Attention and Beam Search Models

CS8004: Deep Learning and Applications

Slides based on : guillaumegenthial.github.io

Sequence to Sequence RNN Model

- The Sequence to Sequence framework is based on the encoder-decoder model.

- The encoder encodes the input sequence, while the decoder produces the target sequence.

Encoder

- Suppose the problem is machine translation from English to French.
- Each word from the input sequence is associated to a vector $w \in \boldsymbol{R}^d$.
- Input sequence is 'How are you"
- 3 words input sequence $w_0, w_1, w_2 \in \boldsymbol{R}^d$.
- Run an LSTM model on these three inputs. We get three hidden states e_0, e_1, e_2.
- Store the last hidden state output $e = e_2$. This is the input vector for the decoder.

Vanilla Encoder

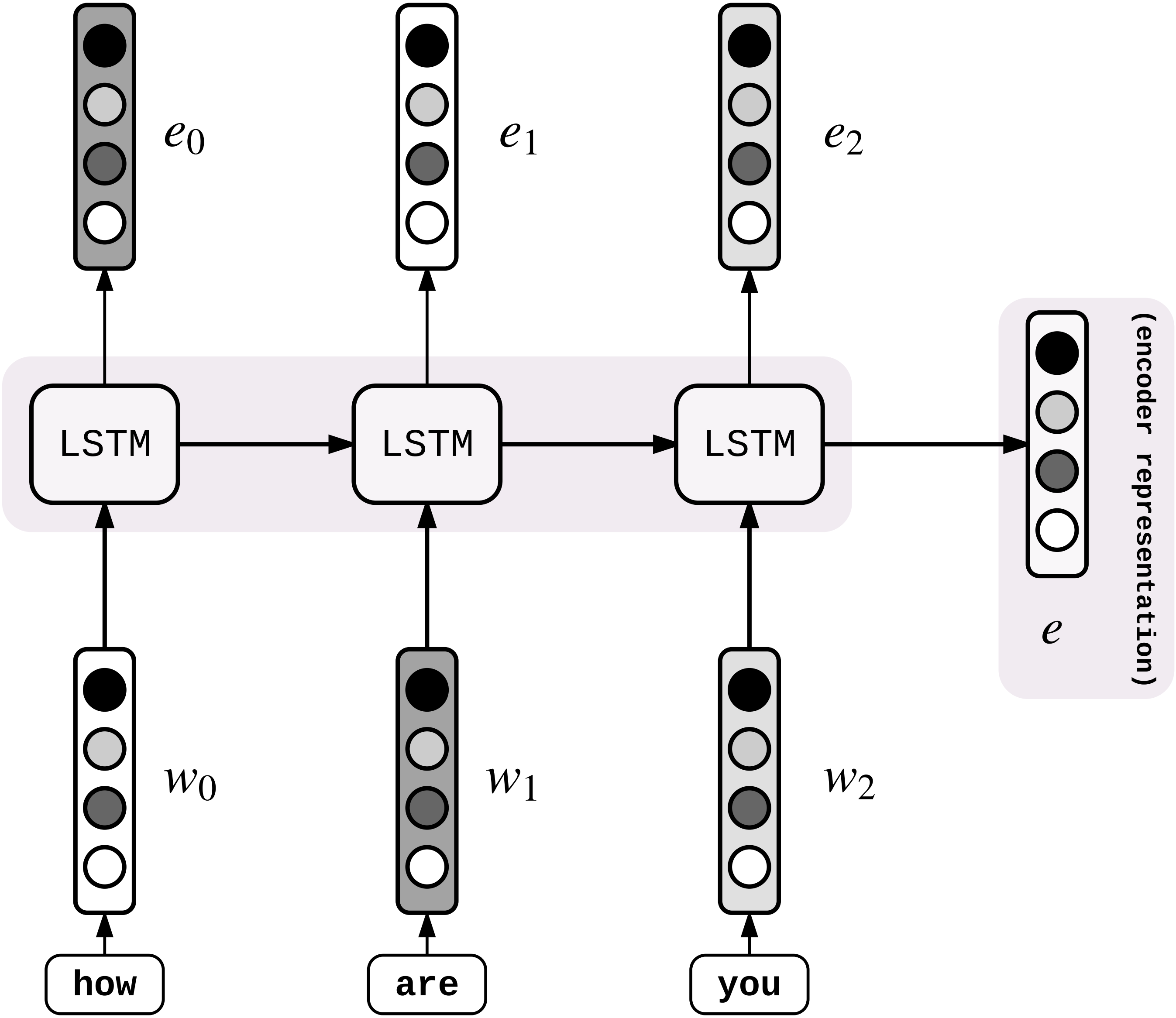


Figure source: guillaumegenthial.github.io

Decoder

- The vector e is assumed to have captured the meaning of the input sequence.

- Is used to generate the target sequence of word.

- It is fed to another LSTM unit as a hidden state with a starting probable word in French, w_{sos} (start of sequence word).

Decoder

- The next hidden state $h_0 \in \boldsymbol{R}^h$ is computed using e and w_{sos}.

- Next, h_0 is transformed to another vector $s_0 \in \boldsymbol{R}^V$ a vector to match the size of the vocabulary in French.

- Finally, apply $softmax$ to s_0 to find out the vector of probabilities in $\boldsymbol{R}^V$. The corresponding word with highest probability i_0 is chosen to be the index of the first output word w_{i_0} in the output sequence.
- In this way w_{i_t} is the output word at $t - th$ time step.

Decoder

Now the next word is fed to the next LSTM unit as the input.

Equations:

$$h_0 = LSTM(w_{sos}, e)$$
$$s_0 = g(h_0)$$
$$p_0 = softmax(s_0)$$
$$i_0 = argmax(p_0)$$

$$h_t = LSTM\left(h_{t-1}, w_{i_{t-1}}\right)$$
$$s_t = g(h_t)$$
$$p_t = softmax(s_t)$$
$$i_t = argmax(p_t)$$

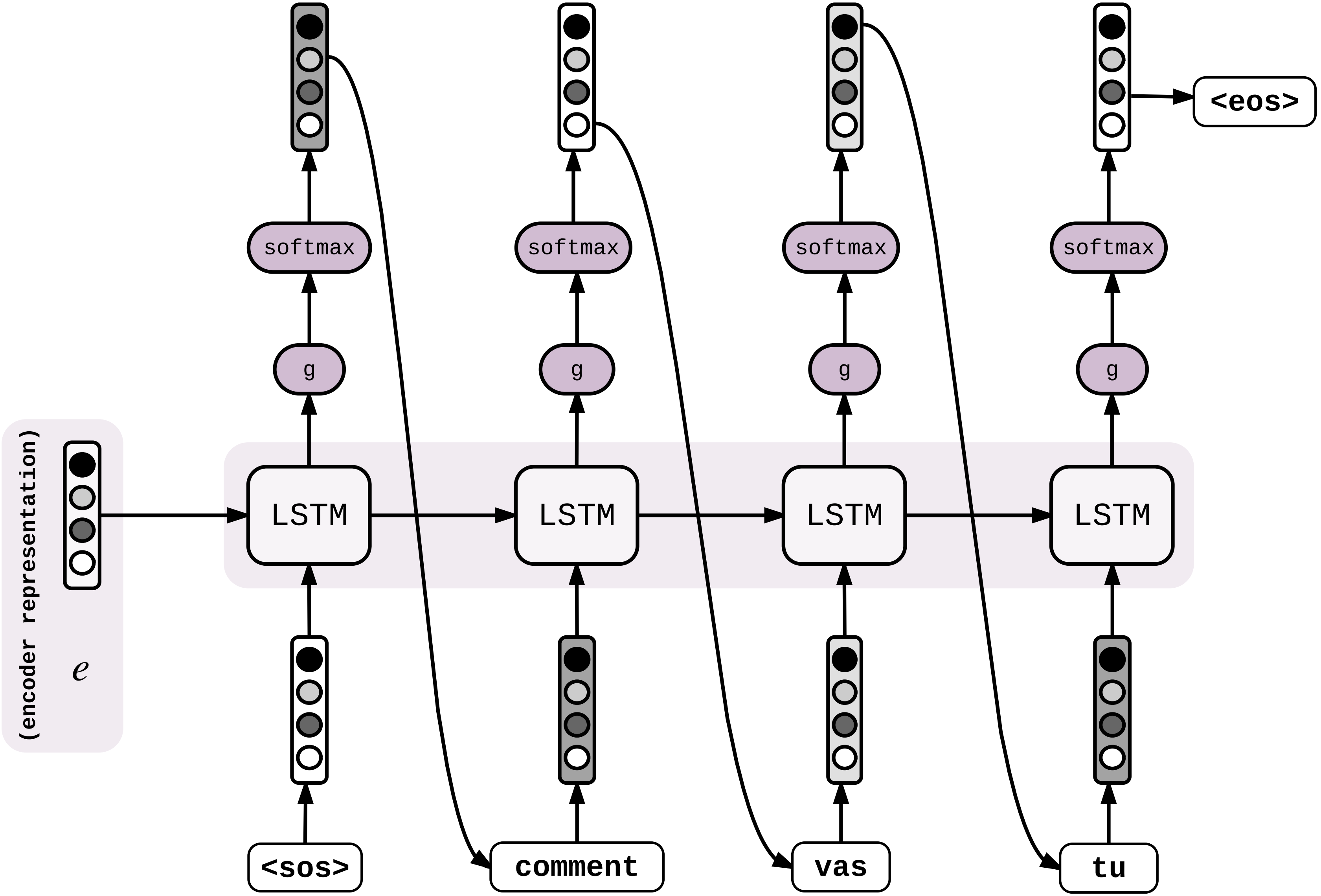


Figure source: guillaumegenthial.github.io

Decoder

- The method aims at finding out the most probable next word by maximizing the probability

$$p(y_{t+1} | y_t, y_{t-1}, \dots, y_0, e)$$

Or

$$p(y_{t+1} | y_t, y_{t-1}, \dots, y_0, x_0, x_1, \dots, x_n)$$

Attention Model: An Improvement

- Pay attention to specific parts of the input sequence, not all the input is important.
- At each time step, a context vector c_t is also added
- Change in equations is as follows.

$$h_t = LSTM(h_{t-1}, w_{i_{t-1}}, c_t)$$
$$s_t = g(h_t)$$
$$p_t = softmax(s_t)$$
$$i_t = argmax(p_t)$$

Attention Model: An Improvement

- How is this context vector computed ?

- At each time step t, compute the score of the hidden state e_t using a function $f(h_{t-1}, e_t) = \alpha_t$.

- The softmax function is applied on the sequence $\alpha = [\alpha_0, \alpha_1, \dots, \alpha_n]$. Then c_t is computed by taking the weighted average of e_t vectors.

- Equations for computing c_t are as follows.

$$\alpha_t = f(h_{t-1}, e_t)$$
$$\bar{\alpha} = softmax(\alpha)$$
$$c_t = \sum_{t=0}^{n} \alpha_t e_t$$

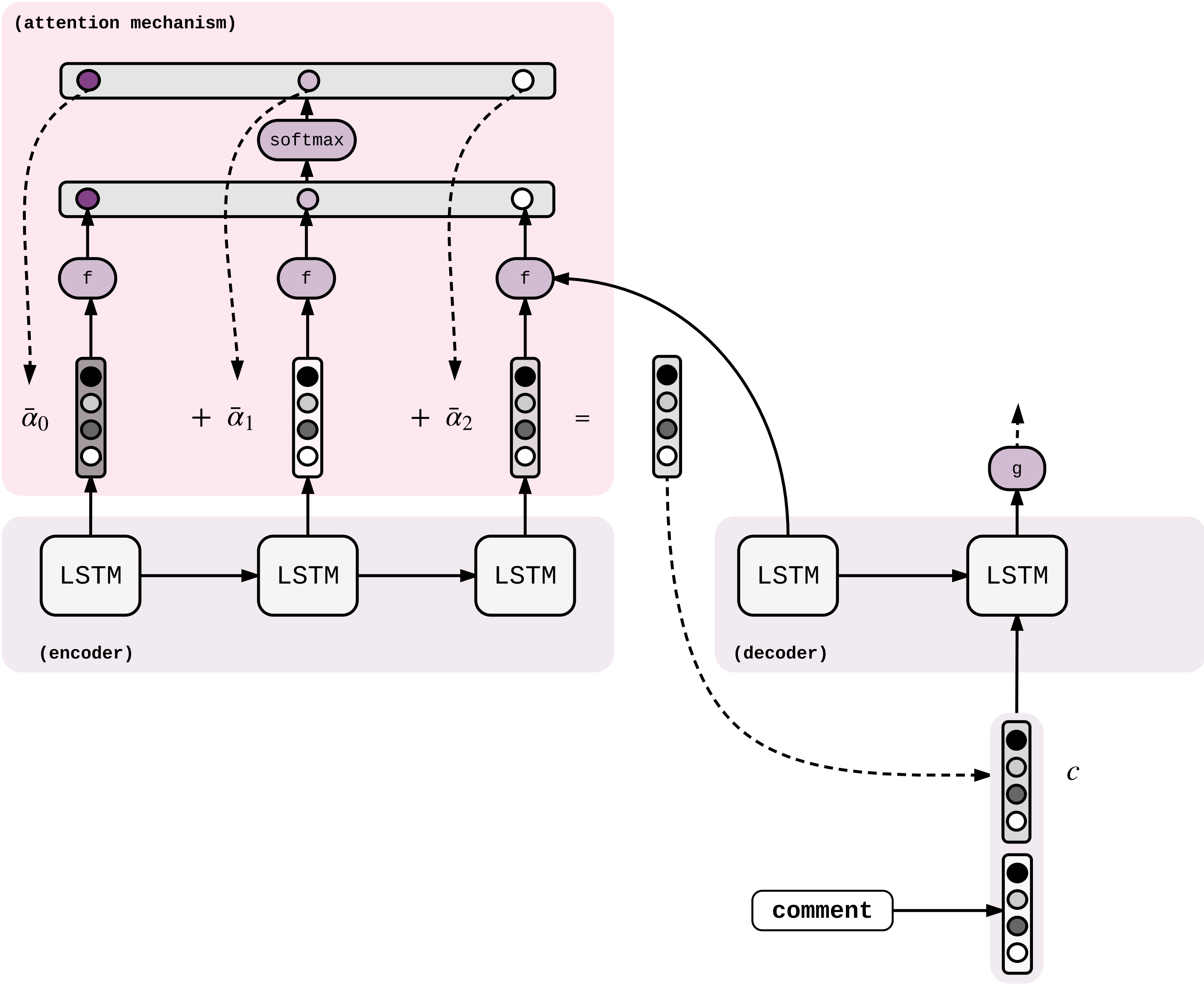
Attention Model

- Choice of the function $f(h_{t-1}, e_t) = \alpha_t$.

- General dot product:
  - $h_{t-1}{}^T e_t$

- Using weight matrix
- $h_{t-1}{}^T W\, e_t$

- Other choices are also used taking activation functions as well.



Challenges during Training

- She is visiting Mount Abu this week end.
- In this weekend, she is visiting Mount Abu.
- This weekend, she is visiting Mount Abu.
- In this weekend she will visit Mount Abu.
- She will visit Mount Abu in this weekend.
- In this weekend, she is going to visit Mount Abu.
- … …. …. ….

Challenges during Training

- And now think about big errors in estimations
- Mount Abu visit she will this weekend.
- This weekend will visit Mount Abu she.

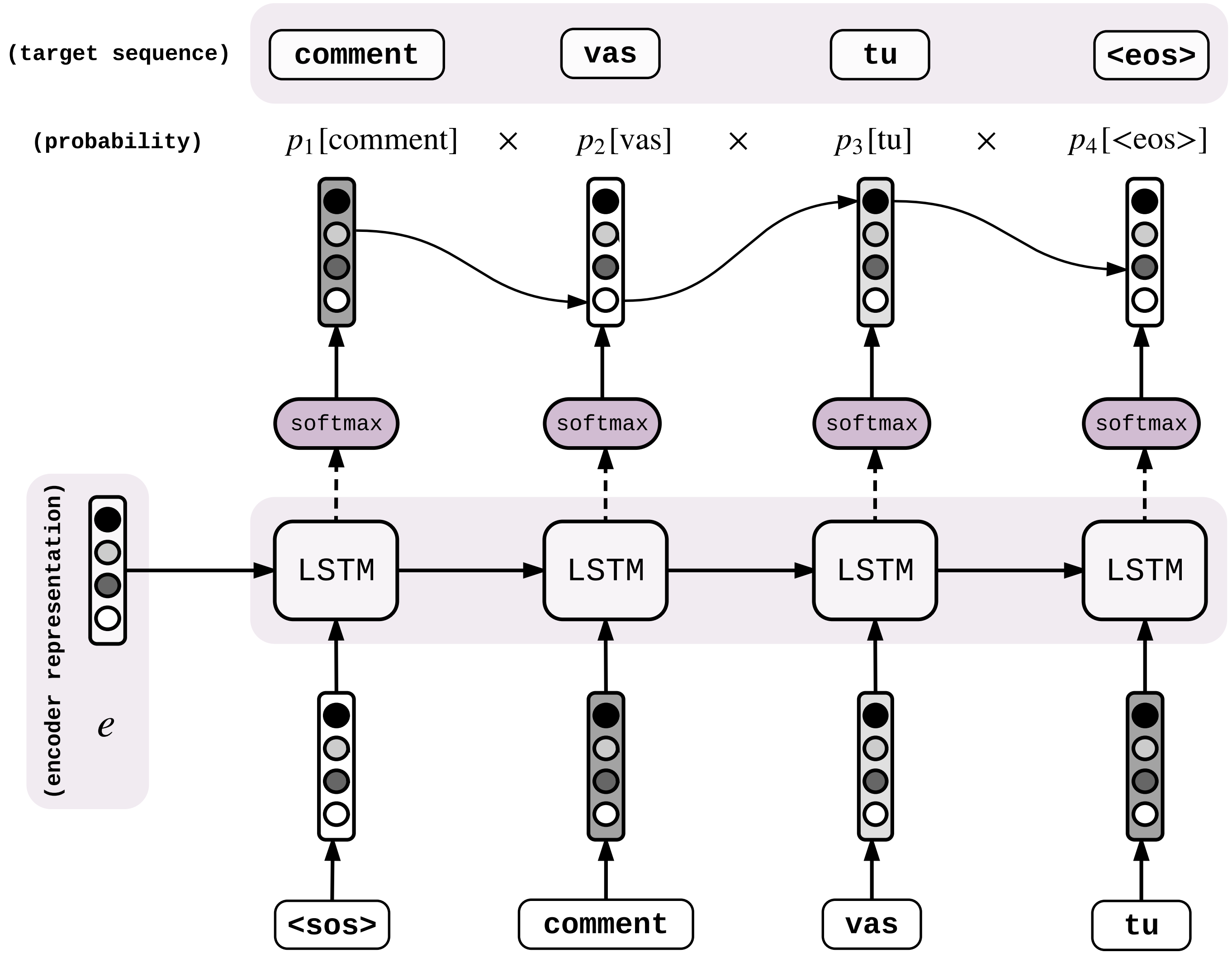
… …. …. ….

Challenges during Training

- What happens if the first time-step is not sure about what it should generate.
- What is the most likely word at the beginning of the training.
- The entire sequence will be affected.
- And the model will hardly learn anything, if the first word, or the first 2-3 words are not correctly estimated.
- Slow training and errors will be accumulated.

How to Overcome ?

One can feed the actual output sequence to the decoder during training.



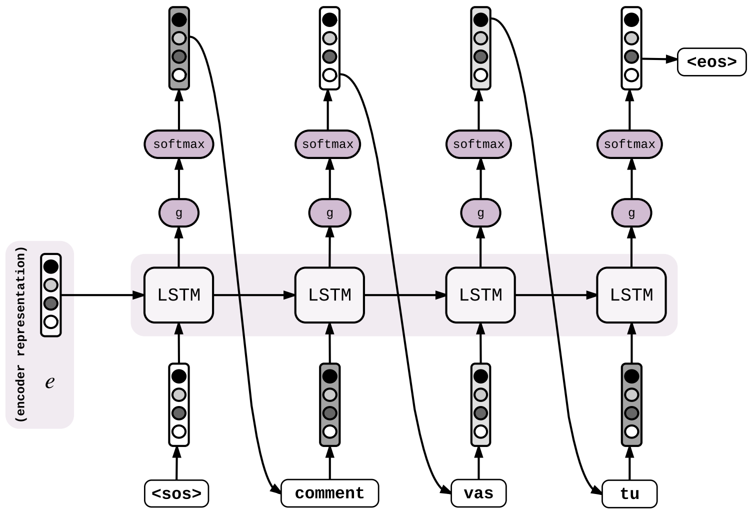
Training

- The decoder outputs vectors of probability over the vocabulary in each time step.

- Then, for a given target sequence $y_0, y_1, \dots, y_n$ its probability is computed as the product of the probabilities of each token being produced at each relevant time step.

- Train the model to maximize the probability of the target sequence (or minimize the –log probabilities).

Testing Time



- During the real time testing, that is, when we don’t have the output sequence, how do we decode ?
- Answer: Use the decoder used earlier.
- But it may accumulate errors.
- How to handle this situation ?

Beam Search Method

- A more refined way of decoding.
- Instead of only predicting the token with the best score, keep track of k hypotheses (**beam size**).
- At each new time step, for these k hypotheses we have V new possible tokens.
- A total of kV new hypotheses.
- Out of these kV new hypotheses, keep only k best hypotheses.

Beam Search Method

- Example Hypotheses.
- Let us take k =3. Then hypotheses can be
  - H = {(She is visiting) (Visiting she is) (She visiting is) }

Beam Search Method

- Consider all k- hypotheses decoded at the time step t.

$$H_t = \left\{(w_1{}^1, w_2{}^1, \dots, w_t{}^1), (w_1{}^2, w_2{}^1, \dots, w_t{}^2), \dots, \left(w_1{}^k, w_2{}^k, \dots, w_t{}^k\right)\right\}$$

- Then to select k candidates in the next time step, find out all possible combinations with all the tokens from the target vocabulary.
- C_{t+1}

$$= \bigcup_j \left\{(w_1{}^1, w_2{}^1, \dots, w_t{}^1, j), (w_1{}^2, w_2{}^2, \dots, w_t{}^2, j), \dots, \left(w_1{}^k, w_2{}^k, \dots, w_t{}^k, j\right)\right\}$$

- Now retain only k-best out of this set with the highest probability scores.

References

- Andrew Ng’s course on Sequence Models, deeplearning.ai
- https://guillaumegenthial.github.io/sequence-to-sequence.html