Attention Model in NLP

Jun Chen

4/20/2019

Outline

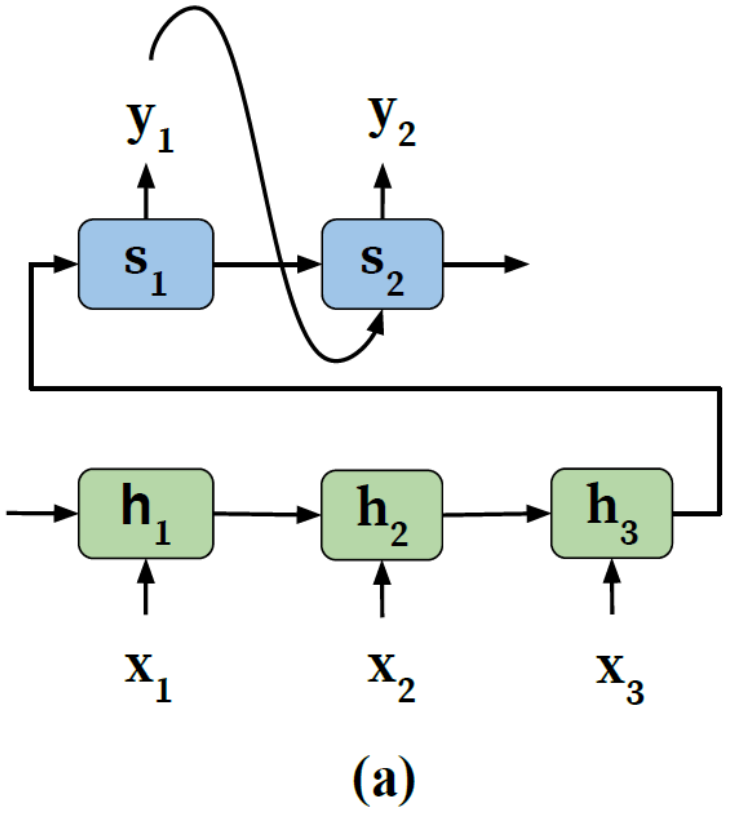
- Motivation
- Attention Model
  - RNNsearch example
  - A unified attention model
- Taxonomy of Attention
  - Number of sequences
  - Number of abstraction levels
  - Number of positions

The slides is based on two survey papers of attention models in NLP:

- Chaudhari, S. , Polatkan, G. , Ramanath, R. , & Mithal, V. . (2019). An attentive survey of attention models.
- Galassi, A. , Lippi, M. , Torroni, P. . (2019). Attention, please! A Critical Review of Neural Attention Models in Natural Language Processing

Motivation

- Sequence-to-sequence model



Encoder: takes an input sequence of tokens $\{x_1, x_2, \ldots, x_T\}$ where T is the length of input sequence, and encodes it into fixed length vectors $\{h_1, h_2, \ldots, h_T\}$.

Decoder: takes a single fixed length vector h_T as its input and generates an output sequence $\{y_1, y_2, \ldots, y_{T'}\}$ token by token, where T' is the length of output sequence.

h_t and s_t denote the hidden states of the encoder and decoder respectively at each position t

Motivation

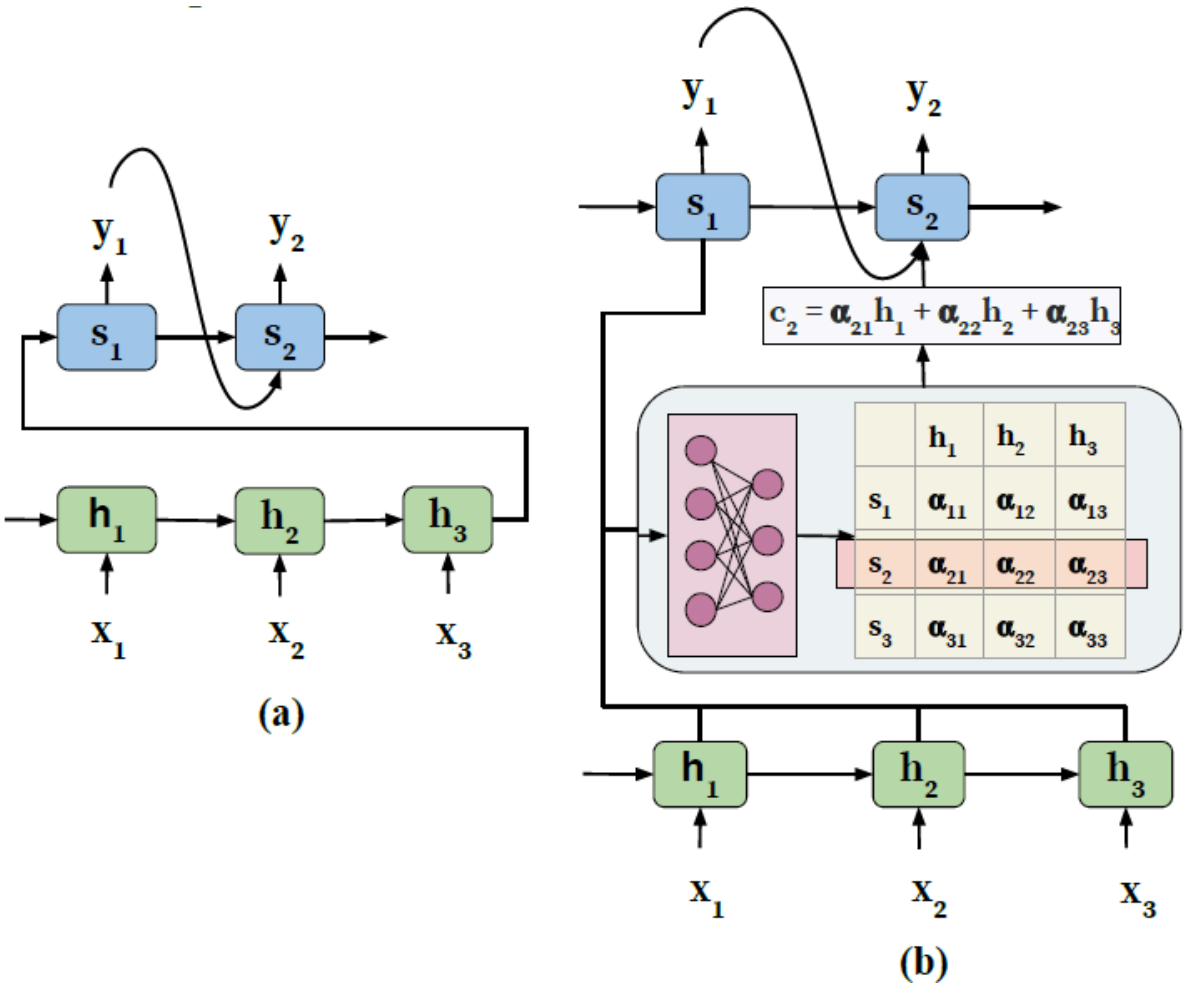
Challenges of traditional encoder-decoder:

- encoder compresses input sequence into a single fixed length vector h_T (loss of information)
- unable to model alignment between input and output sequences, which is an essential aspect of structured output tasks, like translation or summarization

Attention Model

- Key idea
  - Allow the decoder to access the entire encoded input sequence $\{h_1, h_2, \ldots, h_T\}$.
  - Induce attention weights over the input sequence to prioritize the set of positions where relevant information is present for generating the next output token.

Attention Model



Encoder-decoder architecture: (a) traditional (b) with attention

Usage of attention

- The attention block in the architecture is responsible for automatically learning the attention weights α_{ij}, which capture the relevance between h_i (candidate state) and s_j (query state).
- These attention weights are then used for building a context vector c, which is passed as an input to the decoder. At each decoding position j, the context vector c_j is a weighted sum of all hidden states of the encoder and their corresponding attention weights, i.e. $c_j = \sum_{i=1}^{T} \alpha_{ij} h_i$
- This additional context vector is the mechanism by which decoder can access the entire input sequence and also focus on the relevant positions in the input sequence.

Learning attention weights

- The attention weights are learned by incorporating an additional feed forward neural network within the architecture.
- The feed forward network learns a particular attention weight α_{ij} as a function of h_i and s_{j-1} which are taken as input by the neural network.
- It is jointly trained with encoder-decoder components of the architecture.

An Example for NMT

Objective: compute an output sequence y that is a translation of an input sequence x

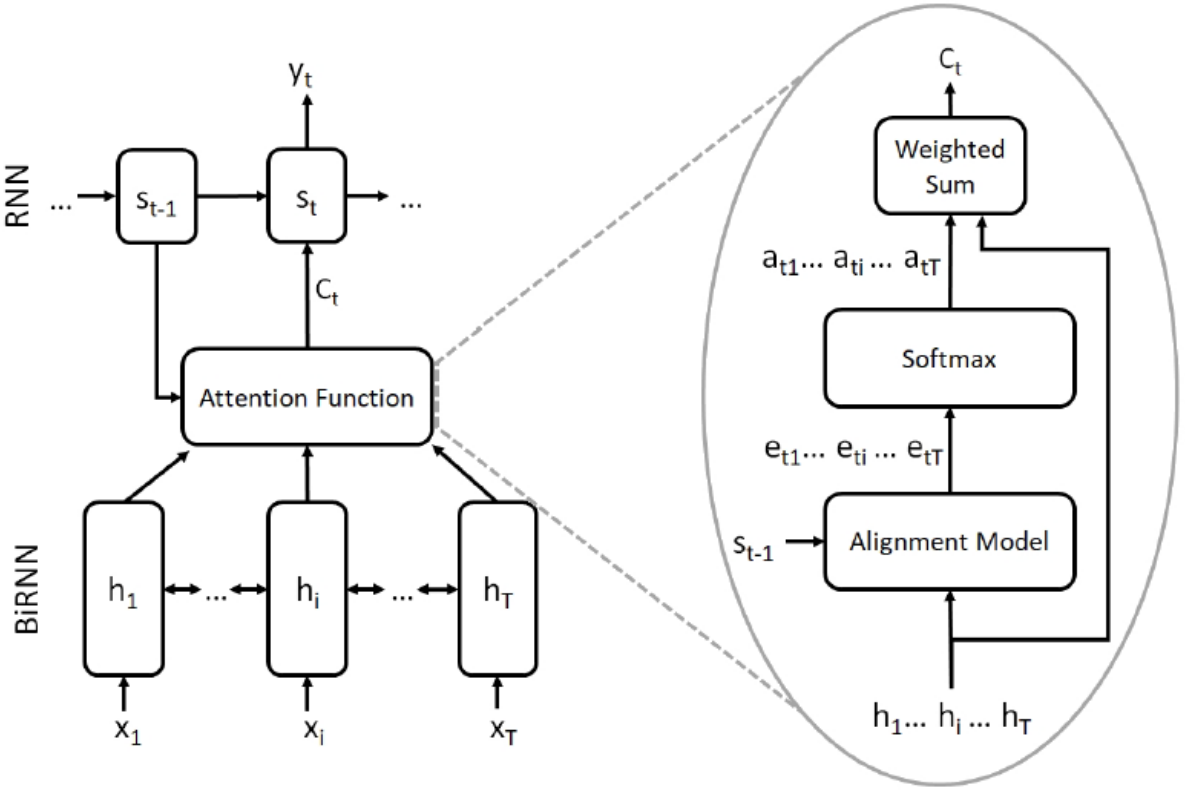


Figure 2: Architecture of RNNsearch (Bahdanau et al., 2015) (left) and its attention model (right).

RNNsearch (Bahdanau et al., 2015)

Encoder:

$$(\boldsymbol{h_1}, \ldots, \boldsymbol{h_T}) = \text{BiRNN}(x_1, \ldots, x_T) \quad (1)$$

Decoder:

$$p(y_t|y_1, ..., y_{t-1}, \boldsymbol{x}) = \text{RNN}(\boldsymbol{c_t}) \quad (2)$$

The context vector c is obtained as follows:

$$e_{ti} = f(\boldsymbol{s_{t-1}}, \boldsymbol{h_i}) \quad \text{matching score} \quad (3)$$

$$a_{ti} = \frac{\exp(e_{ti})}{\sum_{j=1}^{T} \exp(e_{tj})} \quad \text{normalize} \quad (4)$$

$$\boldsymbol{c_t} = \sum_i a_{ti} \boldsymbol{h_i} \quad \text{weighted sum} \quad (5)$$

A Unified Model

Symbol	Name	Definition
x	Input sequence	Sequence of textual elements constituting the raw input.
$\boldsymbol{K}$	Keys	Matrix of d_k vectors ($\boldsymbol{k}_i$) of size n_k, whereupon attention weights are computed: $\boldsymbol{K} \in \mathbb{R}^{n_k \times d_k}$.
$\boldsymbol{V}$	Values	Matrix of d_k vectors ($\boldsymbol{v}_i$) of size n_v, whereupon attention is applied: $\boldsymbol{V} \in \mathbb{R}^{n_v \times d_k}$. Each $\boldsymbol{v}_i$ and its corresponding $\boldsymbol{k}_i$ offer two, possibly different, interpretations of the same entity.
$\boldsymbol{q}$	Query	Vector of size n_q, or sequence thereof, in which respect attention is computed: $\boldsymbol{q} \in \mathbb{R}^{n_q}$.
kaf qaf vaf	Annotation functions	Functions that encode the input sequence and query, producing $\boldsymbol{K}$, $\boldsymbol{q}$ and $\boldsymbol{V}$ respectively.
$\boldsymbol{e}$	Energy scores	Vector of size d_k, whose scalar elements (energy "scores", e_i) represent the relevance of the corresponding $\boldsymbol{k}_i$, according to the compatibility function: $\boldsymbol{e} \in \mathbb{R}^{d_k}$.
$\boldsymbol{a}$	Attention weights	Vector of size d_k, whose scalar elements (attention "weights", a_i) represent the relevance of the corresponding $\boldsymbol{k}_i$ according to the attention model: $\boldsymbol{a} \in \mathbb{R}^{d_k}$.
f	Compatibility function	Function that evaluates the relevance of $\boldsymbol{K}$ with respect to $\boldsymbol{q}$, returning a vector of energy scores: $\boldsymbol{e} = f(\boldsymbol{K}, \boldsymbol{q})$.
g	Distribution function	Function that computes the attention weights from the energy scores: $\boldsymbol{a} = g(\boldsymbol{e})$.
$\boldsymbol{Z}$	Weighted values	Matrix of d_k vectors ($\boldsymbol{z}_i$) of size n_v, representing the application of $\boldsymbol{a}$ to $\boldsymbol{V}$: $\boldsymbol{Z} \in \mathbb{R}^{n_v \times d_k}$.
$\boldsymbol{c}$	Context vector	Vector of size n_v, offering a compact representation of $\boldsymbol{Z}$: $\boldsymbol{c} \in \mathbb{R}^{n_v}$.

A Unified Attention Model

- Core attention model

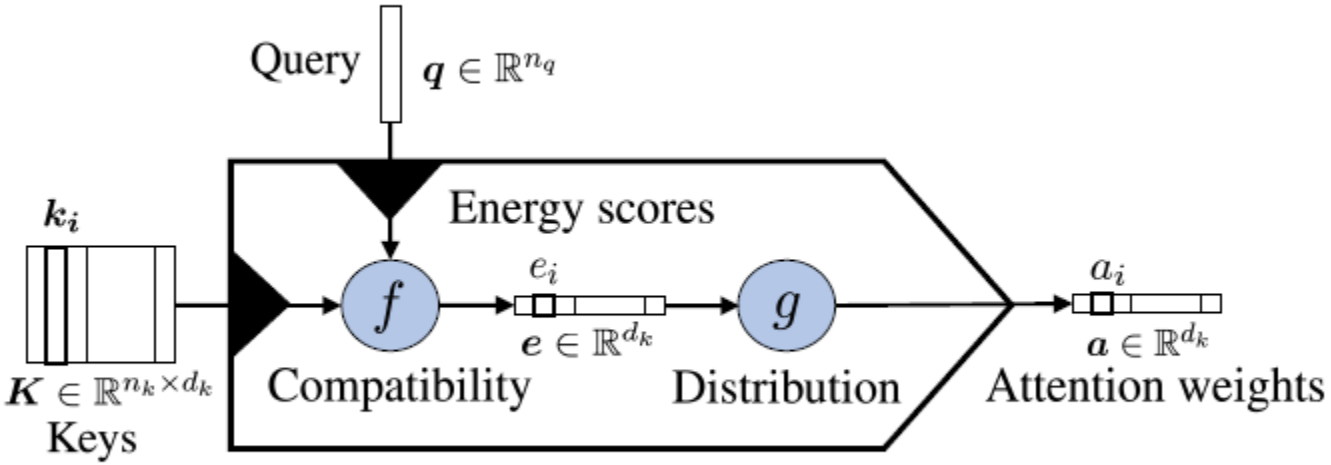


Figure 3: Core attention model.

Core Attention

- The core of the attention mechanism maps a sequence K of d_k vectors k_i, the **keys**, to a distribution a of d_k weights a_i. (h_i in RNNsearch)

- Another input element q, the **query**, is used as a reference when computing the attention distribution. The attention mechanism will give emphasis to the input elements considered to be relevant to the task according to the query.(s_{j-1} in RNNsearch)

Core Attention

- From the keys and query, a vector e of d_k energy scores e_i is computed through a compatibility function f (alignment model in RNNsearch).

$$\boldsymbol{e} = f(\boldsymbol{q}, \boldsymbol{K})$$

- Energy scores are then transformed into attention weights using a distribution function, g:

$$\boldsymbol{a} = g(\boldsymbol{e})$$

Compatibility Functions

Table 3: Summary of compatibility functions found in literature. $\boldsymbol{W}$, $\boldsymbol{W_0}$, $\boldsymbol{W_1}$, ..., and $\boldsymbol{b}$ are learnable parameters.

Name	Equation	Reference
similarity	$f(\boldsymbol{q}, \boldsymbol{K}) = sim(\boldsymbol{q}, \boldsymbol{K})$	Graves et al., 2014
multiplicative or *dot*	$f(\boldsymbol{q}, \boldsymbol{K}) = \boldsymbol{q}^\intercal \boldsymbol{K}$	Luong et al., 2015
scaled multiplicative	$f(\boldsymbol{q}, \boldsymbol{K}) = \frac{\boldsymbol{q}^\intercal \boldsymbol{K}}{\sqrt{d_k}}$	Vaswani et al., 2017
general or *bilinear*	$f(\boldsymbol{q}, \boldsymbol{K}) = \boldsymbol{q}^\intercal \boldsymbol{W} \boldsymbol{K}$	Luong et al., 2015
biased general	$f(\boldsymbol{q}, \boldsymbol{K}) = \boldsymbol{K}^\intercal (\boldsymbol{W}\boldsymbol{q} + \boldsymbol{b})$	Sordoni et al., 2016
activated general	$f(\boldsymbol{q}, \boldsymbol{K}) = act(\boldsymbol{q}^\intercal \boldsymbol{W} \boldsymbol{K} + \boldsymbol{b})$	Ma et al., 2017
concat	$f(\boldsymbol{q}, \boldsymbol{K}) = \boldsymbol{w_{imp}}^\intercal act\big(\boldsymbol{W}[\boldsymbol{K}; \boldsymbol{q}] + \boldsymbol{b}\big)$	Luong et al., 2015
additive	$f(\boldsymbol{q}, \boldsymbol{K}) = \boldsymbol{w_{imp}}^\intercal act(\boldsymbol{W_1}\boldsymbol{K} + \boldsymbol{W_2}\boldsymbol{q} + \boldsymbol{b})$	Bahdanau et al., 2015
deep	$f(\boldsymbol{q}, \boldsymbol{K}) = \boldsymbol{w_{imp}}^\intercal \boldsymbol{E}^{(L-1)} + \boldsymbol{b}^L$ $\boldsymbol{E}^{(l)} = act(\boldsymbol{W}_l \boldsymbol{E}^{(l-1)} + \boldsymbol{b}^l)$ $\boldsymbol{E}^{(1)} = act(\boldsymbol{W_1}\boldsymbol{K} + \boldsymbol{W_0}\boldsymbol{q} + \boldsymbol{b}^1)$	Pavlopoulos et al., 2017
location-based	$f(\boldsymbol{q}, \boldsymbol{K}) = f(\boldsymbol{q})$	Luong et al., 2015

w_{imp}: importance vector

General Attention Model

- In most cases the task requires the computation of new representation of the keys

  -> need another input element: a sequence V of d_k vectors v_i, the values, representing the data whereupon the attention computed from K and q is to be applied.

- Each element of V corresponds to one and only one element of K, and the two can be seen as different representations of the same data.

A Unified Attention Model

- General attention model

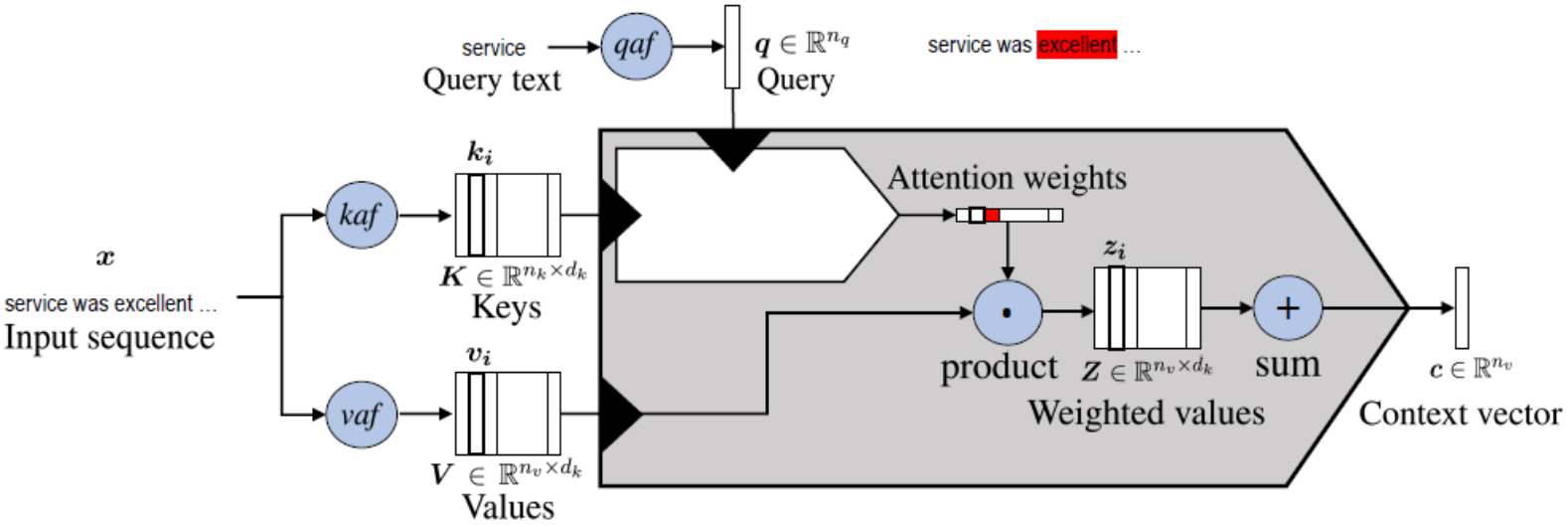


Figure 4: General attention model.

Taxonomy of Attention

Category	Type
Number of Sequences	distinctive, co-attention, self
Number of Abstraction Levels	single-level, multi-level
Number of Positions	soft/global, hard, local
Number of Representations	multi-representational, multi-dimensional

Table 1: Categories and types of attention within each category.

Number of sequences

- **Distinctive**: single input and corresponding output
- **Co-attention**: multiple inputs sequences; attention weights are learned jointly to capture interactions between inputs.
- **Self attention**: input is a sequence but the output is not a sequence; attention learns relevant tokens in the input sequence; the query and candidate states belong to the same sequence (e.g. text classification, recommendation)

Co-attention

- Coarse-grained models compute attention on each input, using an embedding of the other input as a query.
- Fine-grained models consider each element of an input with respect to each element of the other input.

Coarse-grained Co-attention

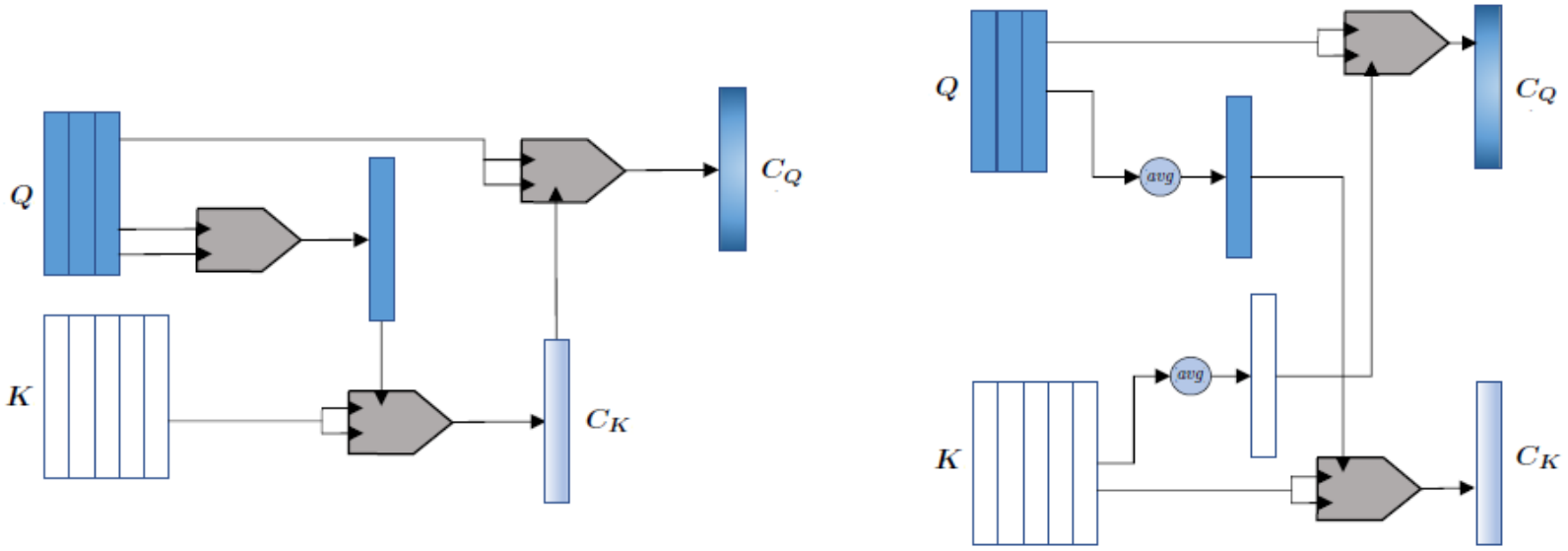


Figure 7: Coarse-grained co-attention by Lu et al. (2016) (left) and Ma et al. (2017) (right).

Fine-grained Co-attention

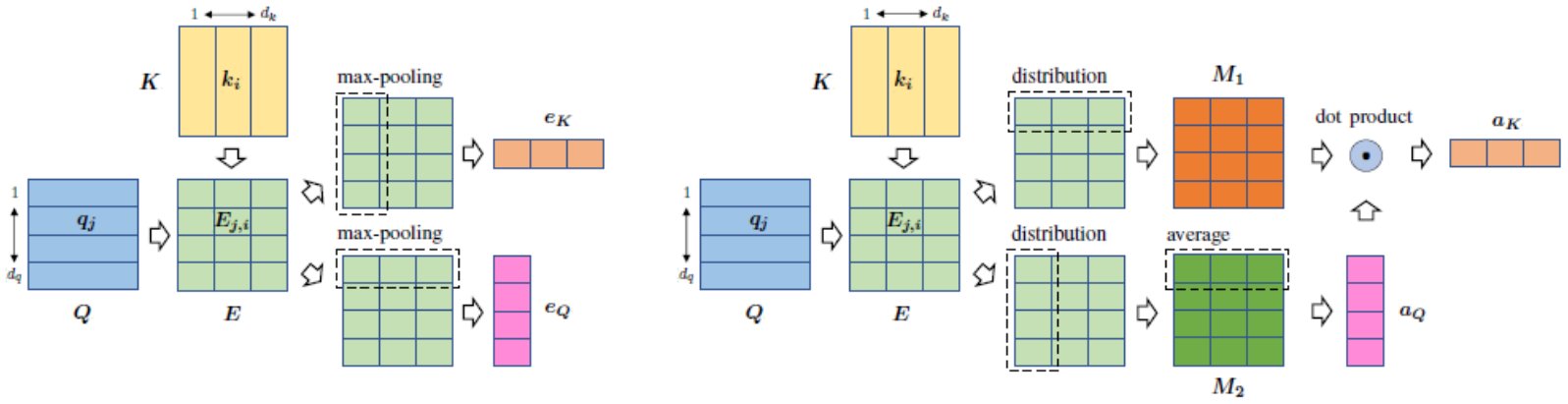


Figure 8: Fine-grained co-attention models presented by dos Santos et al. (2016) (left) and by Cui et al. (2017) (right). Dashed lines show how max pooling/distribution functions are applied (column-wise or row-wise).

Number of abstraction levels

- **Single-level**: attention weights are computed only for the original input sequence.

- **Multi-level**: attention is applied on multiple levels of abstraction of the input sequence in a sequential manner. The output of the lower abstraction level becomes the query for the higher abstraction level.

Hierarchical Attention

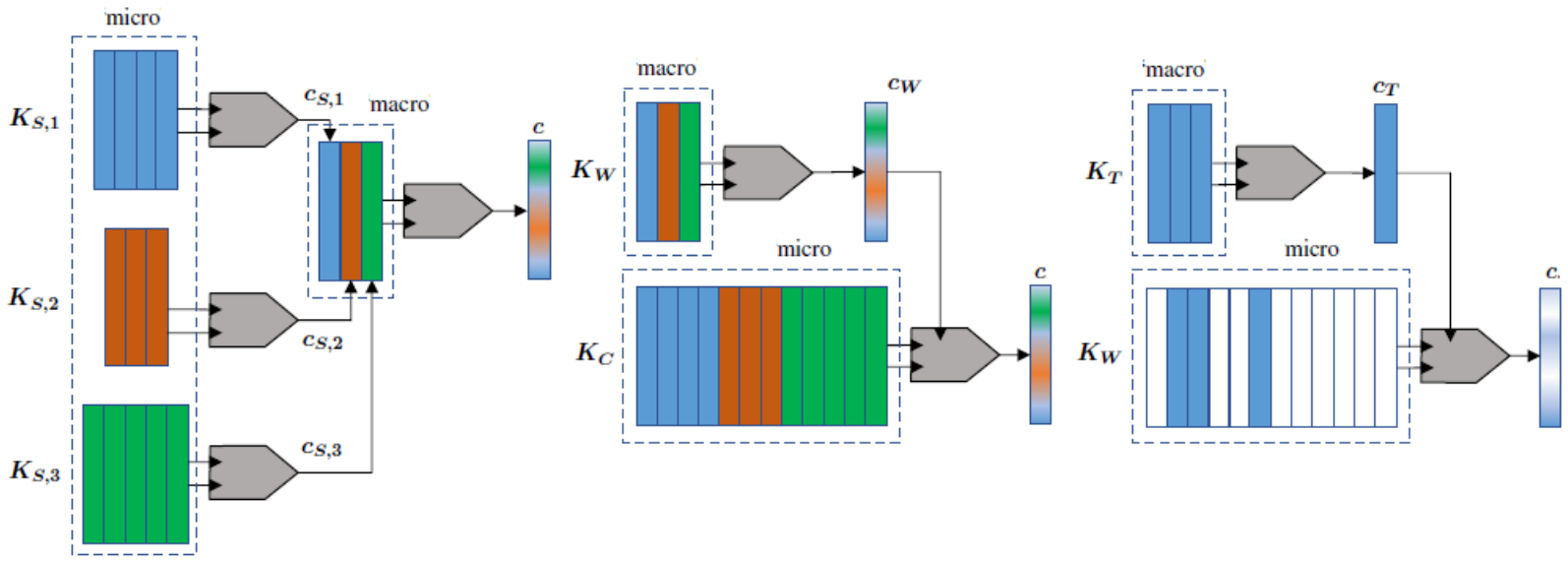


Figure 6: Hierarchical input attention models defined by Yang et al. (2016b) (left), Zhao and Zhang (2018) (center), and Ma et al. (2018) (right). The attention functions on different levels are applied sequentially, left-to-right.

Number of Positions

- **Soft attention**: uses a weighted average of all hidden states of the input sequence to build the context vector.
  - Pro: the model is smooth and differentiable.
  - Con: expensive when the source input is large.
- **Hard attention**: the context vector is computed from stochastically sampled hidden states in the input sequence.
  - Pro: less calculation at the inference time.
  - Con: the model is non-differentiable and requires more complicated techniques such reinforcement learning to train.

Number of Positions

- **Global attention**: similar to the soft attention model.
- **Local attention**: intermediate between soft and hard attention; detect an attention position within the input and pick a window around that position to create a local soft attention model.

Global vs Local Attention

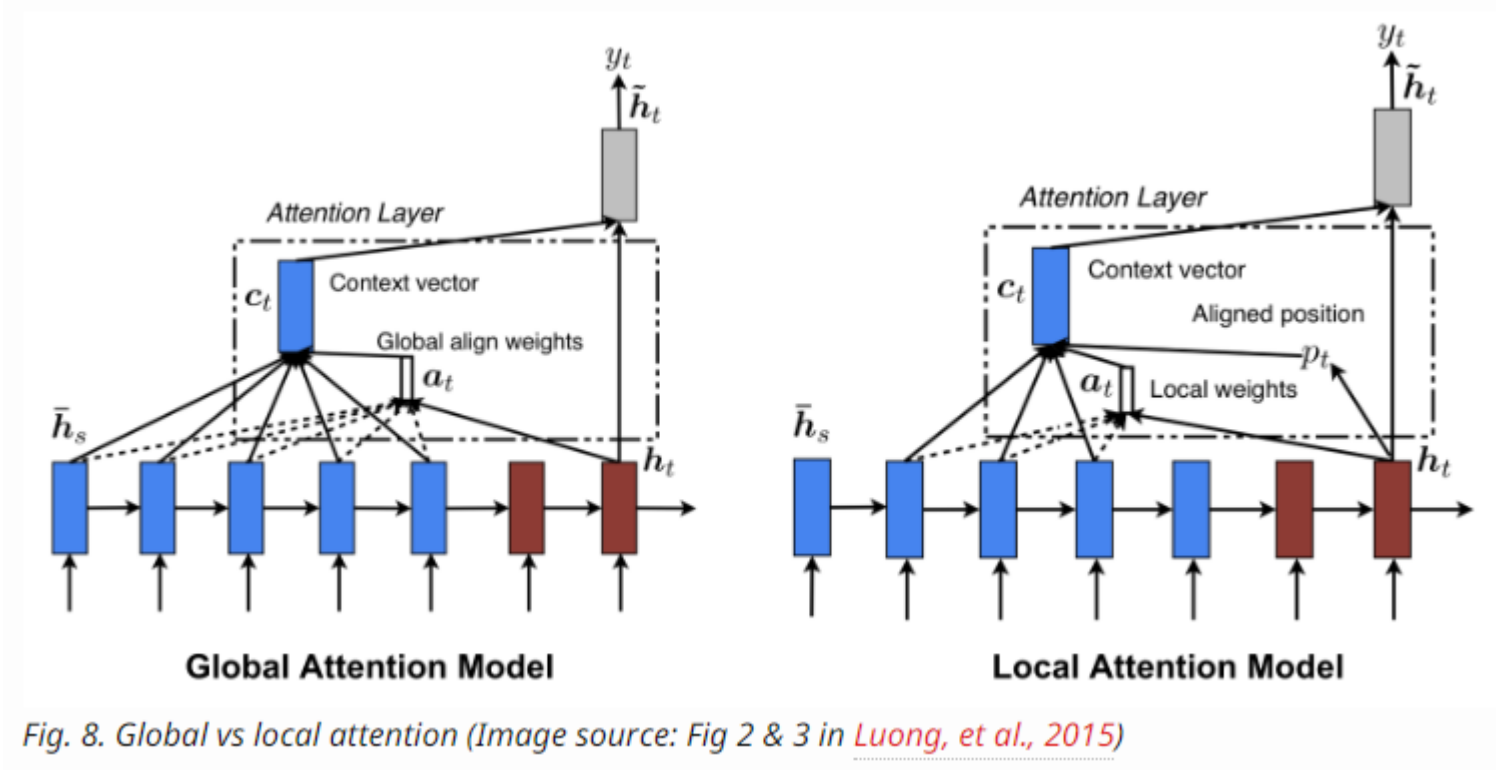


Fig. 8. Global vs local attention (Image source: Fig 2 & 3 in Luong, et al., 2015)

https://arxiv.org/pdf/1508.04025.pdf