

# データサイエンス 『実践コース』

## 数理工学PBL

---

Day 2-2 : アテンション

一関高専 小池 敦

# 概要

---

- 目標

- 文の内容を把握するような深層学習モデルを作る

- 内容

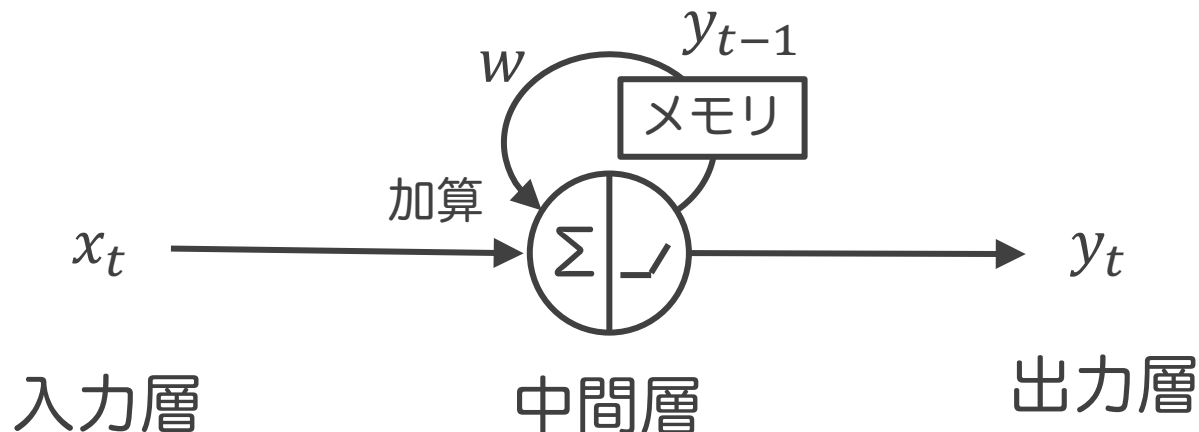
- 再帰型ニューラルネットワーク (RNN)
- エンコーダー・デコーダーモデル
- アテンション
- セルフアテンションとTransformer

# 再帰型ニューラル ネットワーク

---

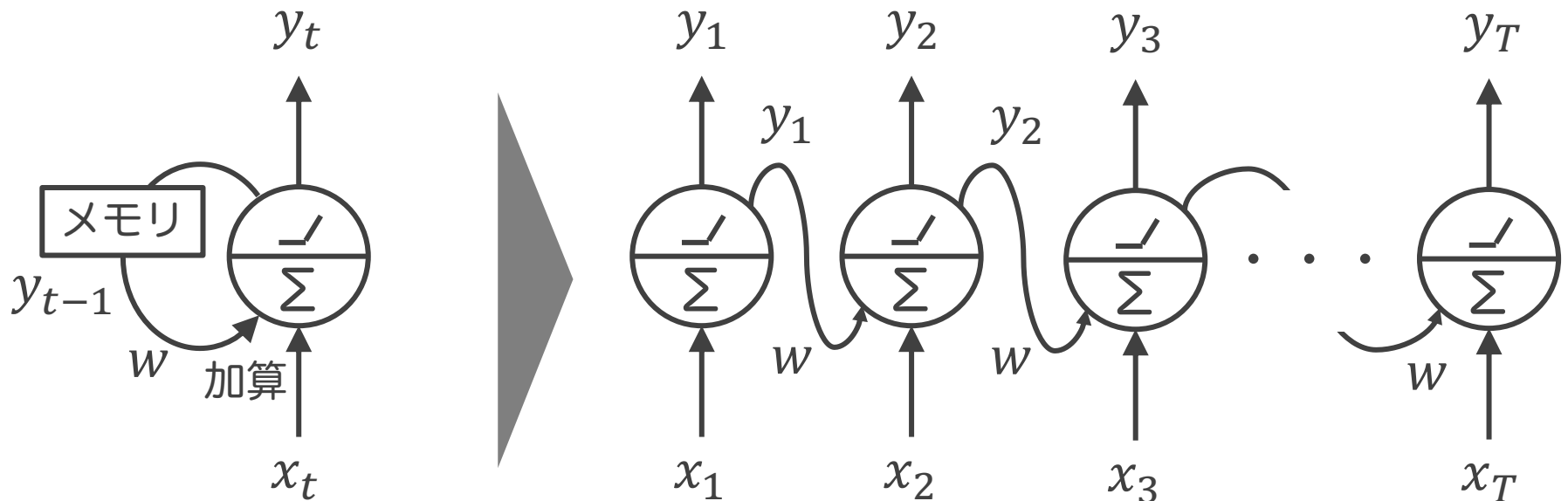
# 再帰型ニューラルネットワーク (RNN)

- 内部に循環構造を持つニューラルネットワーク
- タイムステップごとにひとつの入力
  - ユニットはメモリを持ち、前の時刻の値を出力
- 可変長の時系列データ（や自然言語）を扱える



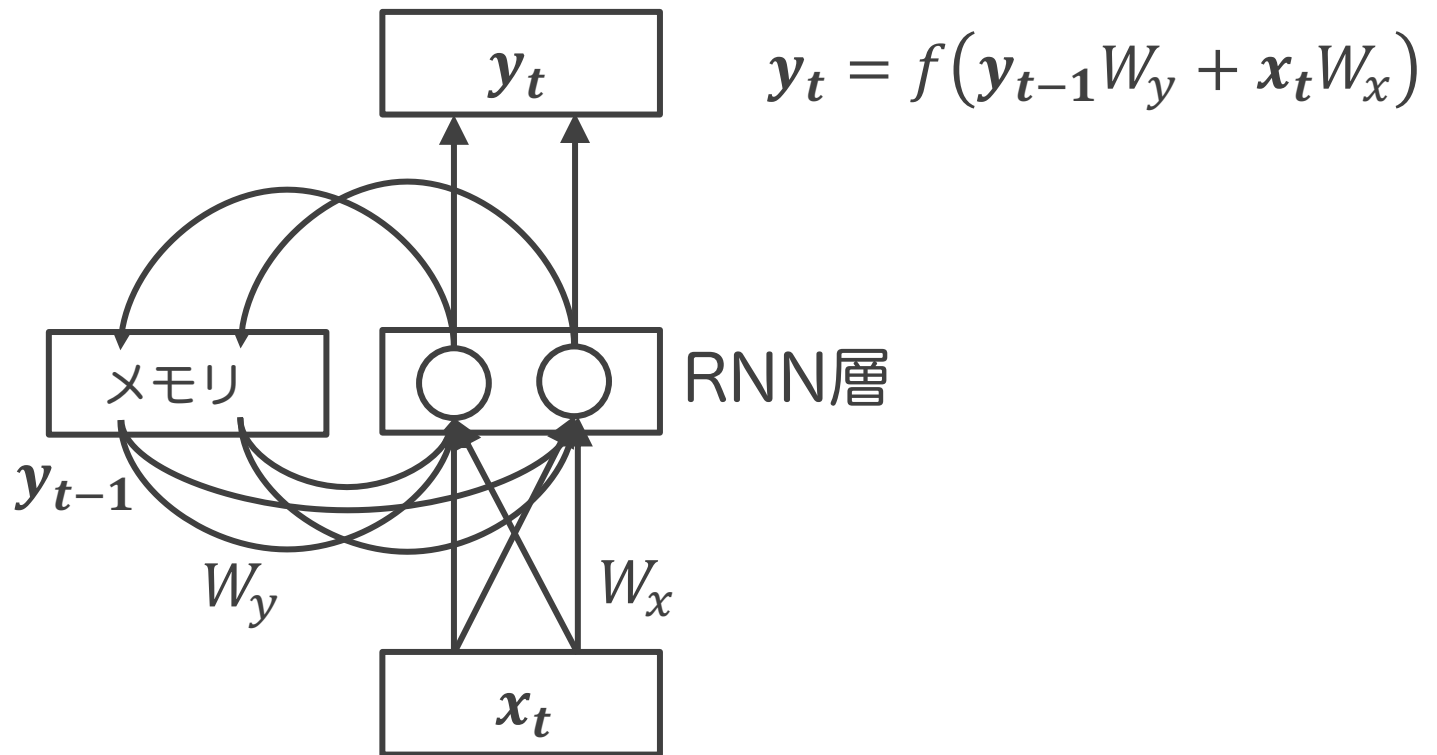
# RNNのアンロール

- RNNを巡回構造を使わず表現

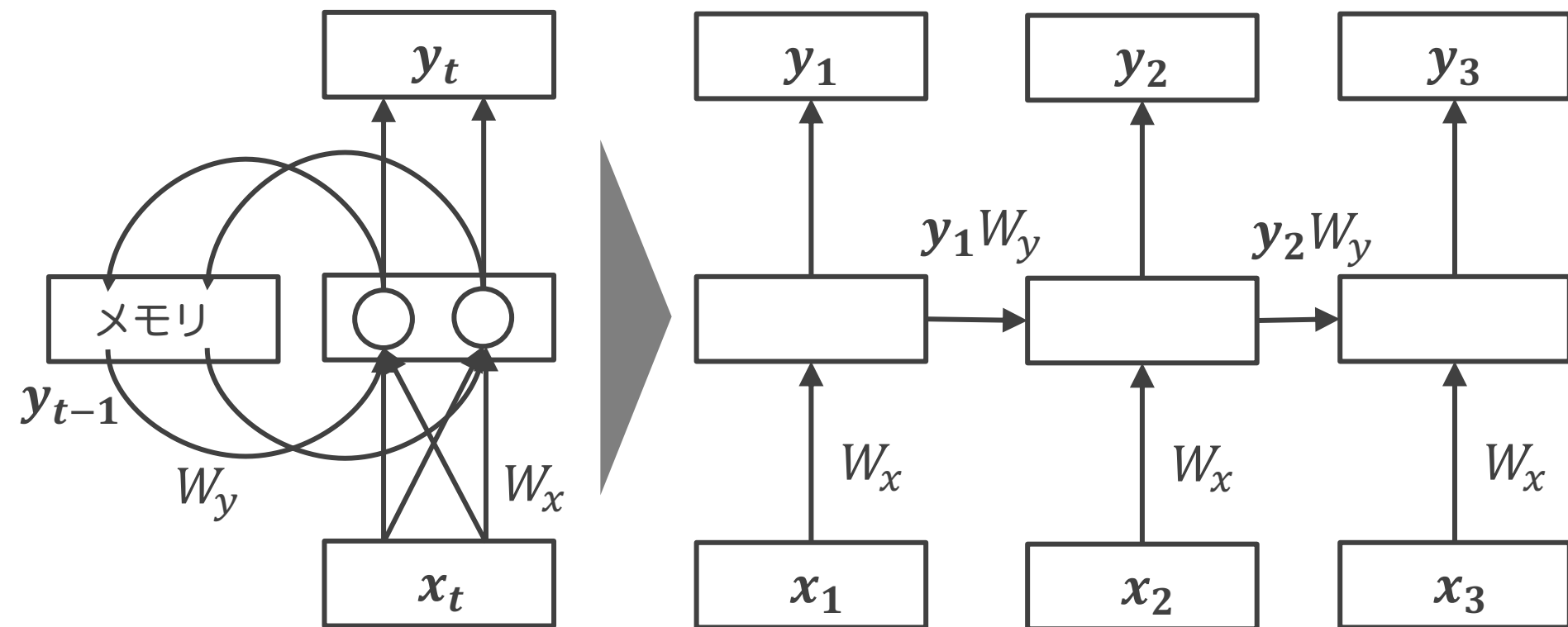


# RNNの行列表現

---

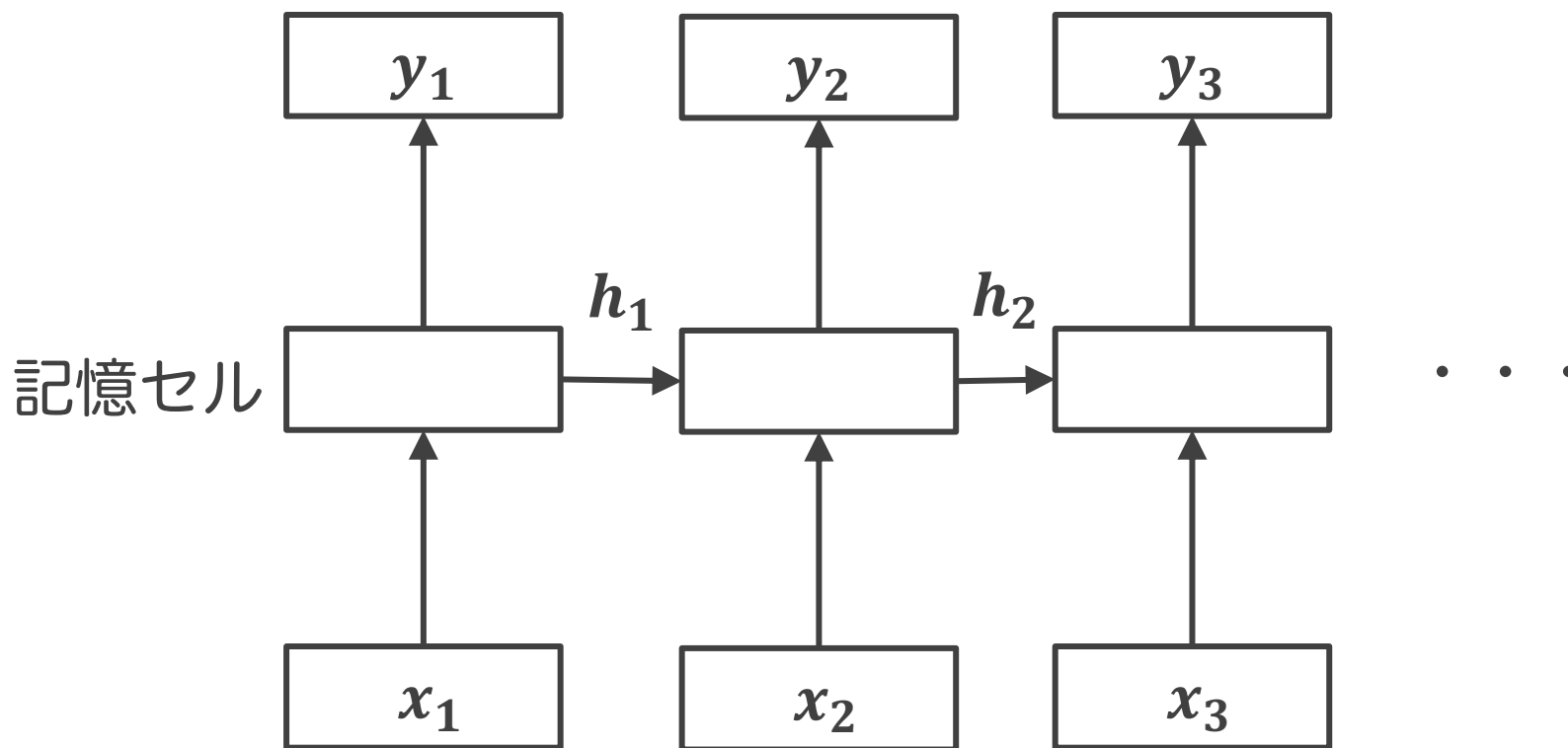


# RNN層のアンロール



# 記憶セル

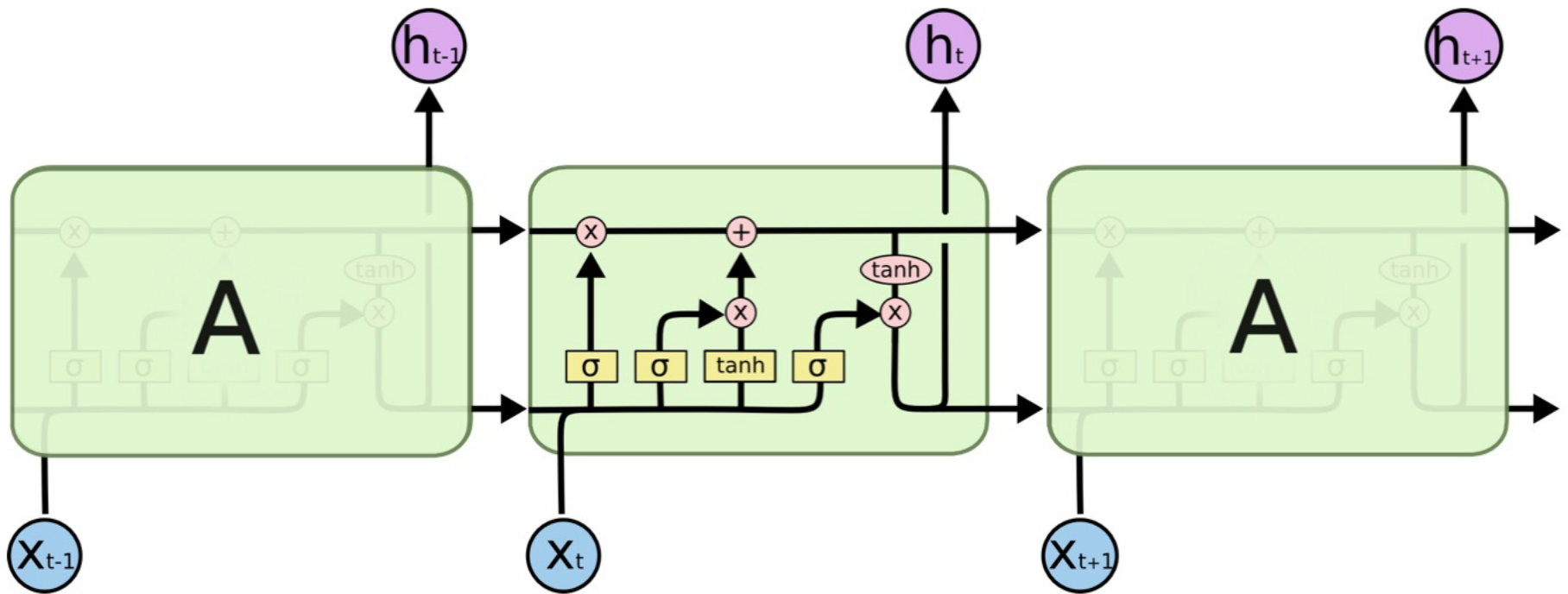
- $f(y_{t-1}W_y + x_tW_x)$ を一般化して $h_{t-1}$ を再帰させる





# LSTM

- 長期記憶に優れた記憶セル（詳細省略）

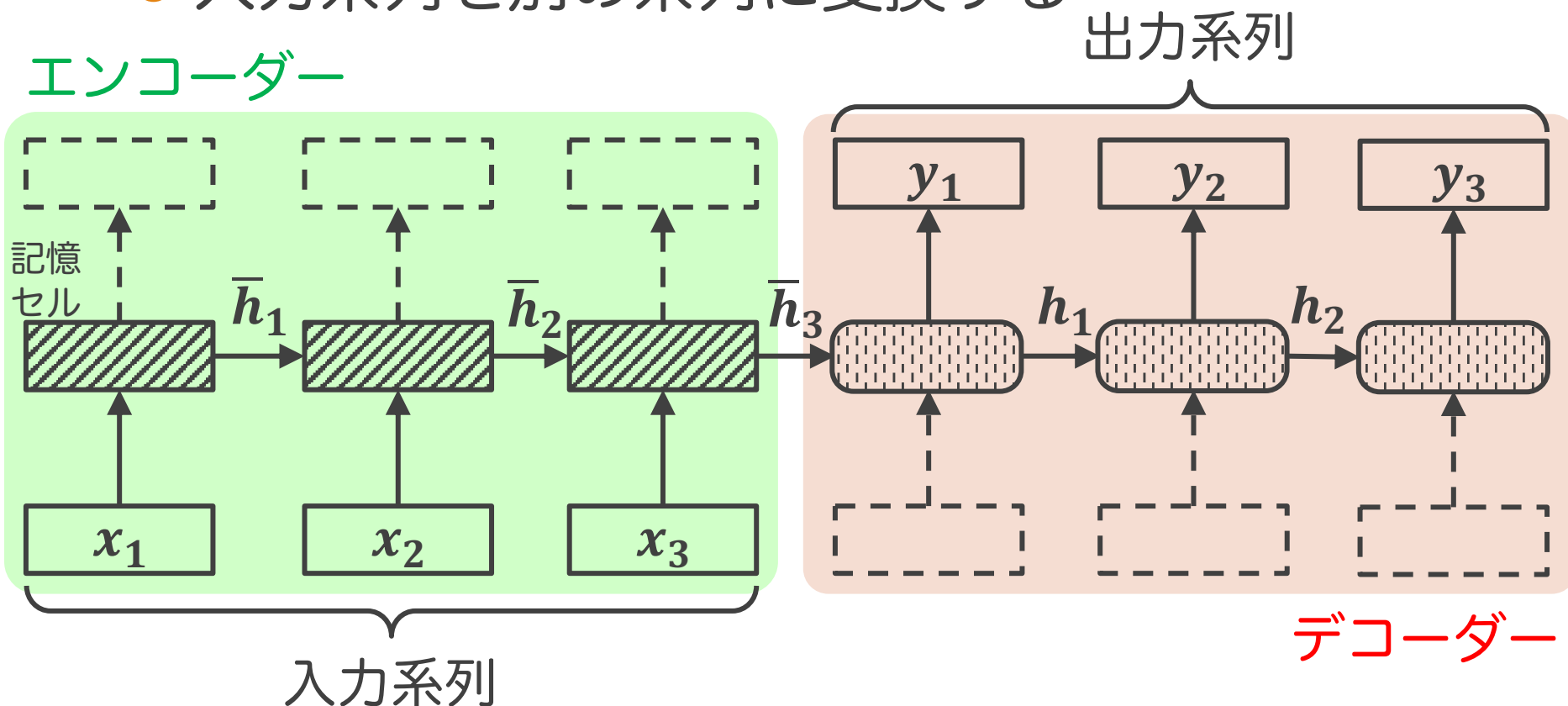


# エンコーダー・デコーダー モデル

---

# エンコーダー・デコーダーモデル

- 入力系列を別の系列に変換する



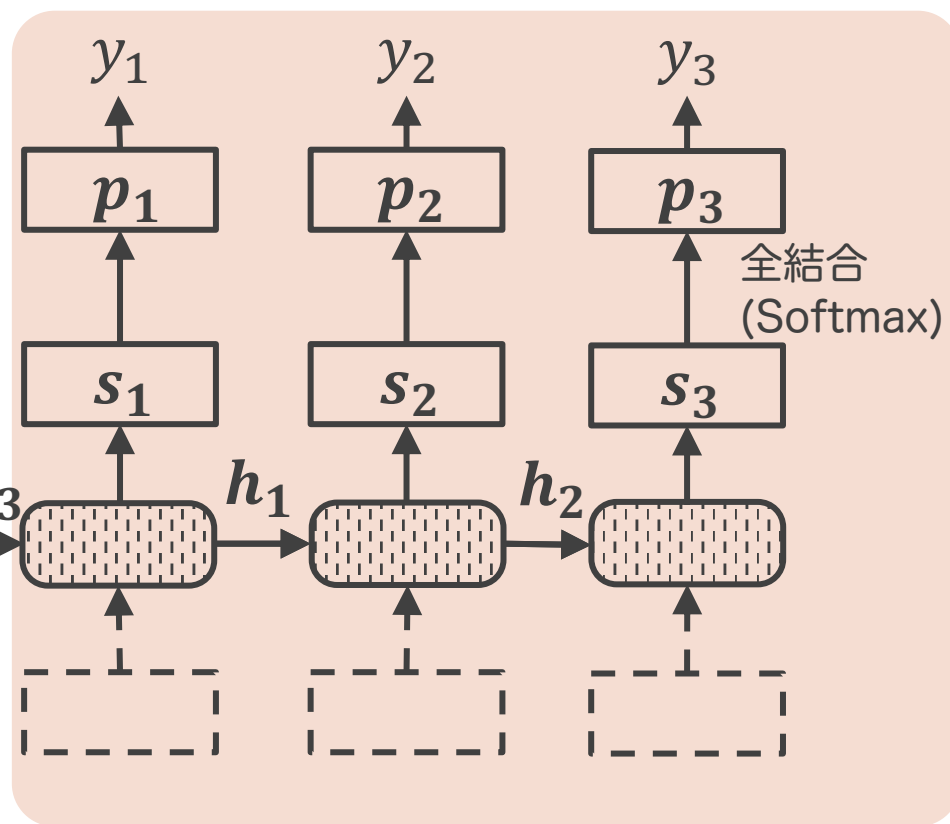
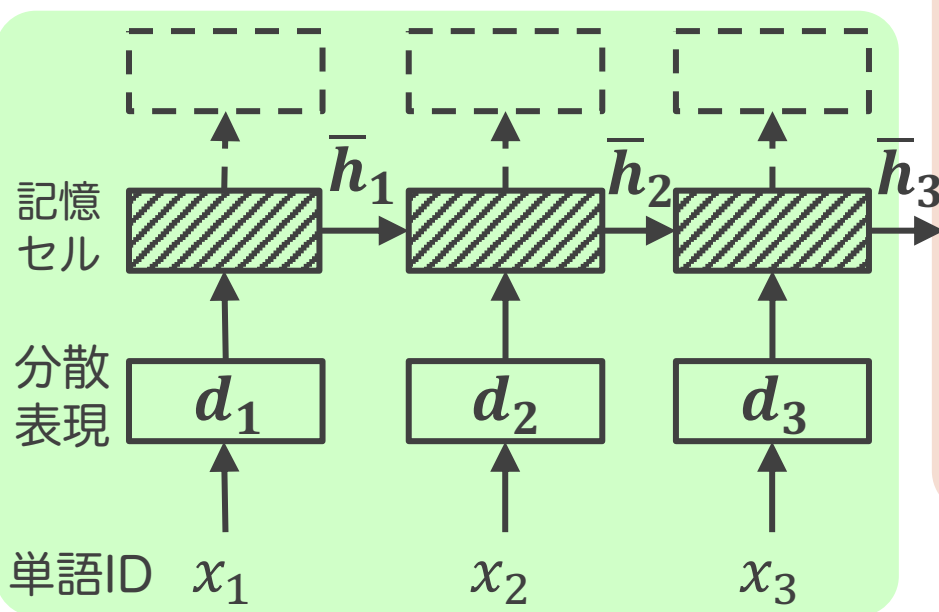
# エンコーダー・デコーダーモデル による機械翻訳

第一言語 → 第二言語

単語ID

確率  
ベクトル

エンコーダー

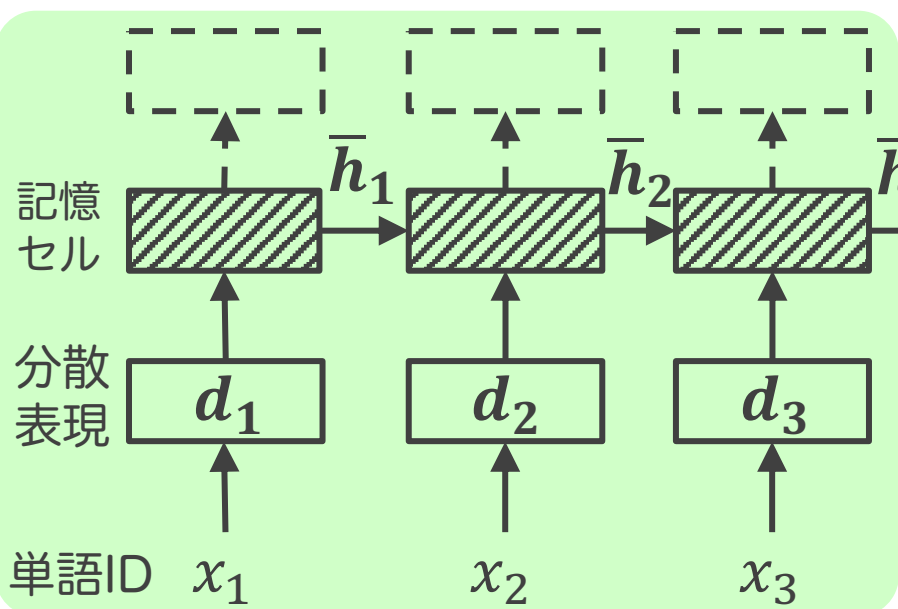


デコーダー

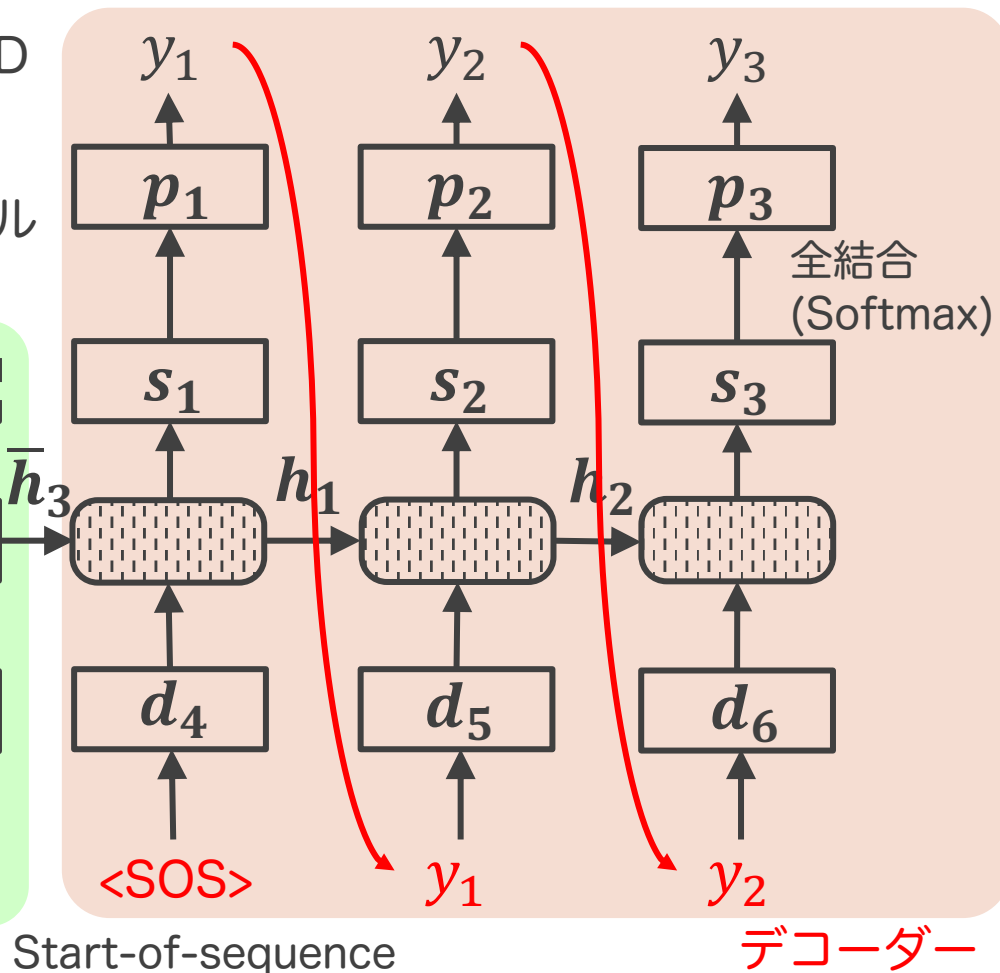
# エンコーダー・デコーダーモデル による機械翻訳

改善：  
デコード時に前の出力を入力  
→ 学習時は正解を混ぜる

## エンコーダー



単語ID  
確率  
ベクトル



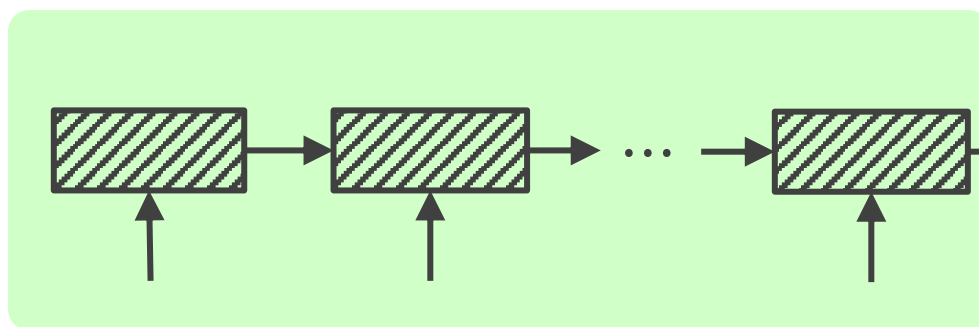
# アテンション

---

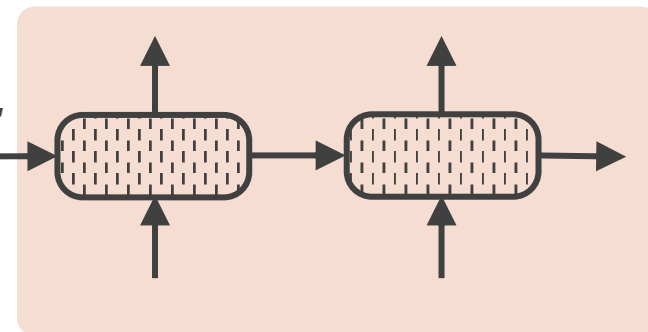
# 背景

- エンコーダー・デコーダーモデルでは、  
エンコーダーの最終状態  $\bar{h}_T$ のみをデコーダーに渡す  
→ 学習により、 $\bar{h}_T$ は元の文の意味全体を含むようになる  
→ しかし文が長くなると、 $\bar{h}_T$ にすべての文意を  
詰め込むことは困難になり、翻訳性能が下がる

エンコーダー

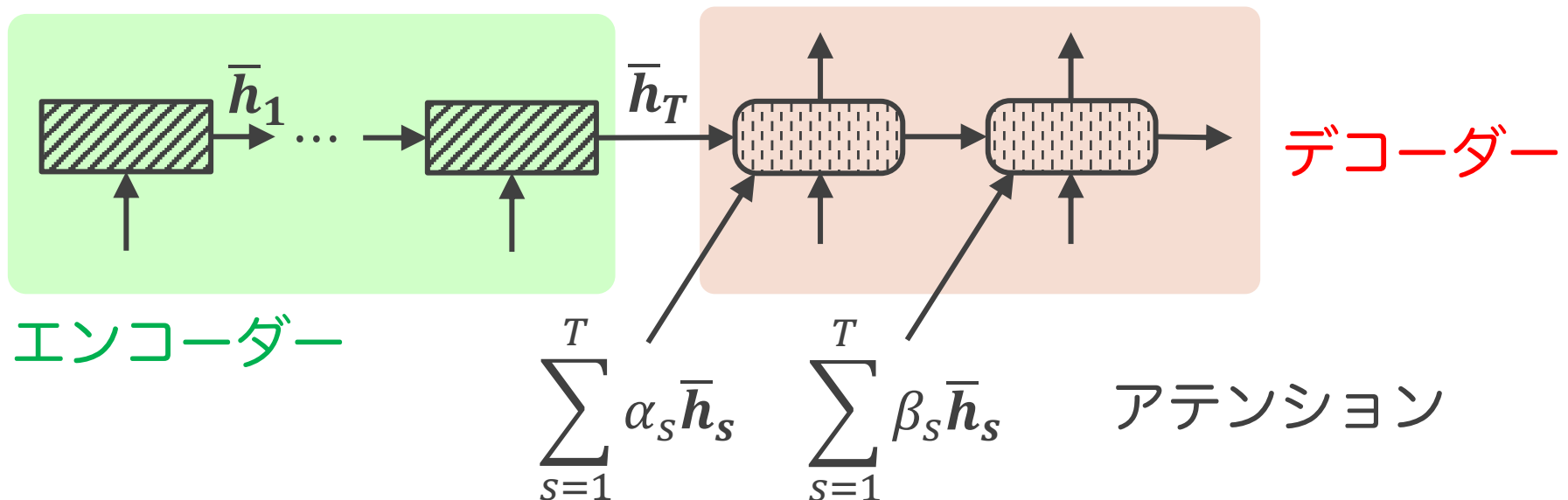


デコーダー



# アテンションとは？

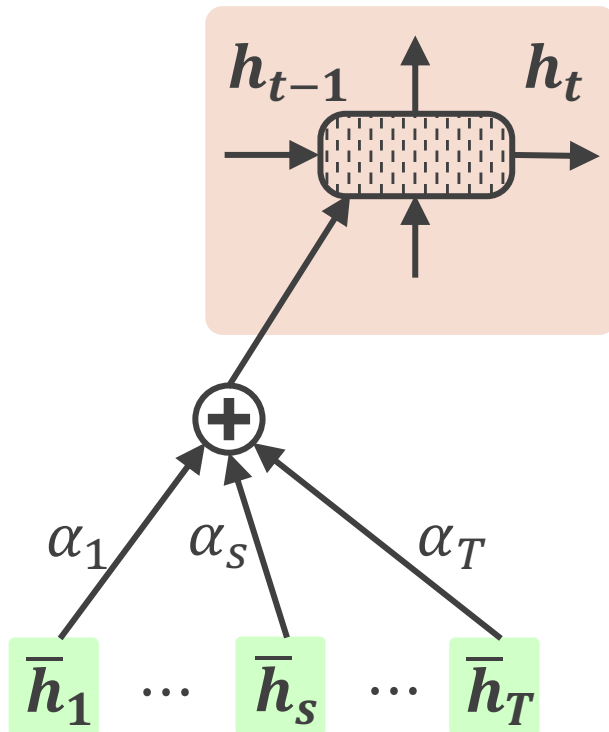
- デコーダーの各時刻に  
エンコーダーの中間状態  $\bar{h}_s$  ( $s = 1, \dots, T$ ) の凸結合を渡す  
→ 文が長くなっても翻訳性能が落ちない





# 凸結合の係数

- 前時刻の状態  $h_{t-1}$  と  $\bar{h}_s$  が類似 → 係数大  
似てない → 係数小



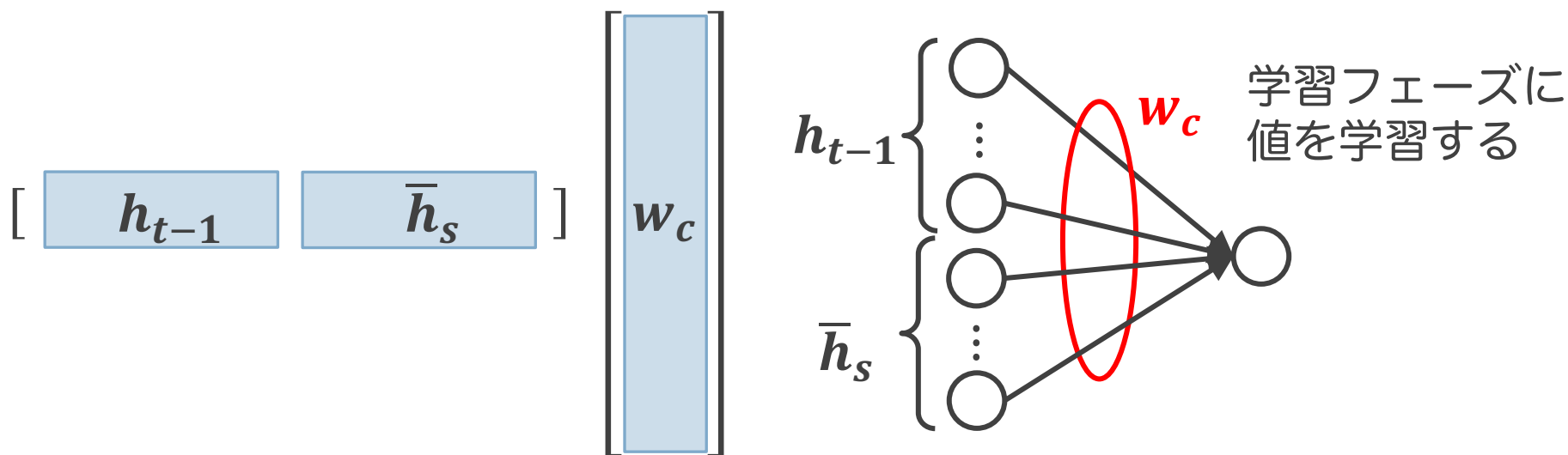
$h_{t-1}$  と  $\bar{h}_1, \dots, \bar{h}_s, \dots, \bar{h}_T$  の類似度スコア：  
 $[e_1, \dots, e_s, \dots, e_T]$

⇓ 凸結合化

$$\begin{aligned} & [\alpha_1, \dots, \alpha_s, \dots, \alpha_T] \\ &= \text{Softmax}[e_1, \dots, e_s, \dots, e_T] \end{aligned}$$

# 類似度スコア

- $h_{t-1}$ ,  $\bar{h}_s$  は横ベクトルとする※
- 連結注意:  $[h_{t-1} \ \bar{h}_s]w_c$



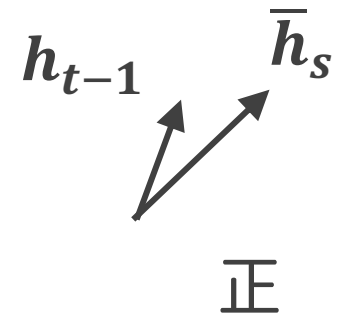
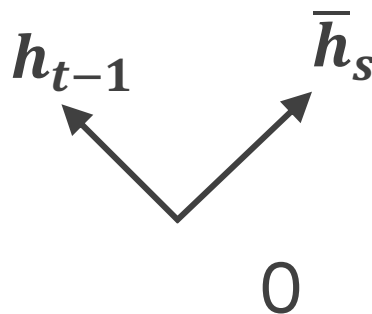
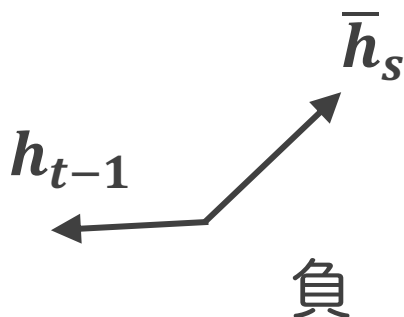
# 類似度スコア

- 内積注意 [Luong 2015] :  $h_{t-1} \bar{h}_s^T$

$$\left[ \boxed{h_{t-1}} \right] \left[ \boxed{\bar{h}_s^T} \right]$$

つまり  $h_{t-1}$  と  $\bar{h}_s$  の内積

内積の値



# 類似度スコア

- 一般注意（Transformer等）： $\mathbf{h}_{t-1} W_a \bar{\mathbf{h}}_s^T$

$$\left[ \boxed{\mathbf{h}_{t-1}} \right] W_a \left[ \bar{\mathbf{h}}_s^T \right]$$

$\mathbf{h}_{t-1}$ の $W_a$ による  
線形変換

つまり  $\mathbf{h}_{t-1} W_a$  と  $\bar{\mathbf{h}}_s$  の内積

$W_a$  は学習フェーズに学習する

直感的なイメージ：

$\mathbf{h}_{t-1}$  をターゲット言語の世界からソース言語の世界に変換（ $W_a$ ）してから  $\bar{\mathbf{h}}_s$  との内積をとる

# 類似度スコア

- Transformerで使用する一般注意
  - $h_{t-1}$ と $\bar{h}_s$ のそれぞれを異なる行列で線形変換してから内積を計算する

$h_{t-1}$ の $W_q$ による  
線形変換

$$\left[ \begin{array}{c} h_{t-1} \end{array} \right] W_q$$

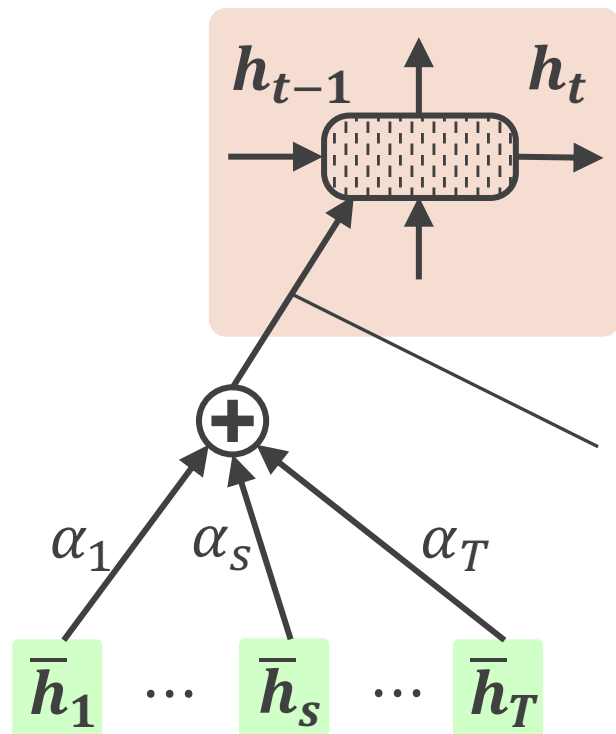
$$W_k^T \left[ \begin{array}{c} \bar{h}_s^T \end{array} \right]$$

$\bar{h}_s$ の $W_k$ による  
線形変換の転置

クエリ キー

$$\begin{aligned} & h_{t-1}W_q \text{ と } \bar{h}_sW_k \text{ の内積} \\ &= (h_{t-1}W_q)(\bar{h}_sW_k)^T \\ &= h_{t-1}W_qW_k^T\bar{h}_s^T \end{aligned}$$

# 解釈



時刻  $t$  で翻訳すべき単語の情報を  
デコーダーに伝えたい

⇒ もはや  $h_{t-1}$  は不要？

⇒ Transformer  
(「Attention is all you need」)

# セルフアテンション

---

# セルフアテンション

---

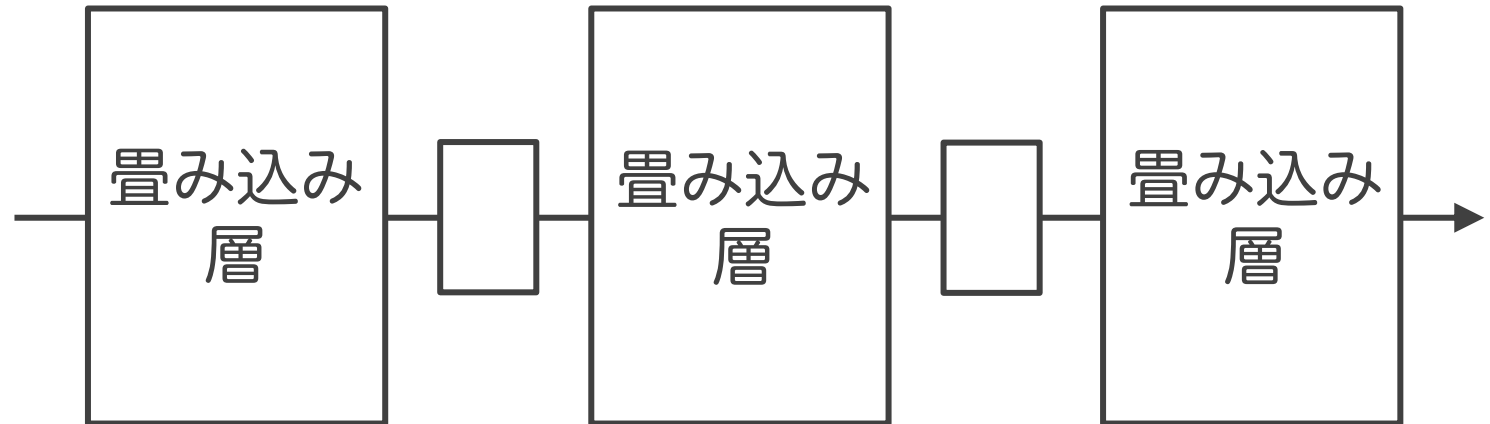
- 言語間の単語の類似性でなく、  
同一言語内で、文の単語間の関係性を学習する
  - 修飾語と被修飾語
  - 主語と述語
  - 「それ」とそれが指し示す名詞
- セルフアテンション層を繰り返すことで  
より複雑な関係性を学習できる



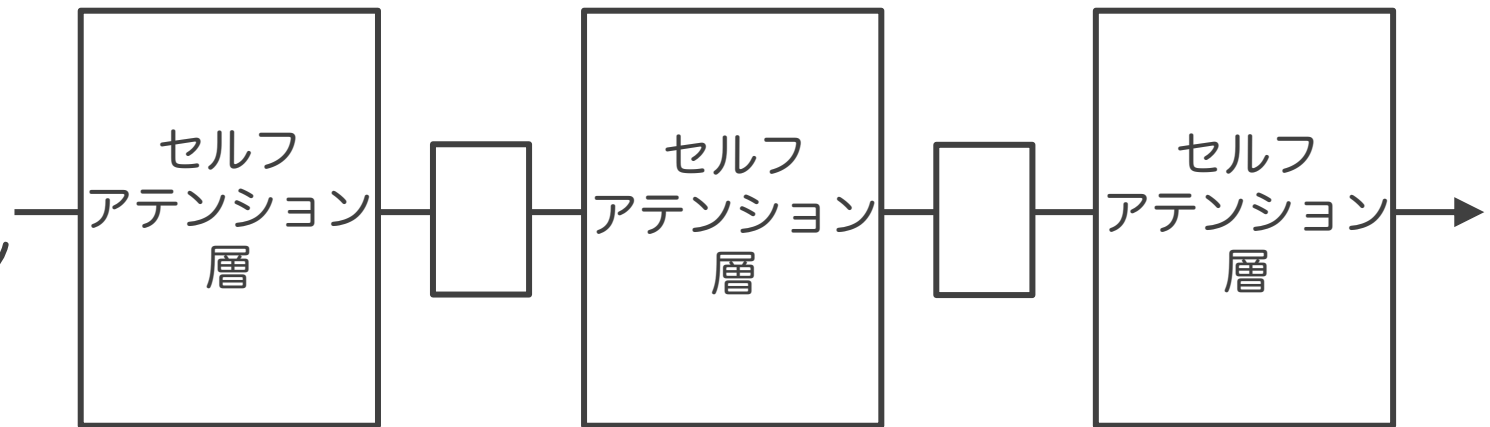
単純なパターンの学習

複雑なパターンの学習

CNN



セルフ  
アテンション



単純な関係性の学習

複雑な関係性の学習

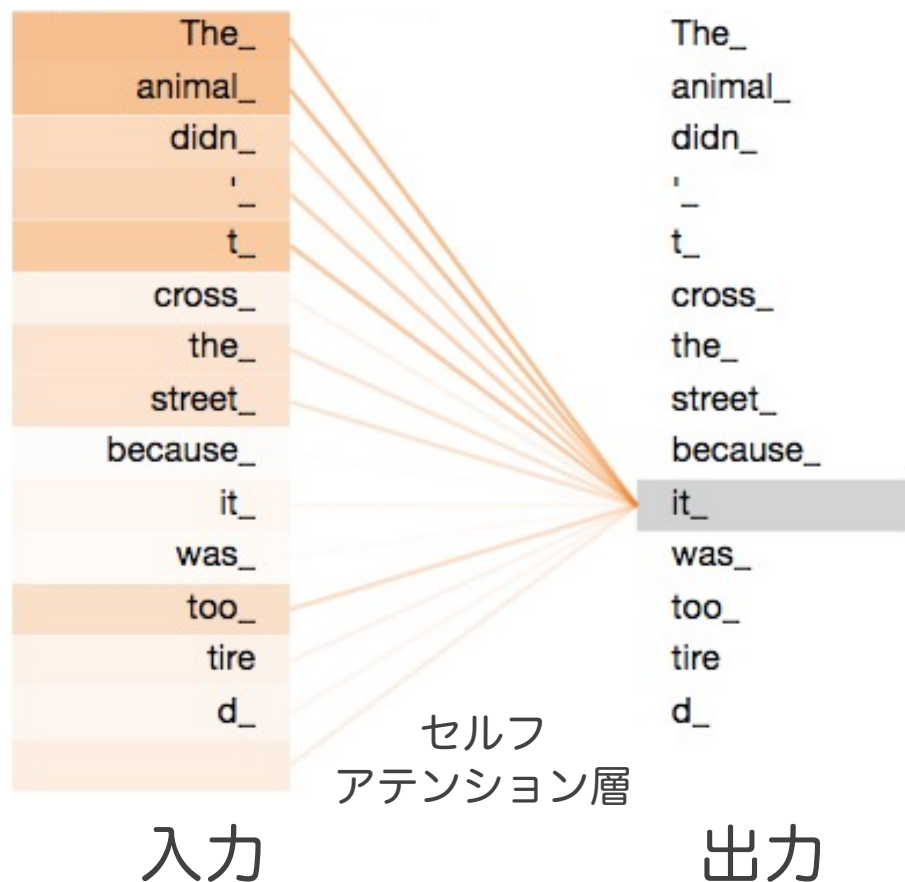
# セルフアテンションを活用したネットワークの学習例（BERT）

---



一部の単語が隠された（もしくは他の単語に置換された）文を受け取り，元の文を出力する（自己教師あり学習）

# セルフアテンションの直感的理解



it を他の単語の凸結合で表現する

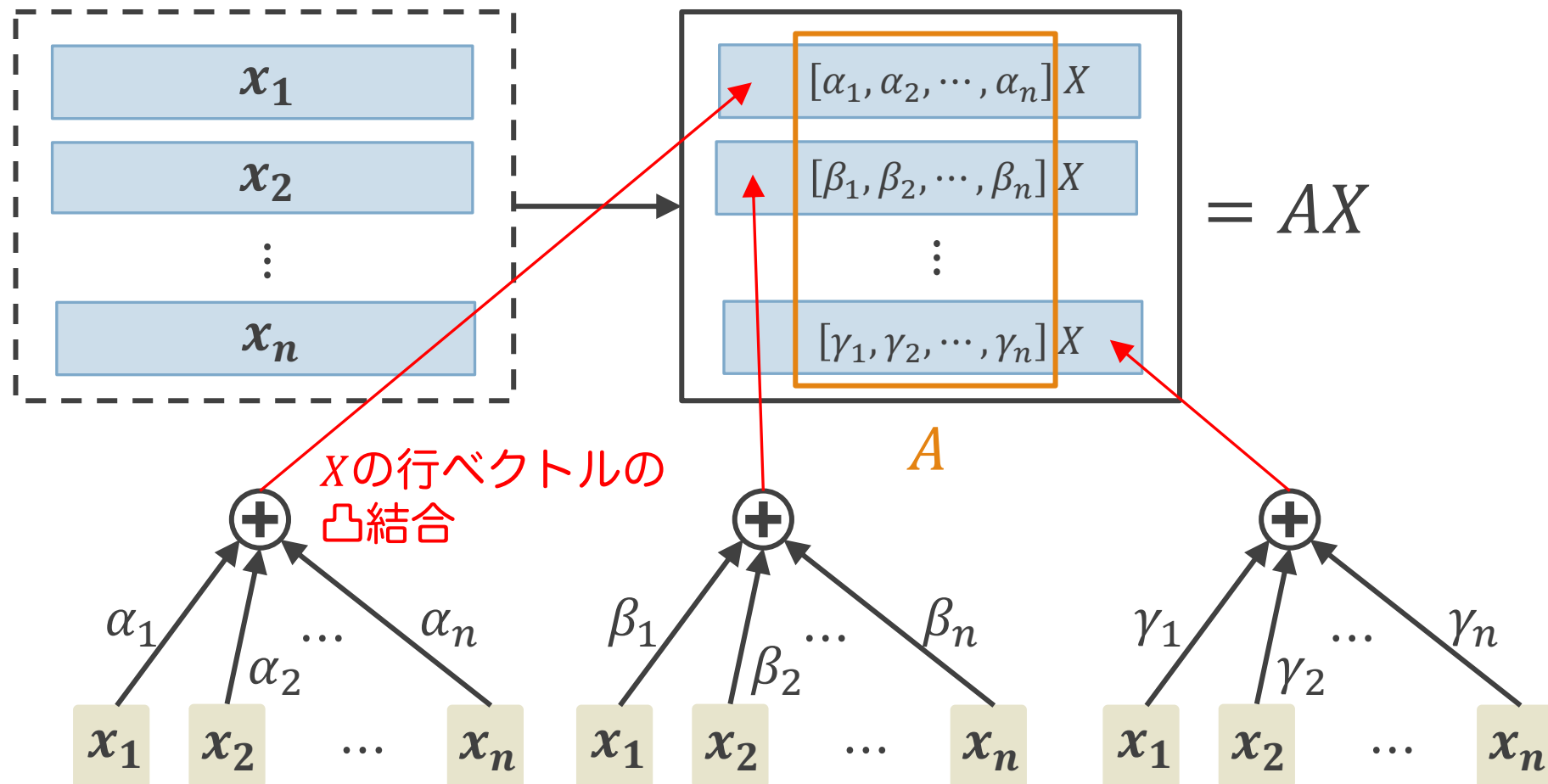
⇒ 「it」という単語と他の単語の結びつきを表現する

※ セルフアテンション単体では出現順を考慮しない

# セルフアテンション層

入力：単語列  $X$  (分散表現)

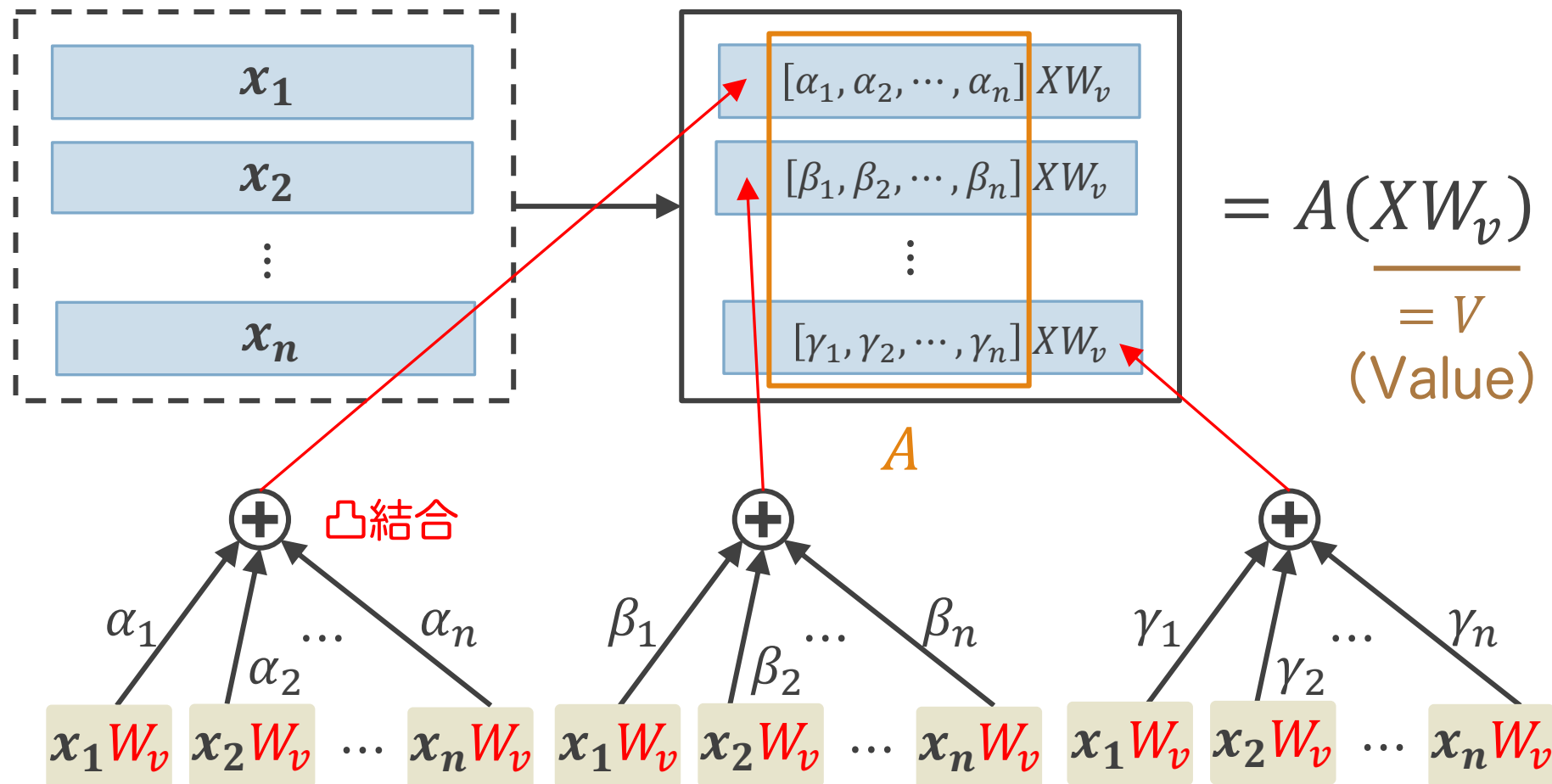
各単語  $x_s$  に対するアテンション



# Transformerのセルフアテンション層

入力：単語列  $X$  (分散表現)

各単語  $x_s$  に対するアテンション



各単語を線形変換したのち，凸結合を求める

# セルフアテンション層における 凸結合の係数

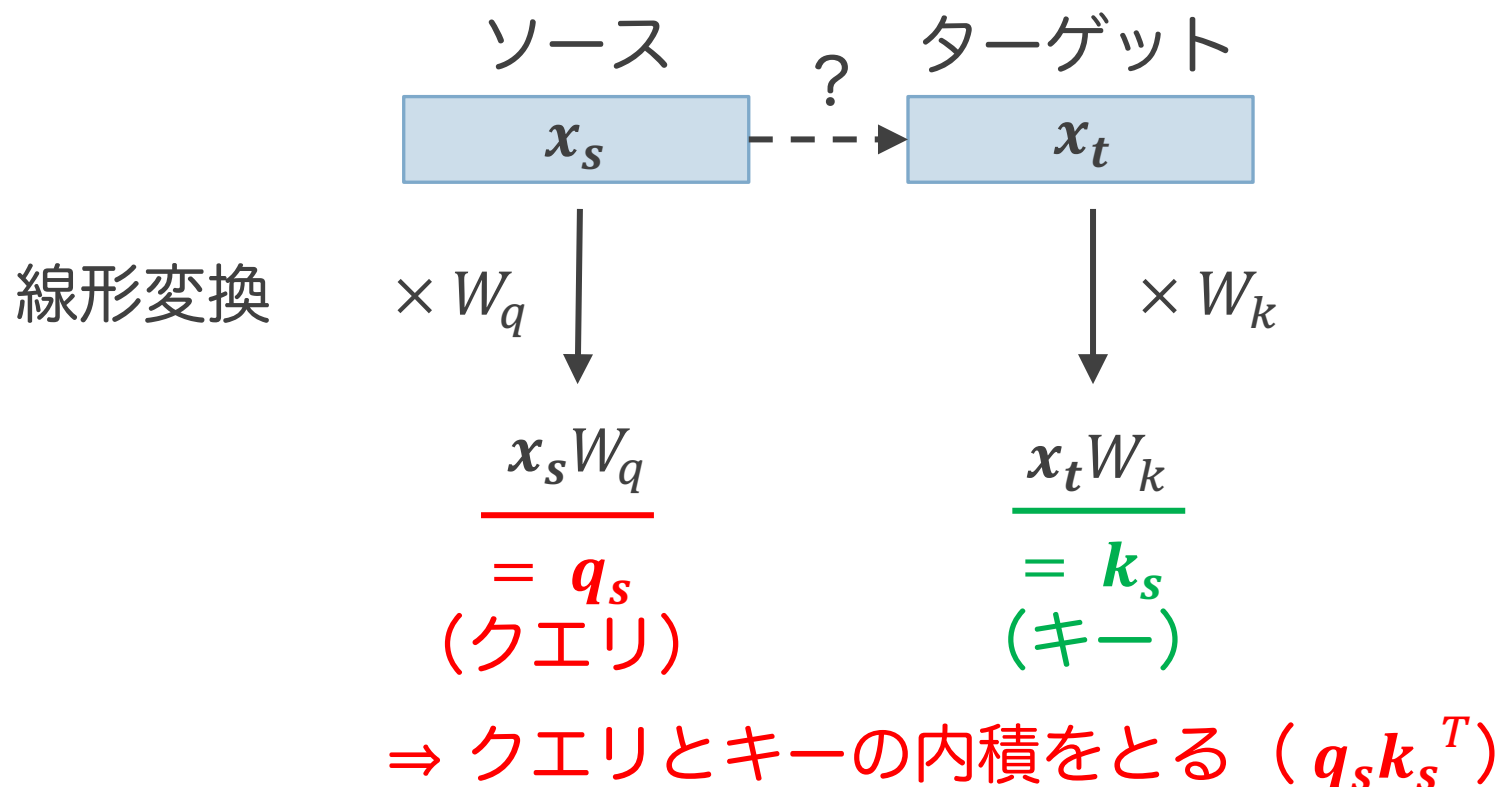
---

- 単に単語どうし似ているというだけでなく、  
単語間の関係を抽出したい

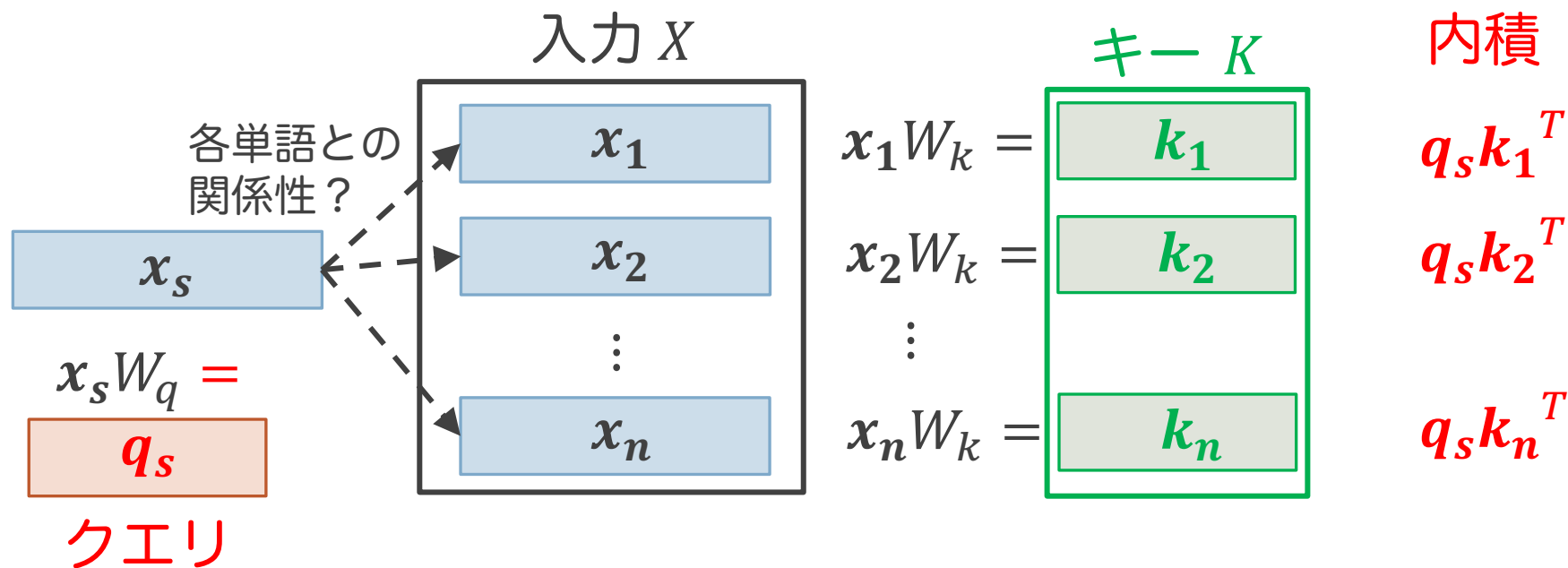


# セルフアテンション層における 凸結合の係数

- 「関係」の抽出



# セルフアテンション層における 凸結合の係数



各単語との関係性:  $[q_s k_1^T, q_s k_2^T, \dots, q_s k_n^T] = q_s K^T$

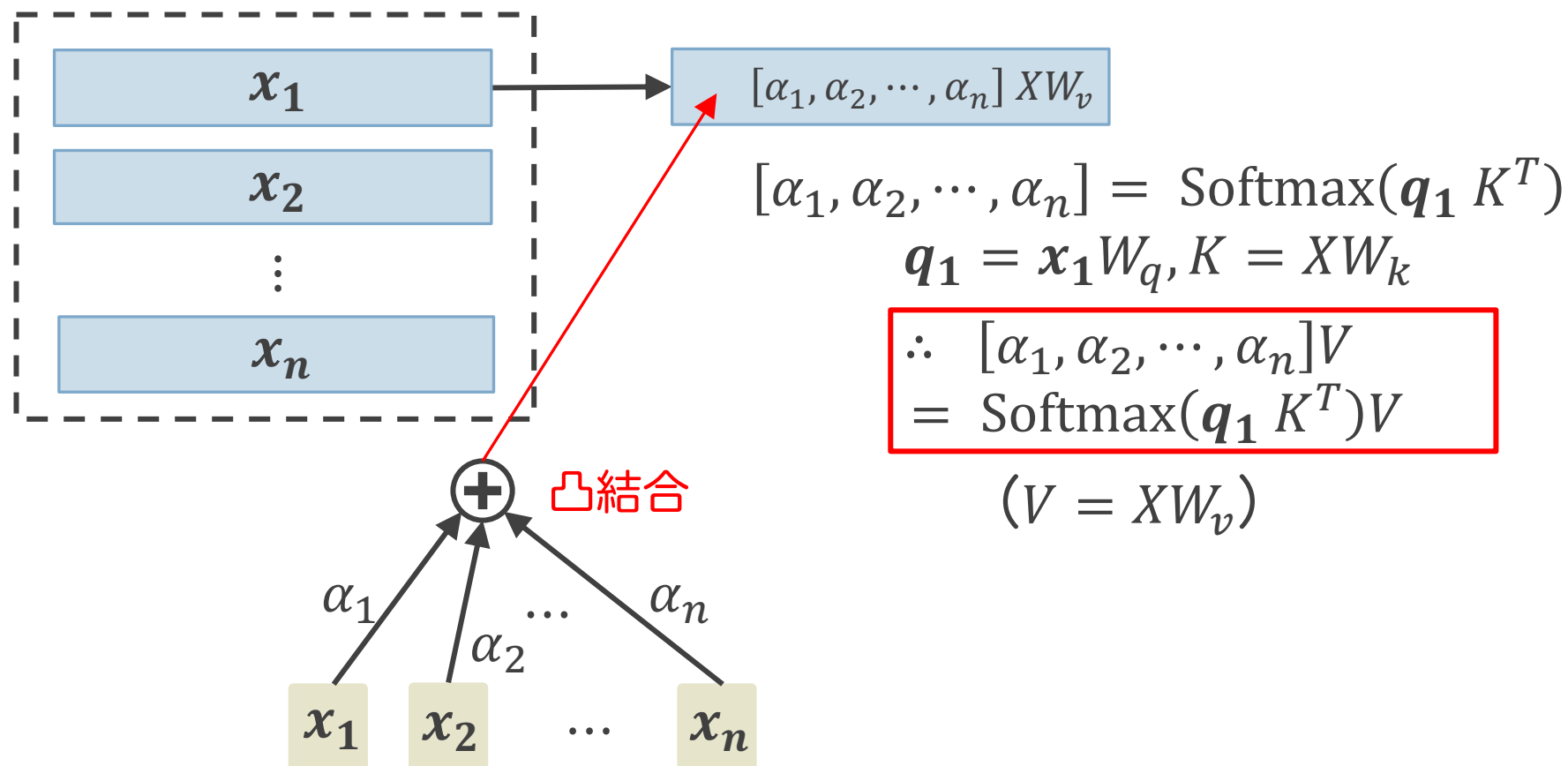
$\Rightarrow$  凸結合になるように変換

係数  $\text{Softmax}[q_s k_1^T, q_s k_2^T, \dots, q_s k_n^T] = \text{Softmax}(q_s K^T)$



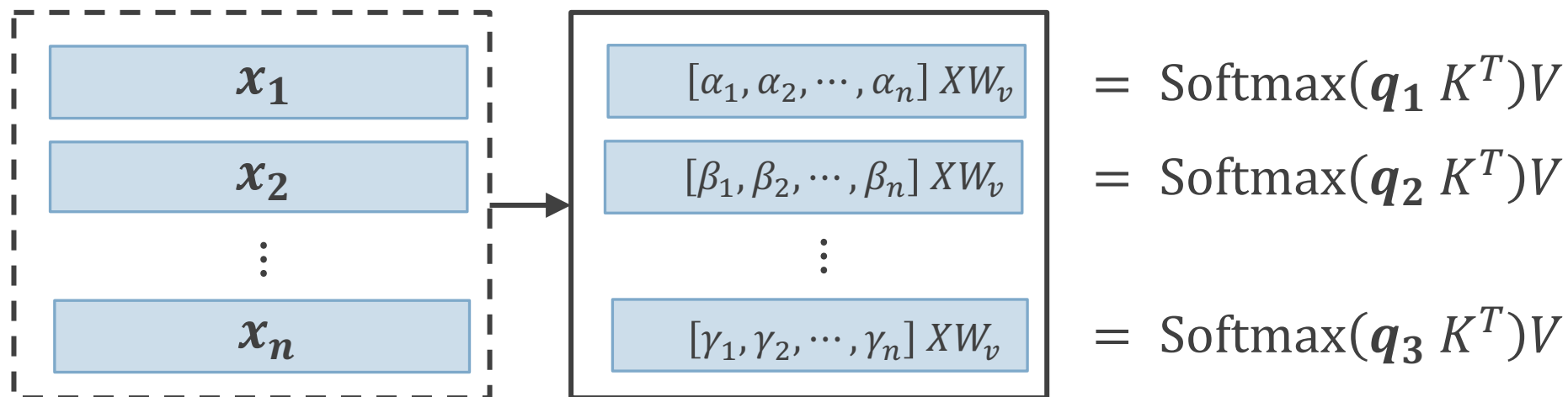
# 1クエリに対する セルフアテンションの導出

入力：単語列  $X$  (分散表現) 単語  $x_1$  に対するアテンション



# セルフアテンション層まとめ

入力：単語列  $X$  (分散表現) 各単語  $x_s$  に対するアテンション



セルフアテンションの出力：  $\text{Softmax}(Q K^T) V$   
( $Q = X W_q, K = X W_k, V = X W_v$ )

Softmax は行ごとに計算する

# Transformerで使われる セルフアテンション層

---

- 凸結合において、一部のベクトルのみの成分が大きくなり過ぎないように  
Softmax 適用前に全体的に値を小さくする  
⇒ 各係数の値を  $\sqrt{d_k}$  で割る  
(  $d_k$  はキーベクトルの次元数)

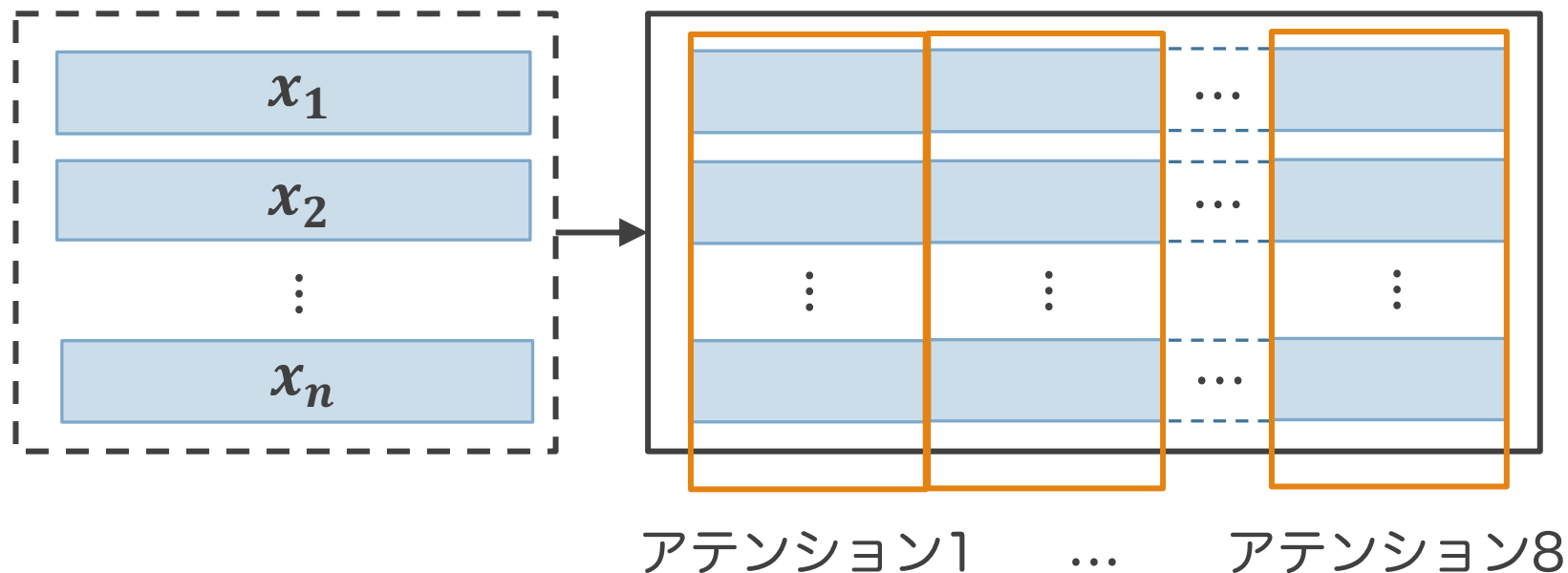
セルフアテンションの出力：  $\text{Softmax}\left(\frac{Q K^T}{\sqrt{d_k}}\right) V$

$$(Q = XW_q, K = XW_k, V = XW_v)$$

# マルチヘッドアテンション

- 異なる行列を用いて複数回アテンションを計算する  
⇒ 複数の異なる関係性を抽出できる

入力：単語列  $X$ （分散表現） 各単語  $x_s$  に対するアテンション



# Transformer

---

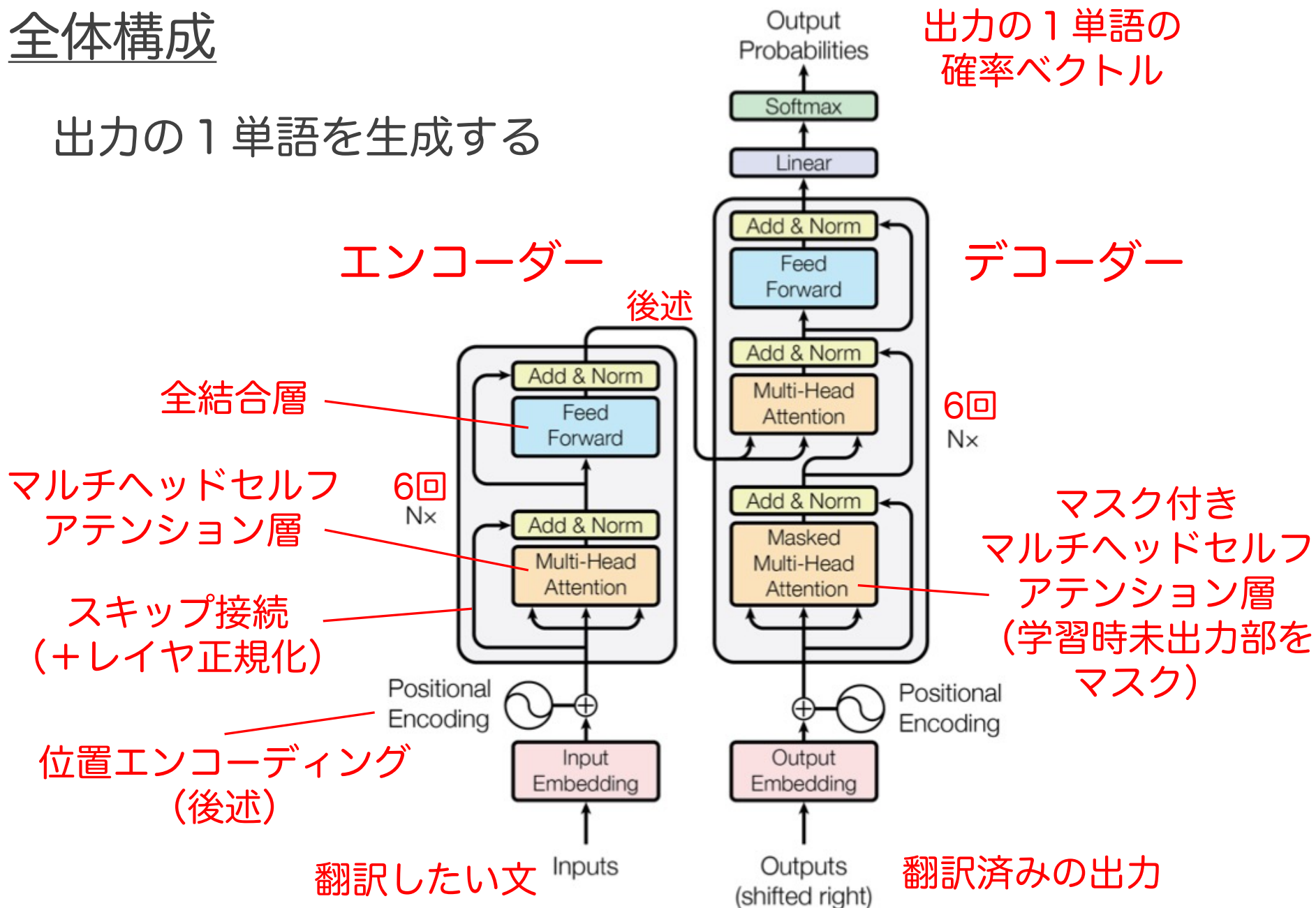
# Transformer概要

---

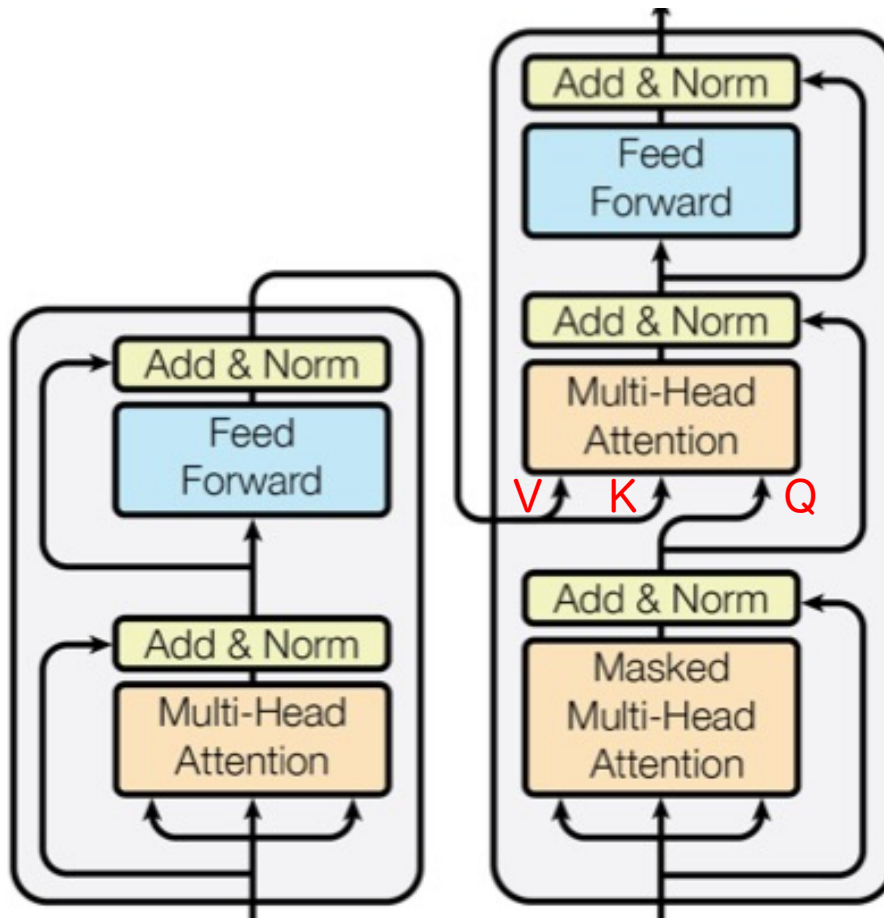
- 高性能な機械翻訳モデル
- 再帰的な層を使用せず，アテンションのみでモデルを構築する
- 入力として文（単語列）を丸ごと入れる
  - 再帰的な層では各時刻に付き1単語しか入力できず並列化がしにくかったが，並列化が容易になる
- エンコーダー・デコーダーともにセルフアテンションを6回積み上げる
  - エンコーダーとデコーダーはアテンションで接続

# 全体構成

出力の1単語を生成する



# エンコーダーとデコーダーの接続



- エンコーダー出力からキーとバリューを作る
- デコーダー側からクエリを作る



# 位置エンコーディング

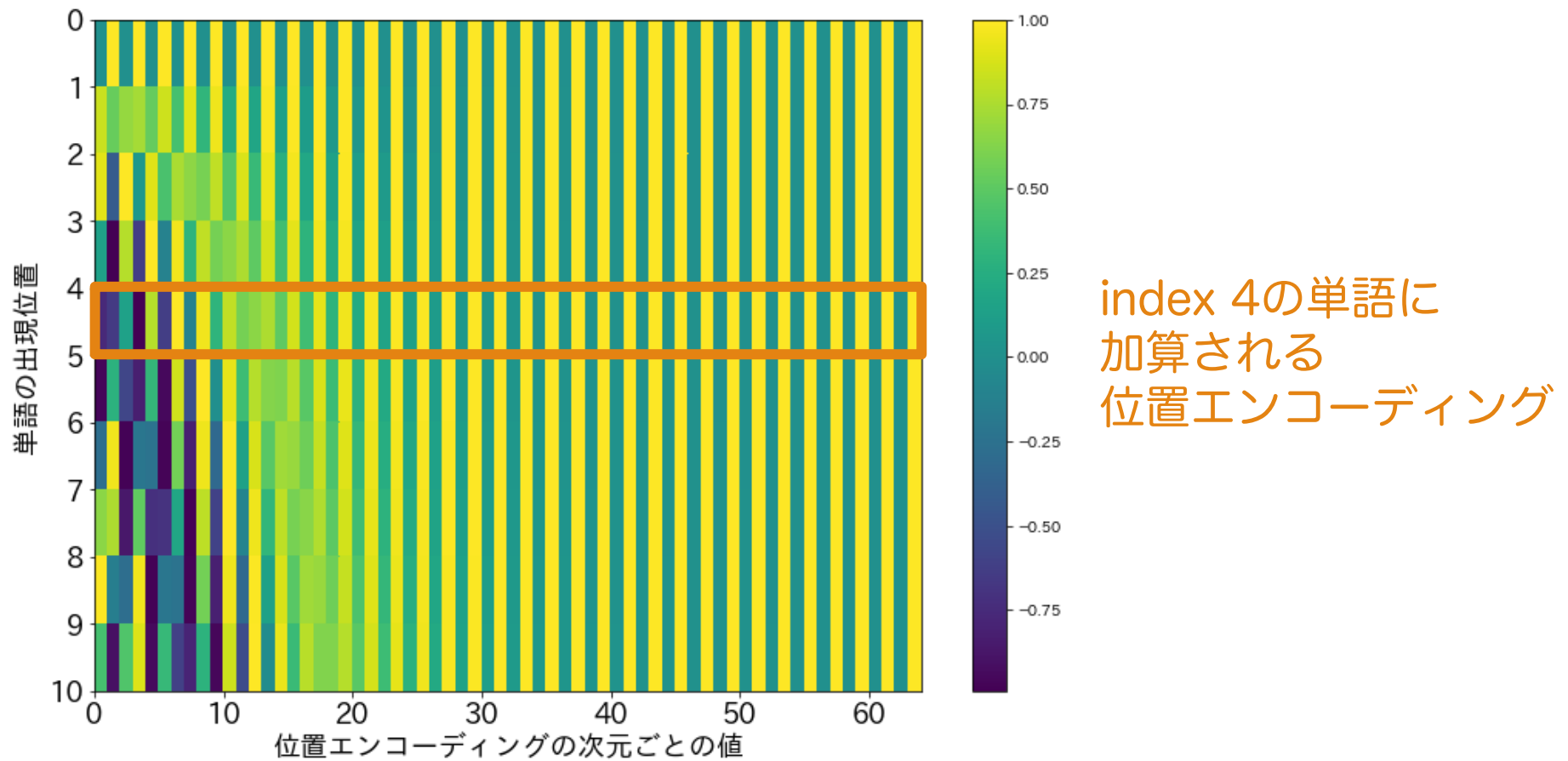
---

- 背景

- ノーマルなセルフアテンションでは，文の中で単語どうしがどれくらい離れているかはまったく考慮されない

- 方針

- 考慮できるようにするため，単語間の距離が離れているとき，分散表現での距離も離すようにする  
⇒ 文中での位置に応じたベクトルを分散表現に加える



⇒ 位置に応じて異なるベクトルが加算される