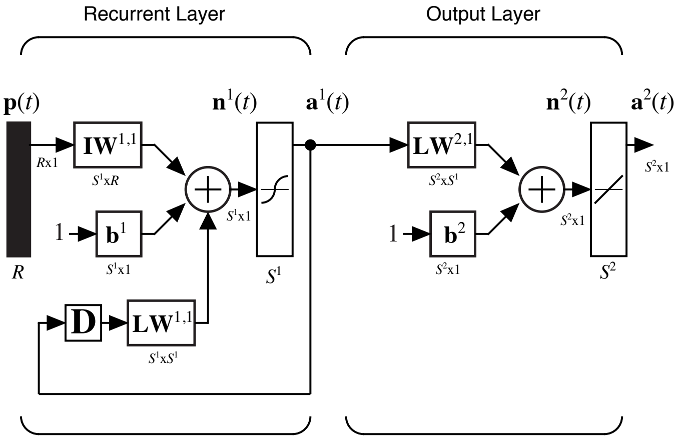
Sequence to Sequence Networks and Attention

Deep Learning

Sequence to sequence (Seq2Seq) networks

- Seq2Seq is a family of networks.
- A sequence is input to the network, which then produces an output sequence.
- Originally developed for machine translation – input sequence is a sentence in language A and the output sequence is the sentence in language B.
- The network has two parts:
  - Encoder processes the input sequence and produces context.
  - Decoder generates the output sequence from the context.
- Many encoder and decoder structures have been used.
  - Originally LSTM networks used for encoder & decoder
  - Then attention added between encoder & decoder
  - Recurrent connections removed – attention-only transformer

RNN

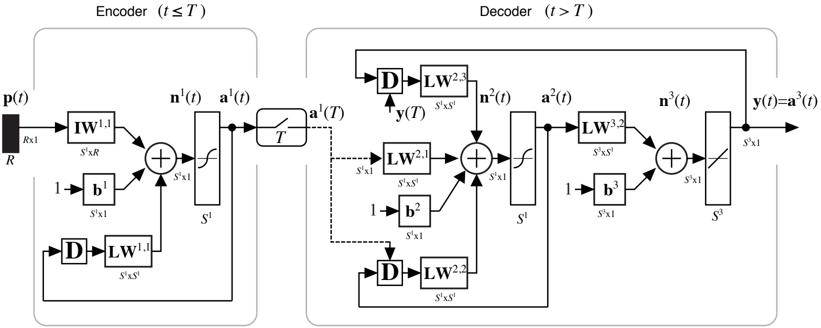


We will start by using RNNs for encoder and decoder.

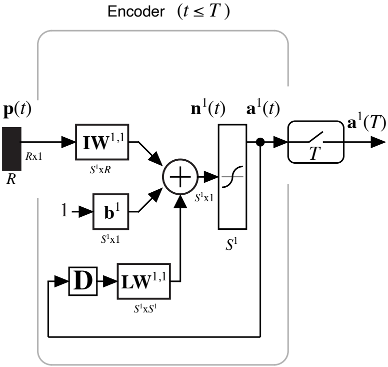
$$\mathbf{a}^1(t) = \mathbf{tansig}\left(\mathbf{IW}^{1,1}\mathbf{p}(t) + \mathbf{LW}^{1,1}\mathbf{a}^1(t-1) + \mathbf{b}^1\right)$$

$$\mathbf{a}^2(t) = \mathbf{LW}^{2,1}\mathbf{a}^1(t) + \mathbf{b}^2$$

Sequence-to-sequence (Seq2Seq) network

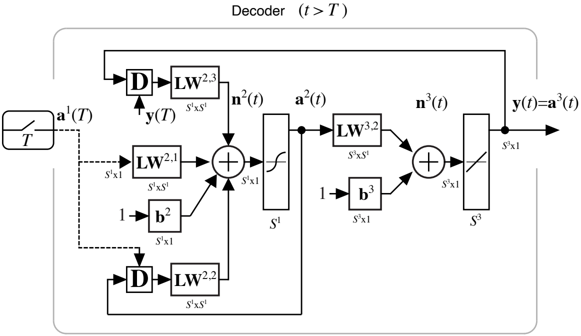


Seq2Seq encoder



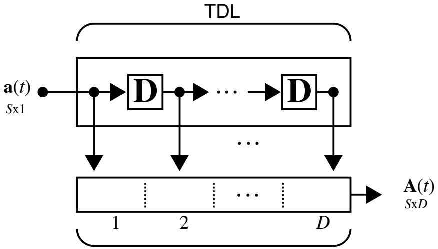
- The encoder takes the input sequence, which has T time points (words, or word fragments), and produces a layer output $\mathbf{a}^1(t)$.
- This output is sampled at the final time point $t = T$.
- The output of the sampler is fixed and does not change with time.

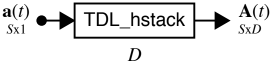
Seq2Seq decoder



- The final encoder state initializes the decoder state.
- It is also a constant input to the first layer of the decoder.
- This final encoder state is referred to as the context.
- The context summarizes the input sequence.

TDL horizontal stack





$$\mathbf{A}^1(t) = \left[\mathbf{a}^1(t) \quad \mathbf{a}^1(t-1) \quad \cdots \quad \mathbf{a}^1(t-D+1)\right]$$

- The encoder recurrent layer has infinite memory.
- It may not store information most efficiently.
- For more flexibility, add a TDL of previous encoder states.

Attention

- Combine the $\mathbf{A}^1(t)$ vectors to form a new context.
- How much should each previous encoder state contribute?
- We want the context to change at each time step.
- When translating an English sentence to French, each French word may depend on different combinations of English words.
- How correlated is the previous decoder state to each previous encoder state.
- Find the inner product between the previous decoder state and all previous encoder states.

$$\mathbf{n}^4(t) = \left[\mathbf{A}^1(T)\right]^T \mathbf{a}^2(t-1)$$

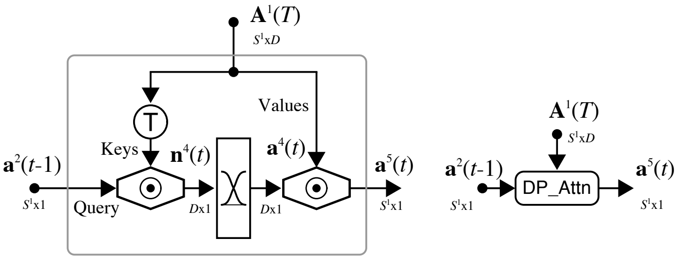
- Normalize with the `softmax`

$$\mathbf{a}^4(t) = \mathbf{softmax}\left(\mathbf{n}^4(t)\right)$$

- Combine according to the amount of correlation.

$$\mathbf{a}^5(t) = \mathbf{A}^1(T)\mathbf{a}^4(t)$$

Dot product attention



- Attention consists of operations between a query vector and sets of key and value pairs.
- The query is compared to each of the keys to determine how much of each corresponding value to include in the result.
- In our case, the query is the previous decoder state $\mathbf{a}^2(t-1)$, the keys are all previous encoder states, and the values are also the previous encoder states.

Database meaning of query, keys and values

- Query: Floy

Rank
2nd
1st
3rd
...

Last Name (Key)	Age (Value)
Flowers	35
Floyd	24
Fly	57
...	...

- Compare the Query to all of the Keys in the database.
- Return the Value associated with the closest matching Key.

Example of attention

$$\mathbf{A}^1(T) = \begin{bmatrix} 0.7 & -0.6 & 0.2 & -0.8 \\ 0.3 & 0.5 & -0.7 & -0.5 \\ -0.6 & 0.1 & 0.6 & 0.4 \end{bmatrix}$$ **Keys/Values**

$$\mathbf{a}^2(t-1) = \begin{bmatrix} 0.4 \\ 0.1 \\ -0.3 \end{bmatrix}$$ **Query**

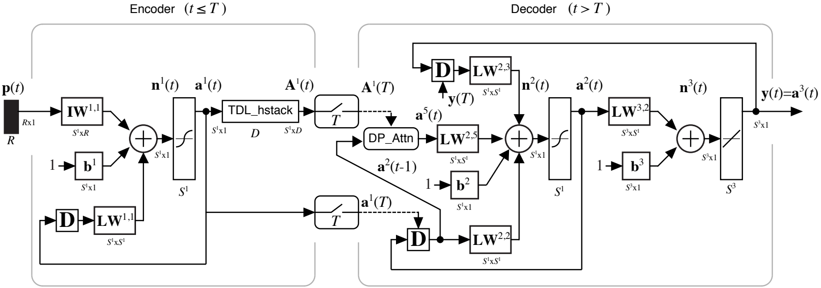
$$\mathbf{n}^4(t) = \left[\mathbf{A}^1(T)\right]^T \mathbf{a}^2(t-1) = \begin{bmatrix} 0.49 \\ -0.22 \\ -0.17 \\ -0.49 \end{bmatrix}$$

$$\mathbf{a}^4(t) = \mathsf{softmax}\left(\mathbf{n}^4(t)\right) = \begin{bmatrix} 0.420 \\ 0.206 \\ 0.217 \\ 0.157 \end{bmatrix}$$ **Attention Weights**

Using attention weights to combine values

$$\begin{aligned}
\mathbf{a}^5(t) &= \mathbf{A}^1(T)\mathbf{a}^4(t) \\
&= 0.420\begin{bmatrix} 0.7 \\ 0.3 \\ -0.6 \end{bmatrix} + 0.206\begin{bmatrix} -0.6 \\ 0.5 \\ 0.1 \end{bmatrix} + 0.217\begin{bmatrix} 0.2 \\ -0.7 \\ 0.6 \end{bmatrix} + 0.157\begin{bmatrix} -0.8 \\ -0.5 \\ 0.4 \end{bmatrix} \\
&= \begin{bmatrix} 0.087 \\ -0.002 \\ -0.038 \end{bmatrix}
\end{aligned}$$

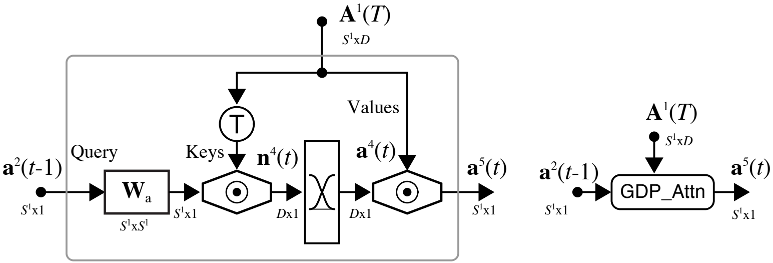
Seq2Seq network with attention



General dot product attention

- For vectors, $\mathbf{x}$ and $\mathbf{y}$, the standard inner product is $\mathbf{x}^T\mathbf{y}$.
- Another common inner product is the weighted dot product $\mathbf{x}^T\mathbf{W}\mathbf{y}$
- The symmetric, positive definite weighting matrix $\mathbf{W}$ is used to emphasize certain components of the dot product.
- This concept can be easily incorporated into the attention mechanism.

$$\mathbf{n}^4(t) = \left[\mathbf{A}^1(T)\right]^T \mathbf{W}_a\mathbf{a}^2(t-1)$$



Concatenation attention

- Attention – how much should each encoder state contribute?
- Instead of inner product between query $(\mathbf{a}^2(t-1))$ and keys (encoder states), use a general functional relationship.
- Concatenate query with the keys and pass the result through a mulitlayer subnetwork.
- Vertically stack, or concatenate, the query $\mathbf{a}^2(t-1)$ with the previous encoder states $\mathbf{A}^1(T)$.

$$
\begin{aligned}
\bar{\mathbf{A}}^{1,2}(t) &= \begin{bmatrix} \mathbf{A}^1(T) \\ \begin{bmatrix} \mathbf{a}^2(t-1) & \mathbf{a}^2(t-1) & \cdots & \mathbf{a}^2(t-1) \end{bmatrix} \end{bmatrix} \\
&= \begin{bmatrix} \begin{bmatrix} \mathbf{a}^1(T) \\ \mathbf{a}^2(t-1) \end{bmatrix} & \begin{bmatrix} \mathbf{a}^1(T-1) \\ \mathbf{a}^2(t-1) \end{bmatrix} & \cdots & \begin{bmatrix} \mathbf{a}^1(T-D+1) \\ \mathbf{a}^2(t-1) \end{bmatrix} \end{bmatrix}
\end{aligned}
$$

Concatenation attention implementation

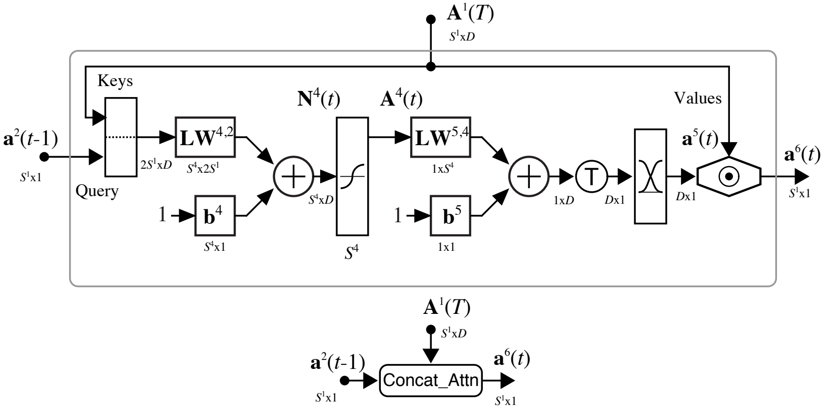
- The matrix $\bar{\mathbf{A}}^{1,2}(t)$ is passed into a two layer network.

$$\begin{aligned} \mathbf{A}^4(t) &= \mathbf{tansig}\left(\mathbf{LW}^{4,2}\bar{\mathbf{A}}^{1,2}(t) + \mathbf{b}^4\right) \\ \mathbf{N}^5(t) &= \mathbf{LW}^{5,4}\mathbf{A}^4(t) + \mathbf{b}^5 \\ \mathbf{a}^5(t) &= \mathbf{softmax}\left(\left[\mathbf{N}^5(t)\right]^T\right) \end{aligned}$$

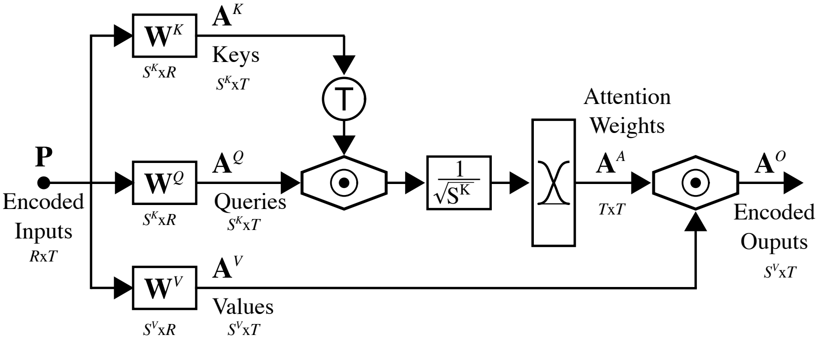
- To get the attention weights, multiply by the values $\mathbf{A}^1(T)$.

$$\mathbf{a}^6(t) = \mathbf{A}^1(T)\mathbf{a}^5(t)$$

Concatenation attention diagram



Self dot product attention (basis for the transformer)



Self attention example (1)

$$\mathbf{P} = \begin{bmatrix} \mathbf{p}(1) & \mathbf{p}(2) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 2 \\ 1 & 0 \end{bmatrix}$$

$$\mathbf{W}^Q = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \mathbf{W}^K = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}, \mathbf{W}^V = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}$$

$$\mathbf{A}^Q = \mathbf{W}^Q\mathbf{P} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 2 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 2 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} \mathbf{a}^Q(1) & \mathbf{a}^Q(2) \end{bmatrix}$$

$$\mathbf{A}^K = \mathbf{W}^K\mathbf{P} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 2 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 2 \\ 2 & 0 \end{bmatrix} = \begin{bmatrix} \mathbf{a}^K(1) & \mathbf{a}^K(2) \end{bmatrix}$$

$$\mathbf{A}^V = \mathbf{W}^V\mathbf{P} = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 2 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 2 \\ 2 & 0 \end{bmatrix} = \begin{bmatrix} \mathbf{a}^V(1) & \mathbf{a}^V(2) \end{bmatrix}$$

Self attention example (2)

$$
\begin{aligned}
\mathbf{N}^A &= \frac{1}{\sqrt{2}} \left[\mathbf{A}^K\right]^T \mathbf{A}^Q = \frac{1}{\sqrt{2}} \begin{bmatrix} 0 & 2 \\ 2 & 0 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 1 & 0 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 2 & 0 \\ 2 & 4 \end{bmatrix} \\
&= \begin{bmatrix} \sqrt{2} & 0 \\ \sqrt{2} & 2\sqrt{2} \end{bmatrix} \\
&= \frac{1}{\sqrt{2}} \begin{bmatrix} \mathbf{a}^K(1)^T \\ \mathbf{a}^K(2)^T \end{bmatrix} \begin{bmatrix} \mathbf{a}^Q(1) & \mathbf{a}^Q(2) \end{bmatrix} \\
&= \frac{1}{\sqrt{2}} \begin{bmatrix} \mathbf{a}^K(1)^T \mathbf{a}^Q(1) & \mathbf{a}^K(1)^T \mathbf{a}^Q(2) \\ \mathbf{a}^K(2)^T \mathbf{a}^Q(1) & \mathbf{a}^K(2)^T \mathbf{a}^Q(2) \end{bmatrix} \\
\mathbf{A}^A &= \text{softmax}\left(\mathbf{N}^A, \text{columns}\right) = \begin{bmatrix} 0.5 & 0.06 \\ 0.5 & 0.94 \end{bmatrix} \\
\mathbf{A}^O &= \mathbf{A}^V \mathbf{A}^A = \begin{bmatrix} 0.5\mathbf{a}^V(1) + 0.5\mathbf{a}^V(2) & 0.06\mathbf{a}^V(1) + 0.94\mathbf{a}^V(2) \end{bmatrix} \\
&= \begin{bmatrix} 1.5 & 1.94 \\ 1 & 0.12 \end{bmatrix}
\end{aligned}
$$