2.4

Monday, March 4, 2024 6:14 PM

1 a)

```python
# 4.1.a
def generate_token_embeddings(s):
    id_map = {}
    s_map = {}
    s_list = s.split(" ")

    # sort by alphabetical order
    s_list_sorted = list(dict.fromkeys(sorted(s.split(" "))))
    for i, s in enumerate(s_list_sorted):
        id_map[i] = s
        s_map[s] = i + 1

    T = len(s_list)
    tensor = []

    for s in s_list:
        tensor.append(s_map[s])

    tensor = torch.as_tensor(tensor)

    # create embedding with torch
    embedding = torch.nn.Embedding(T + 1, 16)

    output_embedding = embedding(tensor)
    print(f"Embedding of word: {output_embedding}")
    print(f"Shape of embedding: {output_embedding.shape}")

    return output_embedding
```

PROBLEMS DEBUG CONSOLE OUTPUT TERMINAL PORTS

```
jackiezhu@Jackies-MacBook-Air assignment_2 % python3 4_1_a.py
Embedding of word: tensor([[ 0.5449,  1.9281, -0.6208,  1.1076,  0.1331,  0.1472, -1.0472,  1.1448,
          0.0128, -0.2665,  0.9189, -2.2912,  0.6965,  1.0309, -1.5584, -1.3782],
        [ 0.7076,  1.1984, -1.5267, -0.2254,  0.0264,  0.6631, -0.3922, -0.2221,
          0.5445, -0.3446, -0.8801, -0.5930, -0.0471,  2.0033, -0.9231, -0.3465],
        [ 0.2683,  0.8987,  0.1029,  0.8214,  2.5925,  1.0087, -0.5200,  0.1927,
          0.6977,  0.4074,  0.9424, -1.7282,  1.1603, -1.2132, -1.4624,  0.2962],
        [ 0.3450,  0.2975,  0.9669, -1.4826,  1.0491, -0.6034, -0.2254,  0.1509,
          0.4969,  0.3232, -0.4563, -0.1595,  0.7242, -0.9060, -0.4738,  0.6987],
        [ 0.0381,  0.1548, -0.2557,  0.0346,  0.3751, -0.3636, -2.5965,  0.5740,
          0.3037,  0.4757, -0.0147, -0.4754, -1.6769,  1.5561,  0.3053, -3.1729],
        [ 0.1769, -0.9805, -2.3564, -0.0269, -1.3488,  1.0367,  0.7383,  0.9799,
         -0.0074, -0.1416, -0.7462,  0.3935,  0.6559, -0.5849,  0.9106, -1.3100],
        [-0.7573, -0.1884,  0.2287,  0.2734,  0.5684,  0.4376, -0.0679,  0.0269,
          1.3526, -0.3357,  1.3992, -1.7147, -1.6434, -1.1107,  0.2154, -0.5452]],
       grad_fn=<EmbeddingBackward0>)
Shape of embedding: torch.Size([7, 16])
```

b)

```python
#4.1.b
def generate_contex(x, d_q, d_k, d_v):
    embeddings_np = x.detach().numpy()
    T = embeddings_np.shape[1]

    # generate q k and v
    q = torch.rand(d_q, T)
    k = torch.rand(d_k, T)
    v = torch.rand(d_v, T)

    q_x = torch.matmul(x, q.transpose(0, 1))
    k_x = torch.matmul(x, k.transpose(0, 1))
    v_x = torch.matmul(x, v.transpose(0, 1))

    w_x = torch.matmul(q_x, k_x.transpose(0, 1))

    alpha_x = torch.nn.functional.softmax(torch.div(w_x, math.sqrt(d_k)), dim=-1)

    context = torch.matmul(alpha_x, v_x)

    print(f"Context: {context}")
    print(f"Shape of single-head context: {context.shape}")

    return context
```

PROBLEMS DEBUG CONSOLE OUTPUT TERMINAL PORTS

```
Context: tensor([[ 0.0812, -0.3247,  0.5198,  0.2025, -0.2158, -0.6486, -0.8076, -0.8843,
         -0.8336, -1.0199, -0.9299, -0.3060, -0.4601,  0.1508, -0.2933, -0.8579,
         -0.6508, -0.4855, -0.6936,  0.2324,  0.1256, -1.0851, -0.4847, -0.7681,
         -0.1082,  0.0191, -0.3791,  0.1280],
        [-0.5022, -2.1867,  0.9639, -1.3133, -2.3291, -1.0358, -3.0479, -2.8921,
         -1.9910, -2.1619, -2.4664, -2.5673, -2.7657,  0.0195, -2.4010, -2.7704,
         -2.7644, -0.8755, -3.2602, -1.4629, -0.5152, -2.9044, -2.1984, -3.8912,
         -0.1395, -1.7397, -1.9709, -0.8409],
        [ 0.7886,  2.5463,  3.0208,  3.1826,  3.0737,  1.2498,  0.9576,  2.5943,
          0.6807,  1.6307,  2.5936,  0.6413,  3.2482,  2.8249,  3.0690,  2.2376,
          0.8775,  0.6318,  2.3635,  3.7491,  3.4594,  1.1780,  3.7918,  2.5255,
          0.3774,  2.2261,  1.8482,  2.7633],
        [ 0.7253,  2.3600,  2.7603,  3.0093,  2.7432,  1.0952,  0.8134,  2.2992,
          0.5193,  1.4080,  2.3549,  0.5579,  2.9768,  2.5822,  2.7740,  1.9851,
          0.7558,  0.5220,  2.1336,  3.5621,  3.1972,  1.0496,  3.5374,  2.2959,
          0.3137,  2.0543,  1.6355,  2.5244],
        [-0.9790, -2.9593,  1.7395, -0.7457, -3.4785, -1.0556, -3.6276, -4.1414,
         -2.8210, -3.1097, -2.4564, -4.1427, -2.8034,  0.1952, -4.0025, -3.3231,
         -4.3137, -1.5543, -4.2333, -1.0514, -0.7028, -3.3224, -1.9985, -5.4794,
         -0.5237, -2.5203, -3.2434, -0.7701],
        [-0.9285, -2.8903,  1.6670, -0.8111, -3.3701, -1.0538, -3.5825, -4.0237,
         -2.7463, -3.0058, -2.4645, -3.9782, -2.8228,  0.1778, -3.8547, -3.2825,
         -4.1494, -1.4744, -4.1455, -1.1110, -0.6764, -3.2933, -2.0217, -5.3296,
         -0.4640, -2.4478, -3.1178, -0.7779],
        [-0.9444, -2.9077,  1.6862, -0.7893, -3.3998, -1.0544, -3.5914, -4.0557,
         -2.7652, -3.0397, -2.4599, -4.0293, -2.8090,  0.1829, -3.8936, -3.2893,
         -4.2013, -1.5022, -4.1681, -1.0872, -0.6871, -3.2975, -2.0142, -5.3706,
         -0.4894, -2.4673, -3.1542, -0.7757]], grad_fn=<MmBackward0>)
Shape of single-head context: torch.Size([7, 28])
```

2.

```python
# 4.2
def generate_multi_contex(embeddings, d_q, d_k, d_v, h):

    c_multi = []

    for _ in range(h):
        c_multi.append(generate_contex(embeddings, d_q, d_k, d_v))

    context = torch.stack(c_multi, dim=0)
    print(f"Multi-head context: {context}")
    print(f"Shape of multi-head context: {context.shape}")
    return context

s = "Attention is all you need for now"

embeddings = generate_token_embeddings(s)
#generate_contex(embeddings, 24, 24, 28)
generate_multi_contex(embeddings, 24, 24, 28, 5)
```

Shape of multi-head context: [5, 7, 28]

```
Multi-head context: tensor([[[ 8.1160e-02, -3.2466e-01,  5.1980e-01,  2.0247e-01, -2.1583e-01,
          -6.4858e-01, -8.0764e-01, -8.8435e-01, -8.3355e-01, -1.0199e+00,
          -9.2990e-01, -3.0602e-01, -4.6006e-01,  1.5085e-01, -2.9333e-01,
          -8.5788e-01, -6.5084e-01, -4.8553e-01, -6.9364e-01,  2.3240e-01,
           1.2556e-01, -1.0851e+00, -4.8471e-01, -7.6808e-01, -1.0818e-01,
           1.9082e-02, -3.7908e-01,  1.2800e-01],
         [-5.0219e-01, -2.1867e+00,  9.6390e-01, -1.3133e+00, -2.3291e+00,
          -1.0358e+00, -3.0479e+00, -2.8921e+00, -1.9910e+00, -2.1619e+00,
          -2.4664e+00, -2.5673e+00, -2.7657e+00,  1.9484e-02, -2.4010e+00,
          -2.7704e+00, -2.7644e+00, -8.7555e-01, -3.2602e+00, -1.4629e+00,
          -5.1523e-01, -2.9044e+00, -2.1984e+00, -3.8912e+00, -1.3953e-01,
          -1.7397e+00, -1.9709e+00, -8.4087e-01],
         [ 7.8859e-01,  2.5463e+00,  3.0208e+00,  3.1826e+00,  3.0737e+00,
           1.2498e+00,  9.5759e-01,  2.5943e+00,  6.8065e-01,  1.6307e+00,
           2.5936e+00,  6.4126e-01,  3.2482e+00,  2.8249e+00,  3.0690e+00,
           2.2376e+00,  8.7753e-01,  6.3184e-01,  2.3635e+00,  3.7491e+00,
           3.4594e+00,  1.1780e+00,  3.7918e+00,  2.5255e+00,  3.7740e-01,
           2.2261e+00,  1.8482e+00,  2.7633e+00],
         [ 7.2532e-01,  2.3600e+00,  2.7603e+00,  3.0093e+00,  2.7432e+00,
           1.0952e+00,  8.1340e-01,  2.2992e+00,  5.1928e-01,  1.4080e+00,
           2.3549e+00,  5.5792e-01,  2.9768e+00,  2.5822e+00,  2.7740e+00,
           1.9851e+00,  7.5581e-01,  5.2204e-01,  2.1336e+00,  3.5621e+00,
           3.1972e+00,  1.0496e+00,  3.5374e+00,  2.2959e+00,  3.1369e-01,
           2.0543e+00,  1.6355e+00,  2.5244e+00],
         [-9.7899e-01, -2.9593e+00,  1.7395e+00, -7.4567e-01, -3.4785e+00,
          -1.0556e+00, -3.6276e+00, -4.1414e+00, -2.8210e+00, -3.1097e+00,
          -2.4564e+00, -4.1427e+00, -2.8034e+00,  1.9525e-01, -4.0025e+00,
          -3.3231e+00, -4.3137e+00, -1.5543e+00, -4.2333e+00, -1.0514e+00,
          -7.0284e-01, -3.3224e+00, -1.9985e+00, -5.4794e+00, -5.2371e-01,
          -2.5203e+00, -3.2434e+00, -7.7006e-01],
         [-9.2847e-01, -2.8903e+00,  1.6670e+00, -8.1110e-01, -3.3701e+00,
          -1.0538e+00, -3.5825e+00, -4.0237e+00, -2.7463e+00, -3.0058e+00,
          -2.4645e+00, -3.9782e+00, -2.8228e+00,  1.7784e-01, -3.8547e+00,
          -3.2825e+00, -4.1494e+00, -1.4744e+00, -4.1455e+00, -1.1110e+00,
          -6.7644e-01, -3.2933e+00, -2.0217e+00, -5.3296e+00, -4.6396e-01,
          -2.4478e+00, -3.1178e+00, -7.7786e-01],
         [-9.4438e-01, -2.9077e+00,  1.6862e+00, -7.8931e-01, -3.3998e+00,
          -1.0544e+00, -3.5914e+00, -4.0557e+00, -2.7652e+00, -3.0397e+00,
          -2.4599e+00, -4.0293e+00, -2.8090e+00,  1.8285e-01, -3.8936e+00,
          -3.2893e+00, -4.2013e+00, -1.5022e+00, -4.1681e+00, -1.0872e+00,
          -6.8713e-01, -3.2975e+00, -2.0142e+00, -5.3706e+00, -4.8938e-01,
          -2.4673e+00, -3.1542e+00, -7.7567e-01]],

        [[ 1.9623e+00,  3.4931e+00,  2.9589e+00,  3.2453e+00,  3.3266e+00,
[illegible]
          -2.8356e+00, -5.8583e+00, -3.1984e+00, -1.6557e+00, -3.8138e+00,
          -3.5578e+00, -2.3392e+00, -2.6536e+00],
         [-4.7745e-01, -1.1123e+00, -1.2607e+00, -2.6344e+00, -9.5075e-01,
          -2.4323e+00, -2.3545e+00, -2.0583e+00, -2.5740e+00,  5.8839e-01,
          -1.1838e+00, -1.6964e+00,  7.3040e-03, -2.1148e+00, -2.6727e+00,
          -2.7397e+00, -1.0387e+00, -1.9044e+00, -2.1886e+00, -8.1066e-01,
          -1.9258e+00, -3.7749e+00, -1.2204e+00, -2.1710e-01, -2.3705e+00,
          -1.6775e+00, -1.2805e+00, -2.0283e+00],
         [-1.0245e+00, -1.8130e+00, -3.6835e+00, -2.2356e+00, -1.0703e-01,
          -9.7210e-01, -4.0641e+00, -3.7946e+00, -4.3335e+00, -2.3424e-01,
          -2.9556e+00, -1.7620e+00, -3.7528e-01, -3.8832e+00, -3.8389e+00,
          -2.8844e+00, -1.6956e+00, -2.7736e+00, -3.0685e+00, -1.4087e+00,
          -2.8346e+00, -5.8560e+00, -3.1973e+00, -1.6546e+00, -3.8131e+00,
          -3.5565e+00, -2.3389e+00, -2.6535e+00]]], grad_fn=<StackBackward0>)
Shape of multi-head context: torch.Size([5, 7, 28])
```

3. Multi-head attention only focuses on a single input sequence while cross-attention extends the idea of attention to multiple inputs. Thus, intuitively, a benefit of cross-attention would be that it is able to capture the contextual relationship between multiple input sequences and understanding how they interact. An example of a use-case would be for translation, where cross-attention can be used to align the words of the source and translatted language to ensure that not only the words are translated correctly, but the grammar and context of the translated sentence is also correct.

Further use-cases of cross-attention:

Question-Answer models
Text-to-speech
Code-generation
Image captioning