対話システム，chatBOT

最新技術を知る → 各自が創造力，問題解決力を発揮する

URL: https://www.kkaneko.jp/a/2023.html

金子邦彦

BY NC SA

chatBOT

- chatBOT は，**人工知能**を利用したコンピュータプログラム
- **人間からの質問**に対して，**自動的に応答**を行う
- chat Bot のさまざまな用途

  商品の販売や予約，問い合わせ，質問応用など
- chat Bot の特徴

1. **既存の大量の文書**と，**今までの対話記録（コンテキスト）**から**学習**
2. **ユーザからのフィードバック**（ユーザが，会話の途中で与える文章）により，chatBOT はさらに学習．chatBOT の**回答が変化**
3. chatBOT に不正確な質問や指示を与えると，chatBOT が適切な回答が**できなくなる**場合がある

chat BOT の仕組み

既存の大量の人間の文章

大規模言語モデル

文章の続きの生成 (Text Completion) の能力を持つ AI

今までの**対話記録**（コンテキスト）

Human: Hello! What can you do?
AI: As an AI assistant. I can answer questions and chat with you.
Human: コンピューターについて教えてください

AI からの回答

AI: 私のコンピュータは，可能な限り賢くなるように設計された汎用AIシステムです

問答のたびに更新

chatBOT の利用例

- 調べごと
- 文章の仕上げ．誤字などのチェック
- 見落としが無いかのチェック
- 内容の充実などの提案

chatBOT の実応用

- **販売用の chatBOT**

  購入希望を伝えると，支払方法，合わせて購入した方がよい商品などを案内

- **問い合わせを受け付ける chatBOT**

  商品の在庫状況，店舗の営業時間などを案内

- **ユーザサポートを行う chatBOT**

  困ったことへの対処法やアドバイスを行う

- **人間と人間のコミュニケーションへの介在**

  専門用語を分かりやすく説明する chatBOT，

  翻訳システムとの連携

- chatBOT の実応用では，**用途に応じたデータベース**が必要

  商品情報，今の価格，今の在庫

- ユーザからの質問や，フィードバックの収集．最新のデータベースの維持も重要.

```
C:¥Users¥user¥FlexGen>python c.py --model facebook/opt-6.7b
Initialize...
A chat between a curious human and a knowledgeable artificial intelligence assistant.
Human: Hello! What can you do?
Assistant: As an AI assistant, I can answer questions and chat with you.
Human: What is the name of the tallest mountain in the world?
Assistant: Everest.
Human: エベレストの高さを教えてください
How high is Everest?
Assistant: It's more than 30,000 feet high.
アシスタント：標高は3万フィート以上です。

Human:
```

FlexGen を用いた chatBOT
エベレストの高さを回答している

```
C:¥Users¥user¥FlexGen>python c.py --model facebook/opt-6.7b
Initialize...
A chat between a curious human and a knowledgeable artificial intelligence assistant.
Human: Hello! What can you do?
Assistant: As an AI assistant, I can answer questions and chat with you.
Human: コンピューターについて教えてください
Tell us about your computer
Assistant: My computer is a general purpose AI system designed to be as smart as possible.
アシスタント：私のコンピュータは、可能な限り賢くなるように設計された汎用AIシステムです。

Human: チャットボットについて教えてください
Tell us about chatbots
Assistant: Chatbots are AI systems that are designed to perform simple, predictable tasks for you.
アシスタント：チャットボットは、単純で、予測可能な作業を代行するように設計されたAIシステムです。

Human:
```

FlexGen を用いた chatBOT
利用者からの質問により，コンピュータ，チャットボットについて説明している

関連ページ

- Meta の言語モデルと日本語で対話できる chatBOT プログラム（FlexGen, DeepL, Python を使用）　（Windows 上）

https://www.kkaneko.jp/ai/repot/deeplandflexgen.html

```python
"""Run a chatBOT with FlexGen and OPT models."""
# usage: python c.py --model facebook/opt-6.7b
import argparse
import sys
import deepl
import win32com.client

auth_key = "f07f7822-073f-bafa-49a7-e91d9efeb78f:fx"
translator = deepl.Translator(auth_key)
speech = win32com.client.Dispatch("Sapi.SpVoice")

from transformers import AutoTokenizer
from flexgen.flex_opt import (Policy, OptLM, ExecutionEnv, CompressionConfig,
    str2bool)


def main(args):
    # Initialize environment
    env = ExecutionEnv.create(args.offload_dir)

    # Offloading policy
    policy = Policy(1, 1,
                    args.percent[0], args.percent[1],
                    args.percent[2], args.percent[3],
                    args.percent[4], args.percent[5],
                    overlap=True, sep_layer=True, pin_weight=args.pin_weight,
                    cpu_cache_compute=False, attn_sparsity=1.0,
                    compress_weight=args.compress_weight,
                    comp_weight_config=CompressionConfig(
                        num_bits=4, group_size=64,
                        group_dim=0, symmetric=False),
                    compress_cache=args.compress_cache,
                    comp_cache_config=CompressionConfig(
                        num_bits=4, group_size=64,
                        group_dim=2, symmetric=False))

    # Model
    print("Initialize...")
    tokenizer = AutoTokenizer.from_pretrained("facebook/opt-30b", padding_side="left")
    tokenizer.add_bos_token = False
    stop = tokenizer("¥n").input_ids[0]

    model = OptLM(args.model, env, args.path, policy)

    context = (
        "A chat between a curious human and a knowledgeable artificial intelligence assistant.¥n"
        "Human: Hello! What can you do?¥n"
        "Assistant: As an AI assistant, I can answer questions and chat with you.¥n"
    )

    # Chat
    print(context, end="")
    while True:
        inp = input("Human: ")
        if not inp:
            print("exit...")
            break

        speech.Speak(inp)
        result = translator.translate_text(inp, target_lang="EN-US")
        print(result)

        context += "Human: " + result.text + "¥n"
        inputs = tokenizer([context])
        output_ids = model.generate(
            inputs.input_ids,
            do_sample=True,
            temperature=0.7,
            max_new_tokens=96,
            stop=stop)
        outputs = tokenizer.batch_decode(output_ids, skip_special_tokens=True)[0]
        try:
            index = outputs.index("¥n", len(context))
        except ValueError:
            outputs += "¥n"
            index = outputs.index("¥n", len(context))

        outputs = outputs[:index + 1]
        print(outputs[len(context):], end="")
        translated = translator.translate_text(outputs[len(context):], target_lang="JA")
        print(translated)
        speech.Speak(translated)
        context = outputs

    # TODO: optimize the performance by reusing context cache and reducing redundant computation.

    # Shutdown
    env.close_copy_threads()


if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument("--model", type=str, default="facebook/opt-6.7b",
        help="The model name.")
    parser.add_argument("--path", type=str, default="~/opt_weights",
        help="The path to the model weights. If there are no cached weights, "
             "FlexGen will automatically download them from HuggingFace.")
    parser.add_argument("--offload-dir", type=str, default="~/flexgen_offload_dir",
        help="The directory to offload tensors. ")
    parser.add_argument("--percent", nargs="+", type=int,
        default=[100, 0, 100, 0, 100, 0],
        help="Six numbers. They are "
         "the percentage of weight on GPU, "
         "the percentage of weight on CPU, "
         "the percentage of attention cache on GPU, "
         "the percentage of attention cache on CPU, "
         "the percentage of activations on GPU, "
         "the percentage of activations on CPU")
    parser.add_argument("--pin-weight", type=str2bool, nargs="?",
        const=True, default=True)
    parser.add_argument("--compress-weight", action="store_true",
        help="Whether to compress weight.")
    parser.add_argument("--compress-cache", action="store_true",
        help="Whether to compress cache.")
    args = parser.parse_args()

    assert len(args.percent) == 6

    main(args)
```

FlexGen を用いた chatBOT のプログラム
（FlexGen の公式ページで公開されているプログラムを利用）