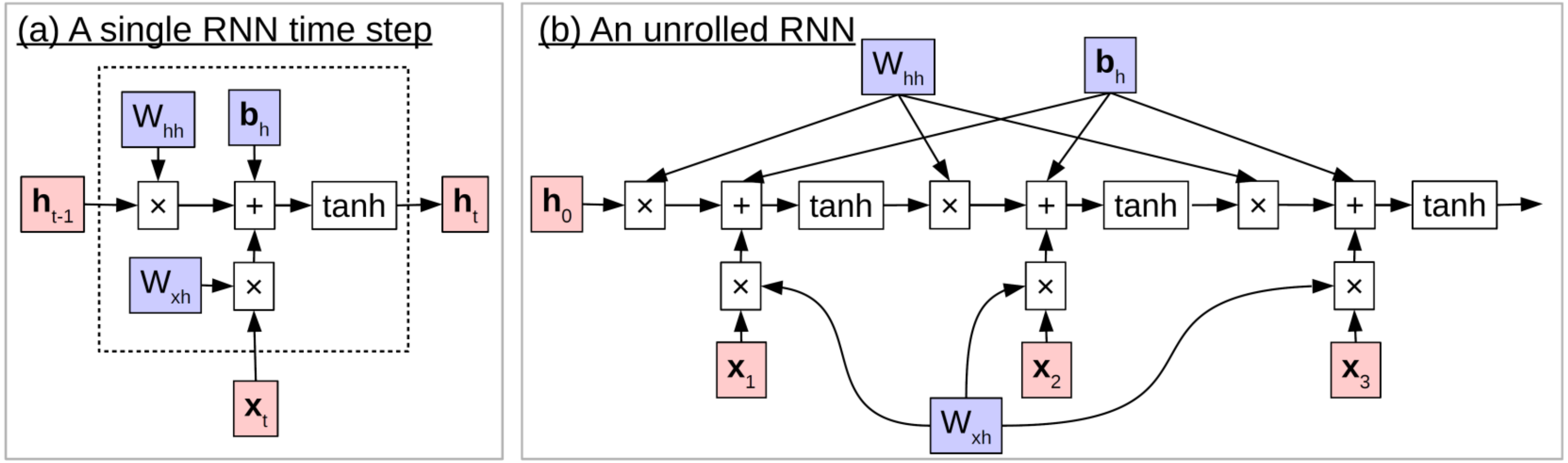
COMPUTER SCIENCE
UNIVERSITY OF MARYLAND

From Recurrent Language Models to Machine Translation

CMSC 470

Marine Carpuat

A recurrent language model

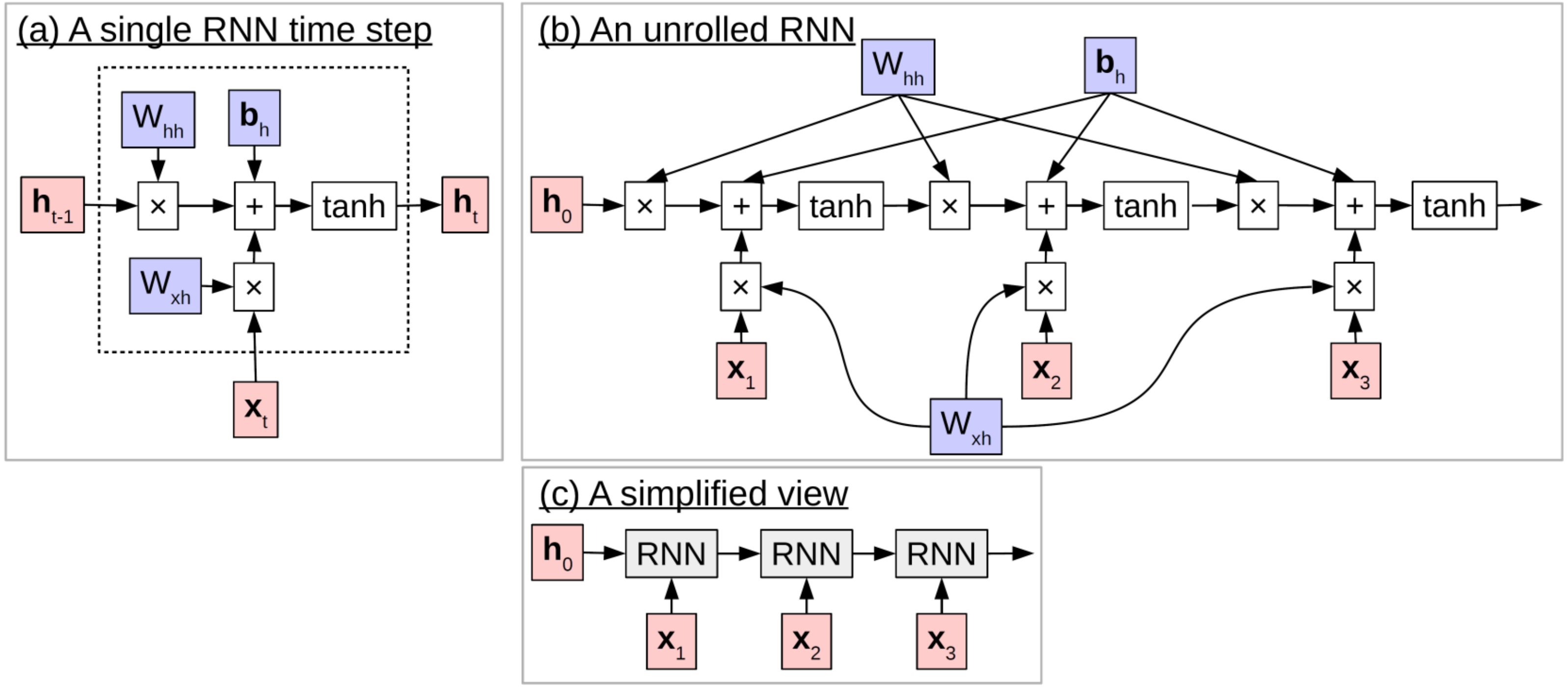


$$\boldsymbol{m}_t = M_{\cdot,e_{t-1}}$$

$$\boldsymbol{h}_t = \begin{cases} \tanh(W_{mh}\boldsymbol{m}_t + W_{hh}\boldsymbol{h}_{t-1} + \boldsymbol{b}_h) & t \geq 1, \\ \boldsymbol{0} & \text{otherwise.} \end{cases}$$

$$\boldsymbol{p}_t = \text{softmax}(W_{hs}\boldsymbol{h}_t + b_s).$$

A recurrent language model



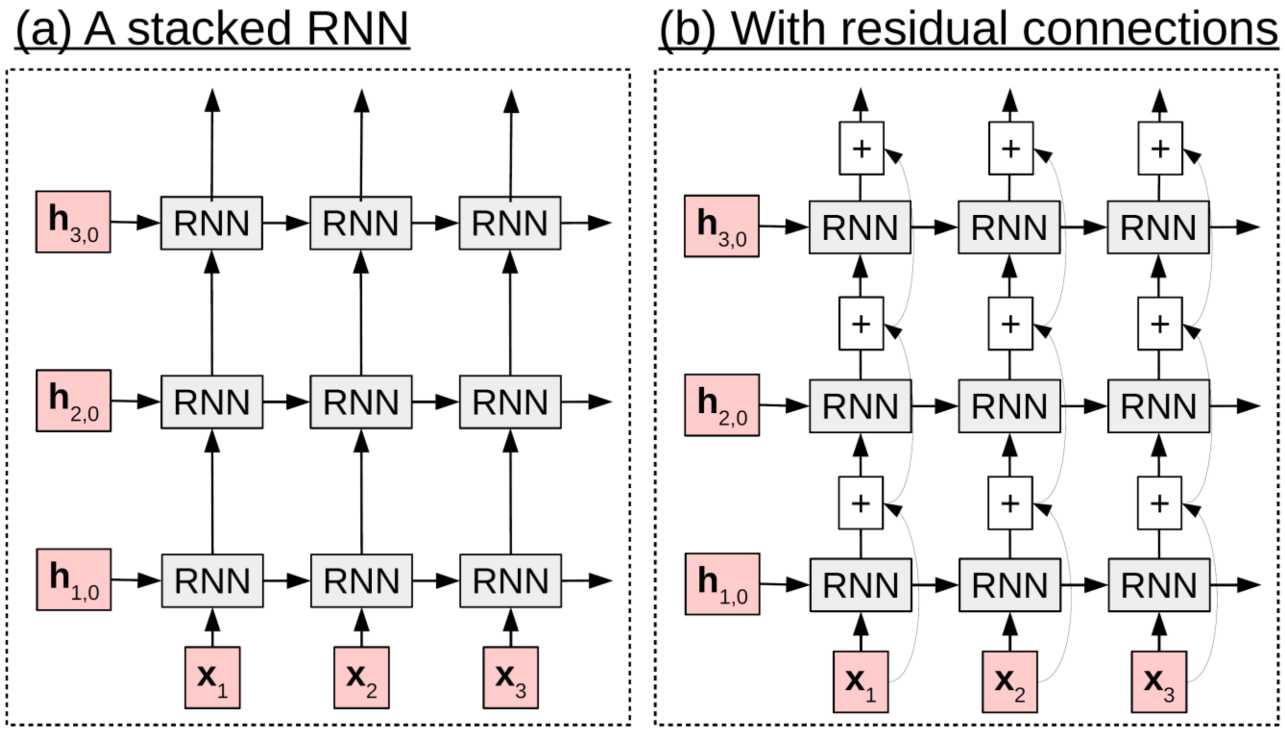
$$\boldsymbol{m}_t = M_{\cdot,e_{t-1}}$$

$$\boldsymbol{h}_t = \mathrm{RNN}(\boldsymbol{m}_t, \boldsymbol{h}_{t-1})$$

$$\boldsymbol{p}_t = \mathrm{softmax}(W_{hs}\boldsymbol{h}_t + b_s).$$

Examples of RNN variants

- LSTMs
  - Aim to address vanishing/exploding gradient issue
- Stacked RNNs



- …

Training in practice: online

Algorithm 1 A fully online training algorithm

```
procedure Online
    for several epochs of training do
        for each training example in the data do
            Calculate gradients of the loss
            Update the parameters according to this gradient
        end for
    end for
end procedure
```

Training in practice: batch

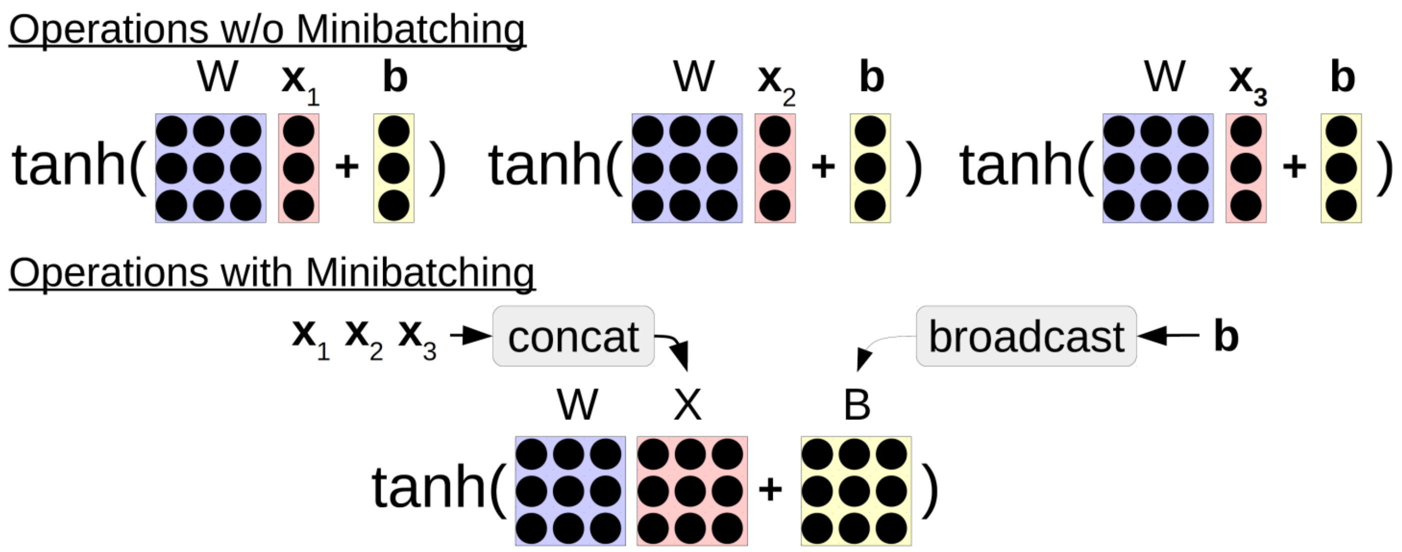
Algorithm 2 A batch learning algorithm

```
procedure Batch
    for several epochs of training do
        for each training example in the data do
            Calculate and accumulate gradients of the loss
        end for
        Update the parameters according to the accumulated gradient
    end for
end procedure
```

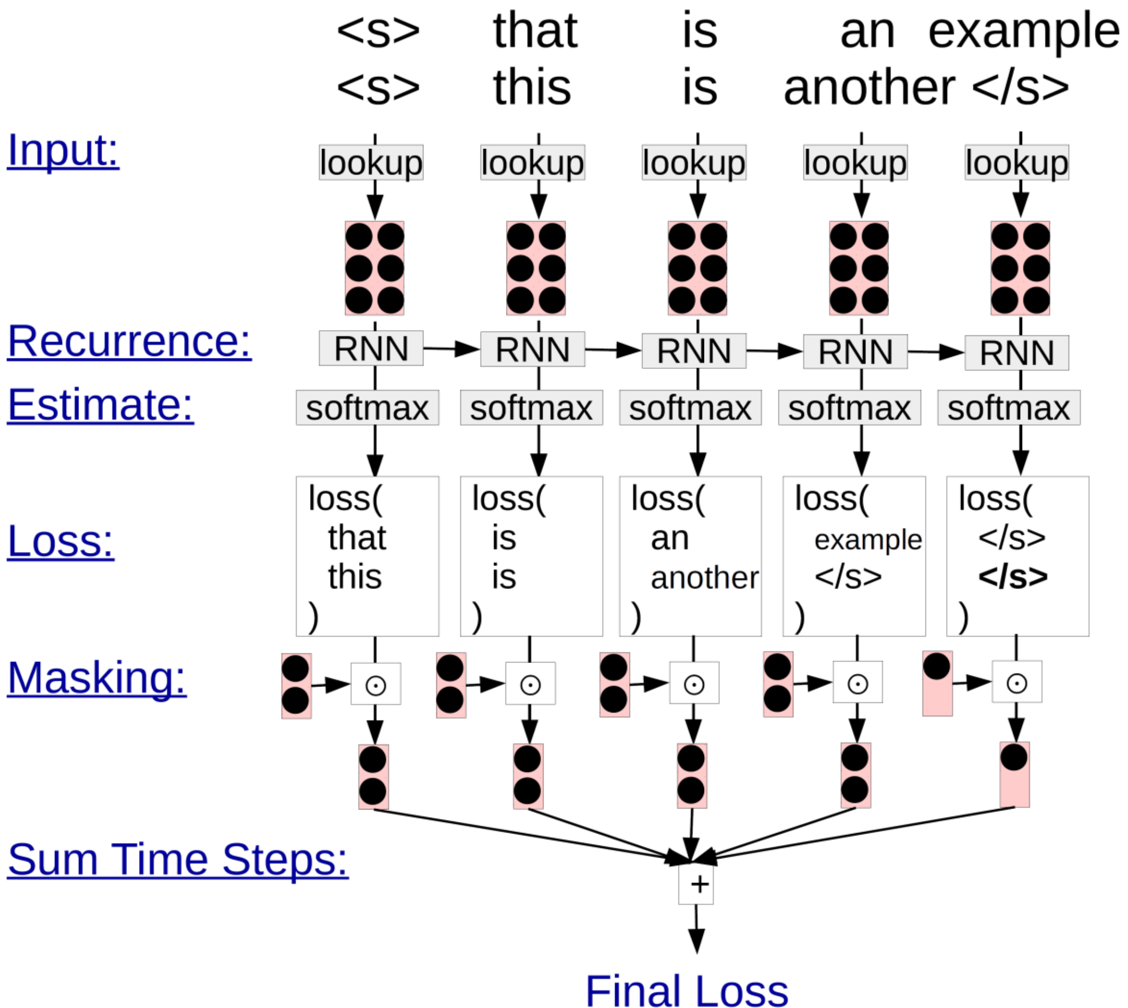
Training in practice: minibatch

- Compromise between online and batch

- Computational advantages
  - Can leverage vector processing instructions in modern hardware
  - By processing multiple examples simultaneously



Problem with minibatches: in language modeling, examples don’t have the same length



- 3 tricks
  - Padding
    - Add </s> symbol to make all sentences same length
  - Masking
    - Multiply loss function calculated over padded symbols by zero
  - + sort sentences by length

Machine Translation

- Translation system
  - Input: source sentence F
  - Output: target sentence E
  - Can be viewed as a function

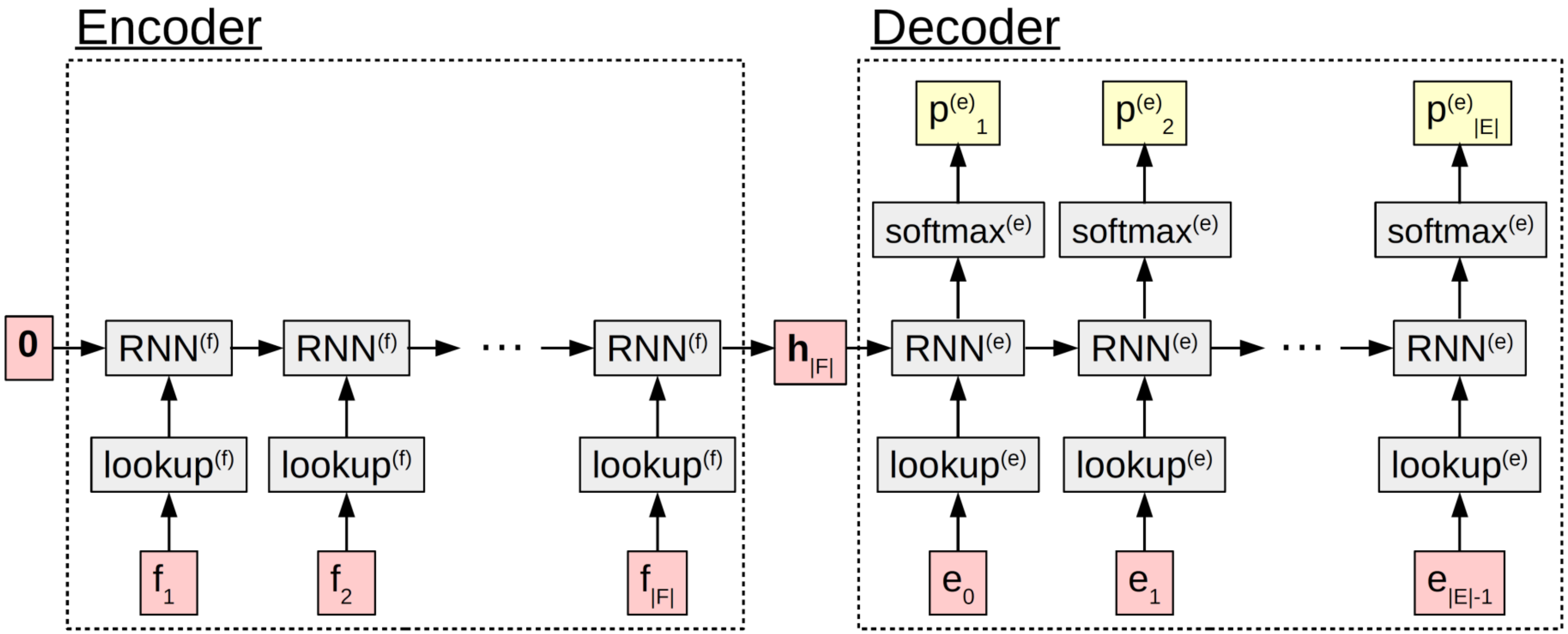
$$\hat{E} = \mathrm{mt}(F)$$

- Statistical machine translation systems

$$\hat{E} = \underset{E}{\mathrm{argmax}}\ P(E \mid F; \theta)$$

- 3 problems
  - Modeling
    - how to define P(.)?
  - Training/Learning
    - how to estimate parameters from parallel corpora?
  - Search
    - How to solve argmax efficiently?

Encoder-decoder model



Encoder-decoder model

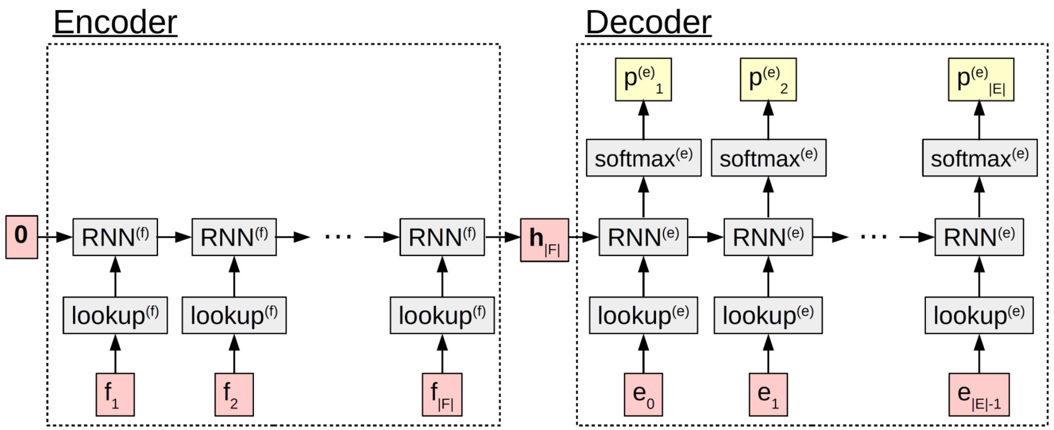
$$\boldsymbol{m}_t^{(f)} = M_{\cdot,f_t}^{(f)}$$

$$\boldsymbol{h}_t^{(f)} = \begin{cases} \mathrm{RNN}^{(f)}(\boldsymbol{m}_t^{(f)}, \boldsymbol{h}_{t-1}^{(f)}) & t \geq 1, \\ \boldsymbol{0} & \text{otherwise.} \end{cases}$$

$$\boldsymbol{m}_t^{(e)} = M_{\cdot,e_{t-1}}^{(e)}$$

$$\boldsymbol{h}_t^{(e)} = \begin{cases} \mathrm{RNN}^{(e)}(\boldsymbol{m}_t^{(e)}, \boldsymbol{h}_{t-1}^{(e)}) & t \geq 1, \\ \boldsymbol{h}_{|F|}^{(f)} & \text{otherwise.} \end{cases}$$

$$\boldsymbol{p}_t^{(e)} = \mathrm{softmax}(W_{hs}\boldsymbol{h}_t^{(e)} + b_s)$$



Generating Output

- We have a model P(E|F), how can we generate translations?
- 2 methods
  - **Sampling**: generate a random sentence according to probability distribution
  - **Argmax**: generate sentence with highest probability

Ancestral Sampling

- Randomly generate words one by one

- Until end of sentence symbol

- Done!

```
while y_{j-1} != "</s>":
  y_j ~ P(y_j | X, y_1, …, y_{j-1})
```

Greedy search

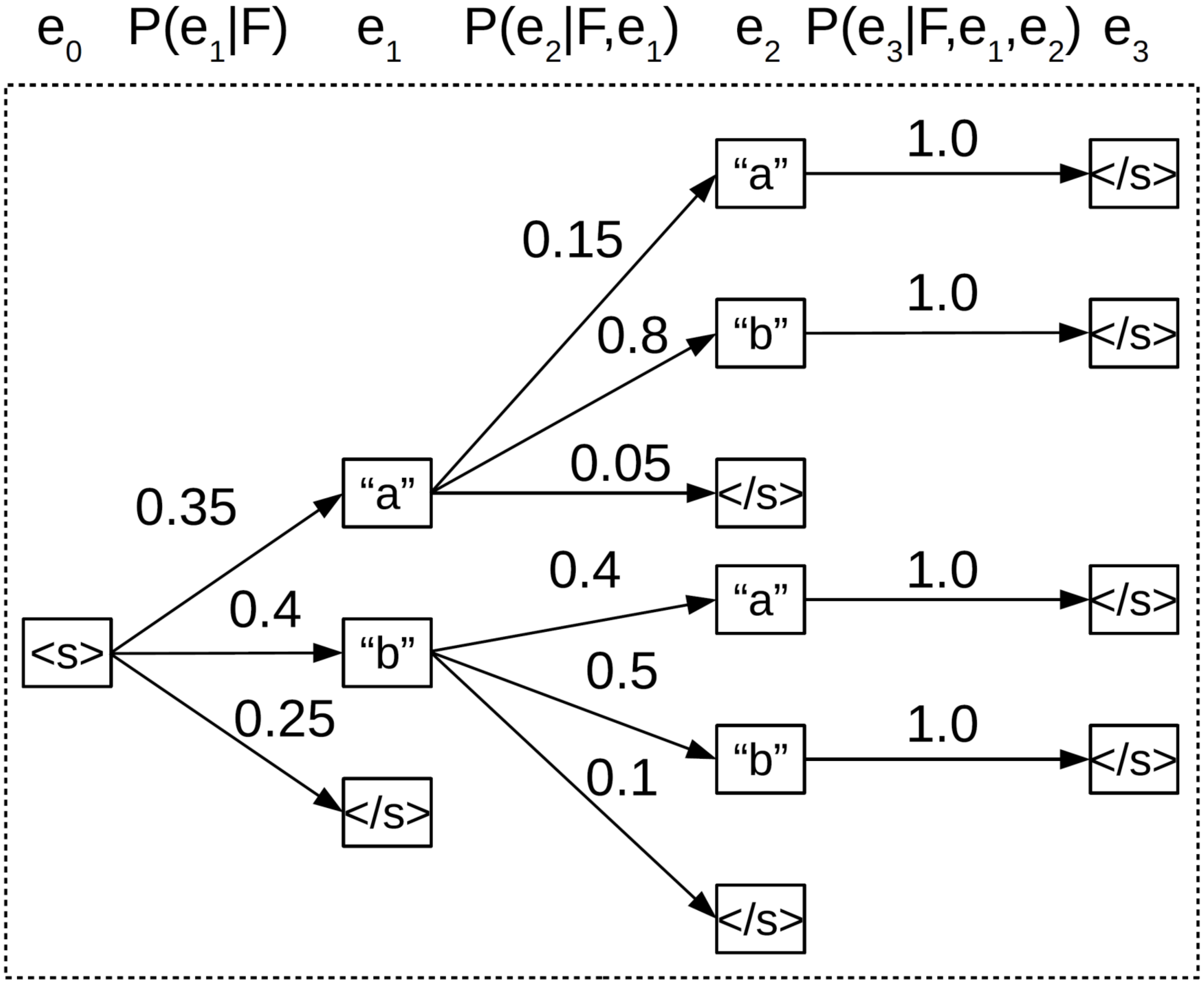
- One by one, pick single highest probability word

while y_{j-1} != “</s>”:
  $y_j = \mathrm{argmax}\ P(y_j \mid X, y_1, \ldots, y_{j-1})$

- Problems
  - Often generates easy words first
  - Often prefers multiple common words to rare words

Greedy Search

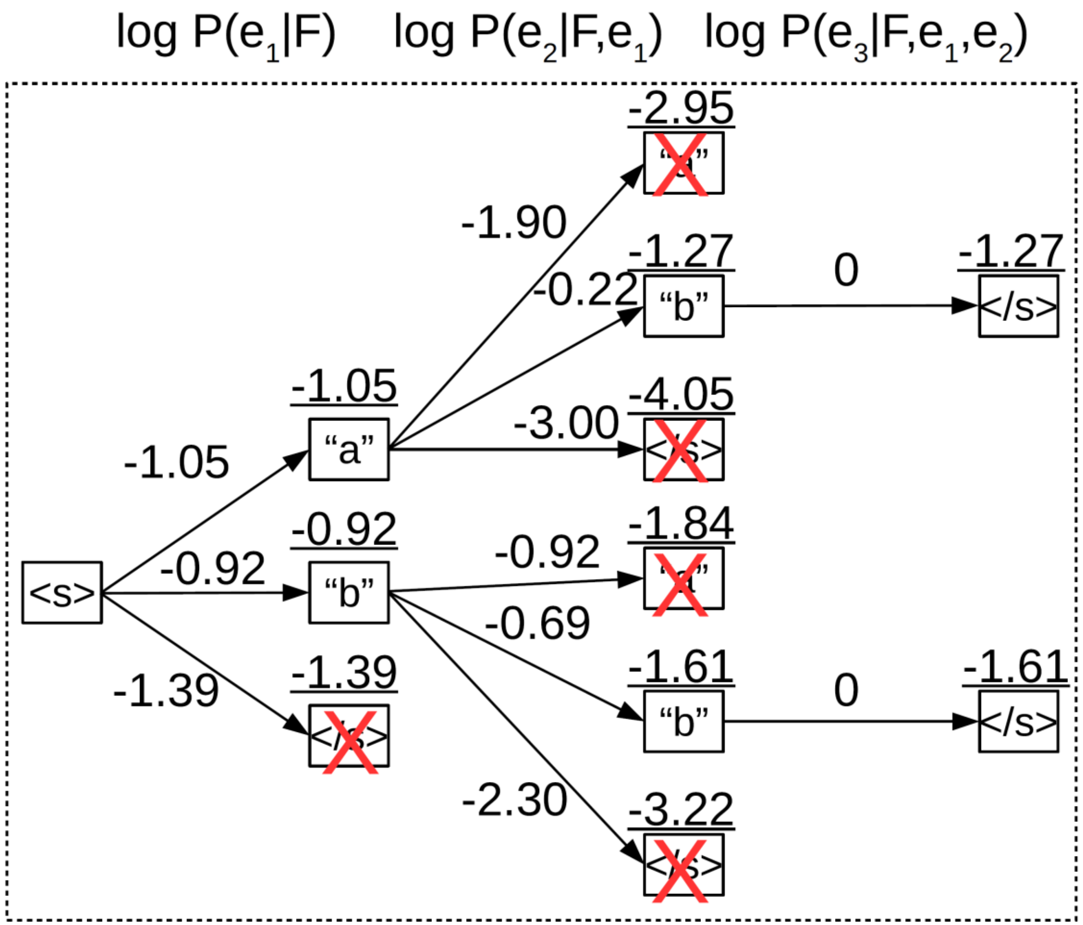
Example



Beam Search

Example with beam size b = 2

We consider b top hypotheses at each time step



Other encoder structures: Bidirectional encoder

$$\overrightarrow{\boldsymbol{h}}^{(f)}_t = \begin{cases} \overrightarrow{\mathrm{RNN}}^{(f)}(\boldsymbol{m}^{(f)}_t, \overrightarrow{\boldsymbol{h}}^{(f)}_{t+-}) & t \geq 1, \\ \mathbf{0} & \text{otherwise.} \end{cases}$$

$$\overleftarrow{\boldsymbol{h}}^{(f)}_t = \begin{cases} \overleftarrow{\mathrm{RNN}}^{(f)}(\boldsymbol{m}^{(f)}_t, \overleftarrow{\boldsymbol{h}}^{(f)}_{t+1}) & t \leq |F|, \\ \mathbf{0} & \text{otherwise.} \end{cases}$$

Motivation:
- Help bootstrap learning
- By shortening length of dependencies

$$\boldsymbol{h}^{(e)}_0 = \tanh(W_{\overrightarrow{f} e} \overrightarrow{\boldsymbol{h}}_{|F|} + W_{\overleftarrow{f} e} \overleftarrow{\boldsymbol{h}}_1 + \boldsymbol{b}_e)$$

Motivation:
- Take 2 hidden vectors from source encoder
- Combine them into a vector of size required by decoder

Introduction to Neural Machine Translation

- Neural language models review
- Sequence to sequence models for MT
  - Encoder-Decoder
  - Sampling and search (greedy vs beam search)
  - **Practical tricks**
- **Sequence to sequence models for other NLP tasks**
- **Attention mechanism**