به نام خدا

نام دانشگاه: پردیس فارابی دانشگاه تهران

نام درس: پردازش زبان های طبیعی

نام تمرین: Machine Translation

دانیال فرهنگی ۲۲۰۷۹۸۰۷۸

ورودی ۹۸

تمرین Machine Translation:

آدرس ریپازیتوری گیتهاب: https://github.com/danfarh/machine-translation

.....

برای تمرین ترجمه ماشینی برای یادگیری بیشتر به سه روش مدل encoder-decoder و مدل با attention و با transformerها زدم.

در مدل encoder-decoder به دقت خوبی نرسیدم ولی نمونه کد آن در کامیت‌هایم موجود است.

در روش attention به دقت مطلوبی رسیدم که دو مدل فارسی به انگلیسی و انگلیسی به فارسی train کرده‌ام، که نتایج را در ادامه ارائه خواهم داد.

با روش transformer هم مدلی train کردم اما متاسفانه بعد از train گوگل‌کولب restart شد و نتوانستم وزن‌ها را سیو کنم تا تابعی برای evaluate آن بنویسم.

در ادامه مدل با روش attention را توضیح خواهم داد.

شرح:

در ابتدا فایل‌های انگلیسی و فارسی را می‌خوانیم و در یک فایل csv میریزیم و در گوگل درایو ذخیره میکنیم تا هر بار از آن دیتا ها را بخوانیم.

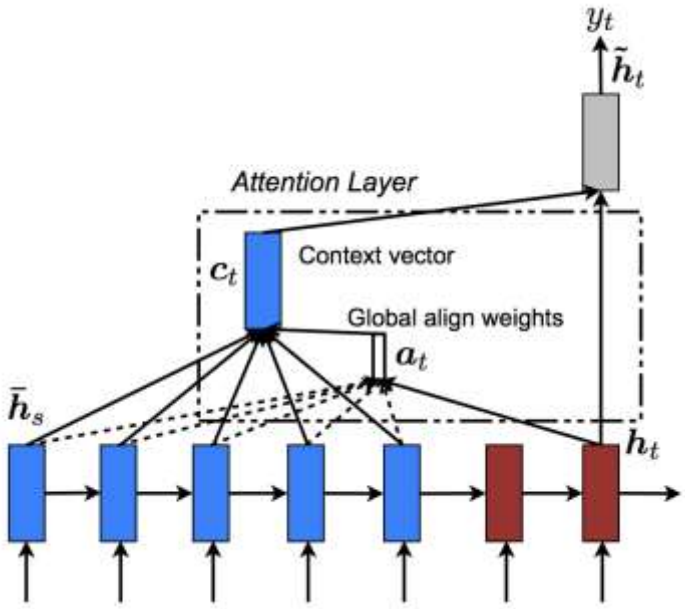
سپس end و start جملات را پیدا میکنیم و دوباره مثل تمرین text classification از توکنایزر کراس استفاده می‌کنیم و تبدیل به عدد میکنیم و padding میزنیم تا طول جملات یکسان شود.

بعد شبکه را که از یک encoder و یک decoder و یک لایه attention تشکیل شده است را طراحی میکنیم. در ادامه نحوه عملکرد شبکه و لایه attention را توضیح خواهم داد.

معماری شبکه:

مدل encoder-decoder ایراداتی داشت که بعد از آن مدل encoder-decoder با attention آمد.

در مدل attention دقیقا مشخص می‌شود که کدام کلمه در encoder تاثیر بیشتری در ترجمه یک کلمه در decoder دارد. پس برای هر کلمه در decoder، ضرب داخلی یا dot product آن را با تک تک کلمات encoder محاسبه می‌کند و یک softmax روی آن می‌زند و محتمل ترین کلمه که بیشترین تاثیر را در ترجمه آن کلمه در decoder دارد حساب می‌کند.



کدهای بخش های encoder و decoder و attention را در تصاویر زیر مشاهده میکنید.

```
class Encoder(keras.Model):
    def __init__(self, vocab_size, embedding_dim, enc_units, batch_size):
        super(Encoder, self).__init__()
        self.batch_size = batch_size
        self.enc_units = enc_units
        self.embedding = keras.layers.Embedding(vocab_size, embedding_dim)
        self.gru = keras.layers.GRU(self.enc_units, return_sequences=True, return_state=True)
    def call(self, x, hidden):
        x = self.embedding(x)
        output, state = self.gru(x, initial_state=hidden)
        return output, state
    def initilize_hidden_state(self):
        return tf.zeros((self.batch_size, self.enc_units))
```

```
class Attention(keras.layers.Layer):
    def __init__ (self, units):
        super(Attention, self).__init__()
        self.W1 = keras.layers.Dense(units)
        self.W2 = keras.layers.Dense(units)
        self.V = keras.layers.Dense(1)
    def call (self, query, values):
        hidden_with_time_axis = tf.expand_dims(query, 1)
        score = self.V(tf.nn.tanh(self.W1(values) + self.W2(hidden_with_time_axis)))
        atteion_weights = tf.nn.softmax(score, axis=1)
        context_vector = atteion_weights * values
        context_vector = tf.reduce_sum(context_vector, axis=1)
        return context_vector, atteion_weights
```

```python
class Decoder(keras.Model):
    def __init__ (self, vocab_size, embedding_dim, dec_units, batch_size):
        super(Decoder, self).__init__()
        self.batch_size = batch_size
        self.dec_units = dec_units
        self.embedding = keras.layers.Embedding(vocab_size, embedding_dim)
        self.gru = keras.layers.GRU(self.dec_units, return_sequences=True, return_state=True)
        self.fc = keras.layers.Dense(vocab_size)
        self.attention = Attention(self.dec_units)
    def call(self, x, hidden, enc_output):
        context_vector, attention_weights = self.attention(hidden, enc_output)
        x = self.embedding(x)
        x = tf.concat([tf.expand_dims(context_vector, 1), x], axis=-1)
        output, state = self.gru(x)
        output = tf.reshape(output, (-1, output.shape[2]))
        x = self.fc(output)
        return x, state, attention_weights
```

توضیح encoder و decoder و attention:

کدها دقیقا به همین صورت است که توضیح داده شد مثلا در encoder طبق تعریفی که از encoder-decoder می‌دانیم از یک لایه embedding و LSTM یا GRU تشکیل شده است و در این هم این لایه ها را قرار می‌دهیم. داده‌ها را از لایه Embedding رد میکنیم و سپس به GRU می‌دهیم.

لایه decoder هم دقیقا به همین صورت است فقط در ابتدا attention خروجی انکودر با hidden state دیکودر حساب میکنیم. و داده‌ها را مثل لایه انکودر از embedding رو میکنیم و خروجی لایه امبدینگ را با خروجی لایه attention کانکت میکنیم و سپس از یک GRU رد میکنیم و سپس از یک لایه dense که به اندازه تعداد لغات است و AF آن softmax است رد میکنیم، و خروجی decoder ما است.

بعد از ۱۰ ایپاک به لاس ۰.۳ رسیدم، اکنون به سراغ تست و ارزیابی مدل میرویم.

تست و ارزیابی مدل:

در تابع evaluate یک جمله به زبان مبدا از کاربر می‌گیریم و آن را به زبان مقصد ترجمه میکنیم.

در ابتدا جمله را preprocess میکنیم و به اول و آخر آن end و start اضافه میکنیم.

سپس تبدیل به عدد میکنیم و به آن padding می‌دهیم.

سپس hidden state را تعریف میکنیم برای این کار یک ماتریس صفر به اندازه یک در تعداد unit ها می‌سازیم.

سپس ورودی‌ها و hidden state را به encoder میدهیم و خروجی آن را دخیره میکنیم زیرا بعدا باید به decoder بدهیم.

سپس decoder هم ورودی‌هایش را می‌دهیم و آن هم می‌سازیم و خروجی decoder یک تابع احتمالاتی بود زیرا از یک softmax رد میکردیم، سپس ماکزیمم آن را پیدا میکنیم و کلمه مربوطه را برمیگردانیم، برای تمام کلمات decoder این کار را انجام می‌دهیم تا به end برسیم در نهایت جمله را return میکنیم.

```
def evaluate(sentence):
    sentence = preprocess_senetence(sentence)
    inputs = [input_lang_tokenizer.word_index[i] for i in sentence.split(' ')]
    inputs = keras.preprocessing.sequence.pad_sequences([inputs], maxlen=max_length_inp, padding='post')
    inputs = tf.convert_to_tensor(inputs)
    result = ''
    hidden = [tf.zeros((1, units))]
    enc_out, enc_hidden = encoder(inputs, hidden)
    dec_hidden = enc_hidden
    dec_input = tf.expand_dims([target_lang_tokenizer.word_index['<start>']], 0)
    for t in range(max_length_targ):
        predictions, dec_hidden, attention_weights = decoder(dec_input, dec_hidden, enc_out)
        attention_weights = tf.reshape(attention_weights, (-1, ))
        predicted_id = tf.argmax(predictions[0]).numpy()
        result += target_lang_tokenizer.index_word[predicted_id] + ' '
        if target_lang_tokenizer.index_word[predicted_id] == '<end>':
            return result, sentence
        dec_input = tf.expand_dims([predicted_id], 0)
    return result, sentence
```

نتایج مترجم انگلیسی به فارسی:

```
evaluate('dad')
```

```
('بابا <end> ', '<start> dad <end>')
```

```
evaluate('maybe its the wind')
```

```
('شاید این باد باشه <end> ', '<start> maybe its the wind <end>')
```

```
evaluate('stop please stop')
```

```
('صبرکنید صبرکنید صبرکنید <end> ', '<start> stop please stop <end>')
```

```
evaluate('you told me we made payments to hollander')
```

```
('تو به من گفته بودي كه ما به هلندر پرداخت كرديم <end> ',
 '<start> you told me we made payments to hollander <end>')
```

```
evaluate('i have great lessons today')
```

```
('امروز صبح دارم <end> ', '<start> i have great lessons today <end>')
```

```
evaluate('boss ')
```

```
('رئیس <end> ', '<start> boss <end>')
```

```
evaluate('zodiac')
```

```
('زودیاک <end> ', '<start> zodiac <end>')
```

```
evaluate('last night,  I saw a dream')
```

```
('دیشب خواب دیدم <end> ', '<start> last night i saw a dream <end>')
```

```
evaluate('would you have made anything different')
```

```
('تو تصمیمه دیگه اي داري <end> ',
 '<start> would you have made anything different <end>')
```

```
evaluate('you lied to me , dan')
```

```
('تو به من دروغ گفتي دن <end> ', '<start> you lied to me dan <end>')
```

```
evaluate('good morning , pinkerton')
```

```
('صبح بخیر پینکرتون <end> ', '<start> good morning pinkerton <end>')
```

```
evaluate('names charlie prince , i expect youve heard of me')
```

```
('من چارلي پرینسم فکر کنم از من شنیدم <end> ',
 '<start> names charlie prince i expect youve heard of me <end>')
```

```
evaluate('he is not what i expected')
```

```
('اون چیزي نیست که انتظارش را کردم <end> ',
 '<start> he is not what i expected <end>')
```

```
evaluate('cows are going to be fat')
```

```
('گاوها چاق خواهند بود <end> ', '<start> cows are going to be fat <end>')
```

نتایج مترجم فارسی به انگلیسی:

```
evaluate('بله')
```

```
('yes <end> ', '<start> بله <end>')
```

```
evaluate('شاید صداي باد باشه')
```

```
('maybe the wind or the wind <end> ', '<start> شاید صداي باد باشه <end>')
```

```
evaluate('سلام')
```

```
('hello <end> ', '<start> سلام <end>')
```

```
evaluate('ما قرار بود تصمیمانو با هم بگیریم')
```

```
('we were supposed to each other convincing <end> ',
 '<start> ما قرار بود تصمیمانو با هم بگیریم <end>')
```

```
evaluate('مگه تو تصمیمه دیگه اي میگرفتي')
```

```
('have you have made anything different <end> ',
 '<start> مگه تو تصمیمه دیگه اي میگرفتي <end>')
```

```
evaluate('شروع شد')
```

```
('its begun <end> ', '<start> شروع شد <end>')
```

```
evaluate('تو میخواي اونجا چیکار کني')
```

```
('what are you going to do <end> ', '<start> تو میخواي اونجا چیکار کني <end>')
```

```
evaluate('امروز درس بزرگی یاد گرفتم')
```

```
('i have great lesson has decided <end> ',
 '<start> امروز درس بزرگی یاد گرفتم <end>')
```

```
evaluate('کجا')
```

```
('where <end> ', '<start> کجا <end>')
```

```
evaluate('بردهها را بکشید')
```

```
('kill the slaves <end> ', '<start> بردهها را بکشید <end>')
```

```
evaluate('ما میریم دنبالش')
```

```
('were going to him <end> ', '<start> ما میریم دنبالش <end>')
```

```
evaluate('آن سریع ترین راه عبور است')
```

```
('that the quickest way of the quickest way <end> ',
 '<start> آن سریع ترین راه عبور است <end>')
```

```
evaluate('من نمیدونم براي چي باید بجنگم')
```

```
('i dont know what to fight <end> ',
 '<start> من نمیدونم براي چي باید بجنگم <end>')
```