Introduction to Language Models

MUSTAFA HAJIJ

Language modeling

Language Modeling is the task of predicting what word comes next.

More formally, given a sequence of words $x_1, x_2, \ldots, x_t$, compute the probability distribution of the next word x_{t+1} :

$$P(x_{t+1} | \, x_1, x_2, \ldots, x_t)$$

x_{t+1} can be any word in the vocabulary $V = \{w_1, \ldots, w_{|V|}\}$

Language modeling

Language Modeling is the task of predicting what word comes next.

More formally, given a sequence of words $x_1, x_2, \ldots, x_t$, compute the probability distribution of the next word x_{t+1} :

$$P(x_{t+1} | \, x_1, x_2, \ldots, x_t)$$

x_{t+1} can be any word in the vocabulary $V = \{w_1, \ldots, w_{|V|}\}$

A system that models this PDF is called a Language Model

Language modeling

Given a sequence of symbols or tokens from V, represented as $X = \{x_1, x_2, \ldots, x_{\mathrm{n}}\}$, a language model L assigns a probability P(X) to that sequence.

P(X) can be computed using the chain rule of probability:

$$P(X) = P(x_1) \times P(x_2 \mid x_1) \times P(x_3 \mid x_1, x_2) \times \ldots \times P(x_{\mathrm{n}} \mid x_1, x_2, \ldots, x_{\mathrm{n}-1})$$

From this perspective, a language model is a system that assigns a probability to a piece of text

Language modeling Usage

During the usage phase, the language model predicts the most likely token x_n given the preceding context x_1, x_2, ..., x_{n-1} by selecting the highest probability from the computed probabilities. Alternatively, sampling techniques can be used to generate likely sequences based on the probability distribution defined by the language model.

N-gram language model

An n-gram is a chunk of n consecutive words.

1. Trigrams (n=3):
   1. "The cat is"
   2. "cat is sleeping"
2. Four-grams (n=4):
   1. "The cat is sleeping"

Question: How do we get these n-gram and (n-1)-gram probabilities?
Answer: By counting them in some large corpus of text

$$P(x_n \mid x_1, x_2, \ldots, x_{n-1}) = \text{Count}((x_1, x_2, \ldots, x_{n-1}, x_n)) / \text{Count}((x_1, x_2, \ldots, x_{n-1}))$$

N-gram language model

An n-gram is a chunk of n consecutive words.

1. Trigrams (n=3):
    1. "The cat is"
    2. "cat is sleeping"
2. Four-grams (n=4):
    1. "The cat is sleeping"

- **Idea:** Collect statistics about how frequent different n-grams are and use these to predict next word.

Question: How do we get these n-gram and (n-1)-gram probabilities?
Answer: By counting them in some large corpus of text

$$P(x_n \mid x_1, x_2, \ldots, x_{n-1}) = \mathrm{Count}((x_1, x_2, \ldots, x_{n-1}, x_n)) \,/\, \mathrm{Count}((x_1, x_2, \ldots, x_{n-1}))$$

How can we use the above as a generative model?

Question: How do we get these n-gram and (n-1)-gram probabilities?
Answer: By counting them in some large corpus of text

$$P(x_n \mid x_1, x_2, \ldots, x_{n-1}) = \text{Count}((x_1, x_2, \ldots, x_{n-1}, x_n)) / \text{Count}((x_1, x_2, \ldots, x_{n-1}))$$

How to build a neural language model?

We start by considering a fixed-window neural Language Model :

output distribution

$$\hat{\boldsymbol{y}} = \mathrm{softmax}(\boldsymbol{U}\boldsymbol{h} + \boldsymbol{b}_2) \in \mathbb{R}^{|V|}$$

hidden layer

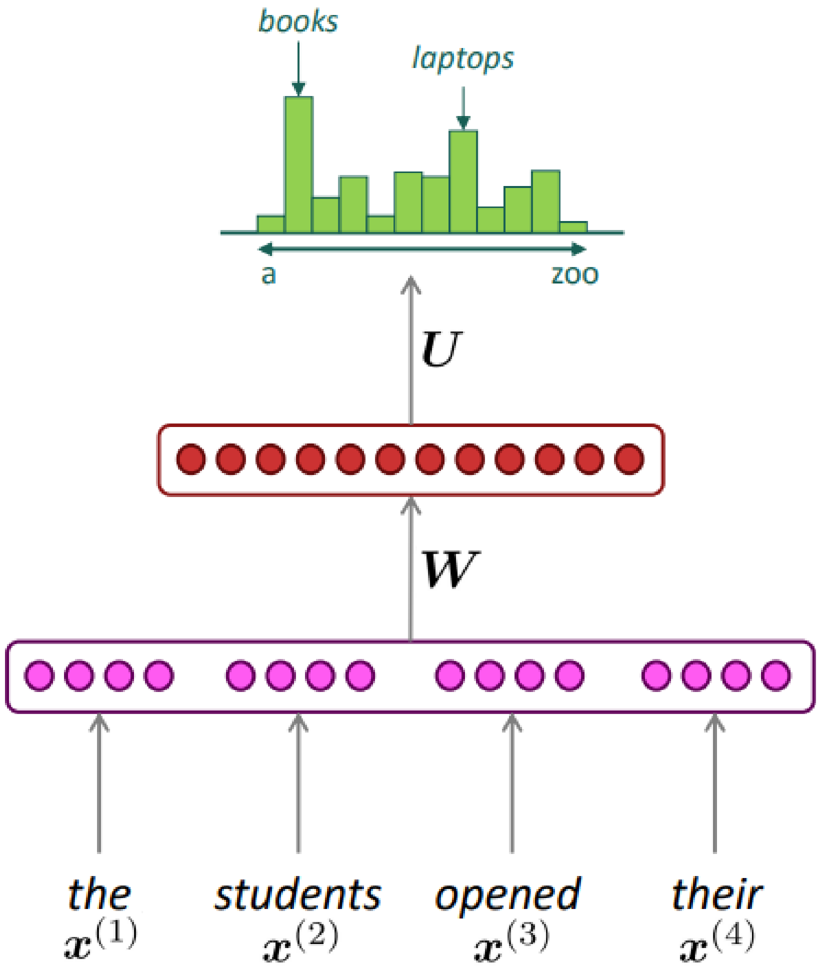
$$\boldsymbol{h} = f(\boldsymbol{W}\boldsymbol{e} + \boldsymbol{b}_1)$$

concatenated word embeddings

$$\boldsymbol{e} = [\boldsymbol{e}^{(1)}; \boldsymbol{e}^{(2)}; \boldsymbol{e}^{(3)}; \boldsymbol{e}^{(4)}]$$

words / one-hot vectors

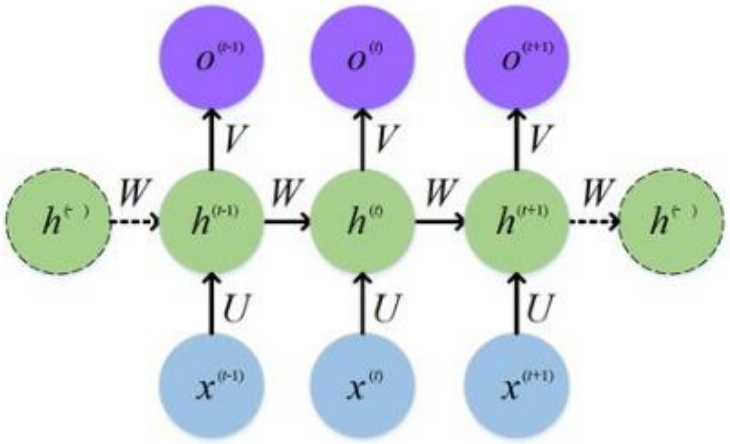
$$\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \boldsymbol{x}^{(3)}, \boldsymbol{x}^{(4)}$$



RNNs

Given an input sequence of length T, denoted as $x = (x_1, x_2, \ldots, x_t)$, and a hidden state h_{t-1} at the previous time step t-1, the RNN computes the hidden state h_t and output y_t at the current time step t as follows:

$$h_t = \sigma(Wh * h_t + Wx * x_t + b1)$$
$$y_t = Wy * h_t + b2$$



- **Wx** is the weight matrix connecting the input x_t to the hidden state h_t
- **Wh** is the weight matrix connecting the previous hidden state h_{t-1} to the current hidden state h_t
- **Wy** is the weight matrix connecting the hidden state h_t to the output y_t
- **σ** is the activation function, commonly the hyperbolic tangent (tanh) function

How to build a neural language model?

How to handle arbitrary sequence ? RNN and other variants

output distribution

$$\hat{\boldsymbol{y}}^{(t)} = \text{softmax}\left(\boldsymbol{U}\boldsymbol{h}^{(t)} + \boldsymbol{b}_2\right) \in \mathbb{R}^{|V|}$$

hidden states

$$\boldsymbol{h}^{(t)} = \sigma\left(\boldsymbol{W}_h\boldsymbol{h}^{(t-1)} + \boldsymbol{W}_e\boldsymbol{e}^{(t)} + \boldsymbol{b}_1\right)$$

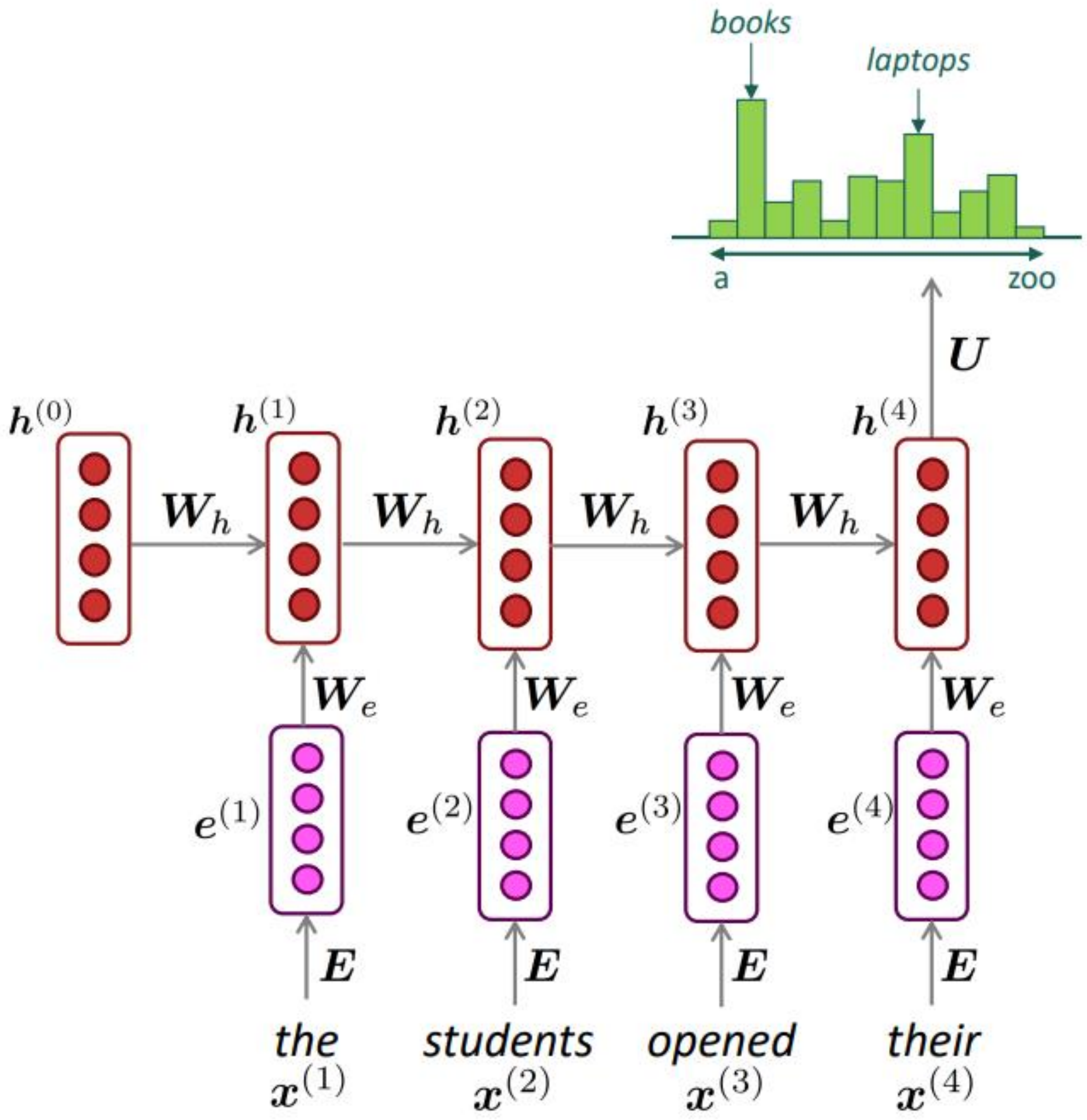
$\boldsymbol{h}^{(0)}$ is the initial hidden state

word embeddings

$$\boldsymbol{e}^{(t)} = \boldsymbol{E}\boldsymbol{x}^{(t)}$$

words / one-hot vectors

$$\boldsymbol{x}^{(t)} \in \mathbb{R}^{|V|}$$



Refs

https://web.stanford.edu/class/cs224n/slides/cs224n-2023-lecture05-rnnlm.pdf

(11) (PDF) Audio visual speech recognition with multimodal recurrent neural networks (researchgate.net)