# Text Classification and Sentence Representation

Kyunghyun Cho
New York University
Courant Institute (Computer Science) and Center for Data Science
Facebook AI Research

# Text Classification

- Input: a natural language sentence/paragraph
- Output: a category to which the input text belongs
  - There are a fixed number $C$ of categories
- Examples
  - Sentiment analysis: is this review positive or negative?
  - Text categorization: which category does this blog post belong to?
  - Intent classification: is this a question about a Chinese restaurant?

# How to represent a sentence

- A sentence is a variable-length sequence of tokens: $X = (x_1, x_2, \ldots, x_T)$
- Each token could be any one from a vocabulary: $x_t \in V$
- Examples
  - (커넥트, 재단에서, 강의, 중, 입니다, .)
    - Vocabulary: All unique, space-separated tokens in Korean
  - (커넥트, 재단, 에서, 강의, 중, 입니다, .)
    - Vocabulary: All uniqued, segmented tokens in Korean
  - (커, 넥, 트, [], 재, 단, 에, 서, [], 강, 의, [], 중, [], 입, 니, 다, .)
    - Vocabulary: All Korean syllables
  - And many more possibilities…

# How to represent a sentence

- A sentence is a variable-length sequence of tokens: $X = (x_1, x_2, \ldots, x_T)$
- Each token could be any one from a vocabulary: $x_t \in V$
- Once the vocabulary is fixed and encoding is done, a sentence or text is just a sequence of "integer indices".
- Examples:
  - (커넥트, 재단, 에서, 강의, 중, 입니다, .)
  - (5241, 827, 20, 288, 12, 19, 5)

$V =$

| Index | Token |
|---|---|
| 5 | . |
| 12 | 중 |
| 19 | 입니다 |
| 20 | 에서 |
| … | … |
| 288 | 강의 |
| 827 | 재단 |
| … | … |

# How to represent a token

- A token is an integer "index".
- How do should we represent a token so that it reflects its "meaning"?
- First, we assume nothing is known: use an one-hot encoding.

$$x = [0, 0, 0, \ldots, 0, 1, 0, \ldots, 0] \in \{0, 1\}^{|V|}$$

  - $|V|$: the size of vocabulary
  - Only one of the elements is 1: $\sum_{i=1}^{|V|} x_i = 1$
- Every token is equally distant away from all the others.

$$\|x - y\| = c > 0, \text{if } x \neq y$$

# How to represent a token

- How do should we represent a token so that it reflects its “meaning”?
- First, we assume nothing is known: use an one-hot encoding.
- Second, the neural network capture the token’s meaning as a vector.
- This is done by a simple matrix multiplication:
  $Wx = W\left[\hat{x}\right]$, if $x$ is one-hot,
  where $\hat{x} = \arg\max_j x_j$ is the token’s index in the vocabulary.

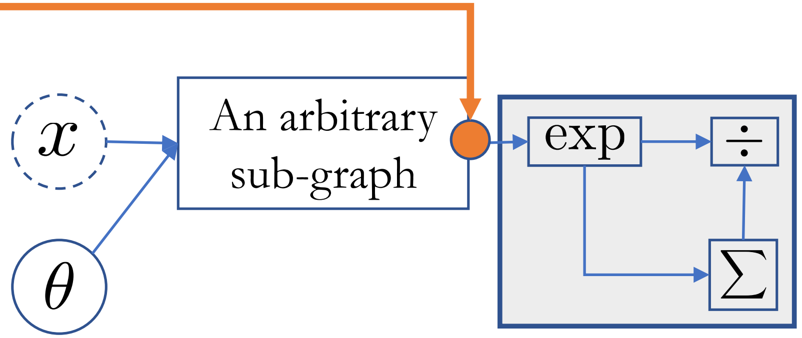
Table Lookup

$$W^\top = \begin{bmatrix} w_{11} & \cdots & w_{1d} \\ \vdots & \ddots & \vdots \\ w_{\hat{x}1} & \cdots & w_{\hat{x}d} \\ \vdots & \ddots & \vdots \\ w_{|V|1} & \cdots & w_{|V|d} \end{bmatrix}$$

# How to represent a sentence – CBoW

- After the table-lookup operation,* the input sentence is a sequence of continuous, high-dimensional vectors:
  $X = (e_1, e_2, \ldots, e_T)$, where $e_t \in \mathbb{R}^d$
- The sentence length $T$ differs from one sentence to another.
- The classifier needs to eventually compress it into a single vector.

**A fixed-size vector representation of the input**





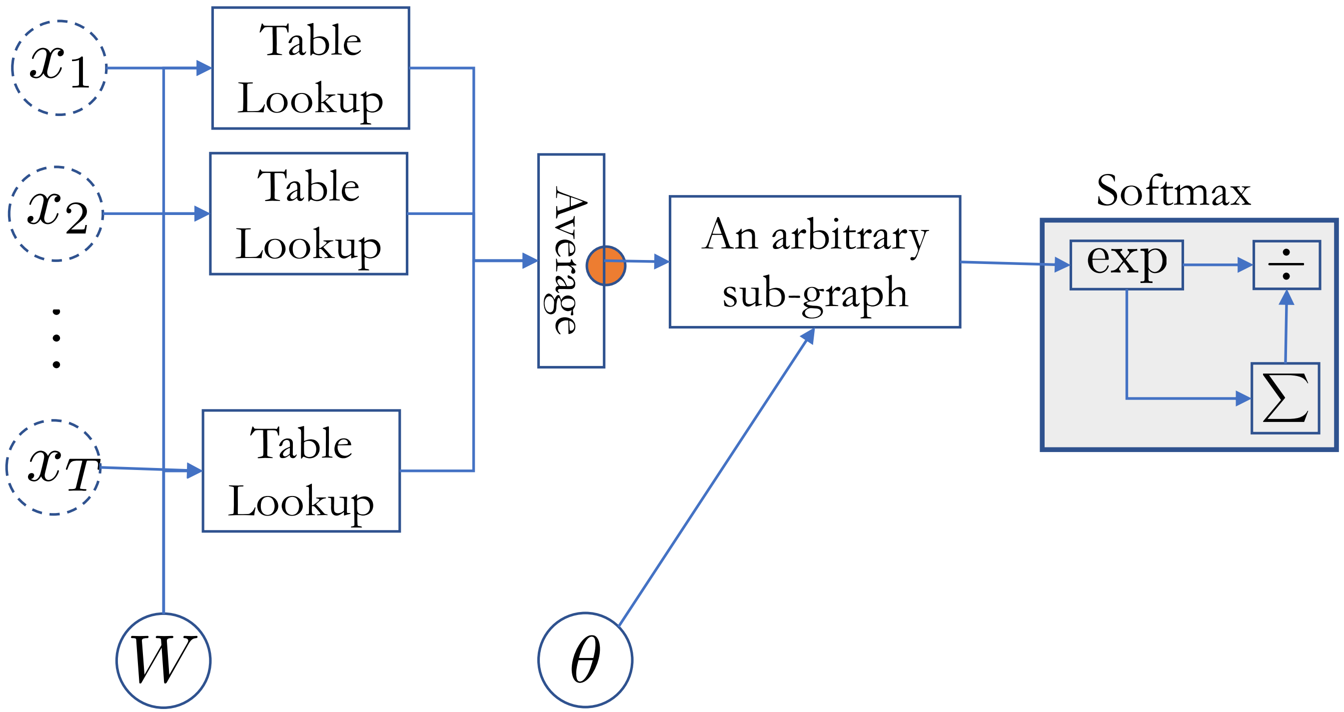

* The table-lookup operation would be one node in the DAG.

# How to represent a sentence – CBoW

- Continuous bag-of-words
  - Ignore the order of the tokens: $(x_1, x_2, \ldots, x_T) \to \{x_1, x_2, \ldots, x_T\}$
  - Simply average the token vectors: $\frac{1}{T}\sum_{t=1}^{T} e_t$
    - Averaging is a differentiable operator.
    - Just one operator node in the DAG.
  - Generalizable to bag-of-n-grams
    - N-gram: a phrase of N tokens
    - *Think of how you would do!*
- Extremely effective in text classification [Iyyer et al., 2016; Cho, 2017; and many more]
  - For instance, if there are many positive words, the review is likely positive.
- In practice, use FastText [Bojanowski et al., 2017]

# How to represent a sentence – CBoW

- Continuous bag-of-words based multi-class text classifier

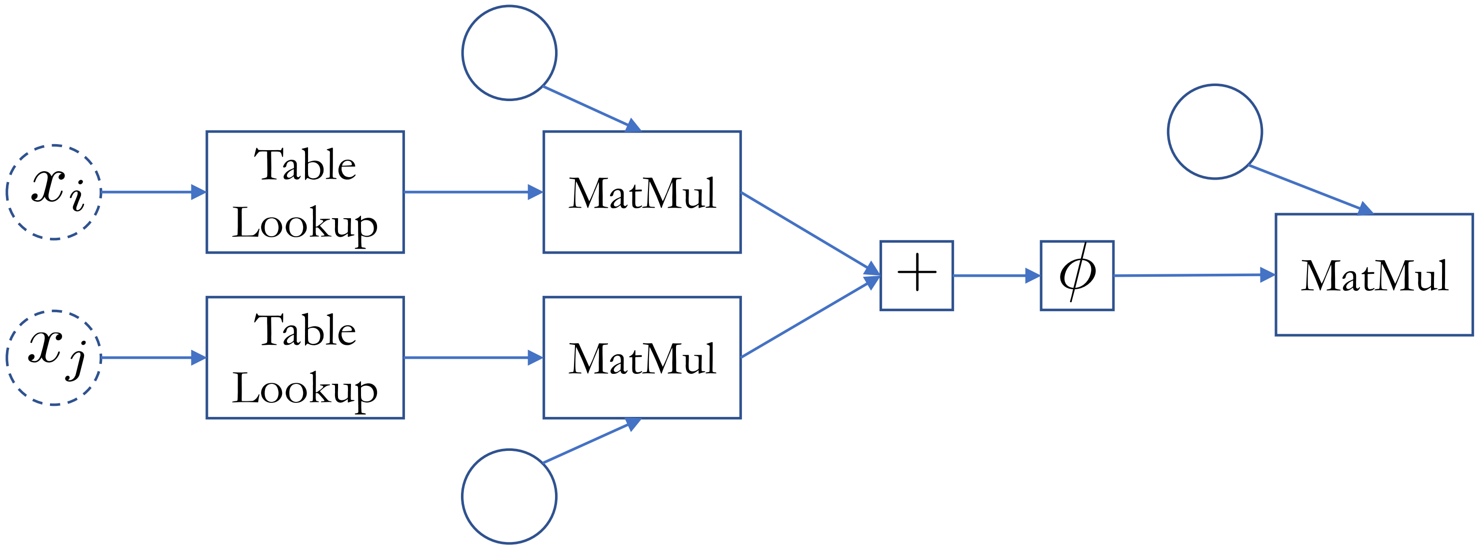



- With this DAG, you use automatic backpropagation and stochastic gradient descent to train the classifier.

# How to represent a sentence – RN

- Relation Network [Santoro et al., 2017]: Skip Bigrams
  - Consider all possible pairs of tokens: $(x_i, x_j), \forall i \neq j$
  - Combine two token vectors with a neural network for each pair

    $$f(x_i, x_j) = W\phi(U_{\text{left}} e_i + U_{\text{right}} e_j)$$
    - $\phi$ is a element-wise nonlinear function, such as tanh or ReLU ($\max(0, a)$)
    - One subgraph in the DAG.

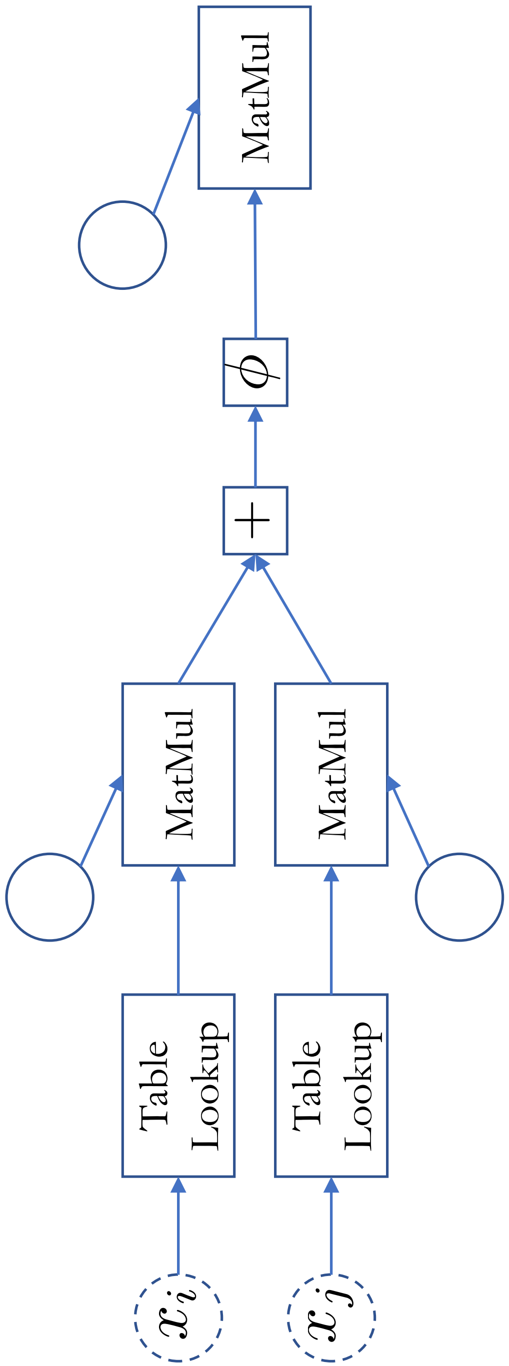


# How to represent a sentence – RN

- Relation Network: Skip Bigrams
  - Considers all possible pairs of tokens: $(x_i, x_j), \forall i \neq j$
    $f(x_i, x_j) = W\phi(U_{\text{left}} e_i + U_{\text{right}} e_j)$
  - Considers the "relation"ship between each pair of words
  - Averages all these relationship vectors

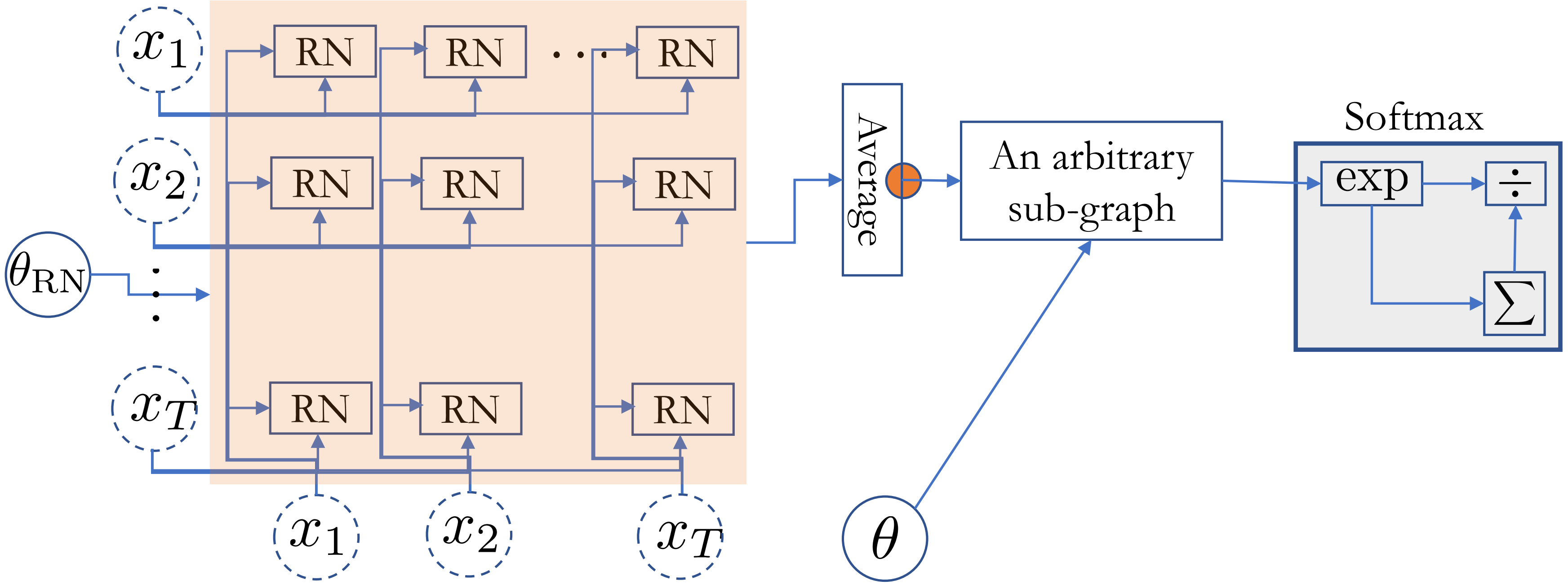
$$\text{RN}(X) = \frac{1}{2N(N-1)} \sum_{i=1}^{T-1} \sum_{j=i+1}^{T} f(x_i, x_j)$$

  - Could be generalized to triplets and so on at the expense of computational efficient.



# How to represent a sentence – RN

- Relation Network: Skip Bigrams
  - Considers all possible pairs of tokens: $(x_i, x_j), \forall i \neq j$
    $f(x_i, x_j) = W\phi(U_{\text{left}} e_i + U_{\text{right}} e_j)$
  - Considers the pair-wise "relation"ship $\text{RN}(X) = \frac{1}{2N(N-1)} \sum_{i=1}^{T-1} \sum_{j=i+1}^{T} f(x_i, x_j)$
  - Averages all these relationship vectors

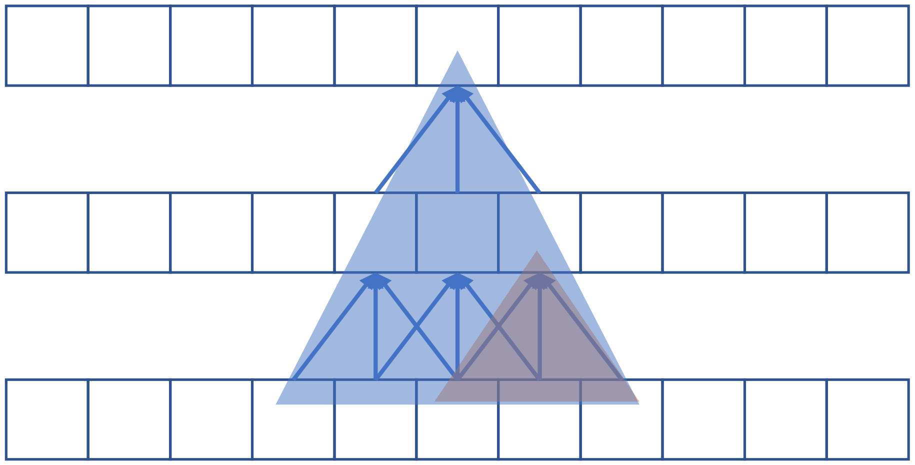


# How to represent a sentence – CNN

- Convolutional Networks [Kim, 2014; Kalchbrenner et al., 2015]
  - Captures $k$-grams hierarchically
  - One 1-D convolutional layer: considers all $k$-grams

$$h_t = \phi\left(\sum_{\tau=-k/2}^{k/2} W_\tau e_{t+\tau}\right), \text{ resulting in } H = (h_1, h_2, \ldots, h_T).$$

  - Stack more than one convolutional layers: progressively-growing window
  - Fits our intuition of how sentence is understood: **tokens→multi-word expressions→phrases→sentence**



# How to represent a sentence – CNN

- Convolutional Networks [Kim, 2014; Kalchbrenner et al., 2015]
  - Captures *k*-grams hierarchically
  - Stack more than one convolutional layers: progressively-growing window
  - **tokens→multi-word expressions→phrases→sentence**
- In practice, just another operation node in a DAG:
  - Extremely efficient implementations are available in all of the major frameworks.
- Recent advances
  - Multi-width convolutional layers [Kim, 2014; Lee et al., 2017]
  - Dilated convolutional layers [Kalchbrenner et al., 2016]
  - Gated convolutional layers [Gehring et al., 2017]

# How to represent a sentence – Self-Attention

- Can we combine and generalize the relation network and the CNN?
- Relation Network:
  - Each token's representation is computed against all the other tokens

$$h_t = f(x_t, x_1) + \cdots + f(x_t, x_{t-1}) + f(x_t, x_{t+1}) + \cdots + f(x_t, x_T)$$

- CNN:
  - Each token's representation is computed against neighbouring tokens

$$h_t = f(x_t, x_{t-k}) + \cdots + f(x_t, x_t) + \cdots + f(x_t, x_{t+k})$$

- RN considers the entire sentence vs. CNN focuses on the local context.

# How to represent a sentence – Self-Attention

- Can we combine and generalize the relation network and the CNN?
- CNN as a weighted relation network:
  - Original: $h_t = f(x_t, x_{t-k}) + \cdots + f(x_t, x_t) + \cdots + f(x_t, x_{t+k})$
  - Weighted:

$$h_t = \sum_{t'=1}^{T} \mathbb{I}(|t' - t| \leq k) f(x_t, x_{t'})$$

where $\mathbb{I}(S) = 1, \text{ if } S \text{ is true, and } 0, \text{otherwise}$.

- Can we compute those weights instead of fixing them to 0 or 1?

# How to represent a sentence – Self-Attention

- Can we compute those weights instead of fixing them to 0 or 1?
- That is, compute the weight of each pair $(x_t, x_{t'})$

$$h_t = \sum_{t'=1}^{T} \alpha(x_t, x_{t'}) f(x_t, x_{t'})$$

- The weighting function could be yet another neural network
  - Just another subgraph in a DAG: easy to use!

    $$\alpha(x_t, x_{t'}) = \sigma(\text{RN}(x_t, x_{t'})) \in [0, 1]$$

  - Perhaps we want to normalize them so that the weights sum to one

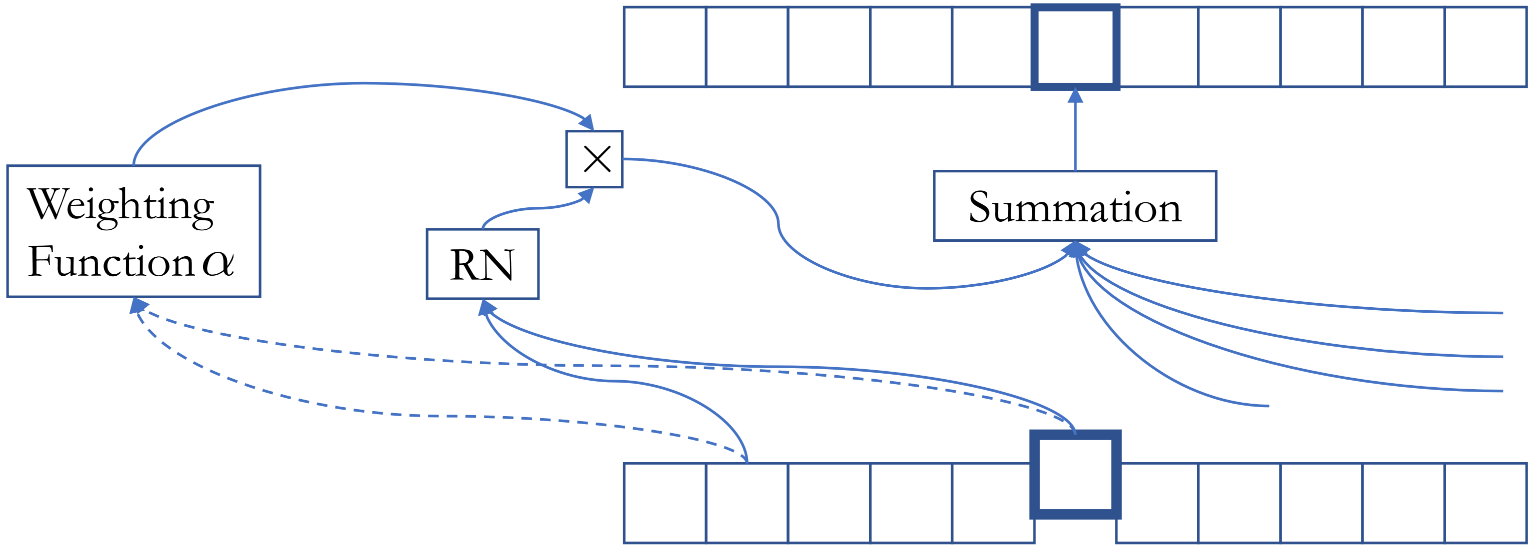
    $$\alpha(x_t, x_{t'}) = \frac{\exp(\beta(x_t, x_{t'}))}{\sum_{t''=1}^{T} \exp(\beta(x_t, x_{t''}))}, \text{where } \beta(x_t, x_{t'}) = \text{RN}(x_t, x_{t'}))$$



[Bahdanau et al., 2015; Vaswani et al., 2017; Google NLP, 2016]

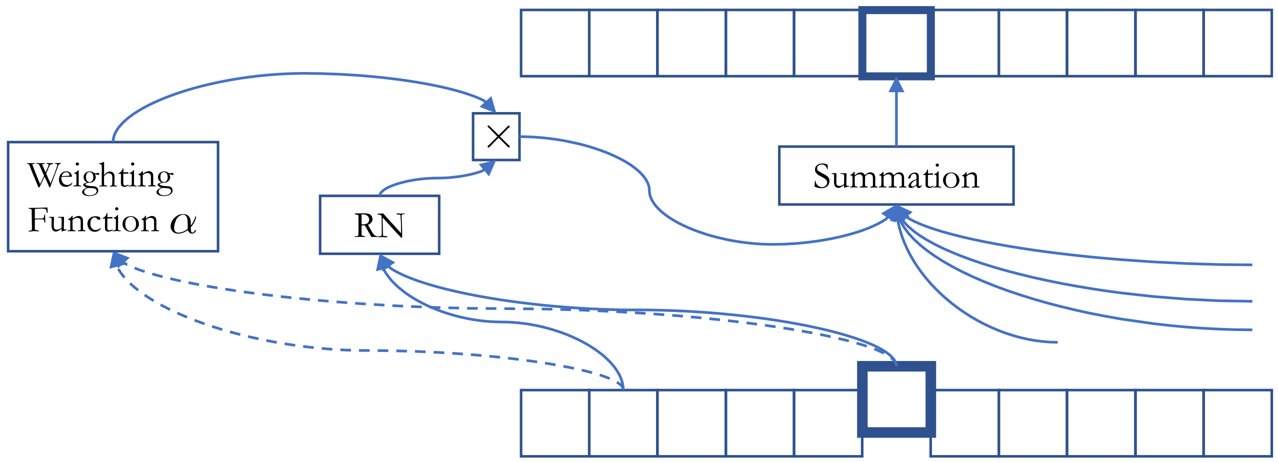
# How to represent a sentence – Self-Attention

- Self-Attention: a generalization of CNN and RN.
- Able to capture long-range dependencies within a single layer.
- Able to ignore irrelevant long-range dependencies.



# How to represent a sentence – Self-Attention

- Self-Attention: a generalization of CNN and RN.
- Able to capture long-range dependencies within a single layer.
- Able to ignore irrelevant long-range dependencies.
- Further generalization via multi-head and multi-hop attention



# How to represent a sentence – RNN

- Weaknesses of self-attention
    1. Quadratic computational complexity $O(T^2)$
    2. Some operations cannot be done easily: e.g., counting, …
- Online compression of a sequence $O(T)$

  $h_t = \text{RNN}(h_{t-1}, x_t)$, where $h_0 = 0$.
- Memory $h_t$ allows it to be Turing complete.*



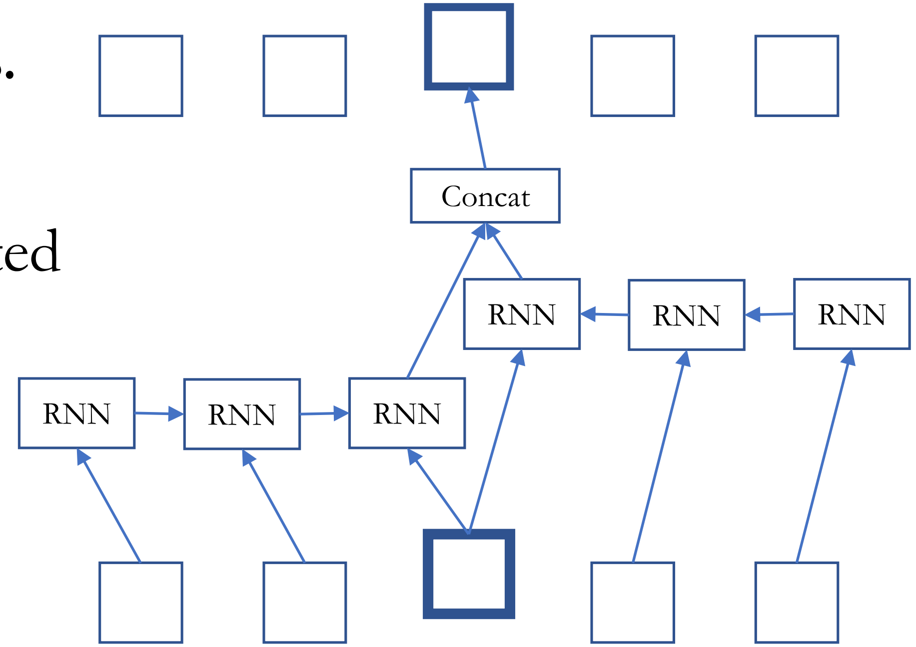
* Under some extreme assumptions…

# How to represent a sentence – RNN

- Recurrent neural network: online compression of a sequence $O(T)$

  $h_t = \mathrm{RNN}(h_{t-1}, x_t)$, where $h_0 = 0$.

- Bidirectional RNN to account for both sides.
- Inherently sequential processing
  - Less desirable for modern, parallelized, distributed computing infrastructure.
- LSTM [Hochreiter&Schmidhuber, 1999] and GRU [Cho et al., 2014] have become de facto standard
  - All standard frameworks implement them.
  - Efficient GPU kernels are available.



# How to represent a sentence

- We have learned five ways to extract a sentence representation:
  - In all but CBoW, we end up with a set of vector representations.
    $H = \{h_1, \ldots, h_T\}$
  - These approaches could be "stacked" in an arbitrary way to improve performance.
    - Chen, Firat, Bapna et al. [2018] combine self-attention and RNN to build the state-of-the-art machine translation system.
    - Lee et al. [2017] stack RNN on top of CNN to build an efficient fully character-level neural translation system.
    - Because all of these are differentiable, the same mechanism (backprop+SGD) works as it is for any other machine learning model.
  - These vectors are often averaged for classification.

# We learned in this lecture…

- Token representation
  - How do we represent a discrete token in a neural network?
  - Training this neural network leads to so-called **continuous word embedding**.
- Sentence representation
  - How do we extract useful representation from a sentence?
  - We learned five different ways to do so: CBoW, RN, CNN, Self-Attention, RNN

# In the next lecture,

- What else can we do with this sentence representation?
  - Language generation: language modelling, machine translation, …
  - Question answering: machine reading, query reformulation, …
- We will focus on language generation.