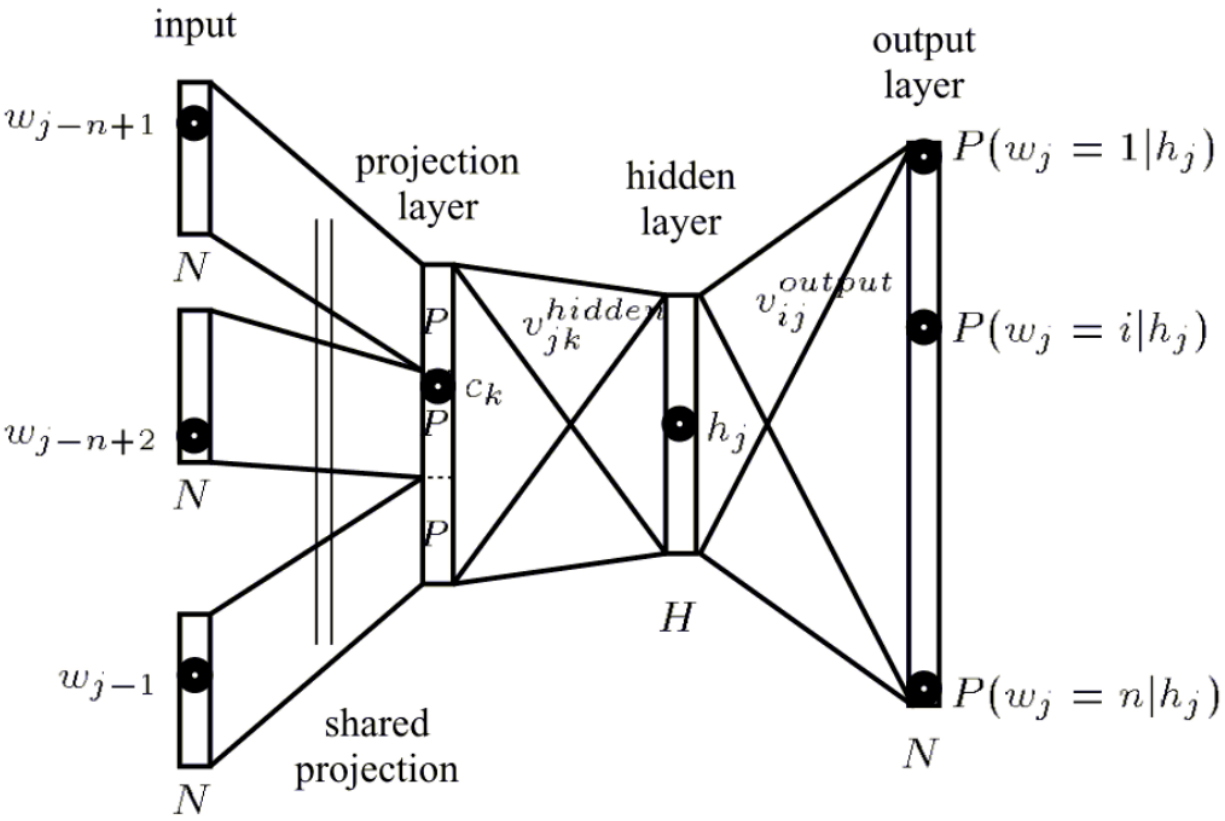
Language modeling

- Natural language is a sequence of sequences
- Some sentences are more likely than others:
  - “How are you ?” has a high probability
  - “How banana you ? “ has a low probability

[Slide: Wojciech Zaremba]

Neural Network Language Models



Bengio, Y., Schwenk, H., Sencal, J. S., Morin, F., & Gauvain, J. L. (2006). Neural probabilistic language models. In Innovations in Machine Learning (pp. 137-186). Springer Berlin Heidelberg.

[Slide: Antoine Border & Jason Weston, EMNLP Tutorial 2014]

Recurrent Neural Network Language Models

Key idea: *input to predict next word is current word plus context fed-back from previous word (i.e. remembers the past with recurrent connection).*

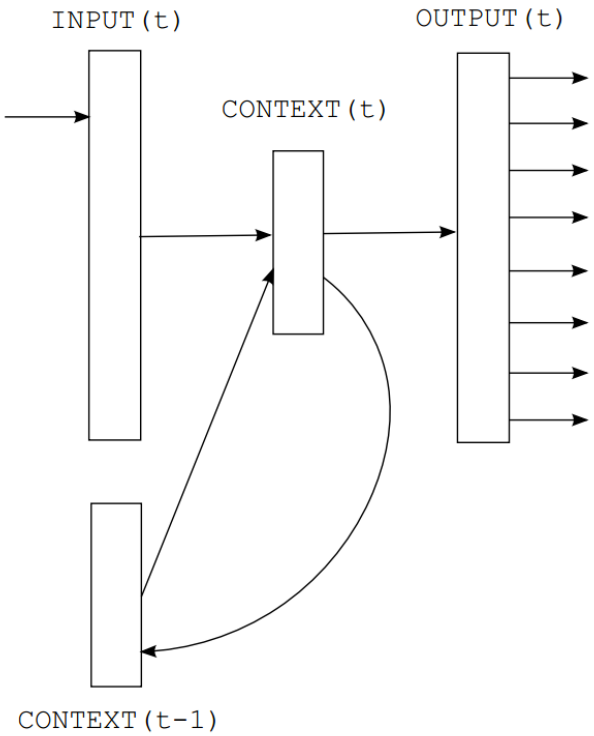
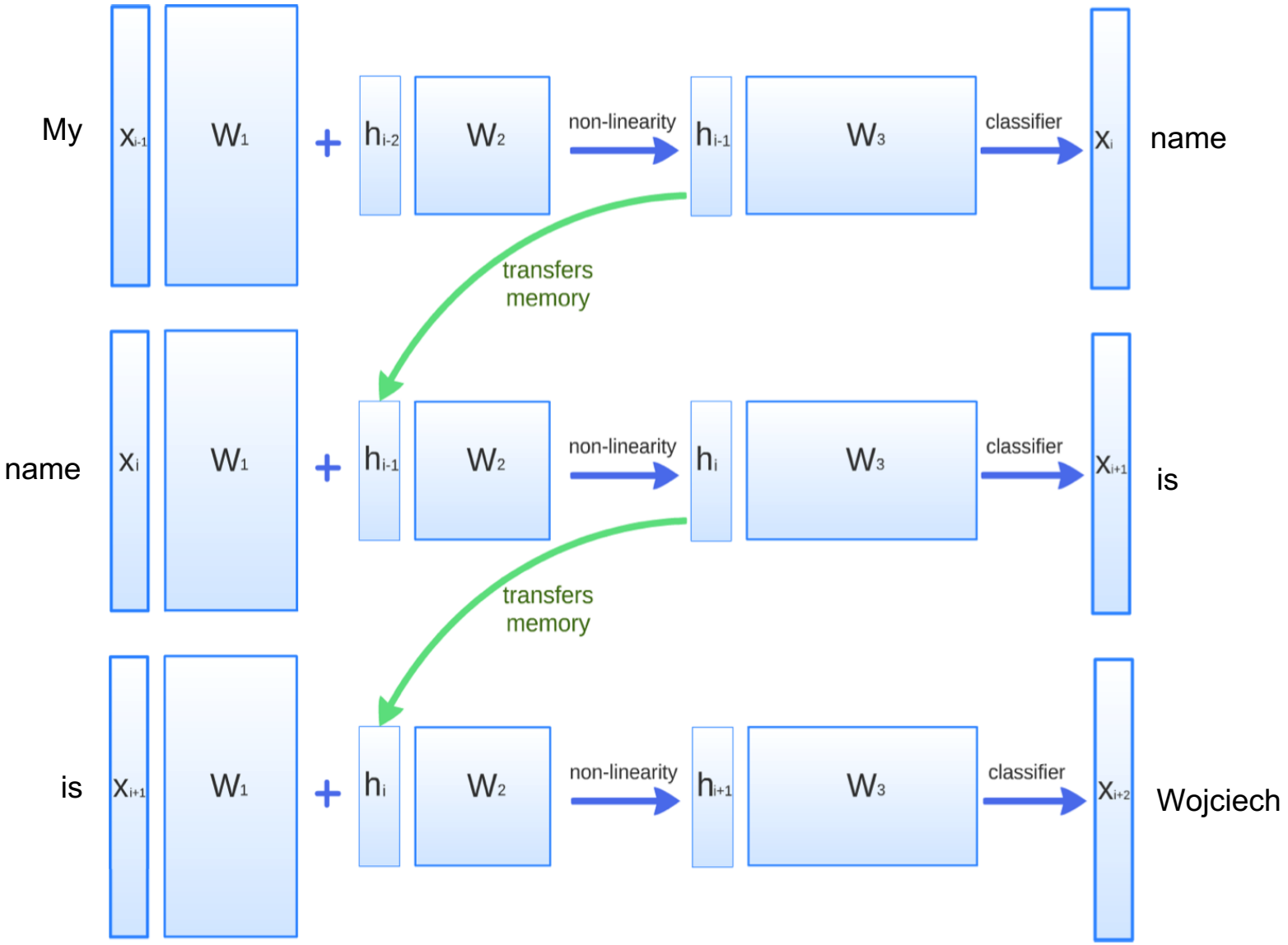


Figure: *Recurrent neural network based LM*

Recurrent neural network based language model. Mikolov et al., Interspeech, '10.

[Slide: Antoine Border & Jason Weston, EMNLP Tutorial 2014]

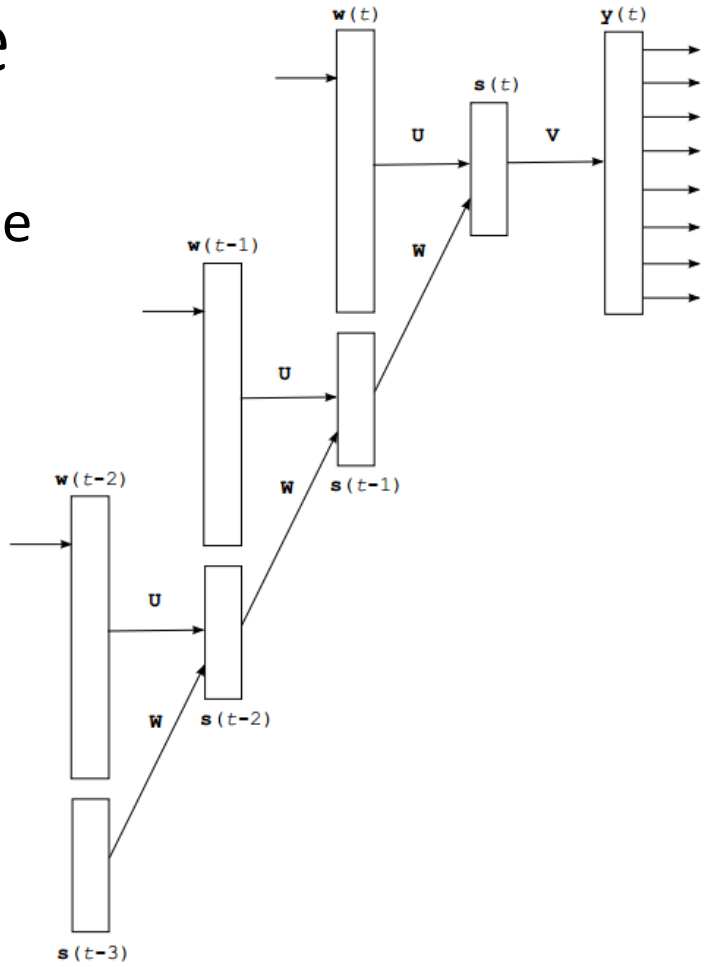
Recurrent neural networks - schema



[Slide: Wojciech Zaremba]

Backpropagation through time

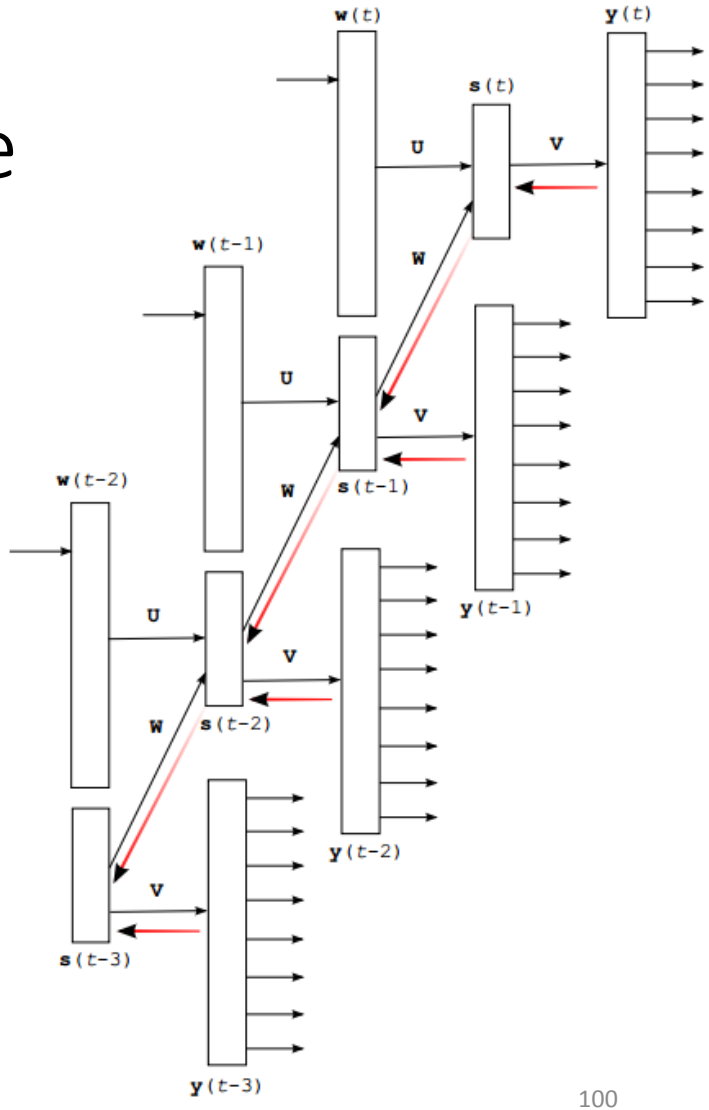
- The intuition is that we unfold the RNN in time
- We obtain deep neural network with shared weights **U** and **W**



[Slide: Thomas Mikolov, COLING 2014]

Backpropagation through time

- We train the unfolded RNN using normal backpropagation + SGD
- In practice, we limit the number of unfolding steps to 5 – 10
- It is computationally more efficient to propagate gradients after few training examples (batch mode)



[Slide: Thomas Mikolov, COLING 2014]

NNLMS vs. RNNS: Penn Treebank Results (Mikolov)

Model	Weight	PPL
3-gram with Good-Turing smoothing (GT3)	0	165.2
5-gram with Kneser-Ney smoothing (KN5)	0	141.2
5-gram with Kneser-Ney smoothing + cache	0.0792	125.7
Maximum entropy model	0	142.1
Random clusterings LM	0	170.1
Random forest LM	0.1057	131.9
Structured LM	0.0196	146.1
Within and across sentence boundary LM	0.0838	116.6
Log-bilinear LM	0	144.5
Feedforward NNLM	0	140.2
Syntactical NNLM	0.0828	131.3
Combination of static RNNLMs	0.3231	102.1
Combination of adaptive RNNLMs	0.3058	101.0
ALL	1	**83.5**

Recent uses of NNLMs and RNNs to improve machine translation:
Fast and Robust NN Joint Models for Machine Translation, Devlin et al, ACL '14.
Also Kalchbrenner '13, Sutskever et al., '14., Cho et al., '14. .

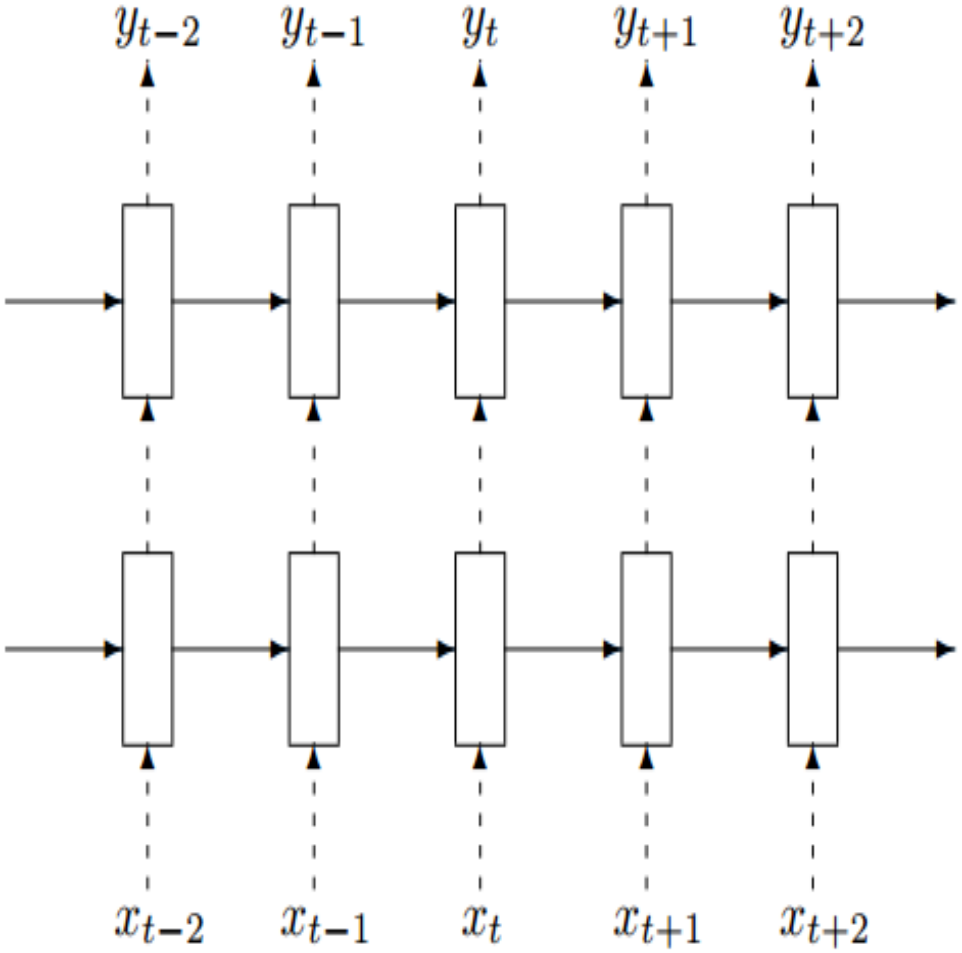
[Slide: Antoine Border & Jason Weston, EMNLP Tutorial 2014]

Language modelling – RNN samples

the meaning of life is that only if an end would be of the whole supplier. widespread rules are regarded as the companies of refuses to deliver. in balance of the nation's information and loan growth associated with the carrier thrifts are in the process of slowing the seed and commercial paper.

[Slide: Wojciech Zaremba]

More depth gives more power

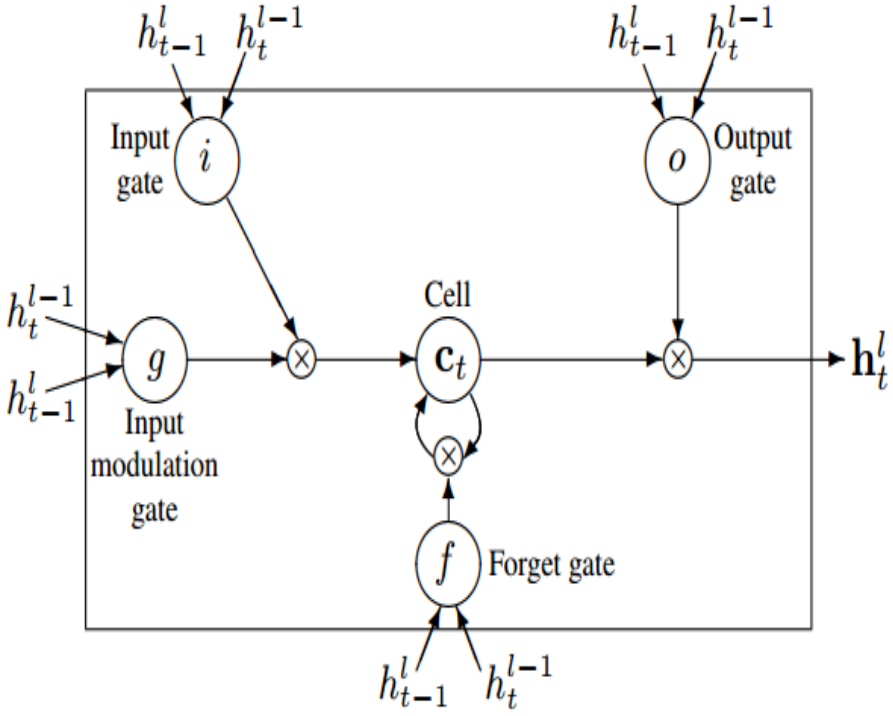


[Slide: Wojciech Zaremba]

LSTM - Long Short Term Memory

[Hochreiter and Schmidhuber, Neural Computation 1997]

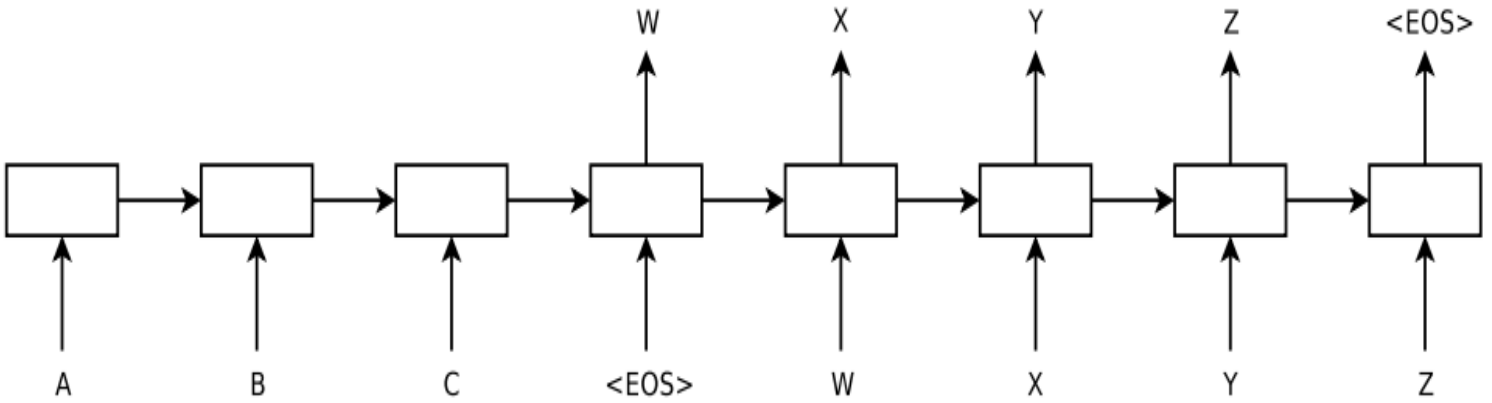
- Ad-hoc way of modelling long dependencies
- Many alternative ways of modelling it
- Next hidden state is modification of previous hidden state (so information doesn't decay too fast).



For simple explanation, see [Recurrent Neural Network Regularization, Wojciech Zaremba, Ilya Sutskever, Oriol Vinyals, arXiv 1409.2329, 2014]

[Slide: Wojciech Zaremba]

RNN-LSTMs for Machine Translation



[Sutskever et. al. (2014)]

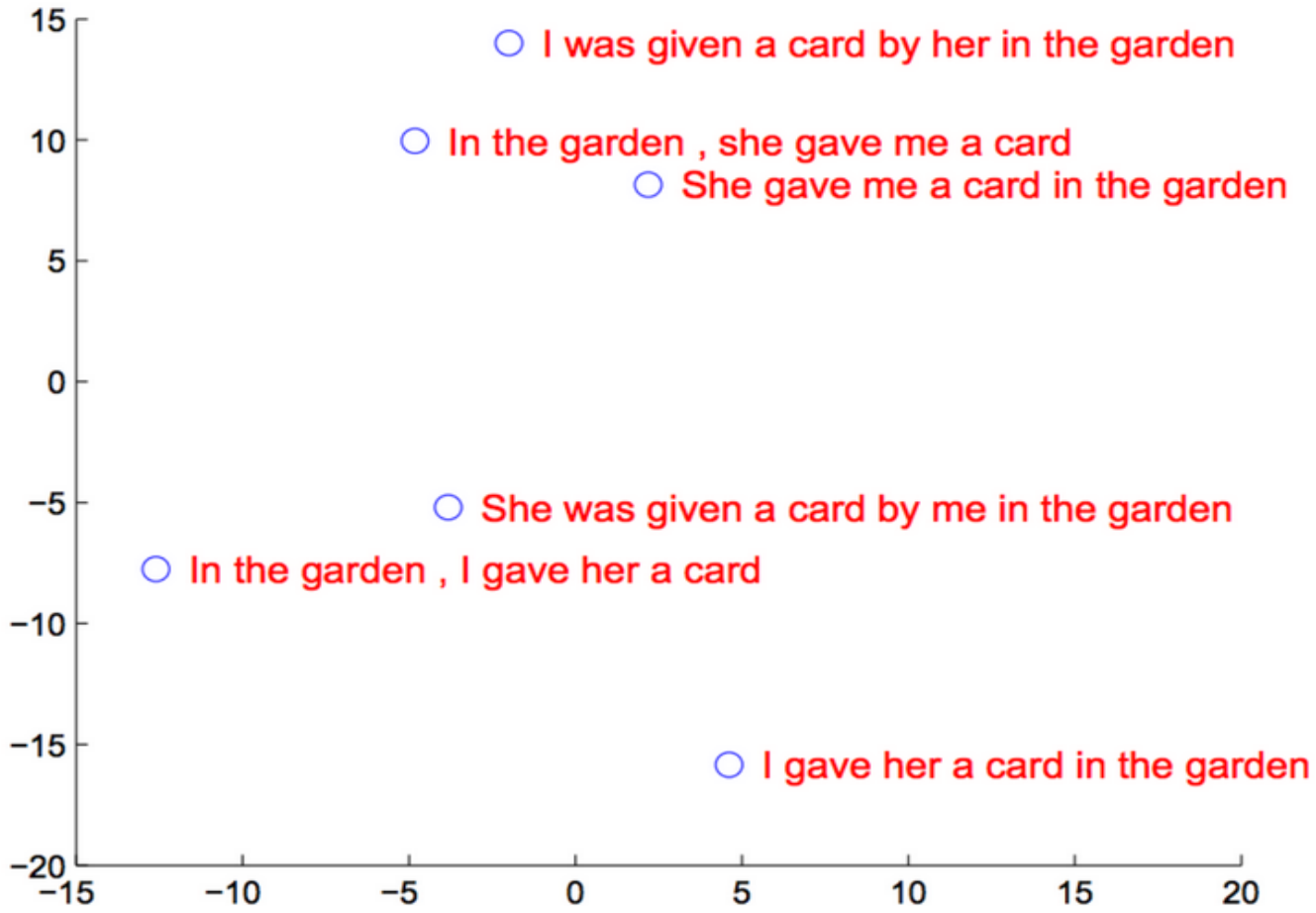
Sequence to Sequence Learning with Neural Networks,
Ilya Sutskever, Oriol Vinyals, Quoc Le, NIPS 2014

Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation, Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, Yoshua Bengio, EMNLP 2014

[Slide: Wojciech Zaremba]

Visualizing Internal Representation

t-SNE projection of network state at end of input sentence



Sequence to Sequence Learning with Neural Networks,
Ilya Sutskever, Oriol Vinyals, Quoc Le, NIPS 2014

Translation - examples

- FR: Les avionneurs se querellent au sujet de la largeur des sièges alors que de grosses commandes sont en jeu

- Google Translate: Aircraft manufacturers are quarreling about the seat width as large orders are at stake

- LSTM: Aircraft manufacturers are concerned about the width of seats while large orders are at stake

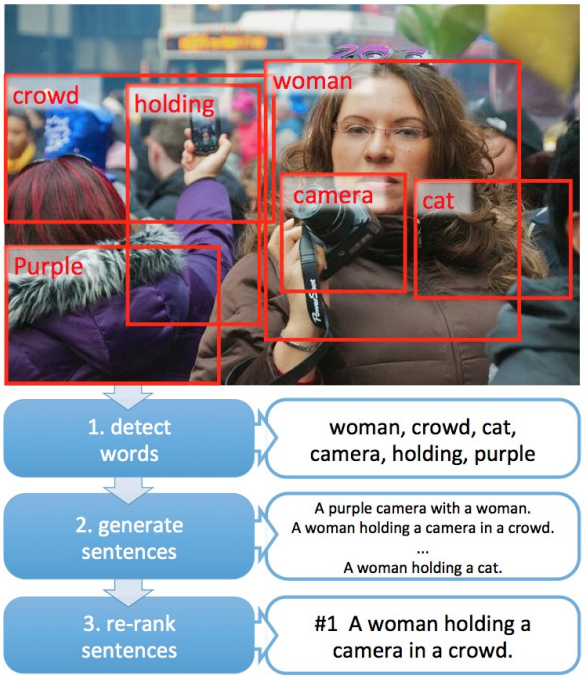
- Ground Truth: Jet makers feud over seat width with big orders at stake

[Sequence to Sequence Learning with Neural Networks,
Ilya Sutskever, Oriol Vinyals, Quoc Le, NIPS 2014]

[Slide: Wojciech Zaremba]

Image Captioning: Vision + NLP

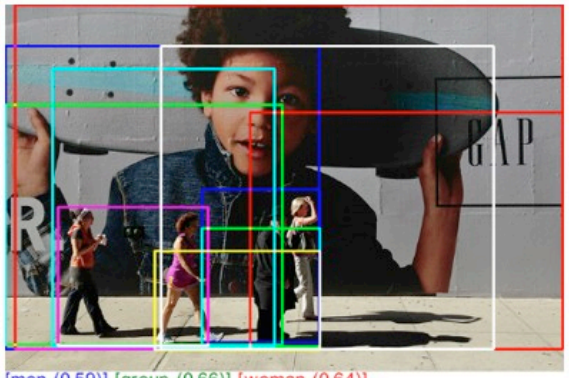
- Generate short text descriptions of image, given just picture.
- Use Convnet to extract image features
- RNN or LSTM model takes image features as input, generates text

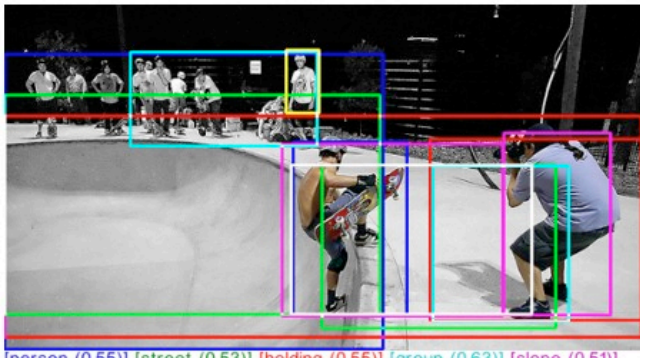


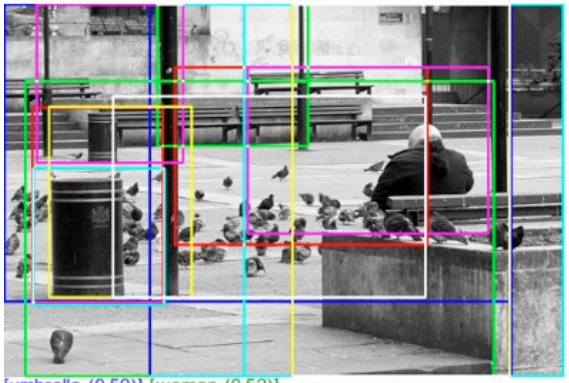
Many recent works on this:

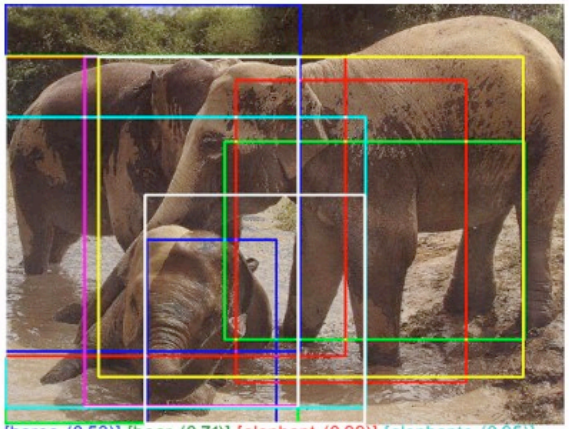
- Baidu/UCLA: Explain Images with Multimodal Recurrent Neural Networks
- Toronto: Unifying Visual-Semantic Embeddings with Multimodal Neural Language Models
- Berkeley: Long-term Recurrent Convolutional Networks for Visual Recognition and Description
- Google: Show and Tell: A Neural Image Caption Generator
- Stanford: Deep Visual-Semantic Alignments for Generating Image Description
- UML/UT: Translating Videos to Natural Language Using Deep Recurrent Neural Networks
- Microsoft/CMU: Learning a Recurrent Visual Representation for Image Caption Generation
- Microsoft: From Captions to Visual Concepts and Back

Image Captioning Examples









From Captions to Visual Concepts and Back, Hao Fang∗ Saurabh Gupta∗ Forrest Iandola∗ Rupesh K. Srivastava∗, Li Deng Piotr Dollar, Jianfeng Gao Xiaodong He, Margaret Mitchell John C. Platt, C. Lawrence Zitnick, Geoffrey Zweig, CVPR 2015.

Memory in Neural Networks

Rob Fergus

New York University
Facebook AI Research

Introduction

- Many tasks require some kind of memory
- But traditional neural networks are not good at remembering things, especially when input is large but only part of it is relevant
- Recently, there has been lot of interest in incorporating memory and attention to neural networks
  - Memory Networks, Neural Turing Machine,…

Outline

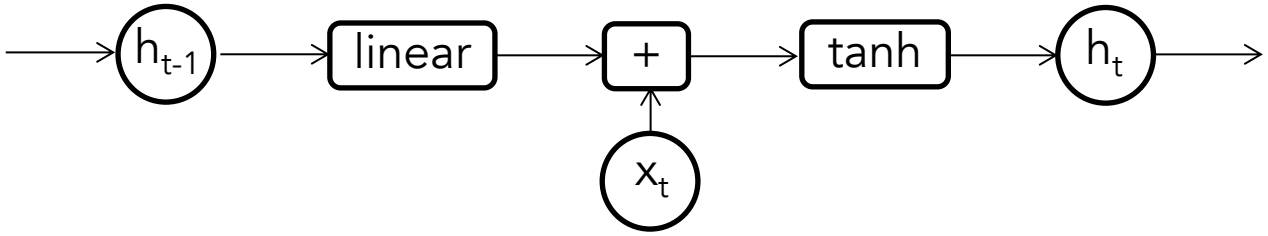
- Implicit Internal memory
  - Recurrent Neural Nets (RNNs)
  - Long-Short Term Memory (LSTMs)
- Explicit External memory
  - StackRNN
  - Memory Networks
  - Neural Turing Machine
- Attention models
  - MT, Speech, Image, Pointer Network
- Discrete Memory
  - Learning algorithms using 1-D tape, 2-D grid

Outline

- Implicit Internal memory
  - Recurrent Neural Nets (RNNs)
  - Long-Short Term Memory (LSTMs)
- Explicit External memory
  - StackRNN
  - Memory Networks
  - Neural Turing Machine
- Attention models
  - MT, Speech, Image, Pointer Network
- Discrete Memory
  - Learning algorithms using 1-D tape, 2-D grid

Implicit Internal Memory

- Internal state of the model can be used for memory
  - Recurrent Neural Networks (RNNs)



- Computation and memory is mixed
  - Complex computation requires many layers of non-linearity
  - But some information is lost with each non-linearity
  - Problems with vanishing/exploding gradients & catastrophic forgetting

Ways to Prevent Forgetting in RNNs

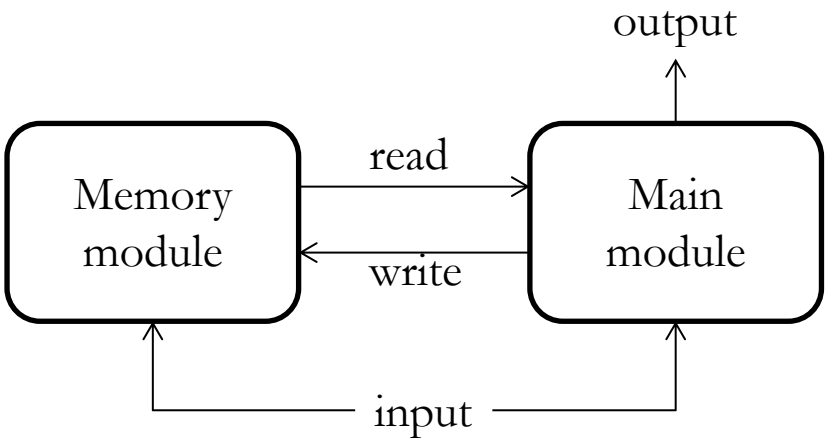
- Split state into fast and slow changing parts: structurally constrained recurrent nets (e.g. Mikolov et al., 2014)
  - Fast changing part is good for computation
  - Slow changing part is good for storing information
- Gated units for internal state
  - Control when to forget/write using gates
  - Long-short term memory (LSTM) (see Graves, 2013)
  - Simpler Gated Recurrent Unit (GRU) (Cho et al., 2014)
- Other problems
  - Memory capacity is fixed and limited by the dimension of state vector (computation is $O(N^2)$ where N is memory capacity)
  - Vulnerable to distractions in inputs
  - Restricted to sequential inputs

Outline

- Implicit Internal memory
  - Recurrent Neural Nets (RNNs)
  - Long-Short Term Memory (LSTMs)
- Explicit External memory
  - StackRNN
  - Memory Networks
  - Neural Turing Machine
- Attention models
  - MT, Speech, Image, Pointer Network
- Discrete Memory
  - Learning algorithms using 1-D tape, 2-D grid

External Global Memory

- Separate memory from computation
  - Add separate memory module for storage
  - Memory contains list/set of items



- Main module can read and write to the memory
- Advantage: long-term, scalable, flexible

Selective Addressing is Key for Memory

- Often, you only want to interact with few items in memory at once
  - Memory needs some addressing mechanism
- Memory addressing types
  - Soft or hard addressing
    - Soft addressing can be trained by backpropagation
    - Hard addressing is not differentiable (e.g. has to be trained with reinforcement learning or additional training signal for where to attend)
  - Context and Location based addressing
    - When input is ordered in some way, location based addressing is useful
    - Location addressing is same as context if location is embedded in the context (e.g. MemN2N)

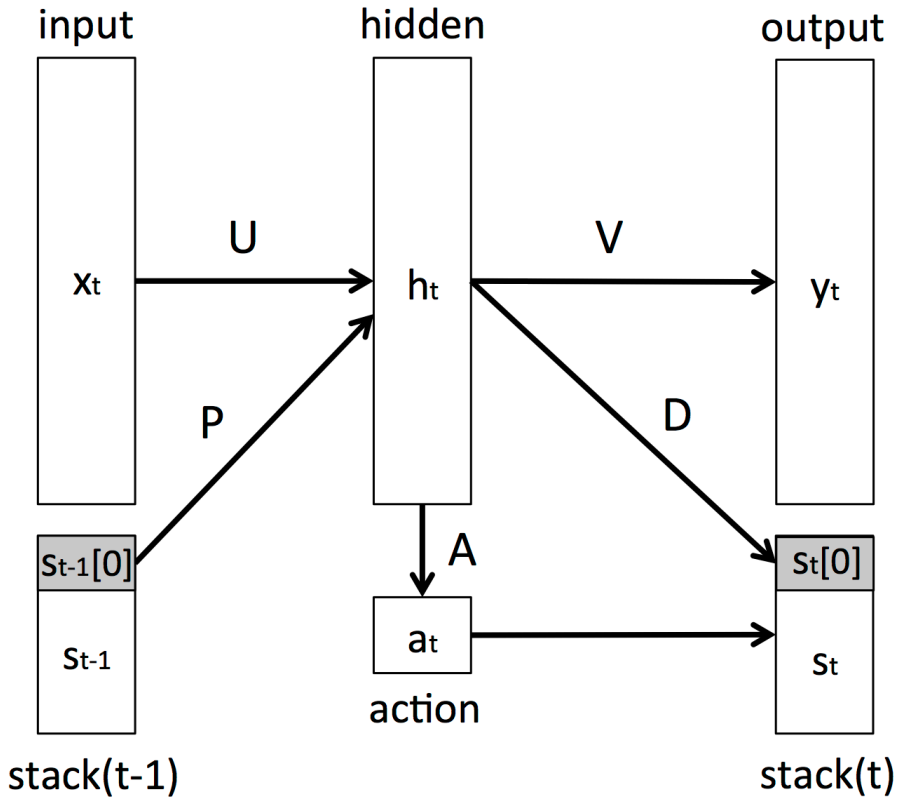
Stack RNNs (Joulin & Mikolov, 2015)

- Simple RNN extended with a stack that the neural net learns to control
- The idea itself is very old (from 80's – 90's)
- Very simple and learns complex toy patterns with much less supervision & scales to more complex tasks

Stack RNN

- Add structured memory to RNN:
  - Trainable [read/write]
  - Unbounded
- Continuous actions:
  PUSH / POP / NO-OP
- Multiple stacks

- Examples of memory structures:
  stacks, lists, queues, tapes, grids, …
- Learns algorithms from examples



Stack RNN - Algorithmic Patterns

Sequence generator	Example
$\{a^n b^n \mid n > 0\}$	aab**ba**aab**bba**b**a**aaaab**bbbb**
$\{a^n b^n c^n \mid n > 0\}$	aaab**bbccca**b**ca**aaaab**bbbbccccc**
$\{a^n b^n c^n d^n \mid n > 0\}$	aab**bccdda**aab**bbccccddda**b**cd**
$\{a^n b^{2n} \mid n > 0\}$	aab**bbba**aab**bbbbba**b**b**
$\{a^n b^m c^{n+m} \mid n, m > 0\}$	aabc**cca**aabbc**ccccca**bc**c**
$n \in [1, k],\ X \rightarrow nXn,\ X \rightarrow =$	$(k = 2)$ 12=**21**2122=**2212**11121=**12111**

- Examples of simple algorithmic patterns generated by short programs (grammars)
- The goal is to learn these patterns in an **unsupervised** manner just by observing the example sequences

Stack RNN - Example

- Sequence: a^6b^{12}

current	next	prediction	proba(next)	action		stack1[top]	stack2[top]
b	a	a	0.99	POP	POP	-1	0.53
a	a	a	0.99	PUSH	POP	0.01	0.97
a	a	a	0.95	PUSH	PUSH	0.18	0.99
a	a	a	0.93	PUSH	PUSH	0.32	0.98
a	a	a	0.91	PUSH	PUSH	0.40	0.97
a	a	a	0.90	PUSH	PUSH	0.46	0.97
a	b	a	0.10	PUSH	PUSH	0.52	0.97
b	**b**	**b**	0.99	PUSH	PUSH	0.57	0.97
b	**b**	**b**	1.00	POP	PUSH	0.52	0.56
b	**b**	**b**	1.00	POP	PUSH	0.46	0.01
b	**b**	**b**	1.00	POP	PUSH	0.40	0.00
b	**b**	**b**	1.00	POP	PUSH	0.32	0.00
b	**b**	**b**	1.00	POP	PUSH	0.18	0.00
b	**b**	**b**	0.99	POP	PUSH	0.01	0.00
b	**b**	**b**	0.99	POP	POP	-1	0.00
b	**b**	**b**	0.99	POP	POP	-1	0.00
b	**b**	**b**	0.99	POP	POP	-1	0.00
b	**b**	**b**	0.99	POP	POP	-1	0.01
b	**a**	**a**	0.99	POP	POP	-1	0.56

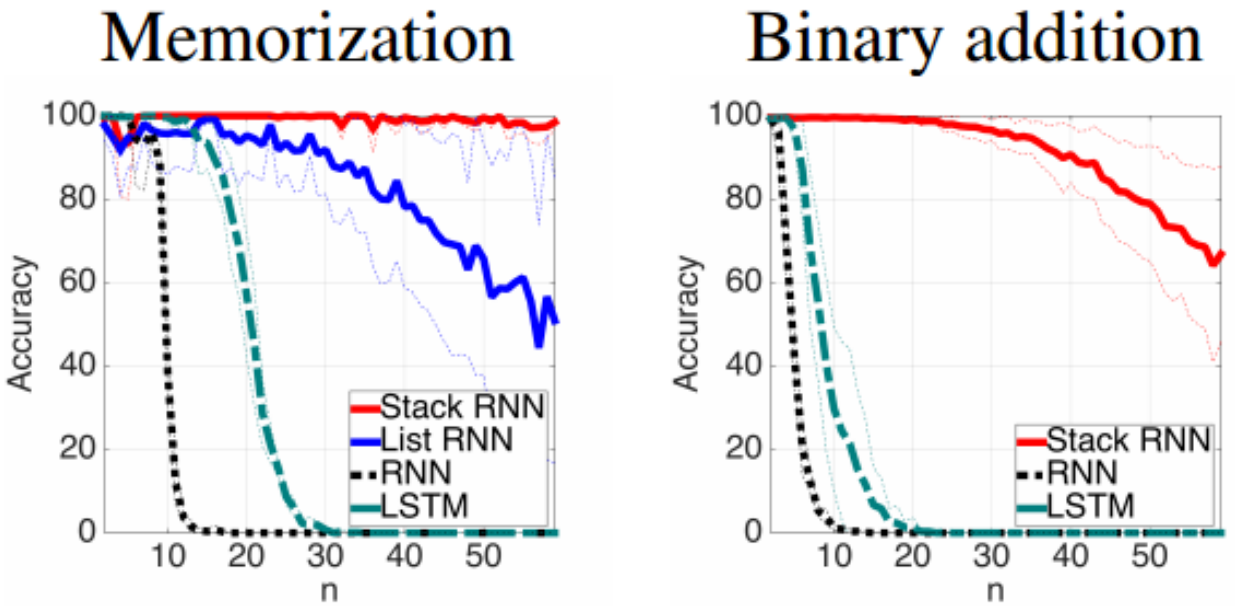
Table 3: Example of the Stack RNN with 20 hidden units and 2 stacks on a sequence $a^n b^{2n}$ with $n = 6$. -1 means that the stack is empty. The depth k is set to 1 for clarity. We see that the first stack pushes an element every time it sees a and pop when it sees b. The second stack pushes when it sees a. When it sees b , it pushes if the first stack is not empty and pop otherwise. This shows how the two stacks interact to correctly predict the deterministic part of the sequence (shown in bold).

Algorithmic Patterns - Counting

method	$a^n b^n$	$a^n b^n c^n$	$a^n b^n c^n d^n$	$a^n b^{2n}$	$a^n b^m c^{n+m}$
RNN	25%	23.3%	13.3%	23.3%	33.3%
LSTM	100%	100%	68.3%	75%	100%
List RNN 40+5	100%	33.3%	100%	100%	100%
Stack RNN 40+10	100%	100%	100%	100%	43.3%
Stack RNN 40+10 + rounding	100%	100%	100%	100%	100%

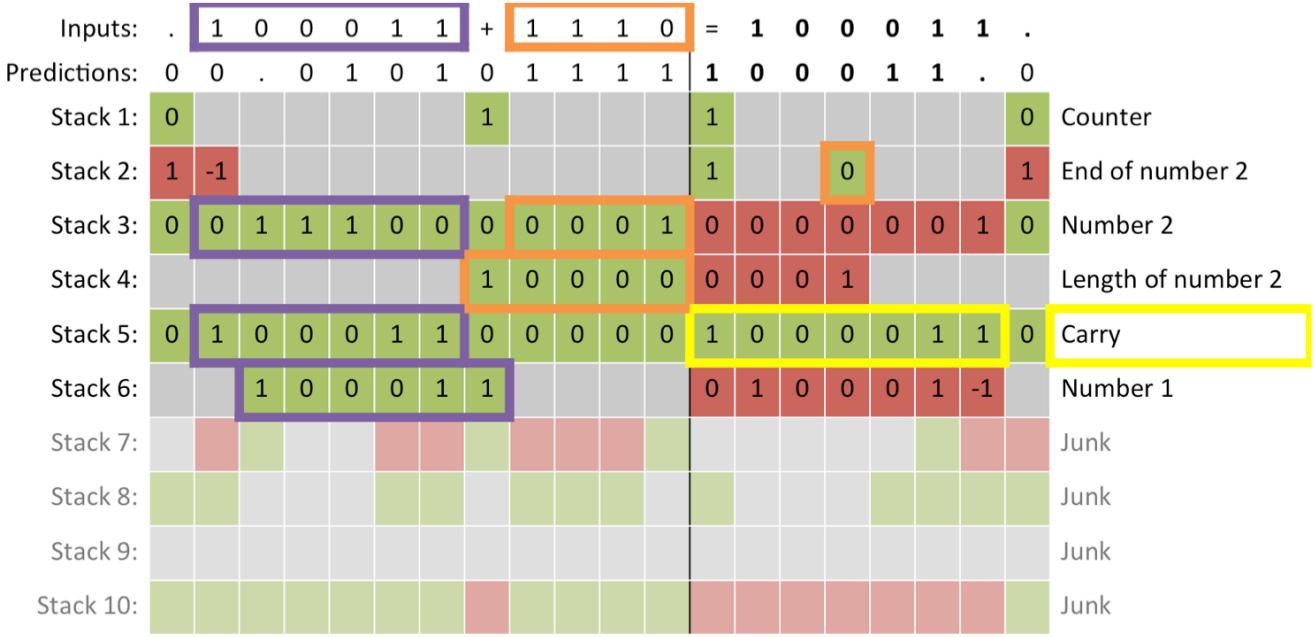
- Performance on simple counting tasks
- RNN with sigmoidal activation function cannot count
- Stack-RNN and LSTM can count

Algorithmic Patterns - Sequences



- Sequence memorization and binary addition are out-of-scope of LSTM
- Expandable memory of stacks allows to learn the solution

Stack RNN - Binary Addition



- **No supervision in training, just prediction**
- Learns to: store digits, when to produce output, carry

Stack RNNs: summary

The good:

- Turing-complete model of computation (with >=2 stacks)
- Learns some algorithmic patterns
- Has long term memory
- Works for some problems that break RNNs and LSTMs
- Reproducible: https://github.com/facebook/Stack-RNN

The bad:

- The long term memory is used only to store partial computation (ie. learned skills are not stored there yet)
- Does not seem to be a good model for incremental learning due to computational inefficiency of the model
- Stacks do not seem to be a very general choice for the topology of the memory

Outline

- Implicit Internal memory
  - Recurrent Neural Nets (RNNs)
  - Long-Short Term Memory (LSTMs)
- Explicit External memory
  - StackRNN
  - Memory Networks
  - Neural Turing Machine
- Attention models
  - MT, Speech, Image, Pointer Network
- Discrete Memory
  - Learning algorithms using 1-D tape, 2-D grid

NEW YORK UNIVERSITY

Facebook AI Research

End-To-End Memory Networks

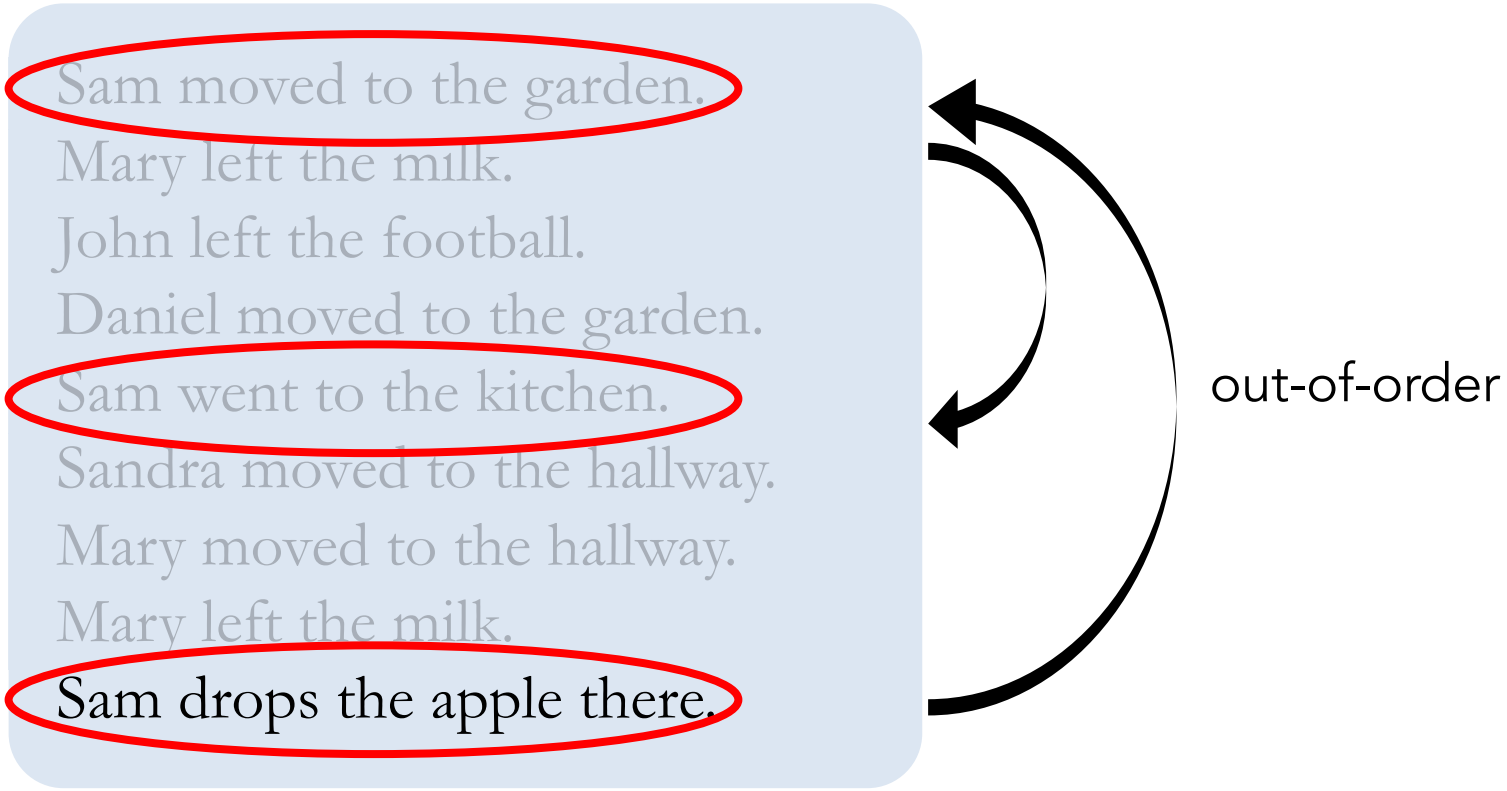
Sainbayar Sukhbaatar[1], Arthur Szlam[2], Jason Weston[2] and Rob Fergus[2]

[1]New York University [2]Facebook AI Research

Motivation

- Good models exist for some data structures
  - RNN for temporal structure
  - ConvNet for spatial structure
- But we still struggle with some type of dependencies
  - out-of-order access
  - long-term dependency
  - unordered set

Ex) Question & Answering on story



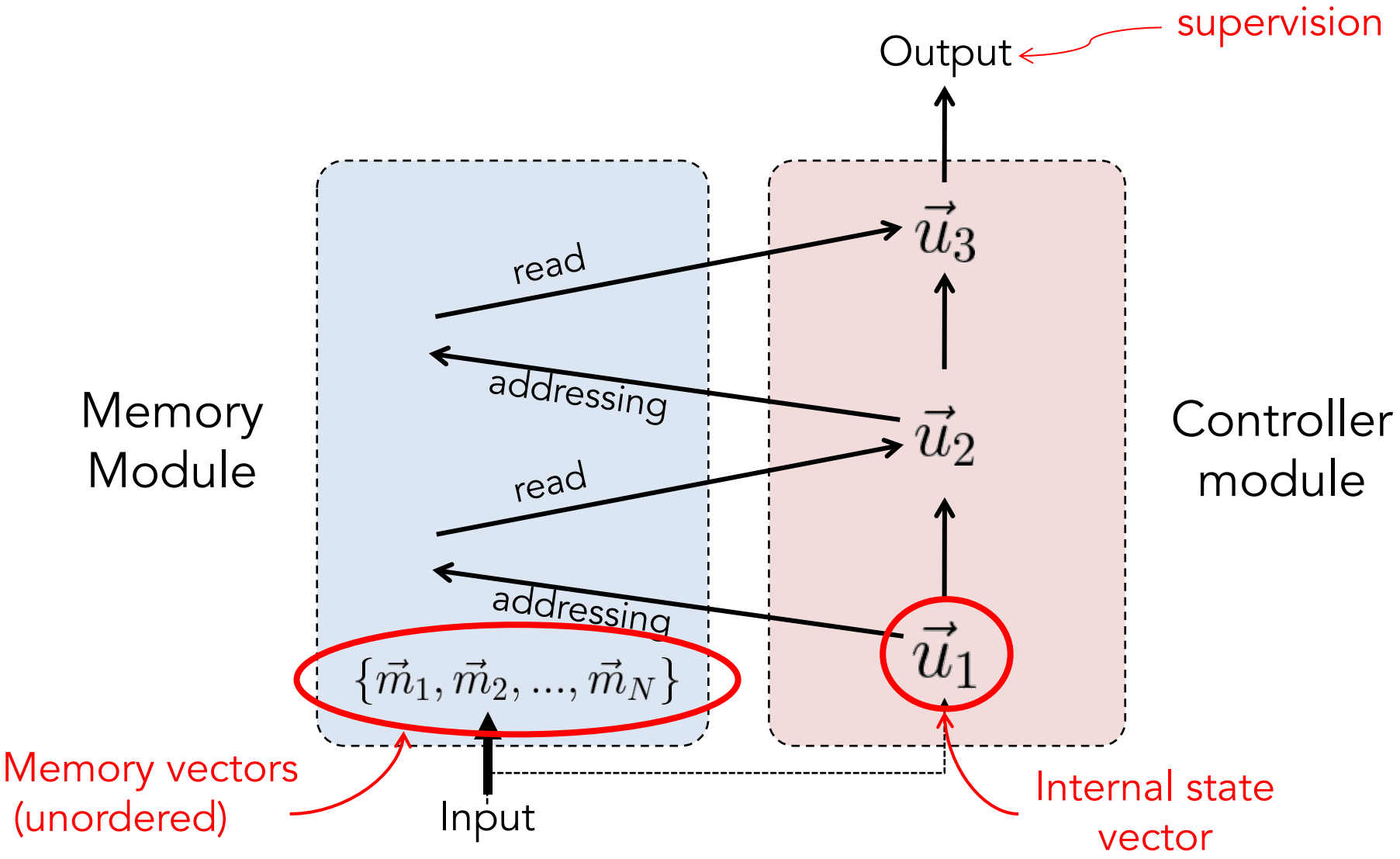
Q: Where was the apple after the garden?

Overview

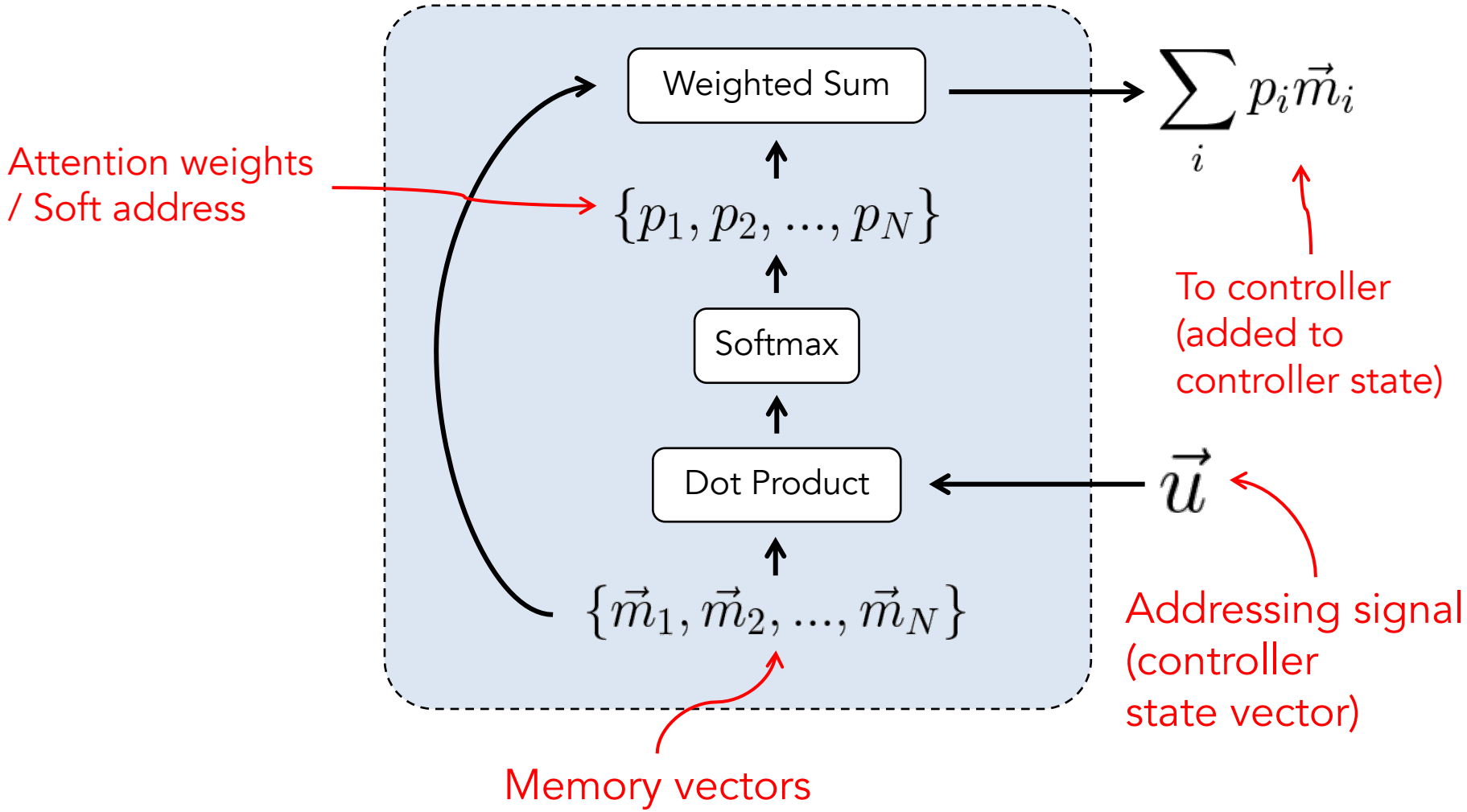
- We propose a neural network model with external memory

- It is based on “Memory Networks” by [Weston, Chopra & Bordes ICLR 2015]
  - Hard attention
  - requires explicit supervision of attention during training
  - Only feasible for simple tasks
  - Severely limits application of the model

MemN2N architecture



Memory Module



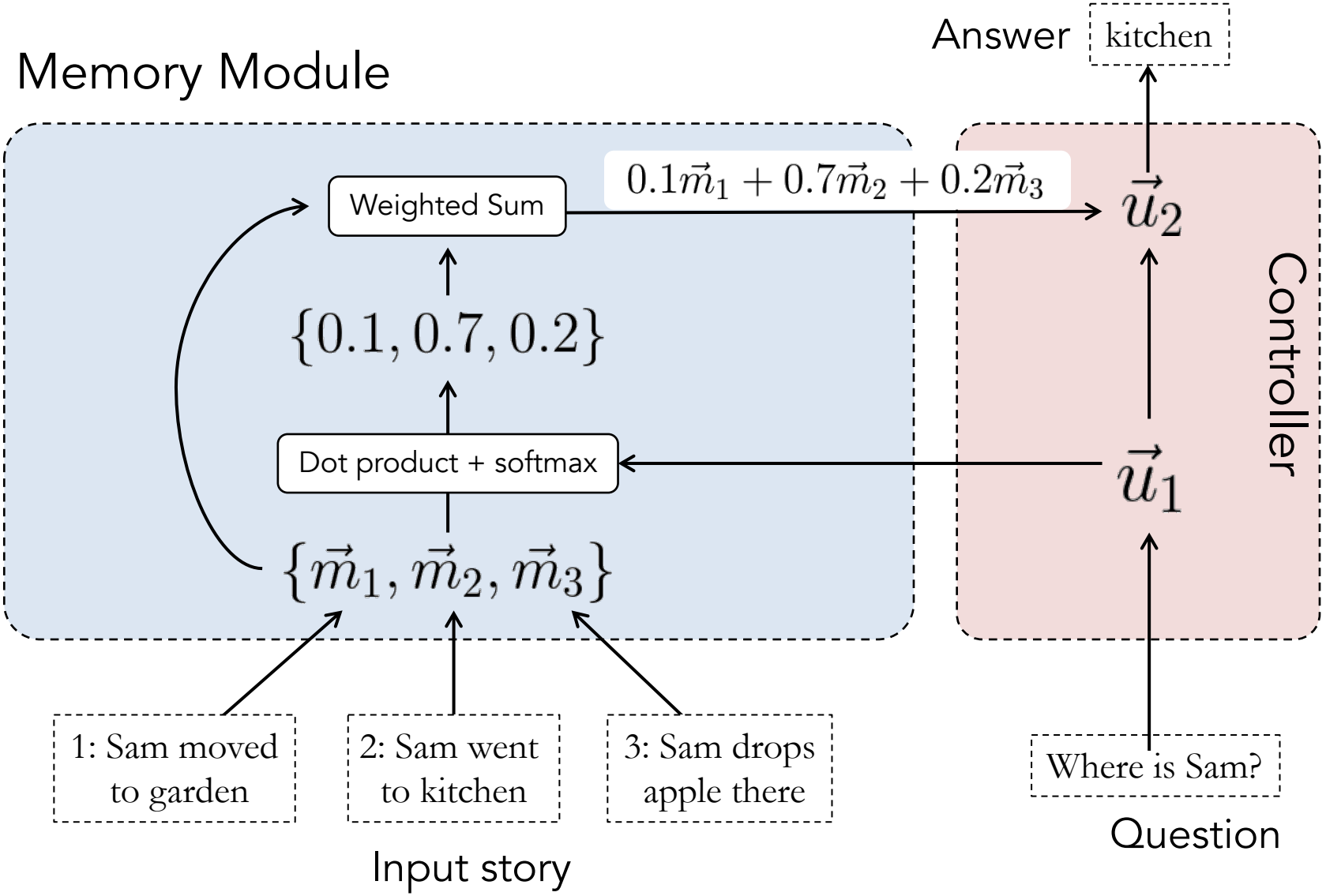
Memory Vectors

E.g.) constructing memory vectors with Bag-of-Words (BoW)

1. Embed each word
2. Sum embedding vectors

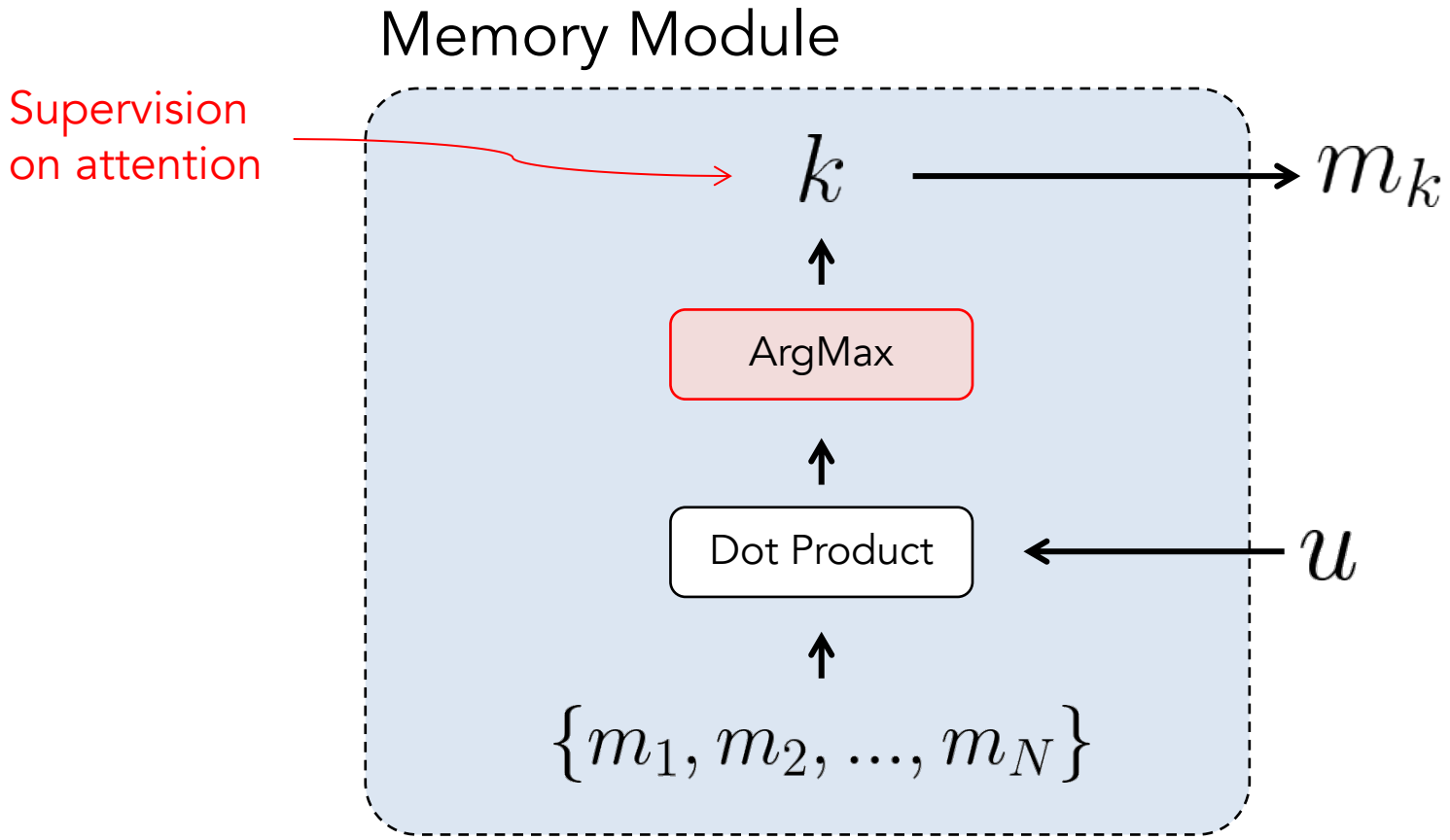
"Sam drops apple"

Question & Answering



Related Work (I)

Hard attention Memory Network [Weston et al. ICLR 2015]



Related Work (II)

- RNNsearch [Bahdanau et al. 2015]
  - Encoder-decoder RNN with attention
  - Our model can be considered as an attention model with multiple hops
- Recent works on external memory
  - Stack memory for RNNs [Joulin & Mikolov. 2015]
  - Neural Turing Machine [Graves et al. 2014]
- Early works on neural network and memory
  - [Steinbuch & Piske. 1963]; [Taylor. 1959]
  - [Das et al. 1992]; [Mozer et al. 1993]
- Concurrent works
  - Dynamic Memory Networks [Kumar et al. 2015]
  - Attentive reader [Hermann et al. 2015]
  - Stack, Queue [Grefenstette et al. 2015]

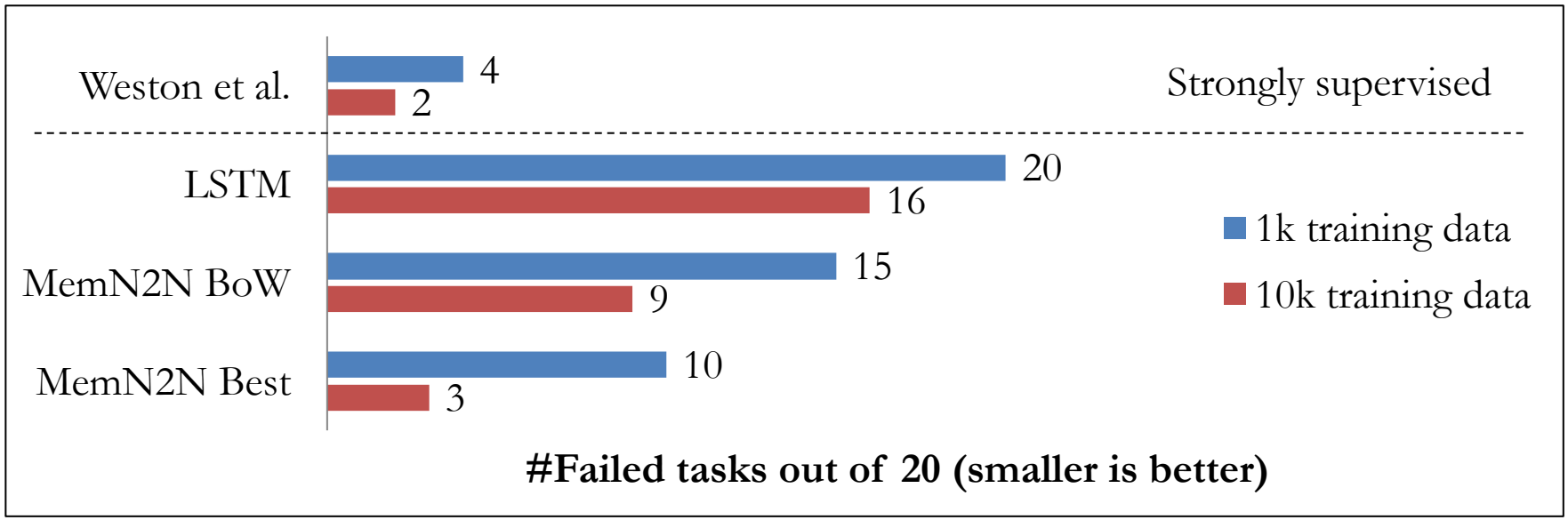
Experiment on bAbI Q&A data

- Data: 20 bAbI tasks [Weston et al. arXiv: 1502.05698, 2015]
- Answer questions after reading short story
- Small vocabulary, simple language
- Different tasks require different reasoning
- Training data size 1K or 10K for each task

```
Sam walks into the kitchen.
Sam picks up an apple.
Sam walks into the bedroom.
Sam drops the apple.
Q: Where is the apple?
A. Bedroom
```

```
Brian is a lion.
Julius is a lion.
Julius is white.
Bernhard is green.
Q: What color is Brian?
A. White
```

Performance on bAbI test set



Examples of Attention Weights

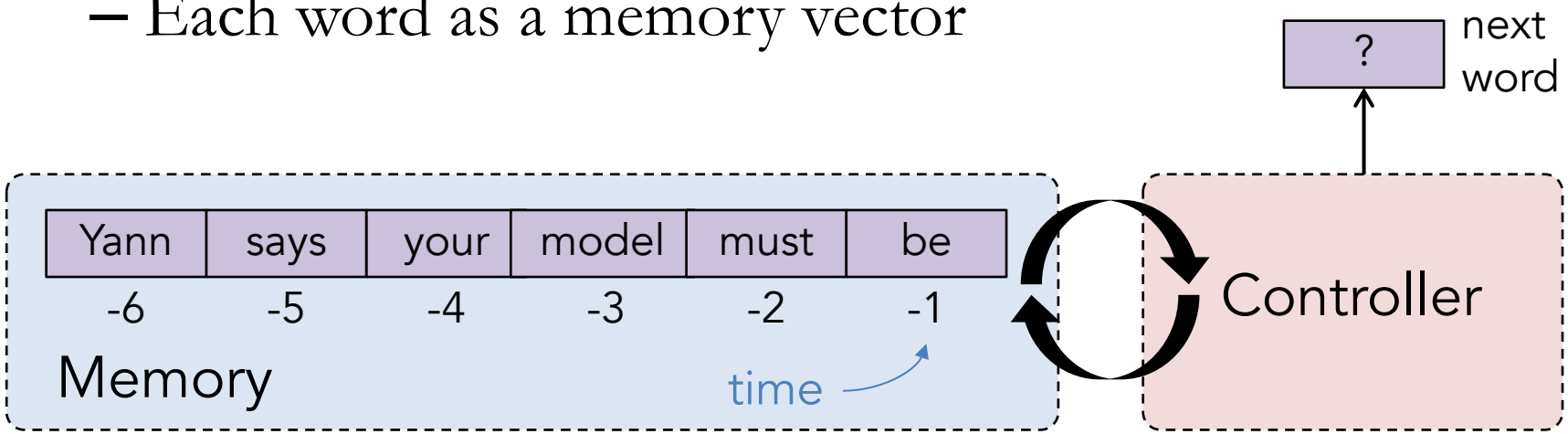
- 2 test cases:

Story (2: 2 supporting facts)	Hop 1	Hop 2	Hop 3
John dropped the milk.	0.06	0.00	0.00
John took the milk there.	0.88	1.00	0.00
Sandra went back to the bathroom.	0.00	0.00	0.00
John moved to the hallway.	0.00	0.00	1.00
Mary went back to the bedroom.	0.00	0.00	0.00
Where is the milk? Answer: hallway Prediction: hallway			

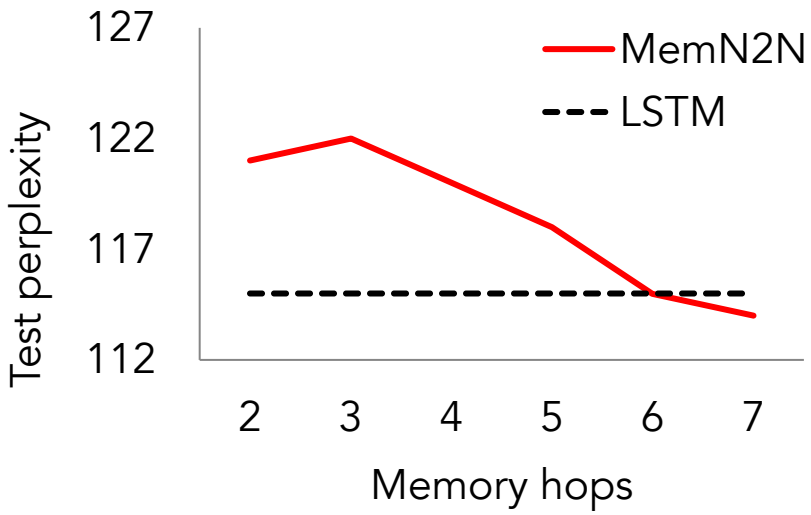
Story (16: basic induction)	Hop 1	Hop 2	Hop 3
Brian is a frog.	0.00	0.98	0.00
Lily is gray.	0.07	0.00	0.00
Brian is yellow.	0.07	0.00	1.00
Julius is green.	0.06	0.00	0.00
Greg is a frog.	0.76	0.02	0.00
What color is Greg? Answer: yellow Prediction: yellow			

Experiment on Language modeling

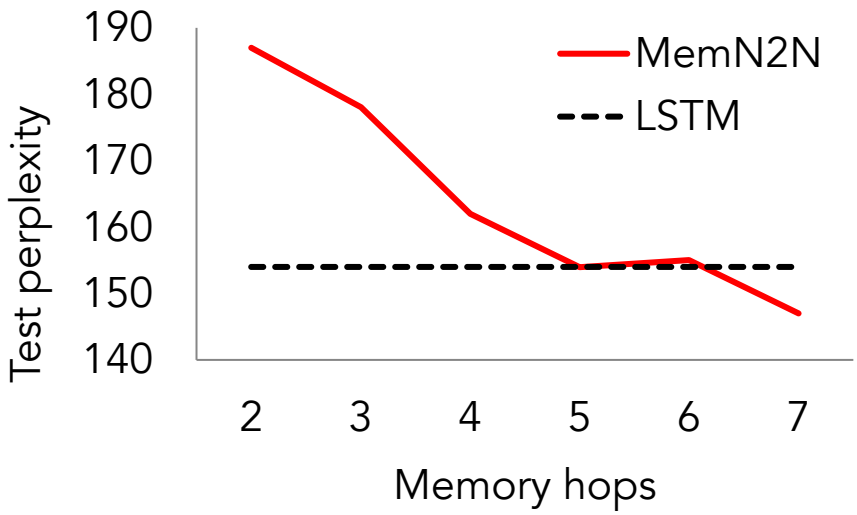
- Data
  - Penn Treebank: 1M words 10K vocab
  - Text8 (Wikipedia): 16M words 40K vocab
- Model
  - Controller module: linear + non-linearity
  - Each word as a memory vector



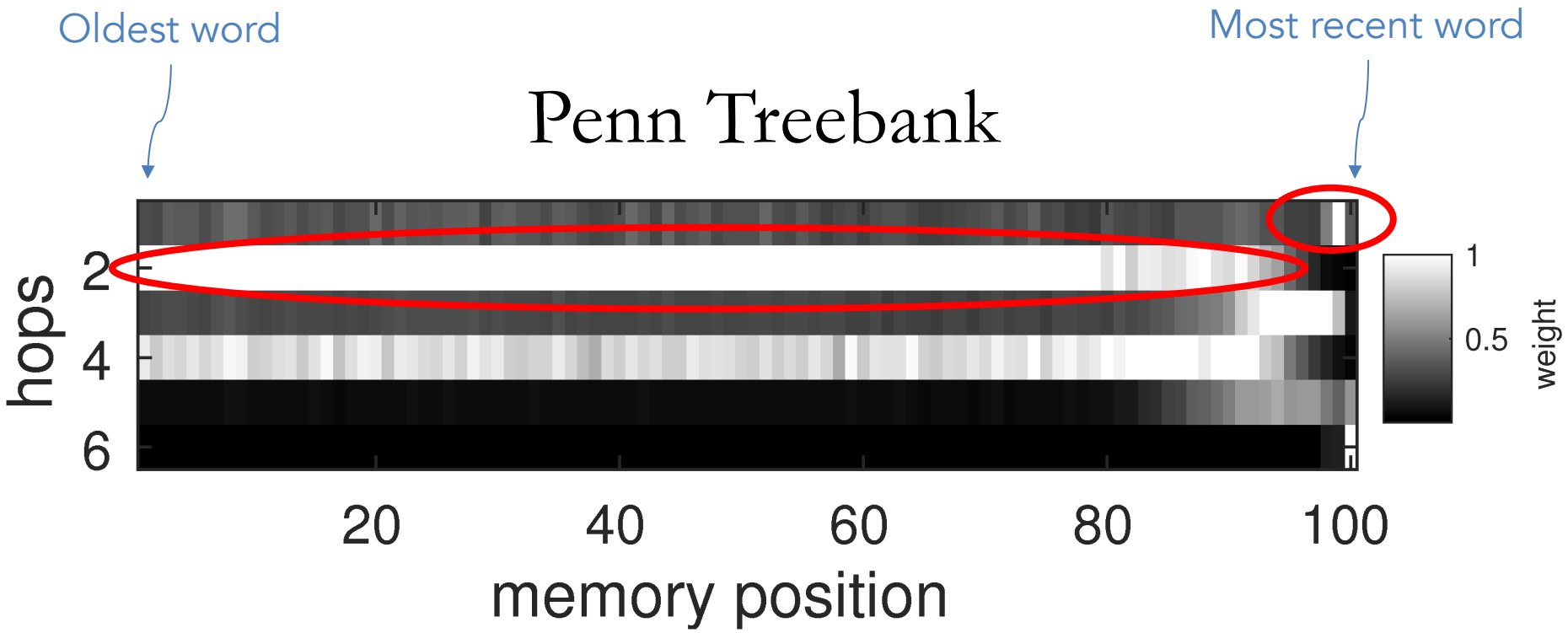
Penn-Treebank



Text8 (Wikipedia)

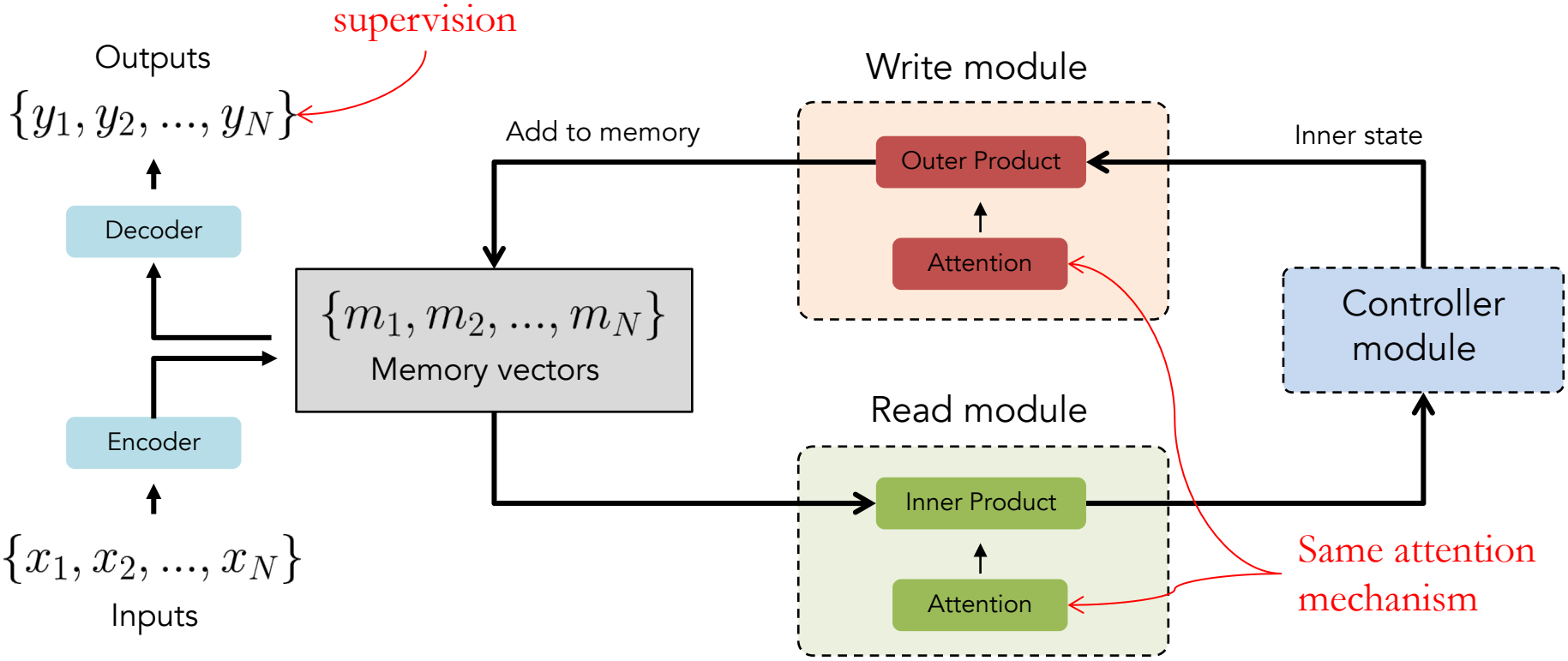


Attention during memory hops



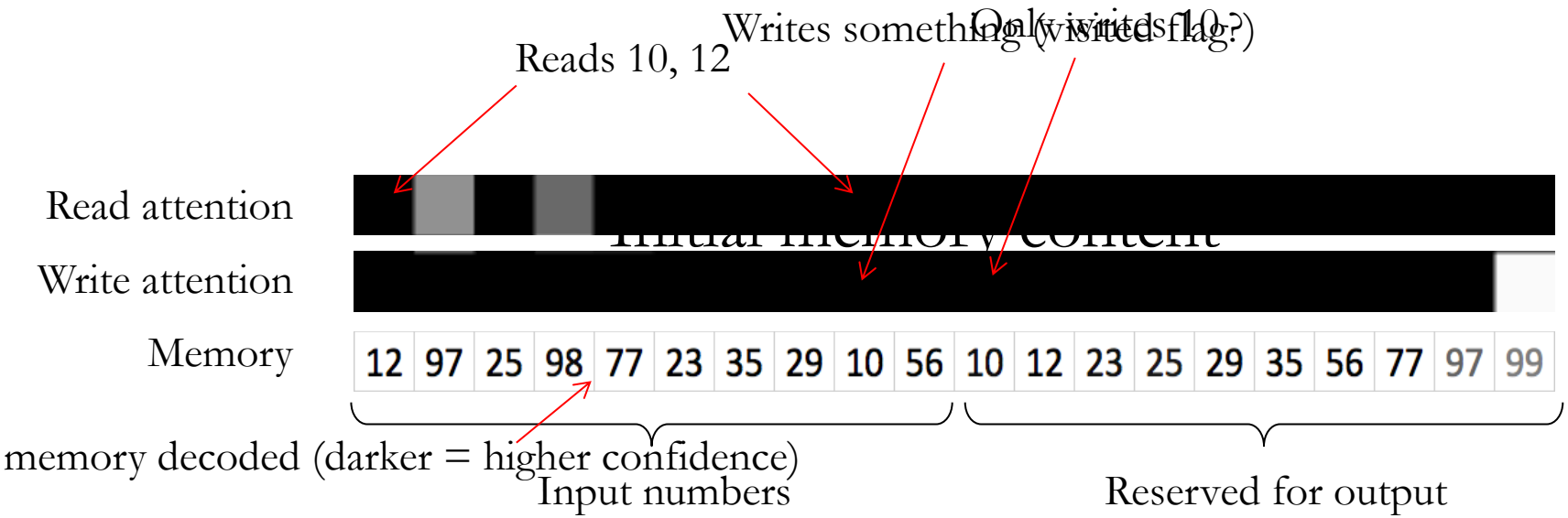
Extension to writable memory

- Every memory location is readable and **writable**
- In each hop, perform **both** read and write
- Write module **adds** to the current memory
- **N** inputs and **N** outputs and **N** memory slots



Learning to sort in memory

- Train MemN2N to sort given numbers
- Input: 10 random numbers
- Output: sorted version of input



Conclusion

- Proposed a neural net model with external memory
  - Soft attention over memory locations
  - End-to-end training with backpropagation
- Good results on a toy QA tasks
- Comparable to LSTM on language modeling
- Versatile model: also apply to writing and games

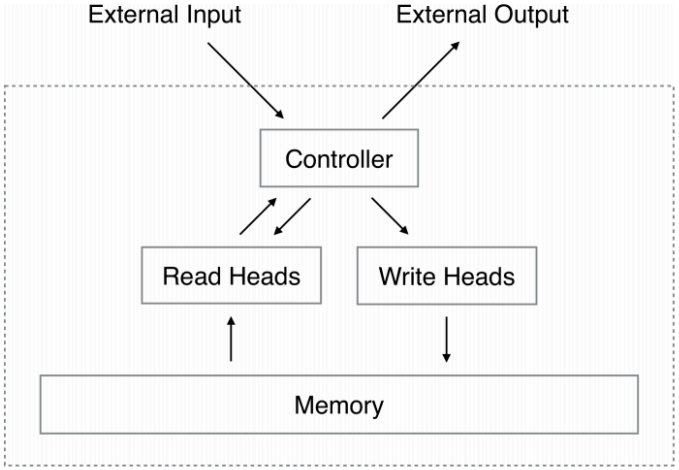
Code http://github.com/facebook/MemNN

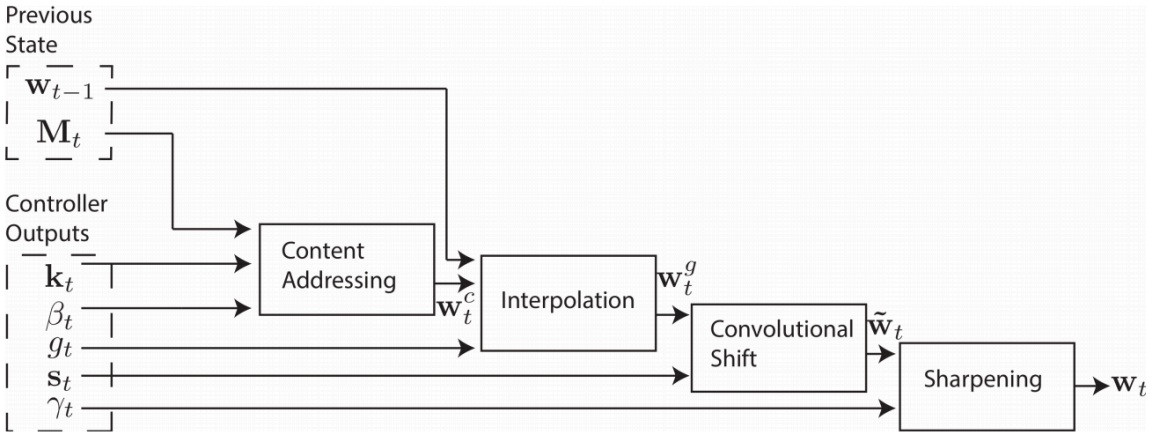
Outline

- Implicit Internal memory
  - Recurrent Neural Nets (RNNs)
  - Long-Short Term Memory (LSTMs)
- Explicit External memory
  - StackRNN
  - Memory Networks
  - Neural Turing Machine
- Attention models
  - MT, Speech, Image, Pointer Network
- Discrete Memory
  - Learning algorithms using 1-D tape, 2-D grid

Neural Turing Machine (Graves et al., 2014)

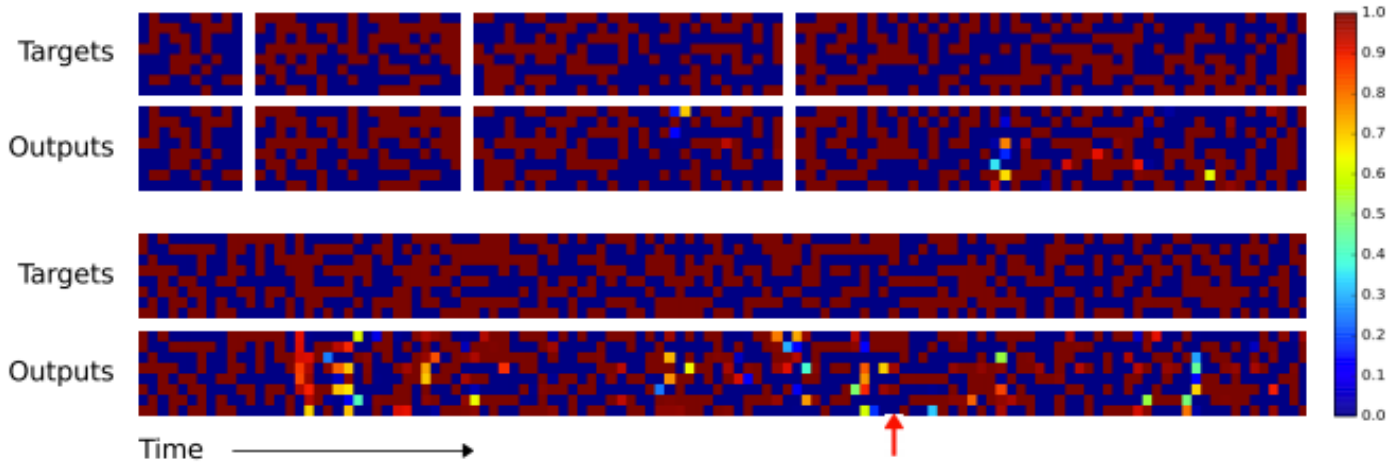
- Learns how to write to the memory
- Soft addressing → backpropagation training
- Location addressing: small continuous shift of attention
- Complex addressing mechanism: need to sharpen after convolution
- Controller can be LSTM-RNN or feed-forward neural network
- Applied to learn algorithms such as sort, associative recall and copy.
- Hard addressing with reinforcement learning (Zaremba et al., 2015)

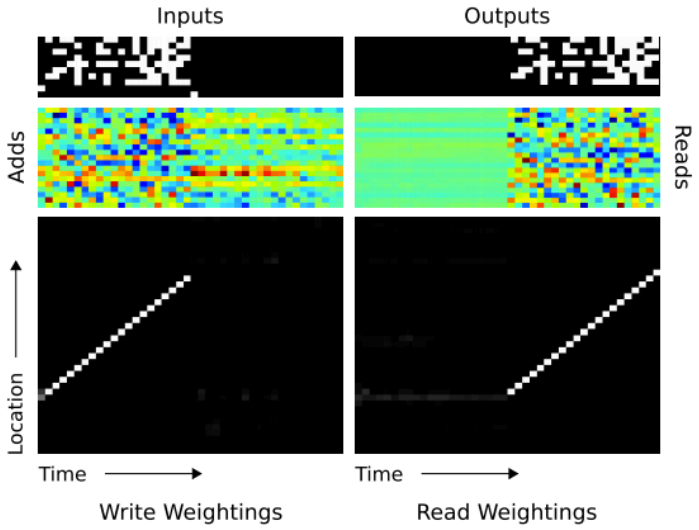




Neural Turing Machine – Copy task

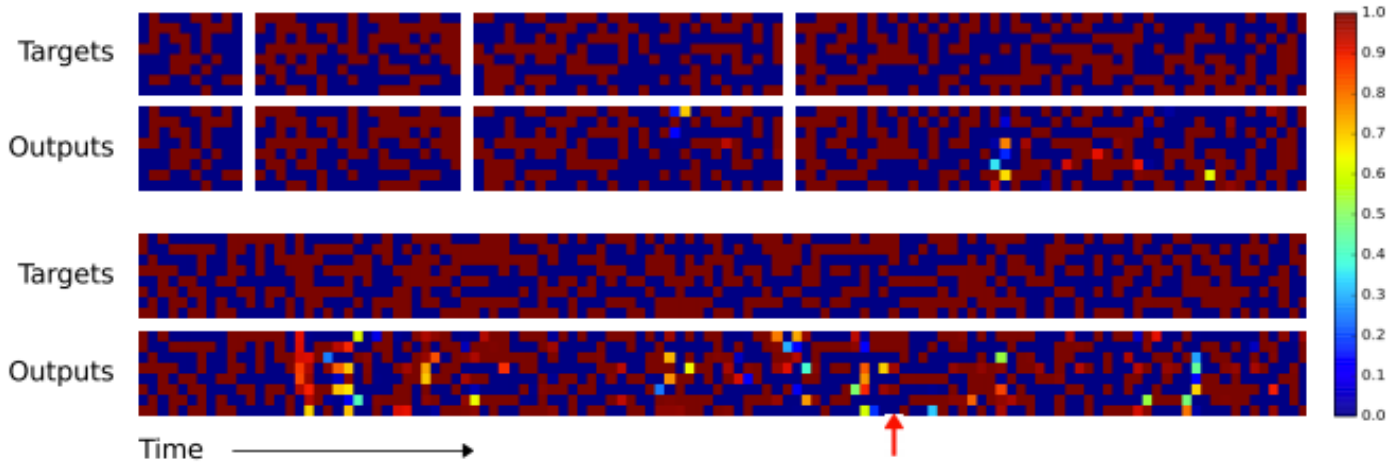
- NTM



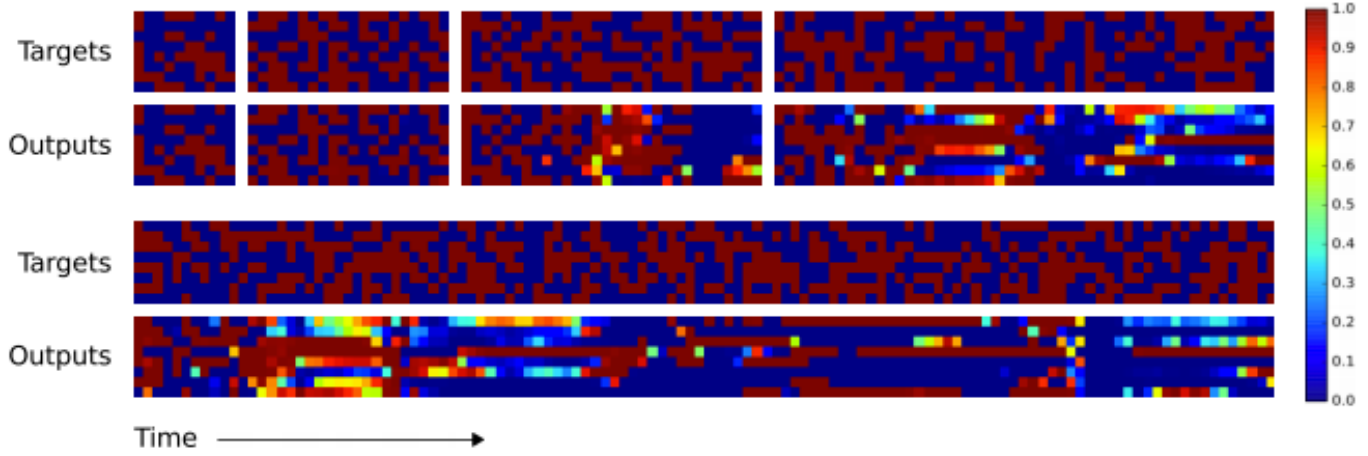


Neural Turing Machine – Copy task

- NTM



LSTM



Neural Turing Machine - Experiments

Task	#Heads	Controller Size	Memory Size	Learning Rate	#Parameters
Copy	1	100	128×20	10^{-4}	17,162
Repeat Copy	1	100	128×20	10^{-4}	16,712
Associative	4	256	128×20	10^{-4}	146,845
N-Grams	1	100	128×20	3×10^{-5}	14,656
Priority Sort	8	512	128×20	3×10^{-5}	508,305

Table 1: NTM with Feedforward Controller Experimental Settings

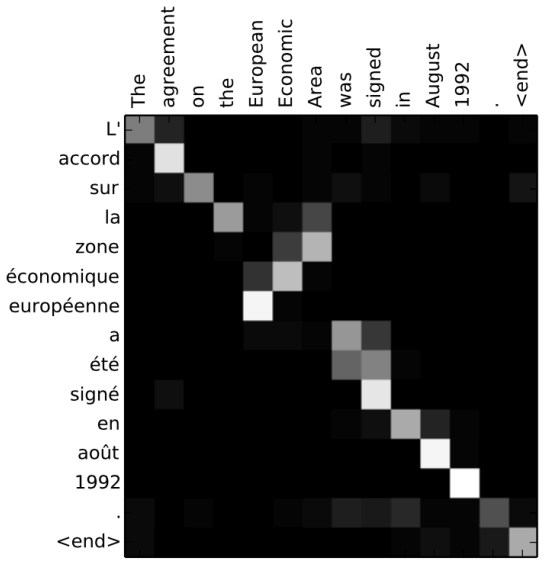
Outline

- Implicit Internal memory
  - Recurrent Neural Nets (RNNs)
  - Long-Short Term Memory (LSTMs)
- Explicit External memory
  - StackRNN
  - Memory Networks
  - Neural Turing Machine
- Attention models
  - MT, Speech, Image, Pointer Network
- Discrete Memory
  - Learning algorithms using 1-D tape, 2-D grid

RNNsearch: Attention in Machine Translation (Bahdanau et al., 2015)

- RNN based encoder and decoder model
- Decoder can look at past encoder states using soft attention
- Attention mechanism is implement by a small neural network
  - It takes the current decoder state and a past encoder state and outputs a score. Then the all scores are fed to softmax to get attention weights
- Applied to machine translation. Significant improvement in translation of longer sentences

Attention weights during English to French machine translation



Significant improvement on long sentences

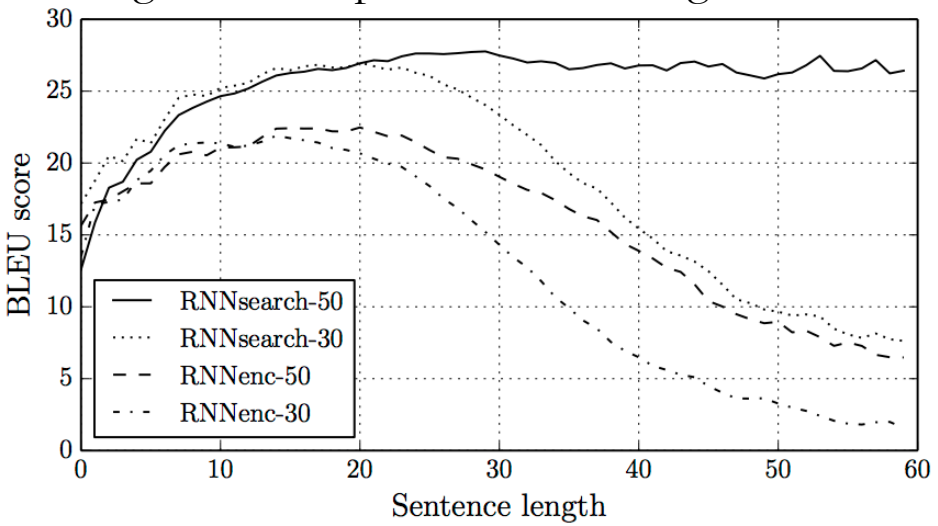


Image caption generation with attention (Xu et al., 2015)

- Encoder: lower convolutional layer of a deep ConvNet (because need spatial information)
- Decoder: LSTM RNN with soft spatial attention
  - Decoder state and encoder state at single location are fed to small NN to get score at that location
- Network attends to the object when it is generating a word for it
- Also hard attention is tried with reinforcement learning

A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

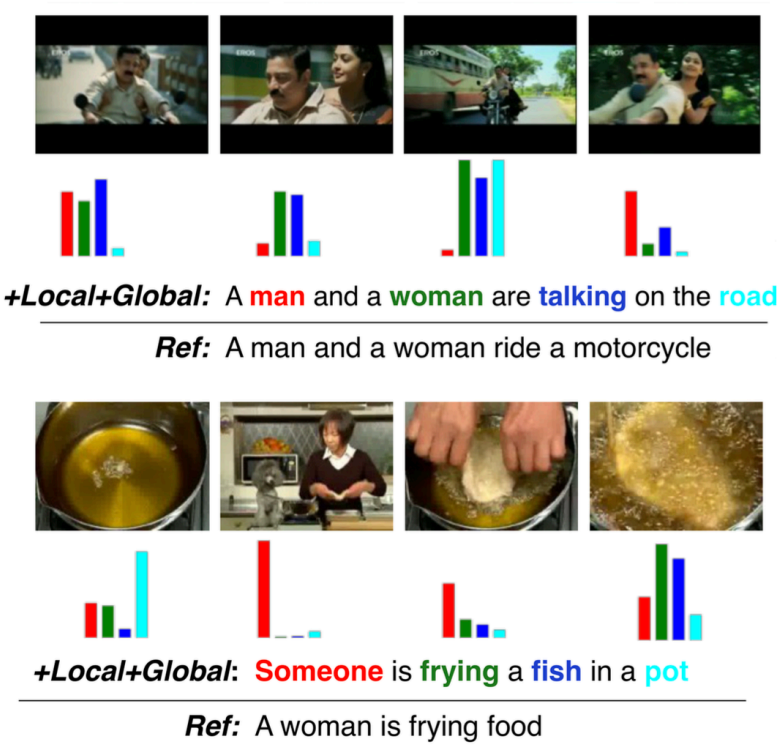
A stop sign is on a road with a mountain in the background.

A little girl sitting on a bed with a teddy bear.

A group of people sitting on a boat in the water.

A giraffe standing in a forest with trees in the background.

Video description generation (Yao et al., 2015)

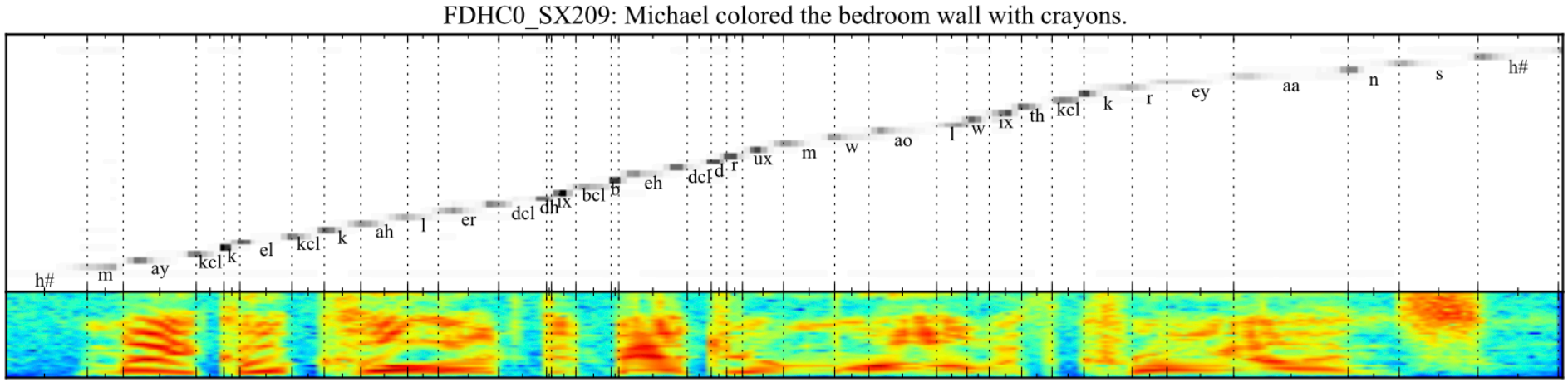


(bottom: ground truth)

L. Yao, A. Torabi, K. Cho, N. Ballas, C. Pal, H. Larochelle, and A. Courville, "Describing videos by exploiting temporal structure," *arXiv: 1502.08029*, 2015.

Location-aware attention for speech (Chorowski et al., 2015)

- RNN based encoder-decoder model with attention (similar to RNNsearch)
- Location based addressing: previous attention weights are used as feature for the current attention (good when subsequent attention locations are highly correlated)
- Improvement with sharpening and smoothing of memory addressing



Pointer Network: attention as an output (Vinyals et al., 2015)

- RNN based encoder-decoder model for discrete optimization problems
- Decoder can attend to previous encoder states (similar to RNNsearch, content based soft attention by a small NN)
- Rather than fixed output classes, attention weights determine output
- Input to the most attended encoder state becomes an output → can output any sequence of inputs