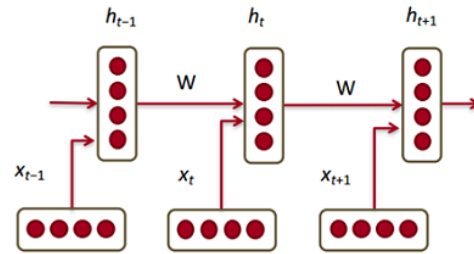


Natural Language Processing



$$x_{shirt} - x_{clothing} \approx x_{chair} - x_{furniture} \quad \log p(o|c) = \log \frac{\exp(u_o^T v_c)}{\sum_{w=1}^W \exp(u_w^T v_c)}$$
$$x_{king} - x_{man} \approx x_{queen} - x_{woman}$$

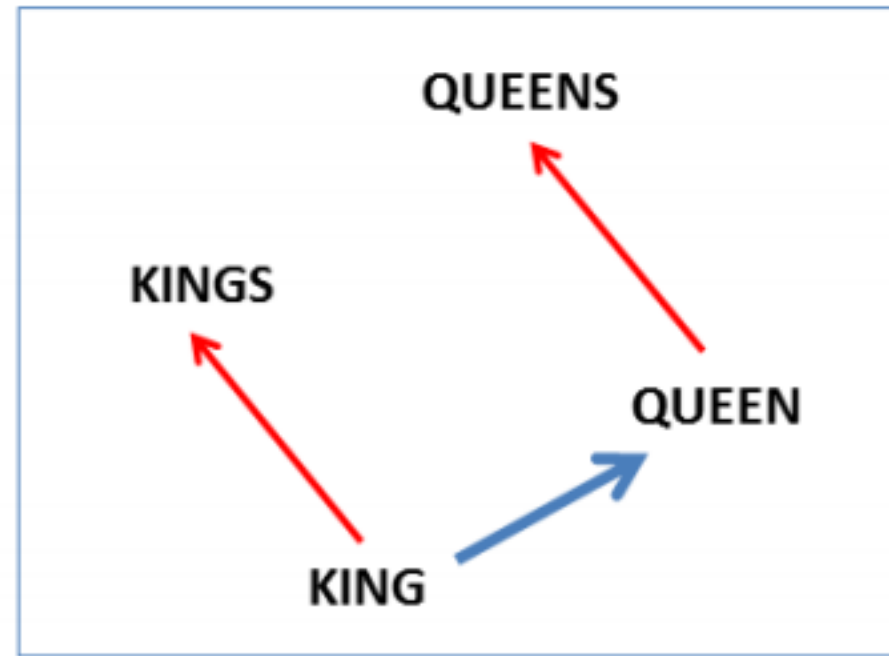
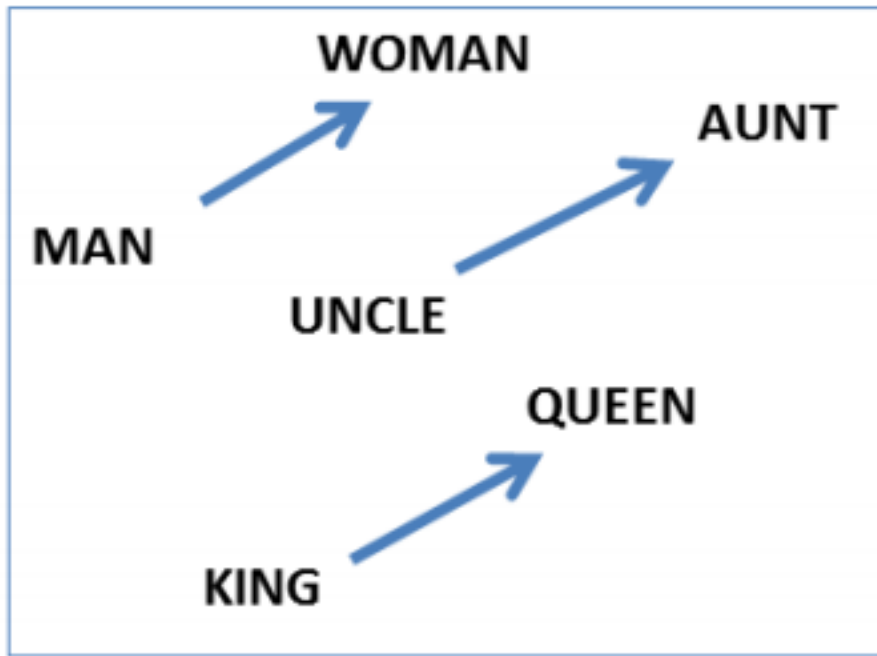
Contents

- Token Representation – Word2Vec
- Sentence Representation – CBOW, RN, CNN, Self Attention, RNN
- Language Model – N-Gram LM, NNLM
- Neural Machine Translation

How to Represent a Token

- Intuitive embedding – one hot encoding
 - Apple, Strawberry, Dog 세 단어가 있을 때,
 - Apple $\rightarrow [1, 0, 0]$
 - Strawberry $\rightarrow [0, 1, 0]$
 - Dog $\rightarrow [0, 0, 1]$
- 장점
 - Easy!
- 단점
 - 단어들 간의 의미관계를 파악할 수 없음(apple과 strawberry, apple과 dog)
 - 단어가 많아지면?

We Want...

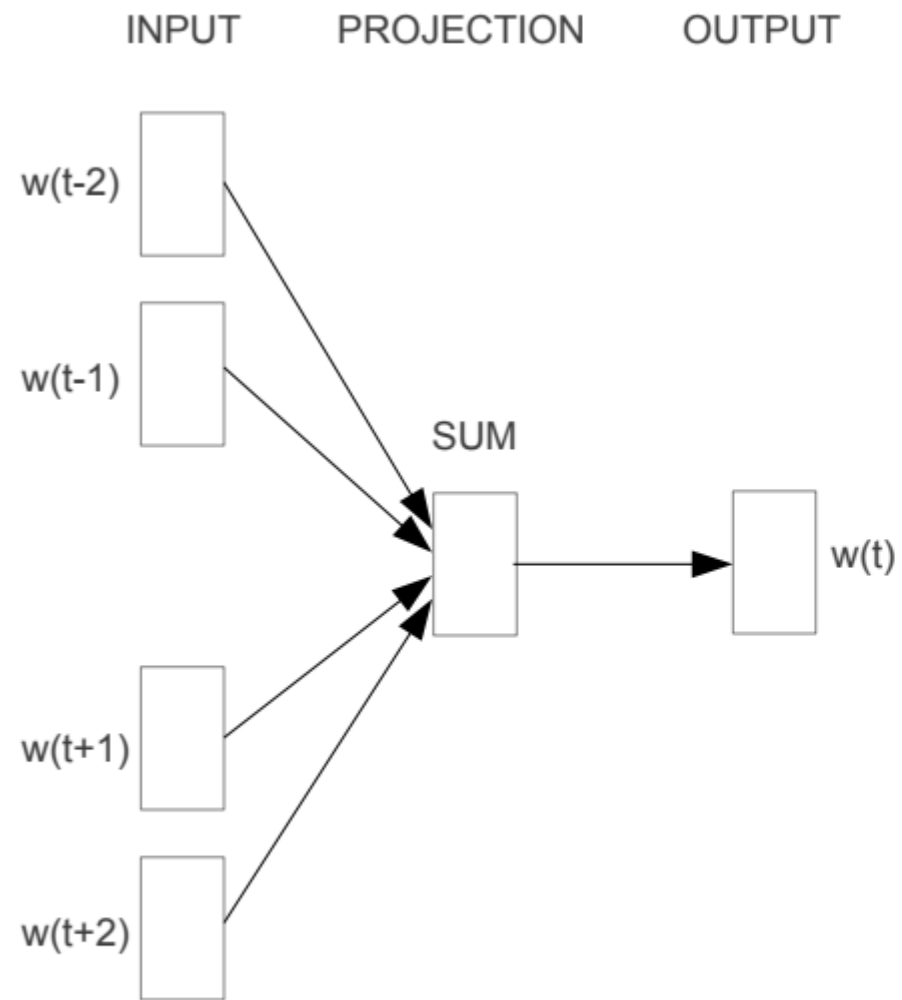


(Mikolov et al., NAACL HLT, 2013)

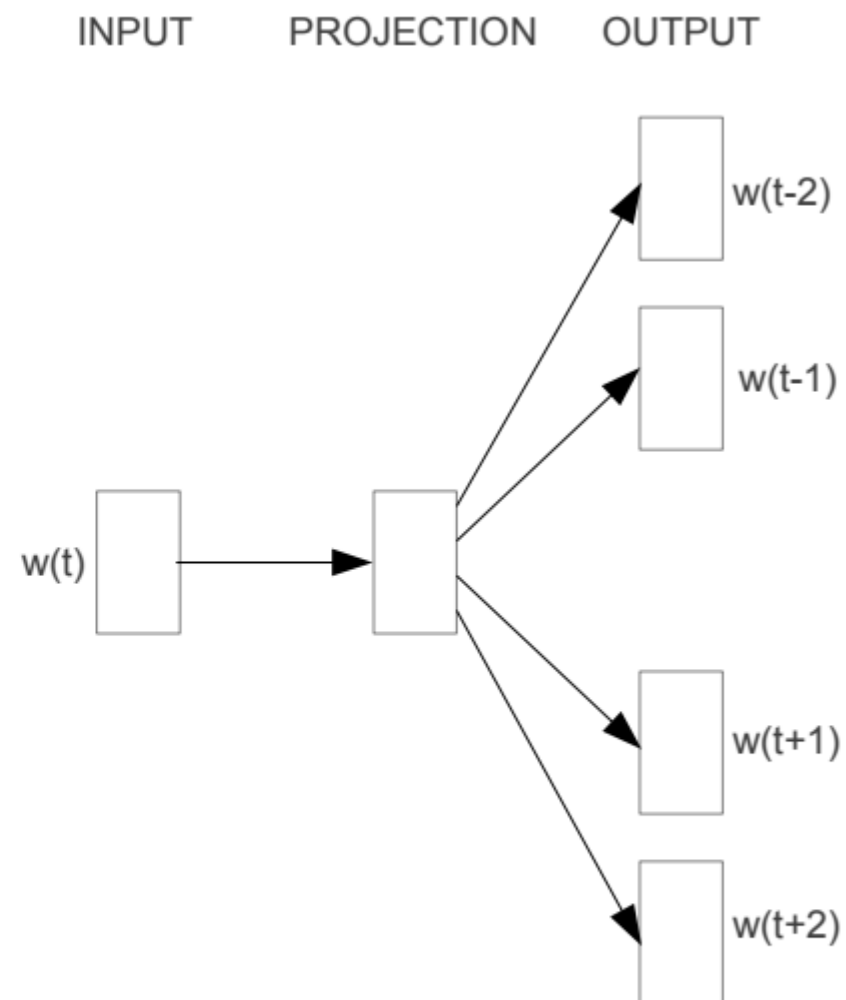
Let's Try It

- <http://w.elnn.kr/search/>
- How to train word vectors?????
 - MLP
 - RNN
 - ...

CBOW & Skip-gram



CBOW



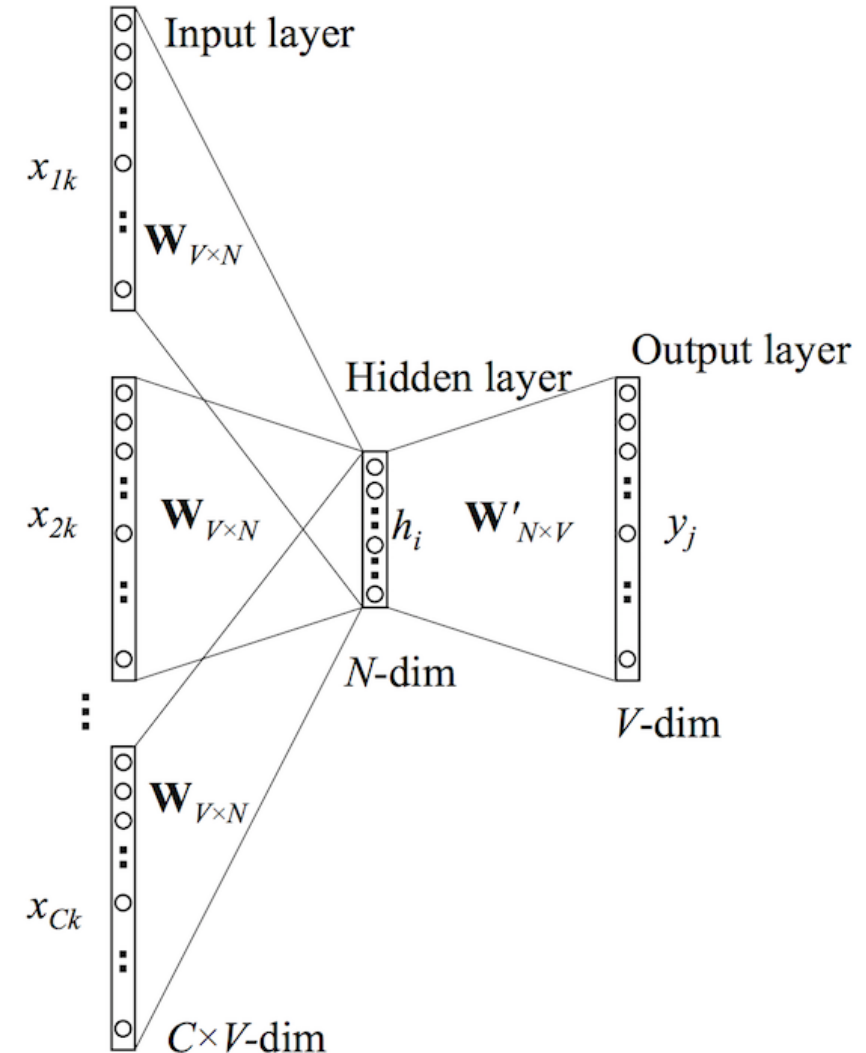
Skip-gram

CBOW – Continuous Bag of Words

- Similar to the feedforward NNLM, but
- Non-linear hidden layer removed
- Projection layer shared for all words
- Projected vectors are just averaged
- Called CBOW because the order of the words is lost

CBOW – Continuous Bag of Words

- Fill the blank
 - 아이스크림을 사 먹었는데, ____ 시려서 먹기가 힘들었다.
- 앞 뒤로 $C/2$ 개의 단어를 input으로 하여 center 단어를 맞추도록 학습
- Input은 one-hot encoding
- Input \rightarrow Hidden layer는 linear mapping($\text{avg}(Wx_{ik})$)
- Hidden \rightarrow Output layer는 $\text{Softmax}(W'h_i)$



Skip-gram

- Similar to CBOW, but instead of predicting the current word based on the context
- Tries to maximize classification of a word based on another word in the same sentence
- Thus, uses each current word as an input to a log-linear classifier
- Predicts words within a certain window

Skip-gram

- CBOW와 반대로 중심 단어를 주고 주변 단어들에 대한 확률 값을 출력함
- Window 내에 있는 단어의 확률이 최대가 되도록 학습
- Objective function

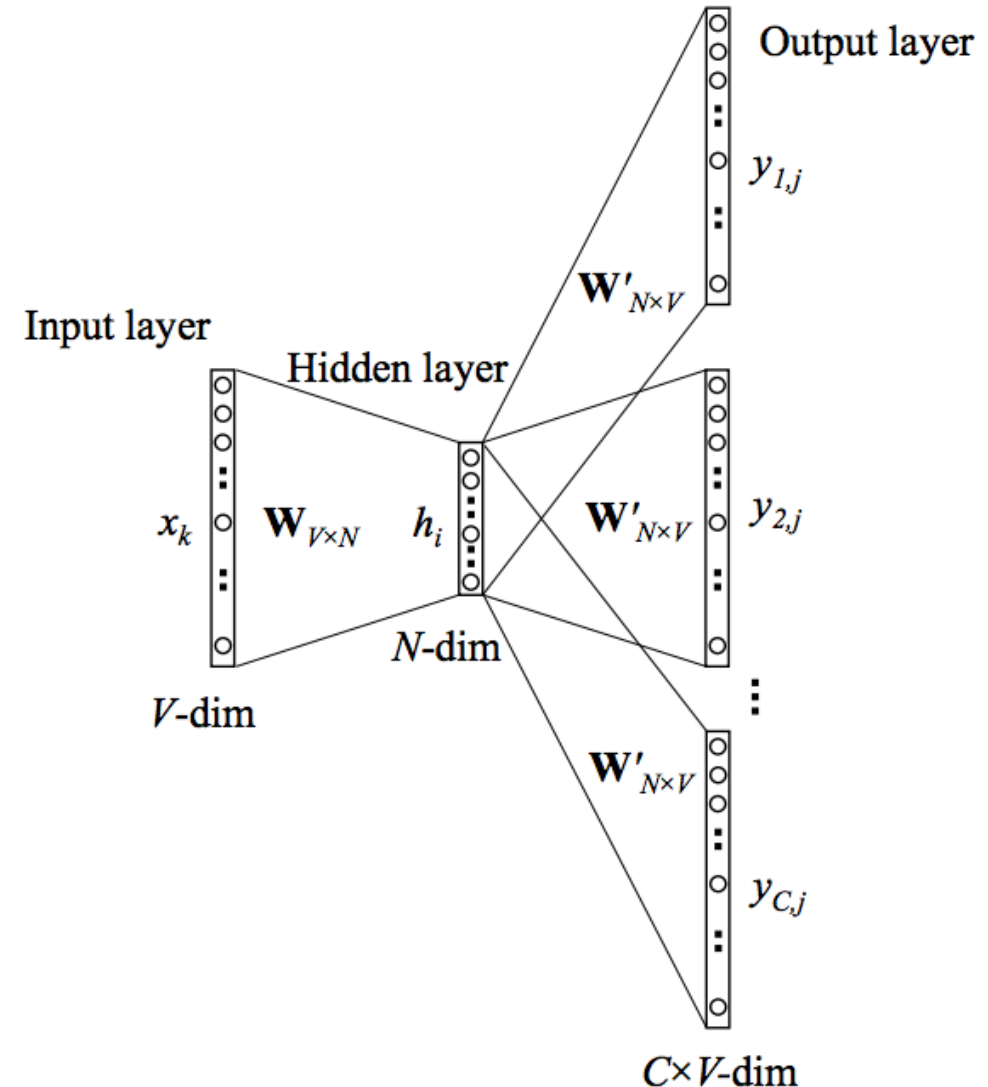
- Maximize

$$J'(\theta) = \prod_{t=1}^T \prod_{-C/2 \leq j \leq C/2, j \neq 0} P(x_{t+j} | x_t; \theta)$$

→ Negative log likelihood

- Minimize

$$J(\theta) = -\frac{1}{T} \sum_{t=1}^T \sum_{-C/2 \leq j \leq C/2, j \neq 0} \log P(x_{t+j} | x_t; \theta)$$

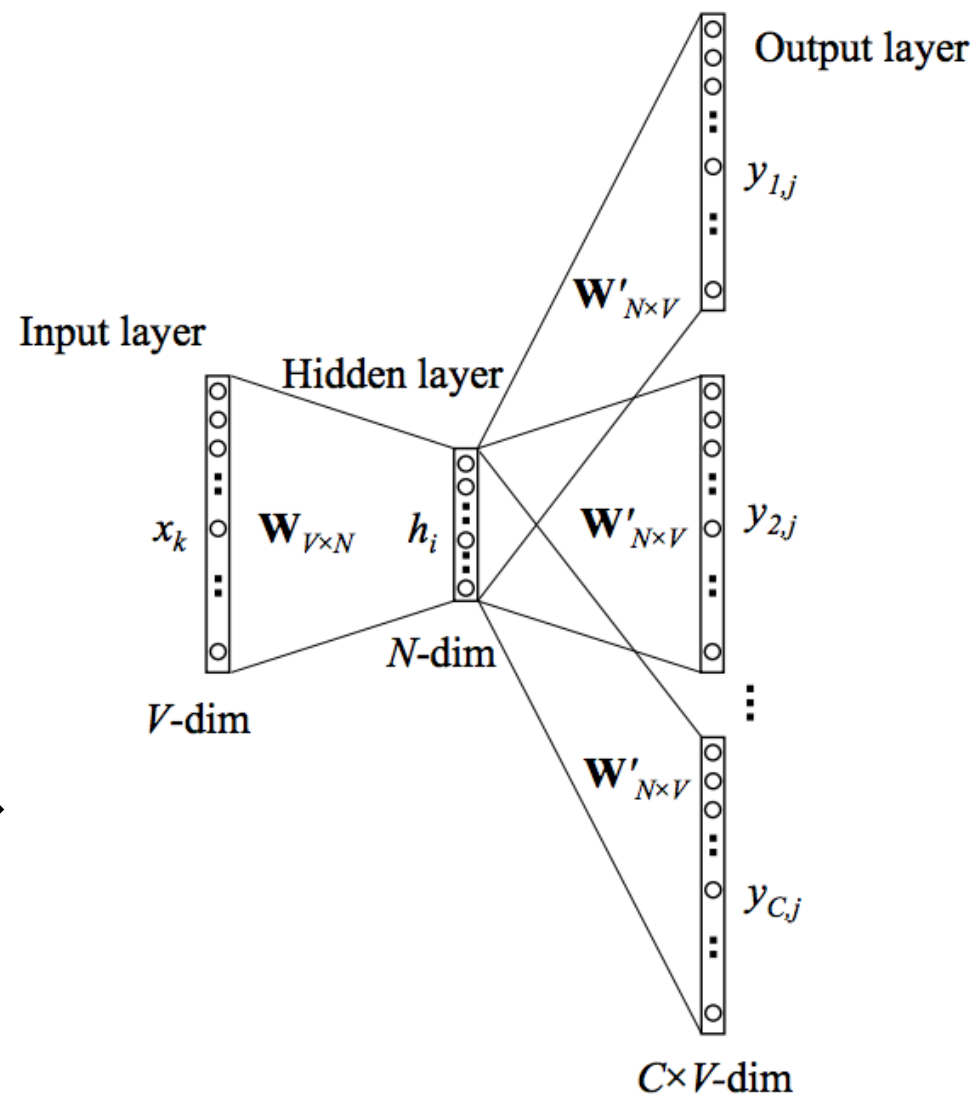


Skip-gram

- $P(o|c) = \frac{\exp(u_o^T \cdot v_c)}{\sum_{w=1}^V \exp(u_w^T \cdot v_c)}$
- $J(\theta) = -\frac{1}{T} \sum_{t=1}^T \sum_{-c/2 \leq j \leq c/2, j \neq 0} \log P(x_{t+j} | x_t; \theta)$

에서, P 의 분모를 구하려면 모든 단어에 대해 고려해야 한다

- Negative sampling 방법으로 window 밖에 있는 단어를 일부만 sampling 하여 근사한다 → NCE(Noise Contrastive Estimation)

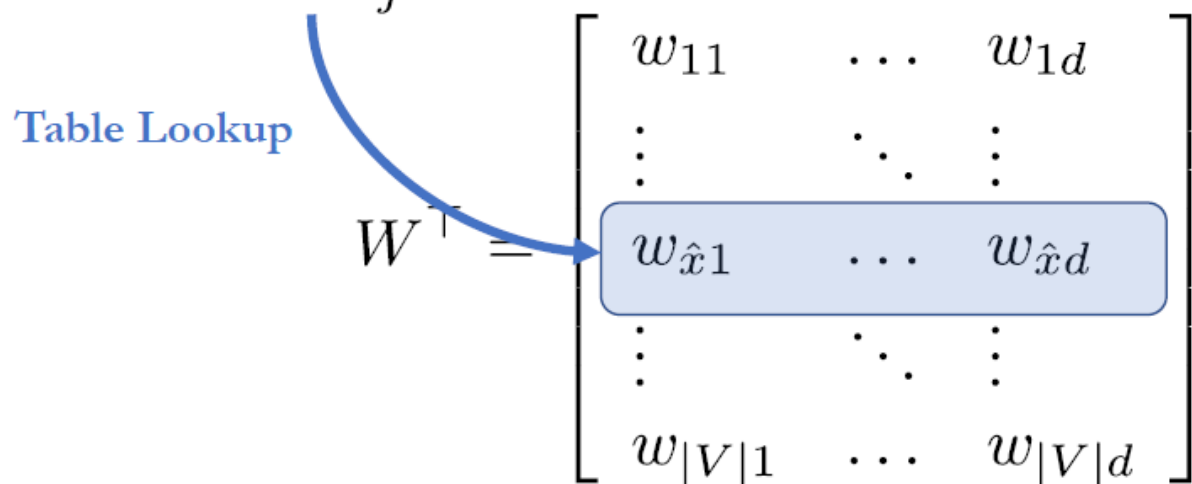


How to Represent a Token

- 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:

$Wx = W[\hat{x}]$, if x is one-hot,

where $\hat{x} = \arg \max_j x_j$ is the token’s index in the vocabulary.

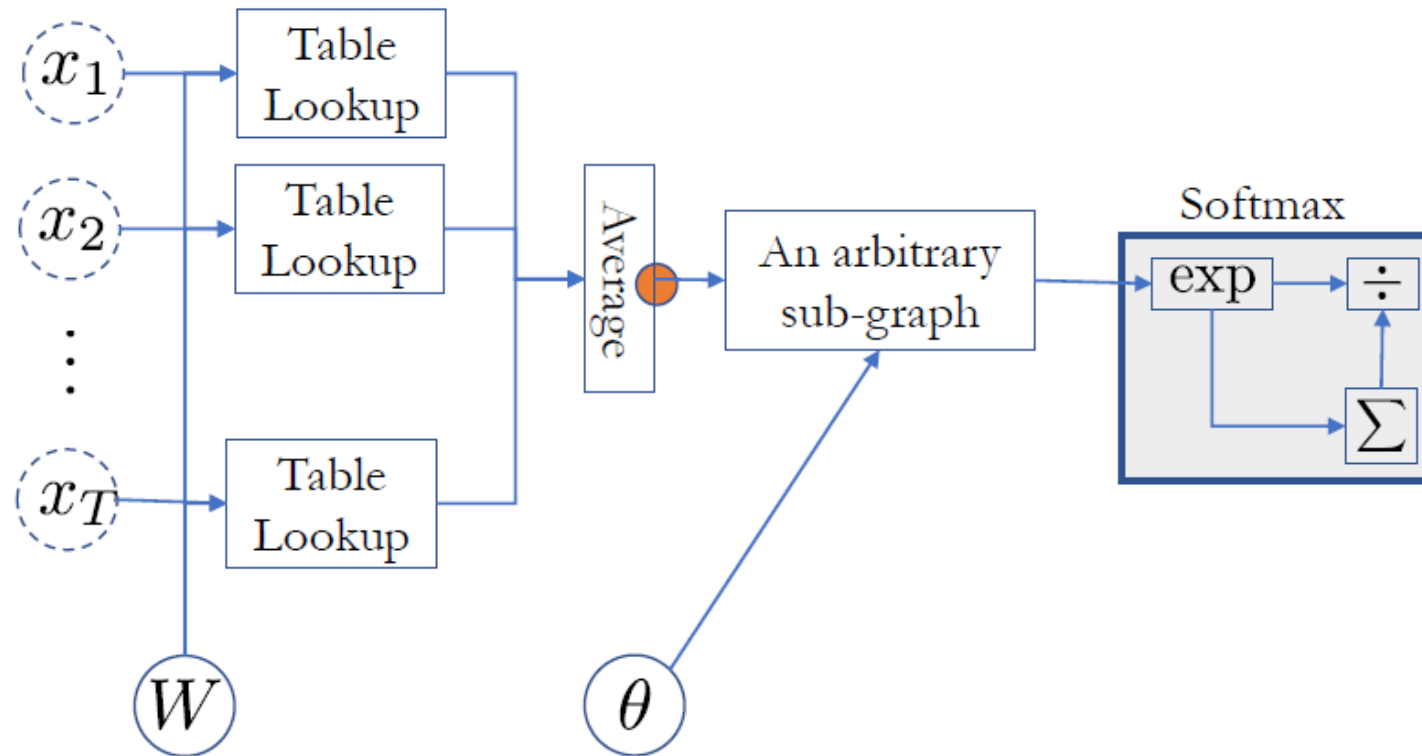


How to Represent a Sentence

- 단어는 Word2Vec을 사용하면 될 텐데, 문장은 어떻게 해야 할까?
- 문장 = 단어들의 sequence
- Sequence의 길이가 문장마다 모두 다름 → fixed length vector로 표현하는 방법을 찾아야 함

How to Represent a Sentence - CBOW

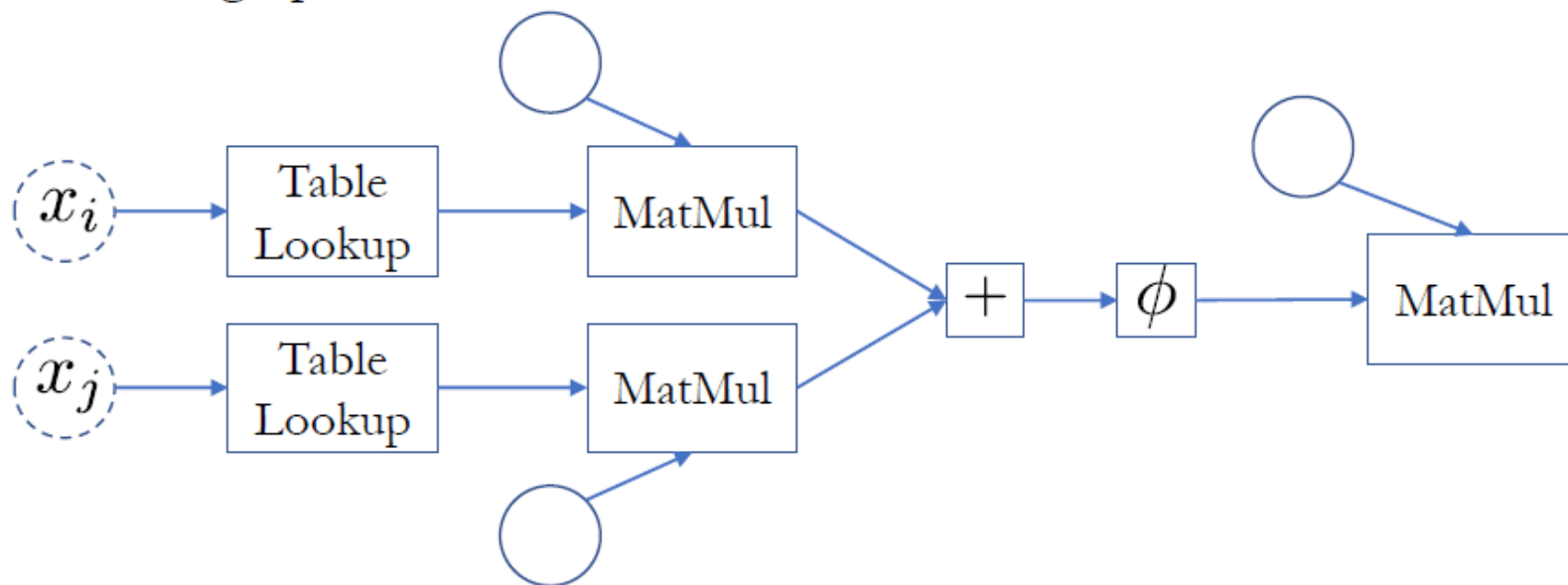
- Continuous bag-of-words based multi-class text classifier



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



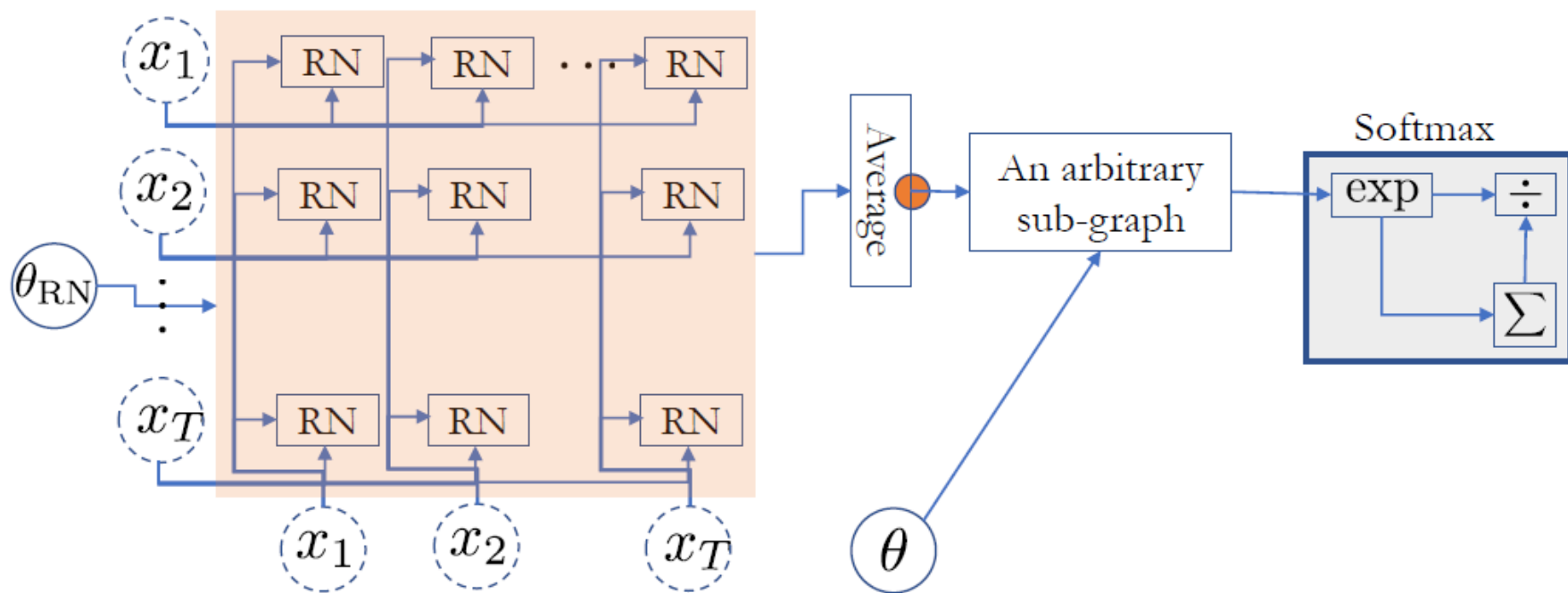
How to Represent a Sentence - RN

- 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 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



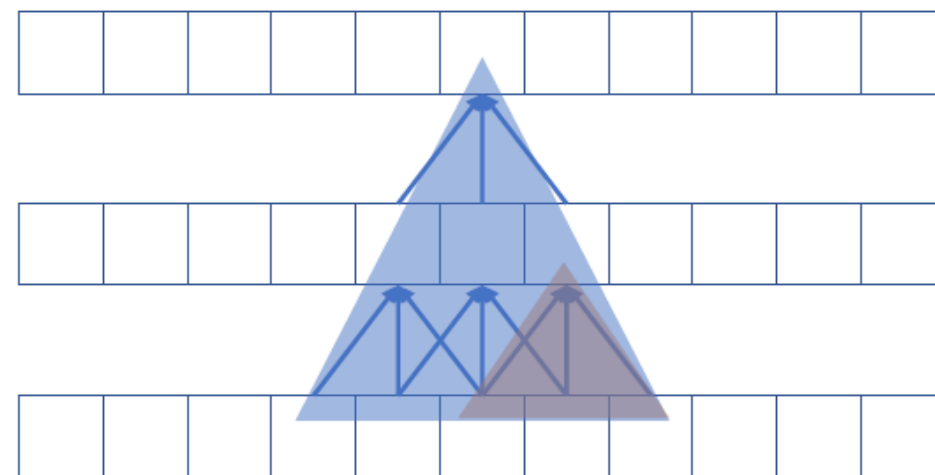
How to Represent a Sentence - CNN

- Convolutional Networks [Kim, 2014; Kalchbrenner et al., 2015]

- Captures k -grams hierarchically
- One 1-D convolutional layer: considers all k -grams

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

- 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 – 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
$$h_t = f(x_t, x_1) + \dots + f(x_t, x_{t-1}) + f(x_t, x_{t+1}) + \dots + f(x_t, x_T)$$
- CNN:
 - Each token's representation is computed against neighbouring tokens
$$h_t = f(x_t, x_{t-k}) + \dots + f(x_t, x_t) + \dots + f(x_t, x_{t+k})$$
- 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 $\mathbb{I}(S) = 1$, if S is true, and 0, 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!

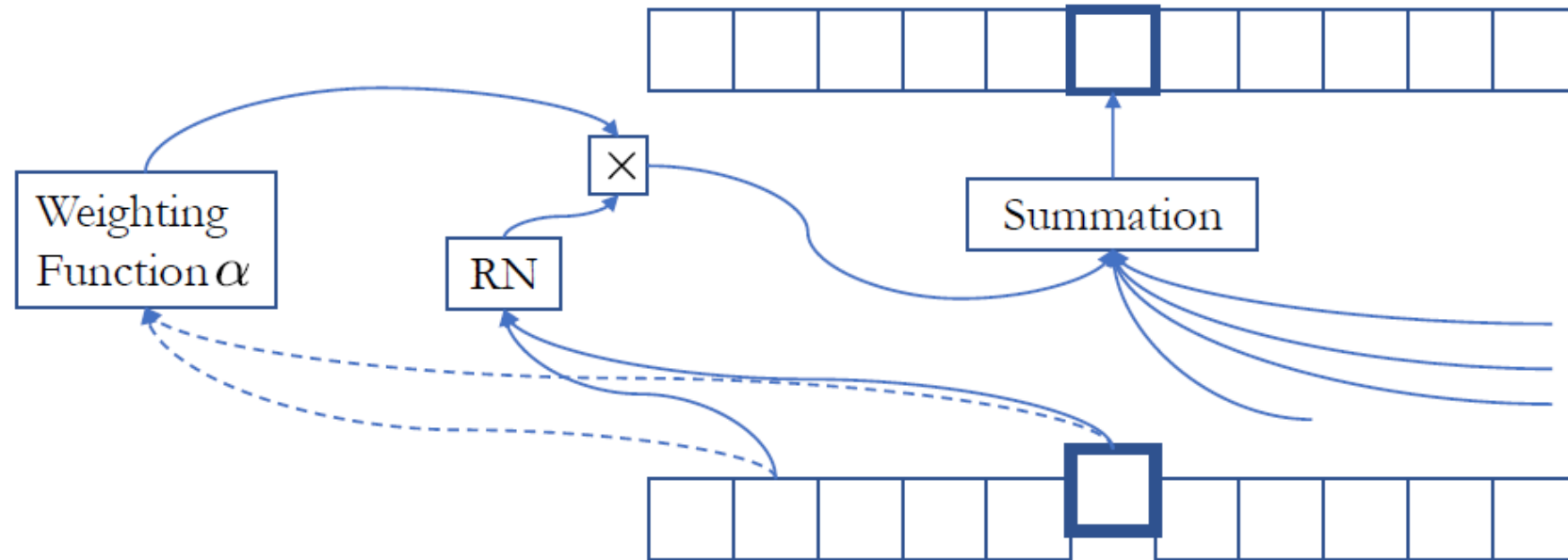
$$\alpha(x_t, x_{t'}) = \sigma(\text{RN}(x_t, x_{t'})) \in [0, 1]$$

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

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

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 – 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 $O(T)$

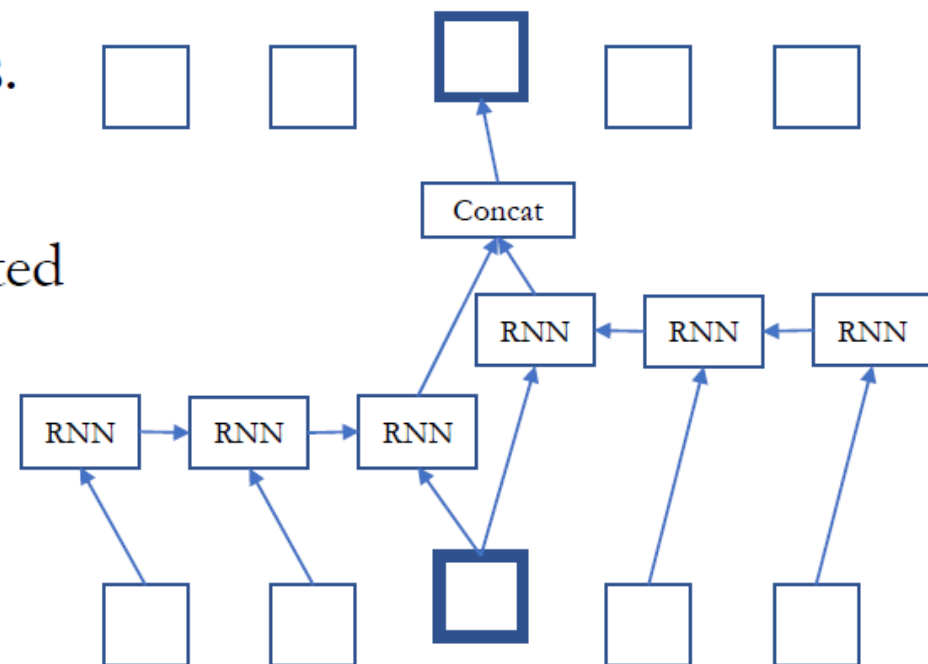
$h_t = \text{RNN}(h_{t-1}, x_t)$, where $h_0 = 0$.
- Memory h_t allows it to be Turing complete.*

How to Represent a Sentence – RNN

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

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

- 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.



Language Model

- Input: a sentence
- Output: the probability of the input sentence
- A language model captures the distribution over all possible sentences.
$$p(X) = p((x_1, x_2, \dots, x_T))$$
- Unlike text classification, it is *unsupervised learning*.
 - We will however turn the problem into a *sequence of supervised learning*.

Autoregressive Language Model

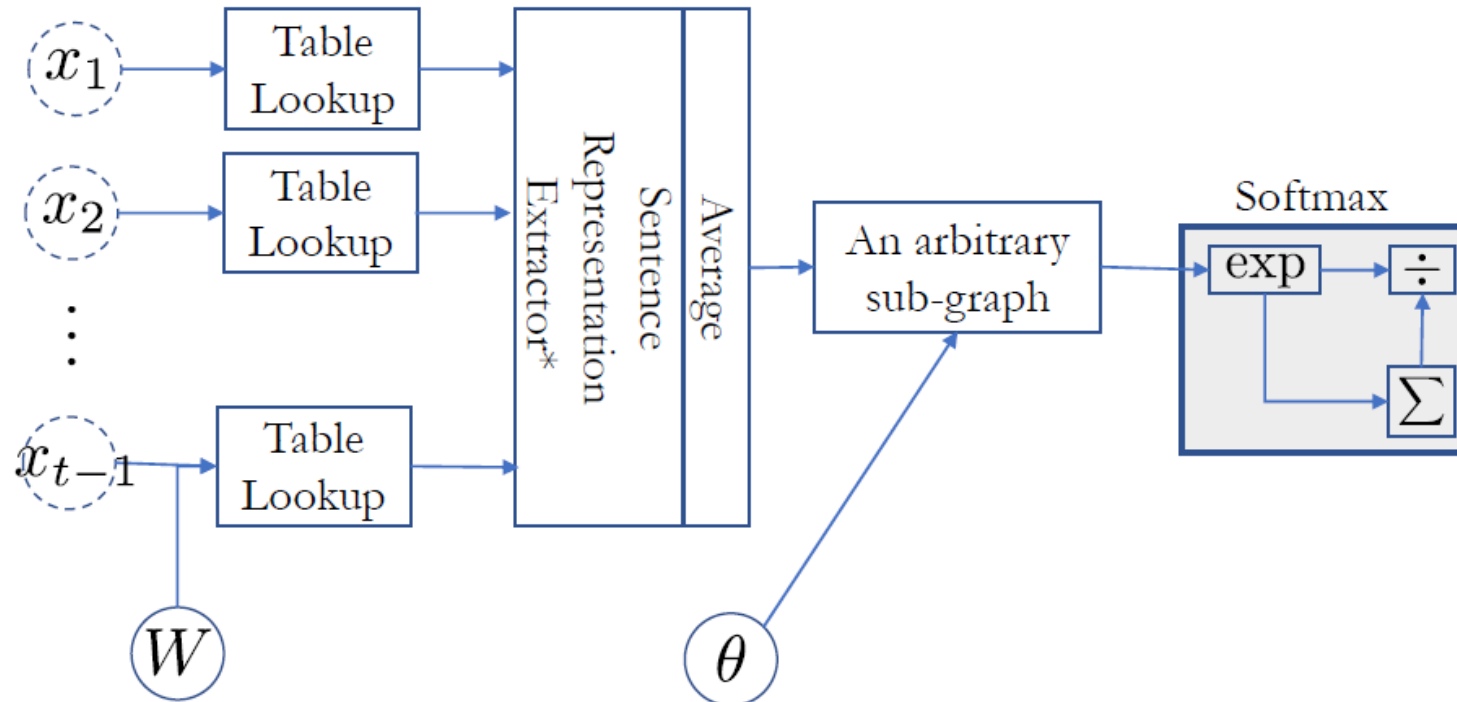
- Autoregressive sequence modelling
 - The distribution over the next token is based on all the previous tokens.
$$p(X) = p(x_1)p(x_2|x_1) \cdots p(x_T|x_1, \dots, x_{T-1})$$
 - This equality holds exactly due to the def. of conditional distribution*
- Unsupervised learning becomes a set of supervised problems.
 - Each conditional is a neural network classifier.
 - Input is all the previous tokens (a partial sentence).
 - Output is the distribution over all possible next tokens (classes).
 - It is a **text classification** problem.

Autoregressive Language Model

- Autoregressive sequence modelling
 - The distribution over the next token is based on all the previous tokens.

$$p(X) = p(x_1)p(x_2|x_1) \cdots p(x_T|x_1, \dots, x_{T-1})$$

- Each conditional is a sentence classifier:



N-Gram Language Model

- Let's back up a little...
- What would we do *without* a neural network?
- We need to estimate n -gram probabilities: $p(x|x_{-N}, x_{-N+1}, \dots, x_{-1})$
- Recall the def. of conditional and marginal probabilities:

$$\begin{aligned} p(x|x_{-N}, x_{-N+1}, \dots, x_{-1}) &= \frac{p(x_{-N}, x_{-N+1}, \dots, x_{-1}, x)}{p(x_{-N}, x_{-N+1}, \dots, x_{-1})} \\ &= \frac{p(x_{-N}, x_{-N+1}, \dots, x_{-1}, x)}{\sum_{x \in V} p(x_{-N}, x_{-N+1}, \dots, x_{-1}, x)} \end{aligned}$$

- V : all possible tokens (=vocabulary)

N-Gram Language Model

- We need to estimate n -gram probabilities:

$$p(x|x_{-N}, x_{-N+1}, \dots, x_{-1}) = \frac{p(x_{-N}, x_{-N+1}, \dots, x_{-1}, x)}{p(x_{-N}, x_{-N+1}, \dots, x_{-1})}$$

- Estimation:

$$\begin{aligned} p(x|x_{-N}, x_{-N+1}, \dots, x_{-1}) &= \frac{p(x_{-N}, x_{-N+1}, \dots, x_{-1}, x)}{\sum_{x \in V} p(x_{-N}, x_{-N+1}, \dots, x_{-1}, x)} \\ &\approx \frac{c(x_{-N}, x_{-N+1}, \dots, x_{-1}, x)}{\sum_{x' \in V} c(x_{-N}, x_{-N+1}, \dots, x_{-1}, x')} \end{aligned}$$

- *Do you see why this makes sense?*

N-Gram Language Model

- We need to estimate n-gram probabilities:

$$\begin{aligned} p(x|x_{-N}, x_{-N+1}, \dots, x_{-1}) &= \frac{p(x_{-N}, x_{-N+1}, \dots, x_{-1}, x)}{\sum_{x \in V} p(x_{-N}, x_{-N+1}, \dots, x_{-1}, x)} \\ &\approx \frac{c(x_{-N}, x_{-N+1}, \dots, x_{-1}, x)}{\sum_{x' \in V} c(x_{-N}, x_{-N+1}, \dots, x_{-1}, x')} \end{aligned}$$

- How likely is “University” given “New York”?
 - Count all “New York University”
 - Count all “New York ?”: e.g., “New York State”, “New York City”, “New York Fire”, “New York Police”, “New York Bridges”, ...
 - How often “New York University” happens among these?

N-Gram Language Model – Two Problems



1. Data sparsity: lack of generalization

- What happens “one” n-gram never happens?

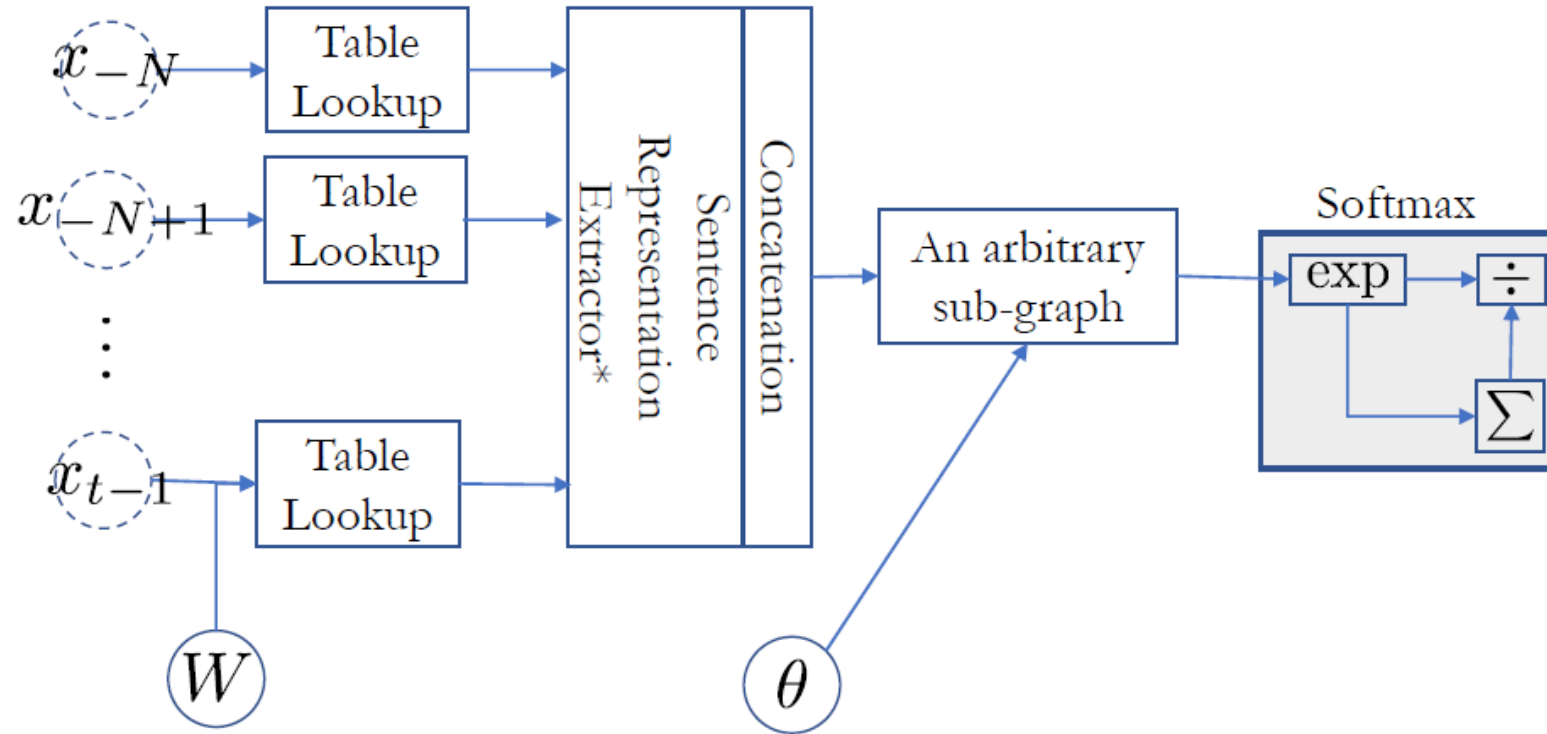
$$\begin{aligned} p(\text{a lion is chasing a llama}) &= p(\text{a}) \times p(\text{lion}|\text{a}) \times p(\text{is}|\text{a lion}) \\ &\quad \times p(\text{chasing}|\text{lion is}) \times p(\text{a}|\text{is chasing}) \\ &\quad \times \underbrace{p(\text{llama}|\text{chasing a})}_{=0} = 0 \end{aligned}$$

2. Inability to capture long-term dependencies

- Each conditional only considers a small window of size n .
- Consider “the same **stump** which had impaled the car of many a guest in the past thirty years and which *he refused to have* **removed**”
- It is impossible to tell “removed” is likely by looking at the four preceding tokens.

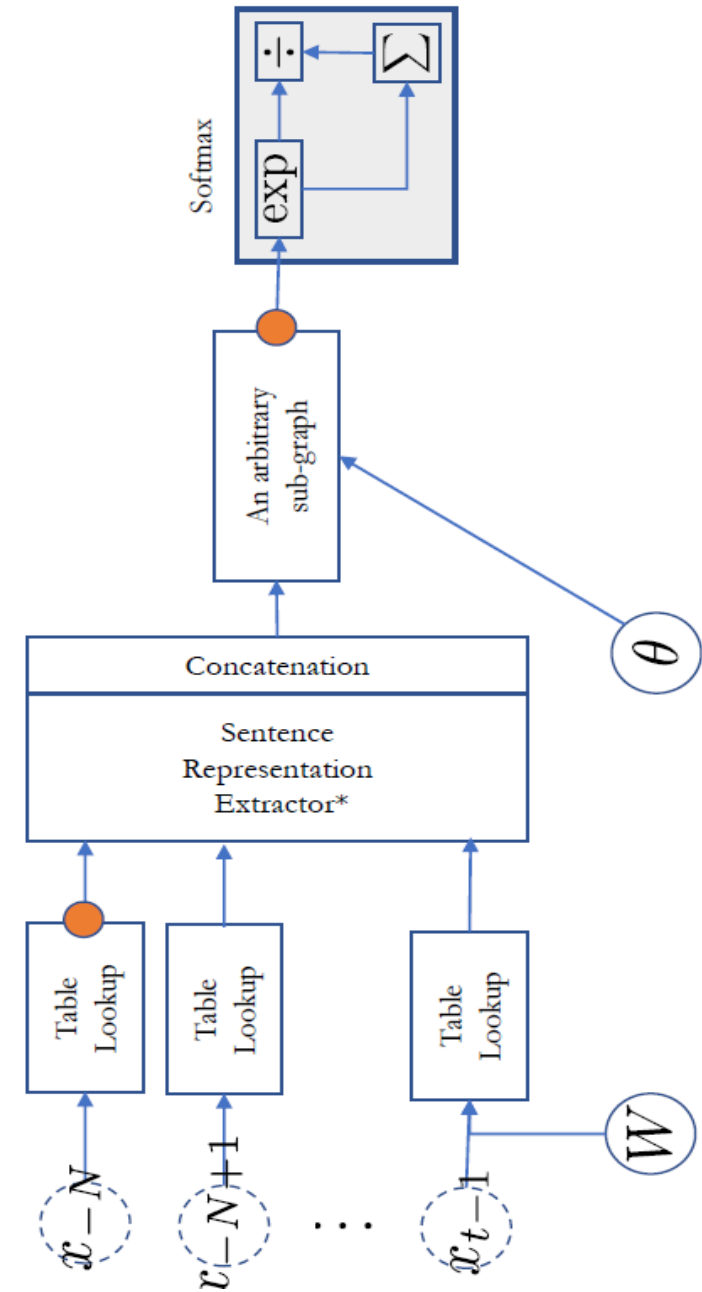
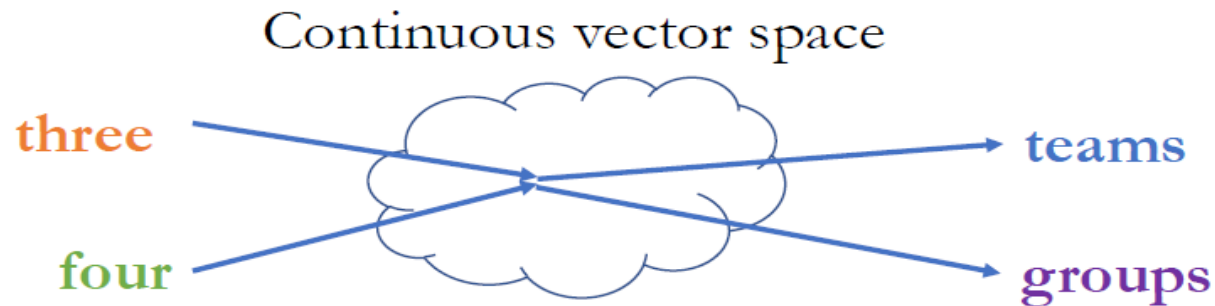
Neural N-Gram Language Model

- The first extension of n-gram language models using a neural network



Neural N-Gram Language Model

- Training examples
 - there are **three** **teams** left for qualification.
 - **four** **teams** have passed the first round.
 - **four** **groups** are playing in the field.
- Q: how likely is “groups” followed by “three”?



Neural N-Gram Language Model

- In practice,
 1. Collect all n-grams from the corpus.
 2. Shuffle all the n-grams to build a training set
 3. Train the neural n-gram language model using stochastic gradient descent on minibatches containing 100-1000 n-grams.
 4. Early-stop based on the validation set.
 5. Report perplexity on the test set.

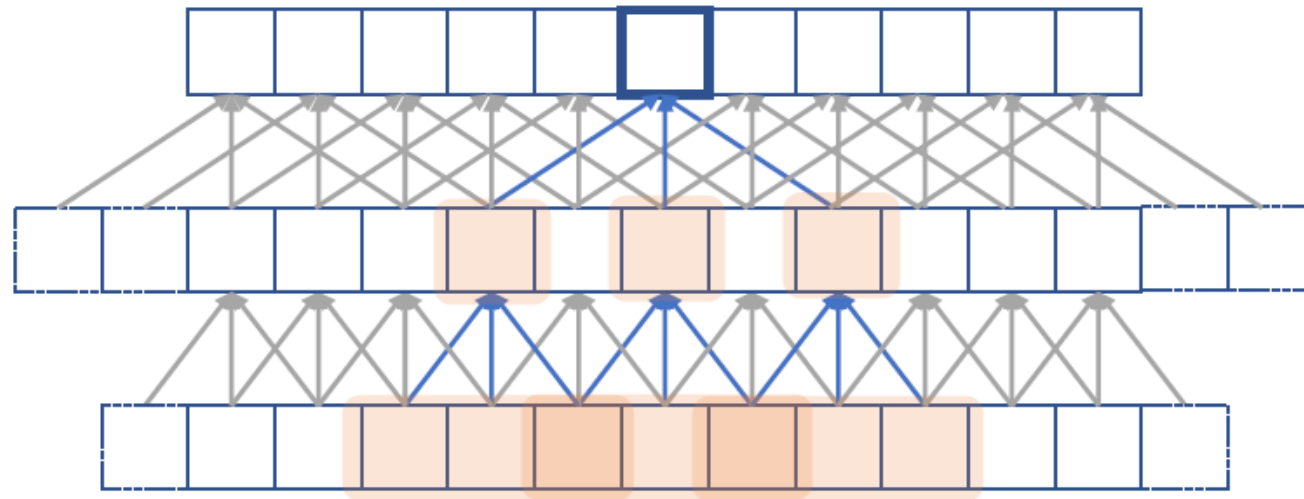
$$\text{ppl} = b^{\frac{1}{|D|} \sum_{(x_1, \dots, x_N) \in D} \log_b p(x_N | x_1, \dots, x_{N-1})}$$

Increasing the Context Size

- Convolutional Language Model

[Kalchbrenner et al., 2015; Dauphin et al., 2016]

- Dilated convolution to rapidly increase the window size
 - Exponential-growth of the window by introducing a multiplicative factor
 - By carefully selecting the multiplicative factor, no loss in the information.



Infinite Context : $n \rightarrow \infty$

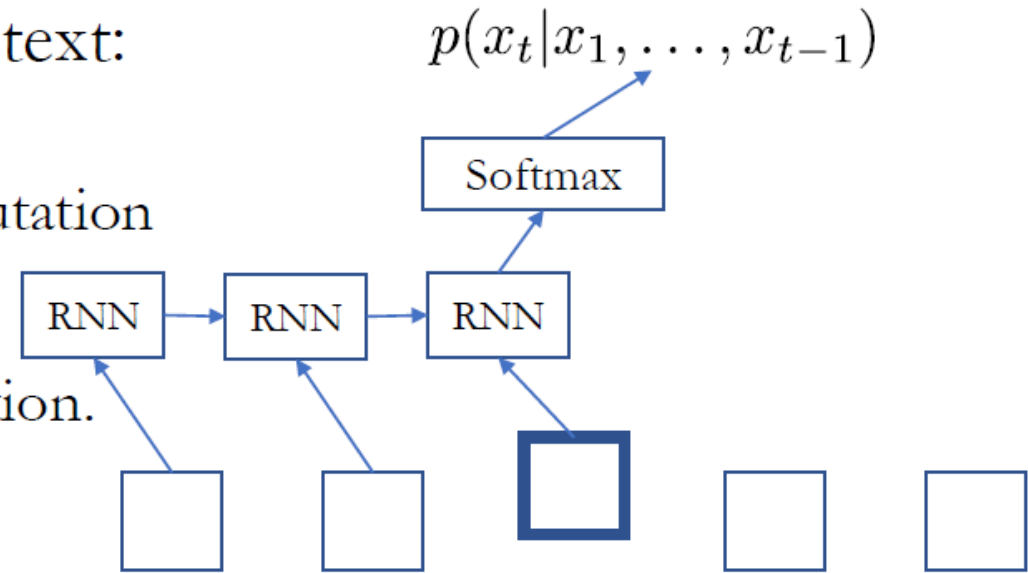
- CBoW Language Model

- Equivalent to the neural LM after replacing “concat” with “average”
 - “Averaging” allows the model to consider the infinite large context window.
- Extremely efficient, but a weak language model
 - Ignores the order of the tokens in the context windows.
 - Any language with a fixed order cannot be modelled well.
 - Averaging ignores the absolute counts, which may be important:
 - If the context window is larger, “verb” becomes less likely in SVO languages.

Infinite Context : $n \rightarrow \infty$

- Recurrent Language Model

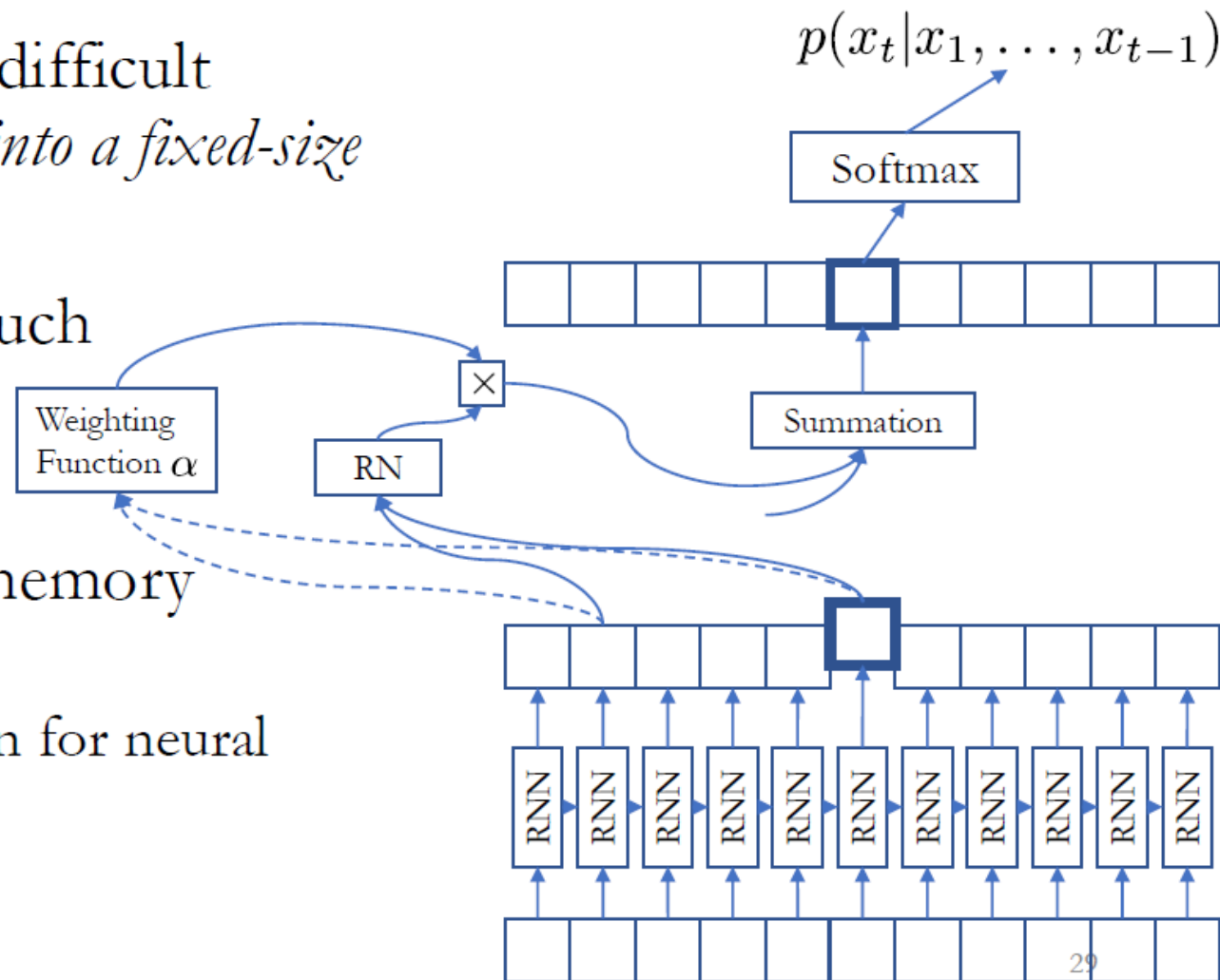
- A recurrent network summarizes all the tokens so far.
- Use the recurrent network's memory to predict the next token.
- Efficient online processing of a streaming text:
 - Constant time per step.
 - Constant memory throughout forward computation
- Useful in practice:
 - Useful for autocomplete and keyword suggestion.
 - Scoring partial hypotheses in generation.



Infinite Context : $n \rightarrow \infty$

- Recurrent Language Model

- The **recurrent network** solves a difficult problem: *compress the entire context into a fixed-size memory vector*.
- **Self-attention** does not require such compression but still can capture long-term dependencies.
- Combine these two: a recurrent memory network (RMN) [Tran et al., 2016]
 - RNMT+: a similar, recent extension for neural machine translation



Machine Translation

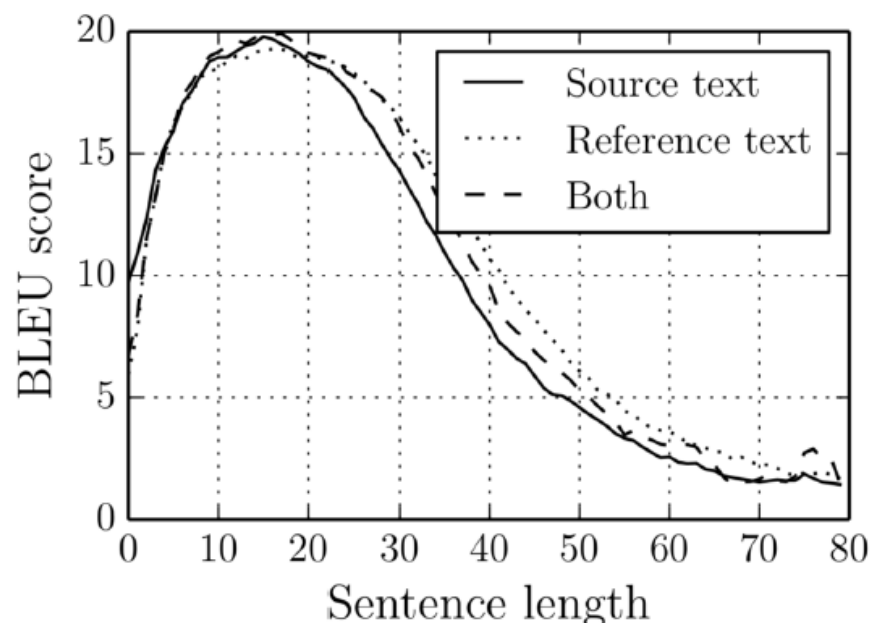
- Input: a sentence written in a source language L_S
- Output: a corresponding sentence in a target language L_T
- Problem statement:
 - Supervised learning: given the input sentence, output its translation
 - Compute the conditional distribution over all possible translation given the input
$$p(Y = (y_1, \dots, y_T) | X = (x_1, \dots, x_{T'}))$$
- *We have already learned every necessary ingredient for building a full neural machine translation system.*

Encoder – Source Sentence Representation

- Encode the source sentence into a set of sentence representation vectors
 - # of encoded vectors is proportional to the source sentence length: often same.
 $H = (h_1, \dots, h_{T'})$
 - Recurrent networks have been widely used [Cho et al., 2014; Sutskever et al., 2014], but CNN [Gehring et al., 2017; Kalchbrenner&Blunsom, 2013] and self-attention [Vaswani et al., 2017] are used increasingly more often. See Lecture 2 for details.
- We do not want to collapse them into a single vector.
 - Collapsing often corresponds to information loss.
 - Increasingly more difficult to encode the entire source sentence into a single vector, as the sentence length increases [Cho et al., 2014b].
 - We didn't know initially until [Bahdanau et al., 2015].

Encoder – Source Sentence Representation

- Encode the source sentence into a set of sentence representation vectors
- We do not want to collapse them into a single vector.
 - Increasingly more difficult to encode the entire source sentence into a single vector, as the sentence length increases [Cho et al., 2014b].



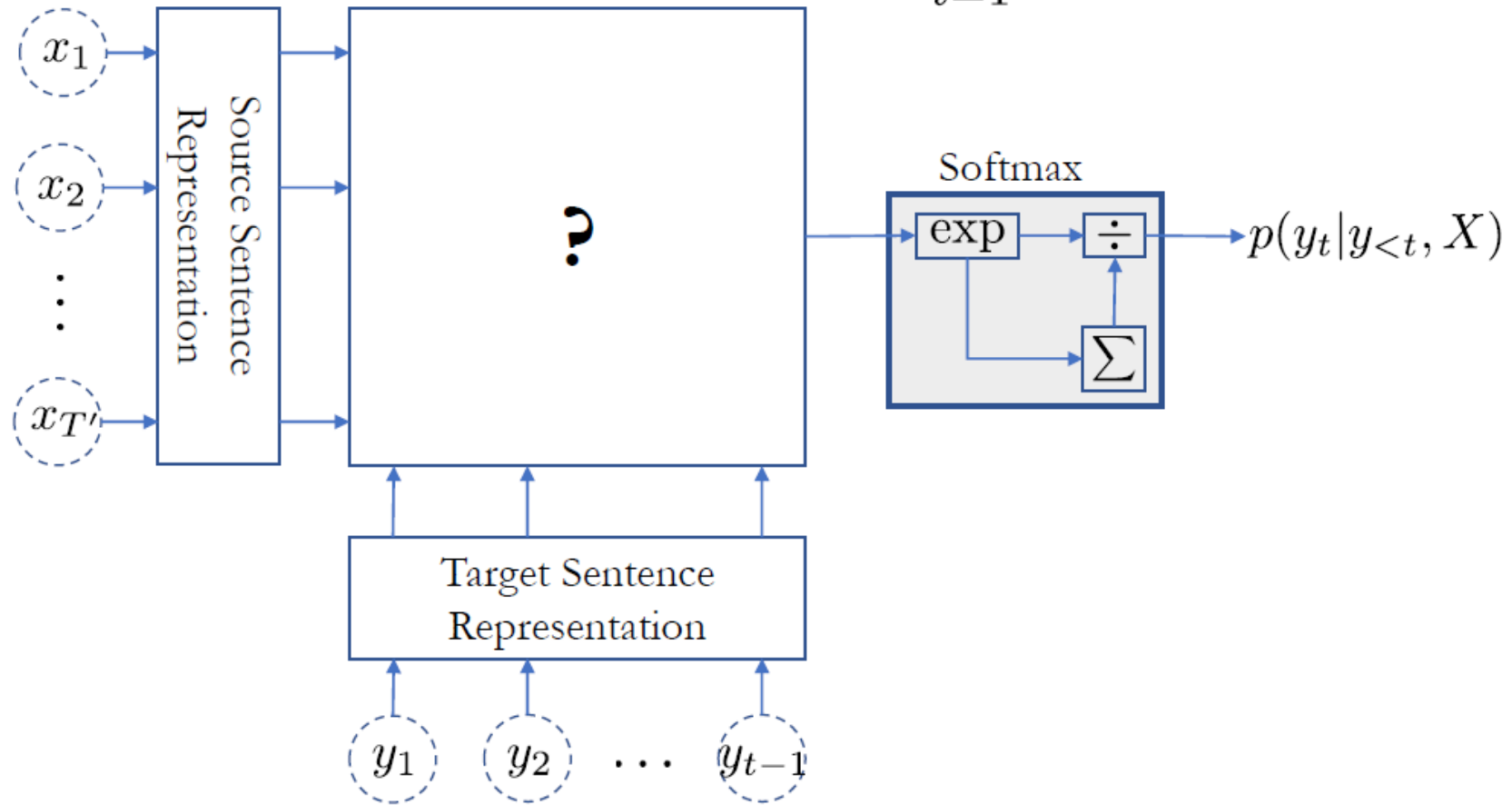
Decoder – Language Modelling

- Autoregressive Language modelling with an infinite context $n \rightarrow \infty$
 - Larger context is necessary to generate a coherent sentence.
 - Semantics could be largely provided by the source sentence, but syntactic properties need to be handled by the language model directly.
 - Recurrent networks, self-attention and (dilated) convolutional networks
 - Causal structure must be followed.
 - See Lecture 3.
- **Conditional** Language modelling
 - The context based on which the next token is predicted is **two-fold**

$$p(Y|X) = \prod_{t=1}^T p(y_t | y_{<t}, X)$$

Decoder – Language Modelling

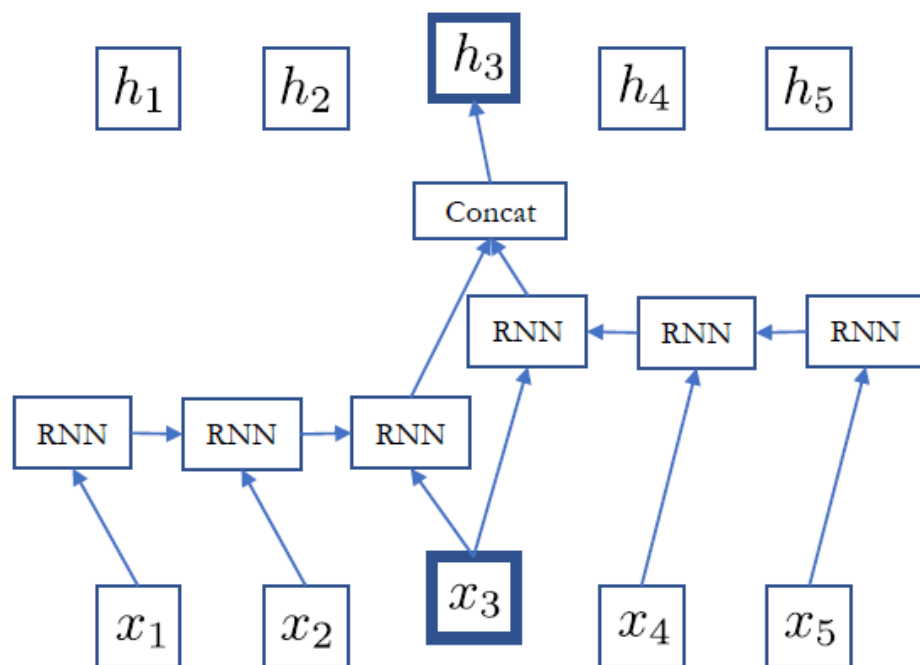
- Conditional Language modelling $p(Y|X) = \prod_{t=1}^T p(y_t|y_{<t}, X)$



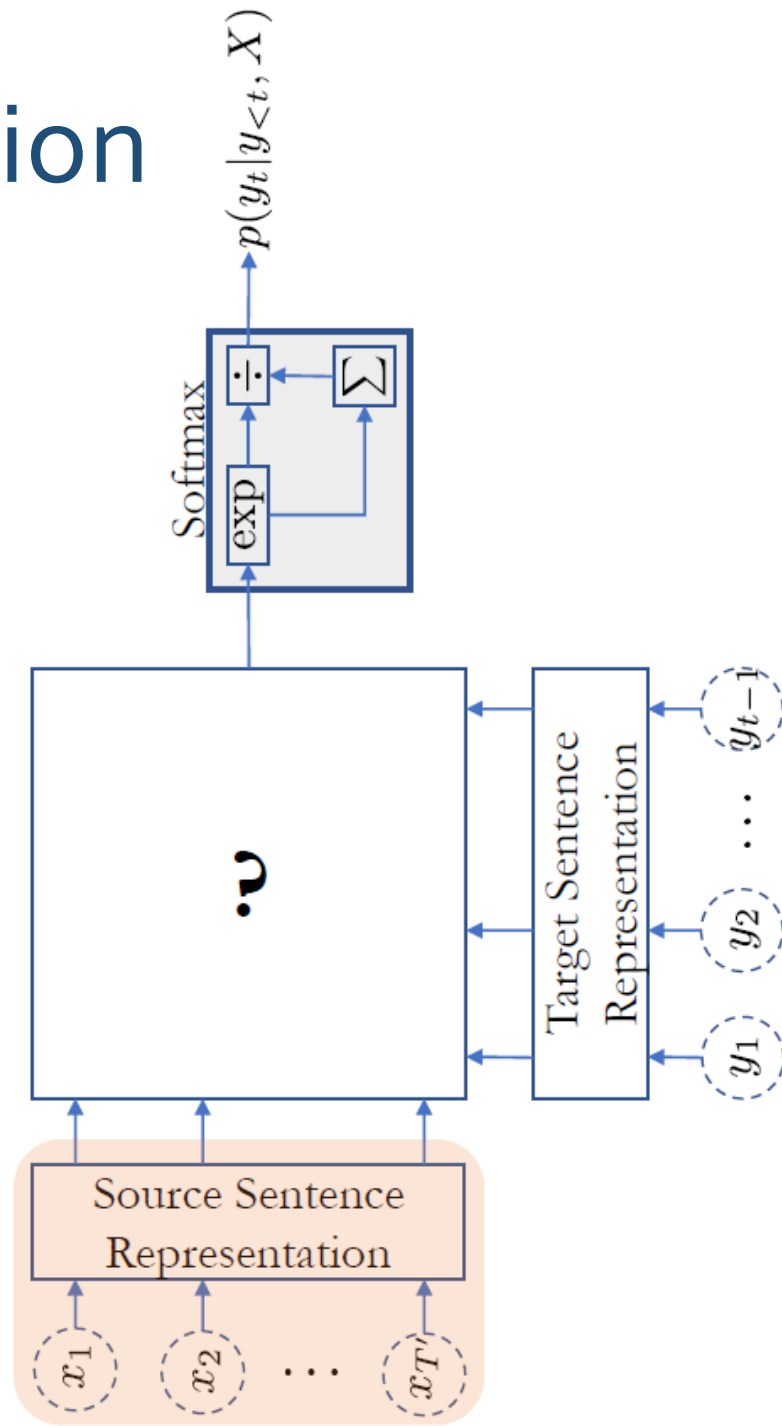
RNN Neural Machine Translation

1. Source sentence representation

- A stack of bidirectional RNN's



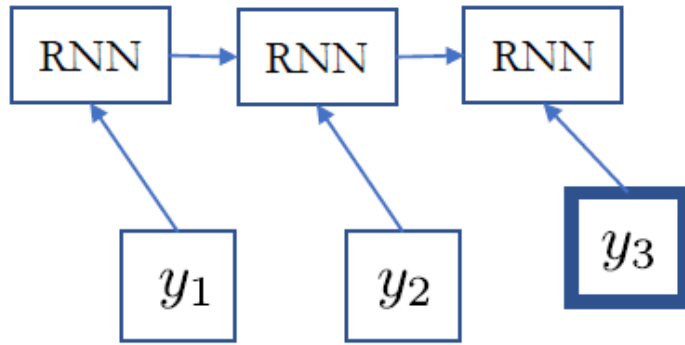
- The extracted vector at each location is a **context-dependent vector representation**.



RNN Neural Machine Translation

2. Target prefix representation

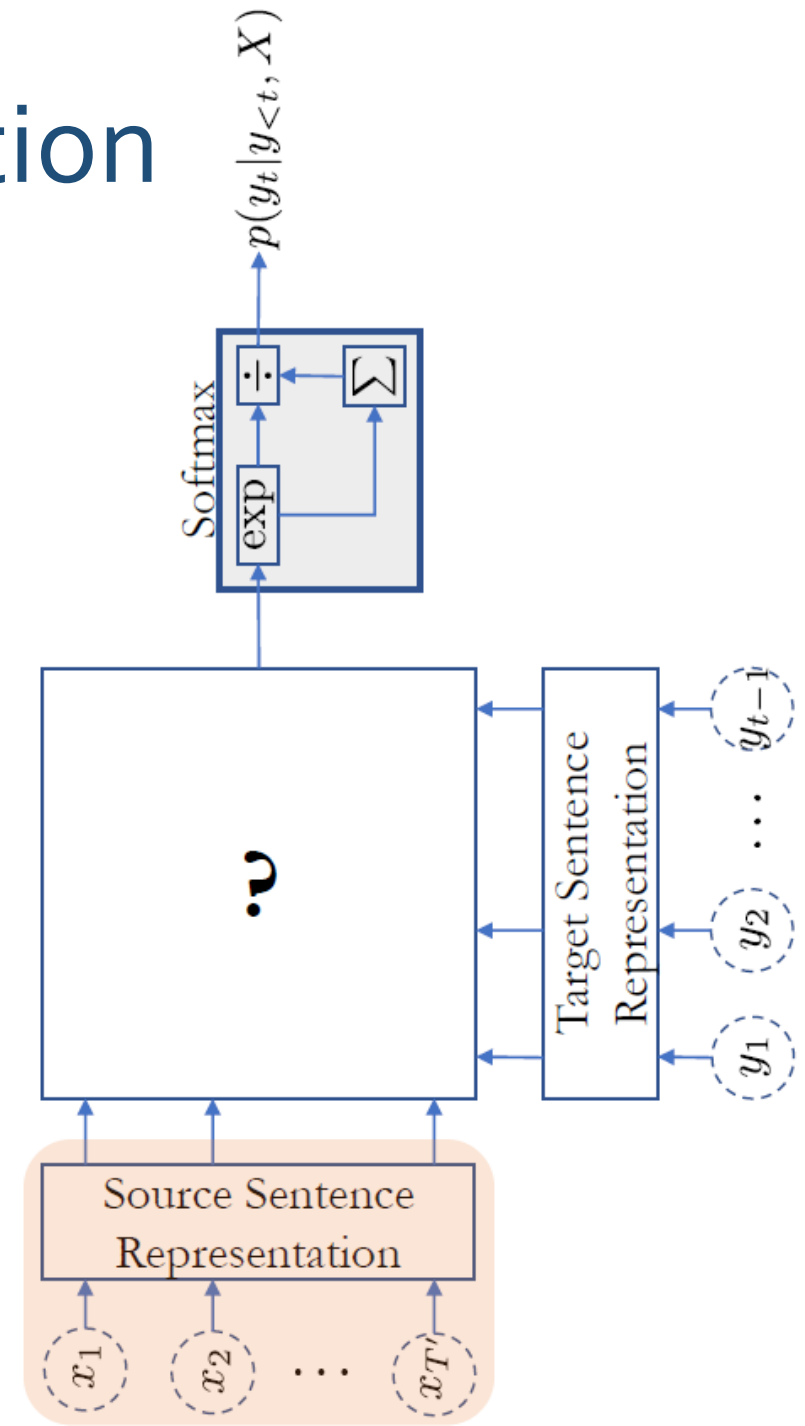
- A unidirectional recurrent network



- Compression of the target prefix

$$z_t = \text{RNN}_{\text{decoder}}(z_{t-1}, y_{t-1})$$

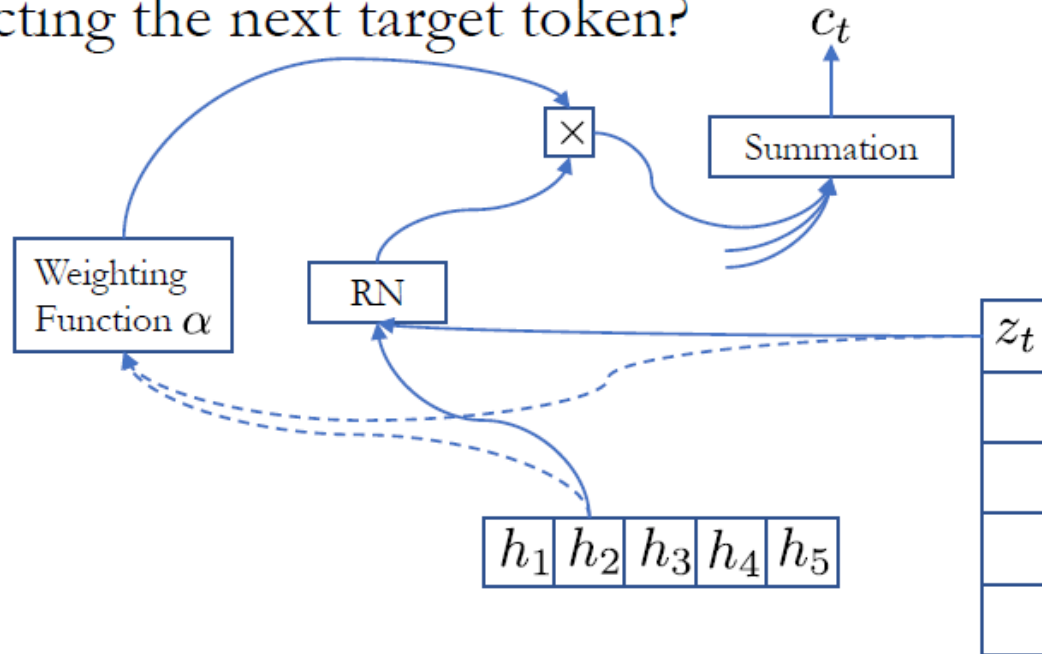
- Summarizes what has been translated so far



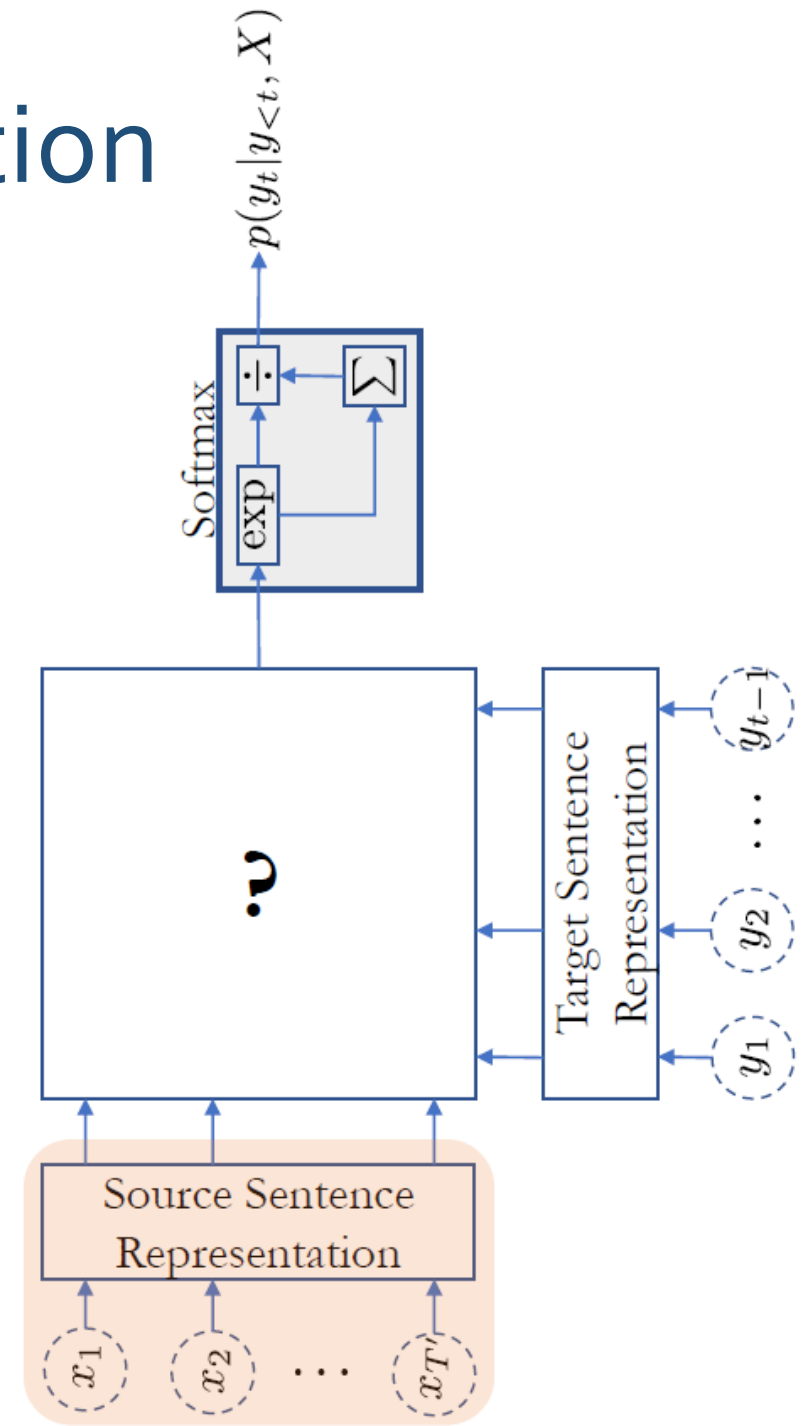
RNN Neural Machine Translation

3. Attention mechanism

- Which part of the source sentence is relevant for predicting the next target token?

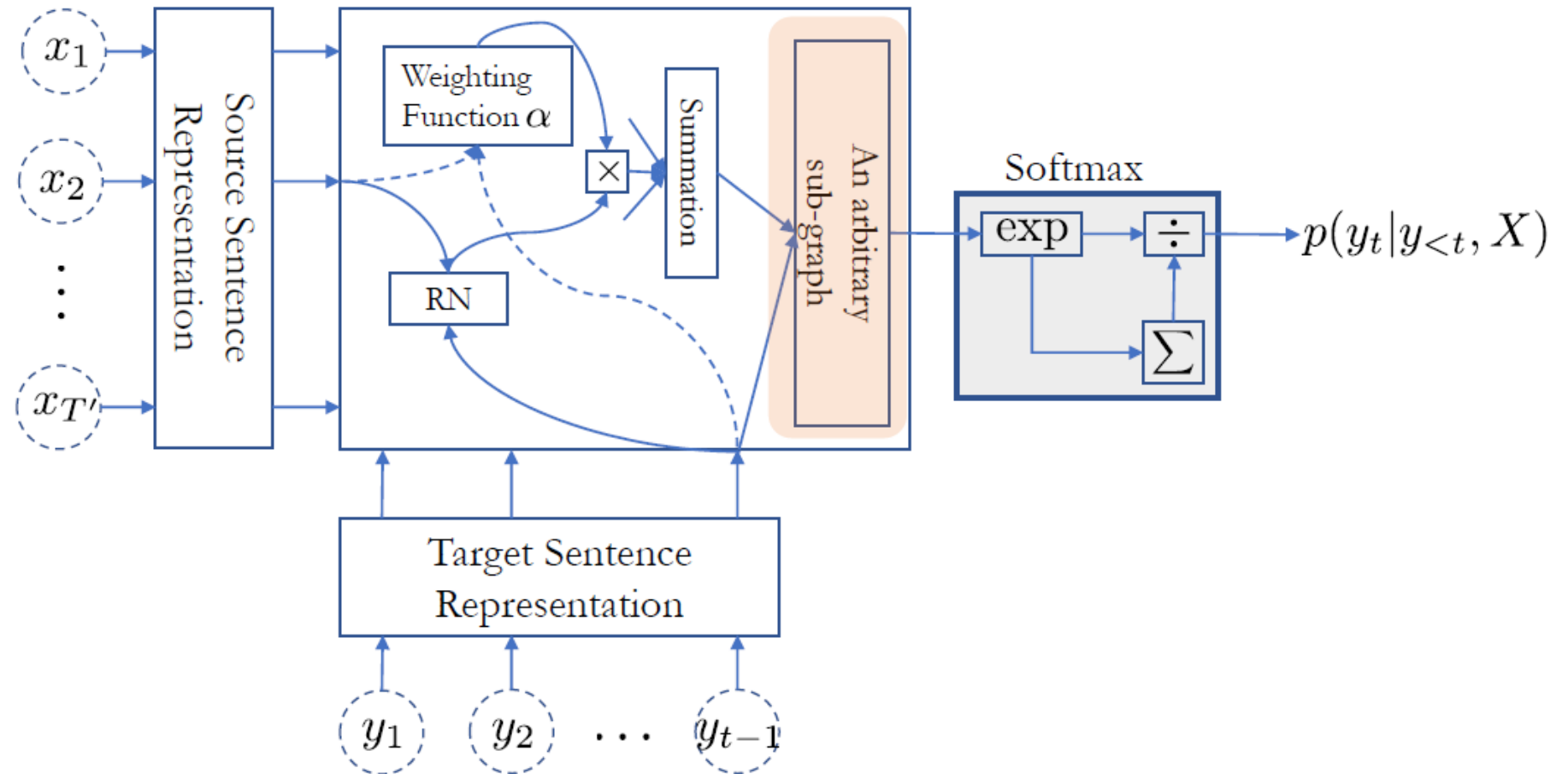


- Time-dependent source context vector c_t

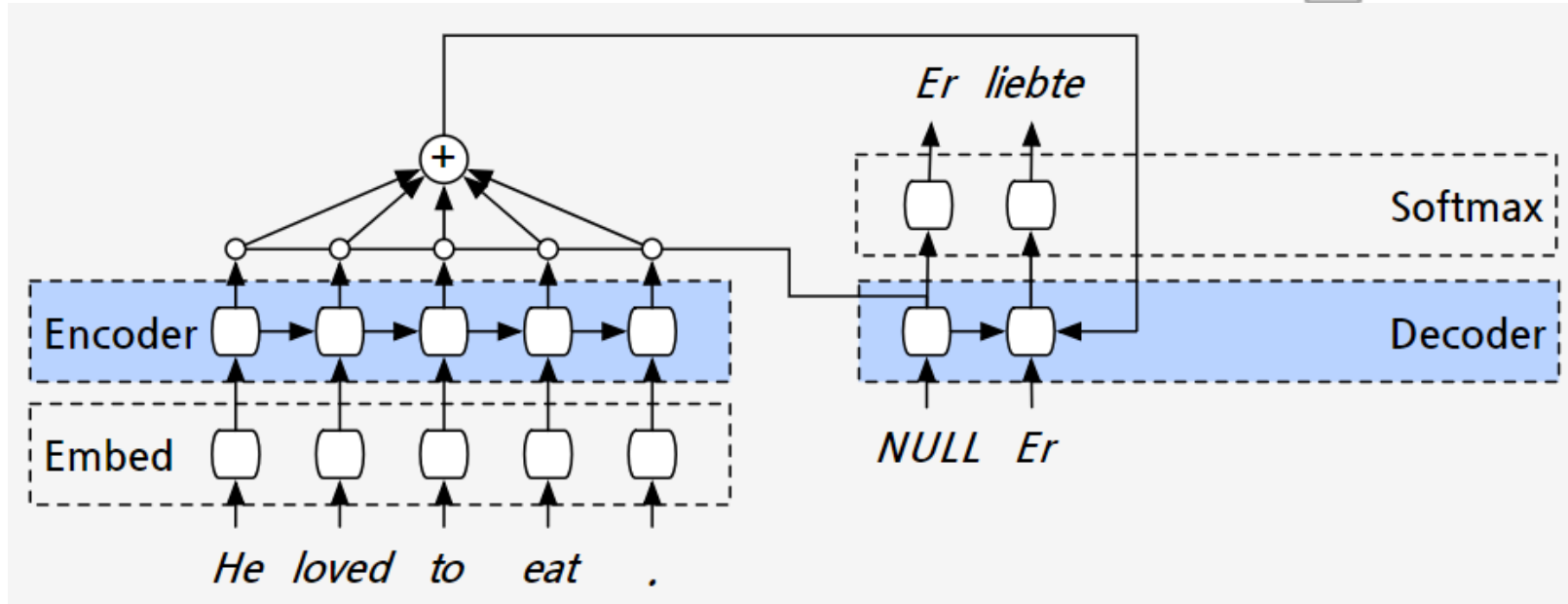
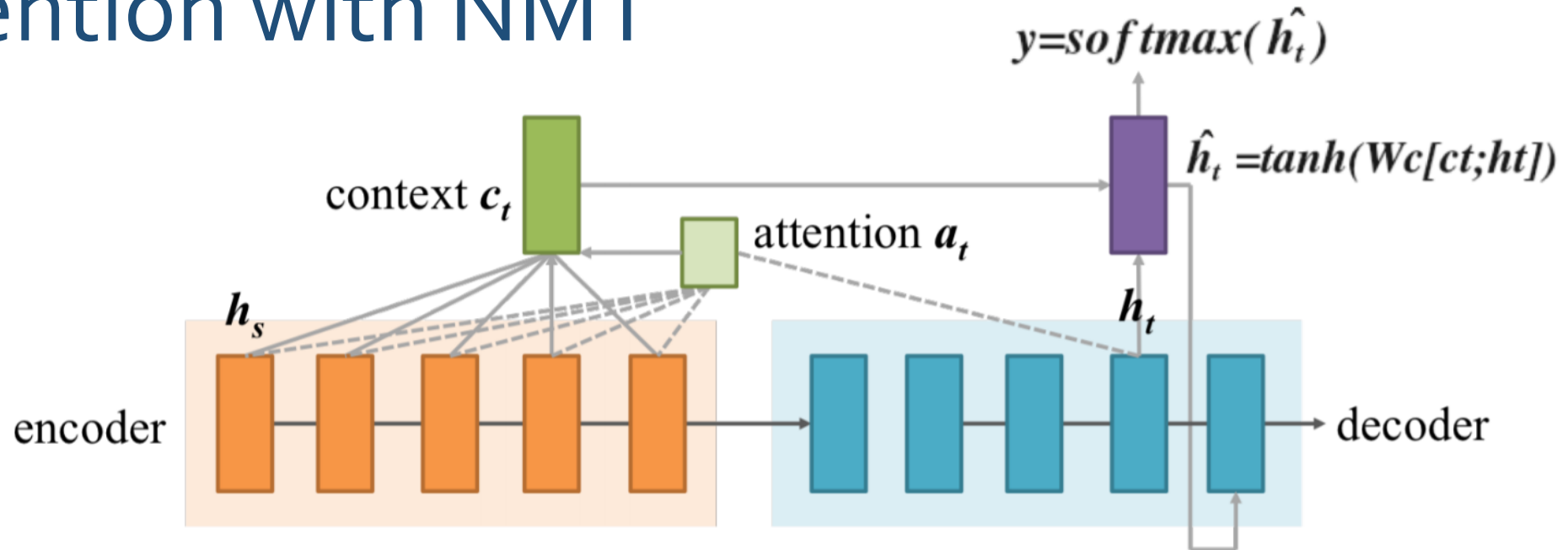


RNN Neural Machine Translation

4. Fuse the source context vector and target prefix vector
- Combines z_t and c_t into a single vector



Attention with NMT



RNN Neural Machine Translation

- **Source:** *An admitting privilege is the right of a doctor to admit a patient to a hospital or a medical centre to carry out a diagnosis or a procedure, based on his status as a health care worker at a hospital.*
- **When collapsed:** *Un privilège d'admission est le droit d'un médecin de reconnaître un patient à l'hôpital ou un centre médical d'un diagnostic ou de prendre un diagnostic en fonction de son état de santé.*
- **RNNSearch:** *Un privilège d'admission est le droit d'un médecin d'admettre un patient à un hôpital ou un centre médical pour effectuer un diagnostic ou une procédure, selon son statut de travailleur des soins de santé à l'hôpital.*

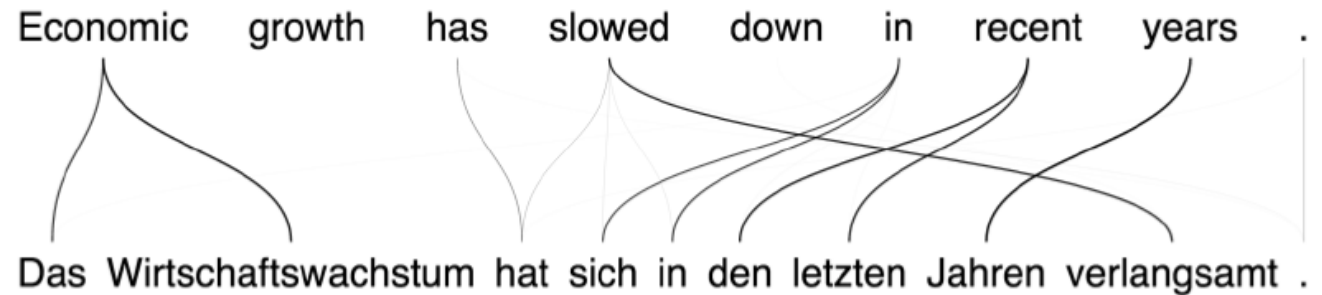
RNN Neural Machine Translation

- Sensible alignment between source and target tokens
- Capture long-range reordering/dependencies
- Without strong supervision on the alignment
 - Weakly supervised learning

English-French

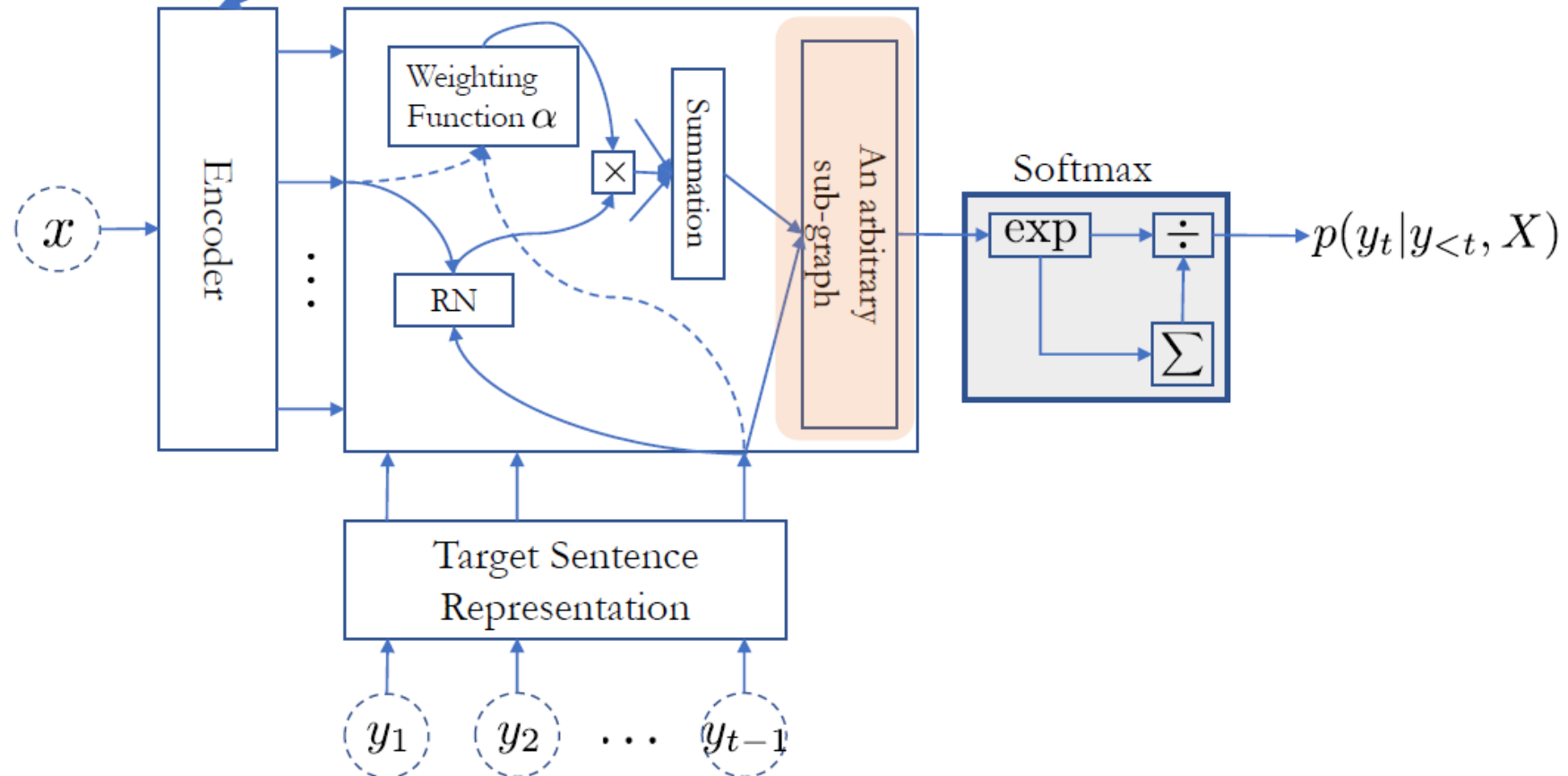


English-German



RNN Neural Machine Translation

- Input: arbitrary as long as encoded into a set of continuous vectors
- Output: a corresponding sentence in a target language



What Does NMT Do?

- Sequence of “discrete” symbols \rightarrow Set of “continuous” vectors
- Continuous vectors *encode* semantics of discrete symbols
- Continuous vectors are *stripped* of hard, linguistic symbols
- *Can we map multiple languages on a single continuous space?*

Continuous space
representation of
a sentence

0.32 0.82 -0.87
0.80 -0.22 1.36

Encoder

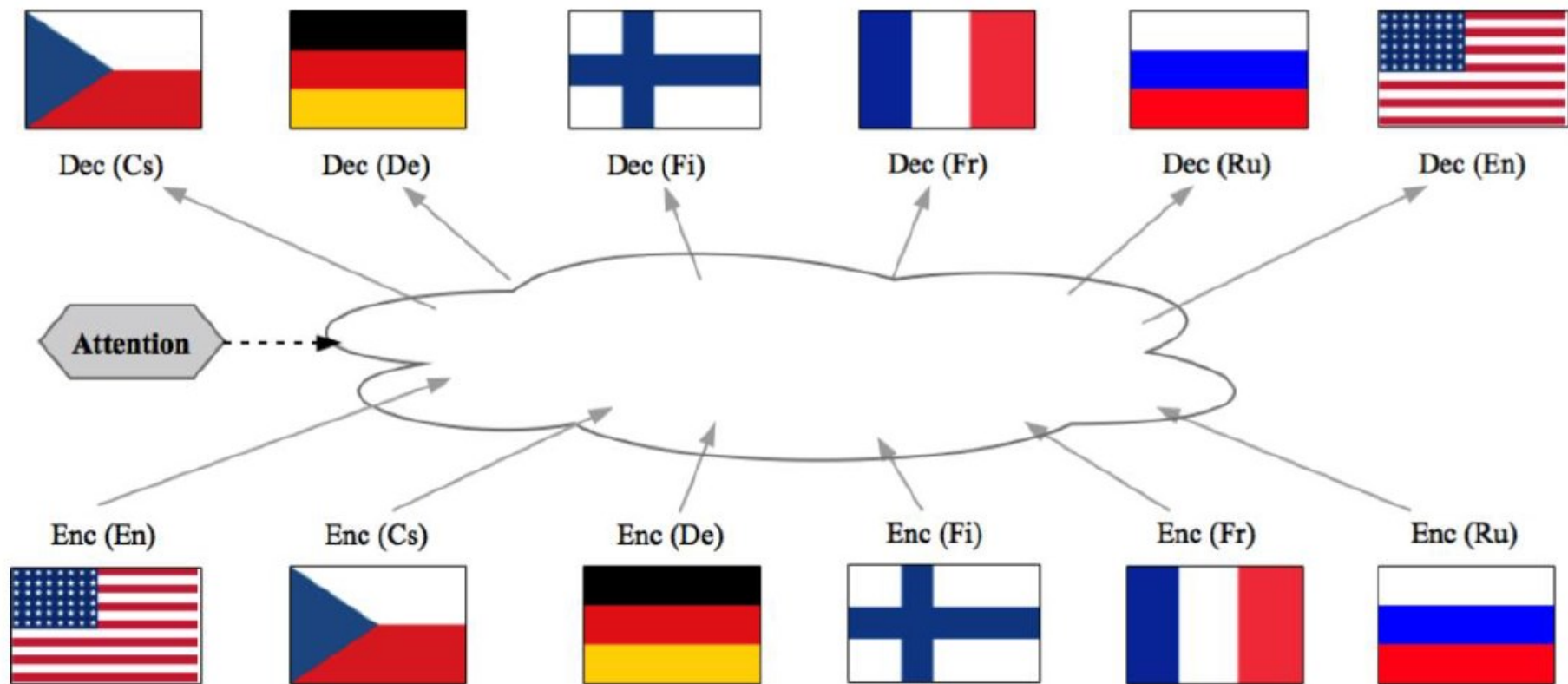
(177, 737, 62, 153, 4)

(he, walk, ed, out, .)

Decoder

(그, 는, 밖, 으로, 나갔, 다, .)

Multilingual Neural Machine Translation



- Can this continuous vector space be shared across multiple languages?

Character-Level, Multilingual Translation

- Robust to intra-sentence code switching
- Huge saving in parameters: 4x less parameters without loss in BLEU

(e) Multilingual

Multi src	Bei der Metropolitního výboru pro dopravu für das Gebiet der San Francisco Bay erklärten Beamte , der Kongress könne das Problem банкротство доверительного Фонда строительства шоссейных дорог einfach durch Erhöhung der Kraftstoffsteuer lösen .
EN ref	At the Metropolitan Transportation Commission in the San Francisco Bay Area , officials say Congress could very simply deal with the bankrupt Highway Trust Fund by raising gas taxes .
bpe2char	During the Metropolitan Committee on Transport for San Francisco Bay , officials declared that Congress could solve the problem of bankruptcy by increasing the fuel tax bankrupt .
char2char	At the Metropolitan Committee on Transport for the territory of San Francisco Bay , officials explained that the Congress could simply solve the problem of the bankruptcy of the Road Construction Fund by increasing the fuel tax .

Image Caption Generation

- Input: an image
- Output: an image caption
- Network Architecture
 - Encoder: deep convolution network
 - Decoder: recurrent language model with the attention mechanism.
- Data: image-caption pairs

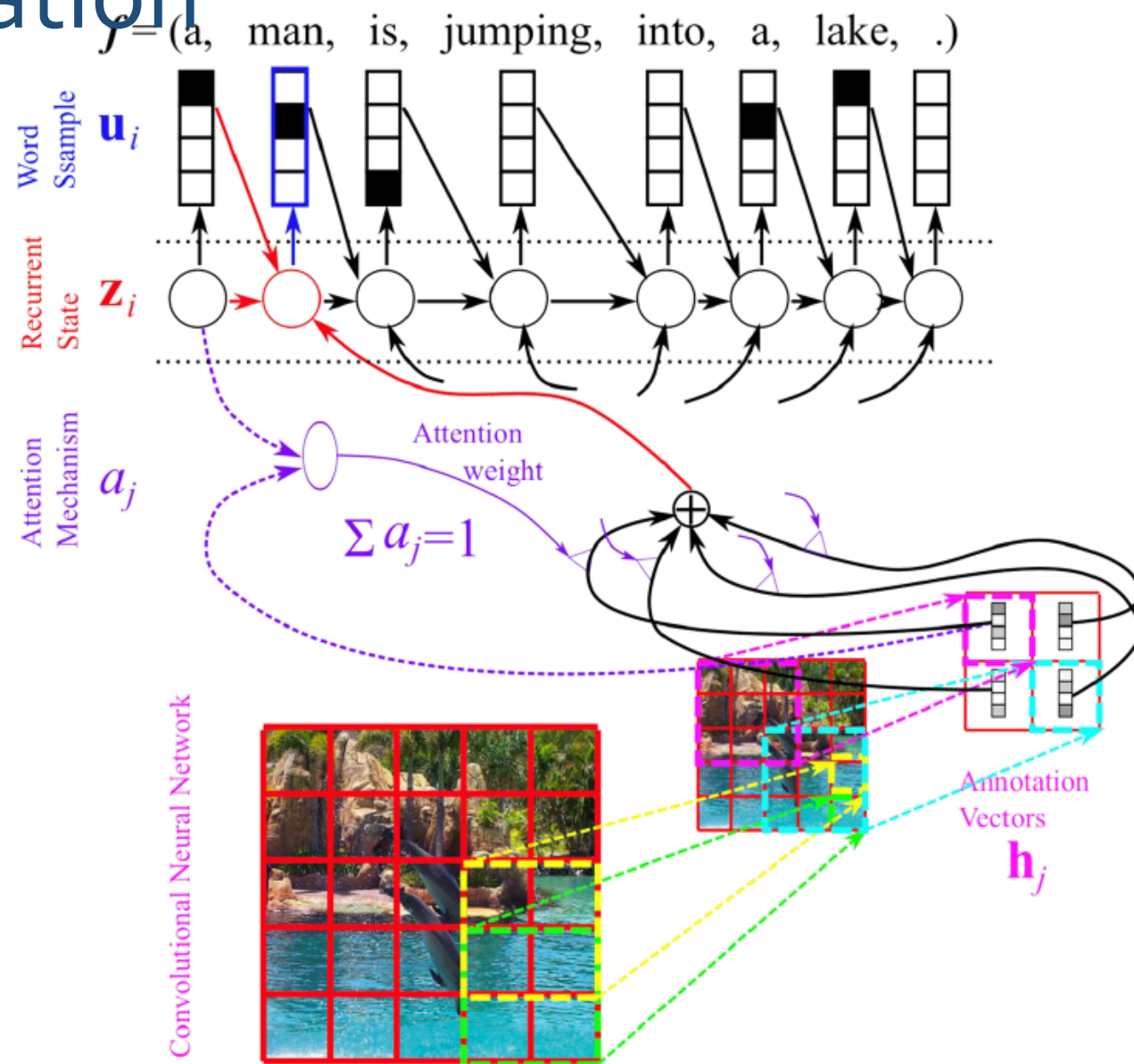


Image Caption Generation



A woman is throwing a frisbee in a park.



A dog is standing on a hardwood floor.



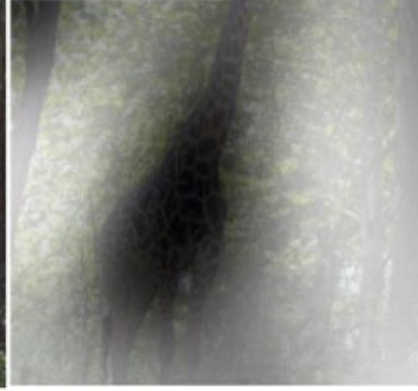
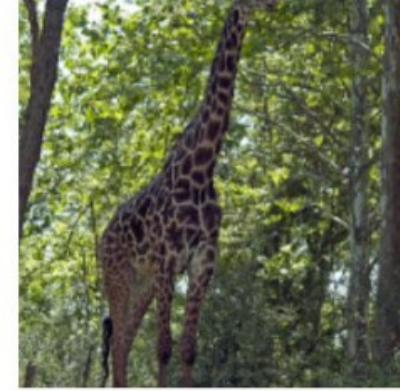
A stop sign is on a road with a mountain in the background.



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.

Speech Recognition

- Input: Speech
- Output: transcription

