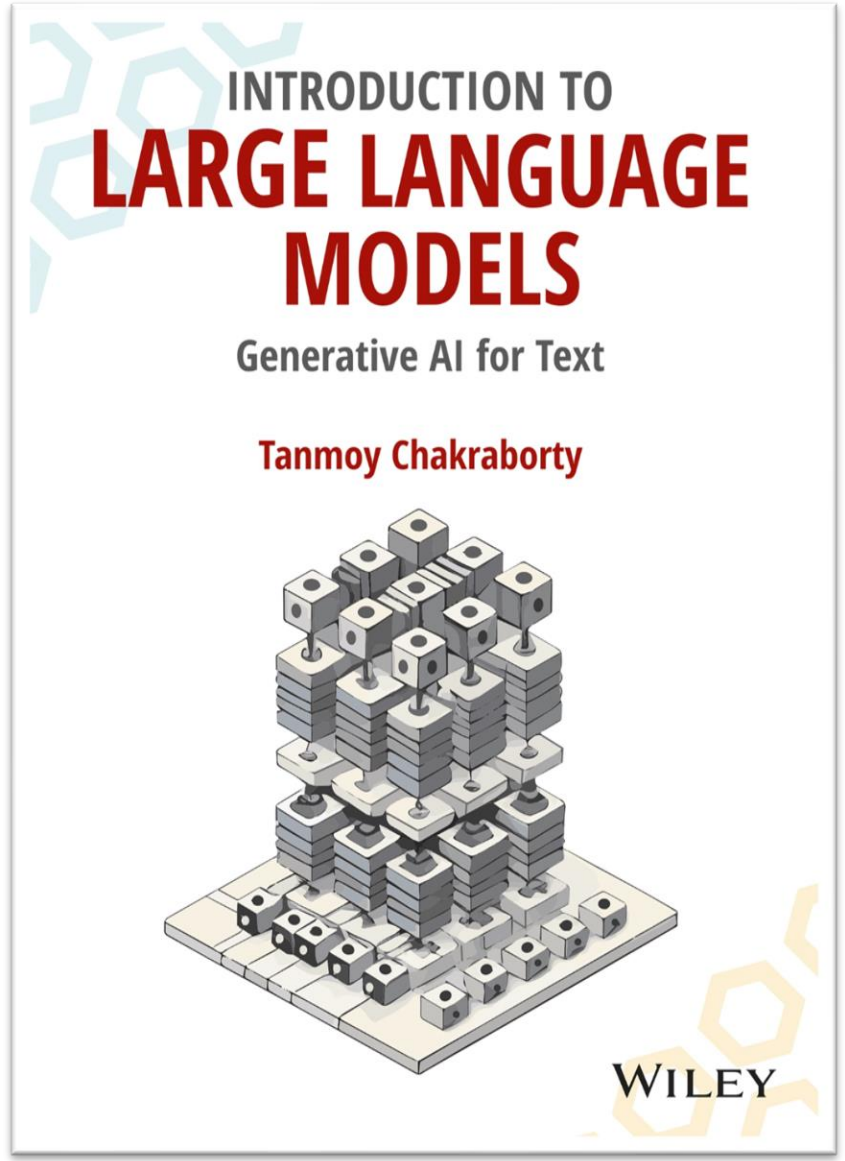
Neural Language Models

Tanmoy Chakraborty
Associate Professor, IIT Delhi
https://tanmoychak.com/

Slides are adopted from the Stanford course 'NLP with DL' by C. Manning

- ▼ Chapter 05. Neural Language Models
  - ▶ 5.1 Convolutional Neural Networks
  - ▶ 5.2 Recurrent Neural Networks
  - ▶ 5.3 Sequence-to-Sequence Models
  - ▶ 5.4 Attention Mechanisms
  - 5.5 Limitations of Neural Language Models
  - 5.6 Summary

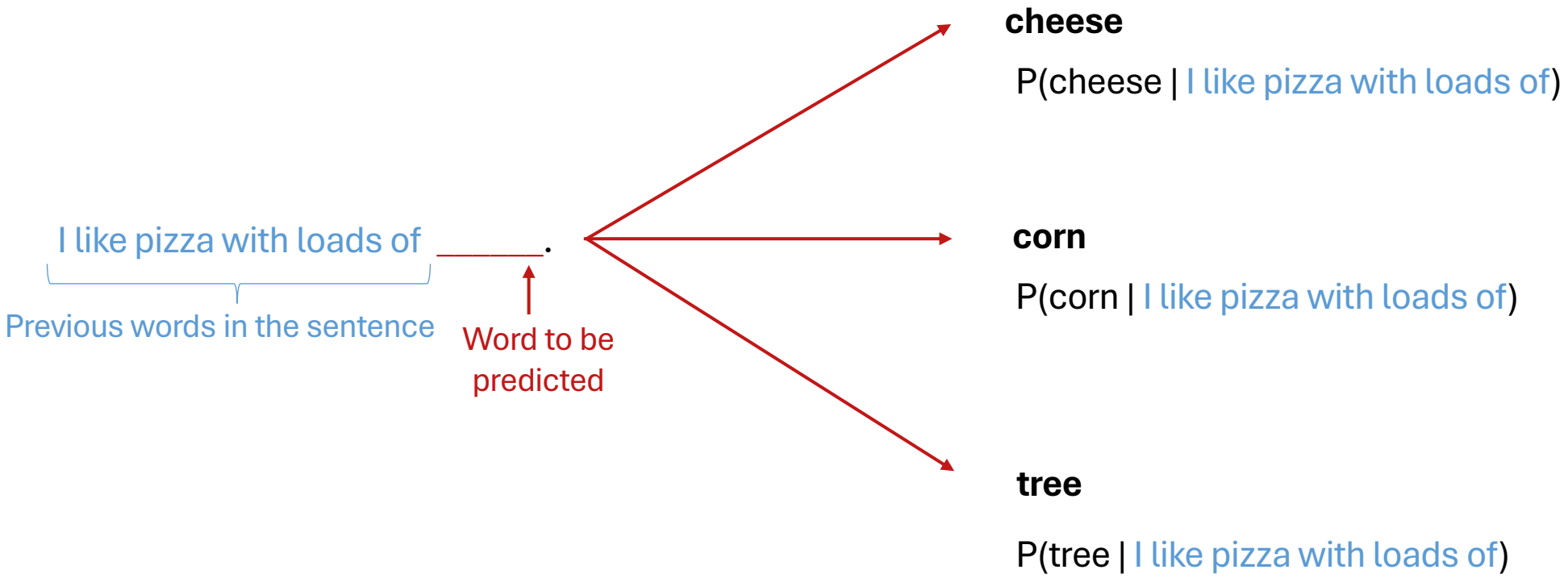


Pre-requisite for this chapter

- Loss function, backpropagation
- CNN
- RNN (LSTM/GRU)

Recall: Language Modeling

- **Language Modeling** is the task of predicting what word comes next



Recall: Language Modeling

- You can also think of a Language Model as a system that assigns a probability to a piece of text.

- For example, if we have some text $x^{(1)}, \ldots, x^{(T)}$, then the probability of this text (according to the Language Model) is:

$$P(\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(T)}) = P(\boldsymbol{x}^{(1)}) \times P(\boldsymbol{x}^{(2)} | \boldsymbol{x}^{(1)}) \times \cdots \times P(\boldsymbol{x}^{(T)} | \boldsymbol{x}^{(T-1)}, \ldots, \boldsymbol{x}^{(1)})$$

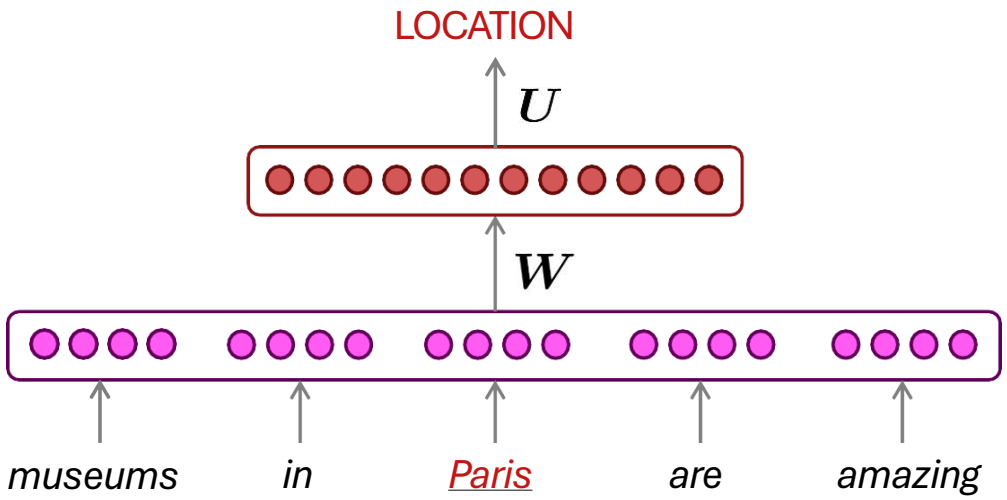
$$= \prod_{t=1}^{T} \underbrace{P(\boldsymbol{x}^{(t)} | \boldsymbol{x}^{(t-1)}, \ldots, \boldsymbol{x}^{(1)})}$$

This is what our LM provides

How to Build a *Neural* Language Model?

- Recall the Language Modeling task:
  - **Input:** sequence of words $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(t)}$
  - **Output:** probability distribution of the next word $\boldsymbol{P}\left(\boldsymbol{x}^{(t+1)} \middle| \boldsymbol{x}^{(t)}, \ldots, \boldsymbol{x}^{(1)}\right)$
- How about a window-based neural model?

Example: NER Task



A Fixed-window Neural Language Model

~~as the proctor started the clock~~ the students opened their _____

discard

fixed window

A Fixed-window Neural Language Model

output distribution

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

hidden layer

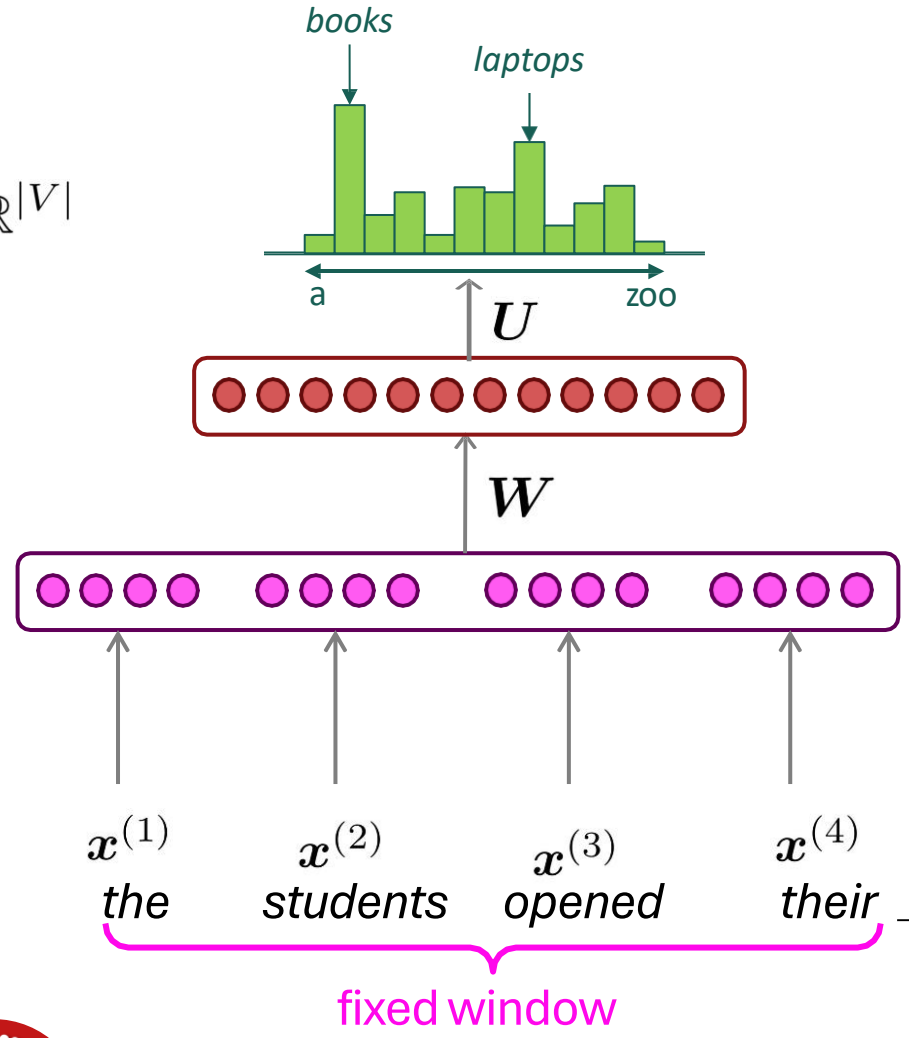
$$\boldsymbol{h} = f(\boldsymbol{W}\boldsymbol{e} + \boldsymbol{b}_1)$$

concatenated word embeddings

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

words / one-hot vectors

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



~~as the proctor started the clock~~

discard

A Fixed-window Neural Language Model

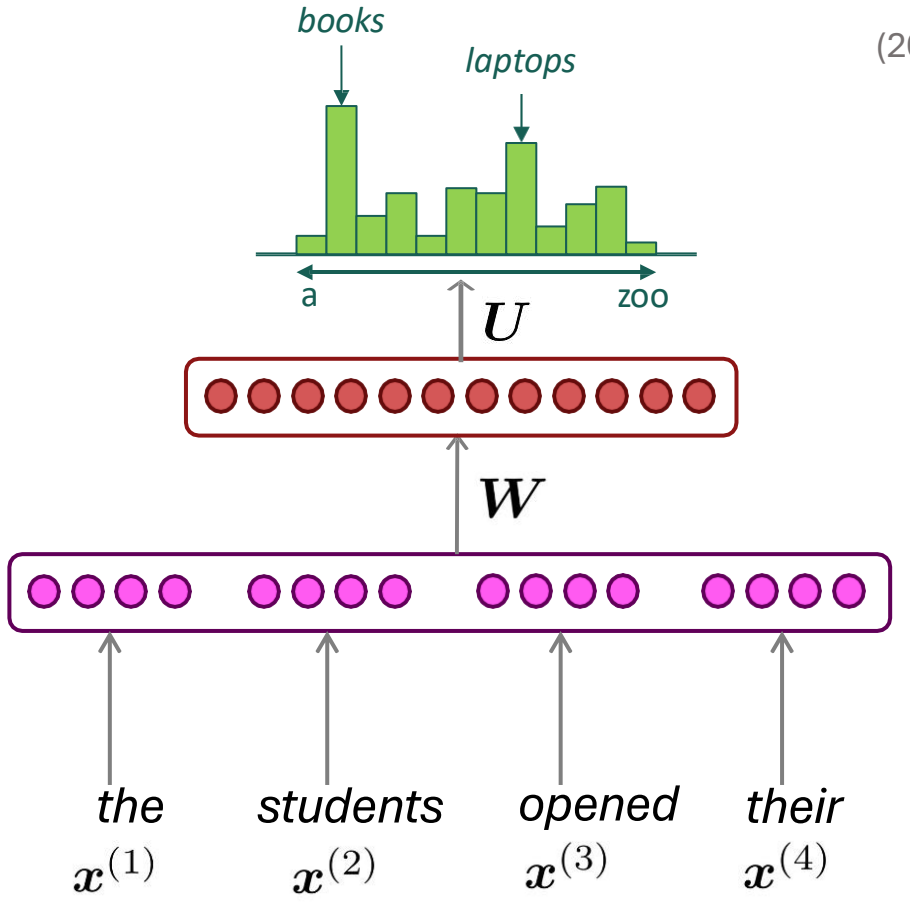
Approximately: Y. Bengio, et al. (2000/2003): A Neural Probabilistic Language Model

Improvements over *n*-gram LM:

- No sparsity problem
- Don't need to store all observed *n*-grams

Remaining problems:

- Fixed window is too small
- Enlarging window enlarges W
- $x^{(1)}$ and $x^{(2)}$ are multiplied by completely different weights in W. No symmetry in how the inputs are processed.



We need a neural architecture that can process ***any length input***