aivancity
SCHOOL FOR
TECHNOLOGY, BUSINESS & SOCIETY
PARIS-CACHAN

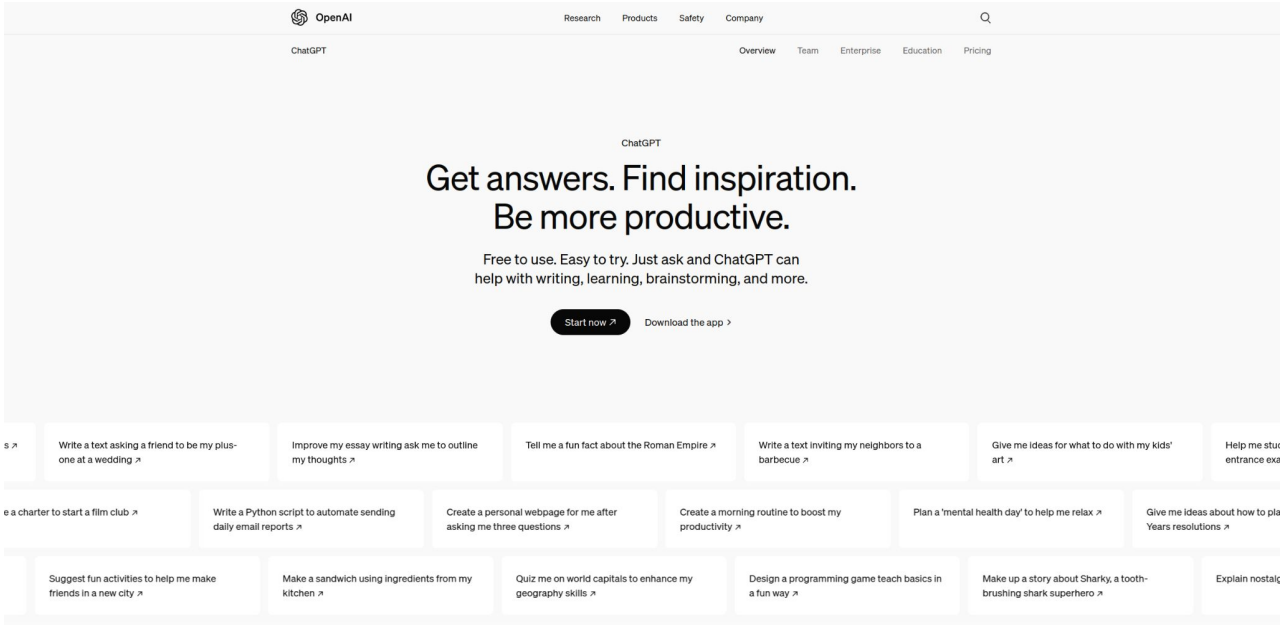
10/11/2025

Natural Language Processing (NLP)

LLM Architectures: Recurrent Neural Networks

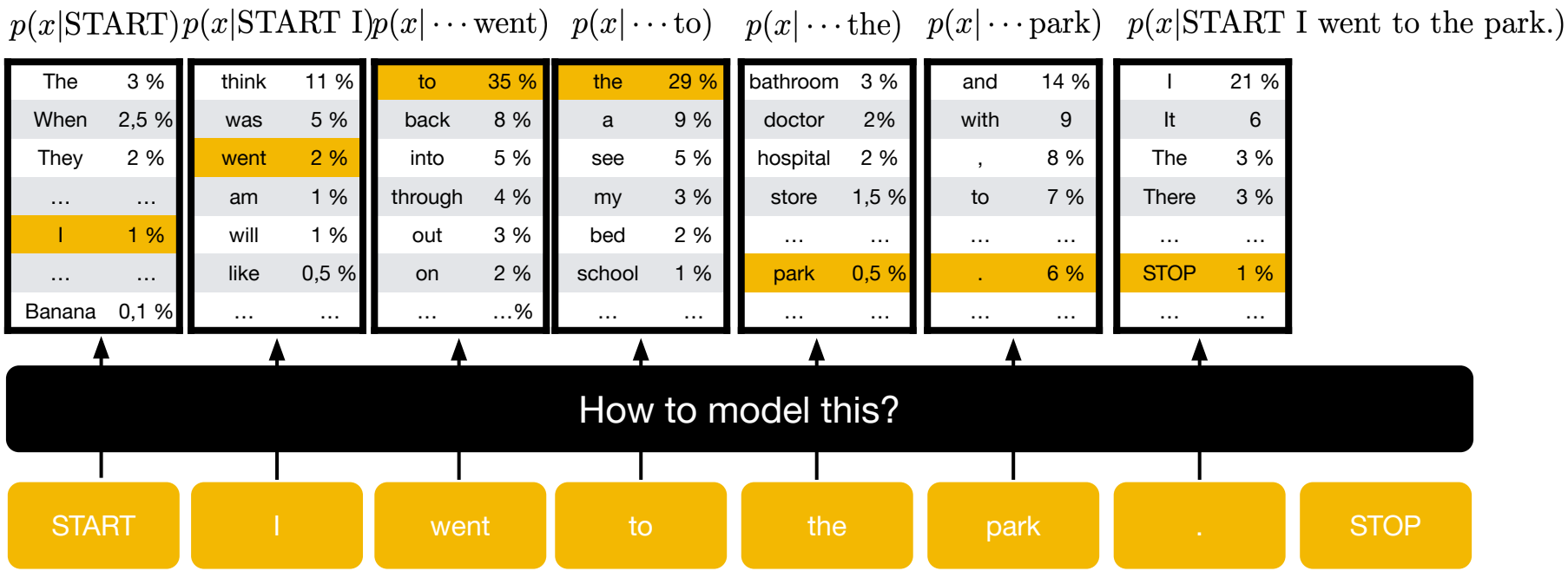
Part 1

Chatbots like ChatGPT rely on LLMs



Part 1

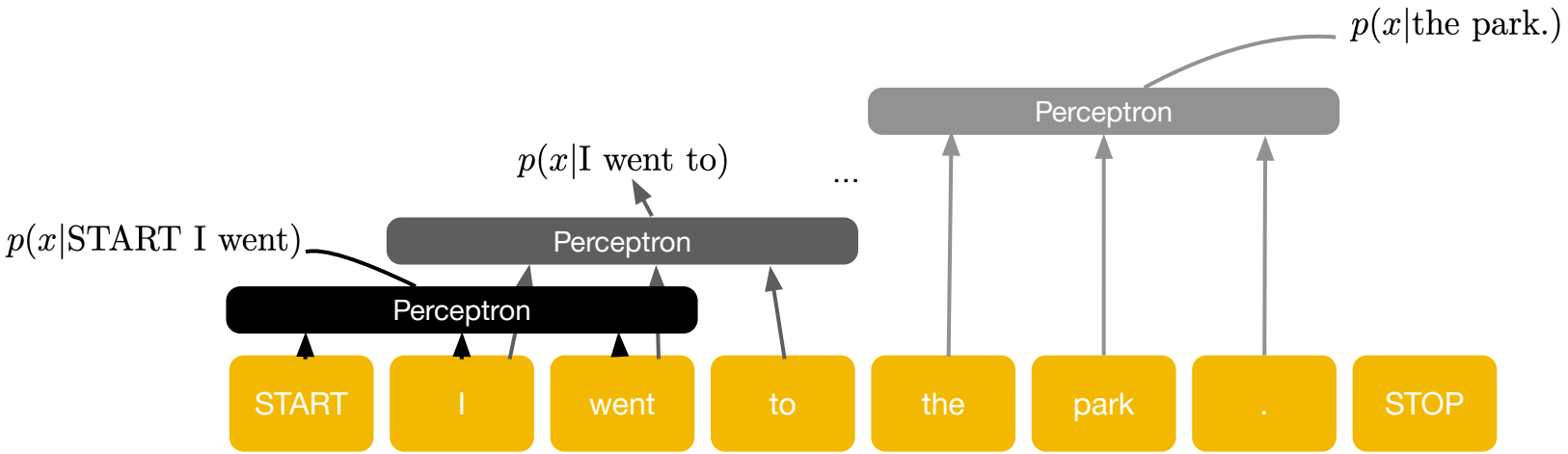
Language Modeling



Part 1

v0: sliding window

- Sliding window of size N, like CBOW or n-grams:
  (N − 1)th-order Markov assumption (prediction only depends on N-1 last)

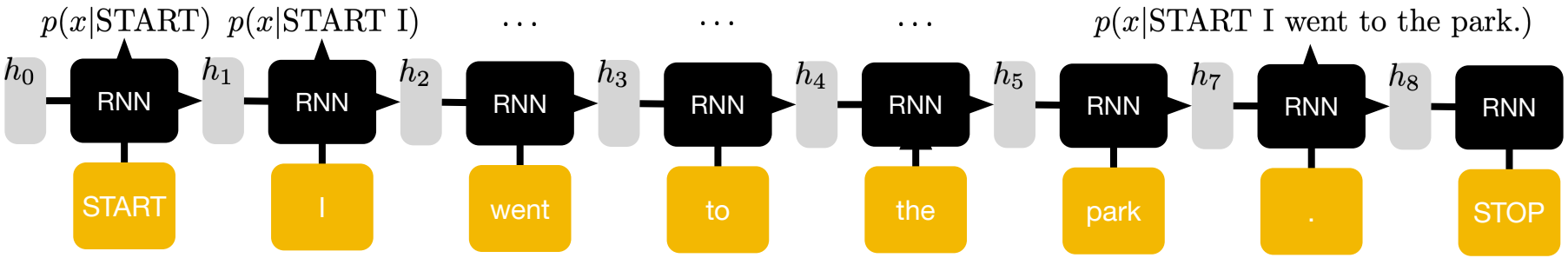


(Bengio et al., 2000)

Part 1

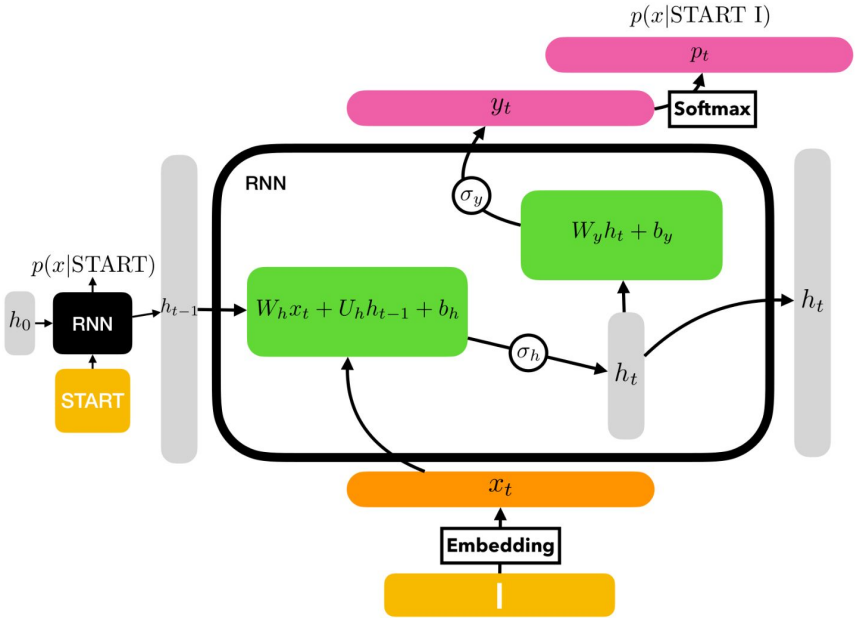
v1: Recurrent Neural Networks (RNN)

- One RNN is applied recursively to the sequence
- Inputs: previous hidden state ***h_t-1***, observation ***x_t***
- Outputs: next hidden state ***h_t***, (optionally) output ***y_t***
- Memory about history is passed through hidden states



Part 1

v1: Recurrent Neural Networks (RNN)



Variables:

x_t: input (embedding) vector

y_t: output vector (logits)

p_t: probability over tokens

h_{t-1}: previous hidden vector

h_t: next hidden vector

σ_h: activation function for hidden state

σ_y: output activation function

Equations:

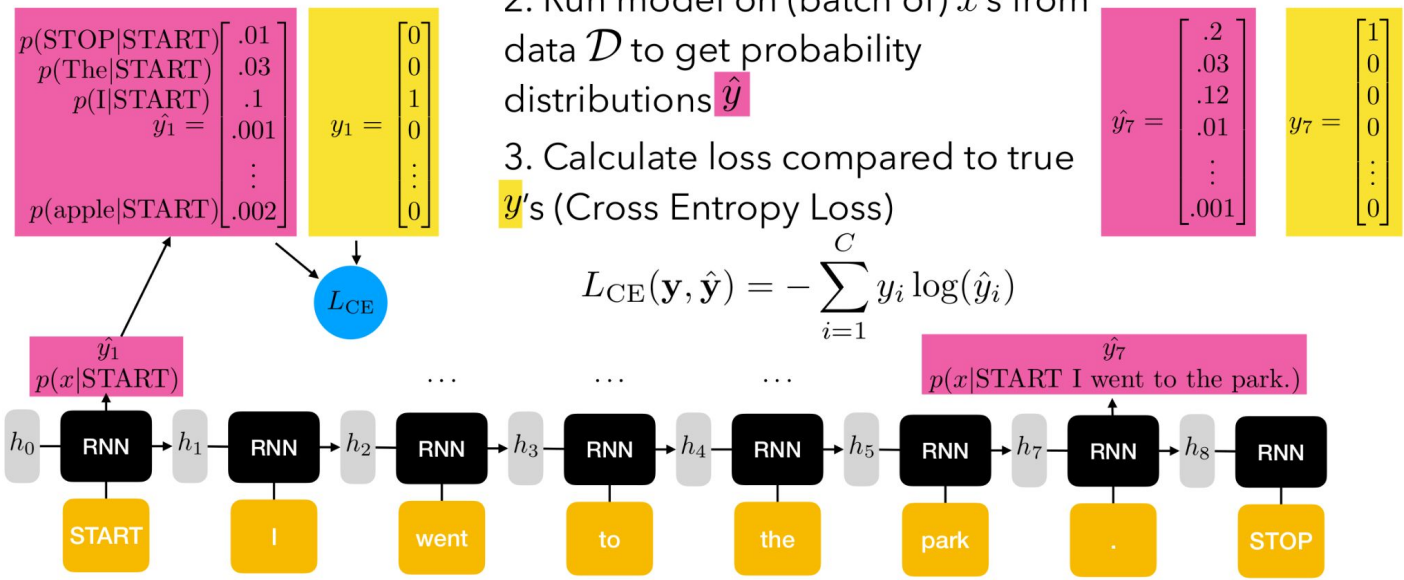
$$h_t := \sigma_h(W_h x_t + U_h h_{t-1} + b_h)$$

$$y_t := \sigma_y(W_y h_t + b_y)$$

$$p_{t_i} = \frac{\exp(y_{t_i})}{\sum_{i=j}^{d} \exp(y_{t_j})}$$

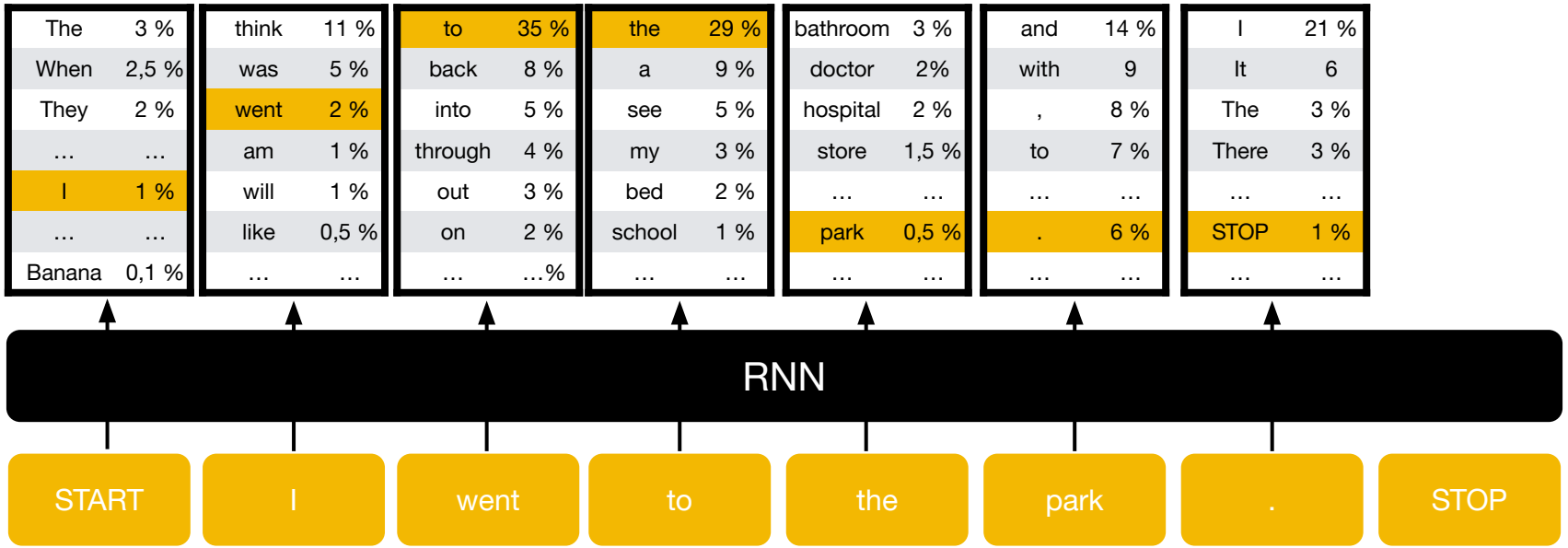
Part 1

Generation as a Sequence of Classifications



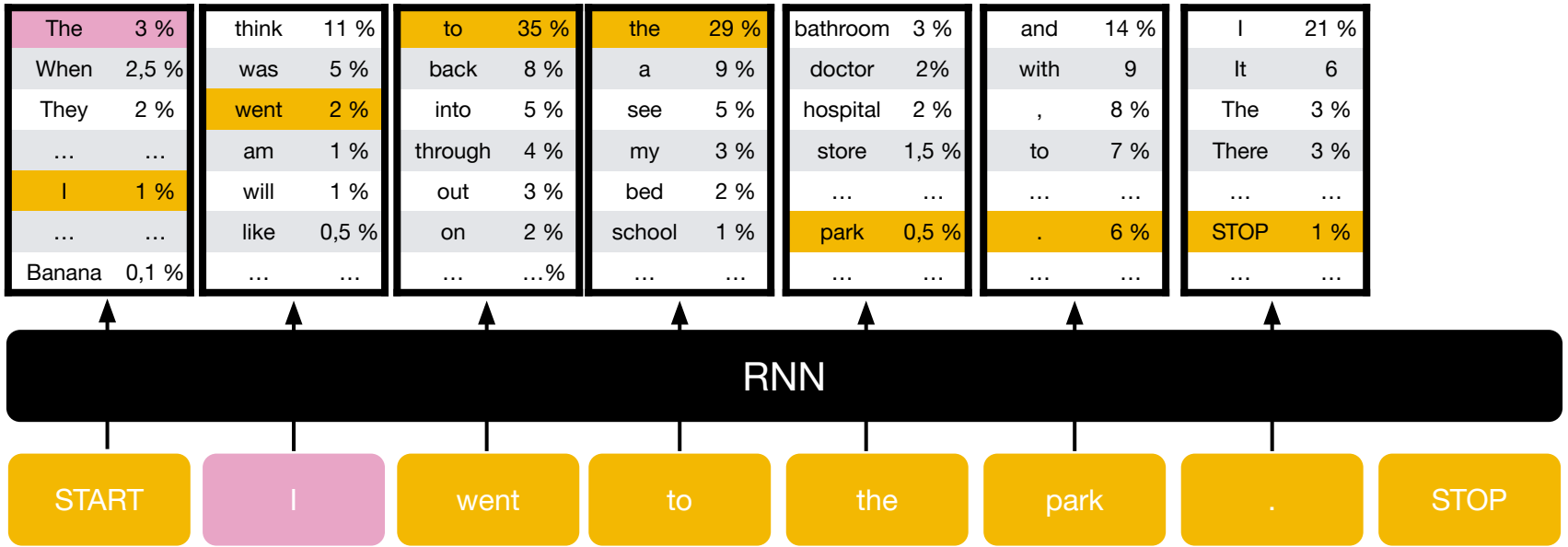
Part 1

Teacher Forcing

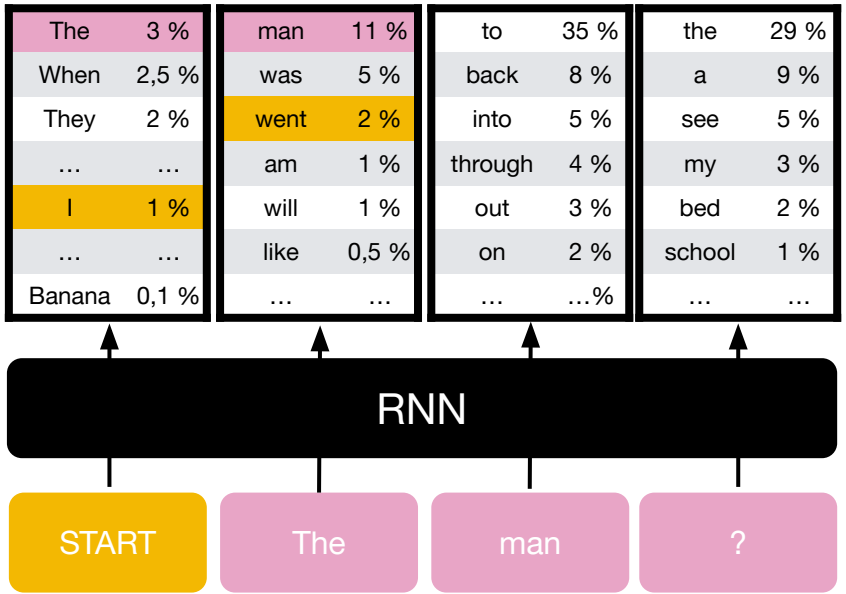


Part 1

Train-test mismatch: exposure bias



Train-test mismatch: exposure bias



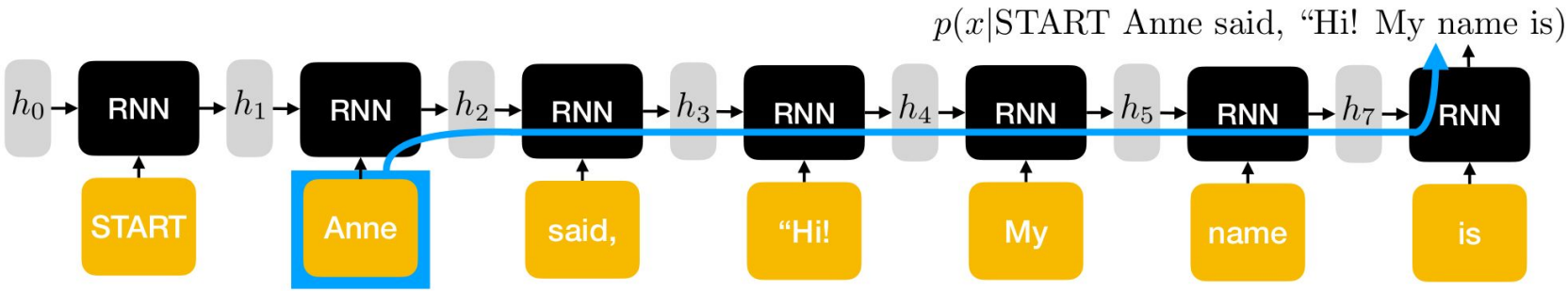
- Keep in mind: with RNN its trivial to train the model with its own generation instead of teacher forcing
- reduces exposure bias

Part 1

RNN limit 1: vanishing gradients

- What word is likely to come next for this sequence?

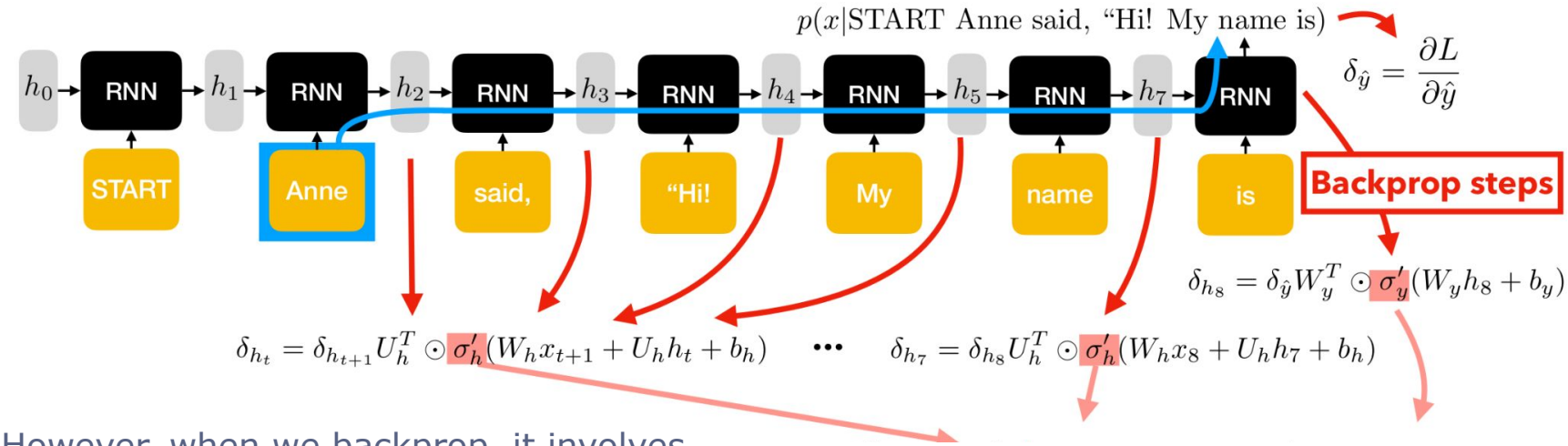
  Anne said, "Hi! My name is



- Need relevant information to flow across many time steps
- When we backpropagate, we want to allow the relevant information to flow

Part 1

RNN limit 1: vanishing gradients



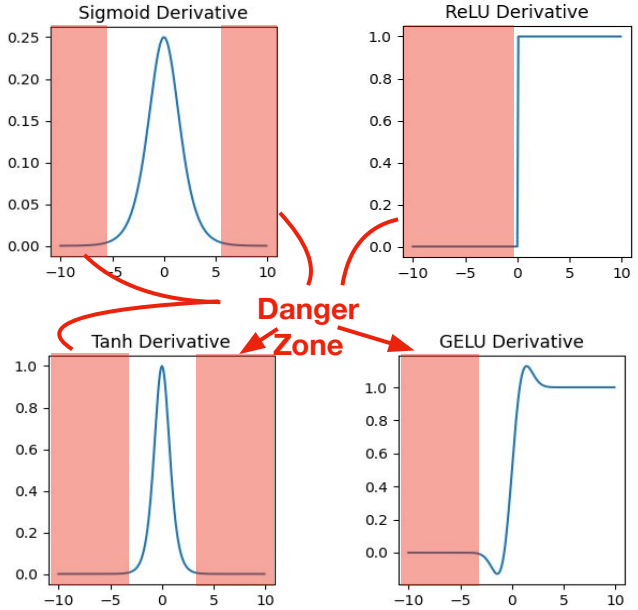
However, when we backprop, it involves multiplying a chain of computations from time to time t_1 t_7

If any of the terms are close to zero, the whole gradient goes to zero (vanishes!)

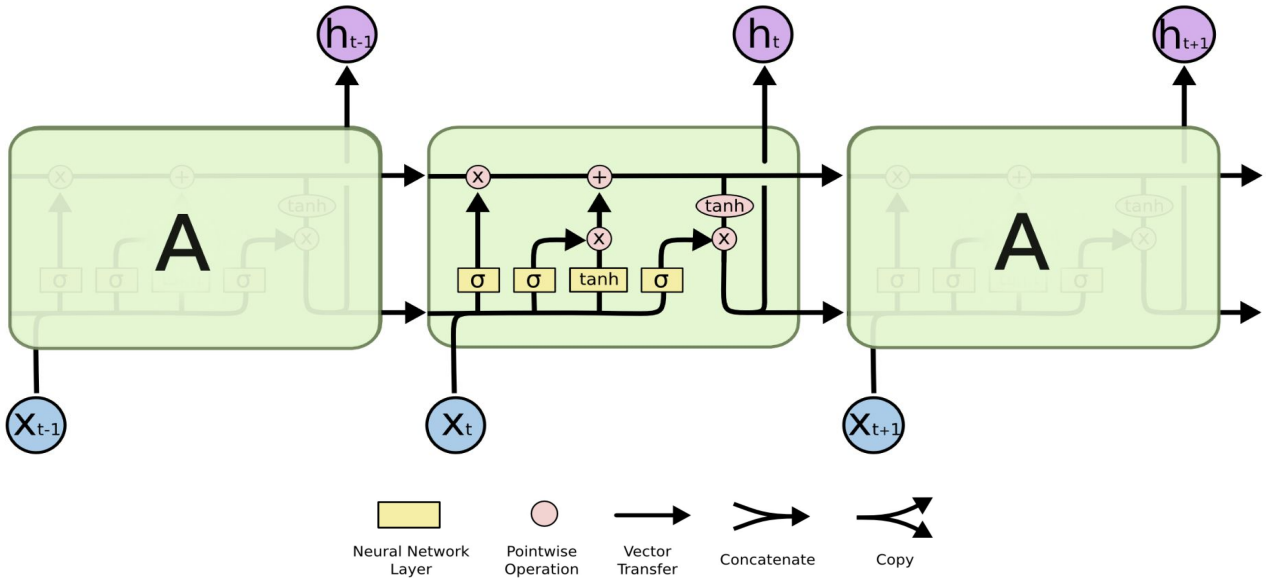
RNN limit 1: vanishing gradients

$$\delta_{h_t} = \delta_{h_{t+1}} U_h^T \odot \sigma'_h(W_h x_{t+1} + U_h h_t + b_h)$$

- If any of the terms are close to zero, the whole gradient goes to zero (vanishes!)
- This happens often for many activation functions… the gradient is close to zero when outputs get very large or small
- The more time steps back, the more chances for a vanishing gradient

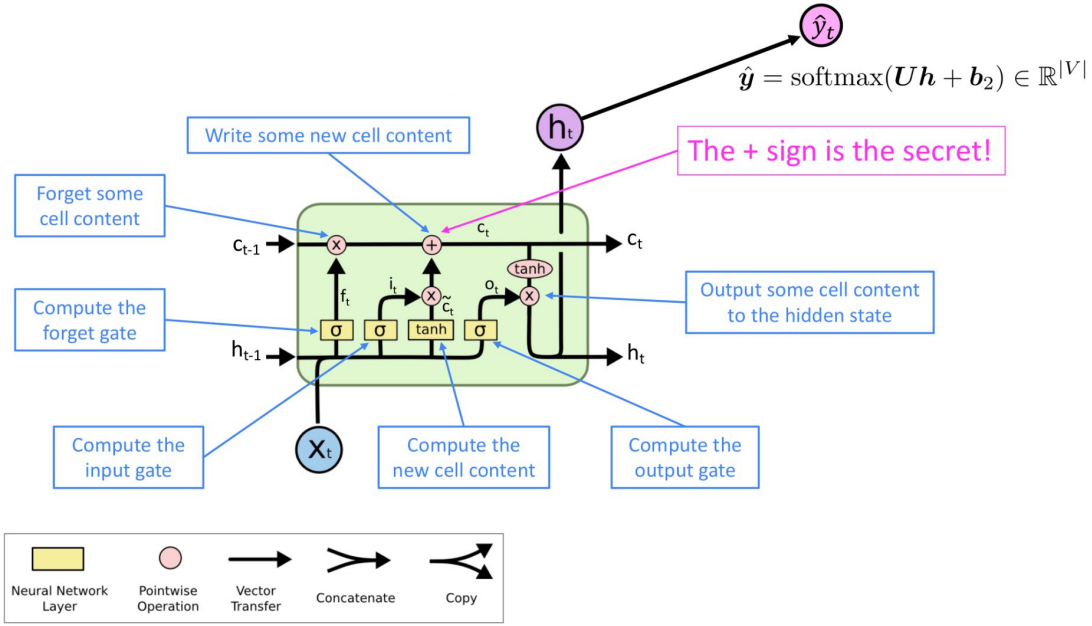


v1.1: Long-Short Term Memory (LSTM)



(Hochreiter & Schmidhuber, 1997)

v1.1: Long-Short Term Memory (LSTM)

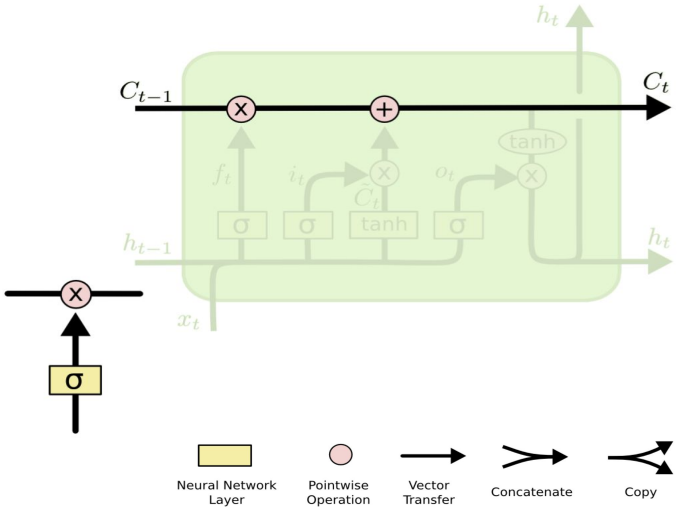


(Hochreiter & Schmidhuber, 1997)

Part 1

v1.1: Long-Short Term Memory (LSTM)

Cell state (**long-term memory**): allows information to flow with only small, **linear interactions** (good for gradients!)



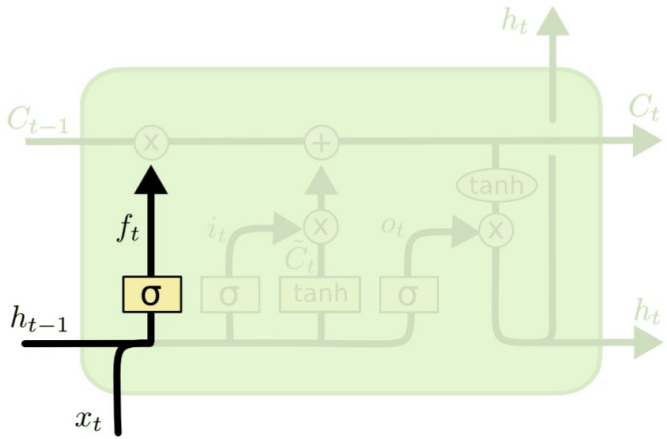
$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f)$$
$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i)$$
$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o)$$
$$\tilde{c}_t = \sigma_c(W_c x_t + U_c h_{t-1} + b_c)$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$$
$$h_t = o_t \odot \sigma_h(c_t)$$

$x_t \in \mathbb{R}^d$: input vector to the LSTM unit
$f_t \in (0,1)^h$: forget gate's activation vector
$i_t \in (0,1)^h$: input/update gate's activation vector
$o_t \in (0,1)^h$: output gate's activation vector
$h_t \in (-1,1)^h$: hidden state vector also known as output vector of the LSTM unit
$\tilde{c}_t \in (-1,1)^h$: cell input activation vector
$c_t \in \mathbb{R}^h$: cell state vector

(Hochreiter & Schmidhuber, 1997)

v1.1: Long-Short Term Memory (LSTM)

Input Gate Layer: Decide what information to "**forget**"



$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f)$$
$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i)$$
$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o)$$
$$\tilde{c}_t = \sigma_c(W_c x_t + U_c h_{t-1} + b_c)$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$$
$$h_t = o_t \odot \sigma_h(c_t)$$

$x_t \in \mathbb{R}^d$: input vector to the LSTM unit

$f_t \in (0,1)^h$: forget gate's activation vector

$i_t \in (0,1)^h$: input/update gate's activation vector

$o_t \in (0,1)^h$: output gate's activation vector

$h_t \in (-1,1)^h$: hidden state vector also known as output vector of the LSTM unit

$\tilde{c}_t \in (-1,1)^h$: cell input activation vector

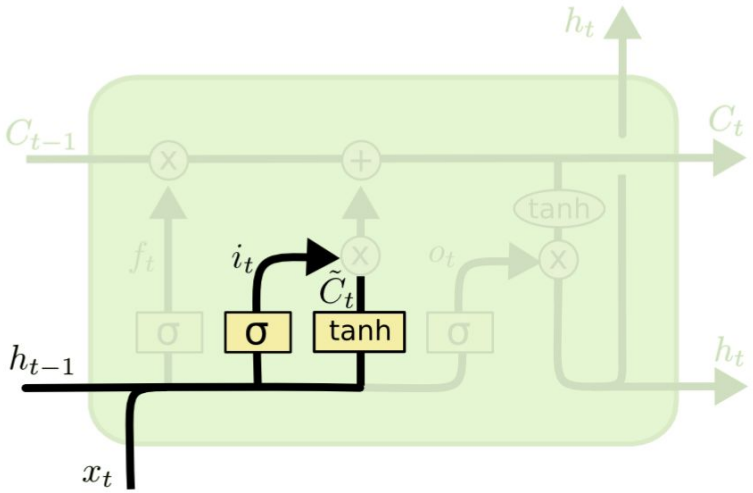
$c_t \in \mathbb{R}^h$: cell state vector

(Gers et al., 2000)

Part 1

v1.1: Long-Short Term Memory (LSTM)

Candidate state values: Extract candidate information to put into the cell vector: "**remember**"



$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f)$$
$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i)$$
$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o)$$
$$\tilde{c}_t = \sigma_c(W_c x_t + U_c h_{t-1} + b_c)$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$$
$$h_t = o_t \odot \sigma_h(c_t)$$

$x_t \in \mathbb{R}^d$: input vector to the LSTM unit

$f_t \in (0,1)^h$: forget gate's activation vector

$i_t \in (0,1)^h$: input/update gate's activation vector

$o_t \in (0,1)^h$: output gate's activation vector

$h_t \in (-1,1)^h$: hidden state vector also known as output vector of the LSTM unit

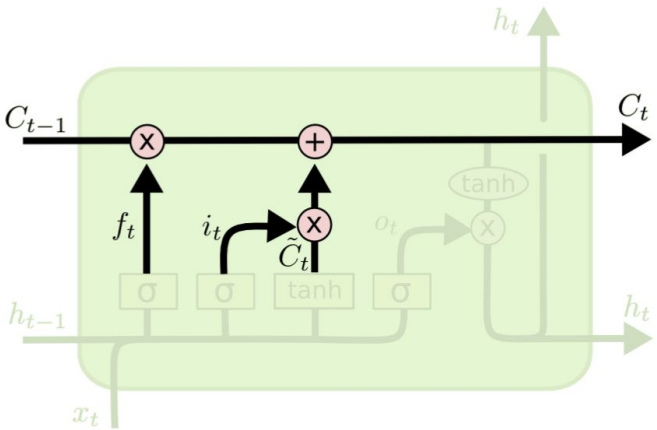
$\tilde{c}_t \in (-1,1)^h$: cell input activation vector

$c_t \in \mathbb{R}^h$: cell state vector

Part 1

v1.1: Long-Short Term Memory (LSTM)

Update cell: "Forget" the information we decided to forget and update with new candidate information



If ***f_t*** is:

- High: we remember more previous info
- Low: we "forget" more info

$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f)$$
$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i)$$
$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o)$$
$$\tilde{c}_t = \sigma_c(W_c x_t + U_c h_{t-1} + b_c)$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$$
$$h_t = o_t \odot \sigma_h(c_t)$$

If ***i_t*** is:

- High: we add more new info
- Low: we add less new info

$x_t \in \mathbb{R}^d$: input vector to the LSTM unit

$f_t \in (0,1)^h$: forget gate's activation vector

$i_t \in (0,1)^h$: input/update gate's activation vector

$o_t \in (0,1)^h$: output gate's activation vector

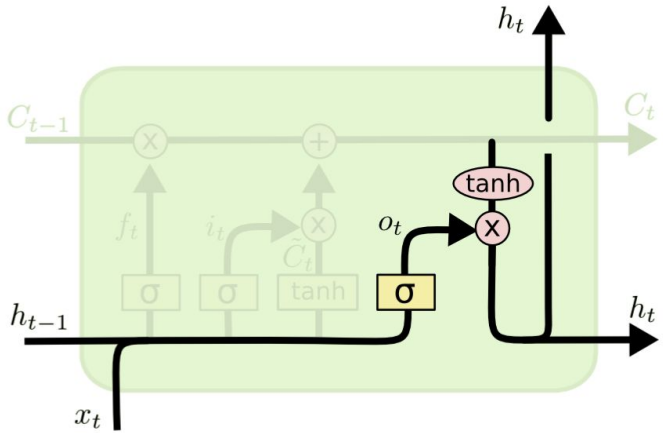
$h_t \in (-1,1)^h$: hidden state vector also known as output vector of the LSTM unit

$\tilde{c}_t \in (-1,1)^h$: cell input activation vector

$c_t \in \mathbb{R}^h$: cell state vector

v1.1: Long-Short Term Memory (LSTM)

Output/**Short-term Memory** (as in RNN):
Pass information onto the next state/for use in output (e.g., probabilities)



$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f)$$
$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i)$$
$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o)$$
$$\tilde{c}_t = \sigma_c(W_c x_t + U_c h_{t-1} + b_c)$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$$
$$h_t = o_t \odot \sigma_h(c_t)$$

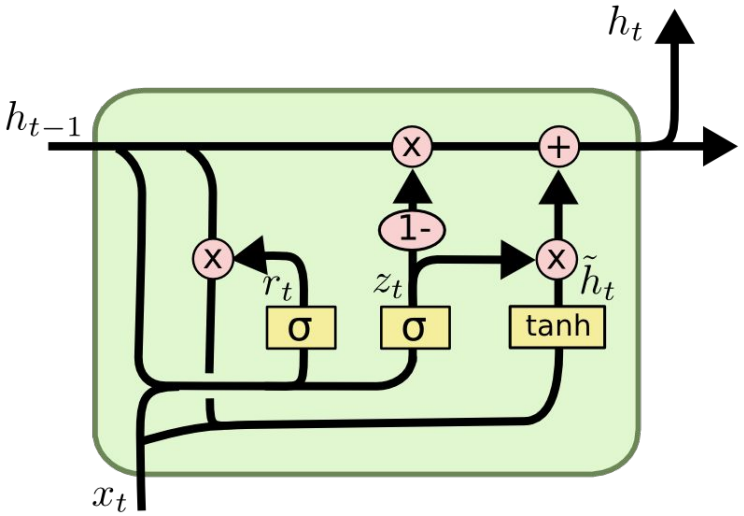
$x_t \in \mathbb{R}^d$: input vector to the LSTM unit
$f_t \in (0,1)^h$: forget gate's activation vector
$i_t \in (0,1)^h$: input/update gate's activation vector
$o_t \in (0,1)^h$: output gate's activation vector
$h_t \in (-1,1)^h$: hidden state vector also known as output vector of the LSTM unit
$\tilde{c}_t \in (-1,1)^h$: cell input activation vector
$c_t \in \mathbb{R}^h$: cell state vector

LSTM for Machine Translation

Type	Sentence
Our model	Ulrich UNK , membre du conseil d' administration du constructeur automobile Audi , affirme qu' il s' agit d' une pratique courante depuis des années pour que les téléphones portables puissent être collectés avant les réunions du conseil d' administration afin qu' ils ne soient pas utilisés comme appareils d' écoute à distance .
Truth	Ulrich Hackenberg , membre du conseil d' administration du constructeur automobile Audi , déclare que la collecte des téléphones portables avant les réunions du conseil , afin qu' ils ne puissent pas être utilisés comme appareils d' écoute à distance , est une pratique courante depuis des années .
Our model	" Les téléphones cellulaires , qui sont vraiment une question , non seulement parce qu' ils pourraient potentiellement causer des interférences avec les appareils de navigation , mais nous savons , selon la FCC , qu' ils pourraient interférer avec les tours de téléphone cellulaire lorsqu' ils sont dans l' air " , dit UNK .
Truth	" Les téléphones portables sont véritablement un problème , non seulement parce qu' ils pourraient éventuellement créer des interférences avec les instruments de navigation , mais parce que nous savons , d' après la FCC , qu' ils pourraient perturber les antennes-relais de téléphonie mobile s' ils sont utilisés à bord " , a déclaré Rosenker .

Sutskever et al. (2014)

Gated Recurrent Units (GRU)



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

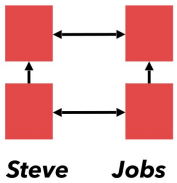
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Cho et al. (2014)

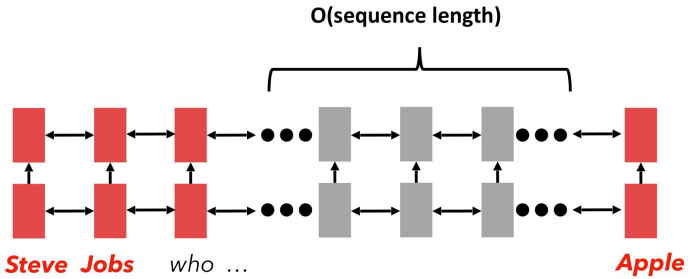
Part 1

RNN limit 2: Linear Interaction Distance

- RNNs are unrolled left-to-right.
- Linear locality is a useful heuristic: nearby words often affect each other's meaning!

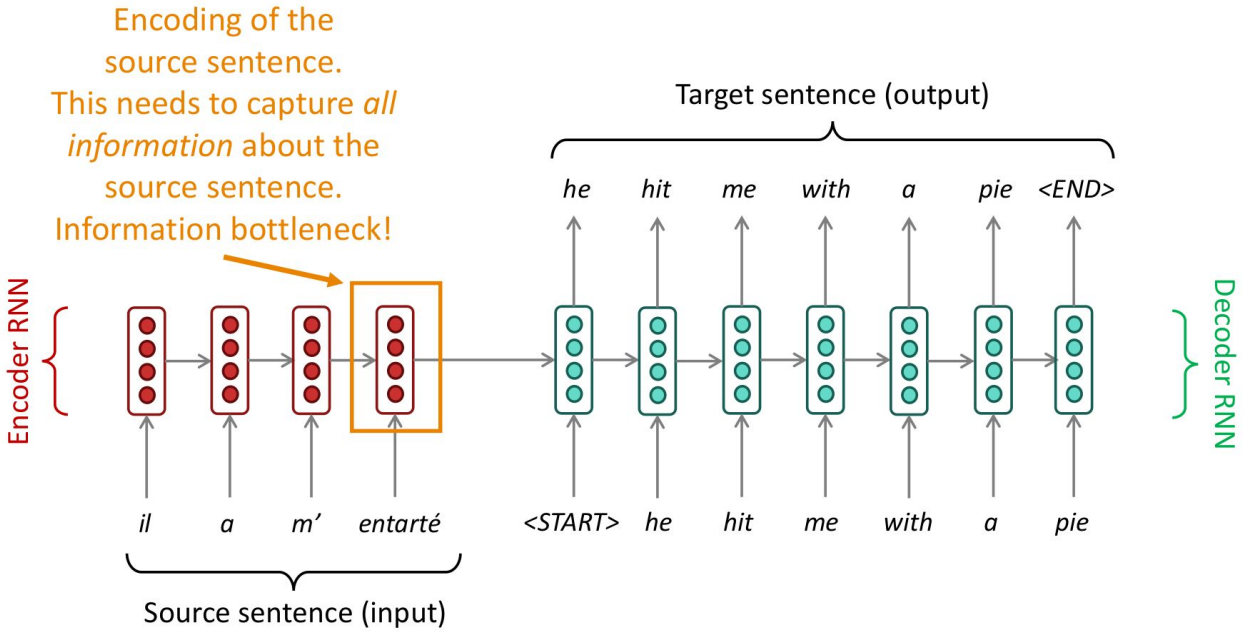


- Better with LSTM then RNN, but still:
  - Failing to capture long-term dependencies
  - Vanishing gradient



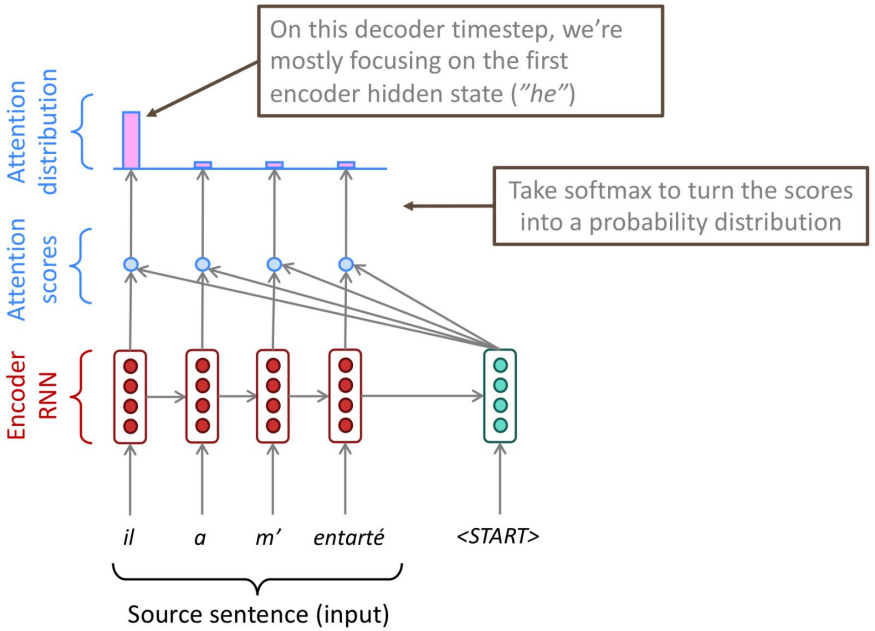
Part 1

Why Attention?



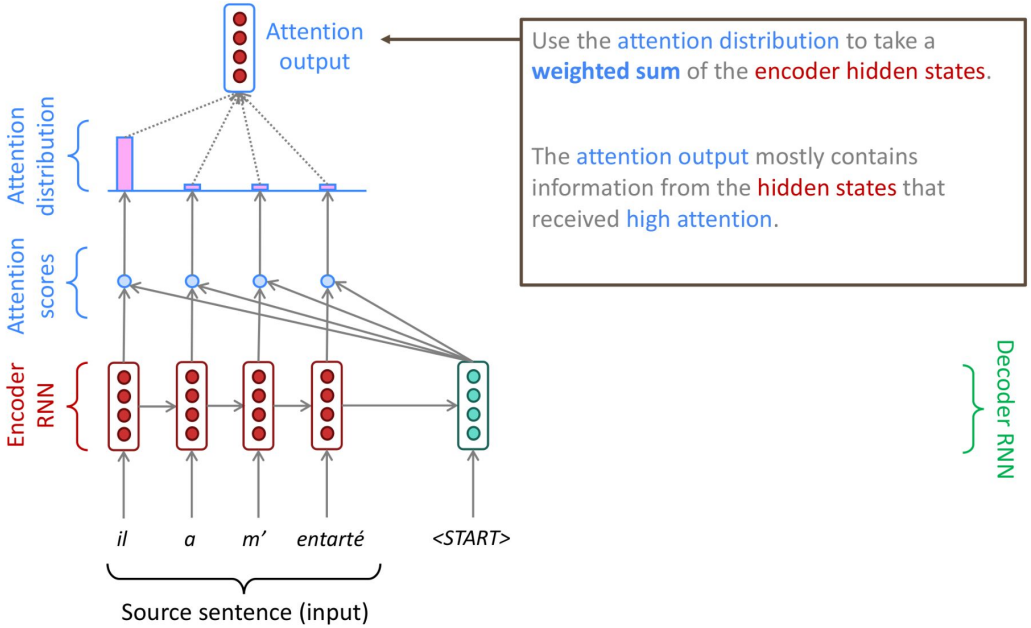
Part 1

LSTM with Attention



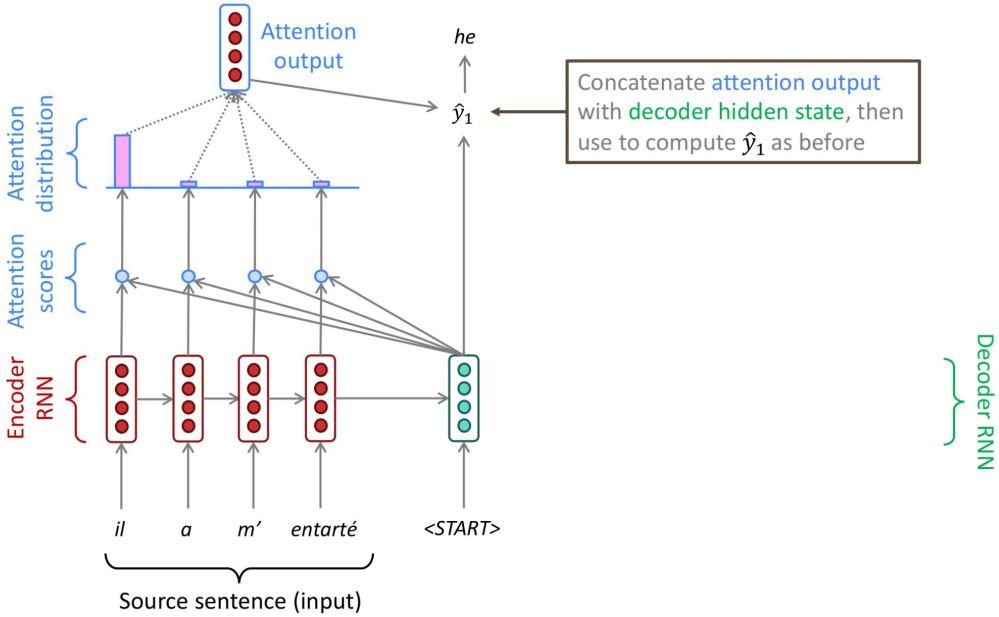
On each step of the decoder, use **direct connection** to the encoder to focus on a particular part of the source sequence

LSTM with Attention



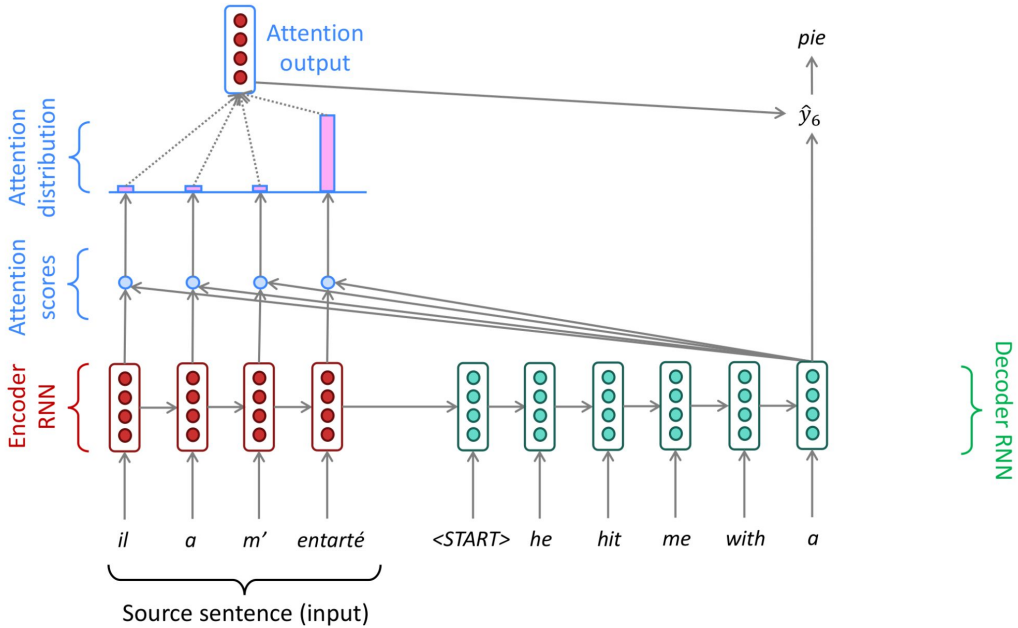
On each step of the decoder, use **direct connection** to the encoder to focus on a particular part of the source sequence

LSTM with Attention



On each step of the decoder, use **direct connection** to the encoder to focus on a particular part of the source sequence

LSTM with Attention



On each step of the decoder, use **direct connection** to the encoder to focus on a particular part of the source sequence

Part 1

GRU+Attention for Machine Translation

Source: An admitting privilege is the right of a doctor to admit a patient to a hospital or a medical centre to carry out a diagnosis or a procedure, based on his status as a health care worker at a hospital.

Reference: Le privilège d'admission est le droit d'un médecin, en vertu de son statut de membre soignant d'un hôpital, d'admettre un patient dans un hôpital ou un centre médical afin d'y délivrer un diagnostic ou un traitement.

RNNenc-50: Un privilège d'admission est le droit d'un médecin de reconnaître un patient à l'hôpital ou un centre médical d'un diagnostic ou de prendre un diagnostic en fonction de son état de santé.

RNNsearch-50: Un privilège d'admission est le droit d'un médecin d'admettre un patient à un hôpital ou un centre médical pour effectuer un diagnostic ou une procédure, selon son statut de travailleur des soins de santé à l'hôpital.

Bahdanau et al. 2015

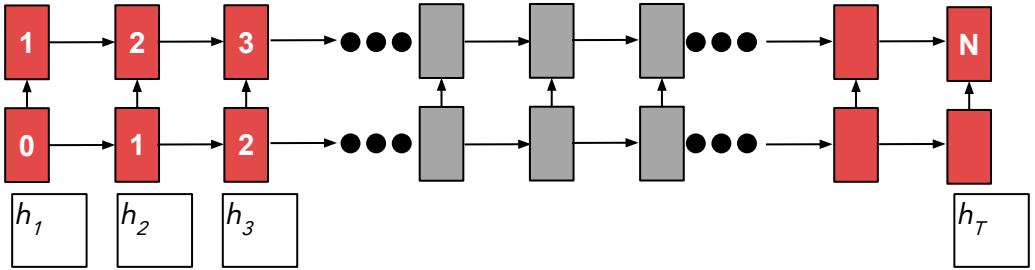
Part 1

Why Attention?

- Attention significantly improves Translation performance:
  It's very useful to allow decoder to focus on certain parts of the source
- Attention provides a more "human-like" model of the MT process:
  You can look back at the source sentence while translating, rather than needing to remember it all
- Attention solves the bottleneck problem:
  allows decoder to look directly at source
- Attention helps with the vanishing gradient problem:
  Provides shortcut to faraway states

RNN limit 3: Parallelization

- Forward and backward passes have O(sequence length) unparallelizable operations
- GPUs can perform many independent computations (like addition) at once!
- But future RNN hidden states can't be computed in full before past RNN hidden states have been computed.
- Training and inference are slow; inhibits on very large datasets!



Numbers indicate min # of steps before a state can be computed

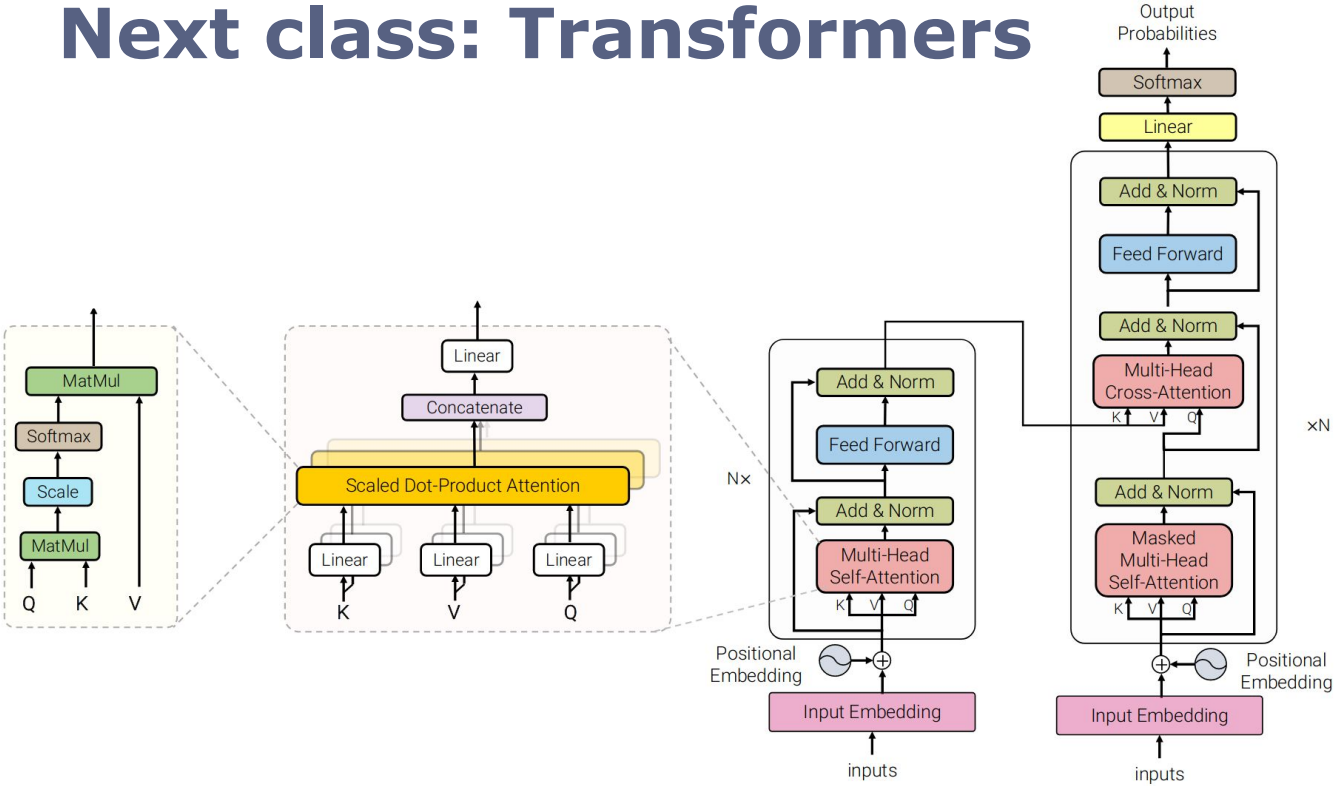
Part 1

Summarizing

- Recurrent Neural Networks allow to process variable-length input/model language without Markov assumption
- Limited by vanishing gradients → LSTM
- Attention was invented for RNNs and Translation, to focus on parts of the source sentence: relieves information bottleneck
- Limited by parallelization → Transformers (next class)

Part 1

Next class: Transformers



Part 0

Acknowledgements

This class directly builds upon:

- **Jurafsky, D., & Martin, J. H.** (2024). Speech and Language Processing : An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models (3rd éd.).
- **Eisenstein, J.** (2019). Natural Language Processing. 587.
- **Yejin Choi**. (Winter 2024). CSE 447/517: Natural Language Processing (University of Washington - Paul G. Allen School of Computer Science & Engineering)
- **Noah Smith**. (Winter 2023). CSE 447/517: Natural Language Processing (University of Washington - Paul G. Allen School of Computer Science & Engineering)
- **Benoît Sagot**. (2023-2024). *Apprendre les langues aux machines* (Collège de France)
- **Chris Manning**. (Spring 2024). Stanford CS224N: Natural Language Processing with Deep Learning
- Classes where I was/am Teacher Assistant:
  - **Christopher Kermorvant**. Machine Learning for Natural Language Processing (ENSAE)
  - **François Landes** and **Kim Gerdes**. Introduction to Machine Learning and NLP (Paris-Saclay)

Also inspired by:

- My PhD thesis: *Répondre aux questions visuelles à propos d'entités nommées* (2023)
- **Noah Smith** (2023): Introduction to Sequence Models (LxMLS)
- **Kyunghyun Cho**: Transformers and Large Pretrained Models (LxMLS 2023), Neural Machine Translation (ALPS 2021)
- My former PhD advisors **Olivier Ferret** and **Camille Guinaudeau** and postdoc advisor **François Yvon**
- My former colleagues at LISN

aivancity

PARIS-CACHAN

advancing education
in artificial intelligence

Part 1

Perplexity

- How "hard" is the task of recognizing digits '0,1,2,..,9' uniformly at random?

$d \sim \text{Uniform}(0, 9)$

$$\text{PP}(d_1, \ldots, d_N) = p(d_1, \ldots, d_N)^{-\frac{1}{N}} = \left(\frac{1}{10}^{N}\right)^{-\frac{1}{N}} = \frac{1}{10}^{-1} = 10$$

  - Perplexity: 10
  - Using entropy (replacing the estimated distribution with the known true dist.):

$$H(D, D) = -\sum_{d \in D} p(d) \log p(d) = -\sum_{i=0}^{9} p(i) \log p(i) = -\sum_{i=0}^{9} \frac{1}{10} \log\left(\frac{1}{10}\right) = -\log\left(\frac{1}{10}\right)$$

$$PP(D) = e^{H(D,D)} = e^{-\log\left(\frac{1}{10}\right)} = \left(e^{\log\left(\frac{1}{10}\right)}\right)^{-1} = \left(\frac{1}{10}\right)^{-1} = 10$$

Same result!

LSTM with Attention: formally

We have encoder hidden states $h_1, \ldots, h_N \in \mathbb{R}^h$

On timestep *t*, we have decoder hidden state $s_t \in \mathbb{R}^h$

We get the attention scores $\boldsymbol{e}^t$ for this step:

$$\boldsymbol{e}^t = [\boldsymbol{s}_t^T \boldsymbol{h}_1, \ldots, \boldsymbol{s}_t^T \boldsymbol{h}_N] \in \mathbb{R}^N$$

We take softmax to get the attention distribution α^t for this step (this is a probability distribution and sums to 1)

$$\alpha^t = \text{softmax}(\boldsymbol{e}^t) \in \mathbb{R}^N$$

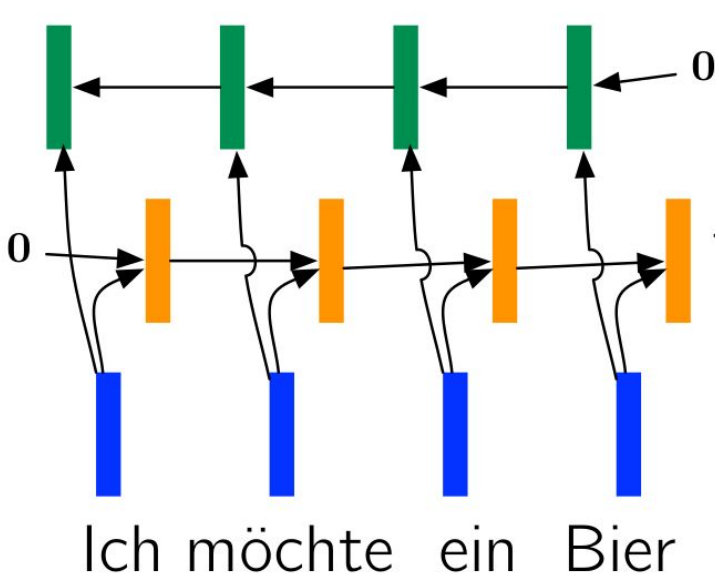
We use α^t to take a weighted sum of the encoder hidden states to get the attention output $\boldsymbol{a}_t$

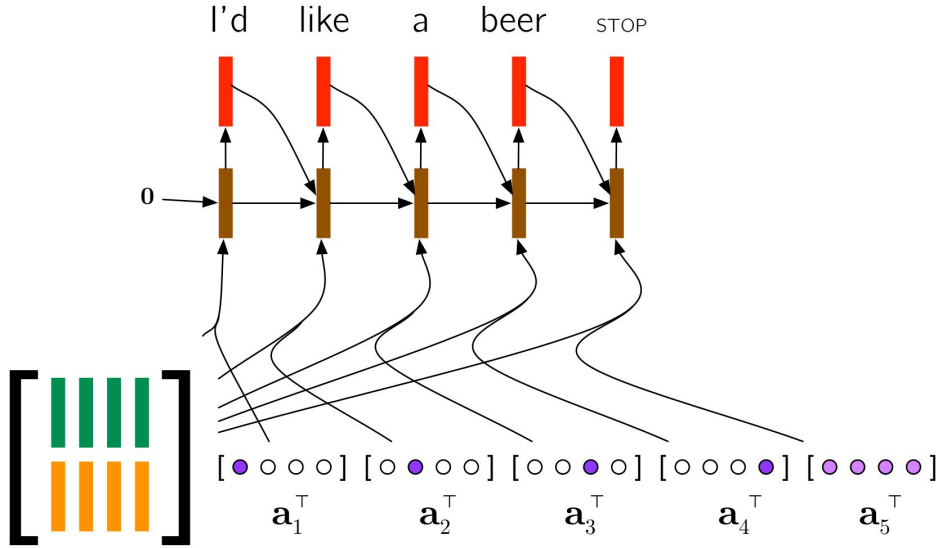
$$\boldsymbol{a}_t = \sum_{i=1}^{N} \alpha_i^t \boldsymbol{h}_i \in \mathbb{R}^h$$

Finally we concatenate the attention output $\boldsymbol{a}_t$ with the decoder hidden state s_t and proceed as in the non-attention seq2seq model

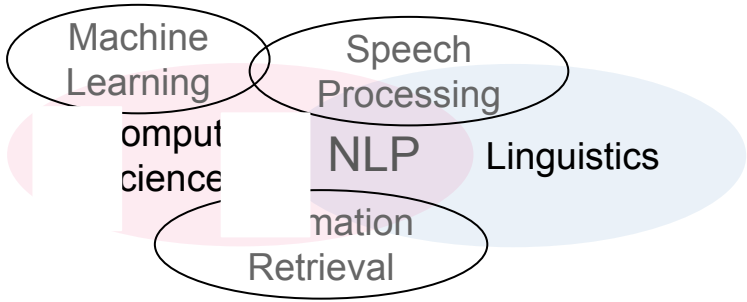
$$[\boldsymbol{a}_t; \boldsymbol{s}_t] \in \mathbb{R}^{2h}$$

Sequence-to-Sequence (Translation)





Why Sequence Models? and Attention?



- Close to Speech Processing (Automatic Speech Recognition etc.)
- Close to Information Retrieval (Search engines like Google)
- Driven by Statistical/Machine Learning methods since the 90s (Brown, P. F., Della Pietra, S. A., Della Pietra, V. J., & Mercer, R. L. (1993). The Mathematics of Statistical Machine Translation : Parameter Estimation. Computational Linguistics, 19(2), 263-311.)
- Driven by Deep Learning since 2013 (Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed Representations of Words and Phrases and their Compositionality. Advances in Neural Information Processing Systems)