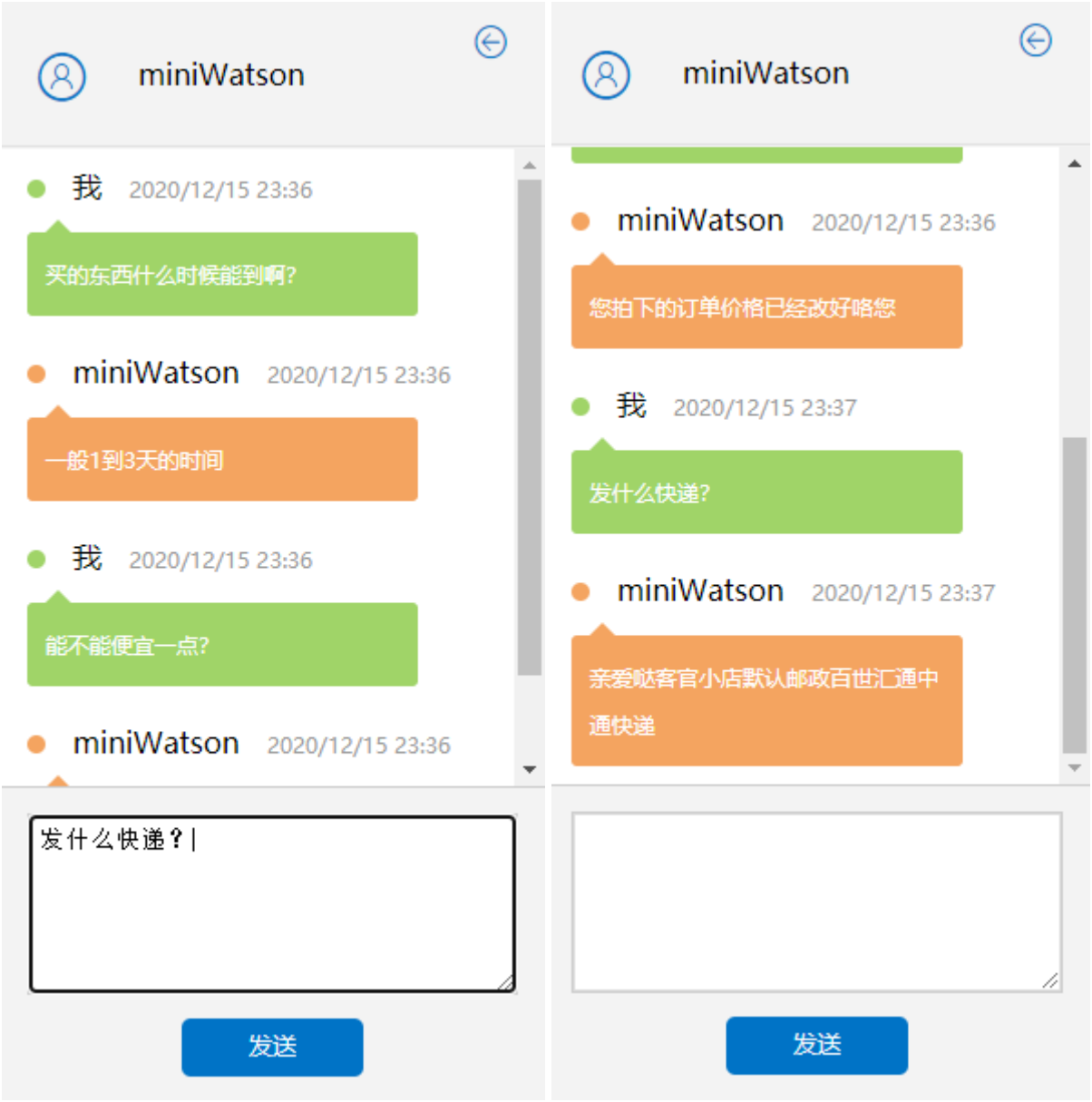
```
# model.py
attn_model = 'dot'
hidden_size = 500
encoder_n_layers = 2
decoder_n_layers = 2
dropout = 0.1
voc = prepare()
embedding = nn.Embedding(voc.num_words, hidden_size)
#调用模型
enstate = torch.load(os.path.join(BASE_DIR, 'module','encoder_10'))
destate = torch.load(os.path.join(BASE_DIR, 'module','decoder_10'))
#实例化
test_encoder = EncoderRNN(hidden_size, embedding, encoder_n_layers, dropout)
test_decoder = LuongAttnDecoderRNN(attn_model, embedding, hidden_size,
voc.num_words,
                    decoder_n_layers, dropout)
#加载参数
test_encoder.load_state_dict(enstate)
test_decoder.load_state_dict(destate)

test_encoder.eval()
test_decoder.eval()
# 构造searcher对象
test_searcher = GreedySearchDecoder(test_encoder, test_decoder)

def evaluateInput(input_sentence, encoder=test_encoder, decoder=test_decoder,
searcher=test_searcher, voc = voc):
    try:
        input_sentence = normalizeString(input_sentence)
        output_words = evaluate(encoder, decoder, searcher, voc, input_sentence)
        words = []
        for word in output_words:
            if word == 'EOS':
                break
            elif word != 'PAD':
                words.append(word)
        return ''.join(words)
    except KeyError:
        return "miniWatson还没有学会怎么回答这个问题……"
```

```
# views.py
def index(request):
    return render(request, os.path.join(BASE_DIR, 'templates','index.html'))

def chat(request):
    context = {}
    context['reply'] = model.evaluateInput(request.GET['messages'])
    return JsonResponse(context)
```

3. 成果展示

- 前端界面：

- 部分测试样例：



4. 附录

- 4.1 模型下载地址

  链接：https://pan.baidu.com/s/1e4wKC2DQRN_7t0BrHz3r-w
  提取码：e8qh
  请将模型encoder_10、decoder_10下载后放到module文件夹下。

- 4.2 程序运行方式

  命令行：在 manage.py 所在目录下，运行
  python manage.py runserver 0.0.0.0:8080
  浏览器访问 127.0.0.1:8080/index