RNN Language Modeling

CS 759/859 Natural Language Processing Lecture 15

Samuel Carton, University of New Hampshire

Last lecture

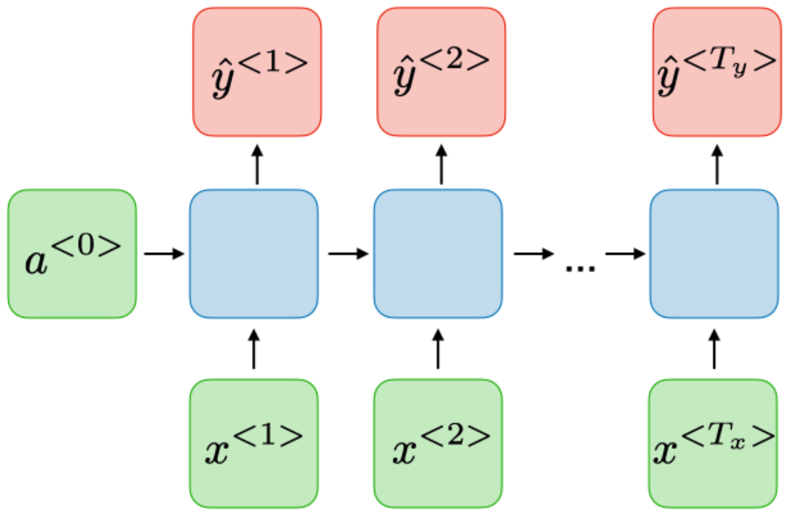
Sequence tagging

- POS tagging

LSTMs as a NLP Swiss army knife

Domain-specific word embeddings

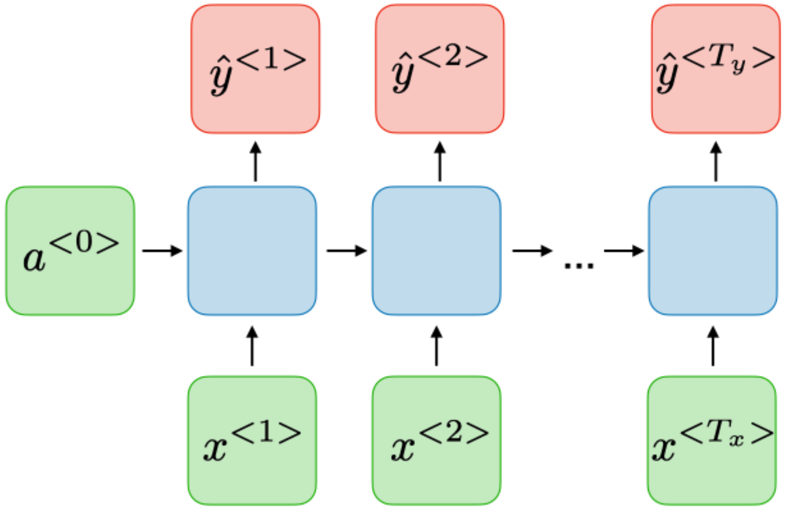
Masked loss



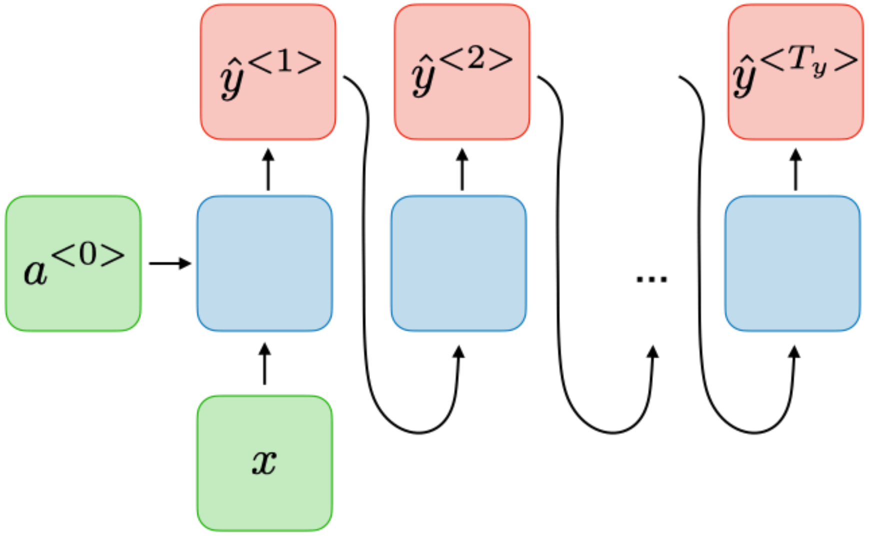
LSTMs: NLP Swiss army knife

LSTMs are exciting for us because they are the Swiss army knife of NLP models.

- Sequence classification
- Sequence tagging
- **Language modeling**
- Text-to-text (e.g. translation)



and



Review: Language modeling

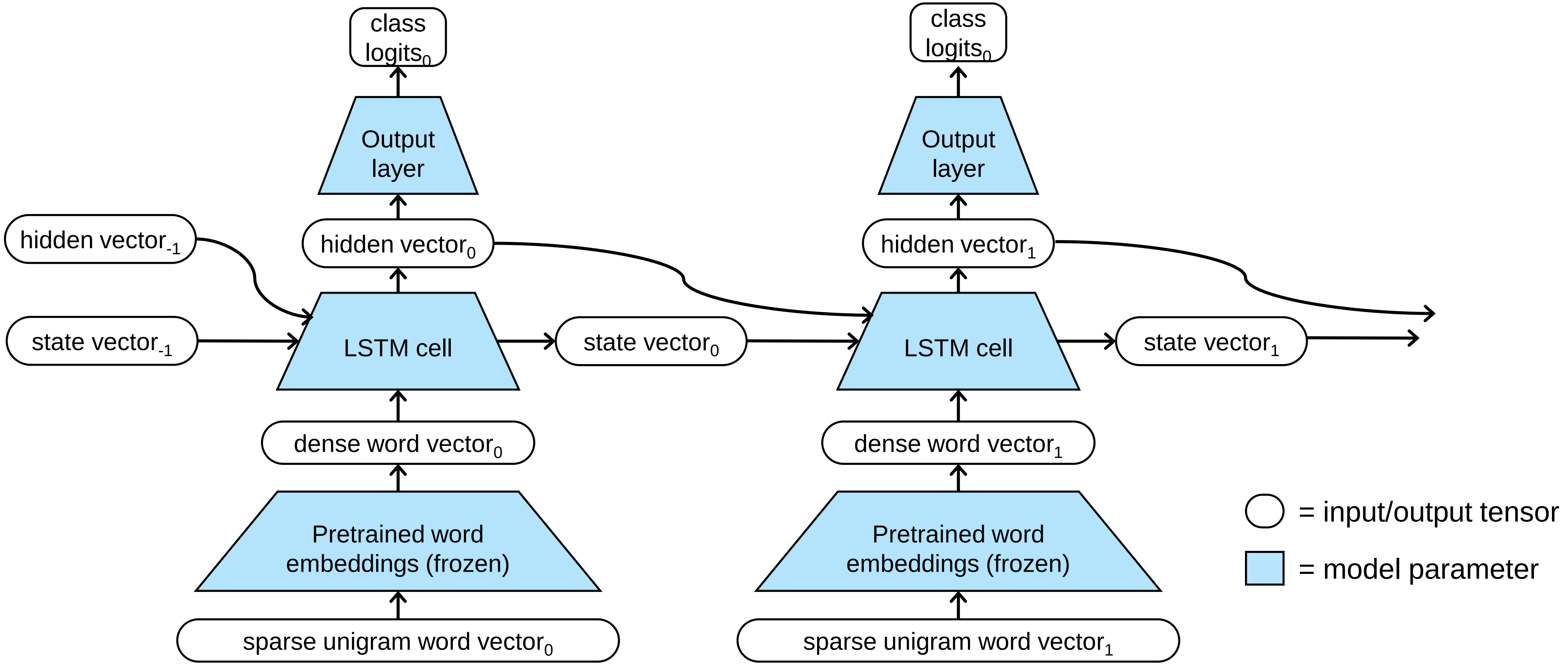
Basic idea: Given words $\{w^0, w^1, w^2, \ldots, w^{t-1}\}$, we want to be able to reliably predict w^t

If we can do this, we can:

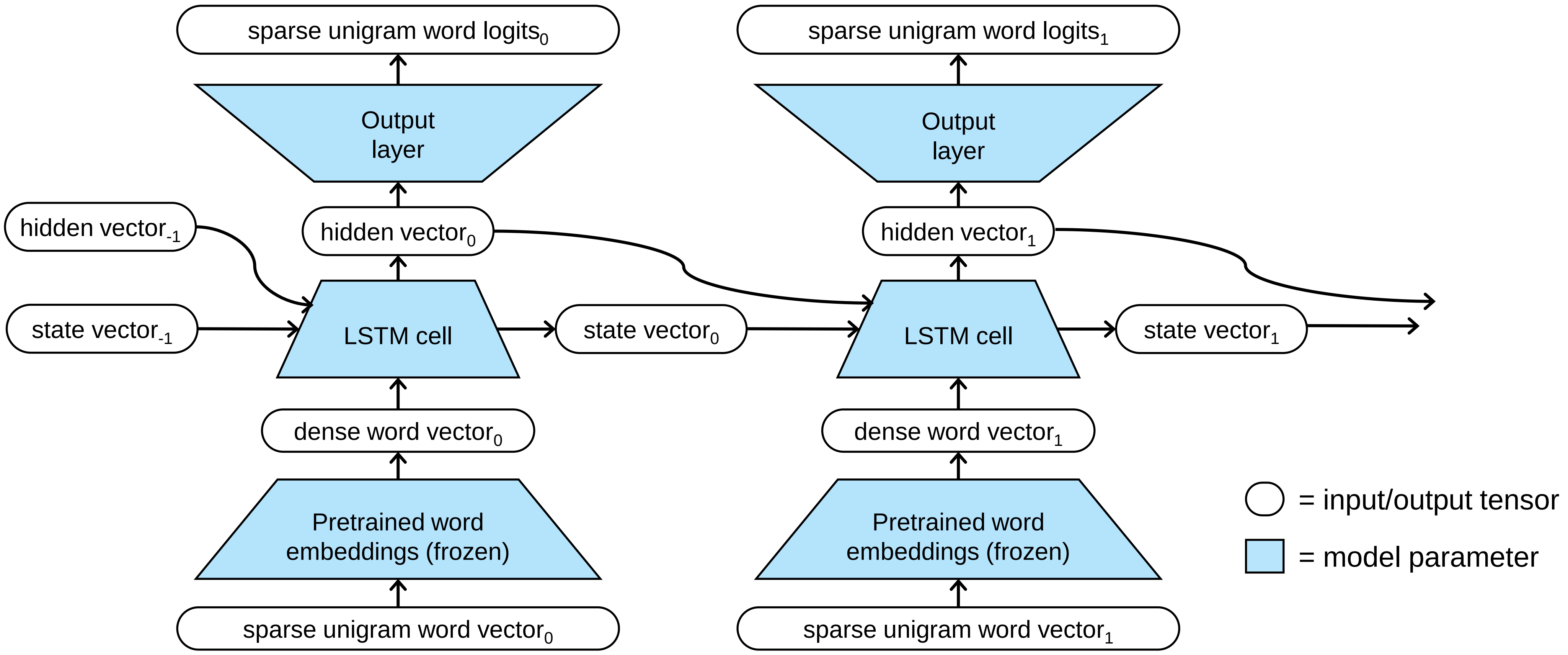
- Generate new text
- Assess the overall likelihood of a piece of text
- (In 2023) talk to the model like it is a person and make it do stuff for us
  - Prompt engineering

Lecture content borrowed from https://courses.engr.illinois.edu/cs447/fa2020/index.html

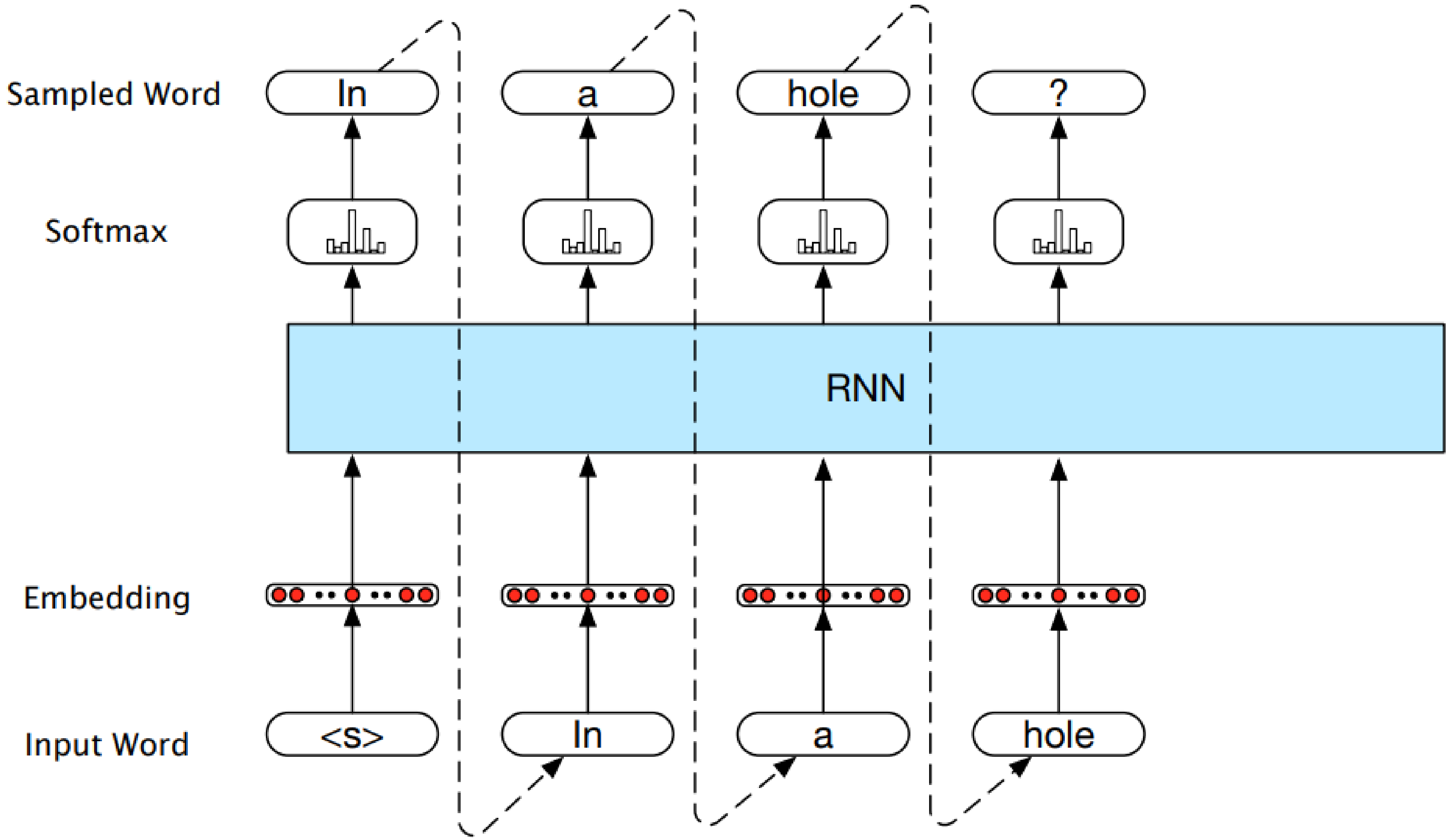
Another view of sequence tagging



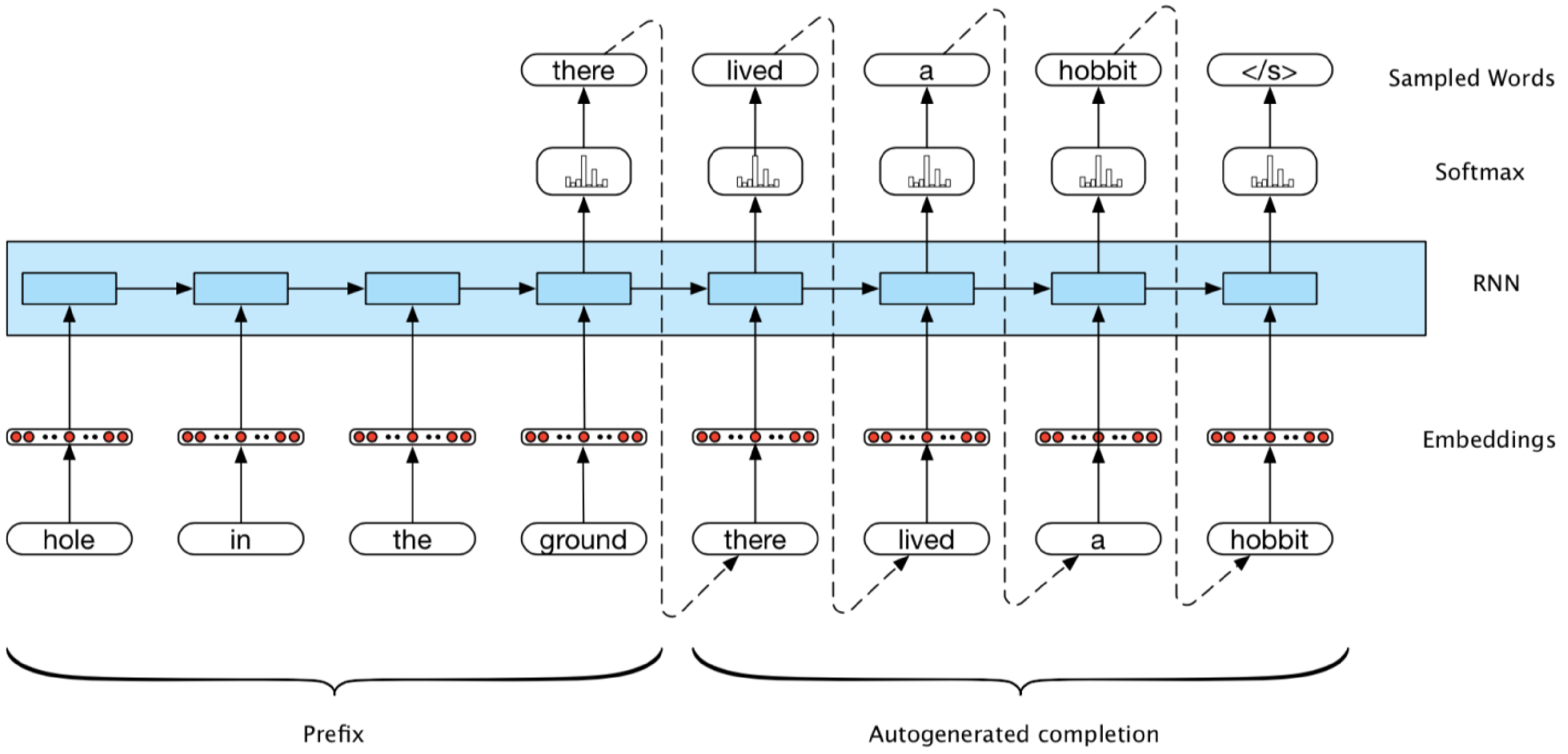
Word logits rather than class logits



Autoregressive generation



Autoregressive completion



Generating with a RNN

Also known as **decoding**: taking the output hidden-state vectors from the RNN at each step and decoding them into a sequence of actual words

Greedy decoding: always pick the most likely word at any given step

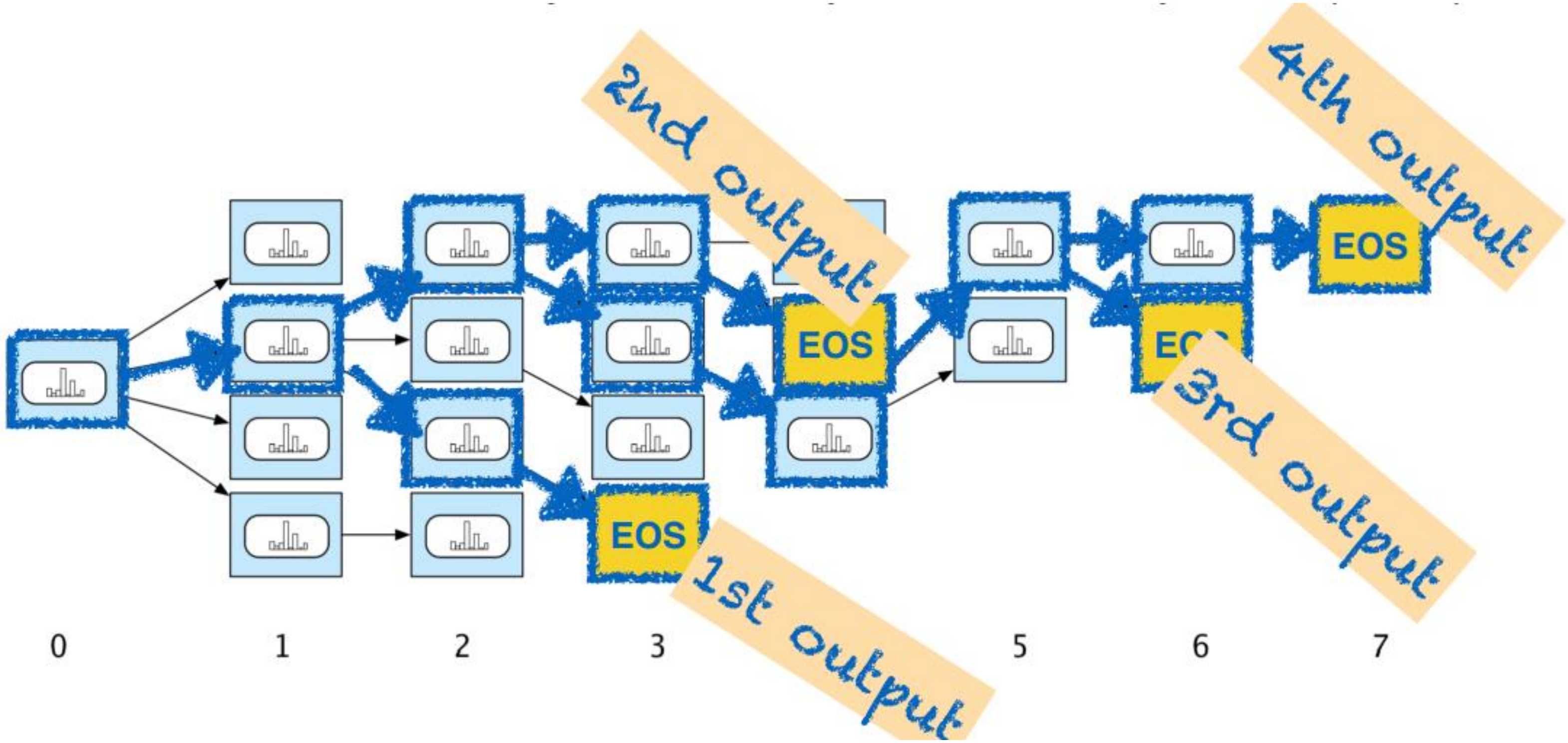
Sampling: randomly sample each word according to output logits

Beam search decoding: keep a number of possible sequences after each time step

- **Fixed-width beam**: keep top-K sequences
- **Variable-width beam**: keep all sequences whose likelihood is within certain threshold of best

Beam search decoding

- Keep the k best options around at each time step.
- Operate breadth-first: keep the k best next hypotheses among the best continuations for each of the current k hypotheses.
- Reduce beam width every time a sequence is completed (EOS)



Training RNN language models

Maximum likelihood estimation (MLE):

Given training samples $w^{(1)}w^{(2)}\ldots w^{(T)}$, find the parameters θ^* that assign **the largest probability to these training samples:**

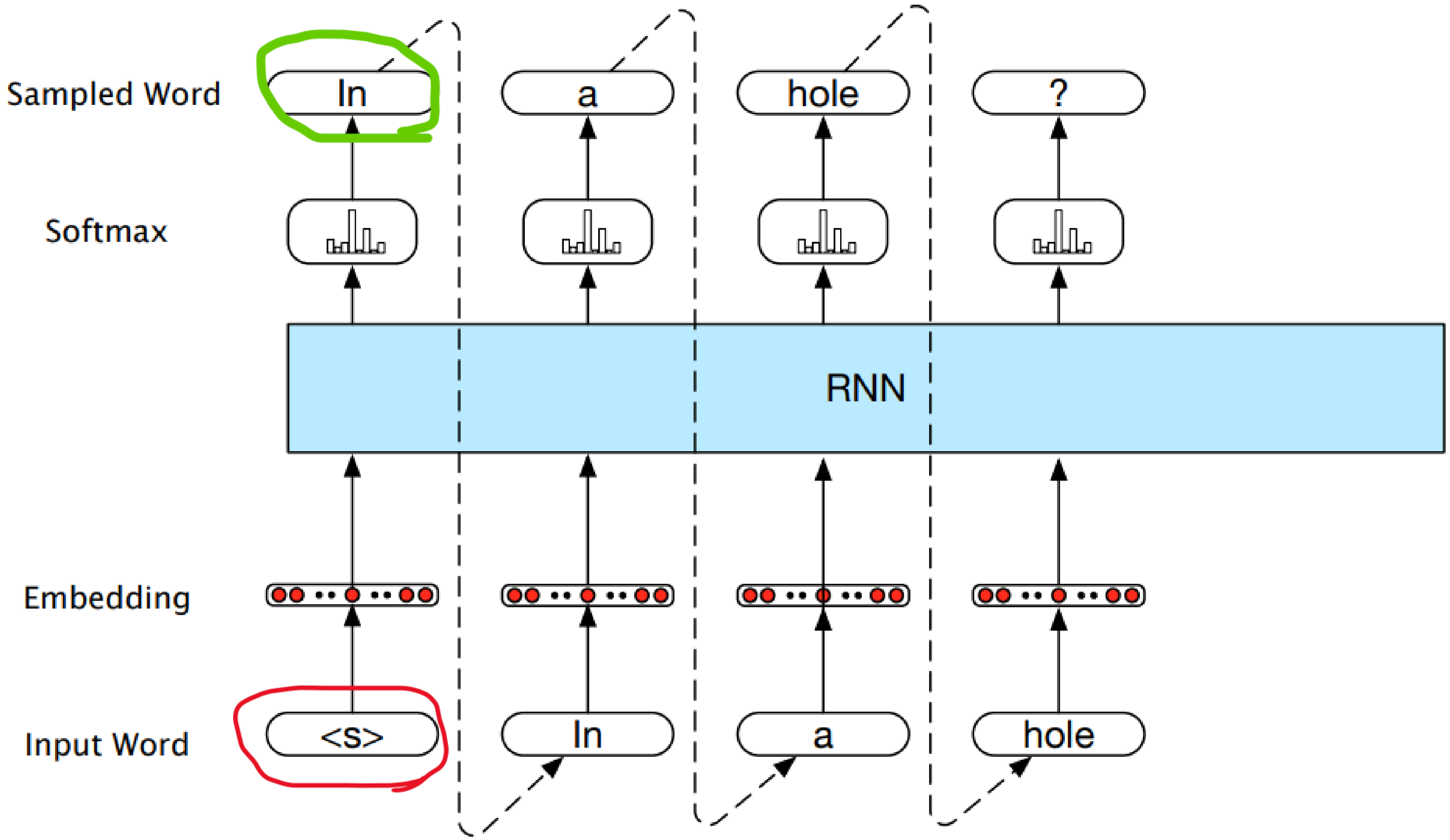
$$\theta^* = \mathrm{argmax}_\theta P_\theta(w^{(1)}w^{(2)}\ldots w^{(T)}) = \mathrm{argmax}_\theta \prod_{t=1..T} P_\theta(w^{(t)} \mid w^{(1)}\ldots w^{(t-1)})$$

- Aka “teacher forcing”

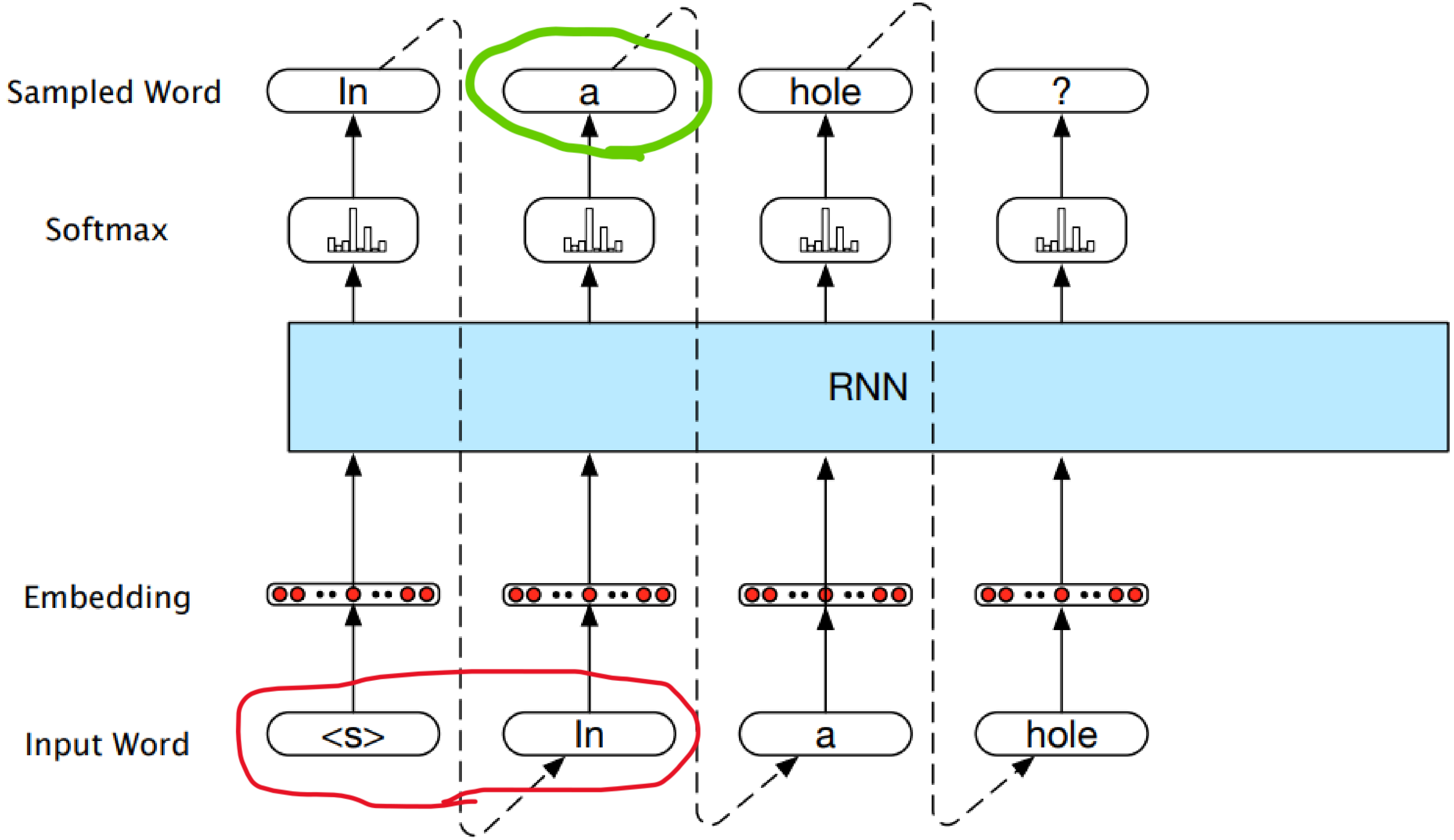
Each training sequence $\{w^0, w^1, w^2, \ldots, w^T\}$ turns into T training items:

Given $\{w^0, w^1, w^2, \ldots, w^{t-1}\}$, train model to maximize probability of w^t

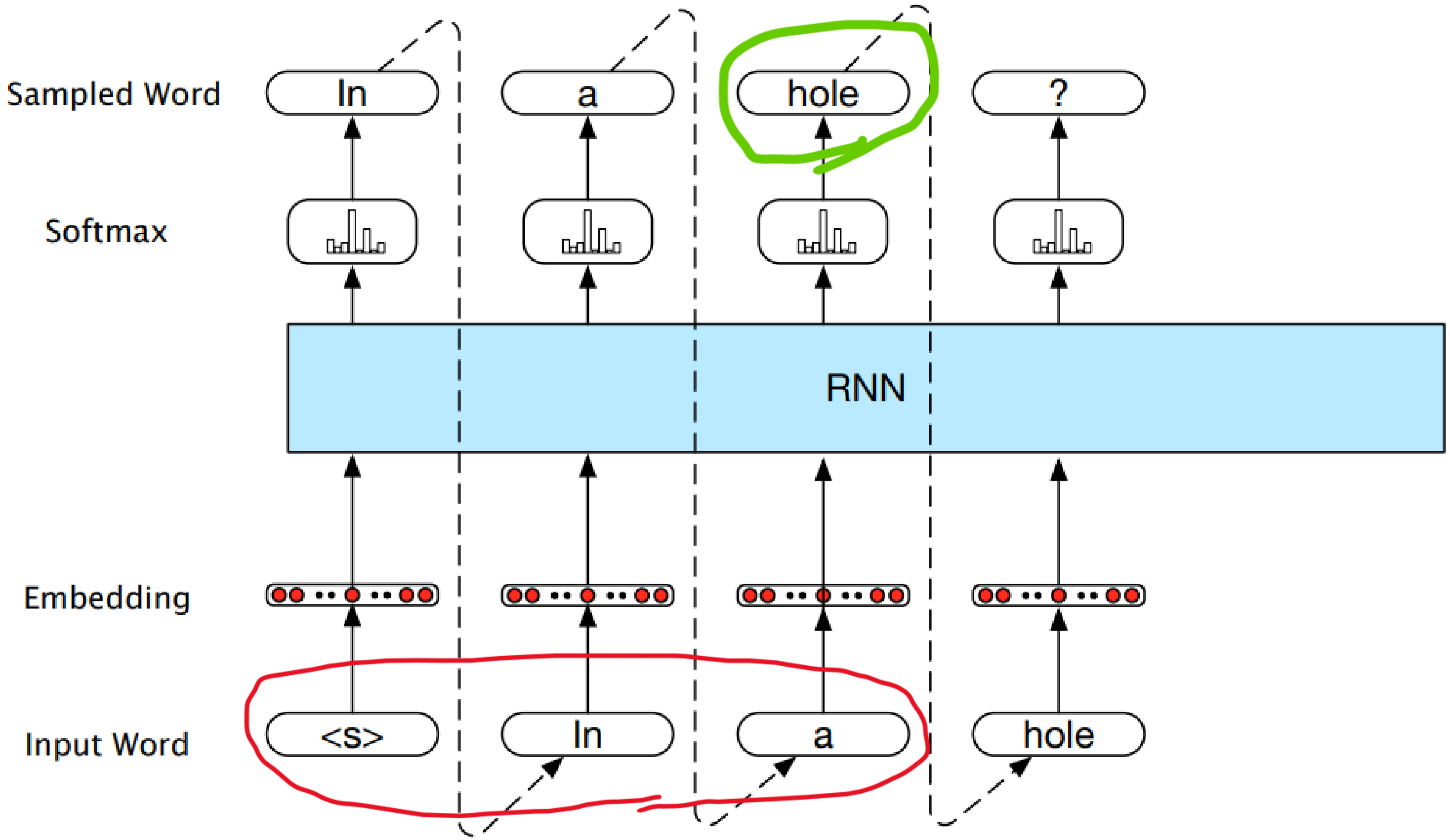
Autoregressive generation



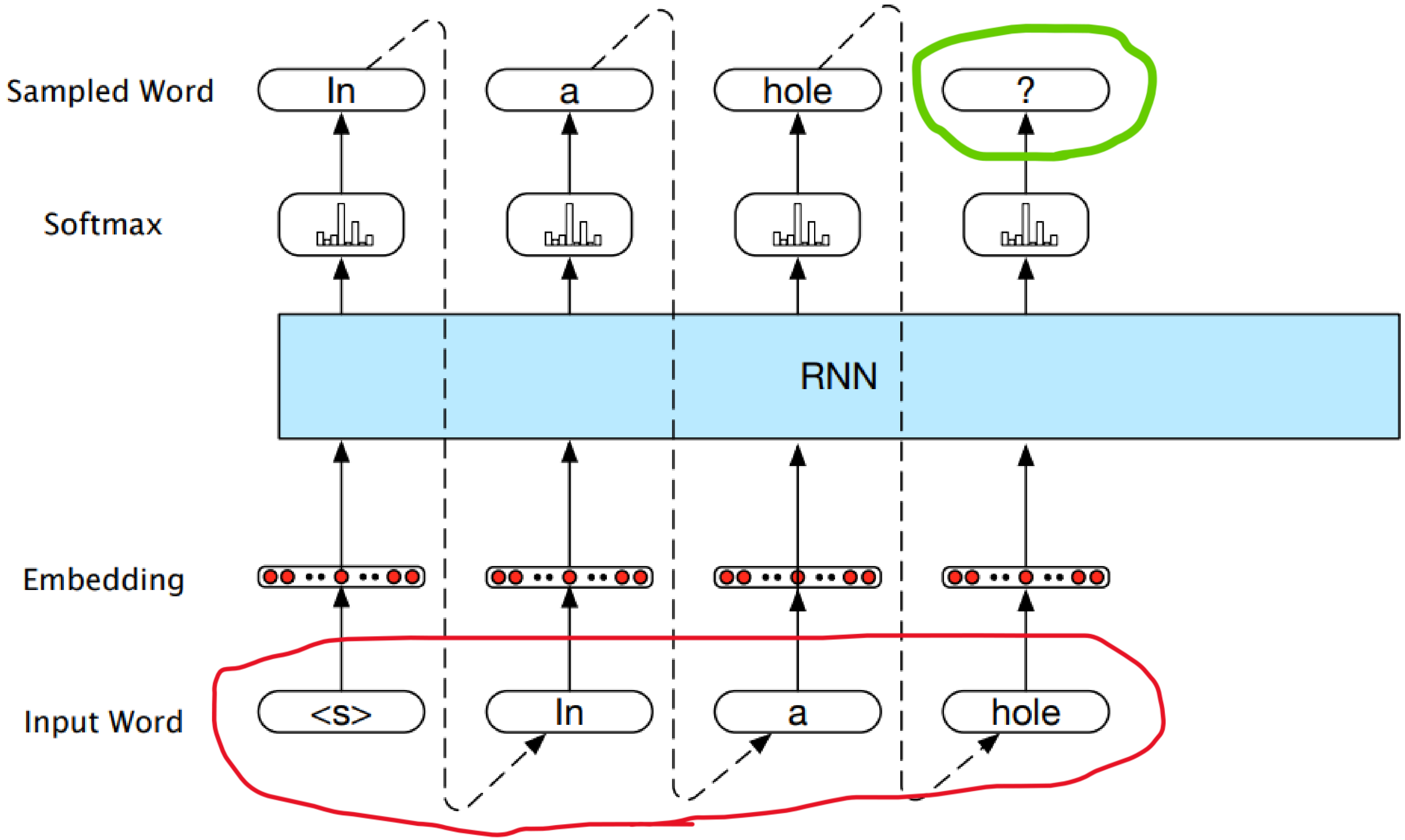
Autoregressive generation



Autoregressive generation



Autoregressive generation



Problem with teacher forcing

Neural networks (and ML models generally) don't do well with **domain shift**

Meaning, if you train the model on data that is distributed one way, it generally will not do well on data that is distributed a different way.

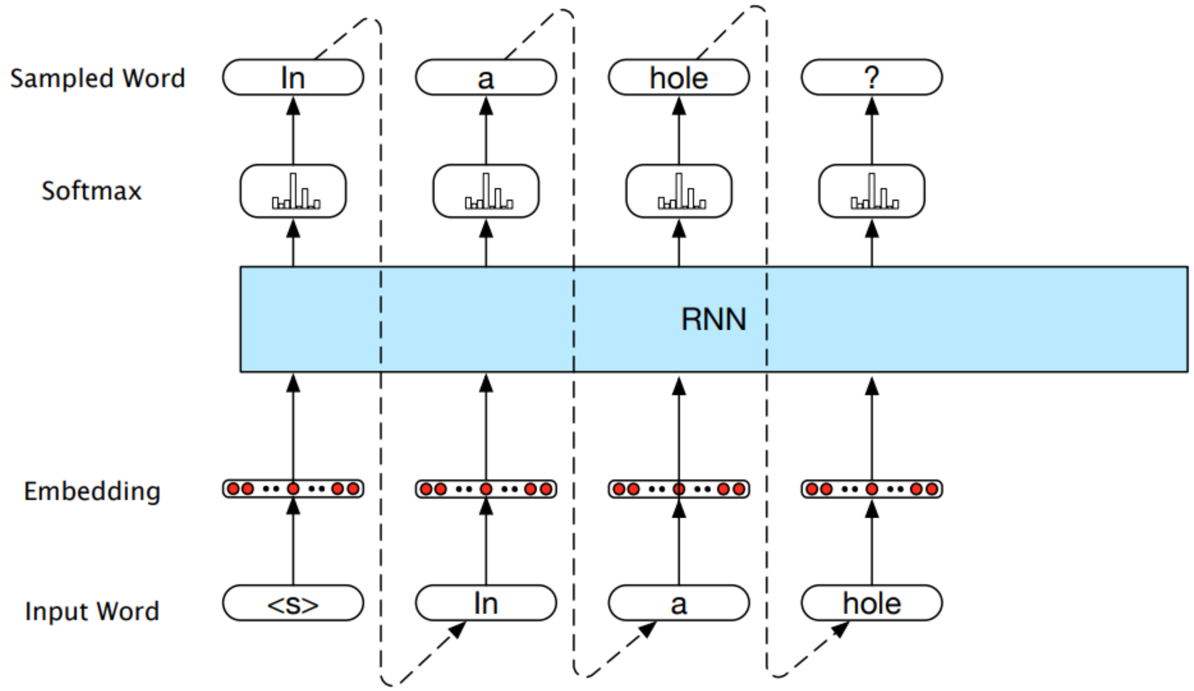
- E.g. Using a Twitter word embedding model on Reddit data
- E.g. Training sentiment detection on movie reviews but testing on product reviews
    - "Kangaroo"

How does this apply to text generation?

Problem with teacher forcing

Exposure bias: We're **training** the model to predict the next word, **given the previous true words**

But when we **generate** text, the model is looking at words **it generated**



Solutions

Minimum risk training:
(Shen et al. 2016, https://www.aclweb.org/anthology/P16-1159.pdf)
— define a loss function (e.g. negative BLEU) to compare
generated sequences against gold sequences
—Minimize risk (expected loss on training data) such that candidates outputs with a smaller loss (higher BLEU score) have higher probability.

Reinforcement learning-based approaches:
(Ranzato et al. 2016 https://arxiv.org/pdf/1511.06732.pdf)
— use BLEU as a reward (i.e. like MRT)
— perhaps pre-train model first with standard teacher forcing.

GAN-based approaches ("professor forcing")
(Goyal et al. 2016, http://papers.nips.cc/paper/6099-professor-forcing-anew-algorithm-for-training-recurrent-networks.pdf)
— combine standard RNN with an adversarial model that aims to distinguish original from generated sequences

Teacher-forcing LSTM LM in Pytorch

Code description

- Loading & preprocessing SST-2 & GloVe vectors, packaging them into a Dataset/Dataloader pair
- Implementation of an LSTM language model which uses teacher forcing for training and sampling for generation
- Testing generation and log-likelihood evaluation of sequences with the trained model

Notebook headings

Concluding thoughts

RNNs for language modeling

Generating text

- Greedy decoding
- Random sampling
- Beam search decoding

Training RNNs

- Teacher forcing
    - Exposure bias
- Alternatives
    - Minimum risk, reinforcement learning, GANs