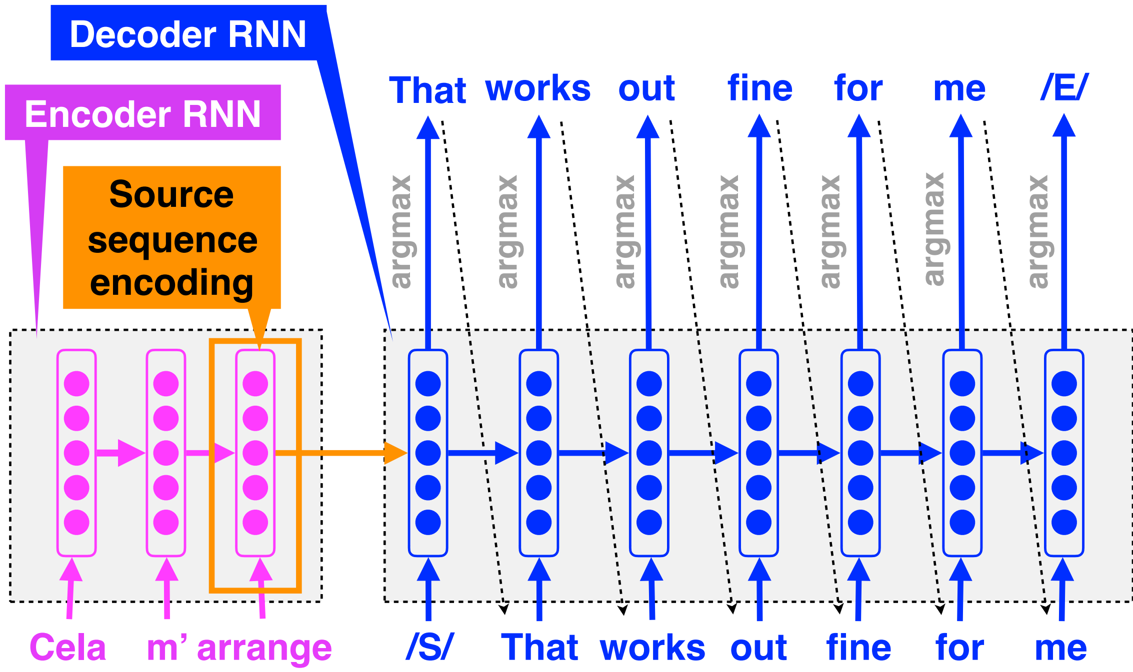
Applied Natural Language Processing

Dr. Jerome J. Braun

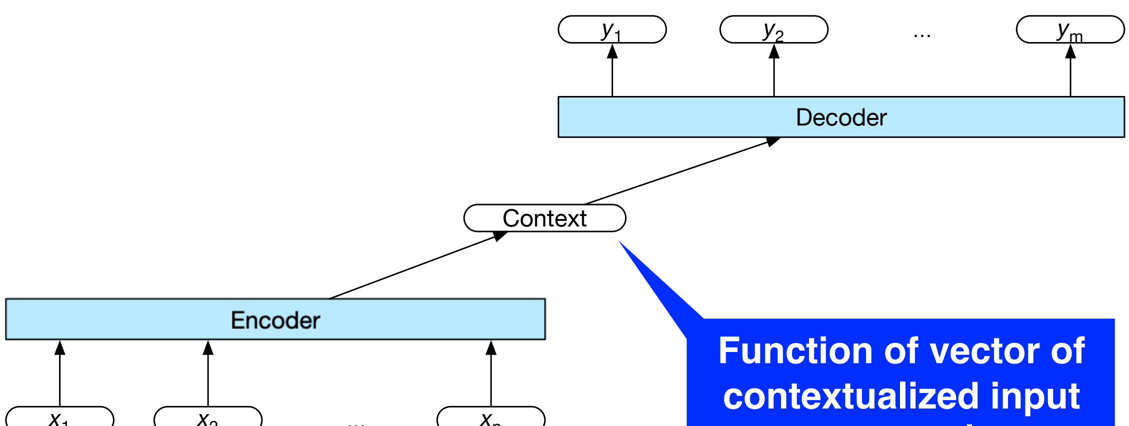
This Lecture: Neural Attention for NLP I

Course: Applied Natural Language Processing in Engineering
IE 7500

NMT Seq2seq at Test-time (Recap)



Abstract Encoder-Decoder Network



Function of vector of contextualized input representations

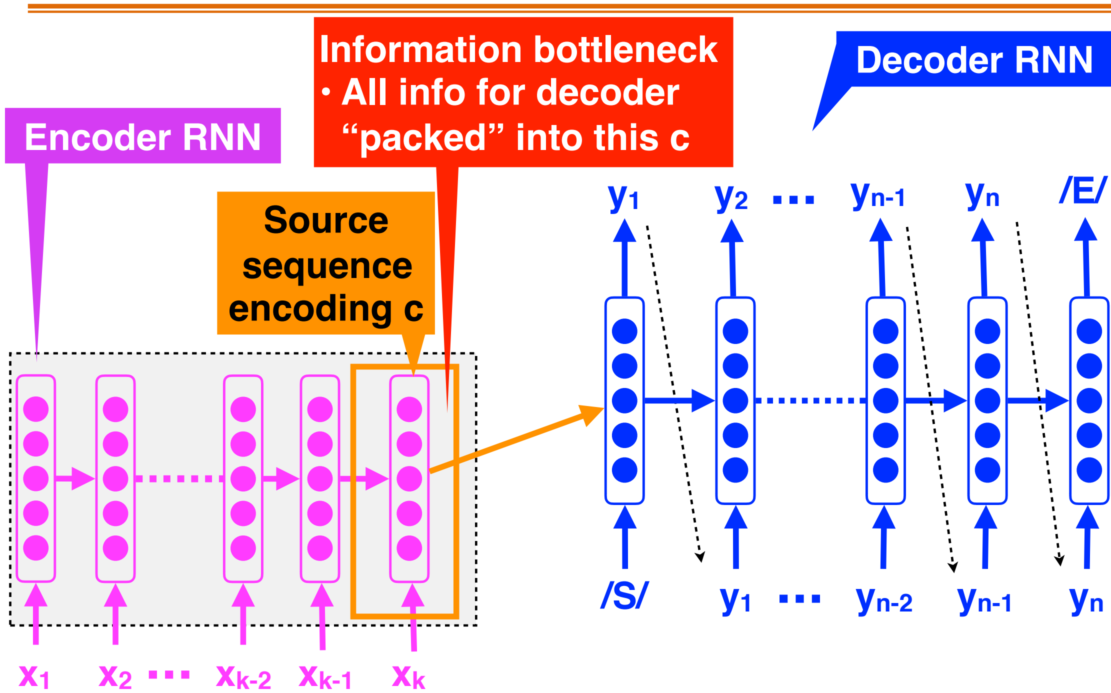
Can be used in decoder in various ways

Can use RNN, LSTM, GRU, etc., as encoder and decoder networks

Context

- Context vector c should be function of encoder's hidden states, i.e., $c = f(h_1 : h_n)$
  - But number of hidden states varies according to length of input
- Using encoder's final hidden state as context
  - Avoids issue of variable length of input
  - But introduces issue of excessive focus on latter parts of input sequence

Encoder-Decoder Information Bottleneck



Context

- **Addressing issue of excessive focus on latter parts of input sequence**
  - **Bi-RNNs — context vector formed by concatenating final states of forward and backward passes**
  - **Produce context vector by summing (or averaging) encoder hidden states**
    - **Loses info about individual encoder states that might be useful in decoding**

Attention Mechanism — Why

Better representation of context, for following reasons:

- **Take entire encoder context into account**
- **Update context dynamically during decoding**
- **Context remains fixed-size vector**

Attention Mechanism — How

- Static context vector replaced by dynamic construct c_i
  - Derived from encoder hidden states
    - At each point during decoding
  - New c_i generated at each decoding step i
    - This takes all hidden states of encoder into account
    - Context c_i made available during decoding
      - Current decoder state conditioned on all of the following
        - c_i
        - Prior hidden state
        - Previous output generated by decoder

Attention Mechanism

Decoder's previous output

Decoder's prior hidden state

$$h_i^d = g(\hat{y}_{i-1}, h_{i-1}^d, c_i)$$

Context vector

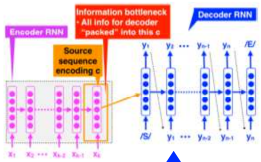
Computing c_i:

Vector of scores representing relevance encoder and decoder hidden states

$$score(h_{i-1}^d, h_j^e)$$

Computed for EACH encoder state j

Attention Mechanism



Without attention

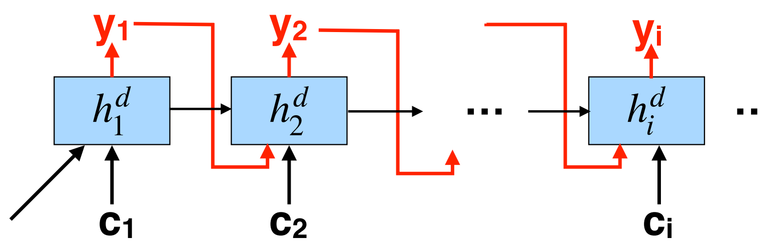
Decoder's prior hidden state

$$h_i^d = g(\hat{y}_{i-1}, h_{i-1}^d, c_i)$$

Decoder's previous output

Context vector

With attention



Computing c_i:

Vector of scores representing how relevant each encoder state is to decoder state h^d_{i-1}

$$score(h_{i-1}^d, h_j^e)$$

Computed for EACH encoder state j

Context c_i Computation

$$h_i^d = g(\hat{y}_{i-1}, h_{i-1}^d, c_i)$$

Computing c_i:

Assume this score is:
- Measure of similarity of (between):
  - Decoder hidden state
  - Each hidden state of encoder

Use dot product as similarity measure

$$score(h_{i-1}^d, h_j^e) = h_{i-1}^d \cdot h_j^e$$

More Robust Similarity Score (1)

- **Dot product is static measure — does not adapt during training**
- **More robust similarity score**

Parametrize score with dedicated set of weights (W_s)

$$score(h^d_{i-1}, h^e_j) = h^d_{i-1} W_s h^e_j$$

Allows network to learn WHICH ASPECTS of SIMILARITY between decoder and encoder states are important to current task

More Robust Similarity Score (2)

Normalize parametrized scores with softmax

$$\begin{aligned}\alpha_{ij} &= \mathrm{softmax}(score(h^d_{i-1}, h^e_j) \ \forall j \in e) \\ &= \frac{exp(score(h^d_{i-1}, h^e_j)}{\sum_k exp(score(h^d_{i-1}, h^e_k))}\end{aligned}$$

Results in vector of weights α_{ij} that express proportional relevance of each encoder hidden state j to current decoder state i

More Robust Similarity Score (3)

$$\alpha_{ij} = \text{softmax}(score(h^d_{i-1}, h^e_j) \;\forall j \in e)$$

$$= \frac{exp(score(h^d_{i-1}, h^e_j)}{\sum_k exp(score(h^d_{i-1}, h^e_k))}$$

Compute fixed-length context vector

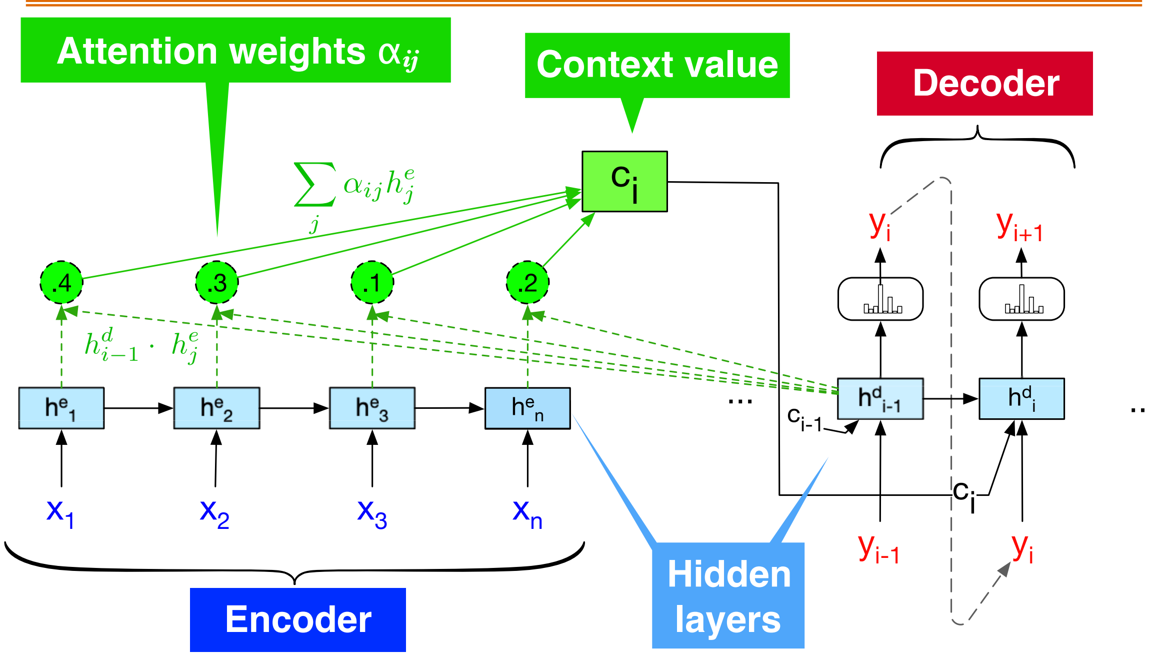
For *current* decoder state

Computed as weighted average over all encoder hidden states

$$c_i = \sum_j \alpha_{ij} h^e_j$$

Results in vector of weights α_{ij} that express proportional relevance of each encoder hidden state j to current decoder state i

Encoder-Decoder with Attention



- **Questions?**