

2023.07.21 情報処理学会連続セミナー2023  
対話AI最前線：ChatGPTとその先にある可能性

# 大規模言語モデルを用いた対話システム

光田航（rinna株式会社）



# 光田 航

Koh Mitsuda

rinna株式会社

## [経歴]

- 2015年 東京工業大学大学院情報理工学研究科 修士課程修了
- 2015年～2023年 日本電信電話株式会社 研究員
- 2021年 筑波大学大学院システム情報工学研究群 博士（工学）
- 2023年 rinna株式会社 Applied Scientist

## [専門]

- 自然言語処理
- 対話システム（チャットボット）

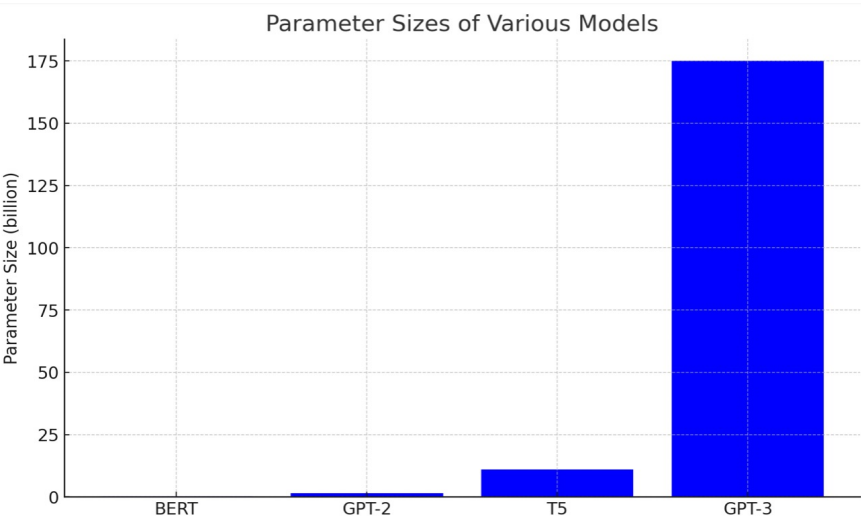
# 大規模言語モデルの基礎

# 大規模言語モデル

- 多量のテキストデータを用いて学習された, 多数のパラメタを持つモデル
  - 例. GPT-3: 500 billionトークン (5000億単語) で学習 (参考: Wikipediaの日本語記事は全部で約20億単語) , パラメタ数は1750億個
- 入力テキストの続きの予測 (次単語予測) に基づき多様なタスクを実施可能
  - パラメタ数のグラフはGPT-4のCode Interpreterを用いて作成 (プログラム生成)

GPT-3の学習データ (<https://arxiv.org/abs/2005.14165>)

Dataset	Quantity (tokens)	Weight in training mix	Epochs elapsed when training for 300B tokens
Common Crawl (filtered)	410 billion	60%	0.44
WebText2	19 billion	22%	2.9
Books1	12 billion	8%	1.9
Books2	55 billion	8%	0.43
Wikipedia	3 billion	3%	3.4



The screenshot shows the ChatGPT interface with a code interpreter. The user prompt asks for a bar chart of parameter sizes for BERT, GPT-2, T5, and GPT-3. The assistant's response includes a list of parameter sizes and a Python code snippet using matplotlib to generate the bar chart.

Model: Code Interpreter

BERT, GPT-2, T5, GPT-3のパラメタサイズを棒グラフにして、棒の色は全部青で、縦軸は線形スケールで、縦軸の目盛はbillionで

指定されたモデルのパラメタサイズは以下の通りです:

- BERT: 340 million
- GPT-2: 1.5 billion
- T5: 11 billion
- GPT-3: 175 billion

これらの値を棒グラフにプロットします。横軸にはモデルの名前、縦軸にはパラメタサイズ (billion)を表示します。また、棒の色は全て青にします。これらの設定をPythonのmatplotlibライブラリを用いて実装します。

```
python
import matplotlib.pyplot as plt

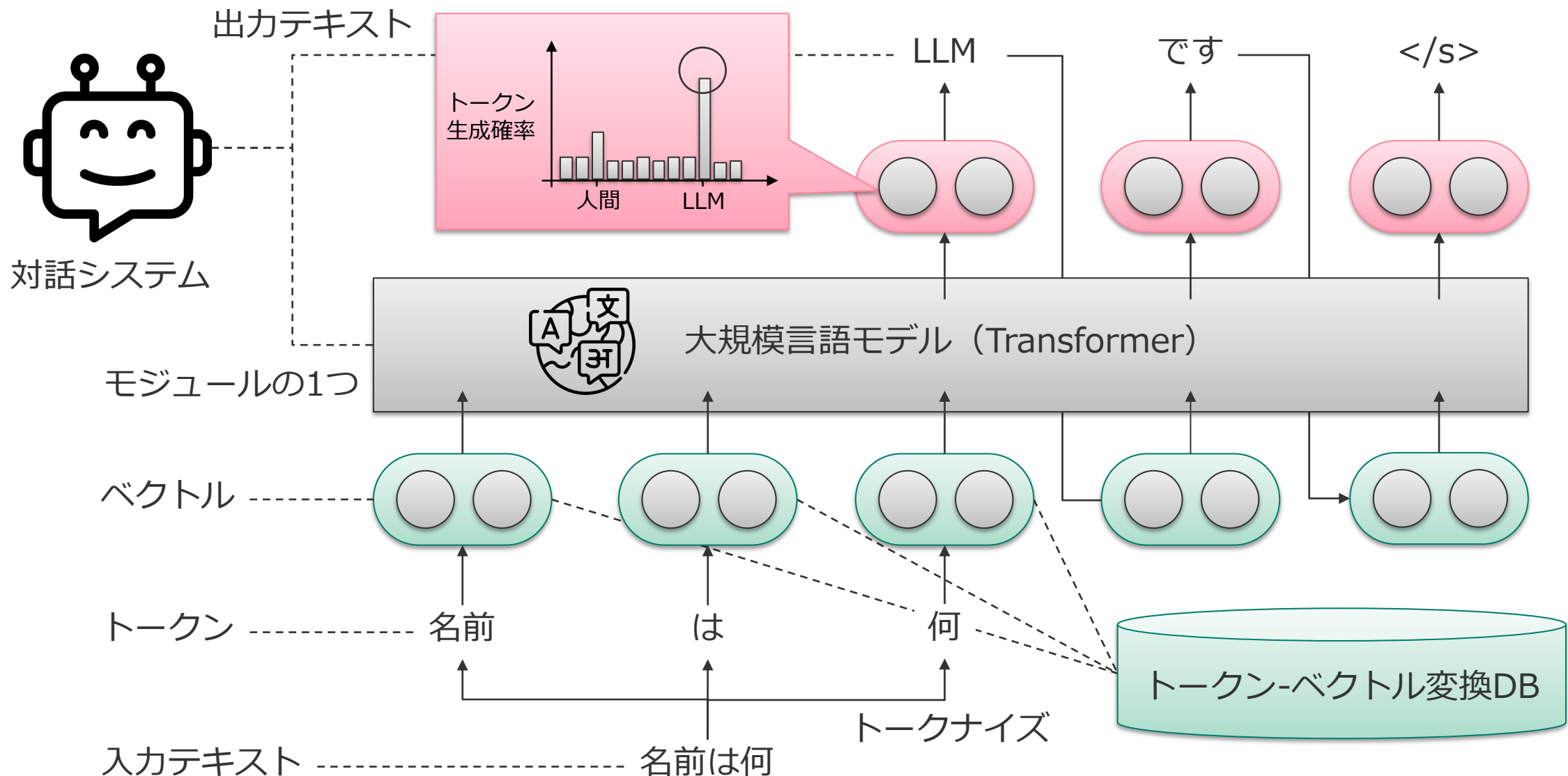
# model names and parameter sizes
models = ['BERT', 'GPT-2', 'T5', 'GPT-3']
param_sizes = [0.34, 1.5, 11, 175] # in billions

# create bar plot
plt.figure(figsize=(10, 6))
plt.bar(models, param_sizes, color='blue')

plt.ylabel('Parameter Size (billion)')
plt.title('Parameter Sizes of Various Models')

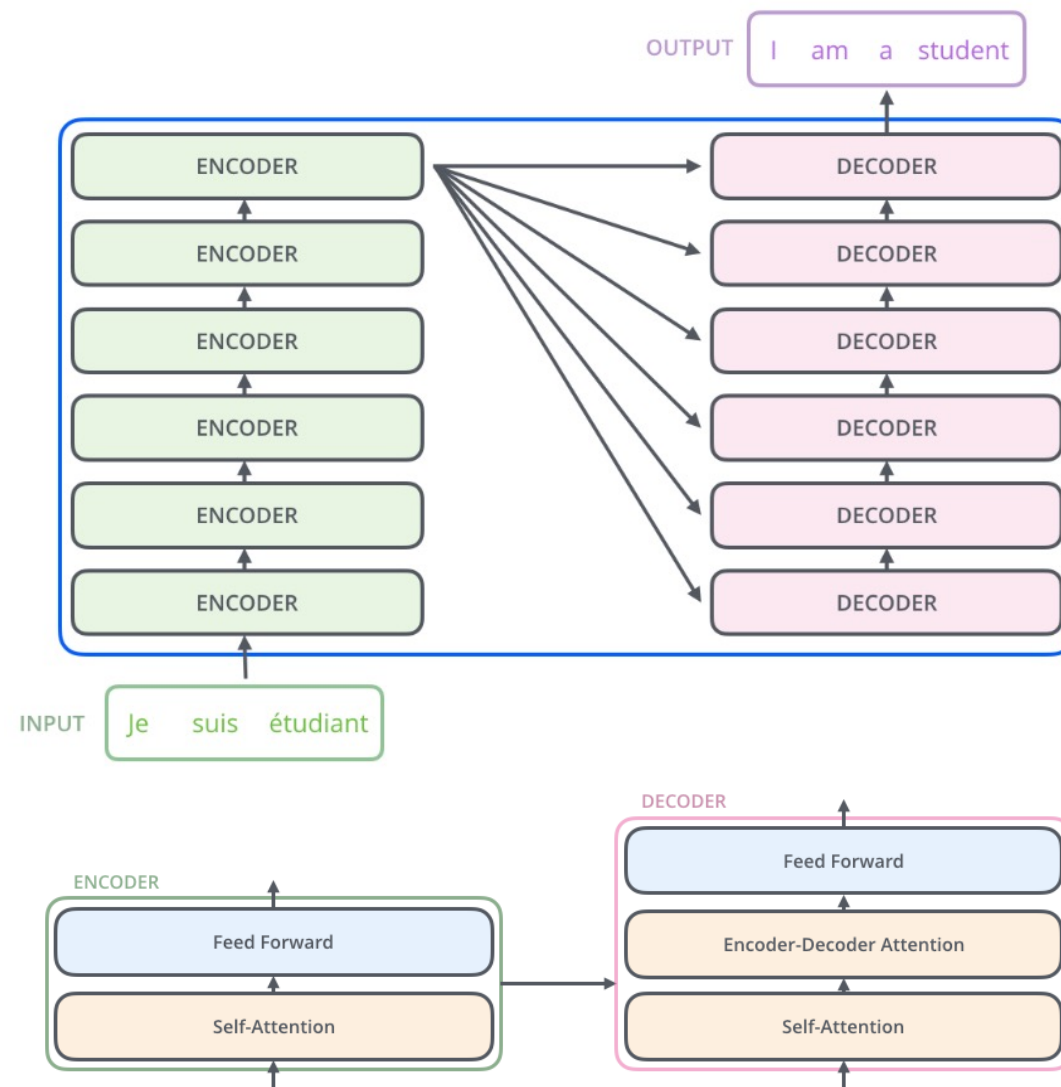
plt.show()
```

# 次単語予測



# Transformer

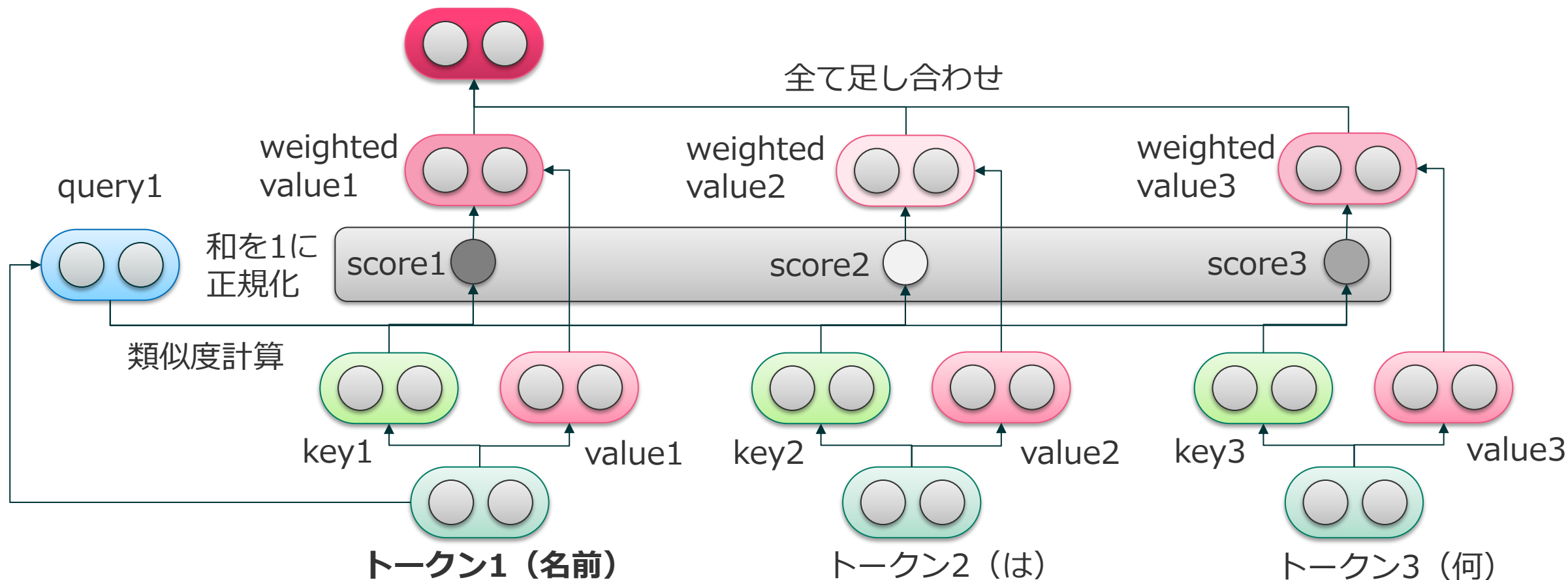
- Transformer-encoderとTransformer-decoderからなる深層学習モデル
- 機械翻訳の研究で提案され、自然言語処理を中心に多様なタスクに適用
  - 翻訳: フランス語テキスト → 英語テキスト
  - 対話: 過去文脈 → システムの応答
- encoderやdecoderを多層に積み重ねることでモデルのスケールが可能になり、性能が大きく向上
  - BERT: encoderを多層に積んだもの
  - GPT-3: decoderを多層に積んだもの
  - 近年の対話が可能で大規模言語モデルは基本的にdecoder-onlyのモデル
- 各encoderやdecoderの中でセルフアテンションという計算を実施



# セルフアテンション

- 入力や出力に含まれる全てのトークン間の関係を考慮するための機構
- 各トークン間の類似度計算をベクトルで表現したQueryとKey（辞書引き）で実現

**Self-Attention(トークン1)** ----- トークン2, 3（は, 何）との関係を考慮したベクトル

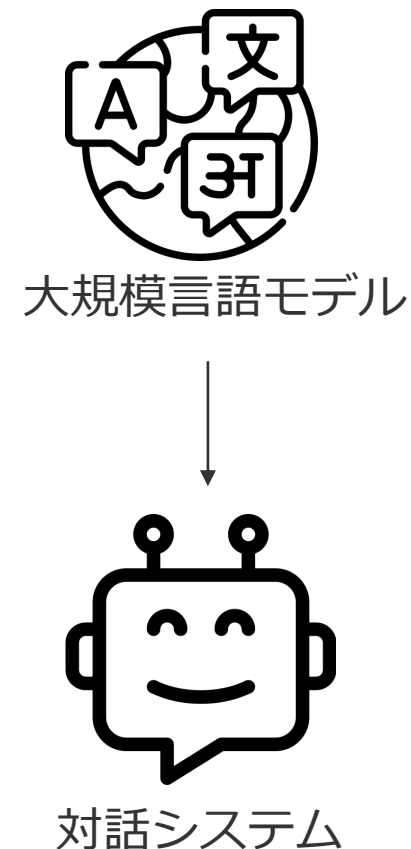


# 大規模言語モデルの技術的なポイント



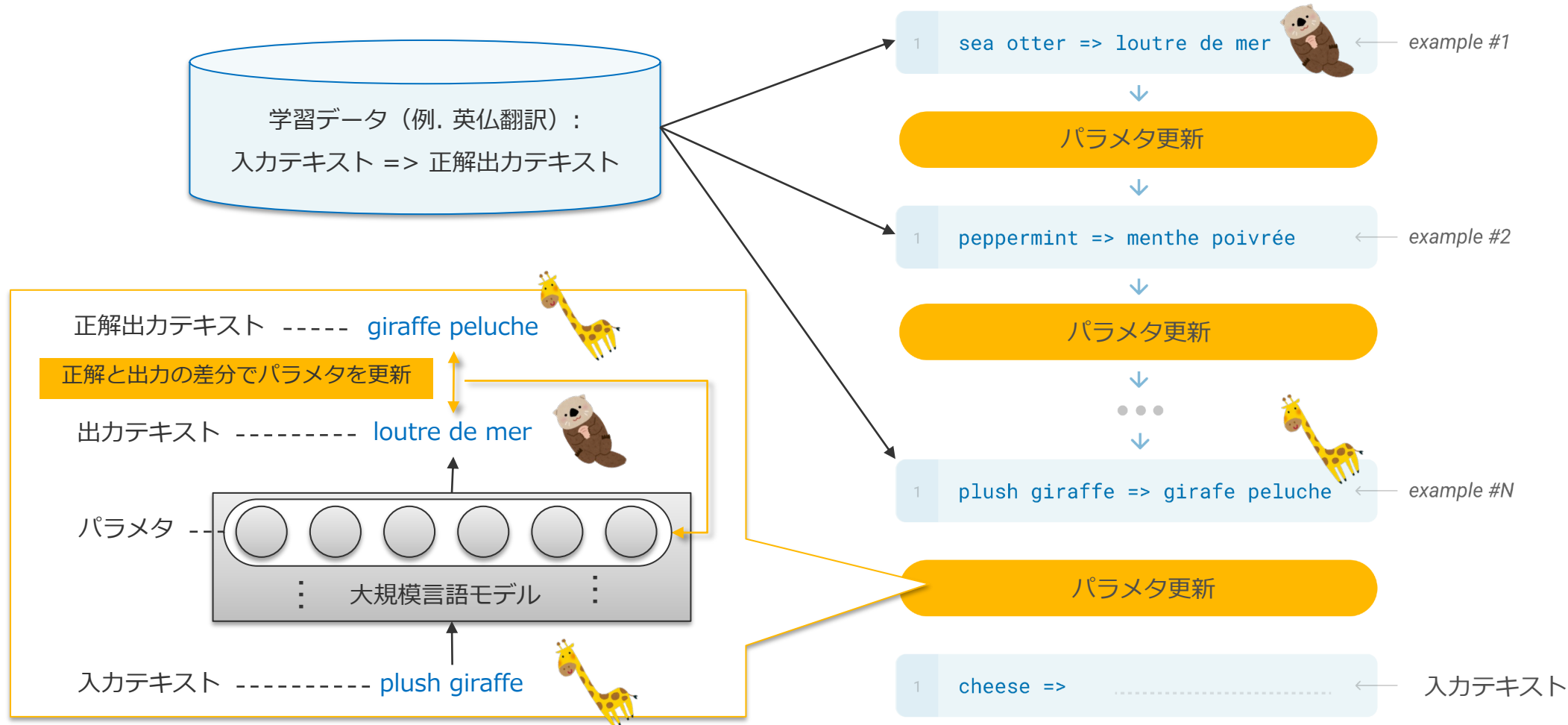
# 大規模言語モデルを用いた対話システムの実現

- 多量のテキストデータで学習（プレトレーニング）された大規模言語モデルは、そのままでは対話システムとしては使うことは難しい
  - 入力されたテキストの続きを出力するのみ
- 対話システムを実現させるために、対話できるようにしたり、個性を持たせたり、知識を入れ込んだりしたい
- 機械学習や自然言語処理の技術を適用することで実現可能
  - ファインチューニング
  - プロンプト（コンテキスト内学習）
  - Reinforcement Learning from Human Feedback (RLHF)
  - 外部知識（組織内の文書や個性等の設定）の参照



# ファインチューニング

- 解きたいタスクに合わせて学習データを用意し，学習データ中のサンプルを使ってプレトレーニングしたモデルのパラメタを更新（微調整）する手法
  - 英仏翻訳のペアと同様に，「文脈 => システムの応答」のペアを用意すれば対話が可能に



# プロンプト（コンテキスト内学習）

- プロンプト：「大規模言語モデルに与える入力テキスト」
  - 例. “次の英語をフランス語に翻訳して：  
cheese =>”
- GPT-3にて、ファインチューニング（モデルのパラメタ更新）を行わず、プロンプトのみで多様な言語処理タスクが高精度に解けることが発見
- ファインチューニング用のデータを用意せずとも、プロンプトを工夫するのみでも良い出力が得られるため注目
  - 入出力サンプルの指定: Zero-shot/One-shot/Few-shot
  - 思考仮定の出力: “Let’s think step by step”
- 柔軟に出力を制御可能
  - 対話の方向性を指定: 「有用な応答を返して」「共感して」
  - 個性を指定: 「ポジティブに」「〇〇のキャラ風に」
  - 外部知識の参照: 組織内の文書やキャラクタの設定等

## Zero-shot

The model predicts the answer given only a natural language description of the task. No gradient updates are performed.

```
1 Translate English to French: ← task description
2 cheese => ..... ← prompt
```

## One-shot

In addition to the task description, the model sees a single example of the task. No gradient updates are performed.

```
1 Translate English to French: ← task description
2 sea otter => loutre de mer ← example
3 cheese => ..... ← prompt
```

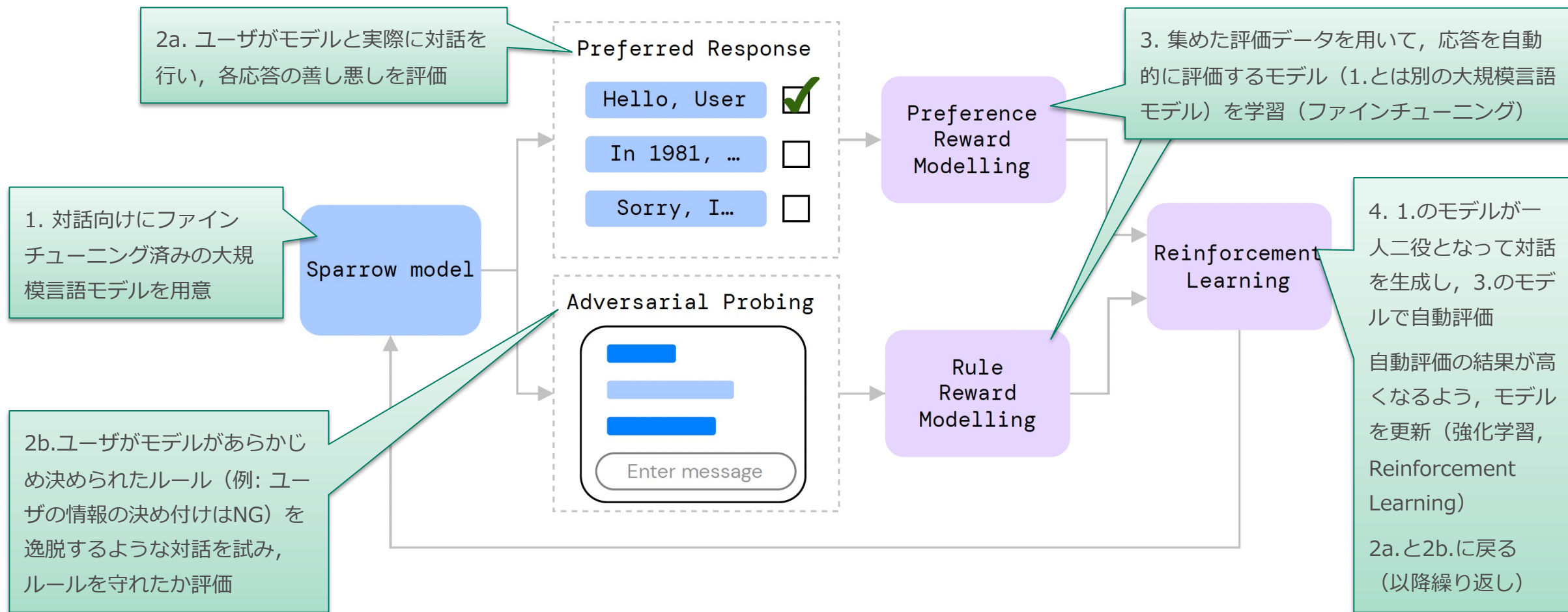
## Few-shot

In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.

```
1 Translate English to French: ← task description
2 sea otter => loutre de mer ← examples
3 peppermint => menthe poivrée ←
4 plush girafe => girafe peluche ←
5 cheese => ..... ← prompt
```

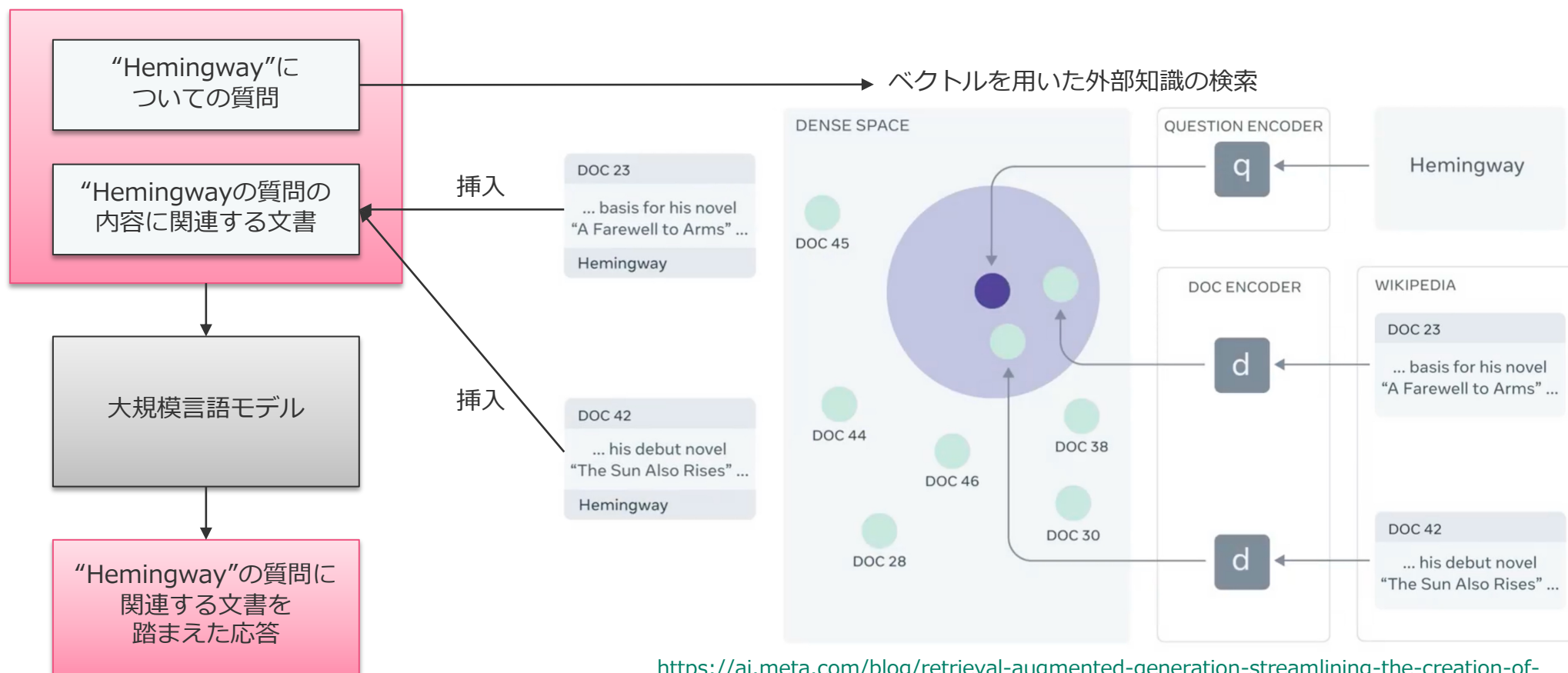
# Reinforcement Learning from Human Feedback (RLHF)

- ユーザにとって好ましい応答を返すよう，人手評価のデータを用いてモデルを調整
- 例. 情報検索を行いつつ対話を行うSparrow (Google DeepMind) におけるRLHF



# 外部知識（組織内の文書や個性等の設定）の参照

- 入力に関連の深いテキストを外部知識から検索し、プロンプトに挿入して生成（Retrieval Augmented Generation）
- 外部知識（例：Wikipediaの新しい記事，組織内の文書，キャラクタ設定）を考慮した生成が可能に  
プロンプト



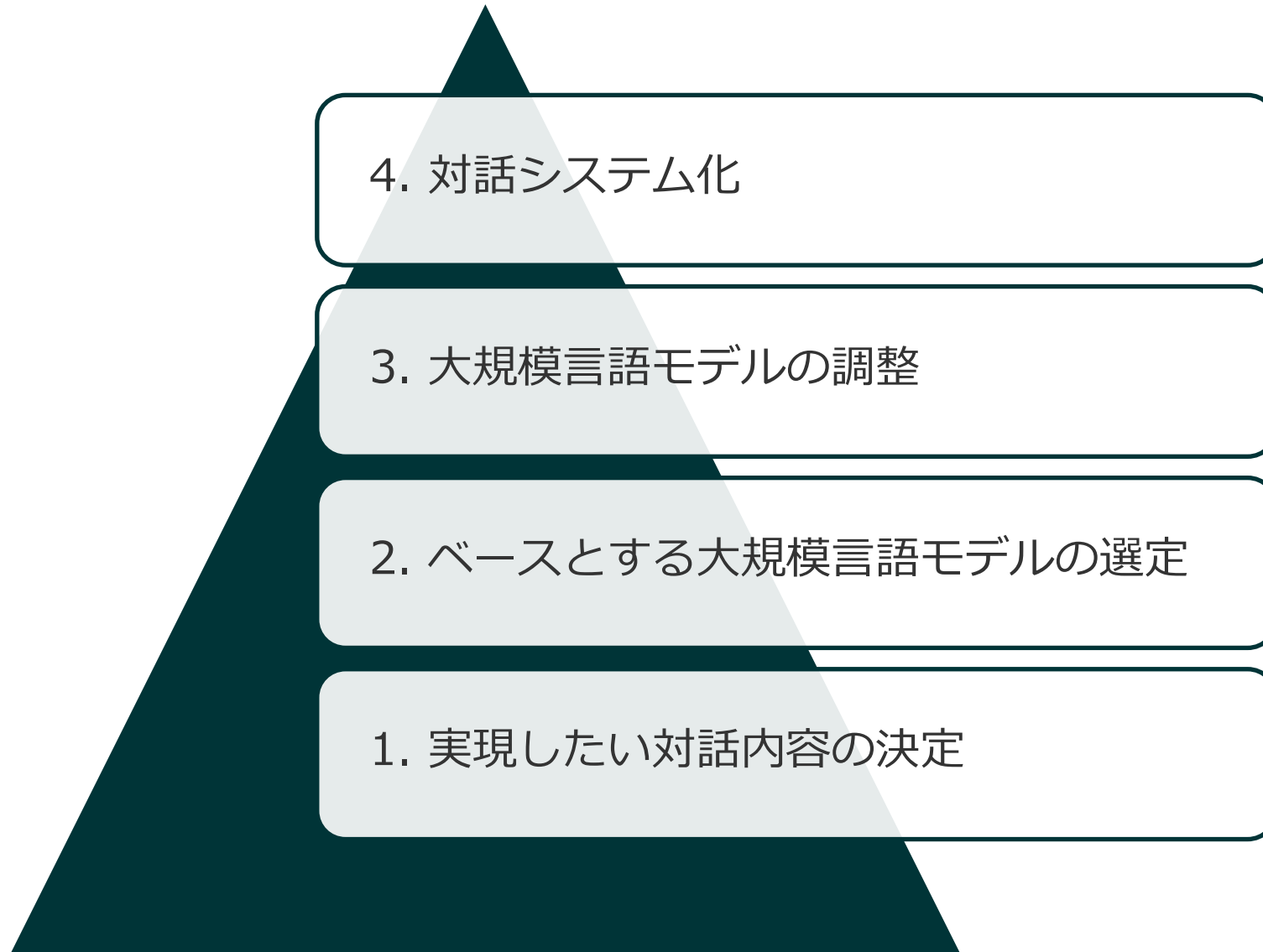
<https://ai.meta.com/blog/retrieval-augmented-generation-streamlining-the-creation-of-intelligent-natural-language-processing-models/>

# 大規模言語モデルの技術的なポイントのまとめ

- 必要なリソースや難易度が異なるため、1番から順に試していくことをおすすめ

技術的なポイント	必要なリソース	調整の難易度	試す順番
ファインチューニング	<ul style="list-style-type: none"><li>モデルのパラメタデータ</li><li>ファインチューニング用データ</li><li>ファインチューニング用プログラム</li><li>GPU</li></ul>	NORMAL 😊	3番
プロンプト (コンテキスト内学習)	なし	VERY EASY 😊	1番
RLHF	<ul style="list-style-type: none"><li>モデルのパラメタデータ</li><li>RLHF用データ</li><li>RLHF用プログラム</li><li>GPU</li></ul>	VERY HARD 😞	4番
外部知識の参照	<ul style="list-style-type: none"><li>外部知識のテキストデータ</li><li>知識検索プログラム</li></ul>	EASY 😊	2番

# 大規模言語モデルに関するツール





## ツール: 大規模言語モデルの選定

- 日本語を対象とした対話を行う場合, 日本語が得意な大規模言語モデルの利用を推奨
  - 英語を対象にしたモデルでは日本語の学習データ量が少なく, 十分な性能が出づらい
- 日本語が得意な大規模言語モデル

ライセンス	モデルの例 (生成型)	特徴 ☺ / ☹	
非オープンソース	<ul style="list-style-type: none"><li>OpenAIのChatGPT, GPT-4</li><li>GoogleのBard 等</li></ul>	<ul style="list-style-type: none"><li>利用ハードル低↓ (チャット画面やAPIを利用)</li></ul>	<ul style="list-style-type: none"><li>カスタマイズ性低↓ (API経由のみ)</li><li>コスト高↑ (API呼び出し数で課金)</li></ul>
オープンソース	<ul style="list-style-type: none"><li>rinnaのjapanese-gpt-neox-3.6b</li><li>CyberAgentのOpenCALM-7B 等</li></ul>	<ul style="list-style-type: none"><li>カスタマイズ性高↑ (柔軟なモデル改善)</li><li>コスト低↓ (GPU利用コストのみ)</li></ul>	<ul style="list-style-type: none"><li>利用ハードル高↑ (自然言語処理や機械学習等の知識が必要)</li></ul>

# ツール: 大規模言語モデルの調整と対話システム化

- 学習データ, 外部知識
  - ウィキペディア
    - <https://www.tensorflow.org/datasets/catalog/wiki40b>
    - 前処理済みのものがおすすめ
  - 日本語対話コーパスのまとめ
    - [https://individuality.jp/dialogue\\_corpus.html](https://individuality.jp/dialogue_corpus.html)
  - オープンソースの日本語LLMまとめ
    - <https://github.com/llm-jp/awesome-japanese-llm>
    - 学習に利用されているデータを参照
  - クリーニング方法
    - <https://arxiv.org/abs/2302.13971>
- 学習や推論のフレームワーク
  - Hugging Face
    - <https://huggingface.co>
    - 大規模言語モデル, データセット, 学習や推論を行うためのプログラムがまとまったもの
  - 大規模言語モデルを各種ツールと組み合わせるための選択肢
    - LangChain: <https://langchain.com>
    - LlamaIndex: <https://www.llamaindex.ai>
    - (Python等で自分で実装)

# ツール: 大規模言語モデルの調整と対話システム化 (つづき)

- 学習や推論を効率化／高速化するライブラリ
  - ファインチューニング
    - Parameter-Efficient Fine-Tuning (PEFT) : <https://github.com/huggingface/peft>
  - 推論
    - CPU (非GPU) 推論: <https://github.com/ggerganov/llama.cpp>
    - 効率化手法の一覧・比較の記事: <https://zenn.dev/rinna/articles/5fd4f3cc12f7c5>
  - モデルの軽量化
    - 量子化: <https://huggingface.co/blog/4bit-transformers-bitsandbytes>
- テキストからベクトルを作成するためのAPIやモデル
  - ベクトル化
    - OpenAI API (Embeddings) : <https://platform.openai.com/docs/guides/embeddings>
    - Sentence-Transformers: <https://www.sbert.net>
  - 高速ベクトル検索
    - FAISS: <https://github.com/facebookresearch/faiss>

## 実装デモ

# デモ：大規模言語モデルを用いた対話システム

<https://colab.research.google.com/drive/1WKAFaeXPSDOJ6ujwOwj55QQnUr2YsNFU?usp=sharing>

Google Collaboratoryという，ブラウザ上でPythonを実行可能な環境を利用

（環境設定が不要で，料金なしでGPUアクセスも可能）

# まとめ

大規模言語モデルの技術や，大規模言語モデルを用いて対話システムを実現する方法について解説

- 大規模言語モデルの基礎
  - 次単語予測に基づいて多様なタスクを実施
  - Transformer, セルフアテンション
- 大規模言語モデルの技術的なポイント
  - ファインチューニング, プロンプト, RLHF, 外部知識の参照
- 大規模言語モデルに関するツール
  - 大規模言語モデルの選択肢
  - 大規模言語モデルを調整する手段
  - 対話システムとして実装する手段
- 実装デモ
  - 対話向けのファインチューニング

**今後の課題:** ChatGPTのリリース記事より

## Limitations

- 一見妥当だが誤ったことや非常識な応答を出力
- 入力の表現に敏感
- 応答が冗長であったり，特定のフレーズを多用したり
- ユーザからの曖昧な入力に対して明確化せず応答
- 有害な入力への反応や偏見を含む応答

<https://openai.com/blog/chatgpt/>

大規模言語モデルを用いた対話システムの性能はまだ100点満点とは言えないかもしれないが，技術発達速度は非常に早く，今後課題の解決とともに，より多様な分野へと応用されていくことが期待