Programming Assignment 4: Reading Comprehension

Written by

Ignacio Cases and Allen Nie

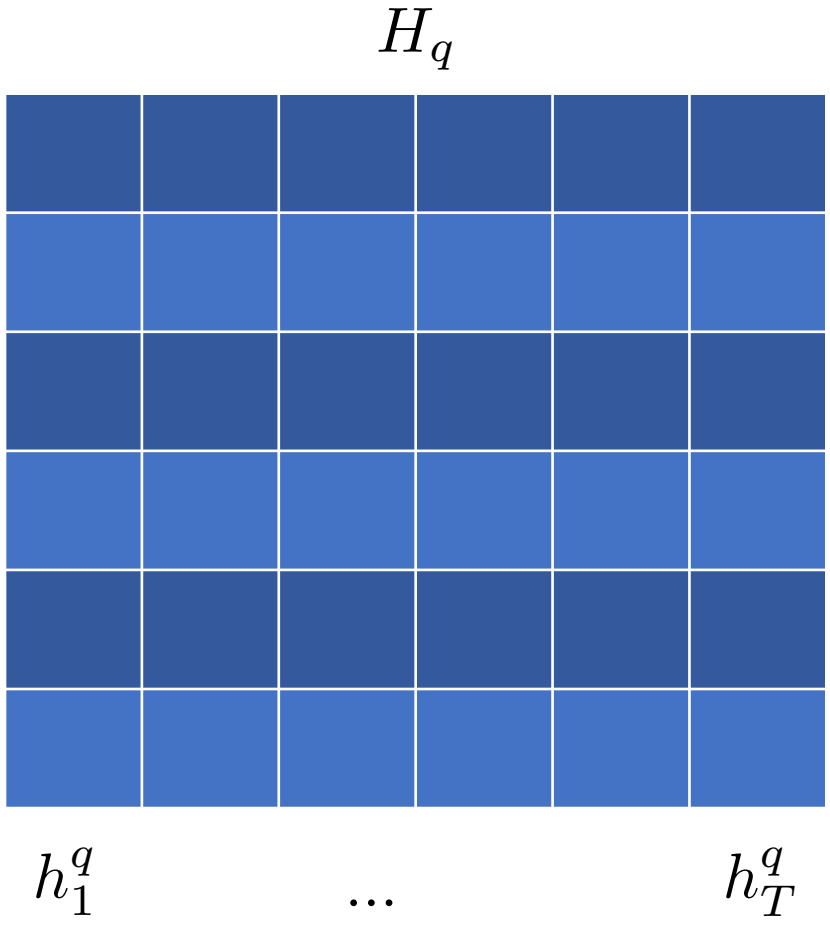
PA4 Task Overview

What was Maria Curie the first female recipient of?

One of the most famous people born in Warsaw was **Maria** Skłodowska-**Curie**, who achieved international recognition for her research on radioactivity and was the first **female recipient** of the **Nobel Prize**. Famous musicians include Władysław Szpilman and Frédéric Chopin. Though Chopin was born in the village of Żelazowa Wola, about 60 km (37 mi) from Warsaw, he moved to the city with his family when he was seven months old. Casimir Pulaski, a Polish general and hero of the American Revolutionary War, was born here in 1745.

The goal is to extract an answer from the paragraph. There are multiple ways to unpack this problem.

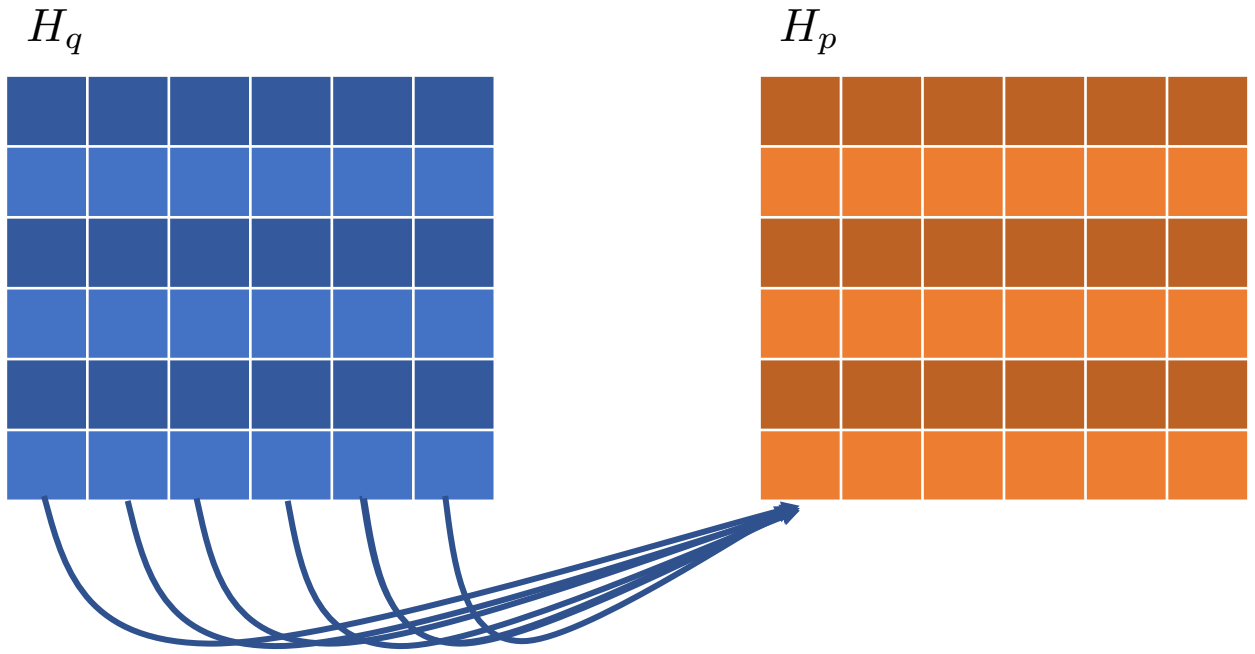
Baseline Model: Encode Question




```
from tensorflow.python.ops.nn import bidirectional_dynamic_rnn

(fw_out, bw_out), _ = bidirectional_dynamic_rnn(self.cell,
                                                self.cell,
                                                inp, srclen,
                                                scope=scope,
                                                time_major=True,
                                                dtype=dtypes.float32)
```

We can use a bidirectional RNN to encode the question! But can you use some other mechanism? Like a CNN?

Baseline Model: Encode Paragraph

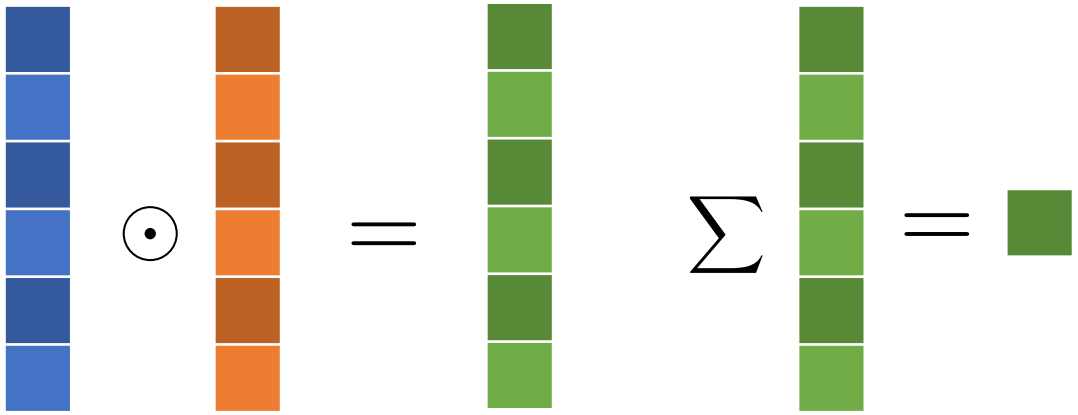



```
from tensorflow.python.ops.nn import dynamic_rnn

def encode_w_attn(self, inputs, masks, prev_states, scope="", reuse=False):
    self.attn_cell = GRUAttnCell(self.size, prev_states)
    with vs.variable_scope(scope, reuse):
        o, _ = dynamic_rnn(self.attn_cell, inputs)
```

The context paragraphs in SQuAD are pretty long. Is sequence to sequence attention the best attention mechanism for this task? It is slow, and complicated to implement!

Baseline Model: Efficient Attention

$$h_t^T h_s = \sum_i (h_t \odot h_s)_i$$



In practice, we need to be able to compute the score between all source hidden states and one target hidden state. So we have to rethink about dot product as two operations:

1. Hadamard product on two vectors
2. Reduce-sum on the resulting vector

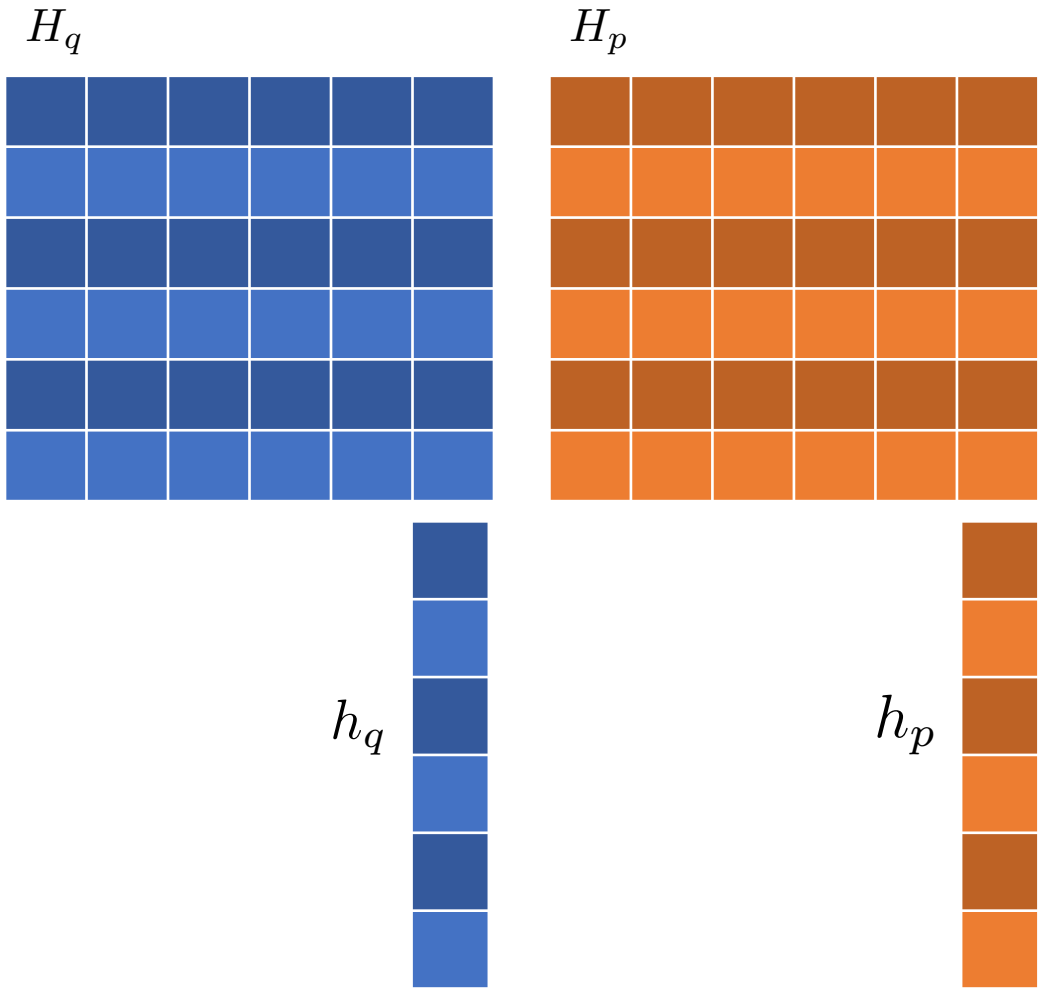
Baseline Model: Sequence Attention

```
class GRUAttnCell(rnn_cell.GRUCell):
    def __init__(self, num_units, encoder_output, scope=None):
        self.hs = encoder_output
        super(GRUCellAttn, self).__init__(num_units)

    def __call__(self, inputs, state, scope=None):
        gru_out, gru_state = super(GRUCellAttn, self).__call__(inputs, state, scope)
        with vs.variable_scope(scope or type(self).__name__):
            with vs.variable_scope("Attn"):
                ht = rnn_cell._linear(gru_out, self._num_units, True, 1.0)
                ht = tf.expand_dims(ht, axis=1)
            scores = tf.reduce_sum(self.hs * ht, reduction_indices=2, keep_dims=True)
            context = tf.reduce_sum(self.hs * scores, reduction_indices=1)
            with vs.variable_scope("AttnConcat"):
                out = tf.nn.relu(rnn_cell._linear([context, gru_out], self._num_units, True, 1.0))

        return (out, out)
```

But you need to ask: is this really the best way to approach this problem? Paragraph is LONG, is incorporating attention at each timestep really a good idea?

This breakdown of dot product operation makes it much easier to implement a sequence to sequence attention when we can’t loop over all hidden states from the source.

Baseline Model: What do we have so far?



We have four representations so far, how can we combine these to create a perfect representation to decode?

It's a great time to start reading the papers listed in the handout!

Baseline Model: Decode

$$a_s = W_a h_p + \tilde{W}_a h_q + b_a$$

$$a_e = W_e h_p + \tilde{W}_e h_q + b_e$$

```
with vs.scope("answer_start"):
    a_s = rnn_cell._linear([h_q, h_p], output_size=self.output_size)
with vs.scope("answer_end"):
    a_e = rnn_cell._linear([h_q, h_p], output_size=self.output_size)
```

The simplest way to build up a naïve decoder is to have just trust the last hidden state of paragraph and question encoder will contain all information needed by the decoder. They represent the un-normalized logits for the start and end index of the answer span.

Baseline Model: Loss

```
from tensorflow.python.ops.nn \
import sparse_softmax_cross_entropy_with_logits as ssce

with vs.variable_scope("loss"):
    l1 = ssce(self.a_s, self.start_answer)
    l2 = ssce(self.a_e, self.end_answer)
    self.loss = l1 + l2
```

Note in your cross-entropy loss, label is a one-hot representation!

You calculate the loss based on how you formulated your objective. If you are doing a sequence 2-class prediction (like PA3), then your loss will look like sequence loss. If you only predict 2 integers, your loss will look like what we have on the left side.

Baseline Model: Evaluation

```
from evaluate import f1_score, exact_match_score

answer = p[a_s, a_e + 1]
true_answer = p[true_s, true_e + 1]
f1_score(answer, true_answer)
exact_match_score(answer, true_answer)
```

Both functions only take one example at a time, so if you are doing mini-batch based training, make sure you are not passing in multiple examples!

You must import EM and F1 evaluation from evaluate.py file. You need to extract your answer from the paragraph based on the start and end index, and call these two functions.

Practical Tips

1. In many implementations, you don't need sequence to sequence attention (co-attention, BiDAF, multi-perspective matching).
2. Have a baseline model working by the end of this week, and start implementing advanced modules as soon as possible!
3. Evaluate F1 and EM score at the end of each epoch, since each takes a long time to train.
4. Simply implementing a paper's model is not enough! Your grade will largely depend on how much you can explore!
5. Good luck!