[無料講座]

LLMの中身を体系的に学ぶ講座

イントロダクション

石津 俊輔

2025年 11月 5日(水)

[本日の目標]

AI の仕組み・裏側について, 今よりも一歩深く理解できるようになること

- ChatGPT が生まれるまでの流れをざっくりつかむ
- ChatGPT の “中身” で使われているモデルの正体を知る
- ChatGPT がどのように学習し, 答えを作っているのかを理解する

今日の講座では, AI の全体像を理解することを目指します。
詳細な数式やコードではなく, 「こういう仕組みなんだ」と感覚でつかめるようになり, 今よりも一歩深く理解できる状態になるのがゴールです.

各項目の詳細は本講座で解説したいと考えています.

[目次]

01

ChatGPT ができるまでの歴史

[ChatGPTができるまでの歴史]

Neural Network を使った自然言語処理が始まったのは2003年. 2014年頃から一気に加速.

モデルの変遷

Bag-of-Words
単語を数えて予測

共起行列
単語の出現回数で意味を獲得

Word2Vec (2013)
ニューラルネットワークで単語の意味を獲得

Neural Network (2003)
ニューラルネットワークを初めて言語モデルに導入し, 分散表現の概念を確立

RNN, LSTM (2010, 2014)
言葉の流れを学習

Attention (2016)
長文を捕捉

Transformer (2017)
並列計算で学習を効率化

事前学習 + ファインチューニング (2015)
前もって言語を学習

GPT, BERT (2018)
モデルと学習手法の融合
初めての大規模言語モデル

ChatGPT (2023)
人に寄り添った対話モデルとして学習

02

単語から予測をする Bag-of-Wordsモデル

[ChatGPTができるまでの歴史]

Neural Network を使った自然言語処理が始まったのは2003年. 2014年頃から一気に加速.

モデルの変遷

Bag-of-Words
単語を数えて予測

共起行列
単語の出現回数で意味を獲得

Word2Vec (2013)
ニューラルネットワークで単語の意味を獲得

Neural Network (2003)
ニューラルネットワークを初めて言語モデルに導入し, 分散表現の概念を確立

RNN, LSTM (2010, 2014)
言葉の流れを学習

Attention (2016)
長文を捕捉

Transformer (2017)
並列計算で学習を効率化

事前学習 + ファインチューニング (2015)
前もって言語を学習

GPT, BERT (2018)
モデルと学習手法の融合
初めての大規模言語モデル

ChatGPT (2023)
人に寄り添った対話モデルとして学習

[Bag-of-Words モデルとは]

Bag-of-Words モデル

- 単語を袋の中にぐちゃっと入れる, という意味.
- 袋 (文章) の中に入っている単語の数で文章を分類したり, センチメントを判定.

センチメントは？

➡ ポジティブ

ネガティブ

中立

どんな話題？

➡ 政治

スポーツ

芸能

スポーツの種類は？

➡ 野球

サッカー

バスケット

自動で振り分けられるので便利

順番の情報はない

[実際のイメージ]

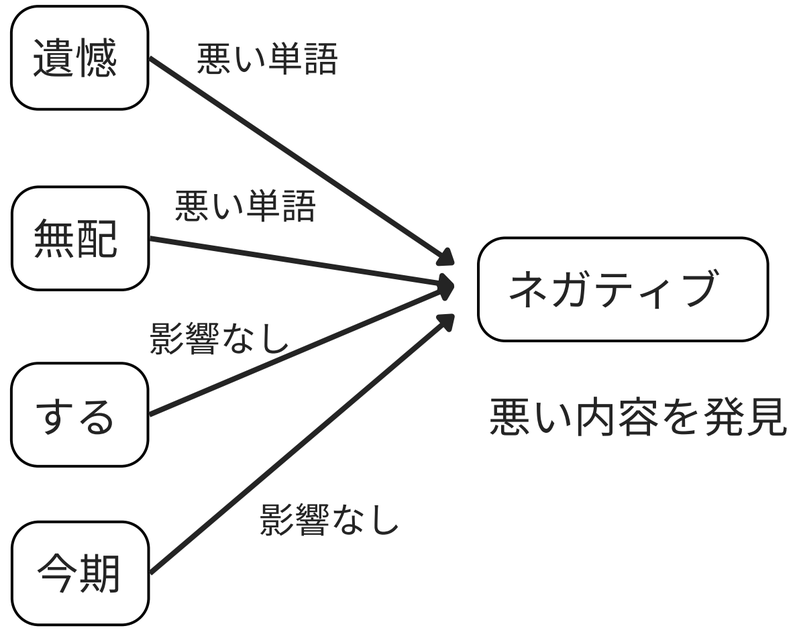
「誠に遺憾ですが, 今期は無配とさせていただきます.」

単語にばらして
(分かち書き; Mecab)

- 誠に
- 遺憾
- です
- ...
- 無配
- と
- させて
- いただ
- ます

袋に入れて個数を数える

学習・予測



復配しましたが無配にします = 無配でしたが復配します

複数の単語をセットにする n-gram や文書中の頻出単語の重みを軽くする TF-IDF といった工夫

03

単語の意味を獲得する Word2Vec

[ChatGPTができるまでの歴史]

Neural Network を使った自然言語処理が始まったのは2003年. 2014年頃から一気に加速.

モデルの変遷

Bag-of-Words
単語を数えて予測

共起行列
単語の出現回数で意味を獲得

Word2Vec (2013)
ニューラルネットワークで単語の意味を獲得

Neural Network (2003)
ニューラルネットワークを初めて言語モデルに導入し, 分散表現の概念を確立

RNN, LSTM (2010, 2014)
言葉の流れを学習

Attention (2016)
長文を捕捉

Transformer (2017)
並列計算で学習を効率化

事前学習 + ファインチューニング (2015)
前もって言語を学習

GPT, BERT (2018)
モデルと学習手法の融合
初めての大規模言語モデル

ChatGPT (2023)
人に寄り添った対話モデルとして学習

[単語を表す方法 ①]

one-hot 表現

辞書の何番目にあるか？

one-hot 表現

k 番目

映画

$$\begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

k 番目だけが1,
そのほかは0

共起行列

単語の意味は, その単語の周りにどんな単語があるかで説明できる.

昨日　観た　**映画**　は　面白い

昨日　× 1
観た　× 1
は　× 1
面白い × 1

$$\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

昨日
観た
面白い
は

語彙数

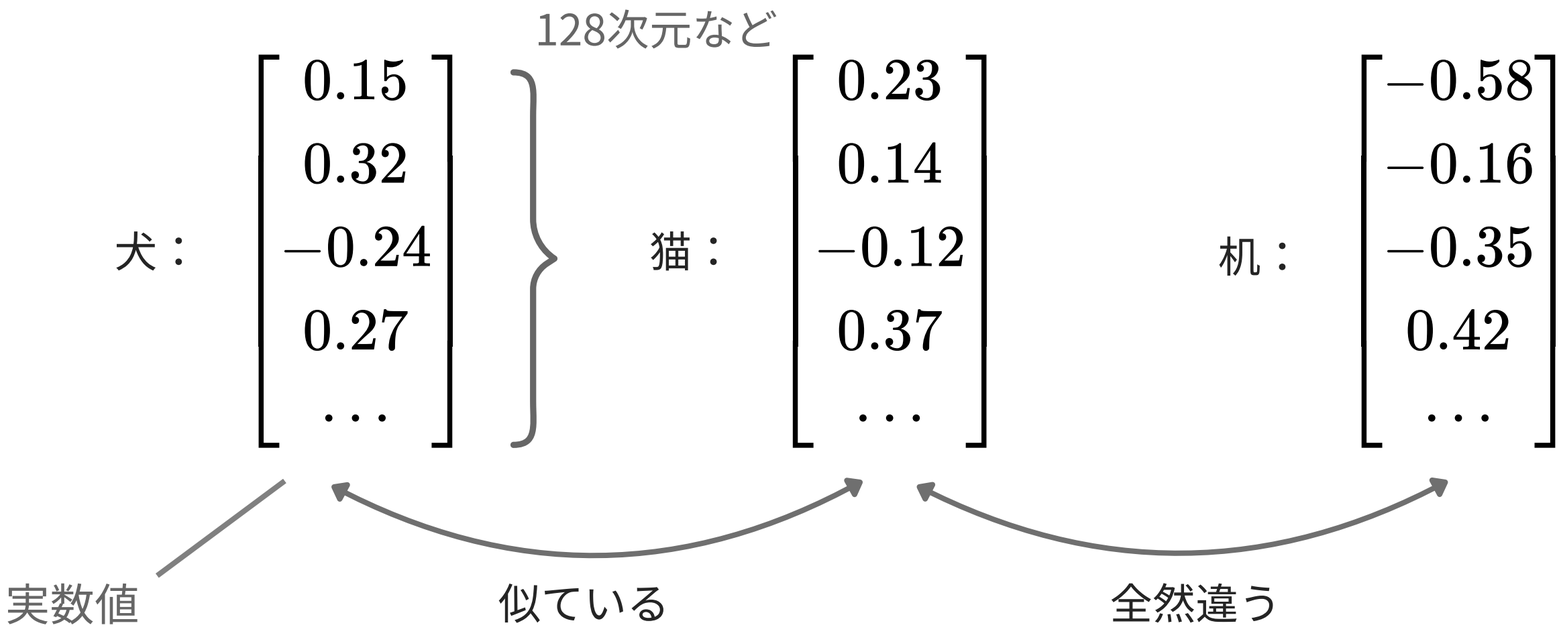
これが映画を表す表現

- 単語数が増えると巨大な行列になる

[単語を表す方法 ②]

埋め込み表現 (分散表現)

単語をその単語の**意味を表現**する**低次元**の**実数値**ベクトル (**埋め込み表現**) に変換する, ということ.



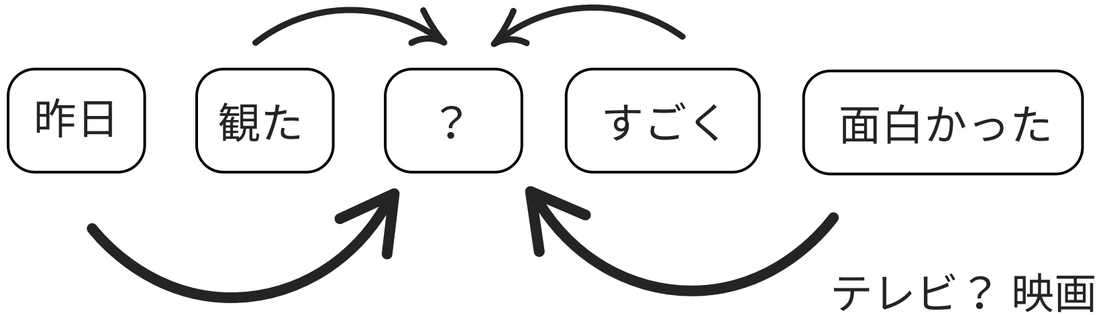
💡 共起行列は自然数のベクトル, 埋め込み表現による単語表現は実数値ベクトル

[Word2Vec]

Word2Vec は埋め込み表現を計算するモデル

CBOW (Continuous Bag of Words) モデル … 連続的な Bag-of-Words

- 単語の意味は **周りの単語** から形成されるという仮説 (分布仮説)
- 周辺の単語から真ん中の単語を予測するタスクを解く



テレビ？ 映画？ 試合？

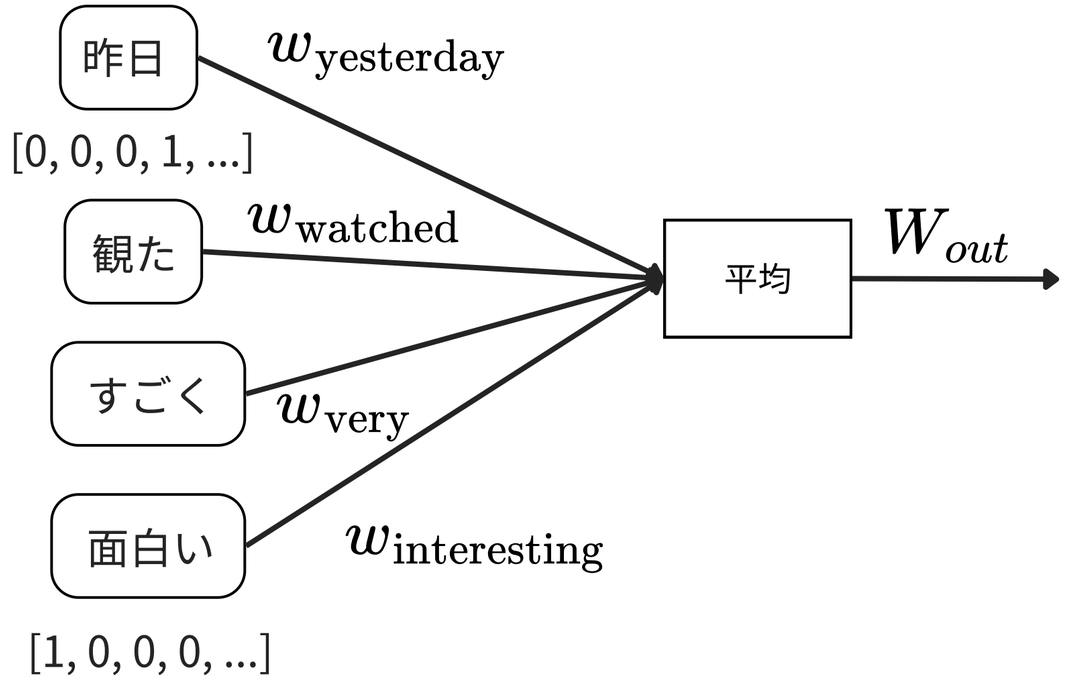
観るもの, 面白いもの (つまらないもの)

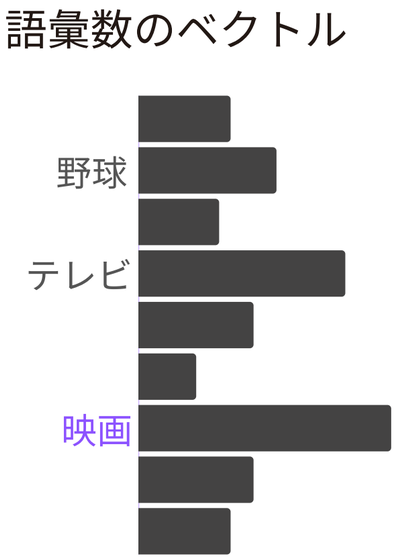
他にも, 真ん中の単語から周りの単語を予測する **Skipgram** モデルも提案

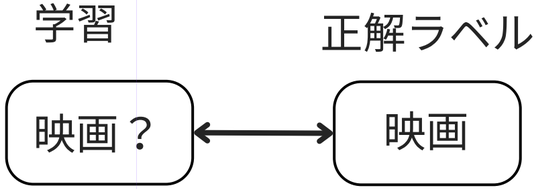
Efficient Estimation of Word Representations in Vector Space (2013)

[CBOW モデル詳細]

CBOW (Continuous Bag of Words) モデル







- このようにして, w_{watched} , w_{movie} などを学習する.
- この wを埋め込み表現と呼ぶ.

[Word2Vec による埋め込み表現の特徴]

- 似たような単語は同じような埋め込み表現となり, 似ている単語を検索することができる.
  (詳細は本講座で)
  - ➡ 文章を埋め込むことで RAG (**R**etrieval-**A**ugmented **G**eneration) に応用
  - 💡 King - Man + Woman = Queen というような計算もできる

- **Word2Vecの欠点**

  1単語に対して1つの埋め込み表現なので, 同じ名前の単語は同じ埋め込みになる

  例えば, 現金の意味の “キャッシュ” と一時記憶の “キャッシュ” は同じ埋め込み表現になる

  - ➡ 文脈を考慮した埋め込み表現のモデルもあり (ELMo)

04

ニューラル・ネットワークを言語モデルに応用

[ChatGPTができるまでの歴史]

Neural Network を使った自然言語処理が始まったのは2003年. 2014年頃から一気に加速.

モデルの変遷

Bag-of-Words
単語を数えて予測

共起行列
単語の出現回数で意味を獲得

→ **Word2Vec (2013)**
ニューラルネットワークで単語の意味を獲得

Neural Network (2003)
ニューラルネットワークを初めて言語モデルに導入し, 分散表現の概念を確立

→ **RNN, LSTM (2010, 2014)**
言葉の流れを学習

→ **Attention (2016)**
長文を捕捉

→ **Transformer (2017)**
並列計算で学習を効率化

事前学習 + ファインチューニング (2015)
前もって言語を学習

→ **GPT, BERT (2018)**
モデルと学習手法の融合
初めての大規模言語モデル

>>> **ChatGPT (2023)**
人に寄り添った対話モデルとして学習

[言語モデルとは]

言語モデルとは

✔ 文章 (単語の並び) がどれくらい**自然か**を判断する仕組み

少し難しく言うと, **単語の並びに確率を与える**モデル → 正しい言語モデルは正しく確率を与えることができる

○ 昨日観た映画は面白かった　　確率99.5%

× 昨日観た映画は丸い　　確率3.2%

💡 次の単語を正しく予測することが重要！

昨日観た映画は ＿＿＿

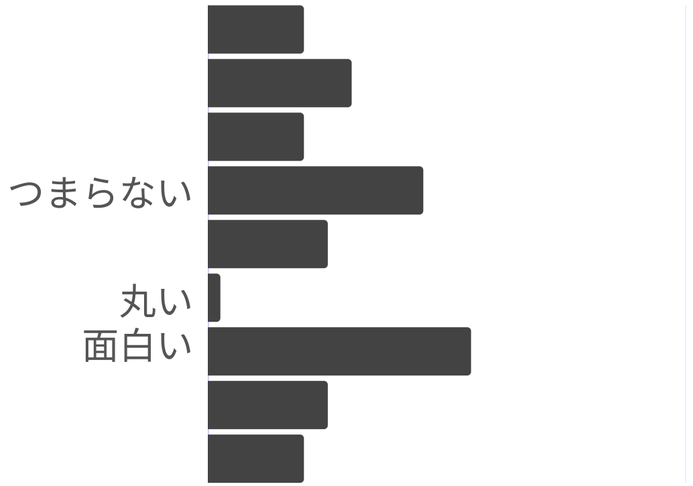
この確率は, 条件付き期待値で書くことができる.

$$P(w_1, w_2, \ldots, w_T) = \prod_{t=1}^{T} P(w_t | w_1, \ldots, w_{t-1})$$

次の単語は何か？

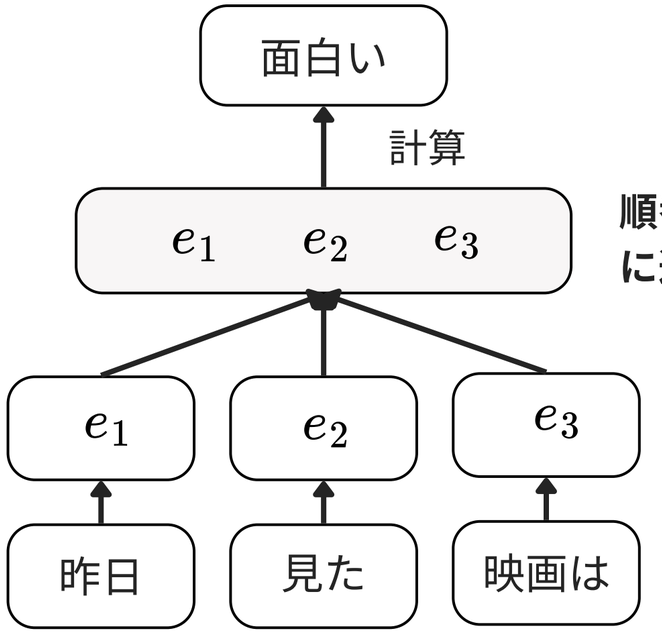
w_t：単語「面白かった」

$w_1, \ldots, w_{t-1}$：前の文章「昨日観た映画は」



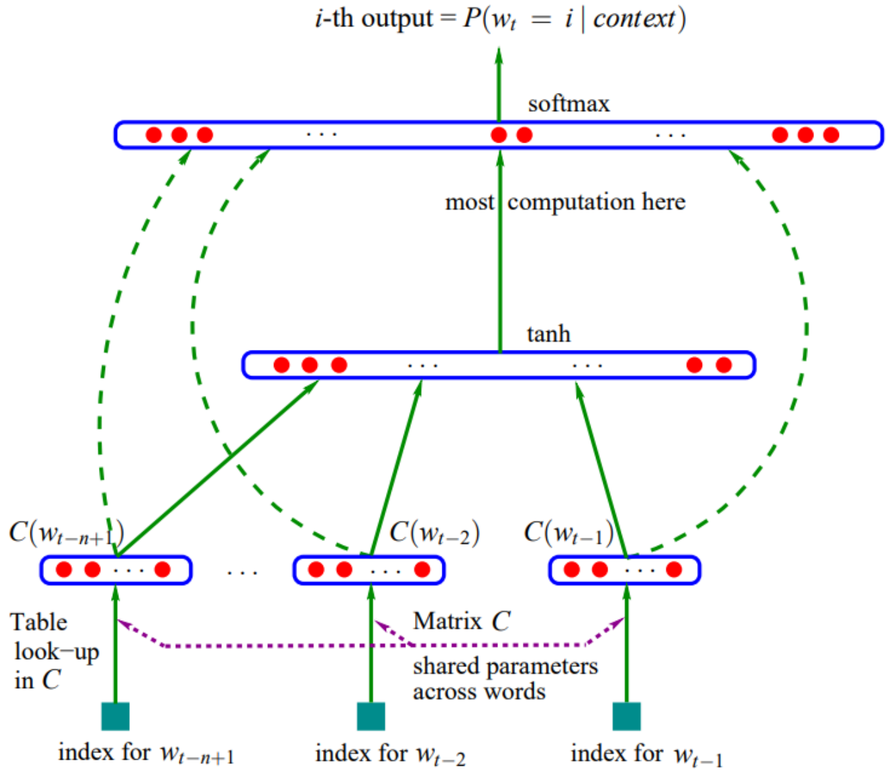
[ニューラル言語モデル]

- 単語の埋め込み表現 (分散表現) と文章の確率を計算



順番の情報を保持するために連結

埋め込み表現



A Neural Probabilistic Language Model (2003)

- 文章が固定長となる (この場合, インプットが3つ)
  長い文章にすると, 重みが増えすぎてしまう.

05

時系列モデルの応用とその限界

[ChatGPTができるまでの歴史]

Neural Network を使った自然言語処理が始まったのは2003年. 2014年頃から一気に加速.

モデルの変遷

Bag-of-Words
単語を数えて予測

共起行列
単語の出現回数で意味を獲得

Word2Vec (2013)
ニューラルネットワークで単語の意味を獲得

Neural Network (2003)
ニューラルネットワークを初めて言語モデルに導入し, 分散表現の概念を確立

RNN, LSTM (2010, 2014)
言葉の流れを学習

Attention (2016)
長文を捕捉

Transformer (2017)
並列計算で学習を効率化

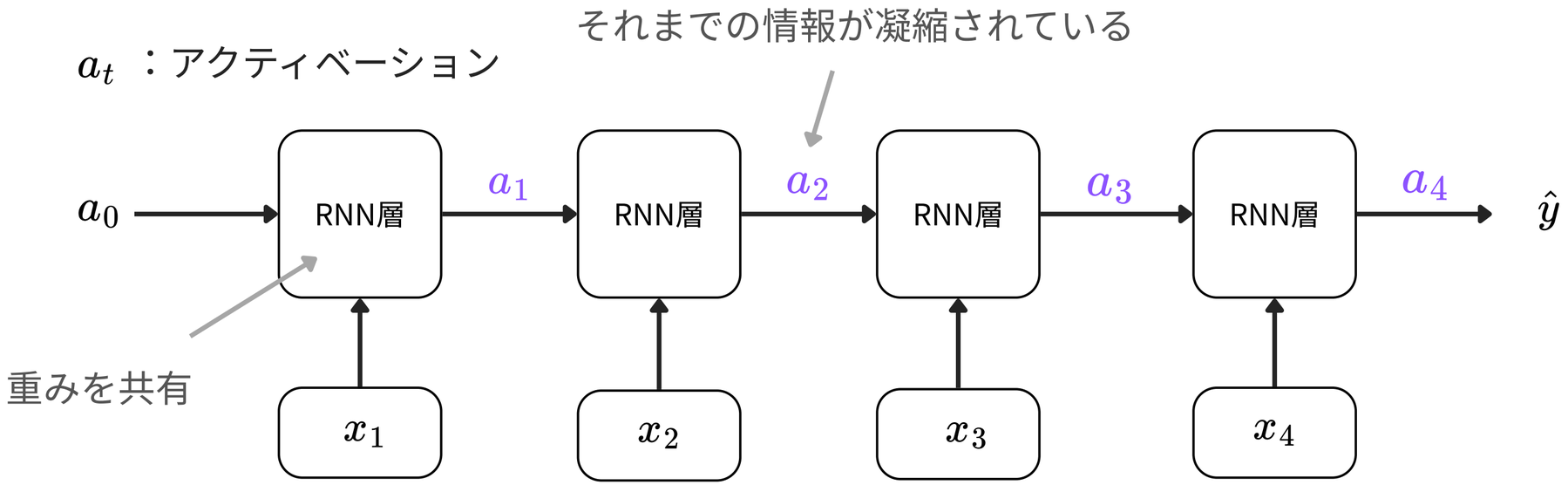
事前学習 + ファインチューニング (2015)
前もって言語を学習

GPT, BERT (2018)
モデルと学習手法の融合
初めての大規模言語モデル

ChatGPT (2023)
人に寄り添った対話モデルとして学習

[再帰的ニューラル・ネットワークとは]

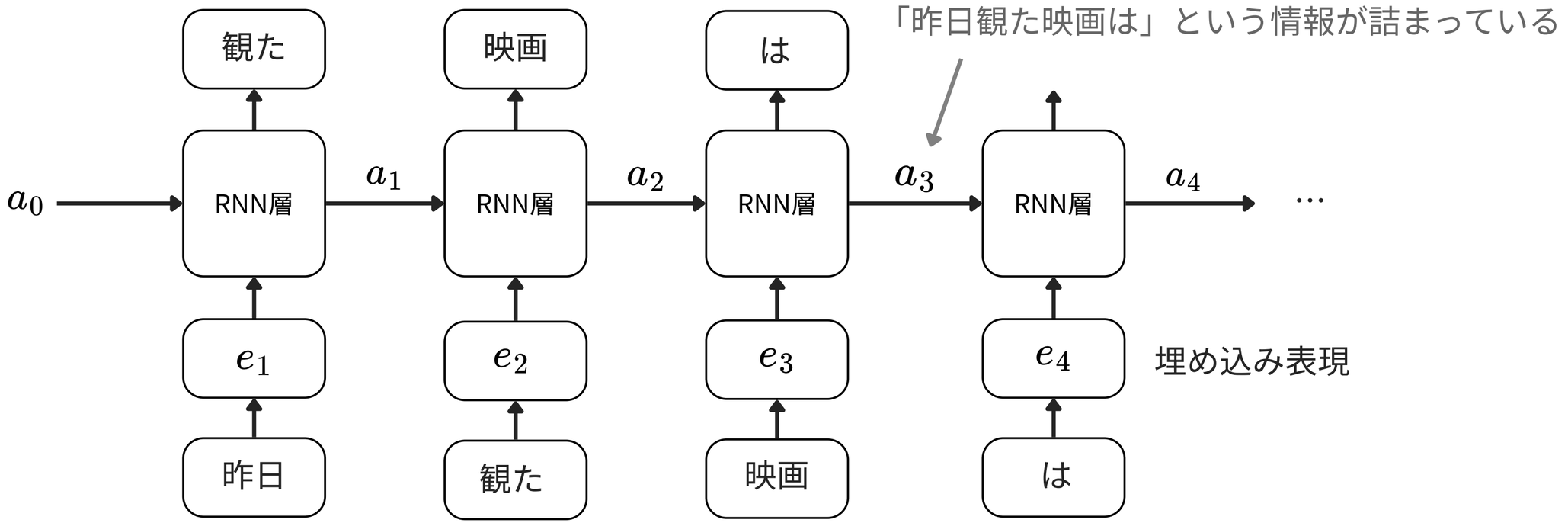
- 時系列データに対するニューラル・ネットワーク
- 前から処理を行い, 順番に次の層にアクティベーション a_t を送っていく
- そのアクティベーションと単語のインプットを使って次の層のアクティベーションを計算する



任意の長さの文章に対応できる

[RNN を言語モデル]

時系列データに対するニューラル・ネットワークである再帰的ニューラル・ネットワーク
(Recurrent Neural Network; RNN) で, **文章を単語の時系列データと見て処理**を行う.



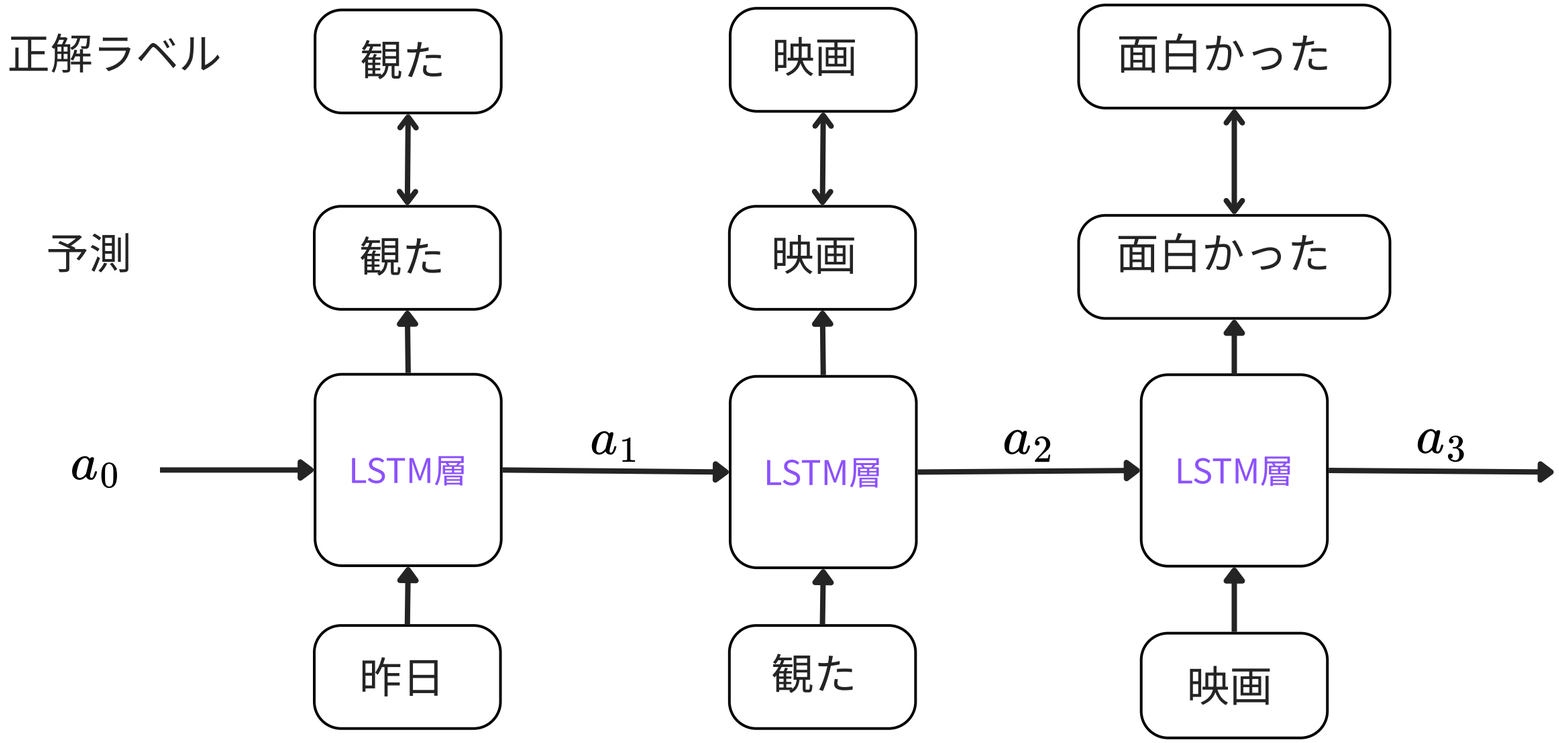
- 文章が長くなると勾配が消失し, 学習できないという問題がある.

Recurrent neural network based language model (2010)

[LSTM (Long Short-Term Memory)とは]

LSTM は RNN の欠点を補うため, 更新ゲート(Update gate), 忘却ゲート(Forget gate), 出力ゲート(Output gate) などを備え, 長い文章になっても学習ができるようにするモデル.

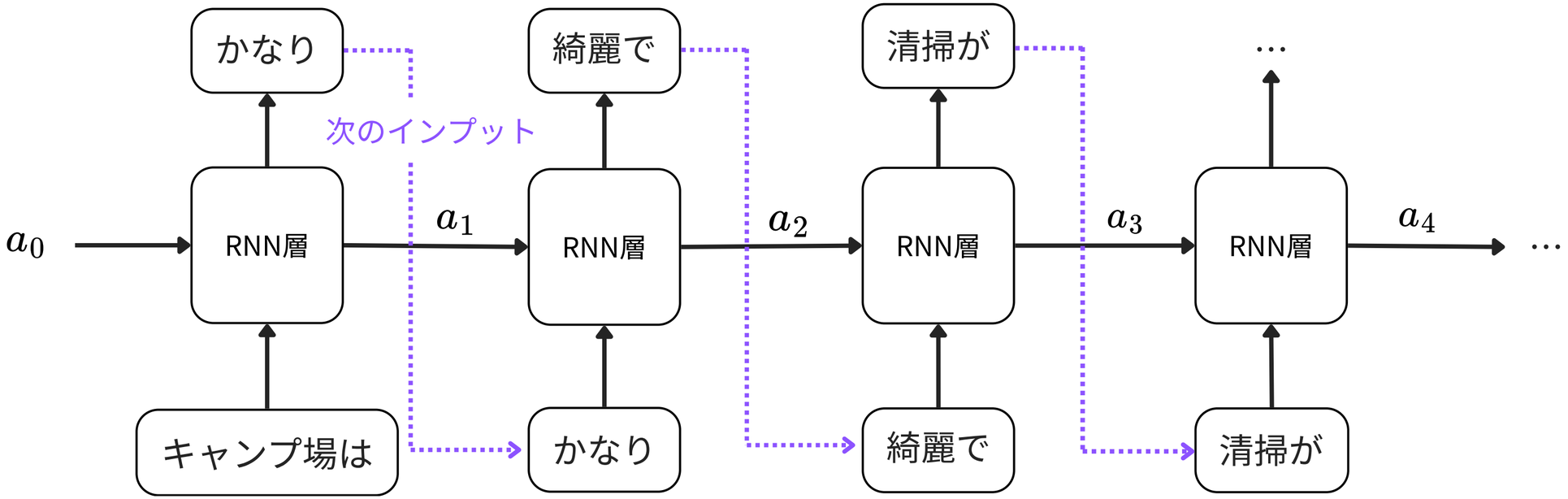
LSTM による言語モデルの学習のイメージ



Long Short-Term Memory (1997)

[（ご参考）RNN による文章生成]

RNN による文章生成例



キャンプ場の口コミで学習した RNN モデルで文章生成.「キャンプ場は」で始めると,

トイレ が かなり 綺麗 で 清掃 が 行き届い て い て 快適 に 過ごせ ま し た ！ 流し台 (ブースオージ で の 冷蔵庫 も あり ま し た 。 …

『【データ分析実践】RNNを使って文章生成を実装してみる』より

[（ご参考）RNN による文章生成]

RNN による文章生成例

悪い口コミを作成してみようと,「スタッフの態度が非常に悪く 」で始めると,

スタッフの態度が非常に悪く 、 日 中 は 暑い です 。 夜 は 星 が 綺麗 でし た 。
芝生 の キャンプ 場 で 、 木々 に 囲ま れ て い て 、 とても 気持ち が 良い です 。
川 の 水 は 冷たく 、 足 を つけ て 遊ん で い ま し た 。 川 が 近く に あり 、
川遊び が 楽しめ ます 。 川 の せせらぎ が 心地よく 、 夜 は 静か で 過ごし やすか
っ た です 。

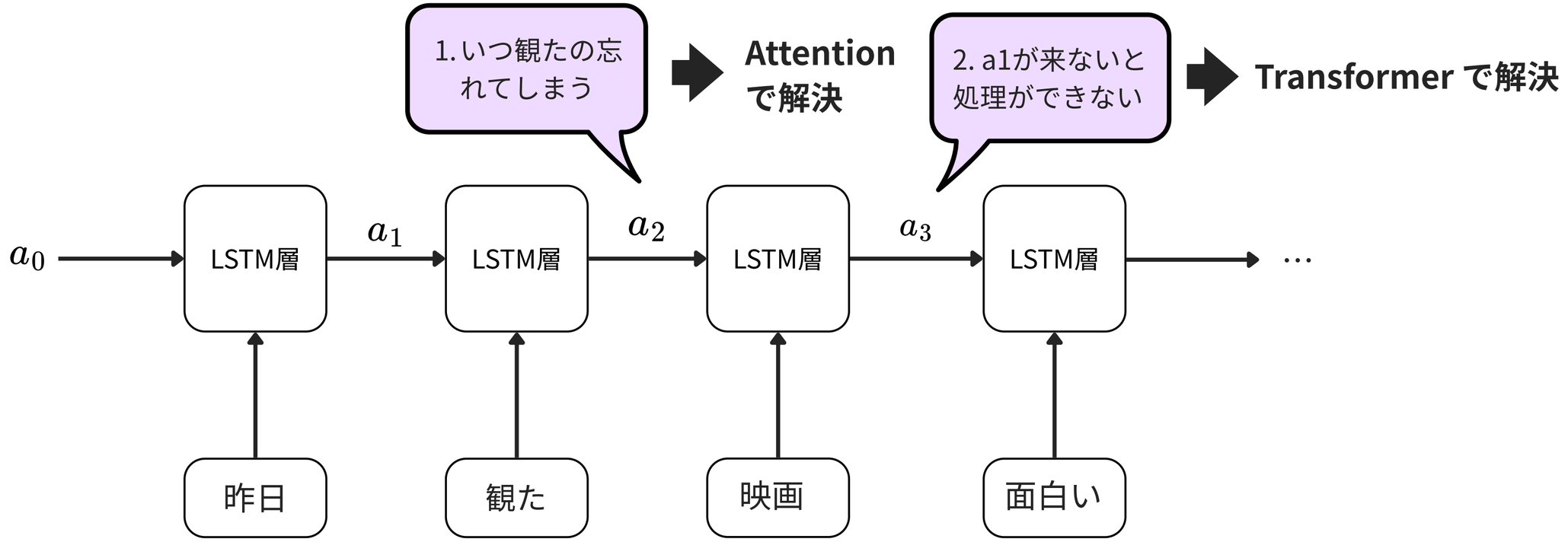
良い口コミがほとんどなので, 悪い文章が作成しにくい.

学習データに強く依存する

『【データ分析実践】RNNを使って文章生成を実装してみる』より

[LSTM の問題点]

1. それでも, 古い情報がどんどん薄まっていき, 長文になると過去の情報を忘れてしまう
2. 次の層に行くにはその前の層の処理を終えて, アクティベーションを次に渡す必要がある. 並列計算の得意な GPU の能力を十分に発揮できない



06

Attention メカニズムの出現

[ChatGPTができるまでの歴史]

Neural Network を使った自然言語処理が始まったのは2003年. 2014年頃から一気に加速.

モデルの変遷

Bag-of-Words
単語を数えて予測

共起行列
単語の出現回数で意味を獲得

→ **Word2Vec (2013)**
ニューラルネットワークで単語の意味を獲得

Neural Network (2003)
ニューラルネットワークを初めて言語モデルに導入し, 分散表現の概念を確立

→ **RNN, LSTM (2010, 2014)**
言葉の流れを学習

→ **Attention (2016)**
長文を捕捉

→ **Transformer (2017)**
並列計算で学習を効率化

事前学習 + ファインチューニング (2015)
前もって言語を学習

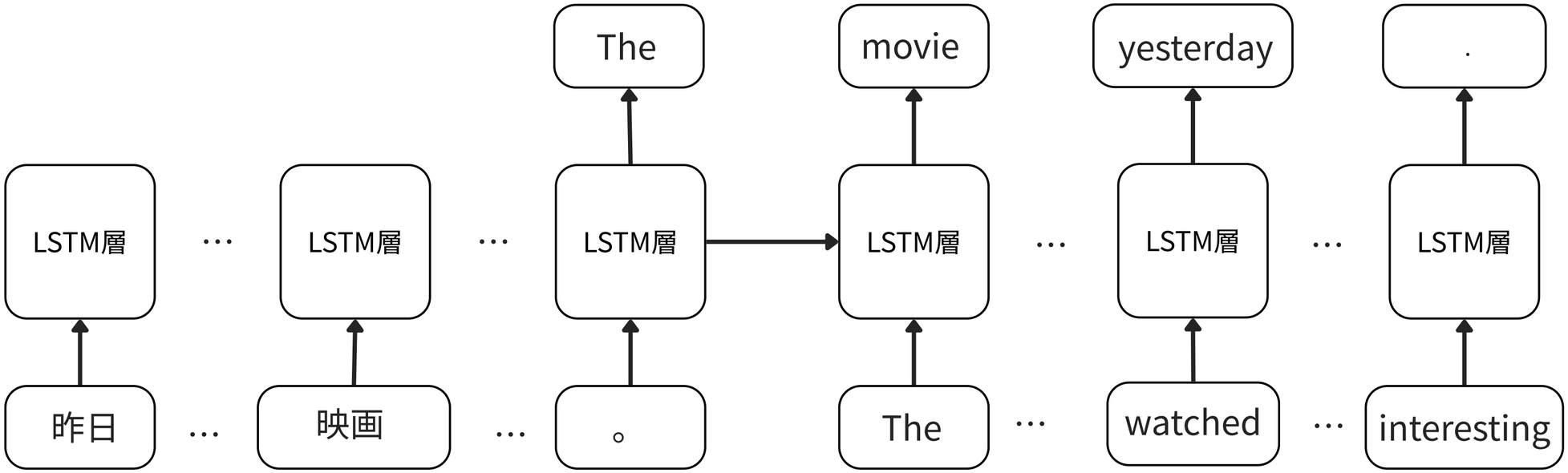
→ **GPT, BERT (2018)**
モデルと学習手法の融合
初めての大規模言語モデル

>>> **ChatGPT (2023)**
人に寄り添った対話モデルとして学習

[Attention メカニズムの仕組み]

LSTM には文章が長くなると過去の情報を忘れてしまうという欠点があり, それを解消することが目的

- **最後の50単語は順番も含めてきちんと記憶**しているが, それより前は**存在は記憶しているものの順番は記憶していない**という研究結果もあり.

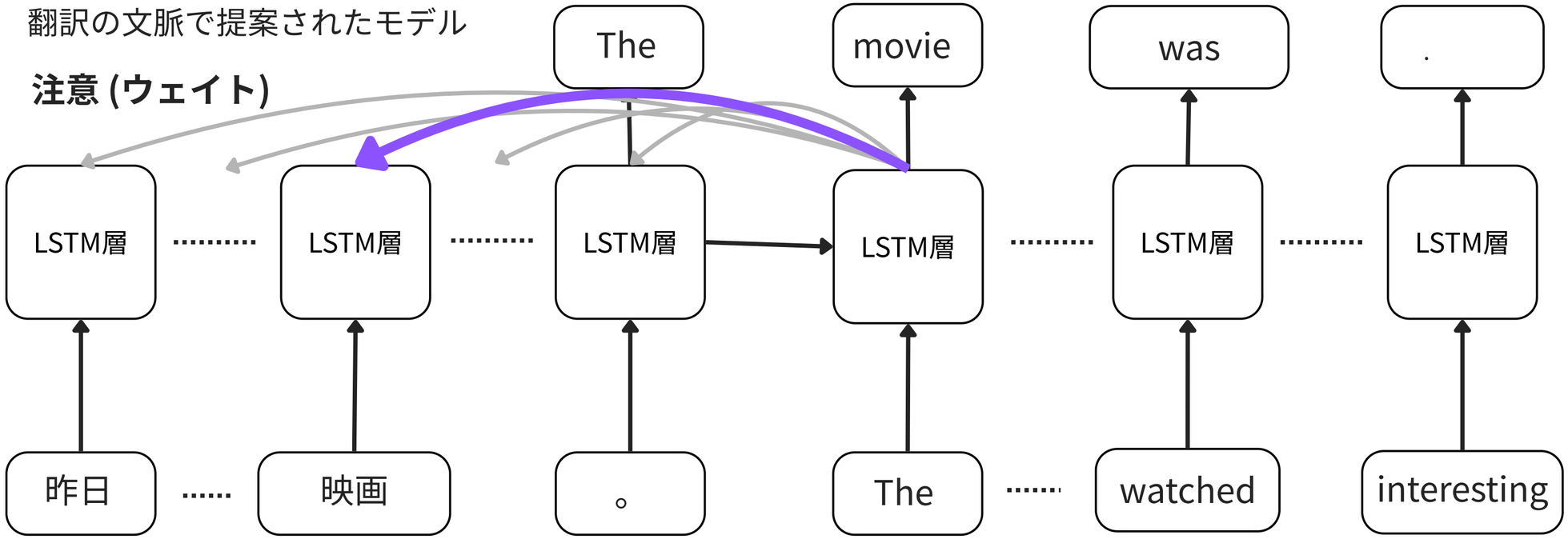


- LSTM でも最初の方の内容を忘れてしまう

[Attention メカニズムとは]

- 翻訳するに当たり, 元の文章のどこがに注意を向けると良いかを学習し, そこに大きなウェイトをつける

翻訳の文脈で提案されたモデル

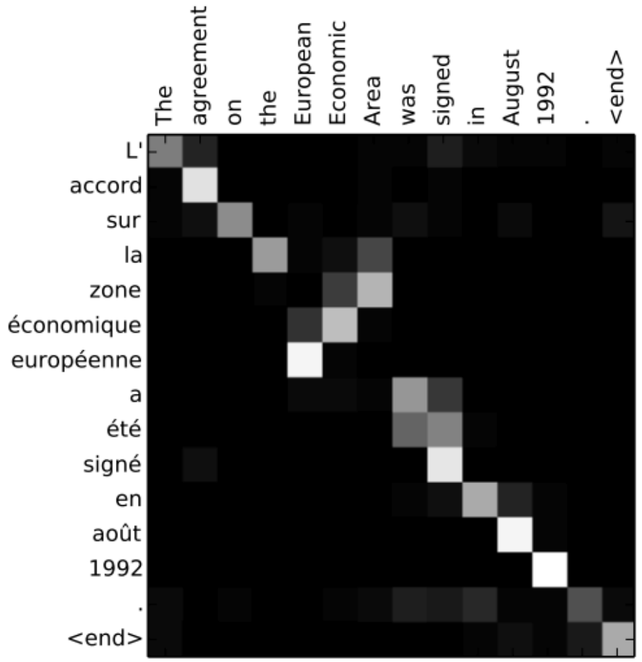


- 注意の向け方を学習することで, 文章の最初の方も忘れなくなる

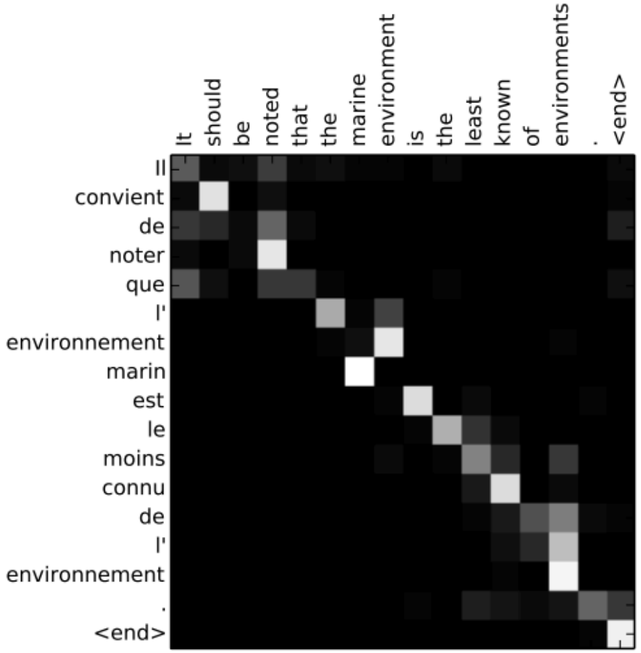
Neural Machine Translation by Jointly Learning to Align and Translate (2016)

[Attention の可視化]

- attention weight を可視化すると, 翻訳後の単語が適切に翻訳前の該当する単語のところに大きなウェイトが設定されている.



(a)



(b)

- どこを重要視しているかがわかる？ → BERT などで試すと, 全然関係のないところに注意が向いていることもあるので何とも言えない？

07

Transformer とその目的

[ChatGPTができるまでの歴史]

Neural Network を使った自然言語処理が始まったのは2003年. 2014年頃から一気に加速.

モデルの変遷

Bag-of-Words
単語を数えて予測

共起行列
単語の出現回数で意味を獲得

Word2Vec (2013)
ニューラルネットワークで単語の意味を獲得

Neural Network (2003)
ニューラルネットワークを初めて言語モデルに導入し, 分散表現の概念を確立

RNN, LSTM (2010, 2014)
言葉の流れを学習

Attention (2016)
長文を捕捉

Transformer (2017)
並列計算で学習を効率化

事前学習 + ファインチューニング (2015)
前もって言語を学習

GPT, BERT (2018)
モデルと学習手法の融合
初めての大規模言語モデル

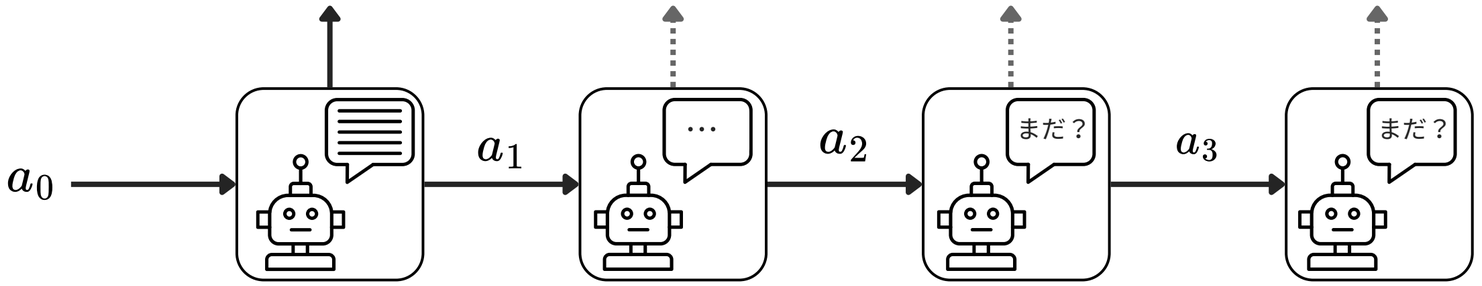
ChatGPT (2023)
人に寄り添った対話モデルとして学習

[Transformer とは]

LSTM は並列処理ができず, **GPU の性能をフル活用できていない**という問題があった.

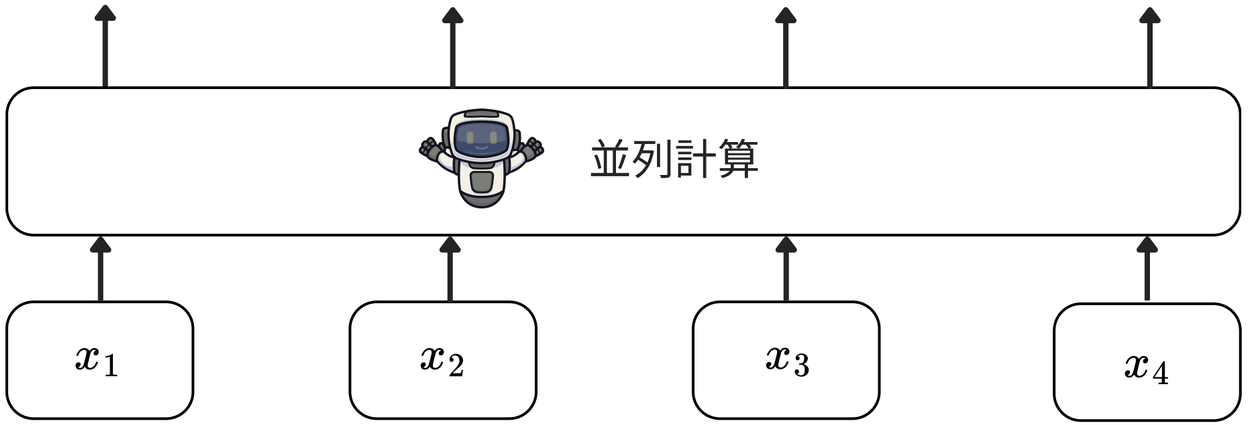
Transformer は LSTM のような逐次処理をなくし, すべての単語を同時に処理をすることを可能にする

LSTM



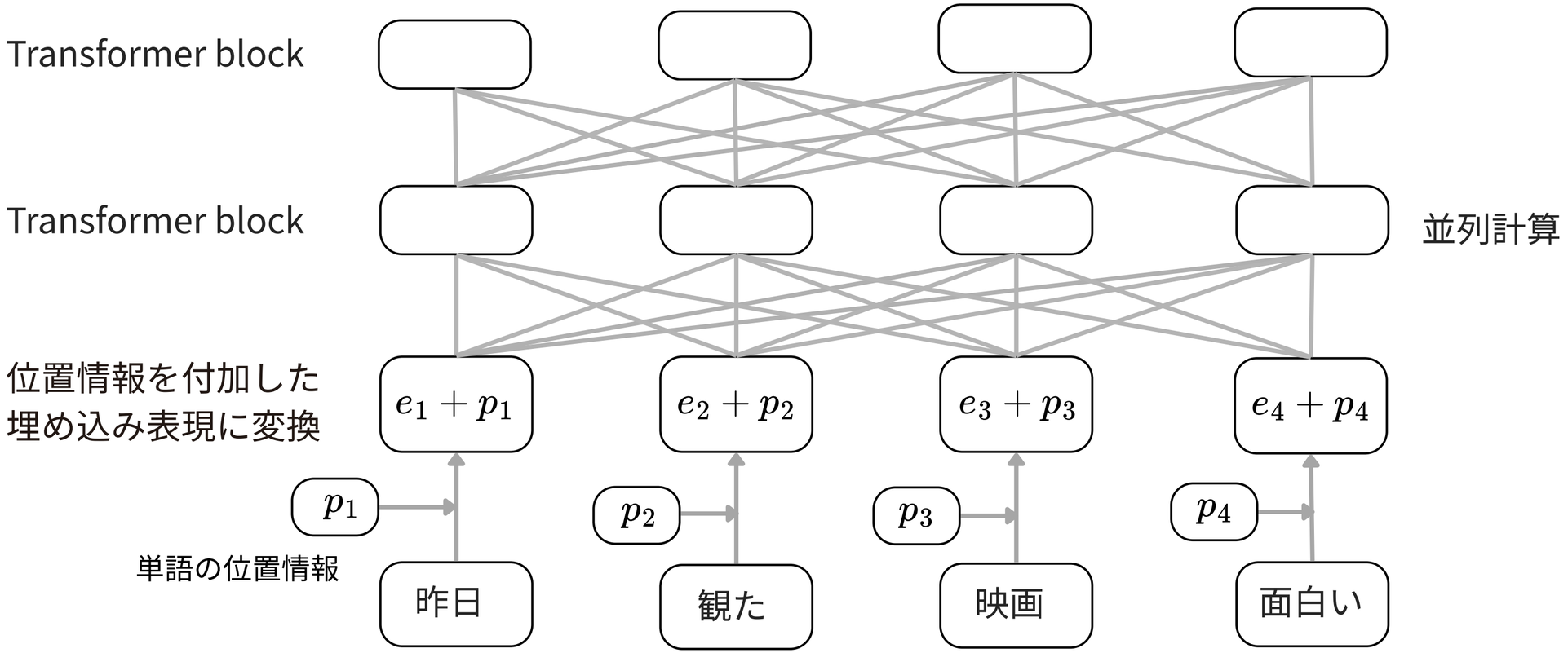
Transformer

Attention is All You Need (2017)



[Transformer の仕組み]

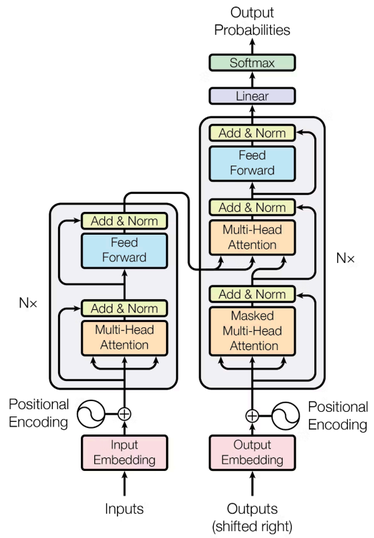
単語の埋め込み表現に位置情報を付加し, Transformer block で処理を行う



[Transformer の特徴]

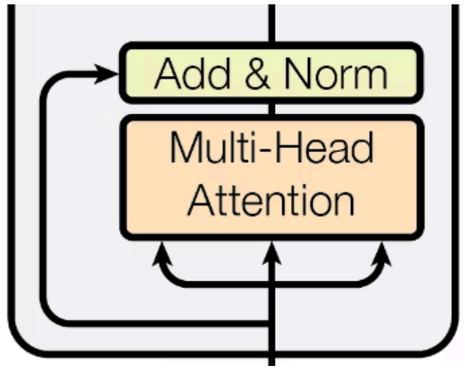
- LSTM のような逐次処理は使わず, 通常の**ニューラル・ネットワークと Attention だけ**で実現
- Atteniton は位置情報を持たないので, 単語の位置情報は Positional Encoding という手法で対応
  (詳細は本講座で)

全体



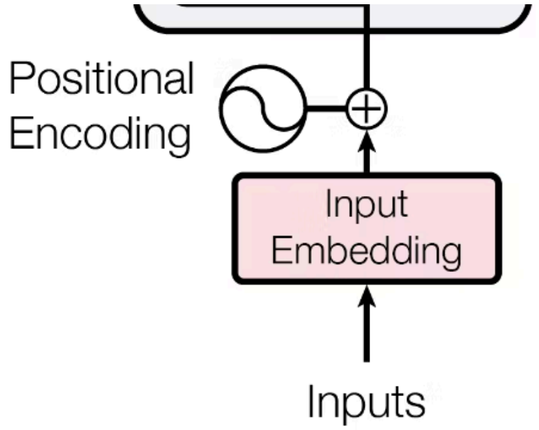
Attention

注意を向けて文脈を理解



Postional Encoding

位置情報を加える



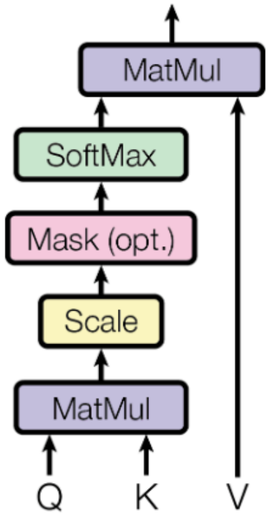
一度, 自分の手で書いてみるとよく理解できます！

[Self-Attention の特徴]

- 翻訳元の文章だけでなく, 自分自身にも注意を向ける
- Transformer では “Scaled Dot-Product Attention” と呼ばれる手法を使う
  (詳細は本講座で)

$$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V$$

Q: Query, K: Key, V: Value



$key_{\text{yesterday}}$
key_{watched}
key_{was}
$key_{\text{interesting}}$

Query との関連度 (注意の大きさ)を計算して,

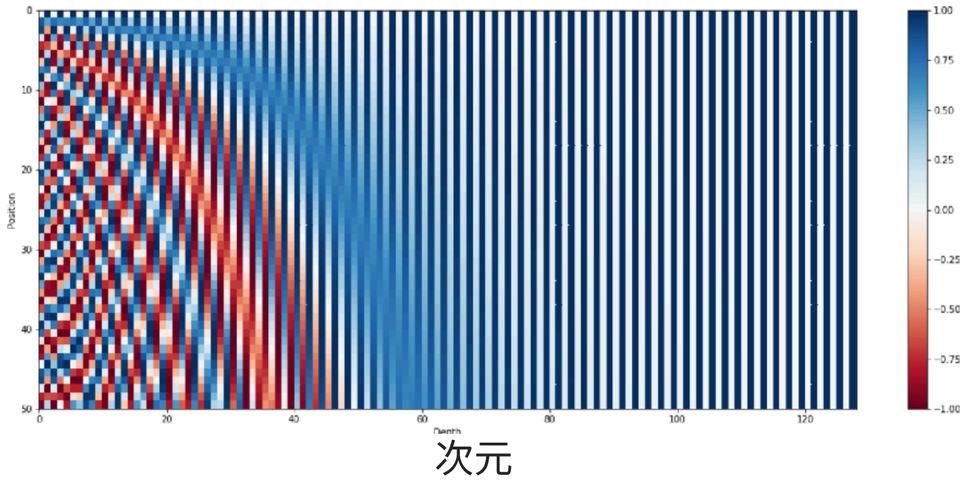
$value_{\text{yesterday}}$
$value_{\text{watched}}$
$value_{\text{was}}$
$value_{\text{interesting}}$

value を重みづけしたものが, “映画”の意味表現

[Positional Encoding]

- Scaled Dot-Product Attention では単語の位置を考慮せず, あくまで単語間の関連性のみを取り込む
- 位置の関係性をモデルに入れるのが Positional Encoding

$$
\begin{cases}
PE_{(pos,2i)} = \sin\left(\dfrac{pos}{10000^{2i/d_{\text{model}}}}\right) \\
PE_{(pos,2i+1)} = \cos\left(\dfrac{pos}{10000^{2i/d_{\text{model}}}}\right)
\end{cases}
$$



- 文章中の位置と次元 (ベクトルの何番目の要素か) によって違う値が振られる
- 必ずしもこの形でなくてよく, この形だと位置が近い単語同士を掛けると値が大きくなるなど, 良い性質を持つ

[Transformer の評価結果]

Transformer を使った結果

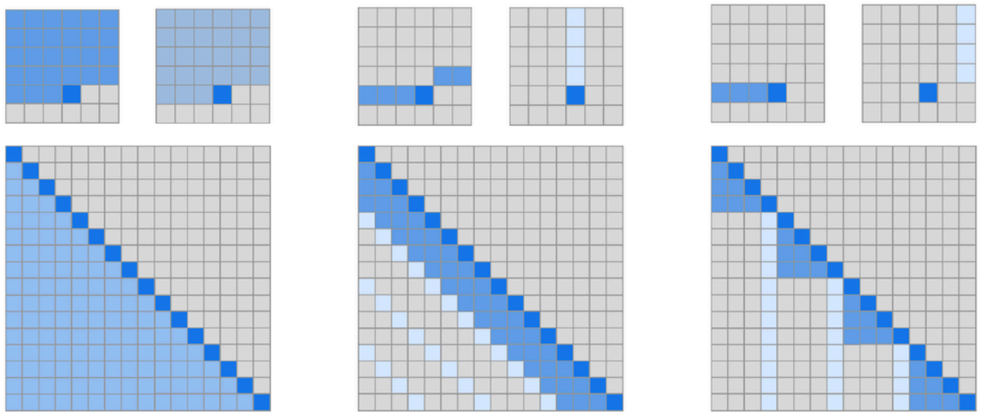
- Transformer (big) では一番良い結果
- Transformer (base model) はそこそこの性能 + 計算効率が非常に良い

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	**$3.3 \cdot 10^{18}$**	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

[Transformer の問題点]

- 文章が長くなると Attention のメモリ使用量が非常に大きい
- OpenAI から “Sparse Transformer” などが提案



(a) Transformer　(b) Sparse Transformer (strided)　(c) Sparse Transformer (fixed)

Figure 3. Two 2d factorized attention schemes we evaluated in comparison to the full attention of a standard Transformer (a). The top row indicates, for an example 6x6 image, which positions two attention heads receive as input when computing a given output. The bottom row shows the connectivity matrix (not to scale) between all such outputs (rows) and inputs (columns). Sparsity in the connectivity matrix can lead to significantly faster computation. In (b) and (c), full connectivity between elements is preserved when the two heads are computed sequentially. We tested whether such factorizations could match in performance the rich connectivity patterns of Figure 2.

Generating Long Sequences with Sparse Transformers (2019)

08

事前学習 + ファインチューニング

[ChatGPTができるまでの歴史]

Neural Network を使った自然言語処理が始まったのは2003年. 2014年頃から一気に加速.

モデルの変遷

Bag-of-Words
単語を数えて予測

共起行列
単語の出現回数で意味を獲得

→ **Word2Vec (2013)**
ニューラルネットワークで単語の意味を獲得

Neural Network (2003)
ニューラルネットワークを初めて言語モデルに導入し, 分散表現の概念を確立

→ **RNN, LSTM (2010, 2014)**
言葉の流れを学習

→ **Attention (2016)**
長文を捕捉

→ **Transformer (2017)**
並列計算で学習を効率化

事前学習 + ファインチューニング (2015)
前もって言語を学習

→ **GPT, BERT (2018)**
モデルと学習手法の融合
初めての大規模言語モデル

>>> **ChatGPT** (2023)
人に寄り添った対話モデルとして学習

[なぜ事前学習が必要か？]

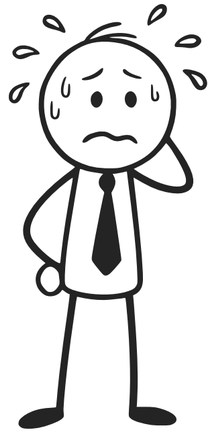
- ラベルデータは人が付ける必要があるので数が限られており, 数百・数千程度のサンプルでは良い精度が出ない.
- 人間で言うと, まったく言葉を知らない人が, 数百・数千の文章だけを覚えて見たことがない文章のセンチメントを当てるといイメージ.

➡ 文章の丸暗記に近い形になってしまう

学習データ

「誠に遺憾ですが, 今期は無配にします」はネガティブ

「**成長のための投資**を行っていきます」はポジティブ



実践

「成長投資が想定よりかさみました」

「成長投資」はポジティブだったような…
「かさみました」なんて見たことない…

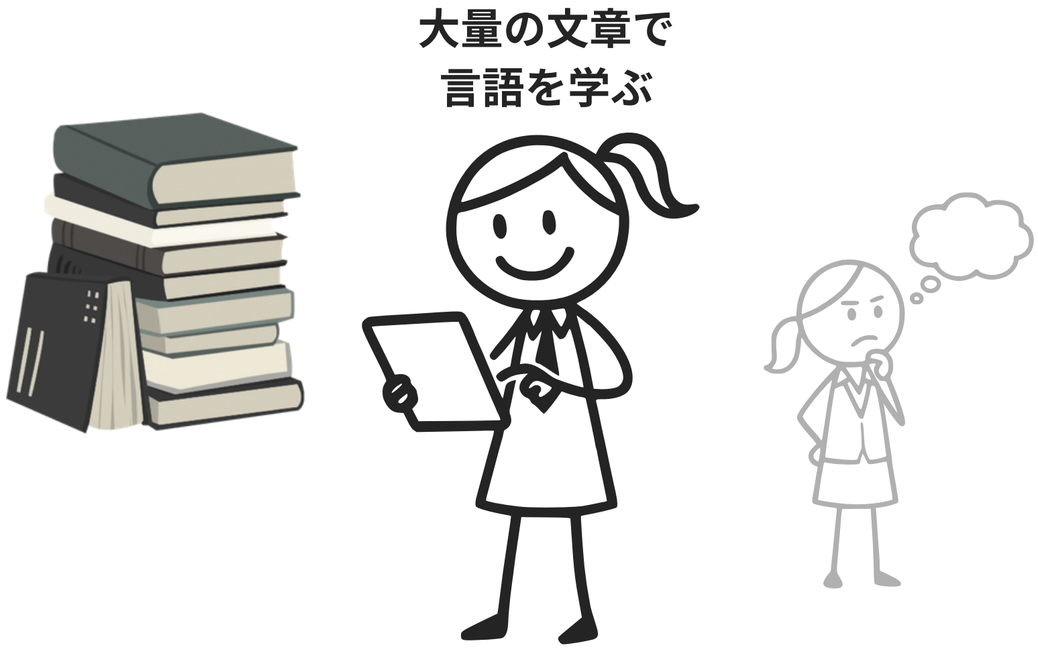
➡ ポジティブ…？

💡 **まずはしっかり言語を学びましょう**

Semi-supervised Sequence Learning (2015)

[事前学習とは]

- タスクを学習する前に, まず, 言語の仕組みや単語などを学びましょうということ.
- 平文があればいいので, 大量のデータを利用できる.



[事前学習の方法]

Auto-Encoder (下図) と Recurrent Language モデル (現在の言語モデルの学習方法) が提案されている.

Auto-Encoder の図. 右半分だけを見ると Recurrent Language Model.

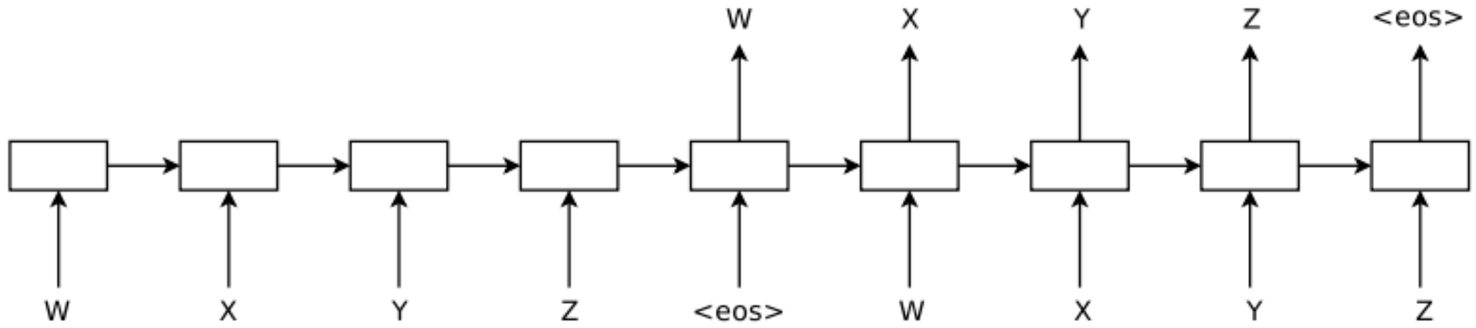


Figure 1: The sequence autoencoder for the sequence "WXYZ". The sequence autoencoder uses a recurrent network to read the input sequence in to the hidden state, which can then be used to reconstruct the original sequence.

- この平文で学習したウェイトを, RNN のパラメータの初期値とする.
- まだ, 何がポジティブで何がネガティブなのかはまったく知らない

Semi-supervised Sequence Learning (2015)

[ファインチューニングとは]

前頁で大量の文章で学習したパラメータをモデルの初期値とし, 解きたいタスクのデータで**パラメータを調整**する.

つまり, 事前学習として大量の文章を学習することで単語や文章の構造を理解する. そして, その後に特定のタスクに必要な知識を学習する.

大量の文章で
言語を学ぶ

言語を理解した上で, タスクを学ぶ

「誠に遺憾ですが, 今期は無配にします」はネガティブ

「成長のための投資を行っていきます」はポジティブ

[事前学習の効果]

Rotten Tomatoes データセット (映画のセンチメント予測) の結果

- LSTM を普通に学習すると20~21%強のerror rate
- Rotten Tomatoes データセット (**小規模データ**) で事前学習し, そのままファイン・チューニングしても精度は**21.7%** と**20.3%**と効果なし

💡 Amazon reviews という大量のラベル無しデータで事前学習すると16.7%と大きく改善

Table 4: Performance of models on the Rotten Tomatoes sentiment classification task.

Model	Test error rate
LSTM with tuning and dropout	20.3%
LSTM with linear gain	21.1%
LM-LSTM	21.7%
SA-LSTM	20.3%
LSTM with word vectors from word2vec Google News	20.5%
SA-LSTM with unlabeled data from IMDB	18.6%
SA-LSTM with unlabeled data from Amazon reviews	16.7%
MV-RNN [28]	21.0%
NBSVM-bi [35]	20.6%
CNN-rand [12]	23.5%
CNN-non-static (ConvNet with vectors from word2vec Google News) [12]	18.5%

変わらず

事前学習のデータが増えると大幅に改善

Semi-supervised Sequence Learning (2015)

09

GPT, BERT = Transformerと事前学習+ファインチューニングの出会い

[ChatGPTができるまでの歴史]

Neural Network を使った自然言語処理が始まったのは2003年. 2014年頃から一気に加速.

モデルの変遷

Bag-of-Words
単語を数えて予測

共起行列
単語の出現回数で意味を獲得

→ **Word2Vec (2013)**
ニューラルネットワークで単語の意味を獲得

Neural Network (2003)
ニューラルネットワークを初めて言語モデルに導入し, 分散表現の概念を確立

→ **RNN, LSTM (2010, 2014)**
言葉の流れを学習

→ **Attention (2016)**
長文を捕捉

→ **Transformer (2017)**
並列計算で学習を効率化

事前学習 + ファインチューニング (2015)
前もって言語を学習

→ **GPT, BERT (2018)**
モデルと学習手法の融合
初めての大規模言語モデル

>>> **ChatGPT** (2023)
人に寄り添った対話モデルとして学習

[GPT, BERT とは]

2018年に事前学習 + ファインチューニングの手法を用いて, 高精度を達成するモデルがいくつか提案された. (GPT, BERT, ELMo, ULMFit)

- その中でも, 特に注目されたモデルは Transformer を採用し、大規模データセットで事前学習し, ターゲットタスクのデータでファイン・チューニングを行った.
  - GPT (**G**enerative **P**re-**T**rained or Generative **P**retrained **T**ransformer)
  - BERT (**B**idirectional **E**ncoder **R**epresentations from **T**ransformers)

💡 **なぜ Transformer か？**

1. Transformer は LSTM に比べ計算効率が高い

   → 大量のデータを学習可能

2. Self-Attention のみを使った Transformer は, 時系列にデータが流れるなどといった仮定が少なく非常に柔軟な, 汎用性が高いモデル

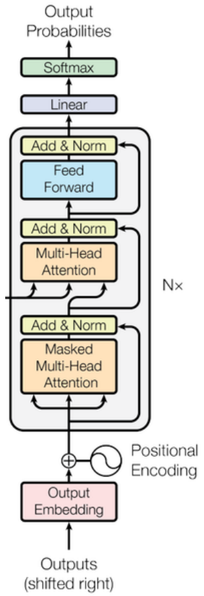
   → 大量データになるほどオーバー・フィッティングせず性能を発揮

[GPT と BERT の概要]

GPT と BERT はモデルに Transformer を採用し,

- 当時では非常に大きなデータセット (Wikipedia, BookCorpus など) で事前学習
- 特定のタスクを処理するために, ファインチューニングを行う

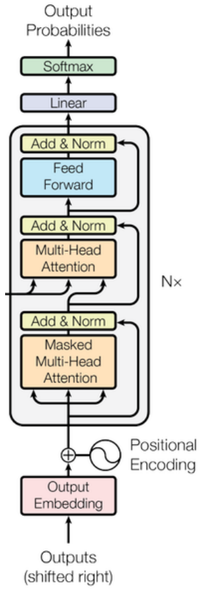
事前学習



Wikipedia

BookCorpus

ファイン・チューニング



センチメント

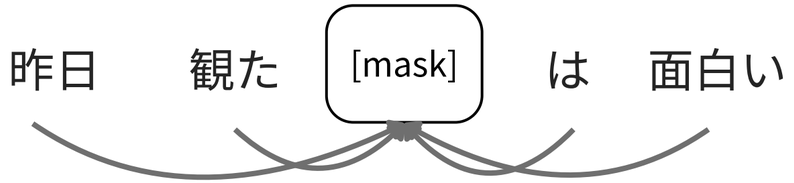
[GPT と BERT の違い]

GPT と BERT の違いは学習方法

- GPT は順番に次の単語, 次の単語を予測して学習する Recurrent Language Model. Attention の向きも自分よりも古い情報のみ. Attention は過去にのみ向く.

Improving Language Understanding by Generative Pre-Training (2018)

- BERT は文章中の単語をランダムにマスクし, そのマスクされた単語を当てることで事前学習する **Masked Language Model**.
  予測する単語の先の単語も参照する双方向の Attention. (なぜ双方向かなどの詳細は本講座で)



BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (2018)

[BERT の発表後]

- 双方向の attention を利用した BERT は自然言語処理のブレイクスルー
- 東北大などが日本語の Wikipedia で事前学習した 日本語 BERT を提供.

  🤗 Hugging Face の Transformers というライブラリで簡単に利用可能
- 💡 それぞれのタスクでファイン・チューニングする

  「未来では教師データを適切に作ることができる人が重要になってくる！」と言った意見も...
- BERT の発展形も多数出現
  - ALBERT (**A** **L**ite **BERT**)
  - RoBERTa (**R**obustly **O**ptimized **BERT** Pretraining **A**pproach)
  - T5 (**T**ext-**t**o-**T**ext **T**ransfer **T**ransformer) など

💡 RoBERTa の示唆 ➡ **事前学習のデータセットを大きくしたら (160 GB) 精度がすごく改善**

[BERT の使い道]

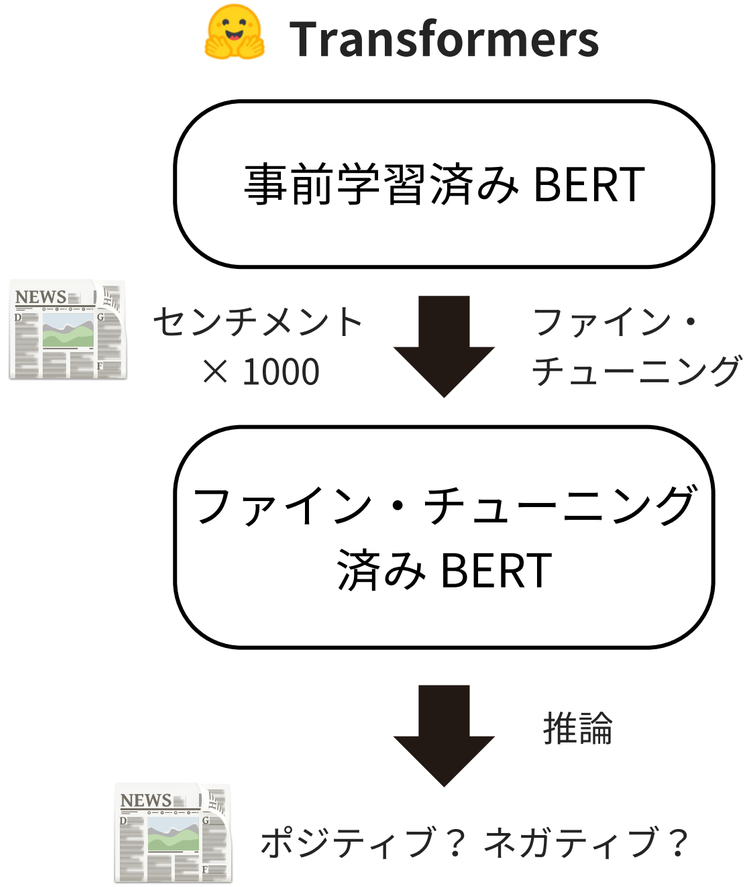
- BERT はファイン・チューニングで高い精度を出すモデルとして現場で非常に使われていた.
  今でも大きな話題にはならないものの開発の需要あり.

BERT のメリット

- 計算コストが安い
  ローカルの CPU でも十分推論可能
- 決定論的 … 毎回, 同じ回答が得られる
- 推論が速い

BERT の使われ方

- 文書の分類
- 固有表現抽出, など



[GPT-2, GPT-3]

- 「人間は特定のタスクをこなすのに1000件も1万件もデータを見て教えてもらう必要はない」
- GPT においては, GPT-2 からは BERT と違い, ファイン・チューニングはせず, **事前学習とわずかな例 (zero-shot, few-shot) だけで良い精度を出せるような汎用モデルを模索**

Language Models are Unsupervised Multitask Learners (2019)

zero-shot
例は見せずにいきなりタスクを解かせる

few-shot
数例サンプルを見せてタスクを解かせる

- モデルのサイズと学習データをとにかく巨大化

GPT-2 の印象 (私見)

- 確かに文章にはなってて面白いけど, さすがにまだ使えない...

GPT-3 の印象 (私見)

Language Models are Few-Shot Learners (2020)

- 文章にはなってるけど, フェイク・ニュースを作るぐらいしか使い道がないのでは...

[zero-shot, few-shot, ファインチューニング]

zero-shot

「昨日観た映画は面白かった」のセンチメントはポジティブですか, ネガティブですか？
=>

few-shot

次のセンチメントを答えてください.
感動した => ポジティブ
つまらなかった => ネガティブ
良かった => ポジティブ

昨日観た映画は面白かった
=>

ファインチューニング

感動した => ポジティブ

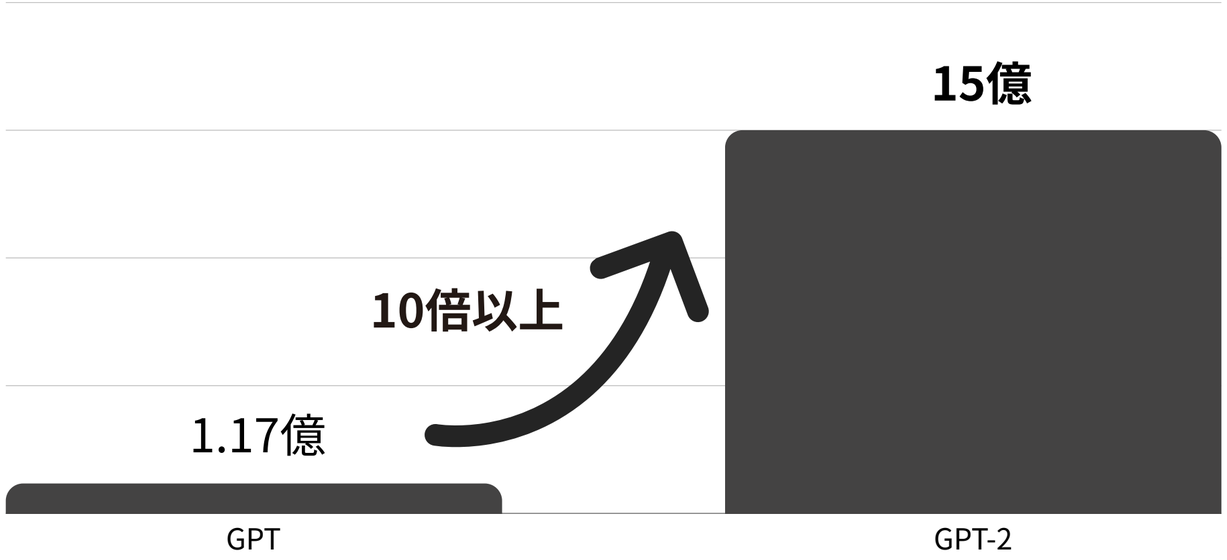
パラメータを更新

つまらなかった => ネガティブ

パラメータを更新

昨日観た映画は面白かった
=>

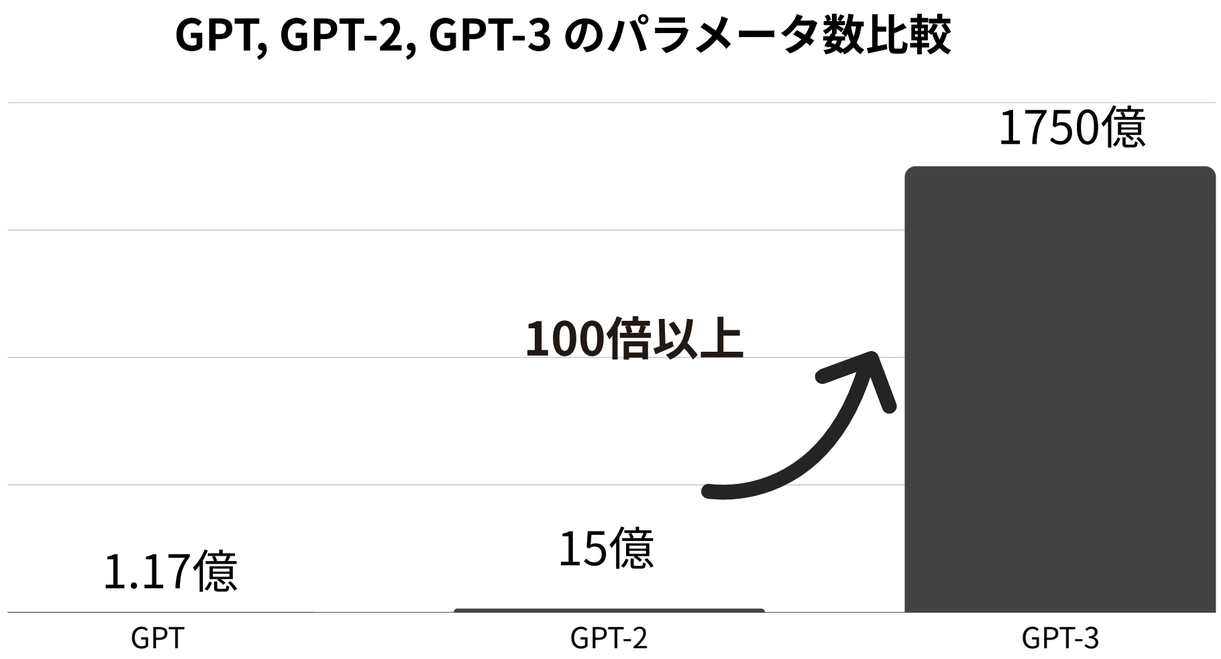
[GPT-2 のモデルサイズとアウトプット]



GPT-2 日本語モデル『rinna』を使って「筋トレするメリットは、1. 健康になる 2. 」と入力すると…

筋トレするメリット 1. 健康になる。 2. 美容・健康効果。2. 代謝アップ。3. 血流が良くなる。4. 基礎体力アップ。5. 筋力低下予防。6. ストレス解消に。7. 普段の生活でも役に立つ。8. 若返る。9. 糖尿病や認知症予防。 11. 病気予防。12. 病気進行予防。13. 病気予防。15. 筋力の衰えを防ぐ。 13. 強弱がある体になる。

[GPT-3 のモデルサイズとアウトプット]



プロンプト

Plotly を使う利点を説明します。
1.インタラクティブな図が生成できること。

生成結果

2. R や Python で簡単にグラフを生成できること。
3.必要に応じてグラフを整形できること。
…

[GPT-3 のすごさ]

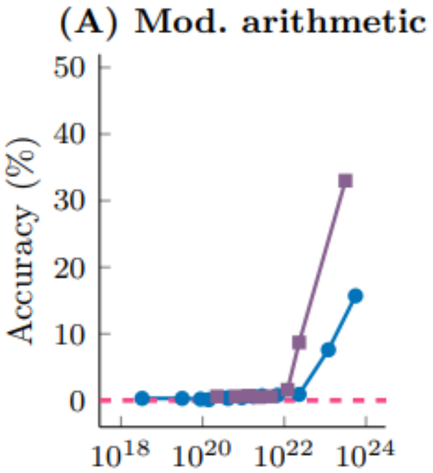
- GPT-3 が生成した偽ニュースは人間でも区別がつかない

	Mean accuracy	95% Confidence Interval (low, hi)	t compared to control (p-value)	"I don't know" assignments
Control	88%	84%–91%	-	2.7%
GPT-3 175B	52%	48%–57%	12.7 (3.2e-23)	10.6%

Table 3.12: People's ability to identify whether ∼ 500 word articles are model generated (as measured by the ratio of correct assignments to non-neutral assignments) was 88% on the control model and 52% on GPT-3 175B. This table shows the results of a two-sample T-Test for the difference in mean accuracy between GPT-3 175B and the control model (an unconditional GPT-3 Small model with increased output randomness).

たくさん事前学習して精度が良くなっても, 単にたくさんのことを覚えて, それを繰り返しているだけでは？

- 事前学習データに, 3桁の足し算の表現は17個のみ, 3桁の引き算に関しては2つしかなかったにもかかわらず, 80%や90%といった精度で計算ができている.
- 「"Gigamaru"は日本の楽器です。その例は」とインプットすると,「叔父が贈り物としてくれた"Gigamaru"を持っています。私は家でそれを演奏するのが大好きです」と知らない単語でも文章を生成する.
- モデルが巨大化することで**汎化性能が発現** (創発的能力, emergent abilities)



Emergent Abilities of Large Language Models (2022)

10

ChatGPT

[ChatGPTができるまでの歴史]

Neural Network を使った自然言語処理が始まったのは2003年. 2014年頃から一気に加速.

モデルの変遷

Bag-of-Words
単語を数えて予測

共起行列
単語の出現回数で意味を獲得

Word2Vec (2013)
ニューラルネットワークで単語の意味を獲得

Neural Network (2003)
ニューラルネットワークを初めて言語モデルに導入し, 分散表現の概念を確立

RNN, LSTM (2010, 2014)
言葉の流れを学習

Attention (2016)
長文を捕捉

Transformer (2017)
並列計算で学習を効率化

事前学習 + ファインチューニング (2015)
前もって言語を学習

GPT, BERT (2018)
モデルと学習手法の融合
初めての大規模言語モデル

ChatGPT (2023)
人に寄り添った対話モデルとして学習

[GPT-3 の課題]

- 特定のタスクであれば, ファインチューニングした BERT 系のモデルの方が良い
  - ➡ さらなる大規模化 (パラメータ, 学習データ)で解決

- すごいけど, 単なる文章生成をするツール

  対話モデルにするには, もっと人に寄り添う必要がある
  - ➡ 人間のフィードバックによる強化学習 (**R**einforcement **L**earning from **H**uman **F**eedback; RLHF) の開発

[ChatGPT の仕組み ~ InstructGPT]

ステップ 1. 大量のテキストデータを事前学習 (GPT-3)

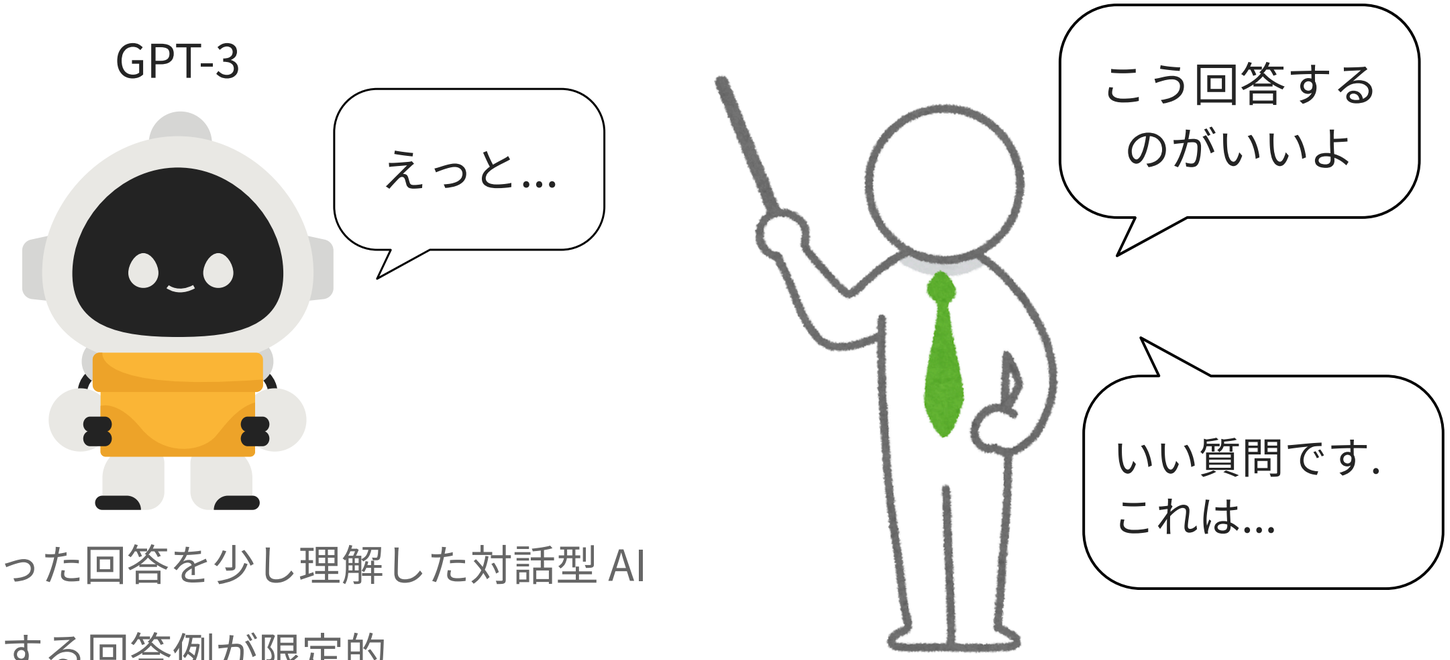
大量の文章を読んで事前学習

Training language models to follow instructions with human feedback (2022)

[人のラベルで教師あり学習]

ステップ 2. 人間が作成した回答案を使って学習 → ファイン・チューニング (指示チューニング)

人間が作成した回答案を使って学習



人に寄り添った回答を少し理解した対話型 AI

- 学習する回答例が限定的
- 良い回答は学べたが, 回答の良し悪しはわからない

[報酬モデルを作成]

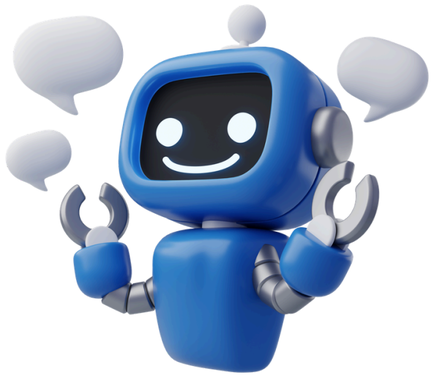
ステップ 3. ステップ2の“人に寄り添った回答を少し理解した対話型 AI” から報酬モデルの構築

良い回答の仕方を学んだ報酬モデルを作成 (GPT-3 とは別)

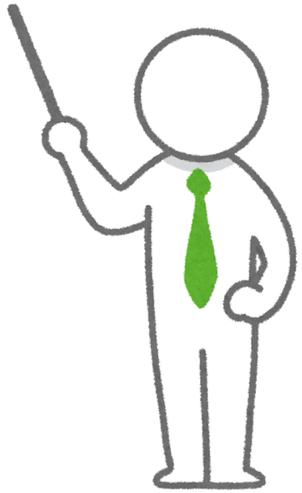
どのような回答が良いかを判定する報酬モデルを構築

良い回答と悪い回答がわかった！

報酬モデル



この回答はどう？



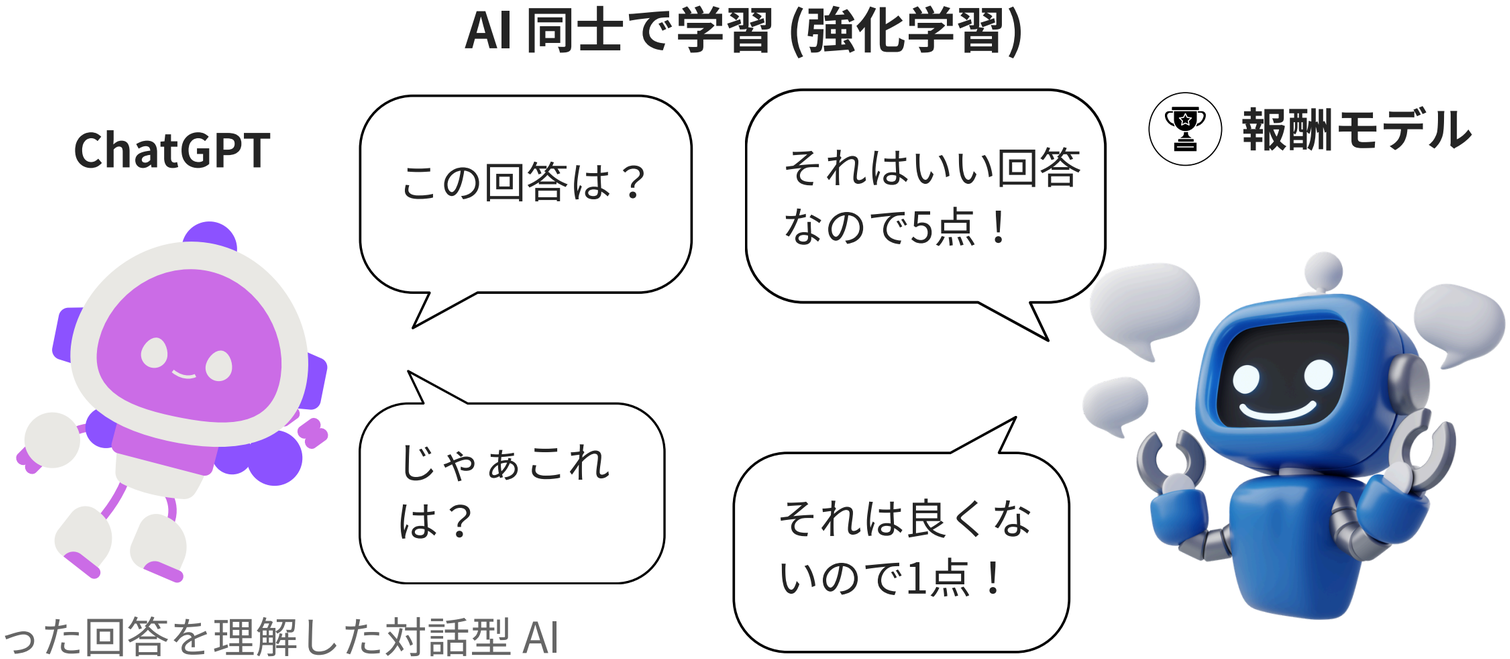
それはいい回答なので5点です

良い回答を判別できる AI

※ 厳密には, 点数をつけるのは人によって基準が違ったりして統一できないため, 2つのペアを比較してどっちが良いかを回答してランキング

[AI 同士の強化学習で成長加速]

ステップ 4. ステップ2の“人に寄り添った回答を少し理解した対話型 AI” と “報酬モデル”を使って, AI 同士で学習 (強化学習)



人に寄り添った回答を理解した対話型 AI

1750億パラメータの GPT-3 よりも100分の1以上小さい13億パラメータの InstructGPT の方が良い評価を得られた

[ChatGPT のその後]

- いい加減なことを知ったかぶって答えてしまう (ハルシネーション)
- 論理的な思考ができない
- 複雑なタスクはこなせない
- CoT, ReAct, Reflection, Tree-of-Thought など様々なアイデア・手法が提案

この辺りの詳細はまた本講座で…

11

まとめ

[振り返り]

初期の自然言語処理モデルからGPT, BERT, ChatGPTまでをざっくり説明しました

Bag-of-Words
単語を数えて予測

共起行列
単語の出現回数で意味を表現

Word2Vec (2013)
ニューラルネットワークで単語の意味を獲得

Neural Network (2003)
ニューラルネットワークを初めて言語モデルに導入し, 分散表現の概念を確立

RNN, LSTM (2010, 2014)
言葉の流れを学習

Attention (2016)
長文を捕捉

Transformer (2017)
並列計算で学習を効率化

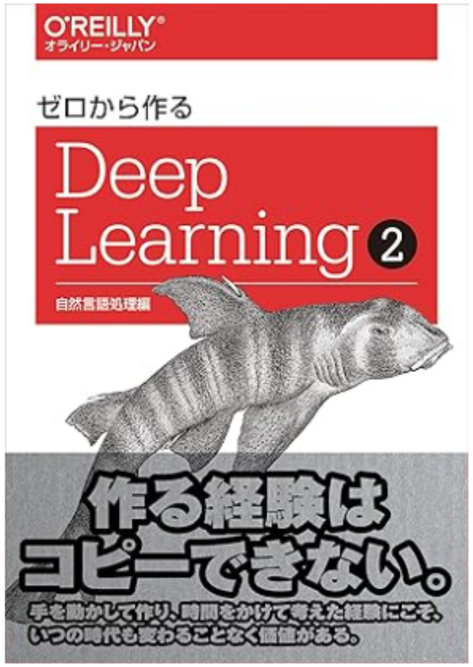
事前学習 + ファインチューニング (2015)
前もって言語を学習

GPT, BERT (2018)
モデルと学習手法の融合
初めての大規模言語モデル

ChatGPT (2023)
人に寄り添った対話モデルとして学習

[参考図書]

🔰自分で実装して自然言語処理の基礎をしっかりと理解するための本 (ただし Attention まで)



🔰 PyTorch で Transformer, BERT を実装する方法がわかりやすく説明されている本

大規模言語モデルの全体像を解説. Transformers を使った実装などもあり.

🔰 入門者向け

[本講座の内容]

本講座では, 以下の内容を想定しています.

1. Word2Vec, RNN, LSTM のを実装で確認
2. Transformer の詳細を説明し, PyTorch での実装を確認
3. BERTの詳細説明とライブラリ Transformers を使って BERT をファインチューニングして実践
4. ChatGPT の仕組みと CoT (Chain-of-Thought), ReAct といった手法・実装例を解説

💡 本講座を受講いただいた方にはできるだけ質問に答えます！

本講座で聞きたいことなどがあればご連絡ください！

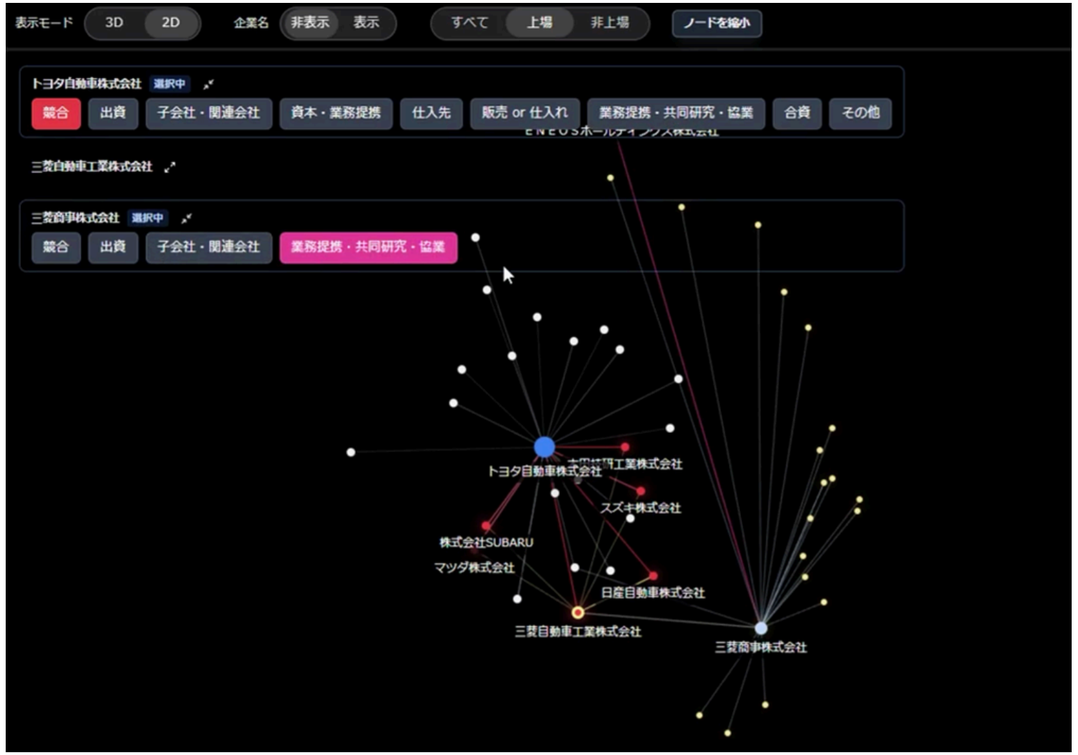
[ご質問への回答]

【質問】

直近のポストでお見かけした「企業ネットワークの可視化」に対して、LLMはどんな使われ方をされていますか？

【回答】

現状では, 対象となる企業の書類から関連する企業との関係を取得する際に, LLM を利用しています.
今後は企業情報を埋め込んで類似企業との距離を表現したり, LLM で分析するといった組み合わせを考えています.



[THANK YOU!]

ありがとうございました！

石津 俊輔

shunsuke.ishizu0106@gmail.com

shunsuke-ishizu@finosight-analytics.co.jp