Deep Learning

14 Encoder-Decoder Models & Attention

Dr. Konda Reddy Mopuri
Dept. of AI, IIT Hyderabad
Jan-May 2025

Sequence-to-sequence Models

- I/p is a sequence: $X_1, X_2, \ldots, X_N$

Sequence-to-sequence Models

- I/p is a sequence: $X_1, X_2, \ldots, X_N$
- O/p is a sequence: $Y_1, Y_2, \ldots, Y_M$

Sequence-to-sequence Models

- I/p is a sequence: $X_1, X_2, \ldots, X_N$
- O/p is a sequence: $Y_1, Y_2, \ldots, Y_M$
  - ASR: Speech i/p $\rightarrow$ word sequence

Sequence-to-sequence Models

- I/p is a sequence: $X_1, X_2, \ldots, X_N$
- O/p is a sequence: $Y_1, Y_2, \ldots, Y_M$
  - ASR: Speech i/p $\rightarrow$ word sequence
  - Machine Translation: word sequence$\rightarrow$ word sequence

Sequence-to-sequence Models

- I/p is a sequence: $X_1, X_2, \ldots, X_N$
- O/p is a sequence: $Y_1, Y_2, \ldots, Y_M$
  - ASR: Speech i/p $\rightarrow$ word sequence
  - Machine Translation: word sequence$\rightarrow$ word sequence
  - Dialog: user statement $\rightarrow$ system response

Sequence-to-sequence Models

- I/p is a sequence: $X_1, X_2, \ldots, X_N$
- O/p is a sequence: $Y_1, Y_2, \ldots, Y_M$
  - ASR: Speech i/p $\rightarrow$ word sequence
  - Machine Translation: word sequence$\rightarrow$ word sequence
  - Dialog: user statement $\rightarrow$ system response
  - Question Answering: Question i/p $\rightarrow$ Answer

Sequence-to-sequence Models

- No synchrony between X and Y $(M \neq N)$

Sequence-to-sequence Models

- No synchrony between X and Y $(M \neq N)$
- May not even maintain the order of the symbols

Sequence-to-sequence Models

- No synchrony between X and Y $(M \neq N)$
- May not even maintain the order of the symbols
- O/p symbols may not seem related to i/p

Sequence-to-sequence Models

- No synchrony between X and Y $(M \neq N)$
- May not even maintain the order of the symbols
- O/p symbols may not seem related to i/p
- E.g., The check I issued could not be encashed. $\rightarrow$ Did you check the balance in your account?

Language Model

Language Model

```
Naturalism and decision for the majority of Arab countries' capitalide was grounded
by the Irish language by [[John Clair]], [[An Imperial Japanese Revolt]], associated
with Guangzham's sovereignty. His generals were the powerful ruler of the Portugal
in the [[Protestant Immineners]], which could be said to be directly in Cantonese
Communication, which followed a ceremony and set inspired prison, training. The
emperor travelled back to [[Antioch, Perth, October 25|21]] to note, the Kingdom
of Costa Rica, unsuccessful fashioned the [[Thrales]], [[Cynth's Dajoard]], known
in western [[Scotland]], near Italy to the conquest of India with the conflict.
Copyright was the succession of independence in the slop of Syrian influence that
was a famous German movement based on a more popular servicious, non-doctrinal
and sexual power post. Many governments recognize the military housing of the
[[Civil Liberalization and Infantry Resolution 265 National Party in Hungary]],
that is sympathetic to be to the [[Punjab Resolution]]
(PJS)[http://www.humah.yahoo.com/guardian.
cfm/7754800786d17551963s89.htm Official economics Adjoint for the Nazism, Montgomery
was swear to advance to the resources for those Socialism's rule,
was starting to signing a major tripad of aid exile.]]
```

Figure: Andrej Karpathy

Language Model

- Models the probability of token sequences in the language(of characters or words)

Language Model

- Models the probability of token sequences in the language(of characters or words)
- Can

Language Model

- Models the probability of token sequences in the language(of characters or words)
- Can
  - Compute the probability of a given token sequence

Language Model

- Models the probability of token sequences in the language(of characters or words)
- Can
  - Compute the probability of a given token sequence
  - Generate sequences from the distribution of language

Language Model

- $p(y_1, y_2, y_3, y_4, \ldots)$

Language Model

- $p(y_1, y_2, y_3, y_4, \ldots)$
- Use Baye's rule to compute this incrementally
  $p(y_1) \cdot p(y_2/y_1) \cdot p(y_3/Y_1, y_2) \cdot p(y_3/y_1, y_2, y_3) \ldots$

Language Model

- $p(y_1, y_2, y_3, y_4, \ldots)$
- Use Baye's rule to compute this incrementally
  $p(y_1) \cdot p(y_2/y_1) \cdot p(y_3/Y_1, y_2) \cdot p(y_3/y_1, y_2, y_3) \ldots$
- They perform next token prediction

Language Model

1. $y^* = \text{argmax}\ P(y_t / y_1, y_2 \ldots y_{t-1})$

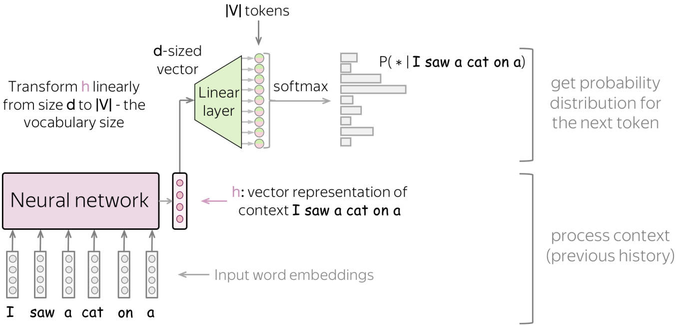
Language Model

1. $y^* = \text{argmax } P(y_t/y_1, y_2 \ldots y_{t-1})$
2. We have an NN (e.g. RNN or LSTM) first consuming the i/p sequence $(y_1^{t-1}) \rightarrow$ representation for the context

Language Model

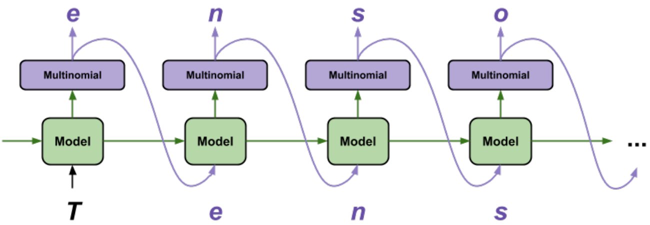
1. $y^* = \text{argmax } P(y_t/y_1, y_2 \ldots y_{t-1})$
2. We have an NN (e.g. RNN or LSTM) first consuming the i/p sequence $(y_1^{t-1}) \rightarrow$ representation for the context
3. Then, predict the probability distribution $P(y_t/y_1, y_2 \ldots y_{t-1})$ over the vocabulary

Language Model



Credits: Elena Voita

Language Model



Credits: TensorFlow

Language Model

- When do we stop?

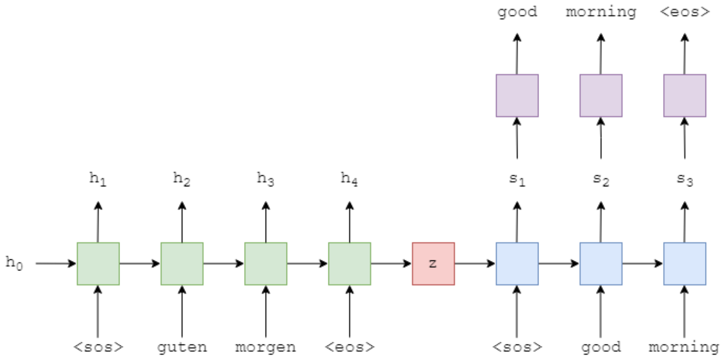
Language Model

- When do we stop?
- Add two additional tokens to the vocabulary

Language Model

- When do we stop?
- Add two additional tokens to the vocabulary
- <sos>: start of the sequence
- <eos>: end of the sequence

Language Model



Credits: PyTorch

Encoder-Decoder Framework

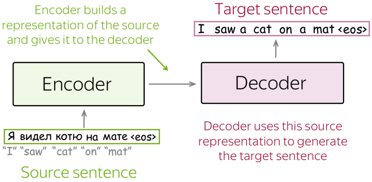
1. Standard modeling paradigm for sequence-to-sequence tasks

Encoder-Decoder Framework

1. Standard modeling paradigm for sequence-to-sequence tasks
2. Consists of two components: **Encoder** and **Decoder**

Encoder-Decoder Framework

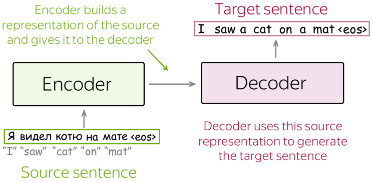
1. **Encoder**: reads source sequence to produce its representation



Credits: Elena Voita

Encoder-Decoder Framework

1. **Encoder**: reads source sequence to produce its representation
2. **Decoder**: uses the source representation given by the encoder to infer the target sequence



Credits: Elena Voita

Encoder-Decoder Model

1. Language modeling learns $p(y)$, where $y = (y_1, y_2, \dots y_n)$ is a sequence of tokens

Encoder-Decoder Model

1. Language modeling learns $p(y)$, where $y = (y_1, y_2, \ldots y_n)$ is a sequence of tokens
2. Seq2Seq need to model the conditional probability $p(y/x)$ of a sequence y given a sequence x (source or context)

Encoder-Decoder Model

1. Language modeling learns $p(y)$, where $y = (y_1, y_2, \ldots y_n)$ is a sequence of tokens
2. Seq2Seq need to model the conditional probability $p(y/x)$ of a sequence y given a sequence x (source or context)
3. Note that x need not be a sequence always (e.g. image in captioning)

Encoder-Decoder Model

1. Hence, Seq2Seq tasks can be modelled as conditional language models

Language Models: $P(y_1, y_2, \dots, y_n) = \prod_{t=1}^{n} p(y_t | y_{<t})$

Conditional Language Models: $P(y_1, y_2, \dots, y_n, | x) = \prod_{t=1}^{n} p(y_t | y_{<t}, x)$

condition on source x

Credits: Elene Voita

Encoder-Decoder Model

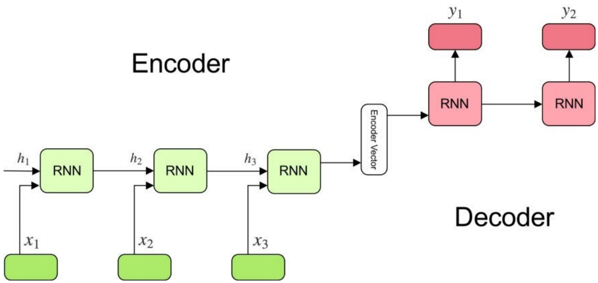
1. Basis for a lot of applications
   - Image (or video) captioning
   - Textual entailment
   - Machine translation
   - Transliteration
   - Document summarization
   - VQA: Visual Question Answering
   - Video classification
   - Chatbot for dialog

Encoder-Decoder Model

1. Basis for a lot of applications
   - Image (or video) captioning
   - Textual entailment
   - Machine translation
   - Transliteration
   - Document summarization
   - VQA: Visual Question Answering
   - Video classification
   - Chatbot for dialog
2. Let's consider machine translation...

Encoder-Decoder Model

- Simplest model is having two RNNs



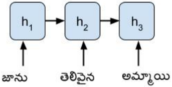
Credits: Simeon Kostadinov

Encoder-Decoder for Machine Translation

Input sequence: $x_1, x_2, \ldots x_T$

Output sequence: $y_1, y_2, \ldots y_{T'}$

Encoder: $h_t = E(x_t, h_{t-1})$



Sequence to sequence learning by Sutskever et al. NeurIPS 2014

Encoder-Decoder for Machine Translation

- Hope is that

Encoder-Decoder for Machine Translation

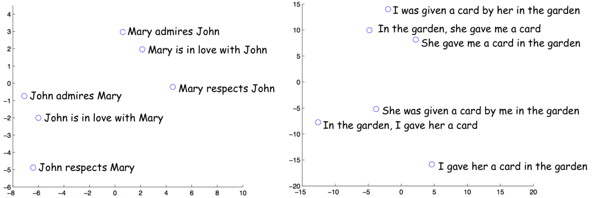
- Hope is that
  - Final encoder state 'encodes' all the information about the source

Encoder-Decoder for Machine Translation

- Hope is that
  - Final encoder state ‘encodes’ all the information about the source
  - This vector is sufficient for the decoder to generate the target sentence

Encoder-Decoder for Machine Translation

- Representations of sentences with similar meaning but different structure are close!



Sequence to sequence learning by Sutskever et al. NeurIPS 2014

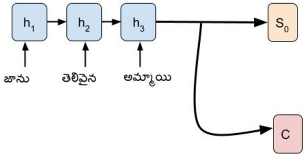
Encoder-Decoder for Machine Translation

Input sequence: $x_1, x_2, \ldots x_T$

Output sequence: $y_1, y_2, \ldots y_{T'}$

Last hidden state $h_T \rightarrow$ Initial state of the Decoder S_0 and the context information C
E.g. $S_0 \leftarrow h_T$ + dense layers, and $C \leftarrow h_T$

Encoder: $h_t = E(x_t, h_{t-1})$



Sequence to sequence learning by Sutskever et al. NeurIPS 2014

Encoder-Decoder for Machine Translation

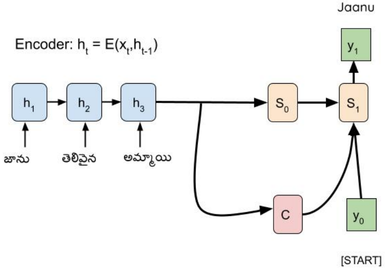
Input sequence: $x_1, x_2, \ldots x_T$

Output sequence: $y_1, y_2, \ldots y_{T'}$

Last hidden state $h_T \rightarrow$ Initial state of the Decoder S_0 and the context information C
E.g. $S_0 \leftarrow h_T$ + dense layers, and $C \leftarrow h_T$

Decoder: $s_t = D(y_{t-1}, s_{t-1}, C)$

Encoder: $h_t = E(x_t, h_{t-1})$



Sequence to sequence learning by Sutskever et al. NeurIPS 2014

Encoder-Decoder for Machine Translation

భారతీయ సాంకేతిక విజ్ఞాన సంస్థ హైదరాబాద్
भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

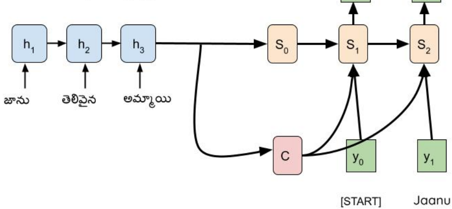
Input sequence: $x_1, x_2, \ldots\ x_T$

Output sequence: $y_1, y_2, \ldots\ y_{T'}$

Last hidden state $h_T \rightarrow$ Initial state of the Decoder S_0 and the context information C
E.g. $S_0 \leftarrow h_T$ + dense layers, and $C \leftarrow h_T$

Decoder: $s_t = D(y_{t-1}, s_{t-1}, C)$

Encoder: $h_t = E(x_t, h_{t-1})$



Sequence to sequence learning by Sutskever et al. NeurIPS 2014

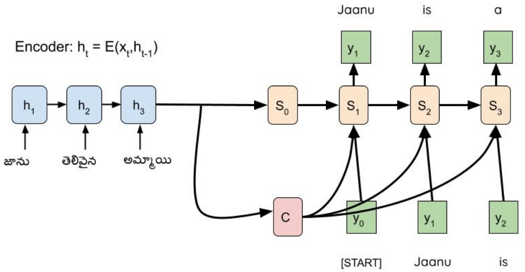
Encoder-Decoder for Machine Translation

Input sequence: $x_1, x_2, \ldots x_T$

Output sequence: $y_1, y_2, \ldots y_{T'}$

Last hidden state $h_T \rightarrow$ Initial state of the Decoder S_0 and the context information C
E.g. $S_0 \leftarrow h_T$ + dense layers, and $C \leftarrow h_T$

Decoder: $s_t = D(y_{t-1}, s_{t-1}, C)$



Sequence to sequence learning by Sutskever et al. NeurIPS 2014

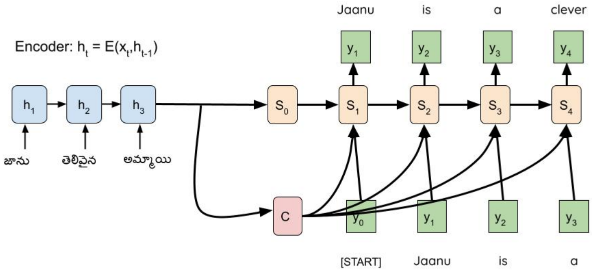
Encoder-Decoder for Machine Translation

Input sequence: x_1, x_2, ... x_T

Output sequence: y_1, y_2, ... $y_{T'}$

Last hidden state h_T → Initial state of the Decoder S_0 and the context information C
E.g. $S_0 \leftarrow h_T$ + dense layers, and $C \leftarrow h_T$

Decoder: $s_t = D(y_{t-1}, s_{t-1}, C)$



Sequence to sequence learning by Sutskever et al. NeurIPS 2014

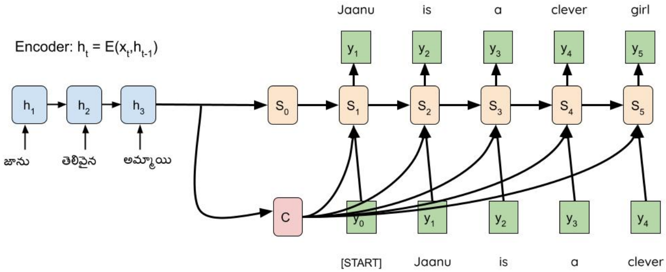
Encoder-Decoder for Machine Translation

Input sequence: $x_1, x_2, \ldots. x_T$

Output sequence: $y_1, y_2, \ldots. y_{T'}$

Last hidden state $h_T \rightarrow$ Initial state of the Decoder S_0 and the context information C
E.g. $S_0 \leftarrow h_T$ + dense layers, and $C \leftarrow h_T$

Decoder: $s_t = D(y_{t-1}, s_{t-1}, C)$



Sequence to sequence learning by Sutskever et al. NeurIPS 2014

Encoder-Decoder for Machine Translation

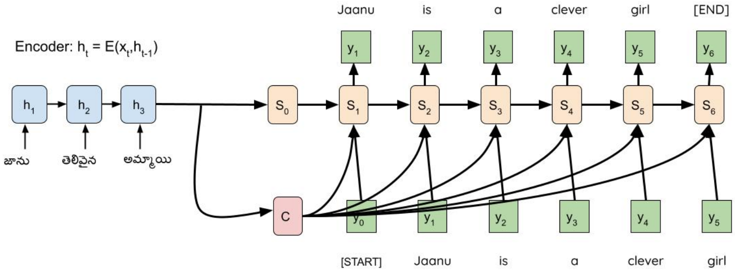
Input sequence: $x_1, x_2, \ldots\, x_T$

Output sequence: $y_1, y_2, \ldots\, y_{T'}$

Last hidden state $h_T \rightarrow$ Initial state of the Decoder S_0 and the context information C
E.g. $S_0 \leftarrow h_T$ + dense layers, and $C \leftarrow h_T$

Decoder: $s_t = D(y_{t-1}, s_{t-1}, C)$

Encoder: $h_t = E(x_t, h_{t-1})$



Sequence to sequence learning by Sutskever et al. NeurIPS 2014

Encoder-Decoder for Machine Translation

1. Encoder got only a single vector to encode the entire source sequence

Encoder-Decoder for Machine Translation

1. Encoder got only a single vector to encode the entire source sequence
2. Harsh compression, may lead to encoder forgetting something!

Encoder-Decoder for Machine Translation

1. Encoder got only a single vector to encode the entire source sequence
2. Harsh compression, may lead to encoder forgetting something!
3. Different information may be relevant for the decoder at different time steps

Encoder-Decoder for Machine Translation

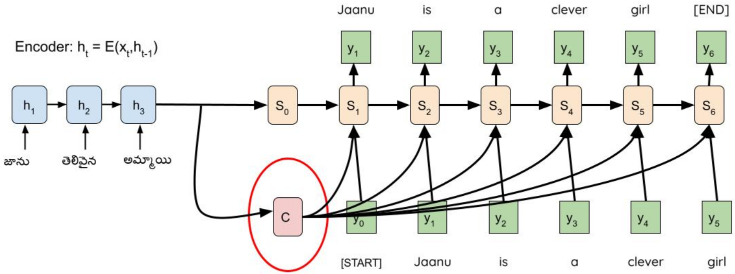
Input sequence: $x_1, x_2, \ldots. x_T$

Output sequence: $y_1, y_2, \ldots. y_{T'}$

Last hidden state $h_T \rightarrow$ Initial state of the Decoder S_0 and the context information C
E.g. $S_0 \leftarrow h_T$ + dense layers, and $C \leftarrow h_T$

Decoder: $s_t = D(y_{t-1}, s_{t-1}, C)$

Encoder: $h_t = E(x_t, h_{t-1})$



Bottleneck: Entire input is summarized by this vector!

Sequence to sequence learning by Sutskever et al. NeurIPS 2014

Encoder-Decoder for Machine Translation

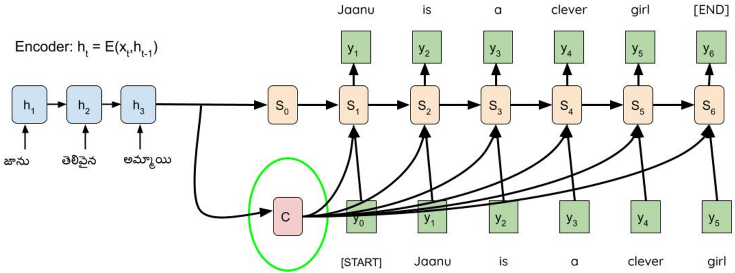
Input sequence: x_1, x_2, ... x_T

Output sequence: y_1, y_2, ... $y_{T'}$

Last hidden state $h_T \rightarrow$ Initial state of the Decoder S_0 and the context information C
E.g. $S_0 \leftarrow h_T$ + dense layers, and $C \leftarrow h_T$

Decoder: $s_t = D(y_{t-1}, s_{t-1}, C)$

Encoder: $h_t = E(x_t, h_{t-1})$



Solution: use different context at each time step!

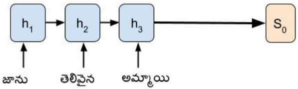
Sequence to sequence learning by Sutskever et al. NeurIPS 2014

Encoder-Decoder for Machine Translation with Attention

Input sequence: $x_1, x_2, \ldots x_T$

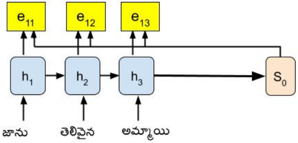
Input sequence: $y_1, y_2, \ldots y_{T'}$

Encoder: $h_t = E(x_t, h_{t-1})$



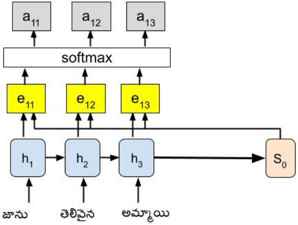
Encoder-Decoder for Machine Translation with Attention

Compute the alignment scores
$e_{t,i} = f_{att}(s_{t-1}, h_i)$ f_{att} - couple of dense layers



Neural Machine Translation with aligning by Bahdanau et al. ICLR 2015

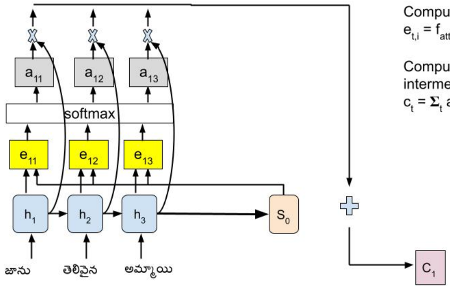
Encoder-Decoder for Machine Translation with Attention



Compute the alignment scores
$e_{t,i} = f_{att}(s_{t-1}, h_i)$ f_{att} - couple of dense layers

Neural Machine Translation with aligning by Bahdanau et al. ICLR 2015

Encoder-Decoder for Machine Translation with Attention



Compute the alignment scores

$e_{t,i} = f_{att}(s_{t-1}, h_i)$ f_{att} - couple of dense layers

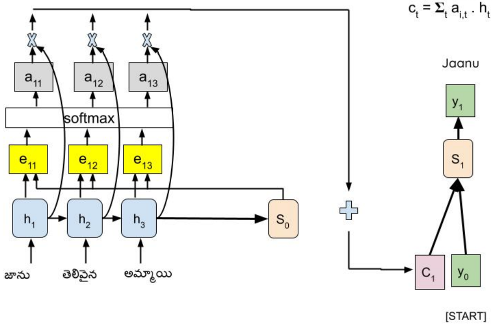
Compute the context as a linear combination of intermediate hidden states

$c_t = \Sigma_t a_{i,t} \cdot h_t$

Neural Machine Translation with aligning by Bahdanau et al. ICLR 2015

Compute the alignment scores
$e_{t,i} = f_{att}(s_{t-1}, h_i)$ f_{att} - couple of dense layers

Compute the context as a linear combination of intermediate hidden states
$c_t = \Sigma_t\, a_{i,t} \cdot h_t$



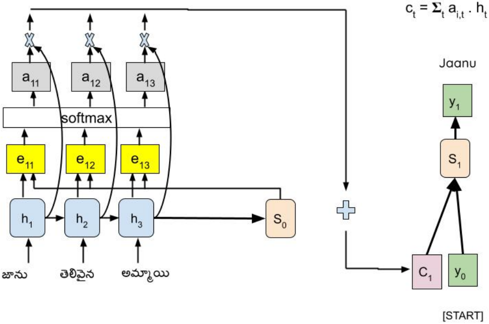
Decoder: $s_t = D(y_{t-1}, C_t)$

Neural Machine Translation with aligning by Bahdanau et al. ICLR 2015

Encoder-Decoder for Machine Translation with Attention

Compute the alignment scores
$e_{t,i} = f_{att}(s_{t-1}, h_i)$ f_{att} - couple of dense layers

Compute the context as a linear combination of intermediate hidden states
$c_t = \Sigma_t a_{i,t} \cdot h_t$

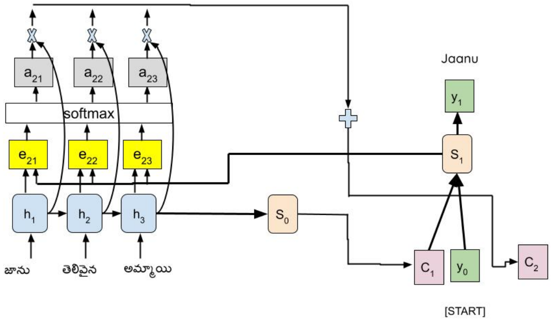


Decoder: $s_t = D(y_{t-1}, C_t)$

All these operations are differentiable!
Attention is learned using backprop!!

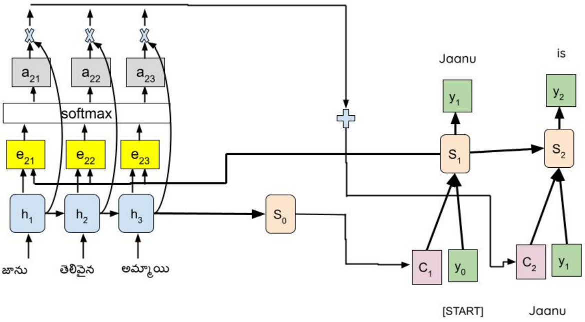
Neural Machine Translation with aligning by Bahdanau et al. ICLR 2015

Encoder-Decoder for Machine Translation with Attention



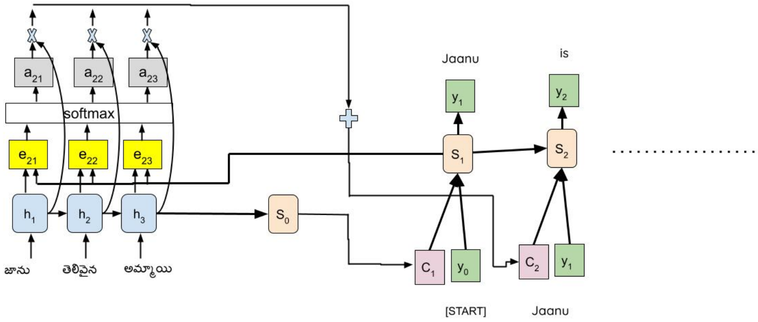
Neural Machine Translation with aligning by Bahdanau et al. ICLR 2015

Encoder-Decoder for Machine Translation with Attention



Neural Machine Translation with aligning by Bahdanau et al. ICLR 2015

Encoder-Decoder for Machine Translation with Attention



Neural Machine Translation with aligning by Bahdanau et al. ICLR 2015

Encoder-Decoder for Machine Translation with Attention

- Employs a different context at each time step of decoding

Neural Machine Translation with aligning by Bahdanau et al. ICLR 2015

Encoder-Decoder for Machine Translation with Attention

- Employs a different context at each time step of decoding
- No more bottleneck-ing of the input

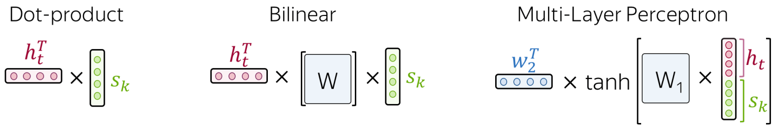
Neural Machine Translation with aligning by Bahdanau et al. ICLR 2015

Encoder-Decoder for Machine Translation with Attention

- Employs a different context at each time step of decoding
- No more bottleneck-ing of the input
- Decoder can 'attend' to different portions of the input at each time step

Neural Machine Translation with aligning by Bahdanau et al. ICLR 2015

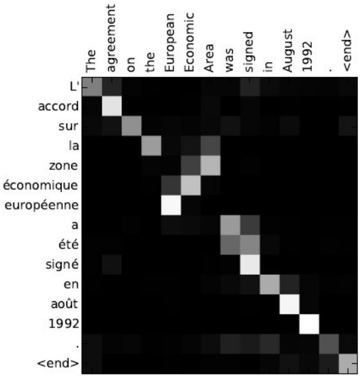
Encoder-Decoder for Machine Translation with Attention

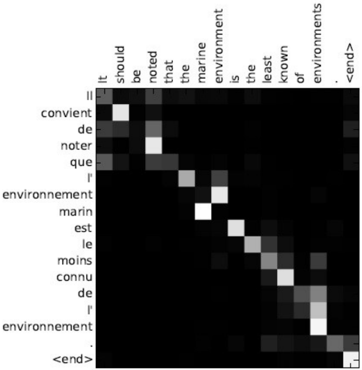


$\text{score}(h_t, s_k) = h_t^T \, s_k$ $\quad$ $\text{score}(h_t, s_k) = h_t^T \, W s_k$ $\quad$ $\text{score}(h_t, s_k) = w_2^T \cdot \tanh(W_1[h_t, s_k])$

Computing Attention
(Credits: Elene Voita)

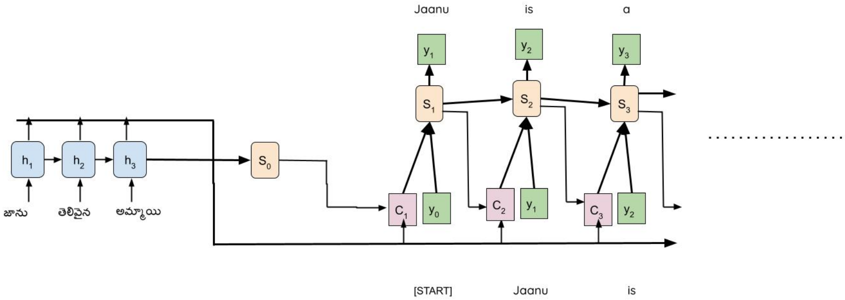
Encoder-Decoder for Machine Translation with Attention





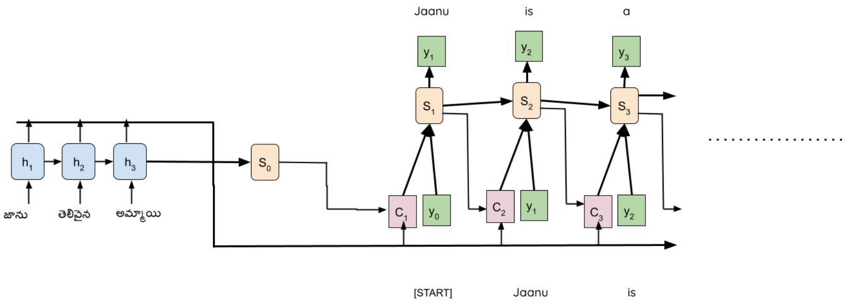
Neural Machine Translation with aligning by Bahdanau et al. ICLR 2015

Encoder-Decoder for Machine Translation with Attention



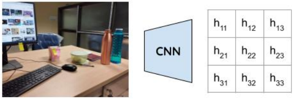
- Decoder doesn't consider the h_i to be an ordered set

Encoder-Decoder for Machine Translation with Attention



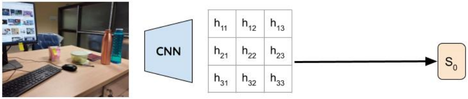
- Decoder doesn't consider the h_i to be an ordered set
- This architecture can be exploited to process a set of inputs h_i

Image captioning using RNNs with Attention



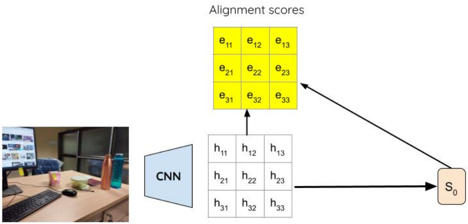
Show Attend and Tell by Xu et al. 2015

Image captioning using RNNs with Attention



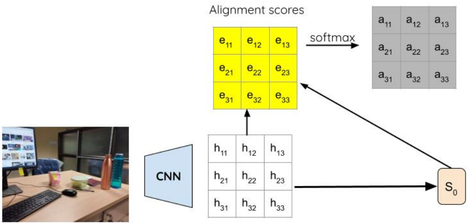
Show Attend and Tell by Xu et al. 2015

Image captioning using RNNs with Attention



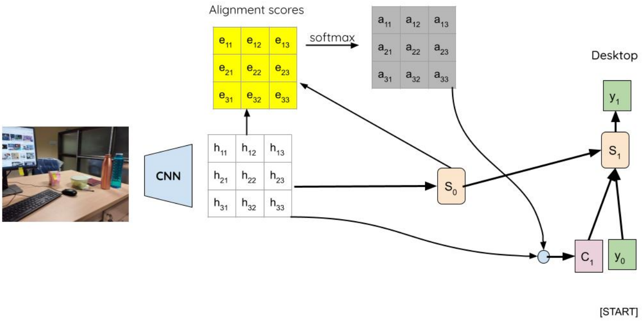
Show Attend and Tell by Xu et al. 2015

Image captioning using RNNs with Attention



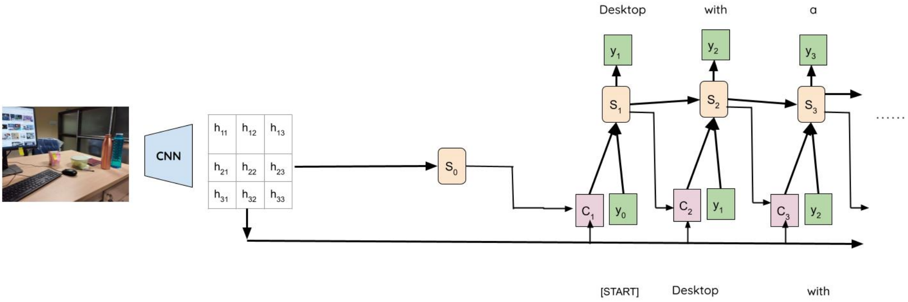
Show Attend and Tell by Xu et al. 2015

Image captioning using RNNs with Attention



Show Attend and Tell by Xu et al. 2015

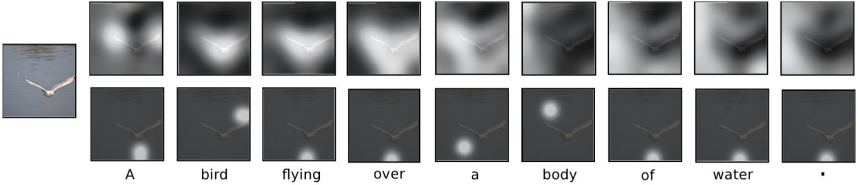
Image captioning using RNNs with Attention



Show Attend and Tell by Xu et al. 2015

Image captioning using RNNs with Attention

Figure 2. Attention over time. As the model generates each word, its attention changes to reflect the relevant parts of the image. "soft" (top row) vs "hard" (bottom row) attention. (Note that both models generated the same captions in this example.)



Show Attend and Tell by Xu et al. 2015

Image captioning using RNNs with Attention

A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

A stop sign is on a road with a mountain in the background.

A little girl sitting on a bed with a teddy bear.

A group of people sitting on a boat in the water.

A giraffe standing in a forest with trees in the background.

Show Attend and Tell by Xu et al. 2015