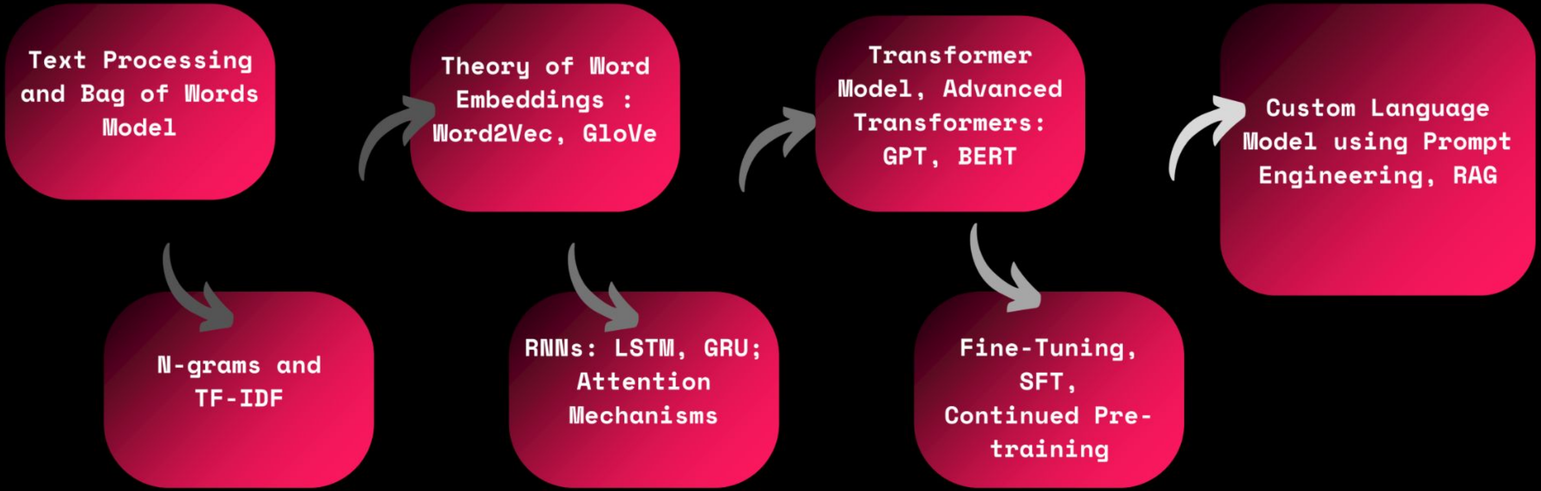
Sequence Modeling

Week #4 - 2 Oct ‘24

NLP Lab

NLP Lab Project : Build your own GPT


Text Processing and Bag of Words Model
N-grams and TF-IDF
Theory of Word Embeddings : Word2Vec, GloVe
RNNs: LSTM, GRU; Attention Mechanisms
Transformer Model, Advanced Transformers: GPT, BERT
Fine-Tuning, SFT, Continued Pre-training
Custom Language Model using Prompt Engineering, RAG

1. Language Models Recap

1.1 Language Models

- Models that assign a probability to each possible next word.
- Language models can also assign a probability to an entire sentence, telling us that the following sequence has a much higher probability of appearing in a text.
- **Goal** : Compute the probability of a sentence or sequence of words.

$$P(W) = P(w_1,w_2,w_3,w_4,w_5 \ldots w_n)$$

1.2 Discriminative vs Generative Models

- **Discriminative model:** a model that calculates the probability of a latent trait given the data.

  $P(Y \mid X)$ (Conditional)

- **Generative model:** a model that calculates the probability of the input data itself.

$P(X)$	$P(X, Y)$
stand-alone	joint

- Discriminative language models are designed to directly learn the boundary between different classes or outputs, typically in tasks involving classification or structured prediction.

1.3 Autoregressive Language Models

$$P(X) = \prod_{i=1}^{I} P(\underbrace{x_i}_{\text{Next Token}} \mid \underbrace{x_1, \ldots, x_{i-1}}_{\text{Context}})$$

- The big problem : How do we predict

$$\boldsymbol{P\ (x_i \mid x_i,\ x_2\ ,\ldots\ldots,\ x_{i-1}\ )}$$

Source : https://phontron.com/class/anlp2024/lectures/#sequence-modeling-jan-25

1.4 Examples of Language Models

- Count-based Language Model
  - Independence assumption: • Count-based maximum-likelihood estimation:

$$P(x_i|x_1, \dots, x_{i-1}) \approx P(x_i)$$

  - Count-based maximum-likelihood estimation.

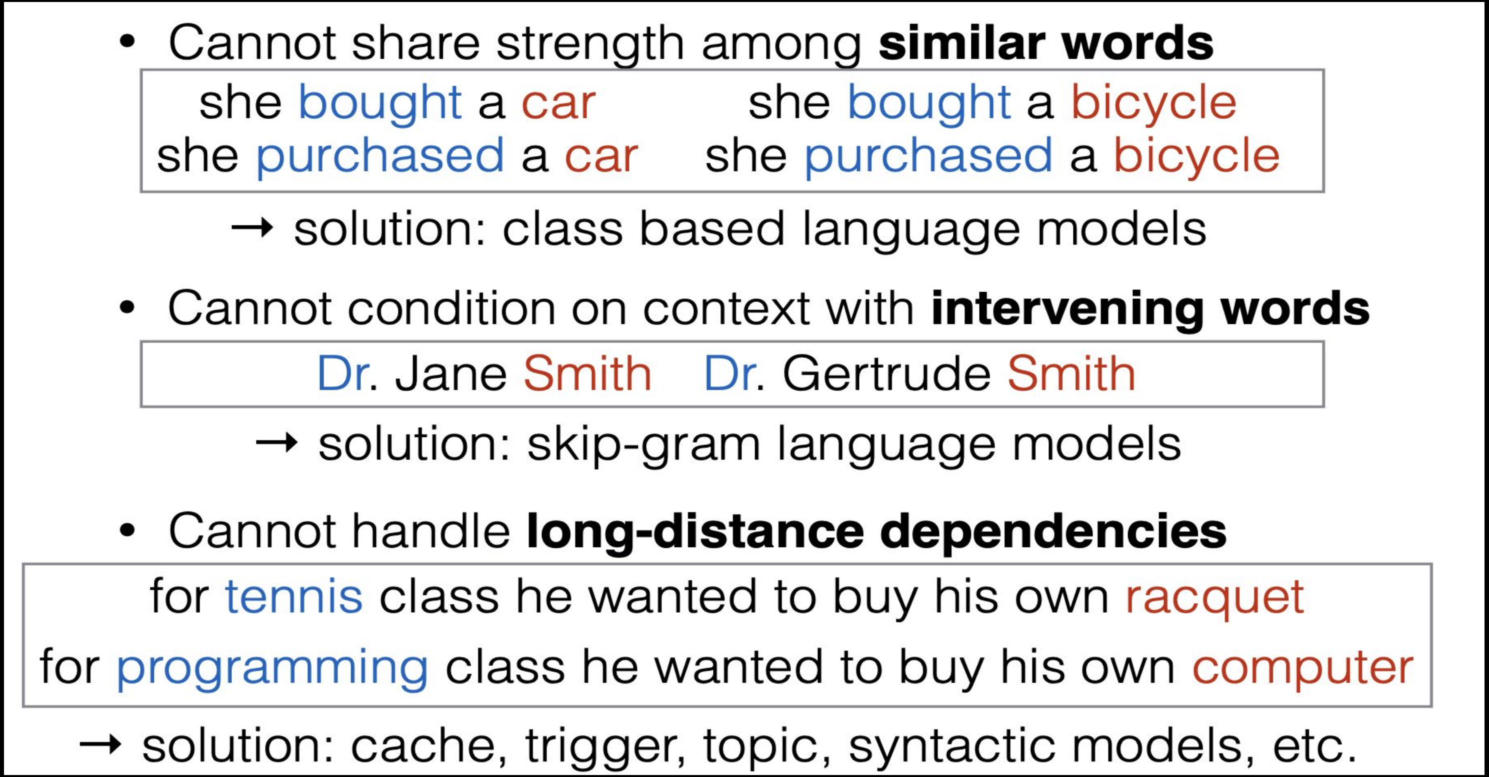
$$P_{\text{MLE}}(x_i) = \frac{c_{\text{train}}(x_i)}{\sum_{\tilde{x}} c_{\text{train}}(\tilde{x})}$$

- Higher-order n-gram Models

$$P_{ML}(x_i \mid x_{i-n+1}, \dots, x_{i-1}) := \frac{c(x_{i-n+1}, \dots, x_i)}{c(x_{i-n+1}, \dots, x_{i-1})}$$

- Featurized Log-Linear Models

1.5 Problems & Solutions



- Cannot share strength among **similar words**

she bought a car   she bought a bicycle
she purchased a car   she purchased a bicycle

→ solution: class based language models

- Cannot condition on context with **intervening words**

Dr. Jane Smith   Dr. Gertrude Smith

→ solution: skip-gram language models

- Cannot handle **long-distance dependencies**

for tennis class he wanted to buy his own racquet
for programming class he wanted to buy his own computer

→ solution: cache, trigger, topic, syntactic models, etc.

Source : https://phontron.com/class/anlp2024/lectures/#sequence-modeling-jan-25

2. Sequence Modeling

2.1 NLP and Sequential Data

- NLP is full of sequential data :
  - Characters in words
  - Words in sentences
  - Sentences in discourse
  - ……

Source : https://phontron.com/class/anlp2024/lectures/#sequence-modeling-jan-25

2.2 Long-distance Dependencies in Language

- Agreement in number, gender, etc.

  He does not have very much confidence in **himself**.

  She does not have very much confidence in **herself**.

- Selectional preference.

  The **reign** has lasted as long as the life of the **queen**.

  The **rain** has lasted as long as the life of the **clouds**.

Source : https://phontron.com/class/anlp2024/lectures/#sequence-modeling-jan-25

2.3 Can be Complicated !

- What is the referent of “**it**”?

The trophy would not fit in the brown suitcase because it was too **big**.

Trophy

The trophy would not fit in the brown suitcase because it was too **small**.

Suitcase

Source : Winograd Schema Challenge: http://commonsensereasoning.org/winograd.html)

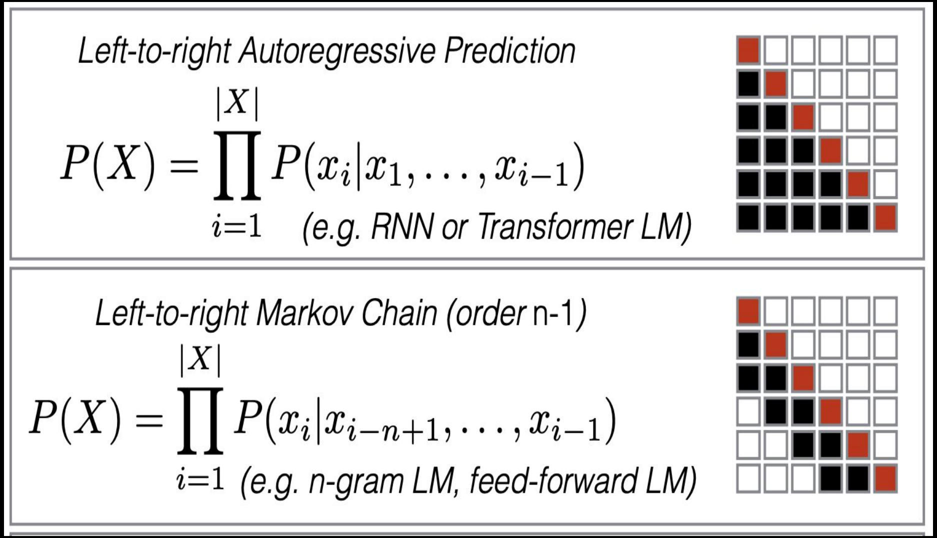
2.4 Types of Sequential Prediction

<u>Unconditioned vs Conditioned</u>

- **Unconditioned Prediction:** Predict the probability of a single variable $P(X)$.
- **Conditioned Prediction:** Predict the probability of an output variable given an input $P(Y \mid X)$.

Source : Winograd Schema Challenge: http://commonsensereasoning.org/winograd.html)

2.4 Types of Unconditioned Prediction



Left-to-right Autoregressive Prediction

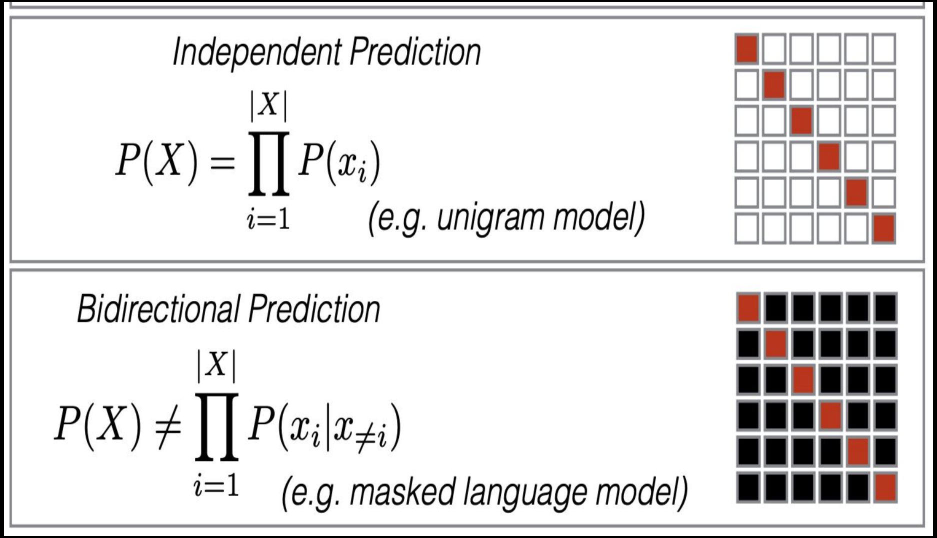
$$P(X) = \prod_{i=1}^{|X|} P(x_i|x_1, \ldots, x_{i-1})$$

(e.g. RNN or Transformer LM)

Left-to-right Markov Chain (order n-1*)*

$$P(X) = \prod_{i=1}^{|X|} P(x_i|x_{i-n+1}, \ldots, x_{i-1})$$

(e.g. n-gram LM, feed-forward LM)



Independent Prediction

$$P(X) = \prod_{i=1}^{|X|} P(x_i)$$

(e.g. unigram model)

Bidirectional Prediction

$$P(X) \neq \prod_{i=1}^{|X|} P(x_i|x_{\neq i})$$

(e.g. masked language model)

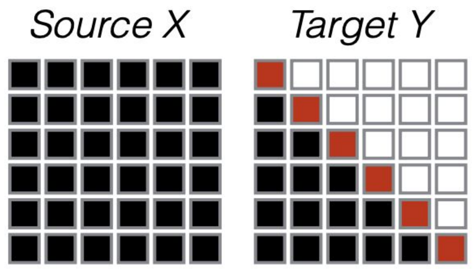
Source : https://phontron.com/class/anlp2024/assets/slides/anlp-04-seqmod.pdf

2.4 Types of Conditioned Prediction

Autoregressive Conditioned Prediction

$$P(Y|X) = \prod_{i=1}^{|Y|} P(y_i|X, y_1, \ldots, y_{i-1})$$

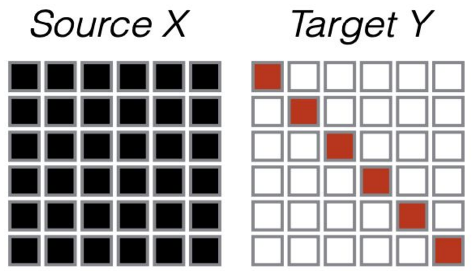
(e.g. seq2seq model)



Non-autoregressive Conditioned Prediction

$$P(Y|X) = \prod_{i=1}^{|Y|} P(y_i|X)$$

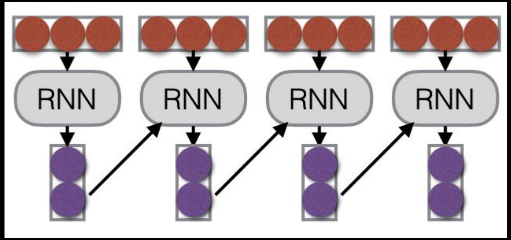
(e.g. sequence labeling, non-autoregressive MT)



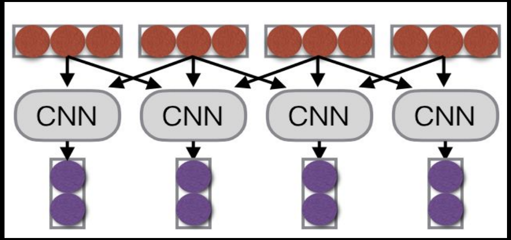
Source : https://phontron.com/class/anlp2024/assets/slides/anlp-04-seqmod.pdf

2.5 Three Major Types of Sequence Models

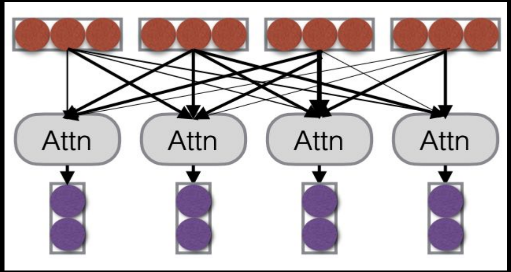
- **Recurrence**: Condition representations on an encoding of the history.



- **Convolution**: Condition representations on local context.



- **Attention**: Condition representations on a weighted average of all tokens

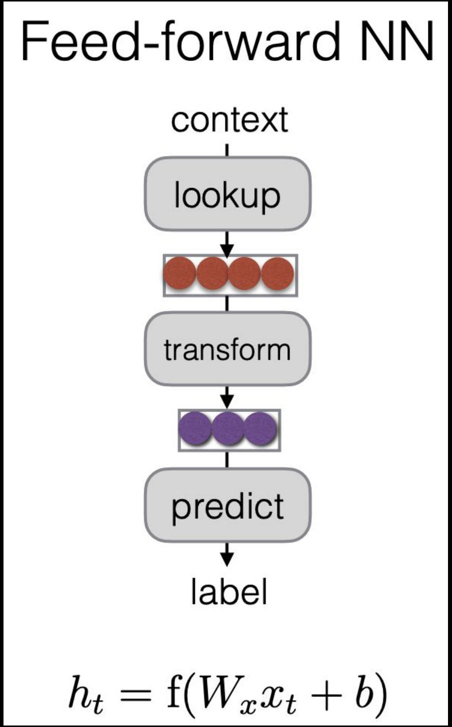


Source : https://phontron.com/class/anlp2024/assets/slides/anlp-04-seqmod.pdf

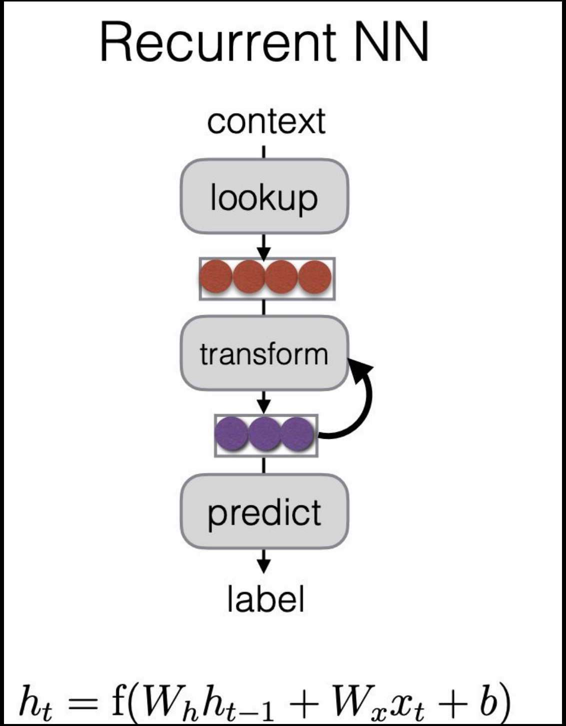
3. Recurrent Neural Networks

3.1 Recurrent Neural Networks

- Tools to “remember” information.

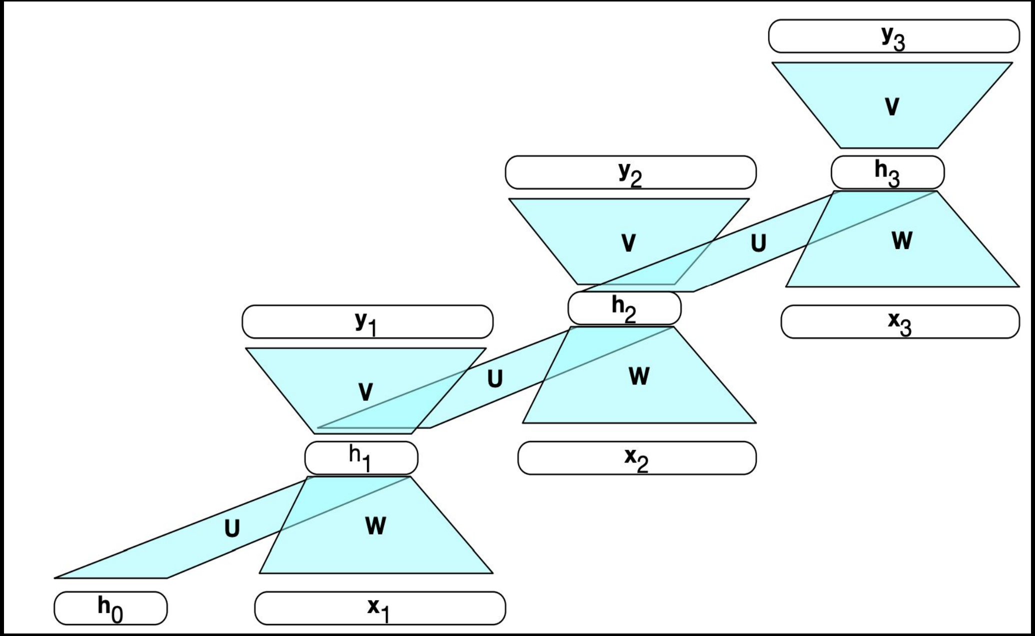


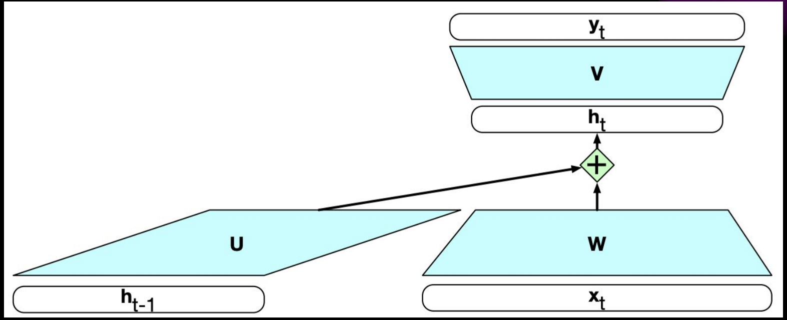
$$h_t = \mathrm{f}(W_x x_t + b)$$

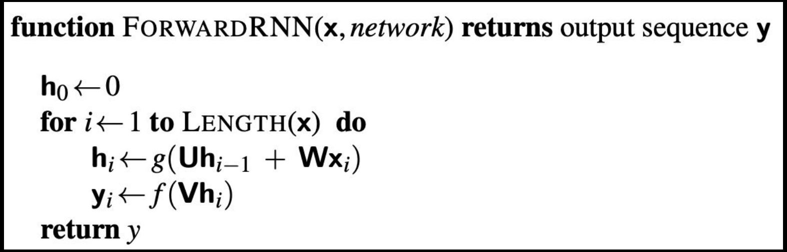


$$h_t = \mathrm{f}(W_h h_{t-1} + W_x x_t + b)$$

3.2 Unrolling in Time





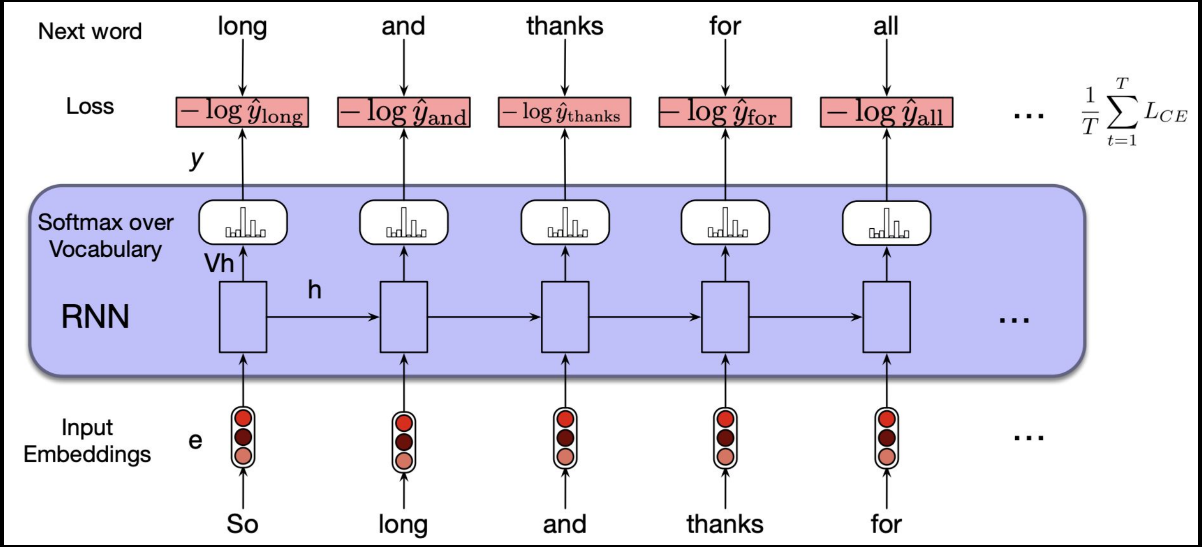


```
function FORWARDRNN($\mathbf{x}$, network) returns output sequence $\mathbf{y}$

  $\mathbf{h}_0 \leftarrow 0$
  for $i \leftarrow 1$ to LENGTH($\mathbf{x}$) do
      $\mathbf{h}_i \leftarrow g(\mathbf{U}\mathbf{h}_{i-1} + \mathbf{W}\mathbf{x}_i)$
      $\mathbf{y}_i \leftarrow f(\mathbf{V}\mathbf{h}_i)$
  return $y$
```

Source : https://web.stanford.edu/~jurafsky/slp3/8.pdf

3.3 Training RNN Language Model

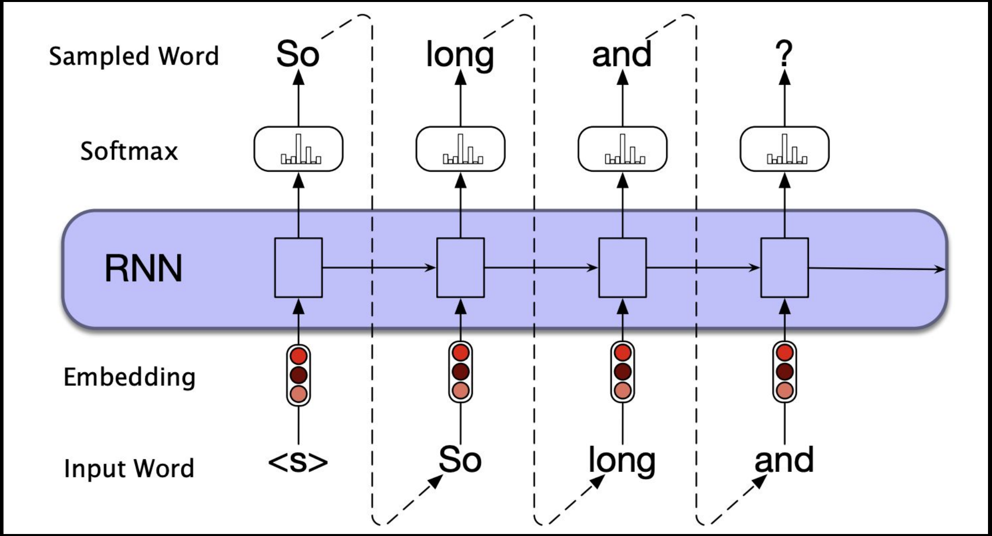


Next word
long
and
thanks
for
all
Loss
$-\log \hat{y}_{\text{long}}$
$-\log \hat{y}_{\text{and}}$
$-\log \hat{y}_{\text{thanks}}$
$-\log \hat{y}_{\text{for}}$
$-\log \hat{y}_{\text{all}}$
...
$\frac{1}{T}\sum_{t=1}^{T} L_{CE}$
y
Softmax over Vocabulary
Vh
h
RNN
...
Input Embeddings
e
...
So
long
and
thanks
for

- **Self-Supervision** : the natural sequence of words is its own supervision!
- **Teacher Forcing** : model is always given the correct history sequence to predict the next word (rather than feeding the model its best case from the previous time step).

Source : https://web.stanford.edu/~jurafsky/slp3/8.pdf

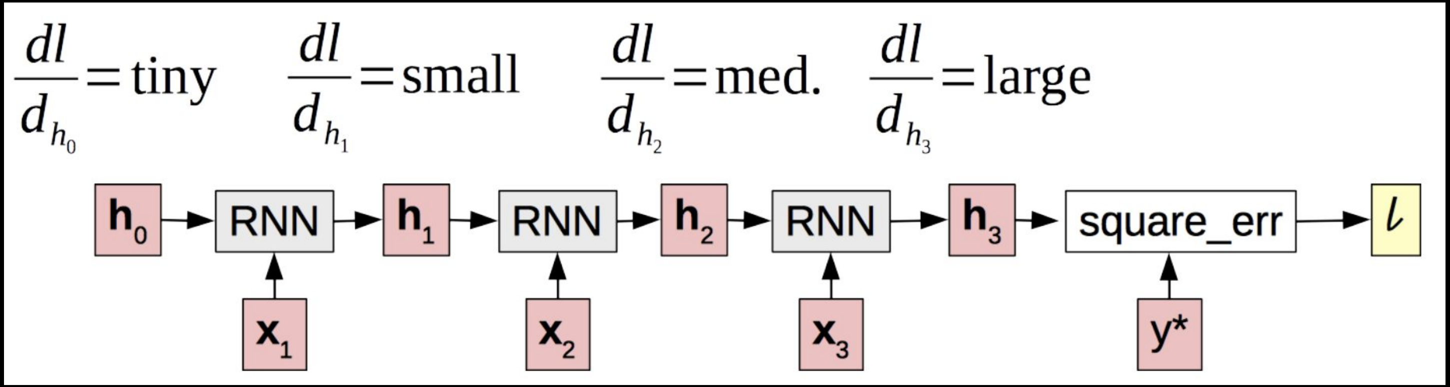
3.4 Generation with RNN-Based Language Models



- **Autoregressive Generation** : Sampling words conditioned on previous choices until a predetermined length is reached, or an end of sequence token is generated.
- The key to these approaches is to prime the generation component with an appropriate context. That is, instead of simply using <s> to get things started we can provide a richer task-appropriate context.

3.5 Vanishing Gradient

- Gradients decrease as they get pushed back.

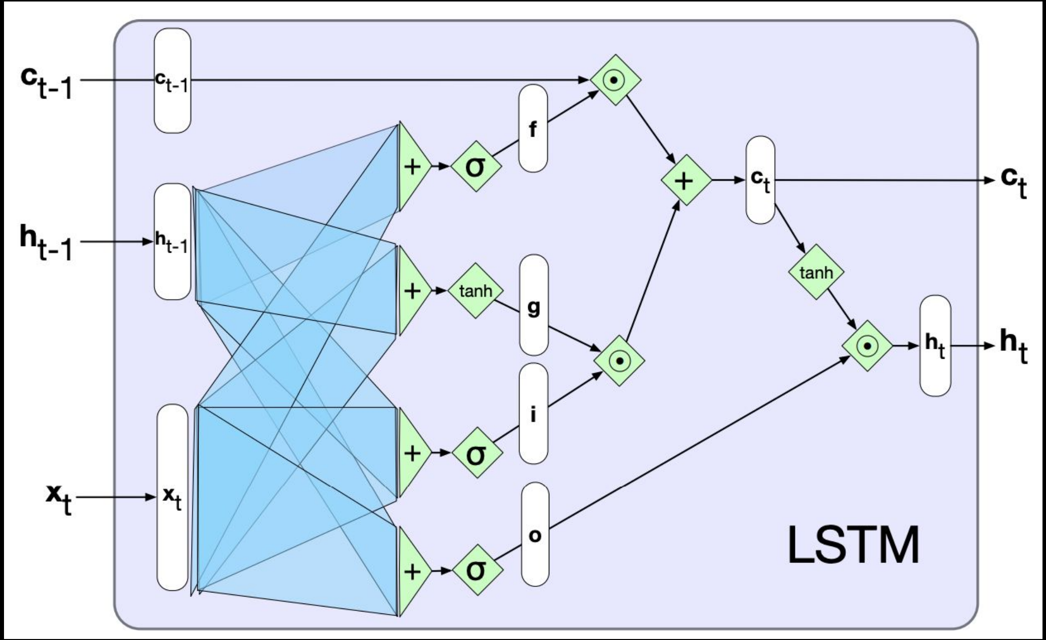

$\frac{dl}{d_{h_0}} = \text{tiny}$
$\frac{dl}{d_{h_1}} = \text{small}$
$\frac{dl}{d_{h_2}} = \text{med.}$
$\frac{dl}{d_{h_3}} = \text{large}$
$\mathbf{h}_0$
RNN
$\mathbf{h}_1$
RNN
$\mathbf{h}_2$
RNN
$\mathbf{h}_3$
square_err
l
$\mathbf{x}_1$
$\mathbf{x}_2$
$\mathbf{x}_3$
y*

- Why? “Squashed” by non-linearities or small weights in matrices.

Source : https://phontron.com/class/anlp2024/assets/slides/anlp-04-seqmod.pdf

4. Long Short Term Memory

4.1 Basic Idea of LSTM



- **Basic idea:** make additive connections between time steps.
- Gates control the flow of the information and the gradients.

Source : https://web.stanford.edu/~jurafsky/slp3/8.pdf

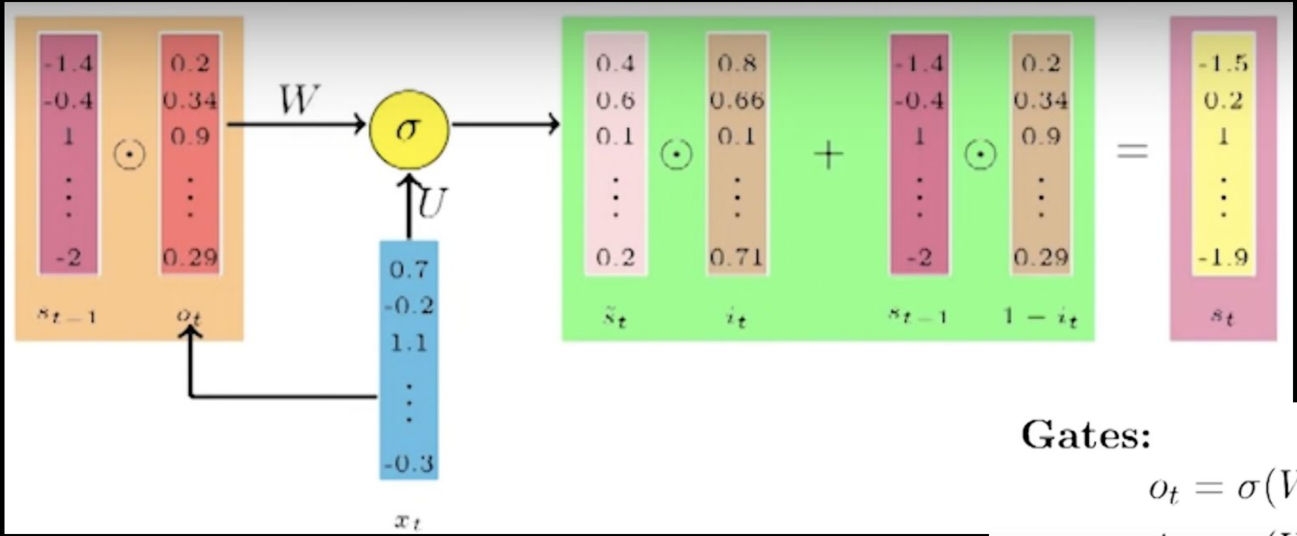
4.1 Equations of Gates in LSTM

- Forget Gate : $\mathbf{f}_t = \sigma(\mathbf{U}_f \mathbf{h}_{t-1} + \mathbf{W}_f \mathbf{x}_t)$
- Update Gate : $\mathbf{g}_t = \tanh(\mathbf{U}_g \mathbf{h}_{t-1} + \mathbf{W}_g \mathbf{x}_t)$
- Input/Add Gate : $\mathbf{i}_t = \sigma(\mathbf{U}_i \mathbf{h}_{t-1} + \mathbf{W}_i \mathbf{x}_t)$
- Output Gate : $\mathbf{o}_t = \sigma(\mathbf{U}_o \mathbf{h}_{t-1} + \mathbf{W}_o \mathbf{x}_t)$

Source : https://web.stanford.edu/~jurafsky/slp3/8.pdf

5. Gated Recurrent Unit

4.1 GRU Architecture



Gates:

$$o_t = \sigma(W_o s_{t-1} + U_o x_t + b_o)$$
$$i_t = \sigma(W_i s_{t-1} + U_i x_t + b_i)$$

States:

$$\tilde{s}_t = \sigma(W(o_t \odot s_{t-1}) + U x_t + b)$$
$$s_t = (1 - i_t) \odot s_{t-1} + i_t \odot \tilde{s}_t$$

Source : https://archive.nptel.ac.in/courses/106/106/106106184/

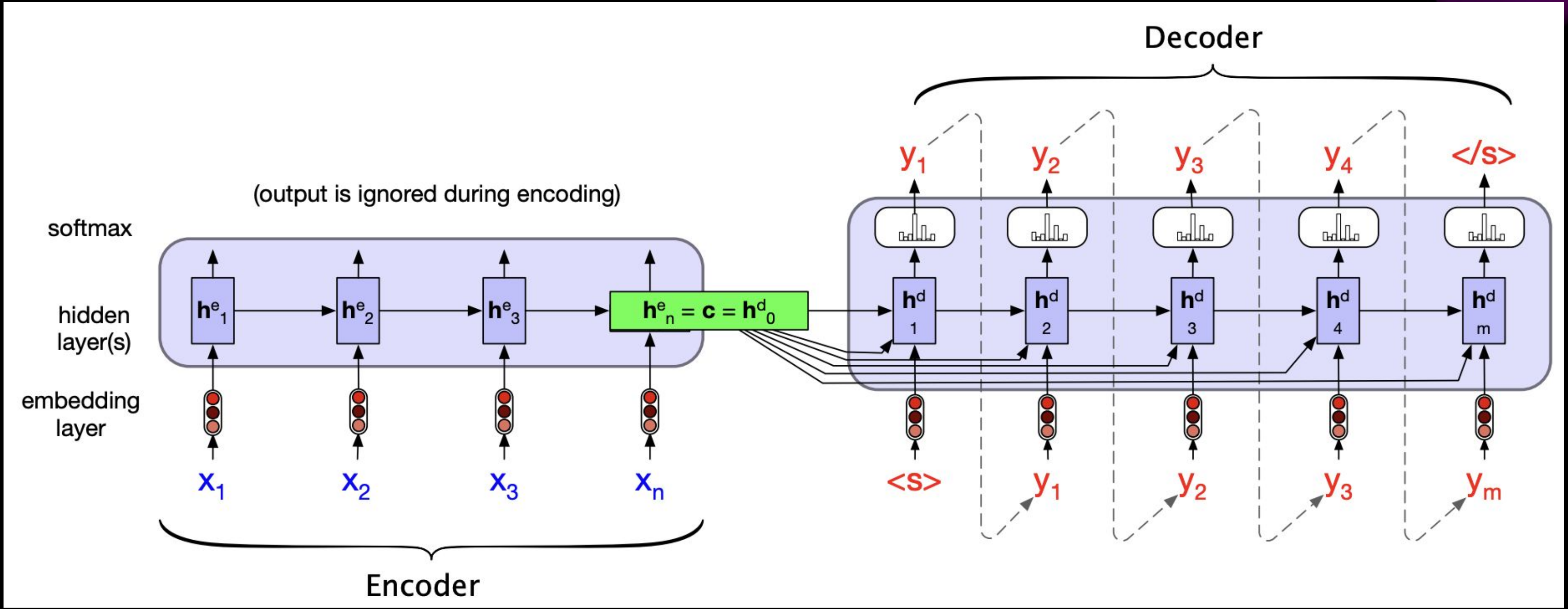
6. Encoder-Decoder

6.1 What is Encoder-Decoder ?

- Encoder-decoder networks, sometimes called sequence-to-sequence networks, encoder-decoder are models capable of generating contextually appropriate, arbitrary length, output sequences given an input sequence.
- The key idea underlying these networks is the use of an encoder network that takes an input sequence and creates a contextualized representation of it, often called the context.
- This representation is then passed to a decoder which generates a taskspecific output sequence.

Source : https://web.stanford.edu/~jurafsky/slp3/8.pdf

6.2 Encoder-Decoder Models with RNN



Source : https://web.stanford.edu/~jurafsky/slp3/8.pdf

7. Attention

7.1 Basic Idea of Attention

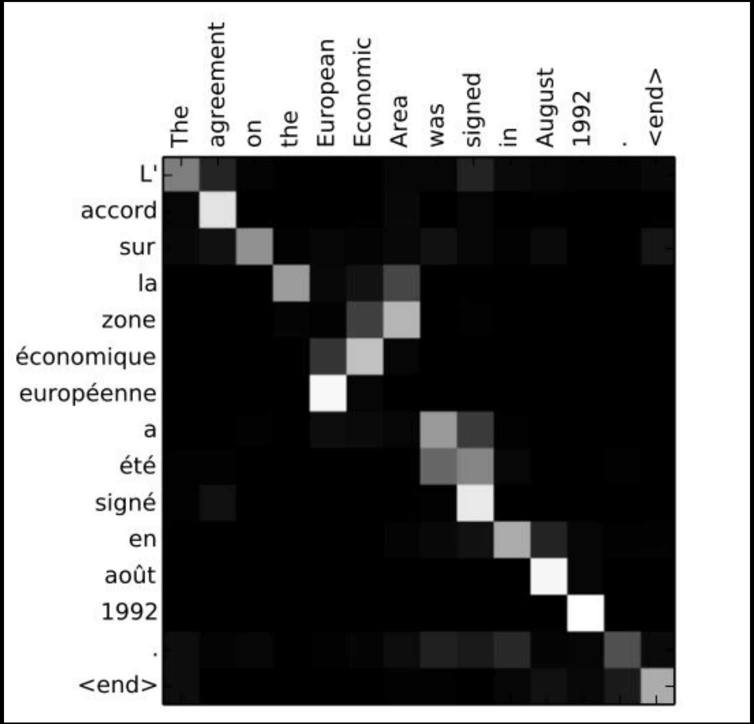
- A mechanism that helps models focus on relevant parts of the input when making predictions.

- Overcomes limitations of traditional sequence models (like RNNs) that struggle with long dependencies.

- Two main types of Attention :
    - Cross-Attention
    - Self-Attention

7.2 Cross Attention

- Each element in a sequence attends to elements of another sequence.



7.3 Cross Attention : Bahdanau et al. 2015

- Encoder-Decoder Model:
  - The encoder processes the input sequence and generates hidden states.
  - The attention mechanism computes a weighted sum of these hidden states to form a context vector.
- Key Components:
  - Encoder Hidden States (h_1, h_2, ..., h□): Represent the input sequence.
  - Decoder Hidden State (s□): Represents the decoder's state at time step t.
  - Alignment Score (e□$_i$): Measures the relevance between the decoder state at time t and the encoder hidden state h_i.
- Attention Weights (α□$_i$):
  - Computed using the alignment scores.
  - Determine how much "attention" should be paid to each input word.

7.3 Bahdanau Attention : Step-by-Step

- Alignment Scores:
  - Compute alignment scores between the current decoder hidden state and each encoder hidden state.
  - The alignment model is parametrized as a feedforward neural network and jointly trained with the remaining system components.

$$e_{ti} = v^T \tanh(W_1 s_{t-1} + W_2 h_i)$$

- Softmax Over Alignment Scores:
  - These attention weights tell how much the model should focus on each encoder hidden state.
- Context Vector (c□):
  - Compute a weighted sum of the encoder hidden states based on the attention weights

$$c_t = \sum_{i=1}^{n} \alpha_{ti} h_i$$

7.4 Luong Attention Mechanism

- Encoder Hidden States (h_1, h_2, ..., h□): Represent the input sequence.
- Decoder Hidden State (s□): Represents the decoder's state at time step t.
- Alignment Scores:
  - Compute alignment scores by taking the dot product of the decoder hidden state s□ and each encoder hidden state h_i.

$$e_{ti} = s_t \cdot h_i$$

- Attention Weights (α□$_i$):
  - Apply softmax to the alignment scores to get attention weights
- Context Vector (c□):
  - Compute a weighted sum of the encoder hidden states based on the attention weights.

$$c_t = \sum_{i=1}^{n} \alpha_{ti} h_i$$

Test you knowledge!