Neural Random Access Machines

a deep learning technique for sequential data

Based on the PhD thesis of
Dr Karol Kurach
Warsaw University/Google

Agenda

1. **Introduction to Deep Neural Architectures**
2. Neural Random Access-Machines
3. Hierarchical Attentive Memory
4. Applications: Smart Reply
5. Applications: Efficient Math Identities
6. Applications: Predicting Events From Sensor Data

A primer on Deep Learning

Deep Learning

Big Data + Big Deep Model
= Success Guaranteed

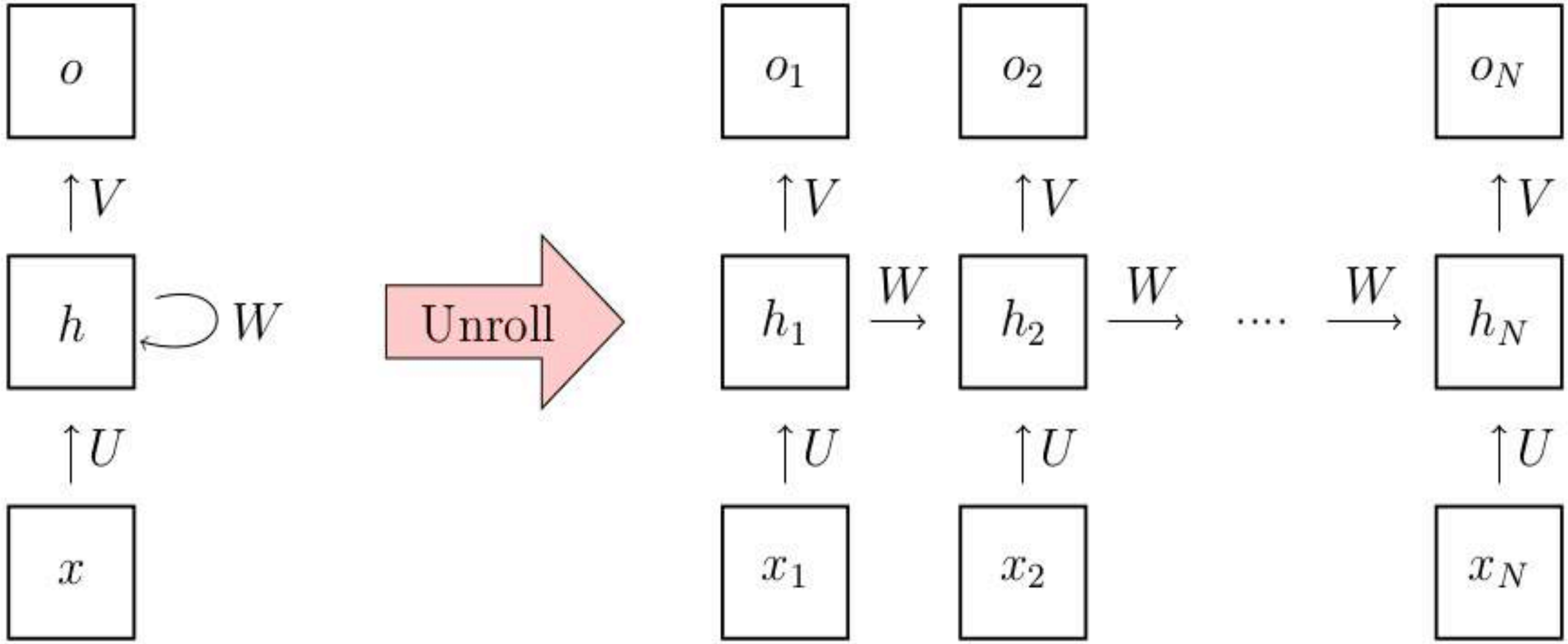
State of the art in:

- computer vision,
- speech recognition,
- machine translation, …

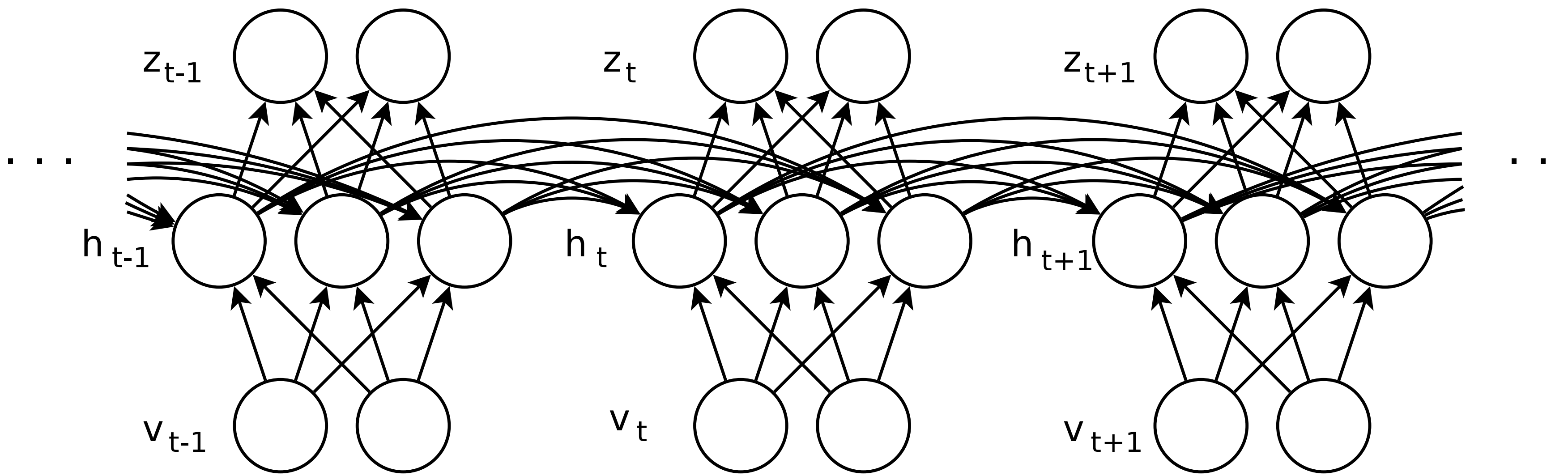
- New techniques (e.g., initialization, pretraining)
- Computing power (GPU, FPGA, TPU…)
- Big datasets

Recurrent Neural Networks

- ➢ Neural networks with cycles
- ➢ Process inputs of variable length
- ➢ Preserve state between timesteps

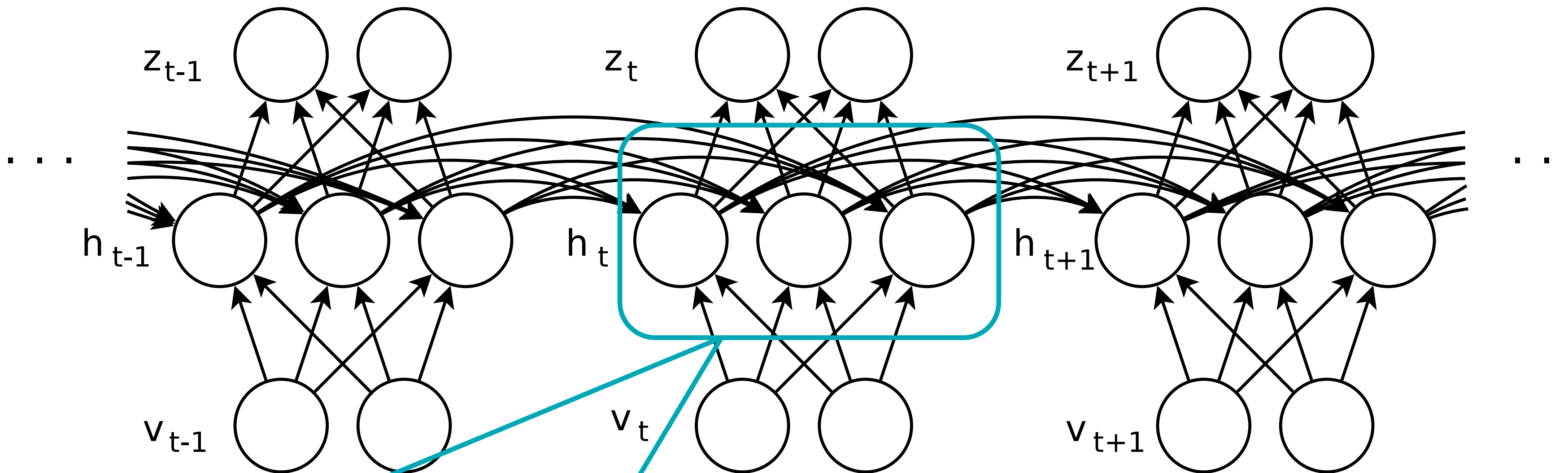


Recurrent Neural Networks



$v_1^T = (v_1, \ldots, v_T) \rightarrow \boxed{[W_{hv}, W_{hh}, W_{oh}, b_h, b_o, h_0]} \rightarrow h_1^T \rightarrow z_1^T$

Recurrent Neural Networks



for t **from** 1 **to** T **do**

$\quad u_t \leftarrow W_{hv} v_t + W_{hh} h_{t-1} + b_h$

$\quad h_t \leftarrow e(u_t)$

$\quad o_t \leftarrow W_{oh} h_t + b_o$

$\quad z_t \leftarrow g(o_t)$

end for

Vanilla RNN

- ➢ Basic version of RNN
- ➢ State: vector h

$$h_t = \tanh(U * x_t + W * h_{t-1})$$

$$o_t = output(V * h_t)$$

Learning RNN: BPTT

$$\boxed{L(z,y) = \sum_{t=1}^{T} L(z_t; y_t)}$$

(BPTT; Werbos, 1990; Rumelhart et al., 1986):

for t **from** T **downto** 1 **do**

$\quad do_t \leftarrow g'(o_t) \cdot dL(z_t; y_t)/dz_t$

$\quad db_o \leftarrow db_o + do_t$

$\quad dW_{oh} \leftarrow dW_{oh} + do_t h_t^\top$

$\quad dh_t \leftarrow dh_t + W_{oh}^\top do_t$

$\quad dz_t \leftarrow e'(z_t) \cdot dh_t$

$\quad dW_{hv} \leftarrow dW_{hv} + dz_t v_t^\top$

$\quad db_h \leftarrow db_h + dz_t$

$\quad dW_{hh} \leftarrow dW_{hh} + dz_t h_{t-1}^\top$

$\quad dh_{t-1} \leftarrow W_{hh}^\top dz_t$

end for

Return $d\theta = [dW_{hv}, dW_{hh}, dW_{oh}, db_h, db_o, dh_0]$.

Vanilla RNN - problems

- ➢ Exploding gradient (idea: use gradient clipping)
- ➢ Vanishing gradient (idea: use ReLU and/or LSTM)

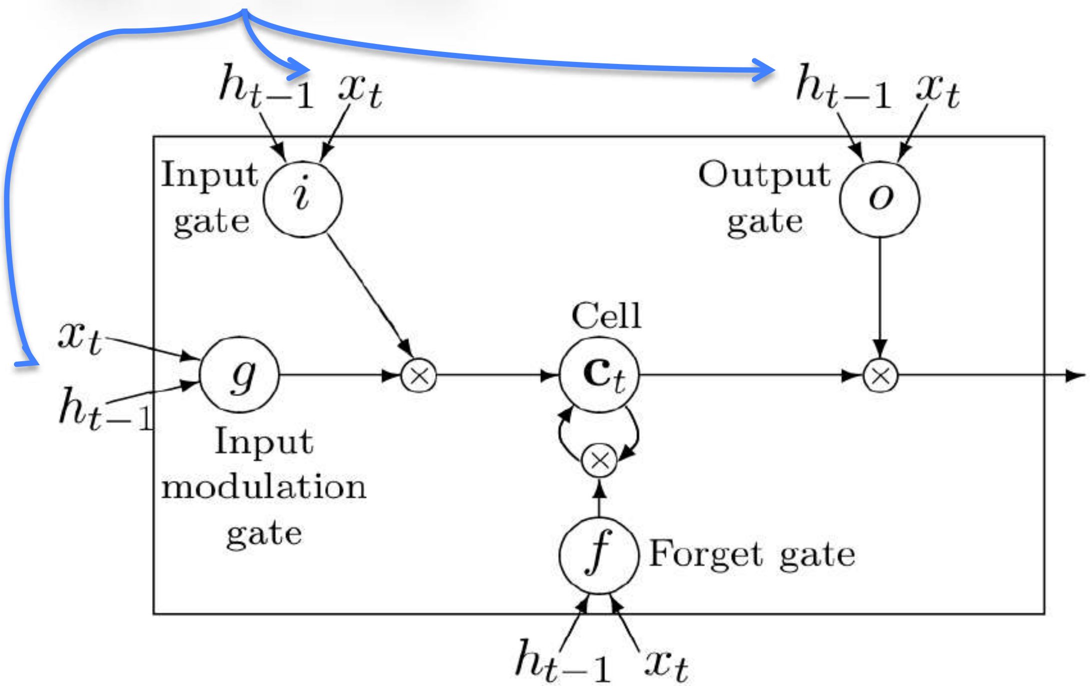
make it difficult to optimize RNNs on sequences with long-range temporal dependencies, and are possible causes for the abandonment of RNNs by machine learning researchers

LSTM: Long Short-Term Memory

(Hochreiter and Schmidhuber, 1997)

Input:

$(h_{t-1},\ c_{t-1}, x_t)$



Output:

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \text{sigm} \\ \text{sigm} \\ \text{sigm} \\ \tanh \end{pmatrix} T_{2n,4n} \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

- ➢ Better at learning long-range dependencies
- ➢ Avoid vanishing gradient problem
- ➢ State: a pair of vectors (c, h)

LSTM Cell

$$W_i, W_f, W_u, W_o$$

$$b_i, b_f, b_u, b_o$$

$$f_t = \mathrm{sigm}(W_f * [h_{t-1} \oplus x_t] + b_f)$$

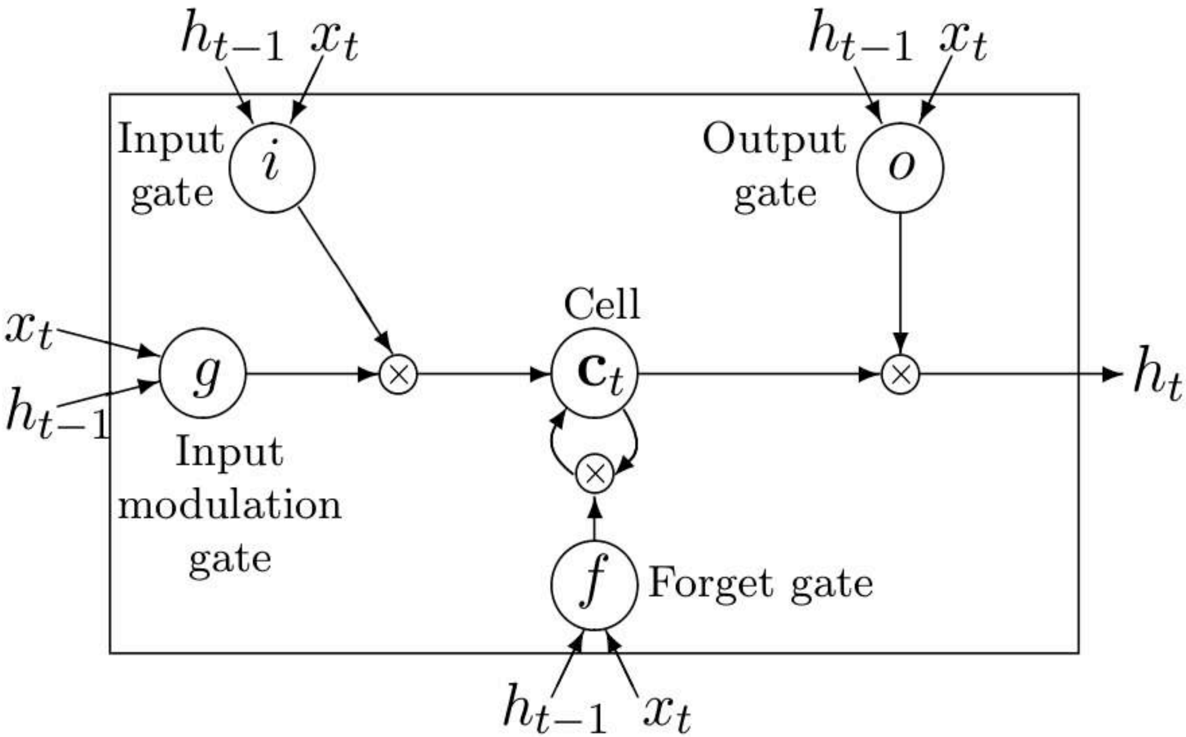
$$i_t = \mathrm{sigm}(W_i * [h_{t-1} \oplus x_t] + b_i)$$

$$u_t = \tanh(W_u * [h_{t-1} \oplus x_t] + b_u)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t$$

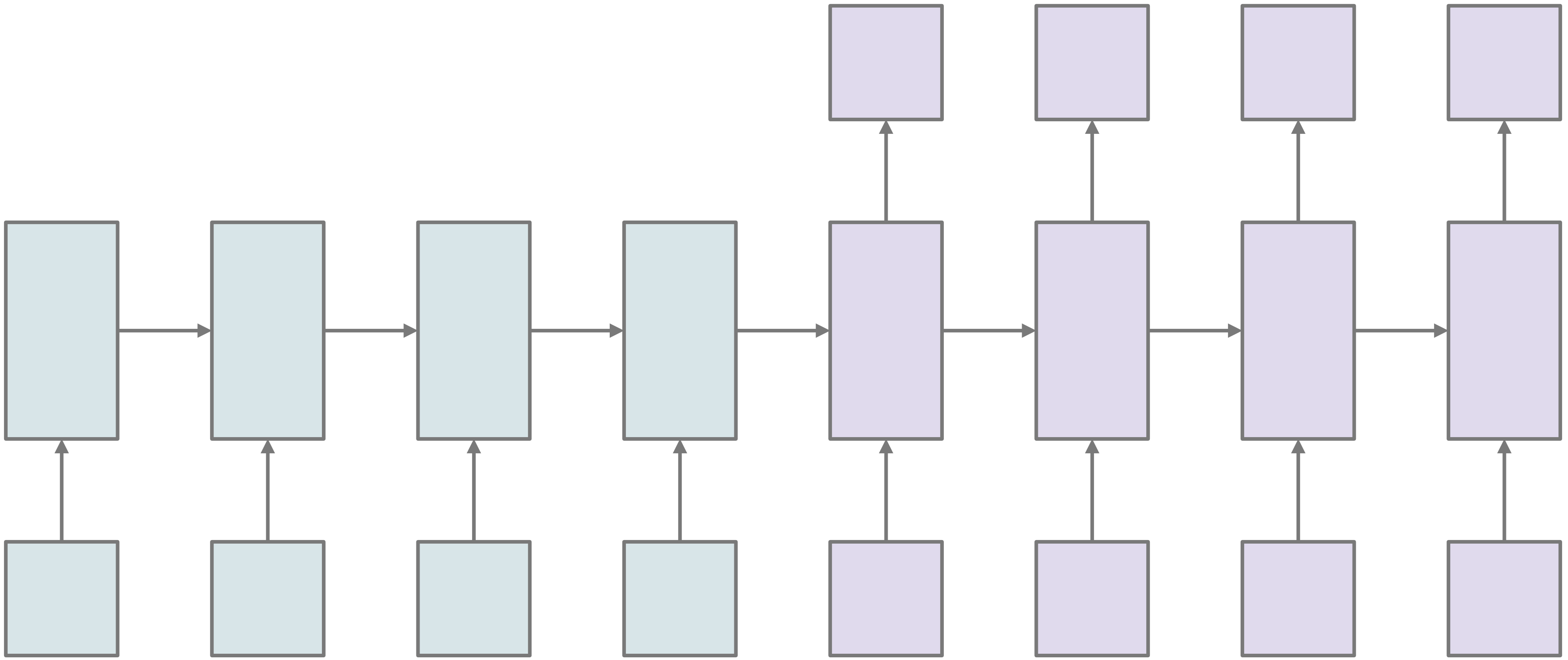
$$o_t = \mathrm{sigm}(W_o * [h_{t-1} \oplus x_t] + b_o)$$

$$h_t = o_t \odot \tanh(c_t)$$



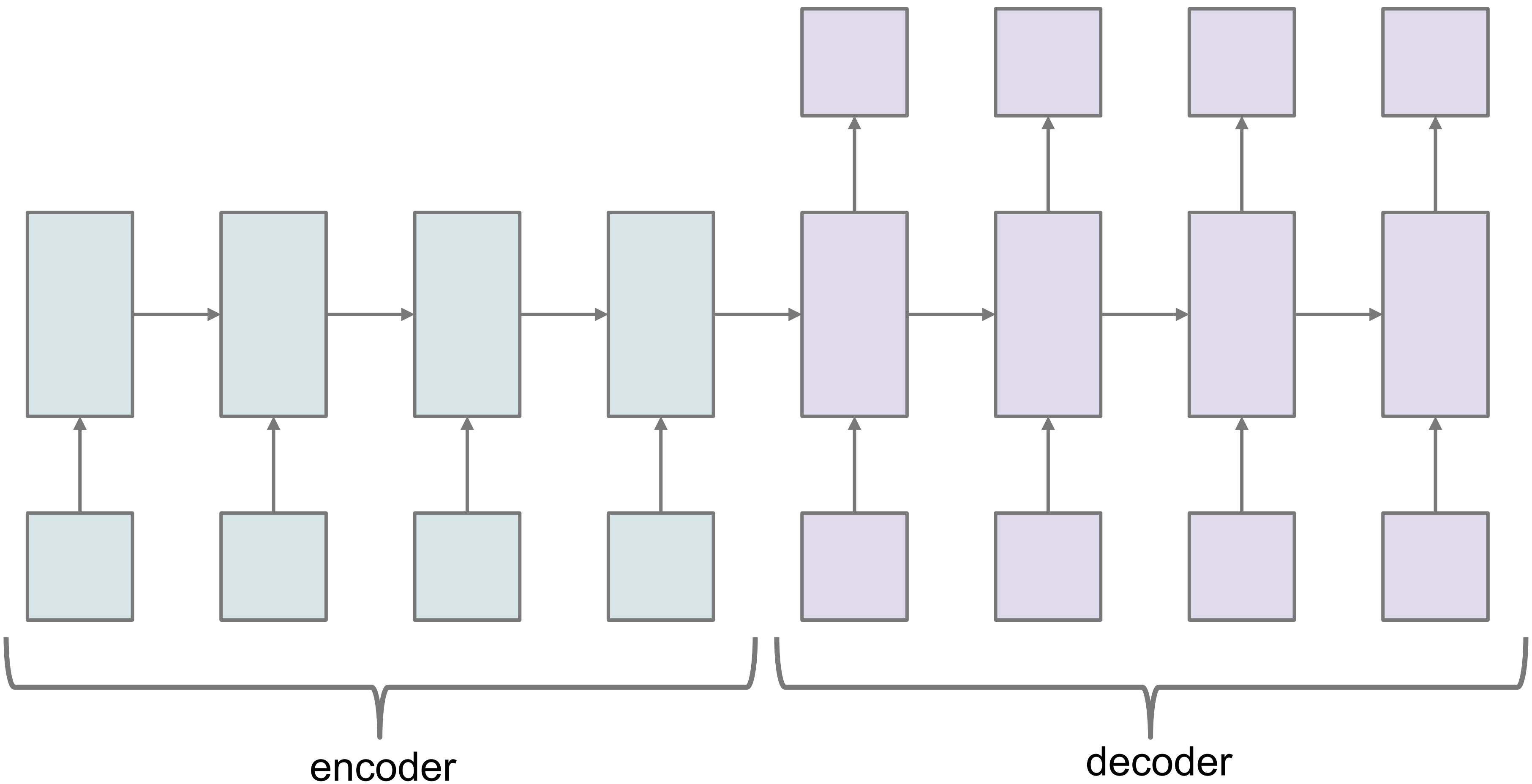
Sequence-to-Sequence & Attention

Sequence-to-sequence model

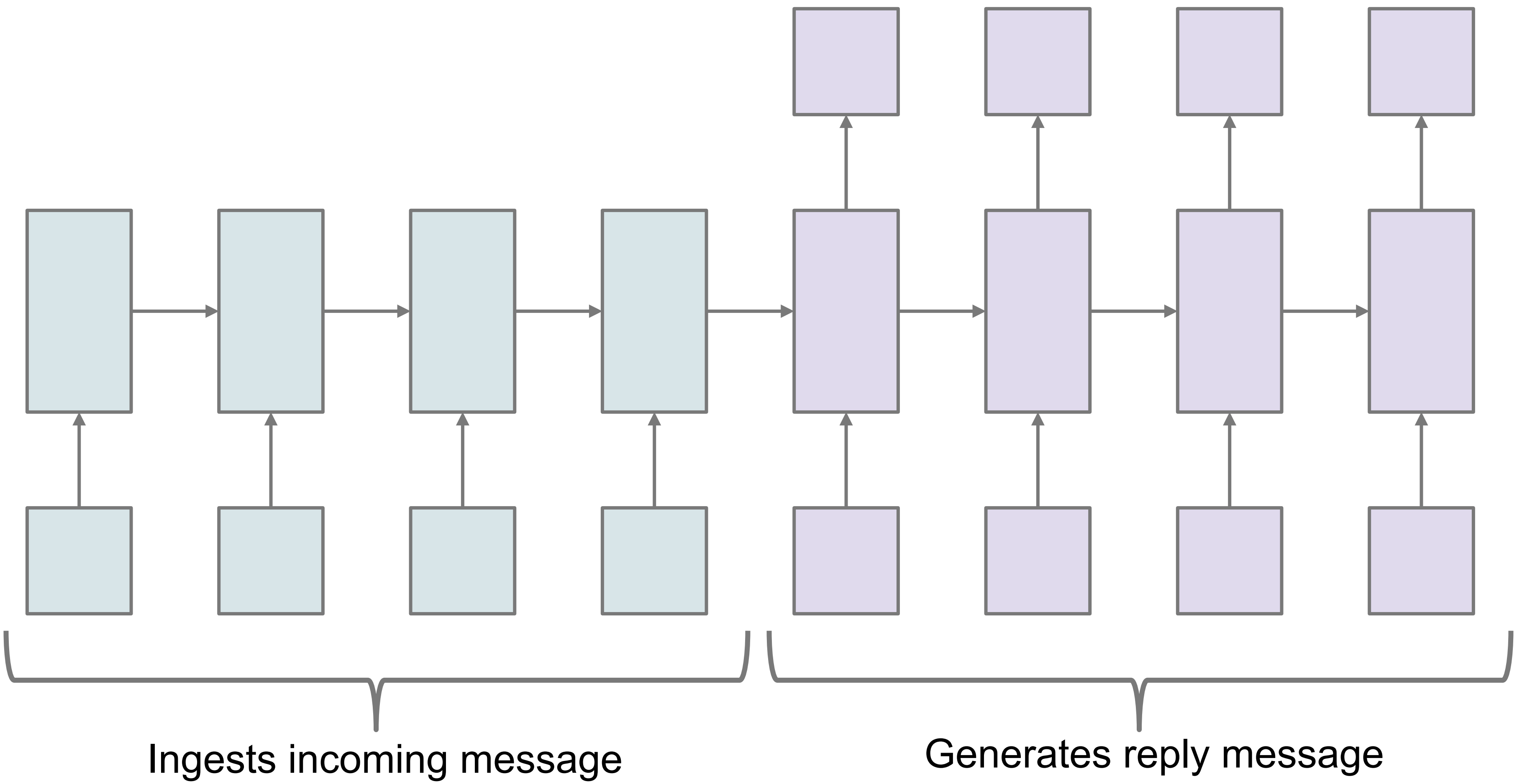


Sutskever et al, NIPS 2014

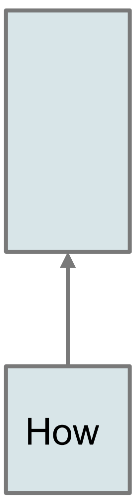
Sequence-to-sequence model



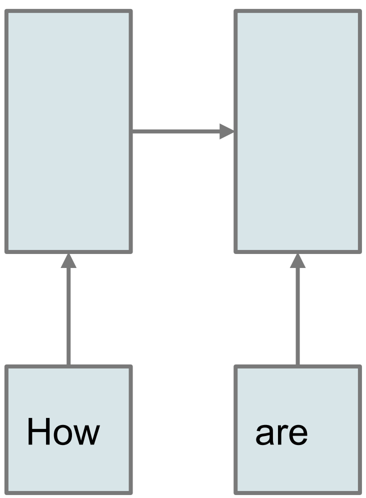
Sequence-to-sequence model



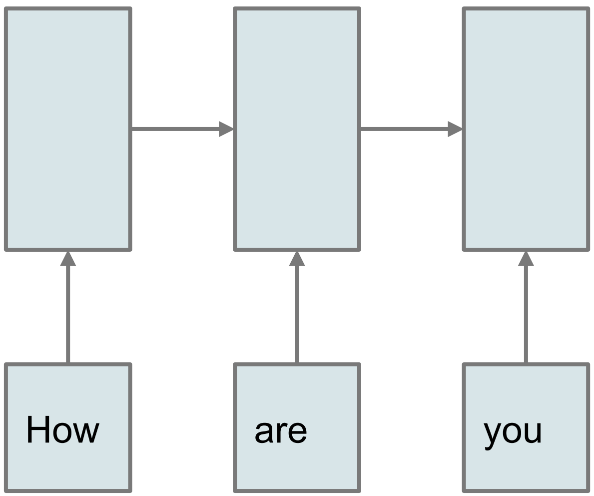
Reading a sequence of words into an RNN



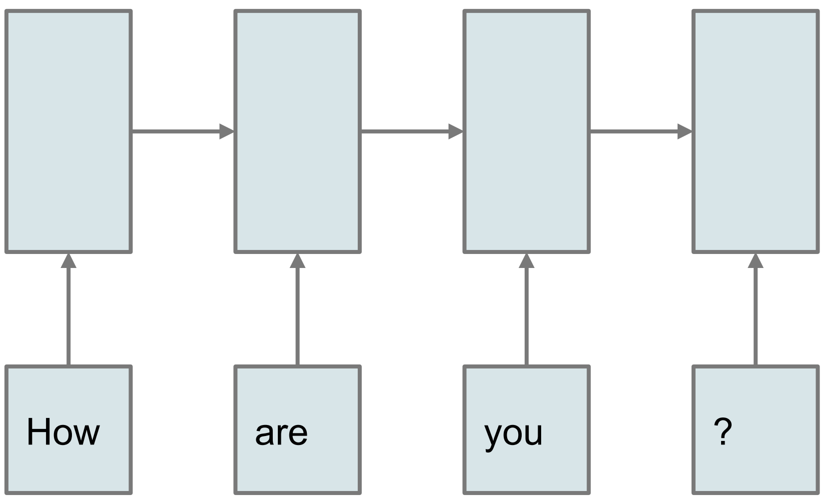
Reading a sequence of words into an RNN



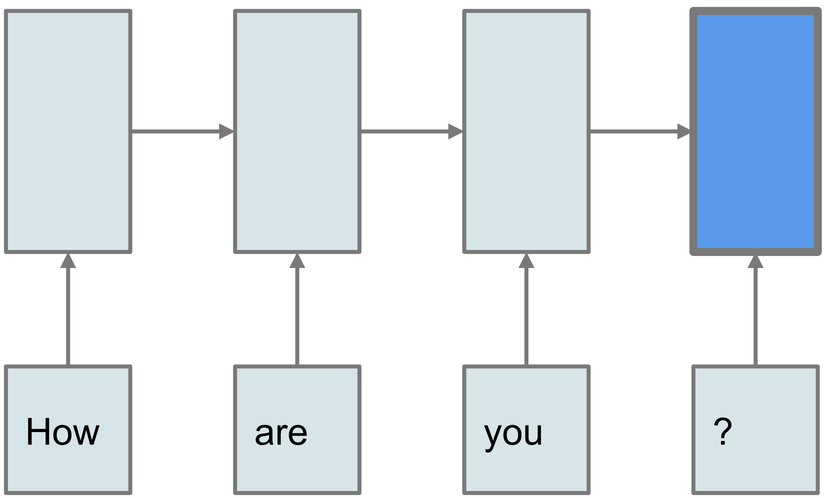
Reading a sequence of words into an RNN



Reading a sequence of words into an RNN

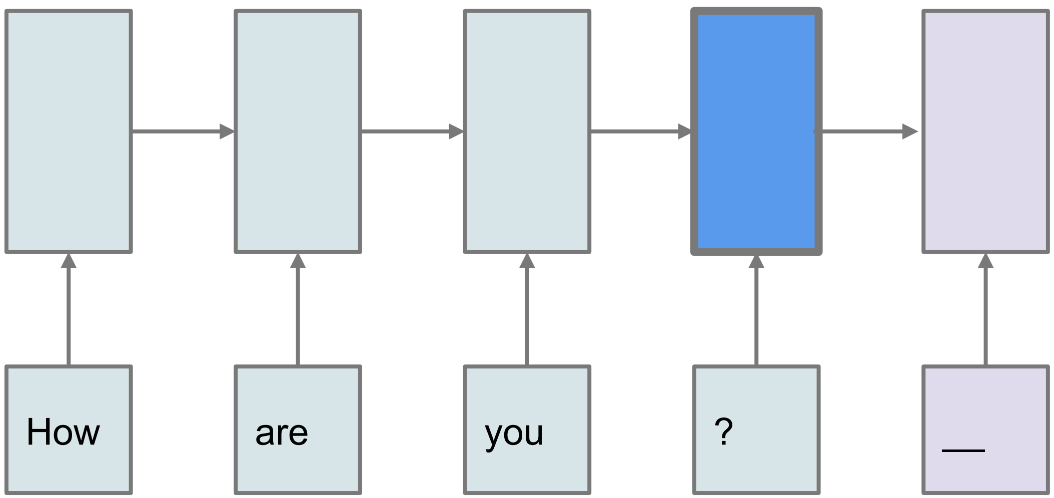


Encoder ingests the incoming message

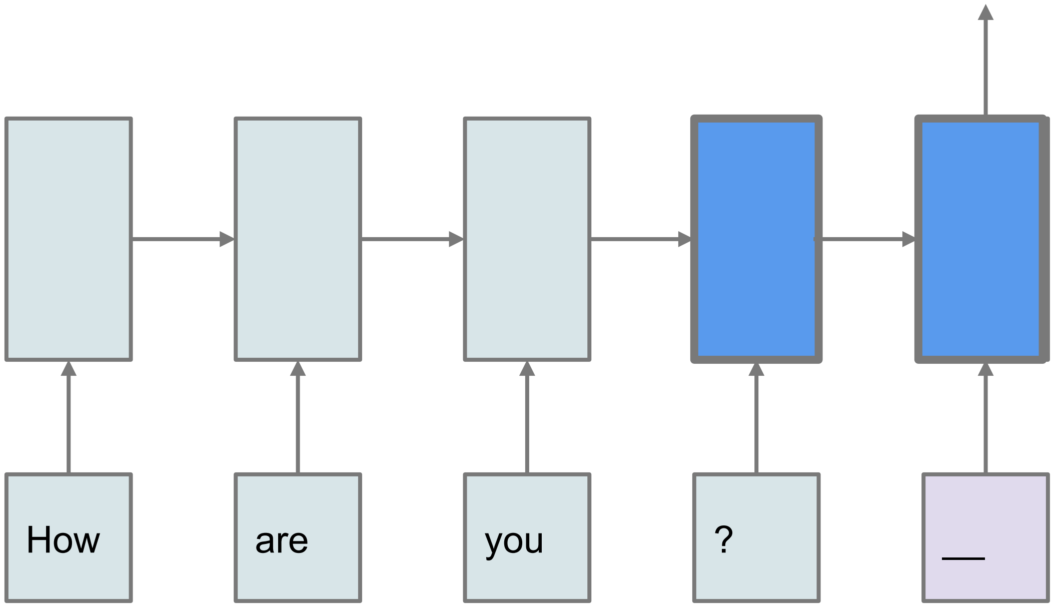


Internal state is a fixed length encoding of the message

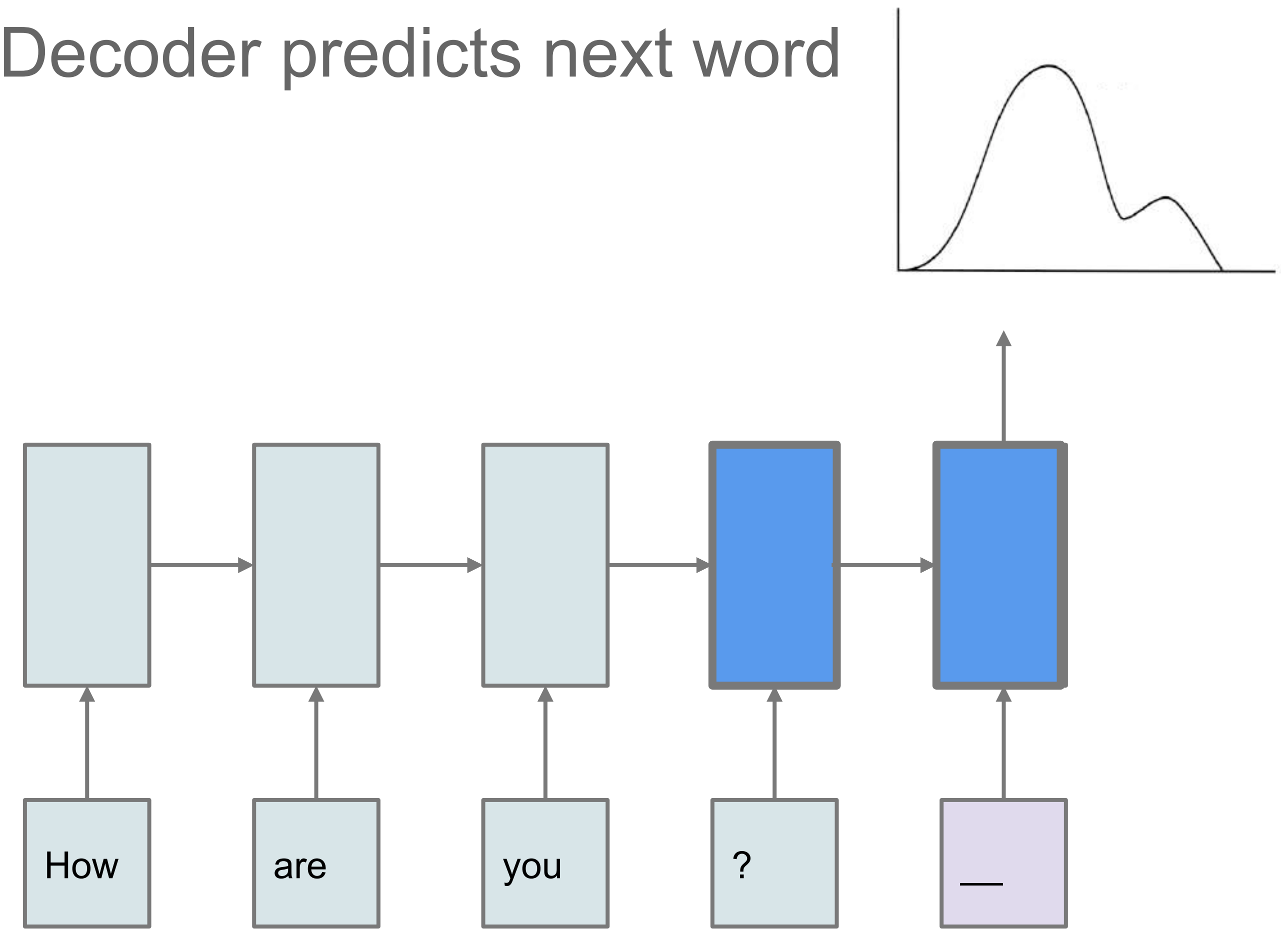
Decoder is initialized with final state of encoder



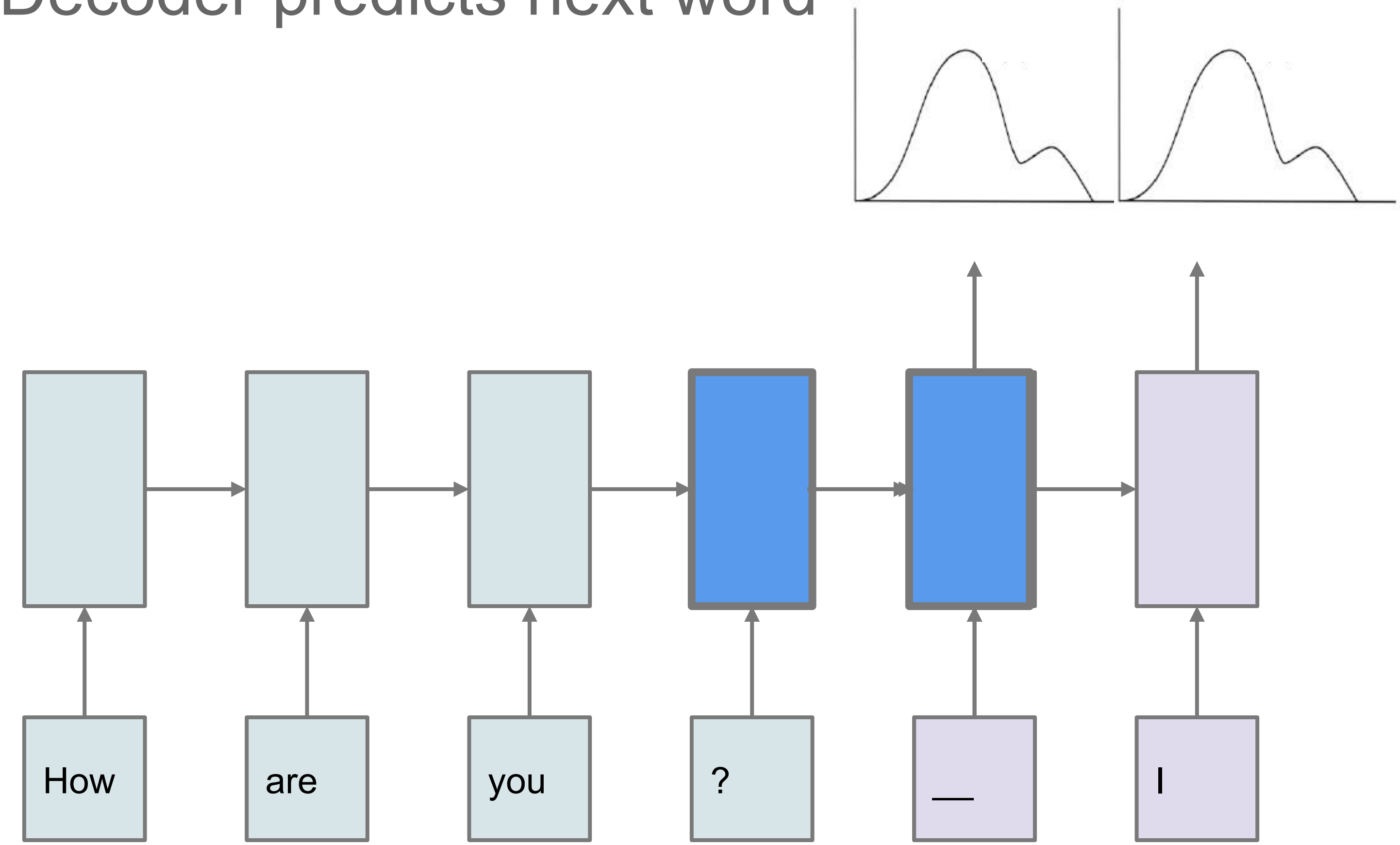
Decoder is initialized with final state of encoder



Decoder predicts next word



Decoder predicts next word



Encoder - decoder

- ➢ People observed that some heuristics made seq2seq better
  - ○ Example: reverse sequence, feed it twice
- ➢ Size of the encoding vector may be a bottleneck

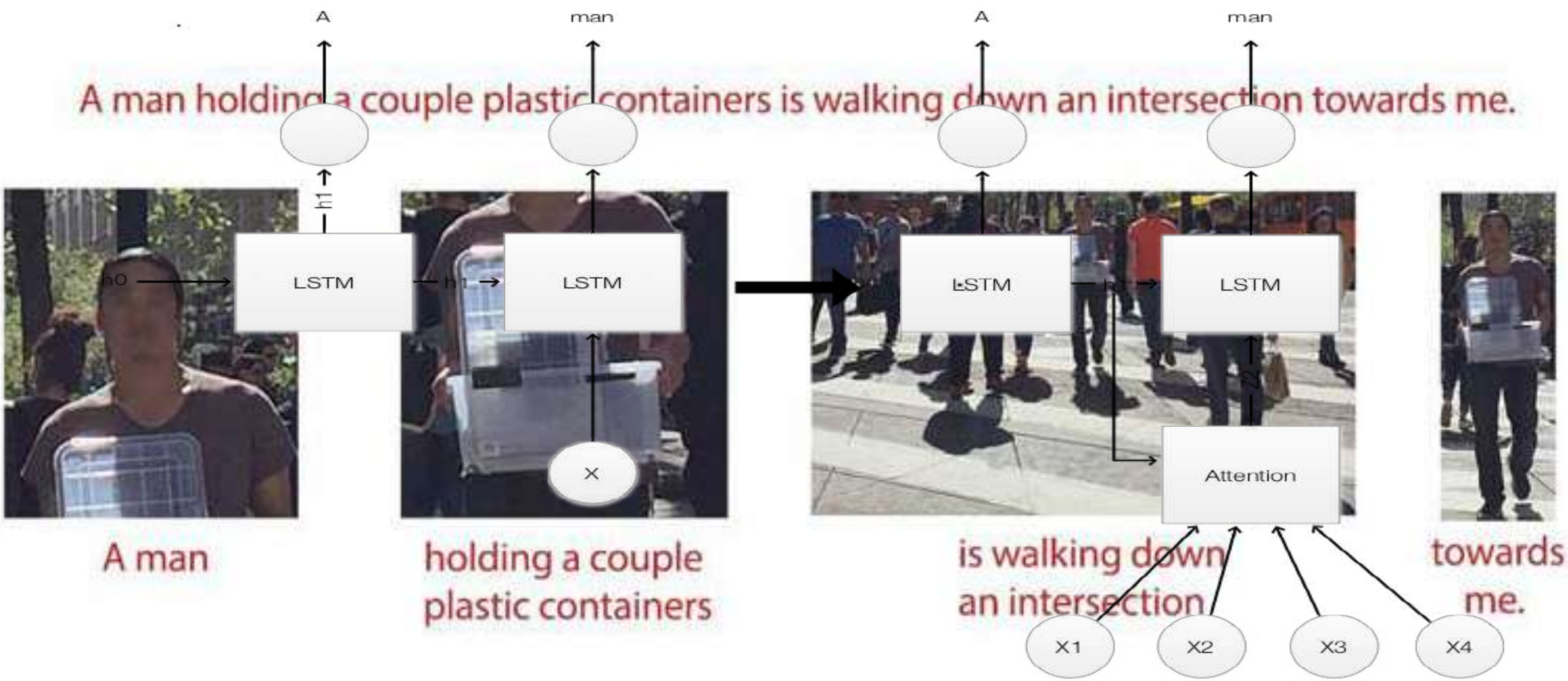
Attention

- ➢ One of the most “hot” / promising techniques for DNNs now
- ➢ Basic idea:
    - network decides which part of input it wants to look at in the next timestep
- ➢ Two variants: soft (differentiable) & hard (RL)

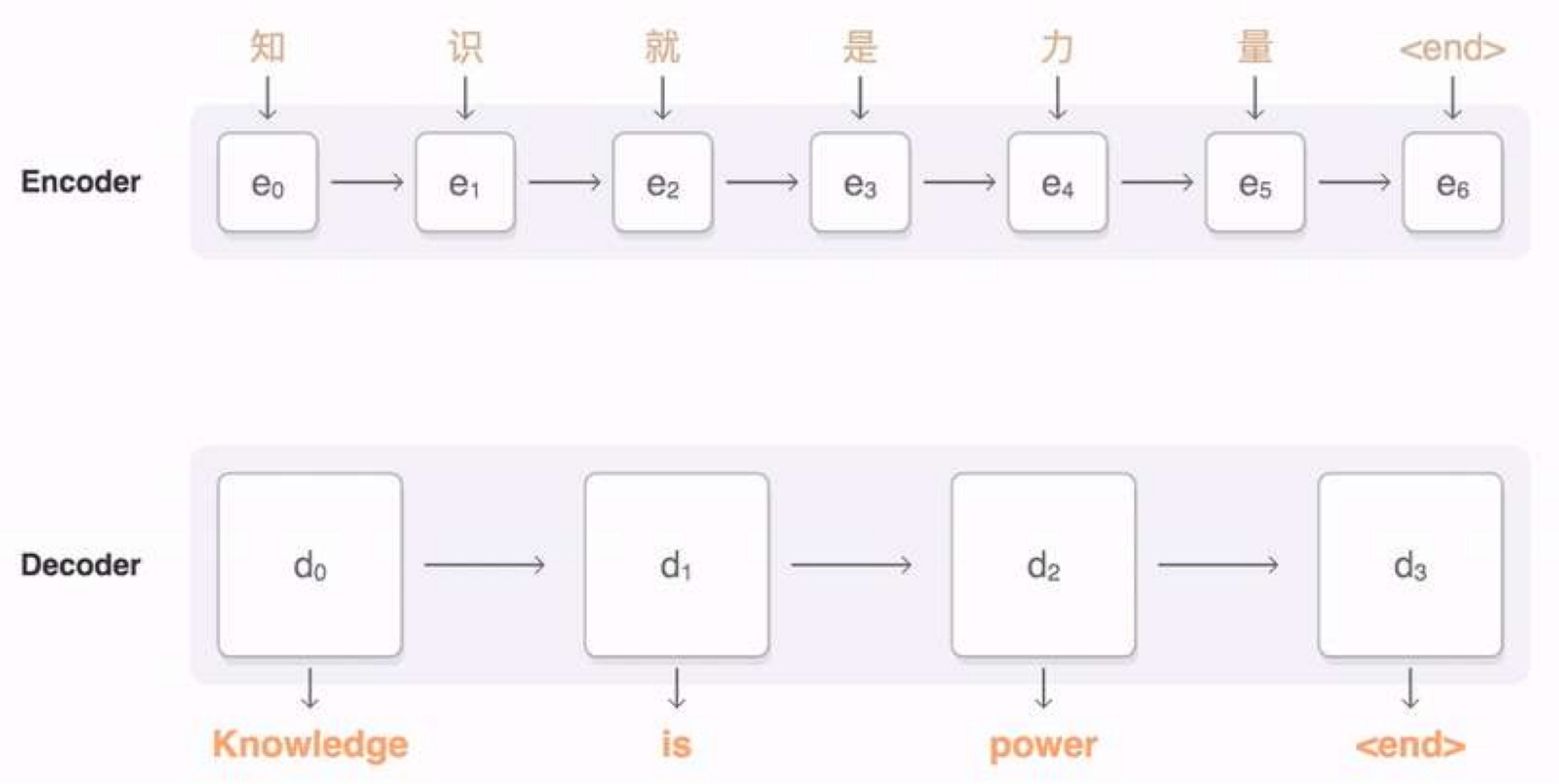
A woman is throwing a frisbee in a park. A dog is standing on a hardwood floor. A stop sign is on a road with a mountain in the background.

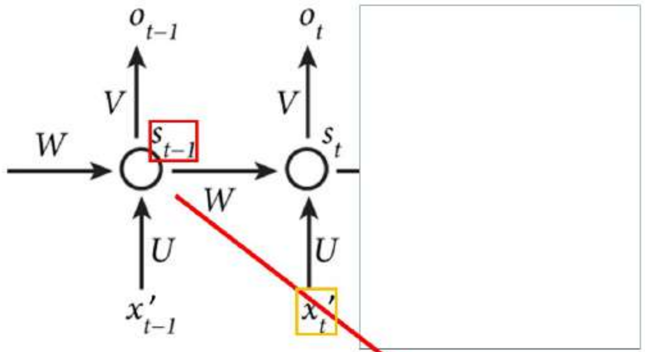
Attention

$$h_t = f(x, h_{t-1}) \qquad h_t = f(attention(x, h_{t-1}), h_{t-1})$$

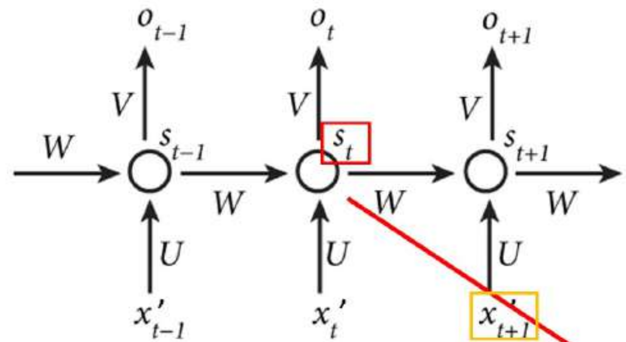


Attention in machine translation





$\boxed{x'_t} = f_{att}(\boxed{x_t}, \boxed{s_{t-1}})$



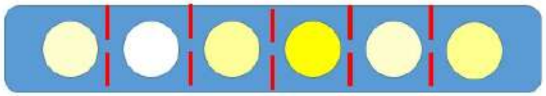
$\boxed{x'_{t+1}} = f_{att}(\boxed{x_{t+1}}, \boxed{s_t})$

$$x'_t = f_{att}(x_t, s_{t-1})$$

$$e_t = g(x_t, s_{t-1}; \theta) \quad (e_t, x_t \in \mathcal{R}^D)$$

$$\alpha_{tj} = \frac{exp(e_{tj})}{\sum_{i=1}^{D} exp(e_{ti})}$$

$$x'_t = \sum_{j=1}^{D} \alpha_{tj} x_{tj}$$



Hard vs Soft

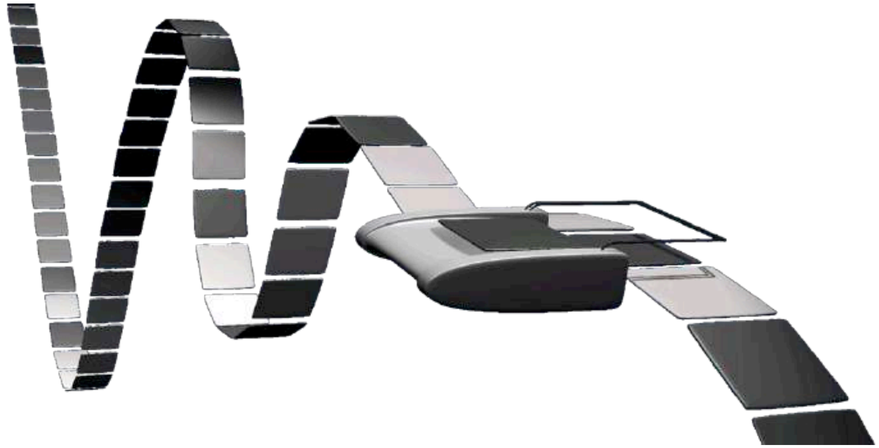
- Hard:
  - Discretely **sampling**
  - **Non-differential**
  - Learning by Reinforcement learning
- Soft:
  - Linear combination/Masking/Weights
  - **Differential**

Turing Machine & Neural Turing Machine

Turing Machine

- Theoretical, abstract machine
- Capable of simulating any computer algorithm
- Operates on *infinite tape* divided into cells (each in N states)
- Machine in one of M states
- Head can read/write to the tape and move left/right
- Finite table of instructions

Turing Machine - formally



Definition: A *Turing Machine* is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$,

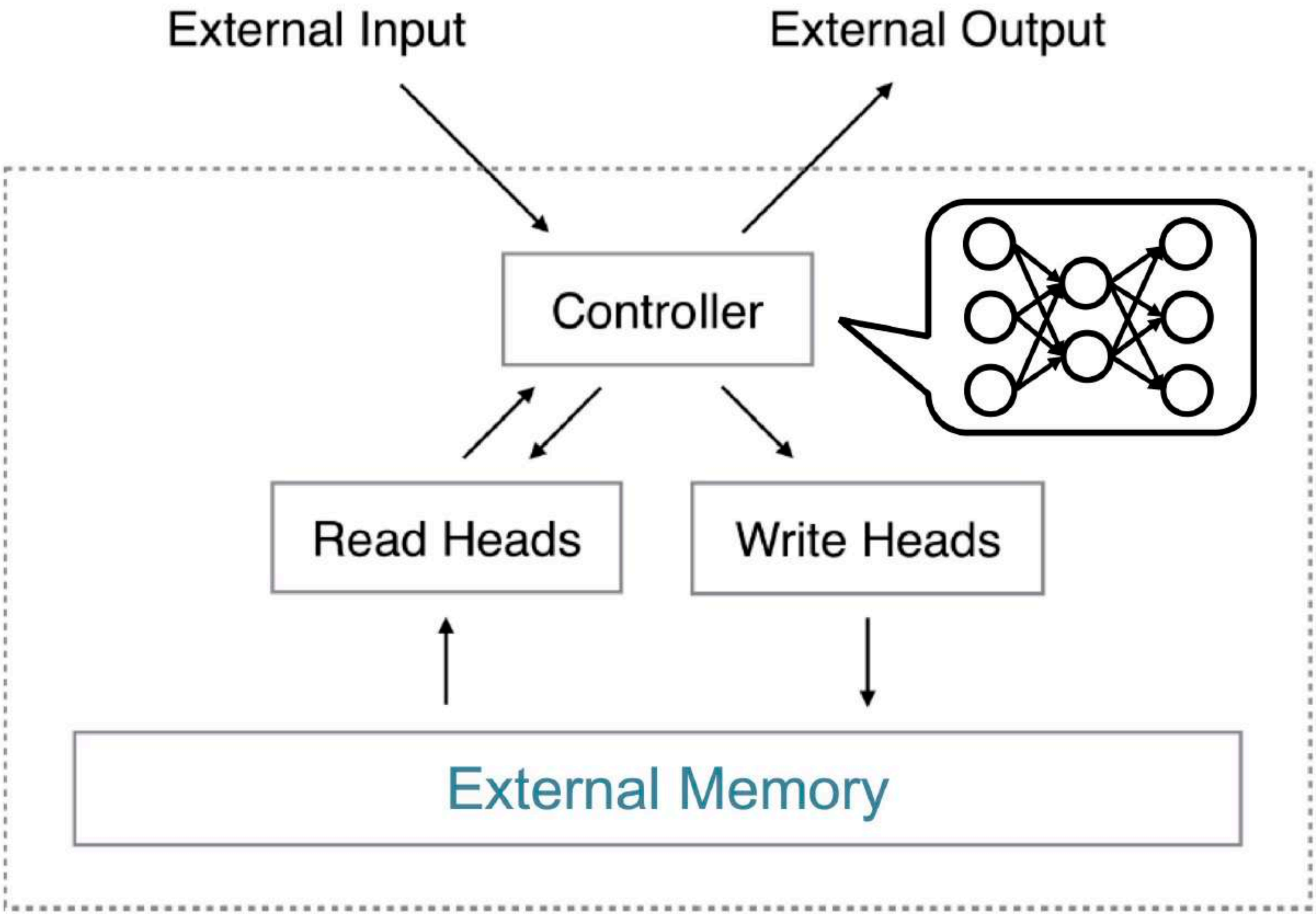
- Q is a finite set of *states*;
- Σ is an *input alphabet*, $\sqcup \notin \Sigma$;
- Γ is a *tape alphabet*, $\sqcup \in \Gamma$, and $\Sigma \subset \Gamma$;
- $\delta : Q \times \Gamma \mapsto (Q \cup \{q_{accept}, q_{reject}\}) \times \Gamma \times \{L, R\}$ is the *transition fun*
- $q_0 in Q$ is the *start state*;
- q_{accept}, q_{reject} are the *accept* and *reject* states, respectively.

Neural Turing Machine

Alex Graves et. al. in 2014.

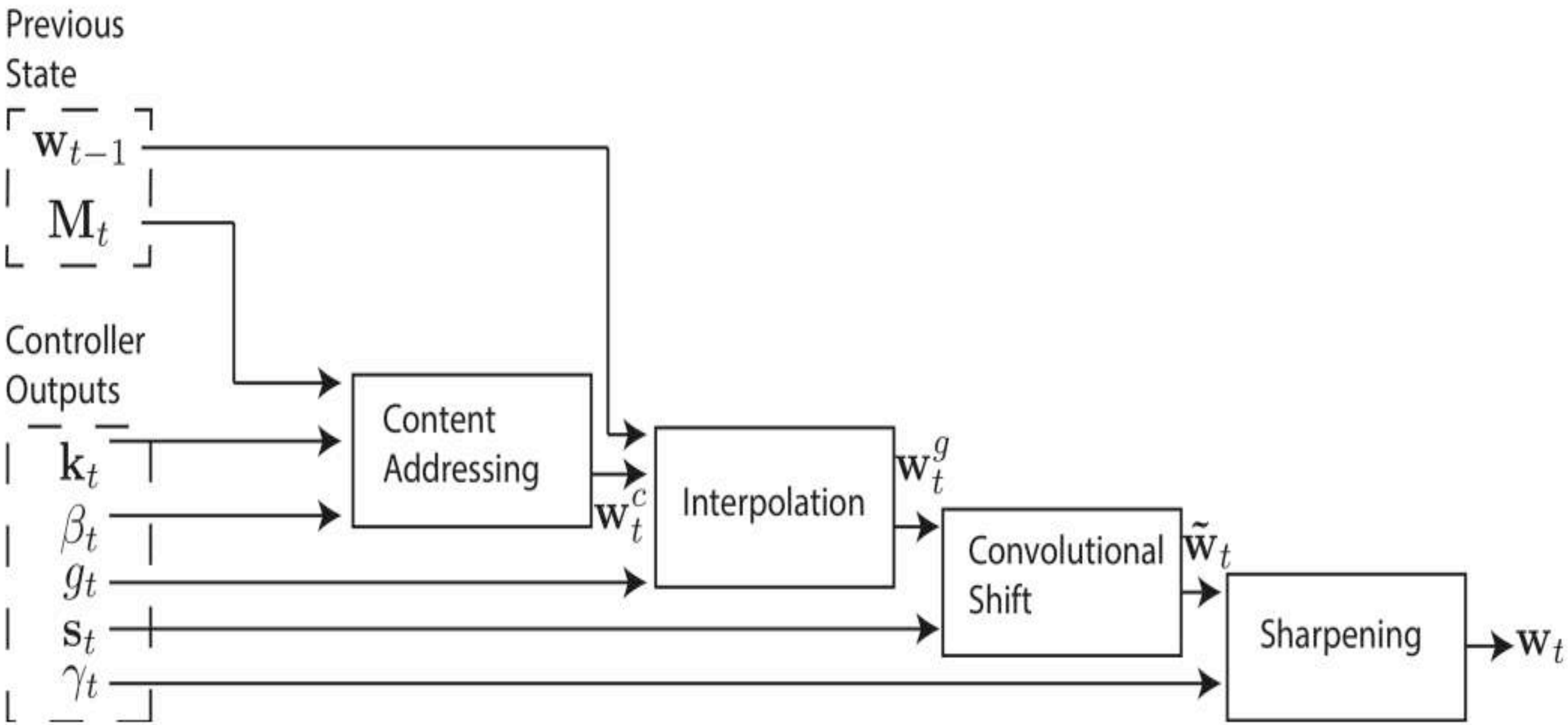
- ➢ Inspired by Turing Machine
- ➢ Fully differentiable
- ➢ Separation of the network into controller & memory
- ➢ Capable of solving simple tasks, like Copy, Reverse

NTM



External Input
External Output
Controller
Read Heads
Write Heads
External Memory

NTM - addressing mechanism



NTM - content addressing

$$w_t^c(i) \longleftarrow \frac{\exp\Big(\beta_t K\big[\mathbf{k}_t, \mathbf{M}_t(i)\big]\Big)}{\sum_j \exp\Big(\beta_t K\big[\mathbf{k}_t, \mathbf{M}_t(j)\big]\Big)}$$

Agenda

1. Introduction to Deep Neural Architectures
2. **Neural Random Access-Machines**
3. Hierarchical Attentive Memory
4. Applications: Smart Reply
5. Applications: Efficient Math Identities
6. Applications: Predicting Events From Sensor Data

Neural Random-Access Machines

Karol Kurach*

Marcin Andrychowicz*

Ilya Sutskever

Overview

- Neural architecture which can dereference pointers
- Can learn concepts such as “linked list” or “Binary Search Tree”
- Can interact with external modules
- Decides when to stop the computation

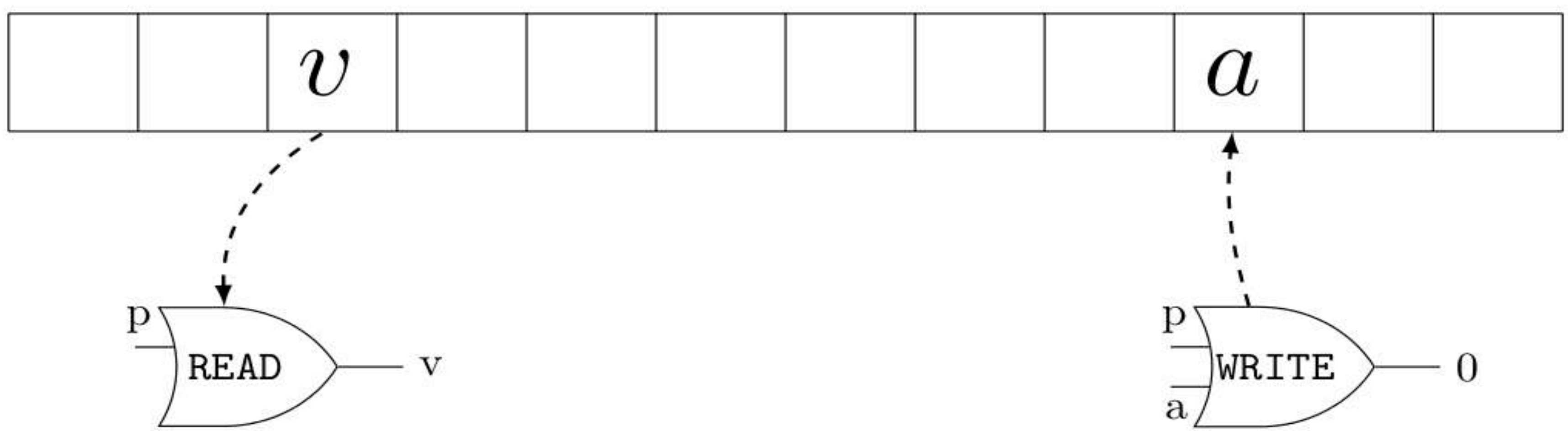
Google

Components

- External random-access memory M
- Fixed number of registers (distributions over Z_M)
- A fixed set of gates, e.g. addition modulo M
- LSTM controller deciding which operations should be applied at every timestep and which values should be stored in the registers

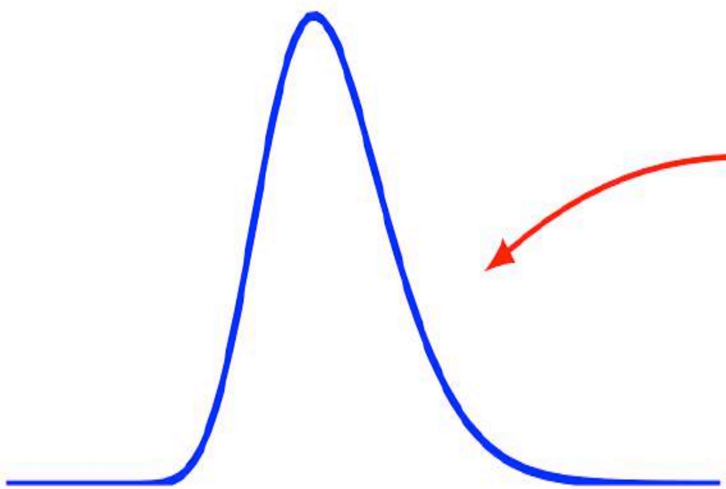
Memory

- Memory cells store distributions over Z_M
- Distributions over Z_M can be interpreted as *fuzzy pointers*
- Number of parameters independent of the memory size
- Interaction with using two special modules



Google

Fuzzy pointer



each memory cell and each register stores a probability distribution over M memory cells

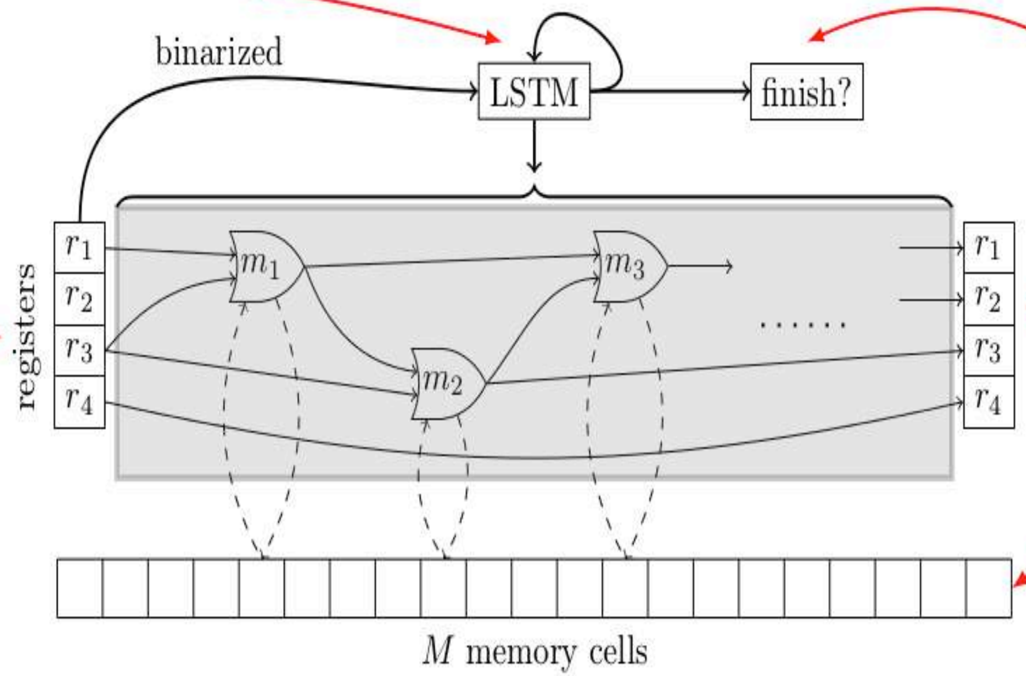
0	1	$\boxed{2}$	3	..	M-1
0	0.02	0.97	0.01	0	0

$\sum = 1$

Architecture overview

controls connections between gates at every timestep

LSTM decides when to finish the computation



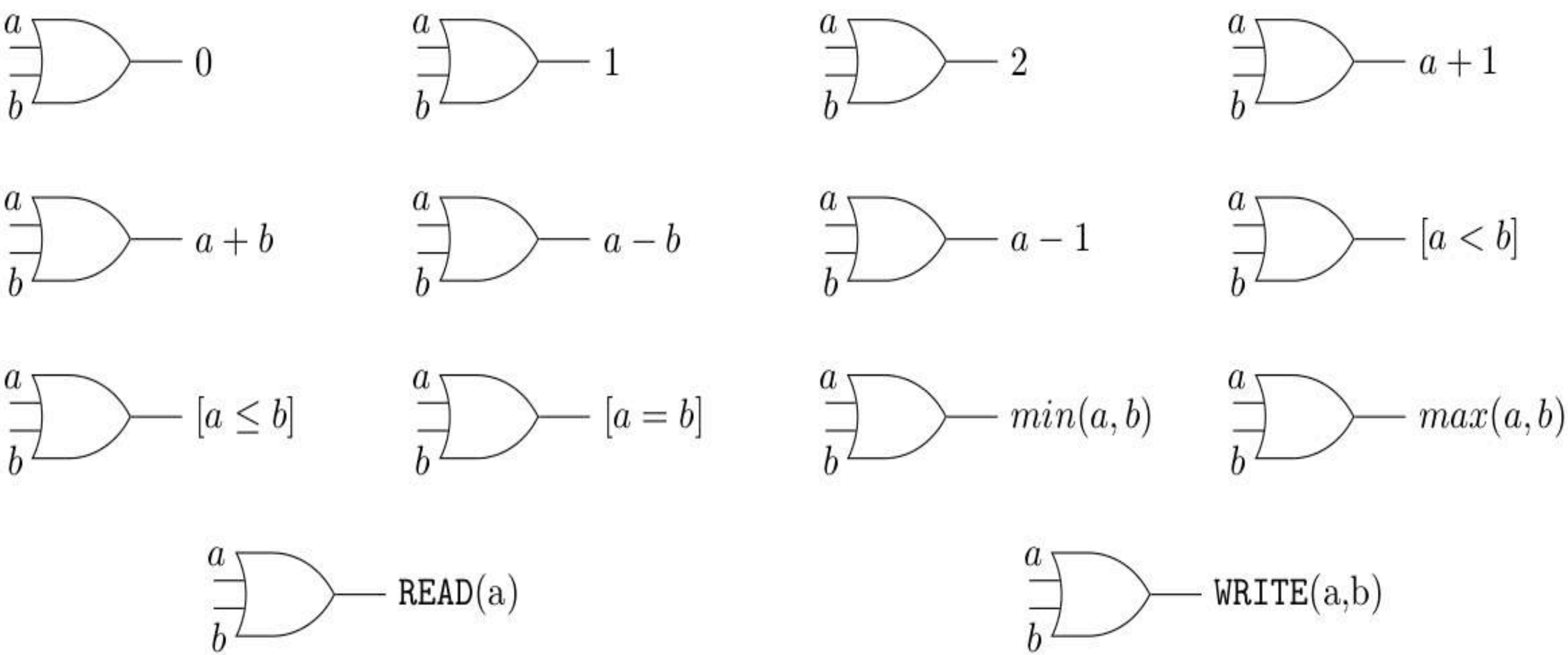
new values of registers

registers storing distributions over {0,1,...,M-1}

variable-size random-access memory

Google

Gates

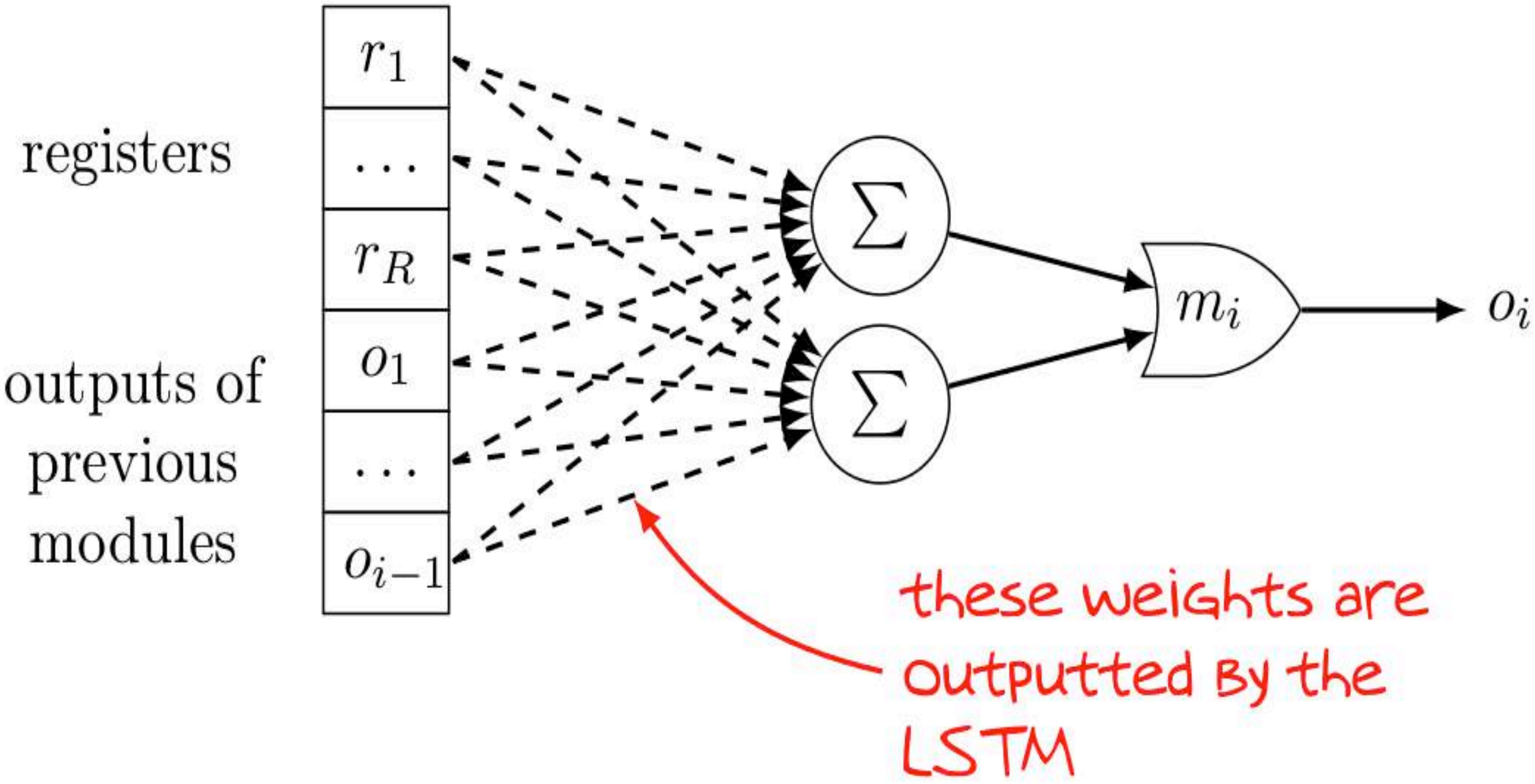


Gates

- But we have distributions, not integers….
- Natural extension:

$$\forall_{0 \leq c < M} \ \mathbb{P}\left(m_i(A, B) = c\right) = \sum_{0 \leq a, b < M} \mathbb{P}(A = a)\mathbb{P}(B = b)[m_i(a, b) = c]$$

Circuit generation



Training

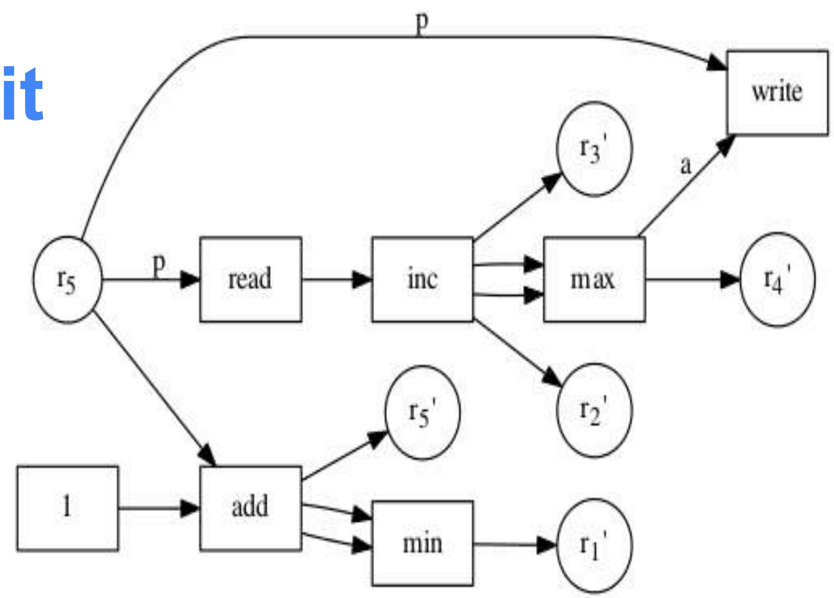
- Only input-output examples
- Log-likelihood cost function
- Gradient clipping both globally and during the backprop
- Curriculum
- Entropy bonus
- **Gradient noise**

Google

Experiments

Task	Train error	Generalization	Discretization
Access	0	perfect	perfect
Increment	0	perfect	perfect
Copy	0	perfect	perfect
Reverse	0	perfect	perfect
Swap	0	perfect	perfect
Apply the given permutation	0	almost perfect	perfect
Find the k-th element on a list	0	strong	hurts performance
Find the given value on a list	0	weak	hurts performance
Merge 2 sorted arrays	1%	weak	hurts performance
Follow the given path in a BST	0.3%	strong	hurts performance

Example circuit



Step	Memory cells											Registers				
	0	1	2	3	4	5	6	7	8	9	10	r_1	r_2	r_3	r_4	r_5
1	1	11	3	8	1	2	9	8	5	3	0	0	0	0	0	0
2	2	11	3	8	1	2	9	8	5	3	0	1	2	2	2	1
3	2	12	3	8	1	2	9	8	5	3	0	2	12	12	12	2
..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..
11	2	12	4	9	2	3	10	9	6	4	0	10	4	4	4	10

Summary

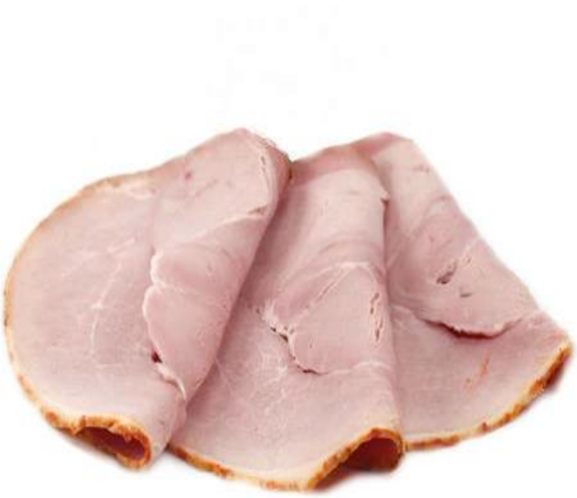
- First neural network that use pointers
- Can use external modules (gates)
- Can interact with external modules
- Decides when to stop the computation

Agenda

1. Introduction to Deep Neural Architectures
2. Neural Random Access-Machines
3. **Hierarchical Attentive Memory**
4. Applications: Smart Reply
5. Applications: Efficient Math Identities
6. Applications: Predicting Events From Sensor Data

Hierarchical Attentive Memory

Karol Kurach*
Marcin Andrychowicz*

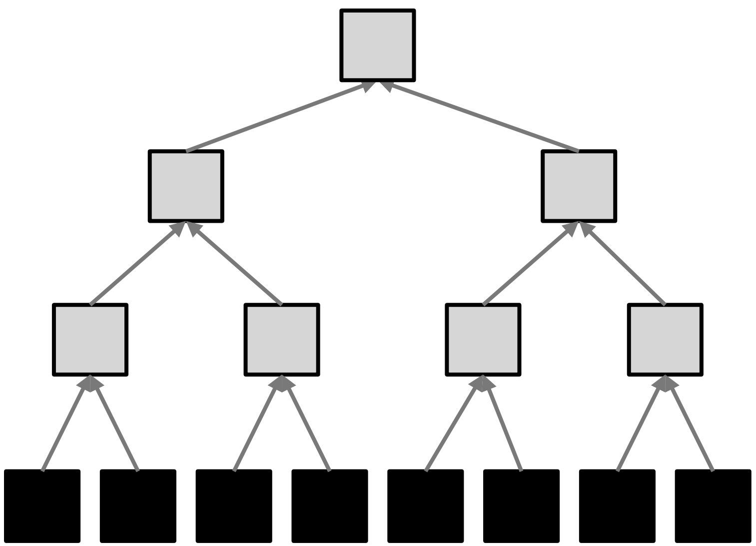


Motivation:

- Recently there have been proposed many memory architectures
- But most of them are not very efficient - copying a sequence of length n requires $O(n^2)$ operations:
- Aim: design an efficient memory architecture
- Means: hierarchical attention

Hierarchical Attentive Memory (HAM)

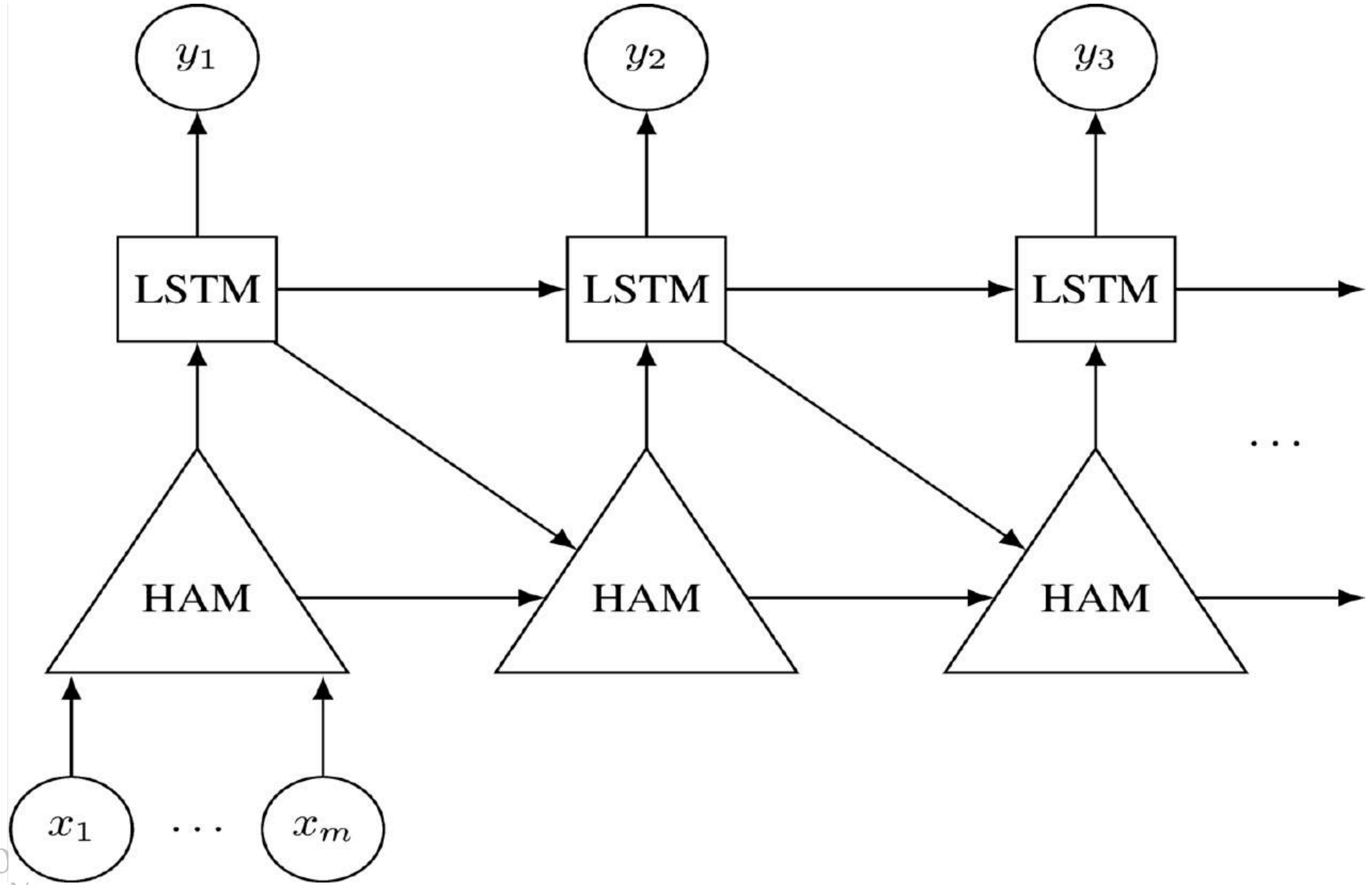
- Memory access in O(log n)
- Memory is structured as a binary tree



Inner nodes: auxiliary hidden values "summarizing" memory cells beneath

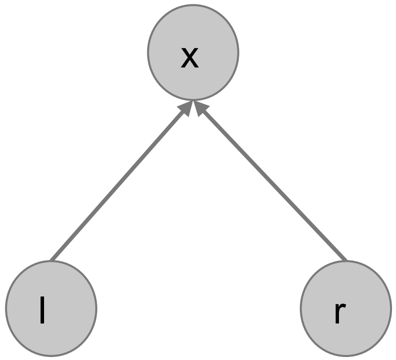
Leafs: memory cells

LSTM + HAM

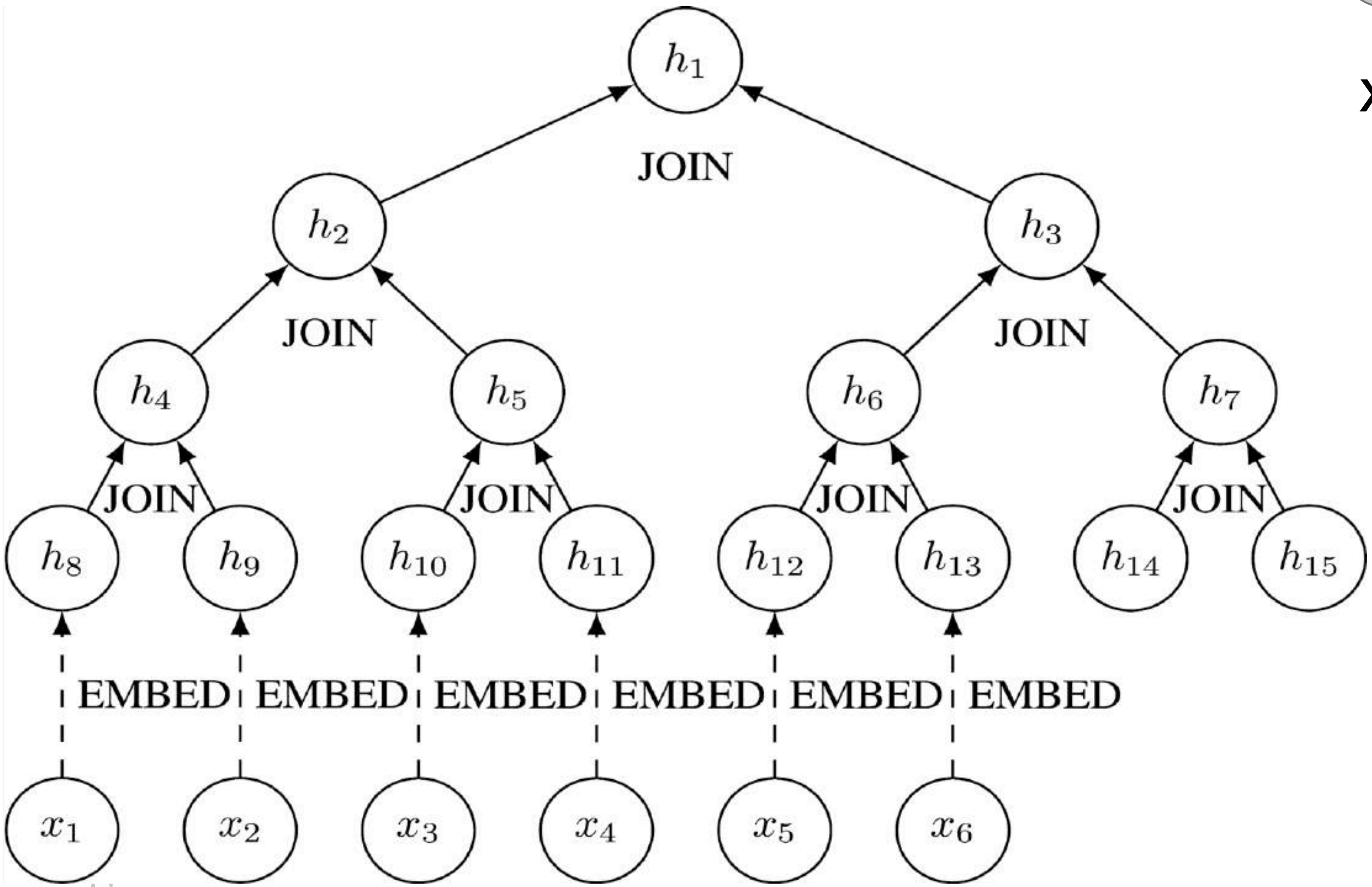


Goo

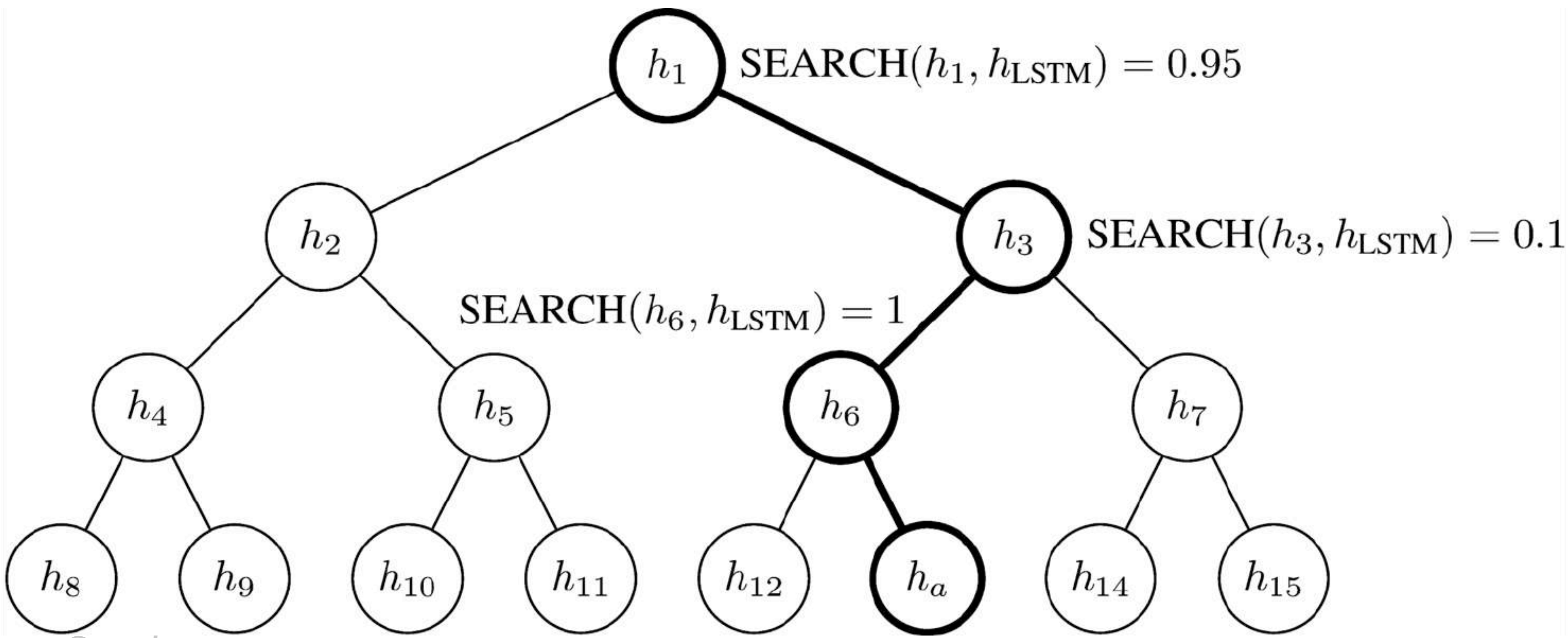
1. Initialization



x=JOIN(l,r)

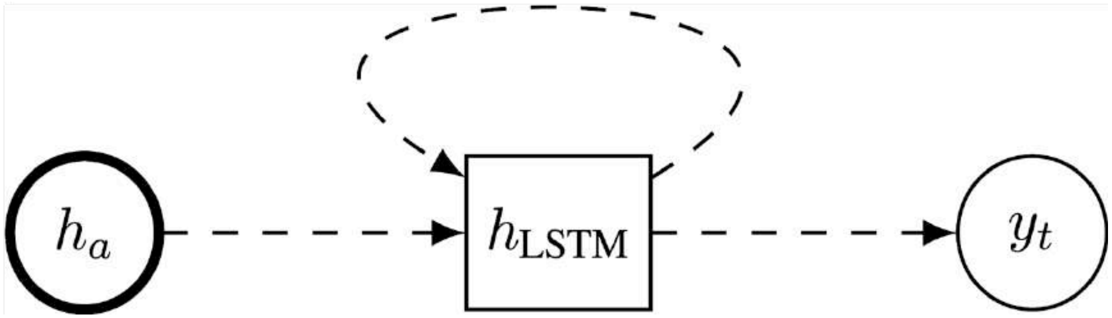


2. Attention phase



Google

3. Output phase



4. Update phase

a) **modify the attended leaf**

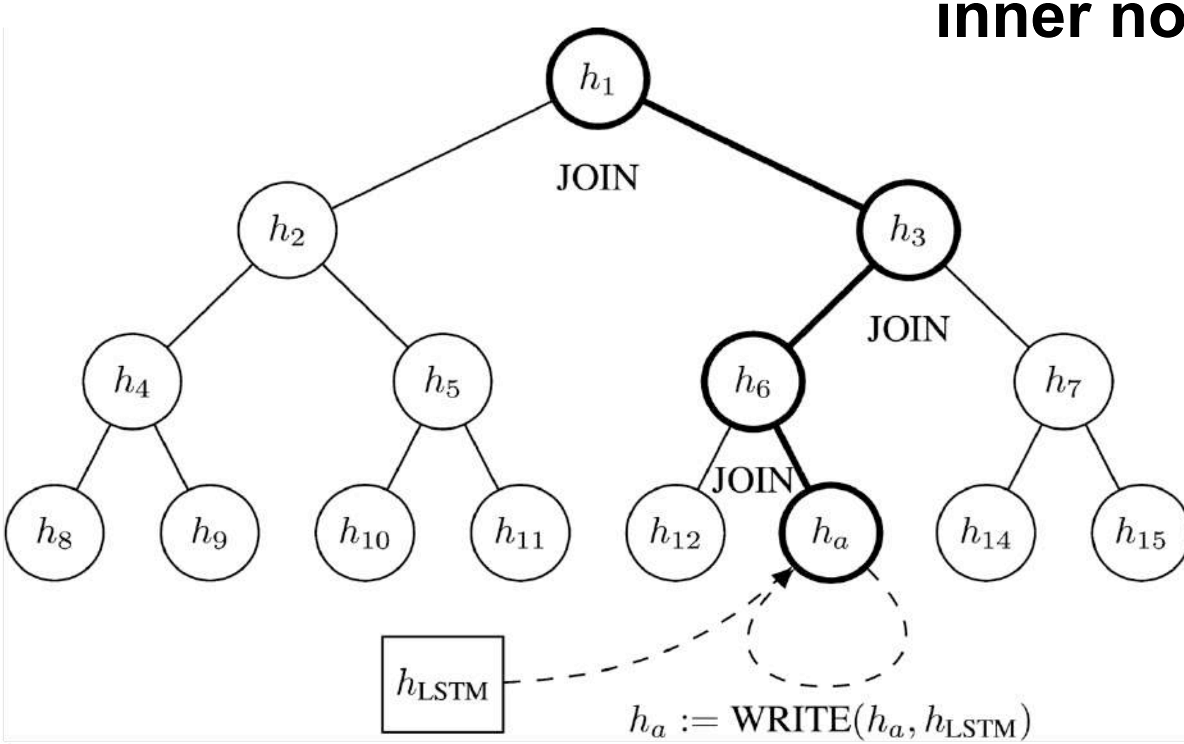
Highway Networks-style write:

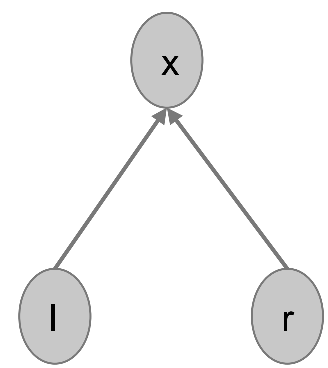
$WRITE(h_a, h_{LSTM}) =$

$T(h_a, h_{LSTM}) \cdot H(h_a, h_{LSTM}) +$

$[1\text{-}T(h_a, h_{LSTM})] \cdot h_a$

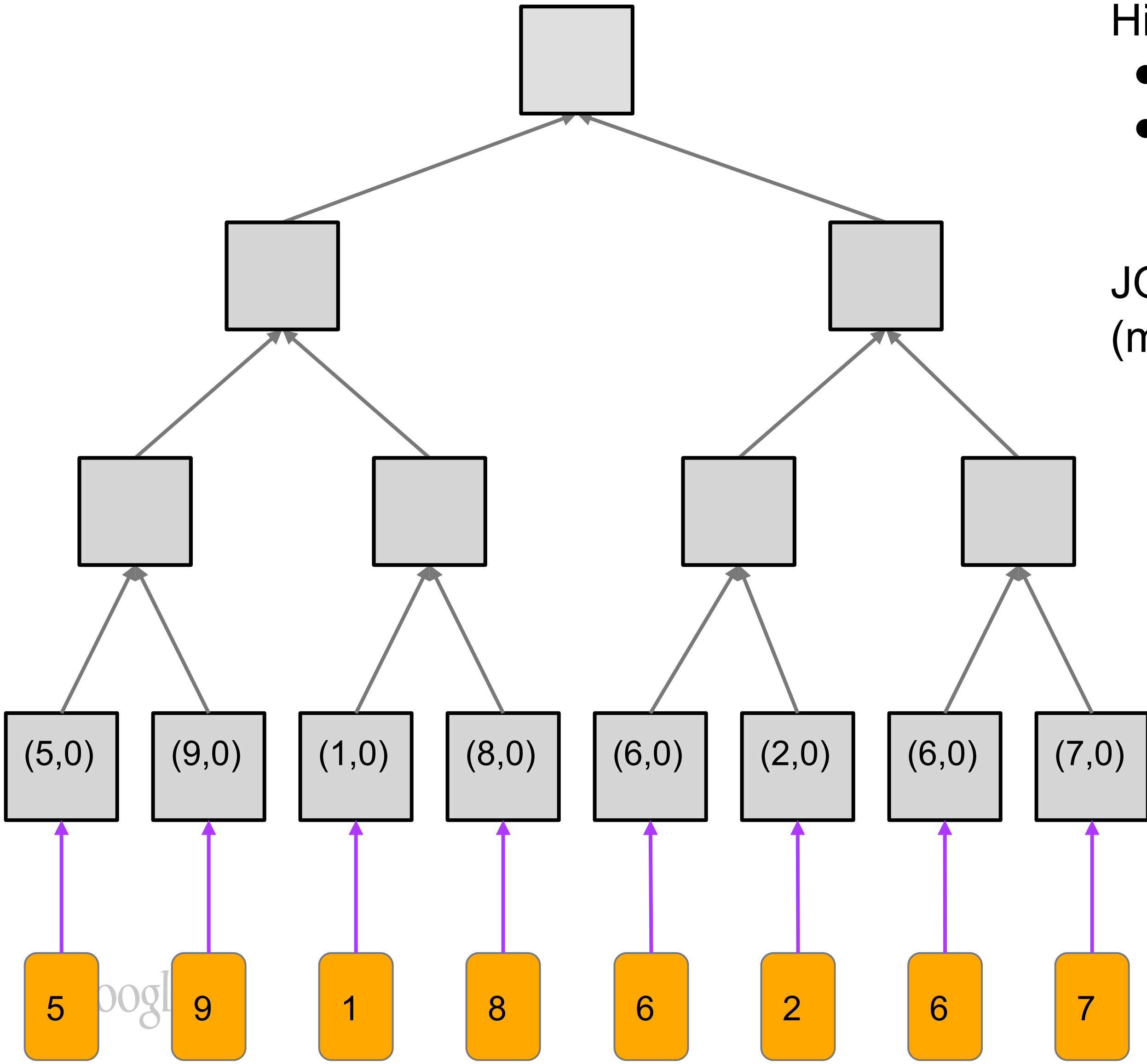
b) update the values in the inner nodes





x=JOIN(l,r)

Example: sorting



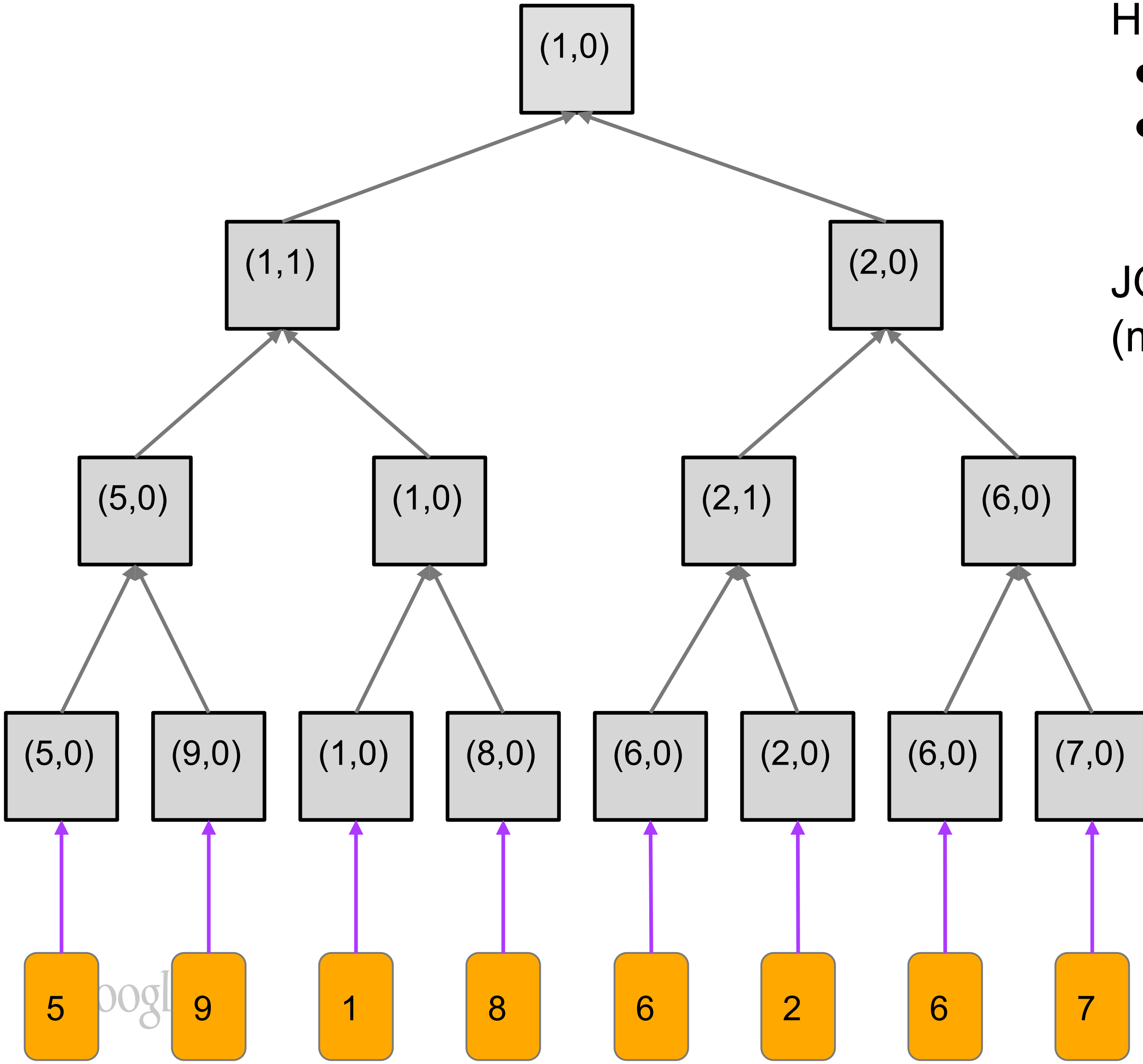
Hidden values:

- Minimum in the subtree
- Is the minimum in the right subtree?

$\text{JOIN}((x_1,r_1),(x_2,r_2)) = (\min(x_1,x_2), [x_1 > x_2])$

$\text{EMBED}(x) = (x,0)$

Example: sorting



Hidden values:

- Minimum in the subtree
- Is the minimum in the right subtree?

$\text{JOIN}((x_1,r_1),(x_2,r_2)) =$
$(\min(x_1,x_2), [x_1 > x_2])$

EMBED(x) = (x,0)

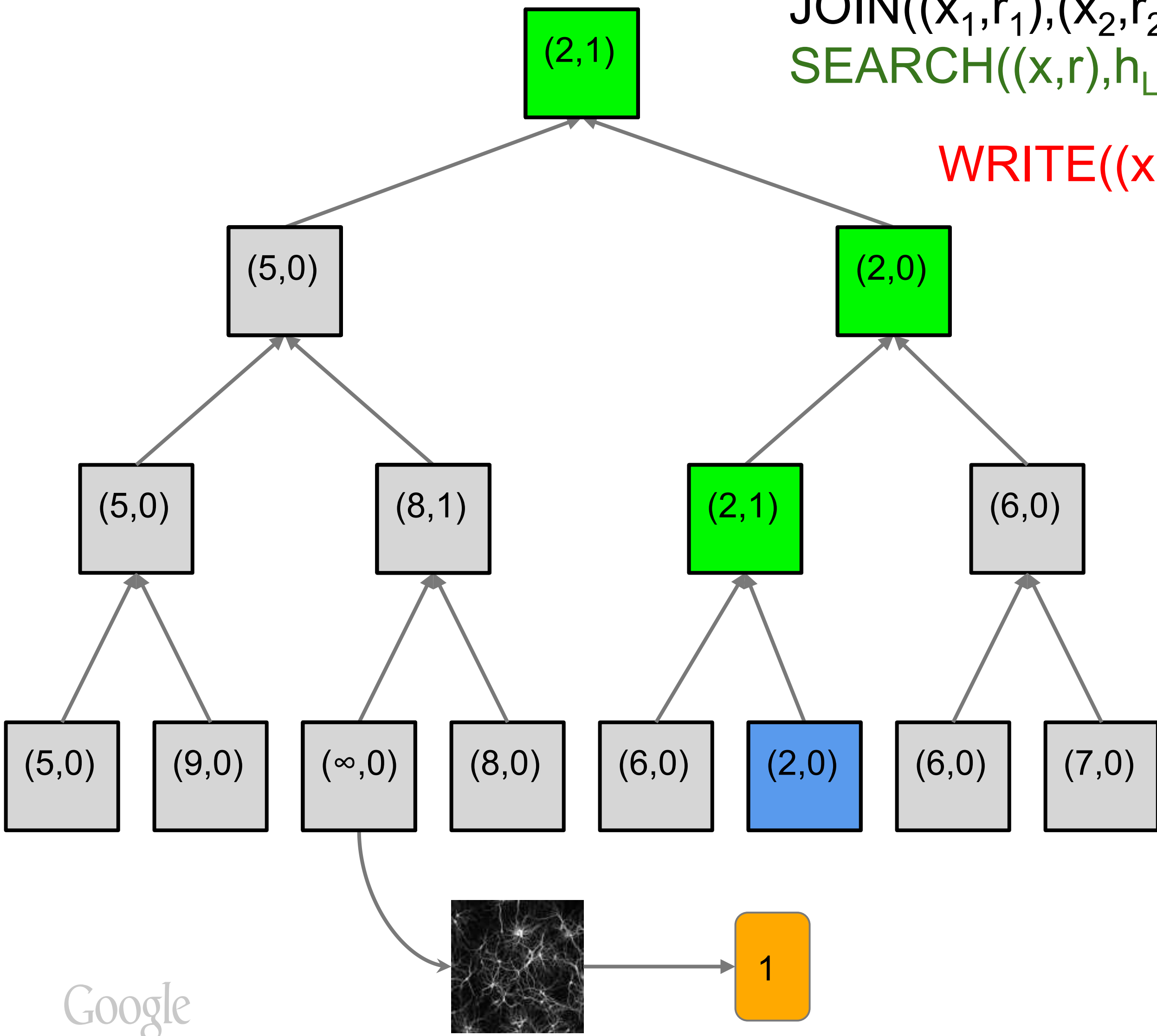
Example: sorting

EMBED(x) = (x,0)

$\mathrm{JOIN}((x_1,r_1),(x_2,r_2)) = (\min(x_1,x_2), [x_1 > x_2])$

$\mathrm{SEARCH}((x,r),h_{\mathrm{LSTM}}) = r$

$\mathrm{WRITE}((x,r),h_{\mathrm{LSTM}})=(\infty,0)$



Google

Properties of HAM

- Number of parameters independent of the memory size
- Memory access complexity: O(log n)
- Supports some operations impossible for normal attention, e.g. extracting the minimum

Training

- BPTT
- REINFORCE with discounted returns for the sampling nodes
- Reward: log-probability → percentage of correctly predicted bits
- Entropy bonus: $\alpha H(p) \rightarrow -\alpha / H(p)$:
  - Forces the model to give non-zero probability to **every** leaf
- Curriculum: [1,4], [1,8], [1,16]...

Experiments

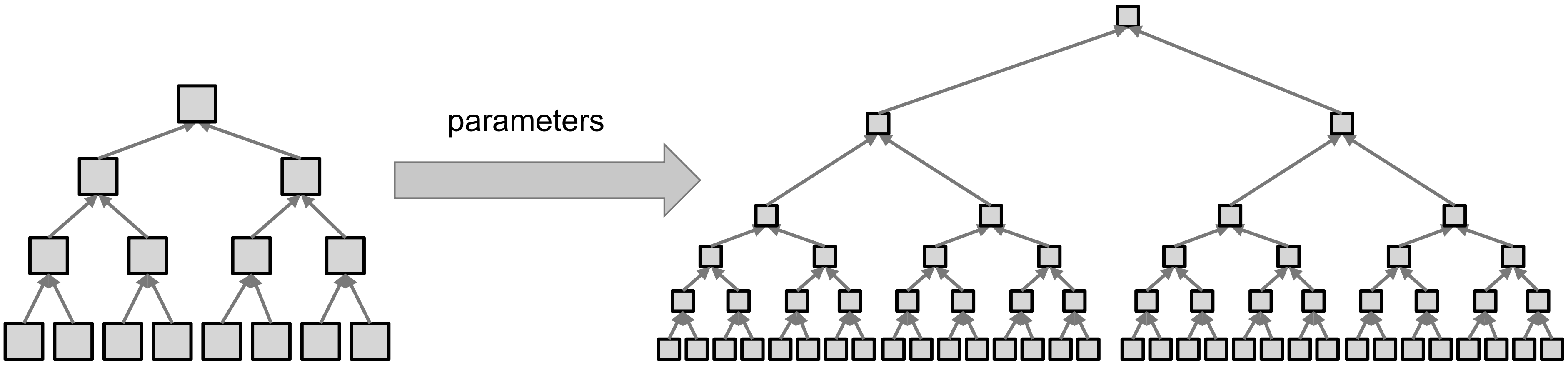
- LSTM+HAM:
  - Reverse
  - Search (binary)
  - Merge
  - Sort
  - Long binary addition
- Raw HAM:
  - Stack
  - FIFO Queue
  - Priority Queue

Experiments

	LSTM	LSTM+A	LSTM+HAM
Reverse	73%	0%	**0%**
Search	62%	0.04%	**0.12%**
Merge	88%	16%	**0%**
Sort	99%	25%	**0.04%**
Add	39%	0%	**0%**
Stack	N/A	N/A	**0%**
FIFO Queue	N/A	N/A	**0%**
Priority Queue	N/A	N/A	**0.08%**

Error rates are percentages of incorrect **output sequences**.

Generalization



training model (n=32)

testing model (n=128)

Generalization results

	LSTM	LSTM+A	LSTM+HAM
Reverse	100%	100%	**0%**
Search	89%	0.52%	**1.68%**
Merge	100%	100%	**2.48%**
Sort	100%	100%	**0.24%**
Add	100%	100%	**100%**
Stack	N/A	N/A	**0%**
FIFO Queue	N/A	N/A	**0%**
Priority Queue	N/A	N/A	**0.2%**

Error rates are percentages of incorrect **output sequences**.

HAM vs. content-based attention:

- Pros:
  - It is more efficient
  - It supports some operations impossible for content-based attention, e.g. extracting the minimum
  - It generalizes better
- Cons:
  - Performing associative recall may be difficult
- After all there is no need to choose: you can use both

Conclusion

- Efficient memory, access in O(log n)
- Possible drop-in replacement for other data structures
- Good generalization (first to learn sorting that generalizes)

Agenda

1. Introduction to Deep Neural Architectures
2. Neural Random Access-Machines
3. Hierarchical Attentive Memory
4. **Applications: Smart Reply**
5. Applications: Efficient Math Identities
6. Applications: Predicting Events From Sensor Data

Smart Reply: Automated Response Suggestion for Email

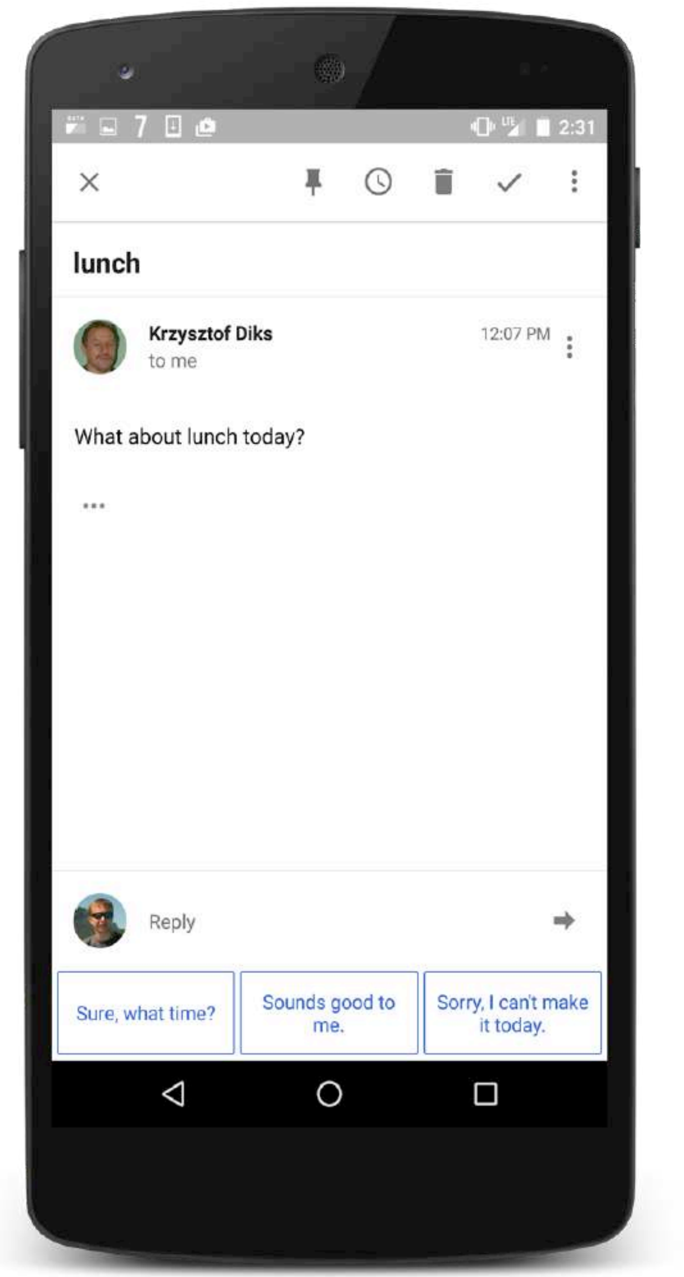
Anjuli Kannan*, Karol Kurach*, Sujith Ravi*, Tobias Kaufmann*, Andrew Tomkins, Balint Miklos, Greg Corrado, Marina Ganea, Laszlo Lukacs, Peter Young, Vivek Ramavajjala

*equal contribution

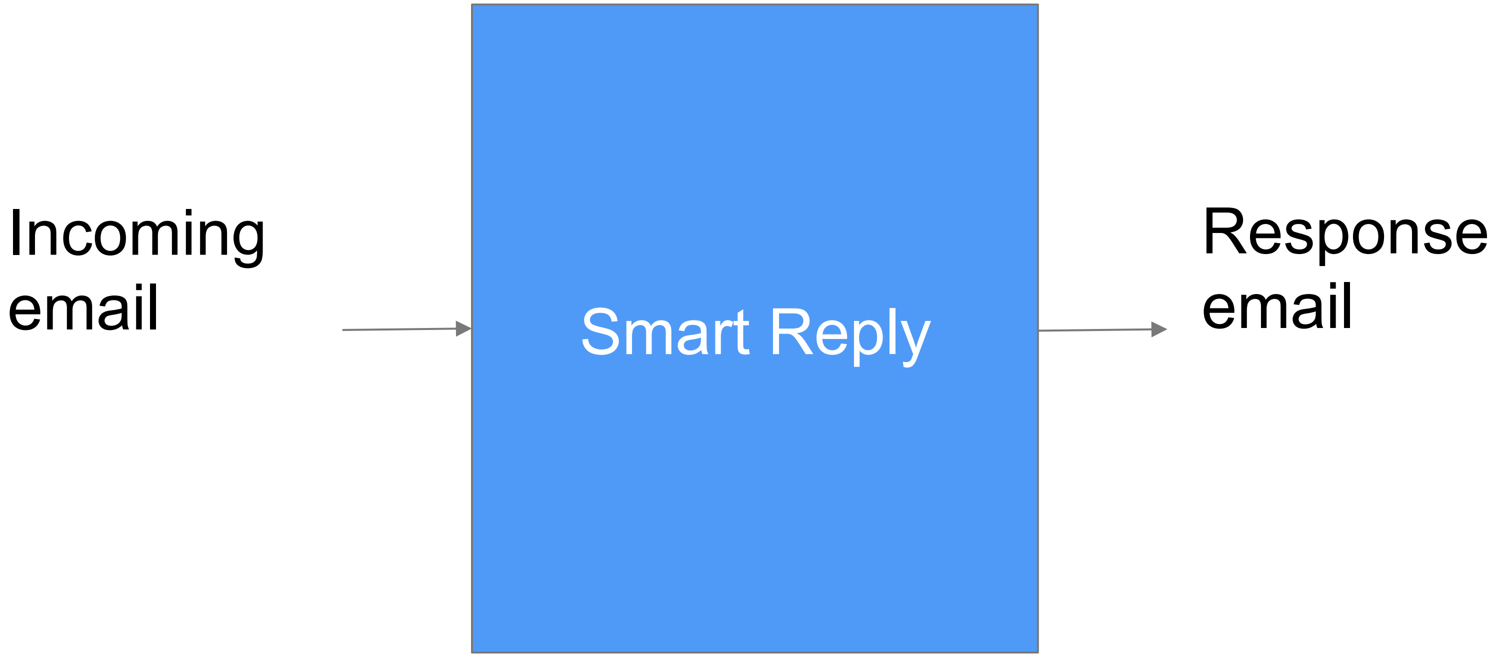
Problem

Smart Reply feature

- Provide text assistance for email reply composition
- Targeted at mobile
- Responses can be sent on their own or extended



Smart Reply feature predicts email responses

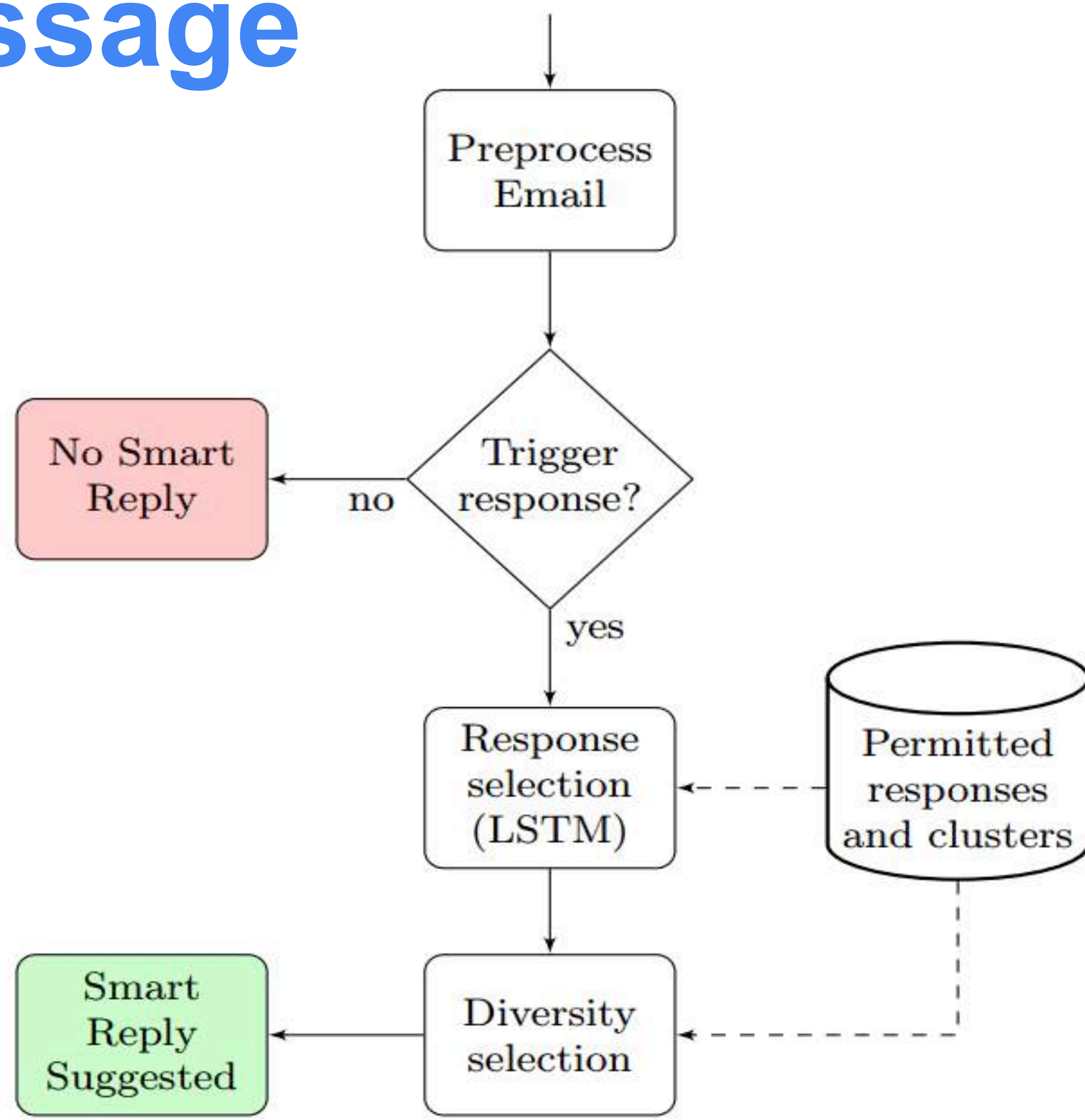


Why is this task hard?

- extracting meaning from previous message
- generating language
- grammatical transformations between call and response
- matching style/tone

Models

Life of a message



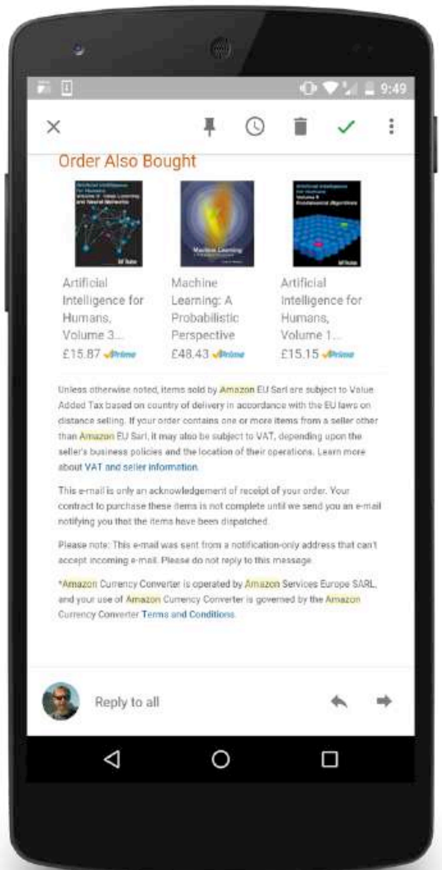
Two main models

- **Triggering: quickly filter bad candidates**

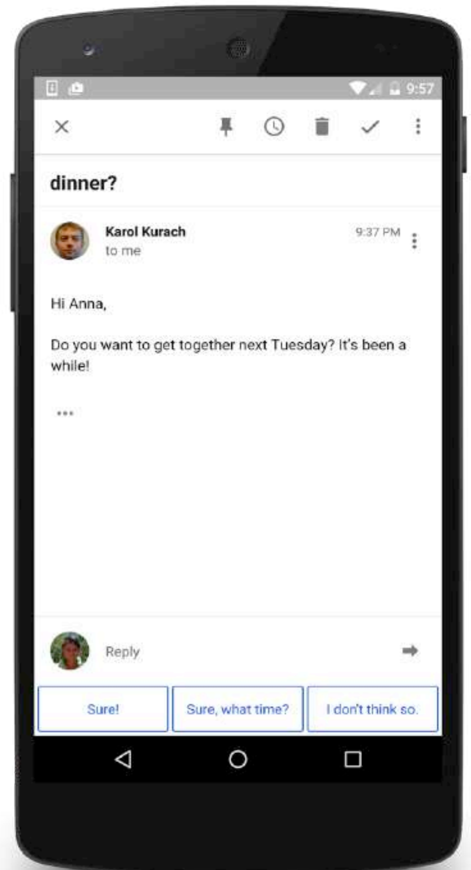
  How do we decide when it is appropriate to show suggestions, and avoid showing them when they would be not only useless but distracting?

- **Scoring: score a whitelist of responses**

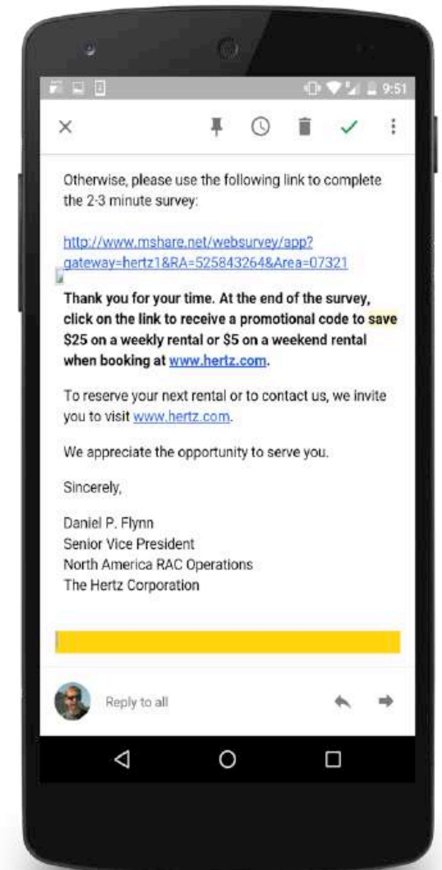
Triggering model



Receipt



Personal



Promo

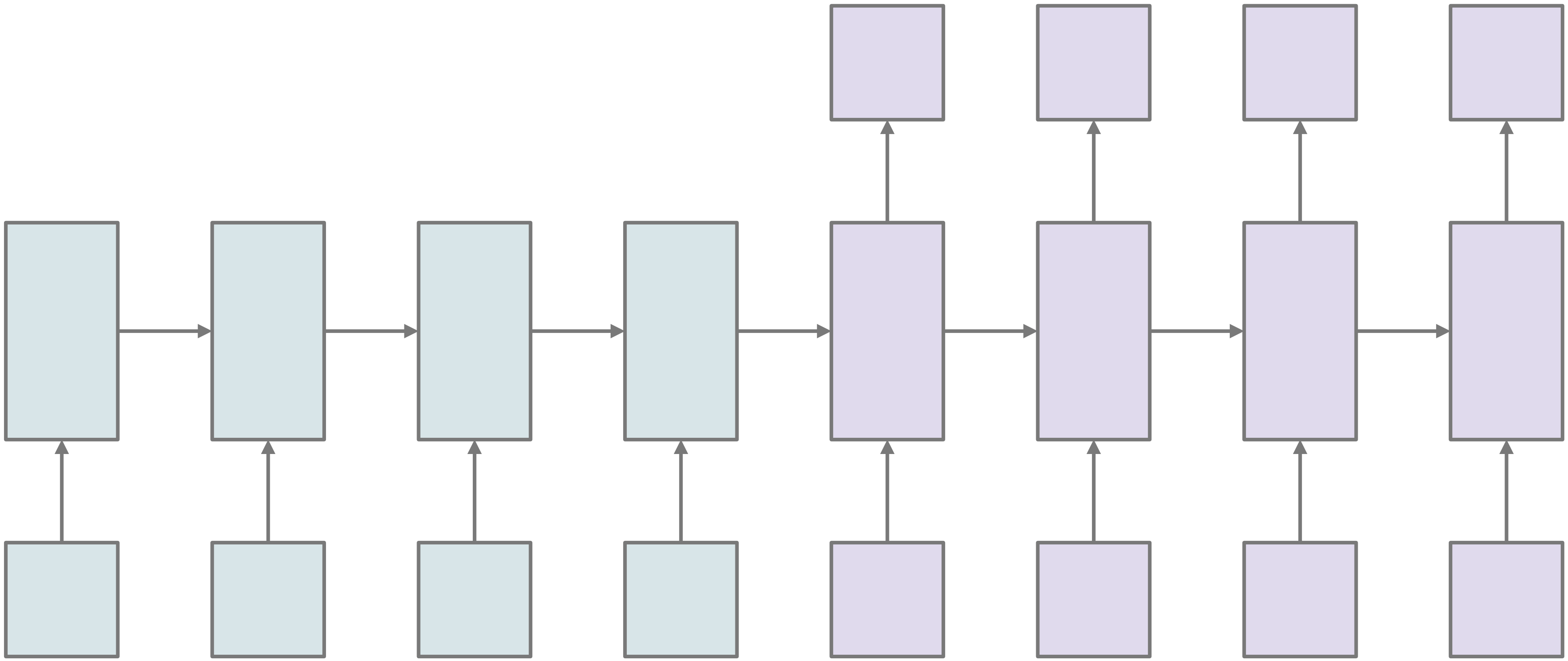
Triggering model

- How do we decide when it is appropriate to show suggestions, and avoid showing them when they would be not only useless but distracting?

Solution: Have a separate feed-forward neural network that decides whether to trigger.

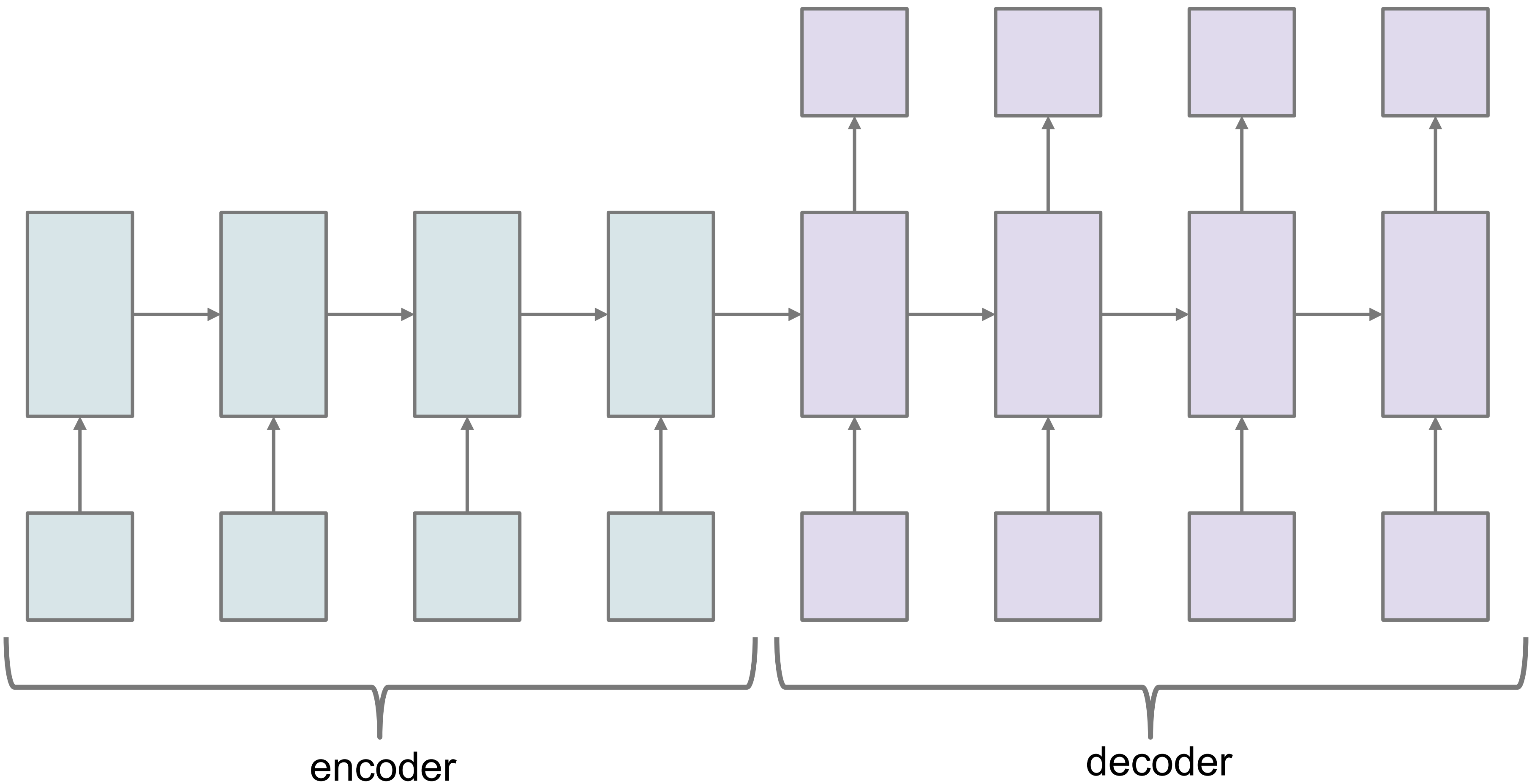
Challenging: mails not directly to me,
predicting replies != predicting smart replies, ...

Sequence-to-sequence model

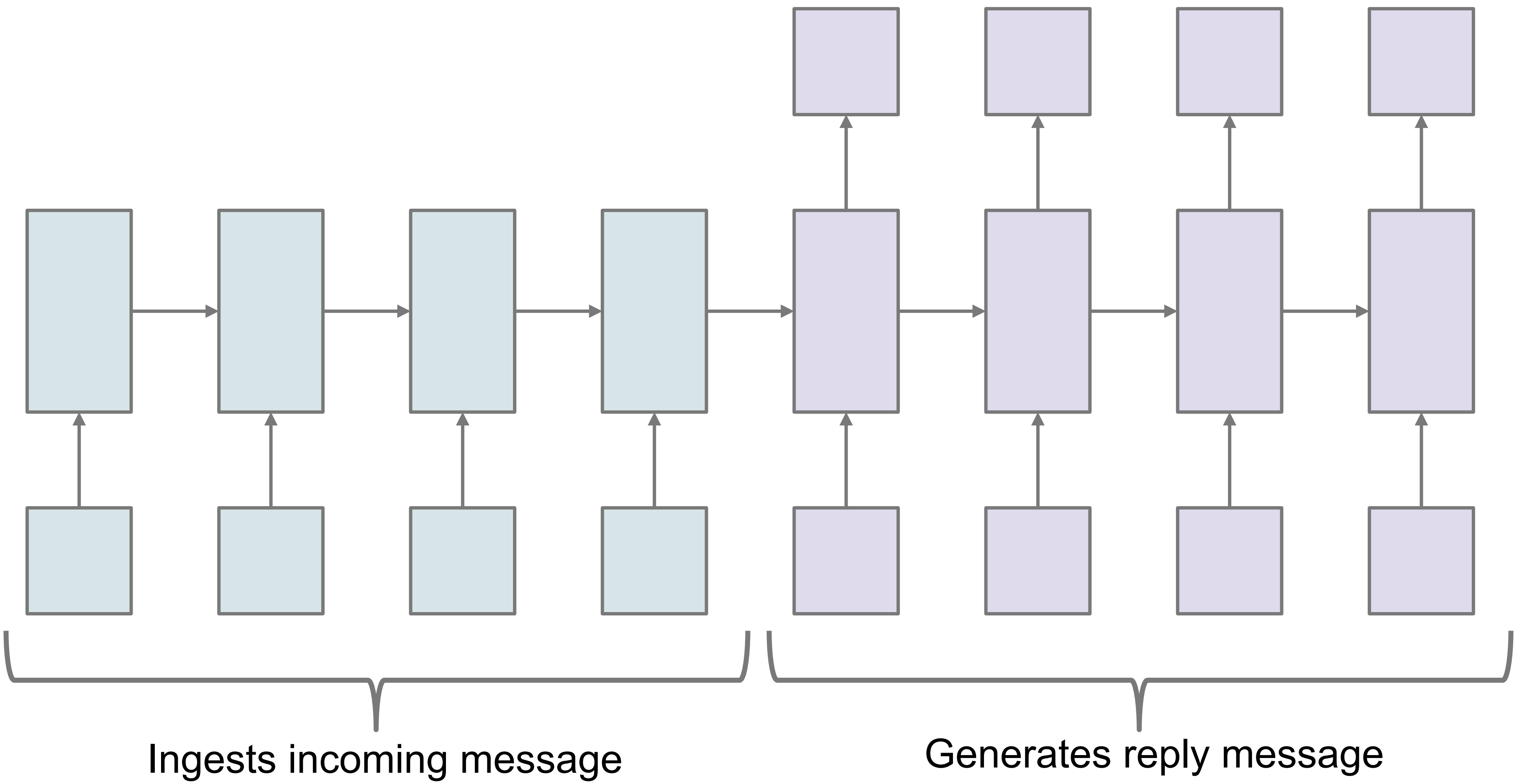


Sutskever et al, NIPS 2014

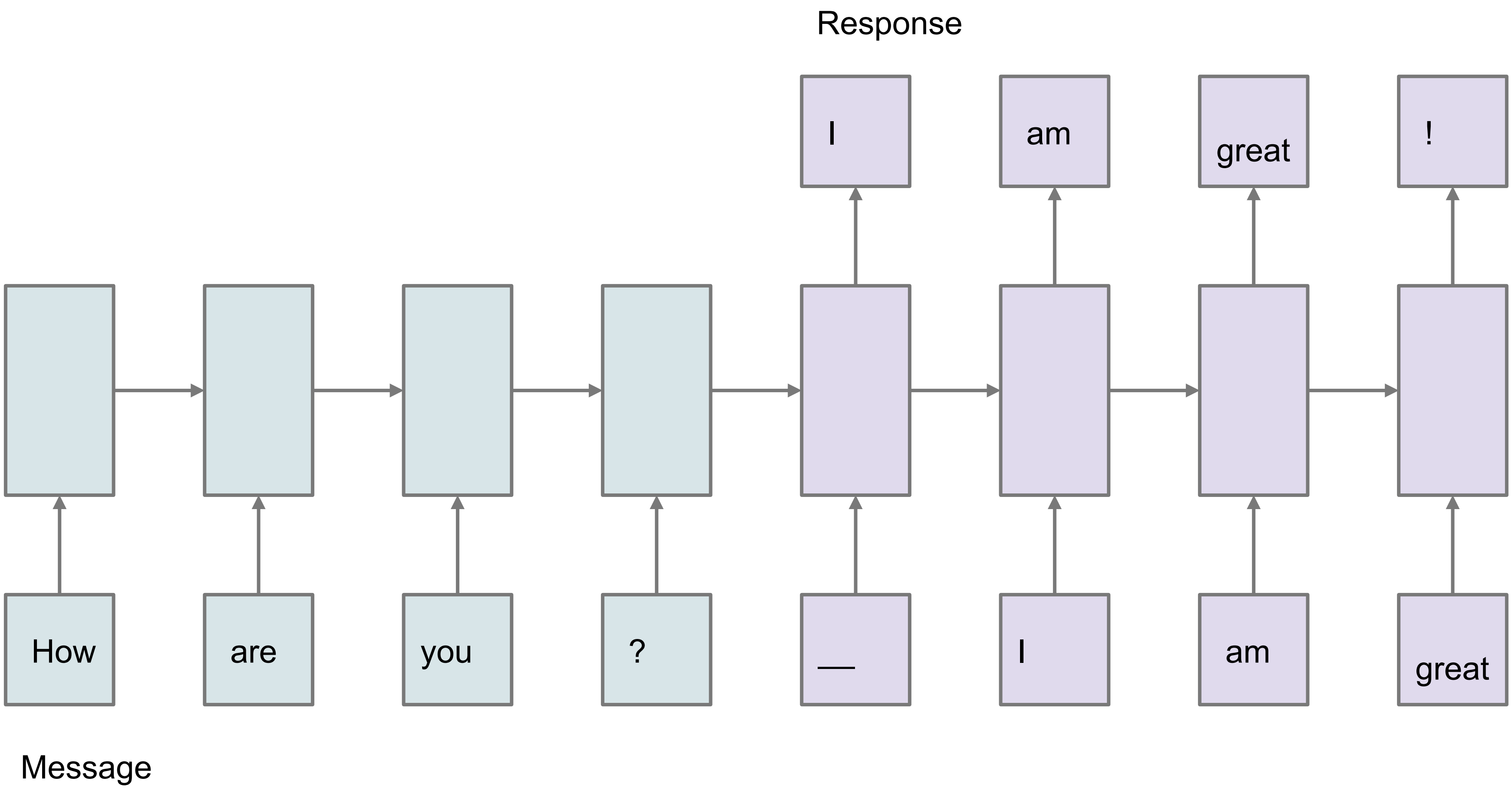
Sequence-to-sequence model



Sequence-to-sequence model



Smartreply model



Training

- Training data is a corpus of email-reply pairs
- Both encoder and decoder are trained together (end-to-end)

Inference

- Resulting model is fully generative
- Output distribution can be used to determine the most likely responses using a beam search

Example

Query	Top generated responses
Hi, I thought it would be great for us to sit down and chat. I am free Tuesday and Wenesday. Can you do either of those days? Thanks! –Alice	I can do Tuesday. I can do Wednesday. How about Tuesday? I can do Tuesday! I can do Tuesday. What time works for you? I can do Wednesday! I can do Tuesday or Wednesday. How about Wednesday? I can do Wednesday. What time works for you? I can do either.

Challenges

Quality

- How do we ensure that the response options are always high quality in content and language?
  - Avoid incorrect grammar and mechanics, misspellings *e.g., your the best*
  - Avoid inappropriate, offensive responses. *e.g., Leave me alone.*
  - Deal with wide variability, informal language. e.g., *got it thx*
- Restricting model vocabulary is not sufficient!

Solution: Restrict to a fixed set of valid responses, derived automatically from data.

Scalability

- How do we scale costly LSTM computation to the requirements of an email delivery pipeline?

Solution:

1. Use
2. Perform an approximate search over set of valid responses

Diversity

- How can we select a semantically diverse set of suggestions?

	Redundant responses	Responses with diversity (more useful)
Can you join tomorrow's meeting?	Yes, I'll be there. Yes, I will be there. I'll be there.	Sure, I'll be there. Yes, I can. Sorry, I won't be able to make it tomorrow.

Solution: Learn semantic intents of responses, then use these to filter out redundant suggestions.

Diversity

Our approach to diversity is based on two heuristics:

- Cluster-based diversity:
  Don’t show suggestions of the same intent.

- Forced positives/negatives:
  If there is an affirmative suggestion, also force a negative one (and vice versa).
  Product decision: offer positive/negative choice, even if the latter is rare.

Cluster-based diversity

«We're waiting for you, are you going to be here soon?»

1. On my way.
2. On my way!
3. I'm here.
4. I'm on my way.
5. I'm here!
6. Yes, I'm here.
7. I'll be there in a few minutes.
8. I am on my way.

On my way.
I am on my way.
I’m on my way.
...

I am here.
Already here!
Yes, I am here.
...

Will be there shortly.
I’ll be there in a few minutes.
Be there in a few!
...

Cluster-based diversity

«We're waiting for you, are you going to be here soon?»

1. On my way.
2. ~~On my way!~~
3. I'm here.
4. ~~I'm on my way.~~
5. ~~I'm here!~~
6. ~~Yes, I'm here.~~
7. I'll be there in a few minutes.
8. ~~I am on my way.~~

On my way.
I am on my way.
I’m on my way.
...

I am here.
Already here!
Yes, I am here.
...

Will be there shortly.
I’ll be there in a few minutes.
Be there in a few!
...

Diversity results

- Removing diversity click-through rate by 7.5% relative.

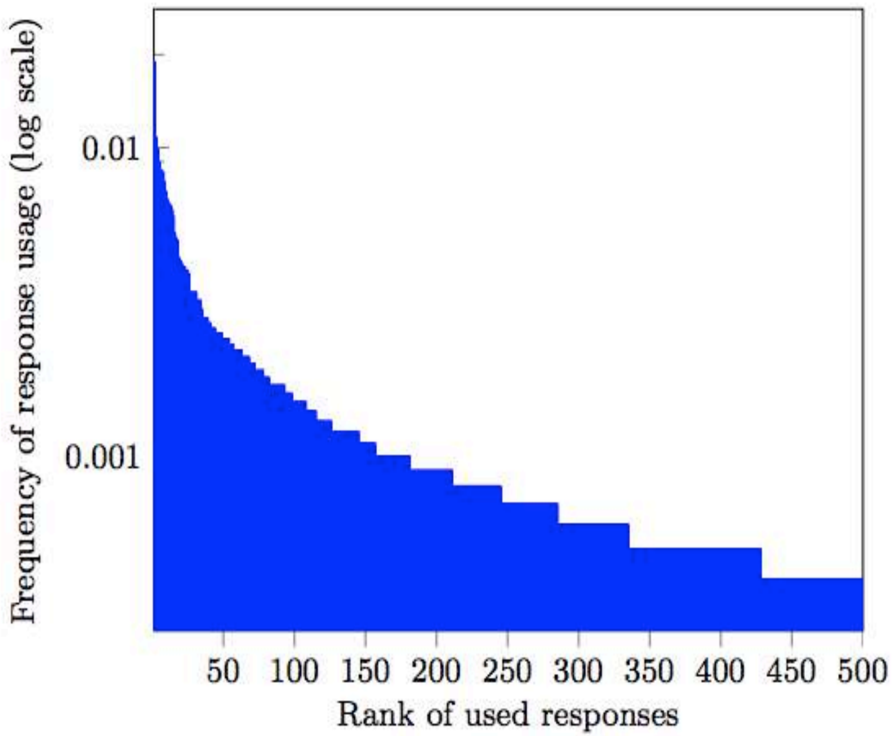
Results

Deployment & coverage

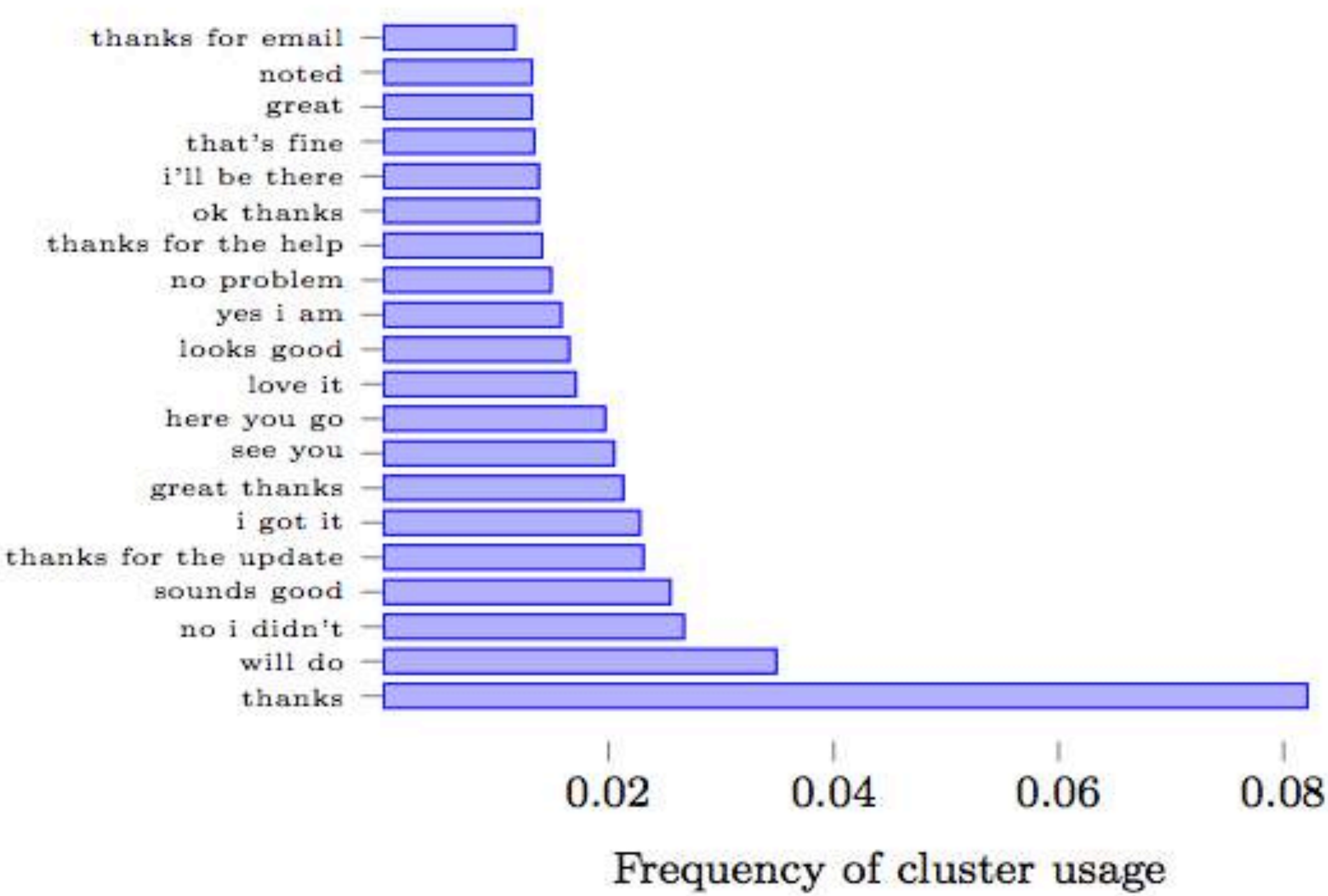
- Deployed in Inbox by Gmail
- Used to assist with more than 10% of all mobile replies

Unique cluster and suggestion usage

	Daily Count	Seen	Used
Unique Clusters	376	97.1%	83.2%
Unique Suggestions	12.9k	78%	31.9%



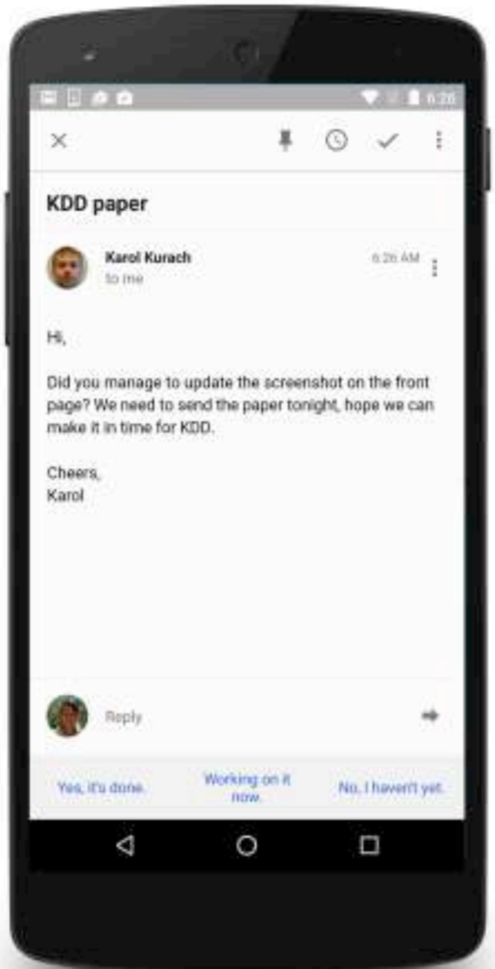
Most frequently used clusters

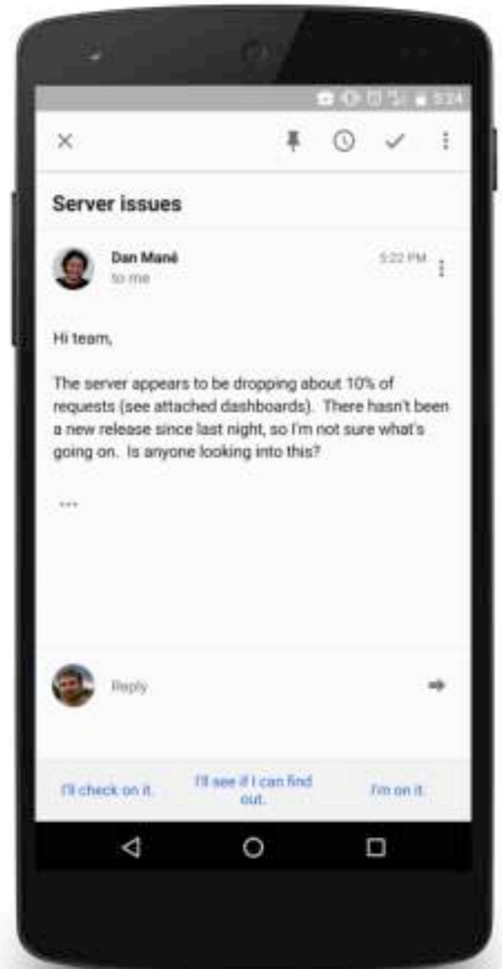


Ranking experiments

Model	Precision@10	Precision@20	MRR
Random	$5.58e-4$	$1.12e-3$	$3.64e-4$
Frequency	0.321	0.368	0.155
Multiclass-BOW	0.345	0.425	0.197
Smart Reply	0.483	0.579	0.267

Examples


KDD paper
Karol Kurach
to me
Hi,
Did you manage to update the screenshot on the front page? We need to send the paper tonight, hope we can make it in time for KDD.
Cheers,
Karol
Reply
Yes, it's done.
Working on it now.
No, I haven't yet.


Server issues
Dan Mané
to me
Hi team,
The server appears to be dropping about 10% of requests (see attached dashboards). There hasn't been a new release since last night, so I'm not sure what's going on. Is anyone looking into this?
Reply
I'll check on it.
I'll see if I can find out.
I'm on it.

Conclusions

- Sequence-to-sequence produces plausible email replies in many common scenarios, when trained on an email corpus
- Smart Reply is deployed in Inbox by Gmail and generates more than 10% of mobile replies
- RNNs show promise not only for assisted communication, but also for other applications where a conversation model is needed, such as virtual assistants

Agenda

1. Introduction to Deep Neural Architectures
2. Neural Random Access-Machines
3. Hierarchical Attentive Memory
4. Applications: Smart Reply
5. **Applications: Efficient Math Identities**
6. Applications: Predicting Events From Sensor Data

Learning to Discover Efficient Mathematical Identities

Karol Kurach*
Wojciech Zaremba*
Rob Fergus

Slides by Wojciech Zaremba & Karol Kurach

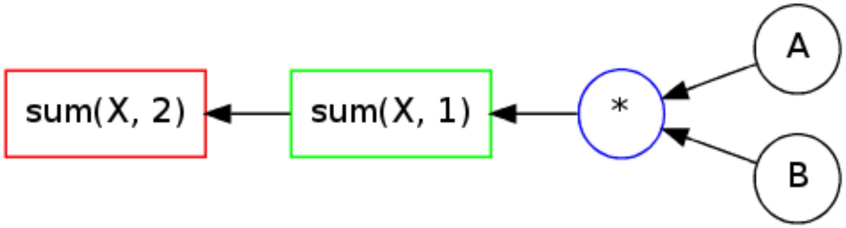
A toy example

Let's consider two matrices A, B

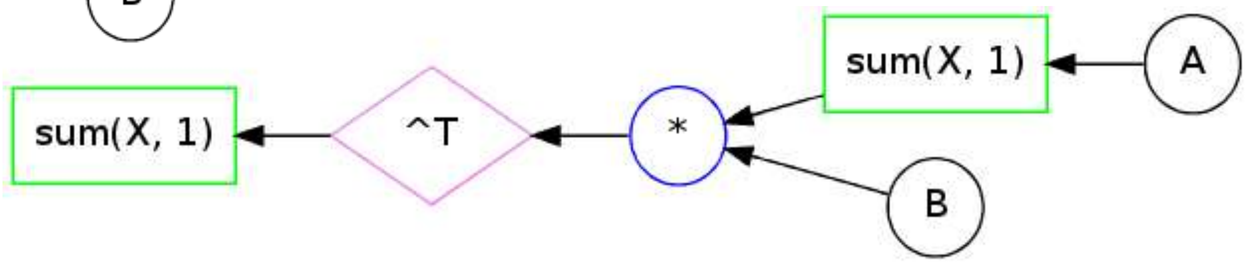
$$\sum_{i,k}(AB)_{i,k} = \sum_i \sum_j \sum_k a_{i,j} b_{j,k}$$

Naive computation takes O(n^3).

Our framework found O(n^2) computation



Naive computation
(O(n^3) time)



Optimized computation
(O(n^2) time)

Representation

- Symobolic - slow
- Floats - numerical issues
- Integers mod P - **works well!**

Allowed computations

Rule	Input	Output	Computation	Complexity
Matrix-matrix multiply	$X \in \mathbb{R}^{n \times m}, Y \in \mathbb{R}^{m \times p}$	$Z \in \mathbb{R}^{n \times p}$	`Z = X * Y`	$\mathrm{O}(nmp)$
Matrix-element multiply	$X \in \mathbb{R}^{n \times m}, Y \in \mathbb{R}^{n \times m}$	$Z \in \mathbb{R}^{n \times m}$	`Z = X .* Y`	$\mathrm{O}(nm)$
Matrix-vector multiply	$X \in \mathbb{R}^{n \times m}, Y \in \mathbb{R}^{m \times 1}$	$Z \in \mathbb{R}^{n \times n}$	`Z = X * Y`	$\mathrm{O}(nm)$
Matrix transpose	$X \in \mathbb{R}^{n \times m}$	$Z \in \mathbb{R}^{m \times n}$	`Z = X`^{T}	$\mathrm{O}(nm)$
Column sum	$X \in \mathbb{R}^{n \times m}$	$Z \in \mathbb{R}^{n \times 1}$	`Z = sum(X,1)`	$\mathrm{O}(nm)$
Row sum	$X \in \mathbb{R}^{n \times m}$	$Z \in \mathbb{R}^{1 \times m}$	`Z = sum(X,2)`	$\mathrm{O}(nm)$
Column repeat	$X \in \mathbb{R}^{n \times 1}$	$Z \in \mathbb{R}^{n \times m}$	`Z = repmat(X,1,m)`	$\mathrm{O}(nm)$
Row repeat	$X \in \mathbb{R}^{1 \times m}$	$Z \in \mathbb{R}^{n \times m}$	`Z = repmat(X,n,1)`	$\mathrm{O}(nm)$
Element repeat	$X \in \mathbb{R}^{1 \times 1}$	$Z \in \mathbb{R}^{n \times m}$	`Z = repmat(X,n,m)`	$\mathrm{O}(nm)$

We can consider arbitrary bigger grammar …

Many computations are in this family

- E.g. finite Taylor expansion of any function ….

Many computations are in this family

- E.g. finite Taylor expansion of any function ….

for instance, partition function of Restricted Boltzmann Machine (RBM)

$$\sum_{v,h} \exp(v^T W h) = \sum_k \sum_{v,h} \frac{1}{k!} (v^T W h)^k$$

$$v \in \{0,1\}^n$$

$$h \in \{0,1\}^m$$

Exact solution for k=1 (first term in Taylor series)

$$\sum_{v,h} v^T W h = 2^{n+m-2} \sum_{i,j} W_{i,j}$$

$$v \in \{0,1\}^n$$

$$h \in \{0,1\}^m$$

this is a polynomial computation vs exponential computation in the naive algorithm

Exact solution for k=2 (second term in Taylor series)

$$\sum_{v,h}(v^T W h)^2 = 2^{n+m-4}\Bigg[$$

$$\sum_{i,j} W_{i,j}^2 + (\sum_{i,j} W_{i,j})^2 +$$

$$\sum_i(\sum_j W_{i,j})^2 + \sum_j(\sum_i W_{i,j})^2\Bigg]$$

$$v \in \{0,1\}^n$$

$$h \in \{0,1\}^m$$

this is a polynomial computation vs exponential computation in the naive algorithm

How to find equivalent computations ?

Exact solution for k=6 (sneak preview) derived by our framework

```
1) .* sum(sum(W, 2), 1)))) .* 10) + (sum((sum((
( repmat(sum(W, 2), [1, m]) .* repmat(sum(W,
1), [n, 1])) .* W), 2) .* repmat(sum(sum(( W .*
W), 2), 1), [n, 1])), 1) .* 60) + ((sum(( sum(W,
2) .* sum(W, 2)), 1) .* ( sum(sum(W, 2), 1) .* (
sum(sum(W, 2), 1) .* sum(sum(W, 2), 1)))) .* 10) +
((sum(sum(( W .* W), 2), 1) .* ( sum(sum(W, 2),
1) .* ( sum(sum(W, 2), 1) .* sum(sum(W, 2), 1))))
.* 10) + (sum((sum(repmat(( sum(W, 1) .* sum(W,
1)), [n, 1]), 2) .* sum((repmat(sum(W, 2), [1, m]) .* (
repmat(sum(W, 2), [1, m]) .* repmat(sum(W, 1), [n,
1]))), 2)), 1) .* 30) + (sum(sum((( repmat(sum(W,
2), [1, m]) .* repmat(sum(W, 1), [n, 1])) .* W) .*
repmat(sum(( W .* W), 1), [n, 1])), 2), 1) .* -120)
+ ((sum(sum(W, 2), 1) .* ( sum(( sum(W, 1) .*
sum(W, 1)), 2) .* sum(sum(( W .* W), 2), 1))) .*
30) + (sum(sum((( repmat(sum(W, 2), [1, m]) .*
repmat(sum(( W .* W), 1), [n, 1])) .* repmat(sum((
W .* W), 1), [n, 1])), 2), 1) .* -30) + ((sum(sum(W,
2), 1) .* ( sum(( sum(W, 2) .* sum(W, 2)), 1)
.* sum(sum(( W .* W), 2), 1))) .* 30) + (sum((
repmat(sum(sum(W, 2), 1), [n, 1]) .* ( sum(( W .*
W), 2) .* sum(( W .* W), 2))), 1) .* -30) + (sum((
repmat(sum(sum(W, 2), 1), [n, 1]) .* sum(( ( W .*
W) .* ( W .* W)), 2)), 1) .* 20) + (( sum(sum(W, 2),
1) .* ( sum(sum(( W .* W), 2), 1) .* sum(sum(( W
.* W), 2), 1))) .* 15) + (sum((sum(( W .* W), 2) .*
sum(( ( repmat(sum(W, 2), [1, m]) .* repmat(sum(W,
1), [n, 1])) .* W), 2)), 1) .* -120) + ( sum(( ( ( W *
(W')) .* ( W * (W'))) * repmat(sum(sum(W, 2), 1),
[n, 1])), 1) .* 30))) / 1024;
```

1.6. $g(x \rightarrow x^6, W)$

```
2^(n+m)*(((sum(sum((( W .* W) .* (( rep-
mat(sum(W, 2), [1, m]) .* repmat(sum(W, 1), [n, 1]))
.* ( repmat(sum(W, 2), [1, m]) .* repmat(sum(W, 1),
[n, 1])))), 2), 1) .* 360) + (sum(sum((( W .* rep-
mat(sum(W, 2), [1, m])) .* ( ( W .* repmat(sum(W, 2),
[1, m])) .* repmat(sum(( W .* W), 1), [n, 1]))), 2), 1) .*
360) + (sum((( sum(W, 1) .* sum(W, 1)) .* ( ( sum(W,
1) .* sum(W, 1)) .* ( sum(W, 1) .* sum(W, 1))))), 2) .*
16) + ((sum(( sum(W, 1) .* sum(W, 1)), 2) .* sum((
( sum(W, 1) .* sum(W, 1)) .* ( sum(W, 1) .* sum(W,
1))), 2)) .* -30) + ( ( ( sum(W, 1) * ( (W') * sum(W,
2))) * ( sum(W, 1) * ( (W') * sum(W, 2)))) .* 360) + (
( ( sum(W, 1) * (W')) * ( ( W .* ( W .* W)) * (sum(W,
1)'))) .* 480) + (sum((( sum(W, 2) .* sum(W, 2)) .* (
( sum(W, 2) .* sum(W, 2)) .* ( sum(W, 2) .* sum(W,
2)))), 1) .* 16) + (sum(sum((( W .* repmat(sum(W,
1), [n, 1])) .* ( ( W .* repmat(sum(W, 1), [n, 1])) .*
repmat(sum(( W .* W), 2), [1, m]))), 2), 1) .* 360)
+ ((sum(( sum(W, 1) .* sum(W, 1)), 2) .* sum(( (
sum(W, 1) .* sum(W, 1)) .* sum(( W .* W), 1)), 2))
.* -180) + (sum((repmat(sum(sum(W, 2), 1), [n, 1]) .*
sum(( ( repmat(sum(W, 2), [1, m]) .* repmat(sum(W,
1), [n, 1])) .* ( W .* ( W .* W))), 2)), 1) .* 480) +
(sum((repmat(sum(sum(W, 2), 1), [n, 1]) .* (sum(W,
2) .* (sum(W, 2) .* sum(( ( repmat(sum(W, 2), [1, m])
.* repmat(sum(W, 1), [n, 1])) .* W), 2)))), 1) .* -240)
+ (sum((sum(( W .* repmat(sum(W, 1), [n, 1])), 2)
.* (sum(W, 2) .* sum(repmat(sum((repmat(sum(W,
2), [1, m]) .* ( repmat(sum(W, 2), [1, m]) .* rep-
mat(sum(W, 1), [n, 1]))), 1), [n, 1]), 2))), 1) .* 360) +
( ( ( sum(sum(W, 2), 1) * ( sum(W, 1) * (W'))) * ( (
W * (W')) * sum(W, 2))) .* 720) + ((( sum(sum(W,
2), 1) .* sum(sum(W, 2), 1)) .* (( sum(sum(W, 2),
1) .* sum(sum(W, 2), 1)) .* ( sum(sum(W, 2), 1)
.* sum(sum(W, 2), 1)))) .* 1) + ((sum(( sum(W,
1) .* sum(W, 1)), 2) .* (( sum(sum(W, 2), 1) .*
sum(sum(W, 2), 1)) .* ( sum(sum(W, 2), 1) .*
sum(sum(W, 2), 1)))) .* 15) + (sum((sum(( W .*
repmat(sum(W, 1), [n, 1])), 2) .* (sum(W, 2) .* rep-
mat(( sum(sum(W, 2), 1) .* ( sum(sum(W, 2), 1) .*
sum(sum(W, 2), 1))), [n, 1]))), 1) .* 120) + ((sum((
sum(W, 2) .* sum(W, 2)), 1) .* (( sum(sum(W, 2),
1) .* sum(sum(W, 2), 1)) .* ( sum(sum(W, 2), 1) .*
sum(sum(W, 2), 1)))) .* 15) + ((sum(sum(( W .* W),
2), 1) .* (( sum(sum(W, 2), 1) .* sum(sum(W, 2),
1)) .* ( sum(sum(W, 2), 1) .* sum(sum(W, 2), 1))))
.* 15) + (sum(sum((repmat(( sum(W, 2) .* sum(W,
2)), [1, m]) .* (( repmat(sum(W, 2), [1, m]) .* rep-
mat(sum(W, 1), [n, 1])) .* ( repmat(sum(W, 2), [1,
m]) .* repmat(sum(W, 1), [n, 1])))), 1), 2) .* -30)
+ (sum((sum((repmat(sum(W, 2), [1, m]) .* ( rep-
mat(sum(W, 2), [1, m]) .* repmat(sum(W, 1), [n,
1]))), 1) .* sum((repmat(sum(W, 2), [1, m]) .* ( rep-
mat(sum(W, 2), [1, m]) .* repmat(sum(W, 1), [n, 1]))),
1)), 2) .* 45) + (sum((sum(( W .* repmat(sum(W,
2), [1, m])), 1) .* (sum(( W .* repmat(sum(W, 2),
[1, m])), 1) .* repmat(sum(( sum(W, 1) .* sum(W,
1)), 2), [1, m]))), 2) .* 180) + (sum((sum(( W .* rep-
mat(sum(W, 2), [1, m])), 1) .* (sum(W, 1) .* sum((
( repmat(sum(W, 2), [1, m]) .* repmat(sum(W, 1),
[n, 1])) .* W), 1))), 2) .* -360) + ((sum(( sum(W, 1)
.* sum(W, 1)), 2) .* ( sum(( sum(W, 1) .* sum(W,
1)), 2) .* sum(( sum(W, 1) .* sum(W, 1)), 2))) .*
15) + (sum((sum((repmat(sum(W, 1), [n, 1]) .* ( rep-
mat(sum(W, 2), [1, m]) .* repmat(sum(W, 1), [n,
1]))), 1) .* sum((repmat(sum(W, 1), [n, 1]) .* ( rep-
mat(sum(W, 2), [1, m]) .* repmat(sum(W, 1), [n, 1]))),
1)), 2) .* -30) + ((sum(sum(W, 2), 1) .* (sum(sum(W,
2), 1) .* ( sum(( sum(W, 1) .* sum(W, 1)), 2) .* sum((
sum(W, 1) .* sum(W, 1)), 2)))) .* 45) + ((( sum(W,
1) .* sum(W, 1)) * ( ( repmat(sum(W, 1), [m, 1]) *
(W')) * ( W * (sum(W, 1)')))) .* 180) + (sum((sum((
W .* repmat(sum(W, 1), [n, 1])), 2) .* (( sum(W, 2) .*
repmat(sum(sum(W, 2), 1), [n, 1])) .* sum(repmat((
sum(W, 1) .* sum(W, 1)), [n, 1]), 2))), 1) .* 360) +
( ( sum(( W .* W), 1) * ( ( (W') * sum(W, 2)) .* (
(W') * sum(W, 2)))) .* -360) + (sum(( sum(( W .*
W), 1) .* ( ( sum(W, 1) .* sum(W, 1)) .* ( sum(W, 1)
.* sum(W, 1)))), 2) .* 240) + ((sum(sum(( W .* W),
2), 1) .* sum(( ( sum(W, 1) .* sum(W, 1)) .* ( sum(W,
1) .* sum(W, 1))), 2)) .* -30) + ((sum(( sum(W, 1) .*
sum(W, 1)), 2) .* ( sum(( sum(W, 1) .* sum(W, 1)), 2)
.* sum(sum(( W .* W), 2), 1))) .* 45) + ((sum(sum(W,
2), 1) .* ( sum(sum(W, 2), 1) .* sum(( ( sum(W, 1) .*
sum(W, 1)) .* sum(( W .* W), 1)), 2))) .* -180) + (((
sum(sum(W, 2), 1) .* sum(sum(W, 2), 1)) .* ( sum((
sum(W, 1) .* sum(W, 1)), 2) .* sum(sum(( W .* W),
2), 1))) .* 90) + (sum(sum((( repmat(sum(( W .* W),
2), [1, m]) .* repmat(sum(( W .* W), 1), [n, 1])) .* (
W .* W)), 2), 1) .* 360) + (sum((sum(( W .* W), 1)
.* ( sum(( W .* W), 1) .* repmat(sum(( sum(W, 1)
.* sum(W, 1)), 2), [1, m]))), 2) .* -90) + ((sum(sum((
W .* W), 2), 1) .* sum(( ( sum(W, 2) .* sum(W, 2))
.* ( sum(W, 2) .* sum(W, 2))), 1)) .* -30) + ((sum((
sum(W, 2) .* sum(W, 2)), 1) .* ( sum(( sum(W, 2) .*
sum(W, 2)), 1) .* sum(sum(( W .* W), 2), 1))) .* 45)
+ (sum((sum(( W .* repmat(sum(W, 2), [1, m])), 1)
.* (sum(( W .* repmat(sum(W, 2), [1, m])), 1) .* rep-
mat(sum(sum(( W .* W), 2), 1), [1, m]))), 2) .* 180)
+ (sum((sum(( W .* repmat(sum(W, 1), [n, 1])), 2)
.* (sum(( W .* repmat(sum(W, 1), [n, 1])), 2) .* rep-
mat(sum(sum(( W .* W), 2), 1), [n, 1]))), 1) .* 180)
+ (sum((( sum(W, 2) .* repmat(sum(sum(W, 2), 1),
[n, 1])) .* (sum(( W .* repmat(sum(W, 1), [n, 1])), 2)
.* repmat(sum(sum(( W .* W), 2), 1), [n, 1]))), 1) .*
360) + (sum((sum(( W .* repmat(sum(W, 1), [n, 1])),
2) .* sum(( repmat(( sum(W, 1) .* sum(W, 1)), [n, 1])
.* ( W .* repmat(sum(W, 1), [n, 1]))), 2)), 1) .* -240)
+ (sum(sum((W .* repmat((repmat(sum(sum(W, 2),
1), [1, m]) .* (sum(W, 1) .* sum(( ( repmat(sum(W,
2), [1, m]) .* repmat(sum(W, 1), [n, 1])) .* W),
1))), [n, 1])), 2), 1) .* -240) + ((sum(( sum(W, 2)
.* sum(W, 2)), 1) .* sum(( ( sum(W, 1) .* sum(W,
1)) .* ( sum(W, 1) .* sum(W, 1))), 2)) .* -30) + (((
sum(sum(W, 2), 1) .* sum(sum(W, 2), 1)) .* sum((
( sum(W, 2) .* sum(W, 2)) .* sum(( W .* W), 2)),
1)) .* -180) + ((( sum(sum(W, 2), 1) .* sum(sum(W,
2), 1)) .* sum(sum(( ( W .* W) .* ( W .* W)), 2),
1)) .* 60) + (sum((sum((repmat(sum(W, 2), [1, m])
.* ( repmat(sum(W, 2), [1, m]) .* repmat(sum(W,
1), [n, 1]))), 2) .* sum((repmat(sum(W, 2), [1, m])
.* ( repmat(sum(W, 2), [1, m]) .* repmat(sum(W,
1), [n, 1]))), 2)), 1) .* -30) + ((sum(sum(W, 2), 1)
.* sum((sum(repmat(( sum(W, 2) .* sum(W, 2)), [1,
m]), 1) .* sum((repmat(sum(W, 2), [1, m]) .* ( rep-
mat(sum(W, 2), [1, m]) .* repmat(sum(W, 1), [n, 1]))),
1)), 2)) .* 45) + (sum((sum(( W .* repmat(sum(W, 2),
[1, m])), 1) .* (sum(( W .* repmat(sum(W, 2), [1, m])),
1) .* repmat(( sum(sum(W, 2), 1) .* sum(sum(W, 2),
1)), [1, m]))), 2) .* 180) + ((( sum(sum(W, 2), 1) .*
sum(sum(W, 2), 1)) .* ( sum(( sum(W, 1) .* sum(W,
1)), 2) .* sum(( sum(W, 2) .* sum(W, 2)), 1))) .* 90)
+ (sum((sum(( W .* repmat(sum(W, 1), [n, 1])), 2)
.* (sum(( W .* repmat(sum(W, 1), [n, 1])), 2) .* rep-
mat(( sum(sum(W, 2), 1) .* sum(sum(W, 2), 1)), [n,
1]))), 1) .* 180) + ( ( ( sum(W, 1) * ( (W') * W)) * ( (
(W') * W) * (sum(W, 1)'))) .* 360) + (sum((sum(( W
.* repmat(sum(W, 1), [n, 1])), 2) .* (sum(( W .* rep-
mat(sum(W, 1), [n, 1])), 2) .* repmat(sum(( sum(W,
2) .* sum(W, 2)), 1), [n, 1]))), 1) .* 180) + ((sum((
sum(W, 1) .* sum(W, 1)), 2) .* ( sum(( sum(W, 1) .*
sum(W, 1)), 2) .* sum(( sum(W, 2) .* sum(W, 2)), 1)))
.* 45) + ( ( ( (sum(W, 2)') * W) * ( ( (W') * W) * (
(W') * sum(W, 2)))) .* 360) + (sum((sum(( W .* rep-
mat(sum(W, 2), [1, m])), 1) .* sum(( repmat(( sum(W,
2) .* sum(W, 2)), [1, m]) .* ( W .* repmat(sum(W, 2),
[1, m]))), 1)), 2) .* -240) + ((( (sum(W, 2)') * W) * (
(W') * ( repmat(sum(W, 2), [1, n]) * ( sum(W, 2) .*
sum(W, 2))))) .* 180) + ( ( ( (sum(W, 2)') * W) * (
(( W .* ( W .* W))') * sum(W, 2))) .* 480) + ((sum((
sum(W, 2) .* sum(W, 2)), 1) .* sum(( ( sum(W, 2)
.* sum(W, 2)) .* ( sum(W, 2) .* sum(W, 2))), 1))
.* -30) + ((sum(( sum(W, 2) .* sum(W, 2)), 1) .* (
sum(( sum(W, 2) .* sum(W, 2)), 1) .* sum(( sum(W,
2) .* sum(W, 2)), 1))) .* 15) + ((sum(( sum(W, 2) .*
sum(W, 2)), 1) .* sum(( ( sum(W, 2) .* sum(W, 2)) .*
sum(( W .* W), 2)), 1)) .* -180) + (sum((( sum(( W .*
W), 2) .* sum(( W .* W), 2)) .* repmat(sum(( sum(W,
2) .* sum(W, 2)), 1), [n, 1])), 1) .* -90) + ((sum((
sum(W, 1) .* sum(W, 1)), 2) .* ( sum(( sum(W, 2) .*
sum(W, 2)), 1) .* sum(sum(( W .* W), 2), 1))) .* 90)
+ ( sum(( ( sum(W, 1) .* sum(W, 1)) .* sum(( ( W .*
W) .* ( W .* W)), 1)), 2) .* -480) + (( sum(( sum(W,
1) .* sum(W, 1)), 2) .* sum(sum(( ( W .* W) .* ( W
.* W)), 2), 1)) .* 60) + ( sum(( sum(( W .* W), 1)
.* sum(( ( W .* W) .* ( W .* W)), 1)), 2) .* -480)
+ (sum((sum(repmat(( sum(W, 2) .* sum(W, 2)), [1,
m]), 1) .* sum(( ( W .* W) .* ( W .* W)), 1)), 2) .*
60) + ( ( ( sum(W, 1) .* sum(( W .* W), 1)) * ( (
(W') * W) * (sum(W, 1)'))) .* -720) + ((sum(sum(W,
2), 1) .* sum(sum((( ( repmat(sum(W, 2), [1, m]) .*
repmat(sum(W, 1), [n, 1])) .* W) .* repmat(sum(( W
.* W), 1), [n, 1])), 2), 1)) .* -720) + ((sum(( sum(W,
2) .* sum(W, 2)), 1) .* sum(( ( sum(W, 1) .* sum(W,
1)) .* sum(( W .* W), 1)), 2)) .* -180) + ((sum(sum((
W .* W), 2), 1) .* sum(( ( sum(W, 1) .* sum(W, 1)) .*
sum(( W .* W), 1)), 2)) .* -180) + (sum(( ( sum(W,
1) .* sum(W, 1)) .* ( sum(( W .* W), 1) .* sum(( W
.* W), 1))), 2) .* 720) + ( sum(( sum(( W .* W), 1)
.* ( sum(( W .* W), 1) .* sum(( W .* W), 1))), 2)
.* 240) + (sum(sum((repmat(sum(( W .* W), 1), [n,
1]) .* (repmat(( sum(W, 2) .* repmat(sum(sum(W, 2),
```

Prior over computation trees

- Explore space of computation efficiently
- Find equivalent expressions to the target one
  - But using operations with lower complexity
- Want to learn prior over sensible computations
  - Humans learn prior over proofs in mathematics

Searching over computation trees

Scheduler picks potential new expressions to append to current expressions

Scorer ranks each possibility (i.e. how likely they are to lead to the solution), using prior.

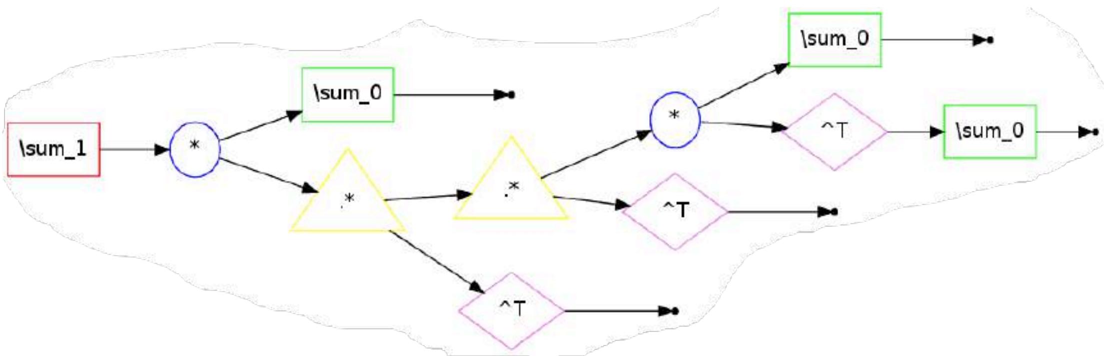
We want to learn a good scorer.

Scoring strategies

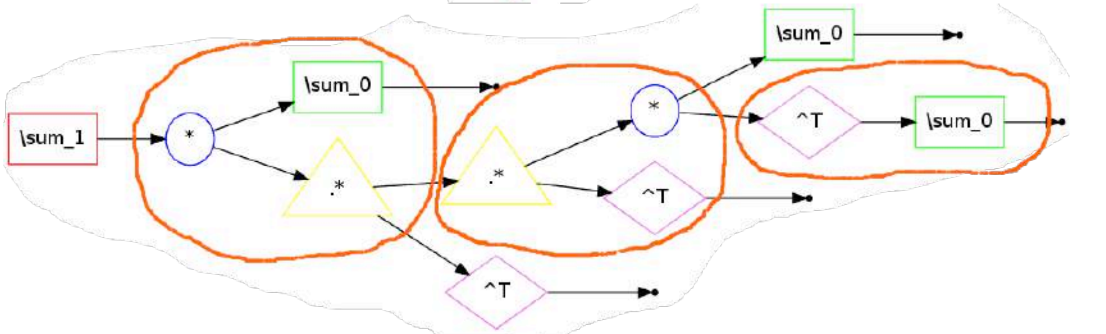
- naive scorer don’t use any prior. All computations are equally probable
- n-gram models
- learnt scorer

n-gram prior over trees

Exemplary intermediate solution:



Bi-grams:



Build n-grams distribution from solutions of simpler expressions

- Patterns that worked before might be useful

Experiments:
5 families of related problems

- $(\sum \mathbf{A}\mathbf{A}^{\mathbf{T}})_{\mathbf{k}}$
- $(\sum(\mathbf{A}. * \mathbf{A})\mathbf{A}^{\mathbf{T}})_k$
- Symmetric polynomials, e.g. $\sum_{i<j<k} A_i A_j A_k$
- RBM-1 $\sum_{v\in\{0,1\}^n} (v^T A)^k$
- RBM-2 $\sum_{v\in\{0,1\}^n, h\in\{0,1\}^n} (v^T A h)^k$

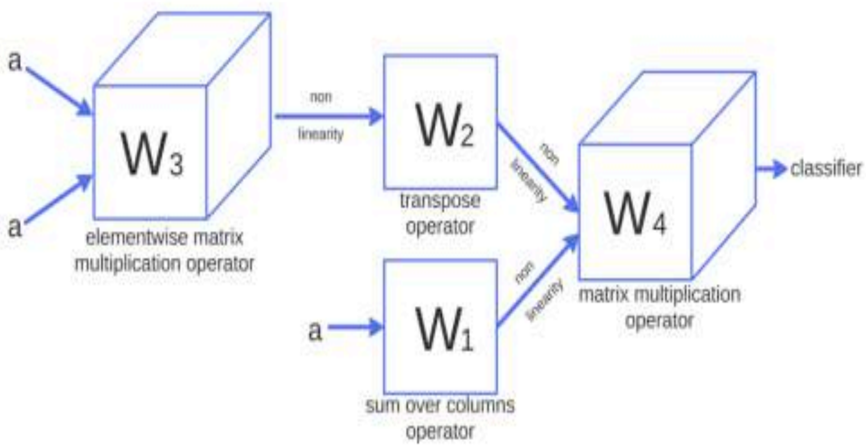
The meaning of a ~~word~~ computation is described by the ~~words~~ computations accompanying it

How we can represent a computation?

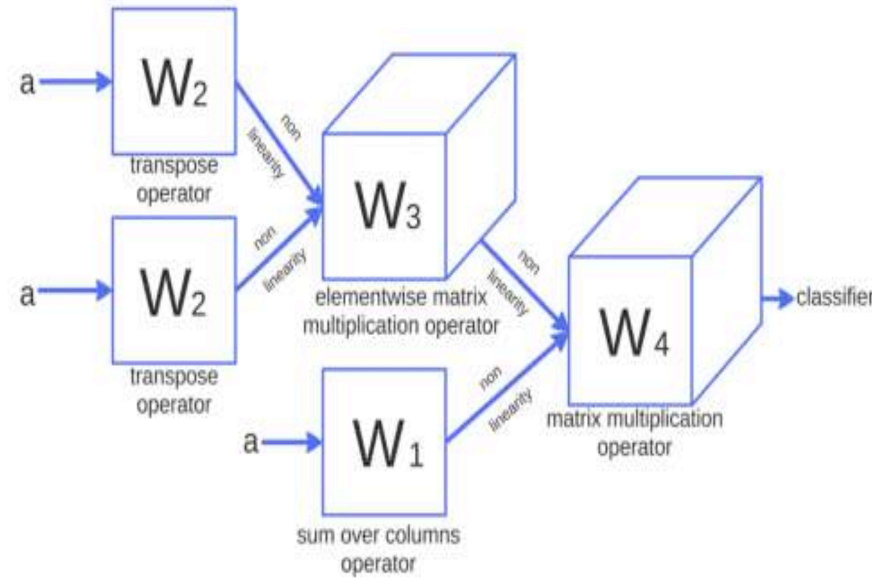
- Vector representation for every computation
  - e.g. A^T = vector_1 , \sum(A^T, 1) = vector_2,
- Want to learn how to compose their vector representations
  - i.e. ((A^T)^T)^T ~ vector_1, \sum(A, 2)^T ~ vector_2

Learnt representation with RNN

Recursive Neural network → RNN



(a) $(A. * A)' * \mathrm{sum}(A, 2)$



(b) $(A'. * A') * \mathrm{sum}(A, 2)$

Task - classify expressions

Example from A class:

```
(((sum((sum((A * (A′)), 1)), 2)) * ((A * (((sum((A′), 1)) * A)′))′)) * A)
```

Example from B class:

```
(((sum((A′), 1)) * (A * ((A′) * ((sum(A, 2)) * ((sum((A′), 1)) * A)))))′)
```

From which class is this example ?

```
((sum((sum((A * (A′)), 1)), 2)) * ((sum((A′), 1)) * (A * ((A′) * A))))
```

RNNs for a better discovery learning

- We have a real vector representation for any computations

- We use a linear classifier on such representation to train scorer

Family sum(AA^T)_k with RNN

RNN gives more diversified solutions (doesn't just copy them), but it doesn't perform as good as n-gram.

Targets → Exemplary solution of RNN:

- sum(A*A') → (sum((A * ((sum(A, 1))')), 1))
- sum(A*A'*A) → ((sum(A, 1)) * ((A') * (sum(A, 2))))
- sum(A*A'*A*A') → ((((sum(A, 1)) * (A')) * A) * ((sum(A, 1))'))
- sum(A*A'*A*A'*A) → ((sum(A, 1)) * ((A') * (A * ((A') * (sum(A, 2))))))

	naive	5-gram	RNN
Hardest possible example to solve	10	>15	~15

Summary

- Simple statistical priors over computations like n-gram allows the discovery of many new math formulae

- Use neural nets to map computational expressions to continuous vectors
  - Also use for formulae discovery

Agenda

1. Introduction to Deep Neural Architectures
2. Neural Random Access-Machines
3. Hierarchical Attentive Memory
4. Applications: Smart Reply
5. Applications: Efficient Math Identities
6. **Applications: Predicting Events From Sensor Data**

Deep Mining

Detecting Methane Outbreaks from Time Series Data with Deep Neural Networks

Karol Kurach* & Krzysztof Pawłowski*

Problem

Problem

Goal

Predict level of methane concentration in a coal mine

Why

Explosion danger

Data

- ➢ Multivariate: collected from 28 sensors
- ➢ Time series (1 read/sensor/sec over months)
- ➢ Non-stationary (concept drift)
- ➢ Correlated and overlapping

Deep Feedforward Neural Network

Architecture & Training

- Multiple hidden layers, sigmoid & ReLU activations
- Backpropagation with SGD, mini-batch, momentum
- Regularization: early stopping & dropout
- Trained on GPU. Cost function: RMSE

Parameters

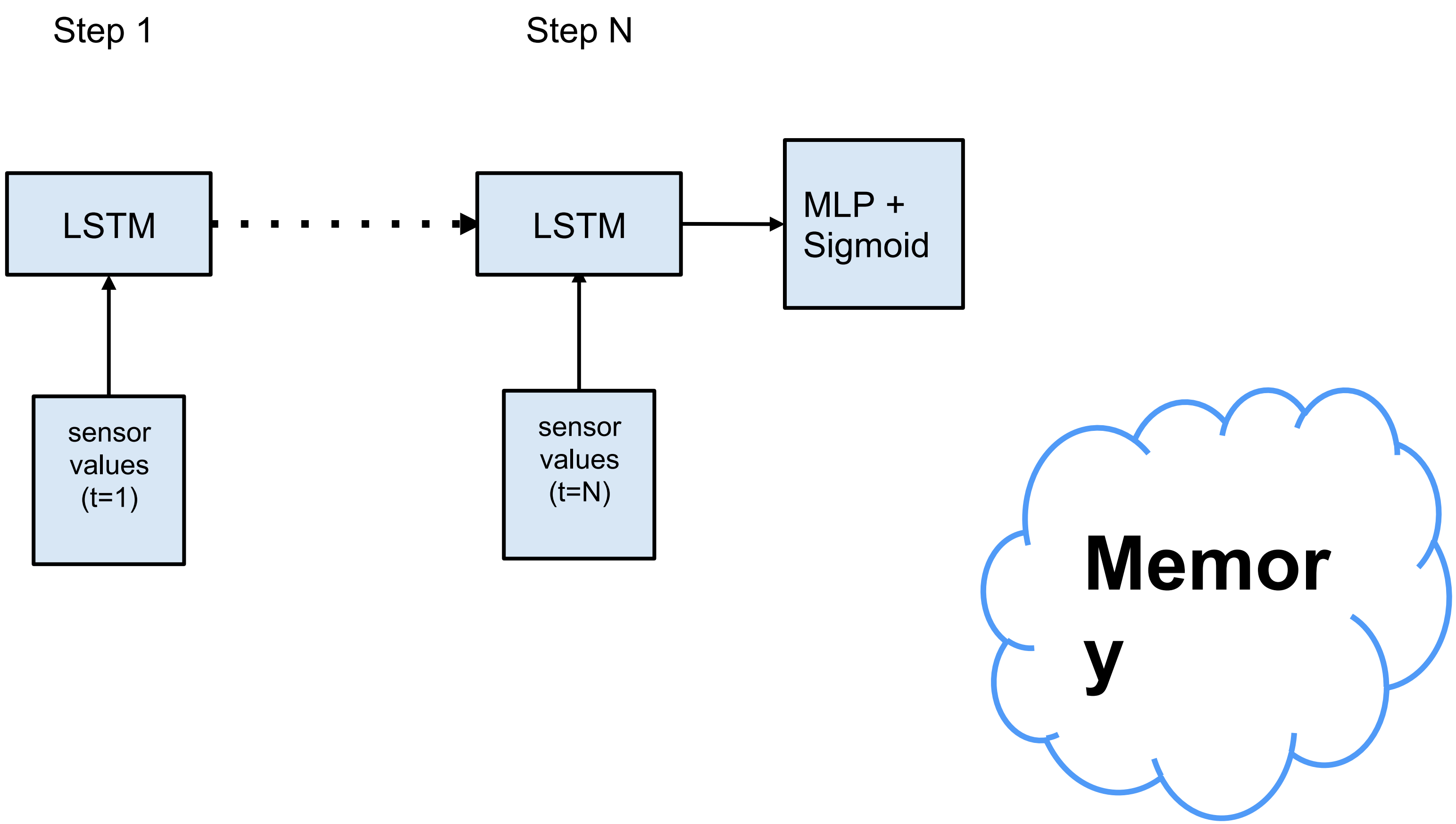
target label	MM263	MM264	MM256
activation	-	sigmoid	ReLU
layer sizes	-	76-15-5-2-1	76-25-7-3-1
dropout	-	none	.5
learning rate	-	0.1	0.1
mini-batch size	-	30	30
number of epochs	-	550	233

Long Short-Term Memory Model

Architecture & Training

- 1 layer LSTM unrolled to 60 time-steps
- Backpropagation through time with SGD
- Gradient clipping to avoid exploding gradients
- Trained on GPU. Cost function: RMSE

Long Short-Term Memory

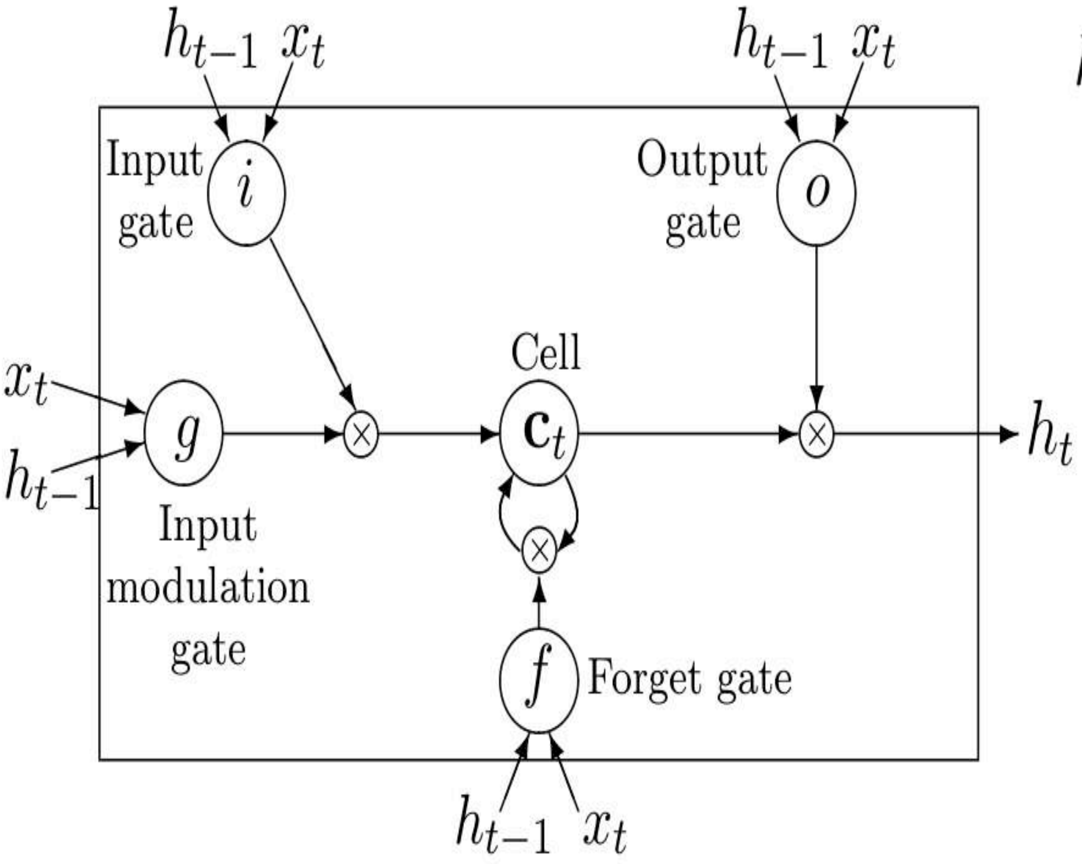


LSTM Cell

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \text{sigm} \\ \text{sigm} \\ \text{sigm} \\ \tanh \end{pmatrix} T_{2n,4n} \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$



Ensemble & Results

Ensembling & Evaluation

- For each of 3 target labels train each of 2 models
- Ensemble predictions - rank averaging
- Evaluate based on public leaderboard score
- For final submission choose the best-performer

Results

Area under ROC

higher is better

➢ Preliminary, public leaderboard scores:

AUC score \ label	MM263	MM264	MM256
LSTM	**0.9599**	0.9560	0.9605
DFFN	-	**0.9773**	0.9602
ensemble	-	0.9722	**0.9683**

➢ Final, private leaderboard score: 0.9400 and 6th place

- A big drop - overfitting?

Analysis

- A big drop due to overfitting to public leaderboard
- Classical cross-validation even worse
- Reason: Concept drift, correlated data points
- Better model selection schemes needed

Conclusion and Future Work

Conclusion

- ➢ A competitive score of 0.9400 score and the 6th place
- ➢ Deep Neural Networks are effective for sequential data
- ➢ Extensive feature engineering not required to perform well

Deep Mining

Predicting Dangerous Seismic Activity with Recurrent Neural Network

Karol Kurach
Krzysztof Pawłowski

Problem

Problem

Goal

Predict increased seismic activity in a coal mine

Why

To prevent life-threatening accidents

Data

- ➢ Multivariate (35 variables)
- ➢ Time series (22 variables are per-hour readings)
- ➢ Non-stationary (concept drift)
- ➢ Overlapping and unbalanced

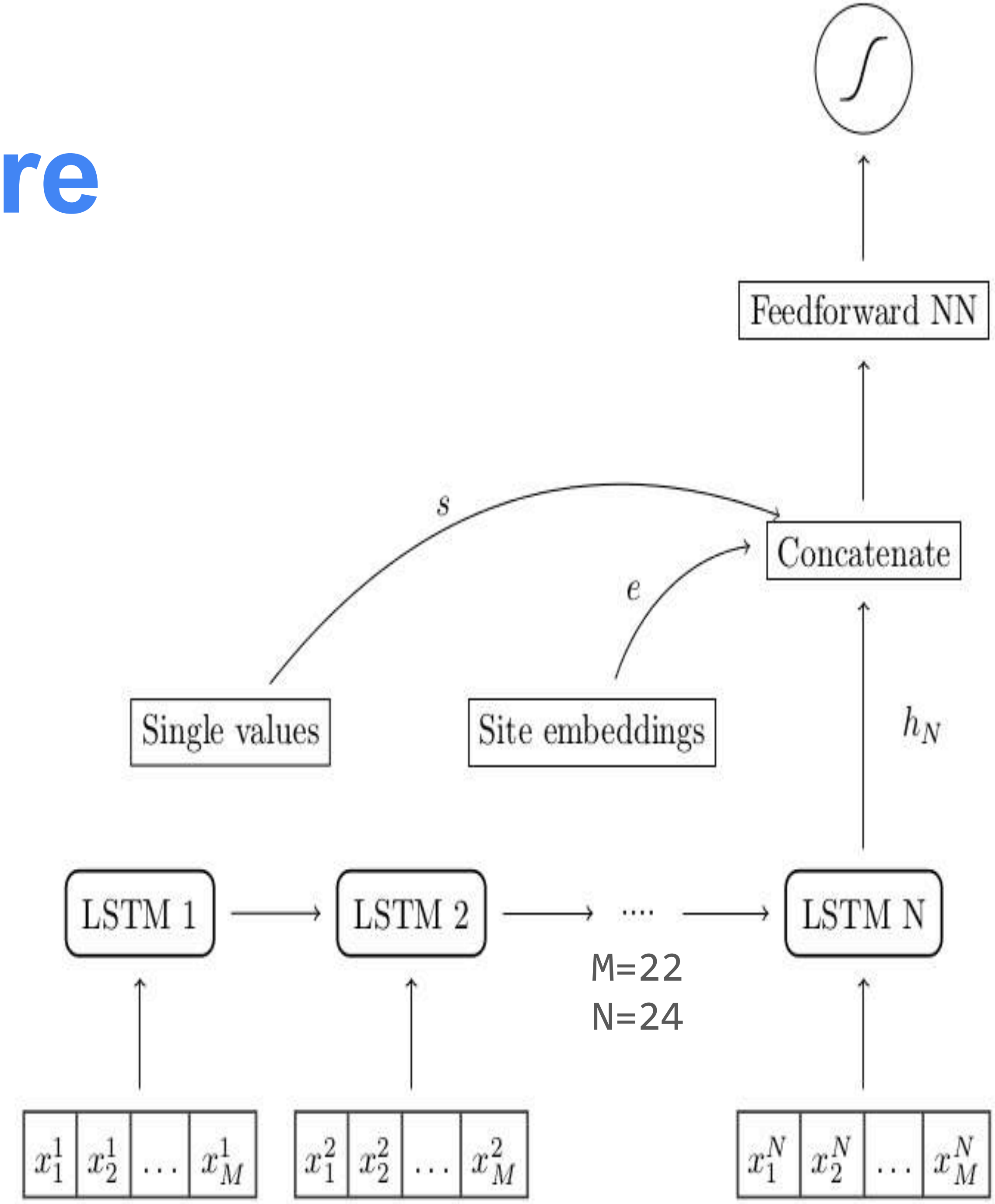
Our Solution

Preprocessing

- Goal: minimal preprocessing
- Normalization: mean = 0, stddev = 1
- Upsampling positives (10-20x)

Architecture

1. Time series data passed through RNN
2. Concatenate non-time series data
3. Pass through feed-forward neural network
4. Apply sigmoid and interpret the result as the likelihood of “warning”



Training

- ➢ Network unrolled for 24 time-steps
- ➢ Backpropagation through time with Adam algorithm
- ➢ Gradient clipping to avoid exploding gradients
- ➢ Trained on GPU. Cost function: Binary Cross Entropy

Model selection

- Overlapping data => cross-validation doesn't work
- CV-like scheme (5 folds, 1 training file per fold)
- Ignore the leaderboard score!

Ensembling

- Single model - nice, but usually ensembles win
- We ensembled our RNN with logistic regression
- We used rank averaging

Results & Conclusion

Results

- ➢ Final score: 5th place (0.934 score, 0.0057 less than winners)
- ➢ Single model (no ensembling): 7th place (0.931 score)

Conclusion

- ➢ Deep Neural Networks are effective for sequential data
- ➢ Extensive feature engineering not required to perform well
- ➢ Ensembling works!

Thank you!