

# LLM (Large Language Model) 大規模言語モデル

津田塾大学 情報科学科

新田善久

nitta@tsuda.ac.jp

# 大規模言語モデル

- LLM (Large Language Model, 大規模言語モデル)は、大規模なテキストデータで訓練された大規模なパラメータで構成されるニューラルネットワーク。
- パラメータ数とテキストデータの拡大により2020年以降性能が飛躍的に向上し、ChatGPT-3.5 (2022) の登場により注目を集めた。この背後にある技術が Transformer (2017) である。
- 従来、言語などの時系列データを扱うために、中間層に再帰構造を持つ深層学習モデル Recurrent Neural Network (RNN, 1986) が使われてた。その後、「入力の情報を全て出力側に渡し重要度のスコア付けを学習する」ことで「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る仕組み」であるAttention (2015)が導入された。

# 大規模言語モデル

- Transformer (2017) は系列データから別の系列データに変換する深層学習モデルだが、RNNを使わずに Attention のみで構築することによりGPUを利用した学習の並列化が可能になり、大規模な自然言語処理モデルを学習できるようになった。Google 翻訳や BERT (2018)/ GPT-2 (2019)/ GPT-3 (2020)/ GPT-3.5 (2022)/ GPT-4 (2023) などで使われている。
- BERT (2018) は、Transformer ベースの encoder 部分のみを重ねた構造の深層学習モデルで、文章内のマスクした単語を予測することを得意とする。学習を「事前学習」と「ファインチューニング」の2段階に分けることにより、効率良い学習が可能。

# 大規模言語モデル

- GPT-2 (2019) は Transformer の decode 部のみを重ねた構造のモデルで、任意の文章に続く次の単語を予測する処理を得意とする。「ファインチューニング」に加えて「Zero Shot (任意の文章に続く次の文章を予測する能力を使って、未学習のタスクの推論を行う)」手法を用いている。
- GPT-3 (2020) では「ファインチューニング」に加えて「プロンプトプログラミング (自然言語でどのような知識をどのように使いたいかを例示する)」を用いて精度を向上させる。
- GPT-3.5 (2022) では、人間のフィードバックによる強化学習を用いて、言語モデルと人間の指示の整合性を高めたモデル。
- GPT-4 (2023) は、テキストだけでなく画像も入力可能となった。

# 単語のベクトル表現 word2vec

- 単語を one-hot-vector (1つの要素だけが1で、残りは全部0)で表現すると、ボキャブラリが大きくなるとベクトルも長くなり、非現実的。1万語を扱うには長さ1万のベクトルが必要となる。

0	技術
0	義務
⋮	
0	魔法
1	王子
0	勇敢
⋮	
⋮	

- 単語を数百次元の特徴ベクトルで表したい。
- ある単語のベクトル表現を、その単語の周囲で頻繁に使用される単語を使って決定する。  
(例) 王子はまだ幼く…  
(例) 王子は勇ましく狩りに出かけ…

# 単語のベクトル表現 word2vec skip-gram

「単語を与えて、その前後の単語を推測する」タスクを行う

[元文章] 美しい白馬に乗った王子は魔法の剣で竜を倒した

[距離3]

□ ? □ ? □ 王子 □ ? □ ? □

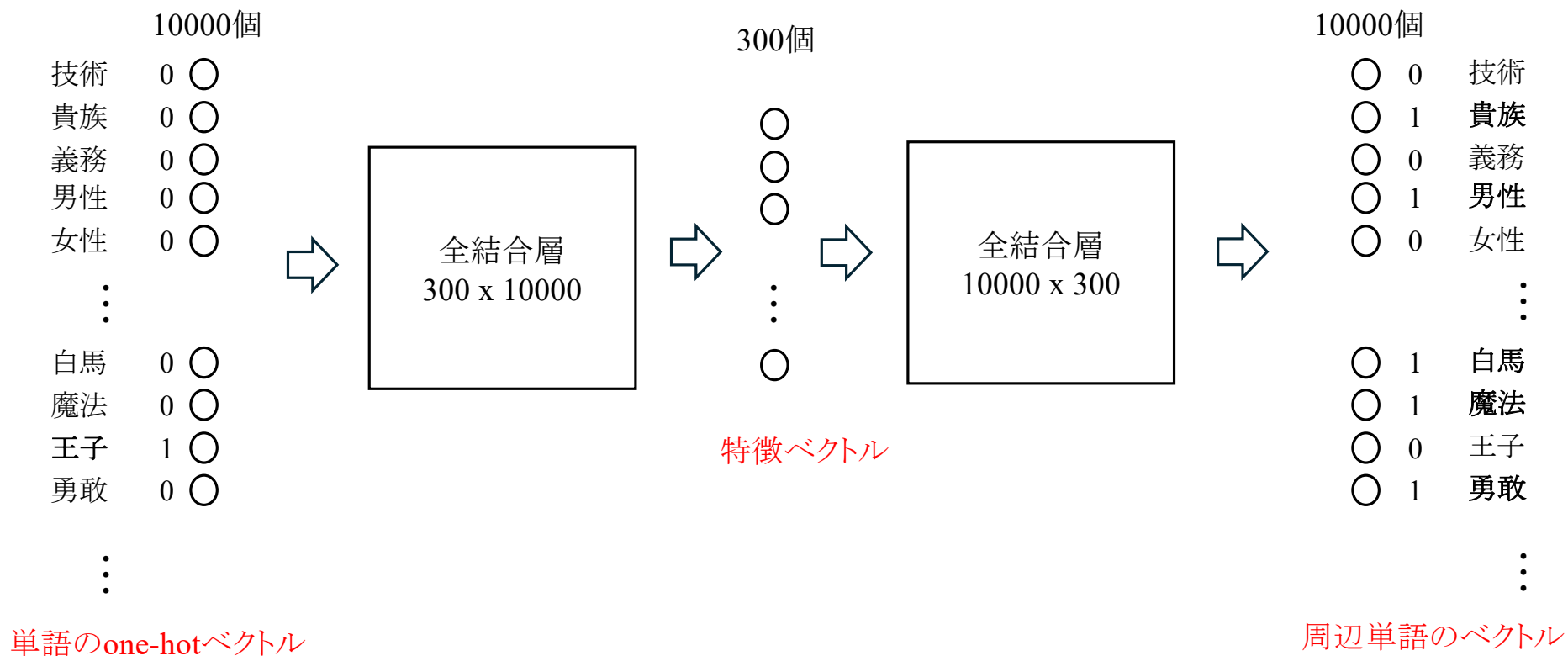
[距離5]

□ ? □ ? □ ? □ ? □ 王子 □ ? □ ? □ ? □ ? □

[https://github.com/udacity/deep-learning-v2-pytorch/blob/master/word2vec-embeddings/Skip\\_Grams\\_Solution.ipynb](https://github.com/udacity/deep-learning-v2-pytorch/blob/master/word2vec-embeddings/Skip_Grams_Solution.ipynb)

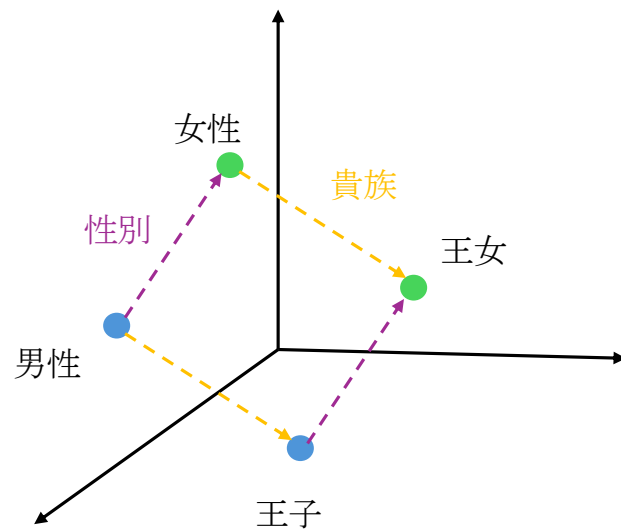
# 単語のベクトル表現 word2vec skip-gram

- ボキャブラリが1万語の場合



# word2vec: 特徴ベクトル

- 単語を特徴ベクトルに変換すると、その方向に「意味や概念」がエンコードされている。
- (例) 王子から王女へのベクトルは、男性から女性へのベクトルと似た方向を向く





# トランスフォーマー (Transformer)

「ある要素のベクトルと、他の要素のベクトルの関連性を学ぶ」

[元文章] 美しい 白馬 に 乗った 王子 は 魔法 の 剣 で 竜 を 倒した



単語位置を表す  
ベクトルも追加  
されている

query

美しい	白馬	に	乗っ	た	王子	は
-----	----	---	----	---	----	---

key

美しい
白馬
に
乗っ
た
王子

回転 (例: 形容詞)

	回転 (例: 名詞)						
美しい	0.02	0.48	0.02	0.01	0.01	0.02	0.03
白馬	0.43	0.12	0.28	0.39	0.15	0.33	0.01
に	0.08	0.11	0.08	0.28	0.08	0.08	0.01
乗っ	0.03	0.28	0.03	0.10	0.33	0.42	0.04
た	0.04	0.05	0.01	0.40	0.11	0.29	0.04
王子	0.02	0.03	0.01	0.52	0.29	0.19	0.27

(例)形容詞は直後の名詞と関連が深い

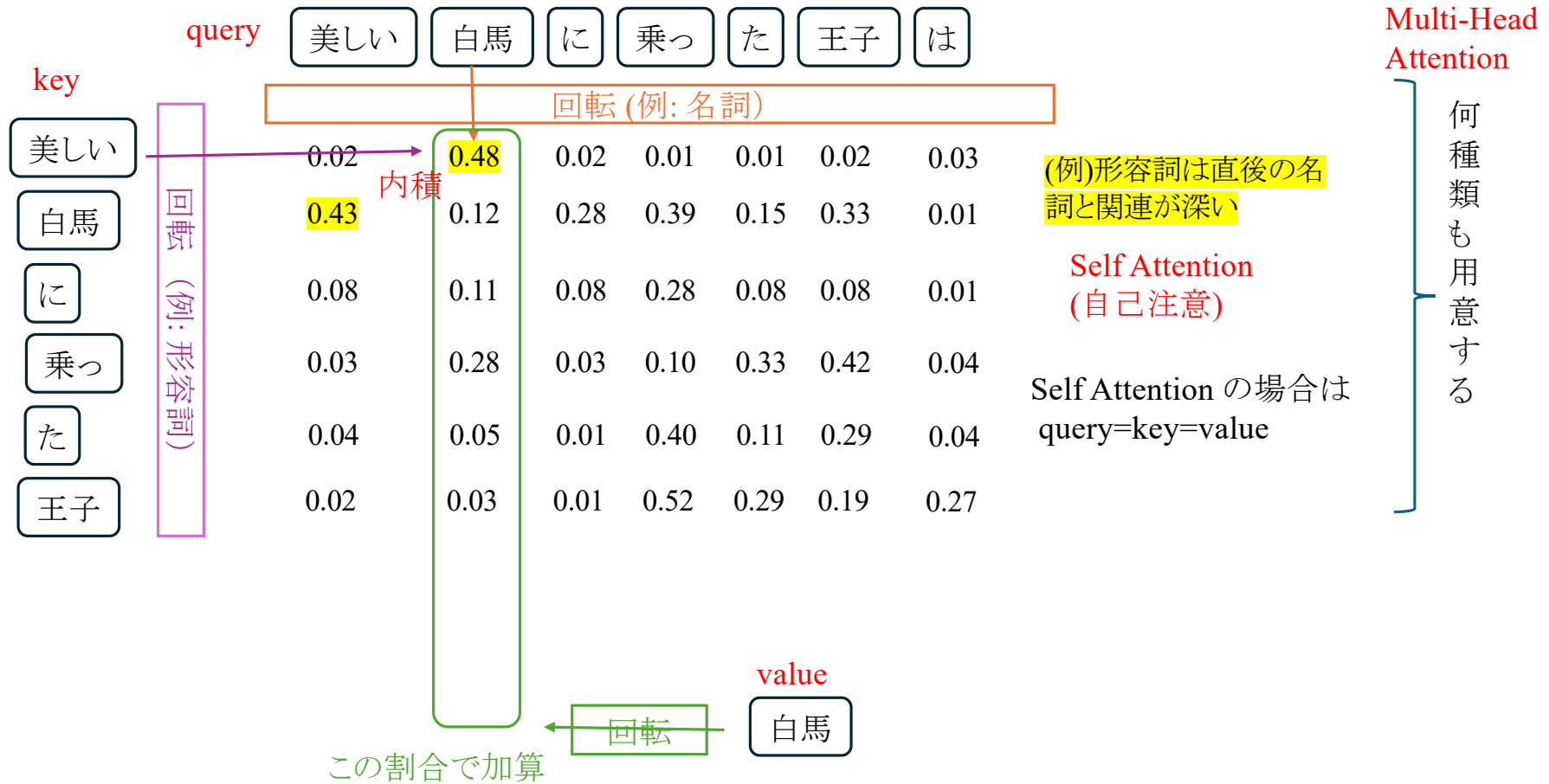
Self Attention  
(自己注意)

(key, value) ペア

Multi-Head Attention

何種類も用意する

# トランスフォーマー (Transformer)



# Transformer

- 「ある語をあらわすベクトルに、周りの語のベクトルを回転させながら重み付けして足していく」イメージ

(例) さくら

1. 桜の木(植物として桜)
2. 桜の花(桜の花そのもの)
3. 偽客(さくら、仕込み)
4. 女性の名前
5. 桜色(薄いピンク色)

彼女は満開のさくらの下で写真を撮るのが大好きです。



観客の中には、盛り上げ役として雇われたさくらが混じっていました。



# GPT: Generative Pre-trained Transformer

## 生成的な事前学習済みのトランスフォーマー

- 文を与えて、その次に続く単語を推測させる

美しい 白馬 に 乗っ た 王子 は ?



0.25 お城

0.21 困難

0.20 魔法

0.18 友

0.15 美しい

⋮

生成した単語を追加して、さらに推測させる

お城に戻りました。

困難に立ち向かいました。

魔法の森に入りました。

友を助けました。

美しい姫に出会いました。

# GPT: Generative Pre-trained Transformer

## 生成的な事前学習済みのトランスフォーマー

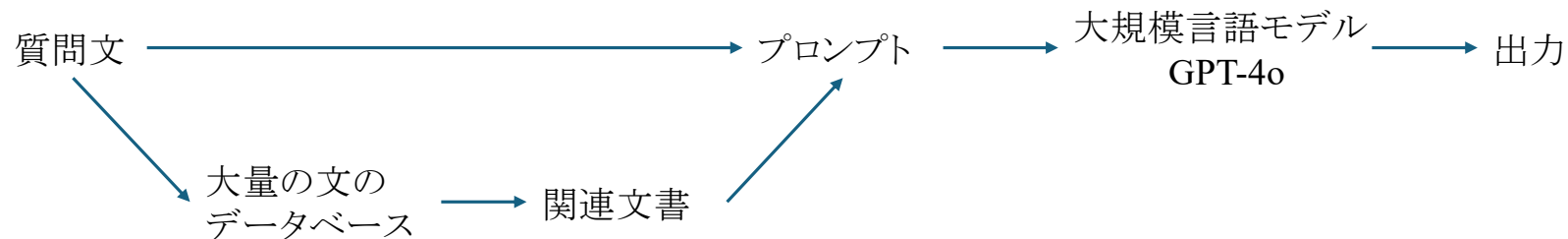
- プロンプト: 「美しい白馬に乗った王子は」の次に続く単語を5通り予測して、確率分布とともに示しなさい。
- 出力: 「美しい白馬に乗った王子は」の次に続く単語を5通り予測し、それぞれの確率分布は以下の通りです:
  1. お城に戻りました。 25%
  2. 困難に立ち向かいました。 20%
  3. 魔法の森に入りました。 15%
  4. 友を助けました。 20%
  5. 美しい姫に出会いました。 20%

# 文のベクトル表現 (文埋め込み)

- 文の意味をベクトルで表現する。
- 正例ペアではベクトルの類似度が高くなるように、負例ペアでは類似度が引くくなるように学習する。(2文の関係)含意 / 中立 / 矛盾
- データ例1
  - 前提文: ガレージで、壁にナイフを投げる男。
  - 含意文: ガレージに男がいます。
  - 矛盾文: 男が台所のテーブルで本を読んでいます。
- データ例2
  - 前提文: ラップトップコンピューターを使用して机に座っている若い白人男
  - 含意文: 人は椅子に座っています
  - 矛盾文: 黒人はデスクトップコンピューターを使用します

# text-to-text

- 文で質問し、文で回答を得る。
- 質問文にいろいろな情報を詰め込める→プロンプト・エンジニアリング
- AIが事前に学習していない事柄にも回答できる。
- 「文」をベクトル化。「似た文」＝「似た方向を向いたベクトル」



# OpenAI Assistants API: File Search

- "File Search" とは、いろいろな形式のテキストを使って、アシスタントに外部からの知識を拡張できる機能
- 知識ベース **VectorStore** に文書をアップロードしておいて、**GPT**に参照を指示する。

(例) Nasdaq100 の目論見書を与えて、その特色を答えさせる。

nissei\_nasdaq100\_prospectus.pdf

「投資信託説明書（交付目論見書）」（使用開始日 2024.06.21）

<購入・換金手数料なし>ニッセイNASDAQ100インデックスファンド」

[https://nw.tsuda.ac.jp/lec/DeepLearning/openai\\_assistant/data/nissei\\_nasdaq100\\_prospectus.pdf](https://nw.tsuda.ac.jp/lec/DeepLearning/openai_assistant/data/nissei_nasdaq100_prospectus.pdf)



投資信託説明書（交付目論見書）  
使用開始日 2024.06.21

<購入・換金手数料なし>  
ニッセイNASDAQ100インデックス  
ファンド

追加型投資 / 海外 / 株式 / インデックス型



ニッセイアセットマネジメント株式会社  
0120-762-506  
https://www.nam.co.jp/



# OpenAI: GPT-4o への質問 (関連文書付き)

```
assistant = client.beta.assistants.create(  
    name = "Financial Analyst Assistant",  
    instructions = "あなたは株式投資の専門家です。知識ベースを利用して、投資信託の目論見書に関する質問に回答してください。",  
    model="gpt-4o",  
    tools=[{'type': 'file_search'}], )
```

```
thread = client.beta.threads.create( messages = [  
    { 'role': 'user',  
      'content': 'nasdaq100 の特色を教えてください。',  
      'attachments': [ { 'file_id': message_file.id,  
                          'tools': [ { 'type': 'file_search' } ] } ],  
    }  
] )
```

# OpenAI: GPT-4o からの回答

NASDAQ100の特色について、以下のポイントが挙げられます：

1. **\*\*インデックス構成\*\***： NASDAQ100指数は、NASDAQ市場に上場している金融銘柄を除いた時価総額上位100銘柄で構成されています [0] [1]。
2. **\*\*投資の目標\*\***： NASDAQ100指数(配当込み、円換算ベース)の動きに連動する投資成果を目指しています [0] [3]。
3. **\*\*為替ヘッジの方針\*\***： 原則として、対円での為替ヘッジは行いません [0] [1]。
4. **\*\*投資対象\*\***： DR(預託証券)や株式等と同等の投資効果が得られる証券および証書等を含みます [0] [1]。

NASDAQ100は、特にハイテク企業が多く含まれており、技術革新の動向による影響を受けやすいインデックスです。また、為替リスクや各国の政治・経済情勢などにも影響を受ける点が留意点となります。これらの特色は、投資判断をする上で重要な情報となりますので、リスク管理を含めて総合的に考えられることが望ましいです。

[0] nissei\_nasdaq100\_prospectus.pdf [1] nissei\_nasdaq100\_prospectus.pdf  
[2] nissei\_nasdaq100\_prospectus.pdf [3] nissei\_nasdaq100\_prospectus.pdf  
[4] nissei\_nasdaq100\_prospectus.pdf [5] nissei\_nasdaq100\_prospectus.pdf  
[6] nissei\_nasdaq100\_prospectus.pdf [7] nissei\_nasdaq100\_prospectus.pdf

# ChatGPTに質問:transformer の機能を、例をつかってわかりやすく説明してください。

Transformerは、自然言語処理(NLP)における強力なモデルアーキテクチャの一つで、特に機械翻訳や文章生成などに優れています。Transformerの主な特徴は自己注意(Self-Attention)メカニズムです。以下に、例を用いてその機能を説明します。

## 自己注意メカニズム

自己注意は、入力された単語(トークン)の間の関係性を計算し、重要な情報を強調します。これにより、文脈を考慮して単語の意味を理解します。

## 例:英語から日本語への翻訳

例えば、「The cat sat on the mat.」という文を「猫はマットの上に座っていた。」と翻訳する場合を考えてみましょう。

### 1. エンコーダ(Encoder):

1. エンコーダは、入力文「The cat sat on the mat.」を受け取り、各単語をベクトル(数値の配列)に変換します。
2. 自己注意メカニズムを使って、各単語の重要度を計算し、文全体のコンテキストを考慮してベクトルを調整します。
3. このプロセスを複数の層で繰り返し、文の意味をエンコードした一連のベクトルを出力します。

### 2. デコーダ(Decoder):

1. デコーダは、エンコーダからの出力ベクトルと、翻訳の初期部分「猫は」を受け取ります。
2. 自己注意メカニズムを使って、現在の出力とエンコーダからのベクトルを基に次の単語の予測を行います。
3. この場合、「マットの」を予測します。
4. このプロセスを繰り返し、最終的に「猫はマットの上に座っていた。」という翻訳を完成させます。

## まとめ

Transformerの自己注意メカニズムは、文中の全ての単語が相互に影響し合うようにすることで、文脈を深く理解し、高精度な翻訳や文章生成を可能にします。このアプローチは、従来のRNNやLSTMよりも並列処理がしやすく、トレーニングの効率も高いため、NLPの多くのタスクで優れた性能を発揮します。