Attention Mechanism

Sunkyung Park

Contents

- **Neural Machine Translation by Jointly Learning to Align and Translate**

  - Existing researches
    - **Encoder-decoders in NMT**
      - **Encoder** reads and encodes a source sentence into **a fixed-length vector**
      - **Decoder** outputs a translation from the encoded vector.

  - Limitation of existing research
    - It requires a neural network to be able to **compress** all the information of a source sentence into a fixed-length vector.
    - It makes it **difficult** for NN to cope with **long sentences**.

  - **Proposed model**
    - Introduce **an extension** to the encoder-decoder model which learns to align and translate jointly **to overcome the limitation**.

- **RNN Encoder-Decoder Framework**

  - **Encoder**

    - It reads **the input sequence** ($X = (x_1, x_2, \dots, x_{T_x})$) into **a vector *c***.

    - **Expression**

      - non-linear function (e.g. LSTM)

      $$\boxed{h_t} = \boxed{f}(x_t, h_{t-1})$$

      - $h_t \in R^n$
      - Hidden state at time t

      - non-linear function
      - $q(\{h_1, \dots, h_T\}) = h_T$

      $$\boxed{c} = \boxed{q}(\{h_1, \cdots, h_{T_x}\})$$

      - ***c* : context vector generated from hidden states**

- **RNN Encoder-Decoder Framework**

  - Decoder

    - It is trained to predict **the next word** $y_{t'}$ given **the context vector** $\boldsymbol{c}$ and all **the previously predicted words** $\{y_1, \ldots, y_{t'-1}\}$.

    - It defines a prob over the translation y by decomposing the joint prob into the ordered conditionals.

Decomposing

$$p(\mathbf{y}) = \prod_{t=1}^{T} p(y_t \mid \{y_1, \cdots, y_{t-1}\}, c)$$

$$p(y_1, c)\ p(y_2|\{y_1\}, c)\ p(y_3|\{y_1, y_2\}, c) \cdots \underbrace{p(y_t|\{y_1, y_2, \ldots, y_{t-1}\}, c)}$$

- where $\boldsymbol{y} = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{T_y})$

$$p(y_t \mid \{y_1, \cdots, y_{t-1}\}, c) = \boxed{g(y_{t-1}, s_t, c)}$$

- Each conditional prob modeled
- $\boldsymbol{c}$**: context vector**
- g : non-linear, multi-layered, function that outputs the prob of y_t
- s_t : hidden state of the RNN.

- **Proposed Model**

  - **Decoder**

$$s_i = f(s_{i-1}, y_{i-1}, c_i)$$

An RNN hidden state for time i

$$p(y_i | y_1, \ldots, y_{i-1}, \mathbf{x}) = g(y_{i-1}, \boxed{s_i}, \boxed{c_i}).$$

- Conditioned on **a distinct context vector c_i** for each target word y_i
- c_i depends on a sequence of $(h_1, \ldots, h_{T_x})$ to which an encoder maps the input sentence.
- h_i
  - Contains information about the **whole** input sequence
  - Be with **focus** on the parts surrounding the i^{th} word of the input sequence.

- **Proposed Model**

  - **Decoder**

    - How to compute context vector c_i?

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

- **Weighted sum of $\boldsymbol{h_1}, \dots, \boldsymbol{h_{T_x}}$**

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

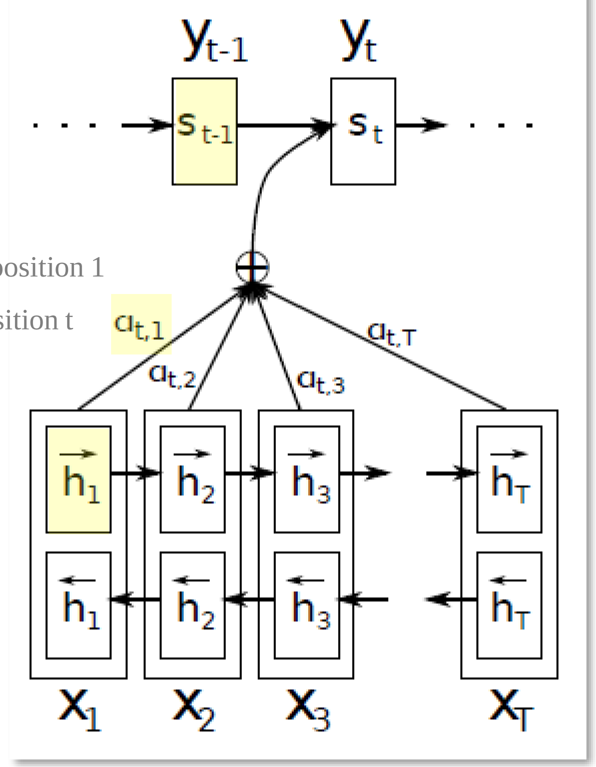
- **Alignment model**
  - Scores **how well** the inputs around position j and the output at position i **match**.

$$e_{ij} = a(\boxed{s_{i-1}}, \boxed{h_j})$$

  - Score is based on the $\boldsymbol{s_{i-1}}$(hidden state) and $\boldsymbol{h_j}$ of the input sequence.
  - a : parametrized feedforward neural network

the inputs around position 1

the output at position t



- **Proposed Model**

  - **Decoder**

    - **How to compute context vector c_i(intuitively)?**

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

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

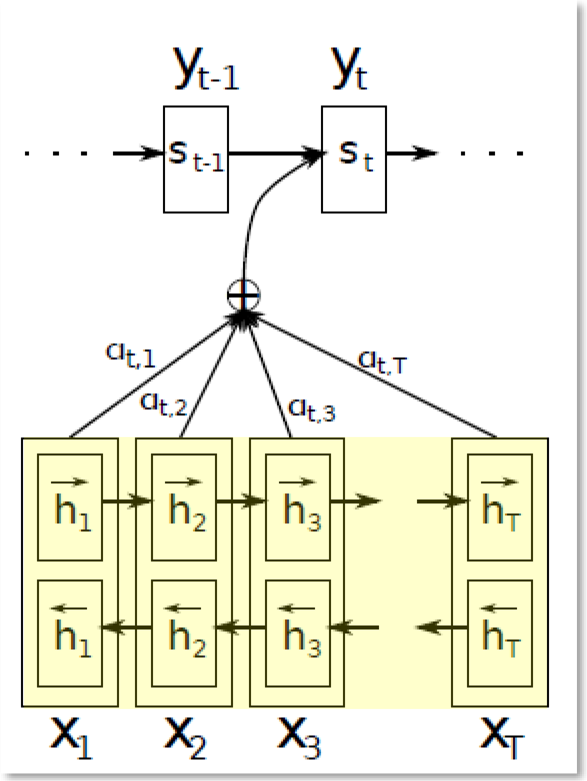
$$e_{ij} = a(\boxed{s_{i-1}}, \boxed{h_j})$$

    - **Prob(a_{ij}) , Energy(e_{ij}) : importance** of h_j w.r.t previous hidden state s_{i-1} in deciding the next state s_i and generating y_i

Attention Mechanism!

- **Relieve** encoder **from** encoding all information in the source sentence into **a fixed-length vector.**
- Information can be spread out the sequence and it can be selectively retrieved by decoder.

- **Proposed Model**
  - **Encoder**
    - **BiRNN(bidirectional RNN)**
      - Summarize not only the preceding words but also the following words.
      - Expression
        - $\vec{f}$ **(forward RNN) :** calculates a sequence of forward hidden states $(\vec{h}_1, \dots, \vec{h}_{T_x})$.
        - $\overleftarrow{f}$ **(backward RNN) :** result in a sequence of backward hidden states $(\overleftarrow{h}_1, \dots, \overleftarrow{h}_{T_x})$.
      - For each word x_j by concatenating $\vec{h}_j$ and $\overleftarrow{h}_j$
        - $h_j = [\,\vec{h}_j^T\,;\,\overleftarrow{h}_j^T\,]$ focuses on the words **around** x_j.



- **Experimental Settings**

Task	**English to French translation**
Dataset	**(Train) Bilingual, parallel corpora by ACL** [Association for computational linguistics] **WMT** [Workshop on Statistical Machine Translation] **'14** • European Parliament(61M words) • News commentary(5.5 M) • UN (421M) • two crawled corpora(90M, 272.5M) 850M words ⟫ Select 348M **(Validation) news-test-2012 + news-test-2013** **(Test) news-test-2014 from WMT '14 with 3003 sentences not present in the training data.**
Preprocessing	• Use a shortlist of 30,000 **most frequent words** in each language(en, fr) to train models. • Any word not included in the shortlist is mapped to **[UNK].**
Baseline	**RNN Encoder-Decoder**

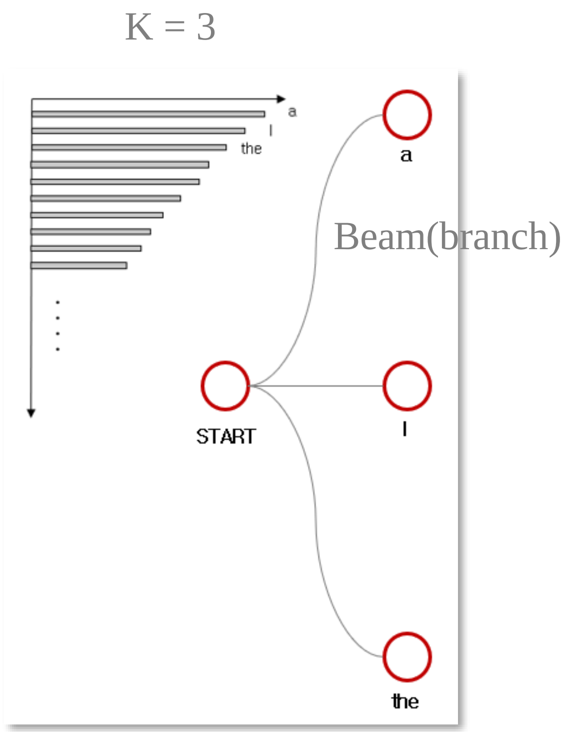
- **Experimental Settings**

<table>
<tr><th></th><th>Train 1
(w/ sentences of length up to 30 words)</th><th>Train 2
(w/ sentences of length up to 50 words)</th><th>Hyperparameters</th></tr>
<tr><td>RNN Encoder-Decoder</td><td>RNNencdec-30</td><td>RNNencdec-50</td><td rowspan="2">• (enc, dec) 1000 hidden units
• A single Maxout [Goodfellow et al., 2013] hidden layer to compute the conditional prob of each target word.
• Mini-batch(80 sentences)
• Stochastic gradient descent(SGD) + Adadelta
• Train for approximately 5 days
• Beam search to find a translation that maximizes the conditional prob.</td></tr>
<tr><td>RNNsearch(Proposed)</td><td>RNNsearch-30</td><td>RNNsearch-50</td></tr>
</table>

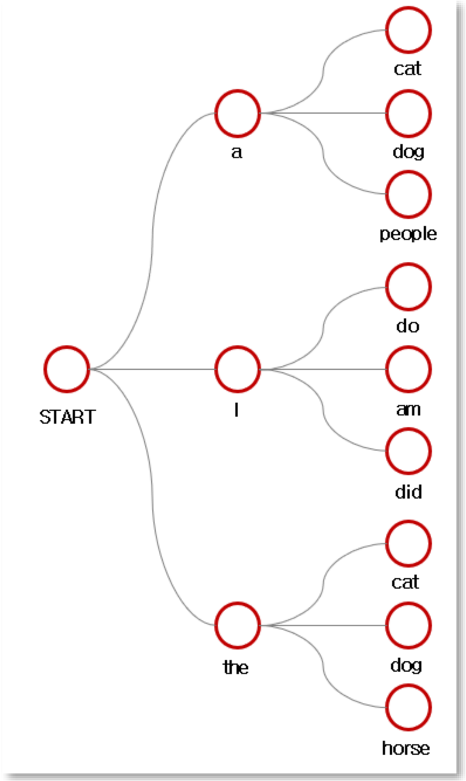
- **Beam Search**

  - (Background) What if only 1st survives where there is little difference between 1st and 2nd ?

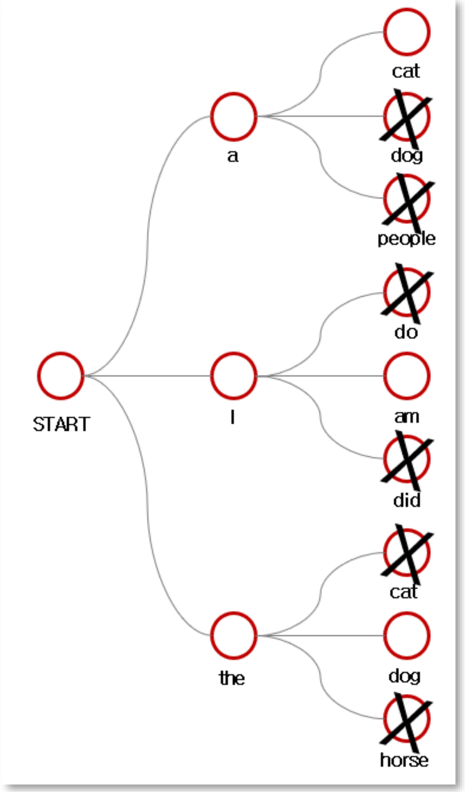
  - Method of **selecting** the num of **promised beams** (≤ **K**) at a given point in time.



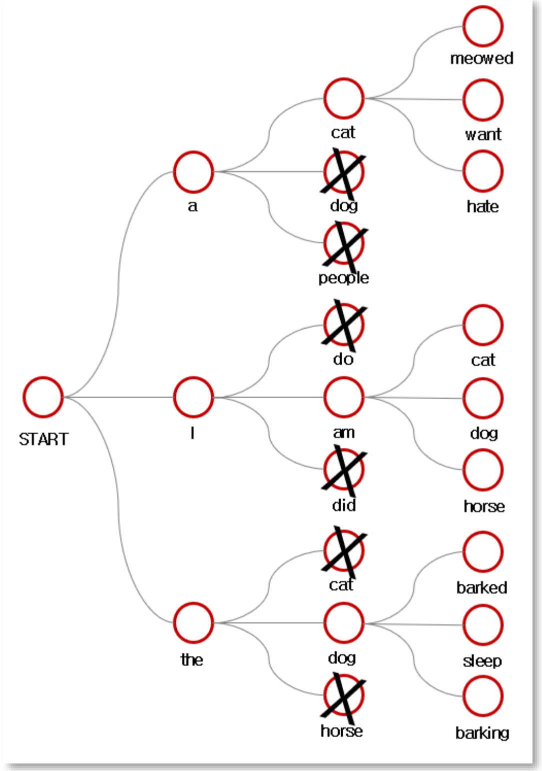
1. Select **K highest prob** among prob distribution of the predicted values.



2. Select **the K highest** of the next prediction value in **each of the K beams.**



3. Select **top K** in the order of **cumulative prob** among K^2 childe nodes.



4. **Set** the top K child nodes as **new beams** and **Create** the top K child nodes until the num of beams w/ <eos> becomes **K.**

- **Results : Quantitative Results**

Scores on all sentences

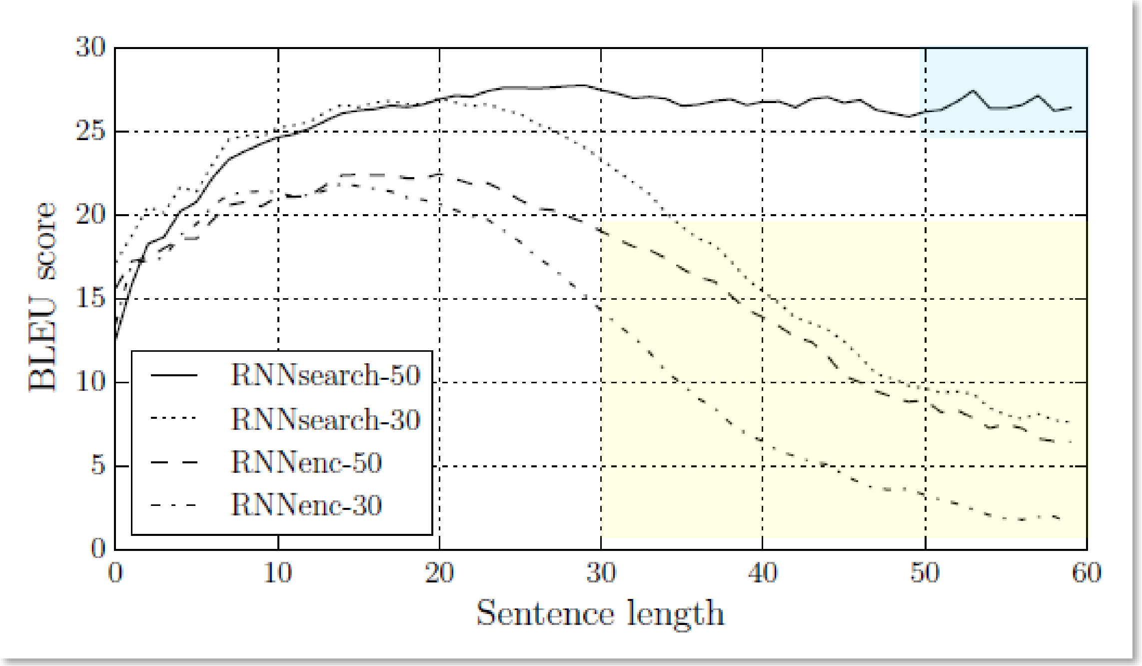
Model	All	No UNK°
RNNencdec-30	13.93	24.19
RNNsearch-30	21.50	31.44
RNNencdec-50	17.82	26.71
RNNsearch-50	26.75	34.16
RNNsearch-50*	28.45	36.15
Moses	33.30	35.63

Trained much longer **until** the performance on validation set stopped improving

Scores on the sentences without any unknown word

- **BLEU Bilingual Evaluation Understudy Score on test set**
- **Proposed RNNsearch > RNNencdec**
- **( ) Better performance w/ fewer resources!**
  - RNNsearch > Moses
  - Moses uses monolingual corpus(418M) + parallel corpora in RNNsearch.

- **Results : Quantitative Results**
  - **Proposed RNNsearch > RNNencdec**
    - RNNsearch does not require encoding a long sentence into a fixed-length vector perfectly.
    - RNNsearch encodes the parts of the input sequence surrounding a particular word.



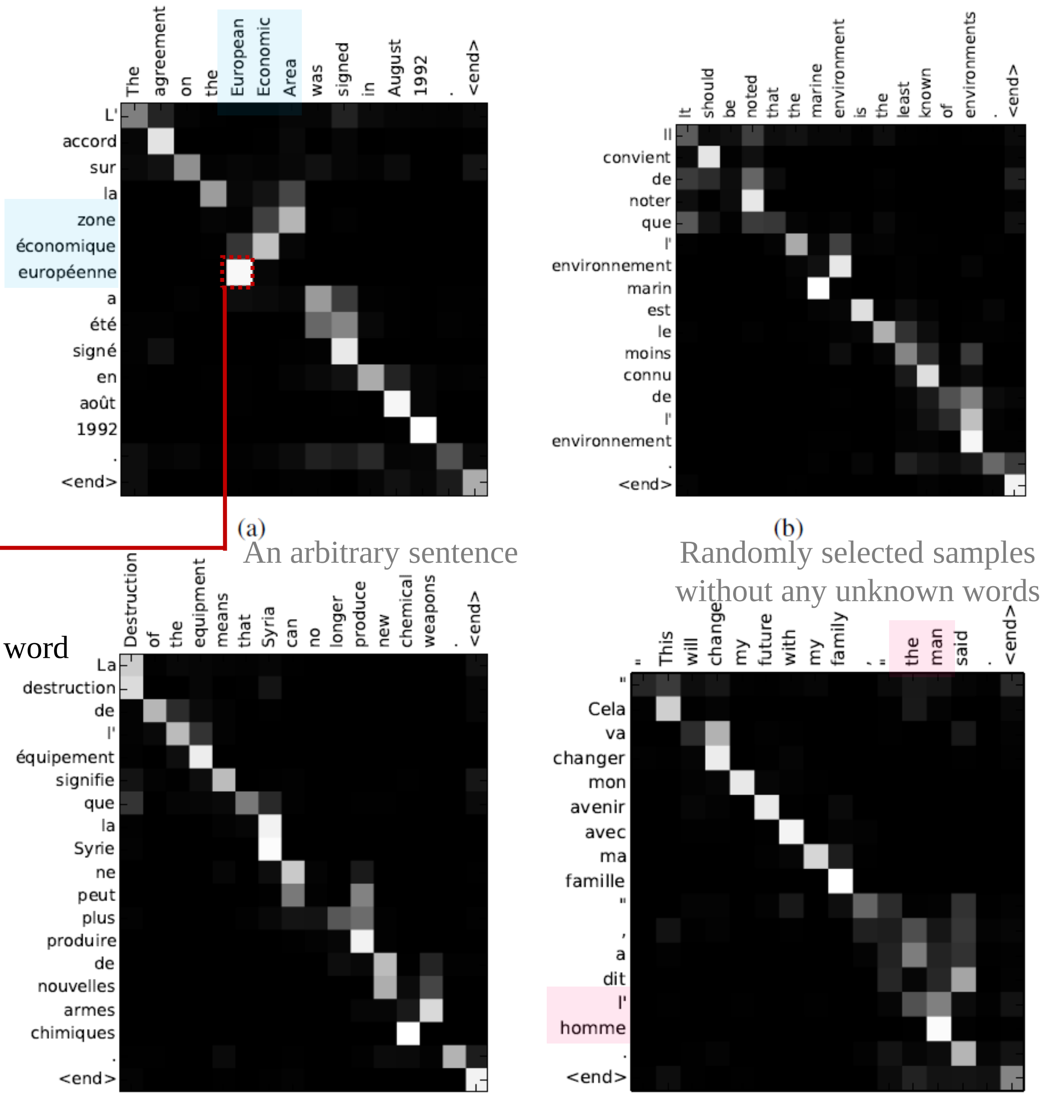
- (●) Enc-dec **drops dramatically** as the length of sentences increases.
- RNNsearch-30, RNNsearch-50 are **more robust** on value of x-axis.
- (●) RNNsearch-50 no performance deterioration w/ **length ≥ 50**

- **Results : Qualitative Results**

Words in source sentence (English)

Generated Translation (French)

Weight a_{ij} of the j^{th} source word for i^{th} target word.
(0 : black, 1 : white)



(a) An arbitrary sentence

(b) Randomly selected samples without any unknown words

(c) Same as (b)

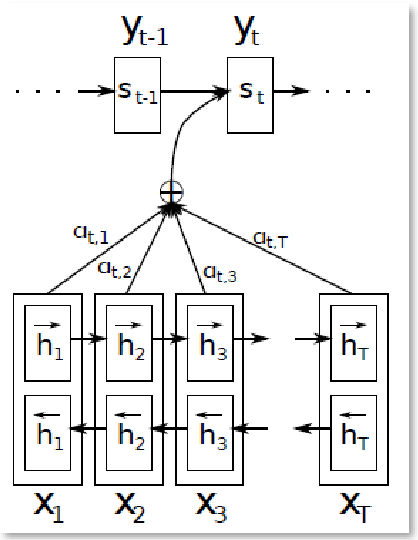
(d) Same as (b)

- **Soft-alignment (≠ hard-alignment(1:1))**
  - **(Fig)** specific positions in the source sentence considered more important in generating the target word.
  - **(Fig) [the man] : [l'homme]**

    Following word ➔ le, la, les, l'
- **Understanding of alignment between English and French**
  - **Monotonic**
    - **(Fig)** Strong weights along the diagonal of matrix.
  - **Non-monotonic: Adjectives and nouns differently ordered**
    - **(Fig)** RNNsearch correctly aligns [zone] with [Area]

- **Hypothesis Verification Structure**

RNNsearch



Quantitative result

Qualitative result

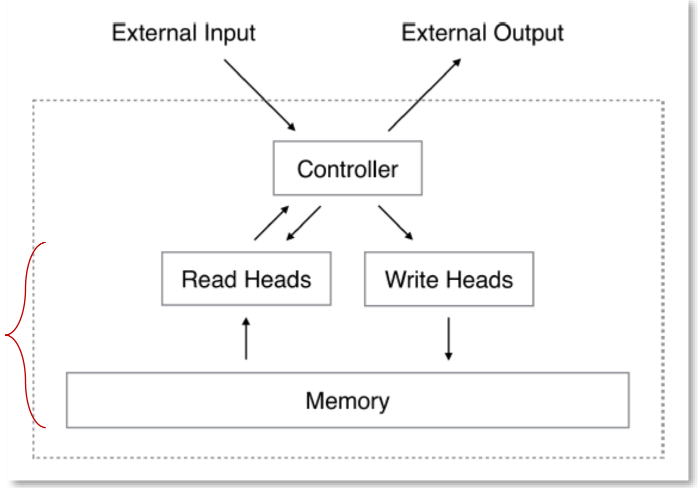
Hypothesis

The proposed model, the RNNsearch enables far more reliable translation of long sentences than the RNNencdec.

- **Augmented RNNs : Neural Turing Machines**

  - **What is Turing Machine?**
    - Virtual Machine, the foundation of **modern computer architecture**.
    - Functions of **numerical operation, memory reading, and memory writing**.
    - Modern computer architecture consists two parts:
      - **CPU** that performs the operation
      - **Memory** that stores the values
  - **Neural Turing Machine**
    - Standard Neural networks have no memory.
    - **Architecture**
      - **Neural Network**
      - **Memory Matrix**
        - **External Memory**(explicit memory structure) outside of neural networks.

Different from Standard Neural Network



Architecture of Neural Turing Machine

- **Neural Turing Machines**

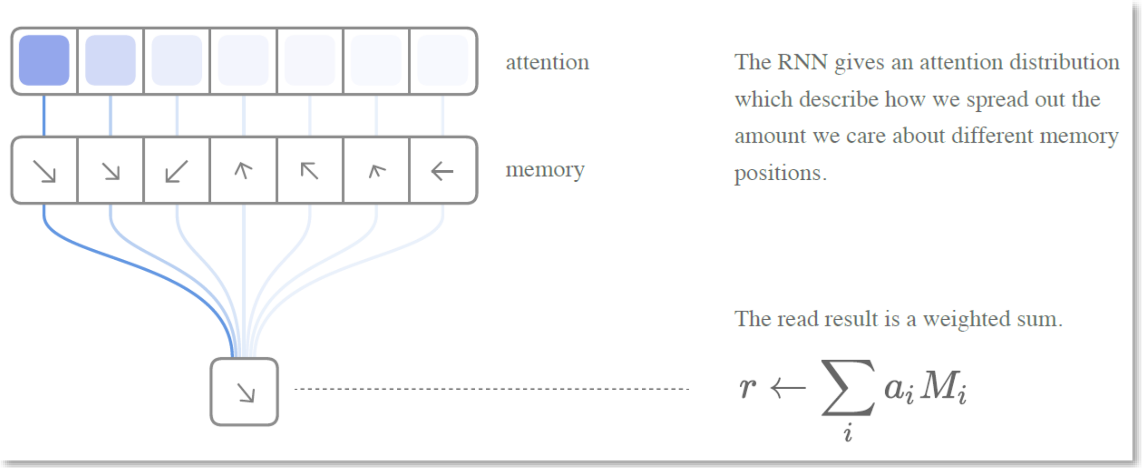
  - How Does Reading and Writing Work?

    - **Attention Distribution**

      - How we spread out the amount we care about **different memory positions.**

      - The result of the read operation is a **weighted sum.**

    - **Reading**



At time step $\boldsymbol{t}$...

- $\boldsymbol{a_i}$: Normalized weight vector indicating **the importance of each location**.

  (**Attention**)

- $\boldsymbol{M_i}$: The value of **memory matrix** corresponding to each location.

- $\boldsymbol{r}$: The actual value we read from memory (**read vector**).

- **Neural Turing Machines**

  - How Does Reading and Writing Work?

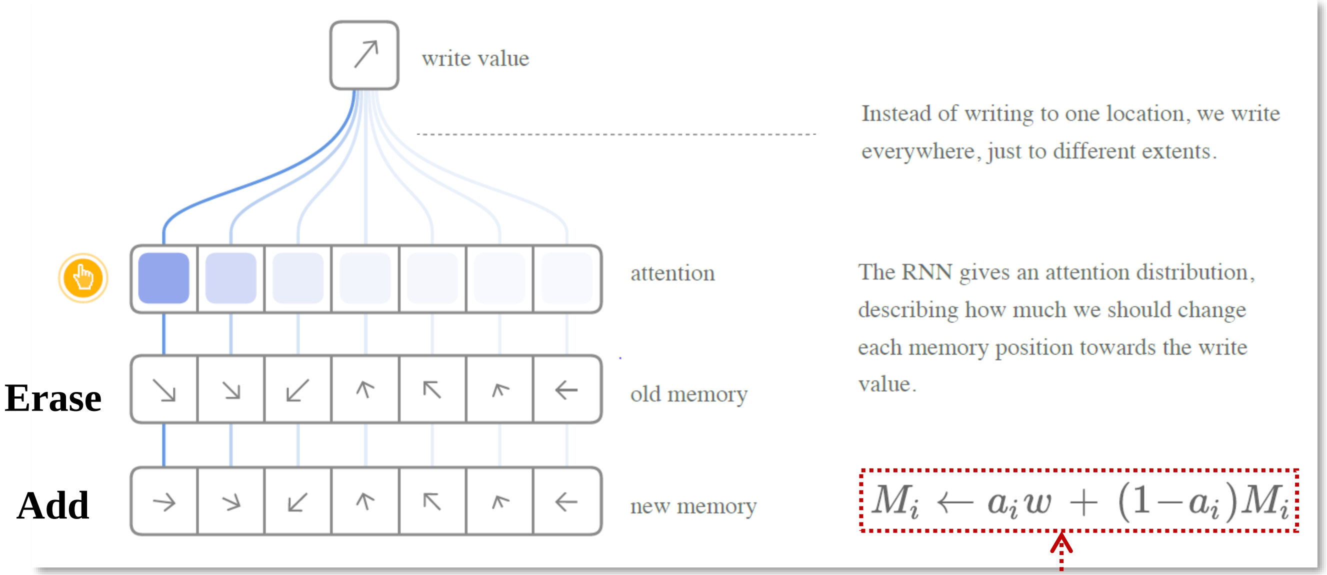
    - **Writing**

      - Erase Operation
        - $\boldsymbol{\widetilde{M}_t(i) \leftarrow M_{t-1}(i)[\,1 - w_t(i)e_t\,]}$
        - e_t : Erase vector
        - w_t : How much to erase

      - Add Operation
        - $\boldsymbol{M_t(i) \leftarrow \widetilde{M}_t(i) + w_t(i)a_t}$
        - $\boldsymbol{a_t}$: Add vector
        - $\boldsymbol{w_t}$: How much to add

Erase

Add



Simplified

- **Neural Turing Machines**

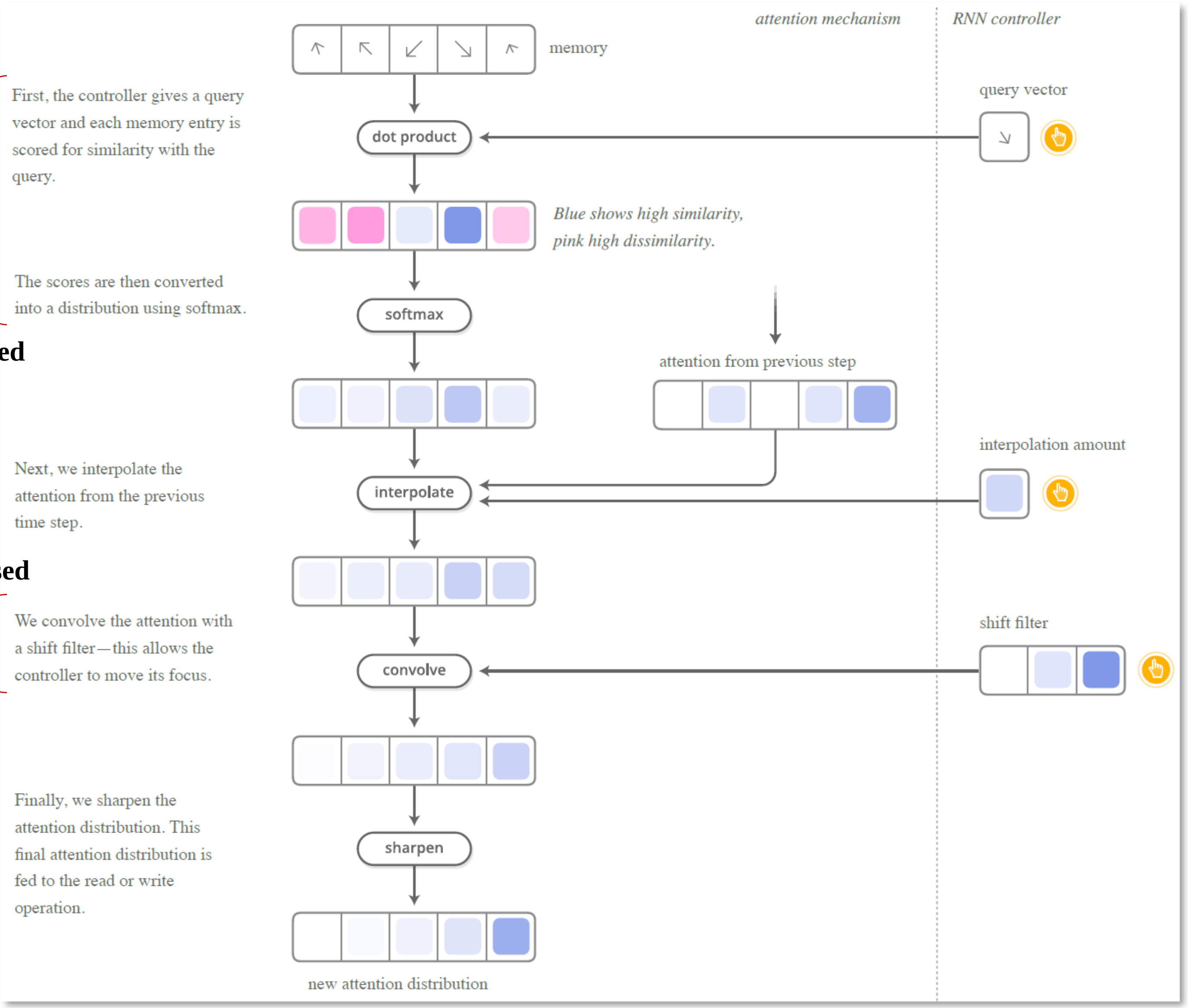
  - How do NTMs decide **which positions in memory** to focus their **attention** on?

  (= **How to compute weight vector w_t?**)

  - A combination of two different methods
    - Content-based attention
      - **Search through** their **memory**
      - **Focus on places** that match they're looking for.
    - Location-based attention
      - **Relative movement** in memory, enabling the NTM to **loop**.
      - **Weight** obtained using content similarity is now **modified based on location**.

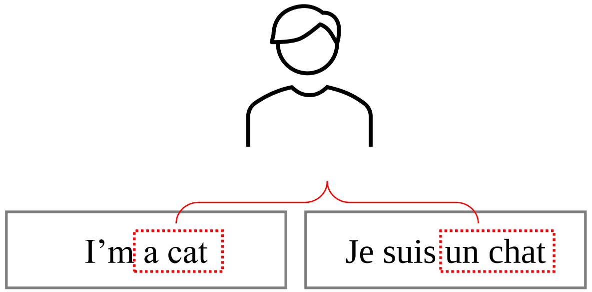
Content-based

Location-based



- **Augmented RNNs : Attentional Interfaces**

Human

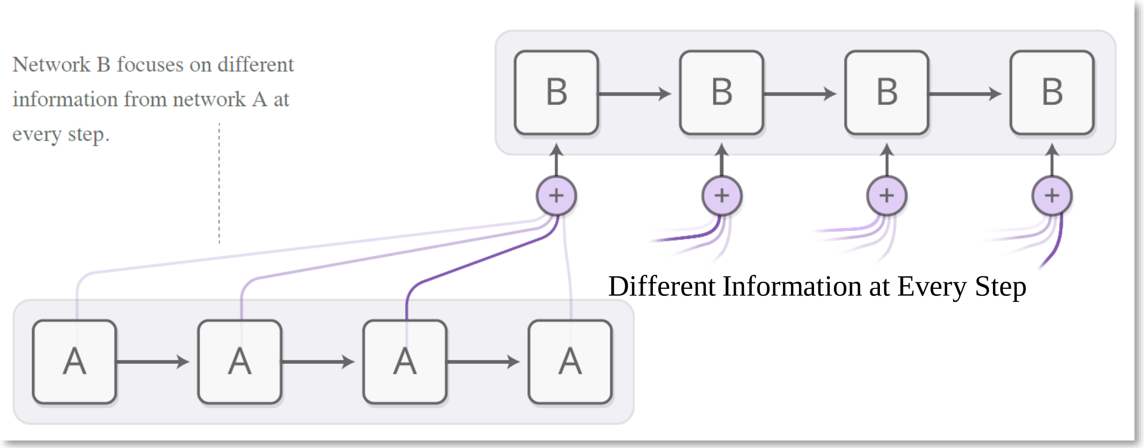


Pay **attention** to the word we are presently translating

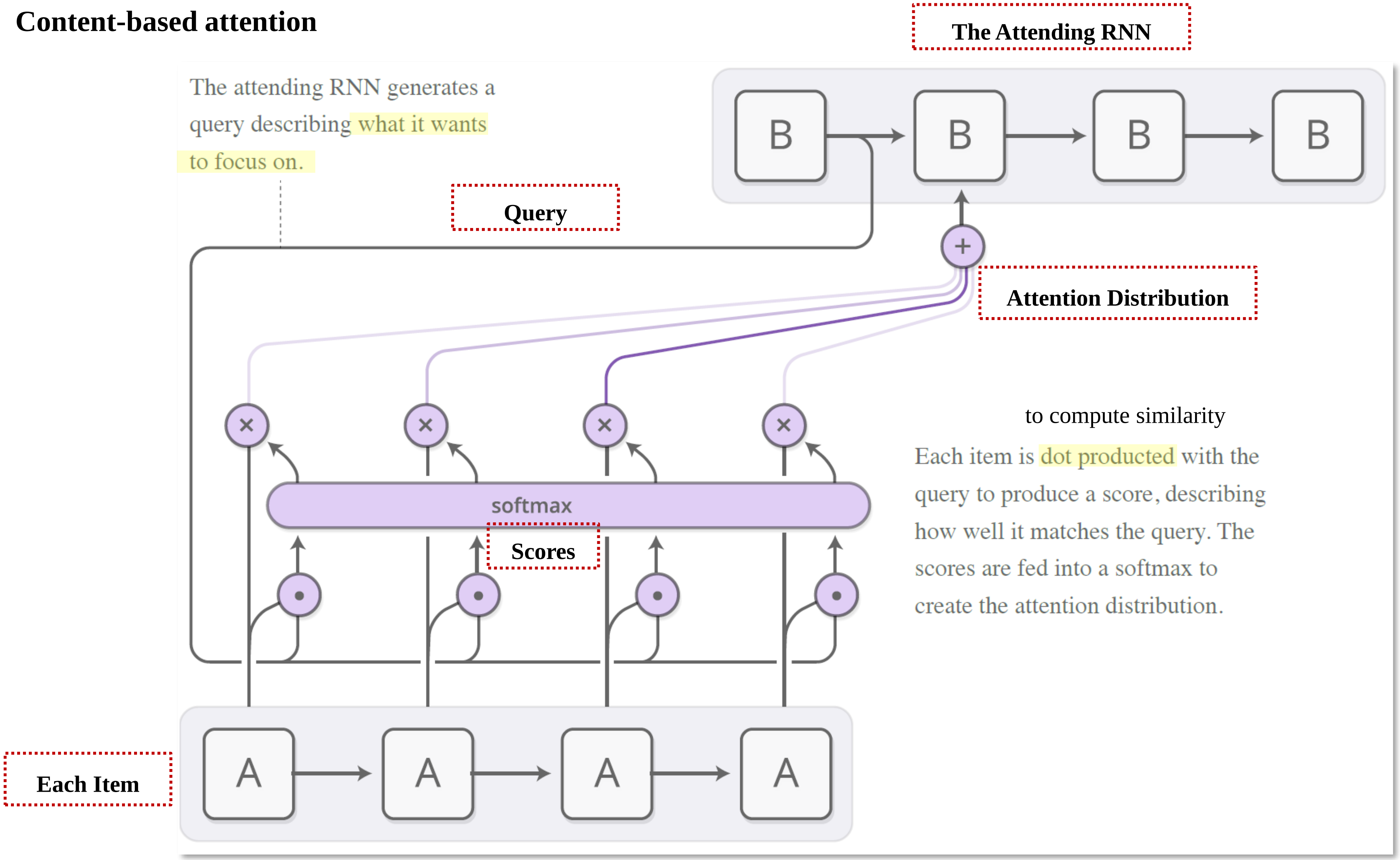
=

Attention in Neural Net

- Make attention **differentiable** ➔ **Learn** where to focus
- **Focus everywhere, just to different extents.**
- Attention distribution is generated with **content-based attention.**



- **Attentional Interfaces**
  - **Content-based attention**



- **Use of Attentional Interfaces : Translation**

  - **Seq-to-Seq Models**

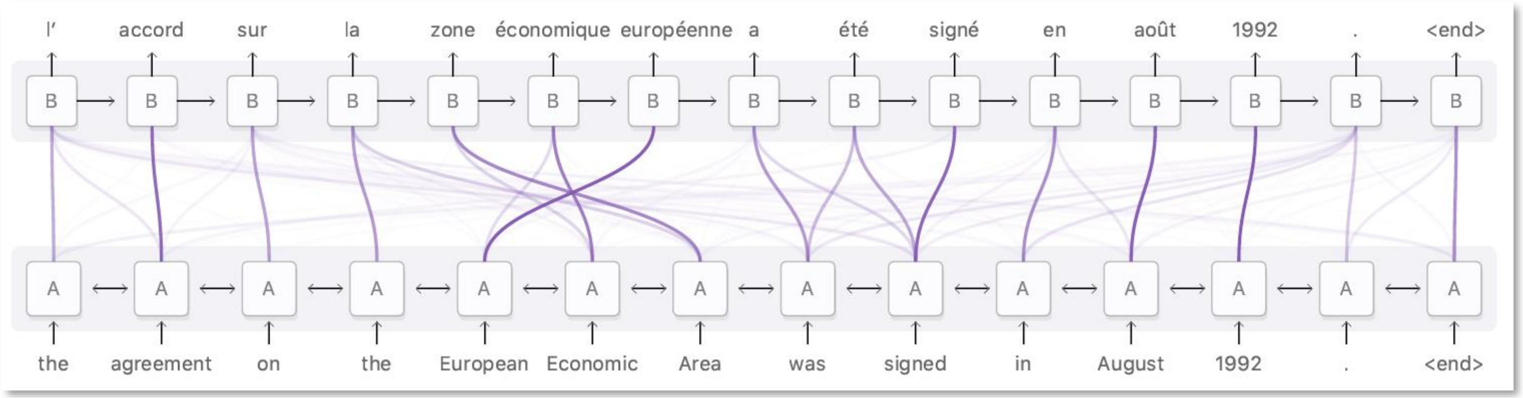
    - Boil **the entire input** down into **a single vector**

    - **Expand** it **back out**

      ➔ As for long sentences, all necessary information **cannot be included** in the vector.

  - **Attention** (same as 5 page)

    - **Avoids** seq2seq’s unreasonable mechanism



Decoder focuses on words relevant

Encoder passes along information about each word

- **Use of Attentional Interfaces : Image Captioning**

    - Attention can be used on the interface **between a CNN and an RNN.**

        - **CNN** processes the image, **extracting** high-level features
        - **RNN focuses** on the CNN’s interpretation of relevant parts and generalizes a description of the image.

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.

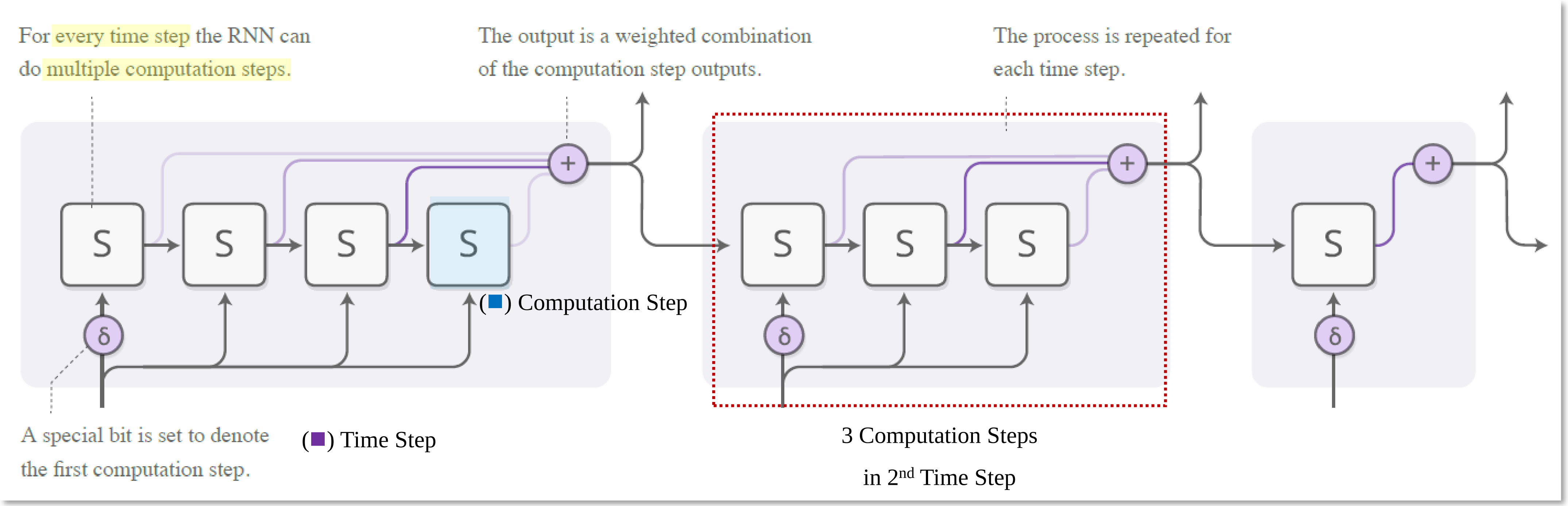
CNN captures noticeable features (frisbee in red etc.)

RNN focuses on that interpretation

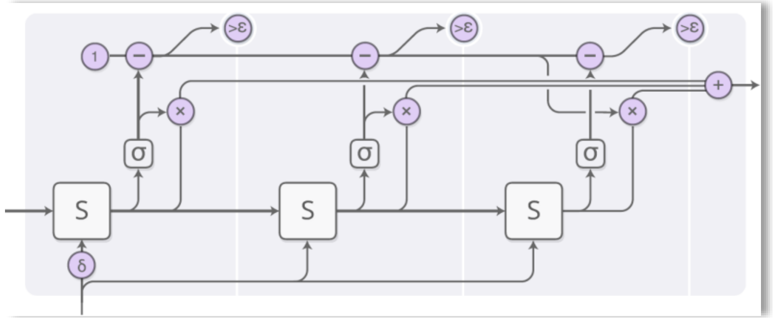
- **Augmented RNNs : Adaptive Computation Time**
  - **Standard RNNs**
    - Do **the same amount of computation** for each time step
    - $\boldsymbol{O}(\boldsymbol{n})$ operations for n length list.
  - **Adaptive Computation Time:**
    - **(Goal)** A way for RNNs to do **different** amounts of computation each step
    - **(How)** Allow the RNN to do **multiple steps** of computation for each time step

Learn How many steps to do ››› Num of Steps to be **Differentiable** ››› **Attention Distribution** over Num of Steps

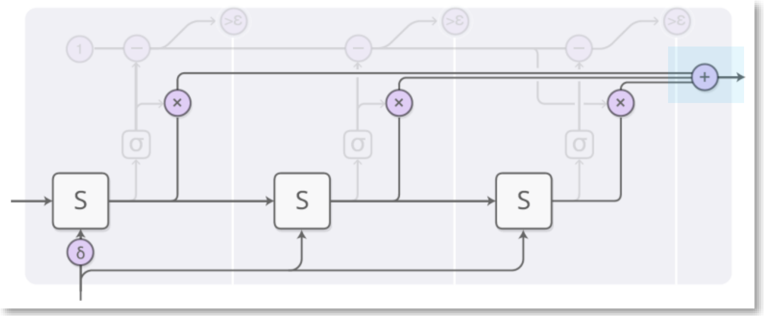
- **Adaptive Computation Time**
  - **Different amounts of computation each step**



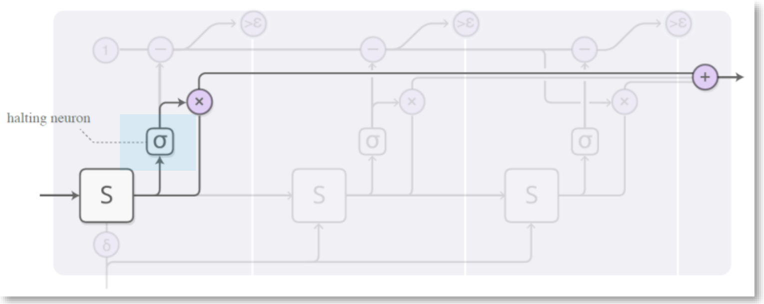
- **Adaptive Computation Time**



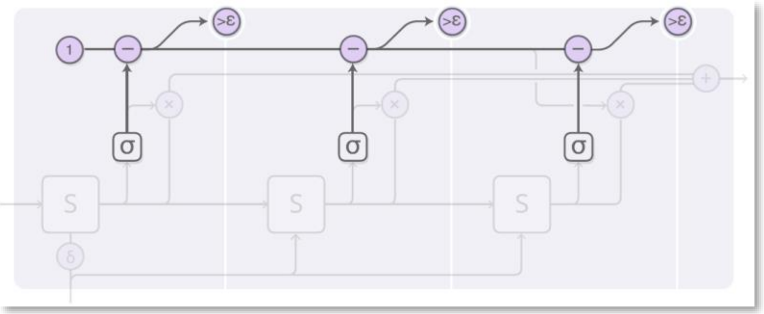
Step by Step



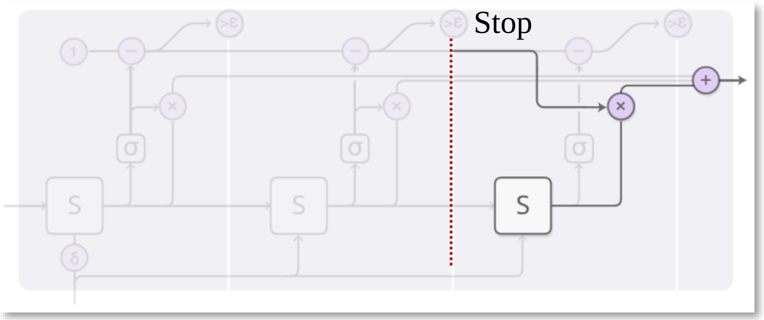
- From RNNs point of view, Output a weighted combination of the states



- The weight for each step determined by a **"halting neuron"**.
- Sigmoid neuron to look at the RNN state
- halting weight $= p\left(Stop_{at\ that\ step}\right)$



- Total budget for the halting weights of 1
- If budget < epsilon(ε), then stop.
- Track that budget.

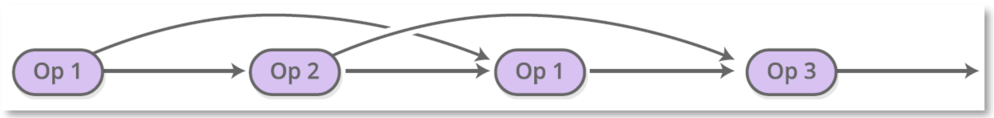


- Some left-over budget belongs to the last step.
- **"Ponder Cost"** term in Cost Function
  - Penalizes the model for the amount of computation
- Ponder Cost ↑ ➔ Computation Time ↓, Performance ↓

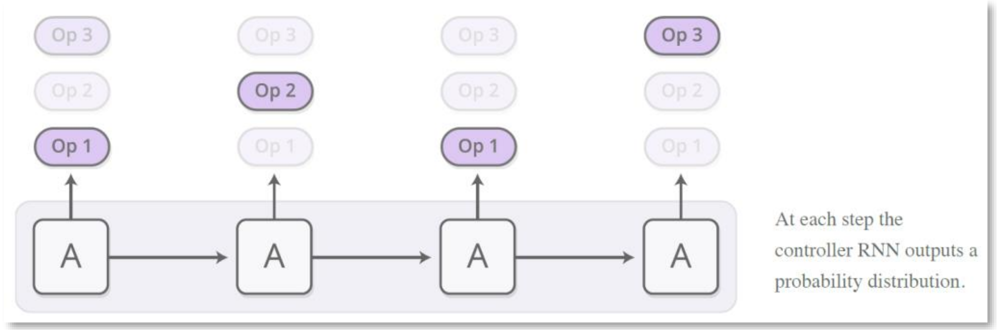
- **Augmented RNNs : Neural Programmer**

  - **Neural Net**
    - Struggle to do some basic things like ‘arithmetic’.
    - Neural net + normal programming ➔ Get the best of both worlds
  - **Neural Programmer:**
    - Learns to create programs in order to solve a task.
    - No need for examples of correct programs.
    - Discover how to produce programs to accomplish some task.
    - Like Unix-Pipe ➔ Why?
      - A mechanism for inter-process communication using message passing.

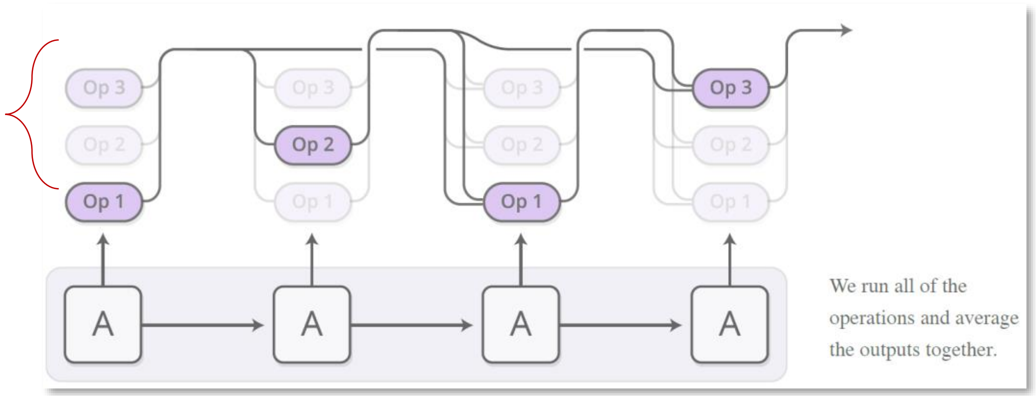
- **Neural Programmer**



- The generated program is **a sequence of operations.**
- **Operation :**
  - $Output_{op\ 2\ steps\ ago} + Output_{op\ 1\ step\ ago}$ ➔ Unix-pipe



- The program is generated one op at a time by a controller RNN.
- Prob distribution for **what the next operation should be.**



- Run **all of ops** and **Average** the outputs together.
- Outputs **weighted by prob** of running that op.
- The Program's output is **differentiable** w.r.t. the prob.
- Define **a Loss**.
- **Train** a neural net to produce programs to give the correct answer.

- **Neural Programmer**
  - **A Few Additional Things**
    - **Multiple Types**
      - Some operations outputs **selections of table columns, selection of cells.**
      - Note that only outputs of **the same type** get merged.
    - **Referencing Inputs**
      - Given a table of cities with a population column, Answer…
        - “ How many cities have **a population** greater than **1,000,000**? ”
      - Some op allow the network to reference **constants** or **the names of columns** by **Attention**

References

- Lecture Material
  - **Neural Machine Translation by Jointly Learning to Align and Translate**
  - Blog Post Overview (**Augmented RNNs**)
- Neural Turing Machines
  - http://solarisailab.com/archives/2162
- Beam Search
  - https://blog.naver.com/sooftware/221809101199