Week 2: Background on Language Models and Transformers

Generative AI
Saarland University – Winter Semester 2024/25

Adish Singla
genai-w24-tutors@mpi-sws.org

MAX PLANCK INSTITUTE
FOR SOFTWARE SYSTEMS

MAX-PLANCK-GESELLSCHAFT

Outline of the Lecture

- Week 1 recap and updates
- Language models (LMs)
- N-gram LMs
- Feedforward neural LMs
- Transformer-based LMs
- Week 2 assignment

Outline of the Lecture

- Week 1 recap and updates
- Language models (LMs)
- N-gram LMs
- Feedforward neural LMs
- Transformer-based LMs
- Week 2 assignment

Weekly Content: Tentative Plan

[15 Oct] Week 1: Introduction

[22 Oct] Week 2: Background on Language Models and Transformers

[29 Oct] Week 3: Large Language Models and In-context Learning

[05 Nov] Week 4: Pre-training and Supervised Fine-tuning

[12 Nov] Week 5: Preference-based Fine-tuning for Alignment

[26 Nov] Week 6: Multi-modal Foundation Models

[03 Dec] Week 7: Trustworthiness Aspects I

[10 Dec] ~~[17 Dec]~~ Week 8: Trustworthiness Aspects II

[07 Jan] Week 9: GenAI-powered Programming Education I

[14 Jan] Week 10: GenAI-powered Programming Education II

[28 Jan] Week 11: Project Discussion

[04 Feb] Week 12: Examination Preparation

Examination Dates and Time

Exam

- 20 Feb (Thursday) 2025
- 10am-1pm
- Written exam
- Room details will be provided later

Re-exam

- 19 Mar (Wednesday) 2025
- 10am-1pm
- Written exam
- Room details will be provided later

Outline of the Lecture

- Week 1 recap and updates
- Language models (LMs)
- N-gram LMs
- Feedforward neural LMs
- Transformer-based LMs
- Week 2 assignment

Predicting Next Words: An Example

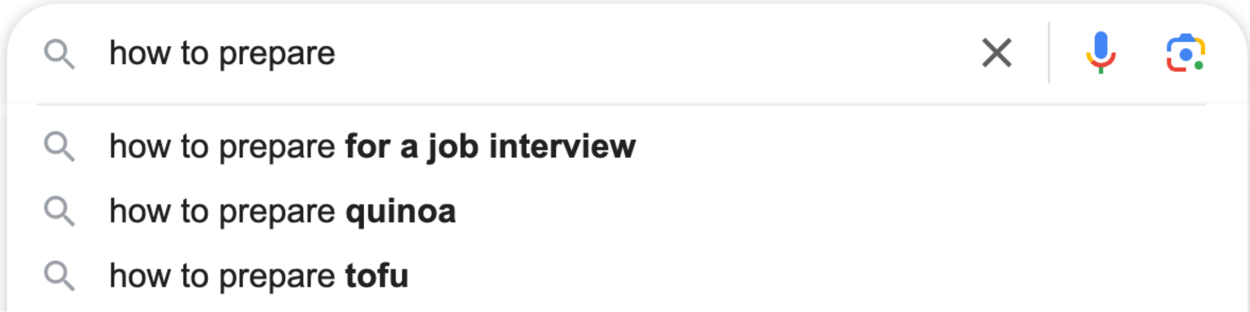
The water of Walden Pond is beautifully …

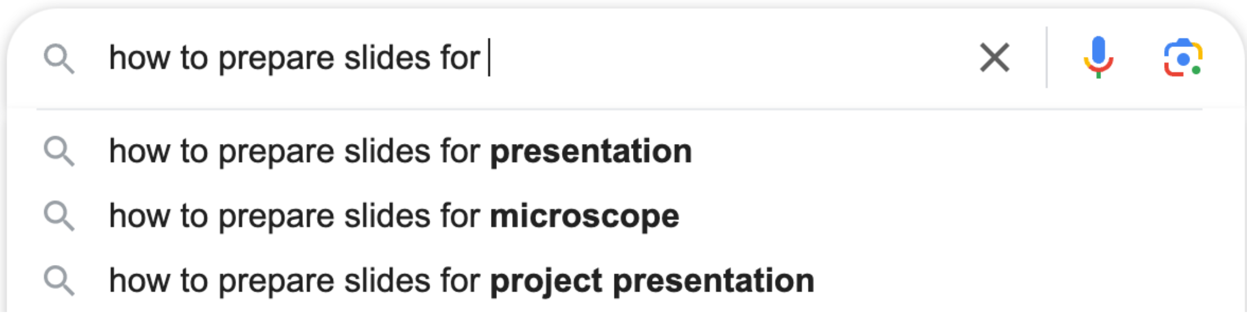
blue
green
clear

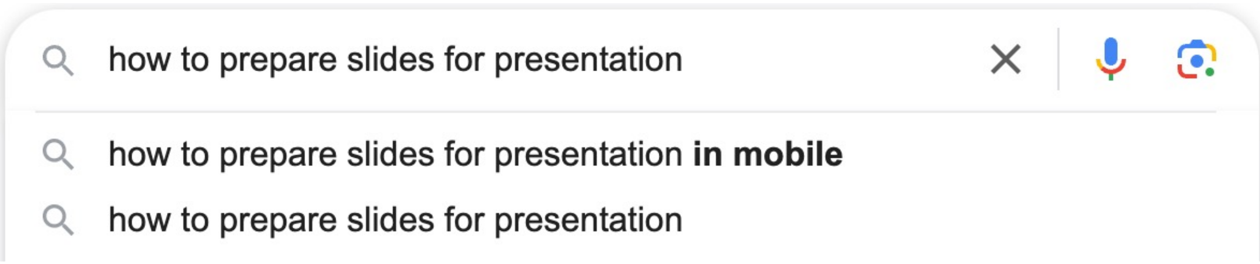
written

refrigerator
that

Predicting Next Words: Why it Matters?

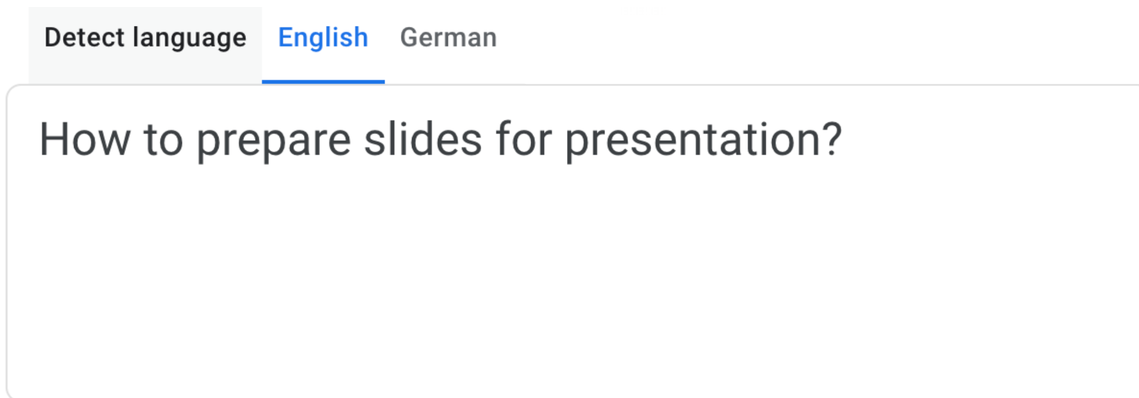


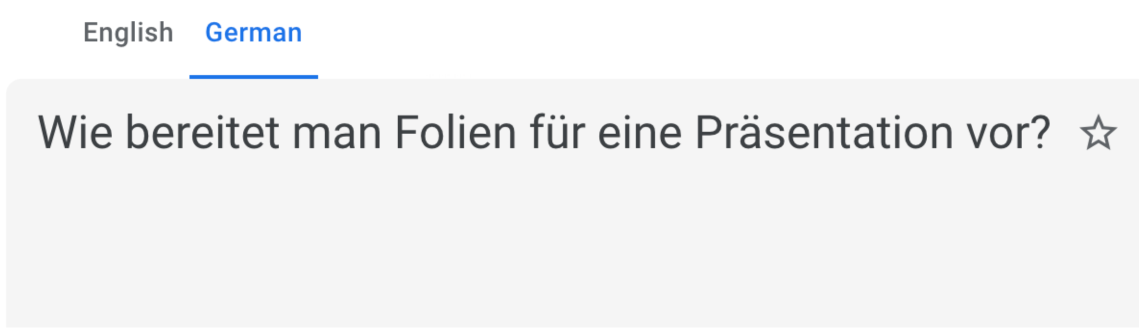




Source: https://www.google.com

Predicting Next Words: Why it Matters?





Predicting Next Words: Why it Matters?

ChatGPT 4o mini

how to prepare slides for presentation

Preparing slides for a presentation requires careful planning and design to effectively communicate your message. Here's a detailed guide to help you create impactful slides:

Step-by-Step Guide to Prepare Presentation Slides

1. Define Your Purpose and Audience

- **Understand the Purpose**: Are you informing, persuading, or training? Your goal will guide the content and design.
- **Know Your Audience**: Tailor your language, tone, and complexity to match the audience's background and interests.

Source: https://chatgpt.com/?model=gpt-4o-mini

Language Models (LMs)

- Language models (LMs) are systems that
  - can predict next words
  - can assign a probability to each potential next word
  - can assign a probability to a whole sentence
- LMs can generate text by predicting the next word over and over again
  - See Chapter 3.4 on “Sampling sentences from a LM”
- Large language models (LLMs) are trained to predict next words

Acknowledgements: Chapter 3 of SLP book at https://web.stanford.edu/~jurafsky/slp3/

LMs More Formally

Probability of a next word

- Probability of t-th word being x_t given history of $t-1$ words

$$P(x_t \mid x_1, x_2, x_3, \dots, x_{t-1})$$

or

$$P(x_t \mid x_{1:t-1})$$

Probability of a whole sentence

- Probability of a whole sentence or sequence comprising of t words

$$P(x_1, x_2, x_3, \dots, x_{t-1}, x_t)$$

or

$$P(x_{1:t})$$

Acknowledgements: Chapter 3 of SLP book at https://web.stanford.edu/~jurafsky/slp3/

LMs: How to Estimate These Probabilities?

Basic idea: Count and divide!

$$P(blue \mid \textit{The water of Walden Pond is beautifully}) = \frac{count(\textit{The water of Walden Pond is beautifully blue})}{count(\textit{The water of Walden Pond is beautifully})}$$

$$P(that \mid \textit{The water of Walden Pond is beautifully}) = \frac{count(\textit{The water of Walden Pond is beautifully that})}{count(\textit{The water of Walden Pond is beautifully})}$$

Not feasible practically

- Too many possible sentences ➔ Not feasible to get reliable estimates

Rest of this lecture

- Different families of models, including basic N-gram models and neural models

Acknowledgements: Chapter 3 of SLP book at https://web.stanford.edu/~jurafsky/slp3/

LMs: How to Measure a Model's Performance?

Space

- Number of parameters in the model to be stored

Time and compute

- Training time and compute required to learn parameters
- Inference time and compute required to sample text

Extrinsic quality evaluation

- Evaluate the quality of a model in a real task, e.g., speech recognition
- Running end-to-end systems is often expensive and time-consuming

Acknowledgements: Chapter 3 of SLP book at https://web.stanford.edu/~jurafsky/slp3/

LMs: How to Measure a Model's Performance?

Intrinsic quality evaluation

- Evaluate the quality of predicting next words on a test set with K points

$$D_{test} = \left\{\left(x_{1:t-1}^{i}, x_{t}^{i}\right)_{i=1:K}\right\}$$

- Geometric mean of probabilities of correctly predicting next words

$$\left(P\left(x_t^1 \mid x_{1:t-1}^1\right) \cdot P\left(x_t^2 \mid x_{1:t-1}^2\right) \cdot \ldots \cdot P\left(x_t^K \mid x_{1:t-1}^K\right)\right)^{1/K}$$

Equivalent to mean of log of the probabilities corresponding to correct words

$$\frac{1}{K}\sum_{i=1}^{K} \log\left(P\left(x_t^i \mid x_{1:t-1}^i\right)\right)$$

- Perplexity is the inverse of this geometric mean: lower perplexity ➔ better quality

Outline of the Lecture

- Week 1 recap and updates
- Language models (LMs)
- N-gram LMs
- Feedforward neural LMs
- Transformer-based LMs
- Week 2 assignment

N-gram LMs: The Markov Assumption

- Approximate the entire history of words with the last few words

- 3-gram (trigram) model: Approximate the history by just the last two words

$$P(x_t \mid x_{1:t-1}) \approx P(x_t \mid x_{t-2:t-1})$$

$$P(\mathit{blue} \mid \textit{The water of Walden Pond is beautifully}) \approx P(\mathit{blue} \mid \textit{is beautifully})$$

- N-gram model: Approximate the history by the last $N-1$ words

$$P(x_t \mid x_{1:t-1}) \approx P(x_t \mid x_{t-N+1:t-1})$$

Acknowledgements: Chapter 3 of SLP book at https://web.stanford.edu/~jurafsky/slp3/

N-gram LMs: Training the Model Parameters

Same basic idea: Count and divide!

- Consider 3-gram model

$$P(blue \mid \textit{The water of Walden Pond is beautifully}) \approx \frac{count(\textit{is beautifully blue})}{count(\textit{is beautifully})}$$

$$P(that \mid \textit{The water of Walden Pond is beautifully}) \approx \frac{count(\textit{is beautifully that})}{count(\textit{is beautifully})}$$

Training from a large corpus

- Compute counts for all possible sequences of N words and $N-1$ words
- $count(.)$ values correspond to model parameters

Acknowledgements: Chapter 3 of SLP book at https://web.stanford.edu/~jurafsky/slp3/

N-gram LMs: Issues with Sparsity and Backoff Algorithm

Sparsity issues

- Any finite training corpus will be missing many word sequences
- Counts do not capture the similarity of words or sequences

Counts in numerator being zero

- 3-gram model: $count(\textit{is beautifully}) > 0$; $count(\textit{is beautifully blue}) = 0$
- Smoothing algorithm: Add some small value to every count

Counts in denominator being zero

- 3-gram model: $count(\textit{is beautifully}) = 0$; $count(\textit{is beautifully blue}) = 0$
- Backoff algorithm: Reduce the history until denominator is non-zero

$$\begin{aligned} P(blue \mid \textit{The water of Walden Pond is beautifully}) &\approx P(blue \mid \textit{is beautifully}) \\ &\approx P(blue \mid \textit{beautifully}) \\ &\approx P(blue) \end{aligned}$$

- Week 1 assignment implementation has Backoff algorithm

N-gram LMs: Issues with Space

Week 1 (E.1) Number of parameters

- N-gram model and vocabular size V
- A count (parameter) is stored for each possible sequence of N words
- An upper bound on parameters is V^N

Week 1 (E.2) Number of parameters

- For $V = 50{,}000$ and $N = 1$, an upper bound on parameters is 5×10^4
- For $V = 50{,}000$ and $N = 3$, an upper bound on parameters is 125×10^{12}

Value of N

- Ideally, we want large N to capture longer context, but higher value
  - increases the sparsity problem
  - increases the number of parameters
- $N > 5$ is typically not considered with large vocabulary

Outline of the Lecture

- Week 1 recap and updates
- Language models (LMs)
- N-gram LMs
- Feedforward neural LMs
- Transformer-based LMs
- Week 2 assignment

Neural LMs Background: 1-hot Encoding of a Word

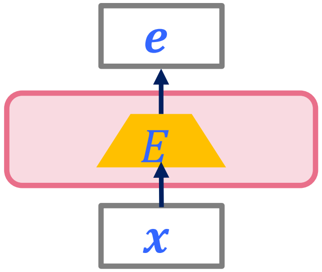
- Consider the index of a word in the vocabulary V
  - Let's say *Pond* has index 5
- 1-hot encoding for *Pond* is a vector $\boldsymbol{x}$ of size $|V|$
- Represent vectors as column vectors, i.e., $\boldsymbol{x}$ has $|V|$ rows and 1 column

$$\boldsymbol{x} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ \dots \\ 0 \\ 0 \end{bmatrix}$$

Acknowledgements: Chapter 7 of SLP book at https://web.stanford.edu/~jurafsky/slp3/

Neural LMs Background: Embedding of a Word

- We consider a d-dimensional vector representation of a word where $d \ll |V|$
- Capture this through an embedding matrix E of size $d \times |V|$
- For a given word with 1-hot encoding vector $\boldsymbol{x}$, we obtain its embedding $\boldsymbol{e} = E\boldsymbol{x}$
- E has $d \times |V|$ parameters that can be learnt during training



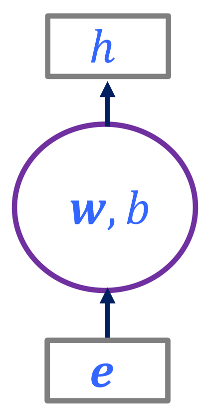
Acknowledgements: Chapter 7 of SLP book at https://web.stanford.edu/~jurafsky/slp3/

Neural LMs Background: Neural Unit

- A neural unit applies a transformation on an input vector
- Consider an input vector $\boldsymbol{e}$ of size d
- A neural unit has
  - a weight vector $\boldsymbol{w}$ of size d (same as input) and a scalar bias term b of size 1
  - an activation function, e.g., $\mathrm{sigmoid}$
- The output h from this neural unit is a scalar value

$$h = \mathrm{sigmoid}(\boldsymbol{w} \cdot \boldsymbol{e} + \ b) = \frac{1}{1 + \exp(-(\boldsymbol{w} \cdot \boldsymbol{e} + \ b))}$$

- $\boldsymbol{w}$ and b are parameters of the unit that are learnt during training

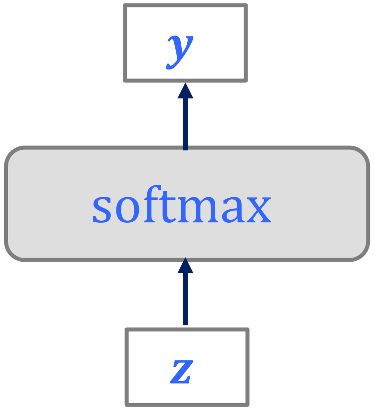


Acknowledgements: Chapter 7 of SLP book at https://web.stanford.edu/~jurafsky/slp3/

Neural LMs Background: Softmax to Obtain Probabilities

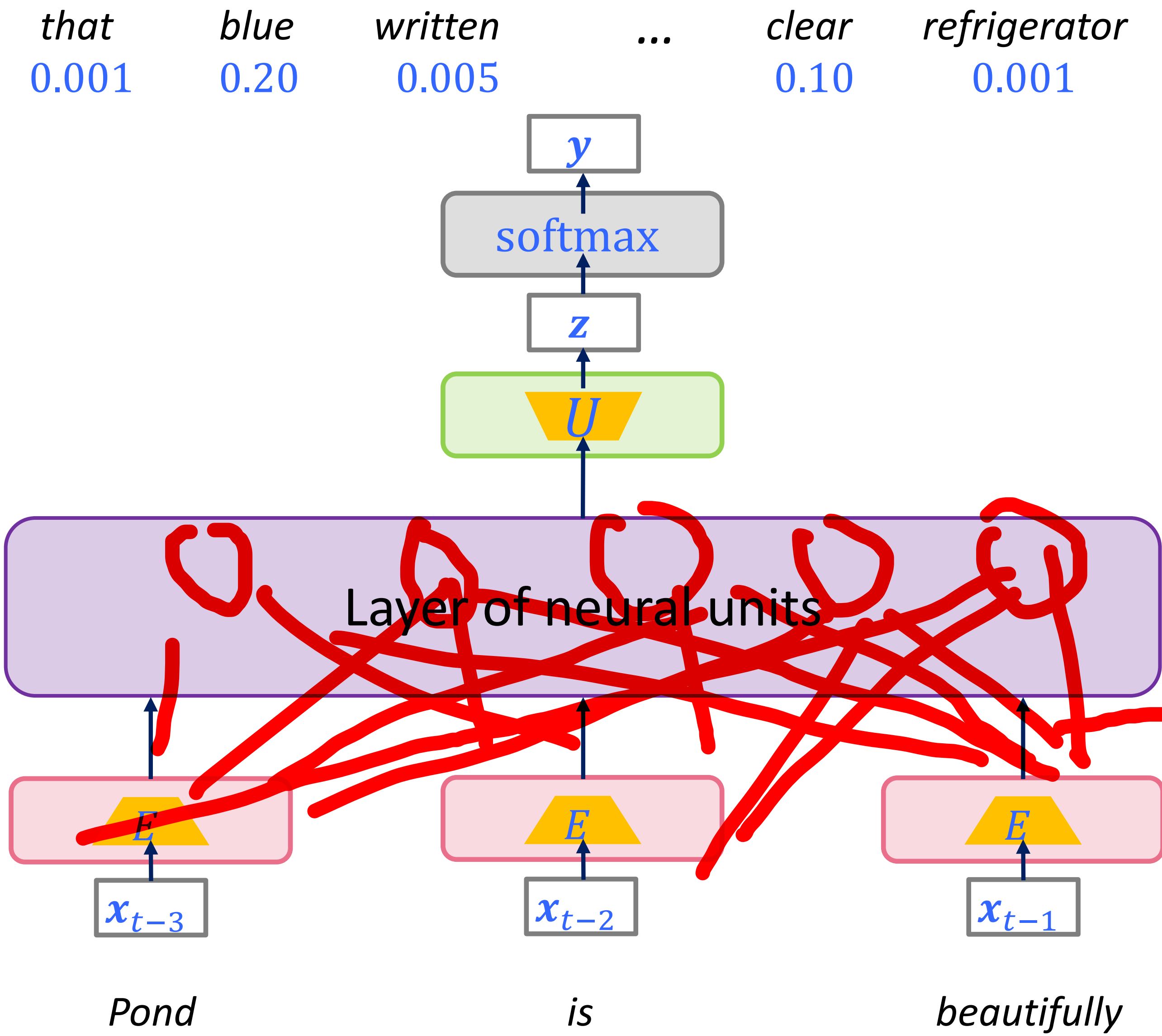
- softmax operation is used to convert a vector of values to probabilities
- Consider an input vector $\boldsymbol{z}$ of size $|V|$ with values $z_1, z_2, \dots, z_i, \dots, z_{|V|}$
- $\boldsymbol{y} = \text{softmax}(\boldsymbol{z})$ is a vector of probabilities with values $y_1, y_2, \dots, y_i, \dots, y_{|V|}$

$$y_i = \frac{\exp(z_i)}{\sum_{j=1}^{|V|} \exp(z_j)}$$



Acknowledgements: Chapter 7 of SLP book at https://web.stanford.edu/~jurafsky/slp3/

Simple Feedforward Neural LM: Overview of Architecture



probability vector, $|V|$

softmax operation

score vector (logits), $|V|$

unembedding matrix, $|V| \times d_h$

d_h neural units

d

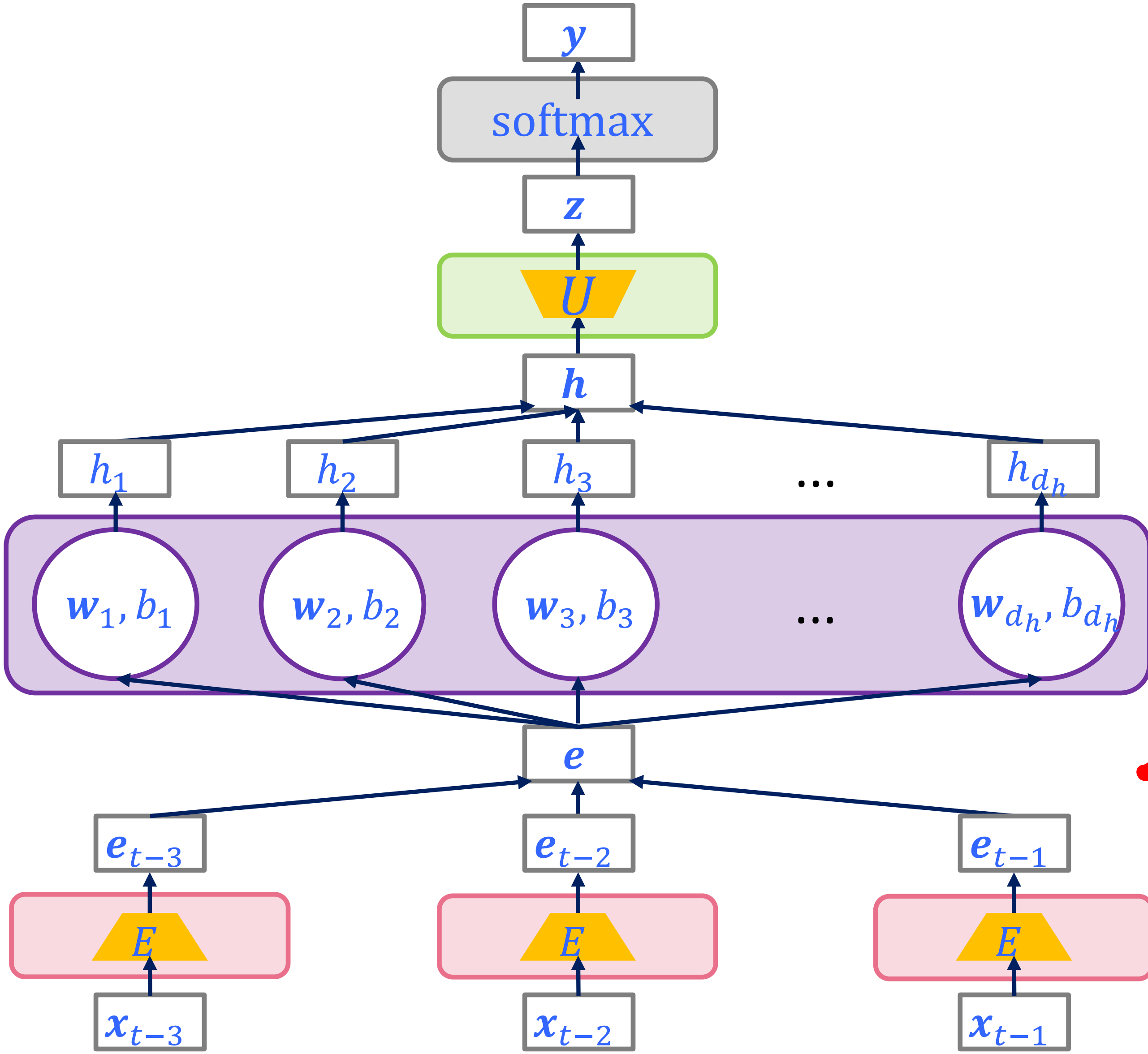
embedding matrix, $d \times |V|$

3 one-hot vectors, $|V|$

3 words

Reference: [Bengio et al., NeurIPS'00; JMLR'03]

Simple Feedforward Neural LM: Detailed Architecture



probability vector, $|V|$

softmax operation

score vector (logits), $|V|$

unembedding matrix, $|V| \times d_h$

joint hidden layer outputs, d_h

d_h hidden layer outputs, 1

d_h neural units:

- each weight vector, $3d$
- each bias term, 1

joint embedding vector, $3d$

3 embedding vectors, d

embedding matrix, $d \times |V|$

3 one-hot vectors, $|V|$

Simple Feedforward Neural LM: Number of Parameters

Parameters for general N

- Consider the architecture with $(N-1)$ context words
  - The previous slide has $N=4$, i.e., 3 context words used to predict the next word
- The number of parameters for this architecture is

$$|V|(d+d_h)+\big((N-1)d+1\big)d_h$$

- To get an idea of scale, consider the following values
  - $N=6$
  - $|V|\approx 18000$
  - $d=100$
  - $d_h=60$

Part of Week 2 assignment

- In the assignment, you will compute the number of parameters step by step

Feedforward Neural LMs: Training the Model Parameters

Basic idea: Training via self-supervision!

- Consider a large training corpus of length K: $x_1, x_2, x_3, \dots, x_{K-1,} x_K$
- At time t, predict the next word x_t from history $x_1, x_2, x_3, \dots, x_{t-1}$
- Optimize a loss function based on negative log likelihood, i.e., log of probabilities corresponding to correct words:

$$-\frac{1}{K}\sum_{t=1}^{K} \log\left(P(x_t \mid x_{1:t-1})\right)$$

- Learn parameters via stochastic gradient decent

Optional reading material

- This course doesn't cover details of neural network training
- Optional reading for details: Chapter 7.5 of SLP book

Feedforward (Ff) Neural LMs: Summary

Improvements over N-gram LMs

- Parameters grow only linearly in N
- Handles the issues with sparsity and generalization

Key issues in simple Ff neural LM

- Fixed window network with small N has several limitations:
  - provides very limited history or context
  - requires sliding window for predictions which limits parallelization
- Ideally we want unlimited N but
  - number of parameters grows as Ndd_h in terms of dependency on N
  - size of vector $\boldsymbol{e}$ grows as $(N-1)d$ ➔ difficult to integrate information

Outline of the Lecture

- Week 1 recap and updates
- Language models (LMs)
- N-gram LMs
- Feedforward neural LMs
- Transformer-based LMs
- Week 2 assignment

From Simple Ff Neural LM to Transformer-based LM

An approximate timeline of LMs' evolution

- 1970s: Resurgence of N-gram LMs in speech recognition, e.g., [Jelinek et al. '75]

  …

- 2000: Simple Ff Neural LM [Bengio et al., NeurIPS'00]

  …

- 2014: Recurrent networks (RNN) for translation, e.g., [Sutskever et al., NeurIPS'14]
  - Encoder-decoder architectures with Long Short-Term Memory (LSTM) hidden units
- 2015: Attention mechanisms, e.g., [Bahdanau et al., ICLR'15]

  …

- 2017: Transformer [Vaswani et al., NeurIPS'17]
- 2018: OpenAI's GPT-1 (~100 million parameters) [Radford et al., '18]
- 2020: OpenAI's GPT-3 (175 billion parameters) [Brown et al., NeurIPS'20]

  …

Acknowledgements: Chapter 3/9 of SLP book at https://web.stanford.edu/~jurafsky/slp3/

How to Deal with Large Contexts?

The water of Walden Pond is beautifully …

The chicken did not cross the road because it …

A woman is throwing a …

Acknowledgements: Chapter 9 of SLP book; [Xu et al., ICML'15]

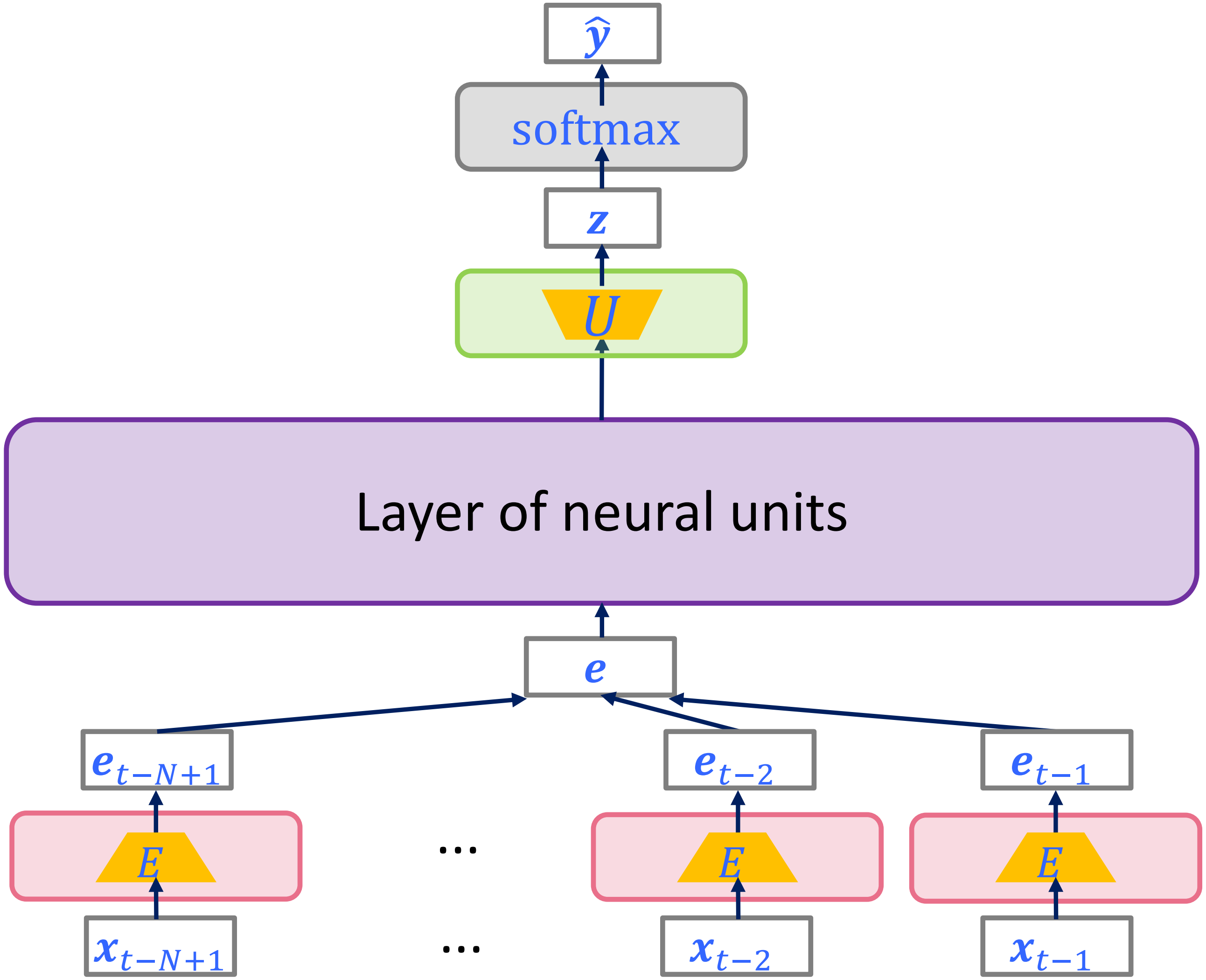
By Attending to Relevant Parts and Integrating Information

The water of Walden Pond is beautifully **blue**

The chicken did not cross the road because it **was**

A woman is throwing a **frisbee**

Incorporating Attention Mechanism in Simple Ff Neural LM



probability vector, $|V|$

softmax operation

score vector (logits), $|V|$

unembedding matrix, $|V| \times d_h$

d_h neural units:
- each weight vec, $(N-1)d$
- each bias term, 1

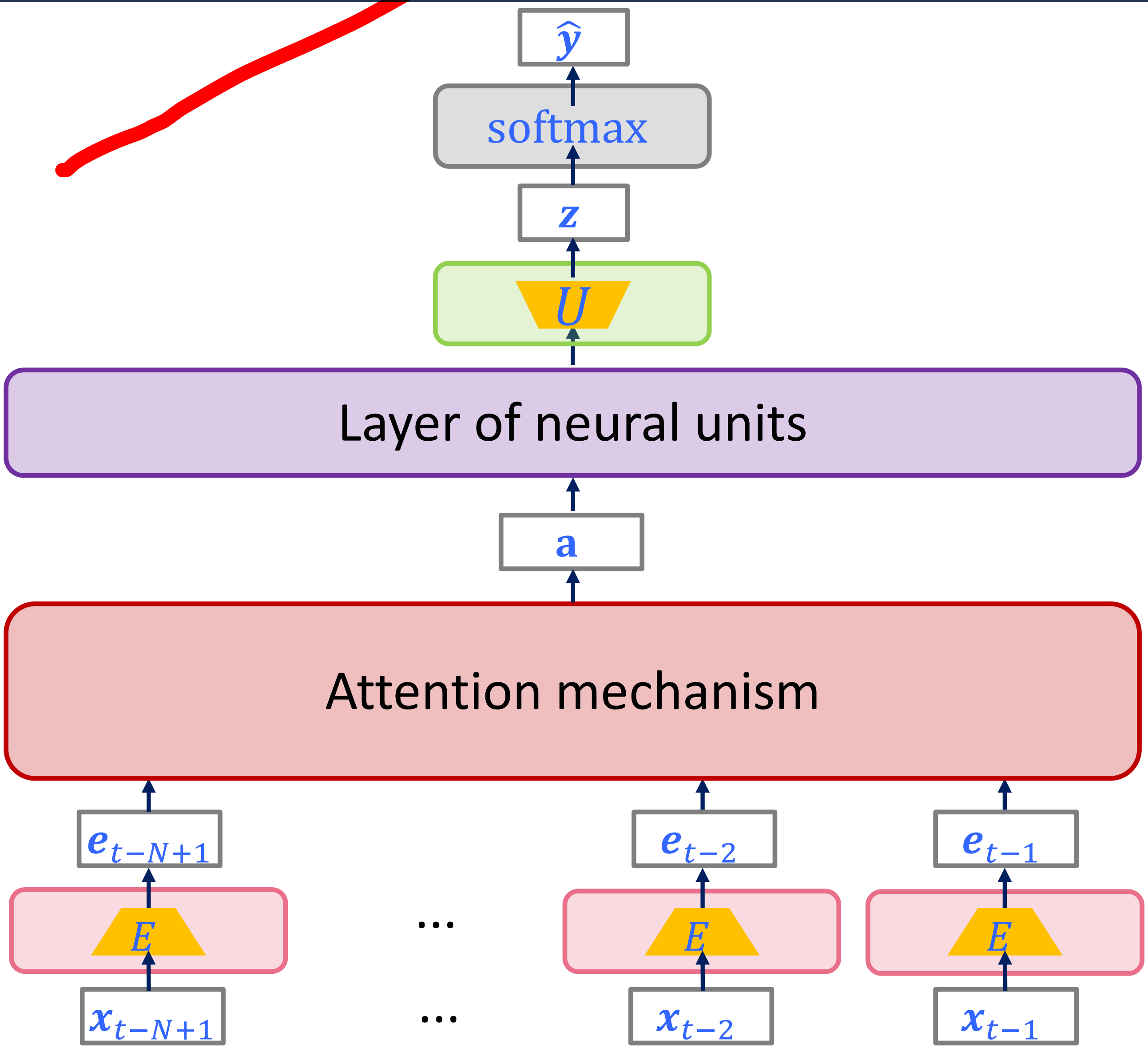
joint emb. vector, $(N-1)d$

$N-1$ embedding vectors, d

embedding matrix, $d \times |V|$

$N-1$ one-hot vectors, $|V|$

Incorporating Attention Mechanism in Simple Ff Neural LM



probability vector, $|V|$

softmax operation

score vector (logits), $|V|$

unembedding matrix, $|V| \times d_h$

d_h neural units:
- each weight vector, d_v
- each bias term, 1

attention vector, d_v

$N-1$ embedding vectors, d

embedding matrix, $d \times |V|$

$N-1$ one-hot vectors, $|V|$

Attention Mechanism with Single Head: Simplified Version

Input vectors

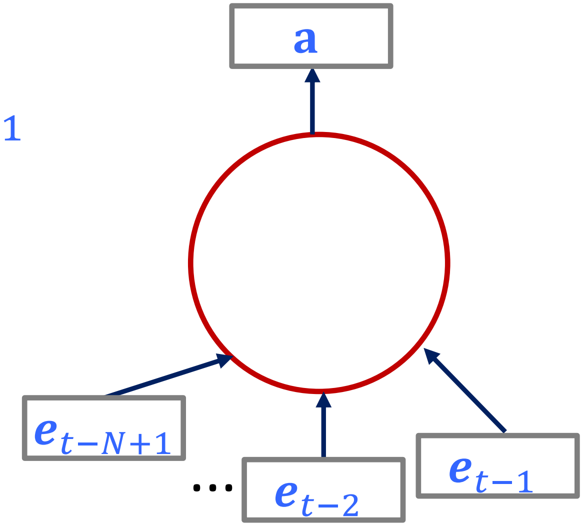
- $N-1$ vectors of size d denoted by $\boldsymbol{e}_{t-N+1}, \ldots, \boldsymbol{e}_{t-2}, \boldsymbol{e}_{t-1}$

Output vector

- 1 vector of size d denoted by $\mathbf{a}$

Parameters

- None



Computation of vector $\mathbf{a}$

- Given scalar values α_i, attention vector is $\mathbf{a} = \sum_{i=t-N+1}^{t-1} \alpha_i \, \boldsymbol{e}_i$
  - Compute similarity scores for $\boldsymbol{e}_{t-1}$ with $\boldsymbol{e}_i$ for $i = t-N+1, \ldots, t-2, t-1$

    using vector dot prodct similarity $\text{score}(\boldsymbol{e}_{t-1}, \boldsymbol{e}_i) = \frac{\boldsymbol{e}_{t-1} \cdot \boldsymbol{e}_i}{\text{sqrt}(d)}$
  - Compute α_i using softmax over scores: $\alpha_i = \text{softmax}(\text{score}(\boldsymbol{e}_{t-1}, \boldsymbol{e}_i))$

Attention Mechanism with Single Head: Actual Version

Input vectors

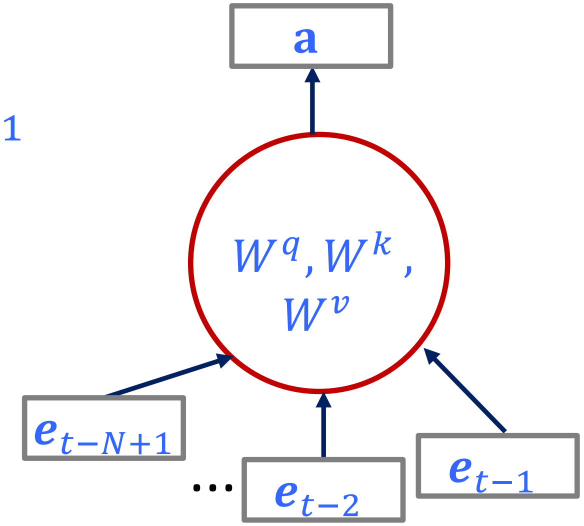
- $N-1$ vectors of size d denoted by $\boldsymbol{e}_{t-N+1}, \dots, \boldsymbol{e}_{t-2}, \boldsymbol{e}_{t-1}$

Output vector

- 1 vector of size d_v denoted by $\mathbf{a}$

Parameters

- query matrix W^q of size $d_k \times d$
- key matrix W^k of size $d_k \times d$
- value matrix W^v of size $d_v \times d$

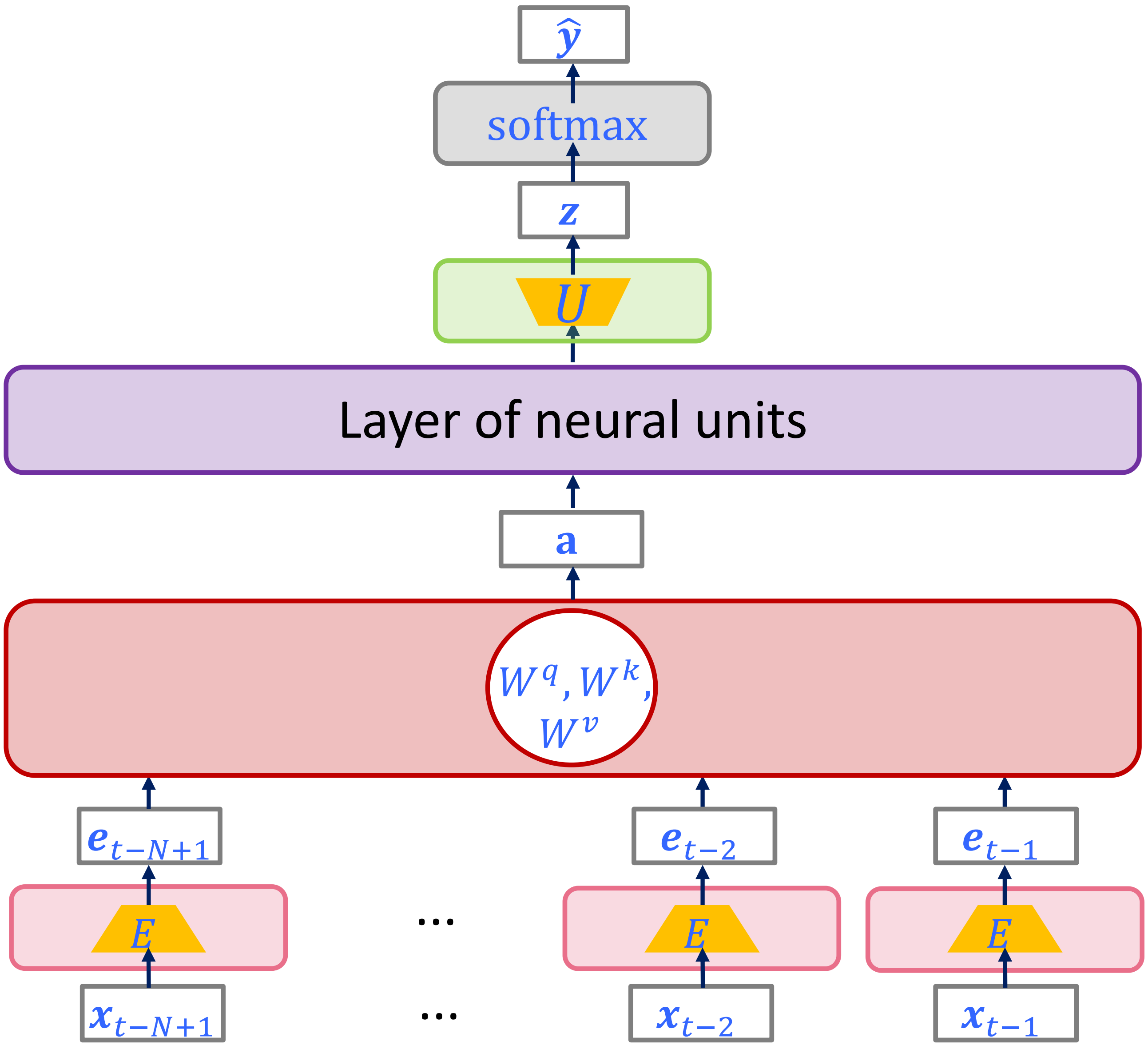


Computation of vector a

- Given scalar values α_i, attention vector is $\mathbf{a} = \sum_{i=t-N+1}^{t-1} \alpha_i W^v \boldsymbol{e}_i$
  - Compute similarity scores for $\boldsymbol{e}_{t-1}$ with $\boldsymbol{e}_i$ for $i = t-N+1, \dots, t-2, t-1$

    using vector dot prodct similarity $\text{score}(\boldsymbol{e}_{t-1}, \boldsymbol{e}_i) = \dfrac{W^q \boldsymbol{e}_{t-1} \cdot W^k \boldsymbol{e}_i}{\text{sqrt}(d_k)}$
  - Compute α_i using softmax over scores: $\alpha_i = \text{softmax}(\text{score}(\boldsymbol{e}_{t-1}, \boldsymbol{e}_i))$

Incorporating Attention Mechanism in Simple Ff Neural LM



probability vector, $|V|$

softmax operation

score vector (logits), $|V|$

unembedding matrix, $|V| \times d_h$

d_h neural units:
- each weight vector, d_v
- each bias value, 1

attention vector, d_v

1 attention head:
- query matrix, $d_k \times d$
- key matrix, $d_k \times d$
- value matrix, $d_v \times d$

$N-1$ embedding vectors, d

embedding matrix, $d \times |V|$

$N-1$ one-hot vectors, $|V|$

Simple Ff Neural LM with Attention: Number of Params

Parameters for general N when $d_k = d_v = d$

- Consider simple Ff neural LM <u>with</u> attention when $d_v = d$ and $d_k = d$
- The number of parameters in this architecture is

$$|V|(d + d_h) + 3d^2 + (d + 1)d_h$$

- Recall the number of parameters in simple Ff neural LM <u>without</u> attention is

$$|V|(d + d_h) + \big((N - 1)d + 1\big)d_h$$

- After adding attention mechanism, there is no dependency on N

Part of Week 2 assignment

- In the assignment, you will compute the number of parameters step by step

From Simple Ff Neural LM to Transformer-based LM

- [This week] N-gram LMs
- [This week] Simple Ff Neural LM [Bengio et al., NeurIPS'00]
- From Simple Ff Neural LM to Transformer-based LM
  - [This week] Single attention head
  - [Next week] Multiple attention heads
  - [Next week] Transformer block
  - [Next week] Stacking transformer blocks and parallelization
- Transformer [Vaswani et al., NeurIPS'17]

- [Next week] LLMs and in-context learning capabilities
  - OpenAI's GPT-1 (~100 million parameters) [Radford et al., '18]
  - OpenAI's GPT-3 (175 billion parameters) [Brown et al., NeurIPS'20]

Outline of the Lecture

- Week 1 recap and updates
- Language models (LMs)
- N-gram LMs
- Feedforward neural LMs
- Transformer-based LMs
- Week 2 assignment

Week 2 Assignment

https://owncloud.mpi-sws.org/index.php/s/9YYZkDAeb58qiT2