تمرین چهارم درس پردازش زبان طبیعی

نام تهیه‌کننده: ملیکا نوبختیان

نسخه: ۱

۱- بخش تئوری

ابتدا RNN معمولی را بررسی می‌کنیم. اگر در h_t که مقدار hidden state در زمان t است همه چیز را ساده کنیم و bias را از فرمول آن حذف کنیم به فرمول زیر خواهیم رسید:

$$h_t = \sigma(wh_{t-1})$$

برای انجام backpropagation نیاز به این داریم که مشتق جزئی hidden state را داشته باشیم که به صورت زیر خواهد بود:

$$\frac{dh_{t'}}{dh_t} = \prod_{k=1}^{t'-t} w\sigma'(wh_{t'-k}) = w^{t'-t}\sigma'(wh_{t'-k})$$

عبارت اول این مشتق یعنی $w^{t'-t}$ نقطه بحرانی این قسمت خواهد بود زیرا اگر کمتر از ۱ یا بیشتر از ۱ باشد در t-t' به صورت expotentiallyیا توانی به سمت صفر یا بی نهایت میل خواهد کرد و مشکل vanishing/exploding gradient را به وجود خواهد آورد.

در LSTM به جای hidden state ، cell state را خواهیم داشت که با s_t آن را نمایش می‌دهیم. مشتق آن به شکل زیر محاسبه می‌شود:

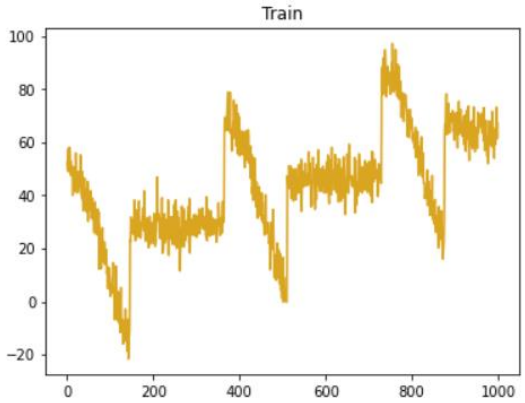
$$\frac{ds_{t'}}{ds_t} = \prod_{k=1}^{t'-t} \sigma'(v_{t+k})$$

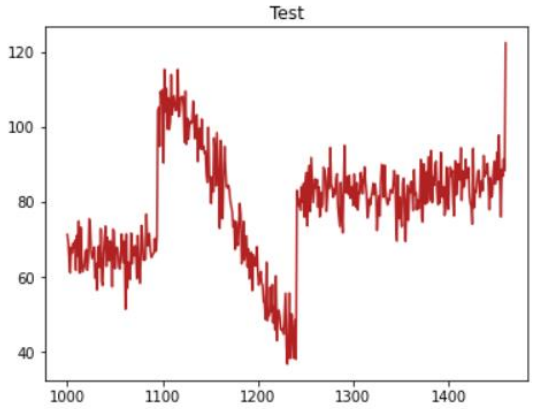
در اینجا v ورودی به forgot gate است و همین طور که می‌بینیم هیچ عاملی در آن وجود ندارد که به شکل توانی به صفر یا بی نهایت میل کند. در نتیجه حداقل یه مسیر وجود دارد که در آن vanishing gradient را نداشته باشیم.

۲- بخش عملی

۲-۱- سوال اول

ابتدا داده‌های train و test را به شکل زیر رسم کردم:





سپس داده‌های train وtest را آماده کرده و به شکل یک دیتاست درآوردم. در اینجا از sequence_length به اندازه ۴۰ استفاده کردم که به این معنی است که با داشتن ۴۰ داده قبلی داده بعد را پیش بینی خواهیم کرد:

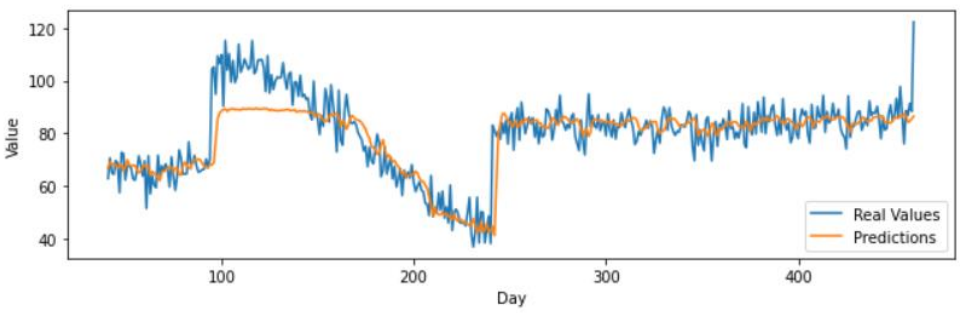
```
train = np.array(train_data.value).reshape(-1, 1)
test = np.array(test_data.value).reshape(-1, 1)
seq_length = 40

train_time_steps = train.shape[0]
test_time_steps = test.shape[0]

train_x = []
train_y = []
# make dataset - train & test
for i in range(train_time_steps - seq_length - 1 ):
  x = train[i : i+seq_length]
  y = train[i + seq_length + 1]
  train_x.append(x)
  train_y.append(y)

test_x = []
test_y = []
for i in range(test_time_steps - seq_length - 1 ):
  x = test[i : i+seq_length]
  y = test[i + seq_length + 1]
  test_x.append(x)
  test_y.append(y)
```

سپس به ساختن مدل پرداختم. مدل من دارای دولایه SimpleRNN است که اولی لایه ورودی است که ابعاد خروجی آن ۸۰ خواهد بود و لایه دوم نیز خروجی ۶۴ خواهد داشت. در این میان یک لایه normalization نیز داریم که کمک می‌کند خروجی‌ها به یک شکل استاندارد برسند و از مشکلاتی که به وجود می‌آید جلوگیری شود. لایه‌های drop out نیز جزء دیگری از مدل ما هستند که از overfitting جلوگیری خواهند کرد. در آخر نیز چندلایه Dense خواهیم داشت. از early stopping نیز استفاده می‌کنیم تا در صورتی بعد از مدتی بهبود خاصی در مقادیر validation loss ایجاد نشد فرآیند آموزش را پایان دهد. Loss را با استفاده از mean squared error محاسبه خواهیم کرد و از Adam به عنوان optimizer استفاده خواهیم کرد. هم چنین چون در صورت سوال گفته شده‌است

که مقادیر mean absolute error را نیز گزارش دهیم، این metric را نیز در هر epoch نشان خواهیم داد. نتایج پایانی train به صورت زیر خواهد بود:

```
Epoch 30/2000
15/15 [==============================] - 1s 37ms/step - loss: 116.3510 -
mean_absolute_error: 8.2197 - val_loss: 93.8877 - val_mean_absolute_error:
7.1515
Epoch 31/2000
15/15 [==============================] - 1s 35ms/step - loss: 113.3653 -
mean_absolute_error: 8.2165 - val_loss: 85.4557 - val_mean_absolute_error:
7.1097
Epoch 32/2000
15/15 [==============================] - 1s 37ms/step - loss: 116.1213 -
mean_absolute_error: 8.2438 - val_loss: 97.5164 - val_mean_absolute_error:
7.1460
Epoch 33/2000
15/15 [==============================] - 1s 37ms/step - loss: 110.9533 -
mean_absolute_error: 7.9699 - val_loss: 99.6764 - val_mean_absolute_error:
7.3801
Epoch 34/2000
15/15 [==============================] - 1s 36ms/step - loss: 105.9039 -
mean_absolute_error: 7.7391 - val_loss: 87.4318 - val_mean_absolute_error:
6.8239
Epoch 35/2000
15/15 [==============================] - 1s 36ms/step - loss: 117.7853 -
mean_absolute_error: 8.3600 - val_loss: 134.6922 -
val_mean_absolute_error: 9.2205
Epoch 36/2000
15/15 [==============================] - 1s 36ms/step - loss: 104.4905 -
mean_absolute_error: 7.8624 - val_loss: 144.7923 -
val_mean_absolute_error: 9.1436
Epoch 37/2000
15/15 [==============================] - 1s 36ms/step - loss: 98.2702 -
mean_absolute_error: 7.6278 - val_loss: 127.8453 -
val_mean_absolute_error: 8.2943
Epoch 38/2000
15/15 [==============================] - 1s 36ms/step - loss: 115.3200 -
mean_absolute_error: 8.0491 - val_loss: 69.8309 - val_mean_absolute_error:
                                                                    5.8166
```

در نهایت آخرین مقدار `val_mean_absolute_error` ما برابر 5.8166 خواهد بود. در نهایت مقادیر اصلی test و مقادیر پیش‌بینی شده آن را به شکل زیر رسم می‌کنیم:



۲-۲- سوال دوم

ابتدا لازم است همان طور که خود سوال به ما گفته‌است وزن‌های ورودی و hidden را تعریف کنیم و این کار را با استفاده از لایه linear و دادن ابعاد ورودی و خروجی انجام می‌دهیم. ابعاد خروجی و ورودی بسته به این که این پارامتر وزن با ورودی x یا hidden state باشد ابعادی شامل input_size و hidden_size خواهد داشت:

```
class MyGRUCell(nn.Module):
    def __init__(self, input_size, hidden_size):
        super(MyGRUCell, self).__init__()

        self.input_size = input_size
        self.hidden_size = hidden_size

        # ------------
        # FILL THIS IN
        # ------------
        ## Input linear layers
        self.Wiz = nn.Linear(input_size, hidden_size)
        self.Wir = nn.Linear(input_size, hidden_size)
        self.Wih = nn.Linear(input_size, hidden_size)

        ## Hidden linear layers
        self.Whz = nn.Linear(hidden_size, hidden_size)
        self.Whr = nn.Linear(hidden_size, hidden_size)
        self.Whh = nn.Linear(hidden_size, hidden_size)
```

متد forward را نیز با توجه به فرمول‌های داده‌شده در خود سوال کامل خواهیم کرد:

```
def forward(self, x, h_prev):
        """Forward pass of the GRU computation for one time step.

        Arguments
            x: batch_size x input_size
            h_prev: batch_size x hidden_size

        Returns:
            h_new: batch_size x hidden_size
        """

        # ------------
        # FILL THIS IN
        # ------------
        r = torch.sigmoid(self.Wir(x) + self.Whr(h_prev))
        z = torch.sigmoid(self.Wiz(x) + self.Whz(h_prev))
        g = torch.tanh(self.Wih(x) + r*self.Whh(h_prev))
        h = (1 - z)*g + z*h_prev
        return h
```

نتایج پایانی train برای RNN encoder که از GRU استفاده می‌کند به شکل زیر خواهد بود:

```
Epoch:  90 | Train loss: 0.778 | Val loss: 1.197 | Gen: eshay airway
onningsray-incepay-i isway orkway
Epoch:  91 | Train loss: 0.795 | Val loss: 1.158 | Gen: eshay airrway
onncay-acitentay isway orkay-awlay
Epoch:  92 | Train loss: 0.788 | Val loss: 1.143 | Gen: eshay airdway
oncisionlay-inentedw isway orkingway
Epoch:  93 | Train loss: 0.768 | Val loss: 1.105 | Gen: eshay airdway
oncisensay-inway-awl isway orkay-awlay
Epoch:  94 | Train loss: 0.759 | Val loss: 1.096 | Gen: eshay airdway
onnsiliscay isway orkay-awlay
Epoch:  95 | Train loss: 0.753 | Val loss: 1.086 | Gen: ethay airdway
onncistionway isway orkway
Epoch:  96 | Train loss: 0.750 | Val loss: 1.100 | Gen: eshay airdway
onncisensay-awelay isway orkay-aidedway
Epoch:  97 | Train loss: 0.748 | Val loss: 1.108 | Gen: ethay airdway
ondingspray isway orkway
Epoch:  98 | Train loss: 0.744 | Val loss: 1.074 | Gen: ethay airdway
oncisionlay isway orkay-awlay
Epoch:  99 | Train loss: 0.739 | Val loss: 1.125 | Gen: ethay airdway
                                                    onncistionway isway orkway
```

و نتیجه ترجمه جمله نمونه نیز به شکل زیر خواهد بود:

```
source:          the air conditioning is working
translated:      ethay airdway onncistionway isway orkway
```

سپس به سراغ پیاده‌سازی attention می‌رویم. تابع forward در additiveAttention را به صورت زیر پیاده‌سازی می‌کنیم:

```
batch_size = queries.size(0)
expanded_queries = queries.unsqueeze(1).expand_as(keys)
concat_inputs = torch.cat((expanded_queries, keys), 2)
unnormalized_attention = self.attention_network(concat_inputs)
attention_weights = self.softmax(unnormalized_attention)
context = torch.bmm(attention_weights.squeeze(2).unsqueeze(1), values)
```

ابتدا لازم است batch_size را به دست بیاوریم که می‌توانیم از پارامتر اول سایز queries آن را پیدا کنیم. سپس برای اینکه بتوانیم key و queries را کنار هم بگذاریم لازم است ابعاد queries متناسب با key باشد که نسخه expand شده از queries این قابلیت را خواهد داشت. برای به دست آوردن مقادیر attention نرمال نشده کافی است query و key که concat شده‌اند را به attention_network بدهیم. در نهایت برای نرمال کردن روی این مقادیر softmax خواهیم زد. بردار context را نیز از ضرب با batch_size مقدار attention و value به دست خواهیم آورد.

تابع forward از RNNAttentionDecoder نیز به صورت زیر خواهد بود:

```
for i in range(seq_len):
            # ------------
            # FILL THIS IN
            # ------------
            embed_current = embed[:, i, :]
            context, attention_weights = self.attention(h_prev, annotation
s, annotations)
            embed_and_context = torch.cat((context.squeeze(1), embed_curre
nt), 1)
            h_prev = self.rnn(embed_and_context, h_prev)
```

با دادن مقادیر query, key و value به attention که h_prev و annonations آن‌ها را تشکیل می‌دهند، وزن‌ها و context را به دست می‌آوریم و سپس نتیجه را با بخشی از embed که انتخاب کردیم concat می‌کنیم. در نهایت hidden state جدید را به دست می‌آوریم.

نتیجه آموزش با train به شکل زیر خواهد بود:

```
Epoch:  90 | Train loss: 0.012 | Val loss: 0.152 | Gen: ethay airway
onditioningcay isway orkingway
Epoch:  91 | Train loss: 0.010 | Val loss: 0.149 | Gen: ethay airway
onditioningcay isway orkingway
Epoch:  92 | Train loss: 0.009 | Val loss: 0.149 | Gen: ethay airway
onditioningcay isway orkingway
Epoch:  93 | Train loss: 0.008 | Val loss: 0.148 | Gen: ethay airway
onditioningcay isway orkingway
Epoch:  94 | Train loss: 0.007 | Val loss: 0.146 | Gen: ethay airway
onditioningcay isway orkingway
Epoch:  95 | Train loss: 0.007 | Val loss: 0.146 | Gen: ethay airway
onditioningcay isway orkingway
Epoch:  96 | Train loss: 0.006 | Val loss: 0.145 | Gen: ethay airway
onditioningcay isway orkingway
Epoch:  97 | Train loss: 0.006 | Val loss: 0.144 | Gen: ethay airway
onditioningcay isway orkingway
Epoch:  98 | Train loss: 0.005 | Val loss: 0.142 | Gen: ethay airway
onditioningcay isway orkingway
Epoch:  99 | Train loss: 0.005 | Val loss: 0.142 | Gen: ethay airway
                                        onditioningcay isway orkingway
```

نتیجه جمله ترجمه شده نیز به صورت زیر خواهد بود:

```
source:          the air conditioning is working
translated:      ethay airway onditioningcay isway orkingway
```

همان طور که از نتایج واضح است اگر train را با attention انجام دهیم به مقادیر کمتری از loss می‌رسیم و در مدت زمان کمتر نتیجه بهتری خواهیم داشت. و همین طور جمله ترجمه شده با attention درست‌تر است.