Papers Related to Memory & Attention

Jack Lanchantin
2016/12

@ https://qdata.github.io/deep2Read

Learning to Rank with (a Lot of) Word Features

Bai et al - Information Retrieval 2010

Basic Bag-O'-Words

Bag-of-words + cosine similarity:

- Each doc. $\{d_t\}_{t=1}^{N} \subset \mathbb{R}^{\mathcal{D}}$ is a *normalized* bag-of-words.
- Similarity with query q is: $f(q, d) = q^\top d$

Doesn't deal with synonyms: bag vectors can be orthogonal

No machine learning at all

Latent semantic indexing (LSI)

Learn a linear embedding $\phi(d_i) = Ud_i$ via a reconstruction objective.

- Rank with: $f(q,d) = q^\top U^\top Ud = \phi(q)^\top \phi(d_i)$ [1].

Uses "synonyms": *low-dimensional latent "concepts"*.

Unsupervised machine learning: useful for goal?

[1] $f(q,d) = q^\top (U^\top U + \alpha I)d$ gives better results.
Also, usually normalize this $\rightarrow$ cosine similarity.

Supervised Semantic Indexing (*SSI*)

- Basic model: rank with

$$f(q,d) \quad = \quad q^\top W d = \quad \sum_{i,j=1}^{\mathcal{D}} q_i \; W_{ij} \; d_j$$

  i.e. learn weights of polynomial terms between documents.

- Learn $W \in \mathbb{R}^{\mathcal{D}\times\mathcal{D}}$ (huge!) with click-through data or other labels.


Uses "synonyms"


Supervised machine learning: targeted for goal


Too Big/Slow?! Solution = Constrain W :

low rank $\rightarrow$ *embedding model!*

SSI Improved model: Low Rank W

- **Constrain** W:

$$W = U^\top V + I.$$

- U and V are $N \times \mathcal{D}$ matrices $\rightarrow$ smaller
- Low dimensional "latent concept" space like LSI (same speed).
- Differences: supervised, asymmetric, learns with I.

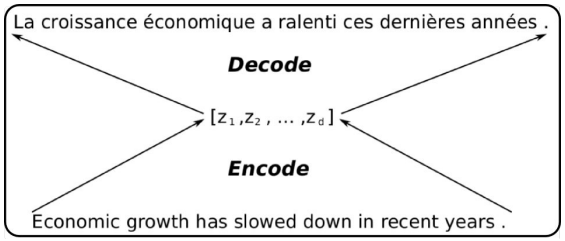
Variants:

- $W = I$: bag-of-words again.
- $W = D$, reweighted bag-of-words related to [Grangier and Bengio, 2005].
- $W = U^\top U + I$: symmetric.

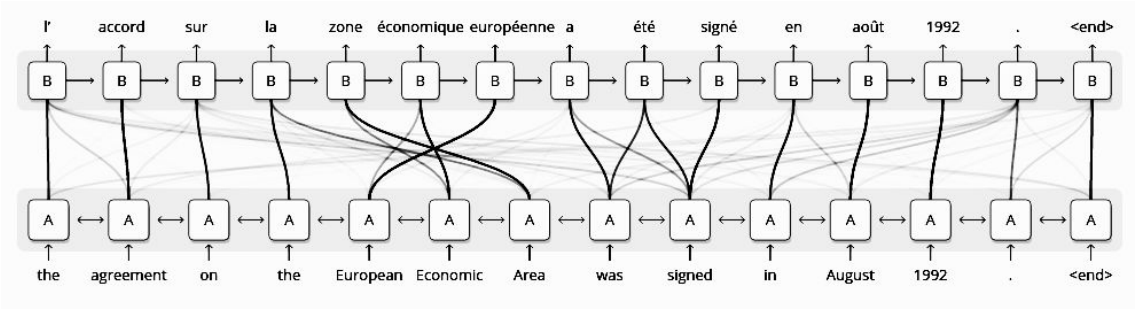
Neural Machine Translation by Jointly Learning to Align and Translate (RNNsearch)

Bahdanau, Cho, Bengio - ICLR 2015

- **Previous models** encode a source sentence into a fixed-length vector from which a decoder generates a translation.



- **This paper** allows a model to automatically (soft-)search and attend to parts of a source sentence that are relevant to predicting a target word



Decoder

Encoder

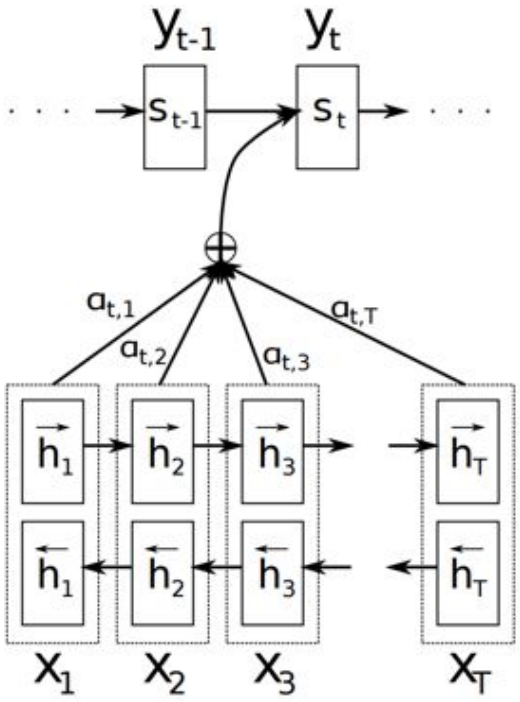


Figure 1: The graphical illustration of the proposed model trying to generate the t-th target word y_t given a source sentence $(x_1, x_2, \ldots, x_T)$.

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})}$$

$$e_{ij} = v_a^\top \tanh(W_a s_{i-1} + U_a h_j)$$

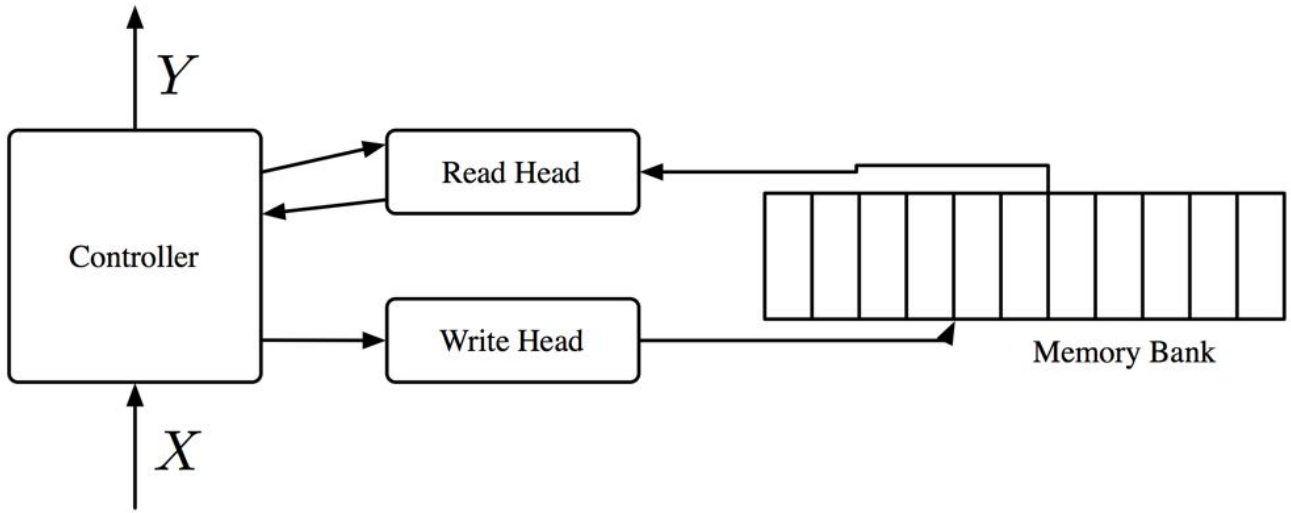
http://distill.pub/2016/augmented-rnns/

End-To-End Memory Networks (torch code)

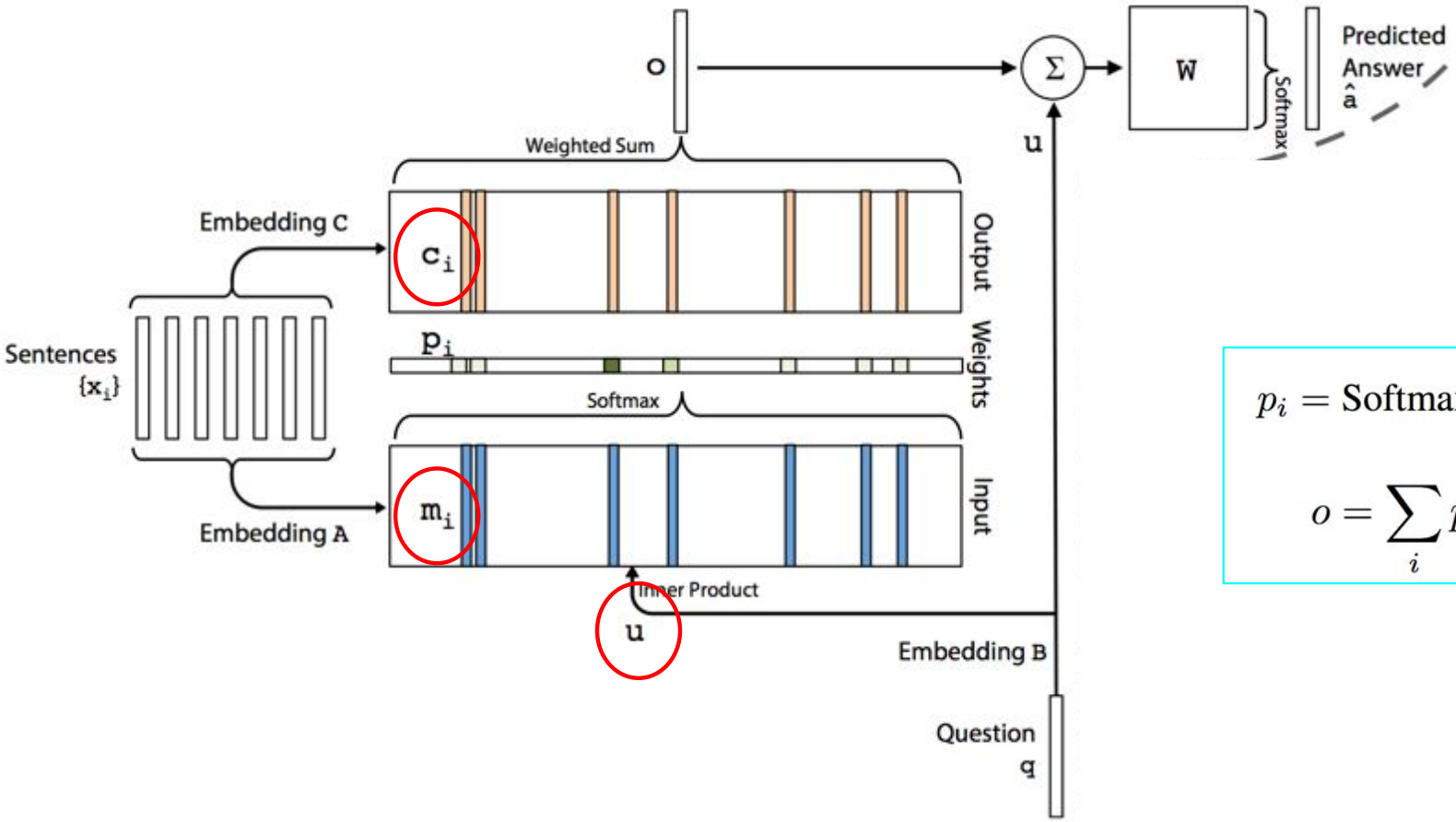
Sukhbaatar, Szlam, Weston, Fergus - NIPS 2015

End-To-End Memory Networks

- **MemNets:** Extend the capabilities of neural networks by coupling them to external memory resources, which they can interact with by attentional processes.
  - Can be seen as an extension of RNNsearch to the case where multiple computational steps (hops) are performed per output symbol.



End-To-End Memory Networks

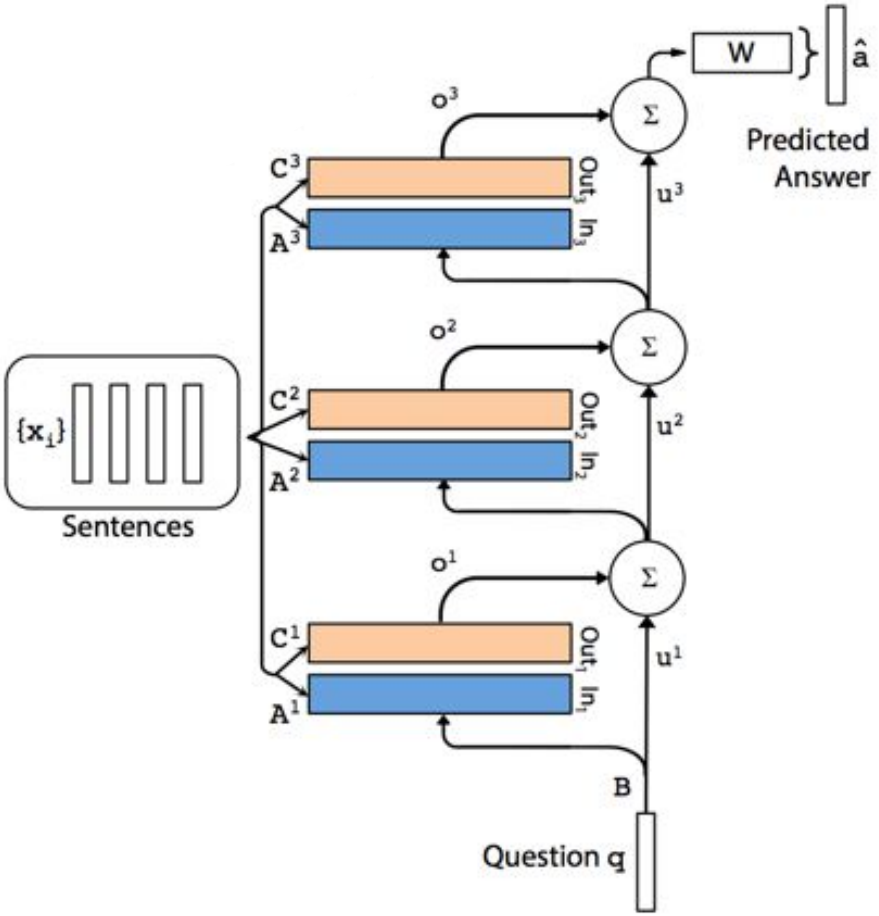


$$p_i = \text{Softmax}(u^T m_i)$$

$$o = \sum_i p_i c_i$$

End-To-End Memory Networks

Extension to multiple "hops" or "layers"

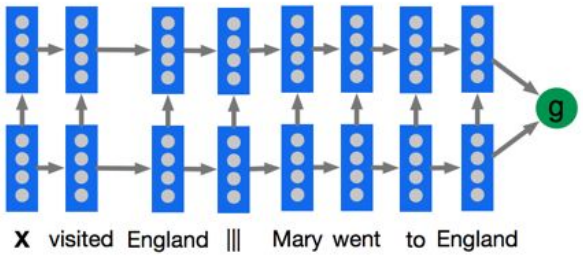


Teaching Machines to Read and Comprehend

Hermann, Kočiský, Grefenstette, Espeholt, Kay, Suleyman, Blunsom - NIPS 2015

Teaching Machines to Read and Comprehend

Extend the LSTM reader for **question answering** by attending to certain parts of the document based on the query



(c) A two layer Deep LSTM Reader with the question encoded before the document.

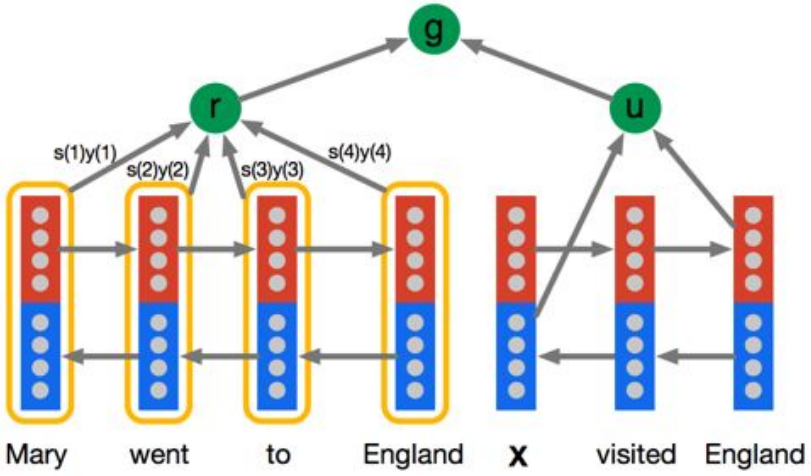
Propose 2 Models for question answering:

1. **Attentive Reader**
2. **Impatient Reader**

Teaching Machines to Read and Comprehend

Attentive Reader:

- Encoding *u* of query is a concatenation of LSTM output of query.
- Encoding *r* of the document is formed by a weighted sum of LSTM output of doc
- Question is answered with joint document and query embedding via a nonlinear function



(a) Attentive Reader.

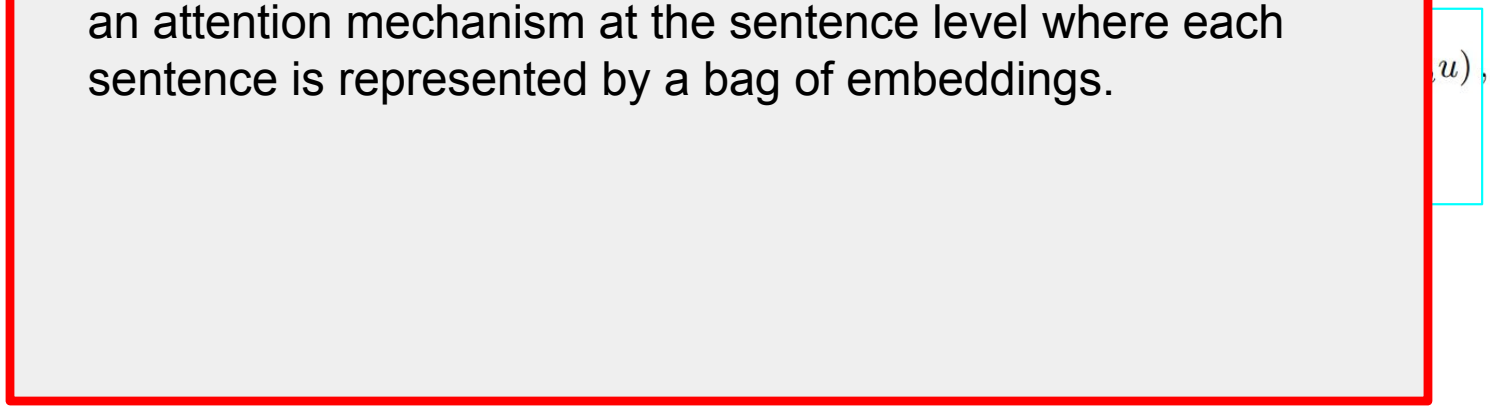
$$
\begin{aligned}
m(t) &= \tanh\left(W_{ym} y_d(t) + W_{um} u\right), \\
s(t) &\propto \exp\left(\mathrm{w}_{ms}^{\intercal} m(t)\right), \\
r &= y_d s,
\end{aligned}
$$

Teaching Machines to Read and Comprehend

Attentive Reader:

- Encoding *u* of query is a concatenation of LSTM output of query.
- Encod[illegible] of doc
- Quest[illegible] near

The Attentive Reader can be viewed as a generalisation of Memory Networks for question answering. MemNets employ an attention mechanism at the sentence level where each sentence is represented by a bag of embeddings.

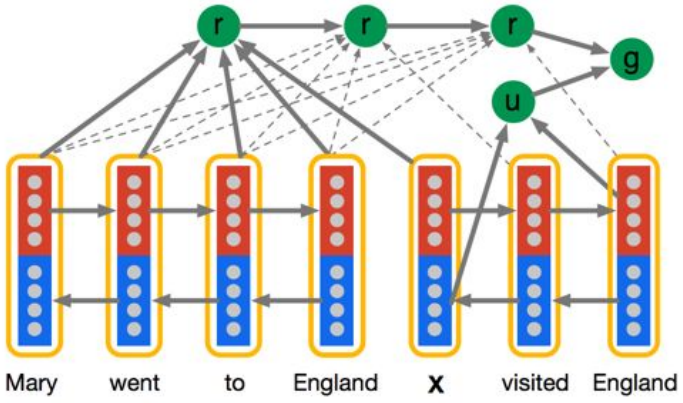


(a) Attentive Reader.

Teaching Machines to Read and Comprehend

Impatient Reader:

- Extend attentive reader by equipping the model with the ability to reread from the document as each query token is read



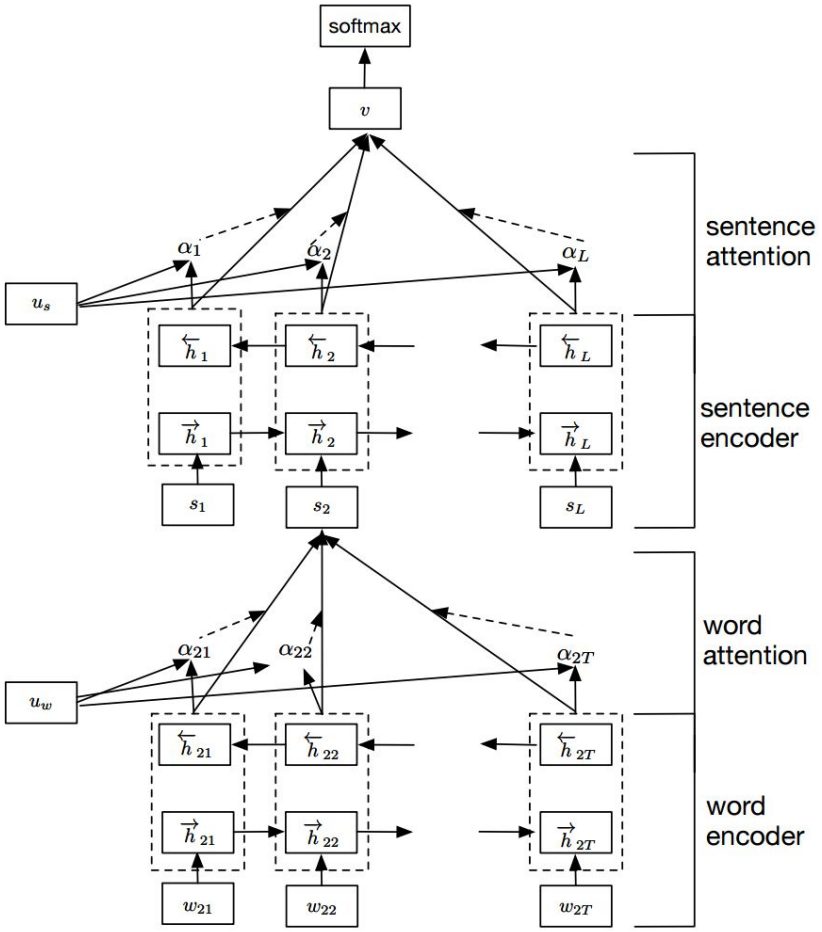
(b) Impatient Reader.

$$
\begin{aligned}
m(i,t) &= \tanh\left(W_{dm}y_d(t) + W_{rm}r(i-1) + W_{qm}y_q(i)\right), \quad 1 \le i \le |q|, \\
s(i,t) &\propto \exp\left(\mathrm{w}_{ms}^{\intercal} m(i,t)\right), \\
r(0) &= \mathbf{r_0}, \quad r(i) = y_d^{\intercal} s(i) + \tanh\left(W_{rr}r(i-1)\right) \quad 1 \le i \le |q|.
\end{aligned}
$$

Hierarchical Attention Networks for Document Classification

Yang, Yang, Dye , He, Smola, Hovy - NAACL 2016

Hierarchical Attention Networks for Document Classification



$$u_i = \tanh(W_s h_i + b_s),$$
$$\alpha_i = \frac{\exp(u_i^\top u_s)}{\sum_i \exp(u_i^\top u_s)},$$
$$v = \sum_i \alpha_i h_i,$$

$$u_{it} = \tanh(W_w h_{it} + b_w)$$
$$\alpha_{it} = \frac{\exp(u_{it}^\top u_w)}{\sum_t \exp(u_{it}^\top u_w)}$$
$$s_i = \sum_t \alpha_{it} h_{it}.$$

Neural Turing Machines (torch code)

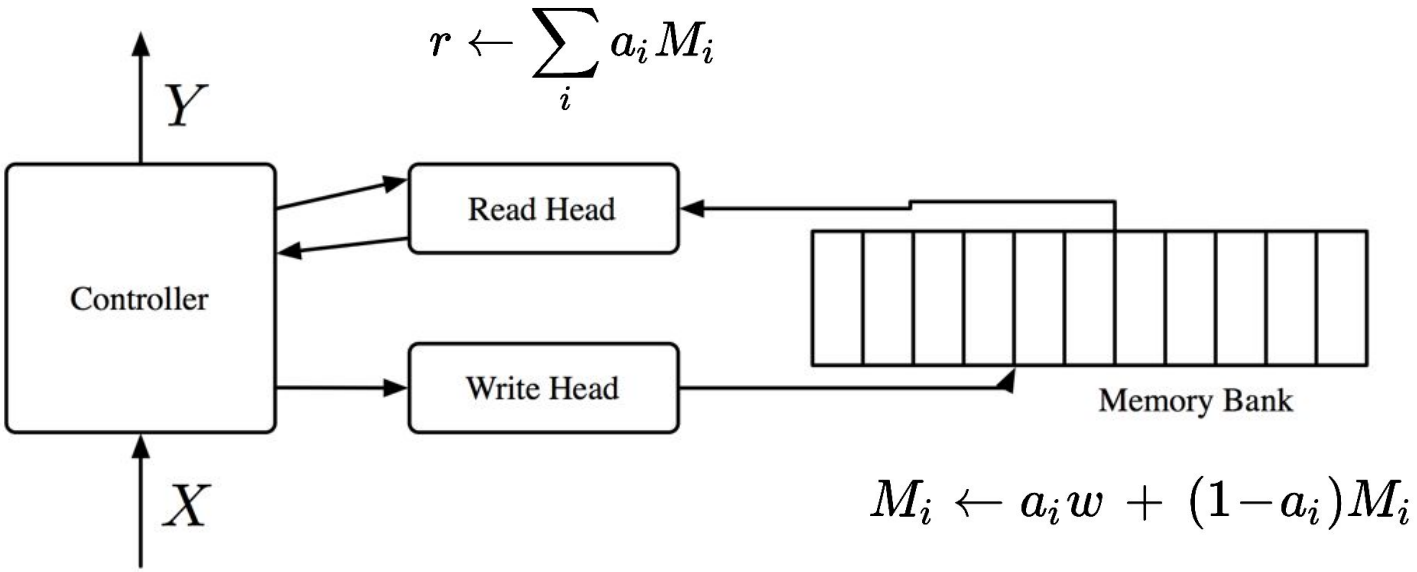
Graves, Wayne, Danihelka - 2014

Neural Turing Machines

- Similar to MemNets, except
  - combine RNN with an external memory bank
  - have different attention method.
- The system is analogous to a Turing Machine but is differentiable end-to-end
- NTMs can infer simple algorithms such as copying and sorting

http://distill.pub/2016/augmented-rnns/

Neural Turing Machines



Neural Turing Machines

How do NTMs decide which positions in memory to focus their attention on?

- Use a combination of two different methods
  - **Content-based** attention allows NTMs to search through their memory and focus on places that match what they're looking for
  - **Location-based** attention allows relative movement in memory, enabling the NTM to loop

Neural Turing Machines

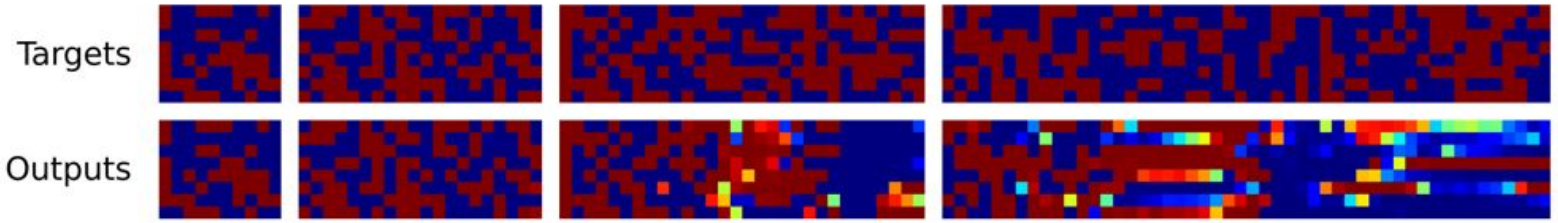
Location-based Attention

In certain tasks the content of a variable is arbitrary, but the variable still needs a recognisable name or address.

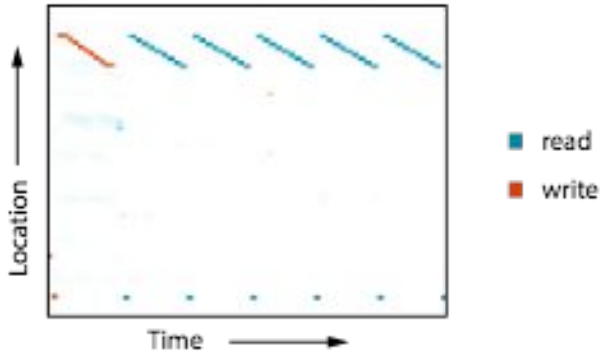
- E.g. arithmetic problems: the variable x and the variable y can take on any two values, but the procedure f(x, y) = x*y should still be defined.
- A controller for this task could take the values of the variables x and y, store them in different addresses, then retrieve them and perform a multiplication algorithm.
    - In this case, the variables are addressed by location, not by content.

Neural Turing Machines

Copy



Repeat copy



Generating Images from Captions with Attention (theano code)

Mansimov, Parisotto, Lei Ba, Salakhutdinov - ICLR 2016

Generating Images from Captions with Attention

Iteratively draw patches on a canvas, while attending to the relevant words in the description.

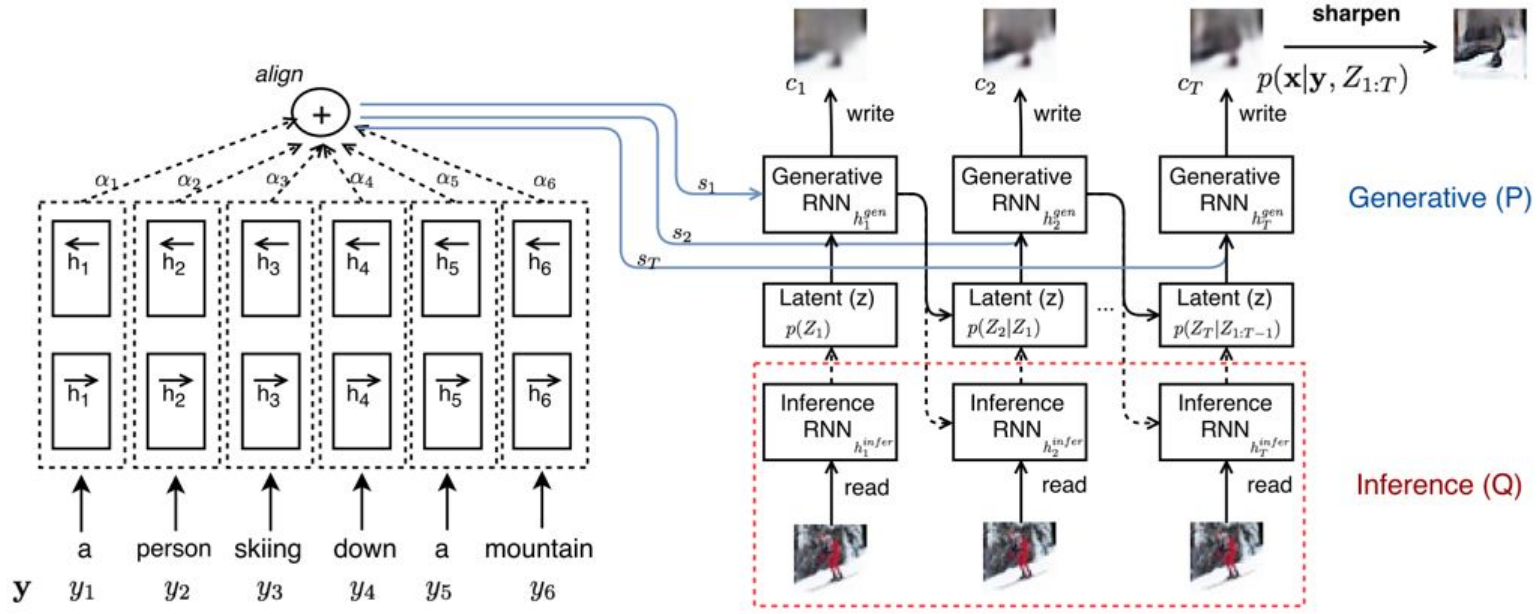


Figure 2: AlignDRAW model for generating images by learning an alignment between the input captions and generating canvas. The caption is encoded using the Bidirectional RNN (left). The generative RNN takes a latent sequence $z_{1:T}$ sampled from the prior along with the dynamic caption representation $s_{1:T}$ to generate the canvas matrix c_T, which is then used to generate the final image $\mathbf{x}$ (right). The inference RNN is used to compute approximate posterior Q over the latent sequence.

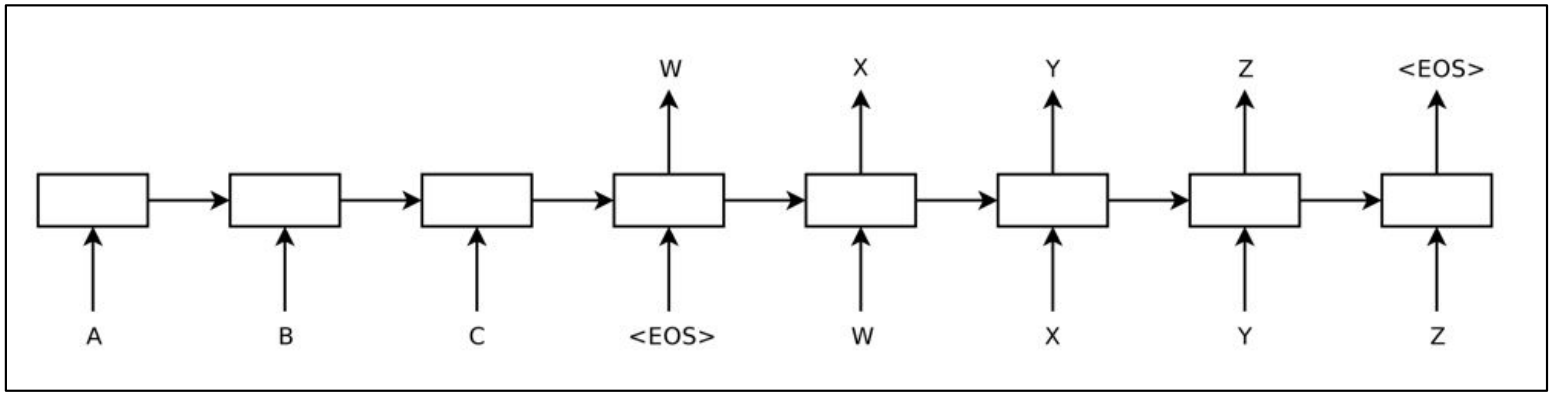
Sequence to Sequence Learning with Neural Networks

Sutskever et al - NIPS 2014

Machine Translation using RNNs

Inputs: sequence of words $x_1, x_2, \ldots, x_T$

Output: sequence of words $y_1, y_2, \ldots, y_{T'}$



$$p(y_1, \ldots, y_{T'} | x_1, \ldots, x_T)$$

Machine Translation using RNNs

Basic RNN:

$$\begin{aligned} h_t &= \text{sigm}\left(W^{\text{hx}} x_t + W^{\text{hh}} h_{t-1}\right) \\ y_t &= W^{\text{yh}} h_t \end{aligned}$$

v = LSTM output of the input sequence = $LSTM(\mathrm{x}_1, \ldots, \mathrm{x}_T)$

RNN for translation:

$$p(y_1, \ldots, y_{T'} | x_1, \ldots, x_T) = \prod_{t=1}^{T'} p(y_t | v, y_1, \ldots, y_{t-1})$$

2 main contributions

1. Use 2 different LSTMs
   - 1 encoder, 1 decoder
2. Reverse input order
   - a,b,c → w,x,y,z
   - c,b,a → w,x,y,z

Order Matters: Sequence to sequence for sets

Vinyals et al - ICLR 2016

Recall: Machine Translation using RNNs

$$X^i = \{x_1^i, x_2^i, \ldots, x_{s_i}^i\}$$

$$Y^i = \{y_1^i, y_2^i, \ldots, y_{t_i}^i\}$$

$$P(Y|X) = \prod_{t=1}^{T} P(y_t|y_1, y_2, \ldots, y_{t-1}, X)$$

Input Sets

When the input X corresponds to a sequence (e.g. a sentence), reasonable to use an RNN

- How should we encode X if it does not correspond naturally to a sequence (e.g. an unordered set of elements)?

Input: unordered set $\{x_1, x_2, \ldots, x_s\}$

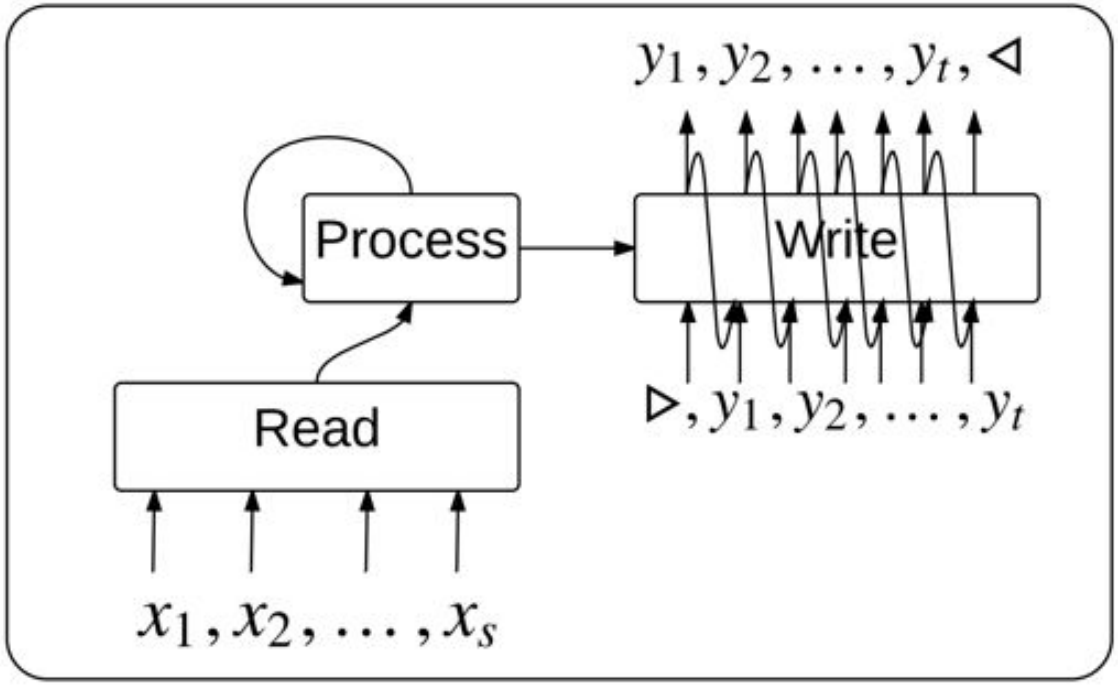
Output: sequence corresponding to the set $y_1, y_2, \ldots, y_t$

Input Sets

Task: Sorting n numbers

Approach: Define a model which input order invariant while also using a memory that increases with the size of the set

Read, Process, Write

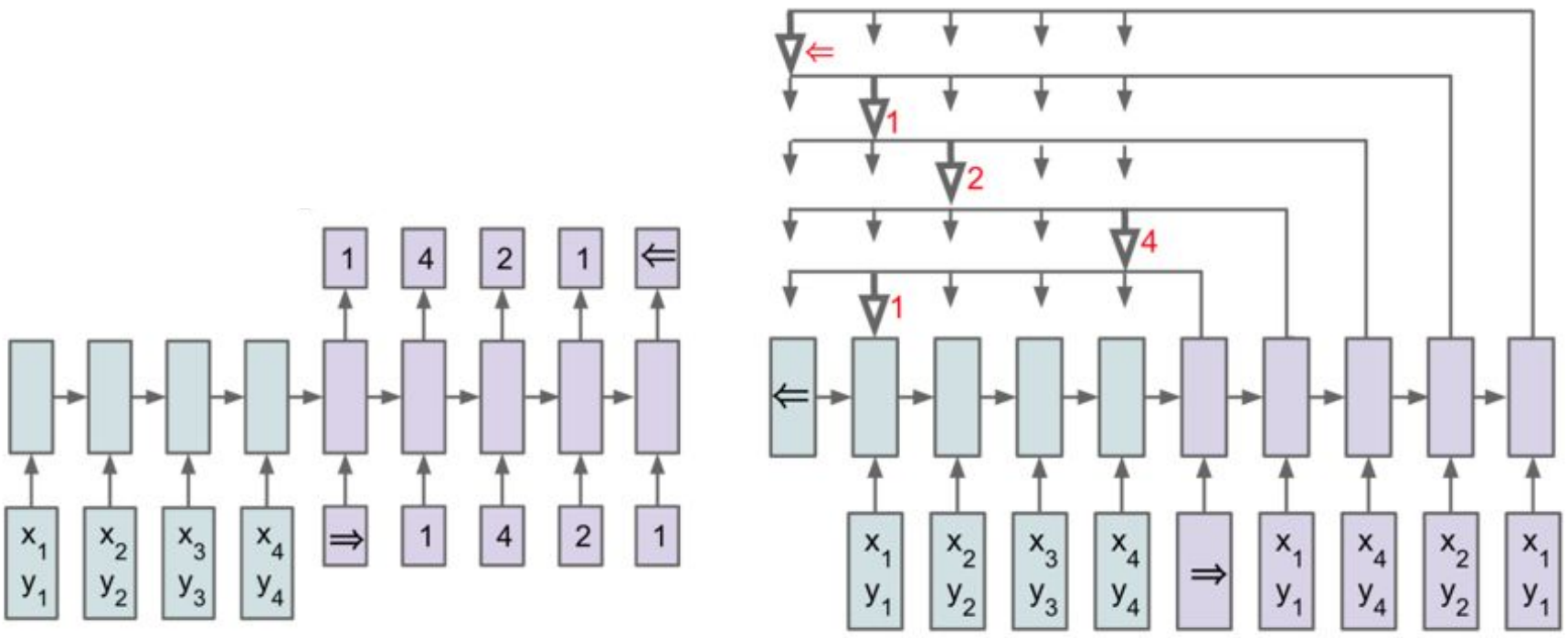


Pointer Networks

Vinyals et al - NIPS 2015

Pointer Networks

Vinyals et al - NIPS 2015



(a) Sequence-to-Sequence

(b) Ptr-Net

Pointer Networks

The sequence-to-sequence model of Section 2.1 uses a softmax distribution over a fixed sized output dictionary to compute $p(C_i|C_1, \ldots, C_{i-1}, \mathcal{P})$ in Equation 1. Thus it cannot be used for our problems where the size of the output dictionary is equal to the length of the input sequence. To solve this problem we model $p(C_i|C_1, \ldots, C_{i-1}, \mathcal{P})$ using the attention mechanism of Equation 3 as follows:

$$\begin{aligned} u_j^i &= v^T \tanh(W_1 e_j + W_2 d_i) \quad j \in (1, \ldots, n) \\ p(C_i|C_1, \ldots, C_{i-1}, \mathcal{P}) &= \text{softmax}(u^i) \end{aligned}$$

where softmax normalizes the vector u^i (of length n) to be an output distribution over the dictionary of inputs, and v, W_1, and W_2 are learnable parameters of the output model. Here, we do not blend