CS 388: Natural Language Processing: LSTM Recurrent Neural Networks

Raymond J. Mooney

University of Texas at Austin

Borrows significantly from:

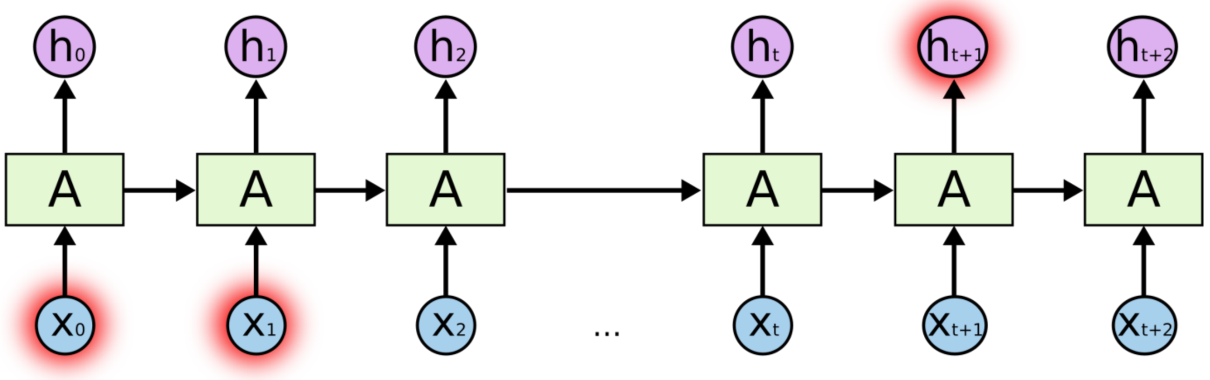
http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Vanishing/Exploding Gradient Problem

- Backpropagated errors multiply at each layer, resulting in exponential decay (if derivative is small) or growth (if derivative is large).
- Makes it very difficult train deep networks, or simple recurrent networks over many time steps.

Long Distance Dependencies

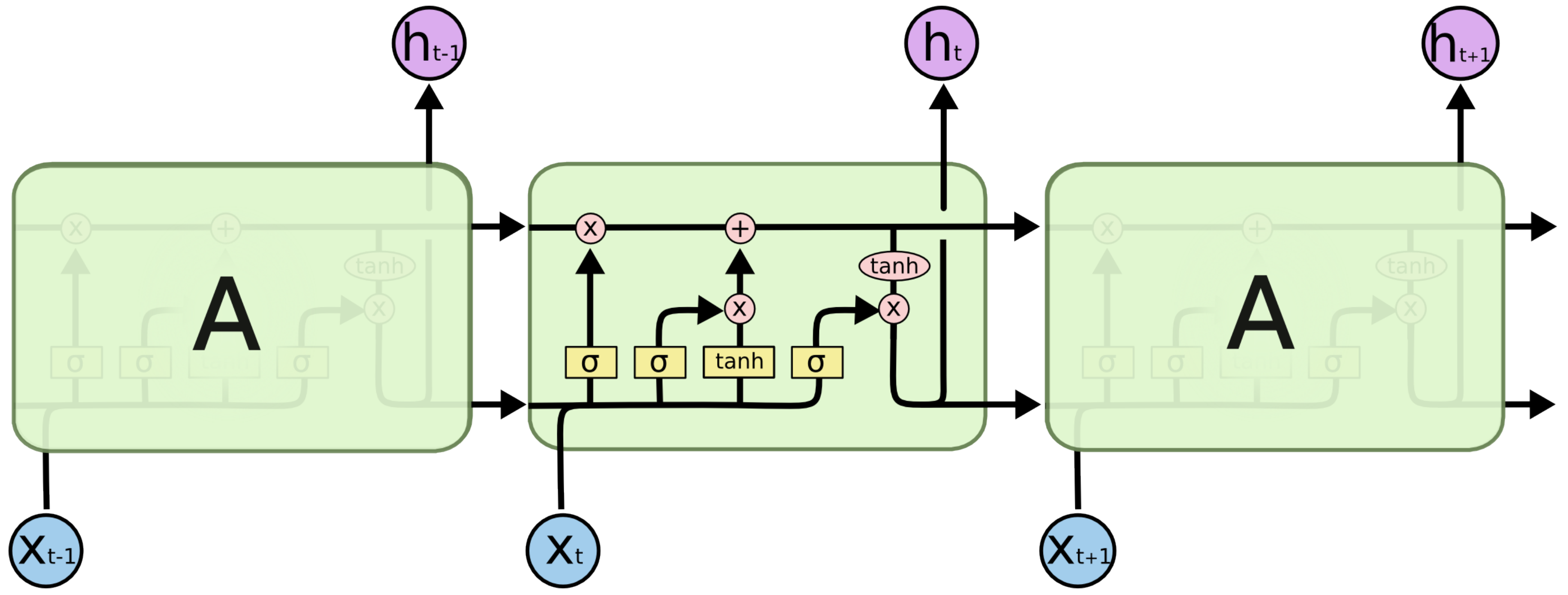
- It is very difficult to train SRNs to retain information over many time steps
- This make is very difficult to learn SRNs that handle long-distance dependencies, such as subject-verb agreement.

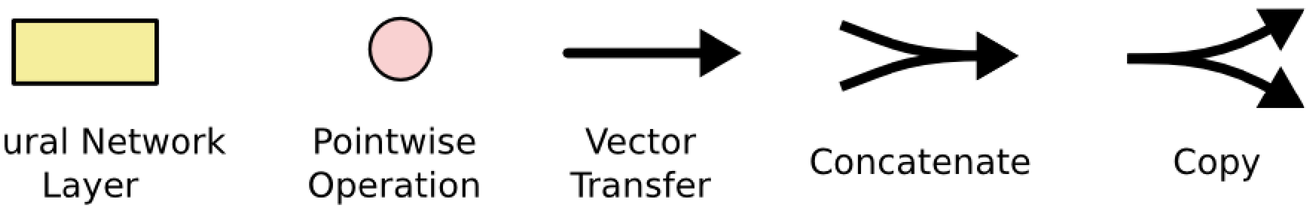


Long Short Term Memory

- LSTM networks, add additional gating units in each memory cell.
  - Forget gate
  - Input gate
  - Output gate
- Prevents vanishing/exploding gradient problem and allows network to retain state information over longer periods of time.

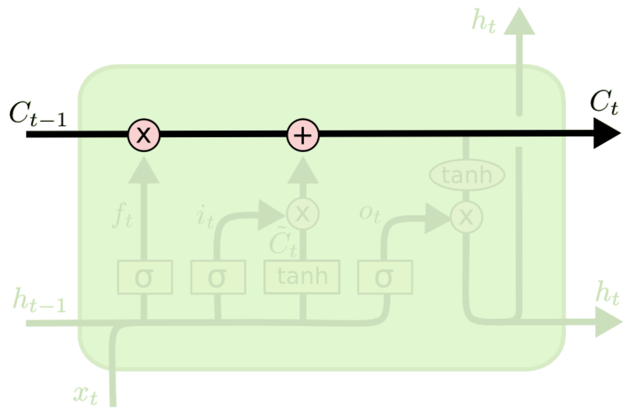
LSTM Network Architecture





Cell State

- Maintains a vector C_t that is the same dimensionality as the hidden state, h_t
- Information can be added or deleted from this state vector via the forget and input gates.

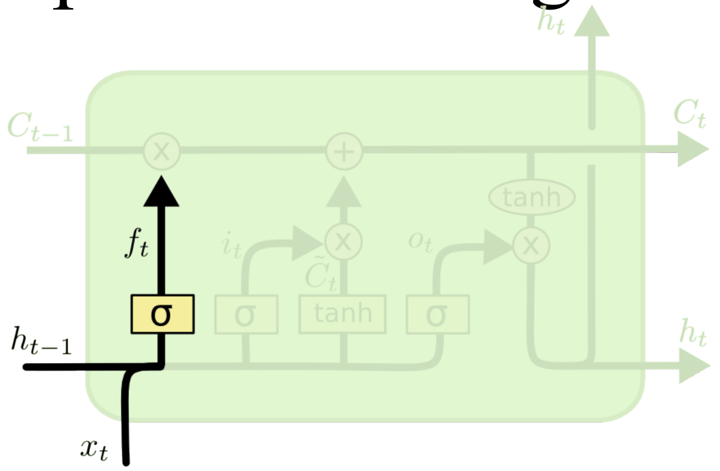


Cell State Example

- Want to remember person & number of a subject noun so that it can be checked to agree with the person & number of verb when it is eventually encountered.
- Forget gate will remove existing information of a prior subject when a new one is encountered.
- Input gate "adds" in the information for the new subject.

Forget Gate

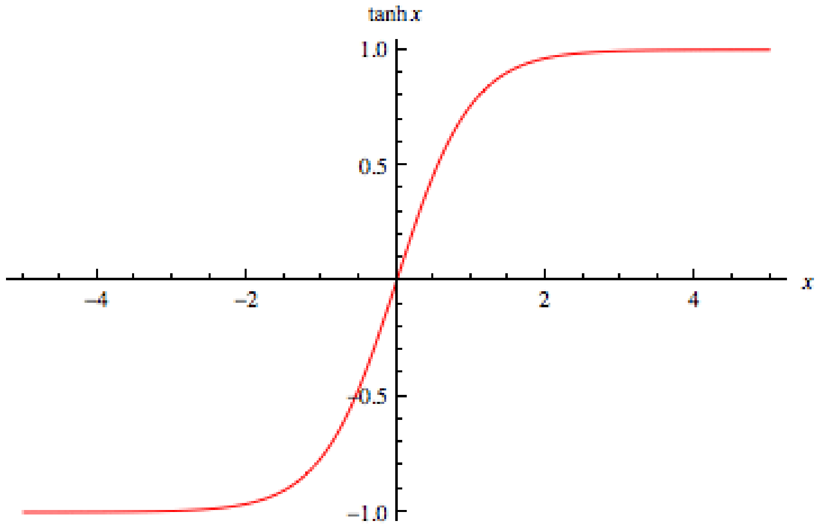
- Forget gate computes a 0-1 value using a logistic sigmoid output function from the input, x_t, and the current hidden state, h_t:
- Multiplicatively combined with cell state, "forgetting" information where the gate outputs something close to 0.



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

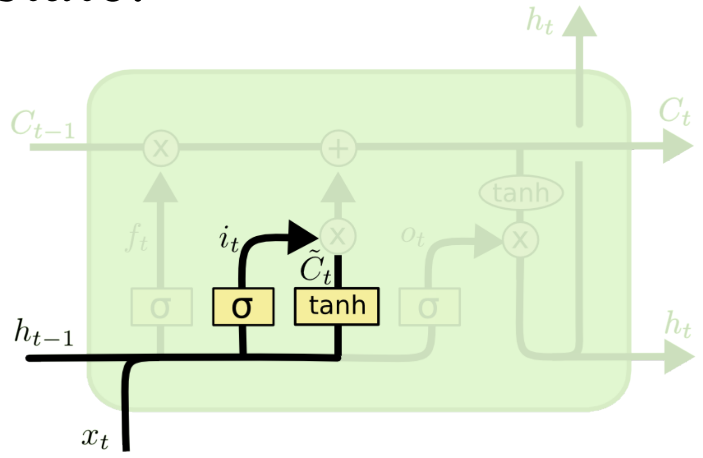
Hyperbolic Tangent Units

- Tanh can be used as an alternative nonlinear function to the sigmoid logistic (0-1) output function.
- Used to produce thresholded output between –1 and 1.



Input Gate

- First, determine which entries in the cell state to update by computing 0-1 sigmoid output.
- Then determine what amount to add/subtract from these entries by computing a tanh output (valued –1 to 1) function of the input and hidden state.

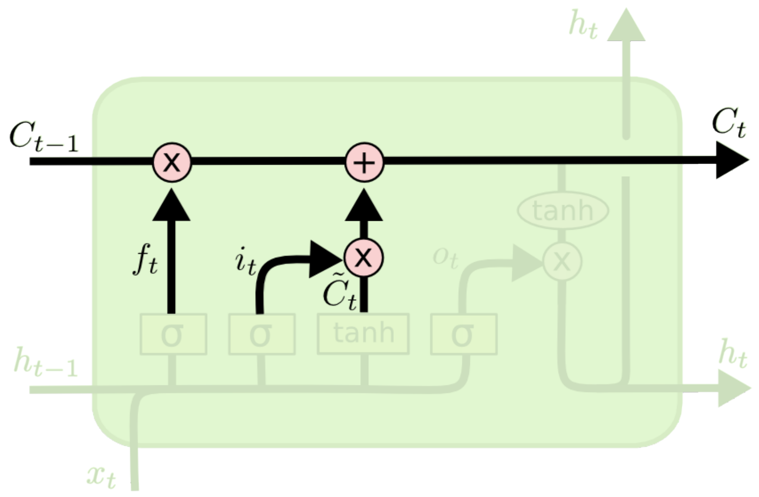


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

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

Updating the Cell State

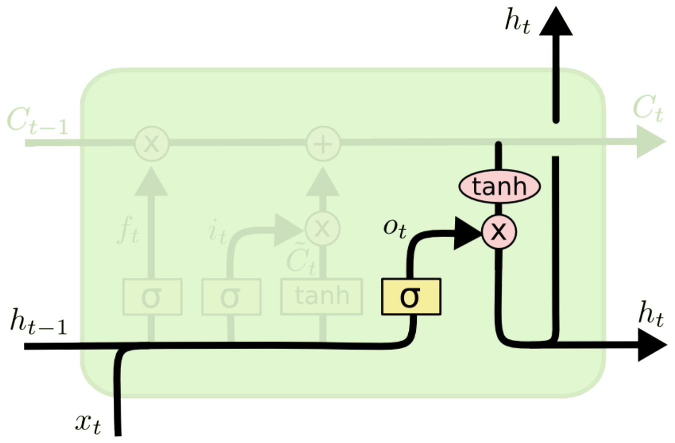
- Cell state is updated by using component-wise vector multiply to "forget" and vector addition to "input" new information.



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

Output Gate

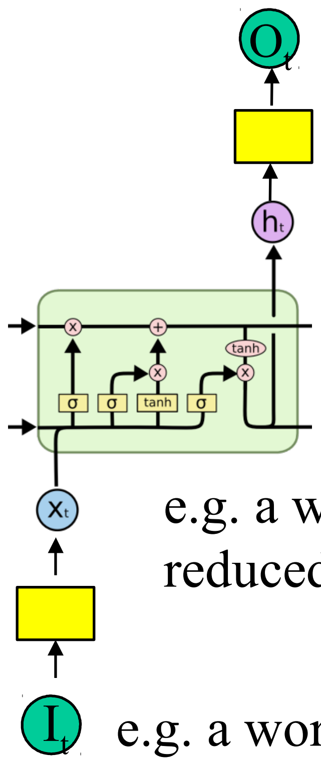
- Hidden state is updated based on a "filtered" version of the cell state, scaled to –1 to 1 using tanh.
- Output gate computes a sigmoid function of the input and current hidden state to determine which elements of the cell state to "output".



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

Overall Network Architecture

- Single or multilayer networks can compute LSTM inputs from problem inputs and problem outputs from LSTM outputs.



e.g. a POS tag as a "one hot" vector

e.g. a word "embedding" with reduced dimensionality

e.g. a word as a "one hot" vector

LSTM Training

- Trainable with backprop derivatives such as:
  - Stochastic gradient descent (randomize order of examples in each epoch) with momentum (bias weight changes to continue in same direction as last update).
  - ADAM optimizer (Kingma & Ma, 2015)
- Each cell has many parameters (W_f, W_i, W_C, W_o)
  - Generally requires lots of training data.
  - Requires lots of compute time that exploits GPU clusters.

General Problems Solved with LSTMs

- Sequence labeling
  - Train with supervised output at each time step computed using a single or multilayer network that maps the hidden state (h_t) to an output vector (O_t).
- Language modeling
  - Train to predict next input ($O_t = I_{t+1}$)
- Sequence (e.g. text) classification
  - Train a single or multilayer network that maps the final hidden state (h_n) to an output vector (O).

Sequence to Sequence Transduction (Mapping)

- Encoder/Decoder framework maps one sequence to a "deep vector" then another LSTM maps this vector to an output sequence.

$$\begin{tikzcd} I_1, I_2, \ldots, I_n \arrow[r] & \boxed{\text{Encoder LSTM}} \arrow[r] & h_n \arrow[r] & \boxed{\text{Decoder LSTM}} \arrow[r] & O_1, O_2, \ldots, O_m \end{tikzcd}$$

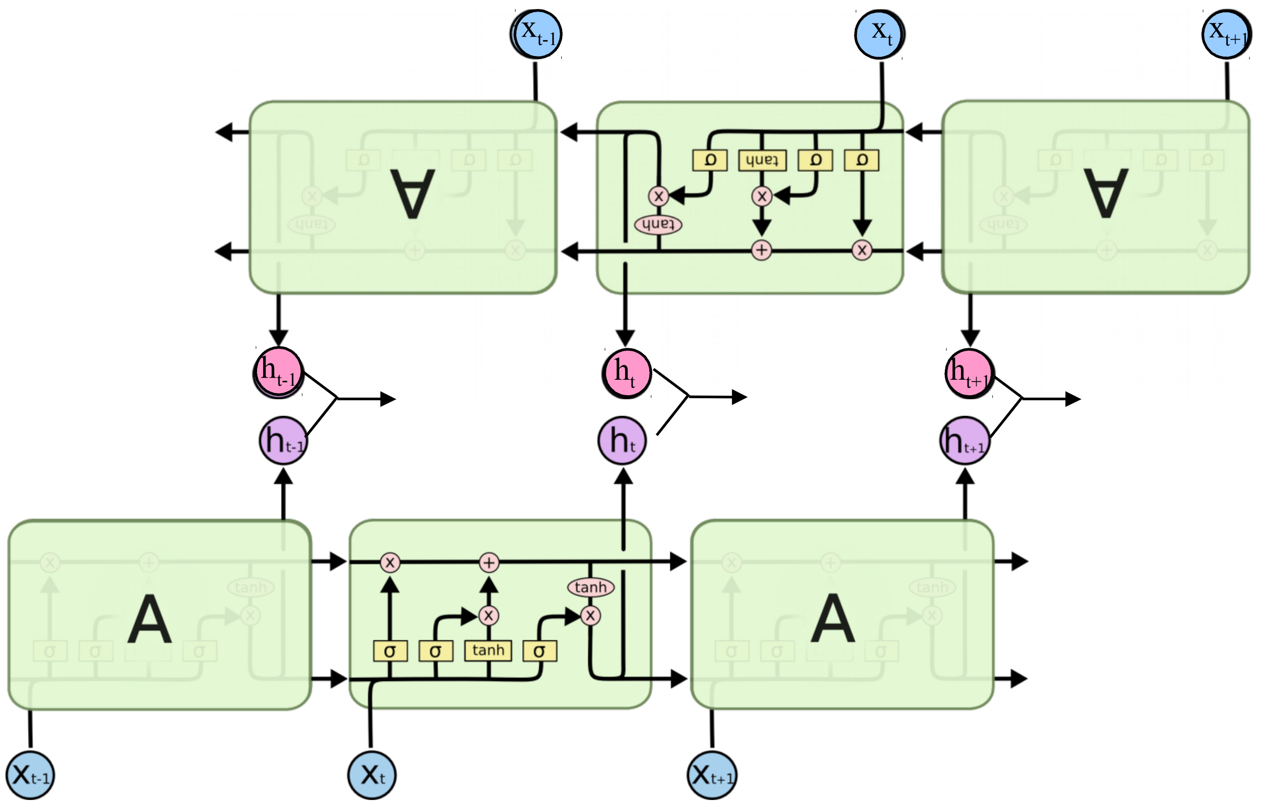
- Train model "end to end" on I/O pairs of sequences.

Successful Applications of LSTMs

- Speech recognition: Language and acoustic modeling
- Sequence labeling
  - POS Tagging
    https://www.aclweb.org/aclwiki/index.php?title=POS_Tagging_(State_of_the_art)
  - NER
  - Phrase Chunking
- Neural syntactic and semantic parsing
- Image captioning: CNN output vector to sequence
- Sequence to Sequence
  - Machine Translation (Sustkever, Vinyals, & Le, 2014)
  - Video Captioning (input sequence of CCN frame outputs)

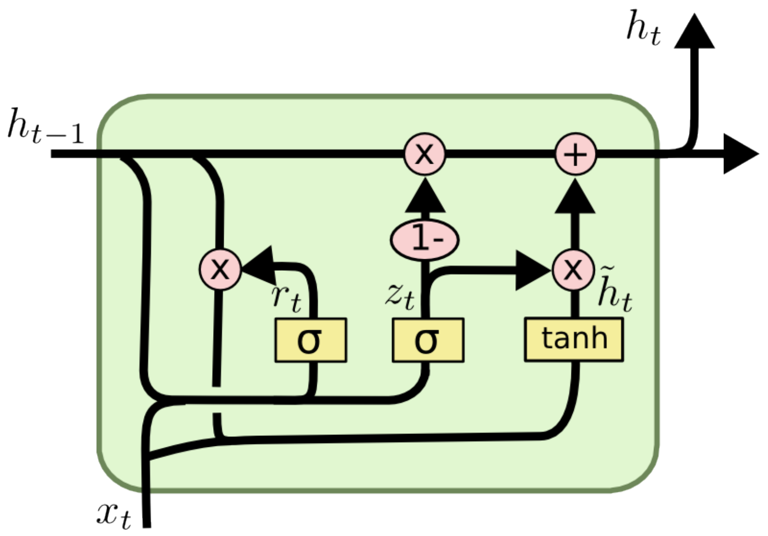
Bi-directional LSTM (Bi-LSTM)

- Separate LSTMs process sequence forward and backward and hidden layers at each time step are concatenated to form the cell output.



Gated Recurrent Unit (GRU)

- Alternative RNN to LSTM that uses fewer gates (Cho, et al., 2014)
  - Combines forget and input gates into "update" gate.
  - Eliminates cell state vector



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

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

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

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

GRU vs. LSTM

- GRU has significantly fewer parameters and trains faster.
- Experimental results comparing the two are still inconclusive, many problems they perform the same, but each has problems on which they work better.

Attention

- For many applications, it helps to add "attention" to RNNs.
- Allows network to learn to attend to different parts of the input at different time steps, shifting its attention to focus on different aspects during its processing.
- Used in image captioning to focus on different parts of an image when generating different parts of the output sentence.
- In MT, allows focusing attention on different parts of the source sentence when generating different parts of the translation.

Attention for Image Captioning (Xu, et al. 2015)

Figure 2. Attention over time. As the model generates each word, its attention changes to reflect the relevant parts of the image. "soft" (top row) vs "hard" (bottom row) attention. (Note that both models generated the same captions in this example.)

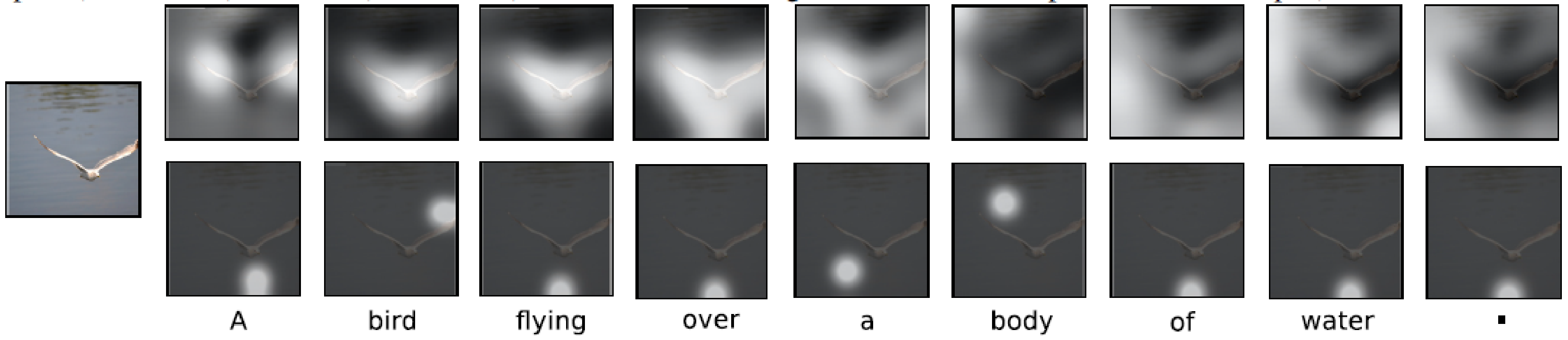
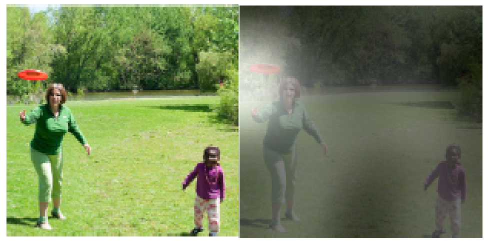
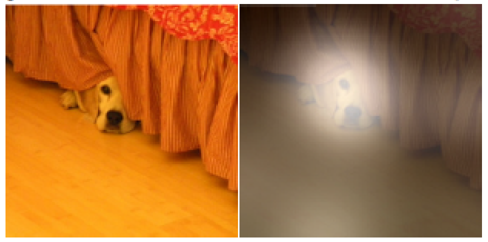
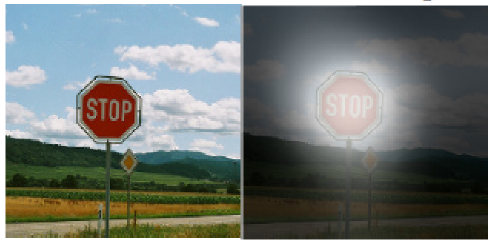
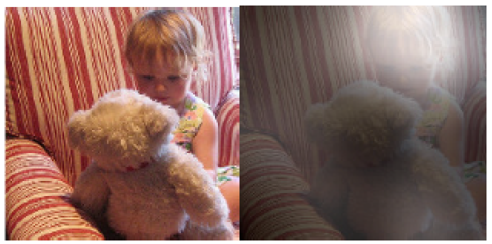


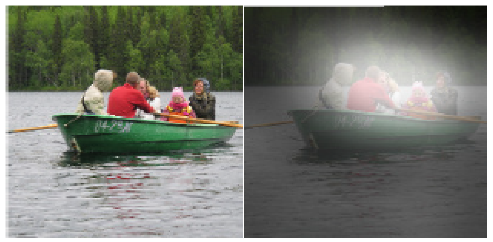
Figure 3. Examples of attending to the correct object (*white* indicates the attended regions, *underlines* indicated the corresponding word)

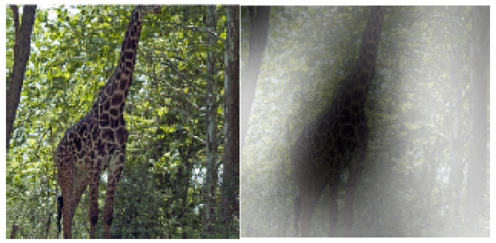

A woman is throwing a frisbee in a park.


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.

Conclusions

- By adding “gates” to an RNN, we can prevent the vanishing/exploding gradient problem.
- Trained LSTMs/GRUs can retain state information longer and handle long-distance dependencies.
- Recent impressive results on a range of challenging NLP problems.