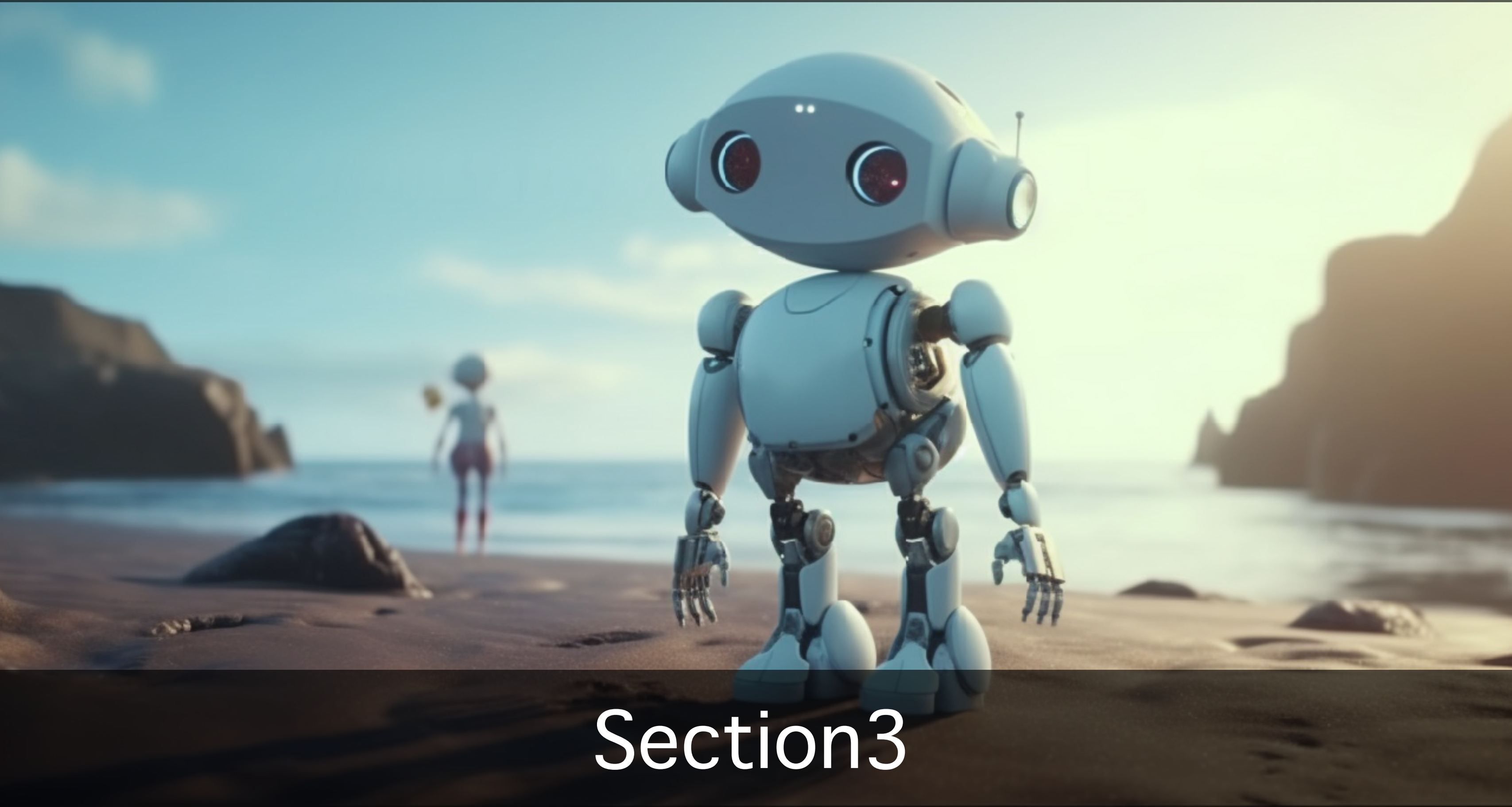


# 大規模言語モデル(LLM)の仕組み入門



## Section3



# Section3の概要



# 講座の内容

Section1. LLMの概要

Section2. ニューラルネットワークの仕組み

 **Section3. Transformerの仕組み**

Section4. LLMの仕組み

# 今回の内容

1. Section3の概要
2. 自然言語処理の概要
3. Transformerの概要
4. Attentionの概要
5. Transformerの利用
6. 演習

# 教材の紹介

- **Pythonの基礎:**

python\_basic

- **Section3の教材**

01\_simple\_bert.ipynb

02\_exercise.ipynb

[https://github.com/yukinaga/llm\\_mechanism/](https://github.com/yukinaga/llm_mechanism/)

# 演習の解答 Section2

[https://github.com/yukinaga/llm\\_mechanism/  
blob/main/section\\_2/02\\_exercise.ipynb](https://github.com/yukinaga/llm_mechanism/blob/main/section_2/02_exercise.ipynb)



# 自然言語処理の概要



# ChatGPT に聞いてみる

「自然言語処理って何ですか？」

→ ?

<https://chat.openai.com/>



# 自然言語処理とは

- 自然言語とは、日本語や英語などの我々が普段使う言語のこと
- 自然言語処理（Natural Language Processing、NLP）とは、  
自然言語をコンピュータで処理する技術のこと

# 自然言語処理の応用

- 検索エンジン
- 機械翻訳
- 予測変換
- スпамフィルタ
- 音声アシスタント
- 小説の執筆
- 対話システム
- etc...

# 自然言語処理技術の要素

- **形態素解析**
  - 文書を単語に分割する技術
- **単語の分散表現**
  - 文書内での関係性を踏まえて、単語をベクトル化する技術
- **再帰型ニューラルネットワーク (RNN)**
  - 時系列を扱うのが得意なニューラルネットワークの一種
- **Seq2Seq**
  - RNNをベースにした、文章などを生成可能なモデル
- etc...

# 日本語の形態素解析

- 形態素とは、言葉が意味を持つまとまりの単語の最小単位のこと
- 形態素解析とは、自然言語を形態素にまで分割すること
- 日本語や中国語、タイ語は単語間にスペースが無いので、  
形態素解析が必要
- 以下は代表的な日本語の形態素解析ライブラリ
  - MeCab → 知名度が高く、高速、高精度
  - Janome → 速度はMeCabに劣るが、導入が簡単
  - etc...



# one-hot表現

すもも も もも も もも の うち

	すもも	も	もも	の	うち
ID	0	1	2	3	4

「すもも」のone-hot表現: [1 0 0 0 0]

「も」のone-hot表現: [0 1 0 0 0]

# 分散表現

- 単語間の関連性や類似度に基づくベクトルで、単語を表現する

200要素程度

男性	0.01	0.58	0.24	...
ロンドン	0.34	0.93	0.02	...
Python	0.97	0.08	0.41	...

- word2vecなどを使えば、足し算や引き算が可能なベクトルを作ることができる

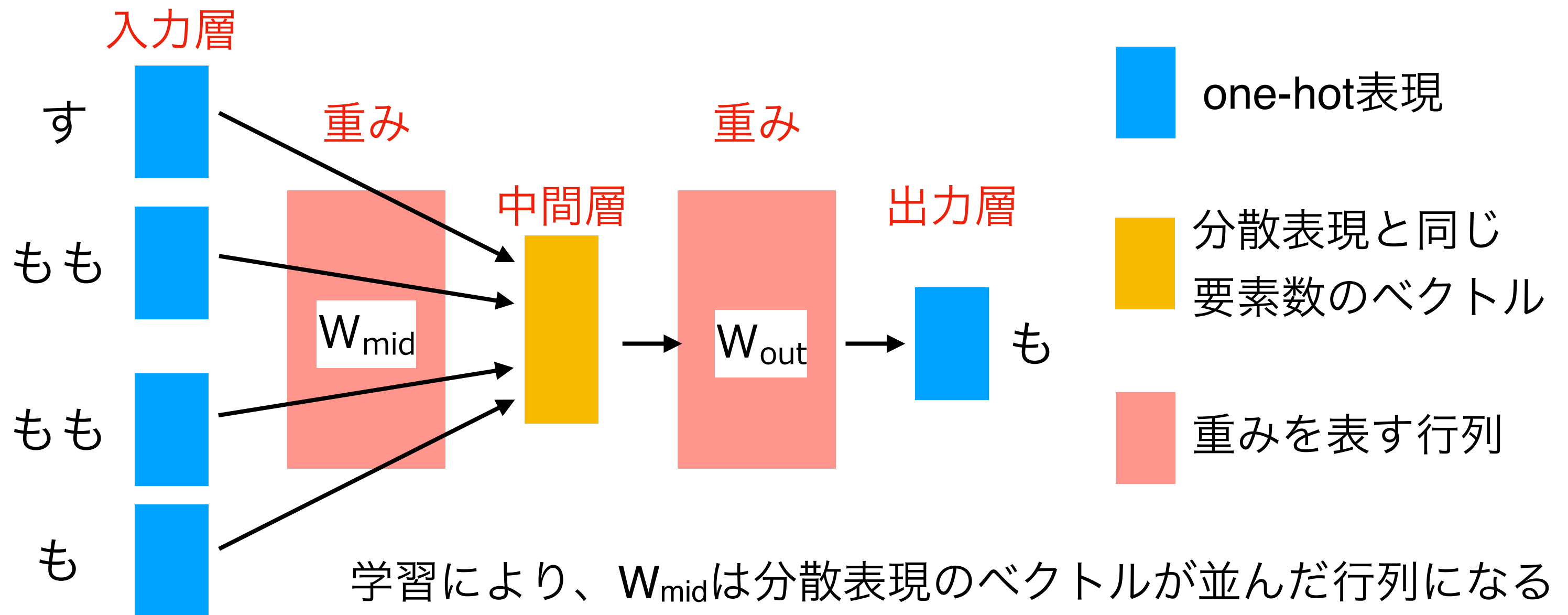
例: 「王」 - 「男」 + 「女」 = 「女王」

# word2vec

- word2vecは、分散表現を作成するための技術
- word2vecでは、CBOW (continuous bag-of-words) もしくは、skip-gramというニューラルネットワークが用いられる

# CBOW (continuous bag-of-words)

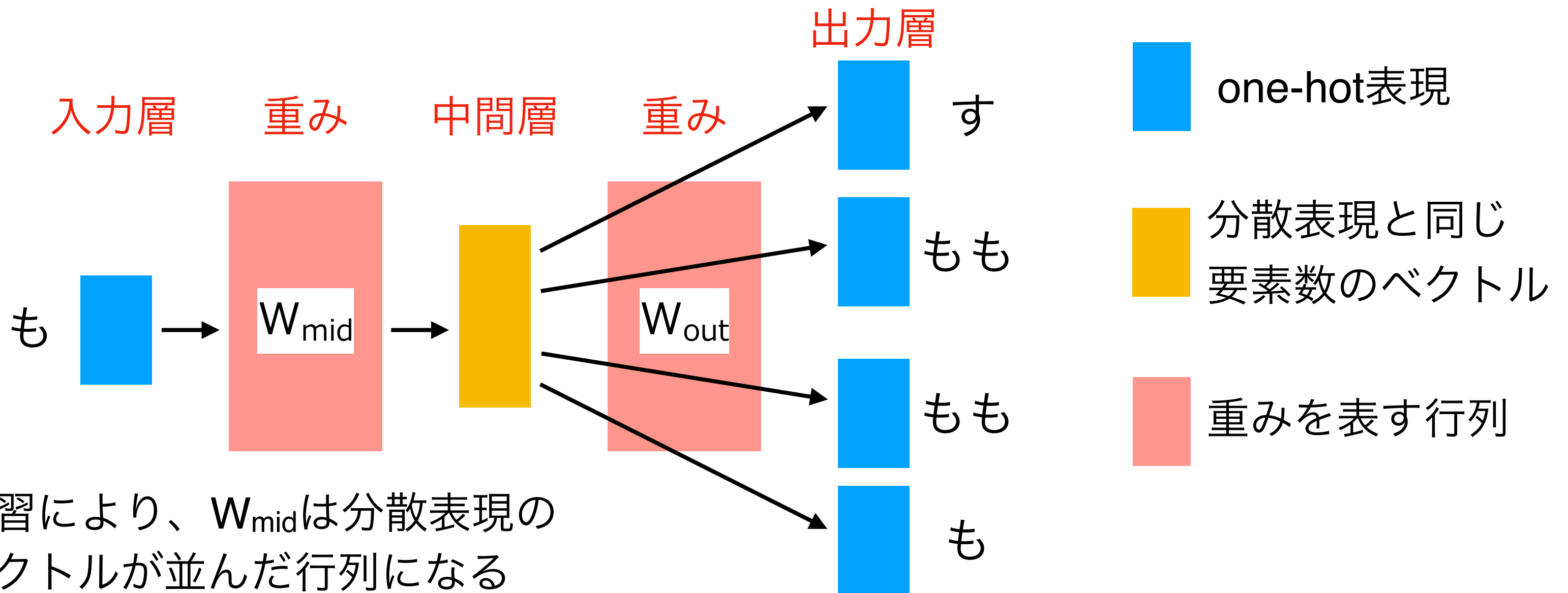
- 前後の単語から対象の単語を予測するニューラルネットワーク
- 学習に要する時間がskip-gramよりも短い





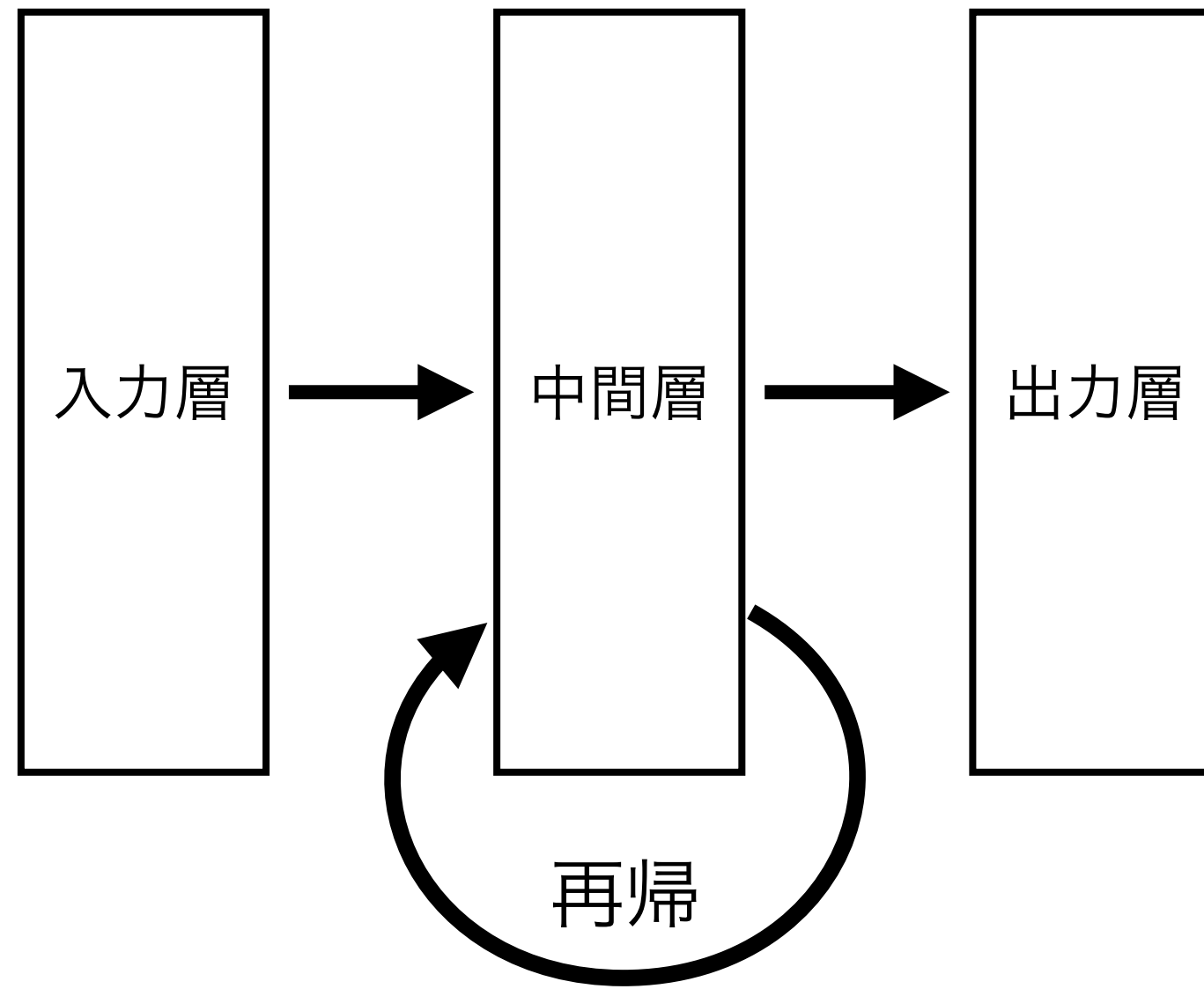
# skip-gram

- ある単語から、前後の単語を予測するニューラルネットワーク
- CBOWよりも学習に時間がかかるが、精度がよい

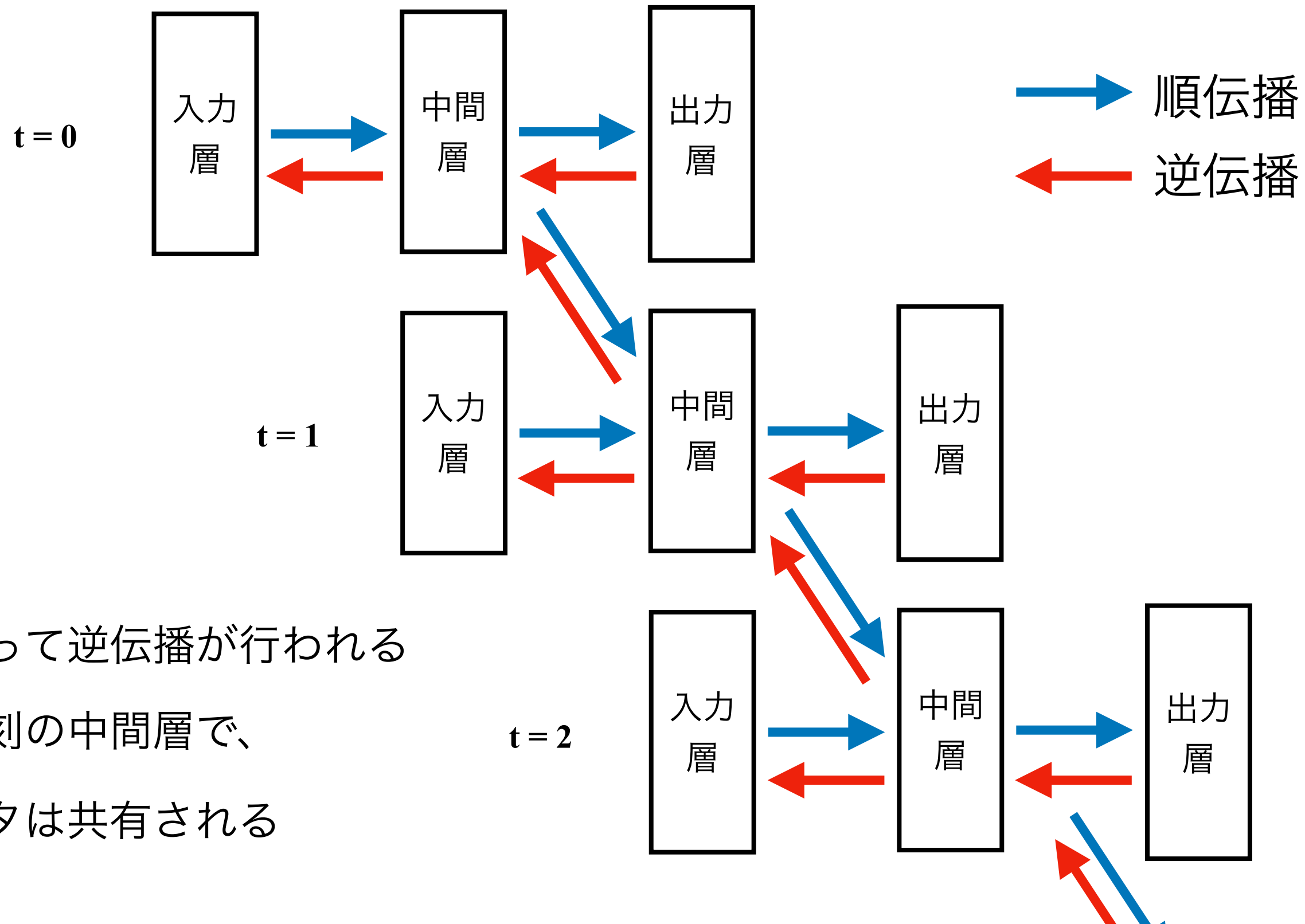


# 再帰型ニューラルネットワーク（RNN）とは？

- 再帰型ニューラルネットワーク（Reccurent Neural Network）
  - 入力と正解が「時系列データ」となる
  - 中間層が「再帰」の構造を持ち、前後の時刻の中間層とつながる



# RNNの順伝播と逆伝播



- 時間を遡って逆伝播が行われる
- 全ての時刻の中間層で、パラメータは共有される

# 時系列データの例

- 文書
- 音声データ
- 動画
- 株価
- 産業用機器の状態
- etc...

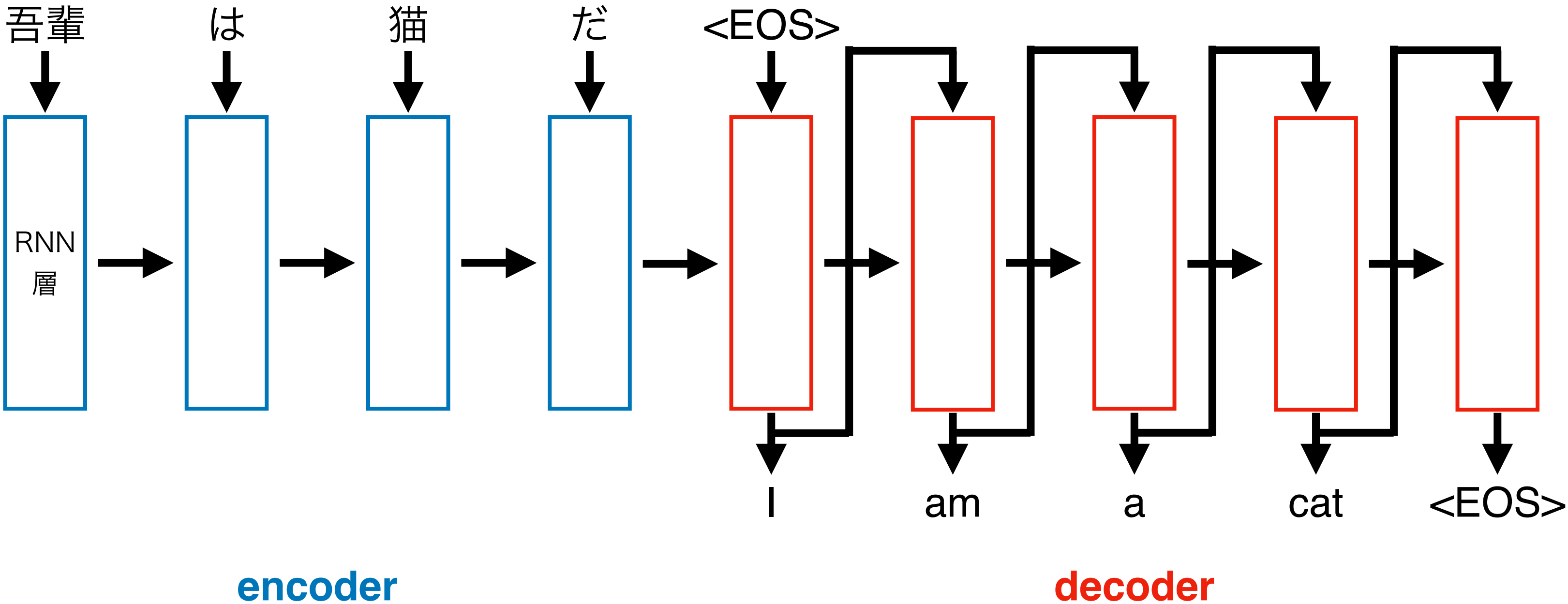


# Seq2Seqとは

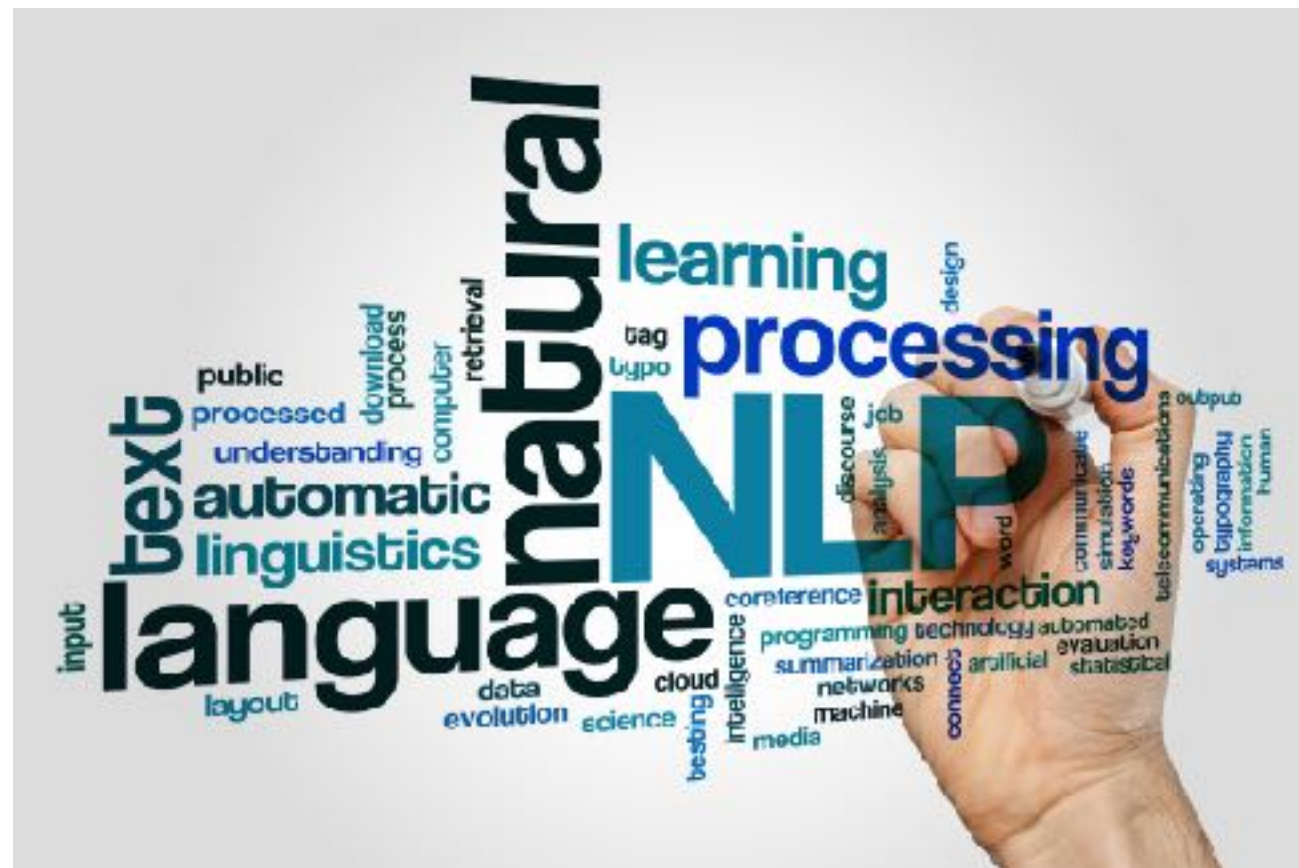
- Seq2Seqは、系列(sequence)を受け取り、別の系列へ変換するモデル
- 自然言語処理でよく利用される
- 文章などの入力を圧縮するencoderと、出力を展開するdecoderからなる
- 以下は活用例
  - 機械翻訳（例: 英語の文章 → フランス語の文章）
  - 文章要約（元の文章 → 要約文）
  - 対話（自分の発言 → 相手の発言）
  - etc...

# Seq2Seqの構造

Seq2Seqによる翻訳の例



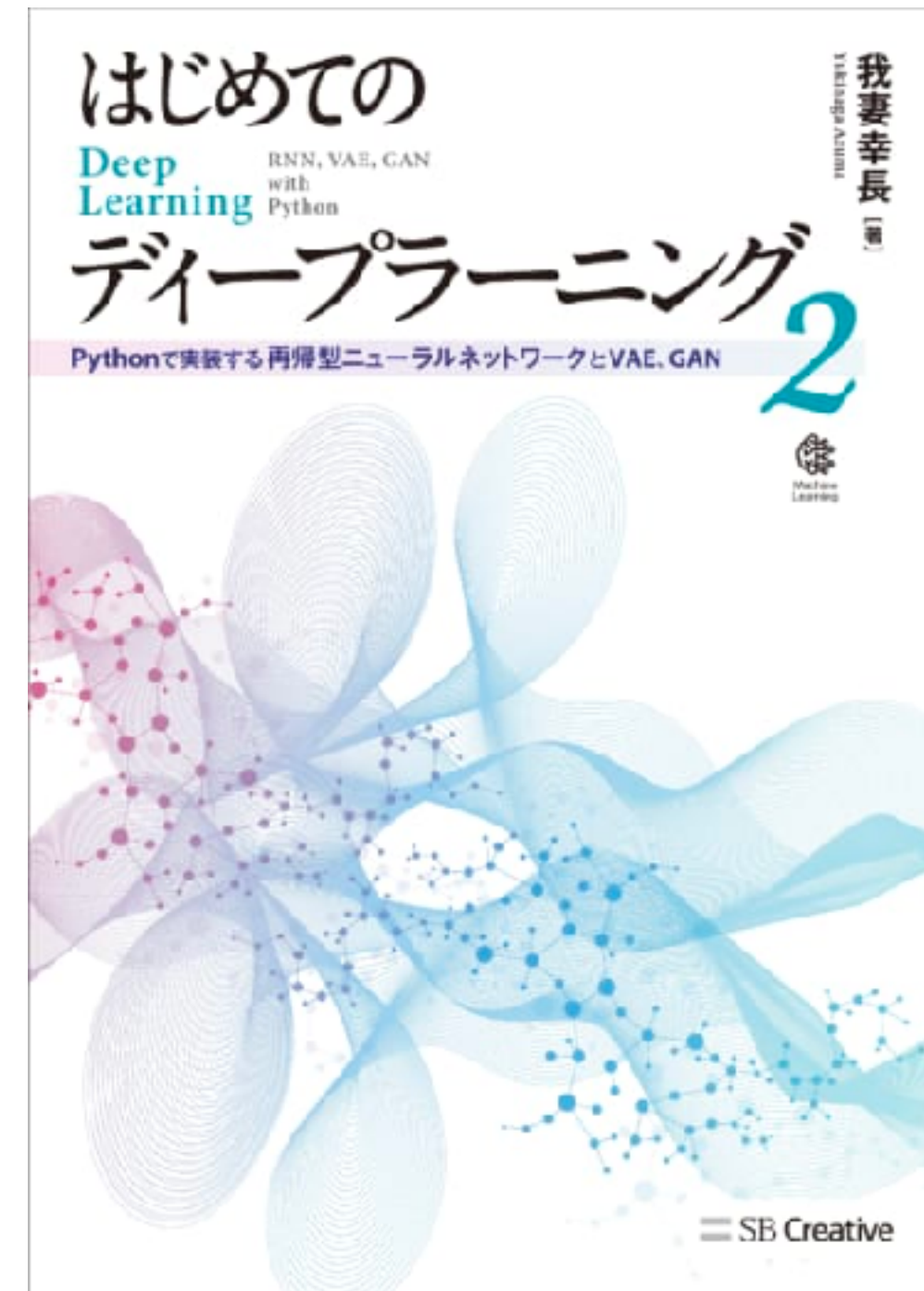
# さらに詳しく学びたい方へ...



Udemyコース

自然言語処理とチャットボット:

AIによる文章生成と会話エンジン開発





# Transformerの概要





# ChatGPT に聞いてみる

「Transformerって何ですか？」

→ ?

<https://chat.openai.com/>

# RNNによる自然言語処理の問題点

- **学習時間が長い**
  - データを並列で処理できないため、学習には長い時間がかかる
- **文脈をとらえるのが難しい**
  - 長時間の関係性をとらえるのが苦手

# Transformerの概要

- **Transformerとは？**

- 2017年に導入されたディープラーニングモデルで、

- 主に自然言語処理の分野で使われる

- RNNと同様に、自然言語などの時系列データを処理するように設計されているが、RNNで用いる再帰、CNNで用いる畳み込みは使わない

- Attention層のみで構築される

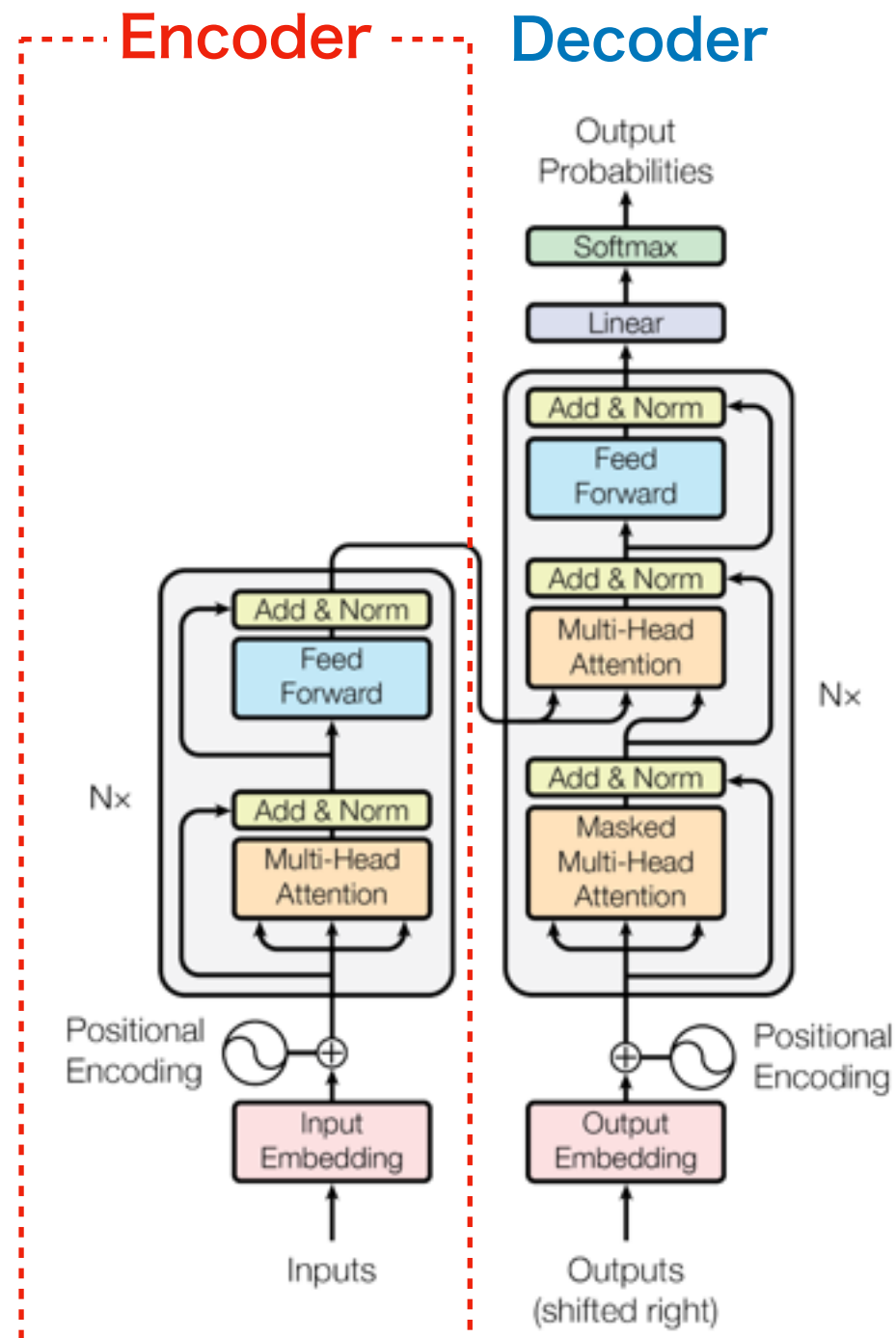
- 翻訳やテキストの要約など、様々なタスクで利用可能

- 並列化が容易であり、訓練時間を大きく削減できる

# Transformerの論文

- **Attention Is All You Need**  
→ <https://arxiv.org/abs/1706.03762>
- 「Attention」は時系列データの特定の部分に  
注意を向けるように学習させていく方法

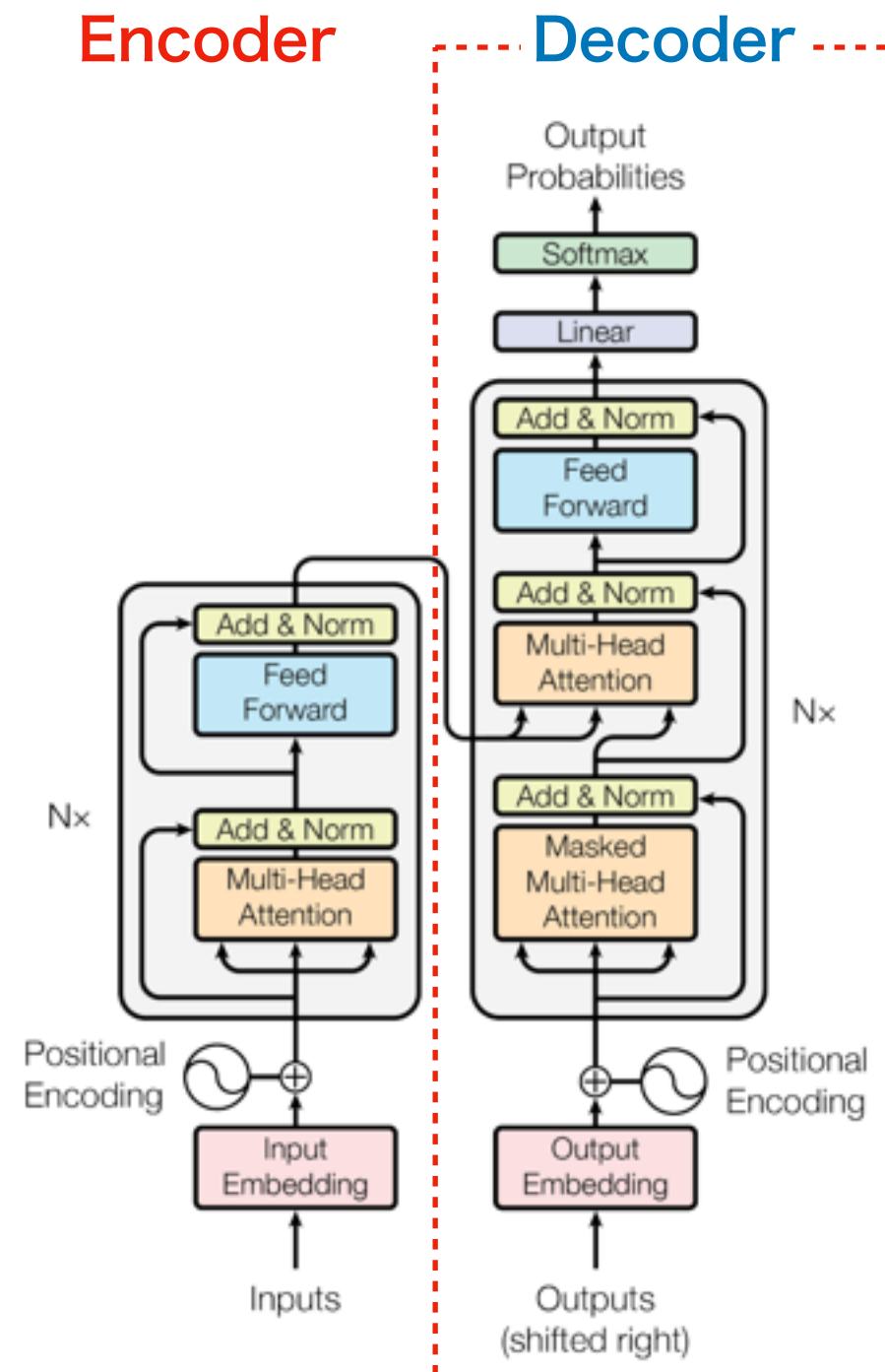
# Transformerのモデル



## Encoderの構造

1. Embedding層により入力文章をベクトルに圧縮
  2. Positional Encoder層によって位置情報を加える
  3. Multi-Head Attention層
  4. normalization（正規化）など
  5. Positionwise fully connected feed-forward network
  6. normalization（正規化）など
- 3-6を6回繰り返す

# Transformerのモデル



## Decoderの構造

1. Embedding層により入力文章をベクトルに圧縮
  2. Positional Encoder層によって位置情報を加える
  3. Multi-Head Attention層
  4. normalization（正規化）など
  5. Multi-Head Attention層（Encoderの入力を使用）
  6. normalization（正規化）など
  7. Positionwise fully connected feed-forward network
  8. normalization（正規化）など
- 3-8を6回繰り返す



# Transformerの構成要素

- **Attention**
  - Self-Attention
  - SourceTarget-Attention
  - Masked Multi-Head Attention
  - Multi-Head Attention
- **Position-wise Feedforward Network**
- **Positional Encoding**

# Positionwise fully connected feed-forward network

- **Positionwise fully connected feed-forward network**
  - 2 層の全結合ニューラルネットワーク
  - 単語の位置ごとに個別の順伝播ネットワーク
  - 他単語との影響関係を排除
  - パラメータは全てのネットワークで共通

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

Attention Is All You Need, Ashish, V. et al. (2017) より引用

# Positional Encoding

- **Positional Encoding**

→ 「単語の位置」の情報を加える

$$PE_{(pos, 2i)} = \sin(pos/10000^{2i/d_{\text{model}}})$$

$$PE_{(pos, 2i+1)} = \cos(pos/10000^{2i/d_{\text{model}}})$$

pos: 単語の位置    2i, 2i+1: Embedding の何番目の次元か     $d_{\text{model}}$ : 次元数

# Attentionの概要



# ChatGPT に聞いてみる

「Attentionって何ですか？」

→ ?

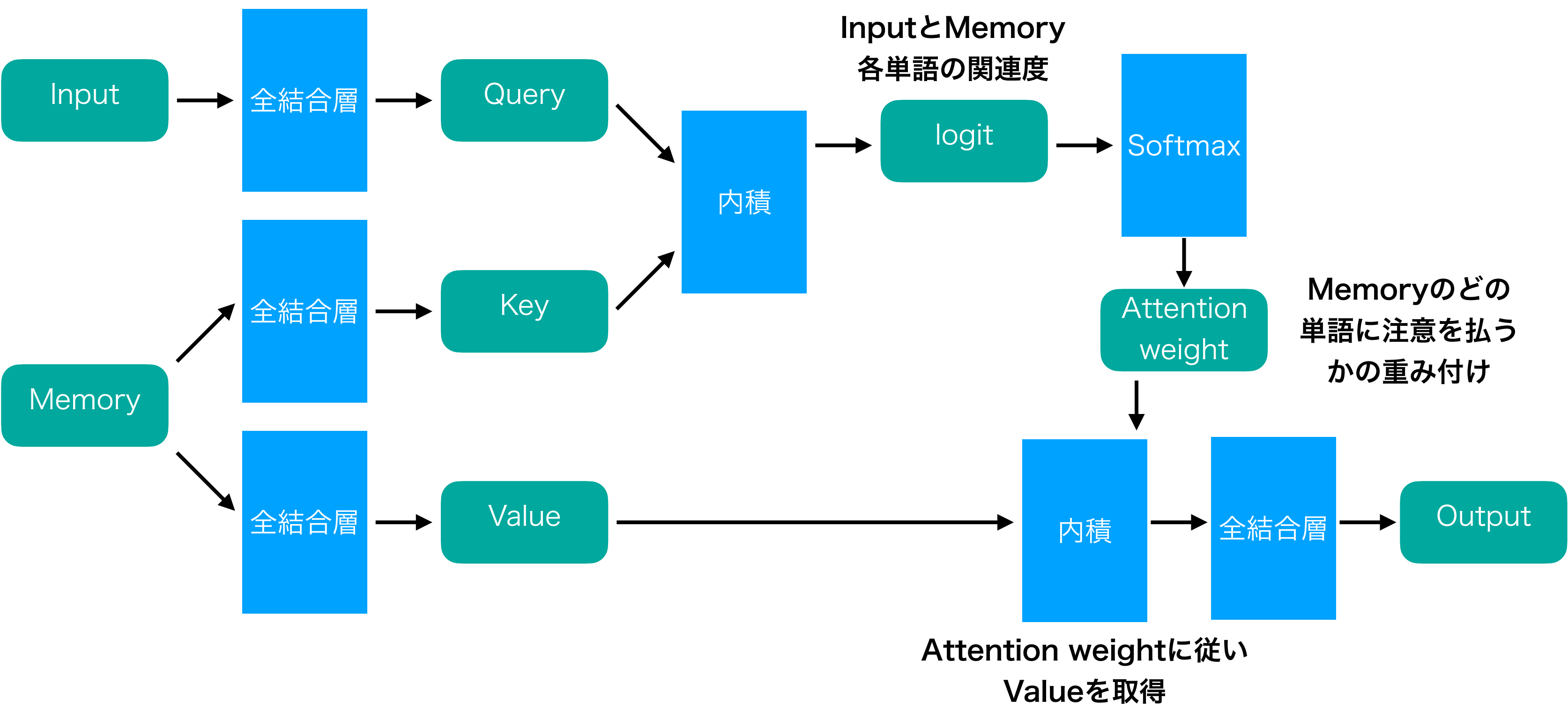
<https://chat.openai.com/>

# Attentionとは？

- **Attentionとは？**
  - 文章中のどの単語に注目すればいいかを表すスコア
  - Query、Key、Valueの3つのベクトルで計算される
- **Query**
  - Inputのうち「検索をかけたいもの」
- **Key**
  - 検索対象とQueryの近さを測る
- **Value**
  - Keyに基づき、適切なValueを出力する



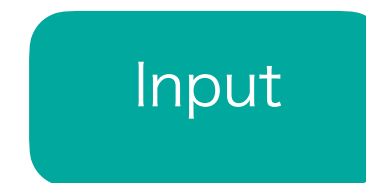
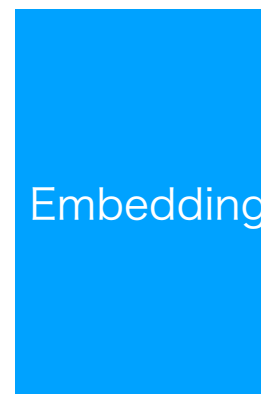
# Attentionとは？



# InputとMemory

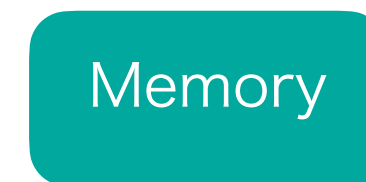
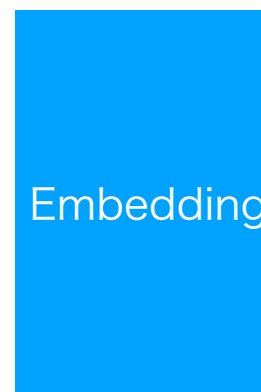
得意 な スポーツ は ？

(各単語はidで表される)

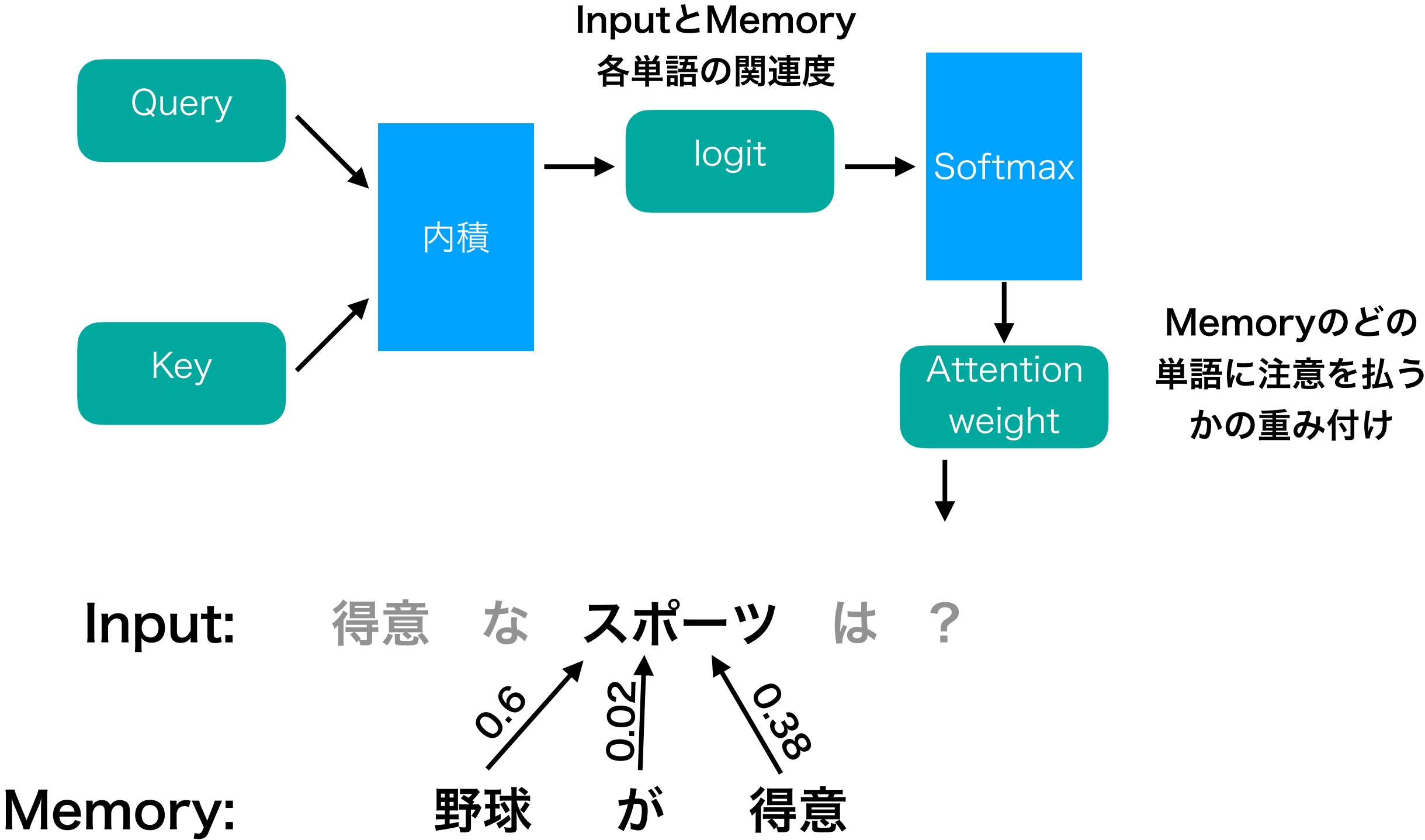


野球 が 得意

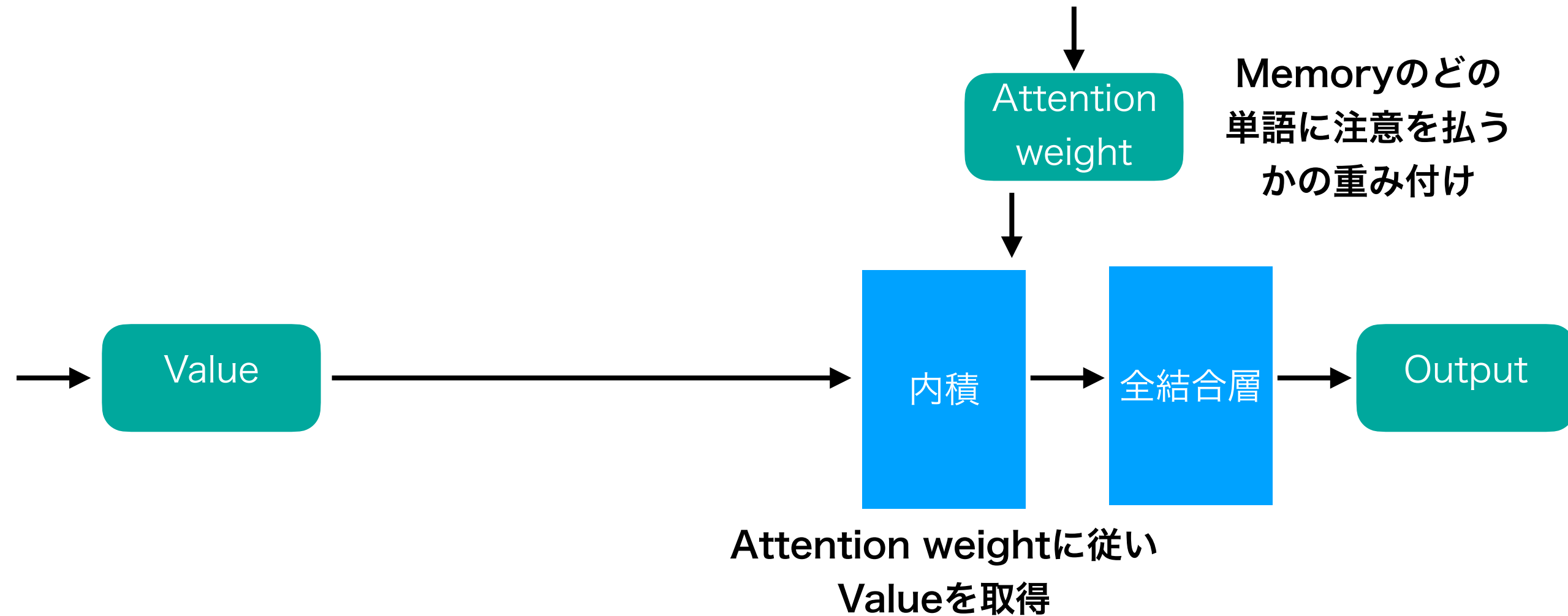
(各単語はidで表される)



# Attention weightの計算



# Valueと内積



Input: 得意 な スポーツ は ?

$$\begin{aligned}\text{内積} &= \text{Value}(\text{野球}) \times 0.6 \\ &\quad + \text{Value}(\text{が}) \times 0.02 \\ &\quad + \text{Value}(\text{得意}) \times 0.38\end{aligned}$$

Memory:

野球 が 得意

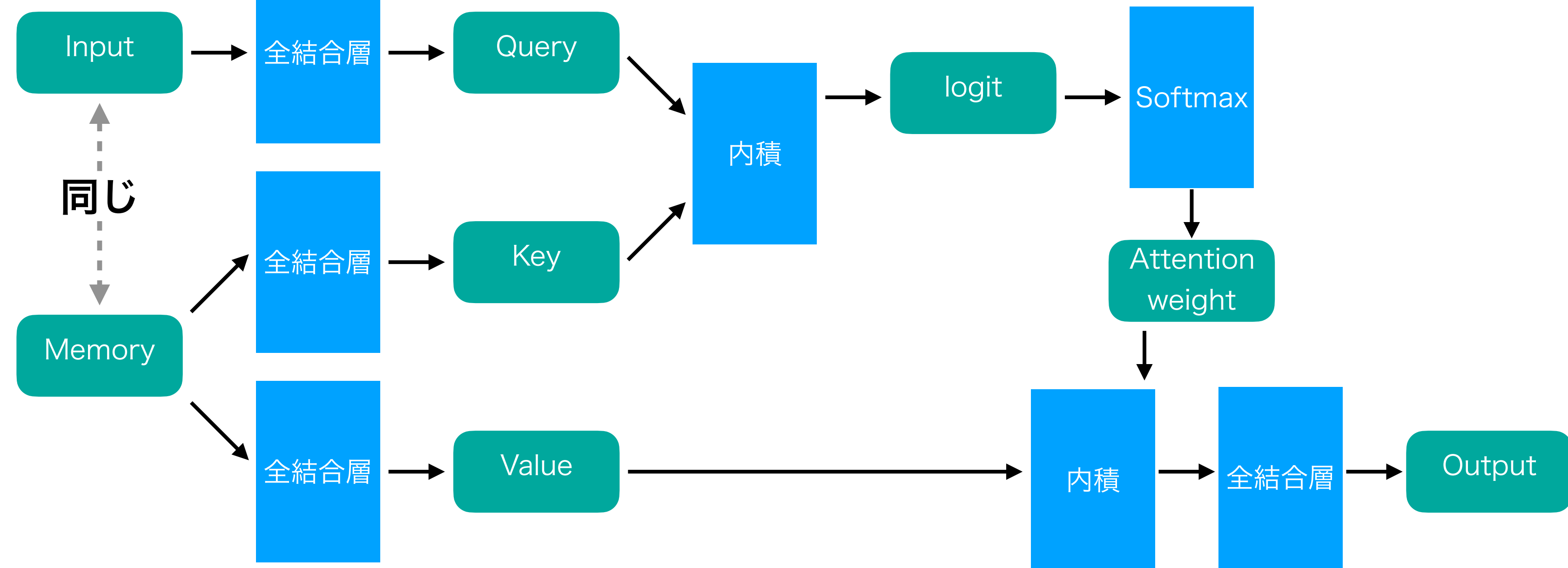
0.6 0.02 0.38

# Self-Attention

- **Self-Attention**

- InputとMemoryが同一のAttention

- 文法の構造や、単語同士の関係性などを獲得するのに使用される

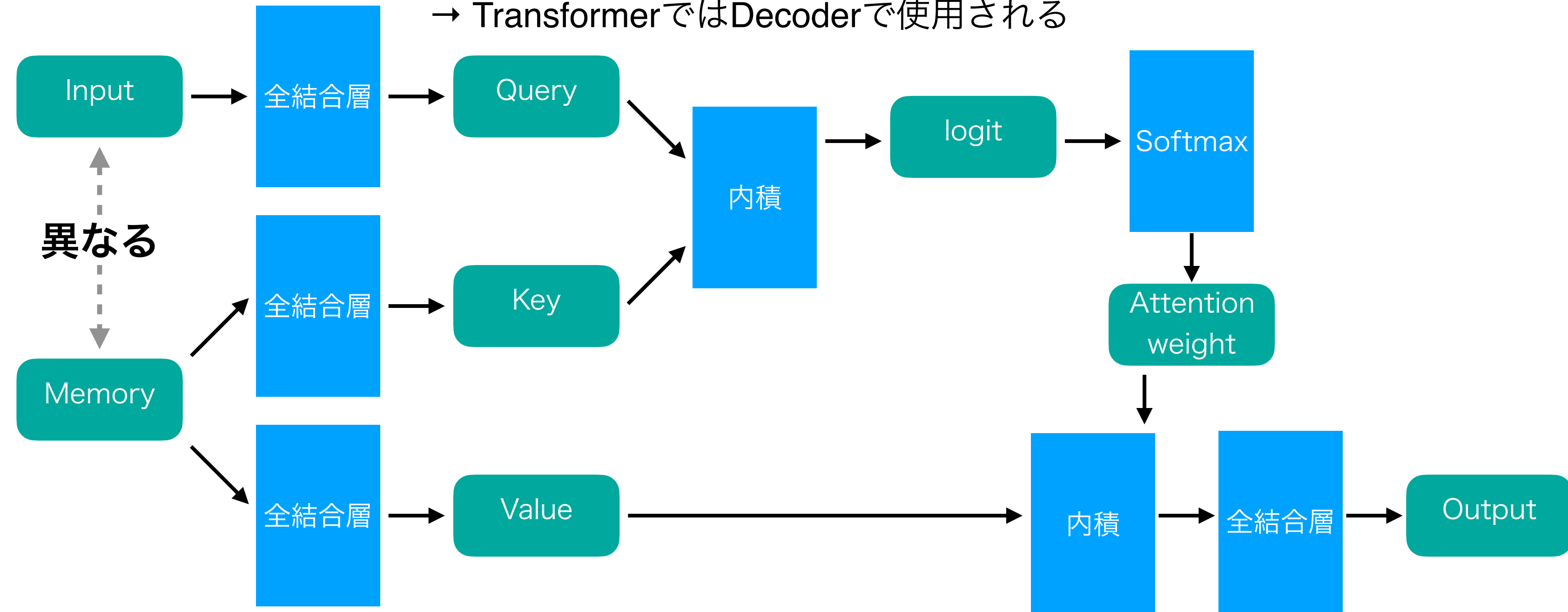


# SourceTarget-Attention

- **Self-Attention**

→ InputとMemoryが異なるAttention

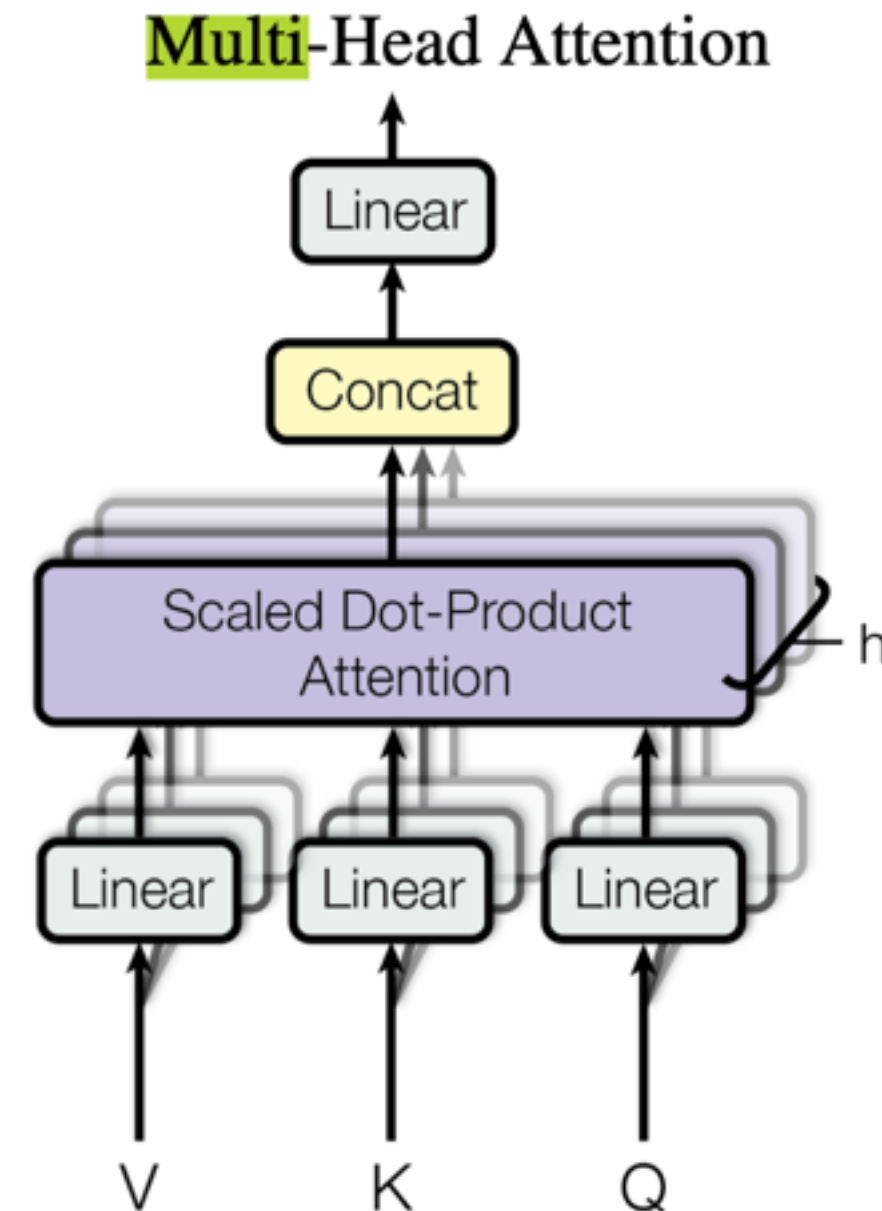
→ TransformerではDecoderで使用される





# Multi-Head Attention

- **Multi-Head Attention**
  - Attentionを並行に並べる
  - それぞれのAttentionはHeadと呼ばれる
  - 「Attention Is All You Need」では  
Multi-Head化による性能の向上が述べられている



# Masked Multi-Head Attention

- **Masked Multi-Head Attention**

- 特定の key に対して、

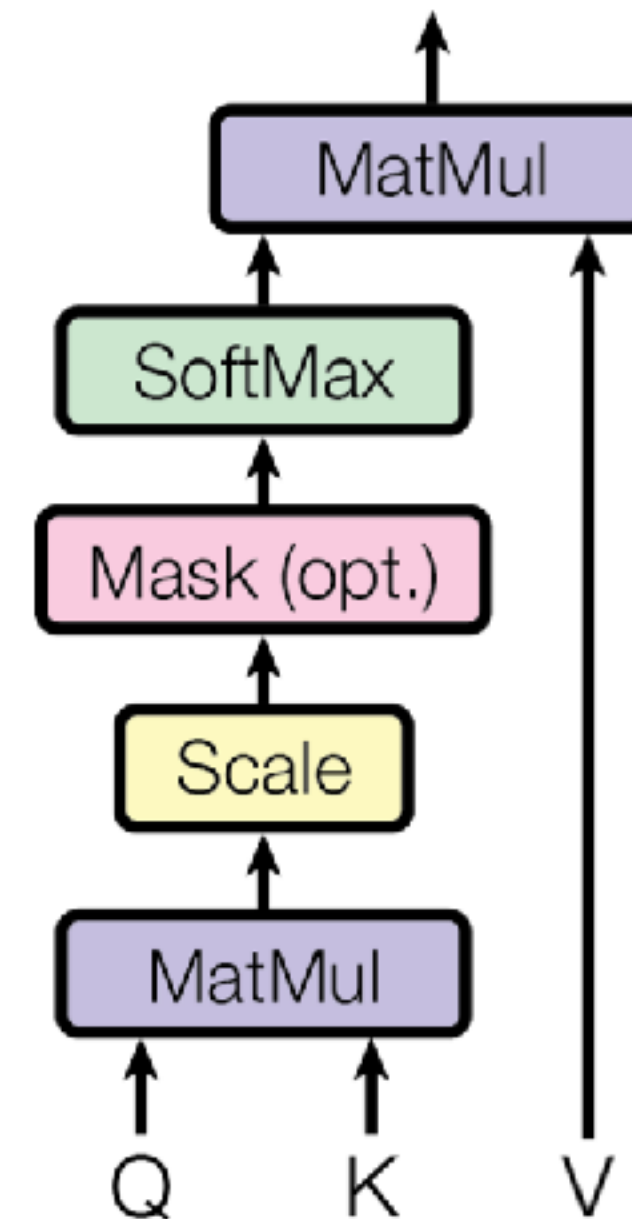
- Attention weight を0にする

- TransformerではDecoderで使われる

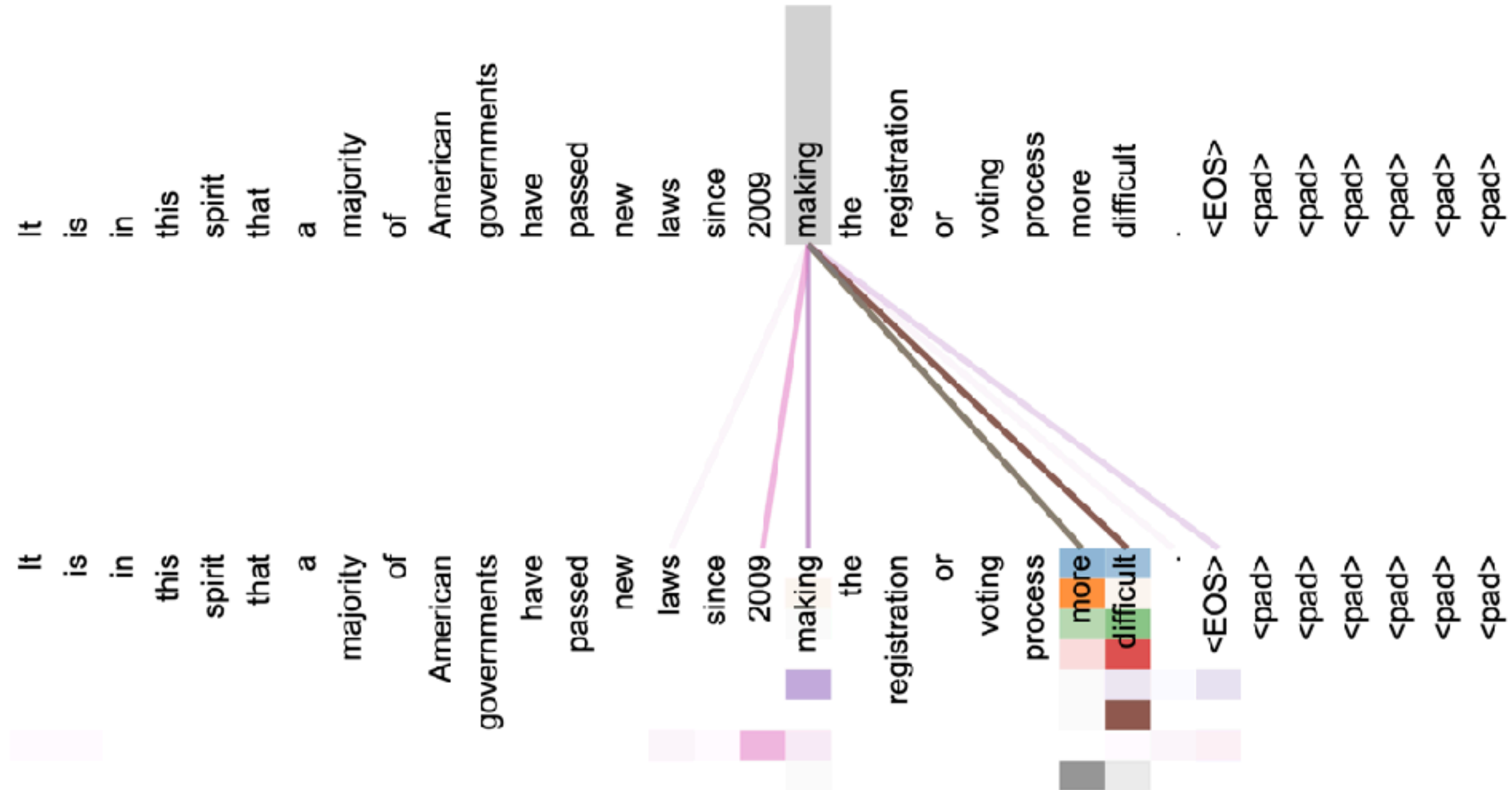
- 入力した単語が先読みを防ぐために、

- 情報をマスクで遮断する

- 言わば、「カンニング」を防ぐ



# Attentionの可視化



異なる色は異なるAttentionのHeadを表す

Attention Is All You Need, Ashish, V. et al. (2017) より引用

# Transformerの利用





# BERTの概要

- **BERT (Bidirectional Encoder Representation from Transformers) とは？**
  - 2018年の後半にGoogleから発表された、  
自然言語処理のための新たなディープラーニングのモデル
  - Transformerがベースとなっている
  - 様々な自然言語処理タスクでファインチューニングが可能
  - 従来の自然言語処理タスクと比較して、高い汎用性

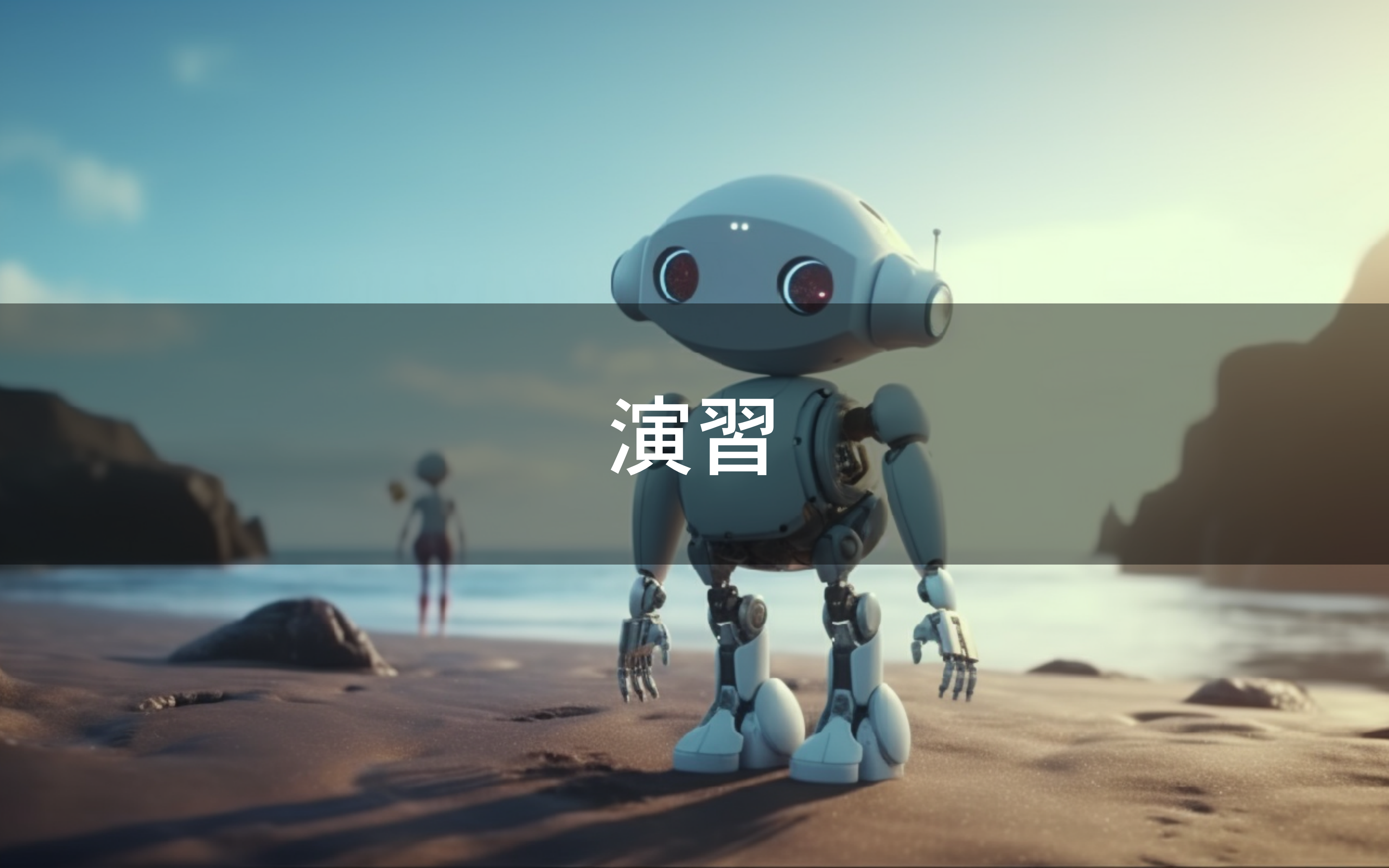


# Transformerの利用



- 01\_simple\_bert.ipynb

# 演習



# 演習


- 02\_exercise.ipynb

# 次回の内容

Section1. LLMの概要

Section2. ニューラルネットワークの仕組み

Section3. Transformerの仕組み

 **Section4. LLMの仕組み**