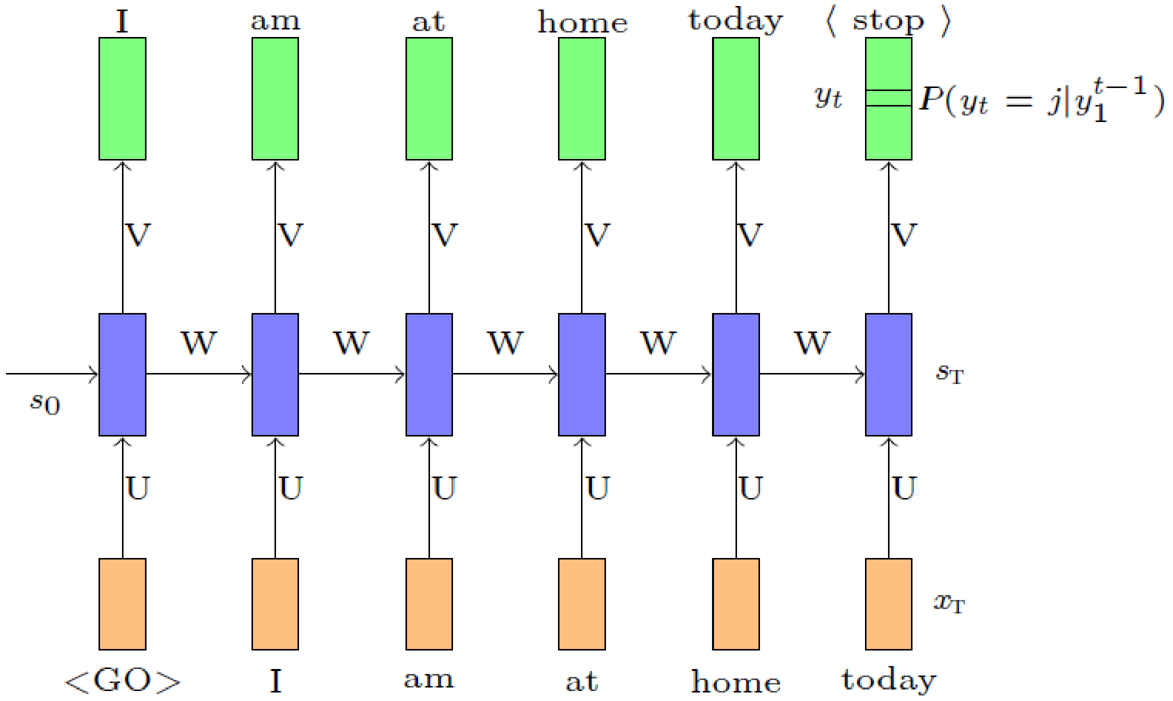
DSE 3121 DEEP LEARNING

Encoder-Decoder Models

Dr. Rohini Rao & Dr. Abhilash K Pai
Department of Data Science and Computer Applications
MIT Manipal

Encoder Decoder Models : Introduction

- Language Modeling:
  Given the t-i words predict the t^{th} word



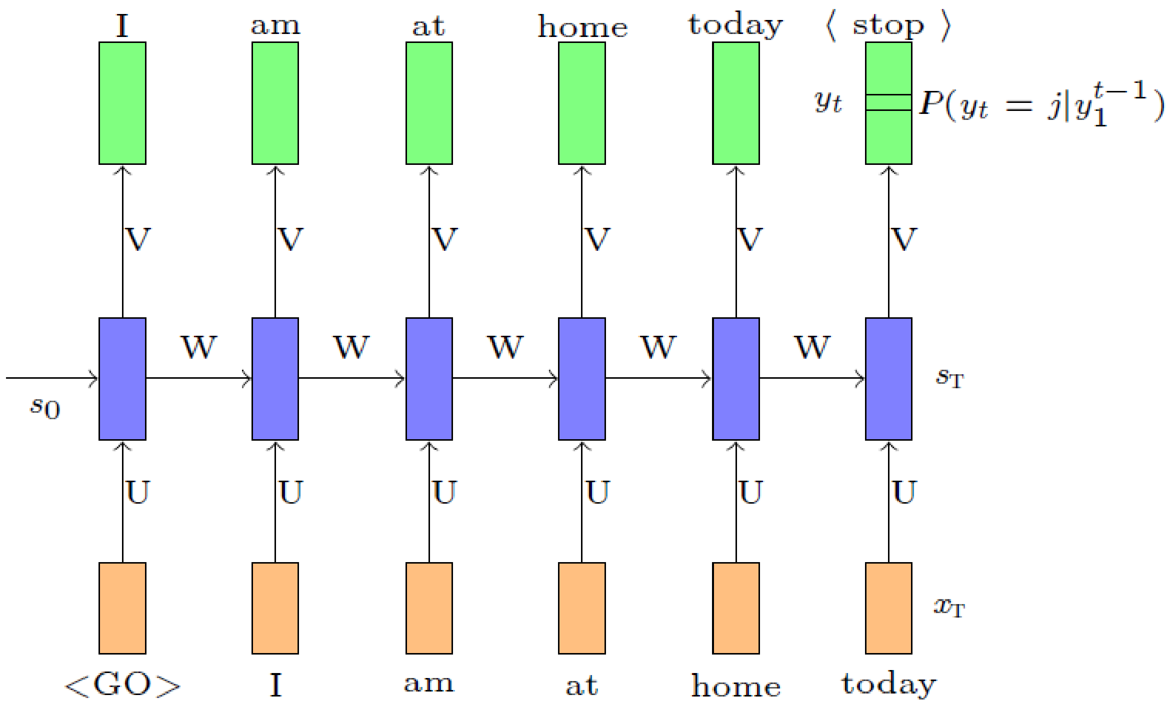
Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

Encoder Decoder Models : Introduction

- Language Modeling:
  Given the t-i words predict the tth word

More formally, given $y_1, y_2, ..., y_{t-1}$ we want to find

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



Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

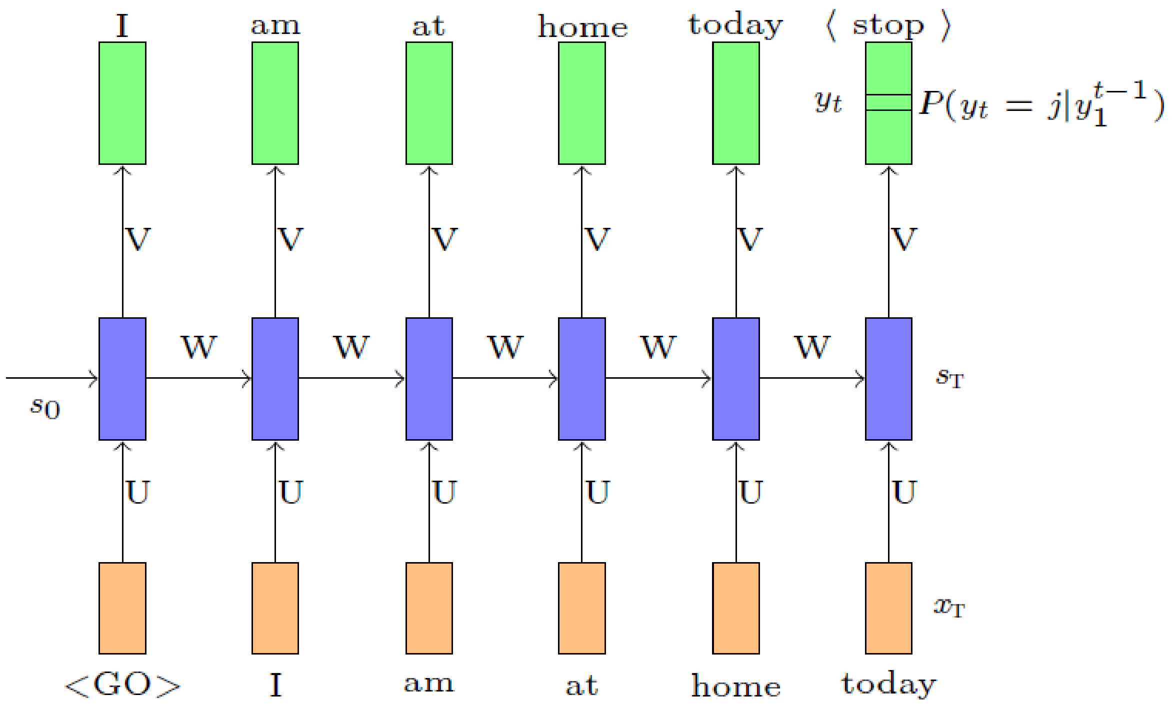
Encoder Decoder Models : Introduction

- Language Modeling:
  Given the t-i words predict the t^{th} word

More formally, given $y_1, y_2, ..., y_{t-1}$ we want to find

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

For simplicity let us denote



Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

Encoder Decoder Models : Introduction

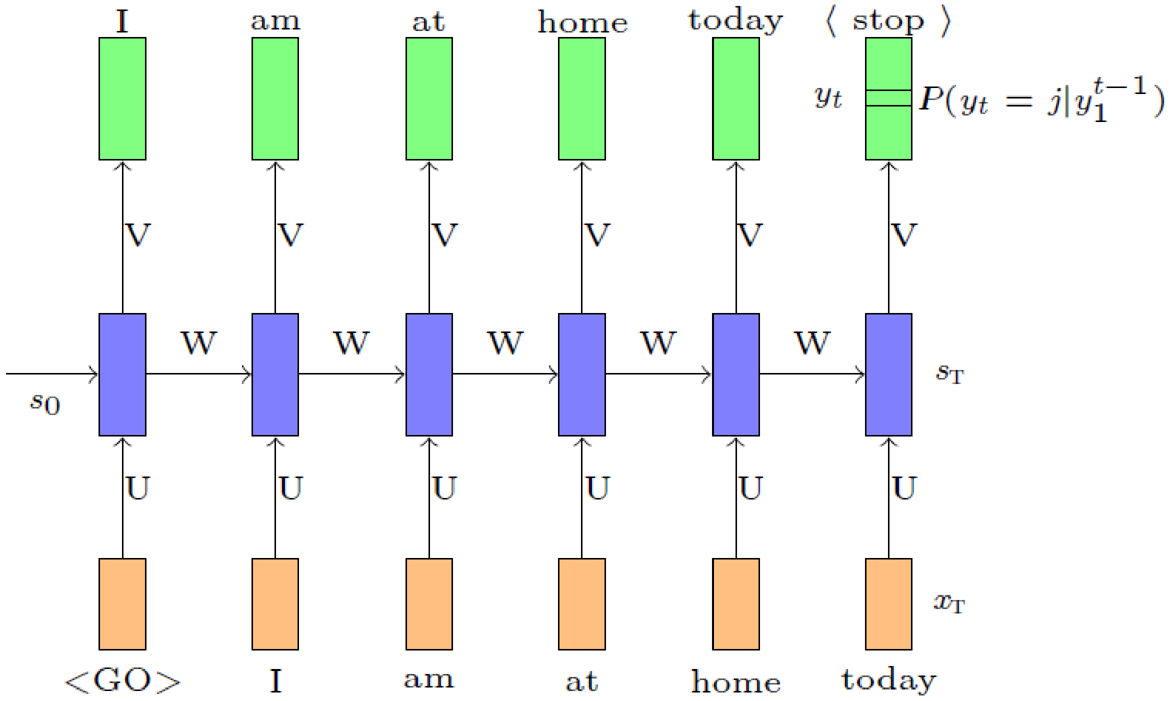
- Language Modeling:
  Given the t-i words predict the t^{th} word

More formally, given $y_1, y_2, ..., y_{t-1}$ we want to find

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

For simplicity let us denote

$$P(y_t | y_1, y_2 ... y_{t-1}) = P(y_t | y_1^{t-1})$$



Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

Encoder Decoder Models : Introduction

- Language Modeling:
  Given the t-i words predict the t^{th} word

More formally, given $y_1, y_2, ..., y_{t-1}$ we want to find

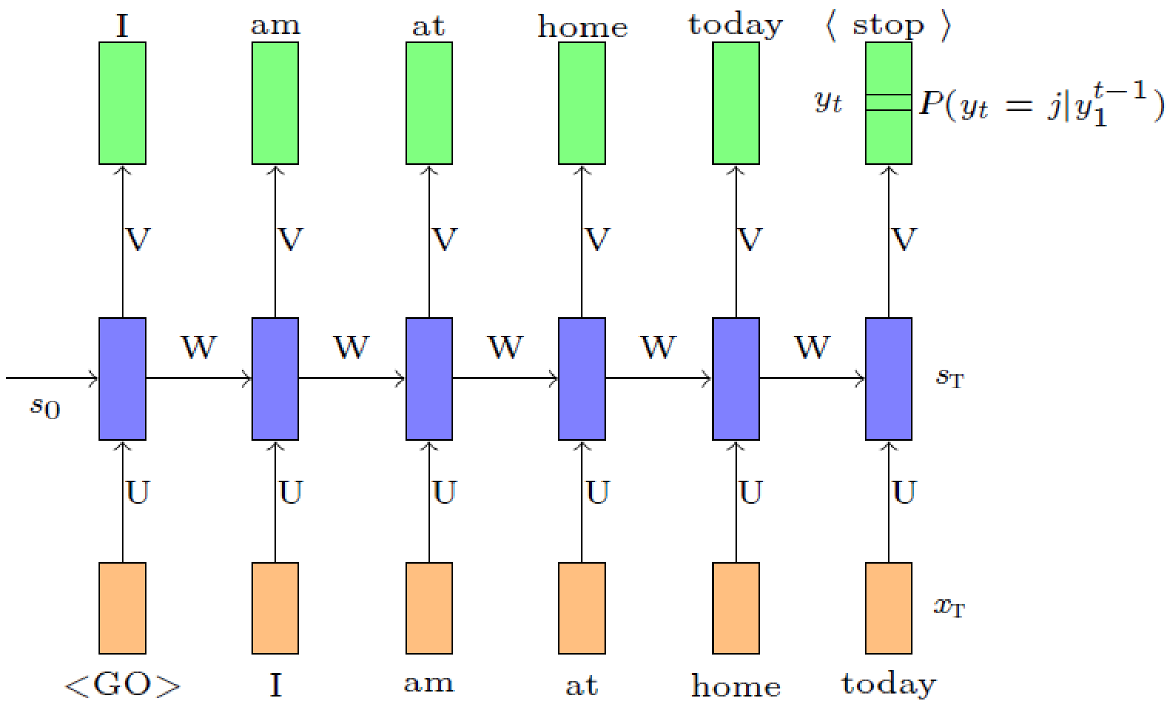
$$y^* = argmax\ P(y_t | y_1, y_2, ..., y_{t-1})$$

For simplicity let us denote

$$P(y_t | y_1, y_2 ... y_{t-1}) = P(y_t | y_1^{t-1})$$

We are interested in:

$$P(y_t = j | y_1, y_2 ... y_{t-1})$$



Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

Encoder Decoder Models : Introduction

- Language Modeling:
  Given the t-i words predict the t^{th} word

More formally, given $y_1, y_2, ..., y_{t-1}$ we want to find

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

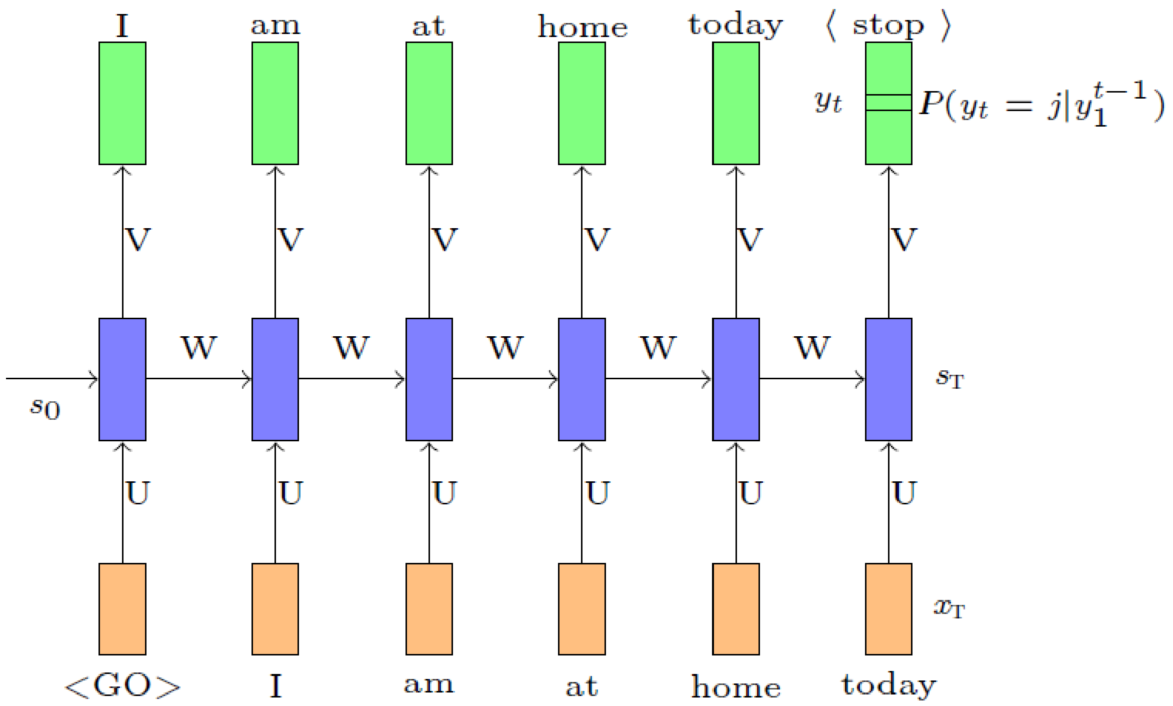
For simplicity let us denote

$$P(y_t|y_1, y_2 ... y_{t-1}) = P(y_t|y_1^{t-1})$$

We are interested in:

$$P(y_t = j|y_1, y_2 ... y_{t-1})$$

j^{th} word in the vocabulary



Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

Encoder Decoder Models : Introduction

- Language Modeling:
  Given the t-i words predict the t^{th} word

More formally, given $y_1, y_2, ..., y_{t-1}$ we want to find

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

For simplicity let us denote

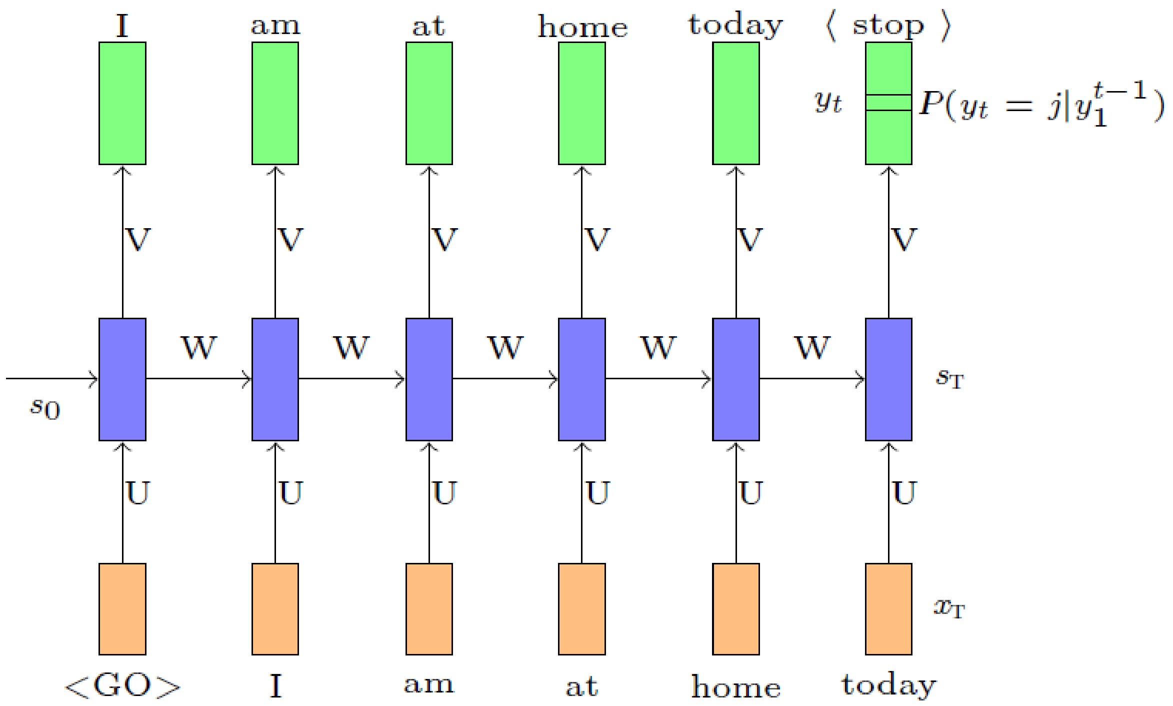
$$P(y_t|y_1, y_2 \ldots y_{t-1}) = P(y_t|y_1^{t-1})$$

We are interested in:

$$P(y_t = j|y_1, y_2 \ldots y_{t-1})$$

j^{th} word in the vocabulary

Using an RNN we will compute this as:



Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

Encoder Decoder Models : Introduction

- Language Modeling:
  Given the t-i words predict the t^{th} word

More formally, given $y_1, y_2, ..., y_{t-1}$ we want to find

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

For simplicity let us denote

$$P(y_t|y_1, y_2 ... y_{t-1}) \;=\; P(y_t|y_1^{t-1})$$

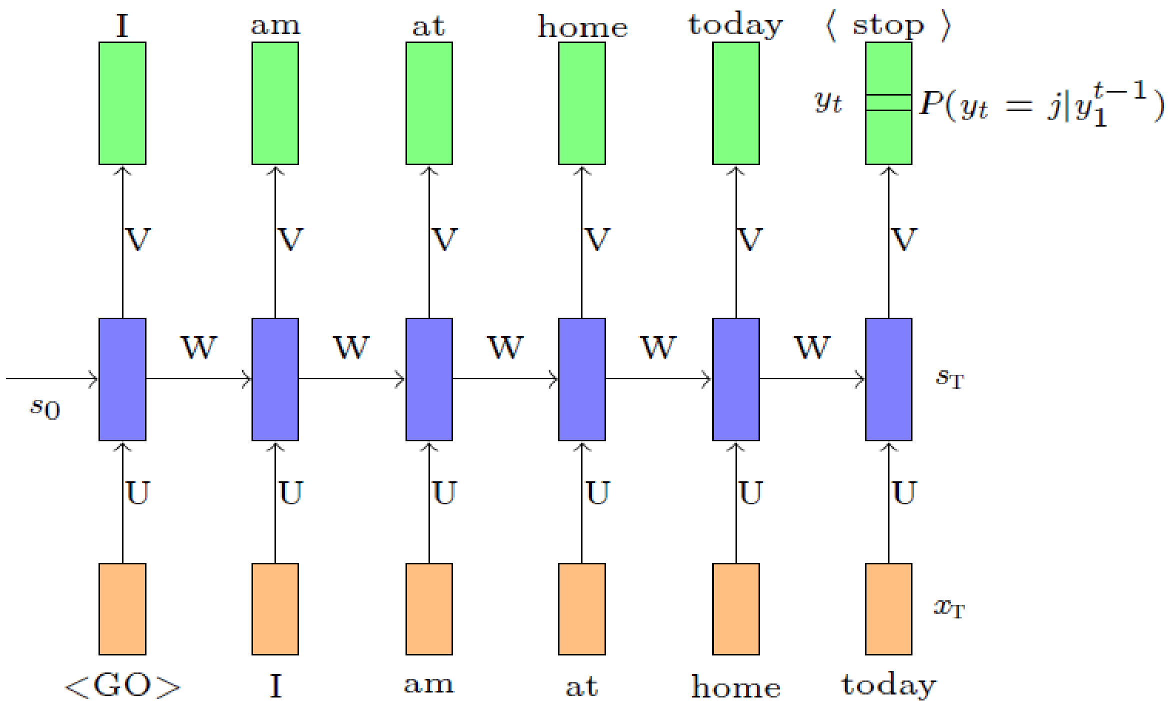
We are interested in:

$$P(y_t = j|y_1, y_2 ... y_{t-1})$$

j^{th} word in the vocabulary

Using an RNN we will compute this as:

$$P(y_t = j|y_1^{t-1}) = softmax(Vs_t + c)_j$$



Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

Encoder Decoder Models : Introduction

- Language Modeling:
  Given the t-i words predict the t^{th} word

More formally, given $y_1, y_2, ..., y_{t-1}$ we want to find

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

For simplicity let us denote

$$P(y_t | y_1, y_2 ... y_{t-1}) = P(y_t | y_1^{t-1})$$

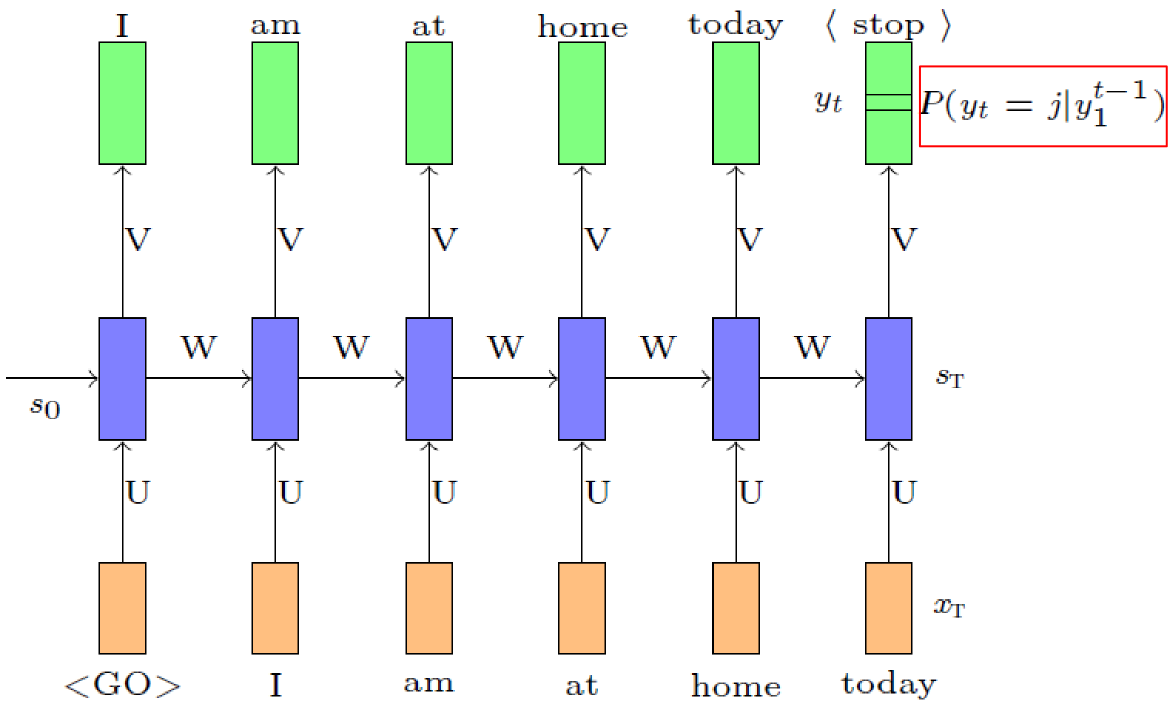
We are interested in:

$$P(y_t = j | y_1, y_2 ... y_{t-1})$$

j^{th} word in the vocabulary

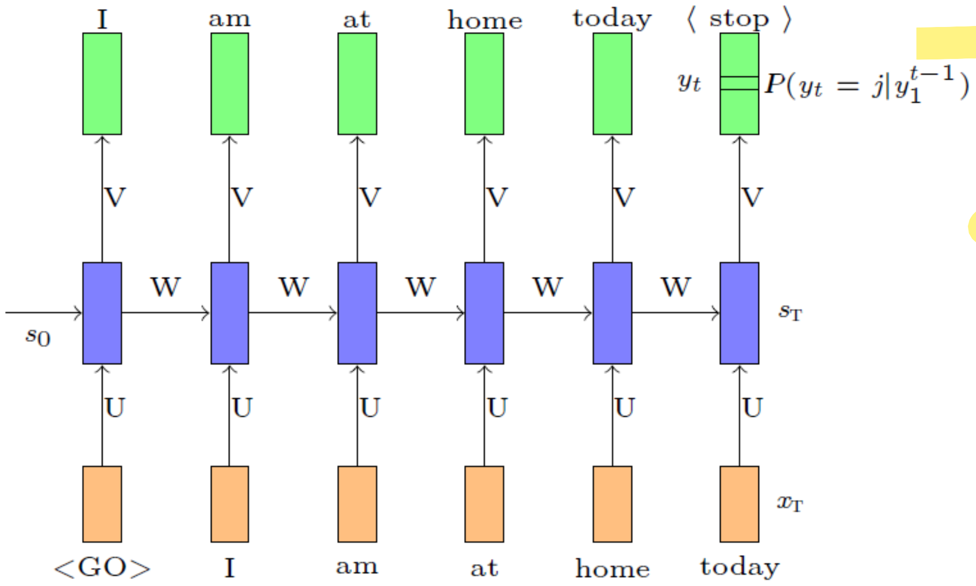
Using an RNN we will compute this as:

$$P(y_t = j | y_1^{t-1}) = softmax(Vs_t + c)_j$$



Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

Encoder Decoder Models : Introduction



Data: India, officially the Republic of India, is a country in South Asia. It is the seventh-largest country by area, ...

- **Data:** All sentences from any large corpus (say wikipedia)
- **Model:**

$$s_t = \sigma(Ws_{t-1} + Ux_t + b)$$

$$P(y_t = j|y_1^{t-1}) = softmax(Vs_t + c)_j$$

- **Parameters:** U, V, W, b, c
- **Loss:**

$$\mathscr{L}(\theta) = \sum_{t=1}^{T} \mathscr{L}_t(\theta)$$

$$\mathscr{L}_t(\theta) = -\log P(y_t = \ell_t|y_1^{t-1})$$

where ℓ_t is the true word at time step t

Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

Encoder Decoder Models : Introduction

- Shorthand notations:

$$s_t = \sigma(U x_t + W s_{t-1} + b)$$

$$\tilde{s}_t = \sigma(W(o_t \odot s_{t-1}) + U x_t + b)$$
$$s_t = i_t \odot s_{t-1} + (1 - i_t) \odot \tilde{s}_t$$

$$\tilde{s}_t = \sigma(W h_{t-1} + U x_t + b)$$
$$s_t = f_t \odot s_{t-1} + i_t \odot \tilde{s}_t$$
$$h_t = o_t \odot \sigma(s_t)$$

$$\Downarrow$$

$$s_t = \text{RNN}(s_{t-1}, x_t)$$

$$s_t = \text{GRU}(s_{t-1}, x_t)$$

$$h_t, s_t = \text{LSTM}(h_{t-1}, s_{t-1}, x_t)$$

Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

Encoder Decoder Models : Introduction

Task: generate a sentence given an image

A man throwing
a frisbee in a park

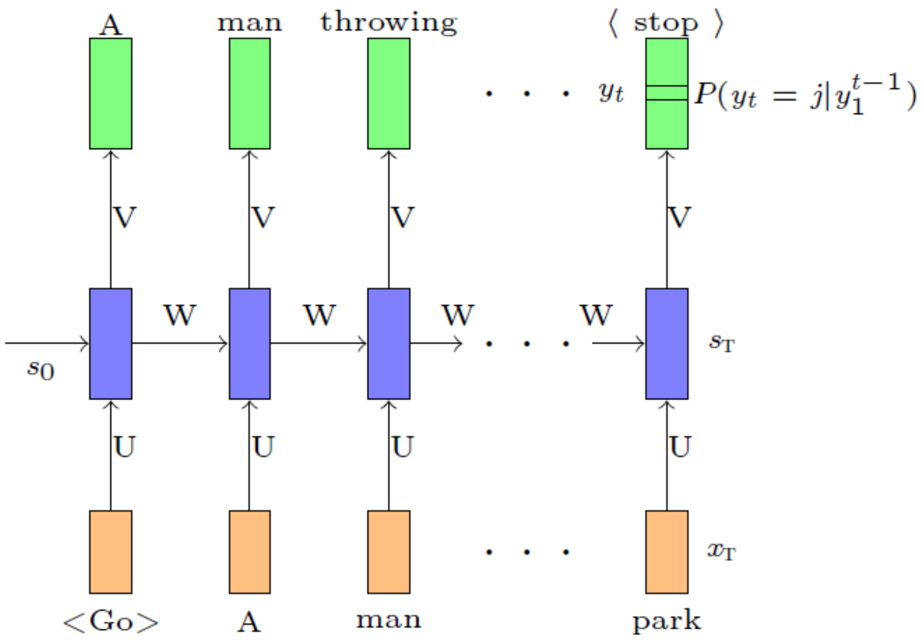
We are now interested in $P(y_t|y_1^{t-1}, I)$ instead of $P(y_t|y_1^{t-1})$ where I is an image

Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

Encoder Decoder Models : Introduction

- Earlier we modeled $P(y_t|y_1^{t-1})$ as

$$P(y_t|y_1^{t-1}) = P(y_t = j|s_t)$$

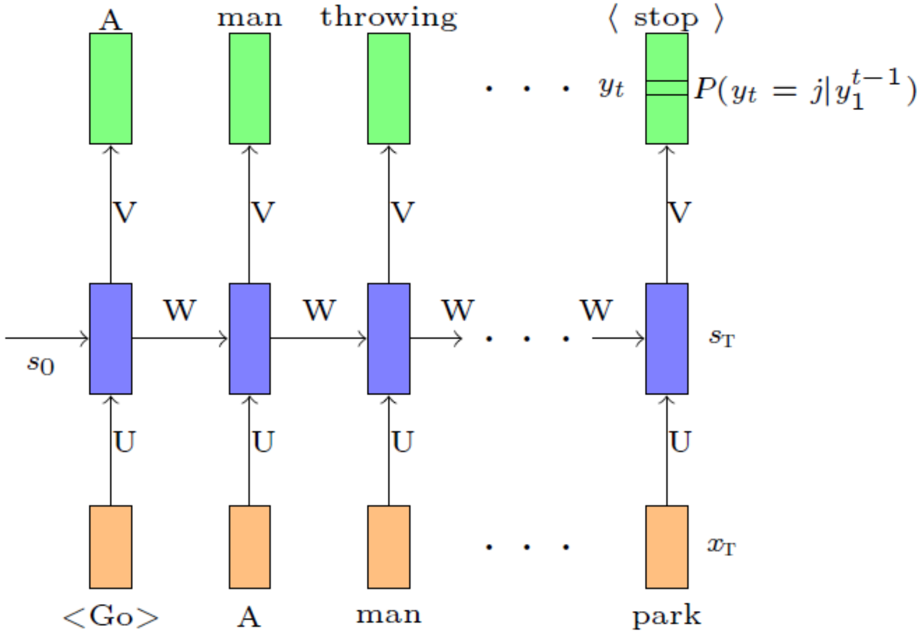


- Earlier we modeled $P(y_t|y_1^{t-1})$ as

$$P(y_t|y_1^{t-1}) = P(y_t = j|s_t)$$

Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

Encoder Decoder Models : Introduction



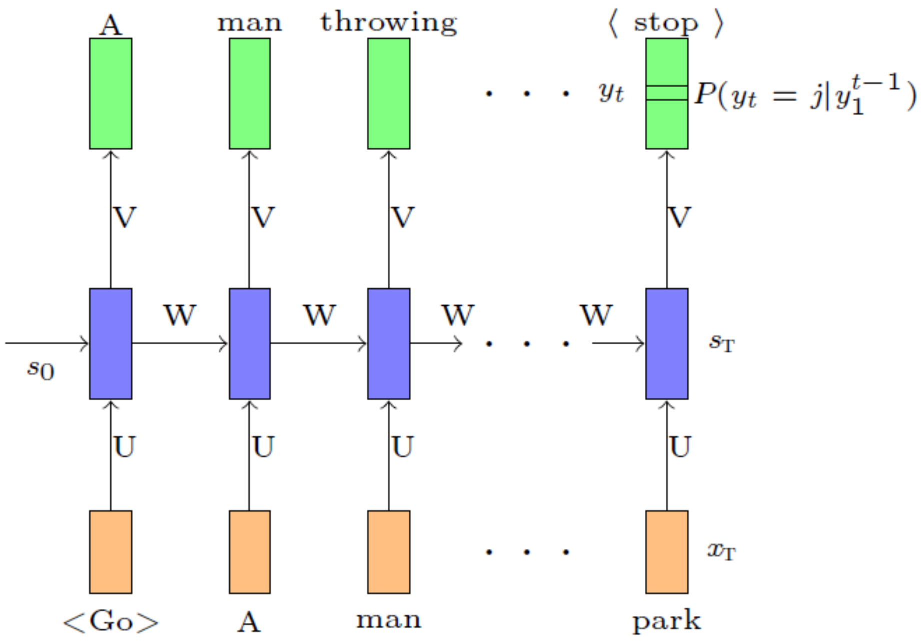
- Earlier we modeled $P(y_t|y_1^{t-1})$ as

$$P(y_t|y_1^{t-1}) = P(y_t = j|s_t)$$

- Where s_t was a state capturing all the previous words

Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

Encoder Decoder Models : Introduction



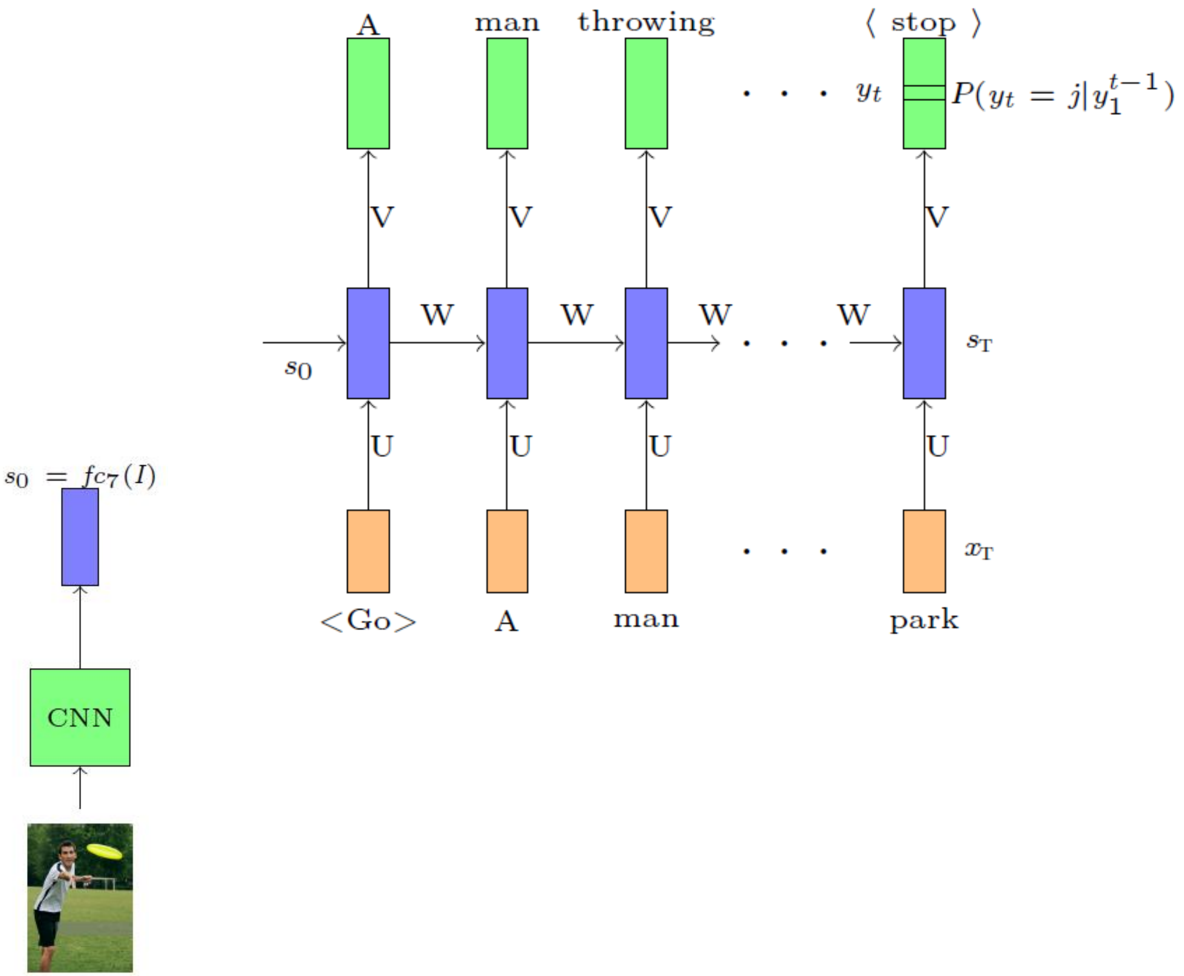
- Earlier we modeled $P(y_t|y_1^{t-1})$ as

$$P(y_t|y_1^{t-1}) = P(y_t = j|s_t)$$

- Where s_t was a state capturing all the previous words
- We could now model $P(y_t = j|y_1^{t-1}, I)$ as $P(y_t = j|s_t, f_{c7}(I))$

Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

Encoder Decoder Models : Introduction



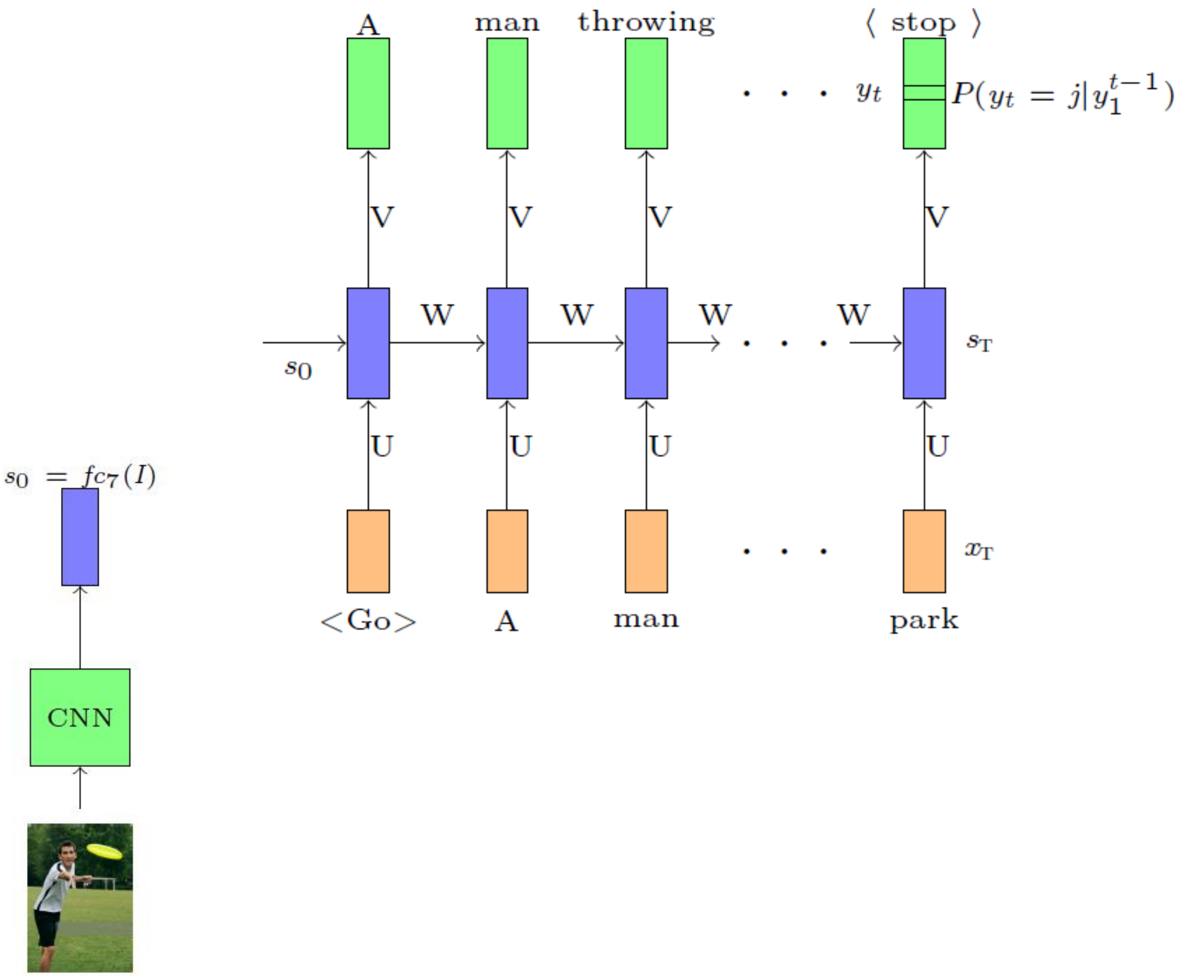
- Earlier we modeled $P(y_t|y_1^{t-1})$ as

$$P(y_t|y_1^{t-1}) = P(y_t = j|s_t)$$

- Where s_t was a state capturing all the previous words
- We could now model $P(y_t = j|y_1^{t-1}, I)$ as $P(y_t = j|s_t, f_{c_7}(I))$

Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

Encoder Decoder Models : Introduction



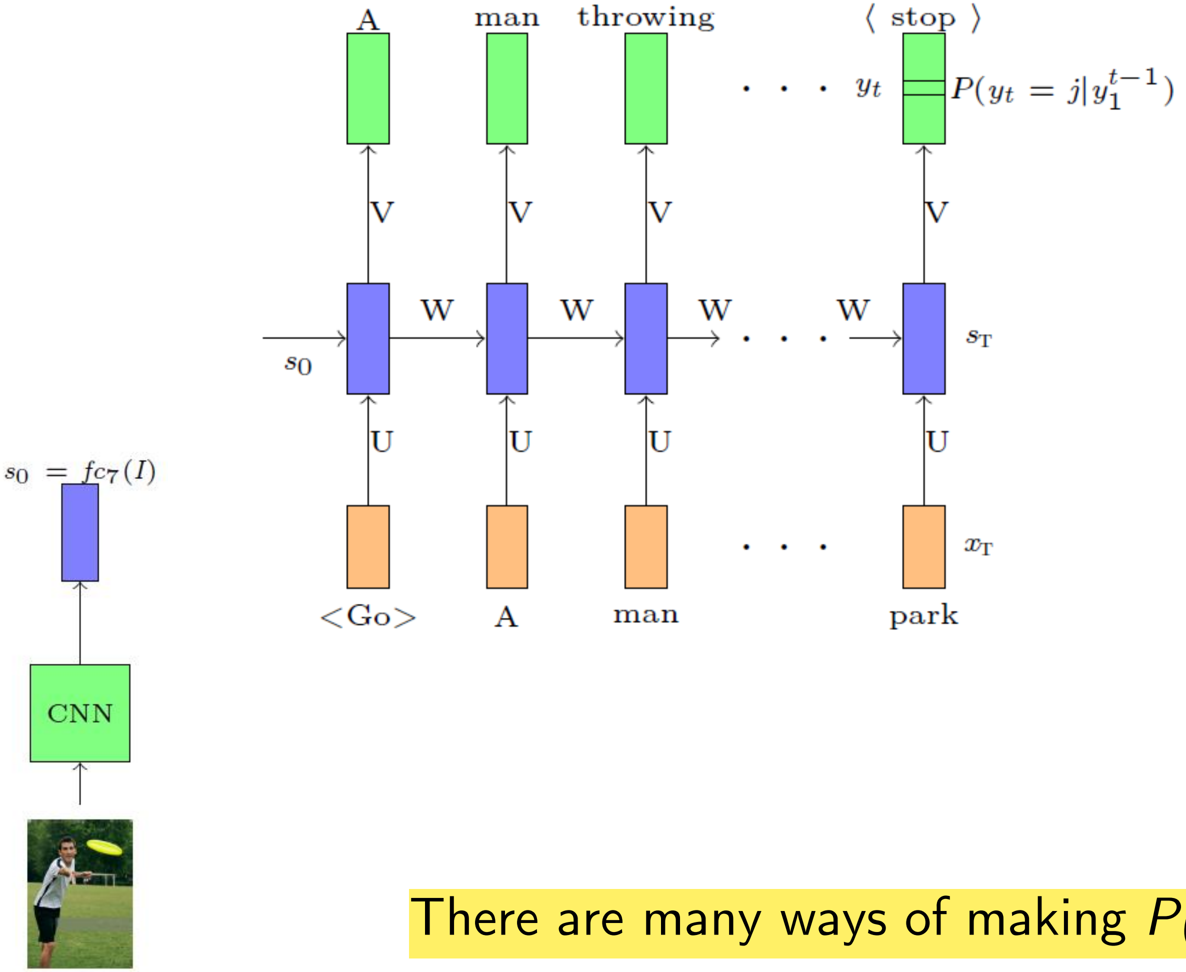
- Earlier we modeled $P(y_t|y_1^{t-1})$ as

$$P(y_t|y_1^{t-1}) = P(y_t = j|s_t)$$

- Where s_t was a state capturing all the previous words
- We could now model $P(y_t = j|y_1^{t-1}, I)$ as $P(y_t = j|s_t, f_{c_7}(I))$
- where $fc_7(I)$ is the representation obtained from the fc_7 layer of an image

Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

Encoder Decoder Models : Introduction



- Earlier we modeled $P(y_t|y_1^{t-1})$ as

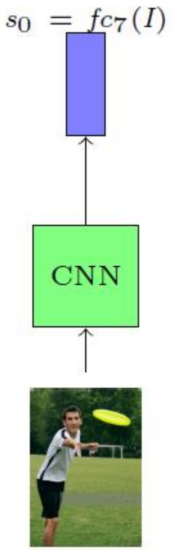
$$P(y_t|y_1^{t-1}) = P(y_t = j|s_t)$$

- Where s_t was a state capturing all the previous words
- We could now model $P(y_t = j|y_1^{t-1}, I)$ as $P(y_t = j|s_t, f_{c_7}(I))$
- where $fc_7(I)$ is the representation obtained from the fc_7 layer of an image

There are many ways of making $P(y_t = j)$ conditional on $f_{c7}(I)$

Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

Encoder Decoder Models : Introduction

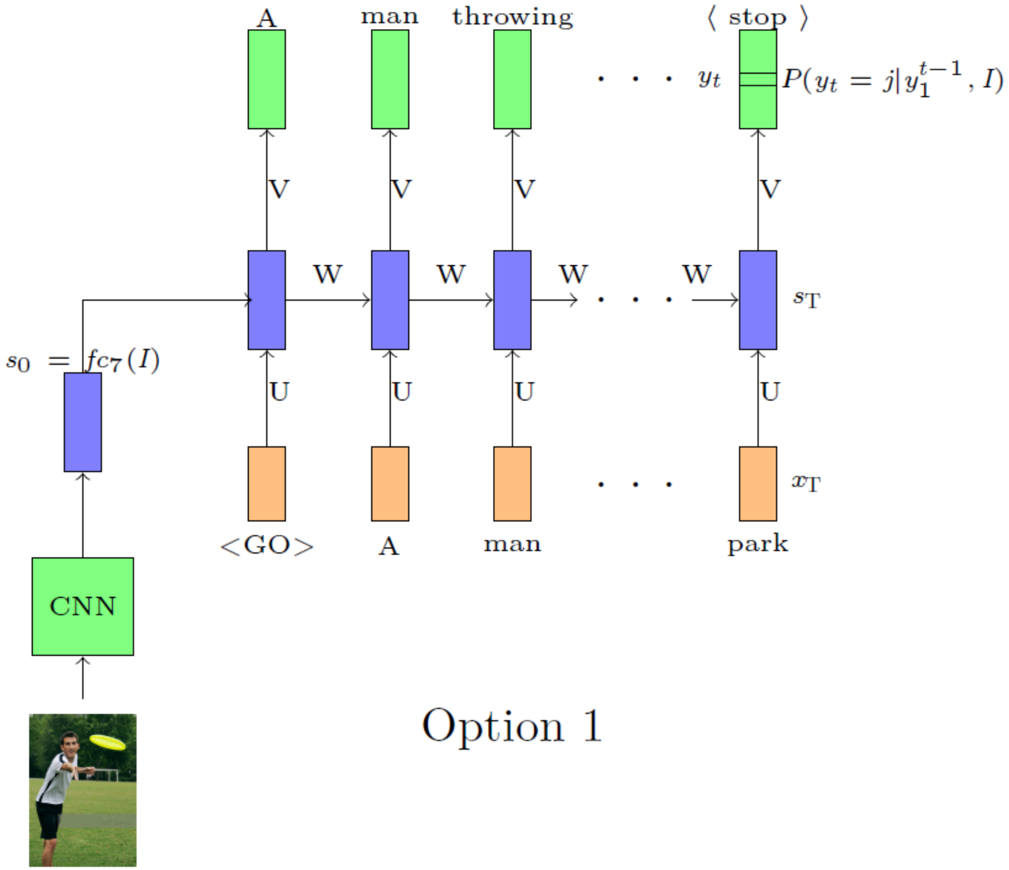


Option 1: Set $s_0 = f_{c_7}(I)$

Now s_0 and hence all subsequent s_t's depend on $f_{c_7}(I)$

Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

Encoder Decoder Models : Introduction



Option 1

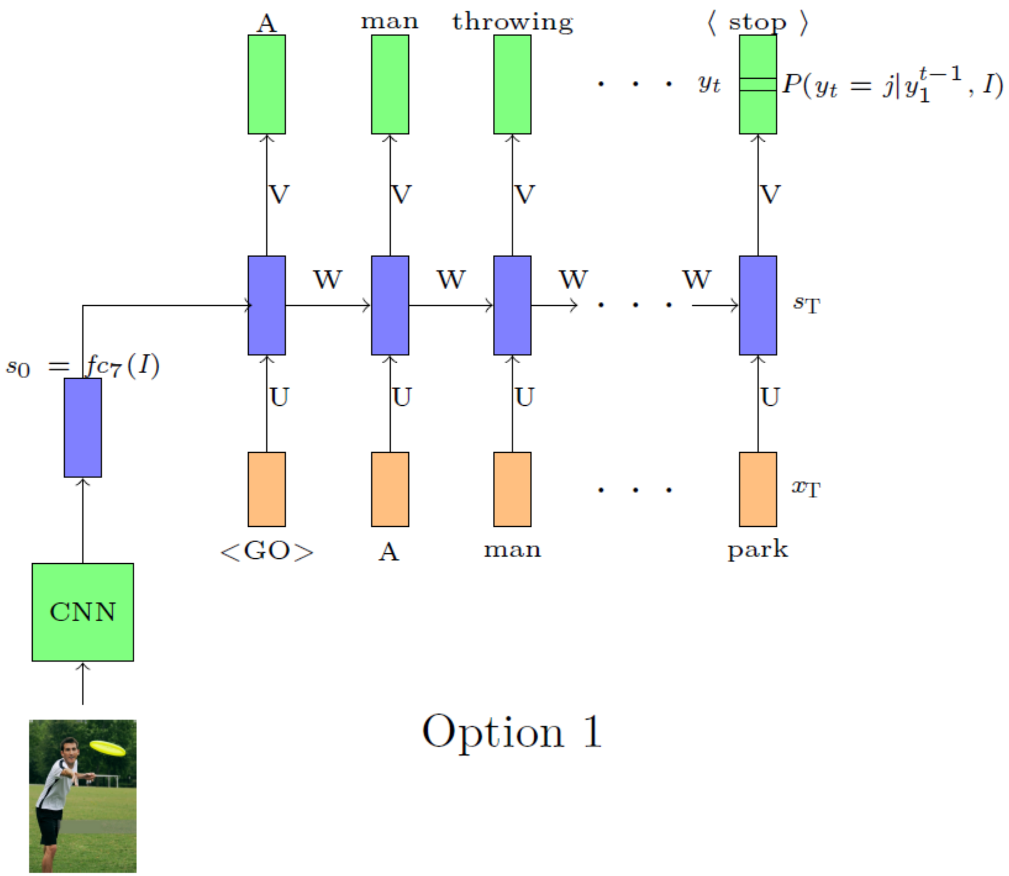
Option 1: Set $s_0 = f_{c7}(I)$

Now s_0 and hence all subsequent s_t's depend on $f_{c7}(I)$

We can thus say that $P(y_t = j)$ depends on $f_{c7}(I)$

In other words, we are computing $P(y_t = j | s_t, f_{c7}(I))$

Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras



Option 1

Option 1: Set $s_0 = f_{c7}(I)$

Now s_0 and hence all subsequent s_t's depend on $f_{c7}(I)$

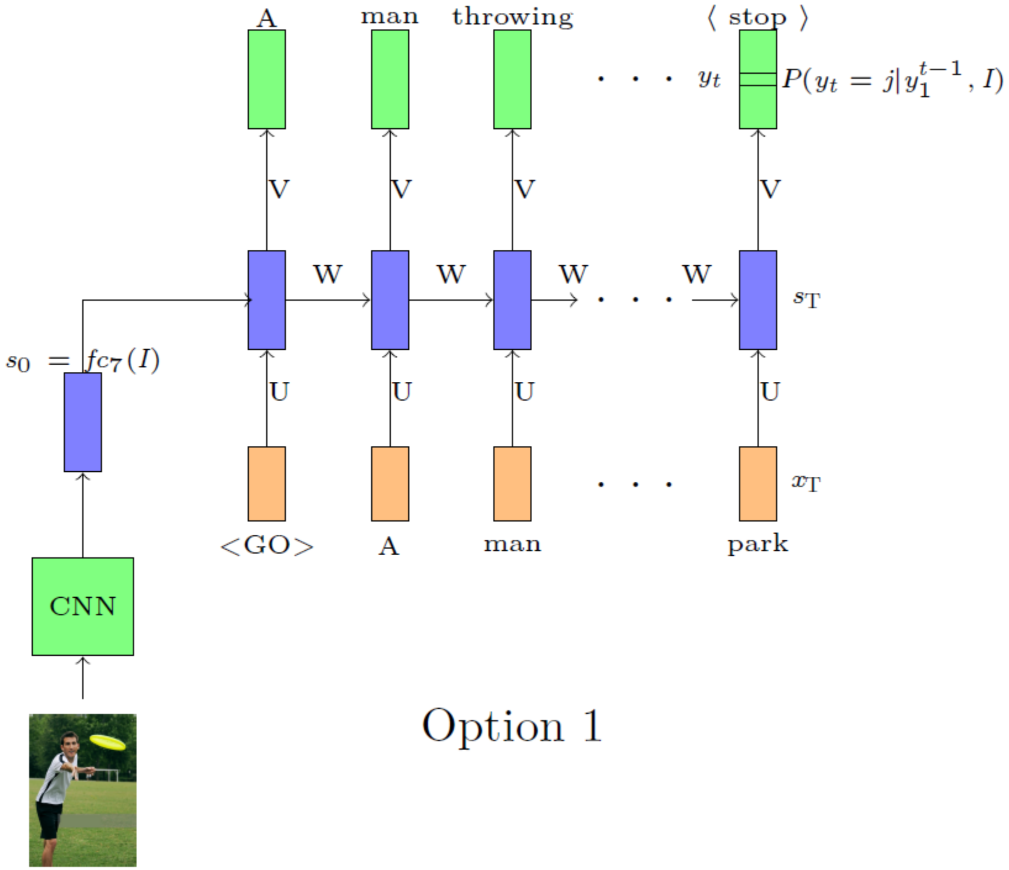
We can thus say that $P(y_t = j)$ depends on $f_{c7}(I)$

In other words, we are computing $P(y_t = j | s_t, f_{c7}(I))$

Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

Encoder Decoder Models : Introduction

Fully connected layer #7 in CNN (here, AlexNet)



Option 1

Option 1: Set $s_0 = f_{c7}(I)$

Now s_0 and hence all subsequent s_t's depend on $f_{c7}(I)$

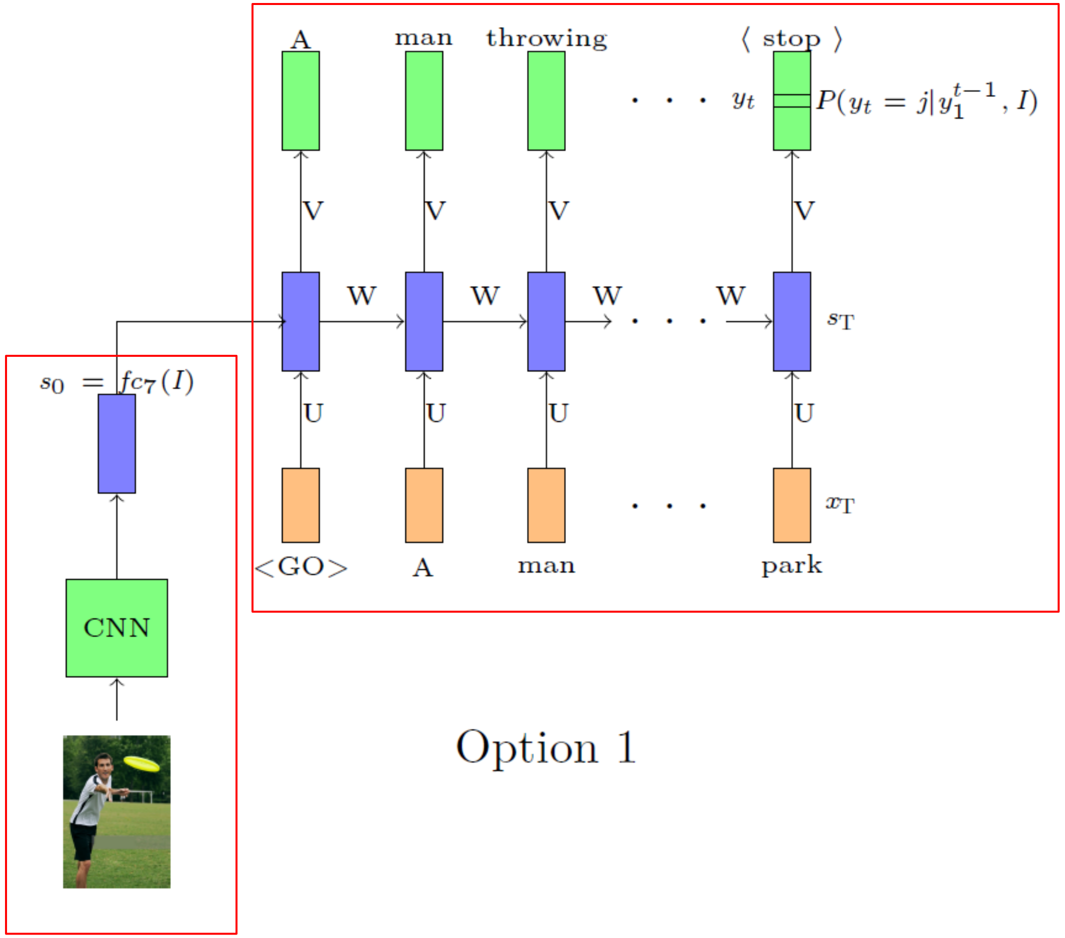
We can thus say that $P(y_t = j)$ depends on $f_{c7}(I)$

In other words, we are computing $P(y_t = j | s_t, f_{c7}(I))$

Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

Encoder Decoder Models : Introduction

Fully connected layer #7 in CNN (here, AlexNet)



Option 1

Option 1: Set $s_0 = f_{c7}(I)$

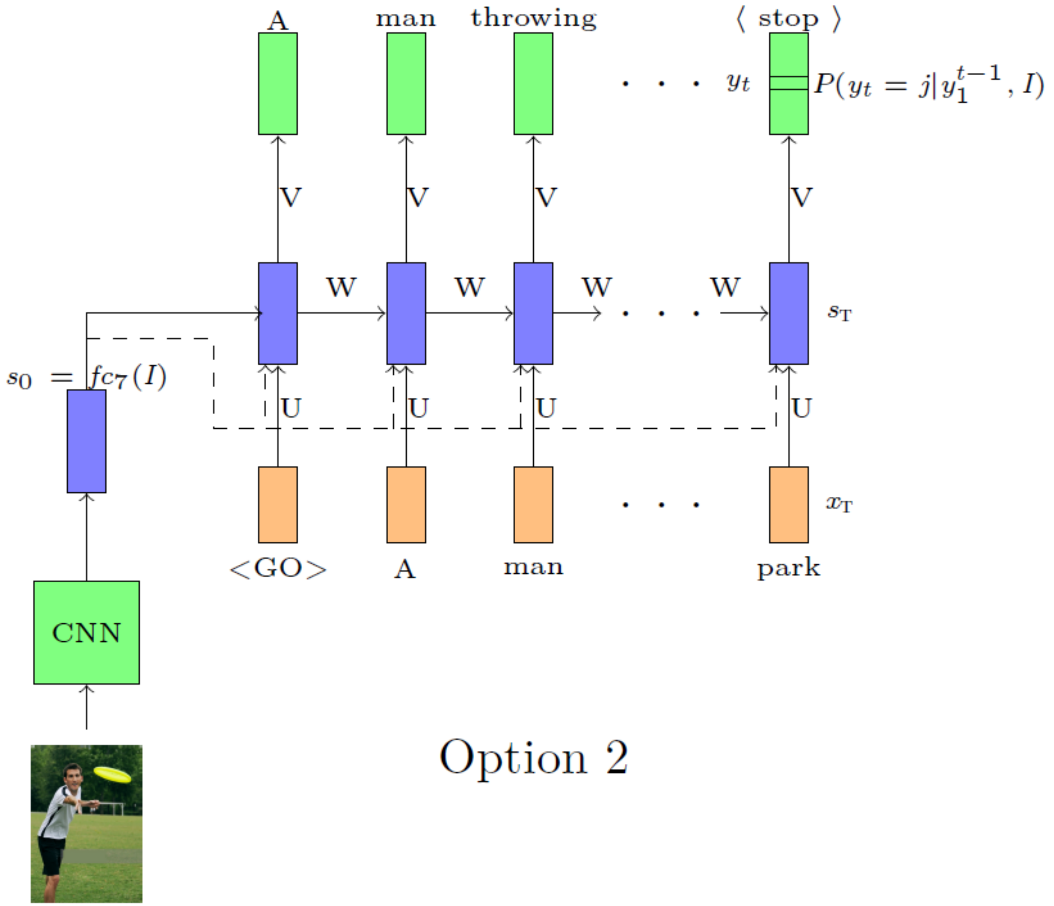
Now s_0 and hence all subsequent s_t's depend on $f_{c_7}(I)$

We can thus say that $P(y_t = j)$ depends on $f_{c_7}(I)$

In other words, we are computing $P(y_t = j | s_t, f_{c_7}(I))$

Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

Encoder Decoder Models : Introduction



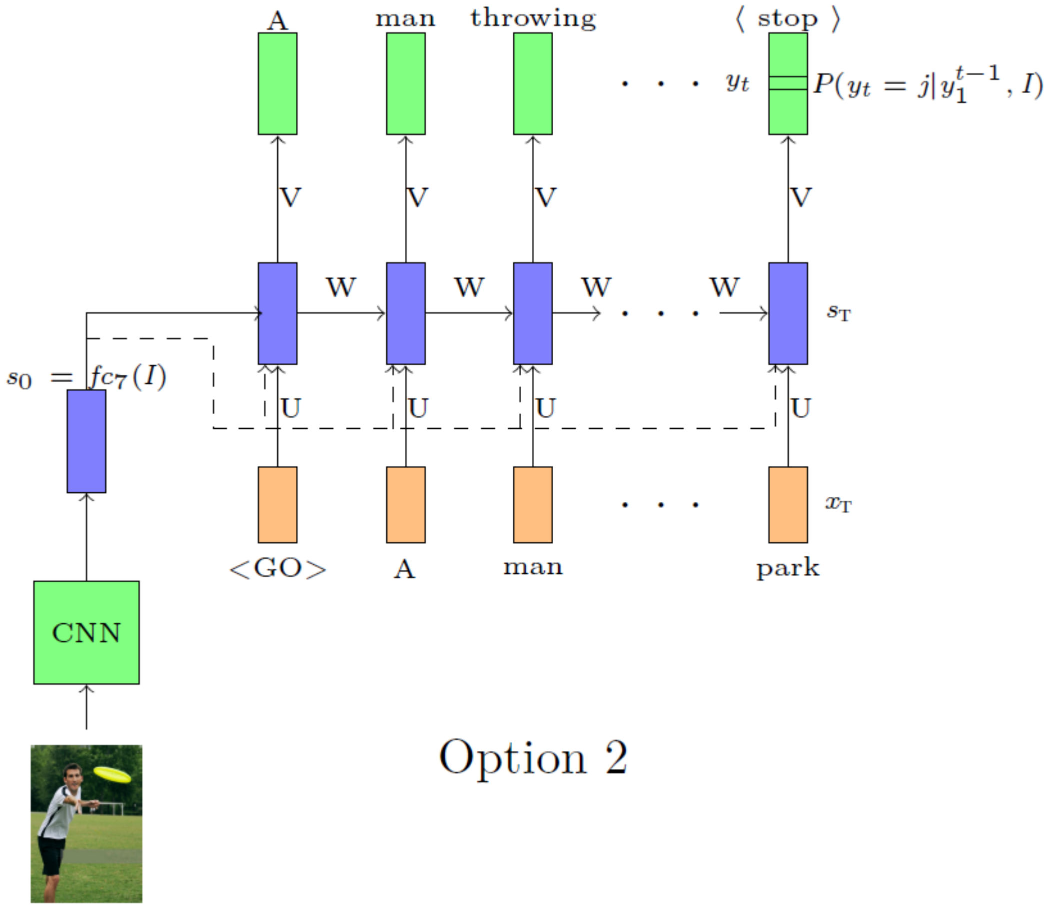
Option 2

Option 2: Another more explicit way of doing this is to compute

$$s_t = RNN(s_{t-1}, [x_t, f_{c_7}(I)])$$

Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

Encoder Decoder Models : Introduction



Option 2

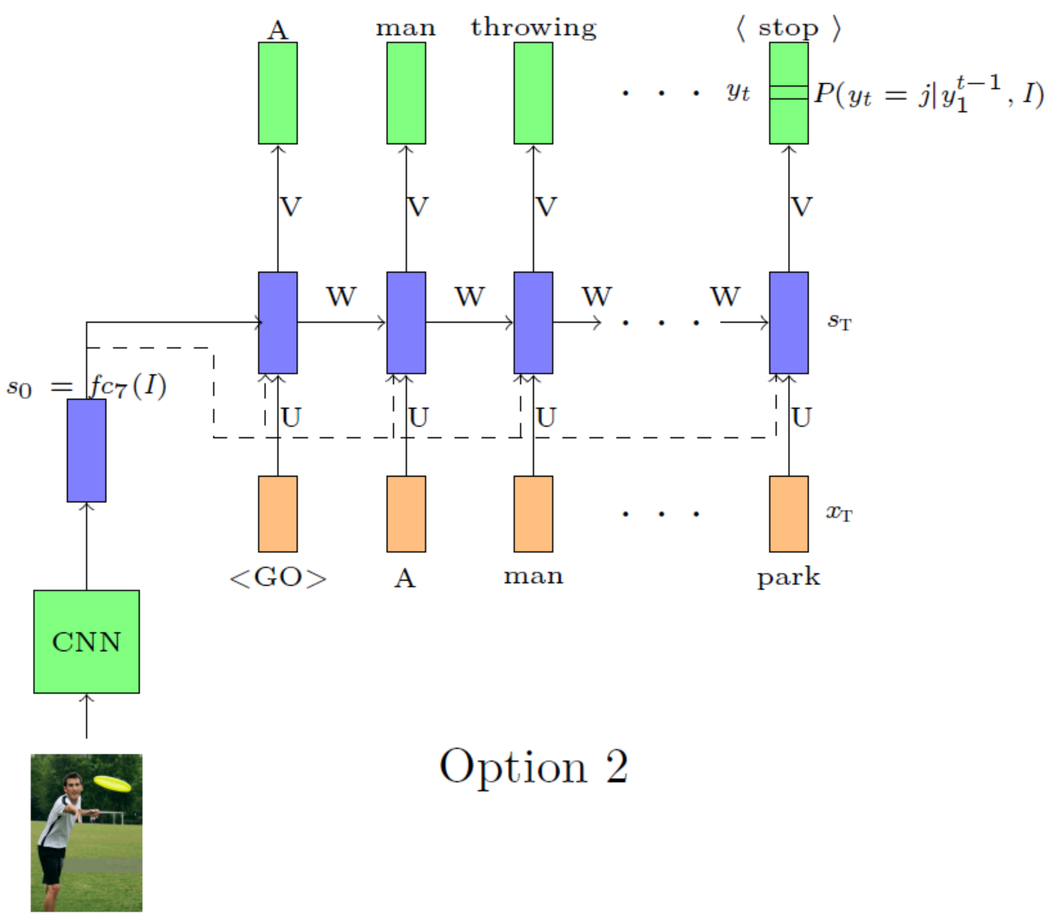
Option 2: Another more explicit way of doing this is to compute

$$s_t = RNN(s_{t-1}, [x_t, f_{c_7}(I)])$$

In other words we are explicitly using $f_{c_7}(I)$ to compute s_t and hence $P(y_t = j)$

Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

Encoder Decoder Models : Introduction



Option 2

Option 2: Another more explicit way of doing this is to compute

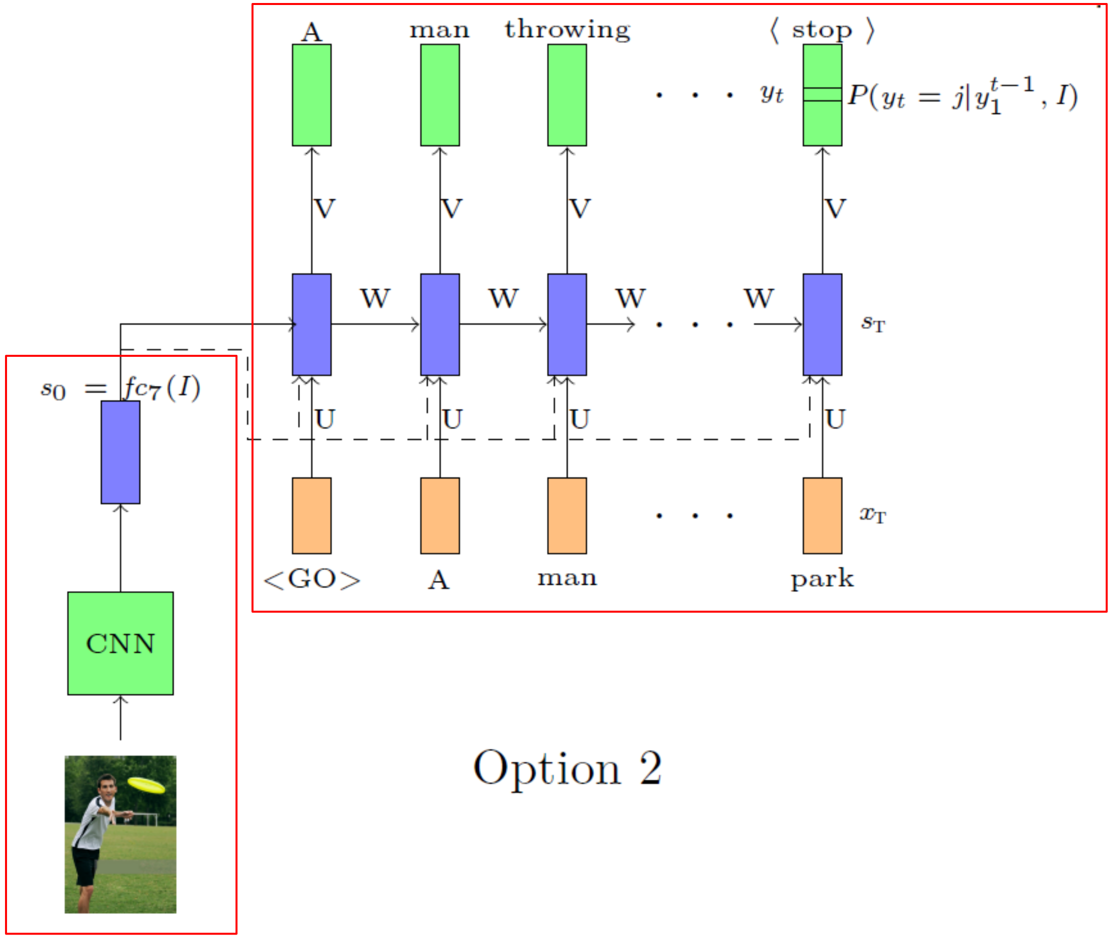
$$s_t = RNN(s_{t-1}, \boxed{[x_t, f_{c_7}(I)]})$$

In other words we are explicitly using $f_{c_7}(I)$ to compute s_t and hence $P(y_t = j)$

The o/p of CNN is concatenated with the input embedding and then fed to the RNN as input at each time step.

Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

Encoder Decoder Models : Introduction



Option 2

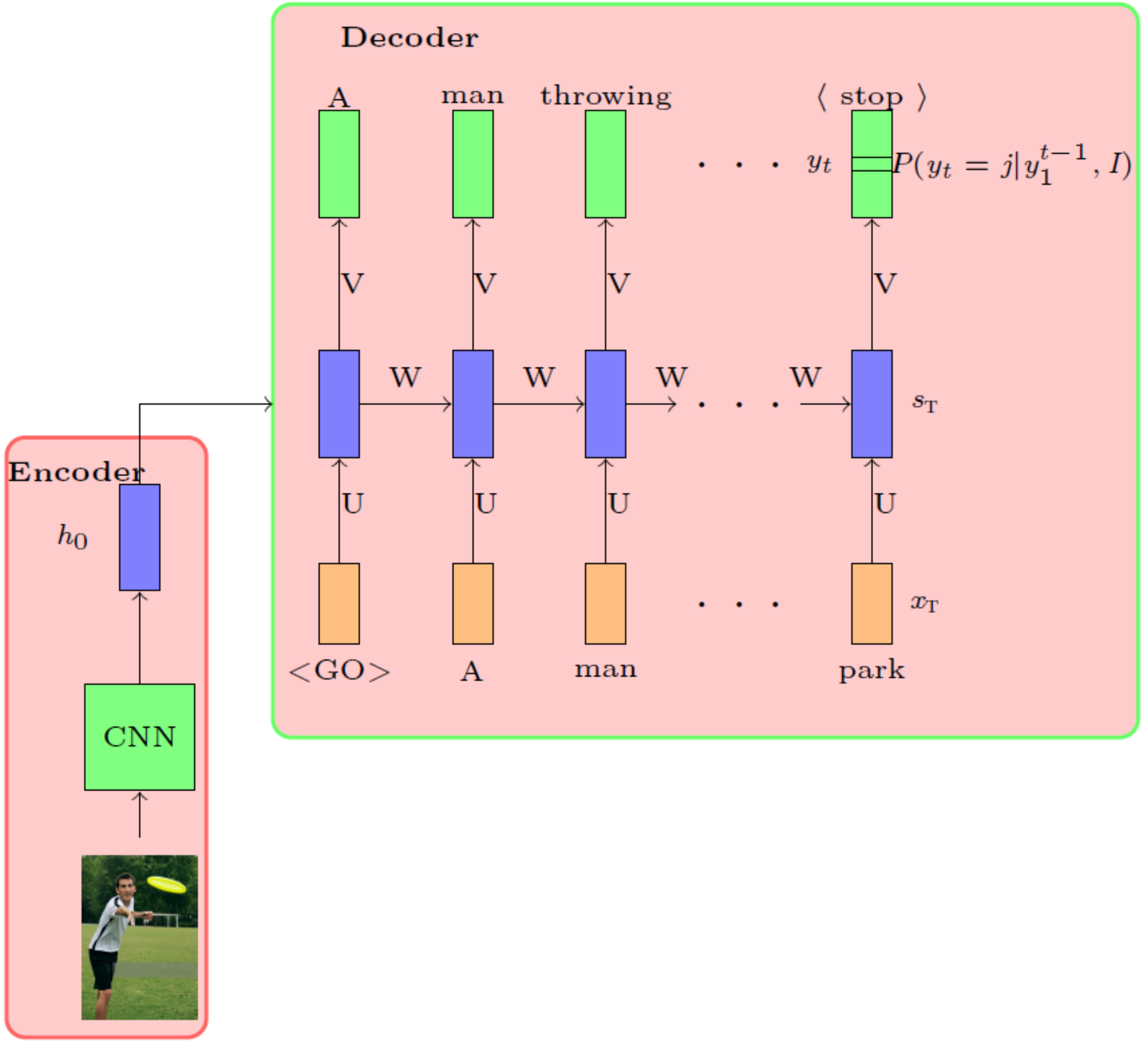
Option 2: Another more explicit way of doing this is to compute

$$s_t = RNN(s_{t-1}, \boxed{[x_t, f_{c_7}(I)]})$$

In other words we are explicitly using $f_{c7}(I)$ to compute s_t and hence $P(y_t = j)$

The o/p of CNN is concatenated with the input embedding and then fed to the RNN as input at each time step.

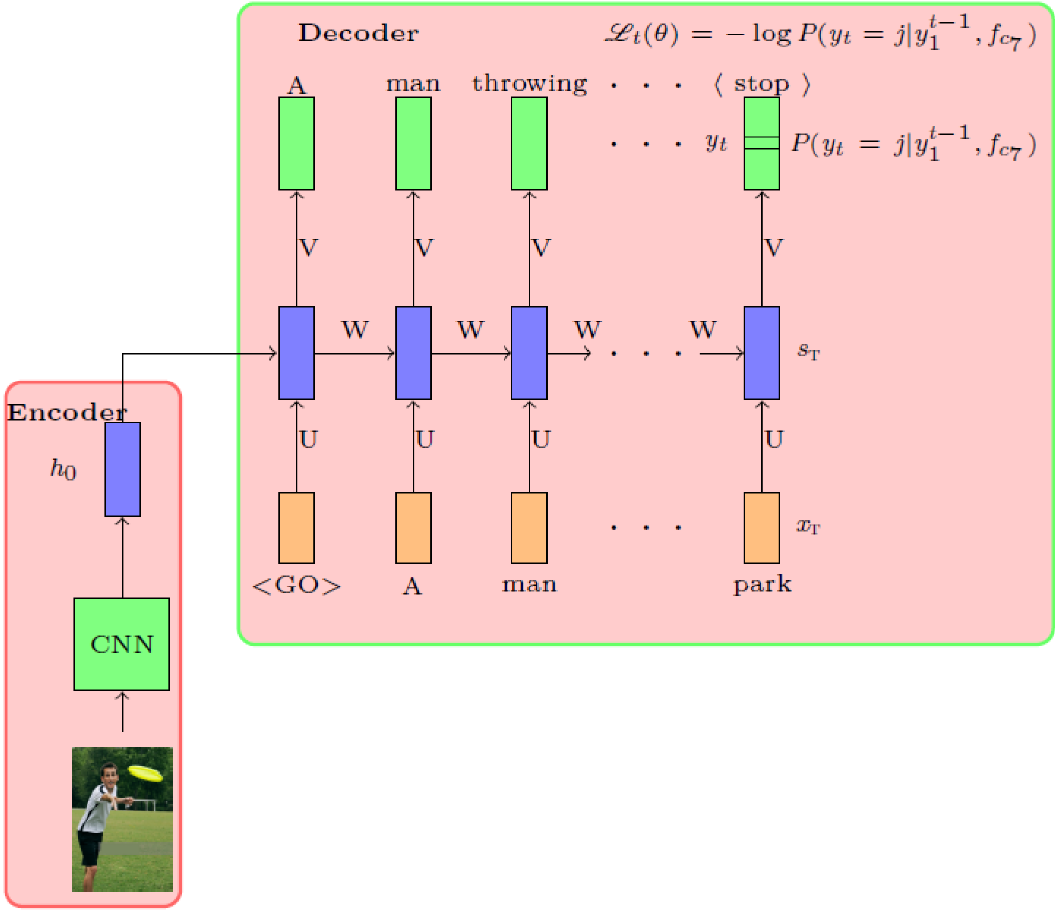
Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras



- This is typical **encoder decoder** architecture

Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

Applications of Encoder Decoder Models: Image Captioning



- **Task:** Image captioning
- **Data:** $\{x_i = image_i,\ y_i = caption_i\}_{i=1}^{N}$
- **Model:**
  - **Encoder:**

$$s_0 = CNN(x_i)$$

  - **Decoder:**

$$s_t = RNN(s_{t-1}, \boxed{e(\hat{y}_{t-1})})$$

Represents the input embedding of the output obtained at time step t-1

$$P(y_t|y_1^{t-1}, I) = softmax(Vs_t + b)$$

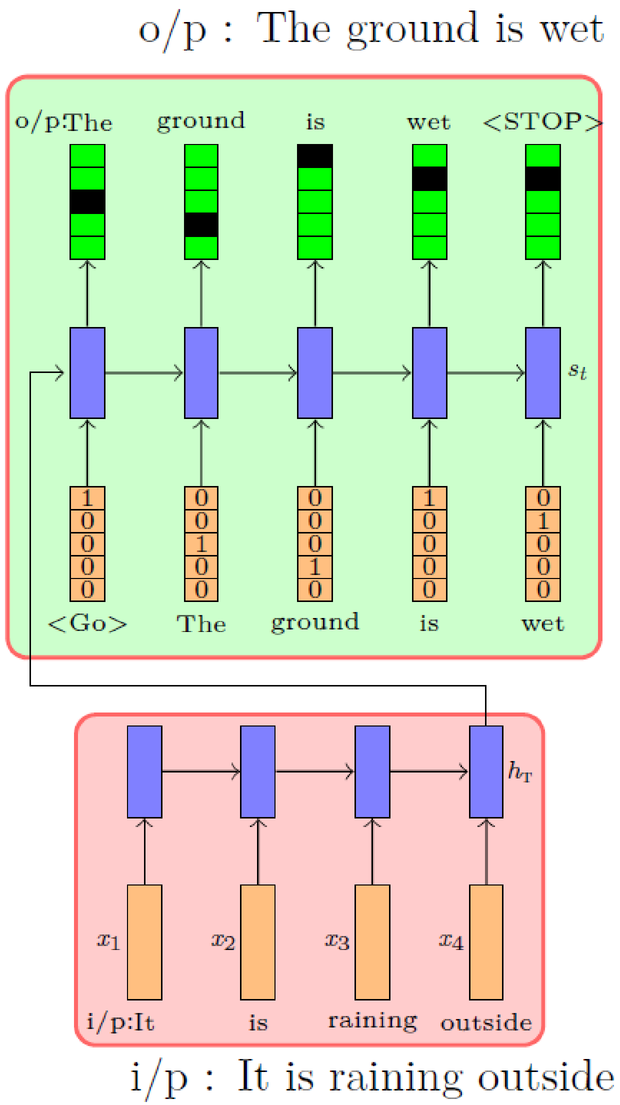
- **Parameters:** U_{dec}, V, W_{dec}, W_{conv}, b
- **Loss:**

$$\mathscr{L}(\theta) = \sum_{i=1}^{T} \mathscr{L}_t(\theta) = -\sum_{t=1}^{T} \log P(y_t = \ell_t|y_1^{t-1}, I)$$

- **Algorithm:** Gradient descent with backpropagation

Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

Applications of Encoder Decoder Models: Textual Entailment



Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

- **Task:** Textual entailment
- **Data:** $\{x_i = premise_i,\ y_i = hypothesis_i\}_{i=1}^{N}$
- **Model (Option 1):**
  - **Encoder:**

$$h_t = RNN(h_{t-1}, x_{it})$$

  - **Decoder:**

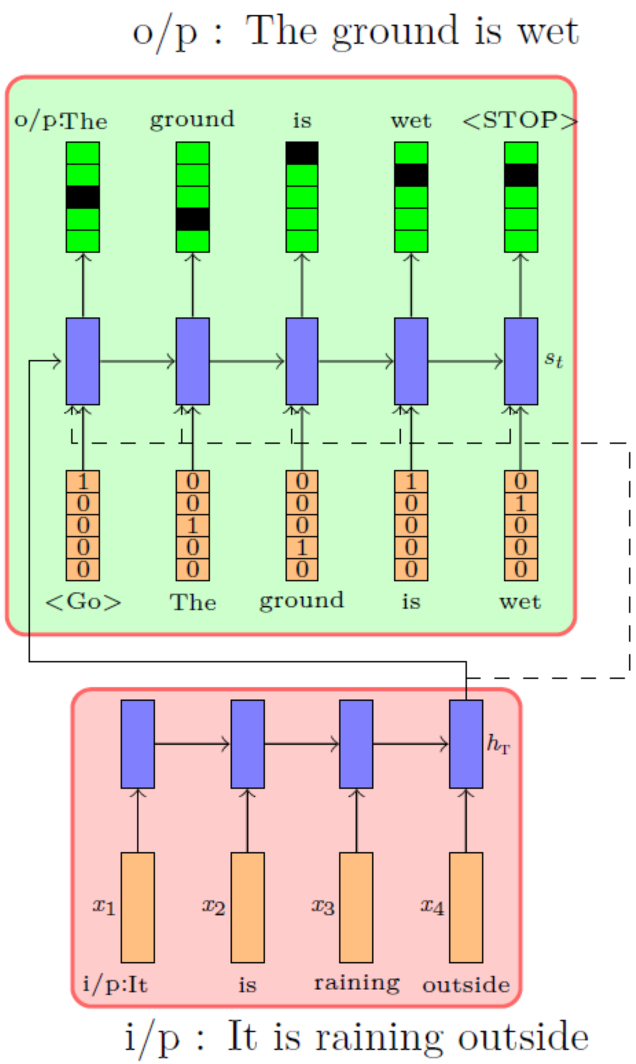
$$s_0 = h_T \quad (T \text{ is length of input})$$

$$s_t = RNN(s_{t-1}, e(\hat{y}_{t-1}))$$

$$P(y_t|y_1^{t-1}, x) = softmax(Vs_t + b)$$

- **Parameters:** $U_{dec},\ V,\ W_{dec},\ U_{enc},\ W_{enc},\ b$
- **Loss:**

$$\mathscr{L}(\theta) = \sum_{i=1}^{T} \mathscr{L}_t(\theta) = -\sum_{t=1}^{T} \log P(y_t = \ell_t | y_1^{t-1}, x)$$



- **Task:** Textual entailment
- **Data:** $\{x_i = premise_i,\ y_i = hypothesis_i\}_{i=1}^{N}$
- **Model (Option 2):**
  - **Encoder:**
    $$h_t = RNN(h_{t-1}, x_{it})$$
  - **Decoder:**
    $$s_0 = h_T \quad (T\ is\ length\ of\ input)$$
    $$s_t = RNN(s_{t-1}, [h_T, e(\hat{y}_{t-1})])$$
    $$P(y_t|y_1^{t-1}, x) = softmax(Vs_t + b)$$
- **Parameters:** $U_{dec}, V, W_{dec}, U_{enc}, W_{enc}, b$
- **Loss:**
  $$\mathscr{L}(\theta) = \sum_{i=1}^{T} \mathscr{L}_t(\theta) = -\sum_{t=1}^{T} \log P(y_t = \ell_t|y_1^{t-1}, x)$$
- **Algorithm:** Gradient descent with backpropagation

Applications of Encoder Decoder Models: Transliteration

o/p : इंडिया

i/p : I N D I A

Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

O/p: White

- **Task:** Image Question Answeing
- **Data:** $\{x_i = \{I, q\}_i,\ y_i = Answer_i\}_{i=1}^{N}$
- **Model:**
  - **Encoder:**

    $$\hat{h}_I = CNN(I),\ \tilde{h}_t = RNN(\tilde{h}_{t-1}, q_{it})$$

    $$s = [\tilde{h}_T; \hat{h}_I]$$
  - **Decoder:**

    $$P(y|q, I) = softmax(Vs + b)$$
- **Parameters:** $V,\ b,\ U_q,\ W_q,\ W_{conv}, b$
- **Loss:**

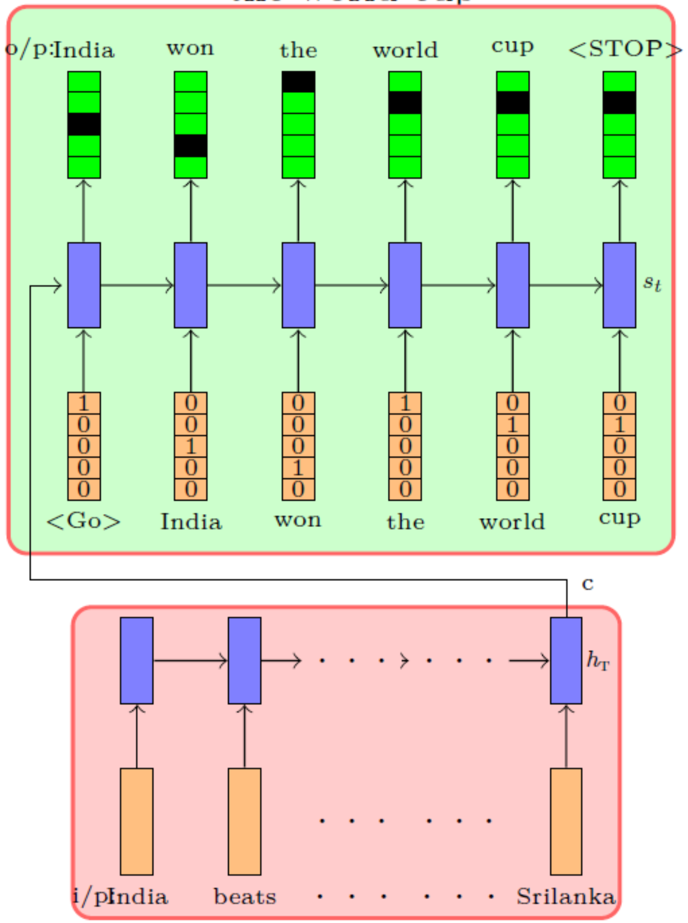
  $$\mathscr{L}(\theta) = -\log P(y = \ell | I, q)$$
- **Algorithm:** Gradient descent with backpropagation

Question: What is the bird's color

Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

Applications of Encoder Decoder Models: Document Summarization

o/p : India won
the world cup

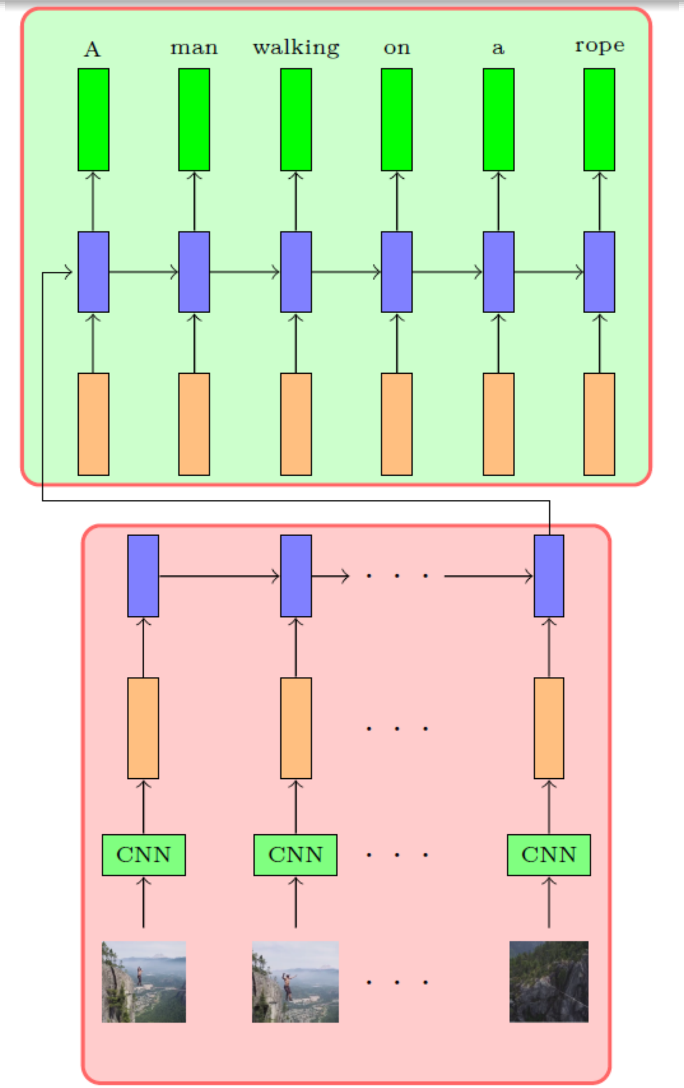


i/p : India beats Srilanka to win ICC WC 2011.
Dhoni and Gambhir's half centuries help beat SL

- **Task:** Document Summarization
- **Data:** $\{x_i = Document_i, \quad y_i = Summary_i\}_{i=1}^{N}$
- **Model:**
  - **Encoder:**
  $$h_t = RNN(h_{t-1}, x_{it})$$
  - **Decoder:**
  $$s_0 = h_T$$
  $$s_t = RNN(s_{t-1}, e(\hat{y}_{t-1}))$$
  $$P(y_t|y_1^{t-1}, x) = softmax(Vs_t + b)$$
- **Parameters:** $U_{dec}, V, W_{dec}, U_{enc}, W_{enc}, b$
- **Loss:**
$$\mathscr{L}(\theta) = \sum_{i=1}^{T} \mathscr{L}_t(\theta) = -\sum_{t=1}^{T} \log P(y_t = \ell_t | y_1^{t-1}, x)$$
- **Algorithm:** Gradient descent with backpropagation

Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

Applications of Encoder Decoder Models: Video Captioning



- **Task:** Video Captioning
- **Data:** $\{x_i = video_i, \; y_i = desc_i\}_{i=1}^{N}$
- **Model:**
  - **Encoder:**

$$h_t = RNN(h_{t-1}, CNN(x_{it}))$$

  - **Decoder:**

$$s_0 = h_T$$
$$s_t = RNN(s_{t-1}, e(\hat{y}_{t-1}))$$
$$P(y_t | y_1^{t-1}, x) = softmax(Vs_t + b)$$

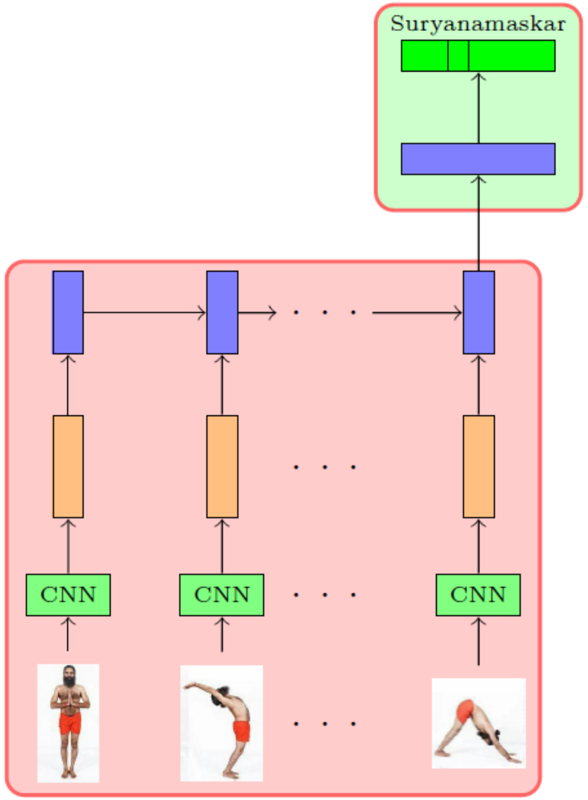
- **Parameters:** U_{dec}, W_{dec}, V, b, W_{conv}, U_{enc}, W_{enc}, b
- **Loss:**

$$\mathscr{L}(\theta) = \sum_{i=1}^{T} \mathscr{L}_t(\theta) = -\sum_{t=1}^{T} \log P(y_t = \ell_t | y_1^{t-1}, x)$$

- **Algorithm:** Gradient descent with backpropagation

Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

o/p: Surya Namaskar

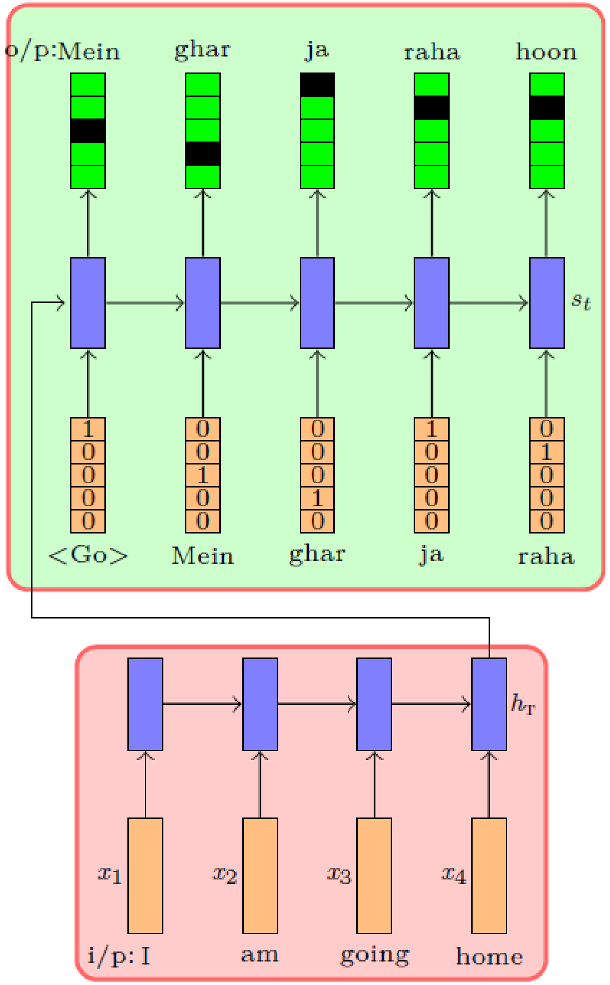


- **Task:** Video Classification
- **Data:** $\{x_i = Video_i, \ y_i = Activity_i\}_{i=1}^{N}$
- **Model:**
  - **Encoder:**
    $$h_t = RNN(h_{t-1}, CNN(x_{it}))$$
  - **Decoder:**
    $$s = h_T$$
    $$P(y|I) = softmax(Vs + b)$$
- **Parameters:** $V, \ b, \ W_{conv}, \ U_{enc}, \ W_{enc}, b$
- **Loss:**
  $$\mathscr{L}(\theta) = -\log P(y = \ell | Video)$$
- **Algorithm:** Gradient descent with backpropagation

Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

Applications of Encoder Decoder Models: Machine Translation

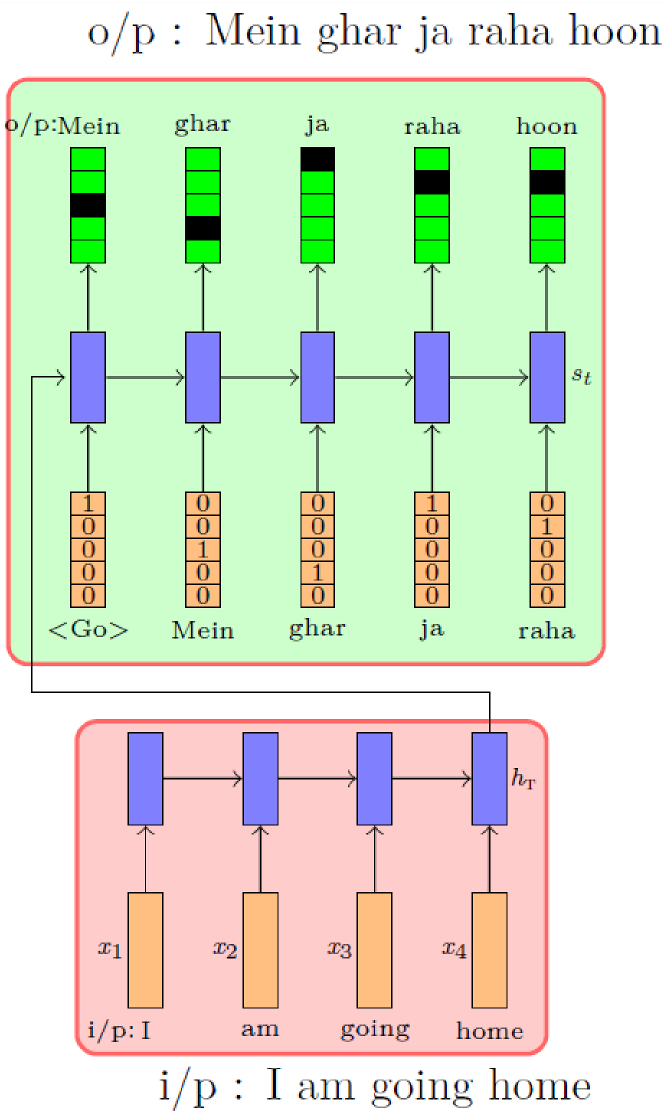
o/p : Mein ghar ja raha hoon



i/p : I am going home

Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

Applications of Encoder Decoder Models: Machine Translation



- **Task:** Machine translation
- **Data:** $\{x_i = source_i,\ y_i = target_i\}_{i=1}^{N}$
- **Model (Option 1):**
  - **Encoder:**

$$h_t = RNN(h_{t-1}, x_{it})$$

  - **Decoder:**

$$s_0 = h_T \quad (T \text{ is length of input})$$

$$s_t = RNN(s_{t-1}, e(\hat{y}_{t-1}))$$

$$P(y_t|y_1^{t-1}, x) = softmax(Vs_t + b)$$

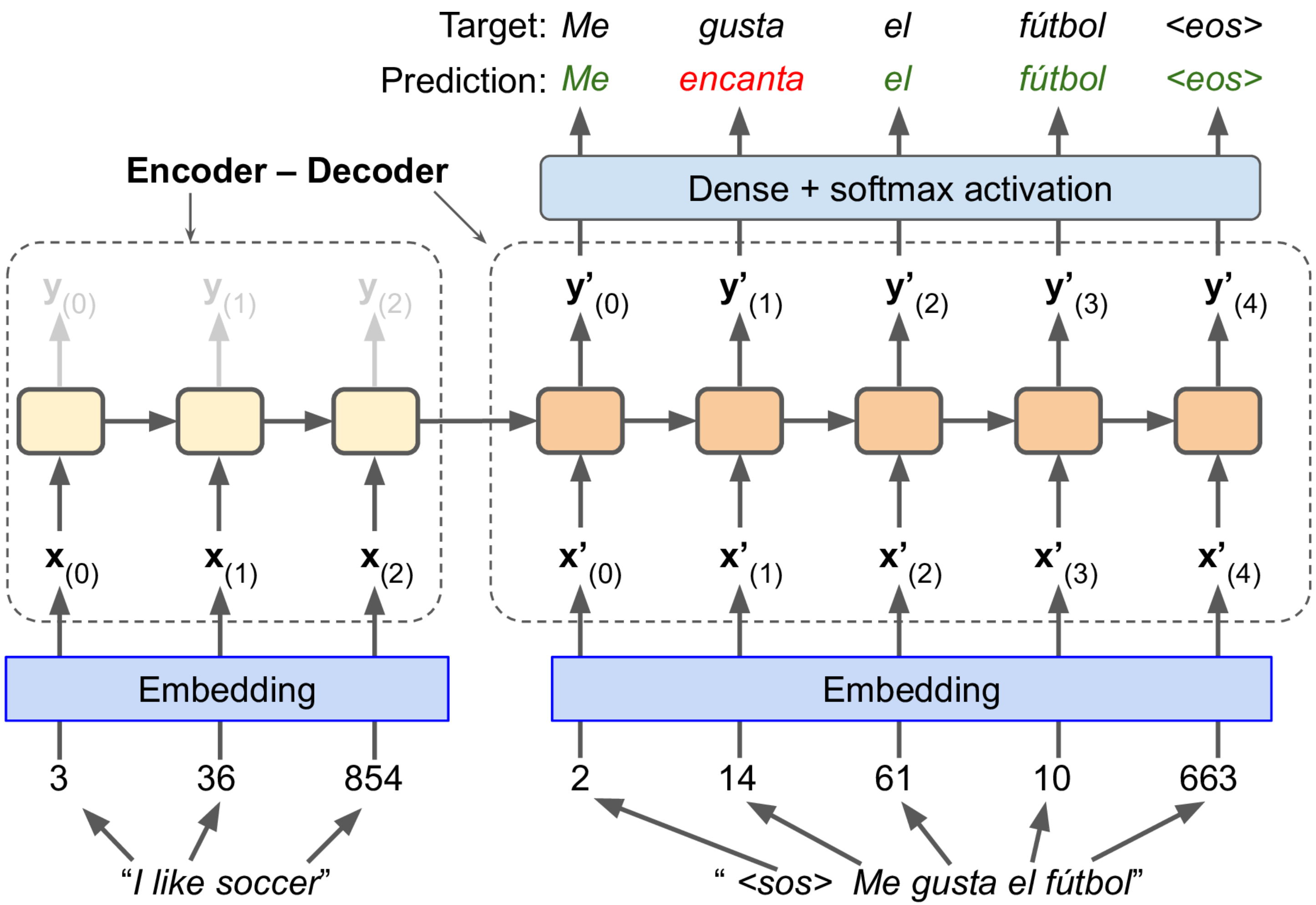
- **Parameters:** $U_{dec},\ V,\ W_{dec},\ U_{enc},\ W_{enc},\ b$
- **Loss:**

$$\mathscr{L}(\theta) = \sum_{i=1}^{T} \mathscr{L}_t(\theta) = -\sum_{t=1}^{T} \log P(y_t = \ell_t | y_1^{t-1}, x)$$

- **Algorithm:** Gradient descent with backpropagation

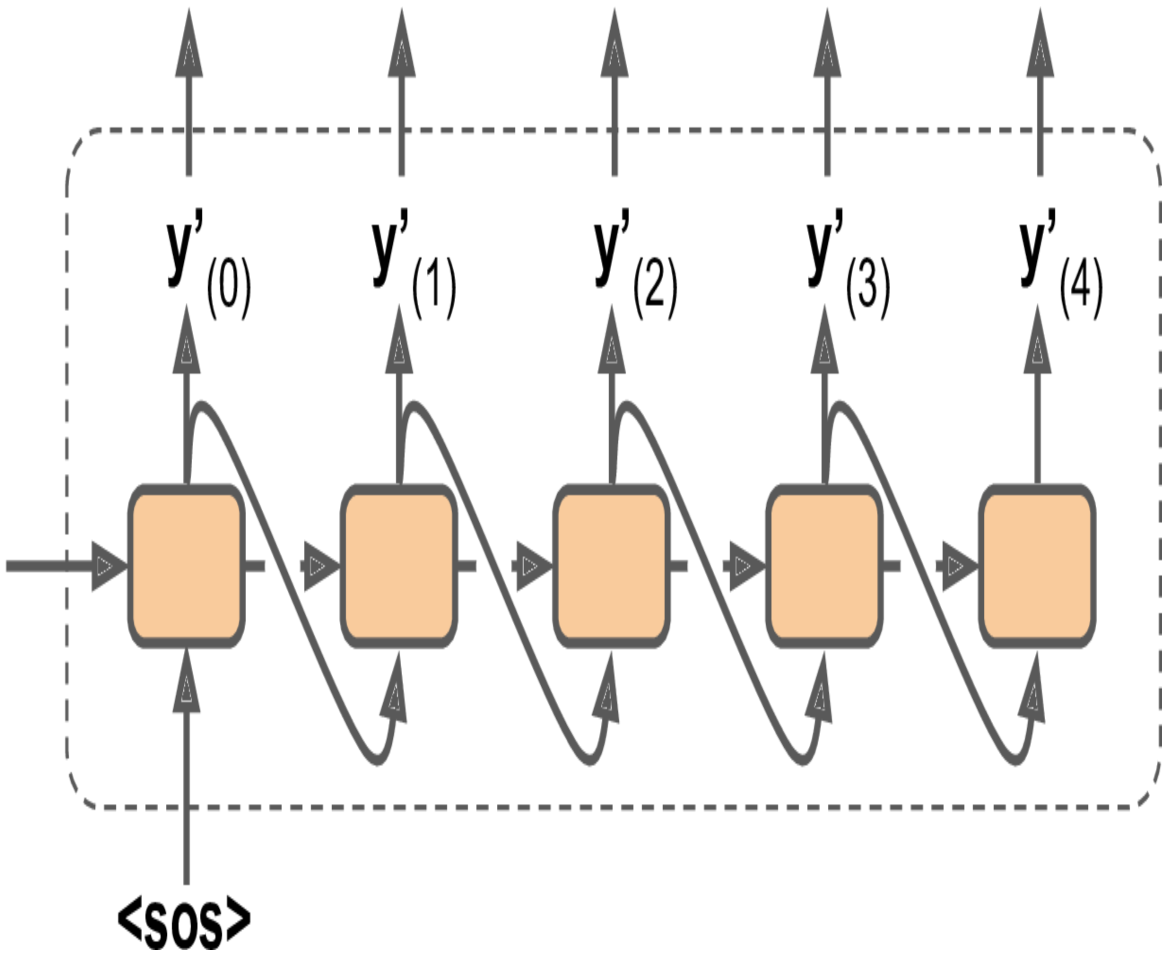
Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

Applications of Encoder Decoder Models: Machine Translation



Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

Encoder-Decoder Network for Machine Translation – in Inference Phase



- Process of the Decoder in the test period-
  - The initial states of the decoder are set to the final states of the encoder.
  - process single word at every time step
  - The output produced at each time step is fed as input in the next time step
  - The internal states generated after every time step is fed as the initial states of the next time step.

Greedy Search

$$Y_{\text{argmax}(y1,..yn)} = P(\underbrace{y1,\ y2,\ y3\ \ldots,\ yn}_{\text{Output seq}} \mid \underbrace{x1, x2.\ \ldots.,\ xm}_{\text{Input seq}})$$

- The greedy search algorithm involves selecting the most likely word at each step to generate a translation.

- **Problem:** Greedy search only considers the most likely word at each step, without considering the overall probability of the entire sentence.
- This can lead to getting stuck in local optima, where a suboptimal word choice at an early step prevents the algorithm from finding a better translation later on.

  - ❑ Jane visitr l'Afrique en Septembre (x:French)
  - ❑ Jane is visiting Africa in September (y: English)
  - ❑ Jane is going to be visiting Africa in September September (y: English)

Beam Search

READ BY COPY PASTING FROM THIS AND ALSO SCROLL DOWN TO GET WHAT IT MAINLY IS

Yes, **greedy search** and **beam search** are decoding strategies that are typically used during the **inference (testing) phase** of sequence-to-sequence models, such as those used in machine translation, text generation, or any task where the model needs to generate a sequence of tokens (e.g., words or characters). These strategies determine how the model selects the next token at each time step, balancing between generating coherent sequences and exploring multiple possibilities.

Greedy Search and Beam Search in Detail:

1. **Greedy Search**:
- **How it works**: Greedy search is a simple decoding strategy where, at each time step, the model chooses the token with the highest probability as the next token. It doesn't look ahead or consider other possibilities—it just picks the "best" option at each step based on the model's predictions.

- **Process**:
  1. The model generates probabilities over the vocabulary for the first token.
  2. The token with the highest probability is chosen as the output for that time step.

3. This token is fed back into the model as input for the next time step.

4. The process is repeated until the model generates an end-of-sequence token (like `<EOS>`).

- **Example**:

Let's say you are using an RNN for text generation. At each time step, the model predicts the next word.
  - Time step 1: "The cat is" (model predicts probability distribution over the vocabulary). If "on" has the highest probability, it's selected.
  - Time step 2: "The cat is on" (the process continues).

- **Pros**:
  - Simple and fast.
  - Low computational cost.

- **Cons**:
  - Greedy search can miss globally optimal solutions because it only considers local (immediate) probabilities. This can lead to suboptimal sequences if a lower-probability token at an earlier step could lead to a better overall sequence.
  - Does not explore alternative sequences, potentially limiting fluency and accuracy in sequence generation.

- **Use Cases**: Greedy search is often used in simpler scenarios where a fast, approximate result is sufficient, but it's typically not ideal for tasks like machine translation or text generation where more complex, coherent output is desired.

2. **Beam Search**:

"Comment vas-tu?" – How are you?

"Comment vas-tu jouer?" – How will you play?

Beam Search

- Keeps track of a short list of the k most promising sentences
- Parameter k is beam width
- At each decoder step it tries to extend them by 1 word
- **Step 1:** First word could be "How" (Prob: 75%), "what" or "you"
- **Step 2:** 3 copies of model made to find the next word
- **Step 3:** First model will look for next word after "How" by computing conditional probabilities
- Output could be "will (Prob:36%) "are", "do",
- **Step 4:** Compute the probabilities of each of the two word sentences the model considered and so on
- Advantage is that good translation for fairly short sentences can be obtained
- Disadvantage – really bad at translating long sentences.
- Due to limited short term memory, lead to **ATTENTION MECHANISM**

Attention Mechanism (Bahdanau et al , 2014)

- Technique that allowed the decoder to focus on the appropriate words at each time steps
- Ensures that the path to from an input word to its translation is much shorter
- For instance , at the time step where the decoder needs to output the word ‘lait’ it focusses on the word “milk”
- Path from input word to translation is smaller, so not affected by the short term limitations of RNNS
- **Visual Attention**
  - For example : Generating image captions using visual attentions
  - CNN processes image and outputs some feature maps
  - Then decoder RNN with attention generates the caption , one word at a time
- **Leads to Explainability (Ribeiro et al. 2016)**
  - **What led the model to produce its output**
  - **Which leads to fairness in results**
    - **Google apologizes after its Vision AI produced racist results**

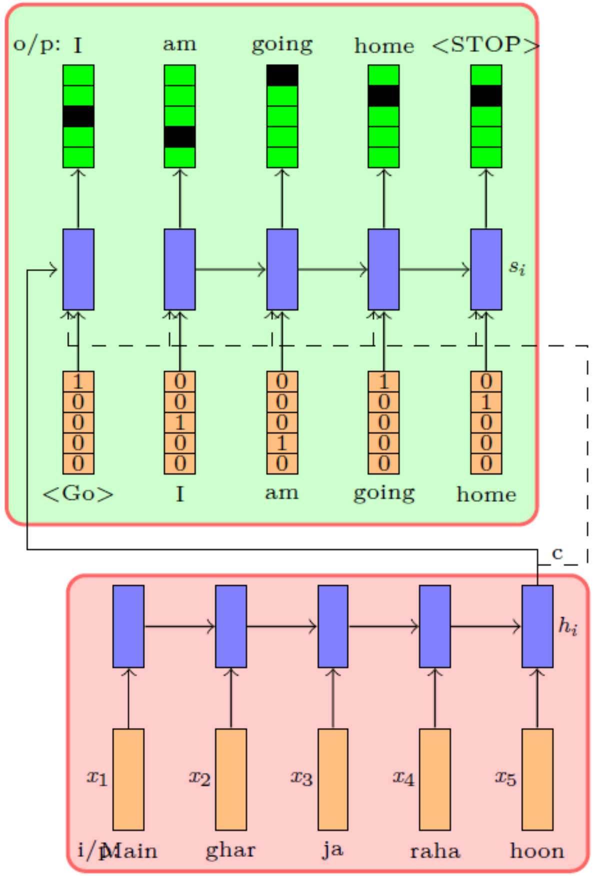
Attention Mechanism (Bahdanau et al , 2014)

- Attention is proposed as a method to both align and translate.
  - **Alignment**
    - is the problem in machine translation that identifies which parts of the input sequence are relevant to each word in the output
  - **translation**
    - is the process of using the relevant information to select the appropriate output
- *"... we introduce an extension to the encoder–decoder model which learns to align and translate jointly. Each time the proposed model generates a word in a translation, it (soft-)searches for a set of positions in a source sentence where the most relevant information is concentrated. The model then predicts a target word based on the context vectors associated with these source positions and all the previous generated target words."*
  - Neural Machine Translation by Jointly Learning to Align and Translate, 2015.

Attention Mechanism: Introduction

Task: Machine translation

o/p : I am going home



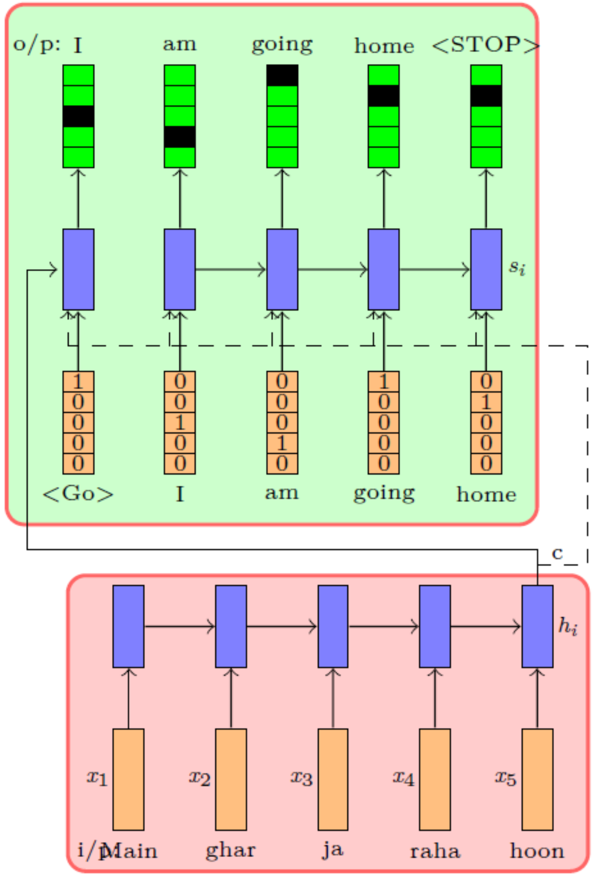
i/p : Main ghar ja raha hoon

Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

Attention Mechanism: Introduction

Task: Machine translation

o/p : I am going home



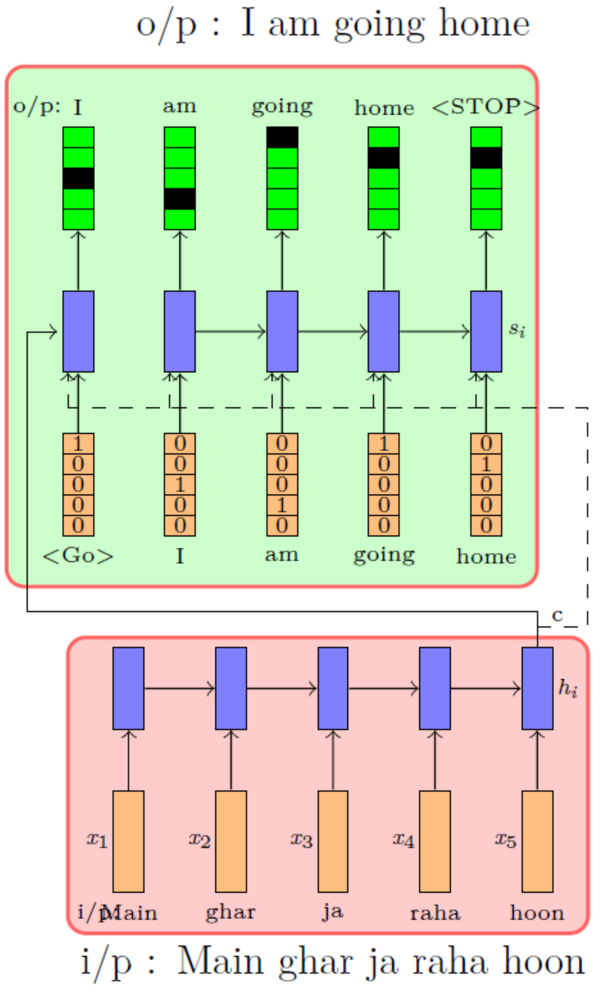
i/p : Main ghar ja raha hoon

- In a typical encoder decoder network, each time step of the decoder uses the information obtained from the last time step of the encoder.

Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

Attention Mechanism: Introduction

Task: Machine translation



- In a typical encoder decoder network, each time step of the decoder uses the information obtained from the last time step of the encoder.
- However, the translation would be effective if the network could focus/or pay attention to specific input word that would contribute to the prediction.

Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

Attention Mechanism: Introduction

- Consider the task of machine translation:

$$\text{o/p : I am going home}$$

$$\text{i/p : Main ghar ja raha hoon}$$

Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

Attention Mechanism: Introduction

While predicting each word in the o/p we would like our model to mimic humans and focus on specific words in the i/p

$$\text{o/p} : \text{I am going home}$$

$$t_1 : [\ 1\ 0\ 0\ 0\ 0\ ]$$

$$\text{i/p} : \text{Main ghar ja raha hoon}$$

Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

Attention Mechanism: Introduction

While predicting each word in the o/p we would like our model to mimic humans and focus on specific words in the i/p

o/p : I am going home

t_1 : [1 0 0 0 0]

t_2 : [0 0 0 0 1]

i/p : Main ghar ja raha hoon

Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

Attention Mechanism: Introduction

While predicting each word in the o/p we would like our model to mimic humans and focus on specific words in the i/p

o/p : I am going home

t_1 : [1 0 0 0 0]

t_2 : [0 0 0 0 1]

t_3 : [0 0 0.5 0.5 0]

i/p : Main ghar ja raha hoon

Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

Attention Mechanism: Introduction

$$\begin{aligned} o/p &: \text{I am going home} \\ t_1 &: [\,1\ 0\ 0\ 0\ 0\,] \\ t_2 &: [\,0\ 0\ 0\ 0\ 1\,] \\ t_3 &: [\,0\ 0\ 0.5\ 0.5\ 0\,] \\ t_4 &: [\,0\ 1\ 0\ 0\ 0\,] \end{aligned}$$

$$i/p : \text{Main ghar ja raha hoon}$$

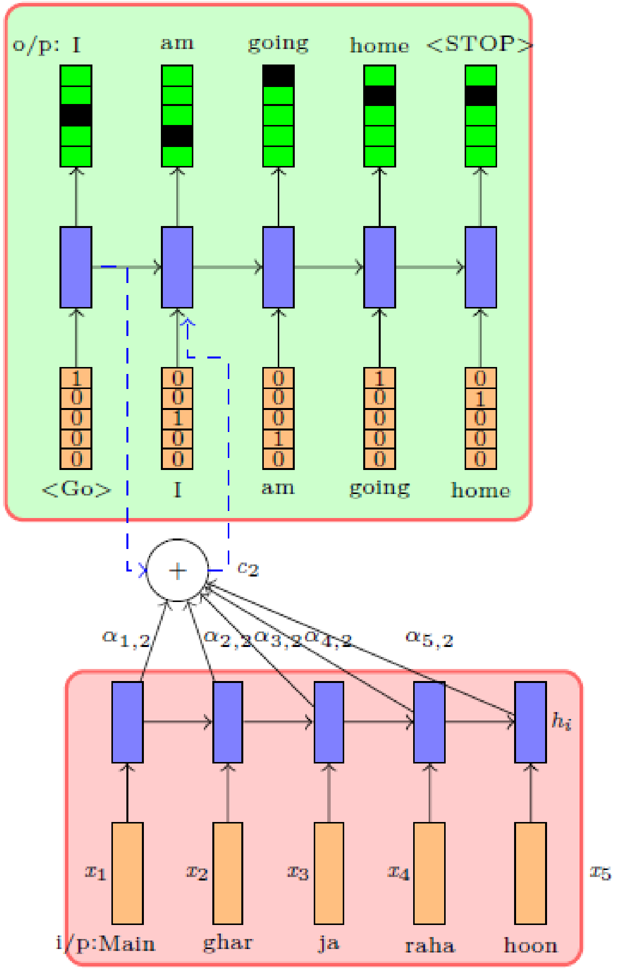
- Essentially, at each time step, a distribution on the input words must be introduced.
- This distribution tells the model how much attention to pay to each input words at each time step.

Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

Encoder Decoder with Attention Mechanism

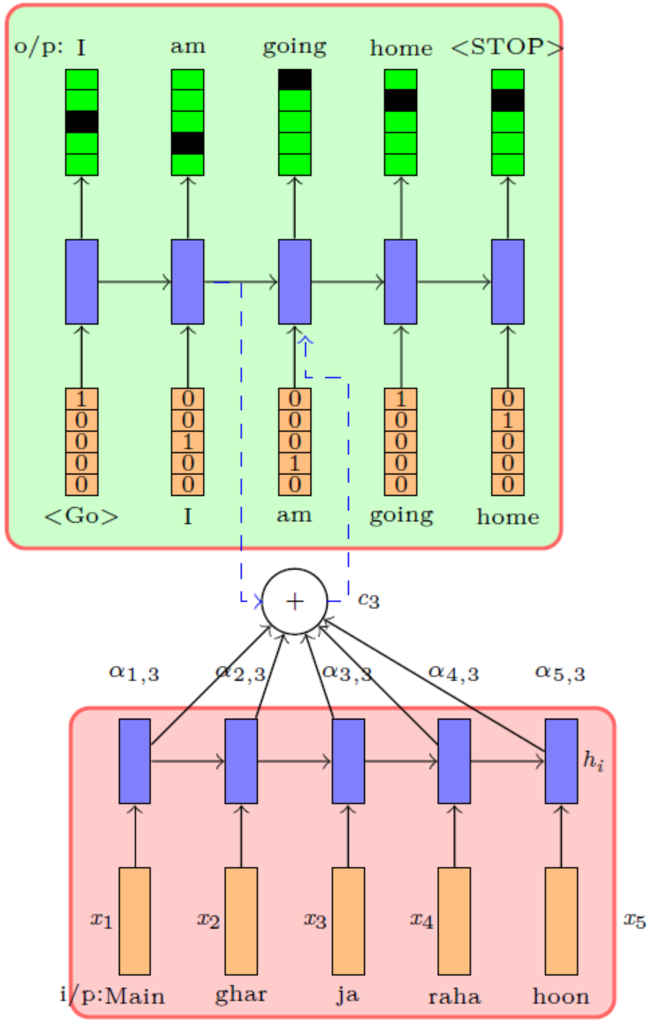
Task: Machine translation

- To do this, we could just take a weighted average of the corresponding word representations and feed it to the decoder



Encoder Decoder with Attention Mechanism

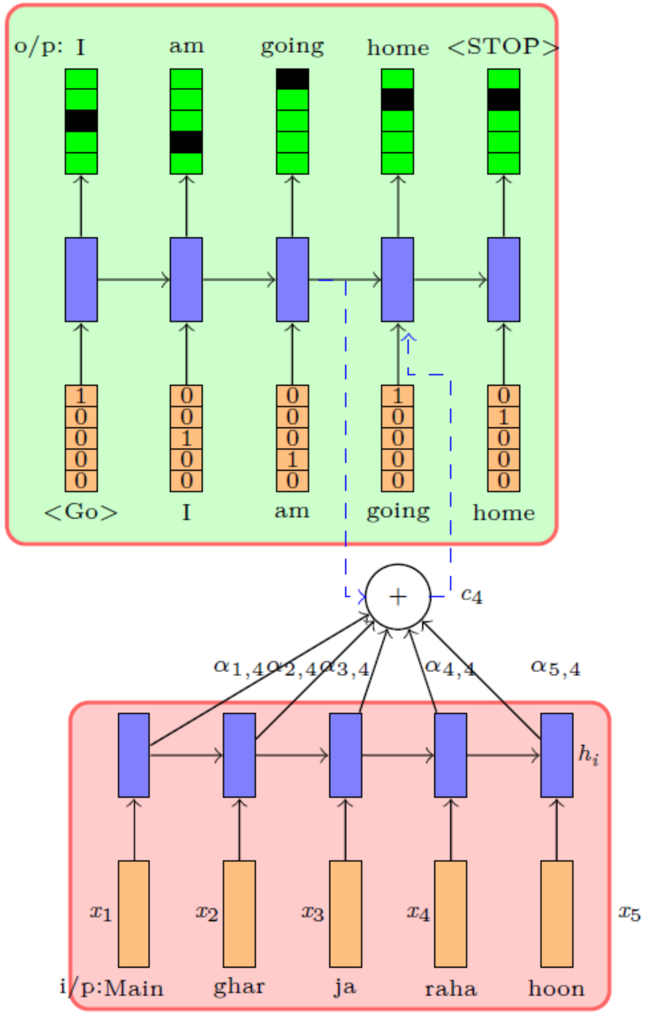
Task: Machine translation



- To do this, we could just take a weighted average of the corresponding word representations and feed it to the decoder

Encoder Decoder with Attention Mechanism

Task: Machine translation

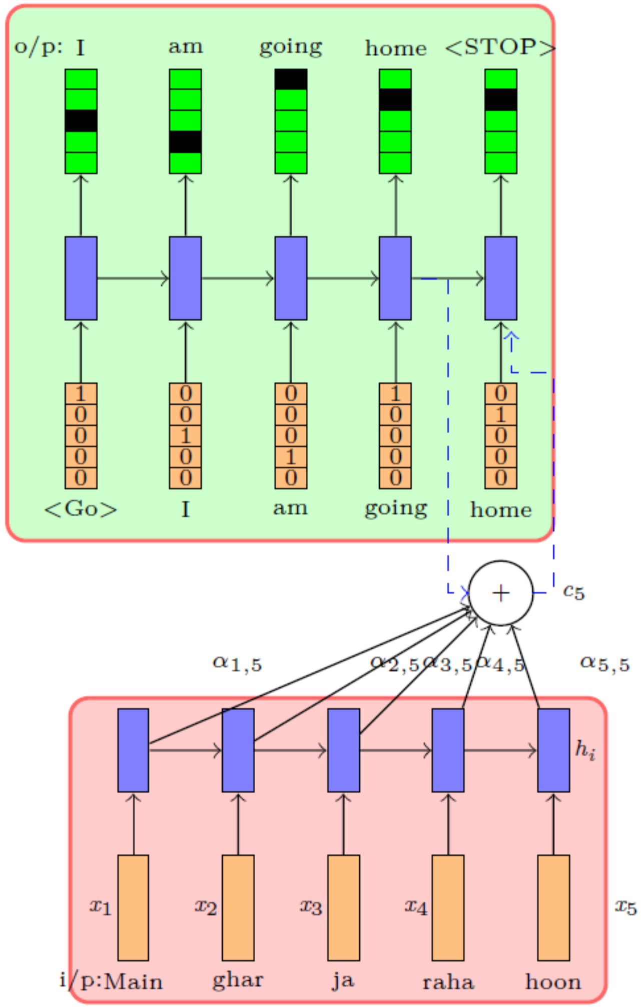


- To do this, we could just take a weighted average of the corresponding word representations and feed it to the decoder

Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

Encoder Decoder with Attention Mechanism

Task: Machine translation

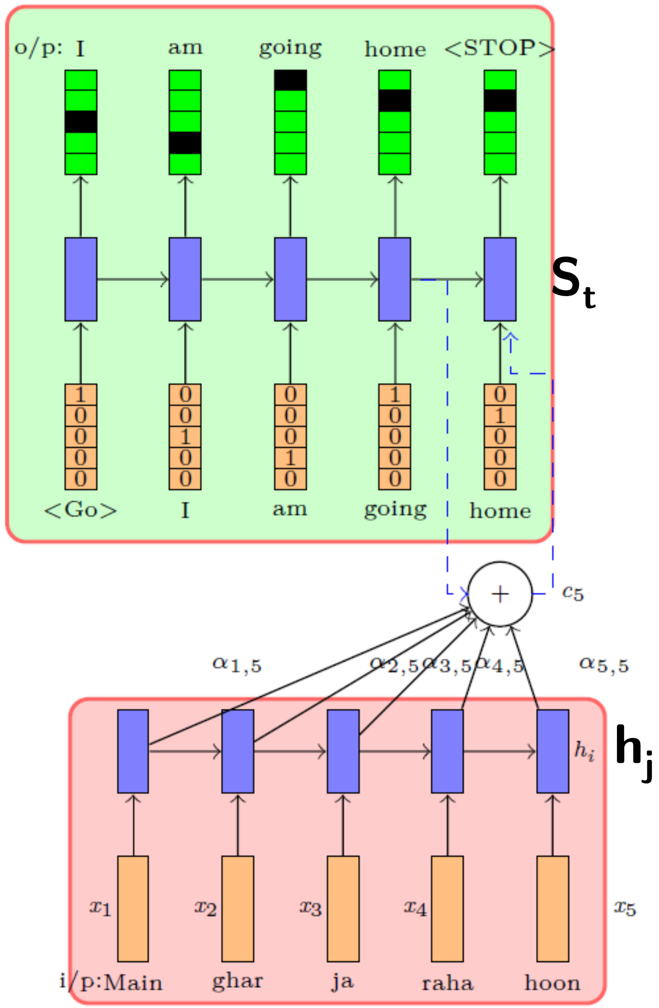


- To do this, we could just take a weighted average of the corresponding word representations and feed it to the decoder

Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

Encoder Decoder with Attention Mechanism

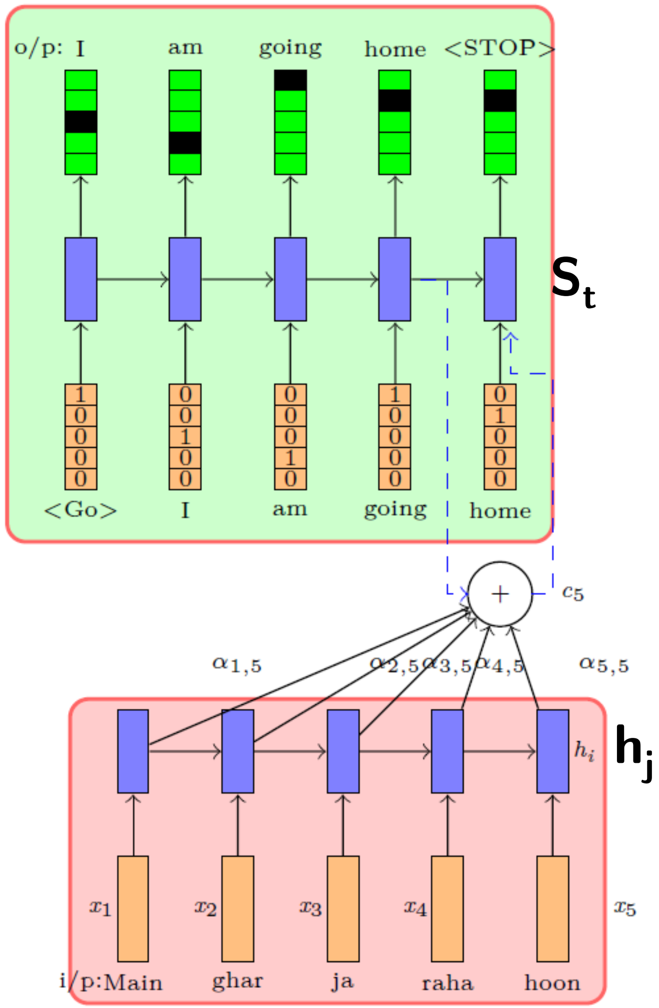
Task: Machine translation



Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

Encoder Decoder with Attention Mechanism

Task: Machine translation

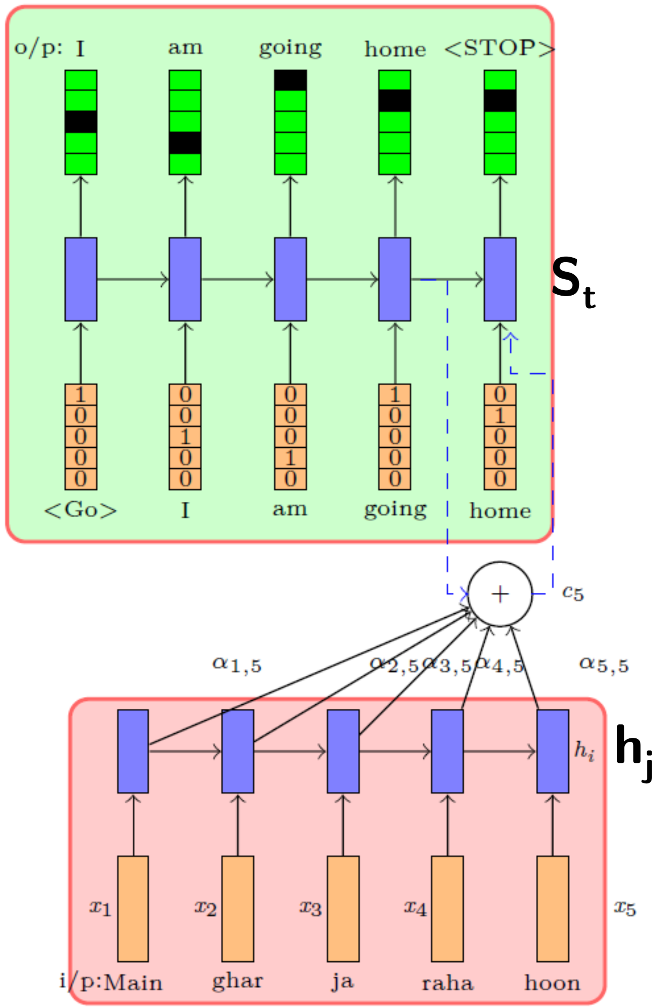


- To enable the network to focus on certain data we define the following function:

Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

Encoder Decoder with Attention Mechanism

Task: Machine translation

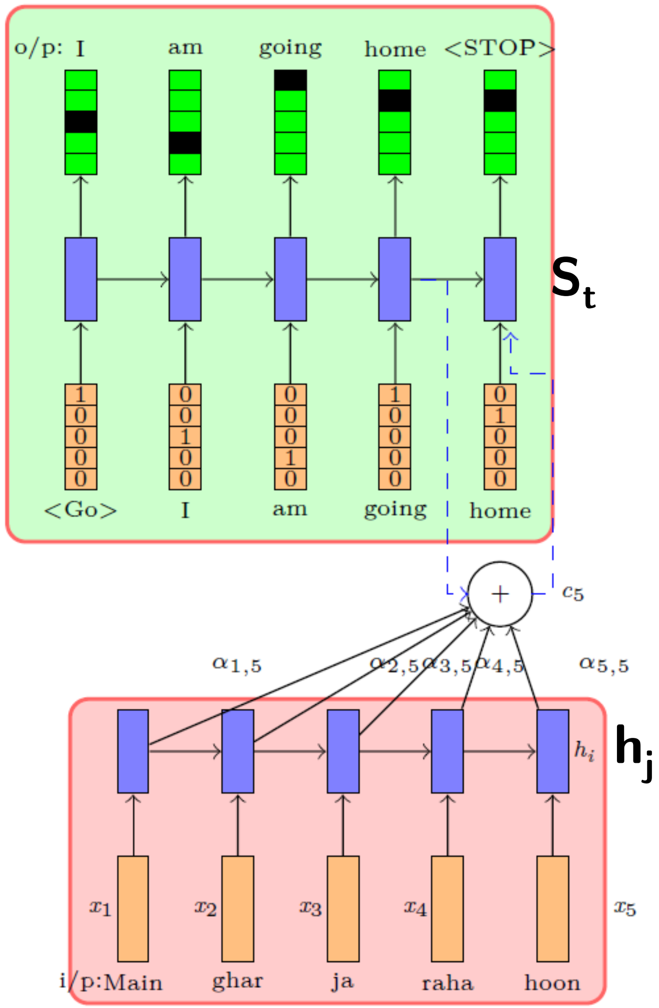


- To enable the network to focus on certain data we define the following function:

$$e_{jt} = f_{ATT}(s_{t-1}, \mathbf{h_j})$$

Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

Task: Machine translation



- To enable the network to focus on certain data we define the following function:

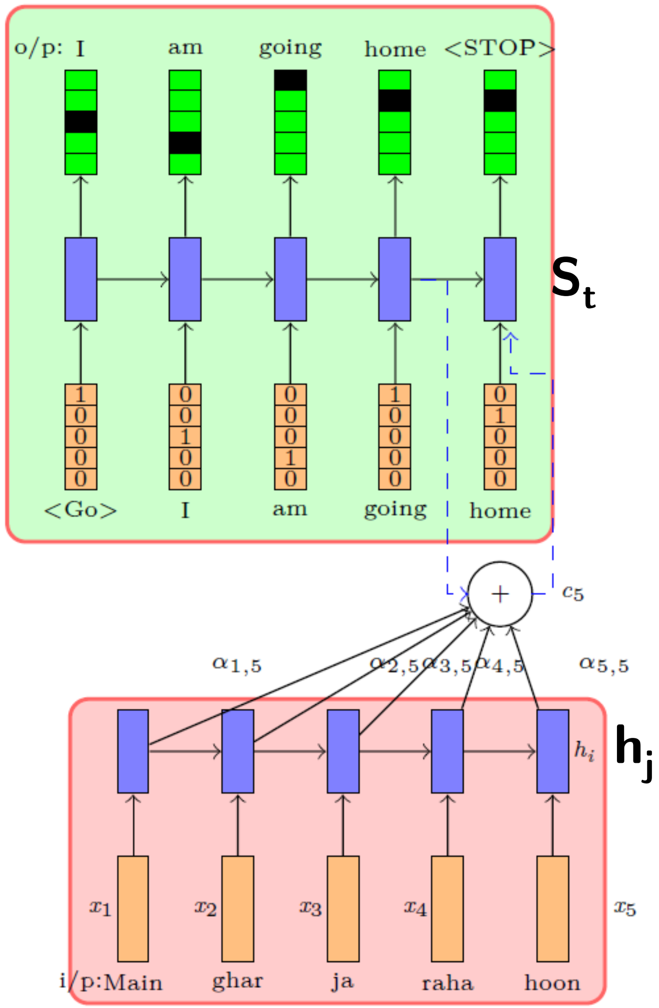
$$e_{jt} = f_{ATT}(s_{t-1}, \mathbf{h_j})$$

Can be considered as a separate feed forward network

Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

Encoder Decoder with Attention Mechanism

Task: Machine translation



- To enable the network to focus on certain data we define the following function:

$$e_{jt} = f_{ATT}(s_{t-1}, \mathbf{h_j})$$

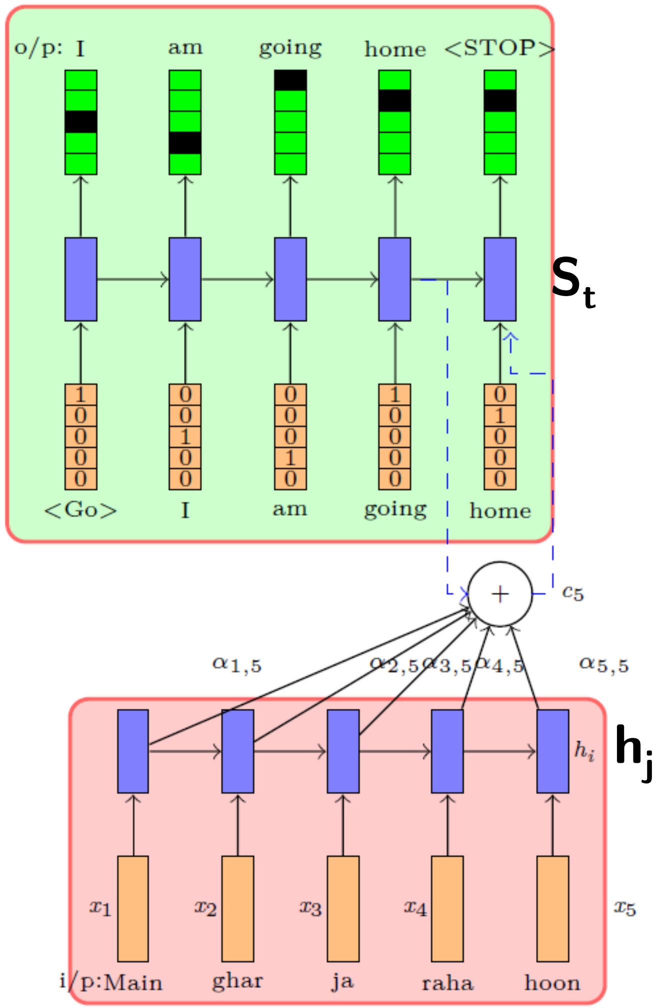
Can be considered as a separate feed forward network

- This quantity captures the importance of the j^{th} input word for decoding the t^{th} output word.

Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

Encoder Decoder with Attention Mechanism

Task: Machine translation



- To enable the network to focus on certain data we define the following function:

$$e_{jt} = f_{ATT}(s_{t-1}, \mathbf{h_j}) \longrightarrow$$

Can be considered as a separate feed forward network

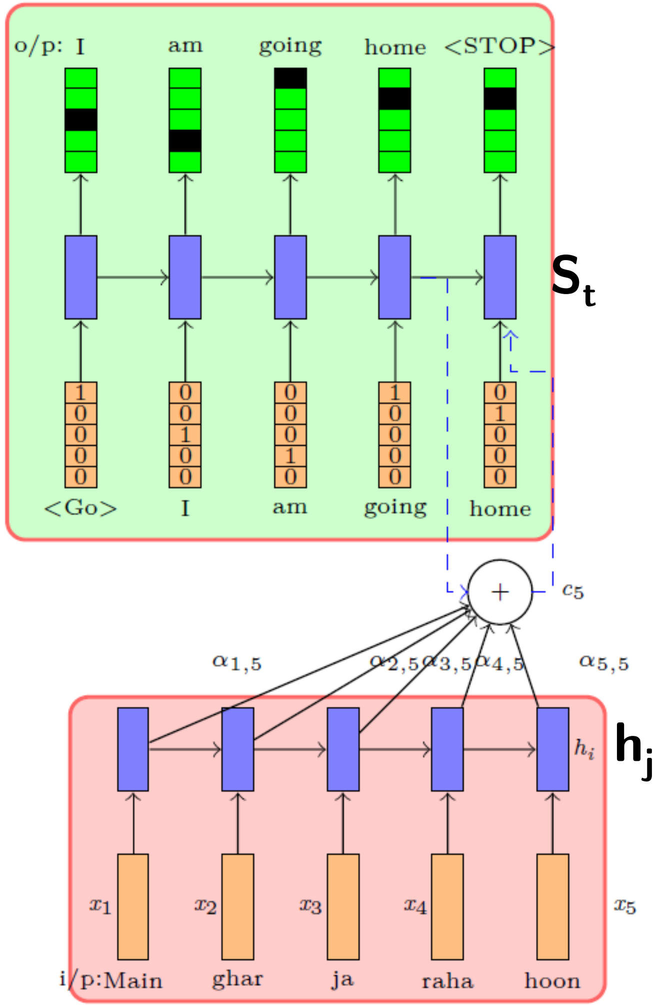
- This quantity captures the importance of the j^{th} input word for decoding the t^{th} output word.

- We could compute α_{jt} by normalizing these weights using the softmax function.

Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

Encoder Decoder with Attention Mechanism

Task: Machine translation



- To enable the network to focus on certain data we define the following function:

$$e_{jt} = f_{ATT}(s_{t-1}, \mathbf{h_j}) \longrightarrow$$

Can be considered as a separate feed forward network

- This quantity captures the importance of the j^{th} input word for decoding the t^{th} output word.

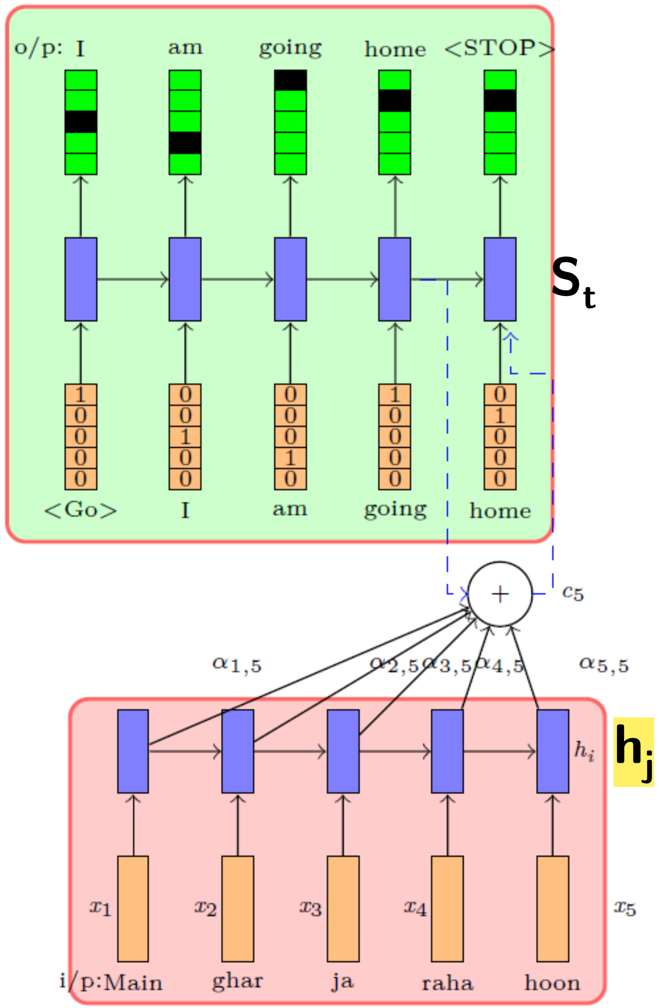
- We could compute α_{jt} by normalizing these weights using the softmax function.

$$\alpha_{jt} = \frac{exp(e_{jt})}{\sum_{j=1}^{M} exp(e_{jt})}$$

Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

Encoder Decoder with Attention Mechanism

Task: Machine translation



- To enable the network to focus on certain data we define the following function:

$$e_{jt} = f_{ATT}(s_{t-1}, \mathbf{h_j}) \longrightarrow$$

Can be considered as a separate feed forward network

- This quantity captures the importance of the j^{th} input word for decoding the t^{th} output word.

- We could compute α_{jt} by normalizing these weights using the softmax function.

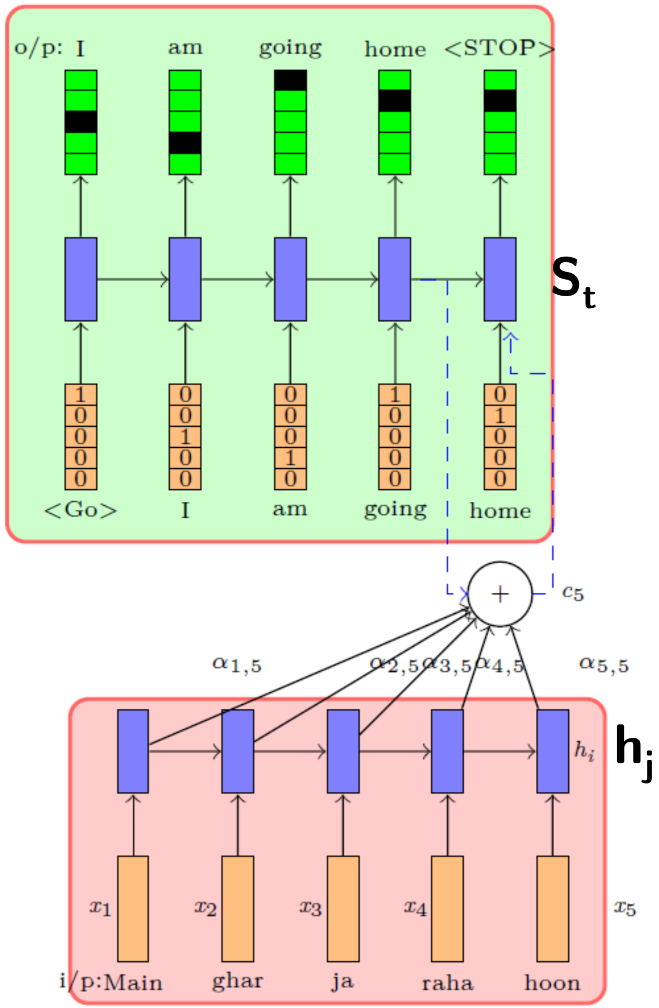
$$\alpha_{jt} = \frac{exp(e_{jt})}{\sum_{j=1}^{M} exp(e_{jt})}$$

- Where, α_{jt} denotes the probability of focusing on the j^{th} word to produce the t^{th} output word

Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

Encoder Decoder with Attention Mechanism

Task: Machine translation



- Introducing the parametric form of α :

$$\boxed{e_{jt} = V_{attn}^{T} tanh(U_{attn} h_j + W_{attn} s_t)}$$

$$\alpha_{jt} = \frac{exp(e_{jt})}{\sum_{j=1}^{M} exp(e_{jt})}$$

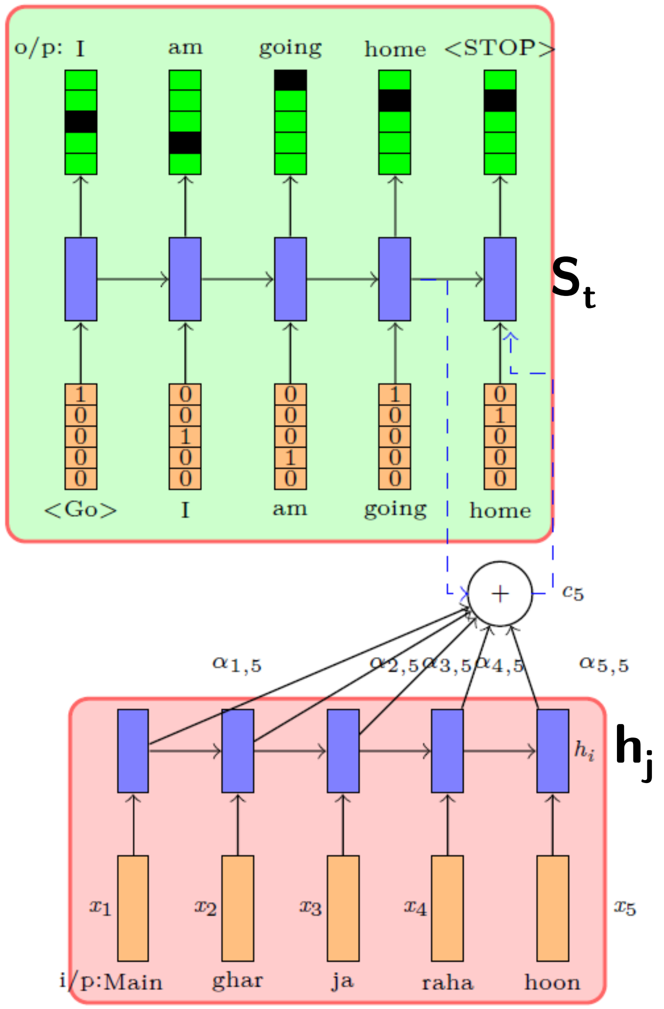
$$\boxed{c_t = \sum_{j=1}^{T} \alpha_{jt} h_j}$$

- Where, c_t (context) gives a weighted sum over the inputs.

Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

Encoder Decoder with Attention Mechanism

Task: Machine translation



- **Data:** $\{x_i = source_i,\ y_i = target_i\}_{i=1}^{N}$
- **Encoder:**

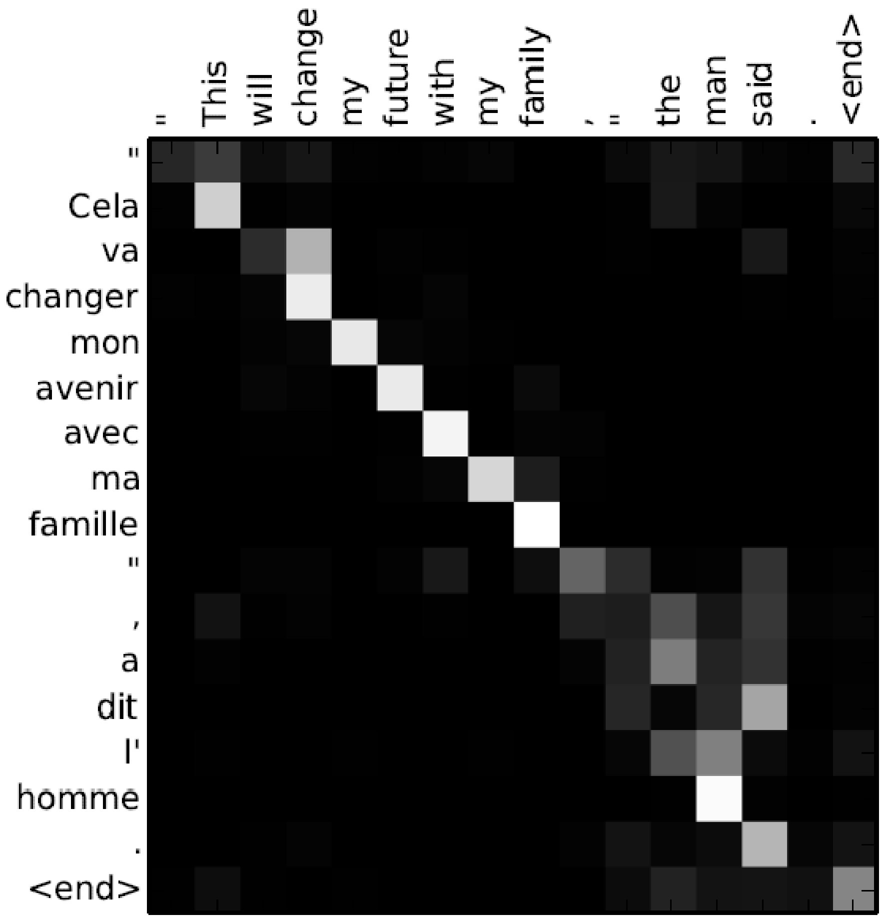
$$h_t = RNN(h_{t-1}, x_t)$$
$$s_0 = h_T$$

- **Decoder:**

$$e_{jt} = V_{attn}^{T} tanh(U_{attn} h_j + W_{attn} s_t)$$
$$\alpha_{jt} = softmax(e_{jt})$$
$$c_t = \sum_{j=1}^{T} \alpha_{jt} h_j$$
$$s_t = RNN(s_{t-1}, [e(\hat{y}_{t-1}), c_t])$$
$$\ell_t = softmax(V s_t + b)$$

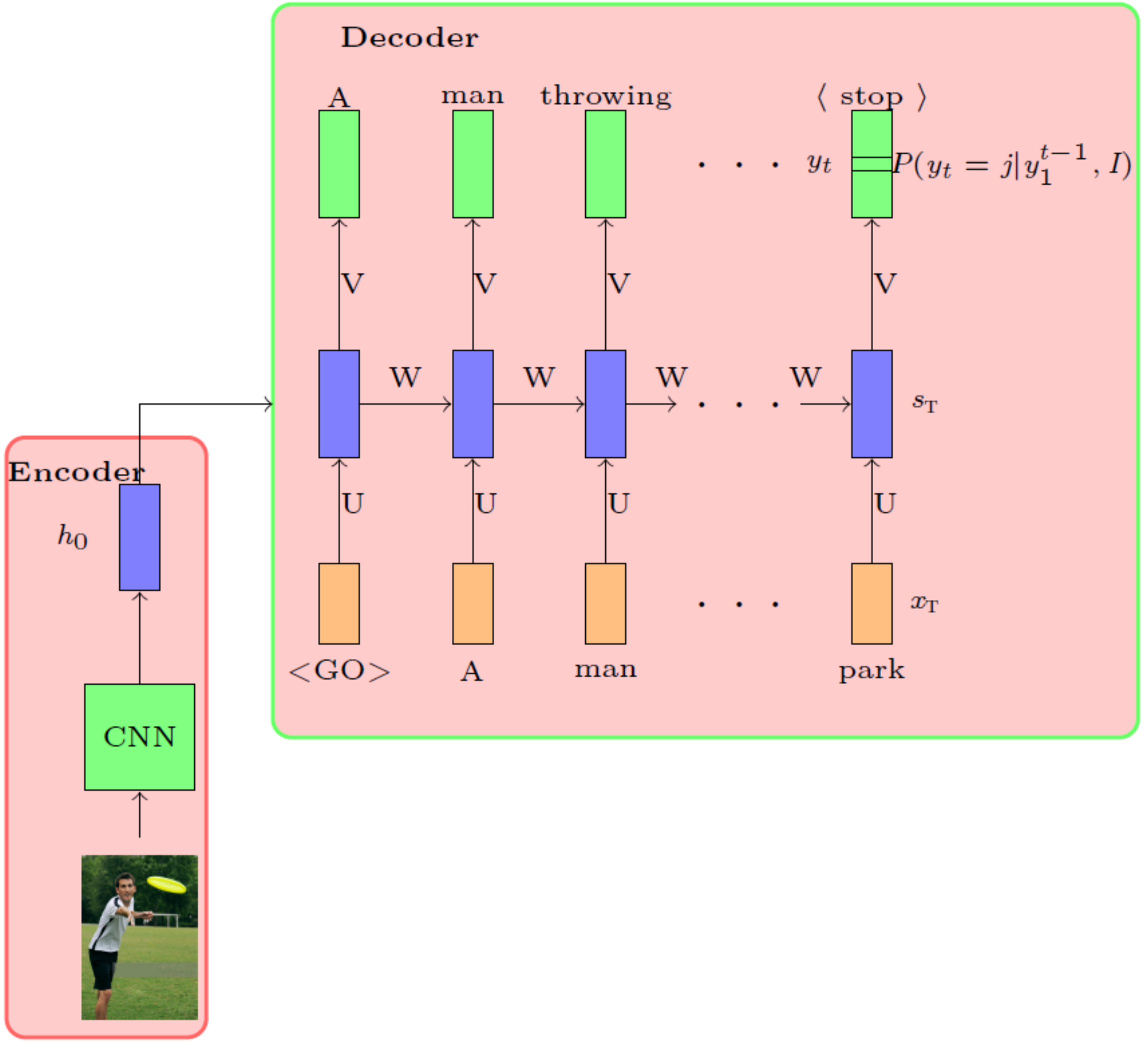
- **Parameters:** $U_{dec},\ V,\ W_{dec},\ U_{enc},\ W_{enc},\ b,\ U_{attn},\ V_{attn}$
- **Loss** and **Algorithm** remains same

Encoder Decoder with Attention Mechanism: Visualization



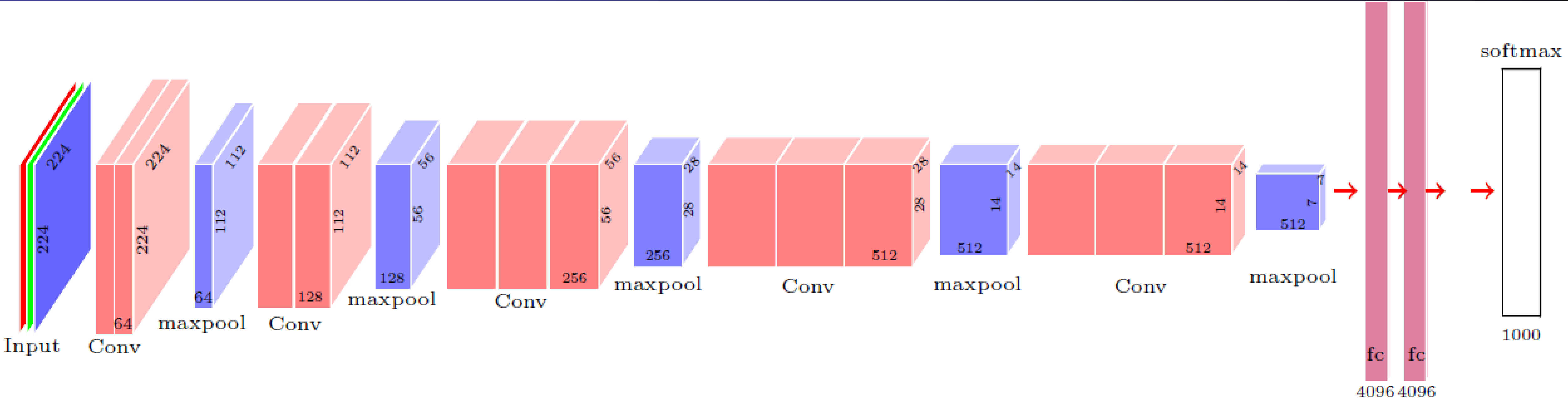
Example output of attention-based neural machine translation model
Bahdanau et al. 2015

Attention over Images



Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

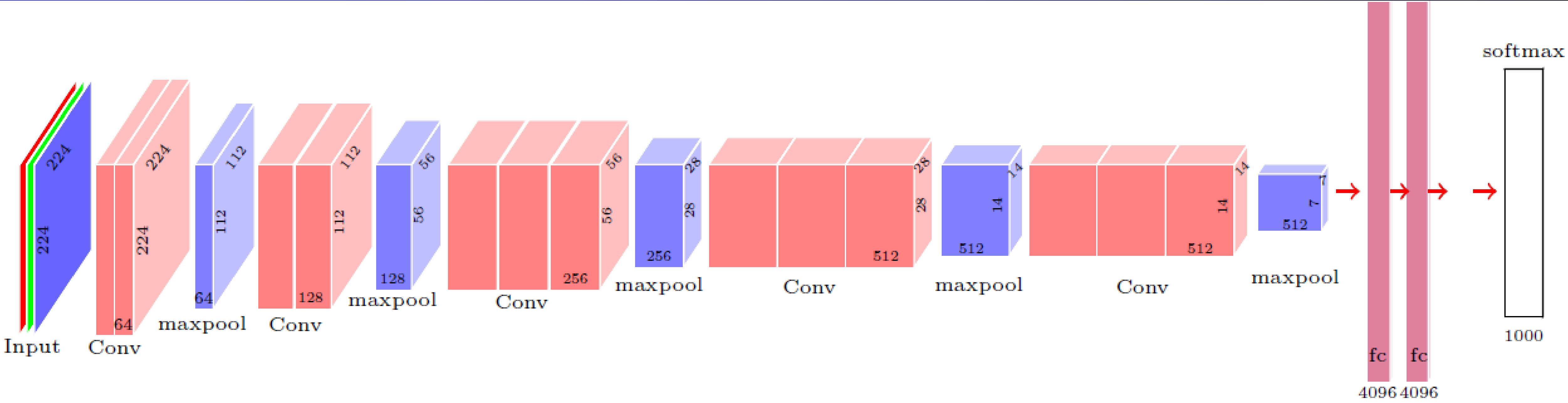
Attention over Images

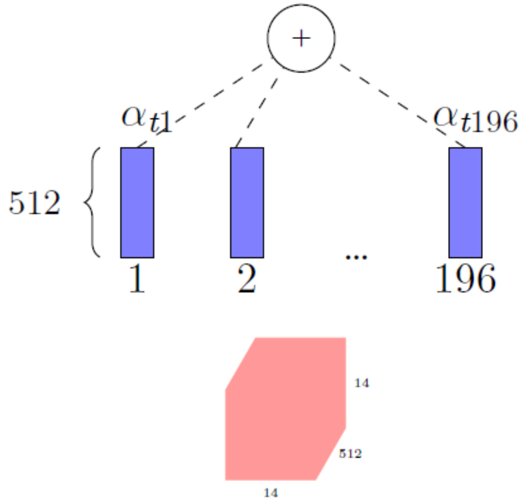


- For a CNN (eg: VGG-16) we would consider the convolution layer to be an input to the decoder, instead of the fully connected layers.
- This is because, the information about the image is contained in the feature maps in the convolution layer.
- Therefore, we could add attention weights to each pixel of the feature map volume to make the model focus on a particular pixel or region in the image.

Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

Attention over Images





- For a CNN (eg: VGG-16) we would consider the convolution layer to be an input to the decoder, instead of the fully connected layers.
- This is because, the information about the image is contained in the feature maps in the convolution layer.
- Therefore, we could add attention weights to each pixel of the feature map volume to make the model focus on a particular pixel or region in the image.

Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

Attention over Images

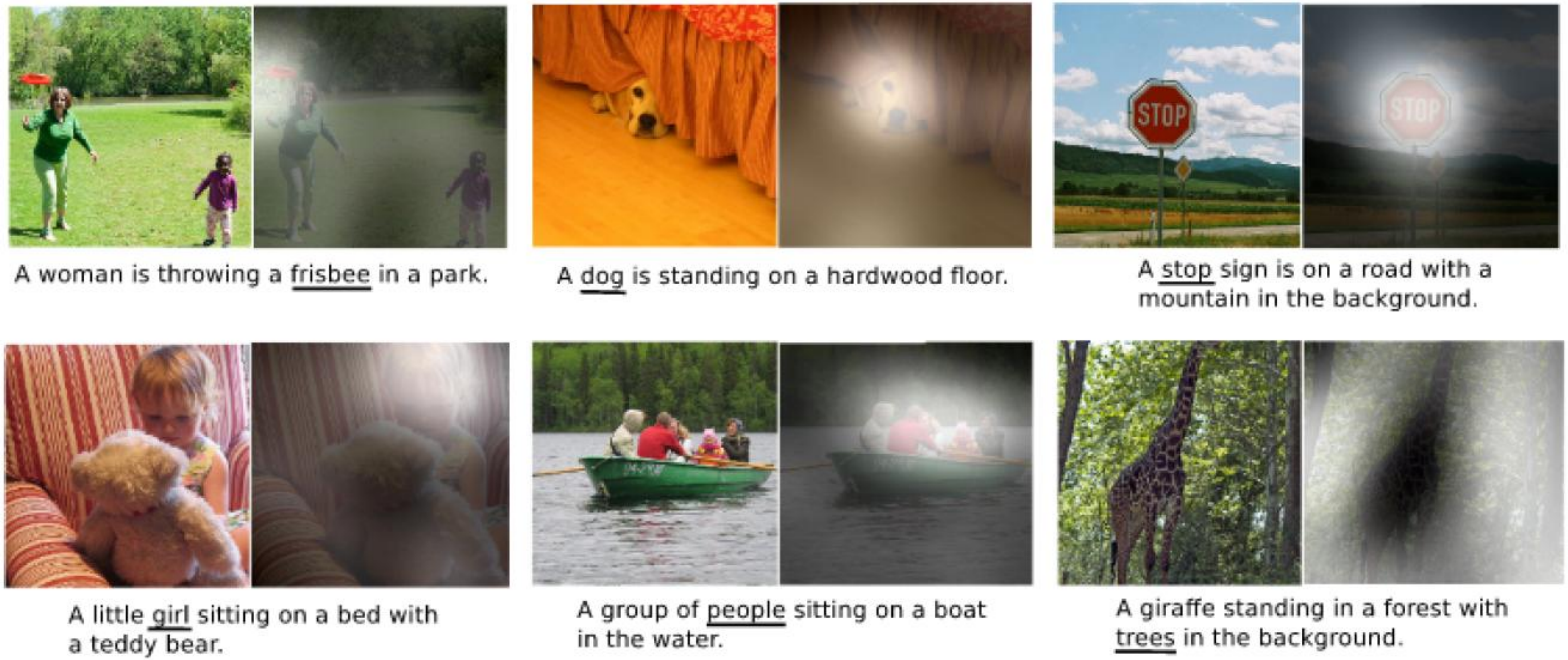


Figure: Examples of the attention-based model attending to the correct object (*white* indicates the attended regions, *underlines* indicates the corresponding word) [Kyunghyun Cho et al. 2015.]

Hierarchical Attention : Introduction to Hierarchical Models

Task: Chat Bot

Context

U: Can you suggest a good movie?

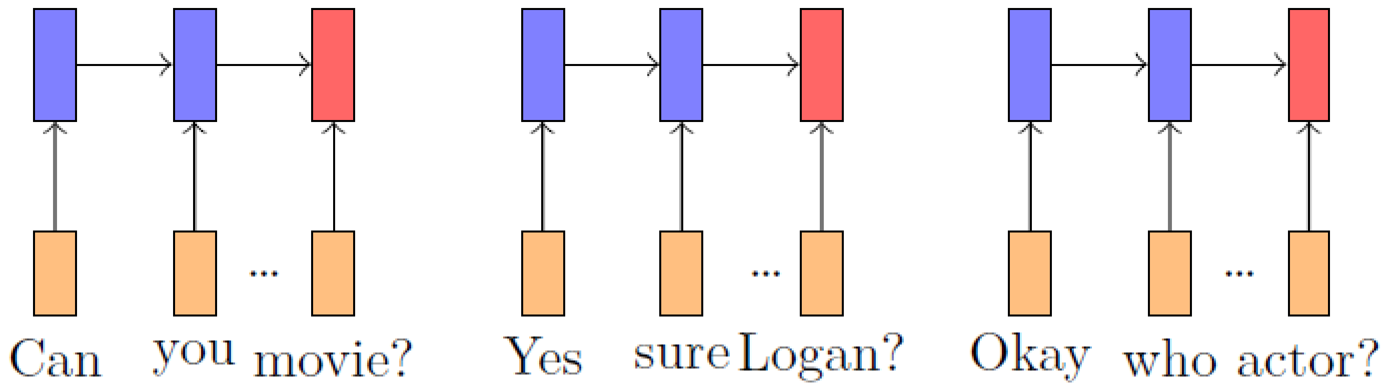
B: Yes, sure. How about Logan?

U: Okay, who is the lead actor?

Response

B: Hugh Jackman, of course

- Consider a dialog between a user (u) and a bot (B)
- The dialog contains a sequence of utterances between the user and the bot
- Each utterance in turn is a sequence of words
- Thus, what we have here is a "sequence of sequences" as input.



Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

Hierarchical Attention : Introduction to Hierarchical Models

Task: Chat Bot

Context

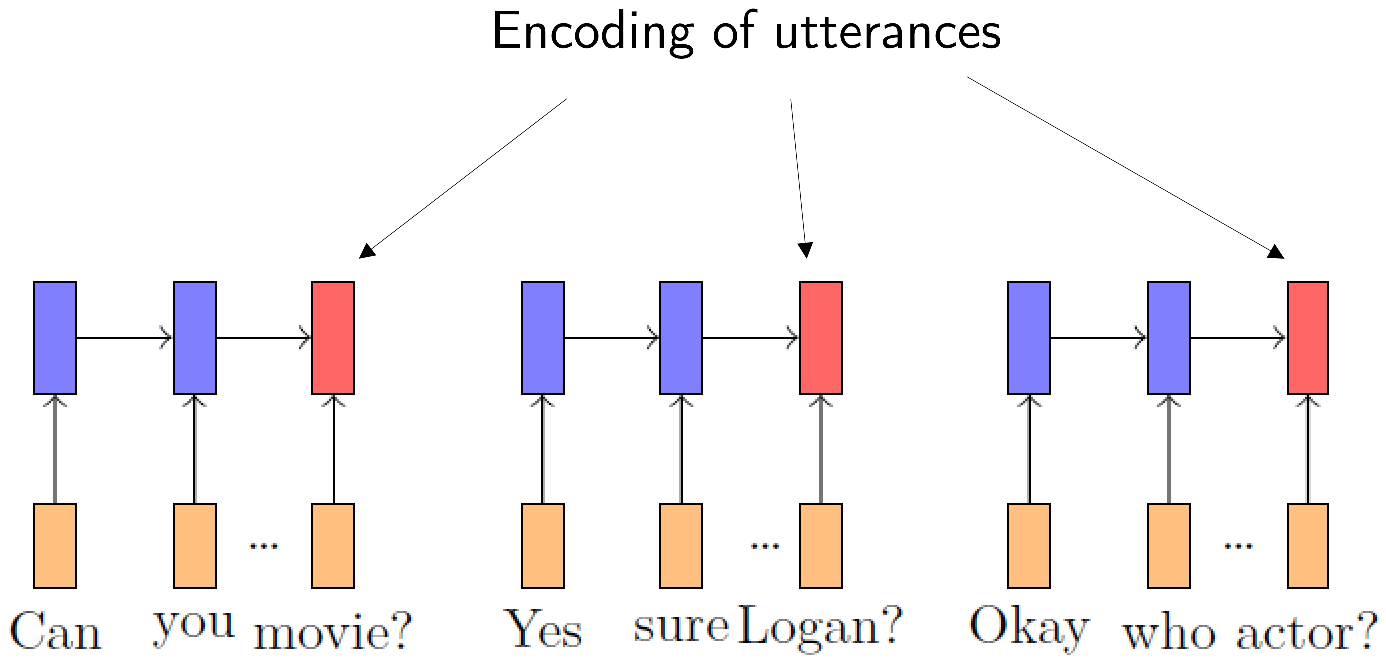
U: Can you suggest a good movie?

B: Yes, sure. How about Logan?

U: Okay, who is the lead actor?

Response

B: Hugh Jackman, of course



Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

Task: Chat Bot

Context

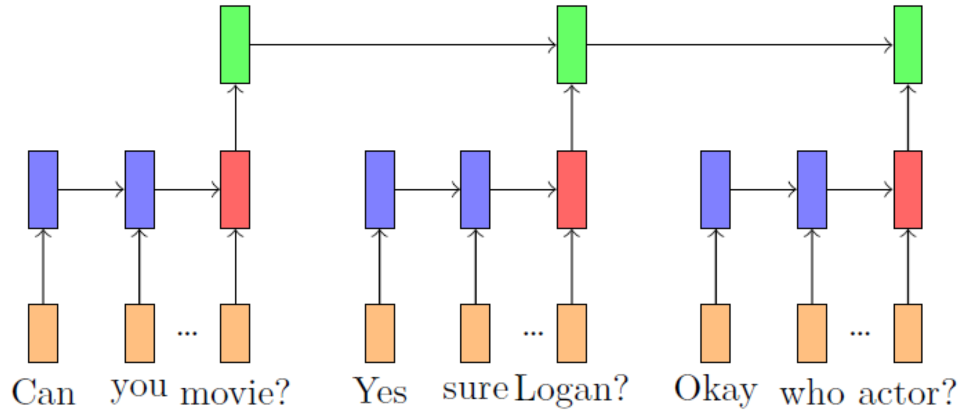
U: Can you suggest a good movie?

B: Yes, sure. How about Logan?

U: Okay, who is the lead actor?

Response

B: Hugh Jackman, of course



Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

Task: Chat Bot

Context

U: Can you suggest a good movie?

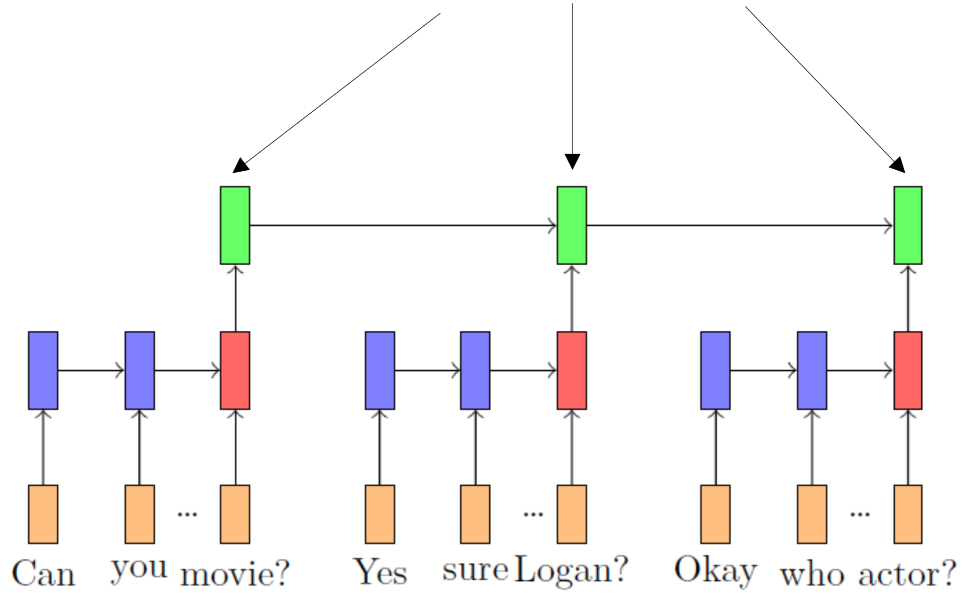
B: Yes, sure. How about Logan?

U: Okay, who is the lead actor?

Response

B: Hugh Jackman, of course

Encoding of sequence of utterances



Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

Task: Chat Bot

Context

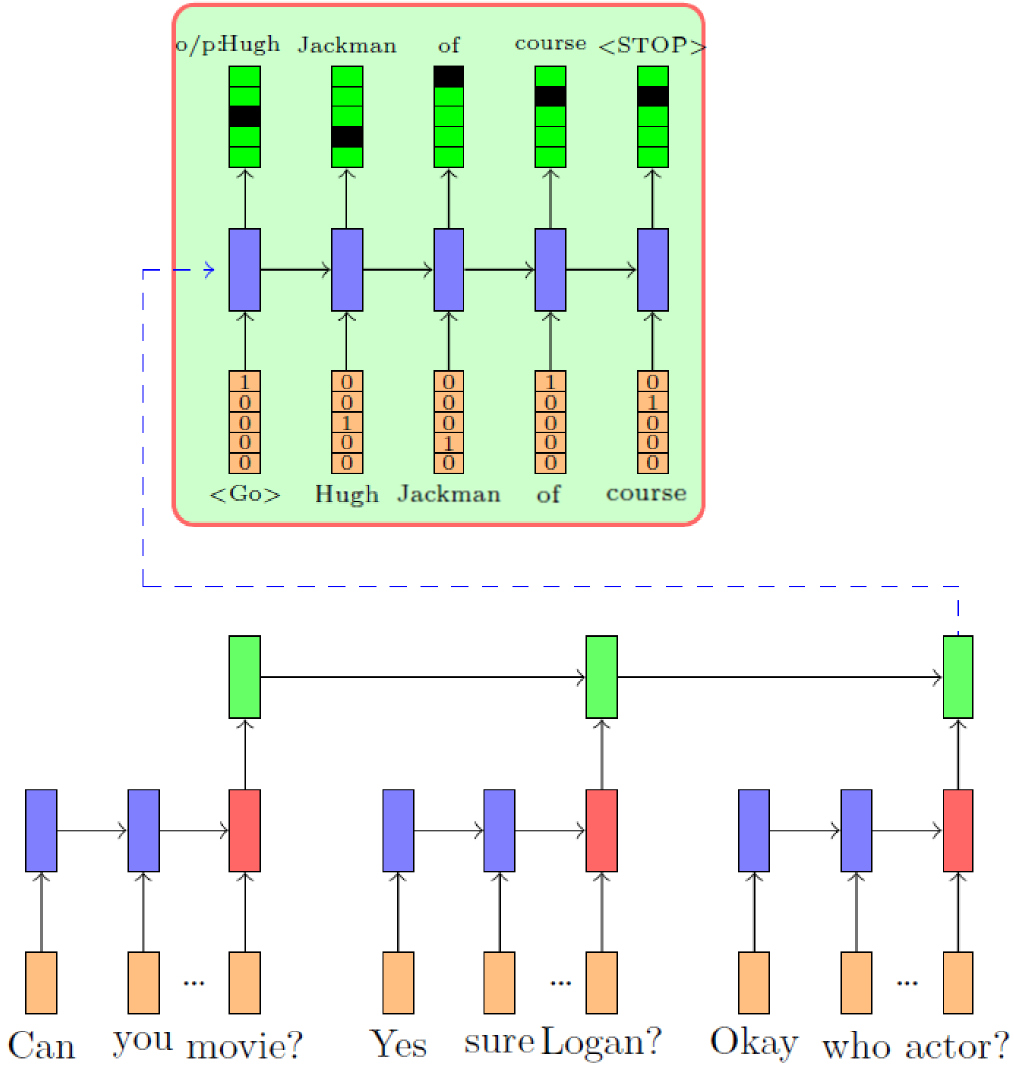
U: Can you suggest a good movie?

B: Yes, sure. How about Logan?

U: Okay, who is the lead actor?

Response

B: Hugh Jackman, of course



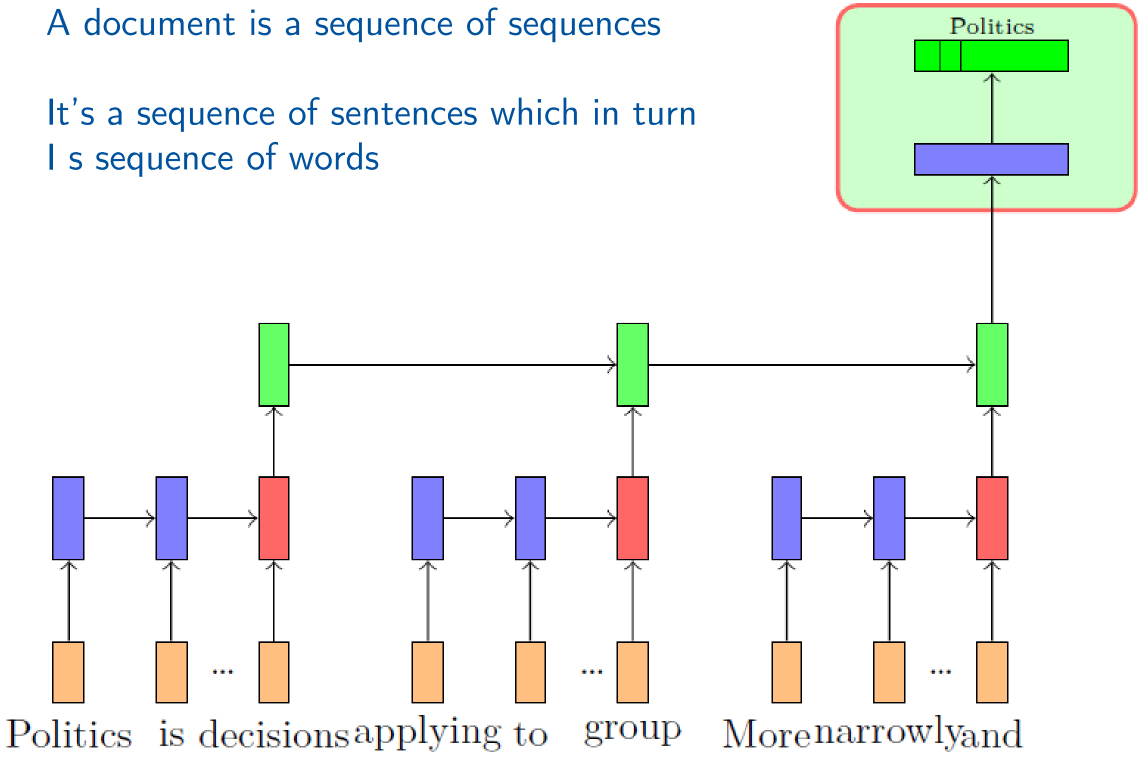
Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

Hierarchical Attention : Introduction to Hierarchical Models

Task: Document Summarization

Politics is the process of making decisions applying to all members of each group. More narrowly, it refers to achieving and ...

- A document is a sequence of sequences
- It's a sequence of sentences which in turn I s sequence of words



- **Data:** $\{Document_i, class_i\}_{i=1}^{N}$
- **Word level (1) encoder:**

$$h_{ij}^1 = RNN(h_{ij-1}^1, w_{ij})$$

$$s_i = h_{iT_i}^1 \quad [T \textit{ is length of sentence } i]$$

- **Sentence level (2) encoder:**

$$h_i^2 = RNN(h_{i-1}^2, s_i)$$

$$s = h_K^2 \quad [K \textit{ is number of sentences}]$$

- **Decoder:**

$$P(y|document) = softmax(Vs + b)$$

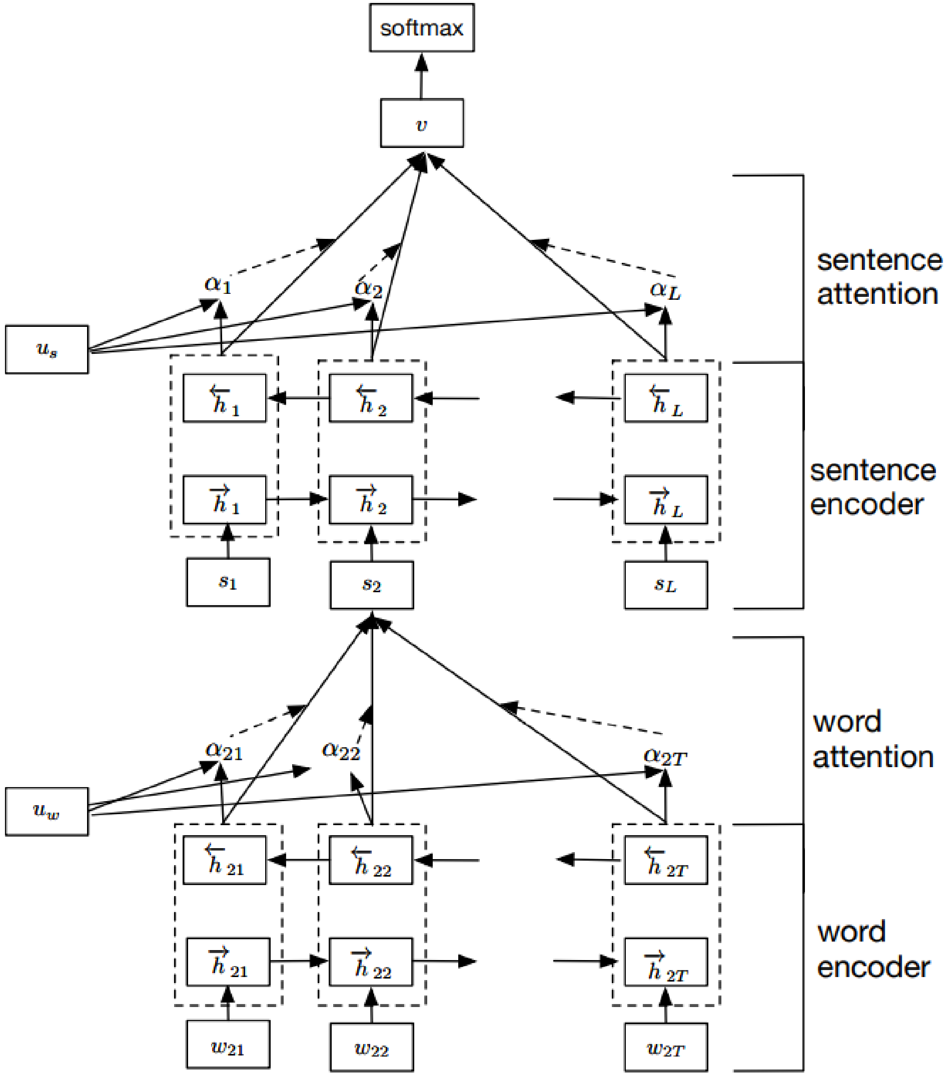
- **Params:** $W_{enc}^1, U_{enc}^1, W_{enc}^2, U_{enc}^2, V, b$
- **Loss:** Cross Entropy
- **Algorithm:** Gradient Descent with backpropagation

Source: CS7015 Deep Learning, Dept. of CSE, IIT Madras

Document Classification using Hierarchical Attention Networks

- To understand the main message of a document
  - Not every word in a sentence and every sentence in a document are equally important
- Meaning of word depends on context
  - For example - "The bouquet of flowers is pretty" vs. "The food is pretty bad".
- HAN
  - considers the hierarchical structure of documents (document - sentences - words)
  - Includes an attention mechanism that is able to find the most important words and sentences in a document while taking the context into consideration

Hierarchical Attention Networks



- **Data:** $\{Document_i, class_i\}_{i=1}^{N}$
- **Word level (1) encoder:**

$$h_{ij} = RNN(h_{ij-1}, w_{ij})$$
$$u_{ij} = tanh(W_w h_{ij} + b_w)$$
$$\alpha_{ij} = \frac{exp(u_{ij}^T u_w)}{\sum_t exp(u_{it}^T u_w)}$$
$$s_i = \sum_j \alpha_{ij} h_{ij}$$

- **Sentence level (2) encoder:**

$$h_i = RNN(h_{i-1}, s_i)$$
$$u_i = tanh(W_s h_i + b_s)$$
$$\alpha_i = \frac{exp(u_i^T u_s)}{\sum_i exp(u_i^T u_s)}$$
$$s = \sum_i \alpha_i h_i$$

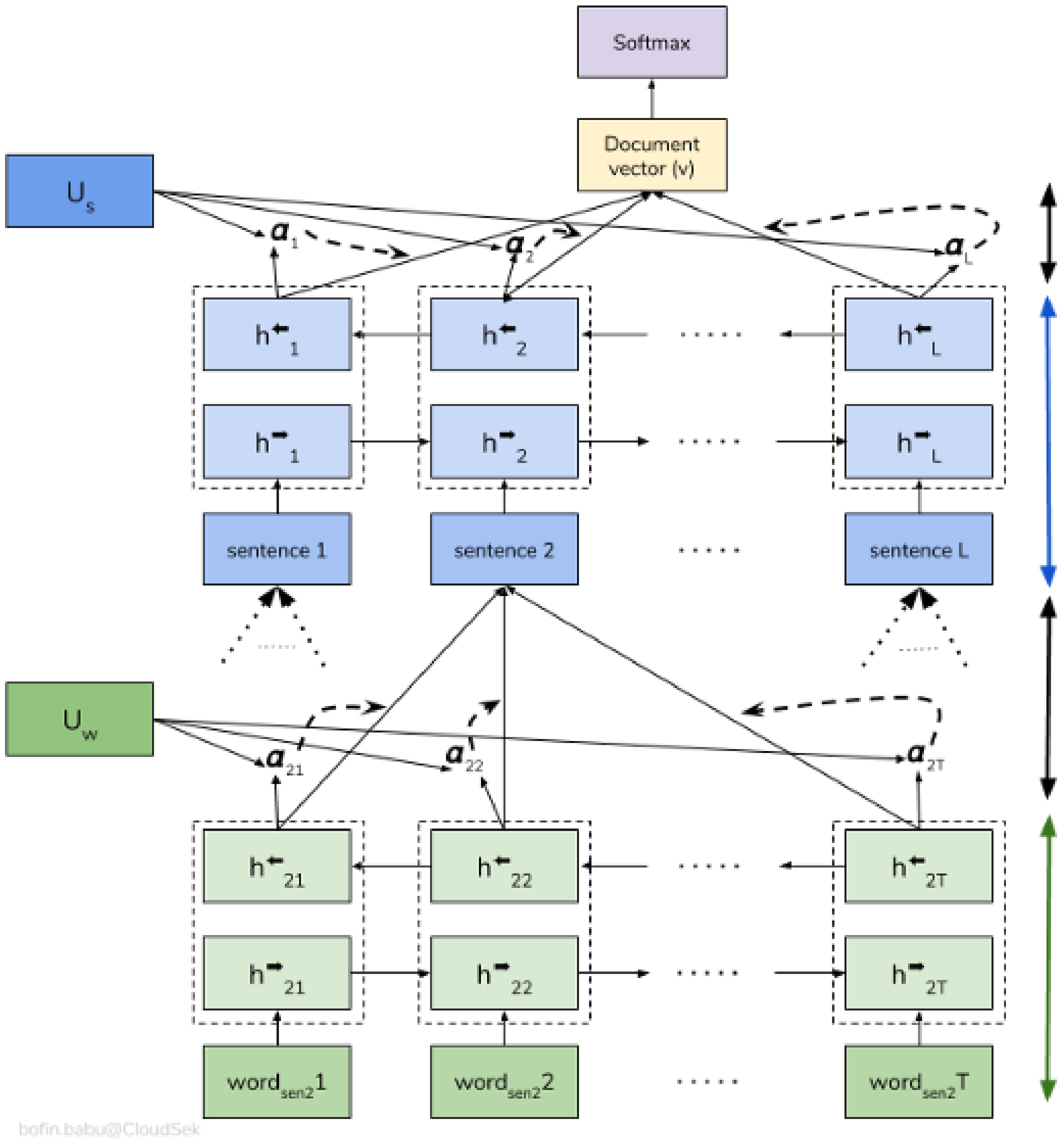
- **Decoder:**

$$P(y|document) = softmax(Vs + b)$$

- **Parameters:** $W_w, W_s, V, b_w, b_s, b, u_w, u_s$
- **Loss:** cross entropy
- **Algorithm:** Gradient Descent and backpropagation

Yang et al. Hierarchical Attention Networks for Document Classification, Proceedings of NAACL-HLT 2016

Hierarchical Attention Networks (Yang et al. 2016)



Sentence attention

U_s: sentence context vector

Sentence encoder

Word attention

α : importance weight
U_w : word context vector (what is the informative word?)

Word encoder

$h^{\rightarrow}$: forward hidden state
$h^{\leftarrow}$: backward hidden state
$word_{sen2}1$: The first word in sentence 2

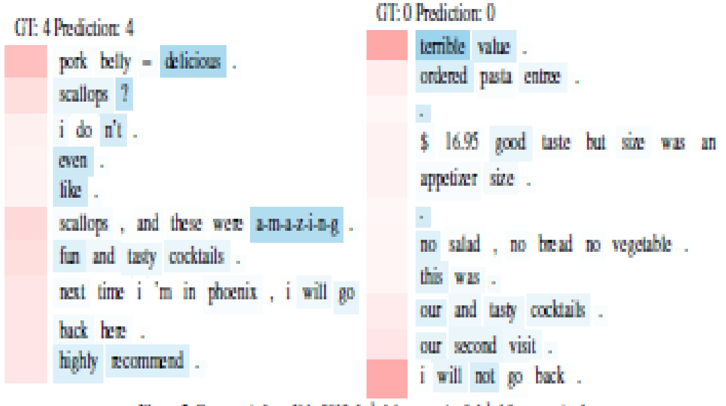


Figure 5: Documents from Yelp 2013. Label 4 means star 5, label 0 means star 1.

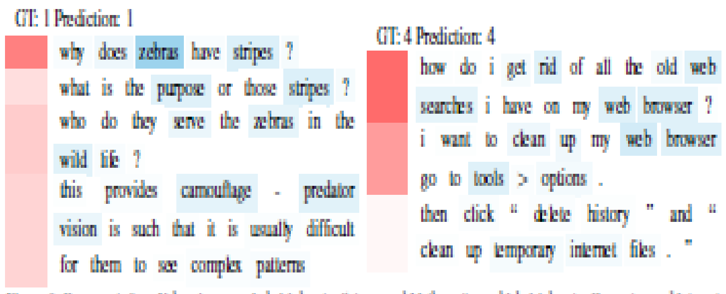


Figure 6: Documents from Yahoo Answers. Label 1 denotes Science and Mathematics and label 4 denotes Computers and Internet.