

AIに携わる技術者を どのように教育するか

東京都市大学 教授
法政大学 兼任講師

神野 健哉

教授
神野 健哉 (Kenya Jin'no)

1986 法政大学工学部電気工学科 電気電子専攻 入学
1991 同大学 電気工学科 卒業
1991 法政大学大学院工学研究科電気工学専攻博士前期課程 入学
1993 同大学院 電気工学専攻 博士前期課程 修了
1993 法政大学大学院工学研究科電気工学専攻博士後期課程 入学
1996 同大学院 電気工学専攻 博士後期課程 修了 博士(工学)

現在、東京都市大学 情報工学部 知能情報工学科 教授
担当科目：機械学習、アルゴリズム設計、データサイエンス応用
機械学習特論

法政大学 電気電子工学科 兼任講師 (線形回路とシステム)
法政大学大学院 理工学研究科 兼任講師 (機械学習特論)
東京理科大学 情報工学科 非常勤講師 (オブジェクト指向開発)

電子情報通信学会 フェロー
信号処理学会 理事
IEEE
人工知能学会
情報処理学会



Pythonによる アルゴリズム設計

Algorithm Design
with Python

神野健哉 [著]
Kenya Jinno

コロナ社

神野健哉 [著]
Kenya Jinno



Python で プログラミングして 理解する 機械学習 アルゴリズム

近代科学社

Pythonによるアルゴリズム設計

出版社：コロナ社

ISBN-13：978-4339029307

Pythonでプログラミングして理解する
機械学習アルゴリズム

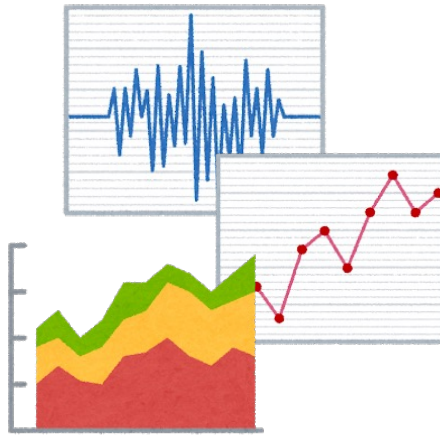
出版社：近代科学社

ISBN-13：978-4764906365

機械学習



データ収集

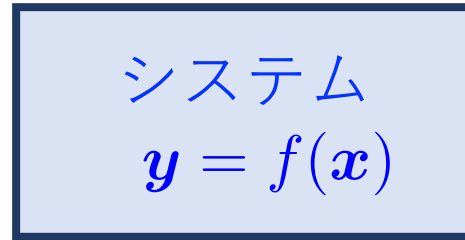


生データ



特徴ベクトル

入力データ x



出力データ y



重要な点

- ・ 生データの量・質
- ・ 特徴ベクトルの質
- ・ データに関する知識

教師あり学習：教師データとの誤差を最小化
教師なし学習：評価関数値を最小（最大）化

機械学習の歴史



C. E. Shannon, "A Chess-Playing Machine",
Scientific American, Vol. 182, No. 2, pp. 48-51, 1950.

情報理論の提唱者

C. E. Shannon, "A Mathematical Theory of Communication",
Bell System Technical Journal,
vol. 27, pp. 379-423, 623-656, 1948

Claude Elwood Shannon
1916 - 2001



Arthur Lee Samuel 1901-1990

1956年 IBM の Arthur Lee Samuel が商用コンピュータ
IBM701 上でチェッカープログラムを発表

- ・ $\alpha\beta$ 法による枝刈りする探索アルゴリズム
- ・ 次の手の選択は **minimax法**

→ 最善手で対戦すると必ず引き分ける

1959年 Arthur Lee Samuel が論文で Machine Learning という言葉を造語

Machine Learning (機械学習) とは

明示的にプログラムしなくても学習する能力をコンピュータに与える研究分野

Field of study that gives computers the ability to learn
without being explicitly programmed.

Arthur Lee Samuel, "Some Studies in Machine Learning Using the Game of Checkers,"
IBM Journal of Research and Development 3 (3): 210-229, 1959. DOI:10.1147/rd.33.0210

人工無能

チャットボット

Joseph Weizenbaum, "ELIZA — A Computer Program For the Study of Natural Language Communication Between Man And Machine", Communications of the ACM, vol. 9, no. 1, pp. 36–45, 1966. DOI:10.1145/365153.365168

```
=====
EEEEEEEE L      IIIIII ZZZZZZZ      AAA
E         L      I       Z          A   A
E         L      I       Z          A   A
EEEEEE   L      I       Z          A   A
E         L      I       Z          AAAAAAA
E         L      I       Z          A   A
EEEEEEEE LLLLLLL IIIIII ZZZZZZ     A   A
=====
ELIZA > (PLEASE TYPE IN ALL CAPS) WHAT'S YOUR NAME DEAR ?
AMIT
ELIZA > HI, I'M ELIZA. WHAT DO YOU WANT TO TALK ABOUT ?
AMIT > CAN I TALK ABOUT MY PROBLEM ?
ELIZA > SURE... !
AMIT > DUE TO ON GOING EXAMS IN COLLEGE, I HAVE A LOT OF STRESS.
ELIZA > PLEASE RELAX A LITTLE. SLEEP WELL.
AMIT > THANKS FOR YOUR ADVICE.
ELIZA > NO MENTION.
AMIT > BYE
ELIZA > BYE AND KEEP IN TOUCH...
=====
```

ELIZE> あなたの名前は？

AMIT> AMIT

ELIZE> HI, 私は ELIZE. 何について話をしたい？

AMIT> 私の問題について話をしてもいい？

ELIZA> もちろん！

AMIT> 大学の試験が続いていて、ストレスが溜まっているんだけど

ELIZE> 少し落ち着いて。よく寝て

AMIT> アドバイスありがとう

ELIZE> とんでもない

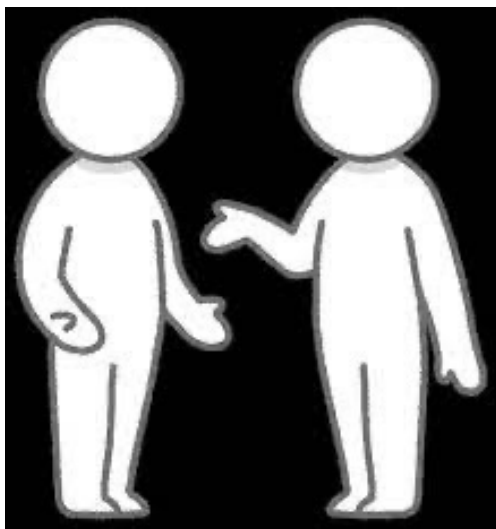
AMIT> さようなら

ELIZE> さようなら、これからもよろしく

相手の発言をあらかじめ用意しておいたパターンと比較し、パターンに合致した発言に対して、そのパターンに応じた発言を返答。基本的には相手の発言をオウム返しで再利用

本物の人間と会話しているような錯覚：**イライザ**効果

自然言語処理



自然言語：
人間が使う言葉

⇔ 人工言語

自然言語処理(Natural Language Processing; NLP)

人間が使用する言語（テキスト）を分析する技術

- ・ 文書分類
- ・ 機械翻訳
- ・ 文書要約
- ・ 質問応答
- ・ 対話

情報理論

- ・よくあることか、滅多にないかが重要
- ・事象が起きる確率と情報が関係



確率的な理論体系化

C. E. Shannon, "A Mathematical Theory of Communication",
Bell System Technical Journal,
vol. 27, pp. 379-423, 623-656, 1948



Claude Elwood Shannon
1916 - 2001

情報量の定義

- ・ 聞いて非常に驚く情報 \Rightarrow 情報量が大きい
- ・ 聞いても驚かない情報 \Rightarrow 情報量が小さい

事象 a の生起確率（発生確率） : $p(a)$

$$\text{情報量} : I(a) \propto \frac{1}{p(a)}$$

$p(a)$ が小さい \Rightarrow $I(a)$ が大きい

$p(a)$ が大きい \Rightarrow $I(a)$ が小さい

3. THE SERIES OF APPROXIMATIONS TO ENGLISH

To give a visual idea of how this series of processes approaches a language, typical sequences in the approximations to English have been constructed and are given below. In all cases we have assumed a 27-symbol “alphabet,” the 26 letters and a space.

1. Zero-order approximation (symbols independent and equiprobable).

XFOML RXKHRJFFJUJ ZLPWCFWKCYJ FFJEYVKCQSGHYD QPAAMKBZAACIBZLHJQD.

2. First-order approximation (symbols independent but with frequencies of English text).

OCRO HLI RGWR NMIELWIS EU LL NBNESEBYA TH EEI ALHENHTTPA OOBTTVA NAH BRL.

3. Second-order approximation (digram structure as in English).

ON IE ANTSOUTINYS ARE T INCTORE ST BE S DEAMY ACHIN D ILONASIVE TU-COOWE AT TEASONARE FUSO TIZIN ANDY TOBE SEACE CTISBE.

4. Third-order approximation (trigram structure as in English).

IN NO IST LAT WHEY CRATICT FROURE BIRS GROCID PONDENOME OF DEMONSTURES OF THE REPTAGIN IS REGOACTIONA OF CRE.

5. First-order word approximation. Rather than continue with tetragram, . . . , n -gram structure it is easier and better to jump at this point to word units. Here words are chosen independently but with their appropriate frequencies.

REPRESENTING AND SPEEDILY IS AN GOOD APT OR COME CAN DIFFERENT NATURAL HERE HE THE A IN CAME THE TO OF TO EXPERT GRAY COME TO FURNISHES THE LINE MESSAGE HAD BE THESE.

6. Second-order word approximation. The word transition probabilities are correct but no further structure is included.

THE HEAD AND IN FRONTAL ATTACK ON AN ENGLISH WRITER THAT THE CHARACTER OF THIS POINT IS THEREFORE ANOTHER METHOD FOR THE LETTERS THAT THE TIME OF WHO EVER TOLD THE PROBLEM FOR AN UNEXPECTED.

意味の近い単語の抽出

- ある単語の意味は、その単語と共に出現している単語群によって特徴付けられる

You shall know a word by the company it keeps.

⇒ 分布仮説 (distribution hypothesis)

Z. S. Harris, "Distributional structure," *Word*, vol. 10, no. 2-3, pp. 146-162, 1954

J. R. Faith, "A synopsis of linguistic theory," *Studies in Linguistic Analysis*, pp. 1-32, 1957

Today's lecture on machine learning is
about natural language processing.

today's / lecture / on / machine / learning / is /
about / natural / language / processing / .

today's / lecture / on / machine / learning / is /
about / natural / language / processing / .

	today's	lecture	on	machine	learning	is	about	natural	language	processing	
today's	0	1	0	0	0	0	0	0	0	0	w_1
lecture	1	0	1	0	0	0	0	0	0	0	w_2
on	0	1	0	1	0	0	0	0	0	0	w_3
machine	0	0	1	0	1	0	0	0	0	0	w_4
learning	0	0	0	1	0	1	0	0	0	0	w_5
is	0	0	0	0	1	0	1	0	0	0	w_6
about	0	0	0	0	0	1	0	1	0	0	w_7
natural	0	0	0	0	0	0	1	0	1	0	w_8
language	0	0	0	0	0	0	0	1	0	1	w_9
processing	0	0	0	0	0	0	0	0	1	0	w_{10}

単語共起行列 (window size=1)

$$\text{similarity}(w_j, w_k) = \frac{w_j \cdot w_k}{\|w_j\|_2 \|w_k\|_2}$$

単語共起行列の問題点

- ・ 特徴ベクトルの次元数増大
- ・ 巨大な疎行列



単語共起行列の次元圧縮

- ・ 特異値分解 (SVD; Singular Value Decomposition)

潜在意味解析 (LSA; Latent Semantic Analysis)

S. Deerwester, et. al. "Indexing by latent semantic analysis," J. Amer. Soc. Info. Sci., 41, 6, 391-407, 1990.

- ・ word2vec

T. Mikolov, et. al. "Efficient Estimation of Word Representations in Vector Space," Proc. ICLR2013, 2013.

word2vec

T. Mikolov, et. al., "Efficient Estimation of Word Representations in Vector Space," Proc. ICLR2013, 2013.

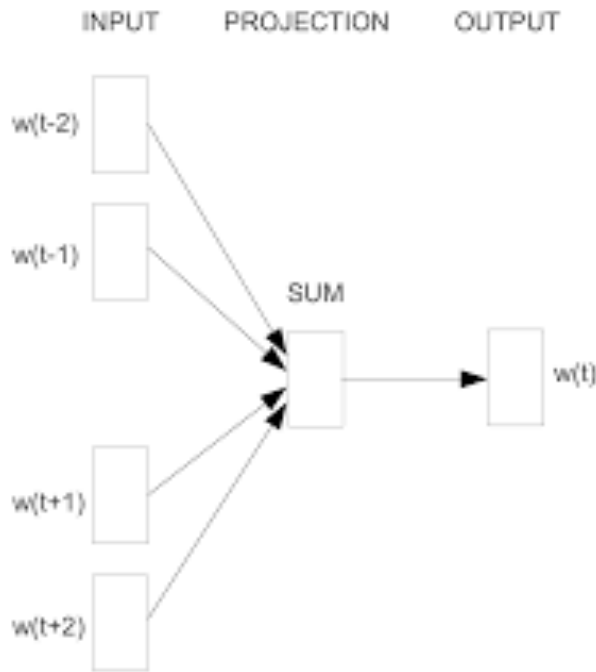
T. Mikolov, et.al., "Distributed Representations of Words and Phrases and their Compositionality," Proc. NIPS2013, vol. 2, pp. 3111-3119, 2013.

- **CBOW (Continuous Bag-of-Words) モデル**

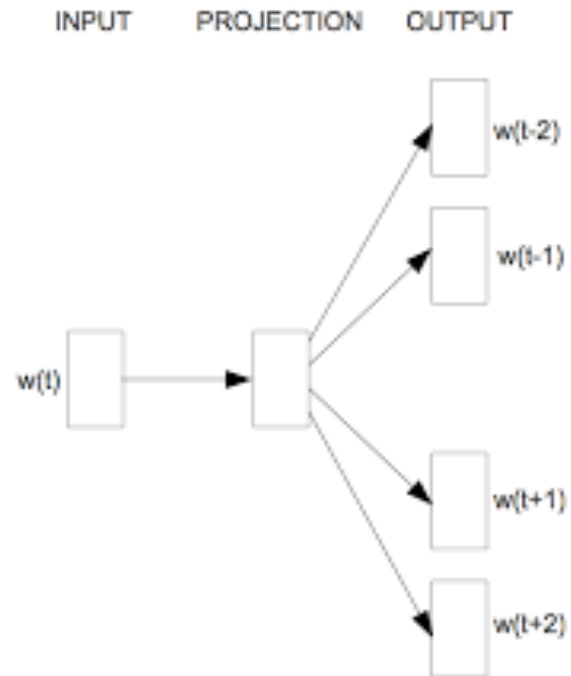
現在の単語の周辺の単語を用いて、現在の単語を予測

- **Skip-gram モデル**

現在の単語から周辺の単語を予測



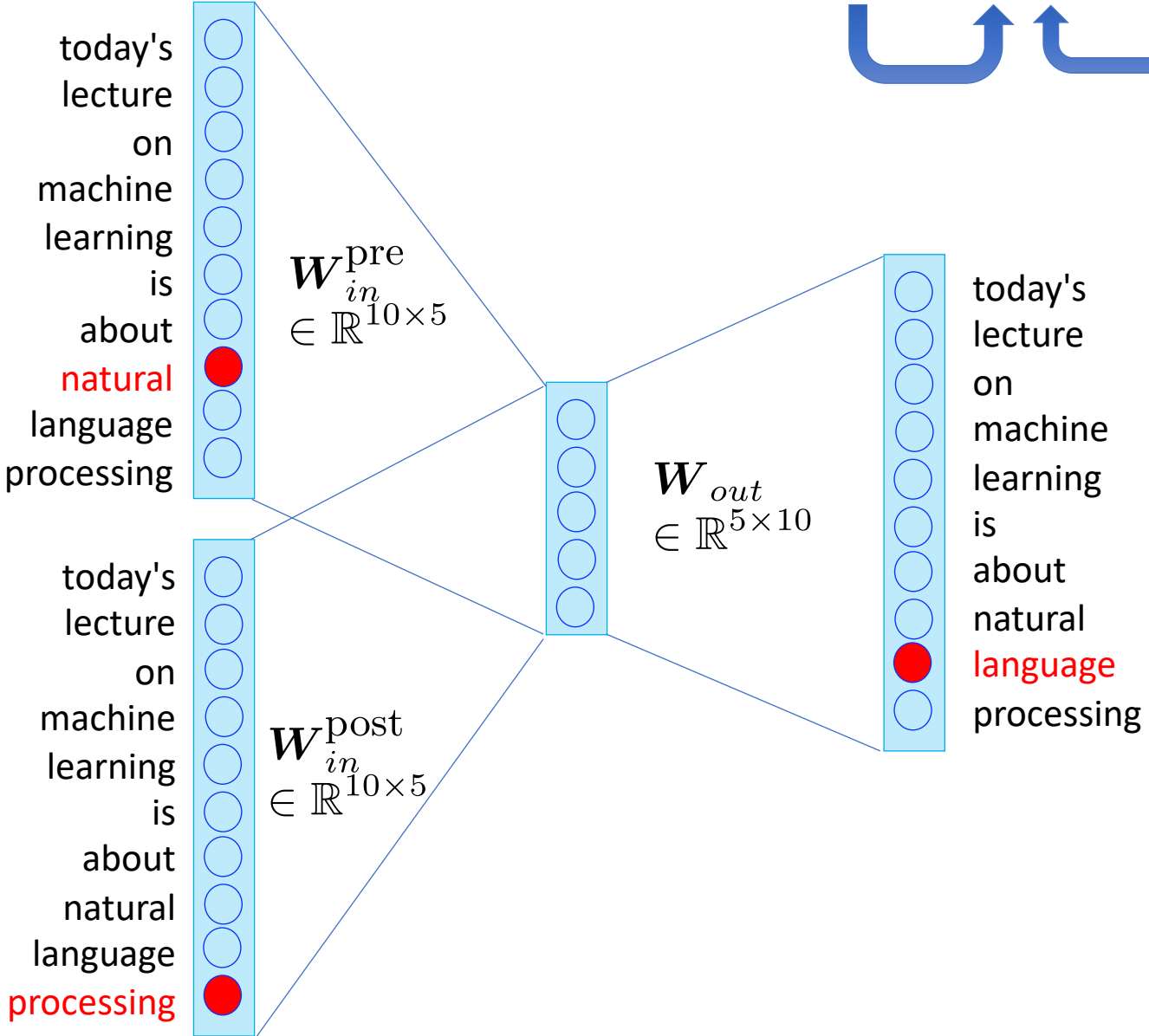
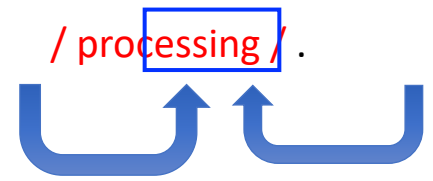
CBOW



Skip-gram

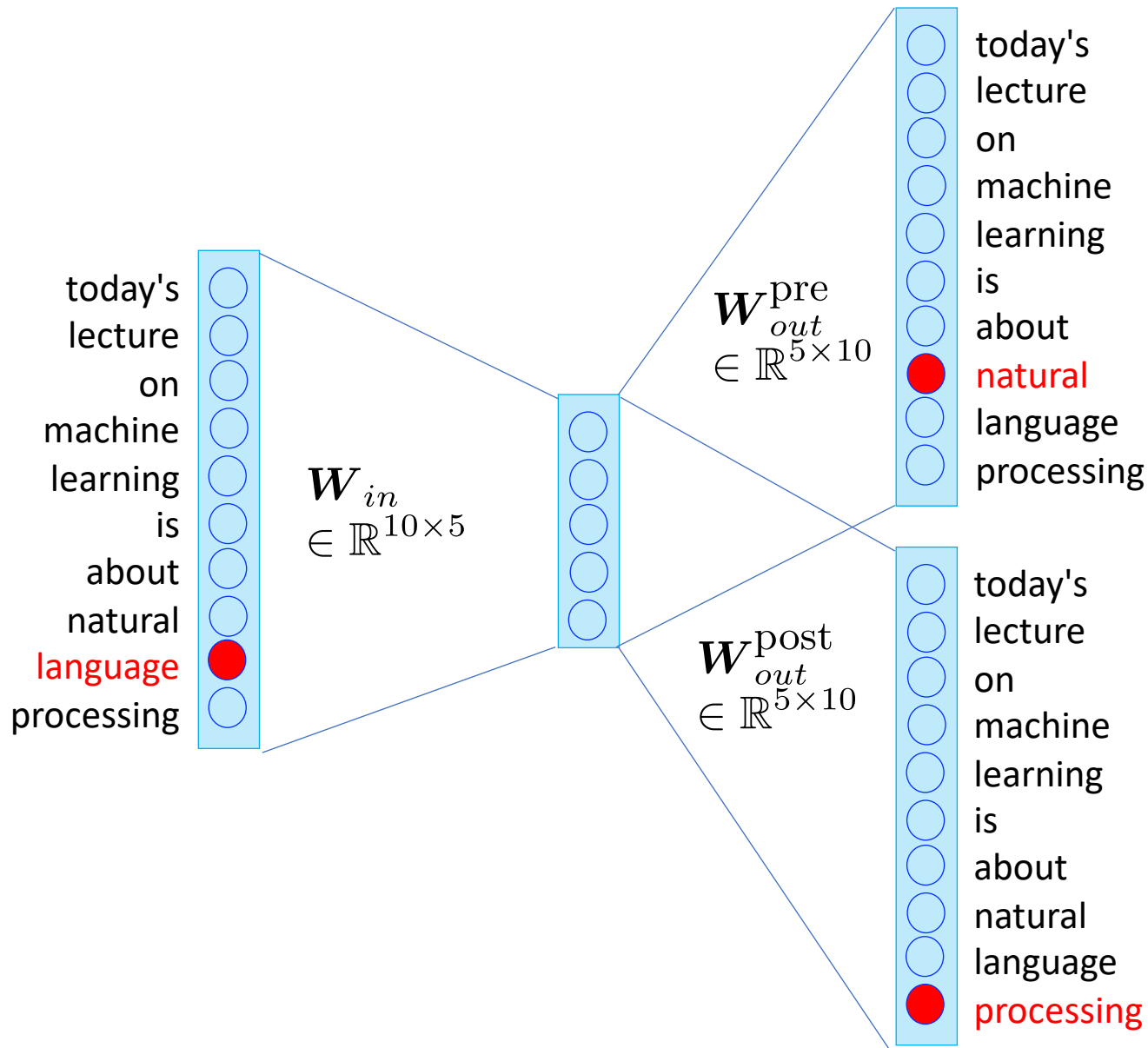
CBOW モデル

today's / lecture / on / machine / learning / is / about / natural / ? / processing / .



Skip-gram モデル

today's / lecture / on / machine / learning / is / about / ? / language / ? / .



類義語

分かち書きした日本語Wikipedia によるコーパスを
vector_size=32, min_count=10, window=5 で CBOW、Skip-gram で学習

「東京」の類義語

CBOW

```
results = wv.most_similar( positive=['東京'] )  
for result in results:  
    print( result )
```

```
('大阪', 0.9703967571258545)  
( '神戸', 0.9284552931785583)  
( '横浜', 0.9233023524284363)  
( '名古屋', 0.9053868651390076)  
( '札幌', 0.9029340744018555)  
( '福岡', 0.8727191686630249)  
( '小樽', 0.8702754378318787)  
( '仙台', 0.8675271272659302)  
( '熊本', 0.8505994081497192)  
( '鹿児島', 0.8490319848060608)
```

Skip-gram

```
results = wv.most_similar( positive=['東京'] )  
for result in results:  
    print( result )
```

```
('大阪', 0.9708964824676514)  
( '名古屋', 0.9396223425865173)  
( '札幌', 0.9280979037284851)  
( '横浜', 0.9249083399772644)  
( '東京八王子', 0.9184554815292358)  
( '新宿美術学院', 0.914902925491333)  
( '仙台青葉', 0.9091632962226868)  
( '大阪市梅田', 0.9087345004081726)  
( '福岡', 0.9073981642723083)  
( '静岡', 0.9063336253166199)
```

類義語

分かち書きした日本語Wikipedia によるコーパスを
vector_size=128, min_count=10, window=15 で CBOW、Skip-gram で学習

「king」 - 「man」 + 「woman」

CBOW

```
positive=['king', 'woman'], negative=['man']
```

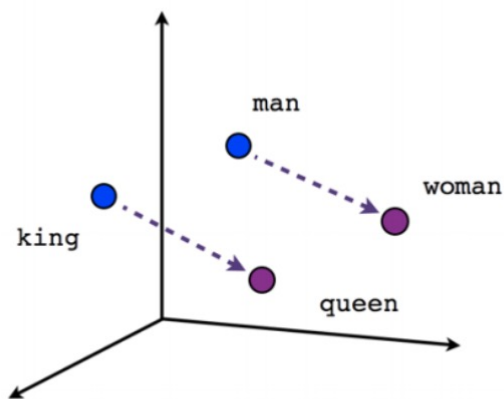
```
('queen', 0.8015924096107483)  
( 'princess', 0.6920022964477539)  
( 'regnant', 0.6867566704750061)  
( 'monarch', 0.6618798971176147)  
( 'throne', 0.6595994830131531)  
( 'queenship', 0.6397891044616699)  
( 'queen-mother', 0.6289969682693481)  
( 'prince', 0.6162654757499695)  
( 'regent', 0.6160593032836914)  
( 'empress', 0.6042488217353821)
```

Skip-gram

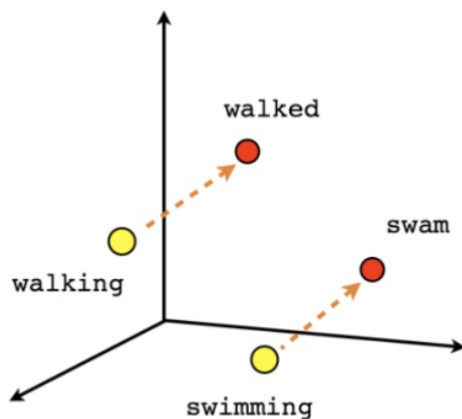
```
positive=['king', 'woman'], negative=['man']
```

```
('queen', 0.8003054857254028)  
( 'princess', 0.7231035232543945)  
( 'regnant', 0.7216320037841797)  
( 'throne', 0.7057614922523499)  
( 'jagellon', 0.704541802406311)  
( 'queen-consort', 0.7011317014694214)  
( 'regnant', 0.6940827965736389)  
( 'prince', 0.6903947591781616)  
( '(consort', 0.6891282796859741)  
( 'consort;', 0.6829870343208313)
```

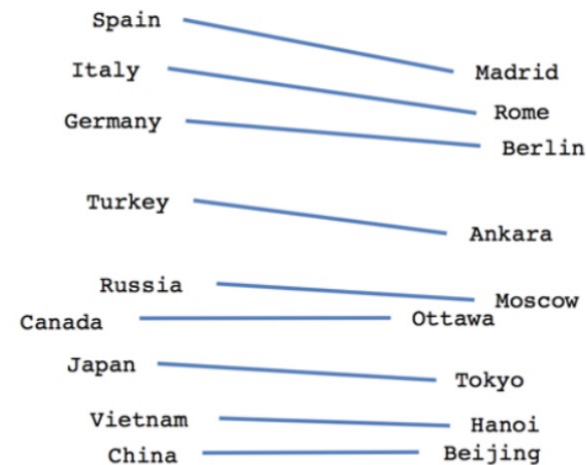
単語埋め込み word embeddings



Male-Female



Verb tense



Country-Capital

word2vec による 単語ベクトルの関係 (Mikolov et al.2013a; 2013b)

word2vec:

分散仮説に基づき単語に対して文脈非依存の（一意な）分散表現を学習するモデル

👉 単語埋め込み（Word Embedding）
単語 → ベクトル

👉 加法構成性（Additive Compositionality）
King - Man + Woman → Queen

word2vec の問題点:

- ・ 文脈に応じた意味の変化（多義語）を扱えない

下記の文章の「甘い」は意味が異なる

「このチョコレートは甘い」

「父は娘に甘い」

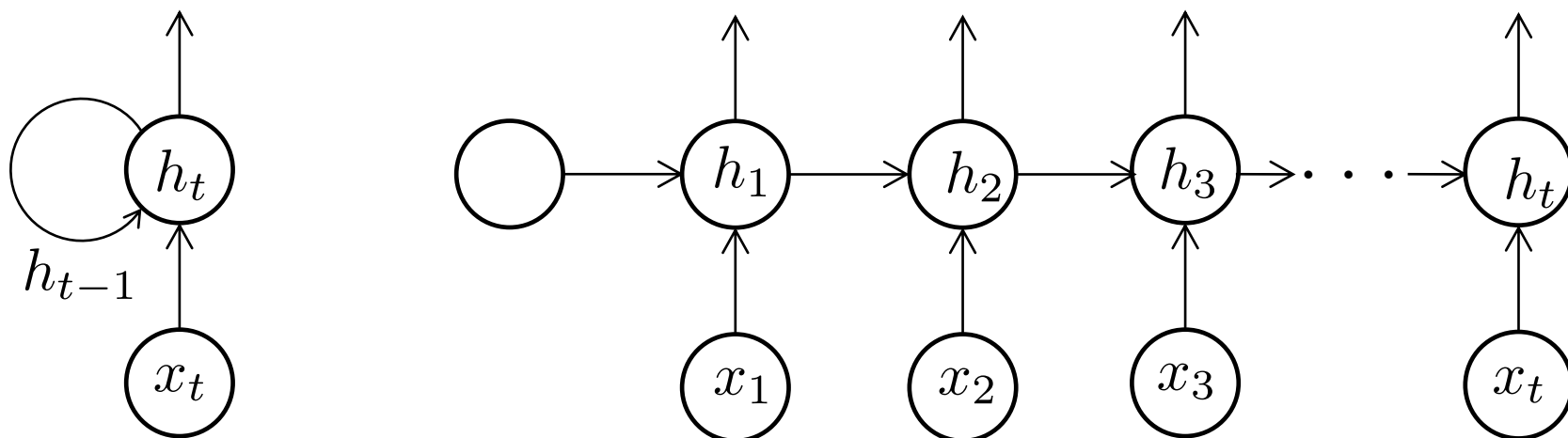
- ・ 分散表現で文章を表現する場合、分散表現の重み付き平均を利用するが、その場合語順が考慮されない

下記の文章の分散表現が同一となる

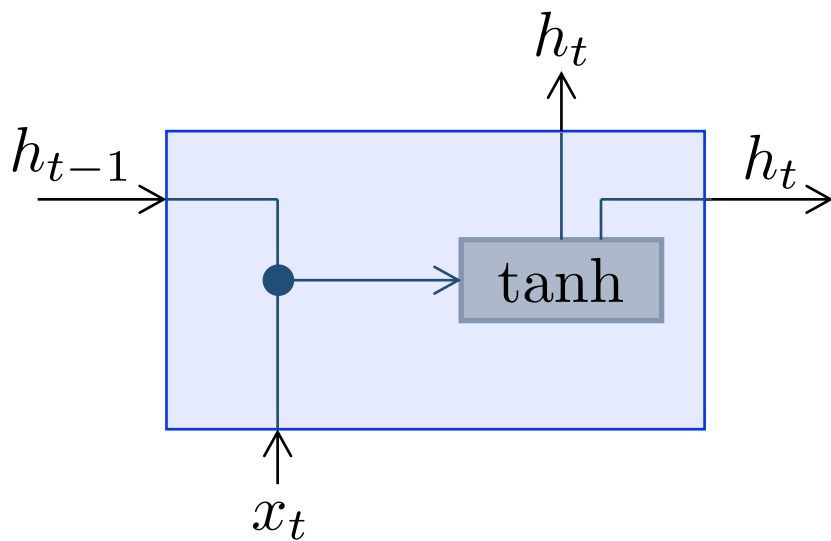
「太郎は二郎に本を貸した。」

「二郎は太郎に本を貸した。」

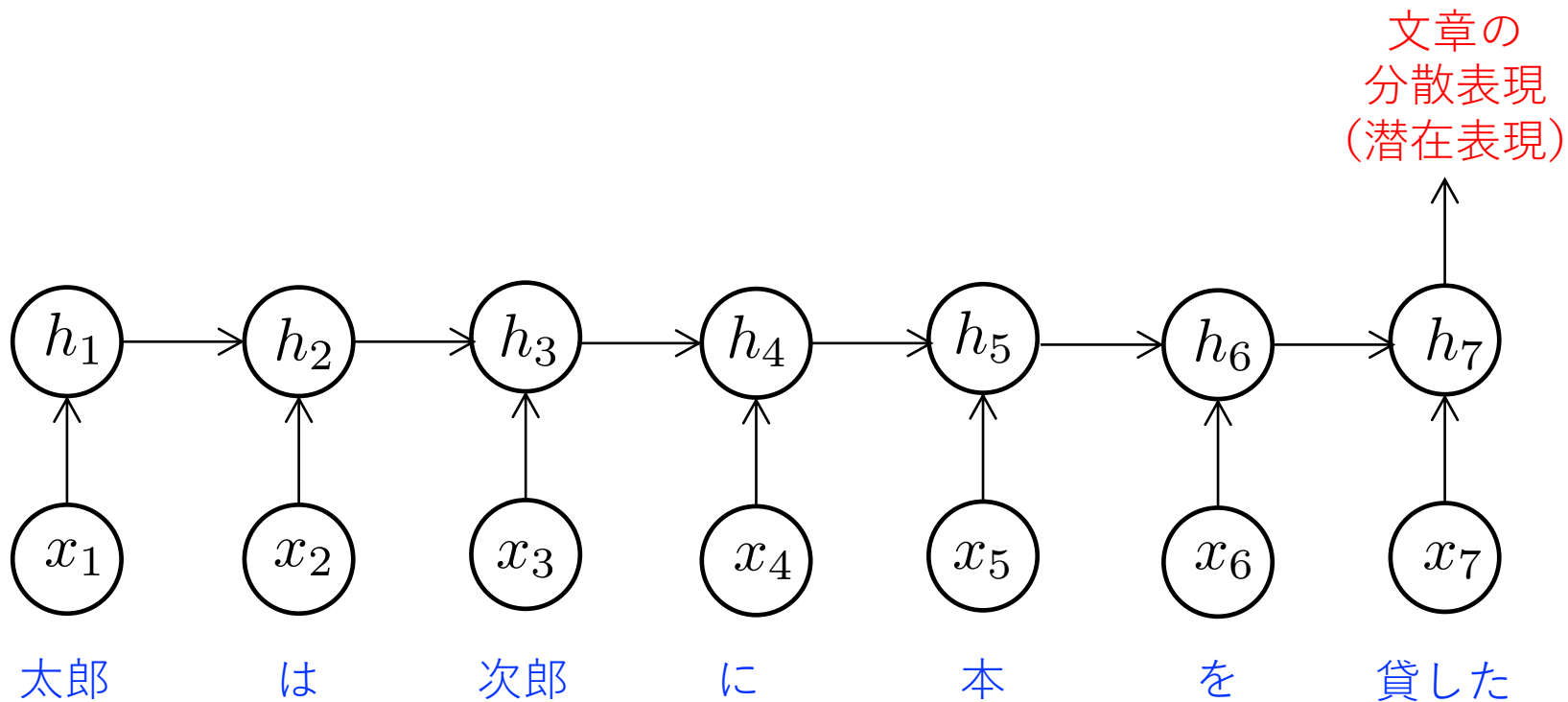
RNN (Recurrent Neural Network)



過去の出力結果を入力
⇒ 時間方向に 'Deep'

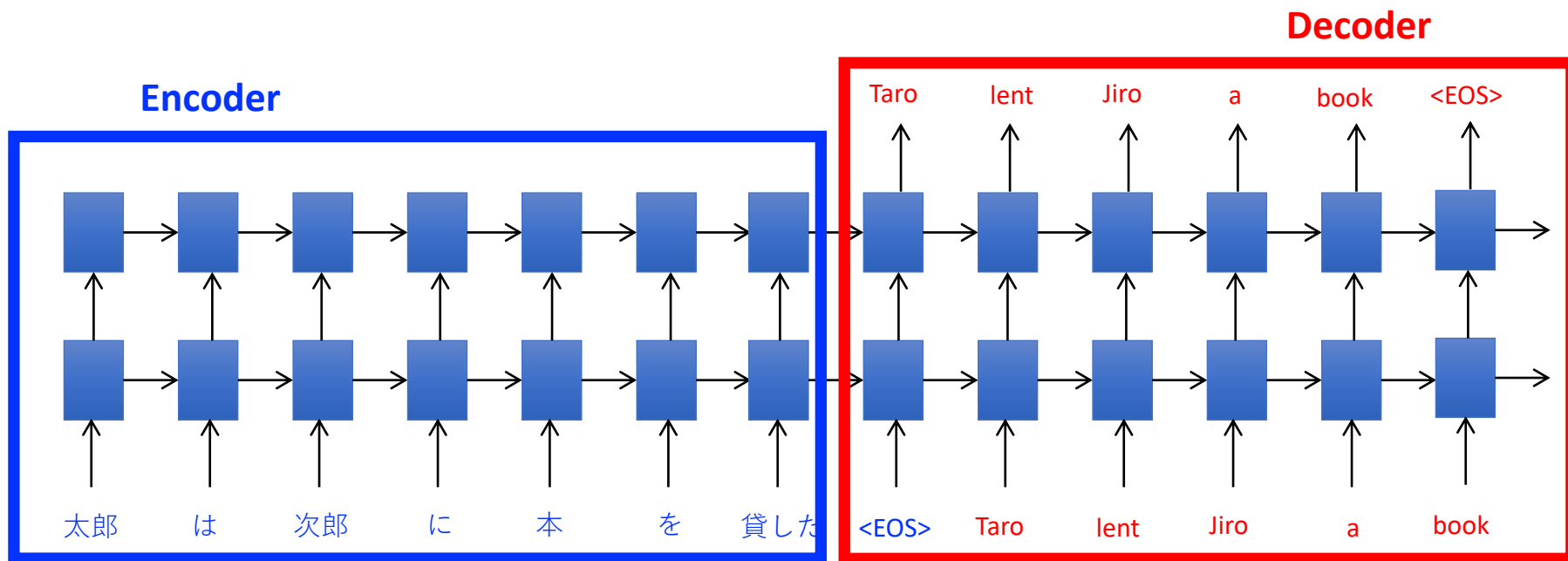


RNN (Recurrent Neural Network)



sequence-to-sequence モデル

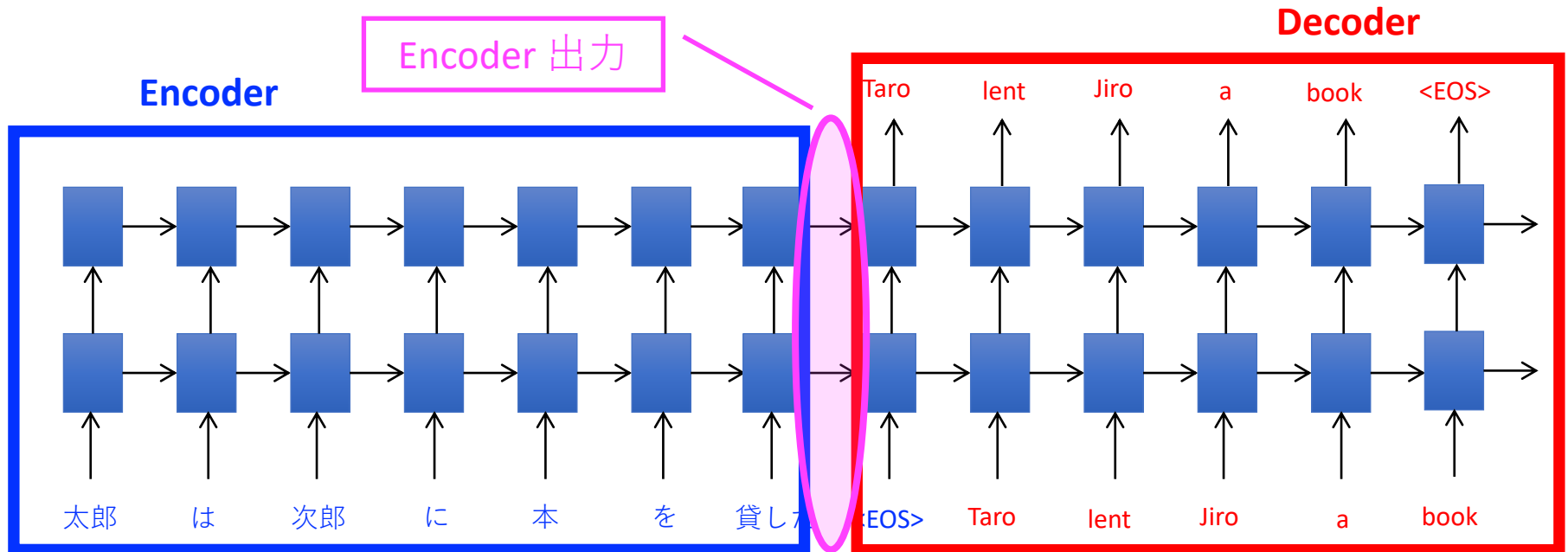
seq2seq



sequence-to-sequence モデル (Sutskever et al.2014)

事前に与えられた系列から次の要素を予測
更に次の予測を行う際には**自身が予測した値も入力**

seq2seq の問題点



sequence-to-sequence モデル (Sutskever et al.2014)

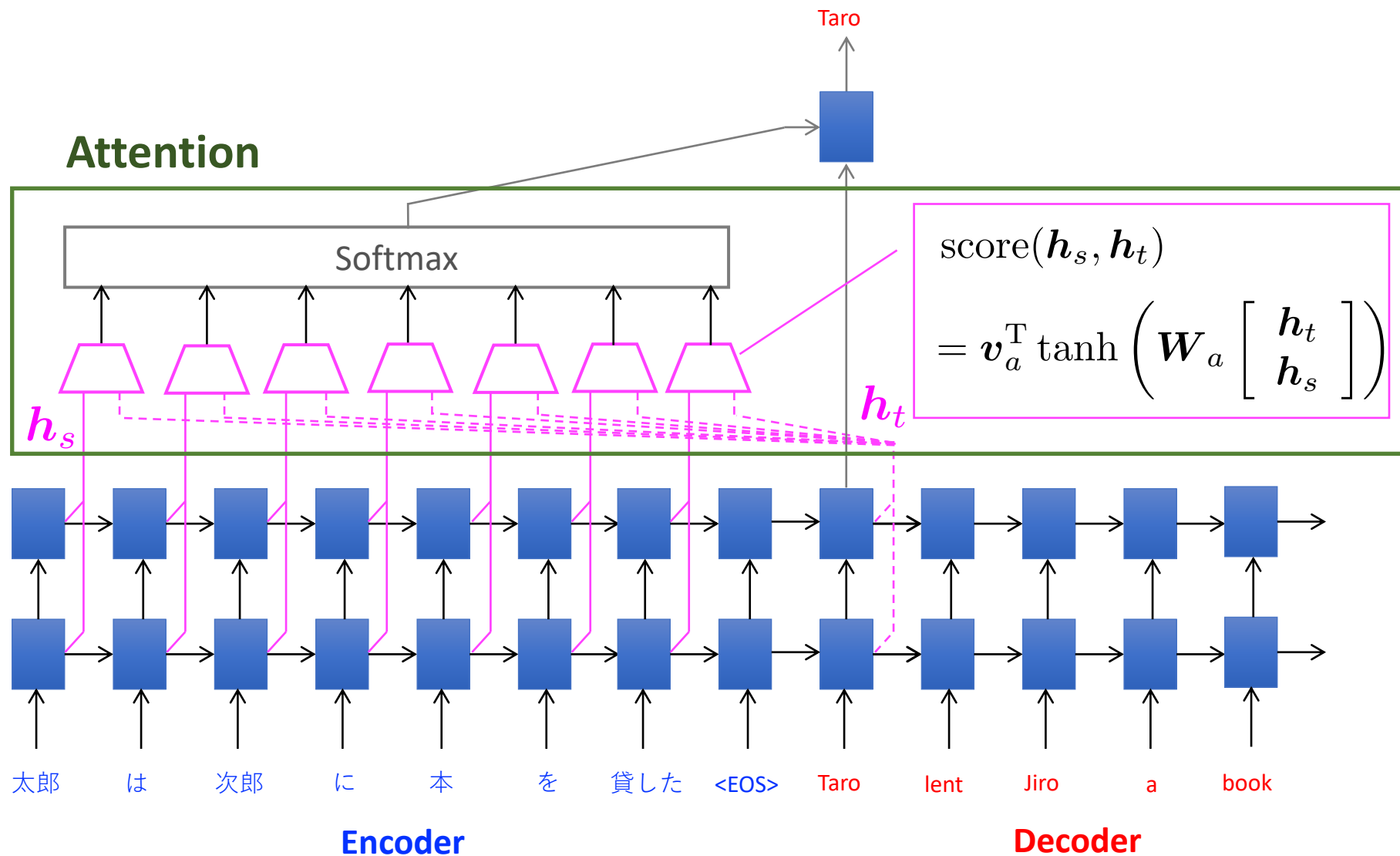
入力系列の情報を**Encoder**で圧縮したベクトルとして
Decoderに伝える

☞ 入力系列が長いと入力系列の情報を**Decoder**に
しっかりと伝えることが困難

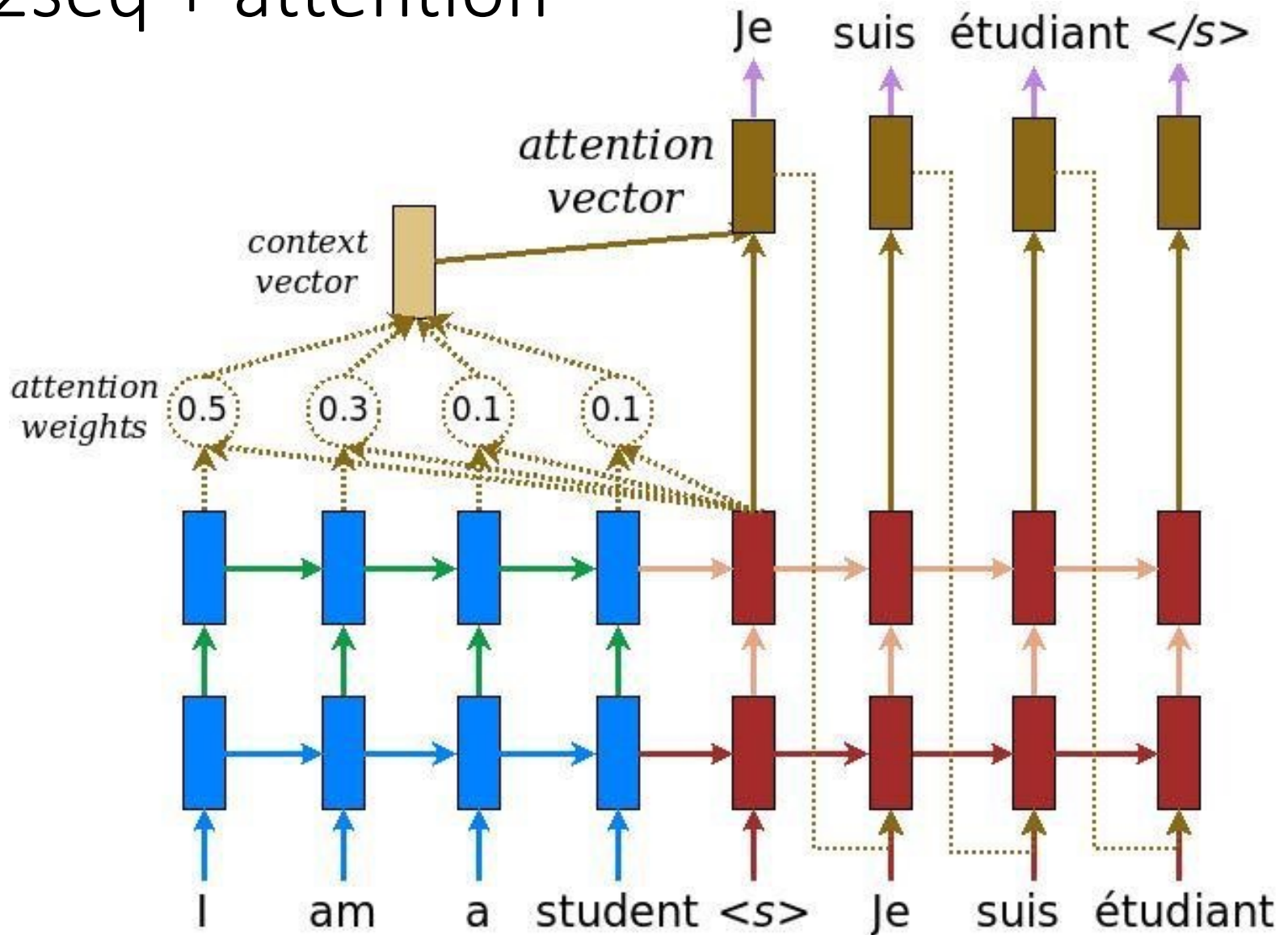
語順構造が異なる言語では性能が得られない

attention (注意機構)

Decode時に入力系列の情報を直接参照できるようにする仕組み



seq2seq + attention

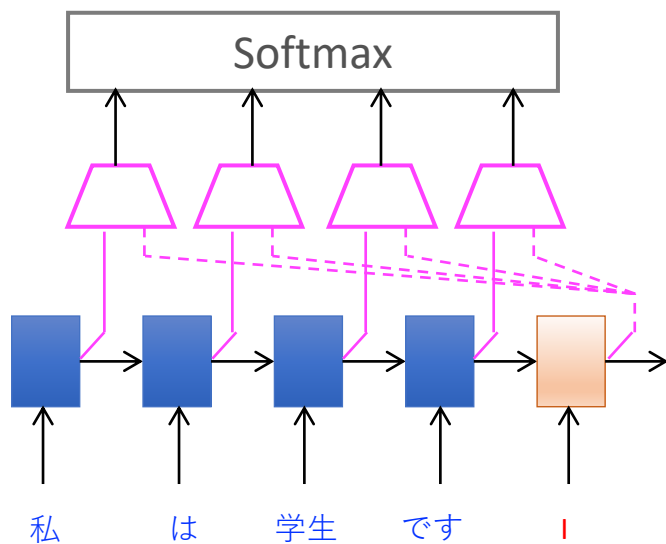


seq2seq + attention を機械翻訳に適用 (Google Translator)

機械翻訳精度が従来と比較して劇的に向上

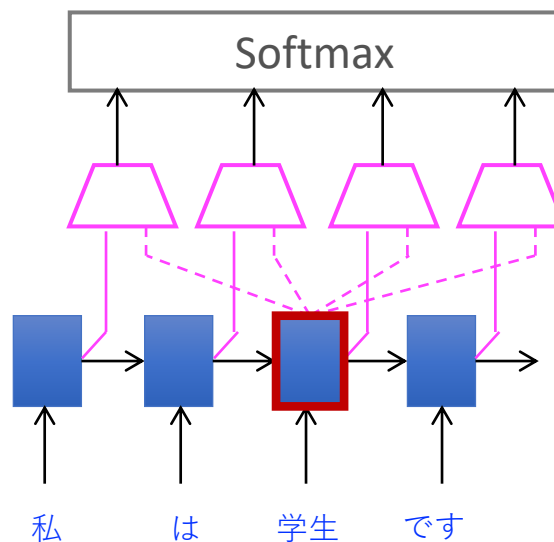
(Wu et al., 2016)

Self-Attention



異なる言語間

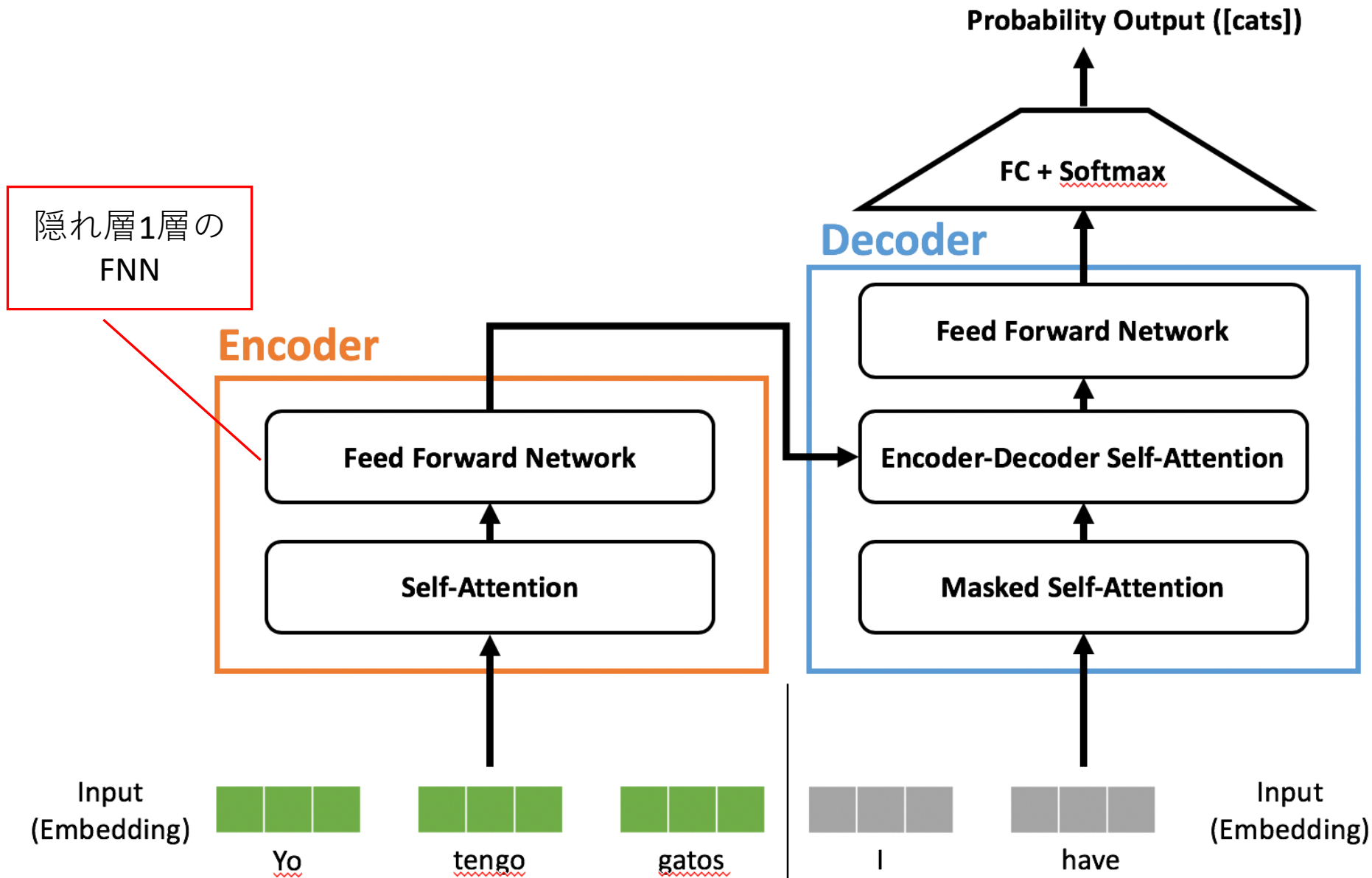
Attention



同一言語内

Self-Attention

Transformer



GPT

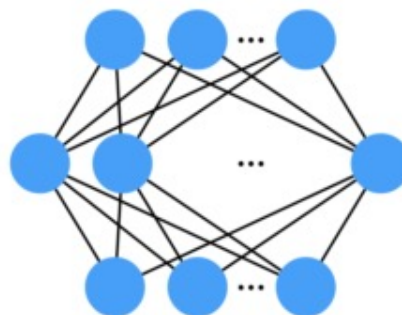
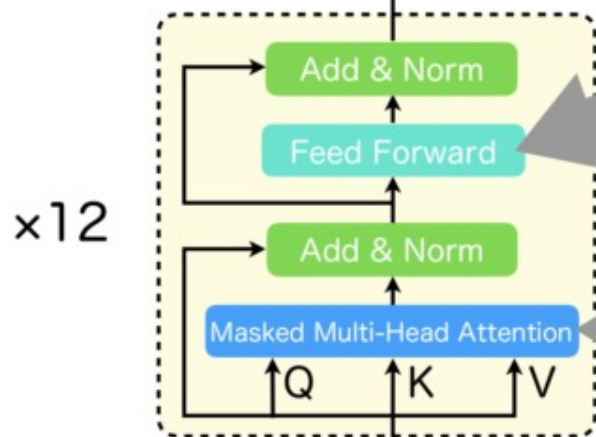
Generative Pre-trained Transformer

Output Block



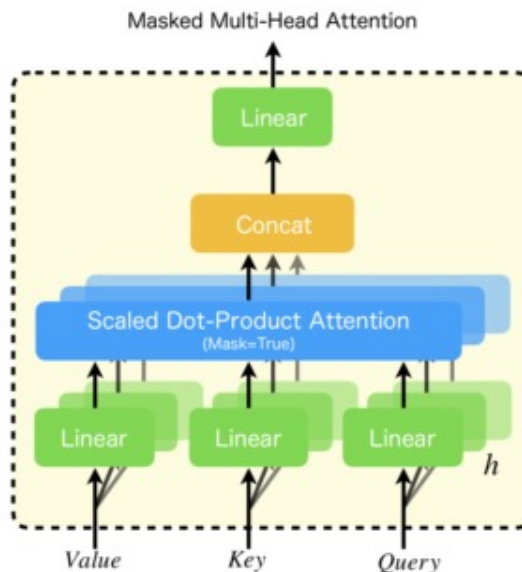
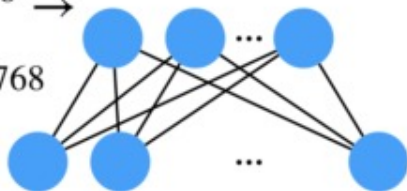
Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever,
Improving Language Understanding by Generative Pre-Training,
Preprint, 2018.
OpenAI

Transformer Block



Embedding Block

$$W_p \in \mathbb{R}^{512 \times 768} \rightarrow$$
$$W_e \in \mathbb{R}^{40478 \times 768}$$



$$W^o \in \mathbb{R}^{768 \times 768}$$

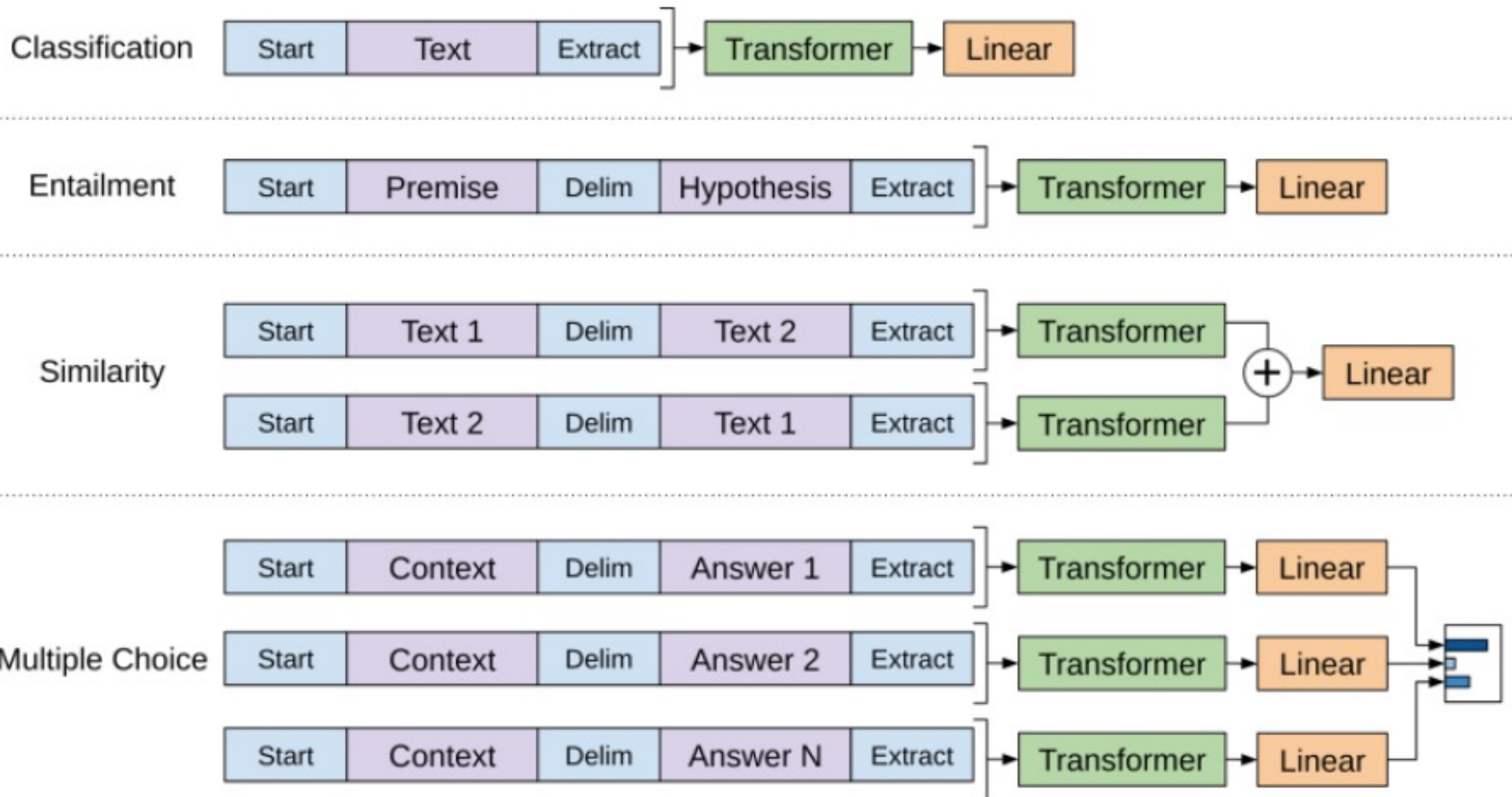
$$W_i^Q, W_i^K, W_i^V \in \mathbb{R}^{768 \times 64}$$
$$(i \in \{1, 2, \dots, 12\})$$

Book Corpus 語彙数 : 40478

GPT

Generative Pre-trained Transformer

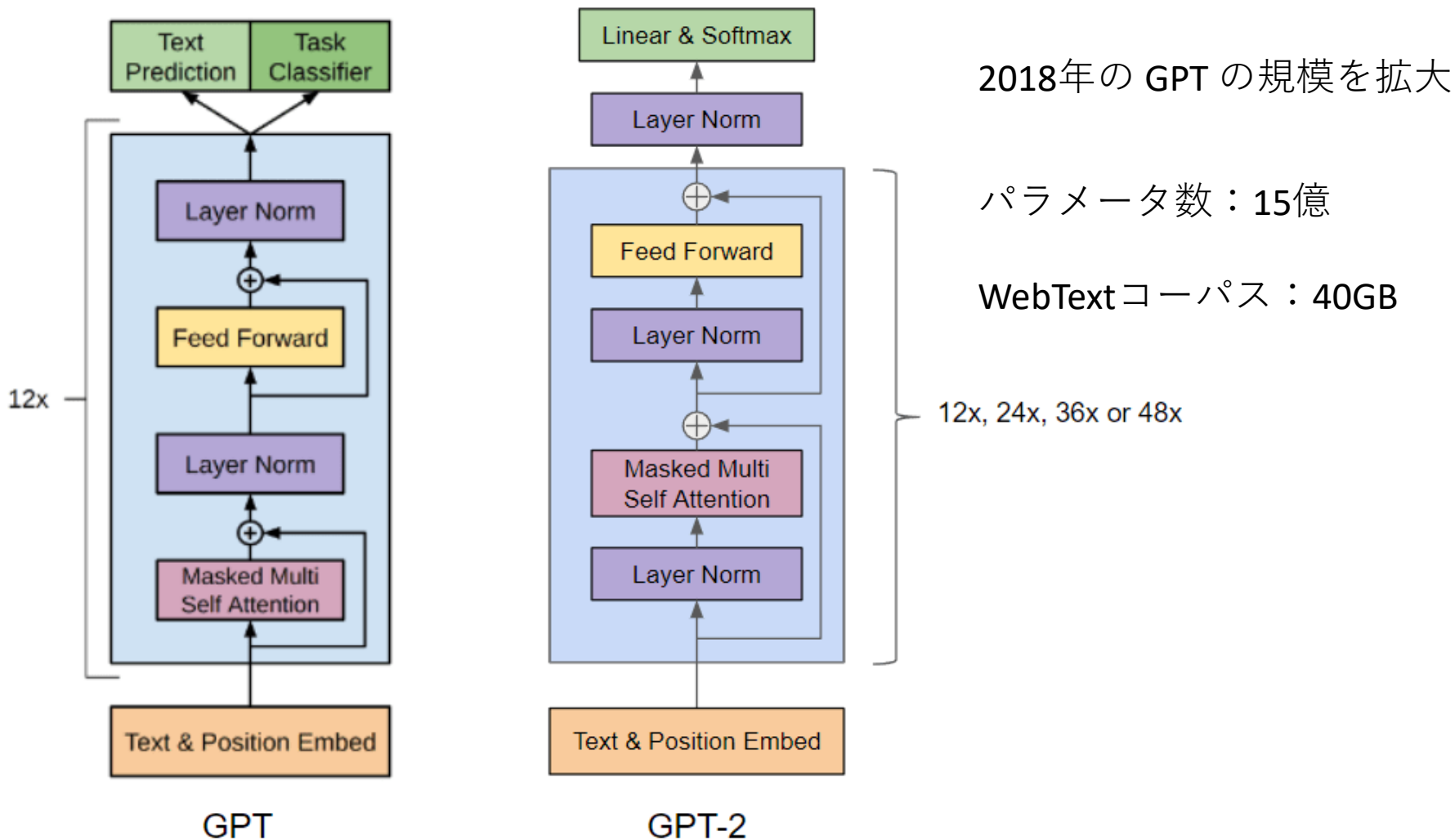
Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever,
Improving Language Understanding by Generative Pre-Training,
Preprint, 2018.
OpenAI



GPT-2

Generative Pre-trained Transformer 2

Alec Radford , Jeffrey Wu , Rewon Child, David Luan, Dario Amodei, Ilya Sutskever
Language Models are Unsupervised Multitask Learners
Technical Report OpenAI, 2019.

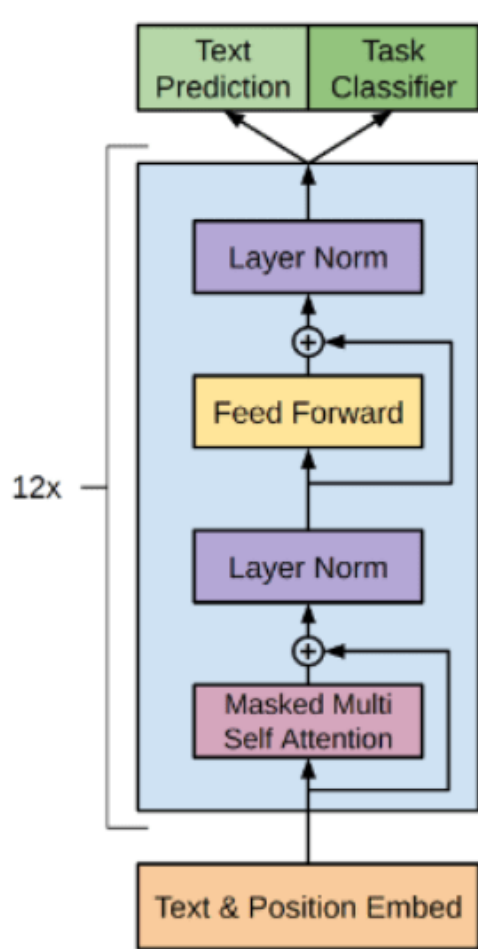


GPT-3

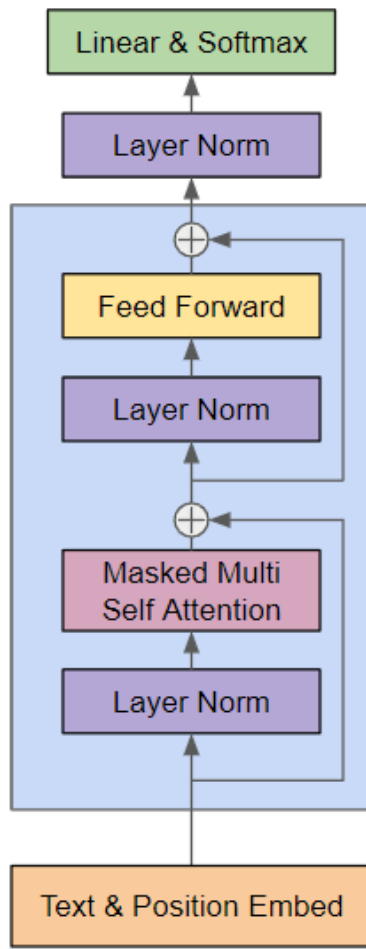
Generative Pre-trained Transformer 3

OpenAIによって開発された入力されたテキストから次の単語の確率を求めるモデル

Tom B. Brown et. al. (31 persons!)
Language Models are Few-Shot Learners
<https://arxiv.org/abs/2005.14165>, 2020



GPT



GPT-3

GPT-2 の規模を拡大

パラメータ数：1750億

570GBの平文（4000億トークン）

96x

GPT-3.5

Generative Pre-trained Transformer 3.5

Step 1

Collect demonstration data and train a supervised policy.

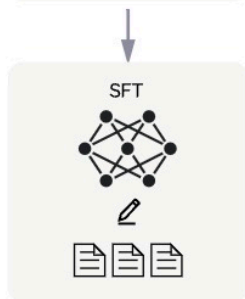
A prompt is sampled from our prompt dataset.



A labeler demonstrates the desired output behavior.



This data is used to fine-tune GPT-3.5 with supervised learning.



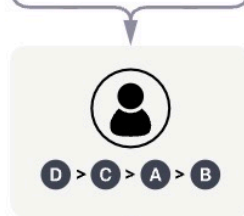
Step 2

Collect comparison data and train a reward model.

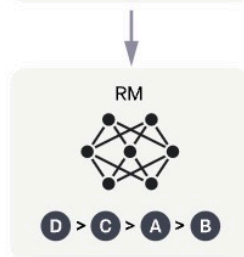
A prompt and several model outputs are sampled.



A labeler ranks the outputs from best to worst.



This data is used to train our reward model.



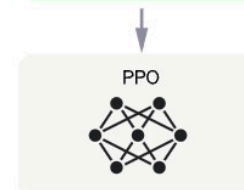
Step 3

Optimize a policy against the reward model using the PPO reinforcement learning algorithm.

A new prompt is sampled from the dataset.



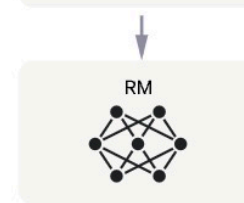
The PPO model is initialized from the supervised policy.



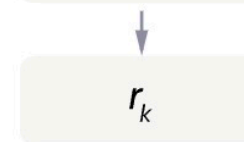
The policy generates an output.



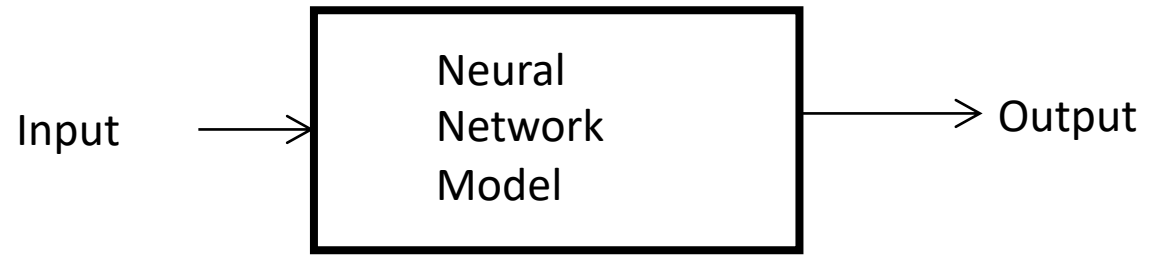
The reward model calculates a reward for the output.



The reward is used to update the policy using PPO.



2010-

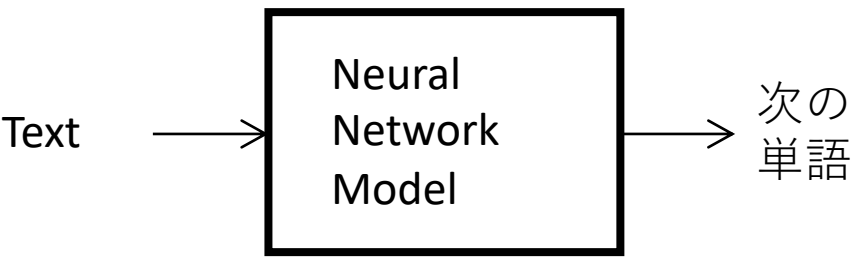


モデルが自動的に特徴量を抽出： 必要データ数：数百万

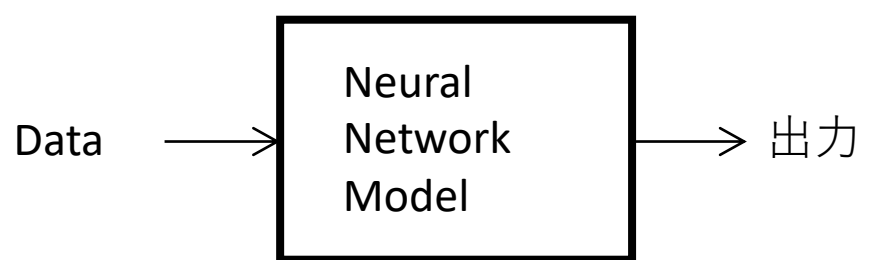
2017-

Transformer, BERT, GPT

特定のタスクのために微調整



事前学習 (pre-train)

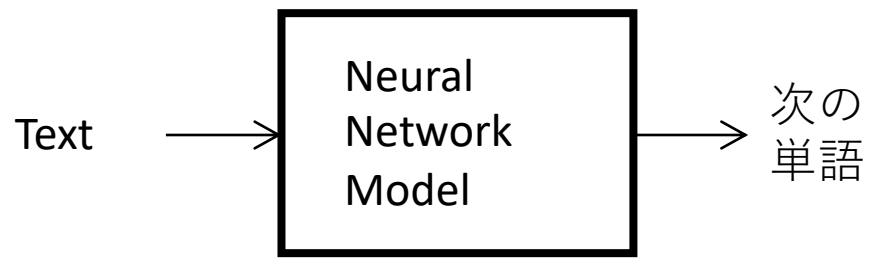


ファインチューニング
必要データ数：数百～数千

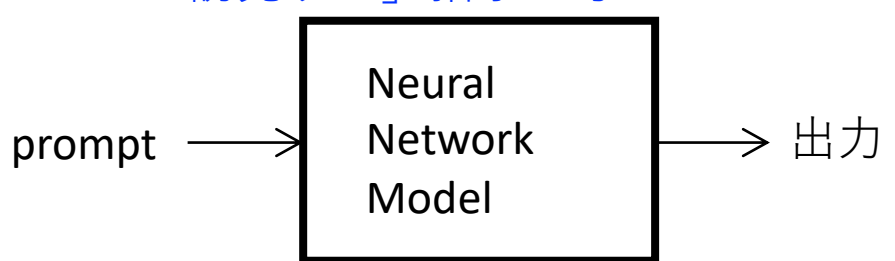
2022-

GPT, PaLM, Gemini

特定のタスクの答を「誘発する」指示を与える



事前学習 (pre-train)



プロンプティング
必要データ数：0～数個



なぜLLMは滑らかな文章を生成することが可能であるのか、分かりやすく説明してください

LLM（Large Language Model、大規模言語モデル）が滑らかな文章を生成することが可能なのは、以下の3つの主要な理由によります：

1.大量のテキストデータによる学習：

LLMは、インターネット上の記事、書籍、ウェブサイト、会話など、膨大な量のテキストデータから学習します。この学習過程で、様々な文脈、言い回し、文法規則、話題に関する知識を吸収します。その結果、言語の自然な使用法を模倣する能力を獲得します。

2.トランスフォーマーアーキテクチャの利用：

多くのLLM、特にGPTシリーズなどは、トランスフォーマーアーキテクチャに基づいています。このアーキテクチャは、文の中の各単語がその前後の単語とどのように関連しているかを理解するのに非常に効率的です。これにより、文脈に基づいて次に来るべき最も自然な単語やフレーズを予測することができます。

3.自己注意メカニズム：

トランスフォーマーアーキテクチャの核心部分である自己注意メカニズムにより、モデルは生成される文章内での各単語間の関係をより深く理解します。これにより、より関連性が高く、文脈に沿った滑らかな文章を生成することが可能になります。

これらの要素が組み合わさることで、LLMは自然で流暢な文章を生成することができ、人間が書いたかのようなテキストを作り出すことが可能になります。

なぜLLMは滑らかな文章を生成することが可能であるのか、分かりやすく説明してください

LLMが滑らかな文章を生成できる理由は、主に以下の3つです。

1. 大量のデータで学習している

LLMは、インターネット上の文章や書籍など、膨大な量のデータで学習しています。そのため、人間の言葉のパターンや規則を理解することができます。

2. さまざまな表現を学習している

LLMは、さまざまな表現を学習しています。そのため、状況に合わせて適切な表現を使うことができます。

3. 文脈を理解している

LLMは、文脈を理解することができます。そのため、前後の文章と矛盾のない文章を生成することができます。

****分かりやすく例えると、LLMは膨大な量の文章を記憶した人間のようなものです。**

****人間は、過去の経験から、状況に合わせて適切な言葉を使い分けることができます。**

LLMも同じように、学習したデータから、滑らかな文章を生成することができます。

**Talk to Claude**

claude.ai

なぜLLMは滑らかな文章を生成することが可能であるのか、分かりやすく説明してください

LLMが滑らかな文章を生成できる主な理由は以下の通りです。

- 膨大な量のテキストデータで事前訓練されている
LLMは数百億~千億規模のパラメータを持ち、Wikipediaなどの大量のテキストデータで事前訓練されています。人間の言語をある程度学習しているため、自然な文章を生成しやすい。
- Transformer構造を利用している
注意機構を持つTransformerアーキテクチャを採用しているため、文脈に応じて適切に単語を予測できる。文章の流れを理解しながら生成できる。
- ニューラルネットワークベース
ニューラルネットによる統計的な単語予測が可能。人間の言語パターンを学習することで、文法的誤りをある程度回避できる。
- 大量のパラメータ
百億規模のパラメータで言語を細かく学習できるため、人間の言語感覚に近い出力を生成しやすい。

といった理由により、自然な文章生成が実現しています。

今後必要となるスキル

- ・ 情報を批判的に評価し、複雑な問題を解決する能力
- ・ 新規性や革新的なアイデアを生み出す能力
- ・ コミュニケーション、共感、チームワークなどの人間関係スキル
- ・ 新しいツールやシステムを迅速に学び、適応する能力
- ・ 大量のデータを理解し、分析する能力
- ・ AI技術を効果的に利用、調整、カスタマイズするためのプログラミングスキル
- ・ 技術的進歩を適切に管理するための倫理的な判断力
- ・ 異なる分野の知識を組み合わせ、複雑な問題を解決する能力