CS291K – Advanced Data Mining

Instructor: Xifeng Yan
Computer Science
University of California at Santa Barbara

Long-Short Term Memory(LSTM) Models and Memory Networks

Lecturer: Izzeddin Gur
Computer Science
University of California at Santa Barbara

Source of slides

- Richard Socher's course in Stanford
- Colah's Blog
- Jason Weston's Memory Networks
- Images from Despicable Me

Recap of Previous Models

Word2Vec $$J_t(\theta) = \log \sigma \left(u_o^T v_c\right) + \sum_{j \sim P(w)} \left[\log \sigma \left(-u_j^T v_c\right)\right]$$

Glove $$J(\theta) = \frac{1}{2} \sum_{i,j=1}^{W} f(P_{ij})(u_i^T v_j - \log P_{ij})^2$$

Nnet & Max-margin $$s = U^T f(Wx + b)$$

$$J = \max(0, 1 - s + s_c)$$

Recap of Previous Models

Multilayer Nnet

&

Backprop

$$
\begin{aligned}
x &= z^{(1)} = a^{(1)} \\
z^{(2)} &= W^{(1)} x + b^{(1)} \\
a^{(2)} &= f\left(z^{(2)}\right) \\
z^{(3)} &= W^{(2)} a^{(2)} + b^{(2)} \\
a^{(3)} &= f\left(z^{(3)}\right) \\
s &= U^T a^{(3)}
\end{aligned}
$$

$$\delta^{(l)} = \left((W^{(l)})^T \delta^{(l+1)}\right) \circ f'(z^{(l)}),$$

$$\frac{\partial}{\partial W^{(l)}} E_R = \delta^{(l+1)} (a^{(l)})^T + \lambda W^{(l)}$$

Recap of Previous Models

Recurrent Neural Networks

$$h_t = \sigma\left(W^{(hh)}h_{t-1} + W^{(hx)}x_{[t]}\right)$$

$$\hat{y}_t = \text{softmax}\left(W^{(S)}h_t\right)$$

Cross Entropy Error

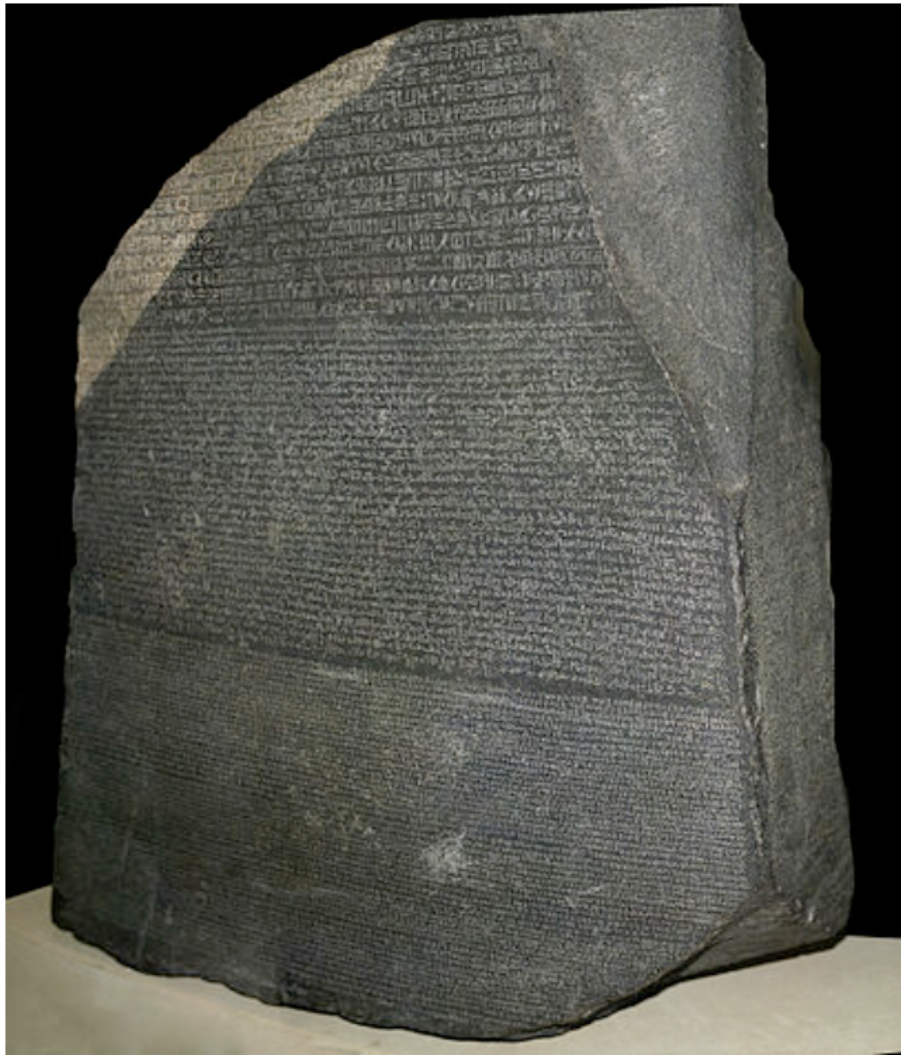
$$J^{(t)}(\theta) = -\sum_{j=1}^{|V|} y_{t,j} \log \hat{y}_{t,j}$$

Mini-batched SGD

$$\theta^{new} = \theta^{old} - \alpha \nabla_\theta J_{t:t+B}(\theta)$$

Machine Translation

- Methods are statistical
- Use Parallel Corpora
  - French to English
  - English to English
- Traditional systems are very complex and human engineered
- Rosetta Stone ———>

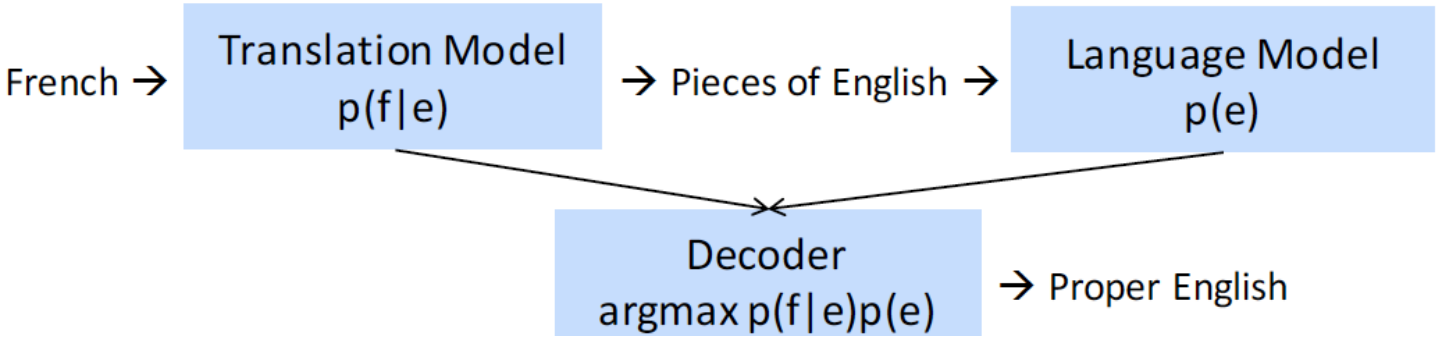


Current MT Systems

- Source language, ex. f = *French*
- Target language, ex. e = *English*
- Probabilistic modeling (Bayes Rule):

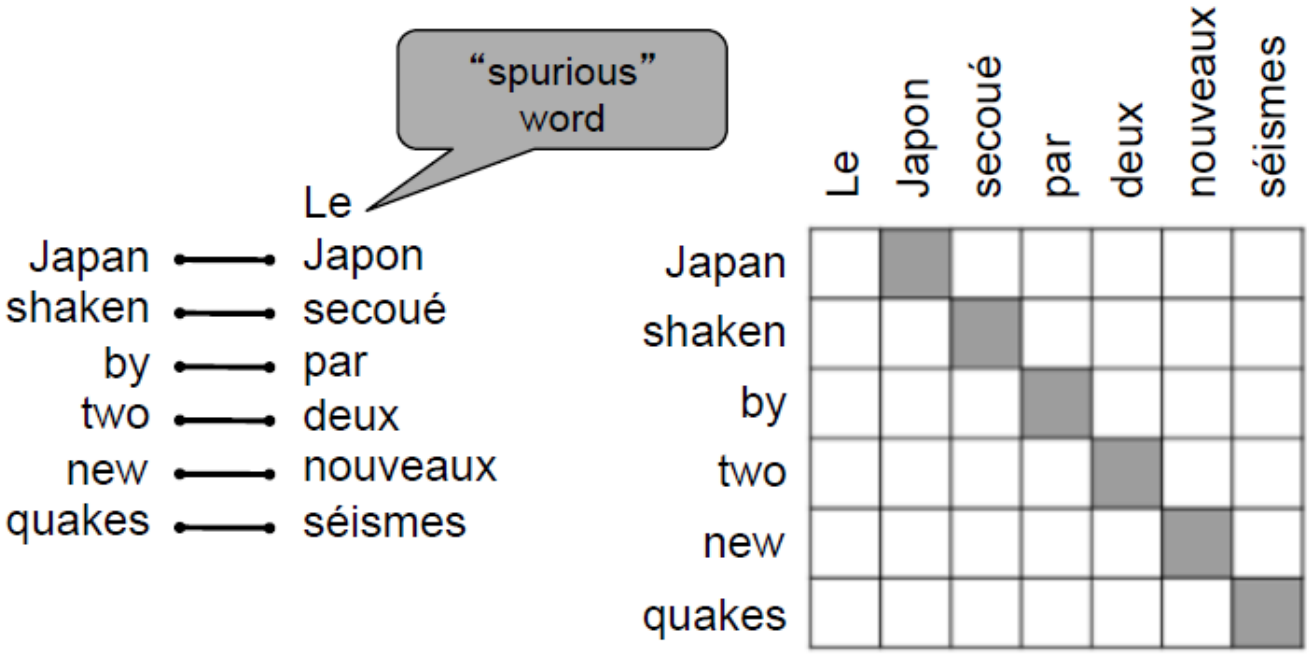
$$\hat{e} = \text{argmax}_e p(e|f) = \text{argmax}_e p(f|e)p(e)$$

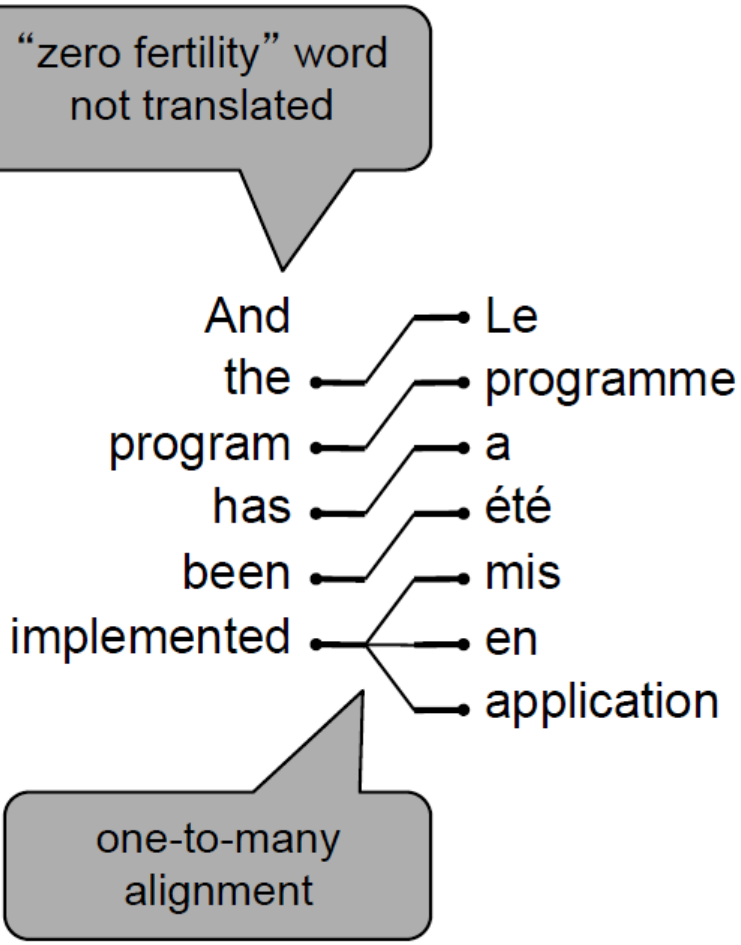
- Translation model *p(f |e)* trained on parallel corpus
- Language modeling *p(e |f)* trained on monolingual corpus

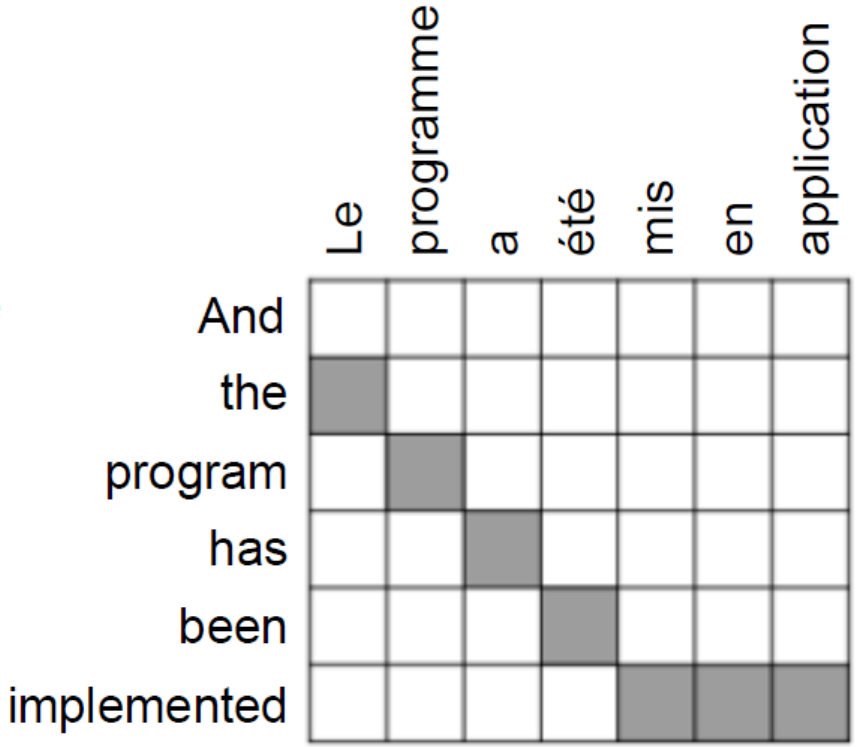


First Steps : Alignment

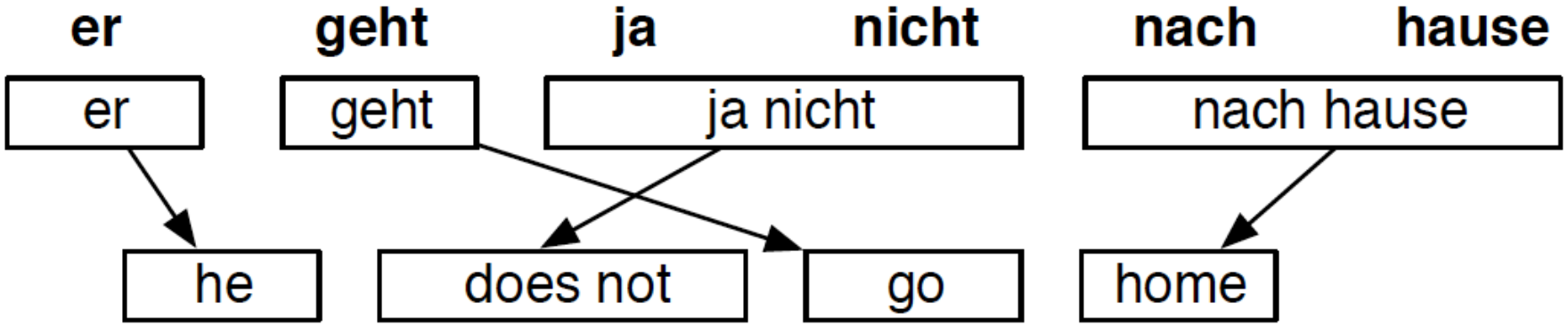
- Which word sequences in the source language would translate well to which word sequences in the target language?




"zero fertility" word not translated
And
the
program
has
been
implemented
Le
programme
a
été
mis
en
application
one-to-many alignment

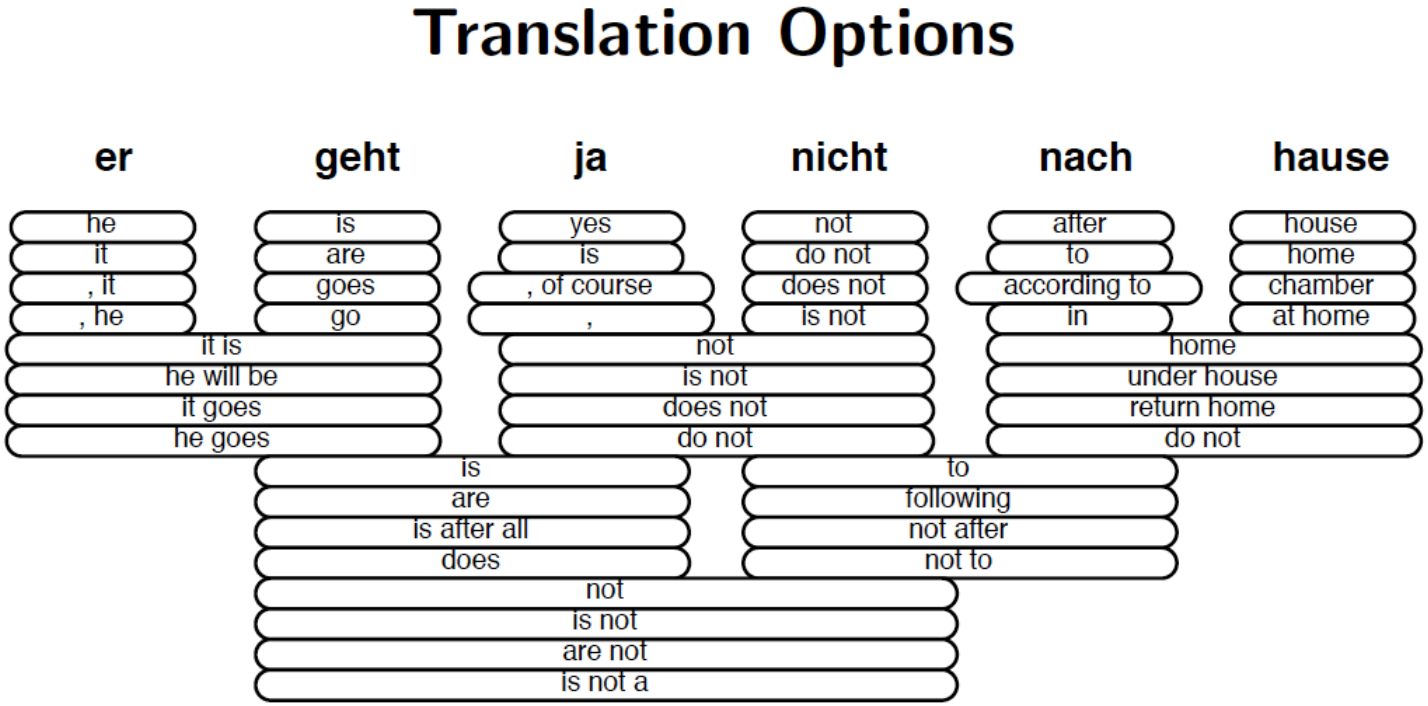

Le
programme
a
été
mis
en
application
And
the
program
has
been
implemented

Second Step : Word Ordering



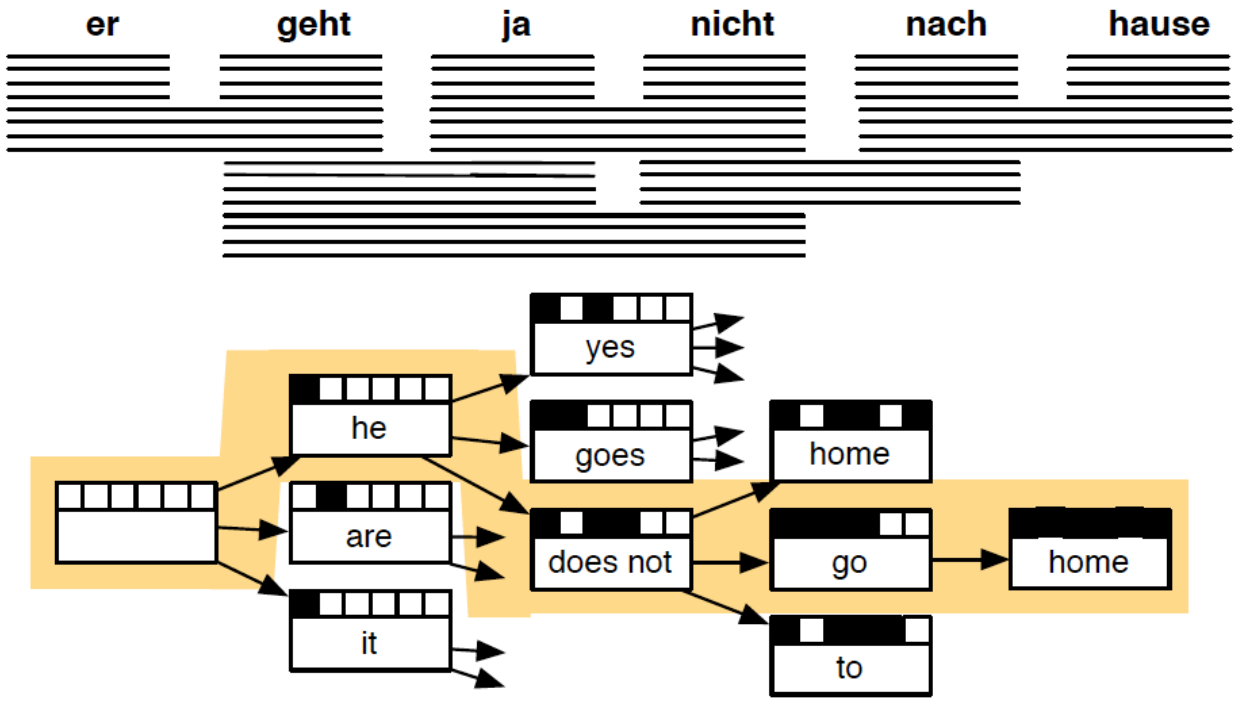
Third Step : Candidate Generation

- Each candidate mapping of word sequences to word sequences result in a large candidate search space :



Fourth Step : Ranking Candidates

- Search for best hypothesis, best?
  - Word sequence translation
  - Reordering and refinement based on language modeling



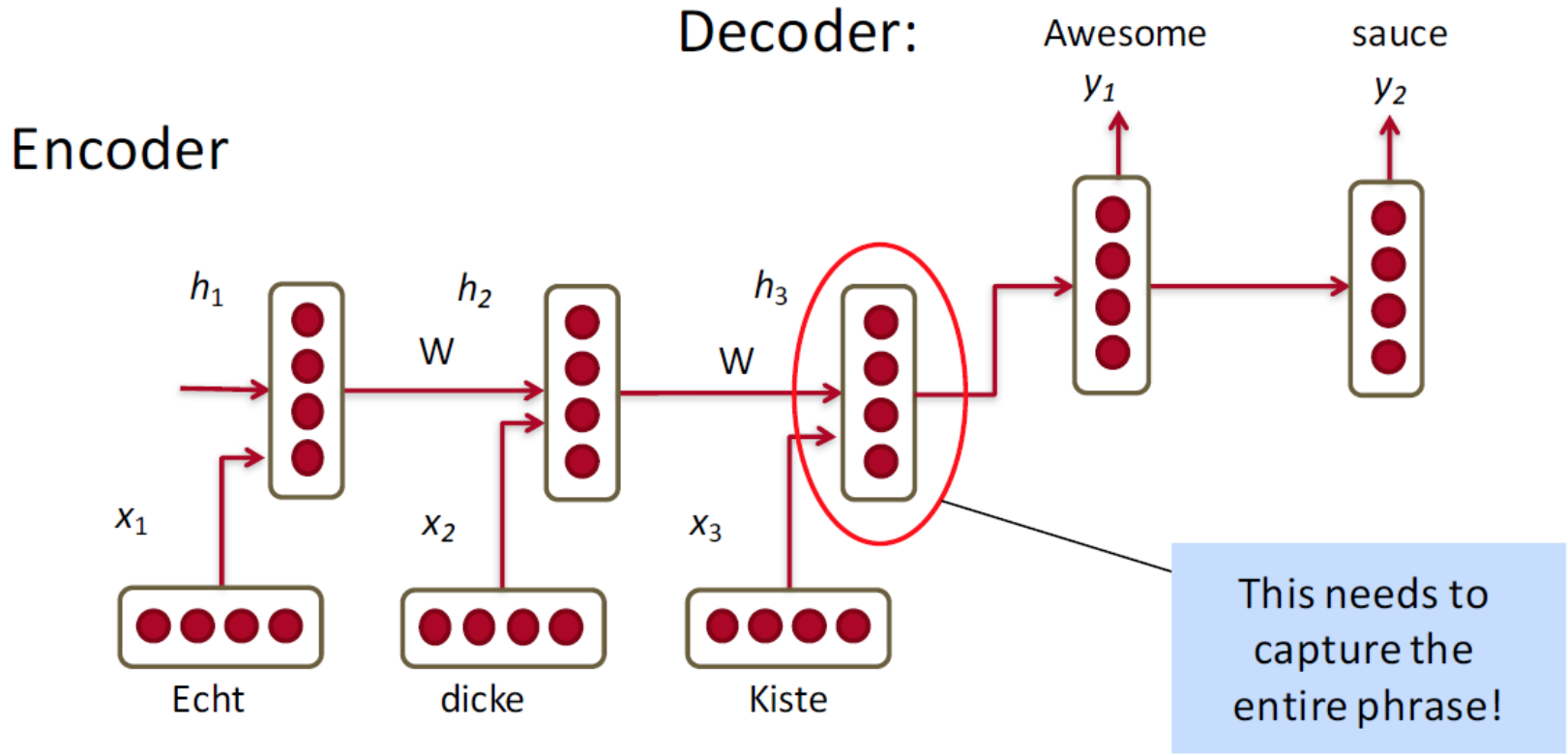
Problems

- Skips a lot of important details, ex. long term sequential dependency
- Lots of human engineering, ex. extracting phrase-level features for phrase translation
- Very complex systems, ex. at least 4 different levels of processes
- Different independent machine learning problems solved, ex.

  parallel corpus generation, feature extraction, scoring different levels of granularity, ranking candidates, handling big data, etc.

Direct Translation by RNN

- Use an RNN to solve all the problems with a single model!



Simplest RNN Translation Model

Encoder - Decoder Architecture

- Encoder : $$h_t = \phi(h_{t-1}, x_t) = f\left(W^{(hh)} h_{t-1} + W^{(hx)} x_t\right)$$
- Decoder : $$h_t = \phi(h_{t-1}) = f\left(W^{(hh)} h_{t-1}\right)$$
$$y_t = softmax\left(W^{(S)} h_t\right)$$

- Given a sequence of source words, minimize cross entropy for each target word :

$$\max_{\theta} \frac{1}{N} \sum_{n=1}^{N} \log p_\theta(y^{(n)} | x^{(n)})$$

Improvements

- Use more complex units in the computation for different purposes

- Main Ideas :
  - Use memories to capture long term dependencies
  - Allow error messages to propagate at different granularity depending on the input

GRU

GRU : Basics

- Standard RNN computes next hidden layer directly :

$$h_t = f\left(W^{(hh)}h_{t-1} + W^{(hx)}x_t\right)$$

- Use intermediary gates instead :
  - *Update Gate :* how novel current input is

$$z_t = \sigma\left(W^{(z)}x_t + U^{(z)}h_{t-1}\right)$$

  - *Reset Gate* : how important previous sequence is

$$r_t = \sigma\left(W^{(r)}x_t + U^{(r)}h_{t-1}\right)$$

GRU

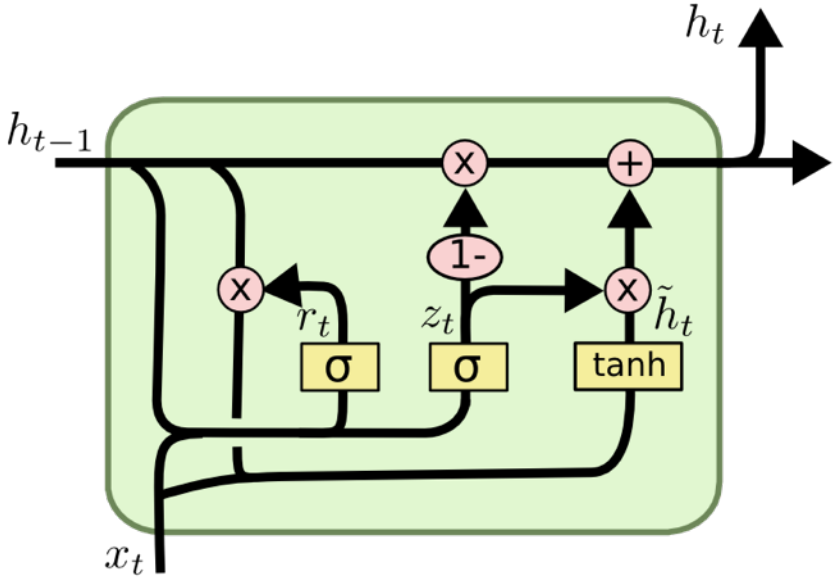
- Update Gate $$z_t = \sigma\left(W^{(z)}x_t + U^{(z)}h_{t-1}\right)$$
- Reset Gate $$r_t = \sigma\left(W^{(r)}x_t + U^{(r)}h_{t-1}\right)$$
- New Memory Content : $$\tilde{h}_t = \tanh\left(Wx_t + r_t \circ Uh_{t-1}\right)$$

- If reset gate is close to 0, previous information is forgotten
- If reset gate is close to 1, current input is not novel
- Final memory (hidden layer) is computed as :

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

- If update gate is close to 0, previous memory is forgotten
- If update gate is close to 1, carry previous information

GRU Illustration



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

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

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

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

GRU Intuition

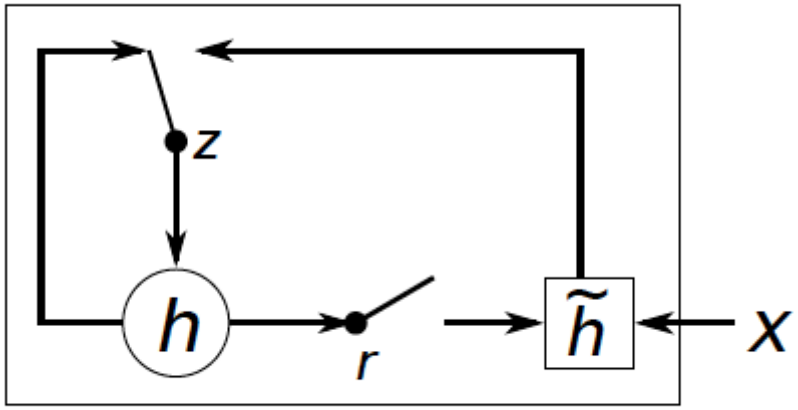
$$z_t = \sigma\left(W^{(z)}x_t + U^{(z)}h_{t-1}\right)$$
$$r_t = \sigma\left(W^{(r)}x_t + U^{(r)}h_{t-1}\right)$$
$$\tilde{h}_t = \tanh\left(Wx_t + r_t \circ Uh_{t-1}\right)$$
$$h_t = z_t \circ h_{t-1} + (1 - z_t) \circ \tilde{h}_t$$

- If reset gate is close to 0, ignore previous hidden state
  —> Model ignores previous information, uses current input
- Update gate z controls, how much previous information matters :
  - If z = 1 for many time steps, we can copy the information from many many time steps away :
    Vanishing Gradient
- Units with short term memory often have reset gates very active

GRU Intuition

- Units with long-term memories generally have very active update gates

$$z_t = \sigma\left(W^{(z)}x_t + U^{(z)}h_{t-1}\right)$$
$$r_t = \sigma\left(W^{(r)}x_t + U^{(r)}h_{t-1}\right)$$
$$\tilde{h}_t = \tanh\left(Wx_t + r_t \circ Uh_{t-1}\right)$$
$$h_t = z_t \circ h_{t-1} + (1 - z_t) \circ \tilde{h}_t$$



Derivatives

- Some parts can be
- reduced to RNN :

$$h_t = f\left(W^{(hh)} h_{t-1} + W^{(hx)} x_t\right)$$

$$z_t = \sigma\left(W^{(z)} x_t + U^{(z)} h_{t-1}\right)$$
$$r_t = \sigma\left(W^{(r)} x_t + U^{(r)} h_{t-1}\right)$$
$$\tilde{h}_t = \tanh\left(W x_t + r_t \circ U h_{t-1}\right)$$
$$h_t = z_t \circ h_{t-1} + (1 - z_t) \circ \tilde{h}_t$$

- Both reset and update games have the same structure
- Derivative of *tanh :*

$$\frac{d}{dx} \tanh x = 1 - \tanh^2 x$$

- Derivative of element wise product *x * y :*

*d(x * y) = d(x) * y + x * d(y)*

- Rest is chain rule

Long-Short Term Memories

- Additional gated units
- Units :
  - Input Gate :
  - Forget Gate :
  - Output Gate :
  - New Memory Cell :
- Final Memory Cell :
- Final Hidden State :

$$i_t = \sigma\left(W^{(i)}x_t + U^{(i)}h_{t-1}\right)$$

$$f_t = \sigma\left(W^{(f)}x_t + U^{(f)}h_{t-1}\right)$$

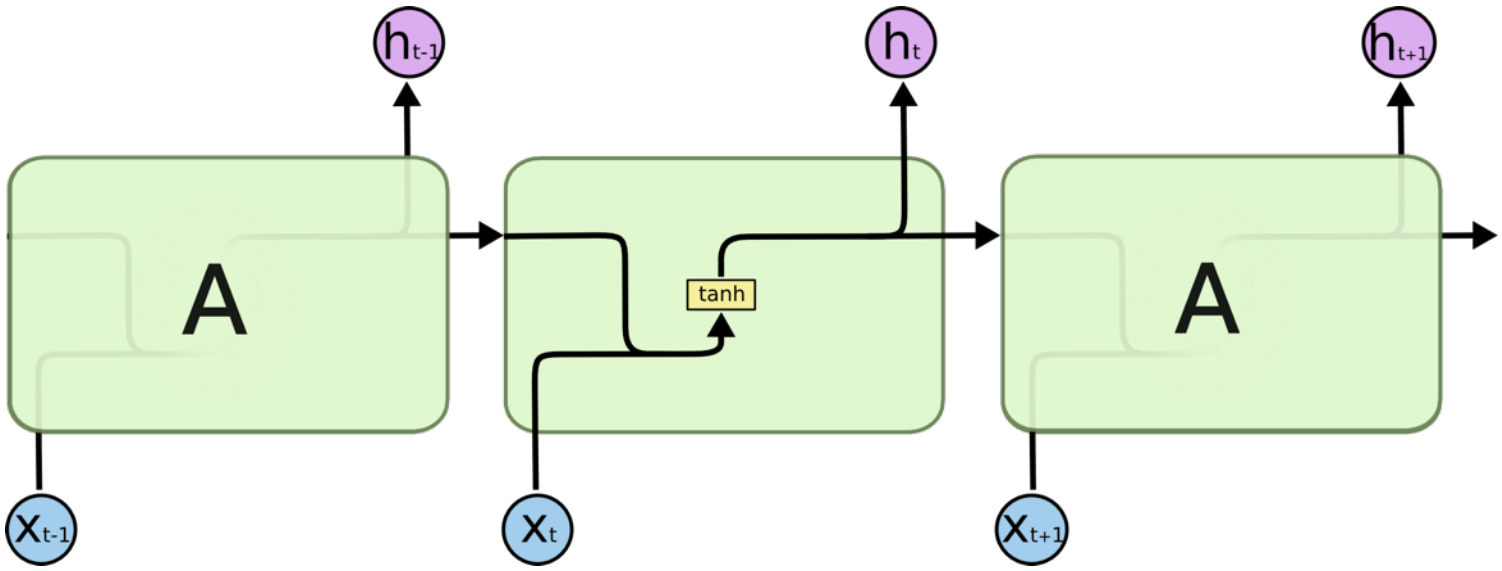
$$o_t = \sigma\left(W^{(o)}x_t + U^{(o)}h_{t-1}\right)$$

$$\tilde{c}_t = \tanh\left(W^{(c)}x_t + U^{(c)}h_{t-1}\right)$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t$$

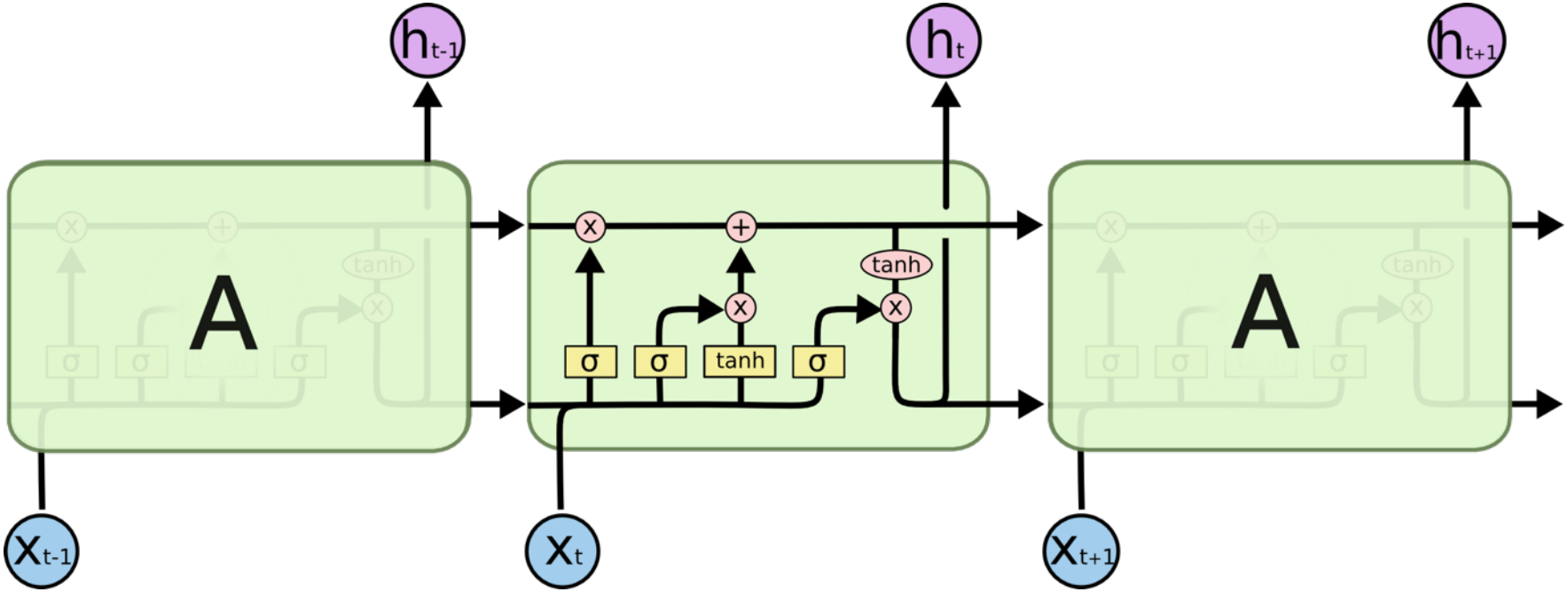
$$h_t = o_t \circ \tanh(c_t)$$

RNN Illustration



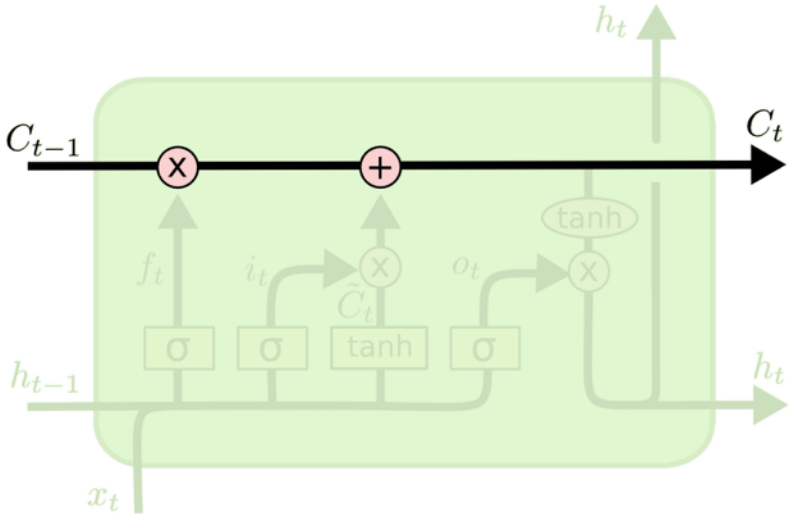
- Only one non-linear unit

LSTM Illustration



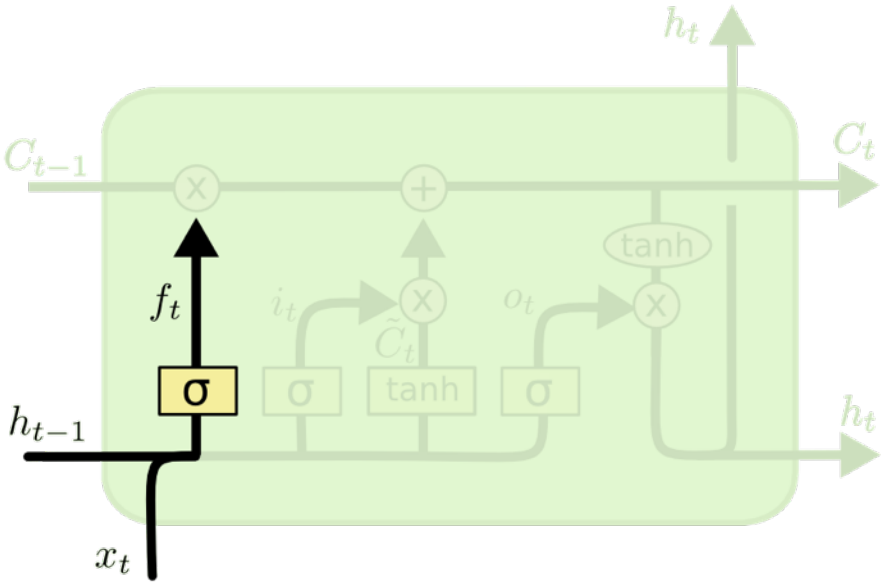
- Various units for different level of information propagation

Main Cell



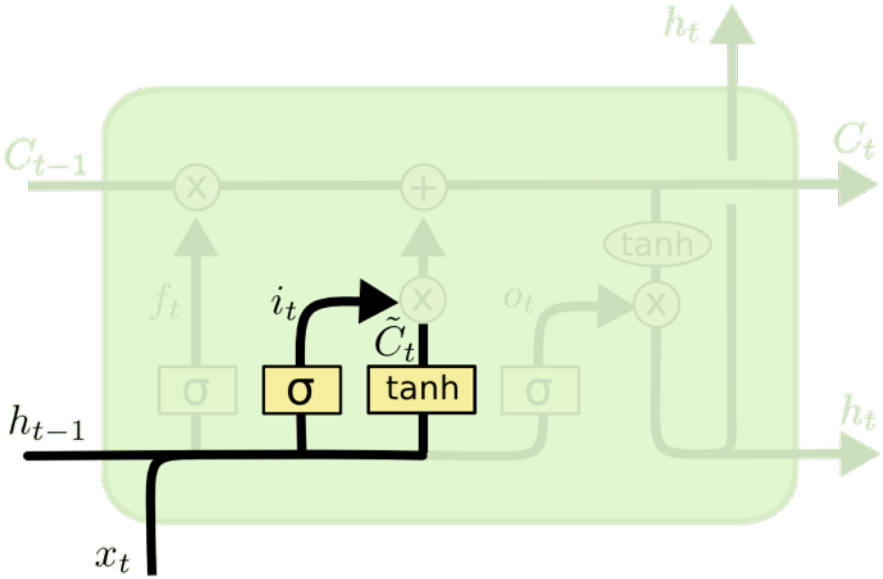
- Cell state carries information between LSTM Cells

First Step : Forget Gate



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

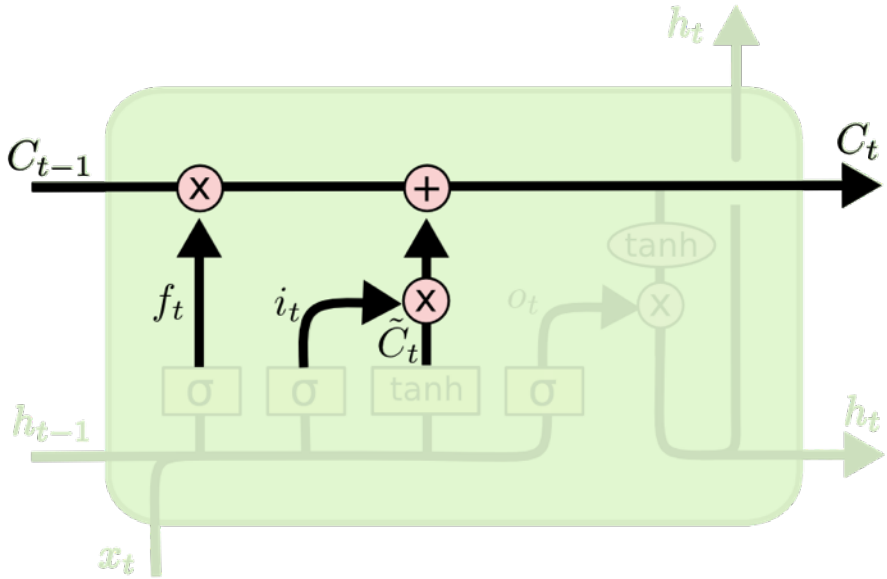
Second Step : Input Gate



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

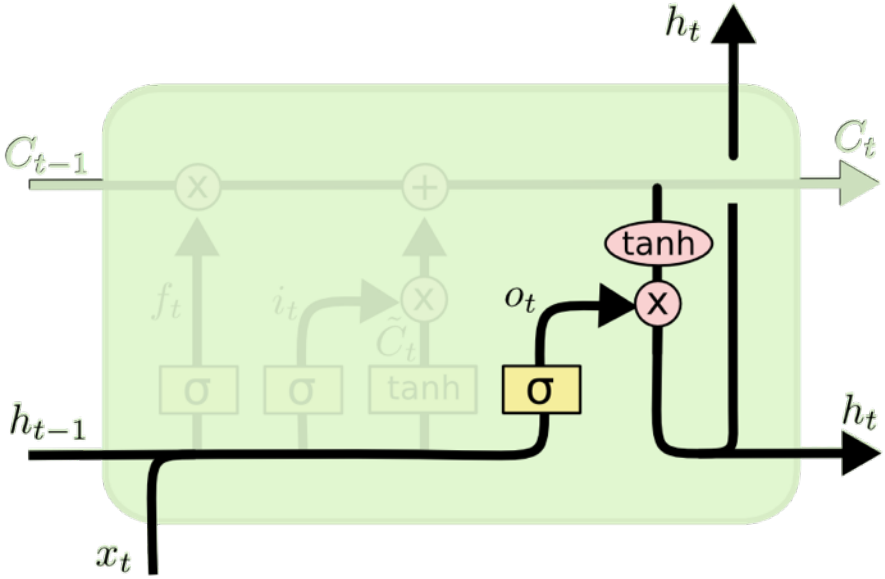
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Third Step : Cell State



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Fourth Step : Hidden State



$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$
$$h_t = o_t * \tanh\left(C_t\right)$$

LSTM Comparison

Method	test BLEU score (ntst14)
Bahdanau et al. [2]	28.45
Baseline System [29]	33.30
Single forward LSTM, beam size 12	26.17
Single reversed LSTM, beam size 12	30.59
Ensemble of 5 reversed LSTMs, beam size 1	33.00
Ensemble of 2 reversed LSTMs, beam size 12	33.27
Ensemble of 5 reversed LSTMs, beam size 2	34.50
Ensemble of 5 reversed LSTMs, beam size 12	**34.81**

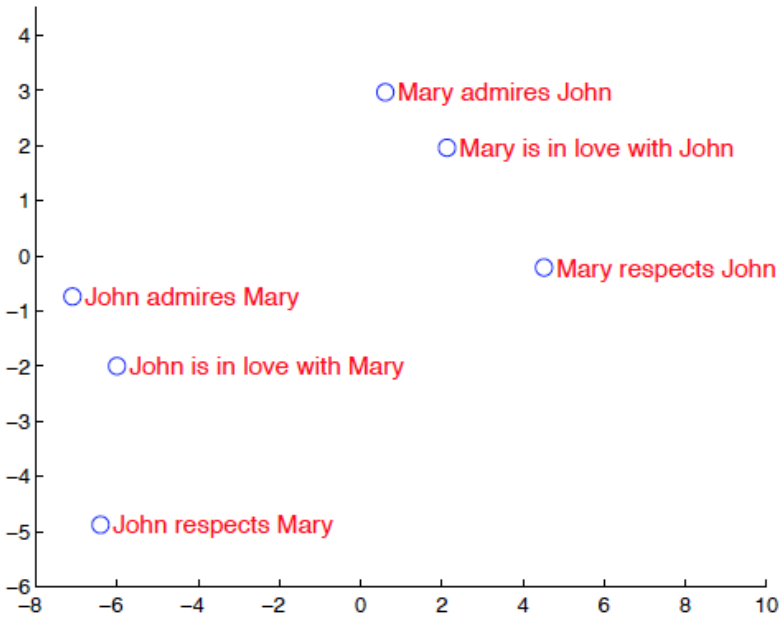
Table 1: The performance of the LSTM on WMT'14 English to French test set (ntst14). Note that an ensemble of 5 LSTMs with a beam of size 2 is cheaper than of a single LSTM with a beam of size 12.

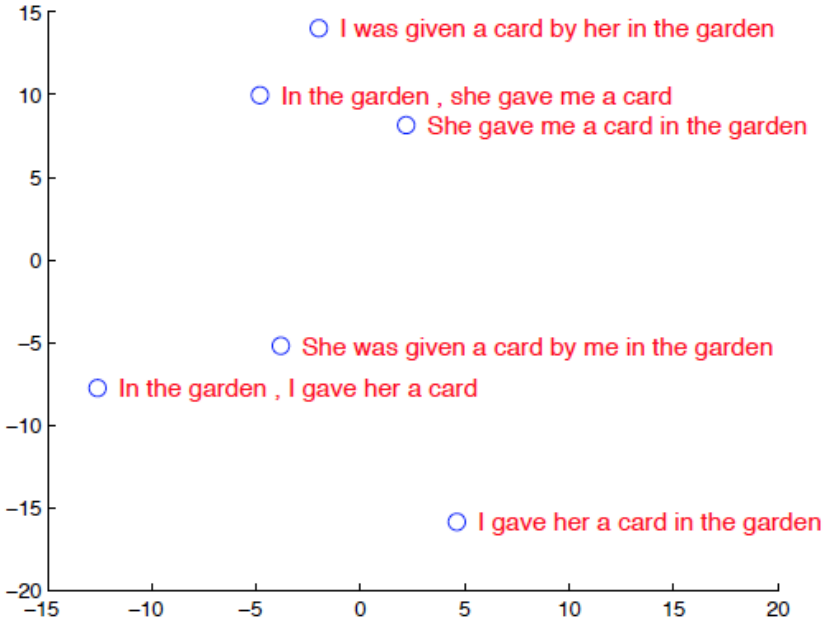
Method	test BLEU score (ntst14)
Baseline System [29]	33.30
Cho et al. [5]	34.54
Best WMT'14 result [9]	**37.0**
Rescoring the baseline 1000-best with a single forward LSTM	35.61
Rescoring the baseline 1000-best with a single reversed LSTM	35.85
Rescoring the baseline 1000-best with an ensemble of 5 reversed LSTMs	**36.5**
Oracle Rescoring of the Baseline 1000-best lists	~45

Sequence to Sequence Learning by Sutskever et al. 2014

Machine Translation Examples

PCA of vectors from last time step hidden layer





Sequence to Sequence Learning by Sutskever et al. 2014

Memory Networks

- Class of models that combine large memory with learning component that can read and write to it.
- Most ML has limited memory which is more-or-less all that's needed for "low level" tasks e.g. object detection.
- **Motivation**: long-term memory is required to read a story (or watch a movie) and then e.g. answer questions about it.
- Study this by building a simple simulation to generate "stories" and try on some real QA data

Reading Comprehension

James the Turtle was always getting in trouble. Sometimes he'd reach into the freezer and empty out all the food. Other times he'd sled on the deck and get a splinter. His aunt Jane tried as hard as she could to keep him out of trouble, but he was sneaky and got into lots of trouble behind her back.

One day, James thought he would go into town and see what kind of trouble he could get into. **He went to the grocery store and pulled all the pudding off the shelves and ate two jars.** Then he walked to the fast food restaurant and ordered 15 bags of fries. He didn't pay, and instead headed home.

His aunt was waiting for him in his room. She told James that she loved him, but he would have to start acting like a well-behaved turtle.

After about a month, and after getting into lots of trouble, James finally made up his mind to be a better turtle.

Q: What did James pull off of the shelves in the grocery store?

A) pudding B) fries C) food D) splinters

Problems

- It's hard for this data to lead to design good ML models …

1) Not enough data to train on (660 stories total).

2) If we get something wrong we don't really understand why: every question potentially involves a different kind of reasoning, our model has to do a lot of different things.

- **Solution**: focus on **simpler** (toy) subtasks where we can generate data to check what the models we design can and cannot do.

Example

Dataset in simulation command format. Dataset after adding a simple grammar.

antoine go kitchen	Antoine went to the kitchen.
antoine get milk	Antoine picked up the milk.
antoine go office	Antoine travelled to the office.
antoine drop milk	Antoine left the milk there.
antoine go bathroom	Antoine went to the bathroom.
where is milk ? (A: office)	Where is the milk now? (A: office)
where is antoine ? (A: bathroom)	Where is Antoine? (A: bathroom)

Data Generation

Aim: built a simple simulation which behaves much like a classic text adventure game. The idea is that generating text within this simulation allows us to ground the language used.

Actions:

go <location>, get <object>, get <object1> from <object2>,
put <object1> in/on <object2>, give <object> to <actor>,
drop <object>, look, inventory, examine <object>.

Constraints on actions:

- an actor cannot get something that they or someone else already has
- they cannot go to a place they are already at
- cannot drop something they do not already have

(1) Factoid QA with Single Supporting Fact

John is in the playground.
Bob is in the office.
Where is John? A:playground

(2) Factoid QA with Two Supporting Facts

John is in the playground.
Bob is in the office.
John picked up the football.
Bob went to the kitchen.
Where is the football? A:playground
Where was Bob before the kitchen? A:office

Memory Networks

MemNNs have four component networks (which may or may not have shared parameters):

I: (input feature map) this converts incoming data to the internal feature representation.

G: (generalization) this updates memories given new input.

O: this produces new output (in feature representation space) given the memories.

R: (response) converts output O into a response seen by the outside world.

This process is applied both train and test time, only difference is model parameter I, G, O and R are not update during test time.

Basic MemNN

I: (input feature map) no conversion, keep original text x.

G: (generalization) stores I(x) in next available slot m_N

O: Loops over all memories k=1 or 2 times:

- 1st loop max: finds best match m_i with x.
- 2nd loop max: finds best match m_J with (x, m_i).
- The output o is represented with (x, mi, m_J).

R: (response) ranks all words in the dictionary given o and returns best single word. (OR: use a full RNN here)

RNN: [x, o1, o2,…, r] feed into RNN, Test time: [x, o1, o2,…]

Matching Function : First Loop

- For a given Q, we want a good match to the relevant memory slot(s) containing the answer, e.g.:

  Match (Where is the football ?, John picked up the football)

- Use a $q^T U^T U d$ embedding model with word embedding features.
  - LHS features: Q:Where Q:is Q:the Q:football Q:?
  - RHS features: D:John D:picked D:up D:the D:football
  - QDMatch:the QDMatch:football

(QDMatch:football is a feature to say there's a Q&A word match, which can help.)

The parameters U are trained with a margin ranking loss: supporting facts should score higher than non-supporting facts.

Matching Function : Second Loop

- On the 2nd hop, match question & 1st hop to new fact:

Match([Where is the football ?, John picked up the football],
John is in the playground)

- Use the same $q^T U^T U d$ embedding model:
  - LHS features: Q:Where Q:is Q:the Q:football Q:? Q2: John Q2:picked Q2:up Q2:the Q2:football
  - RHS features: D:John D:is D:in D:the D:playground QDMatch:the QDMatch:is ..Q2DMatch:John

generated by PCA

bAbI Experiment

10k sentences. (Actor: only ask questions about actors.)

- Difficulty: how many sentences in the past when entity mentioned.
- Fully supervised (supporting sentences are labeled).
- Compare RNN (no supervision)

and MemNN hops k = 1 or 2

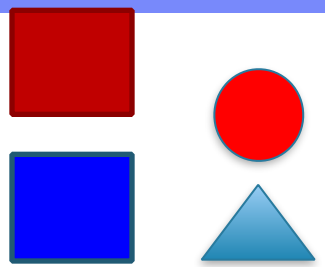
Table 3: Test accuracy on the simulation QA task.

	Difficulty 1			Difficulty 5	
Method	actor w/o before	actor	actor+object	actor	actor+object
RNN	100%	60.9%	27.9%	23.8%	17.8%
LSTM	100%	64.8%	49.1%	35.2%	29.0%
MemNN $k = 1$	97.8%	31.0%	24.0%	21.9%	18.5%
MemNN $k = 1$ (+time)	99.9%	60.2%	42.5%	60.8%	44.4%
MemNN $k = 2$ (+time)	100%	100%	100%	100%	99.9%

Shortcomings

- They model sentences with a bag of words so are likely to fail on tasks such as the 2-argument and 3-argument relation problems.
- They perform only two max operations (k = 2) so they cannot handle questions involving more than two supporting facts
- Unless a RNN is employed in the R module, they are unable to provide multiple answers in the standard setting. This is required for the list (3.8) and path finding (3.19) tasks.

Failing Tasks



3.17. Positional Reasoning

The triangle is to the right of the blue square.
The red square is on top of the blue square.
The red sphere is to the right of the blue square.
Is the red sphere to the right of the blue square? A:yes
Is the red square to the left of the triangle? A:yes

3.19. Path Finding

The kitchen is north of the hallway.
The den is east of the hallway.
How do you go from den to kitchen? A: west, north

Failing Tasks

3.7. Counting

Daniel picked up the football.
Daniel dropped the football.
Daniel got the milk.
Daniel took the apple.
How many objects is Daniel holding? A: two

3.8. Lists / Sets

Daniel picks up the football.
Daniel drops the newspaper.
Daniel picks up the milk.
What is Daniel holding? milk, football

Problems

- Not easy to train via back propagation
- Required supervision at each layer of the network.