Deep Learning for NLP
Recurrent Neural Networks

Nils Reimers

TECHNISCHE UNIVERSITÄT DARMSTADT

Course-Website: www.deeplearning4nlp.com

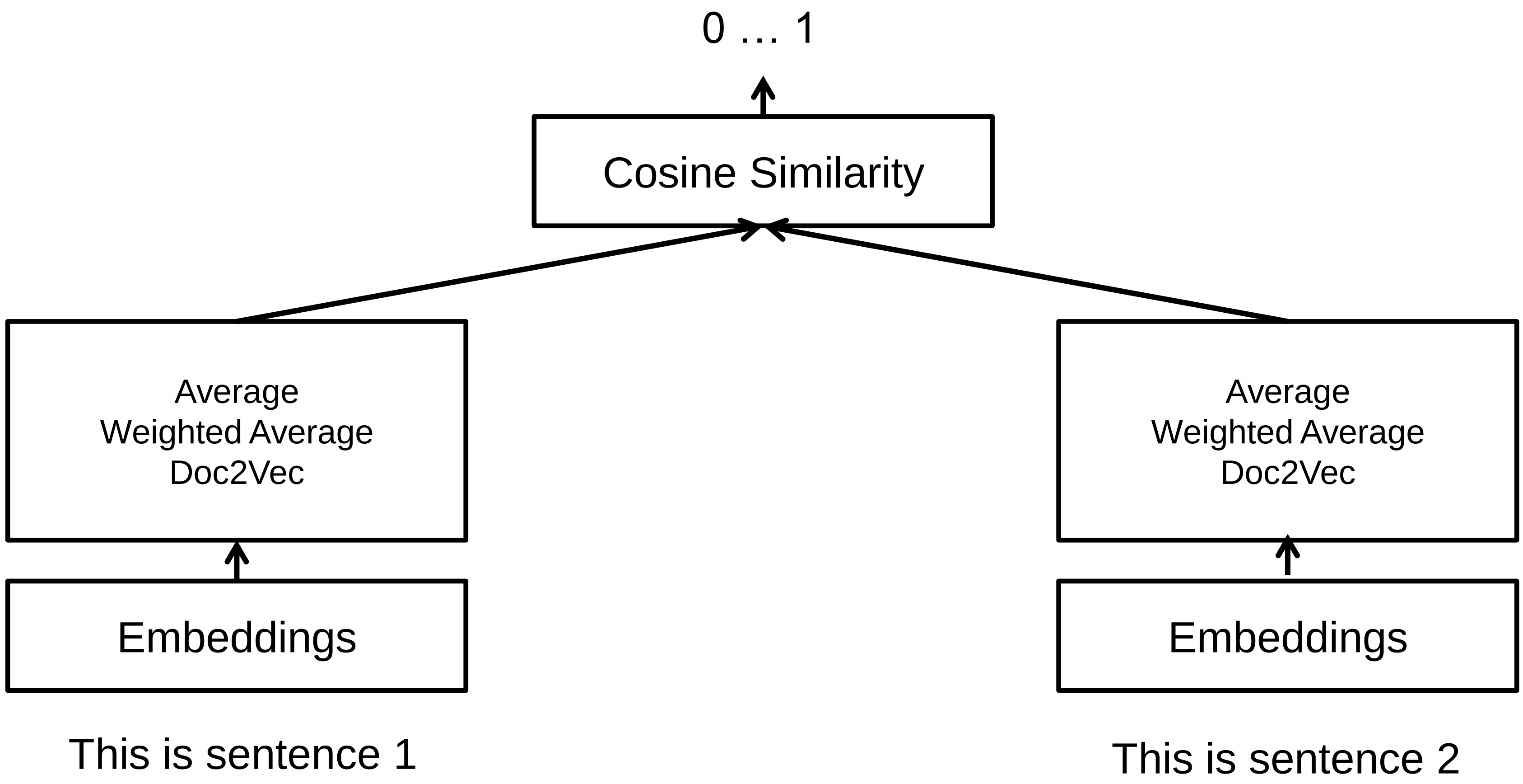
UKP

When you start with Deep Learning…

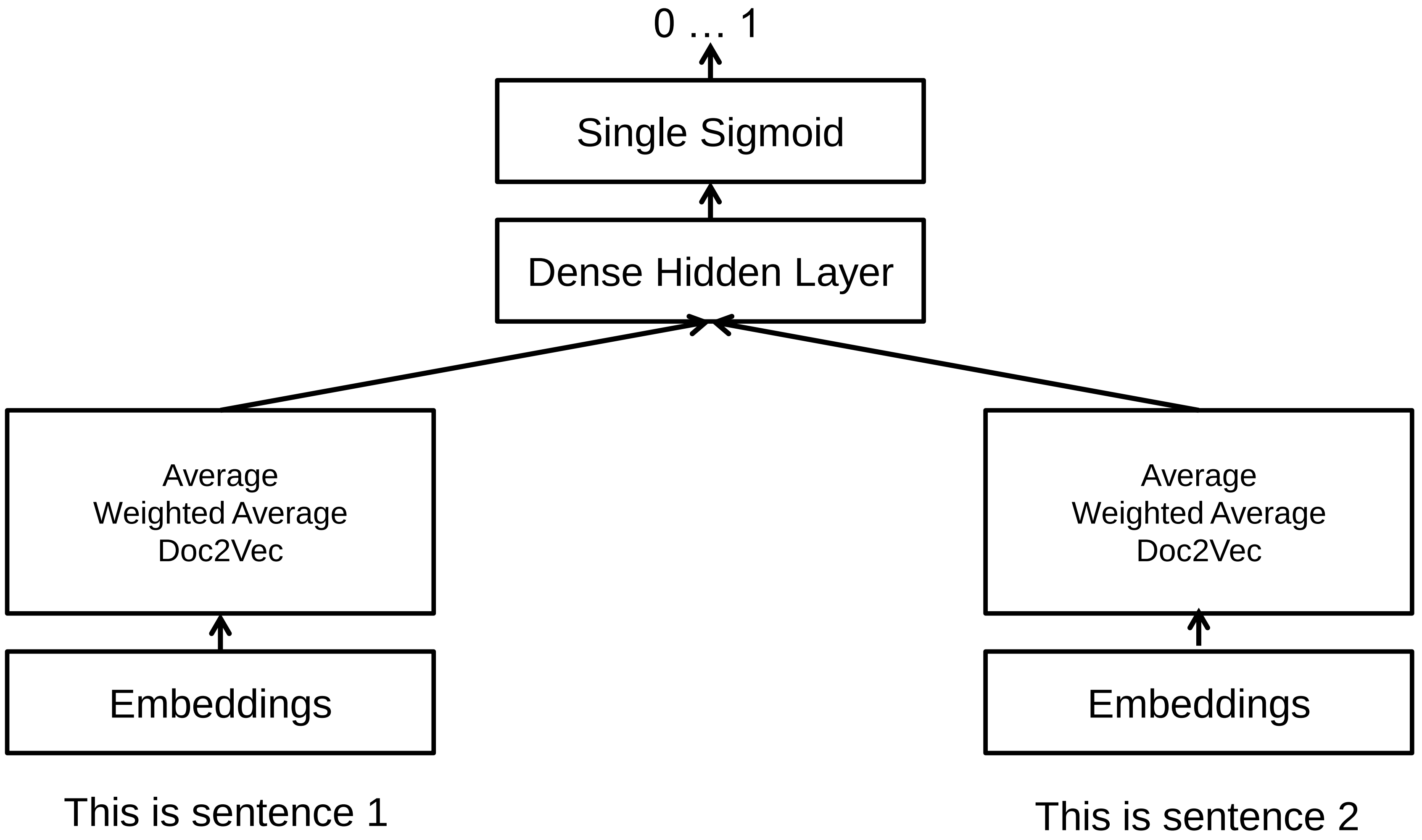
- … start with the most simple approach!
- These complex approaches sound quite cool, but …
  - … when you just adapt them, they will often fail to learn the task
  - … you have no idea why they will not produce reasonable output
  - … there is no easy way to debug complex Deep Neural Networks

- The difference between simple and complex networks are not so large
  - Results on Stanford Sentiment Treebank (Socher 2013):
    - Average of word embeddings: 73.3%
    - Recursive Network: 79.0%
    - Recursive Neural Tensor Networks: 80.7%

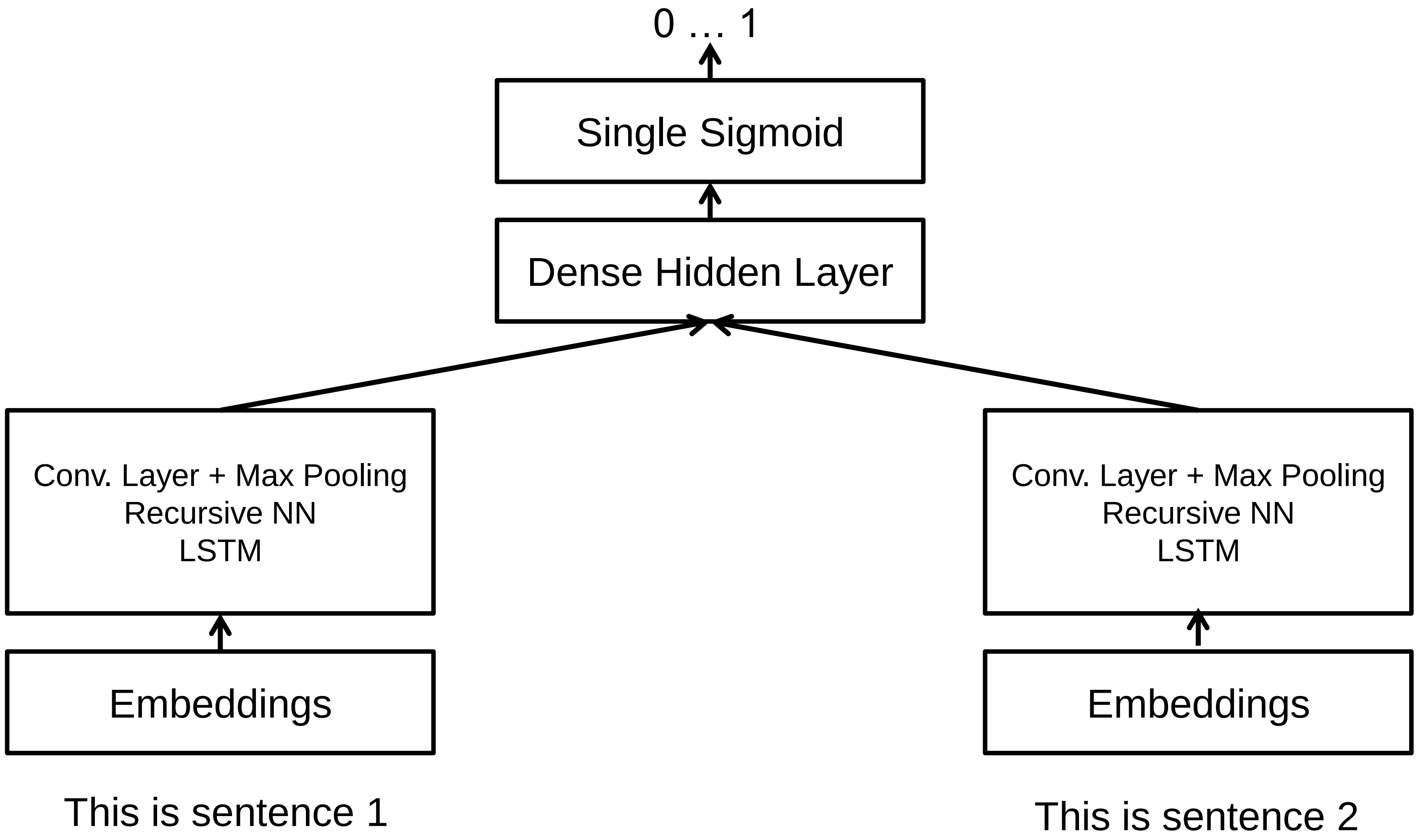
Semantic Text Similarity



Semantic Text Similarity



Semantic Text Similarity



Recommended Readings

- http://karpathy.github.io/2015/05/21/rnn-effectiveness/
  - Video: https://skillsmatter.com/skillscasts/6611-visualizing-and-understanding-recurrent-networks
- http://colah.github.io/posts/2015-08-Understanding-LSTMs/
- C224d Lecture 7: https://www.youtube.com/watch?v=rFVYTydGLr4

Models Overview

- Word Classifcation
  - SENNA
  - Convolutional Neural Networks
  - Recurrent Neural Networks
- Sentence Classification
  - Recursive Neural Networks,
  - Convolutional Neural Networks
  - Recurrent Neural Networks
- Document Classification
  - Bag of Words with Deep Feed Forward Network
  - (Convolutional Neural Network)
  - Recurrent Neural Network
- Generative Models
  - Recurrent Neural Network

Excursion: Language Model

- Compute the probability of a sentence
- Useful in machine translation
  - Word ordering: p(the cat is small) > p(small the cat is)
  - Word choice: p(walking home after school) > p(walking house after school)

Excursion: Language Model

- unigram language model:

$$P(w_i|w_{i-1}) \qquad\qquad P(w_i|w_{i-1}) = \frac{\text{count}(w_{i-1}w_i)}{\text{count}(w_{i-1})}$$

- bigram language model:

$$P(w_i|w_{i-1}, w_{i-2}) \qquad\qquad P(w_i|w_{i-1}, w_{i-2}) = \frac{\text{count}(w_{i-2}w_{i-1}w_i)}{\text{count}(w_{i-2}w_{i-1})}$$

- Such models can also be used to generate new sentences
  - Sample word w_i with probability $P(w_i|w_{i-1}, w_{i-2})$
- Longer n-gram models give a better accuracy
  - Required training data &model size increases extremely
- Long term relationships impossible to capture
  - p(I grew up in *France* and lived there until I was 18. Therefore I speak fluent *French*) >
    p(I grew up in *France* and lived there until I was 18. Therefore I speak fluent *English*)

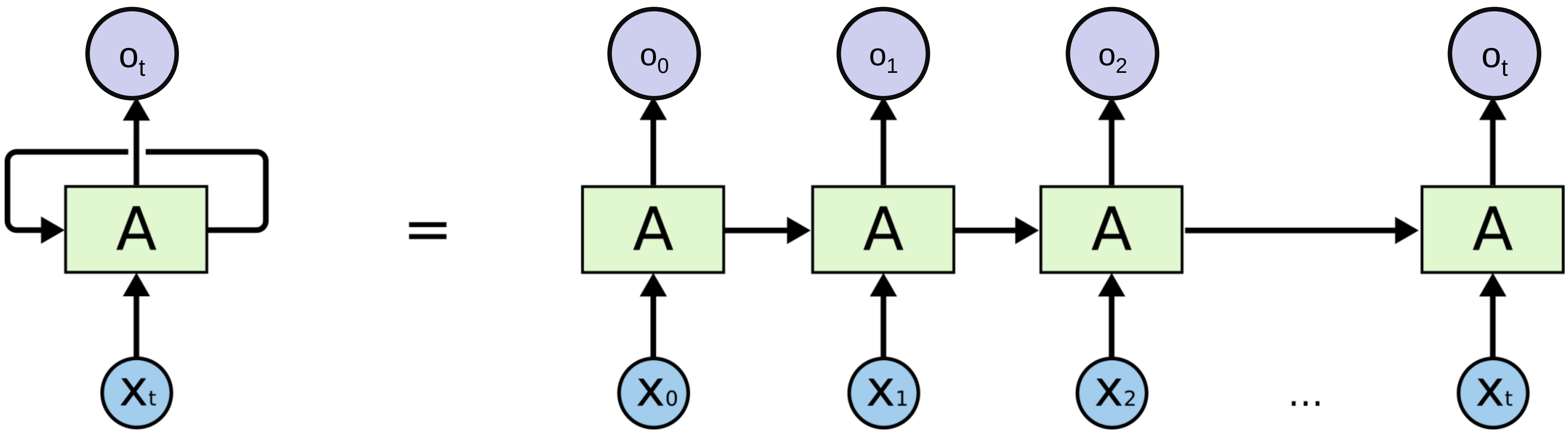
Excursion: Language Models

TECHNISCHE UNIVERSITÄT DARMSTADT

- Performance increases with longer n-grams
- There are a lot of n-grams
  - 500 000 German words (according to Duden)
  - 2-grams: 250 billion combinations
  - 3-grams: > 10^{17} combinations
  - 4-grams: > 10^{22} combinations

- Gigantic training corpus & RAM requirement
  - *“Using one machine with 140GB RAM for 2.8 days, we built an unpruned model on 126 billion tokens”* (Heafield et al.)

Recurrent Neural Network

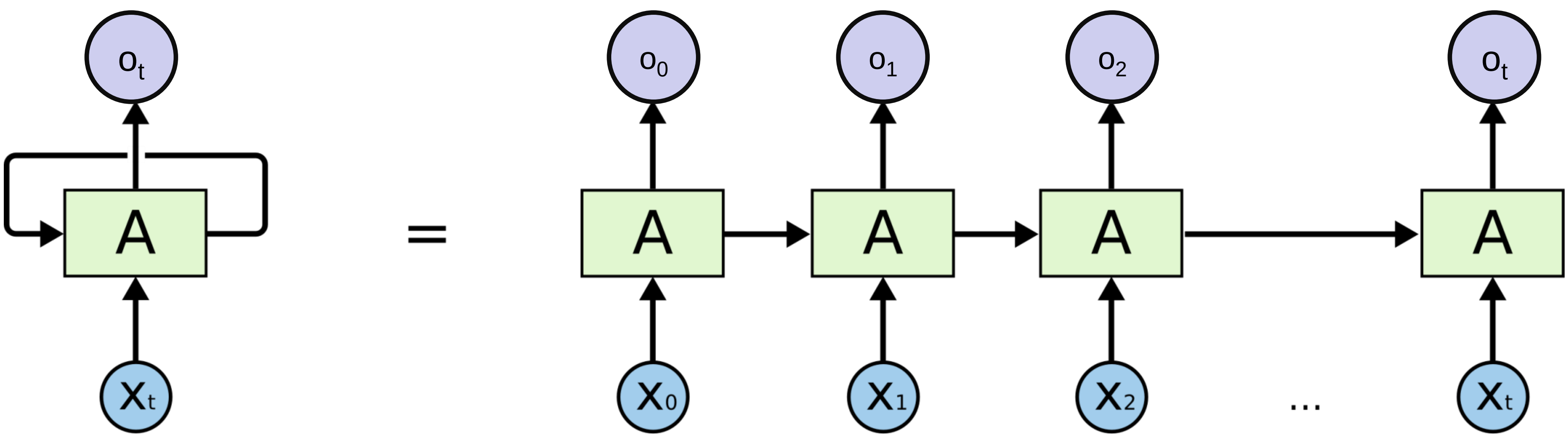


- Recurrent Neural Network have an internal state
- State is passed from input x_t to x_{t+1}

Img Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Language Models with RNN

- Let x_0, x_1, x_2… denote words (or characters)
- Let o_0, o_1, o_2… denote the probability of the sentence
- Memory requirement scales nicely (linear with the number of word embeddings / number of character)



RNN as Generative Language Models

Proof. Omitted. □

Lemma 0.1. *Let $\mathcal{C}$ be a set of the construction.*

Let $\mathcal{C}$ be a gerber covering. Let $\mathcal{F}$ be a quasi-coherent sheaves of $\mathcal{O}$-modules. We have to show that

$$\mathcal{O}_{\mathcal{O}_X} = \mathcal{O}_X(\mathcal{L})$$

.

Proof. This is an algebraic space with the composition of sheaves $\mathcal{F}$ on $X_{\acute{e}tale}$ we have

$$\mathcal{O}_X(\mathcal{F}) = \{morph_1 \times_{\mathcal{O}_X} (\mathcal{G}, \mathcal{F})\}$$

where $\mathcal{G}$ defines an isomorphism $\mathcal{F} \to \mathcal{F}$ of $\mathcal{O}$-modules. □

Lemma 0.2. *This is an integer $\mathcal{Z}$ is injective.*

Proof. See Spaces, Lemma **??**. □

Lemma 0.3. *Let S be a scheme. Let X be a scheme and X is an affine open covering. Let $\mathcal{U} \subset \mathcal{X}$ be a canonical and locally of finite type. Let X be a scheme. Let X be a scheme which is equal to the formal complex.*

The following to the construction of the lemma follows.

Let X be a scheme. Let X be a scheme covering. Let

$$b : X \to Y' \to Y \to Y \to Y' \times_X Y \to X.$$

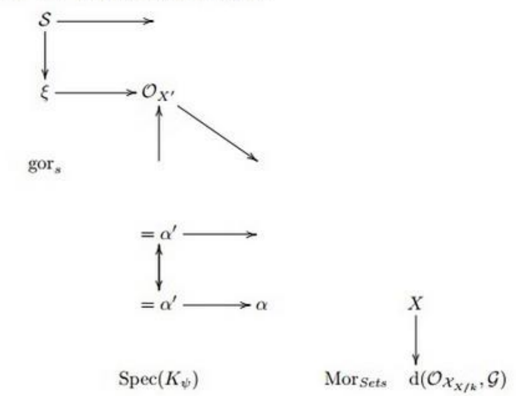
be a morphism of algebraic spaces over S and Y.

Proof. Let X be a nonzero scheme of X. Let X be an algebraic space. Let $\mathcal{F}$ be a quasi-coherent sheaf of $\mathcal{O}_X$-modules. The following are equivalent

(1) $\mathcal{F}$ is an algebraic space over S.
(2) If X is an affine open covering.

Consider a common structure on X and X the functor $\mathcal{O}_X(U)$ which is locally of finite type. □

This since $\mathcal{F} \in \mathcal{F}$ and $x \in \mathcal{G}$ the diagram



is a limit. Then $\mathcal{G}$ is a finite type and assume S is a flat and $\mathcal{F}$ and $\mathcal{G}$ is a finite type f_*. This is of finite type diagrams, and

- the composition of $\mathcal{G}$ is a regular sequence,
- $\mathcal{O}_{X'}$ is a sheaf of rings.

□

Proof. We have see that $X = \mathrm{Spec}(R)$ and $\mathcal{F}$ is a finite type representable by algebraic space. The property $\mathcal{F}$ is a finite morphism of algebraic stacks. Then the cohomology of X is an open neighbourhood of U. □

Proof. This is clear that $\mathcal{G}$ is a finite presentation, see Lemmas **??**.
A *reduced above* we conclude that U is an open covering of $\mathcal{C}$. The functor $\mathcal{F}$ is a "field

$$\mathcal{O}_{X,x} \longrightarrow \mathcal{F}_{\overline{x}} \ \text{-1}(\mathcal{O}_{X_{\acute{e}tale}}) \longrightarrow \mathcal{O}_{X_\ell}^{-1}\mathcal{O}_{X_\lambda}(\mathcal{O}_{X_\eta}^{\overline{v}})$$

is an isomorphism of covering of $\mathcal{O}_{X_i}$. If $\mathcal{F}$ is the unique element of $\mathcal{F}$ such that X is an isomorphism.
The property $\mathcal{F}$ is a disjoint union of Proposition **??** and we can filtered set of presentations of a scheme $\mathcal{O}_X$-algebra with $\mathcal{F}$ are opens of finite type over S.
If $\mathcal{F}$ is a scheme theoretic image points. □

If $\mathcal{F}$ is a finite direct sum $\mathcal{O}_{X_\lambda}$ is a closed immersion, see Lemma **??**. This is a sequence of $\mathcal{F}$ is a similar morphism.

Generated LaTeX-Code from an Character-RNN

Img Source: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

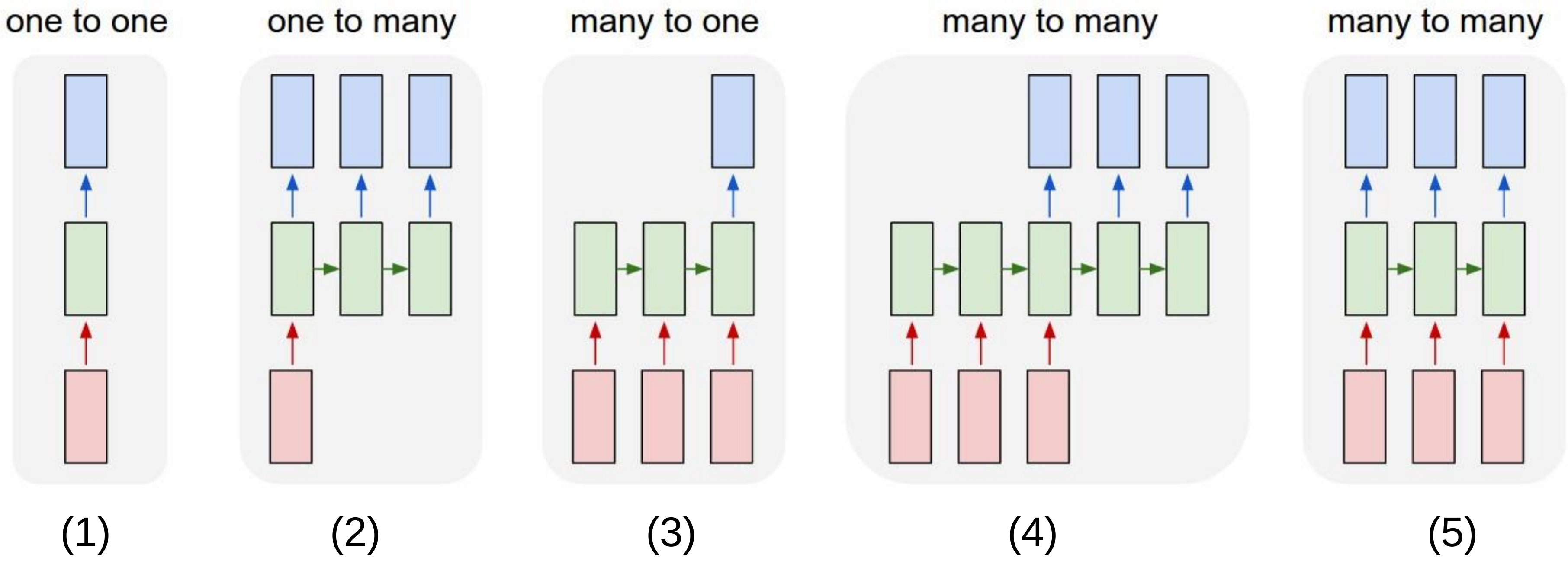
RNN as Generative Language Models

```
/*
 * If this error is set, we will need anything right after that BSD.
 */
static void action_new_function(struct s_stat_info *wb)
{
  unsigned long flags;
  int lel_idx_bit = e->edd, *sys & ~((unsigned long) *FIRST_COMPAT);
  buf[0] = 0xFFFFFFFF & (bit << 4);
  min(inc, slist->bytes);
  printk(KERN_WARNING "Memory allocated %02x/%02x, "
    "original MLL instead\n"),
    min(min(multi_run - s->len, max) * num_data_in),
    frame_pos, sz + first_seg);
  div_u64_w(val, inb_p);
  spin_unlock(&disk->queue_lock);
  mutex_unlock(&s->sock->mutex);
  mutex_unlock(&func->mutex);
  return disassemble(info->pending_bh);
}
```

Generated C-Code from an Character-RNN

Img Source: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Topologies of Recurrent Neural Network



1) Common Neural Network (e.g. feed forward network)
2) Prediction of future states base on single observation
3) Sentiment classification
4) Machine translation
5) Simultaneous interpretation

Img Source: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

(Vanilla) RNN

```
rnn = RNN()
y = rnn.step(x) # x is an input vector, y is the RNN's output vector
```

```
class RNN:
  # ...
  def step(self, x):
    # update the hidden state
    self.h = np.tanh(np.dot(self.W_hh, self.h) + np.dot(self.W_xh, x))
    # compute the output vector
    y = np.dot(self.W_hy, self.h)
    return y
```

- Compute the hidden state: $h_{t+1} = \tanh(W_{hh}h_t + W_{xh}x_t)$
- Compute the output: $y_{t+1} = W_{hy}h_{t+1}$

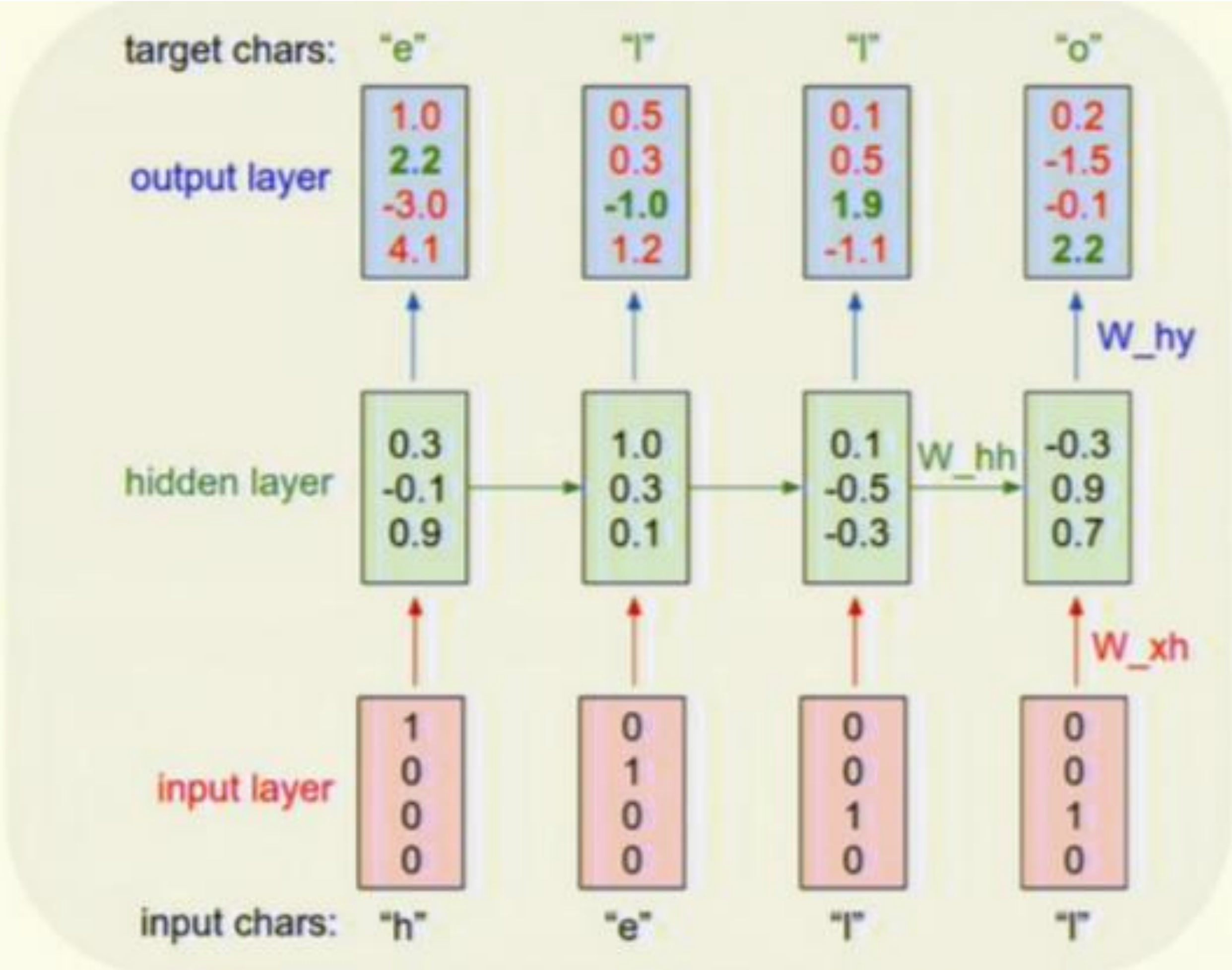
Img Source: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

(Vanilla) RNN

Character-level language model example

Vocabulary:
[h,e,l,o]

Example training sequence:
“hello”



Img Source: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

No Magic Involved (in Theory)

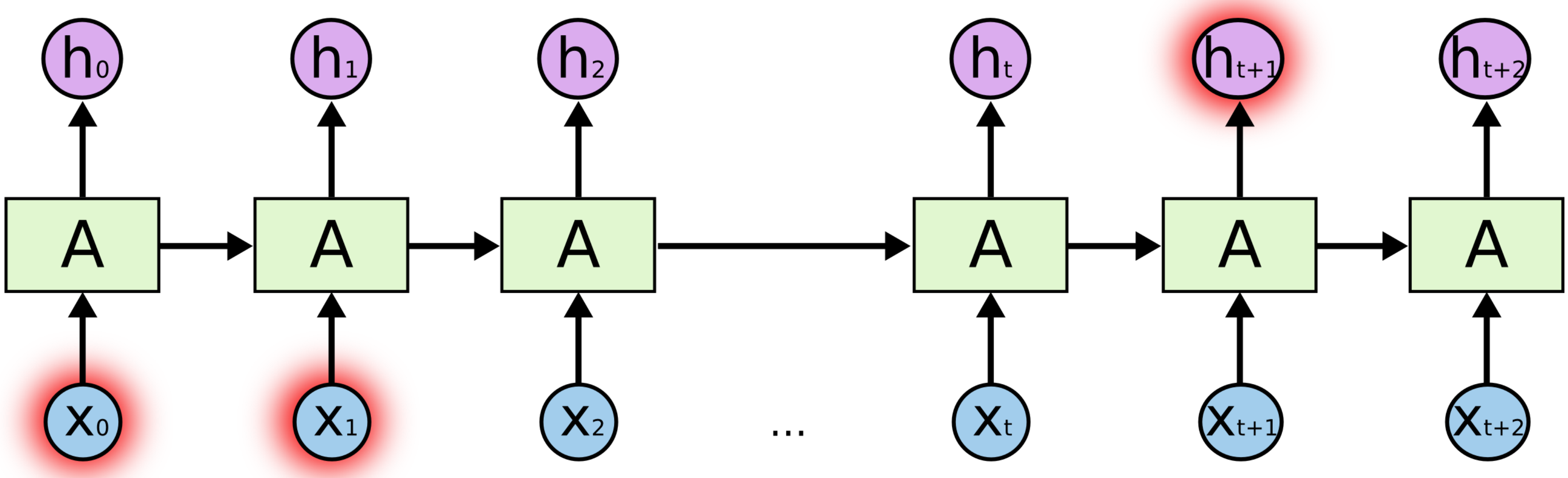
TECHNISCHE UNIVERSITÄT DARMSTADT

- You unroll your data in time
- You compute the gradients
- You use back propagation to train your network
- Karpathy presents a Python implementation for Char-RNN with 112 lines

- Training RNNs is hard:
  - Inputs from many time steps ago can modify output
  - Vanishing / Exploding Gradient Problem

- Vanishing gradients can be solved by Gated-RNNs like Long-Short-Term-Memory (LSTM) Models
  - LSTM became popular 2015 in NLP

Long-Short-Term Memory (LSTM)

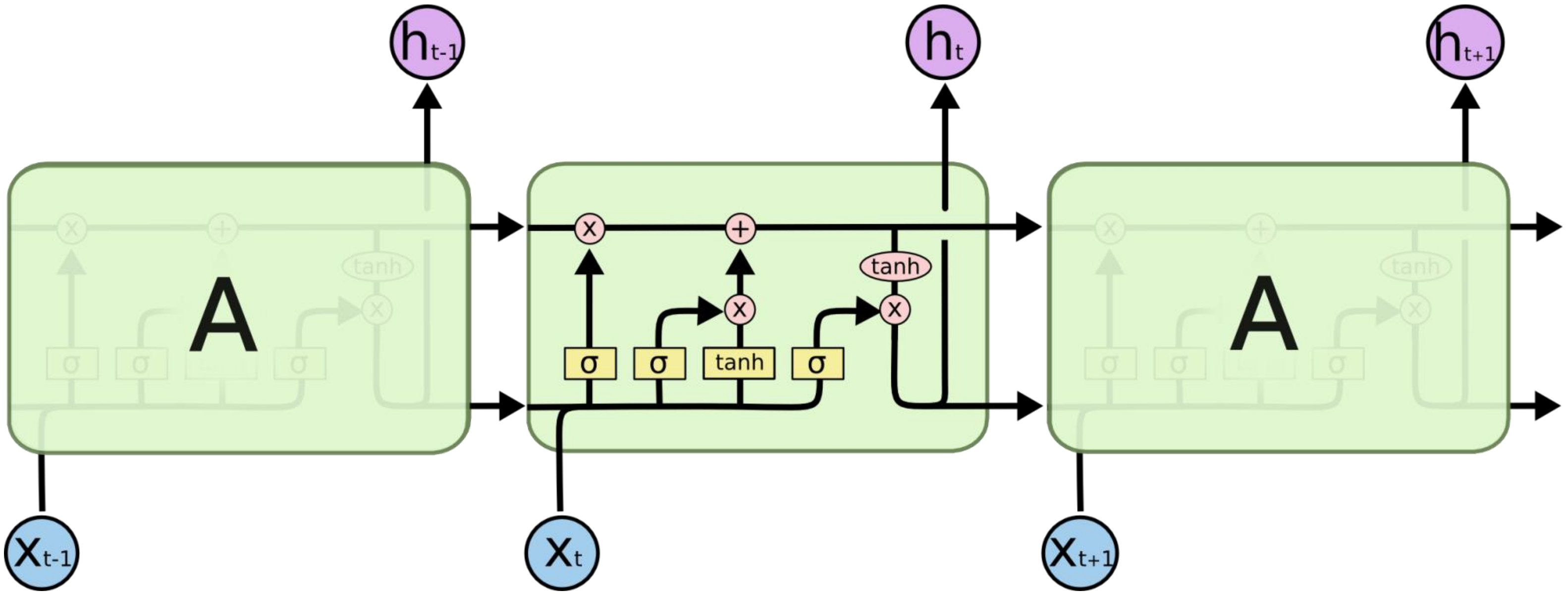


- Long-term dependencies:

  I grew up in France and lived there until I was 18. Therefore I speak fluent ???
- Presented (vanilla) RNN is unable to learn long term dependencies
  - Issue: More recent input data has higher influence on the output
- Long-Short-Term Memory (LSTM) models solves this problem

Img Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

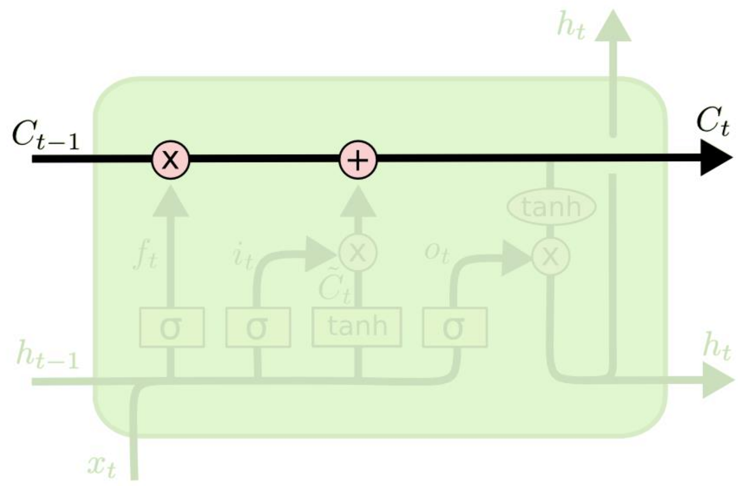
LSTM Model



- The LSTM model implements a *forget-gate* and an *add-gate*
- The models learns when to forget something and when to update internal storage

Img Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

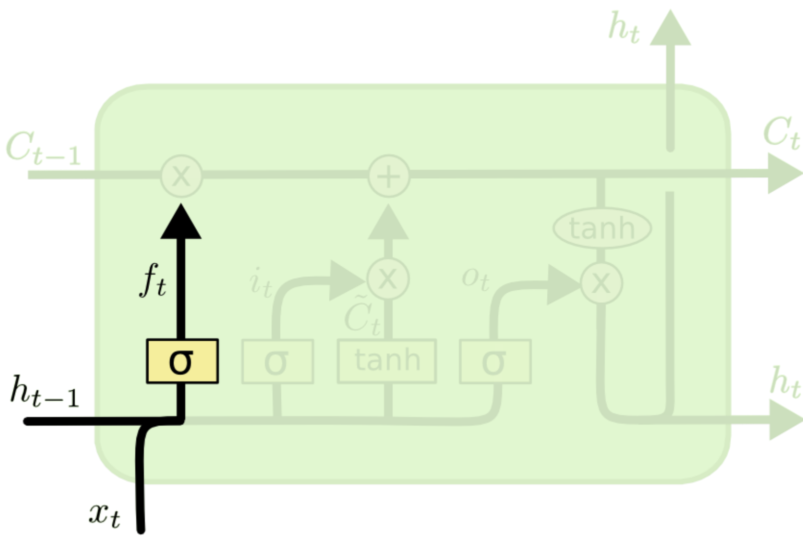
LSTM Model



- Core: Cell-state *C* (a vector of certain size)
- The model has the ability to remove or add information using Gates

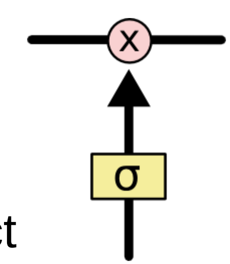
Img Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Forgot-Gate



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

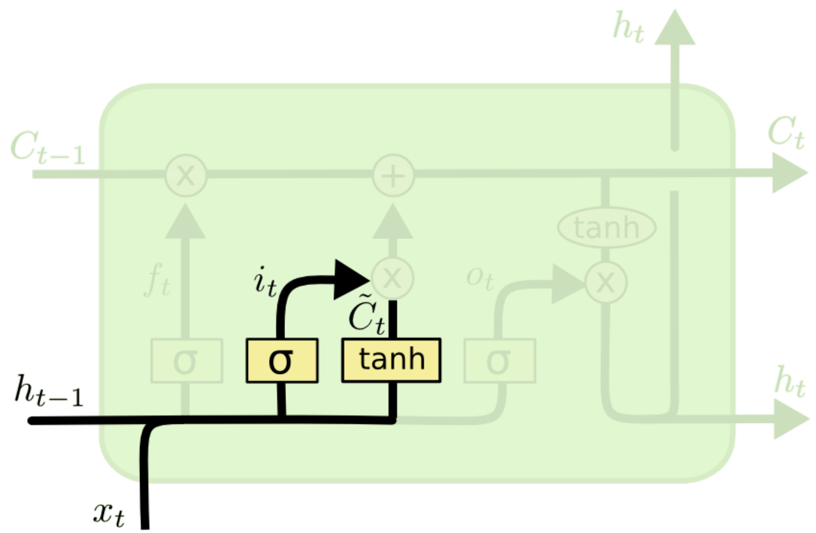
- Sigmoid function σ output a value between 0 and 1
- The output is point-wise multiplied with the cell state C_{t-1}
- Interpretation:
  - 0: *Let nothing through*
  - 1: *Let everything through*



- Example: When we see a new subject, forget gender of old subject

Img Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Set-Gate



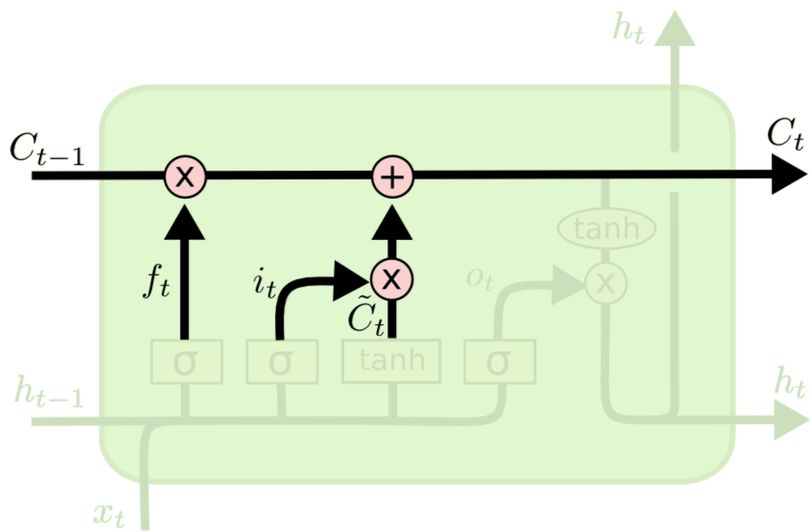
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

- Compute i_t which cells we want to update and to which degree (σ: 0 … 1)
- Compute the new cell value using the *tanh* function

Img Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

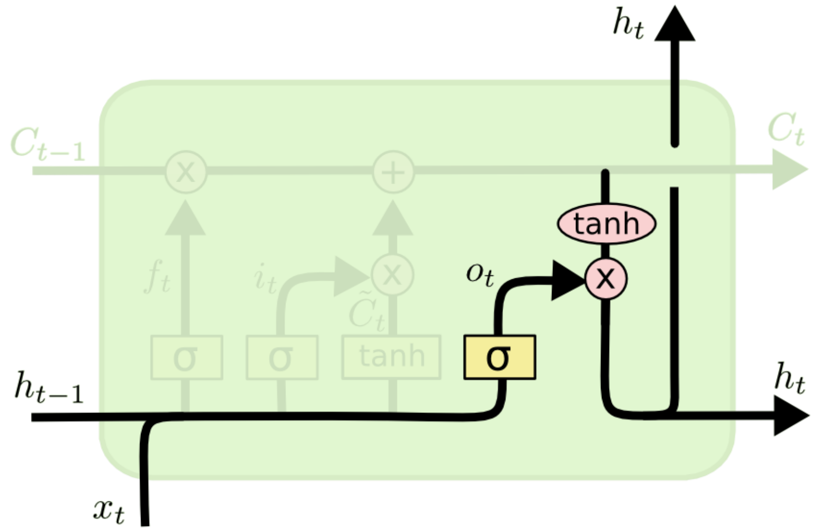
Update Internal Cell State



$$C_t = \underbrace{f_t * C_{t-1}}_{\text{Forget state cells}} + \underbrace{i_t * \tilde{C}_t}_{\text{Update state cells}}$$

Img Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Compute Output h_t



$$o_t = \sigma \left(W_o \ [h_{t-1}, x_t] \ + \ b_o \right)$$

$$h_t = o_t * \tanh \left(C_t \right)$$

- We use the updated cell state C_t to compute the output
- We might not need the complete cell state as output
  - Compute o_t, defining how relevant each cell is for the output
  - Pointwise multiply o_t with *tanh*(C_t)

- Cell state C_t and output h_t is passed to the next time step

Img Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Training

- **RMSProp** / Adam / Adagrad (SGD can work too, but has much higher sensitivity to learning rate)
- **Clip gradients** (at 5.0 is a common value to use)
- **Initialize forget gates** with high bias (to encourage remembering at start) can help
- **L2 regularization** not very common, can even hurt sometimes
- **Dropout** always good along depth, but NOT along time (in the recurrent part). [*Zaremba et al.*]
- Typical training on good GPU: ~10mil params, 1-2 days

Img Source: https://skillsmatter.com/skillscasts/6611-visualizing-and-understanding-recurrent-networks

Searching for interpretable cells

Cell that turns on inside quotes:

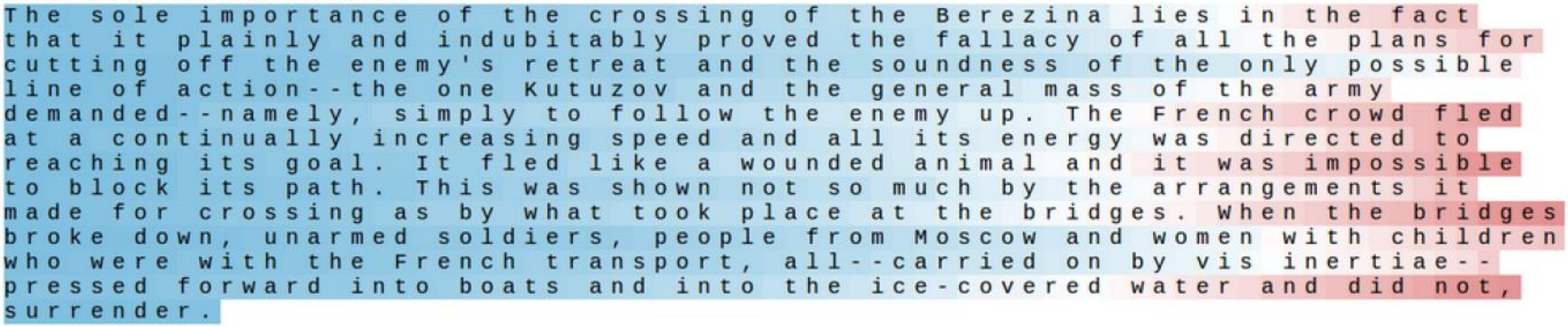
"You mean to imply that I have nothing to eat out of.... On the contrary, I can supply you with everything even if you want to give dinner parties," warmly replied Chichagov, who tried by every word he spoke to prove his own rectitude and therefore imagined Kutuzov to be animated by the same desire.

Kutuzov, shrugging his shoulders, replied with his subtle penetrating smile: "I meant merely to say what I said."

Source: Karpathy et al., 2015, *Visualizing and Understanding Recurrent Networks*

Searching for interpretable cells

Cell sensitive to position in line:



Source: Karpathy et al., 2015, *Visualizing and Understanding Recurrent Networks*

Searching for interpretable cells

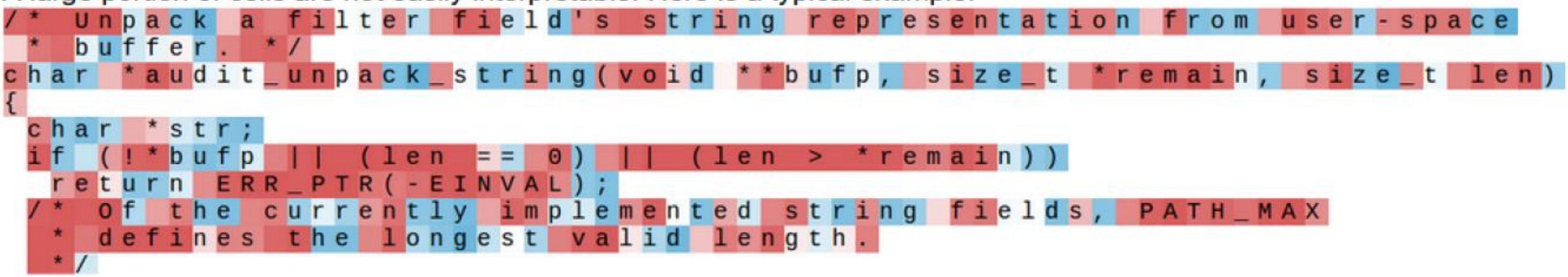
Cell that is sensitive to the depth of an expression:

```
#ifdef CONFIG_AUDITSYSCALL
static inline int audit_match_class_bits(int class, u32 *mask)
{
 int i;
 if (classes[class]) {
  for (i = 0; i < AUDIT_BITMASK_SIZE; i++)
   if (mask[i] & classes[class][i])
    return 0;
 }
 return 1;
}
```

Source: Karpathy et al., 2015, *Visualizing and Understanding Recurrent Networks*

Searching for interpretable cells

A large portion of cells are not easily interpretable. Here is a typical example:



Source: Karpathy et al., 2015, *Visualizing and Understanding Recurrent Networks*

Variants of LSTM

- GRU
  - Cho et al., 2014, http://arxiv.org/pdf/1406.1078v3.pdf
- Depth Gated RNN
  - Yao et al., 2015, http://arxiv.org/pdf/1508.03790v2.pdf
- Clockwork RNN
  - Koutnik et al., 2014, http://arxiv.org/pdf/1402.3511v1.pdf

- Does the difference matter? Not really
  - Greff et al., 2015, http://arxiv.org/pdf/1503.04069.pdf
  - Jozefowicz et al., 2015, http://jmlr.org/proceedings/papers/v37/jozefowicz15.pdf

Img Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Tree-LSTM

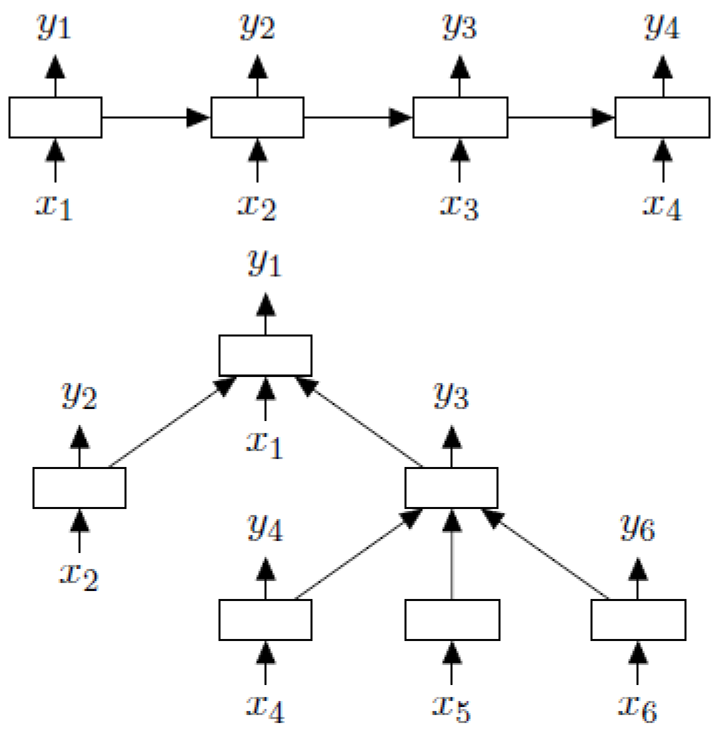
TECHNISCHE UNIVERSITÄT DARMSTADT

Kai Sheng Tai, Richard Socher, and Christopher Manning, 2015, "Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks"

- Code: https://github.com/stanfordnlp/treelstm

Tree-LSTM

- Idea: Use a syntax tree to process the sentence instead of a sequential left-to-right approach



Source: Sheng et al., 2015, *Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks*

Tree-LSTM - Performance

Method	Fine-grained	Binary
RAE (Socher et al., 2013)	43.2	82.4
MV-RNN (Socher et al., 2013)	44.4	82.9
RNTN (Socher et al., 2013)	45.7	85.4
DCNN (Blunsom et al., 2014)	48.5	86.8
Paragraph-Vec (Le and Mikolov, 2014)	48.7	87.8
CNN-non-static (Kim, 2014)	48.0	87.2
CNN-multichannel (Kim, 2014)	47.4	**88.1**
DRNN (Irsoy and Cardie, 2014)	49.8	86.6
LSTM	46.4 (1.1)	84.9 (0.6)
Bidirectional LSTM	49.1 (1.0)	87.5 (0.5)
2-layer LSTM	46.0 (1.3)	86.3 (0.6)
2-layer Bidirectional LSTM	48.5 (1.0)	87.2 (1.0)
Dependency Tree-LSTM	48.4 (0.4)	85.7 (0.4)
Constituency Tree-LSTM		
– randomly initialized vectors	43.9 (0.6)	82.0 (0.5)
– Glove vectors, fixed	49.7 (0.4)	87.5 (0.8)
– Glove vectors, tuned	**51.0** (0.5)	88.0 (0.3)

Performance on the Stanford Sentiment Treebank

Source: Sheng et al., 2015, *Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks*

Tree-LSTM - Performance

Method	Pearson's r		Spearman's ρ		MSE	
Illinois-LH (Lai and Hockenmaier, 2014)	0.7993		0.7538		0.3692	
UNAL-NLP (Jimenez et al., 2014)	0.8070		0.7489		0.3550	
Meaning Factory (Bjerva et al., 2014)	0.8268		0.7721		0.3224	
ECNU (Zhao et al., 2014)	0.8414		–		–	
Mean vectors	0.7577	(0.0013)	0.6738	(0.0027)	0.4557	(0.0090)
DT-RNN (Socher et al., 2014)	0.7923	(0.0070)	0.7319	(0.0071)	0.3822	(0.0137)
SDT-RNN (Socher et al., 2014)	0.7900	(0.0042)	0.7304	(0.0076)	0.3848	(0.0074)
LSTM	0.8528	(0.0031)	0.7911	(0.0059)	0.2831	(0.0092)
Bidirectional LSTM	0.8567	(0.0028)	0.7966	(0.0053)	0.2736	(0.0063)
2-layer LSTM	0.8515	(0.0066)	0.7896	(0.0088)	0.2838	(0.0150)
2-layer Bidirectional LSTM	0.8558	(0.0014)	0.7965	(0.0018)	0.2762	(0.0020)
Constituency Tree-LSTM	0.8582	(0.0038)	0.7966	(0.0053)	0.2734	(0.0108)
Dependency Tree-LSTM	**0.8676**	(0.0030)	**0.8083**	(0.0042)	**0.2532**	(0.0052)

Table 3: Test set results on the SICK semantic relatedness subtask. For our experiments, we report mean scores over 5 runs (standard deviations in parentheses). Results are grouped as follows: (**1**) SemEval 2014 submissions; (**2**) Our own baselines; (**3**) Sequential LSTMs; (**4**) Tree-structured LSTMs.

Source: Sheng et al., 2015, *Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks*

The Future of Deep Learning

NLP as a Q&A-Problem?

- Most NLP tasks can be modeled as a Question-Answering problem
  - What are the part-of-speech tags in the sentence?
  - What are the named entities in the sentence?
  - Which pronouns refer to the same entities?
  - What is the translated version of the sentence?
  - What is the major claim in the sentence?

- Kumar et al. presented a general architecture for Q&A
  - Ask Me Anything: Dynamic Memory Networks for Natural Language Processing
  - http://arxiv.org/abs/1506.07285

Example

I:	Jane went to the hallway.
I:	Mary walked to the bathroom.
I:	Sandra went to the garden.
I:	Daniel went back to the garden.
I:	Sandra took the milk there.
Q:	Where is the milk?
A:	garden
I:	Everybody is happy.
Q:	What's the sentiment?
A:	positive
Q:	What are the POS tags?
A:	NN VBZ JJ .

I:	Peter's sister is called Isabelle.
Q:	What are the mentions?
A:	[[Peter] 's sister] is called [Isabelle] .
I:	Peter's sister is called [Isabelle].
Q:	Is it coreferent with: [Peter's sister] is called Isabelle?
A:	yes
Q:	Is it coreferent with: [Peter] 's sister is called Isabelle?
A:	no
I:	The answer is far from obvious.
Q:	In French?
A:	La réponse est loin d'être évidente.

Source: Kumer et al., 2015, Ask Me Anything: Dynamic Memory Networks for Natural Language Processing

Overview of the Dynamic Memory Model

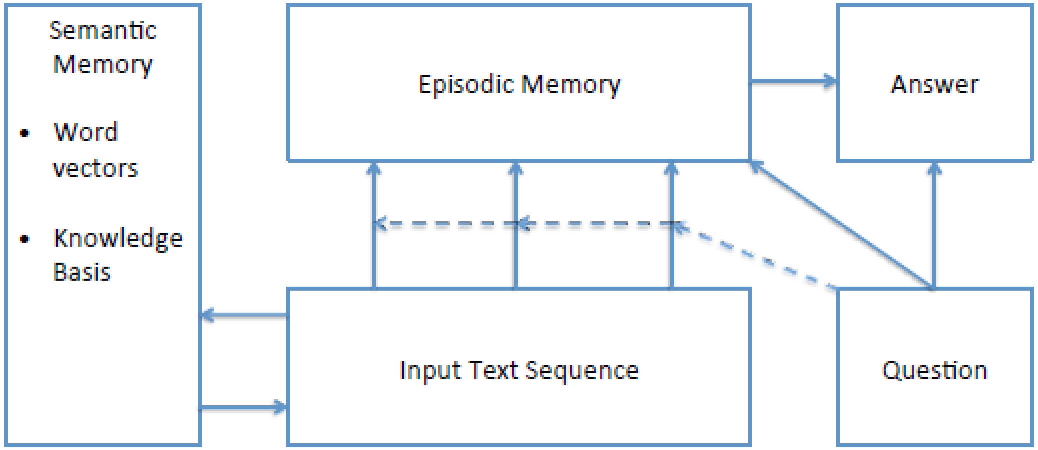


Figure 2: Overview of DMN modules. Communication between them is indicated by arrows and uses only vector representations. Questions trigger gates which allow vectors for certain input words or sentences to be given to the episodic memory module. The final state of the episodic memory is the input to the answer module.

Source: Kumer et al., 2015, Ask Me Anything: Dynamic Memory Networks for Natural Language Processing

DMN – Q&A Results

The Facebook bAbI dataset is a synthetic dataset meant to test a model's ability to retrieve facts and reason over them. Each task tests a different skill that a good question answering model ought to have, such as coreference resolution, deduction, and induction. Training on the bAbI dataset

Task	MemNN	DMN	Task	MemNN	DMN
1: Single Supporting Fact	100	100	11: Basic Coreference	100	99.9
2: Two Supporting Facts	100	98.2	12: Conjunction	100	100
3: Three Supporting facts	100	95.2	13: Compound Coreference	100	99.8
4: Two Argument Relations	100	100	14: Time Reasoning	99	100
5: Three Argument Relations	98	99.3	15: Basic Deduction	100	100
6: Yes/No Questions	100	100	16: Basic Induction	100	99.4
7: Counting	85	96.9	17: Positional Reasoning	65	59.6
8: Lists/Sets	91	96.5	18: Size Reasoning	95	95.3
9: Simple Negation	100	100	19: Path Finding	36	34.5
10: Indefinite Knowledge	98	97.5	20: Agent's Motivations	100	100
			Mean Accuracy (%)	93.3	**93.6**

Table 1: Test accuracies on the bAbI dataset. MemNN numbers taken from Weston et al. [18]. The DMN passes (accuracy $> 95\%$) 18 tasks, whereas the MemNN passes 16.

Source: Kumer et al., 2015, Ask Me Anything: Dynamic Memory Networks for Natural Language Processing

DMN – Results on NLP Tasks

POS (Wall Street Journal)

Model	SVMTool	Sogaard	Suzuki et al.	Spoustova et al.	SCNN	DMN
Acc (%)	97.15	97.27	97.40	97.44	97.50	**97.56**

Sentiment (Stanford Sentiment Treebank)

Task	MV-RNN	RNTN	DCNN	PVec	CNN-MC	DRNN	CT-LSTM	DMN
Binary	82.9	85.4	86.8	87.8	88.1	86.6	88.0	**88.6**
Fine-grained	44.4	45.7	48.5	48.7	47.4	49.8	51.0	**51.2**

Coreference Resolution (Dataset Gua et al., 2015)

Metric	Guha et al., 2015			Durrett and Klein, 2013			DMN		
	P	R	F1	P	R	F1	P	R	F1
MUC	56.8	57.8	57.8	70.2	40.2	49.6	74.6	66.0	**70.0**
B^3	68.1	74.8	70.4	88.5	64.7	74.2	82.0	77.5	**79.7**
$CEAF_e$	73.3	76.1	74.2	56.5	80.0	65.7	70.7	79.3	**74.7**

Source: Kumer et al., 2015, Ask Me Anything: Dynamic Memory Networks for Natural Language Processing

Hot Future Trends in Machine Learning

- Computer use 3 fundamental mechanisms:
  - elementary operations, logical flow control, external memory
- Most machine learning algorithms only use elementary operations
  - flow control and external memory largely neglected in machine learning
- RNN can learn how to use external memory & logical flow
  - Siegelmann & Sontag, 1995, showed that RNN are Turing-Complete

- Hot research directions
  - How to use external memory for machine learning (Graves et al., 2014, Neural Turing Machines)
  - Development of multi-task models
  - Multimodal inputs and questions