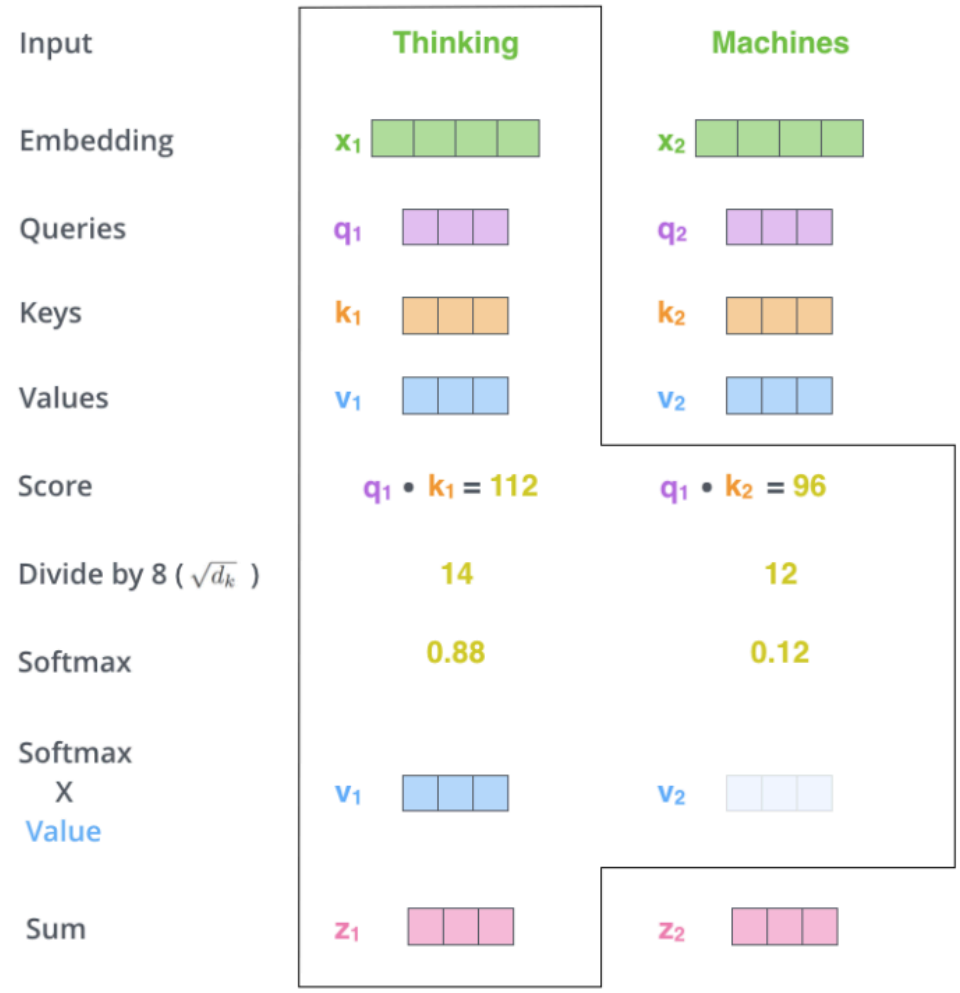
Self-attention: exercise

- “Computers are thinking machines”
- Compute z for machines
- Q = K = V = $\begin{bmatrix} 0.2 & 0.8 \\ -0.2 & 0.5 \\ -0.3 & -0.4 \\ 0.7 & 0.7 \end{bmatrix}$
- Computers = [1 0 0 0], are = [0 1 0 0], thinking = [0 0 1 0], machines = [0 0 0 1]
- Softmax

z = [0.24 0.55]



Python source code

```
import torch
from torch.nn.functional import softmax

x = [
    [1, 0, 0, 0], # Input 1
    [0, 1, 0, 0], # Input 2
    [0, 0, 1, 0], # Input 3
    [0, 0, 0, 1]  # Input 4
]

w_key = np.array([
    [0.2, 0.8],
    [-0.2, 0.5],
    [-0.3, -0.4],
    [0.7, 0.7]
])
w_query = np.array([
    [0.2, 0.8],
    [-0.2, 0.5],
    [-0.3, -0.4],
    [0.7, 0.7]
])
w_value = np.array([
    [0.2, 0.8],
    [-0.2, 0.5],
    [-0.3, -0.4],
    [0.7, 0.7]
])

x = torch.tensor(x, dtype=torch.float32)
w_key = torch.tensor(w_key, dtype=torch.float32)
w_query = torch.tensor(w_query, dtype=torch.float32)
w_value = torch.tensor(w_value, dtype=torch.float32)

keys = x @ w_key
querys = x @ w_query
values = x @ w_value
```

```
attn_scores = querys @ keys.T

attn_scores_softmax = softmax(attn_scores, dim=-1)

weighted_values = values[:,None] * attn_scores_softmax.T[:,:,None]

outputs = weighted_values.sum(dim=0)

print(outputs)
```

```
tensor([[0.2150, 0.5622],
        [0.1277, 0.4783],
        [0.0019, 0.2600],
        [0.2816, 0.5898]])
```

Therefore, the attention score of Machines (4th input) is [0.28, 0.58]