Natural Language Processing (CS-472) Spring-2023

Muhammad Naseer Bajwa

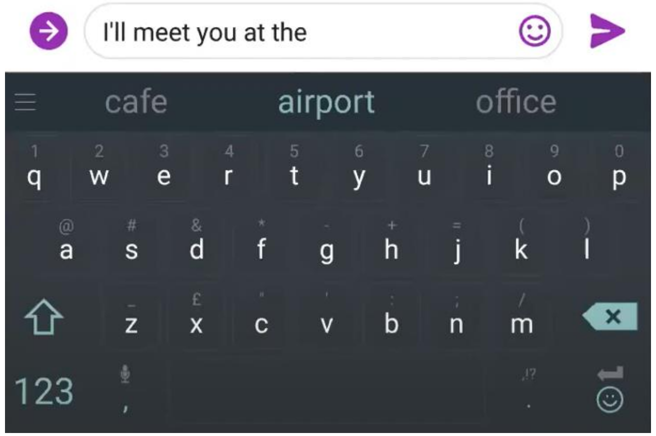
Assistant Professor,
Department of Computing, SEECS
Co-Principal Investigator,
Deep Learning Lab, NCAI
NUST, Islamabad
naseer.bajwa@seecs.edu.pk

Overview of this week's lecture

Language Models

- n-gram Language Models
- Neural Language Models
- Recurrent Neural Networks
- GRUs and LSTMs

Language Modelling is the task of predicting next word in a sequence

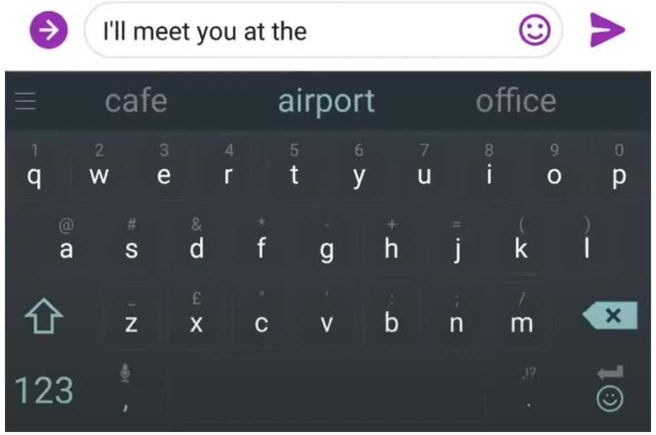


Language Modelling is the task of predicting next word in a sequence

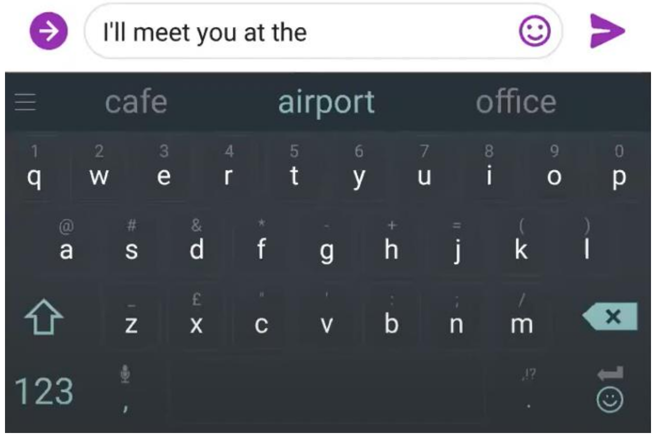
- A language model predicts the probability distribution of the word $x^{<t+1>}$, given a sequence of words $x^{<1>}, x^{<2>}, \ldots, x^{<t>}$.

$$P(x^{<t+1>} | x^{<1>}, x^{<2>}, \ldots, x^{<t>})$$

where $x^{<t+1>}$ can be any word in the vocabulary $V = \{w_1, w_2, \ldots, w_V\}$



Language Modelling is the task of predicting next word in a sequence



- A language model predicts the probability distribution of the word $x^{<t+1>}$, given a sequence of words $x^{<1>}, x^{<2>}, \ldots, x^{<t>}$.

$$P(x^{<t+1>} | x^{<1>}, x^{<2>}, \ldots, x^{<t>})$$

where $x^{<t+1>}$ can be any word in the vocabulary $V = \{w_1, w_2, \ldots, w_V\}$

- In other words, a language model is a system that assigns probability to a piece of text.

$$P(x^{<1>}, x^{<2>}, \ldots, x^{<T>}) = P(x^{<1>}) \times P(x^{<2>} | x^{<1>}) \times \cdots \times P(x^{<T>} | x^{<T>}, \ldots, x^{<1>})$$
$$= \prod_{t=1}^{T} P(x^{<t+1>} | x^{<t>}, \ldots, x^{<1>})$$

How to learn a language model?

- n-gram language models were used before the advent of deep learning.
  - n-gram is a sequence of n consecutive words.
    - unigrams: the, students, opened, their
    - bigrams: the students, students opened, opened their
    - trigrams: the students opened, students opened their
    - 4-grams: the students opened their

How to learn a language model?

- n-gram language models were used before the advent of deep learning.
  - n-gram is a sequence of n consecutive words.
    - unigrams: the, students, opened, their
    - bigrams: the students, students opened, opened their
    - trigrams: the students opened, students opened their
    - 4-grams: the students opened their
- They use statistical data to calculate how frequent different n-grams are in a given corpus and use those statistics to predict the next word.
  - Make a **Markov assumption**: The next word depends only on $n-1$ preceding words.

$$P(x^{<t+1>}|x^{<t>}, \dots, x^{<1>}) = P(x^{<t+1>}|x^{<t>}, \dots, x^{<t-n+2>})$$

$$= \frac{P(x^{<t+1>}, x^{<t>}, \dots, x^{<t-n+2>})}{P(x^{<t>}, \dots, x^{<t-n+2>})}$$

How to learn a language model?

- n-gram language models were used before the advent of deep learning.
  - n-gram is a sequence of n consecutive words.
    - unigrams: the, students, opened, their
    - bigrams: the students, students opened, opened their
    - trigrams: the students opened, students opened their
    - 4-grams: the students opened their
- They use statistical data to calculate how frequent different n-grams are in a given corpus and use those statistics to predict the next word.
  - Make a **Markov assumption**: The next word depends only on $n - 1$ preceding words.

$$P(x^{<t+1>} | x^{<t>}, \ldots, x^{<1>}) = P(x^{<t+1>} | x^{<t>}, \ldots, x^{<t-n+2>})$$

$$= \frac{P(x^{<t+1>}, x^{<t>}, \ldots, x^{<t-n+2>})}{P(x^{<t>}, \ldots, x^{<t-n+2>})}$$

Probability of n-gram

Probability of $n - 1$-gram

Let's take an example of n-gram language model

- Suppose we are learning a 4-gram Language Model.

~~as the proctor started the clock, the~~ students opened their ________

$$P(w|students\ opened\ their) = \frac{count(students\ opened\ their\ w)}{count(students\ opened\ their)}$$

Let's take an example of n-gram language model

- Suppose we are learning a 4-gram Language Model.

~~as the proctor started the clock, the~~ students opened their ________

$$P(w|students\ opened\ their) = \frac{count(students\ opened\ their\ w)}{count(students\ opened\ their)}$$

- If in the corpus we find,
    - students opened their ($\times$ 1000).
    - students opened their books ($\times$ 400). $\rightarrow$ $P(books|students\ open\ their) = 0.4$
    - students opened their exams ($\times$ 100). $\rightarrow$ $P(exams|students\ open\ their) = 0.1$

Let's take an example of n-gram language model

- Suppose we are learning a 4-gram Language Model.

~~as the proctor started the clock, the~~ students opened their ________

$$P(w|students\ opened\ their) = \frac{count(students\ opened\ their\ w)}{count(students\ opened\ their)}$$

- If in the corpus we find,
  - students opened their ($\times$ 1000).
  - students opened their books ($\times$ 400). $\rightarrow$ $P(books|students\ open\ their) = 0.4$
  - students opened their exams ($\times$ 100). $\rightarrow$ $P(exams|students\ open\ their) = 0.1$
- Should we have discarded the 'proctor' context?

n-gram language models have sparsity problem

$$P(w|students\ opened\ their) = \frac{count(students\ opened\ their\ w)}{count(students\ opened\ their)}$$

n-gram language models have sparsity problem

$$P(w|students\ opened\ their) = \frac{count(students\ opened\ their\ w)}{count(students\ opened\ their)}$$

- What if $students\ opened\ their\ w$ never appeared in the training data?

n-gram language models have sparsity problem

$$P(w|students\ opened\ their) = \frac{count(students\ opened\ their\ w)}{count(students\ opened\ their)}$$

- What if $students\ opened\ their\ w$ never appeared in the training data?
  - Apply **Smoothing**: Add a small δ to the count for every $w \in V$.

n-gram language models have sparsity problem

$$P(w|students\ opened\ their) = \frac{count(students\ opened\ their\ w)}{count(students\ opened\ their)}$$

- What if $students\ opened\ their\ w$ never appeared in the training data?
  - Apply **Smoothing**: Add a small δ to the count for every $w \in V$.
- What if $students\ opened\ their$ never appeared in the training data?

n-gram language models have sparsity problem

$$P(w|students\ opened\ their) = \frac{count(students\ opened\ their\ w)}{count(students\ opened\ their)}$$

- What if $students\ opened\ their\ w$ never appeared in the training data?
    - Apply **Smoothing**: Add a small δ to the count for every $w \in V$.
- What if $students\ opened\ their$ never appeared in the training data?
    - Use **Backoff**: Condition on smaller context. In the case above, use only $opened\ their$ as the conditioning context.

n-gram language models have sparsity problem

$$P(w|students\ opened\ their) = \frac{count(students\ opened\ their\ w)}{count(students\ opened\ their)}$$

- What if $students\ opened\ their\ w$ never appeared in the training data?
  - Apply **Smoothing**: Add a small δ to the count for every $w \in V$.
- What if $students\ opened\ their$ never appeared in the training data?
  - Use **Backoff**: Condition on smaller context. In the case above, use only $opened\ their$ as the conditioning context.
- Increasing n makes sparsity problems worse. Typically $n \leq 5$ is used.

n-gram language models have storage problem also

$$P(w|students\ opened\ their) = \frac{count(students\ opened\ their\ w)}{count(students\ opened\ their)}$$

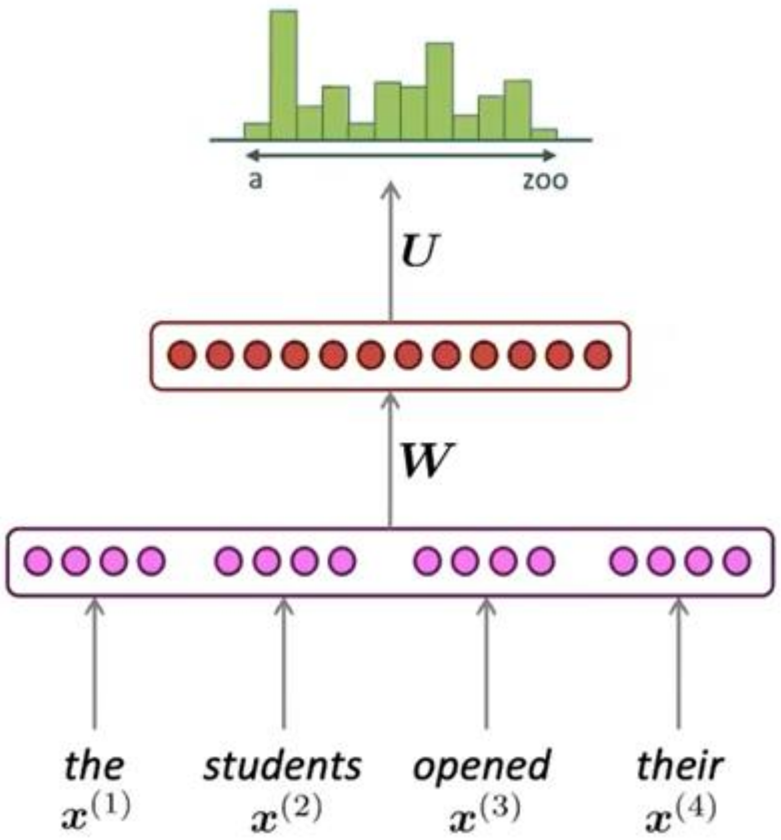
- We need to store count for **all** n-grams in the corpus.

- Increasing n or corpus size increases model size exponentially.

How to build a neural language model?

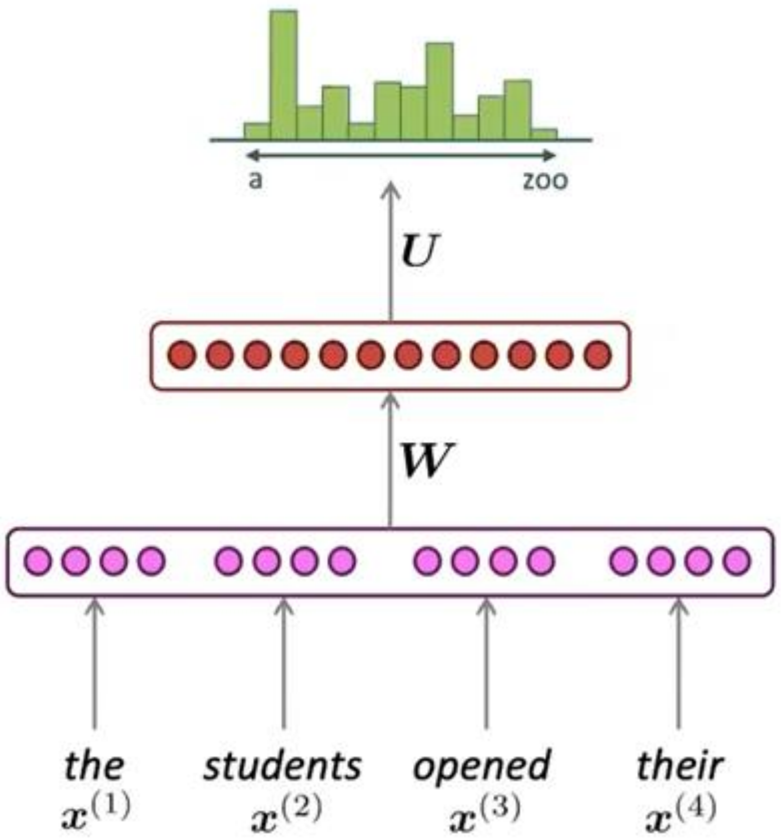
Words

$x^{<1>}, x^{<2>}, x^{<3>}, x^{<4>}$



Bengio, Yoshua, Réjean Ducharme, and Pascal Vincent. "A neural probabilistic language model." Advances in neural information processing systems 13 (2000).

How to build a neural language model?



Concatenated word embeddings

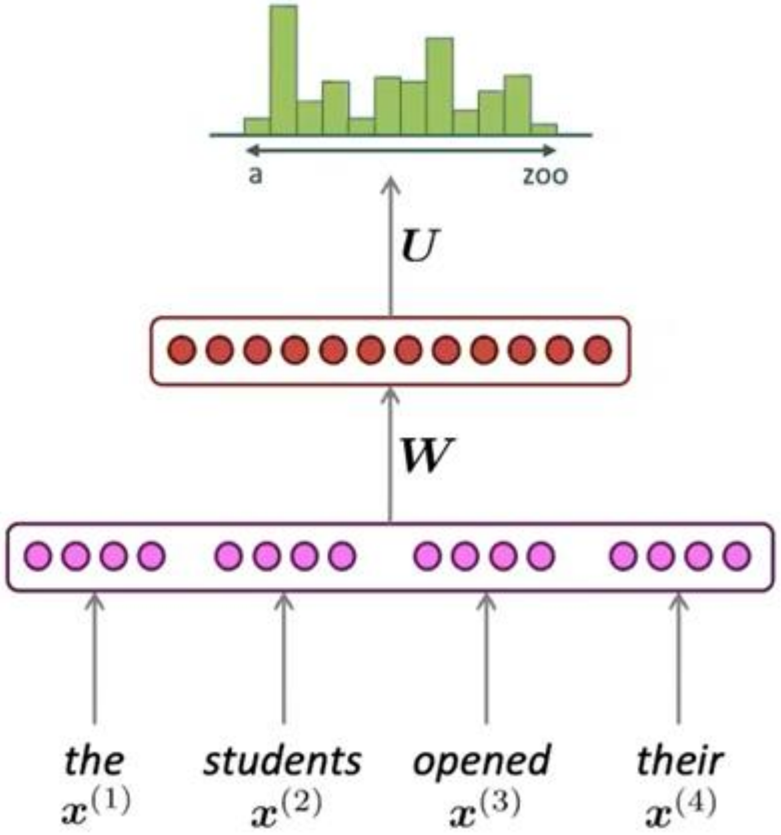
$$\boldsymbol{e} = [\boldsymbol{e}^{<1>}, \boldsymbol{e}^{<2>}, \boldsymbol{e}^{<3>}, \boldsymbol{e}^{<4>}]$$

Words

$$\boldsymbol{x}^{<1>}, \boldsymbol{x}^{<2>}, \boldsymbol{x}^{<3>}, \boldsymbol{x}^{<4>}$$

Bengio, Yoshua, Réjean Ducharme, and Pascal Vincent. "A neural probabilistic language model." Advances in neural information processing systems 13 (2000).

How to build a neural language model?



Hidden layer

$$a = g(W.e + b_a)$$

Concatenated word embeddings

$$e = [e^{<1>}, e^{<2>}, e^{<3>}, e^{<4>}]$$

Words

$$x^{<1>}, x^{<2>}, x^{<3>}, x^{<4>}$$

$x^{(1)}$ $x^{(2)}$ $x^{(3)}$ $x^{(4)}$

Bengio, Yoshua, Réjean Ducharme, and Pascal Vincent. "A neural probabilistic language model." Advances in neural information processing systems 13 (2000).

How to build a neural language model?

Output distribution

$$\hat{\boldsymbol{y}} = \boldsymbol{softmax}(\boldsymbol{U}.\boldsymbol{a} + \boldsymbol{b_y}) \in \mathbb{R}^V$$

Hidden layer

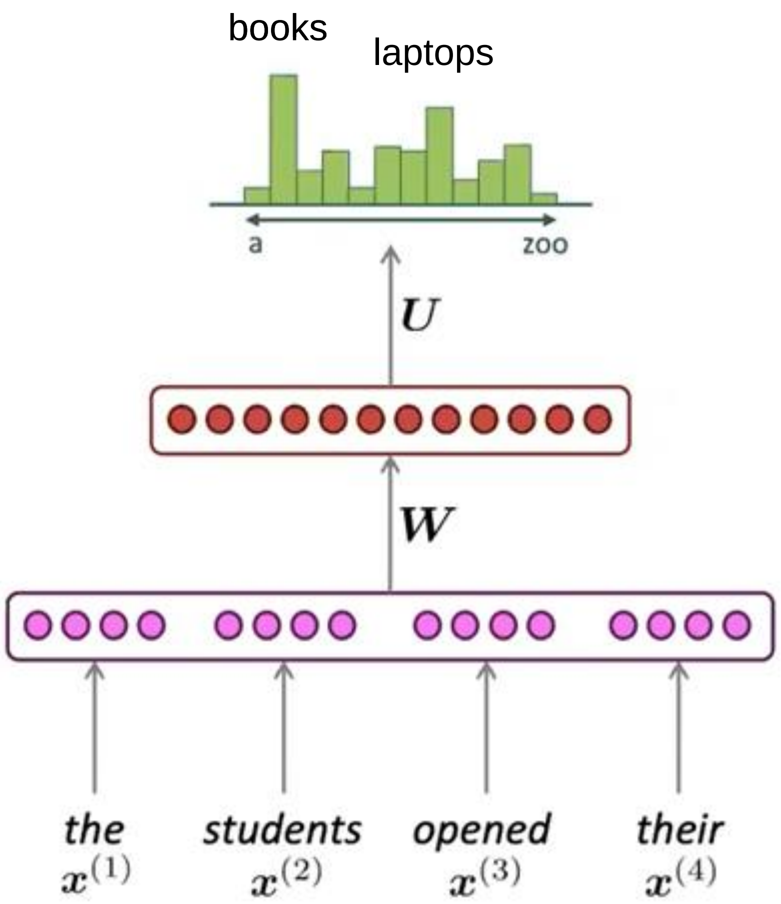
$$\boldsymbol{a} = \boldsymbol{g}(\boldsymbol{W}.\boldsymbol{e} + \boldsymbol{b_2})$$

Concatenated word embeddings

$$\boldsymbol{e} = [\boldsymbol{e}^{<1>}, \boldsymbol{e}^{<2>}, \boldsymbol{e}^{<3>}, \boldsymbol{e}^{<4>}]$$

Words

$$\boldsymbol{x}^{<1>}, \boldsymbol{x}^{<2>}, \boldsymbol{x}^{<3>}, \boldsymbol{x}^{<4>}$$



Bengio, Yoshua, Réjean Ducharme, and Pascal Vincent. "A neural probabilistic language model." Advances in neural information processing systems 13 (2000).

- Improvements over n-grams LMs:
  - No Sparsity, low storage.
  - Use of distributional semantics

Output distribution

$\hat{\boldsymbol{y}} = \boldsymbol{softmax}(\boldsymbol{U}.\boldsymbol{a} + \boldsymbol{b_y}) \in \mathbb{R}^V$

Hidden layer

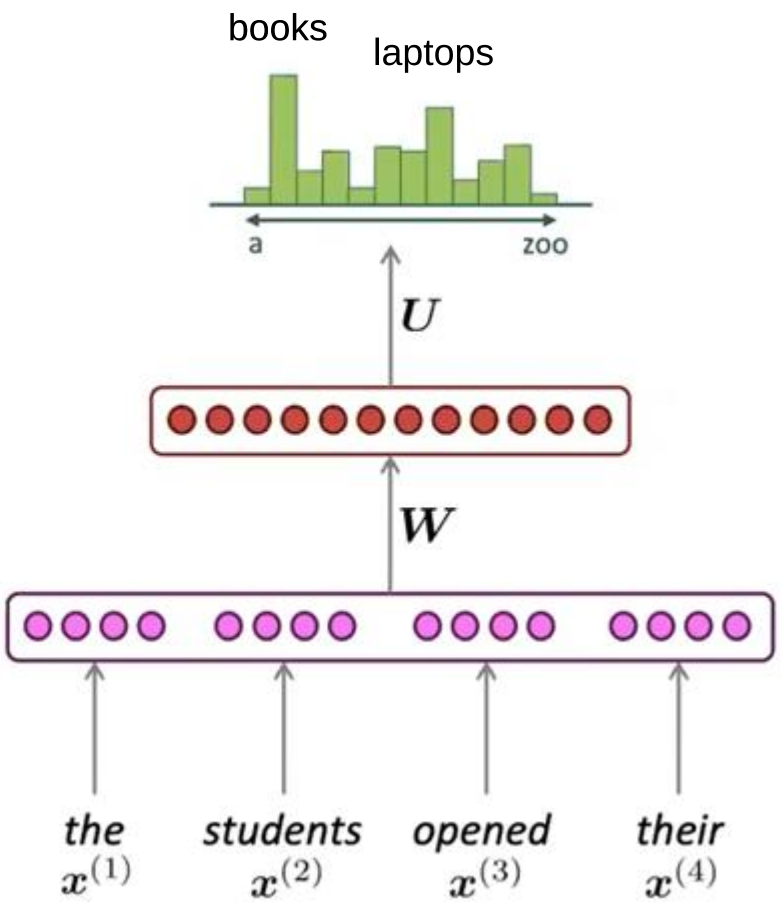
$\boldsymbol{a} = \boldsymbol{g}(\boldsymbol{W}.\boldsymbol{e} + \boldsymbol{b_2})$

Concatenated word embeddings

$\boldsymbol{e} = [\boldsymbol{e}^{<1>}, \boldsymbol{e}^{<2>}, \boldsymbol{e}^{<3>}, \boldsymbol{e}^{<4>}]$

Words

$\boldsymbol{x}^{<1>}, \boldsymbol{x}^{<2>}, \boldsymbol{x}^{<3>}, \boldsymbol{x}^{<4>}$



Bengio, Yoshua, Réjean Ducharme, and Pascal Vincent. "A neural probabilistic language model." Advances in neural information processing systems 13 (2000).

How to build a neural language model?

- Improvements over n-grams LMs:
  - No Sparsity, low storage.
  - Use of distributional semantics.
- Remaining Problems:
  - Fixed window is too small.
  - Enlarging window enlarges W.
  - Window can never be large enough.
  - Consecutive words are multiplied by different weights.

Output distribution

$$\hat{\boldsymbol{y}} = \boldsymbol{softmax}(\boldsymbol{U}.\boldsymbol{a} + \boldsymbol{b_y}) \in \mathbb{R}^V$$

Hidden layer

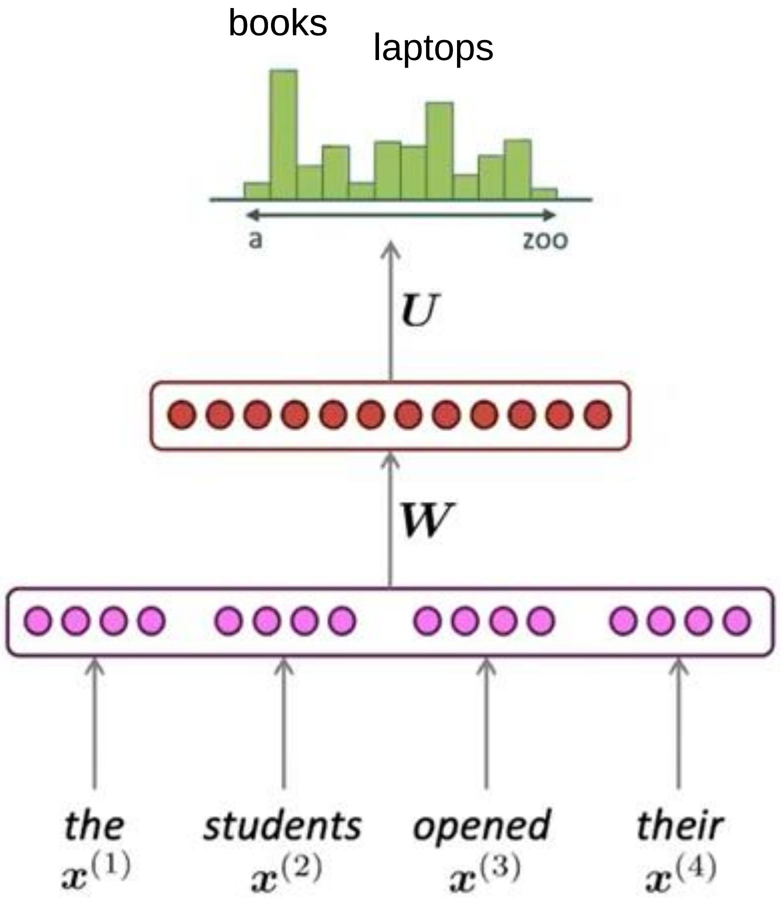
$$\boldsymbol{a} = \boldsymbol{g}(\boldsymbol{W}.\boldsymbol{e} + \boldsymbol{b_2})$$

Concatenated word embeddings

$$\boldsymbol{e} = [\boldsymbol{e}^{<1>}, \boldsymbol{e}^{<2>}, \boldsymbol{e}^{<3>}, \boldsymbol{e}^{<4>}]$$

Words

$$\boldsymbol{x}^{<1>}, \boldsymbol{x}^{<2>}, \boldsymbol{x}^{<3>}, \boldsymbol{x}^{<4>}$$



Bengio, Yoshua, Réjean Ducharme, and Pascal Vincent. "A neural probabilistic language model." Advances in neural information processing systems 13 (2000).

How to build a neural language model?

- Improvements over n-grams LMs:
  - No Sparsity, low storage.
  - Use of distributional semantics.
- Remaining Problems:
  - Fixed window is too small.
  - Enlarging window enlarges W.
  - Window can never be large enough.
  - Consecutive words are multiplied by different weights.
- Solution?
  - A neural architecture that can share weights and process variable-length input.

Output distribution

$$\hat{\boldsymbol{y}} = \boldsymbol{softmax}(\boldsymbol{U}.\boldsymbol{a} + \boldsymbol{b_y}) \in \mathbb{R}^V$$

Hidden layer

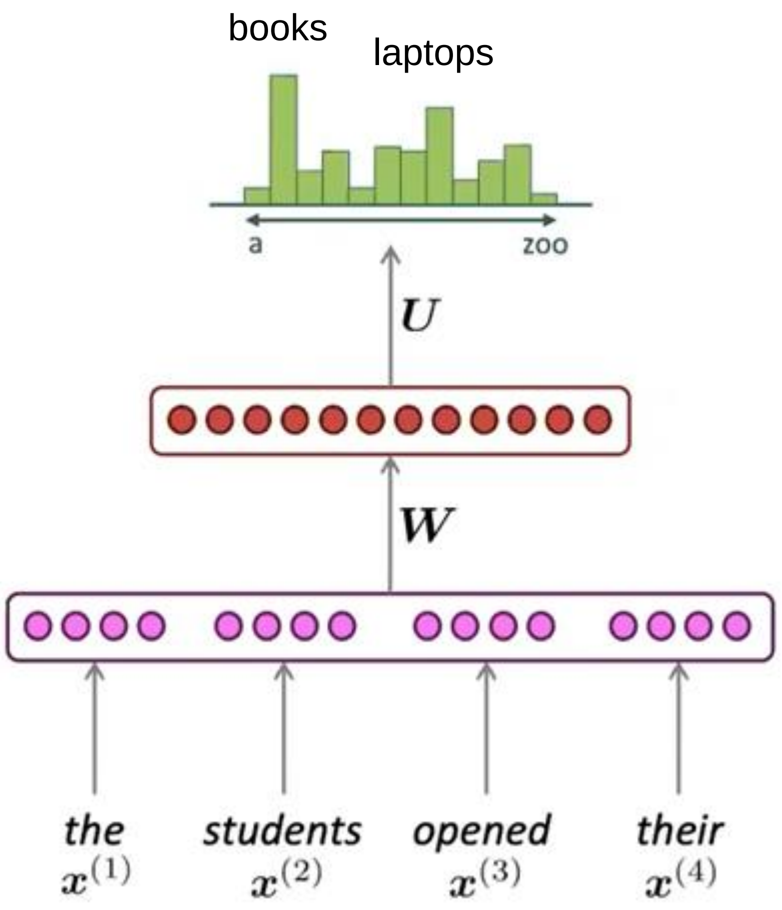
$$\boldsymbol{a} = \boldsymbol{g}(\boldsymbol{W}.\boldsymbol{e} + \boldsymbol{b_2})$$

Concatenated word embeddings

$$\boldsymbol{e} = [\boldsymbol{e}^{<1>}, \boldsymbol{e}^{<2>}, \boldsymbol{e}^{<3>}, \boldsymbol{e}^{<4>}]$$

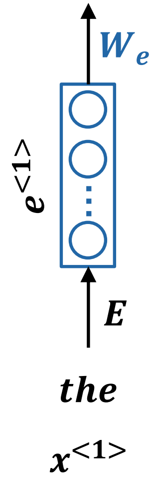
Words

$$\boldsymbol{x}^{<1>}, \boldsymbol{x}^{<2>}, \boldsymbol{x}^{<3>}, \boldsymbol{x}^{<4>}$$

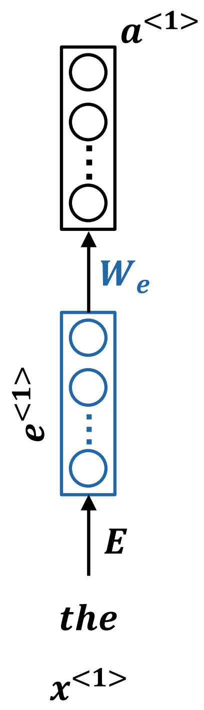


Bengio, Yoshua, Réjean Ducharme, and Pascal Vincent. "A neural probabilistic language model." Advances in neural information processing systems 13 (2000).

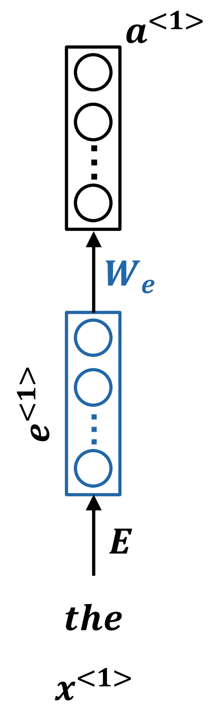
Recurrent Neural Networks are better suited for sequence modelling



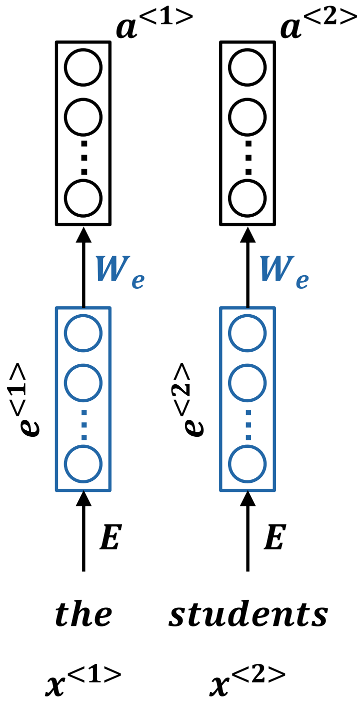
Recurrent Neural Networks are better suited for sequence modelling



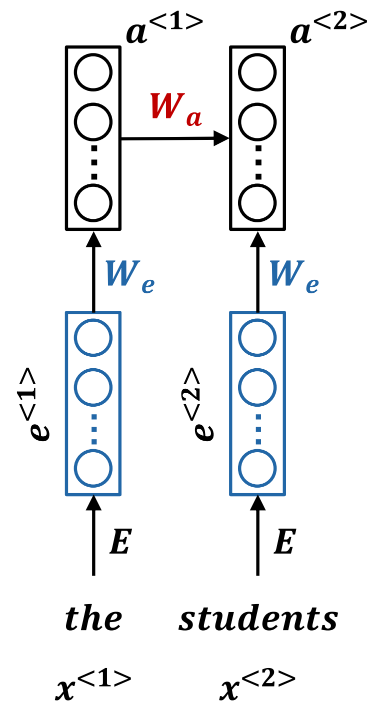
Recurrent Neural Networks are better suited for sequence modelling



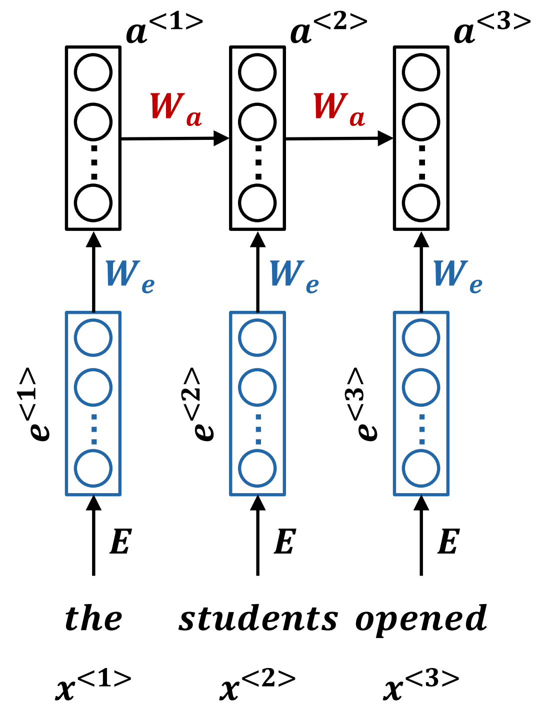
Recurrent Neural Networks are better suited for sequence modelling



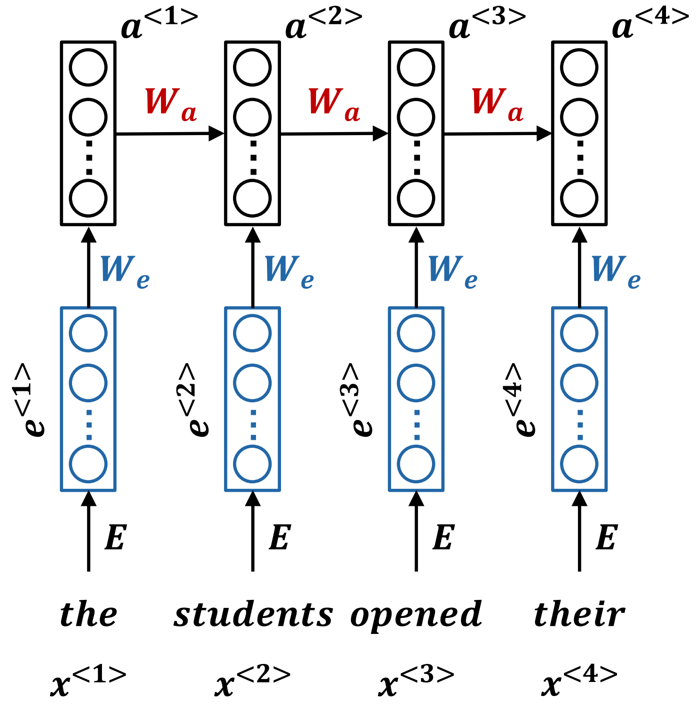
Recurrent Neural Networks are better suited for sequence modelling



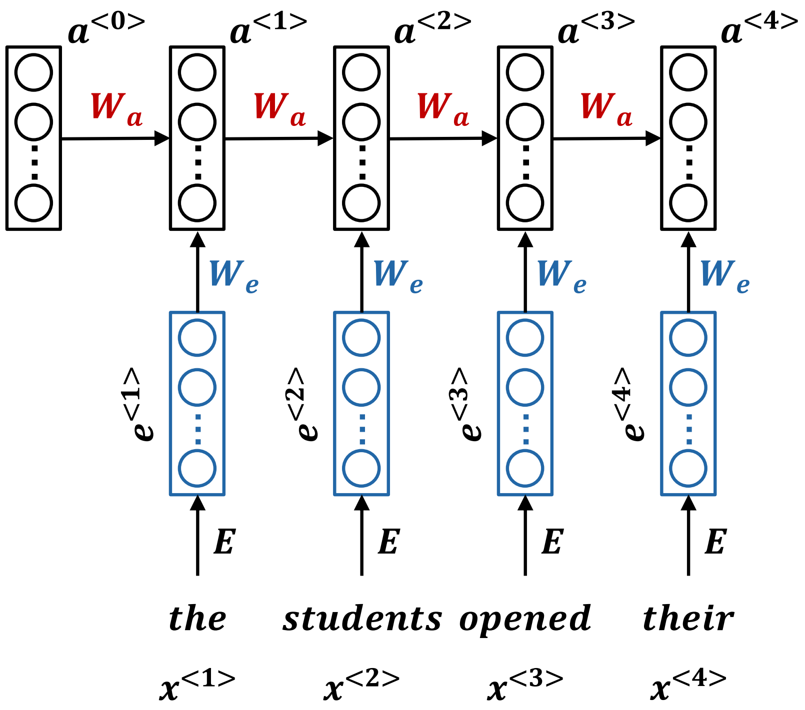
Recurrent Neural Networks are better suited for sequence modelling



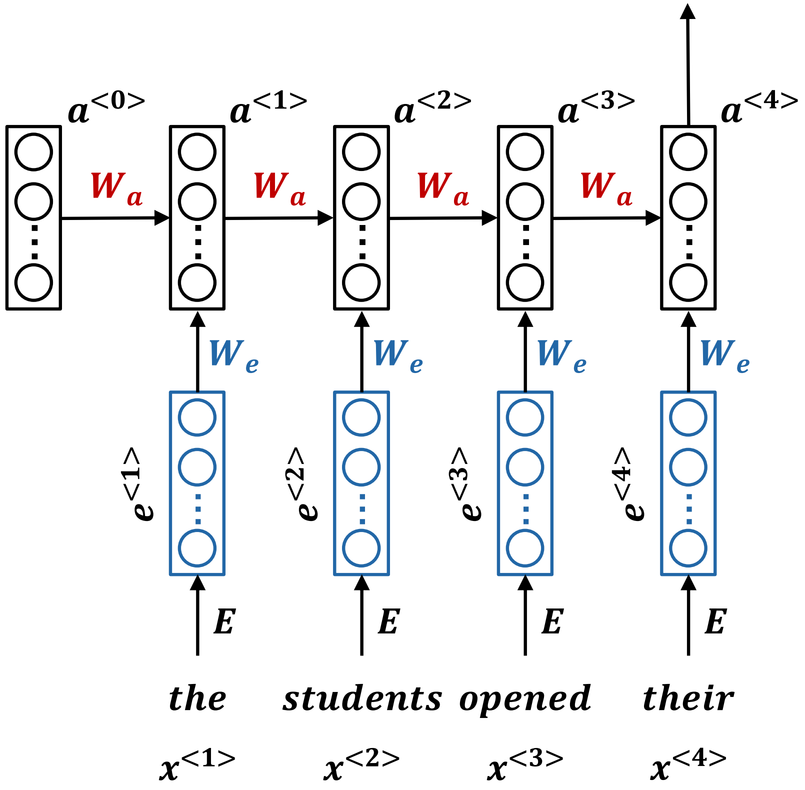
Recurrent Neural Networks are better suited for sequence modelling



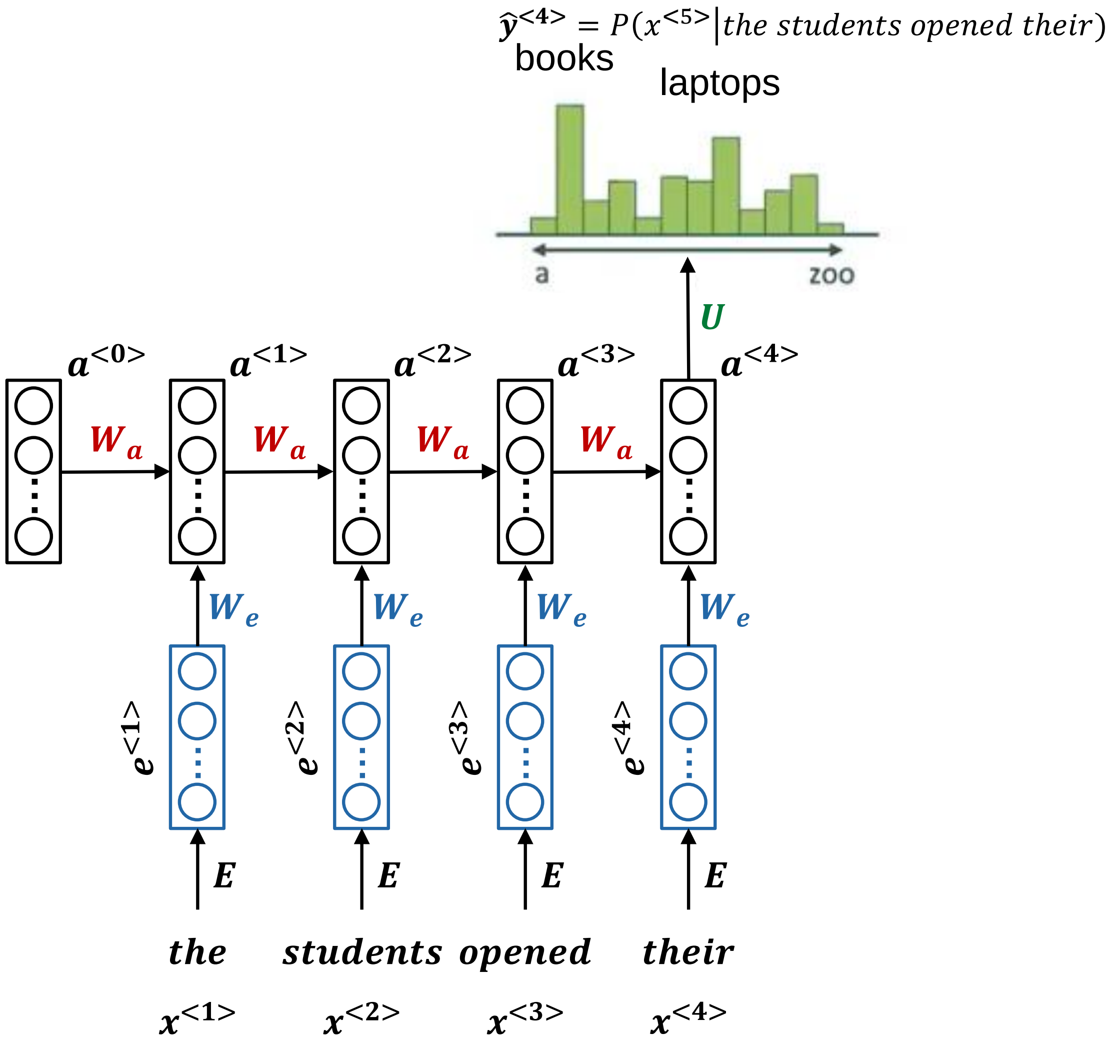
Recurrent Neural Networks are better suited for sequence modelling



Recurrent Neural Networks are better suited for sequence modelling



Recurrent Neural Networks are better suited for sequence modelling



Recurrent Neural Networks are better suited for sequence modelling

Output distribution

$$\hat{y} = softmax(Ua^{<t>} + b_y) \in \mathbb{R}^V$$

Hidden states

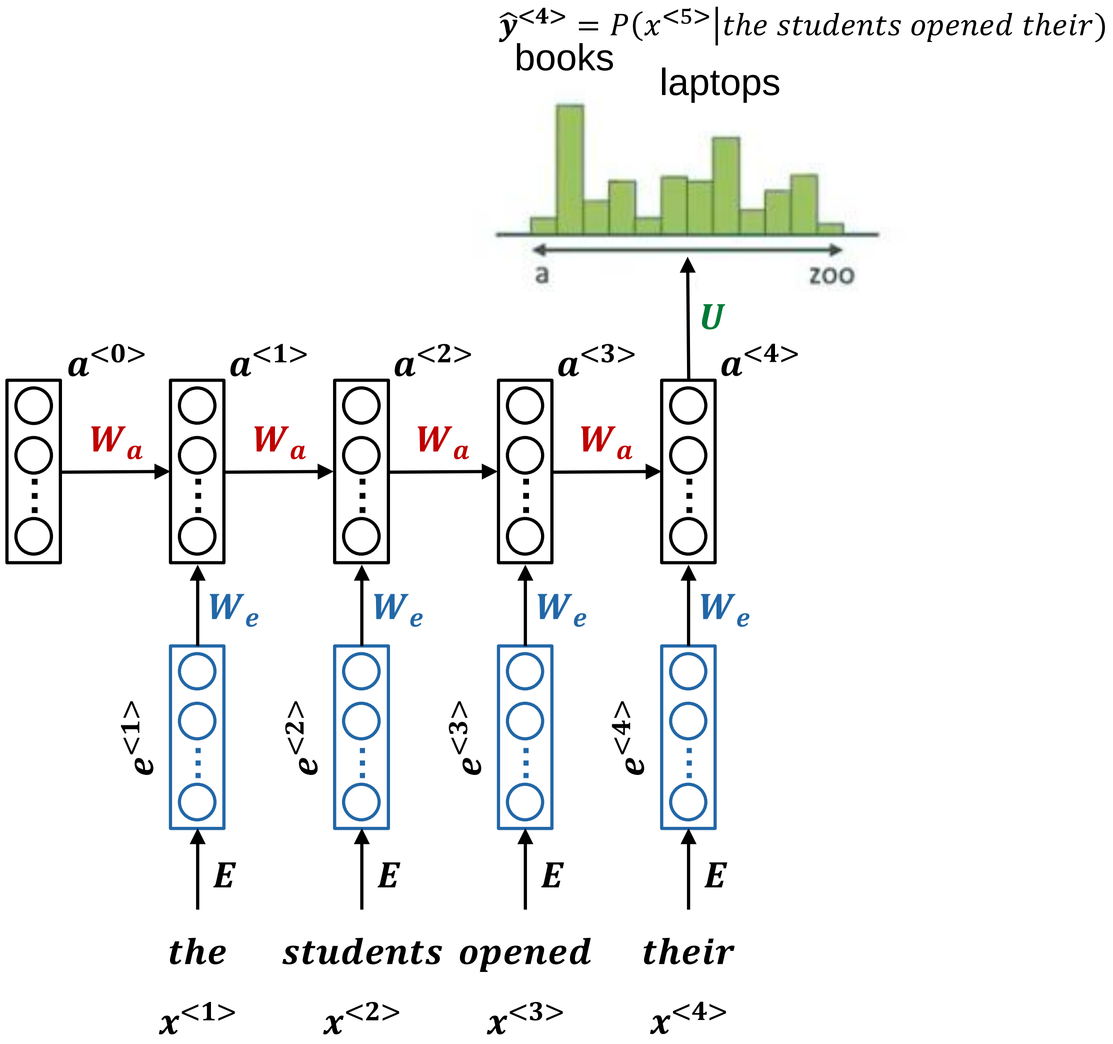
$$a^{<t>} = tanh(W_a a^{<t-1>} + W_e e^{<t>} + b_a)$$

Word embeddings

$$e^{<t>} = Ex^{<t>}$$

Words/one-hot vectors

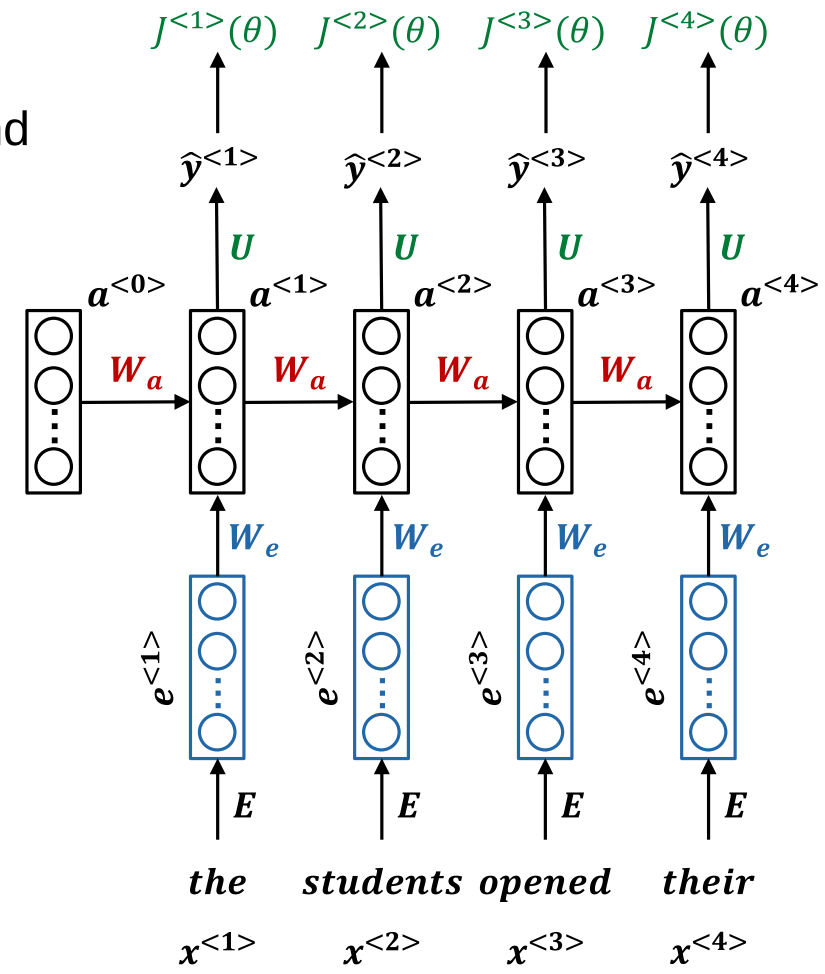
$$x^{<t>} \in \mathbb{R}^V$$



How to train an RNN-based Language Model?

- Get a large corpus of text.

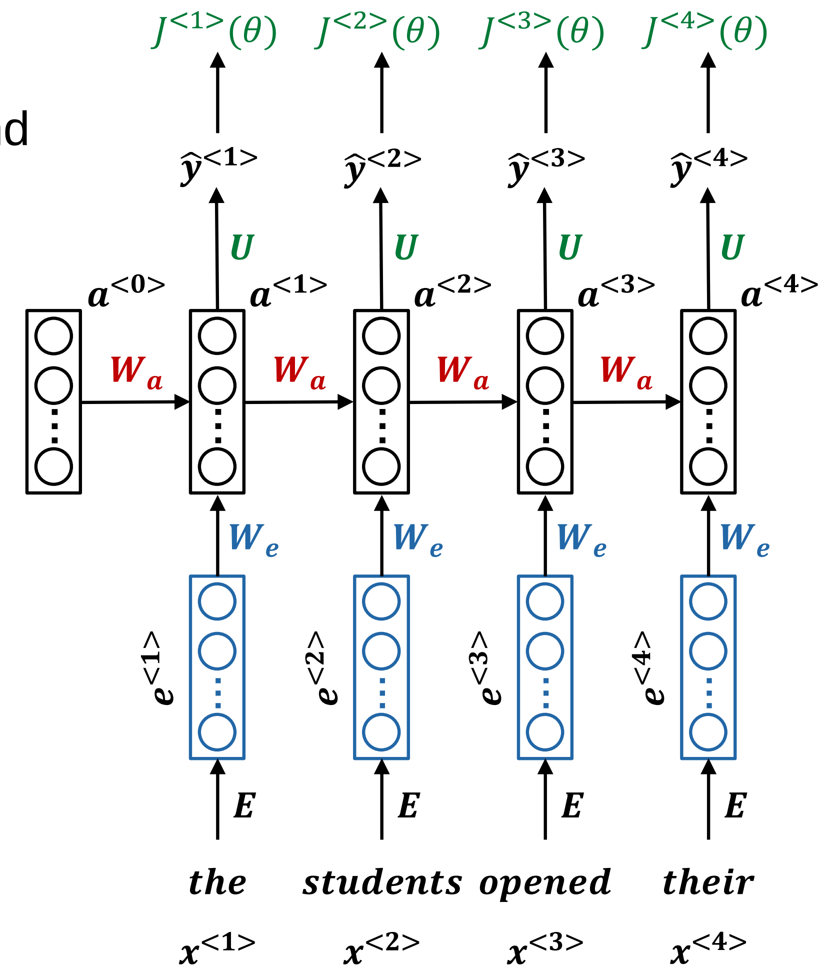
- Feed the sequence of word into RNN model one by one and calculated $\hat{y}^{<t>}$ for every time step t.



How to train an RNN-based Language Model?

- Get a large corpus of text.
- Feed the sequence of word into RNN model one by one and calculated $\hat{y}^{<t>}$ for every time step t.
- Calculate loss function at each time step.
- Average the loss values to get cost for the entire training set.

$$J\,(\theta) = \frac{1}{T}\sum_{t=1}^{T} J^{<t>}(\theta) = \frac{1}{T}\sum_{t=1}^{T} -\log \hat{y}^{<t>}_{x_{t+1}}$$

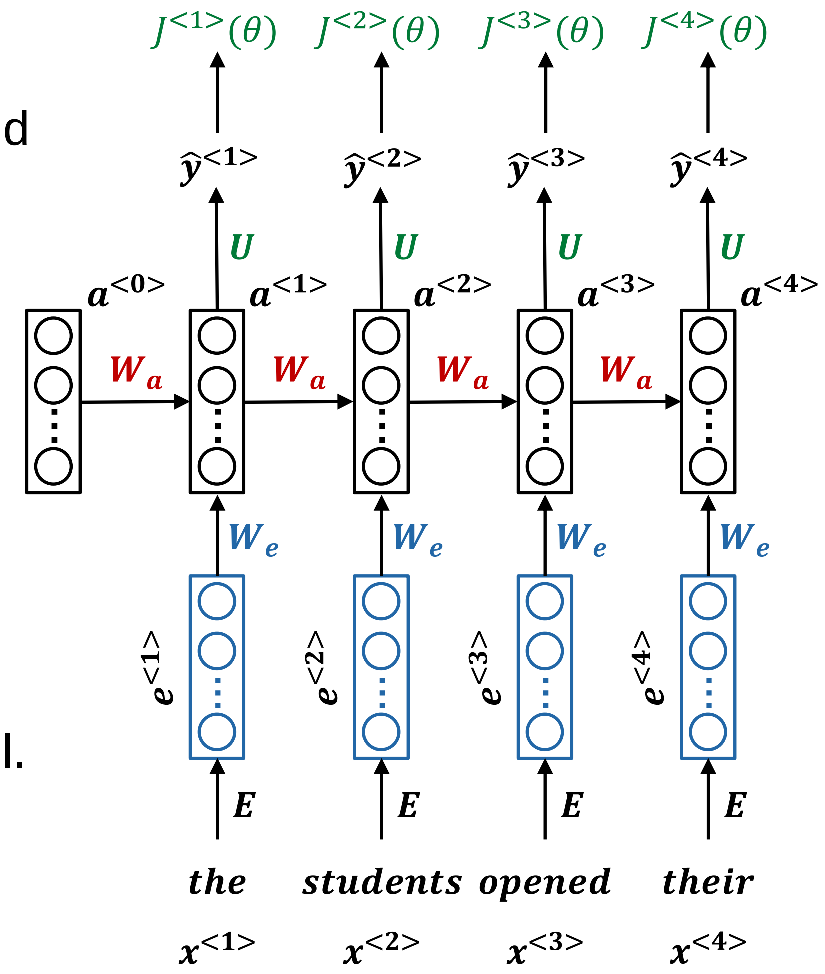


How to train an RNN-based Language Model?

- Get a large corpus of text.
- Feed the sequence of word into RNN model one by one and calculated $\hat{y}^{<t>}$ for every time step t.
- Calculate loss function at each time step.
- Average the loss values to get cost for the entire training set.

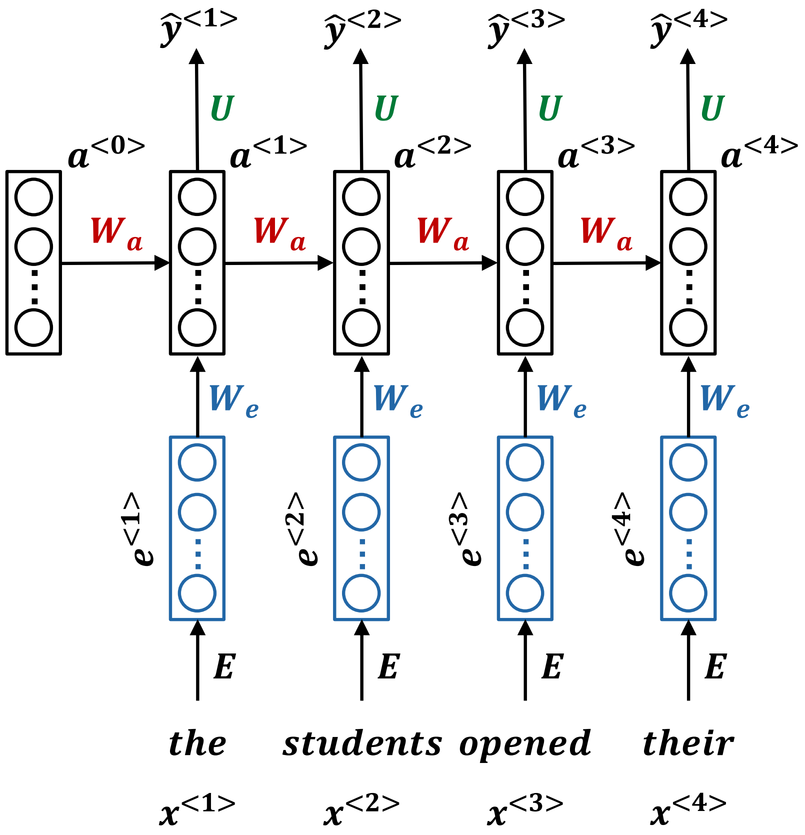
$$J(\theta) = \frac{1}{T}\sum_{t=1}^{T} J^{<t>}(\theta) = \frac{1}{T}\sum_{t=1}^{T} -\log \hat{y}_{x_{t+1}}^{<t>}$$

- **Teacher Forcing** is used during training of language model.

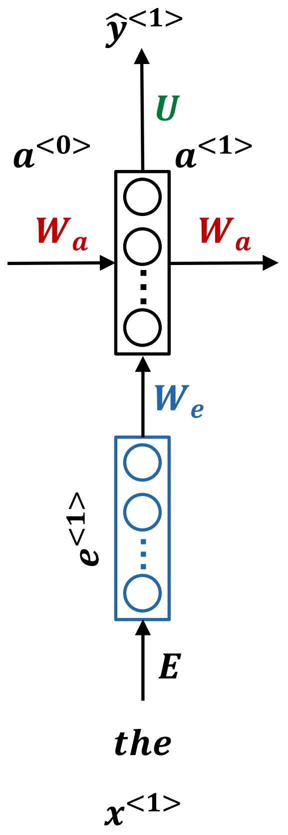


NUST Defining futures
School of Electrical Engineering & Computer Science

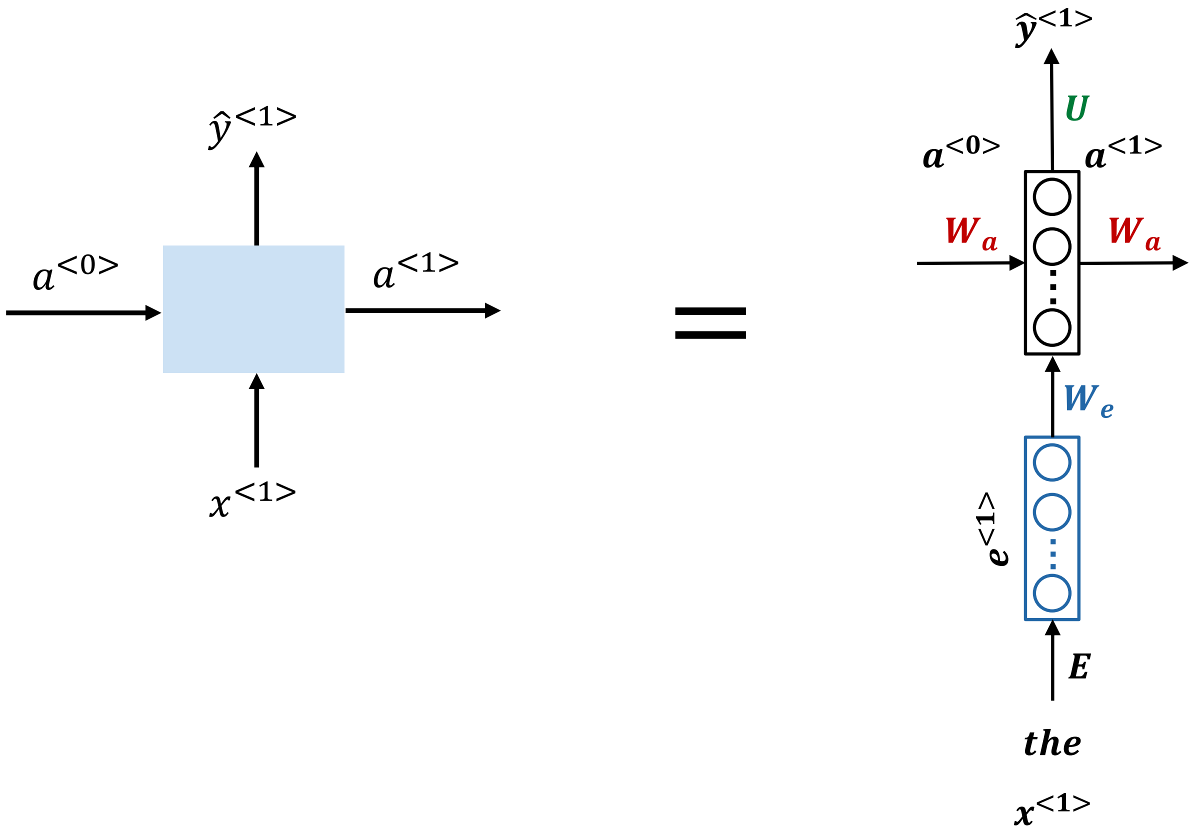
RNN-based models make recurrent use of the same unit



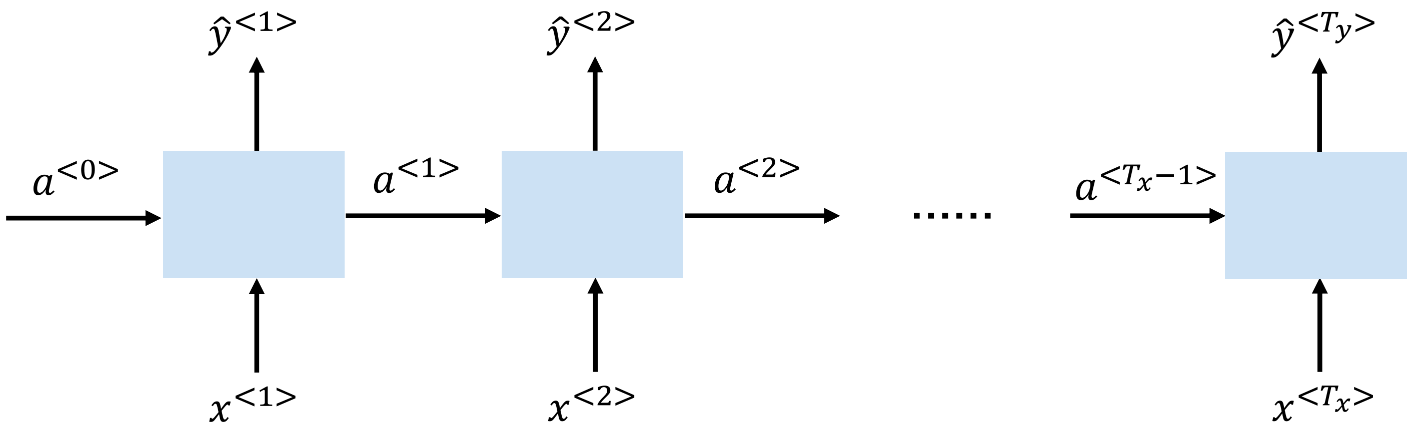
RNN-based models make recurrent use of the same unit



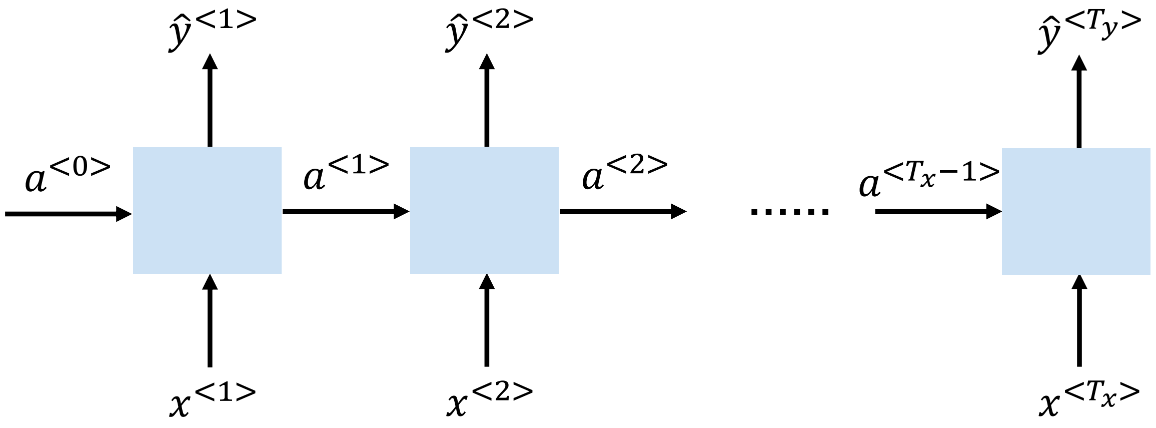
RNN-based models make recurrent use of the same unit



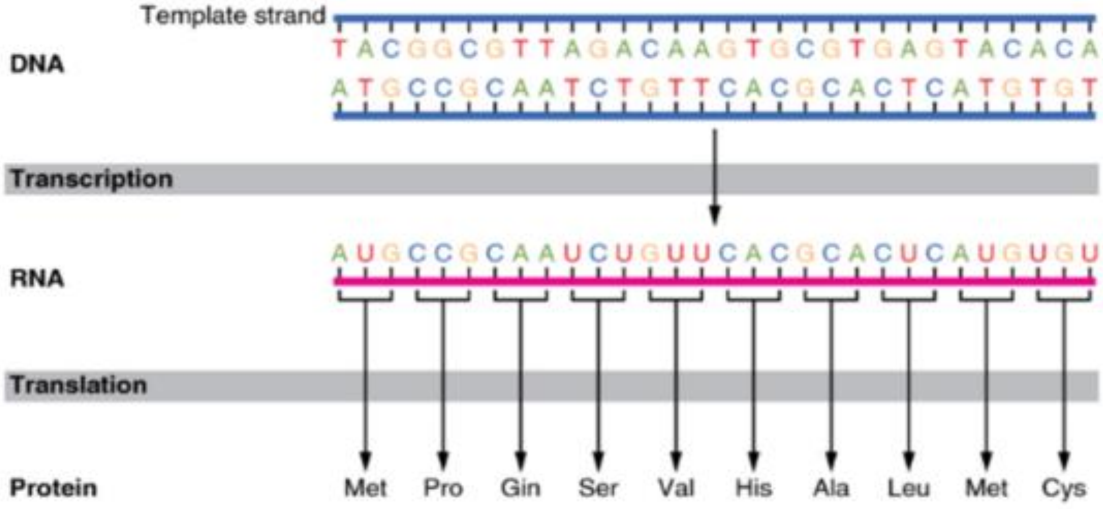
RNN-based models make recurrent use of the same unit



Architecture of RNN will change depending on input/output relationships

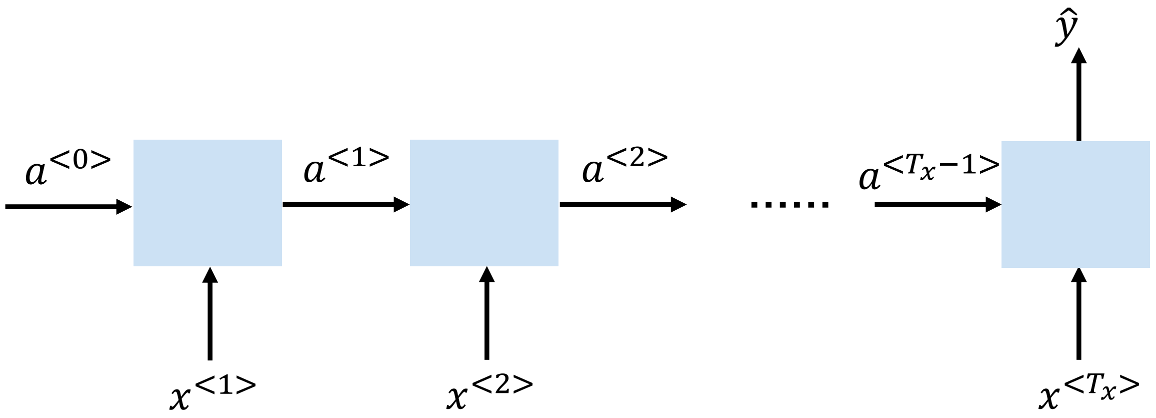


Many-to-Many



DNA Sequencing

Architecture of RNN will change depending on input/output relationships



Many-to-One

I paid 100 Euros for a really flavourless food and not so delightful ambience.

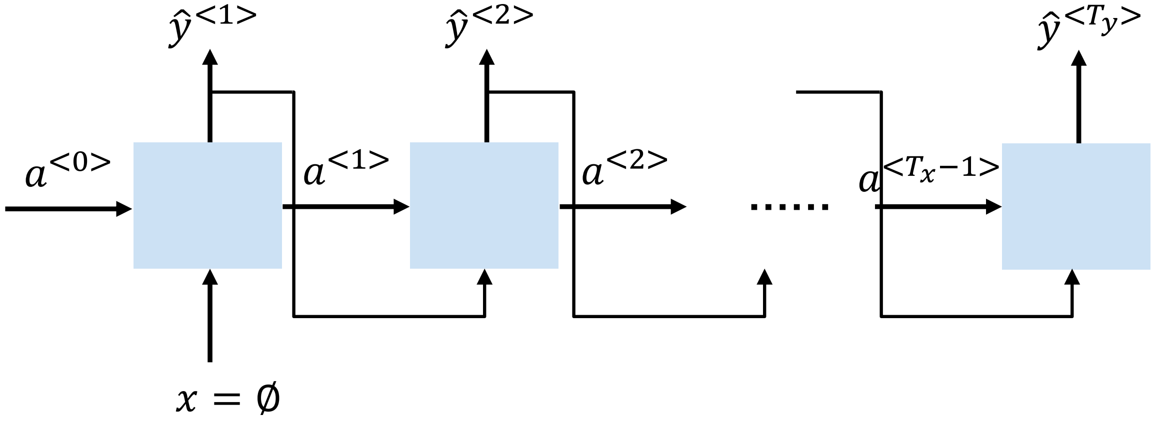
Food was fine and I wouldn't say it was the best place I have ever tried.

We loved the food. Menu is perfect in here, something for everyone. Visiting this one again.

Sentiment Analysis

Architecture of RNN will change depending on input/output relationships

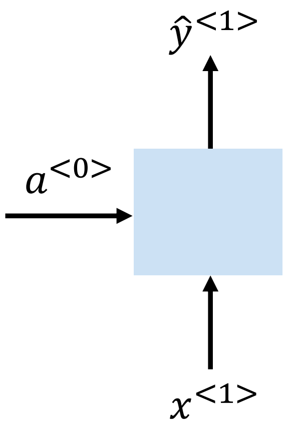
Music Generation



One-to-Many

Architecture of RNN will change depending on input/output relationships

- There’s no NLP task that requires a one-to-one model architecture.

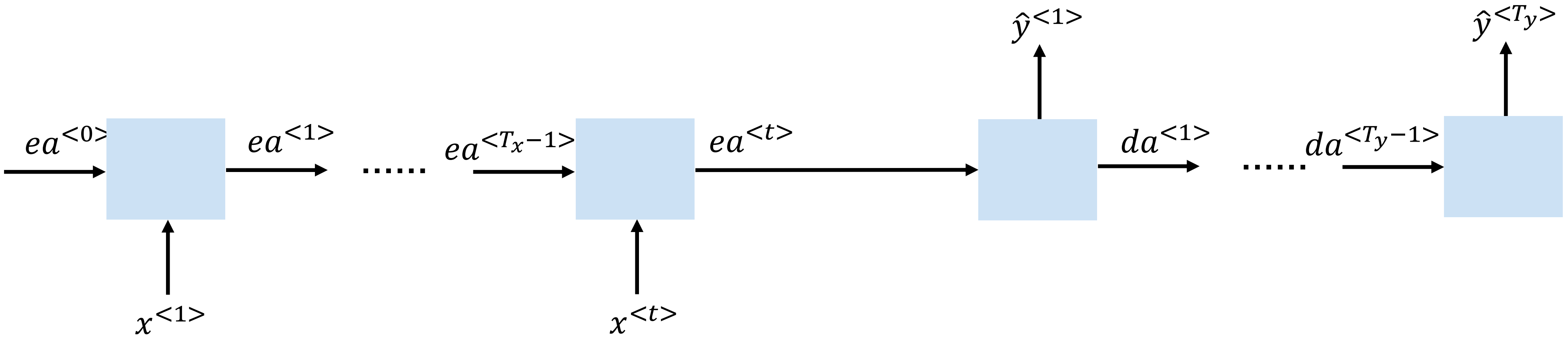


One-to-One

Encoder-Decoder model is a special case of many-to-many architecture

Ich werde nächste Woche Urlaub nehmen.

I am taking a vacation next week.

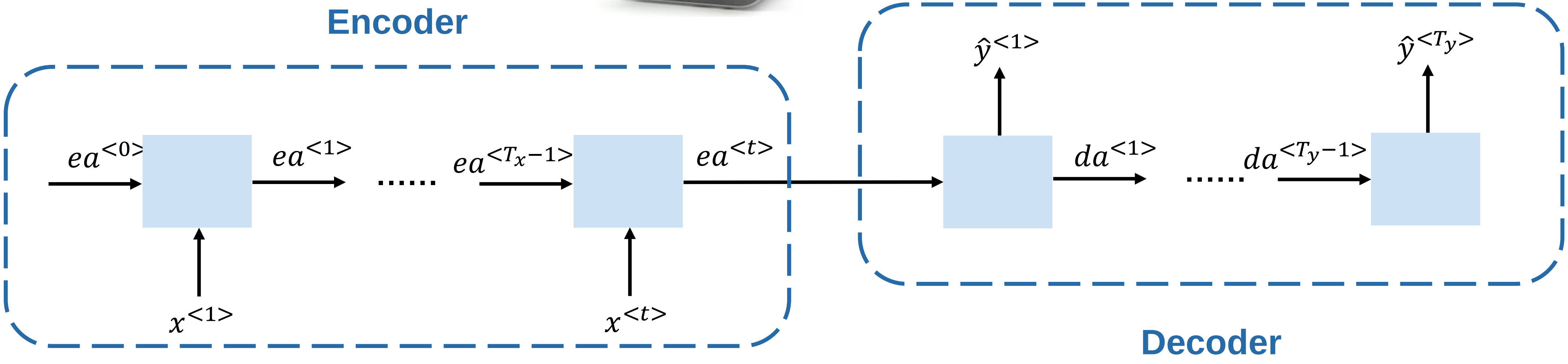


Many-to-Many V2

Encoder-Decoder model is a special case of many-to-many architecture

Ich werde nächste Woche Urlaub nehmen.

I am taking a vacation next week.



Many-to-Many V2

Let's simplify RNN equations and schematics

$$a^{<t>} = g\left(W[a^{<t-1>},\ x^{<t>}] + b_y\right)$$

$$\hat{y}^{<t>} = g(Ua^{<t>} + b_a)$$

$[W_a \,\vdots\, W_e] = W$ (Think of it as a vector of matrices)

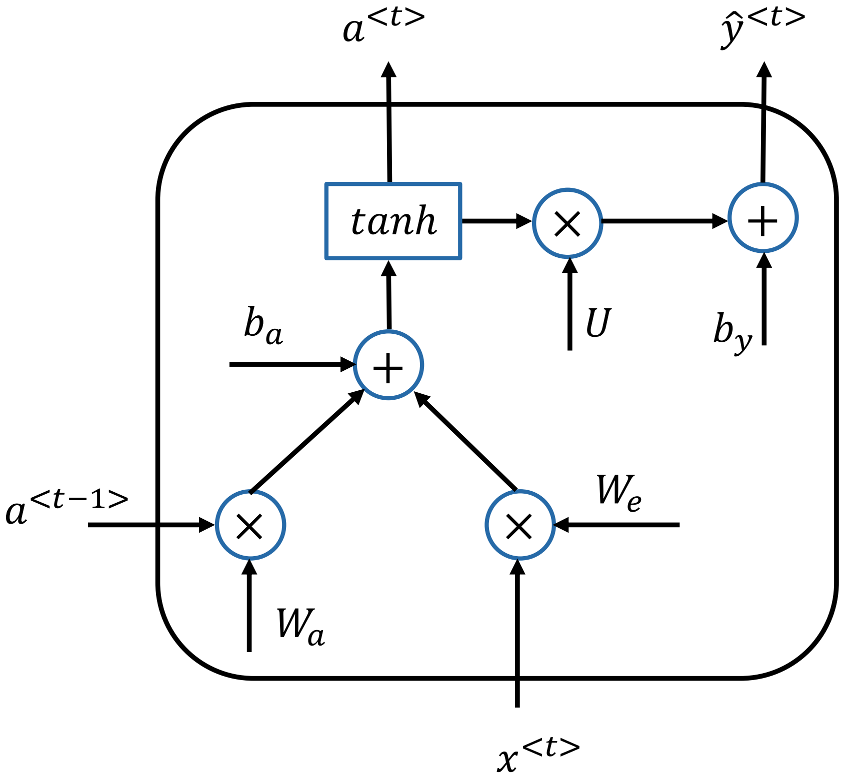
$\begin{bmatrix} a^{<t-1>} \\ x^{<t>} \end{bmatrix} = [a^{<t-1>},\ x^{<t>}]$ (Think of it as a vector of vectors)

$$[W_a \,\vdots\, W_e] \times \begin{bmatrix} a^{<t-1>} \\ x^{<t>} \end{bmatrix} = W_a a^{<t-1>} + W_e x^{<t>}$$

Let's simplify RNN equations and schematics

$$a^{<t>} = g\left(W[a^{<t-1>},\ x^{<t>}] + b_y\right)$$

$$\hat{y}^{<t>} = g(Ua^{<t>} + b_a)$$



$[W_a \mid W_e] = W$ (Think of it as a vector of matrices)

$\begin{bmatrix} a^{<t-1>} \\ x^{<t>} \end{bmatrix} = [a^{<t-1>},\ x^{<t>}]$ (Think of it as a vector of vectors)

$[W_a \mid W_e] \times \begin{bmatrix} a^{<t-1>} \\ x^{<t>} \end{bmatrix} = W_a a^{<t-1>} + W_e x^{<t>}$

Let’s simplify RNN equations and schematics

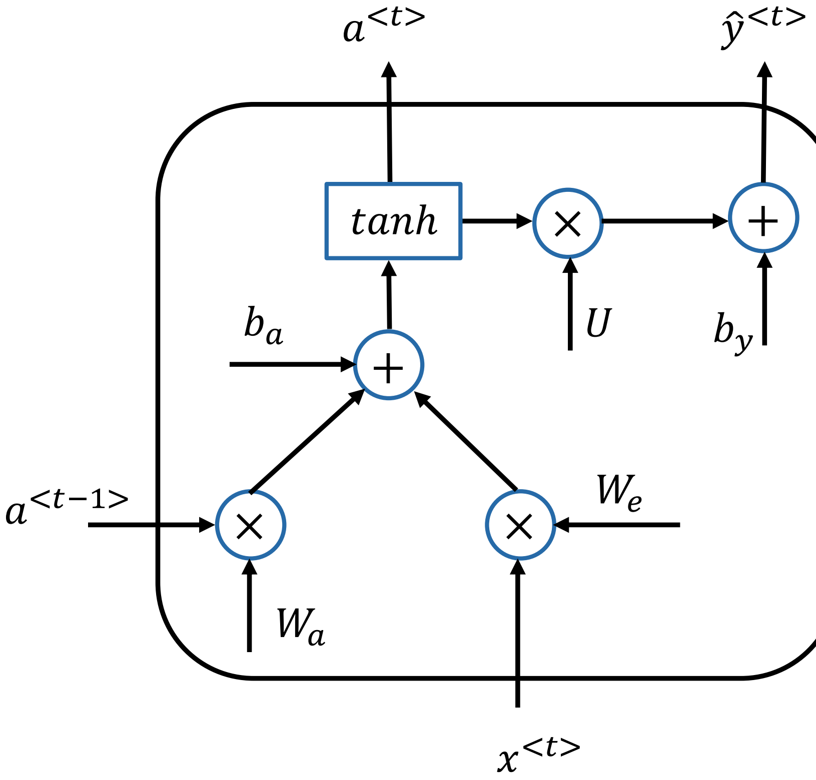
$$a^{<t>} = g\left(W[a^{<t-1>},\ x^{<t>}] + b_y\right)$$

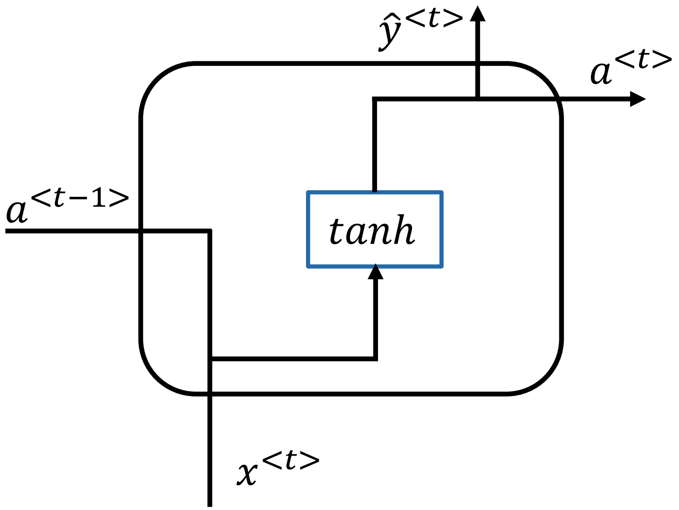
$$\hat{y}^{<t>} = g(Ua^{<t>} + b_a)$$

$[W_a \,\vert\, W_e] = W$ (Think of it as a vector of matrices)

$\begin{bmatrix} a^{<t-1>} \\ x^{<t>} \end{bmatrix} = [a^{<t-1>},\ x^{<t>}]$ (Think of it as a vector of vectors)

$$[W_a \,\vert\, W_e] \times \begin{bmatrix} a^{<t-1>} \\ x^{<t>} \end{bmatrix} = W_a a^{<t-1>} + W_e x^{<t>}$$



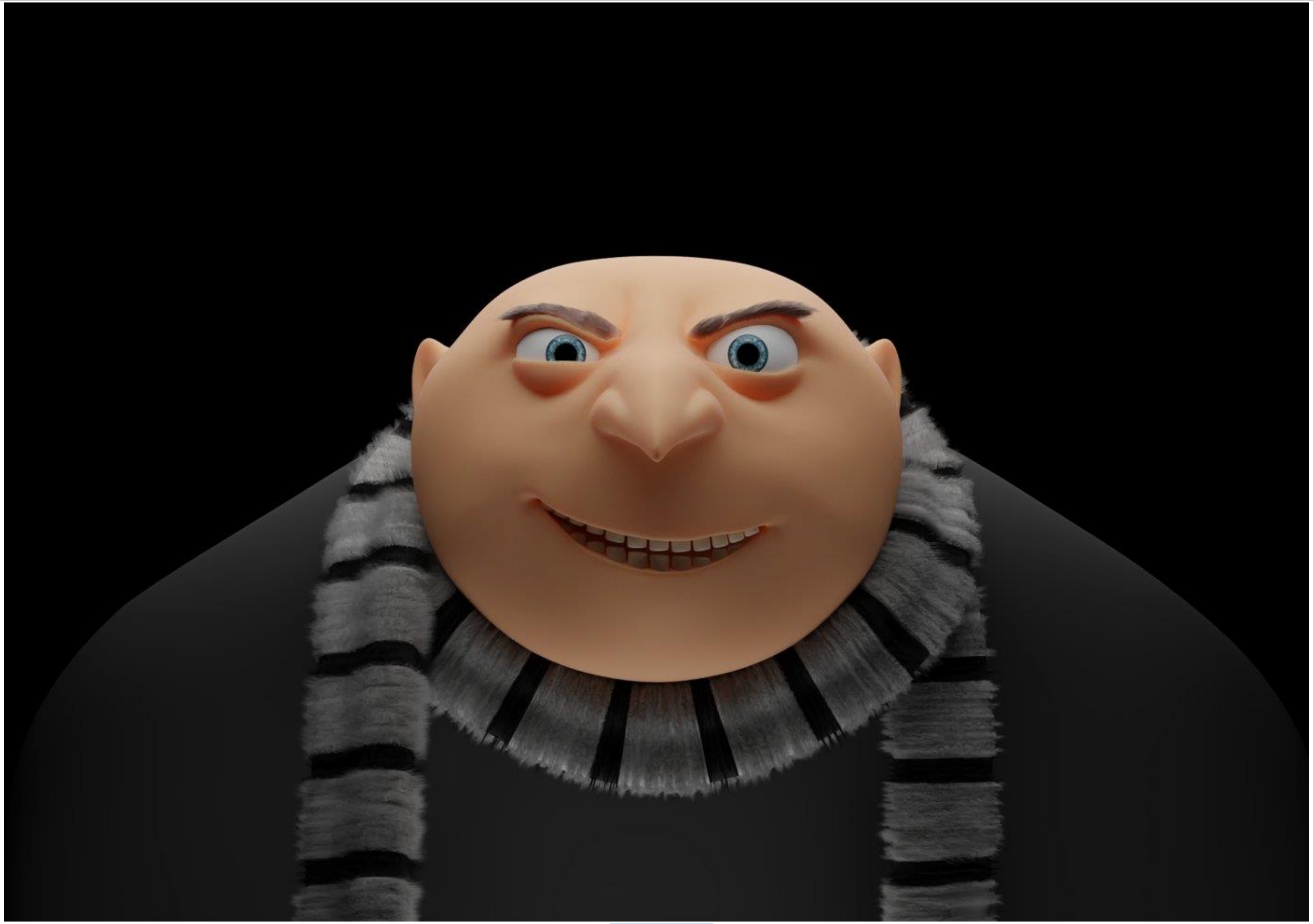


Recurrent Neural Networks are still not perfect

- Advantages of RNNs:
  - Can process variable-length input.
  - In theory, may incorporation of indefinite context.
  - Model size is fixed.

Recurrent Neural Networks are still not perfect

- Advantages of RNNs:
  - Can process variable-length input.
  - In theory, may incorporation of indefinite context.
  - Model size is fixed.
- Disadvantages of RNNs:
  - Computations are sequential.
  - Difficult to access information from many steps back.
- Solution?



Gated Recurrent Unit (GRU) stores previous information for future use

- Let c is a Memory **C**ell.

- For now, let $c^{<t>} = a^{<t>}$.

https://www.youtube.com/watch?v=xSCy3q2ts44

Gated Recurrent Unit (GRU) stores previous information for future use

- Let c is a Memory **C**ell.
- For now, let $c^{<t>} = a^{<t>}$.
- At every time step t, we calculate a new candidate memory $\tilde{c}$, which may or may not replace older c.

$$\tilde{c}^{<t>} = tanh(W_c[c^{<t-1>}, \quad x^{<t>}] + b_c)$$

https://www.youtube.com/watch?v=xSCy3q2ts44

Gated Recurrent Unit (GRU) stores previous information for future use

- Let c is a Memory **C**ell.

- For now, let $c^{<t>} = a^{<t>}$.

- At every time step t, we calculate a new candidate memory $\tilde{c}$, which may or may not replace older c.

$$\tilde{c}^{<t>} = tanh(W_c[c^{<t-1>}, \quad x^{<t>}] + b_c)$$

- Then we have a gate (hence the name) that decides whether c should be updated.

$$\Gamma_u = \sigma(W_u[c^{<t-1>}, \quad x^{<t>}] + b_u)$$

https://www.youtube.com/watch?v=xSCy3q2ts44

Gated Recurrent Unit (GRU) stores previous information for future use

- Let c is a Memory **C**ell.

- For now, let $c^{<t>} = a^{<t>}$.

- At every time step t, we calculate a new candidate memory $\tilde{c}$, which may or may not replace older c.

$$\tilde{c}^{<t>} = tanh(W_c[c^{<t-1>}, \quad x^{<t>}] + b_c)$$

- Then we have a gate (hence the name) that decides whether c should be updated.

$$\Gamma_u = \sigma(W_u[c^{<t-1>}, \quad x^{<t>}] + b_u)$$

- So the update equation for c becomes,

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + (1 - \Gamma_u) * c^{<t-1>}$$

https://www.youtube.com/watch?v=xSCy3q2ts44

Simplified model of GRU has only one gate

- Governing equations of simplified GRU:

$$\tilde{c}^{<t>} = tanh(W_c[c^{<t-1>},\ x^{<t>}] +\ b_c)$$

$$\Gamma_u = \sigma(W_u[c^{<t-1>},\ x^{<t>}] +\ b_u)$$

$$c^{<t>} = \Gamma_u\ * \tilde{c}^{<t>} + (1 - \Gamma_u)\ * c^{<t-1>}$$

https://www.youtube.com/watch?v=xSCy3q2ts44

Simplified model of GRU has only one gate

- Governing equations of simplified GRU:

$$\tilde{c}^{<t>} = tanh(W_c[c^{<t-1>},\ x^{<t>}] + \ b_c)$$

$$\Gamma_u = \sigma(W_u[c^{<t-1>},\ x^{<t>}] + \ b_u)$$

$$c^{<t>} = \Gamma_u \ * \tilde{c}^{<t>} + (1 - \Gamma_u) \ * c^{<t-1>}$$

- c and Γ are not single values. Therefore, they can capture multiple aspects from previous time steps.

https://www.youtube.com/watch?v=xSCy3q2ts44

Simplified model of GRU has only one gate

- Governing equations of simplified GRU:

$$\tilde{c}^{<t>} = tanh(W_c[c^{<t-1>},\ x^{<t>}] + \ b_c)$$

$$\Gamma_u = \sigma(W_u[c^{<t-1>},\ x^{<t>}] + \ b_u)$$

$$c^{<t>} = \Gamma_u \ * \tilde{c}^{<t>} + (1 - \Gamma_u) \ * c^{<t-1>}$$

- c and Γ are not single values. Therefore, they can capture multiple aspects from previous time steps.

- What if Γ is in between 0 and 1?

https://www.youtube.com/watch?v=xSCy3q2ts44

Complete model of GRU has two gates

- Governing equations of complete GRU

$$\tilde{c}^{<t>} = tanh(W_c[\Gamma_r * c^{<t-1>},\ x^{<t>}] + \ b_c)$$

$$\Gamma_u = \sigma(W_u[c^{<t-1>},\ x^{<t>}] + \ b_u)$$

$$\Gamma_r = \sigma(W_r[c^{<t-1>},\ x^{<t>}] + \ b_r)$$

$$c^{<t>} = \Gamma_u \ * \tilde{c}^{<t>} + (1 - \Gamma_u)\ * c^{<t-1>}$$

- Governing equations of simplified GRU:

$$\tilde{c}^{<t>} = tanh(W_c[c^{<t-1>},\ x^{<t>}] + \ b_c)$$

$$\Gamma_u = \sigma(W_u[c^{<t-1>},\ x^{<t>}] + \ b_u)$$

$$c^{<t>} = \Gamma_u \ * \tilde{c}^{<t>} + (1 - \Gamma_u)\ * c^{<t-1>}$$

https://www.youtube.com/watch?v=xSCy3q2ts44

Complete model of GRU has two gates

- Governing equations of complete GRU

$$\tilde{c}^{<t>} = tanh(W_c[\Gamma_r * c^{<t-1>},\ x^{<t>}] + b_c)$$

$$\Gamma_u = \sigma(W_u[c^{<t-1>},\ x^{<t>}] + b_u)$$

$$\Gamma_r = \sigma(W_r[c^{<t-1>},\ x^{<t>}] + b_r)$$

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + (1 - \Gamma_u) * c^{<t-1>}$$

- Governing equations of simplified GRU:

$$\tilde{c}^{<t>} = tanh(W_c[c^{<t-1>},\ x^{<t>}] + b_c)$$

$$\Gamma_u = \sigma(W_u[c^{<t-1>},\ x^{<t>}] + b_u)$$

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + (1 - \Gamma_u) * c^{<t-1>}$$

- Γ_r is reset/relevance/regulate gate. It controls how much of the past information, if any, may be passed on to the next stage.

https://www.youtube.com/watch?v=xSCy3q2ts44

Complete model of GRU has two gates

- Governing equations of complete GRU

$$\tilde{c}^{<t>} = tanh(W_c[\Gamma_r * c^{<t-1>},\ x^{<t>}] +\ b_c)$$

$$\Gamma_u = \sigma(W_u[c^{<t-1>},\ x^{<t>}] +\ b_u)$$

$$\Gamma_r = \sigma(W_r[c^{<t-1>},\ x^{<t>}] +\ b_r)$$

$$c^{<t>} = \Gamma_u\ * \tilde{c}^{<t>} + (1 - \Gamma_u)\ * c^{<t-1>}$$

- Governing equations of simplified GRU:

$$\tilde{c}^{<t>} = tanh(W_c[c^{<t-1>},\ x^{<t>}] +\ b_c)$$

$$\Gamma_u = \sigma(W_u[c^{<t-1>},\ x^{<t>}] +\ b_u)$$

$$c^{<t>} = \Gamma_u\ * \tilde{c}^{<t>} + (1 - \Gamma_u)\ * c^{<t-1>}$$

- Γ_r is reset/relevance/regulate gate. It controls how much of the past information, if any, may be passed on to the next stage.

- More expensive than standard RNN but more powerful and robust as well.

https://www.youtube.com/watch?v=xSCy3q2ts44

Long Short Term Memory (LSTM) Cell is older yet more powerful than GRU

Governing equations of complete GRU

$$\tilde{c}^{<t>} = tanh(W_c[\Gamma_r * c^{<t-1>},\ x^{<t>}] + b_c)$$

$$\Gamma_u = \sigma(W_u[c^{<t-1>},\ x^{<t>}] + b_u)$$

$$\Gamma_r = \sigma(W_r[c^{<t-1>},\ x^{<t>}] + b_r)$$

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + (1 - \Gamma_u) * c^{<t-1>}$$

$$a^{<t>} = c^{<t>}$$

Long Short Term Memory (LSTM) Cell is older yet more powerful than GRU

Governing equations of complete GRU

$$\tilde{c}^{<t>} = tanh(W_c[\Gamma_r * c^{<t-1>},\ x^{<t>}] + b_c)$$

$$\Gamma_u = \sigma(W_u[c^{<t-1>},\ x^{<t>}] + b_u)$$

$$\Gamma_r = \sigma(W_r[c^{<t-1>},\ x^{<t>}] + b_r)$$

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + (1 - \Gamma_u) * c^{<t-1>}$$

$$a^{<t>} = c^{<t>}$$

Governing equations of LSTM

$$\tilde{c}^{<t>} = tanh(W_c[a^{<t-1>},\ x^{<t>}] + b_c)$$

Long Short Term Memory (LSTM) Cell is older yet more powerful than GRU

Governing equations of complete GRU

$$\tilde{c}^{<t>} = tanh(W_c[\Gamma_r * c^{<t-1>},\ x^{<t>}] + b_c)$$

$$\Gamma_u = \sigma(W_u[c^{<t-1>},\ x^{<t>}] + b_u)$$

$$\Gamma_r = \sigma(W_r[c^{<t-1>},\ x^{<t>}] + b_r)$$

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + (1 - \Gamma_u) * c^{<t-1>}$$

$$a^{<t>} = c^{<t>}$$

Governing equations of LSTM

$$\tilde{c}^{<t>} = tanh(W_c[a^{<t-1>},\ x^{<t>}] + b_c)$$

$$\Gamma_u = \sigma(W_u[a^{<t-1>},\ x^{<t>}] + b_u)$$

Long Short Term Memory (LSTM) Cell is older yet more powerful than GRU

Governing equations of complete GRU

$$\tilde{c}^{<t>} = tanh(W_c[\Gamma_r * c^{<t-1>},\ x^{<t>}] + b_c)$$

$$\Gamma_u = \sigma(W_u[c^{<t-1>},\ x^{<t>}] + b_u)$$

$$\Gamma_r = \sigma(W_r[c^{<t-1>},\ x^{<t>}] + b_r)$$

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + (1 - \Gamma_u) * c^{<t-1>}$$

$$a^{<t>} = c^{<t>}$$

Governing equations of LSTM

$$\tilde{c}^{<t>} = tanh(W_c[a^{<t-1>},\ x^{<t>}] + b_c)$$

$$\Gamma_u = \sigma(W_u[a^{<t-1>},\ x^{<t>}] + b_u)$$

$$\Gamma_f = \sigma\left(W_f[a^{<t-1>},\ x^{<t>}] + b_f\right)$$

Long Short Term Memory (LSTM) Cell is older yet more powerful than GRU

Governing equations of complete GRU

$$\tilde{c}^{<t>} = tanh(W_c[\Gamma_r * c^{<t-1>},\ x^{<t>}] +\ b_c)$$

$$\Gamma_u = \sigma(W_u[c^{<t-1>},\ x^{<t>}] +\ b_u)$$

$$\Gamma_r = \sigma(W_r[c^{<t-1>},\ x^{<t>}] +\ b_r)$$

$$c^{<t>} = \Gamma_u\ * \tilde{c}^{<t>} + (1 - \Gamma_u)\ * c^{<t-1>}$$

$$a^{<t>} =\ c^{<t>}$$

Governing equations of LSTM

$$\tilde{c}^{<t>} = tanh(W_c[a^{<t-1>},\ x^{<t>}] +\ b_c)$$

$$\Gamma_u = \sigma(W_u[a^{<t-1>},\ x^{<t>}] +\ b_u)$$

$$\Gamma_f = \sigma\left(W_f[a^{<t-1>},\ x^{<t>}] +\ b_f\right)$$

$$\Gamma_o = \sigma(W_o[a^{<t-1>},\ x^{<t>}] +\ b_o)$$

Long Short Term Memory (LSTM) Cell is older yet more powerful than GRU

Governing equations of complete GRU

$$\tilde{c}^{<t>} = tanh(W_c[\Gamma_r * c^{<t-1>},\ x^{<t>}] + b_c)$$

$$\Gamma_u = \sigma(W_u[c^{<t-1>},\ x^{<t>}] + b_u)$$

$$\Gamma_r = \sigma(W_r[c^{<t-1>},\ x^{<t>}] + b_r)$$

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + (1 - \Gamma_u) * c^{<t-1>}$$

$$a^{<t>} = c^{<t>}$$

Governing equations of LSTM

$$\tilde{c}^{<t>} = tanh(W_c[a^{<t-1>},\ x^{<t>}] + b_c)$$

$$\Gamma_u = \sigma(W_u[a^{<t-1>},\ x^{<t>}] + b_u)$$

$$\Gamma_f = \sigma\left(W_f[a^{<t-1>},\ x^{<t>}] + b_f\right)$$

$$\Gamma_o = \sigma(W_o[a^{<t-1>},\ x^{<t>}] + b_o)$$

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + \Gamma_f * c^{<t-1>}$$

Long Short Term Memory (LSTM) Cell is older yet more powerful than GRU

Governing equations of complete GRU

$$\tilde{c}^{<t>} = tanh(W_c[\Gamma_r * c^{<t-1>},\ x^{<t>}] + b_c)$$

$$\Gamma_u = \sigma(W_u[c^{<t-1>},\ x^{<t>}] + b_u)$$

$$\Gamma_r = \sigma(W_r[c^{<t-1>},\ x^{<t>}] + b_r)$$

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + (1 - \Gamma_u) * c^{<t-1>}$$

$$a^{<t>} = c^{<t>}$$

Governing equations of LSTM

$$\tilde{c}^{<t>} = tanh(W_c[a^{<t-1>},\ x^{<t>}] + b_c)$$

$$\Gamma_u = \sigma(W_u[a^{<t-1>},\ x^{<t>}] + b_u)$$

$$\Gamma_f = \sigma\left(W_f[a^{<t-1>},\ x^{<t>}] + b_f\right)$$

$$\Gamma_o = \sigma(W_o[a^{<t-1>},\ x^{<t>}] + b_o)$$

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + \Gamma_f * c^{<t-1>}$$

$$a^{<t>} = \Gamma_o * \tanh(c^{<t>})$$

Long Short Term Memory (LSTM) Cell is older yet more powerful than GRU

Governing equations of complete GRU

$$\tilde{c}^{<t>} = tanh(W_c[\Gamma_r * c^{<t-1>},\ x^{<t>}] + \ b_c)$$

$$\Gamma_u = \sigma(W_u[c^{<t-1>},\ x^{<t>}] + \ b_u)$$

$$\Gamma_r = \sigma(W_r[c^{<t-1>},\ x^{<t>}] + \ b_r)$$

$$c^{<t>} = \Gamma_u \ * \tilde{c}^{<t>} + (1 - \Gamma_u) \ * c^{<t-1>}$$

$$a^{<t>} = \ c^{<t>}$$

Governing equations of LSTM

$$\tilde{c}^{<t>} = tanh(W_c[a^{<t-1>},\ x^{<t>}] + \ b_c)$$

$$\Gamma_u = \sigma(W_u[a^{<t-1>},\ x^{<t>}] + \ b_u)$$

$$\Gamma_f = \sigma\left(W_f[a^{<t-1>},\ x^{<t>}] + \ b_f\right)$$

$$\Gamma_o = \sigma(W_o[a^{<t-1>},\ x^{<t>}] + \ b_o)$$

$$c^{<t>} = \Gamma_u \ * \tilde{c}^{<t>} + \Gamma_f \ * c^{<t-1>}$$

$$a^{<t>} = \ \Gamma_o * \tanh(c^{<t>})$$

- We can add new memory without losing the old memory. (Sentimental, right?)

Long Short Term Memory (LSTM) Cell is older yet more powerful than GRU

Governing equations of complete GRU

$$\tilde{c}^{<t>} = tanh(W_c[\Gamma_r * c^{<t-1>},\ x^{<t>}] + b_c)$$

$$\Gamma_u = \sigma(W_u[c^{<t-1>},\ x^{<t>}] + b_u)$$

$$\Gamma_r = \sigma(W_r[c^{<t-1>},\ x^{<t>}] + b_r)$$

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + (1 - \Gamma_u) * c^{<t-1>}$$

$$a^{<t>} = c^{<t>}$$

Governing equations of LSTM

$$\tilde{c}^{<t>} = tanh(W_c[a^{<t-1>},\ x^{<t>}] + b_c)$$

$$\Gamma_u = \sigma(W_u[a^{<t-1>},\ x^{<t>}] + b_u)$$

$$\Gamma_f = \sigma\left(W_f[a^{<t-1>},\ x^{<t>}] + b_f\right)$$

$$\Gamma_o = \sigma(W_o[a^{<t-1>},\ x^{<t>}] + b_o)$$

Peephole connections

$$, c^{<t-1>}$$

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + \Gamma_f * c^{<t-1>}$$

$$a^{<t>} = \Gamma_o * \tanh(c^{<t>})$$

- We can add new memory without losing the old memory. (Sentimental, right?)

Long Short Term Memory (LSTM) Cell is older yet more powerful than GRU

Governing equations of complete GRU

$$\tilde{c}^{<t>} = tanh(W_c[\Gamma_r * c^{<t-1>},\ x^{<t>}] + b_c)$$

$$\Gamma_u = \sigma(W_u[c^{<t-1>},\ x^{<t>}] + b_u)$$

$$\Gamma_r = \sigma(W_r[c^{<t-1>},\ x^{<t>}] + b_r)$$

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + (1 - \Gamma_u) * c^{<t-1>}$$

$$a^{<t>} = c^{<t>}$$

Which of the two is better?

Governing equations of LSTM

$$\tilde{c}^{<t>} = tanh(W_c[a^{<t-1>},\ x^{<t>}] + b_c)$$

$$\Gamma_u = \sigma(W_u[a^{<t-1>},\ x^{<t>}] + b_u)$$

$$\Gamma_f = \sigma\left(W_f[a^{<t-1>},\ x^{<t>}] + b_f\right)$$

$$\Gamma_o = \sigma(W_o[a^{<t-1>},\ x^{<t>}] + b_o)$$

Peephole connections

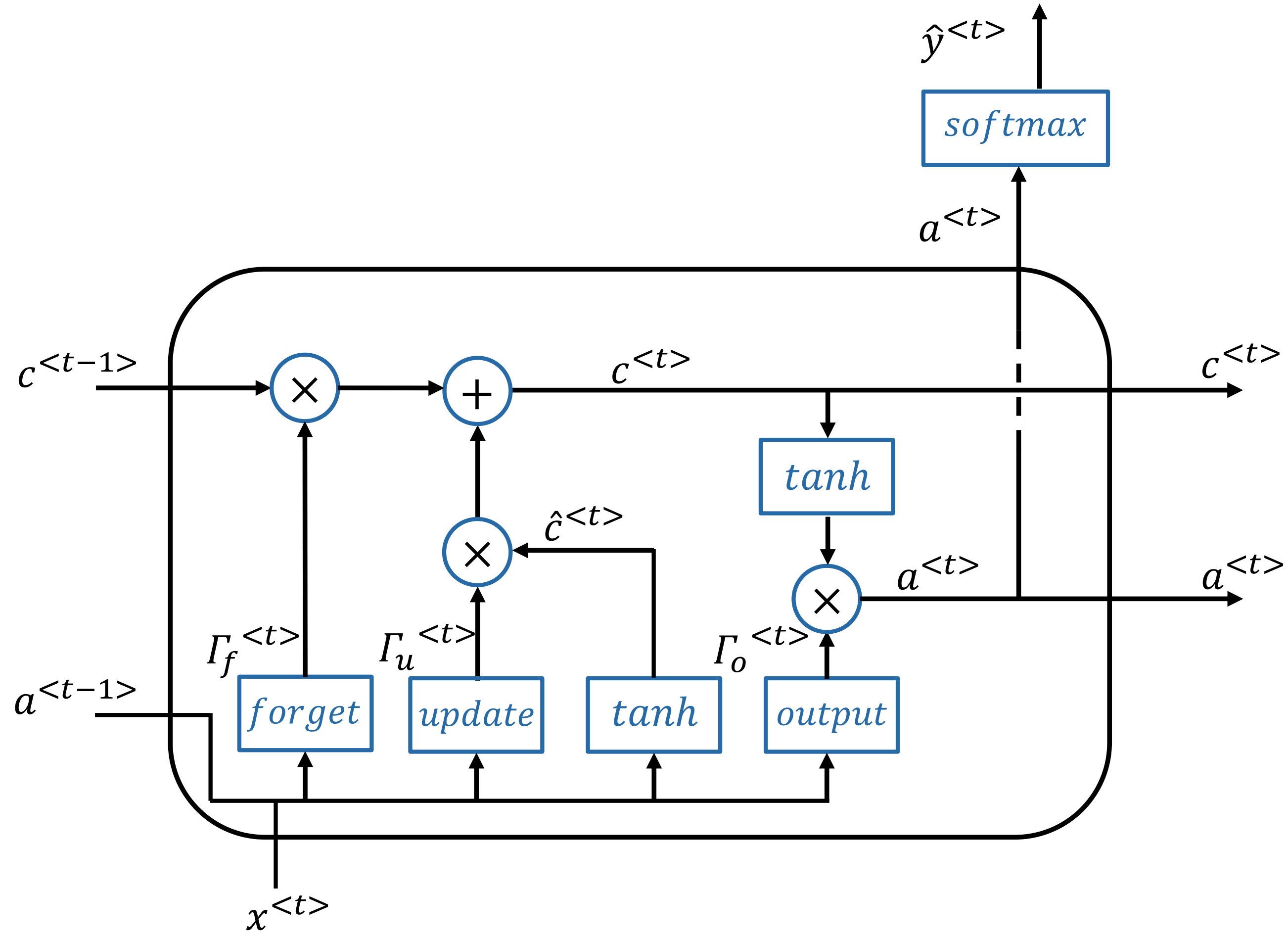
$, c^{<t-1>}$

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + \Gamma_f * c^{<t-1>}$$

$$a^{<t>} = \Gamma_o * \tanh(c^{<t>})$$

- We can add new memory without losing the old memory. (Sentimental, right?)

Schematic Diagram of LSTM



Bidirectional RNNs use past and future information at the same time

In Pakistan, cricket is a

https://www.youtube.com/watch?v=bTXGpATdKRY

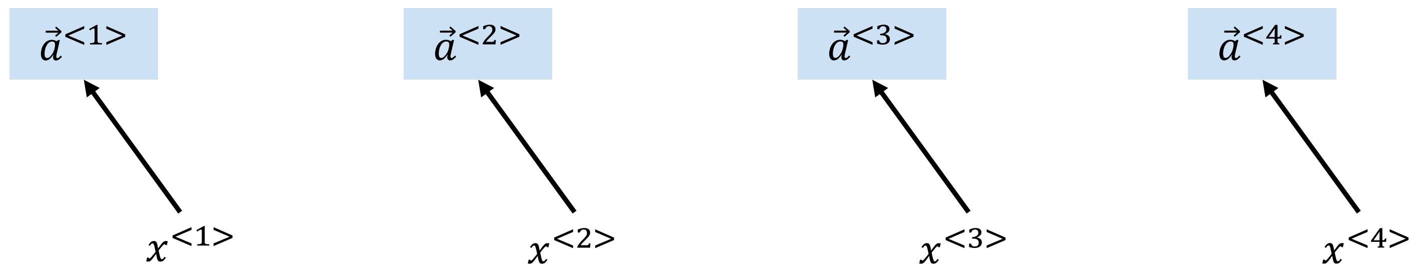
Bidirectional RNNs use past and future information at the same time

In Pakistan, cricket is a common orthopteran insects.
In Pakistan, cricket is a popular sport.

https://www.youtube.com/watch?v=bTXGpATdKRY

Bidirectional RNNs use past and future information at the same time

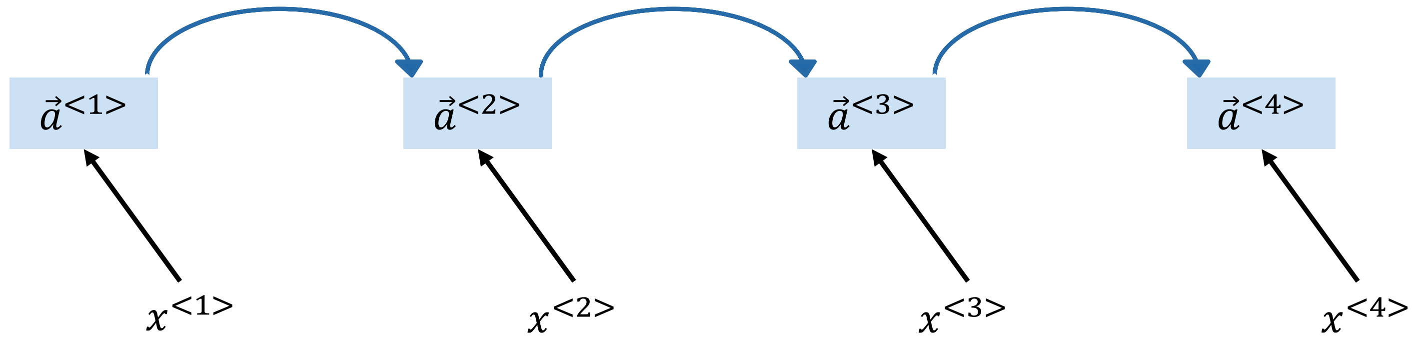
In Pakistan, cricket is a common orthopteran insects.
In Pakistan, cricket is a popular sport.



https://www.youtube.com/watch?v=bTXGpATdKRY

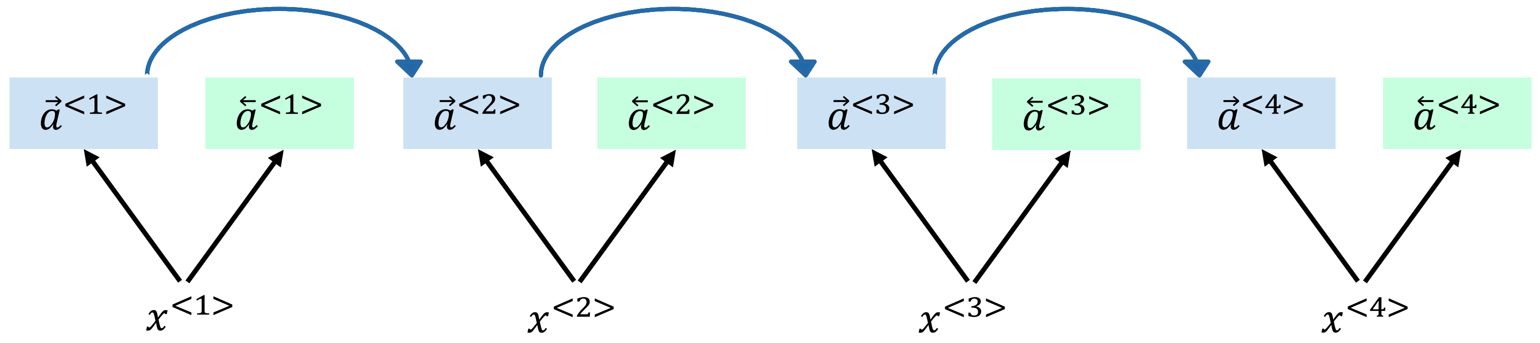
Bidirectional RNNs use past and future information at the same time

In Pakistan, cricket is a common orthopteran insects.
In Pakistan, cricket is a popular sport.



https://www.youtube.com/watch?v=bTXGpATdKRY

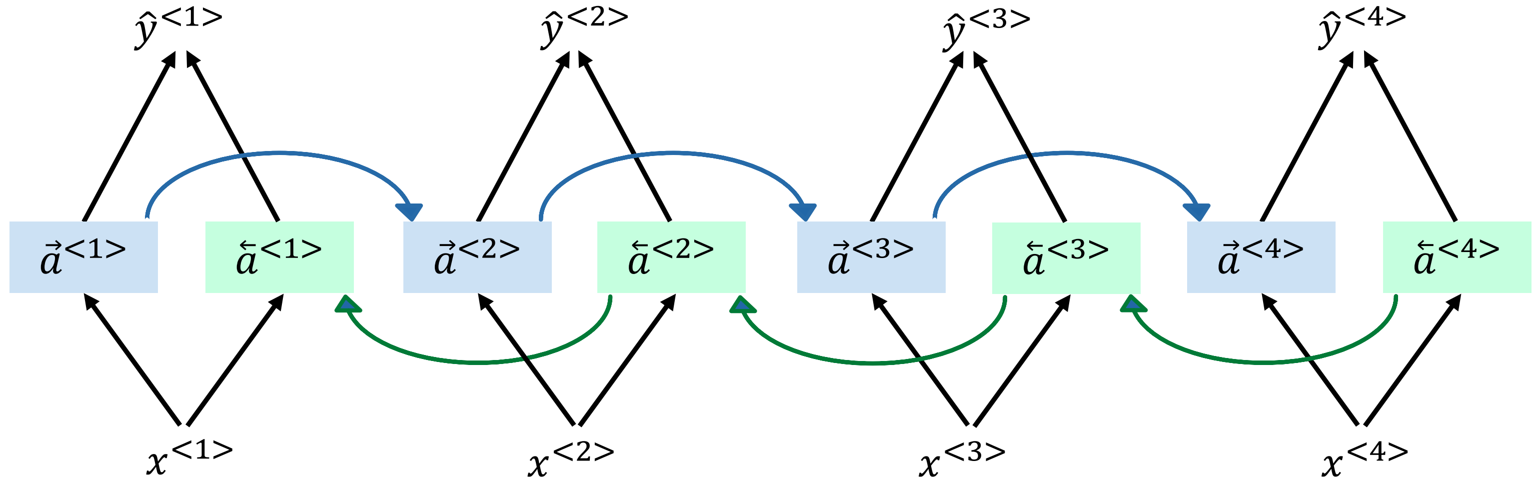
In Pakistan, cricket is a common orthopteran insects.
In Pakistan, cricket is a popular sport.



https://www.youtube.com/watch?v=bTXGpATdKRY

Bidirectional RNNs use past and future information at the same time

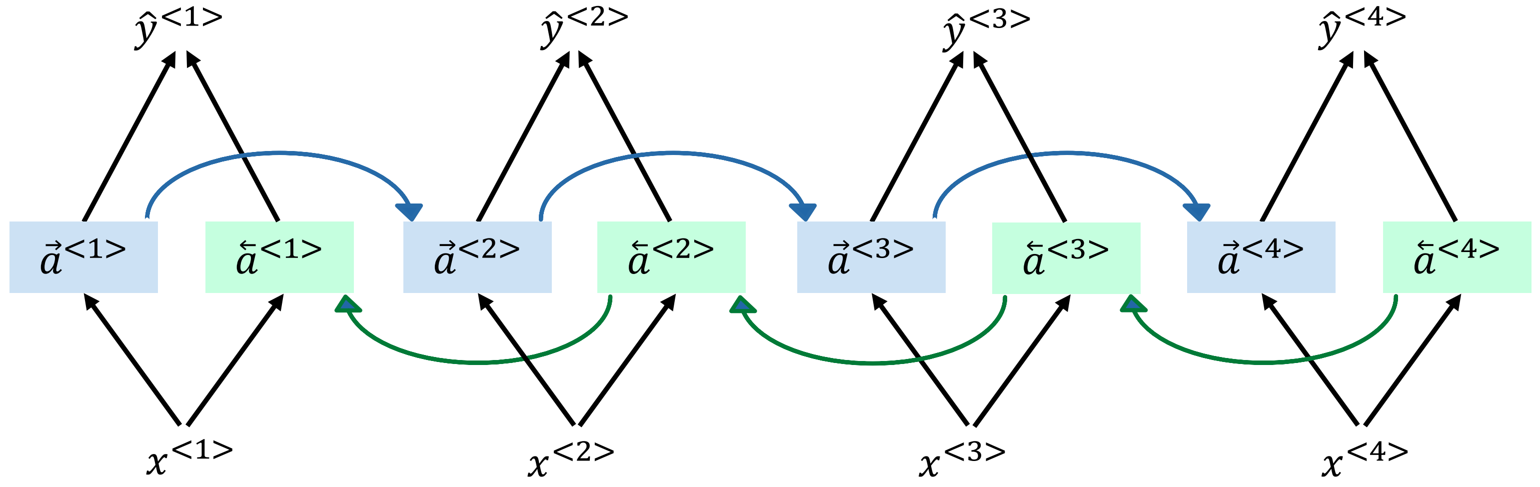
In Pakistan, cricket is a common orthopteran insects.
In Pakistan, cricket is a popular sport.



https://www.youtube.com/watch?v=bTXGpATdKRY

Bidirectional RNNs use past and future information at the same time

In Pakistan, cricket is a common orthopteran insects.
In Pakistan, cricket is a popular sport.

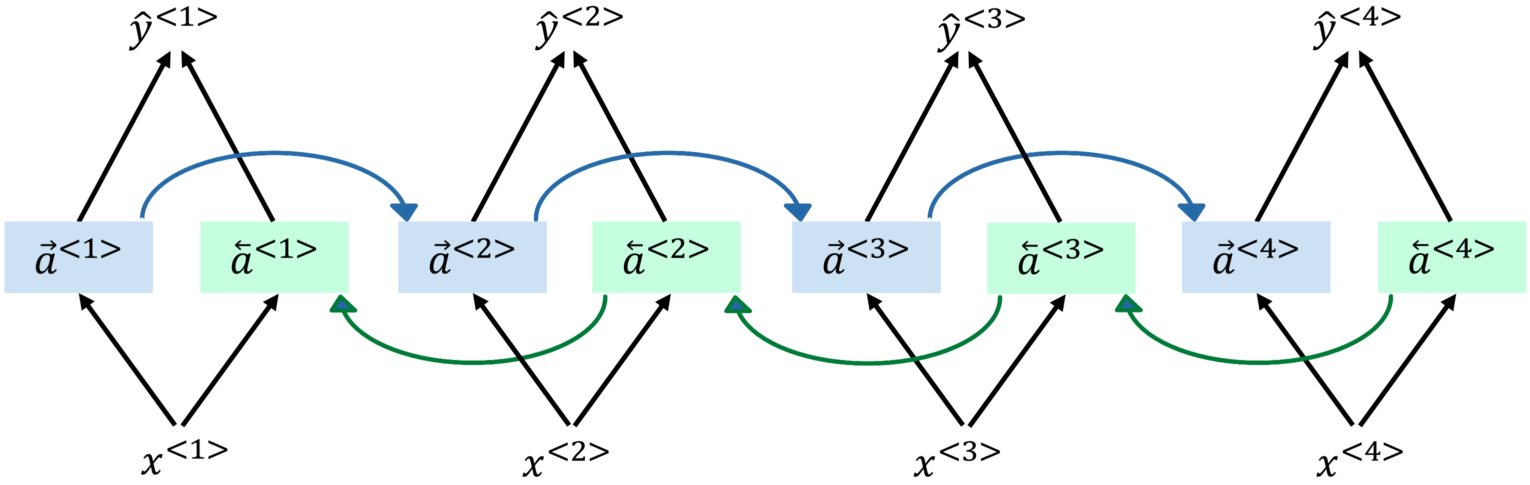


It's an acyclic graph

https://www.youtube.com/watch?v=bTXGpATdKRY

Bidirectional RNNs use past and future information at the same time

In Pakistan, cricket is a common orthopteran insects.
In Pakistan, cricket is a popular sport.



It's an acyclic graph

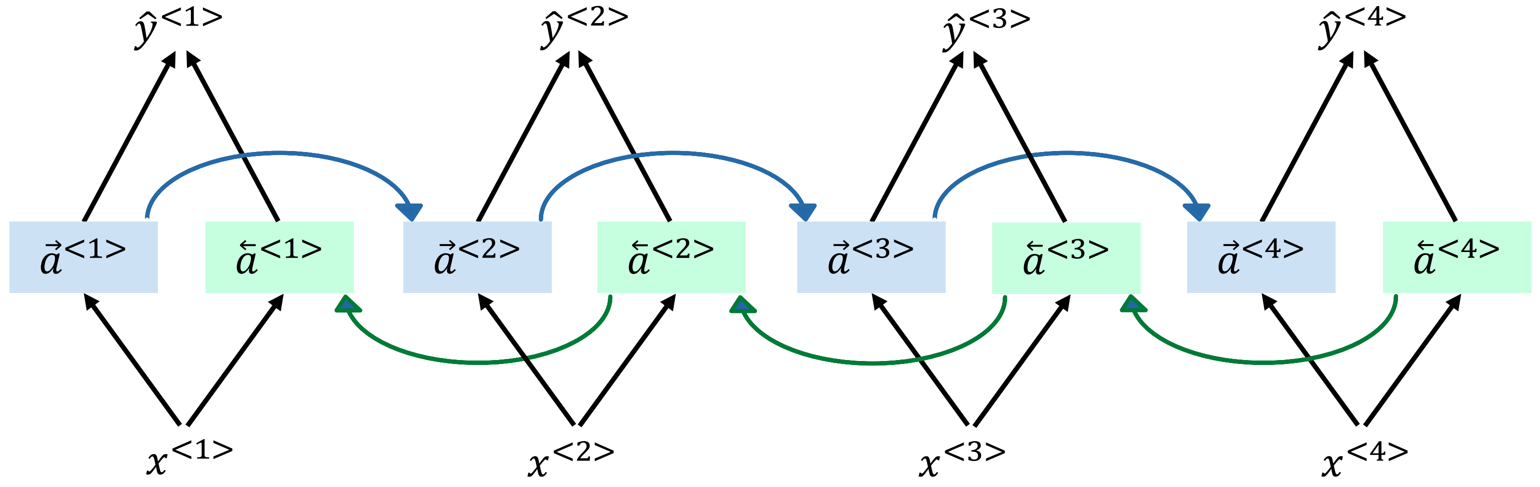
$$\hat{y}^{<t>} = g\left(W_y[\vec{a}^{<1>}, \overleftarrow{a}^{<1>}] + b_y\right)$$

- Forward pass consists of going from left to right and coming back from right to left.

https://www.youtube.com/watch?v=bTXGpATdKRY

Bidirectional RNNs use past and future information at the same time

In Pakistan, cricket is a common orthopteran insects.
In Pakistan, cricket is a popular sport.



It's an acyclic graph

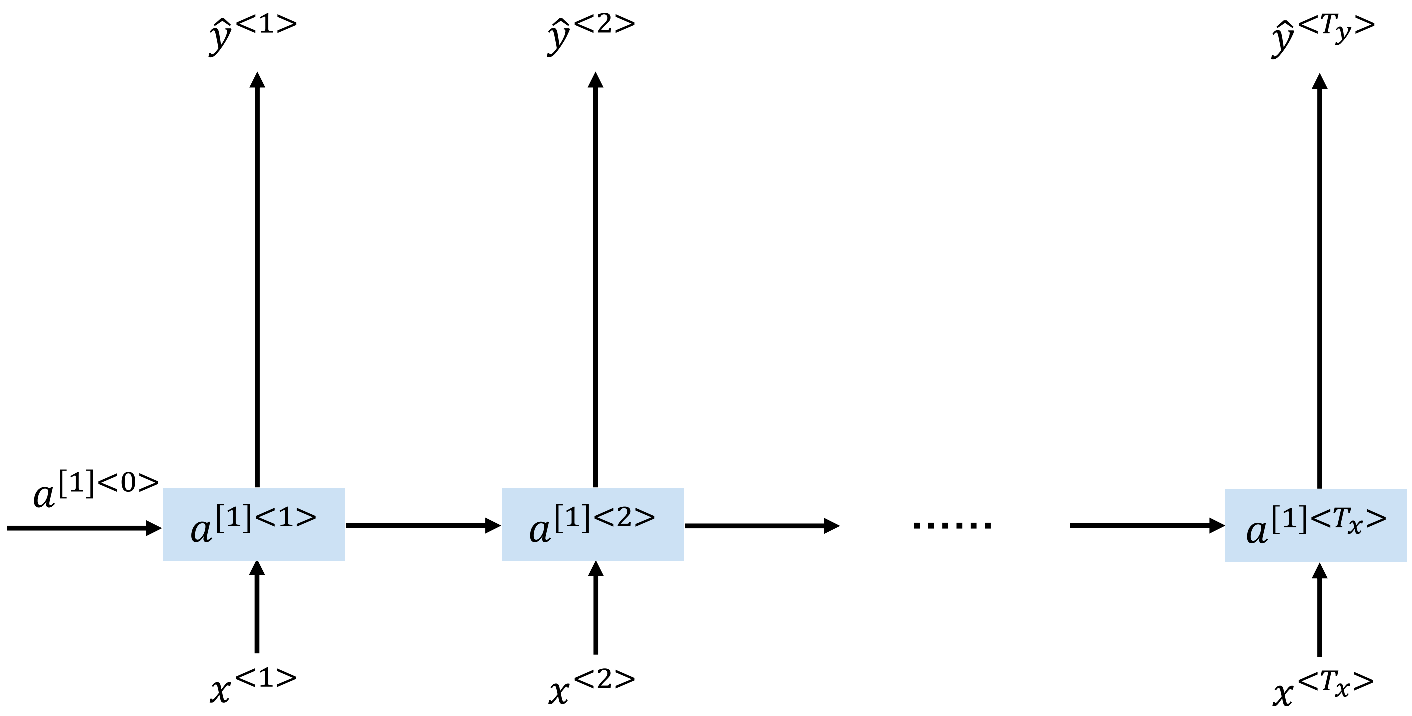
$$\hat{y}^{<t>} = g\left(W_y[\vec{a}^{<1>}, \overleftarrow{a}^{<1>}] + b_y\right)$$

- Forward pass consists of going from left to right and coming back from right to left.

Can we use BRNN for real-time language translation?

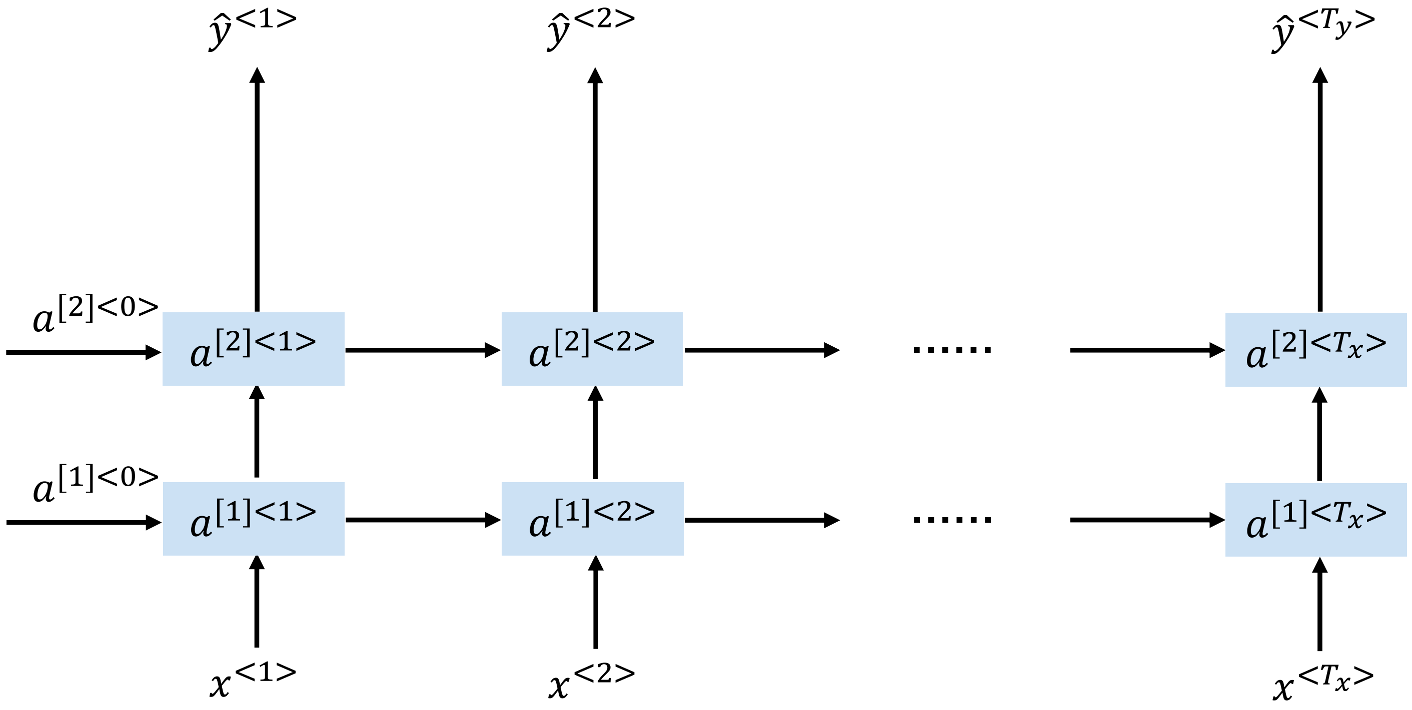
https://www.youtube.com/watch?v=bTXGpATdKRY

Deep RNNs are too computationally expensive



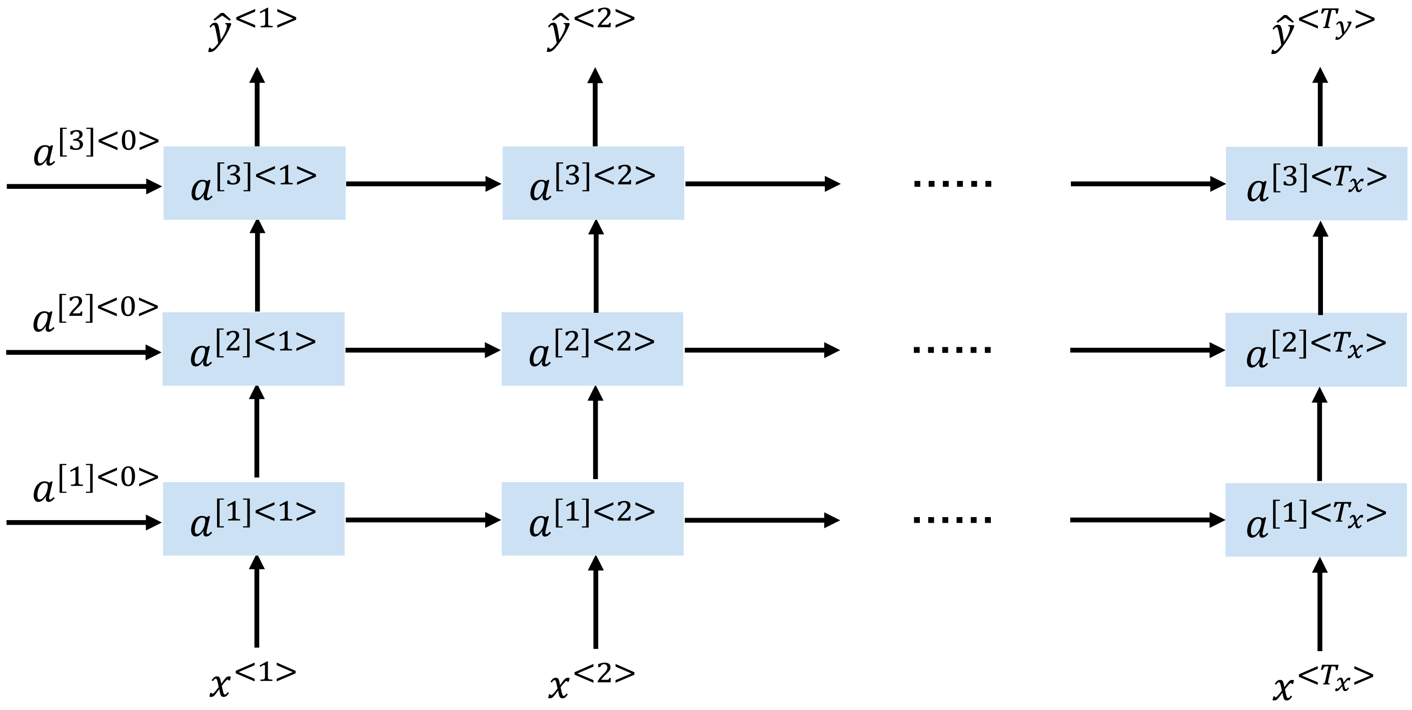
https://www.youtube.com/watch?v=U7wN1x8zsG8

Deep RNNs are too computationally expensive



https://www.youtube.com/watch?v=U7wN1x8zsG8

Deep RNNs are too computationally expensive



$$a^{[l]<t>} = g\left(W_a^l \, [a^{[l]<t-1>}, a^{[l-1]<t>}] + b_a^l\right)$$

https://www.youtube.com/watch?v=U7wN1x8zsG8

Do you have any problem?

Some material (images, tables, text etc.) in this presentation has been borrowed from different books, lecture notes, and the web. The original contents solely belong to their owners, and are used in this presentation only for clarifying various educational concepts. Any copyright infringement is ***not at all*** intended.