DARTMOUTH

Attention

Lecture 21 of “Mathematics and AI”

Outline

1. Self-supervised learning
2. Neural-model architectures for learning structured data
3. Attention

Self-supervised learning

Structured data

- (Semantic) relationships between input features
- Examples of structured data:
  - Time series, text, images, video, matrices, tensors, graphs
- How should a model account for those?
- Examples of neural models for structured data:
  - CNNs, RNNs, …

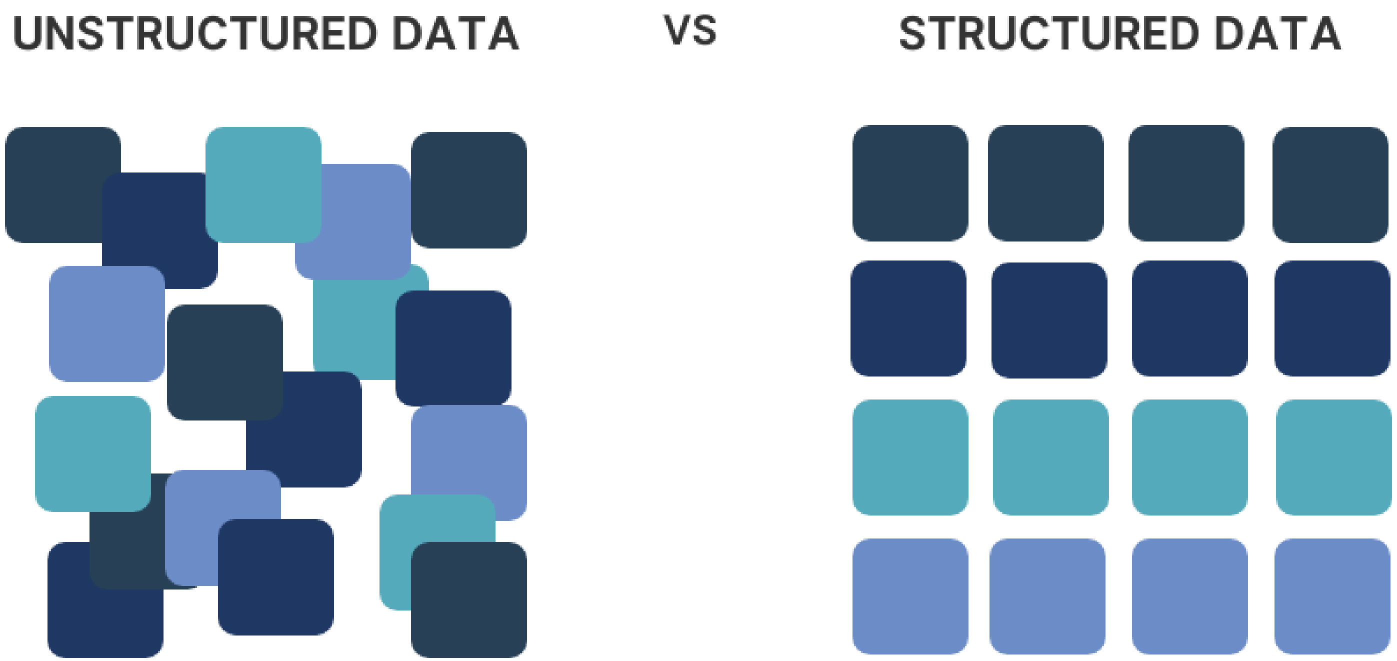


Image source: https://dryviq.com/unstructured-vs-structured-data-4-key-management-differences/

Self-supervised learning (SSL)

- Predict a missing part of x_i from other parts of x_i
- Use structured data $x_1, x_2, \ldots$
- No need for labeled observations $(x_1, y_1), (x_2, y_2), \ldots$
- Examples:
  - Fill in the blanks

    ```
    <sos> I think I **** apples more than oranges <eos>
    ```

  - Complete the sentence

    ```
    <sos> Thank you for your kind ****
    ```

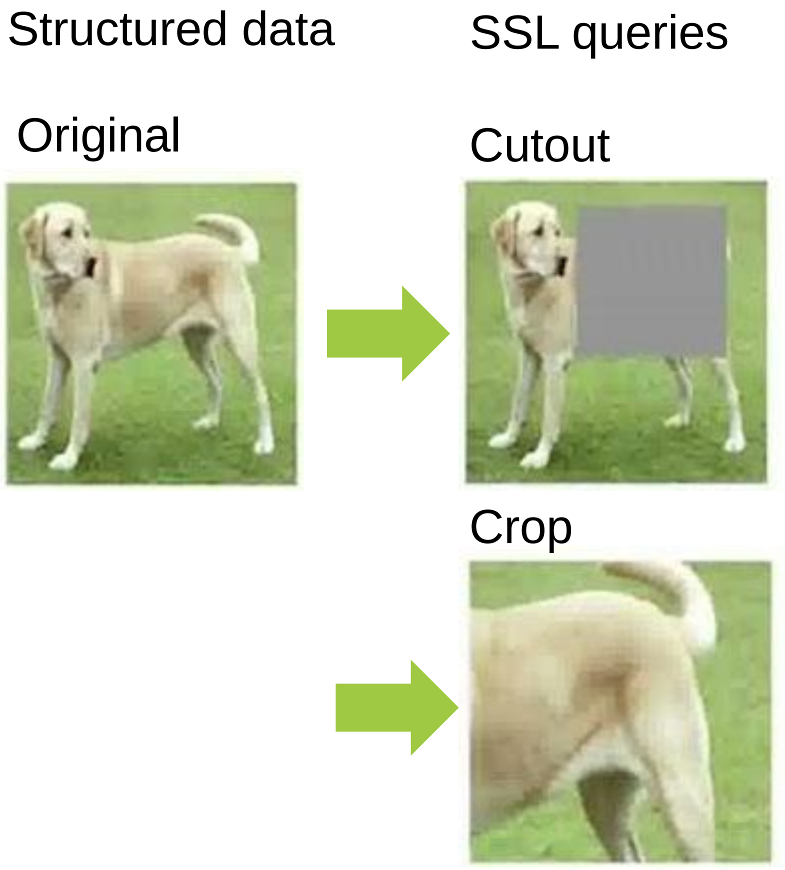


Image source: https://www.youtube.com/watch?v=vICrdSUS9vg

Neural-model architectures for learning structured data

Neural approach to learning structured data

- Structured learning:

  Weights should reflect aspects of the structure of the data

  (specifically the structural aspects that we expect to be important for learning)

- Ensemble learning:

  Train multiple learners in parallel that can focus on different aspects of the structure

Image processing via CNNs

- Structured learning:
  - CNN layers have sparse weight tensors (compared to complete layers)
  - Similarity of the relationship between adjacent pixels corresponds to filters being independent of pixel position
- Ensemble learning:
  - Each filter corresponds to a separate channel in the next layer

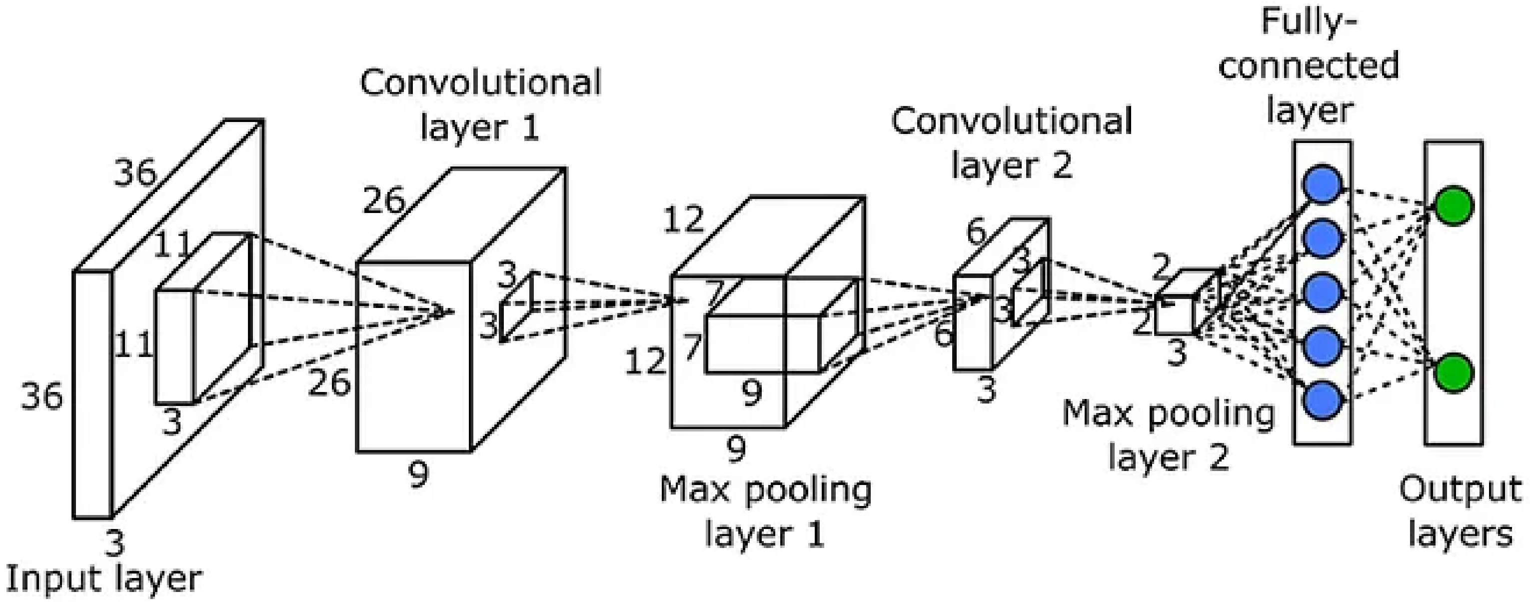


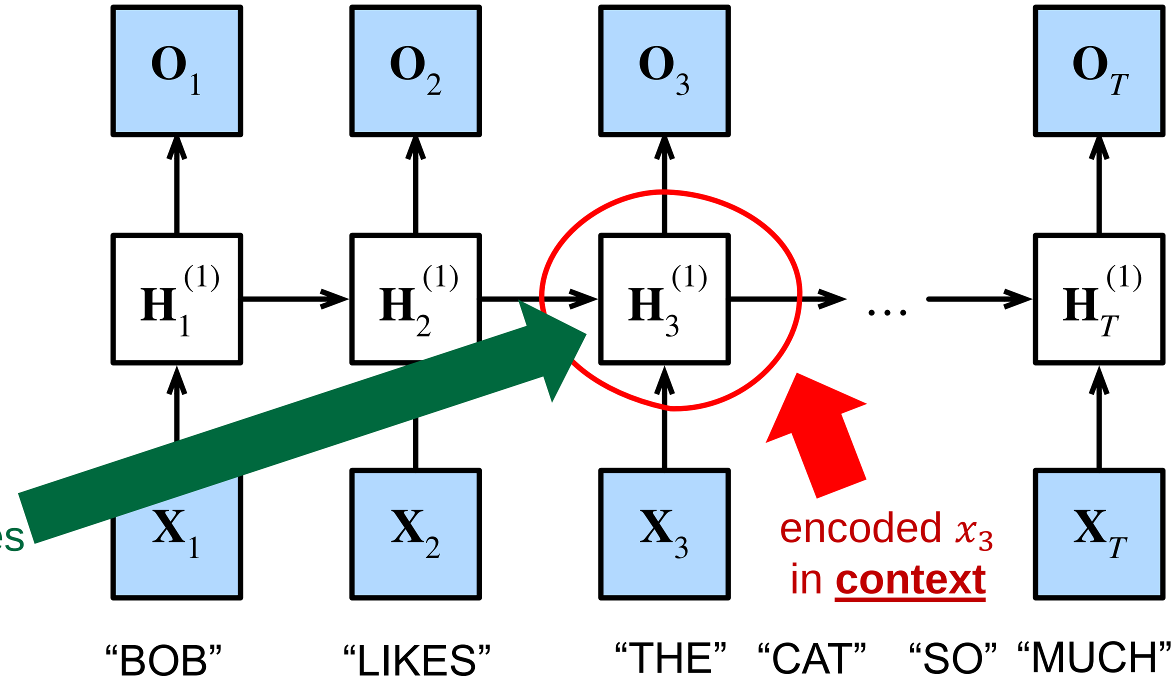
Image source: https://nafizshahriar.medium.com/what-is-convolutional-neural-network-cnn-deep-learning-b3921bdd82d5

Sequence learning via RNNs

- Structured learning:

  Hidden states $H_i^{(k)} = \left(x_i^{(k)}, h_i^{(k)}\right)$ in k –th layer include decaying memory of previous hidden states

Shallow RNN “unfolded”



Hidden state H_3 encodes
“BOB LIKES THE”

Sequence learning via RNNs

- Structured learning:

  Hidden states $H_i^{(k)} = \left(x_i^{(k)}, h_i^{(k)}\right)$ in k –th layer include decaying memory of previous hidden states

- Ensemble learning:

  Can include multiple $H_i^{(k)}$ for each input $x_i^{(in)}$

Shallow RNN "unfolded"

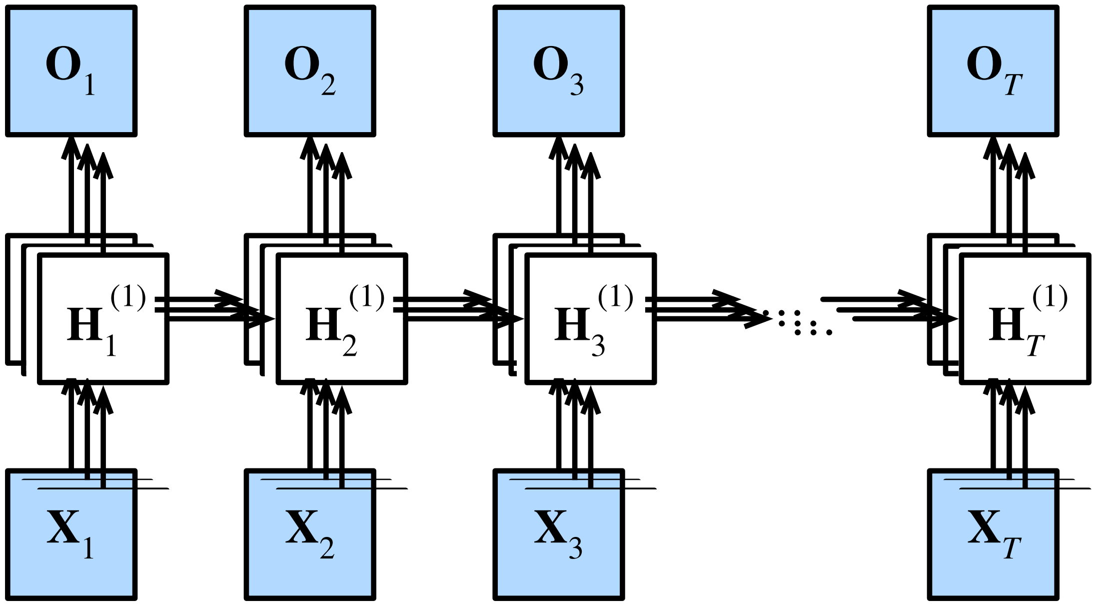
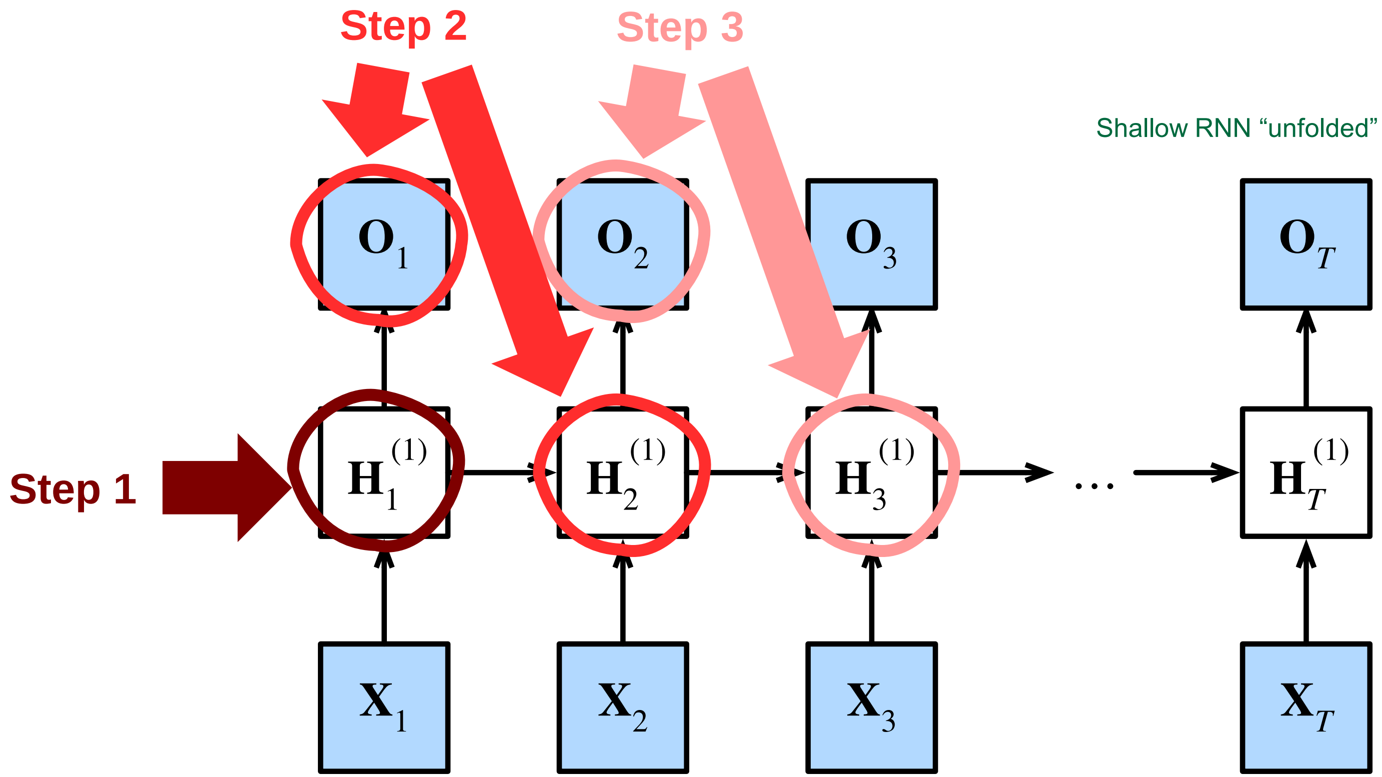


Image source: https://d2l.ai/chapter_recurrent-modern/deep-rnn.html

Forward pass in a RNN

- Sequential processing of information
- No parallel computation scheme
- Training RNNs tends to be slow



Sequence learning via bidirectional RNNs

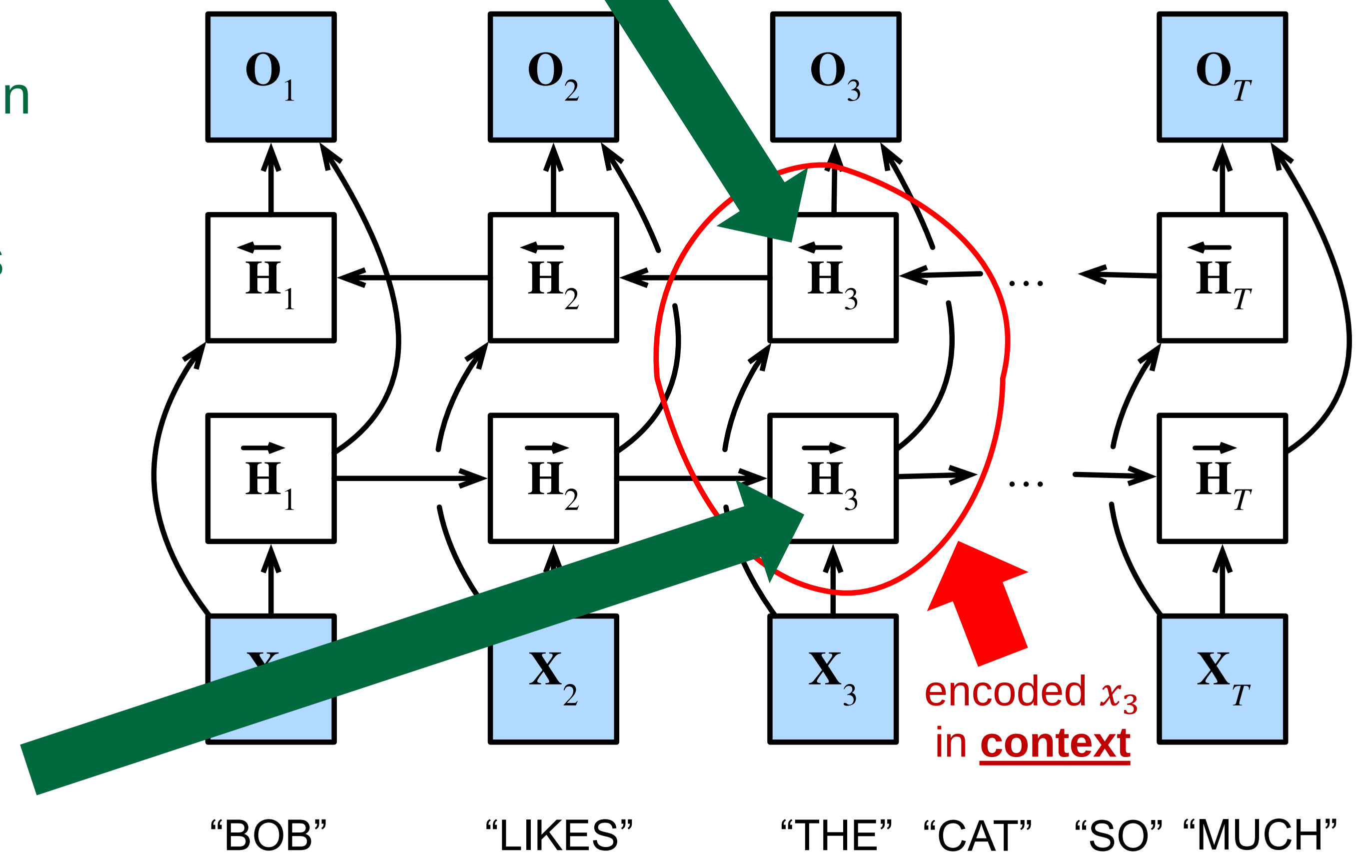
- Structured learning:

  Forward hidden states $\vec{H}_i^{(k)} = \left(x_i^{(k)}, \vec{h}_i^{(k)}\right)$ in the k-th encoder layer include decaying memory of previous forward hidden states

- Backward hidden states $\overleftarrow{H}_i^{(k)} = \left(x_i^{(k)}, \overleftarrow{h}_i^{(k)}\right)$ in the k-th decoder layer include decaying memory of subsequent backward hidden states

Backward hidden state $\overleftarrow{H}_3$ encodes "THE CAT SO MUCH"

Bidirectional RNN "unfolded"



Forward hidden state $\vec{H}_3$ encodes "BOB LIKES THE"

Image source: https://d2l.ai/chapter_recurrent-modern/bi-rnn.html

Forward pass in a bidirectional RNN

- Sequential processing of information
- Whole input sequence read before output is produced
- No parallel computation scheme
- Training biRNNs tends to be *even slower* than training RNNs

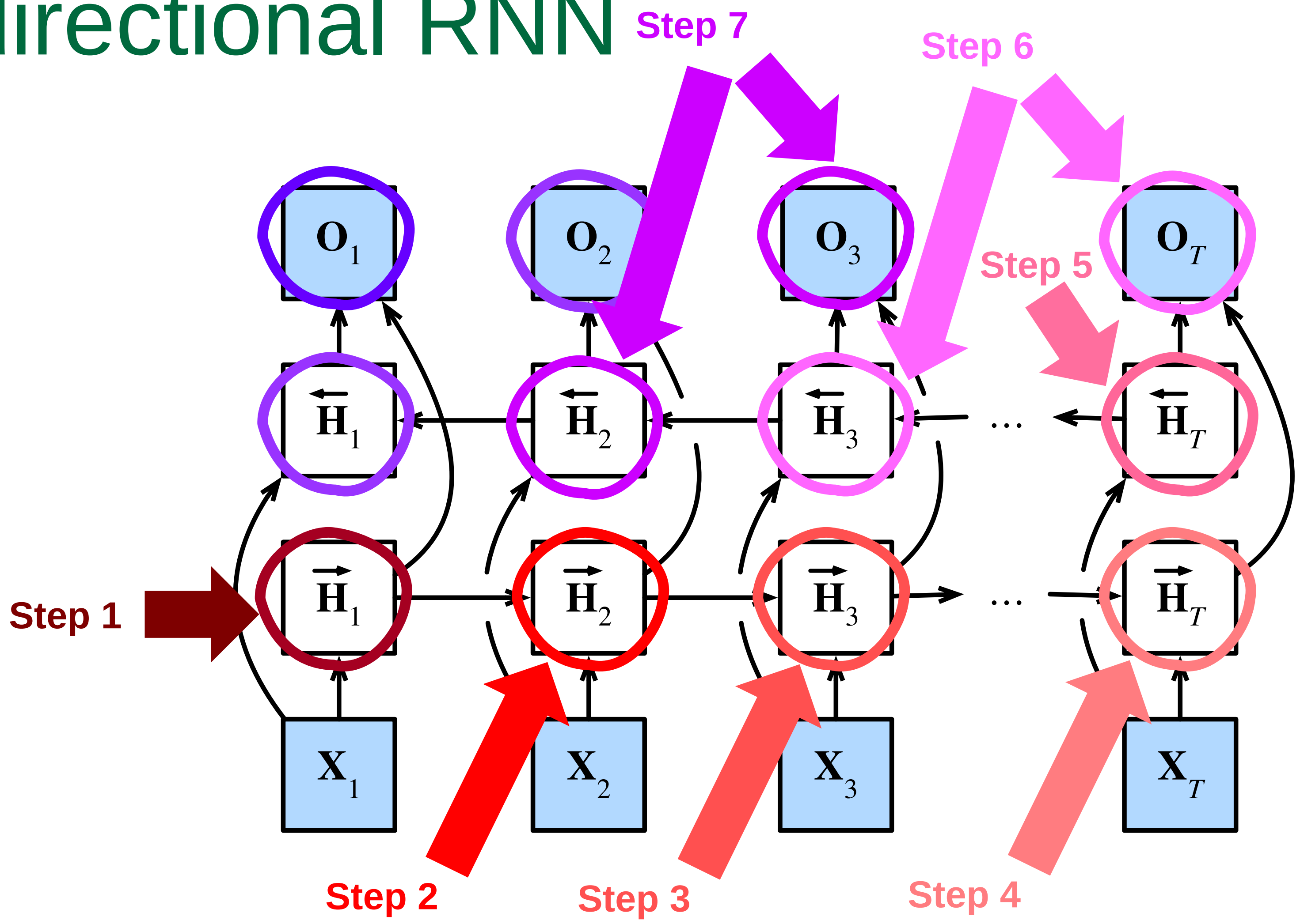


Image source: https://d2l.ai/chapter_recurrent-modern/bi-rnn.html

Context

"BOB LIKES THE CAT SO MUCH"

Proximity in ***word order*** matters less than proximity in ***meaning***

Attention

Attention in bi-directional RNNs

- Direct attention to forward hidden states (i.e., embedded features) that seem particularly relevant to the context

- Measure relevance via some similarity

$$a_{i,j} = \overleftarrow{H}_i \cdot \overrightarrow{H}_j$$

(or kernel function)

- Context for prediction O_2 is

$$\overleftarrow{H}_2 = f\left(x_2^{(in)}, \sum_j softmax(a_{3,j})\overrightarrow{H}_j\right)$$

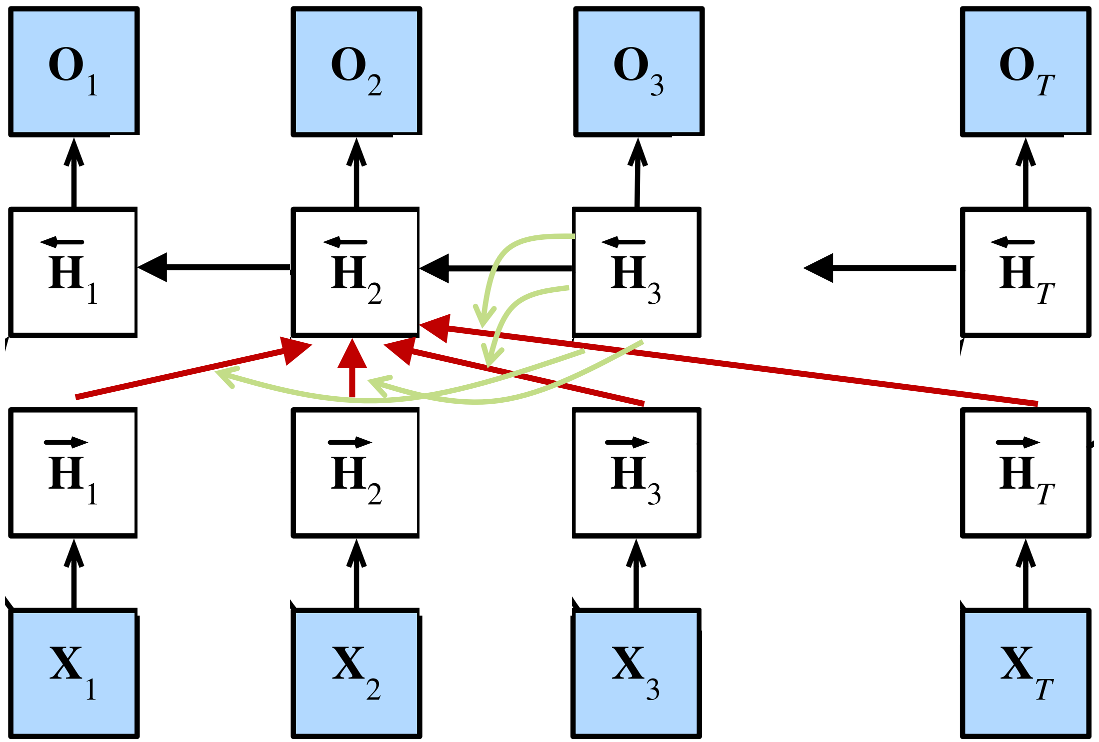


Image source: https://d2l.ai/chapter_recurrent-modern/bi-rnn.html

Query-key-value (QKV) formulation

- Motivated by information retrieval from data bases
- Previous example used backward hidden states $\overleftarrow{H}_i$ as queries and forward hidden states as $\overrightarrow{H}_j$ keys and values

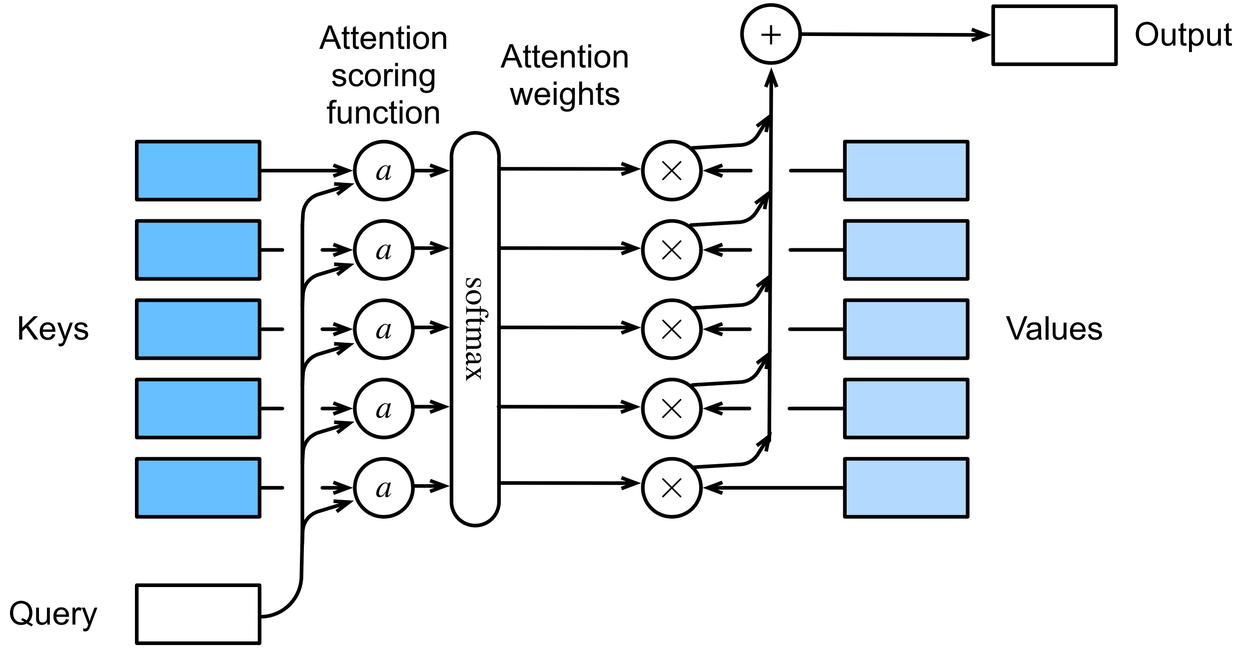


Image source: https://d2l.ai/chapter_attention-mechanisms-and-transformers/queries-keys-values.html

Self-attention

- Direct attention to forward hidden states (i.e., embedded features) that seem particularly relevant to the context
- Measure relevance via some similarity

$$a_{i,j} = \overleftarrow{H}_j \cdot \overrightarrow{H}_j$$

(or kernel function)

- Context for prediction O_2 is

$$\overleftarrow{H}_2 = f\left(x_2^{(in)}, \sum_j softmax(a_{2,j})\overrightarrow{H}_j\right)$$

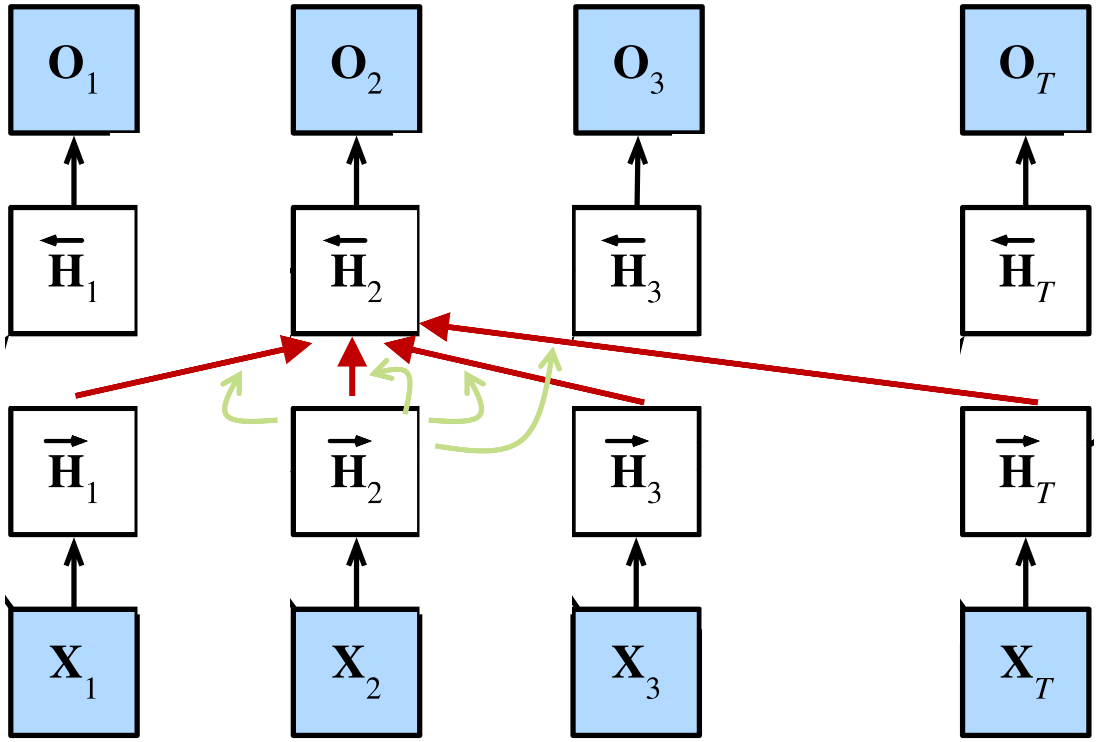


Image source: https://d2l.ai/chapter_recurrent-modern/bi-rnn.html

Self-attention with QKV formulation

- Create 3 linear projections of $x_j^{(in)}$ using weights $W^{(key)}$, $W^{(value)}$, $W^{(query)}$
- Use queries, keys, and values
  - Queries: $q_j \;= W^{(query)} x_j^{(in)}$
  - Keys: $k_j \;= W^{(key)} x_j^{(in)}$
  - Values: $v_j \;= W^{(value)} x_j^{(in)}$
- Context for prediction O_2 is

$\vec{H}_2 \;= f\left(x_2^{(in)}, \sum_j softmax(a_{2,j}) v_j\right)$ with $a_{i,j} = q_i \cdot k_j$

Image source: https://pixabay.com/photos/page-book-words-letters-read-3380967

Examples

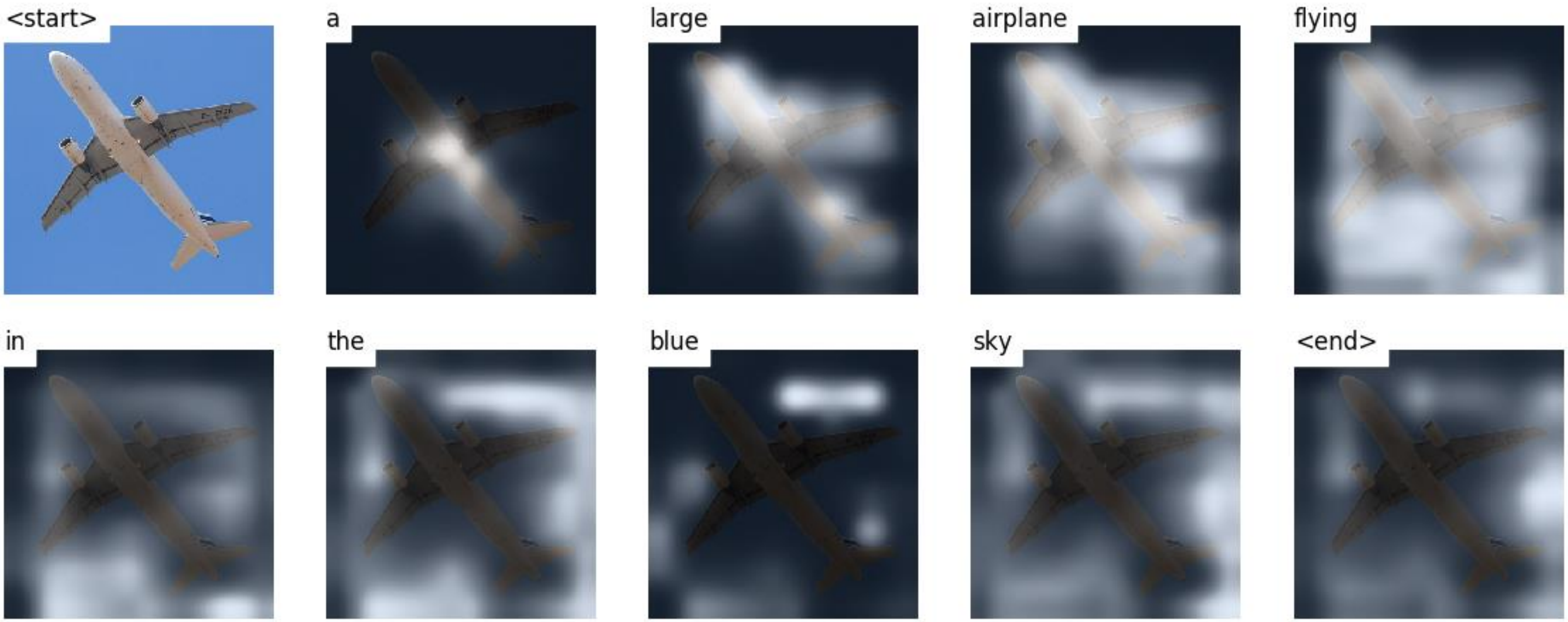


Image source: https://github.com/yunjey/show-attend-and-tell