一般公開用

ChatGPTを業務に組み込むためのハンズオン

2023/06/26　　デジタル庁　Fact&Data Unit　大杉直也

↑マイナンバー交付数のダッシュボードを作っているところです

講義の狙い

アイデアの価値は「検証」してみないとわからない

「Microsoft でテストされたアイデアのうち、改善を示すメトリクスを実際に改善できたのは3分の1にすぎない」 (Microsoft社 元Vice President)

「もしあなたが実験主導のチームにいるなら、70%の仕事が捨てられることに慣れてください。それに応じてプロセスを構築しましょう」(Slack社 Director)

A/Bテスト実践ガイド p14より

一方で

「アイデアの価値を見積もることは難しい。このケースでは、年間１億ドルの価値ある単純な変更が何か月も遅れていた。」(同著 p5より) こともあります

午前中のアイデアソンで出たアイデアはちゃんと検証するまで価値があるかは不明です。

ちゃんと検証するためには、ちゃんとアイデアを形にする必要があり、そのためには「どう作るか」「そもそもできそうか」を試すことが重要です

要するに、作って終わりではないです。が、作らないと話にならないので、さっさと作りましょう

このハンズオンではGPT のAPIを実際に叩いてみて、自身のアイデアを形にするための手ごたえをつかむ時間を設けます

この時間だけで形にしきることは難しいと思いますが、明日以降に**独力で検証を進めるための方法論**を身につけられれば幸いです

（明日以降のデジ庁の何らかの支援が必要な場合は追って相談とさせてください）

A/Bテスト実践ガイド 真のデータドリブンへ至る信用できる実験とは https://www.kadokawa.co.jp/product/302101000901/

ChatGPTとGPT APIの関係性

GPT3.5 やGPT4といった高品質なLLMを「どのように利用するか」でサービスが2系統ある
ChatGPTの方が制約が大きく、GPT APIの方ができることが多い
ただし、GPT APIによるChatGPTの完全再現は困難

このスライドの目的

GPT APIを用いた業務改善に重要な「試行錯誤」を独力でできる状態を目指します

ChatGPT的な処理を、既存業務フローに組み込んだり、独自インターフェース上で実現したり、業務DBと接続したり、の設計を自在にできるようになることが目標です

例1: 電話での注文依頼を文字起こしたものから、注文画面の入力フォームに入力してくれる
例2: 職員の使っているチャットツールから、チャット形式でマニュアルを検索してくれる
例3: 複雑な独自発注ルールを読み込ませ、そのルールについて質問したら答えるチャットボットを発注画面に組み込む
例4: 諸外国の法令検索結果をリアルタイムで和訳・関連法令と比較してくれる

このスライドの目的

GPT APIを用いた業務改善に重要な「試行錯誤」を独力でできる状態を目指します

ChatGPT的な処理を、既存業務フローに組み込んだり、独自インターフェース上で実現したり、業務DBと接続したり、の設計を自在にできるようになることが目標です

誰に:

ChatGPTで何ができそうかは既知だが、実際にシステムに組み込むイメージを持っていない人向けに

何を:

ChatGPT的なAIを業務に組み込むための要件定義に必要な技能の取得

なぜ:

ChatGPTの使い方解説系は巷に溢れているがGPT APIの非エンジニア向けの実践的な解説がまだまだ少ないから

どのように:

講義+実際にGPT APIの挙動を見ながら

※ChatGPTでできそうなことは既知とします。「実用」に注力するため歴史の話などもしません
※このAPI を使い始めるまでの技術的な手順もググると大量に出てくるため割愛します

ChatGPT風アプリケーションのデモ

「これはペンです。」と返答
利用料金:0.000074 USD

※当日は実際に動く検証環境をもとにライブデモを行いました

デジタル庁内での利用環境整備

庁内一般環境

Global ProtectのIP制限
キー①による認証

クラウド環境 (Azure)

API Management

キー②による認証

OpenAI Service (ChatGPT系)

リクエスト・レスポンスのロギング
GPTへの入出力文字列も保存

Blob Storage

要機密情報が送られていないかを確認可能な状態
最悪のケースではネットワークのログと突合し不正利用の個人を特定可能

非エンジニアでも検証・改修が可能なHTMLテンプレートを用意

- デフォルトではChatGPT風アプリケーションとして動く
- LLMのモデルをボタン一つで切り替え
- 1リクエストごとの利用料金を表示することでコストを意識
- プロンプトをHTMLに埋め込むことで多様なユースケースに対応するAIに
    - プロンプトの埋め込みは非エンジニアがメモ帳で作業可能な難易度
- 簡易的な検証・開発環境として利用可能
    - 利用しなくても良い

コスト

token 数について

tokenとは入出力の文章を分割したもの

英語ではスペースで分割できるのでわかりやすい。日本語では文字数より細かくなることもよくある

token数が多いと(1)レスポンスタイム悪化(2)API利用料増(3)回答精度低下傾向などがあり、**日本語は不利**

入出力のtoken数最大値には制限があり、**GPT3.5では4k, 4では8kまたは 32k**までとなっている

長大なルールをひとつのプロンプトに入れることはできない

コストはtoken数に比例して発生

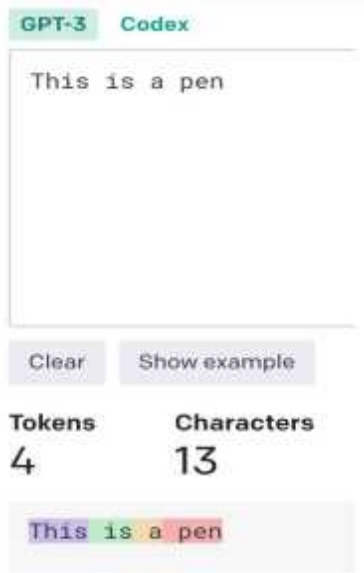
3.5の場合: 入出力のtoken 1000個 あたり 0.002 USD

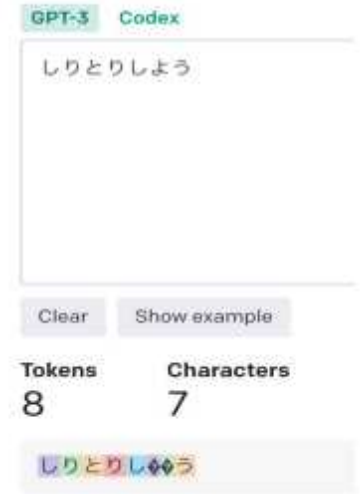
4 (8k)の場合: token 1000個あたり 入力は 0.03USD、出力は0.06USD。32kはこの倍

※ハンズオンのコストはデジタル庁 AI班もち

6/13のアプデで新しいモデルが発表

6/13のアプデで変更。安くなる。
Azureの対応時期は未定





画像は https://platform.openai.com/tokenizer より作成

質疑応答 (1/4)

※当日はここで休憩をいれました

特に意識してほしいこと

ブラックボックスを相手にする

入力と出力の対から性質を判断するしかない

いろんな入力パターンを試してみることが重要

不確実性が極めて高い。入力文字列が少し変わるだけで出力文字列のニュアンスが変わってしまうこともある

完全制御は難しいので、それが許容できるユースケースを探し出す

「どのくらい実現できそうか」に対して、全くできない〜できる時がある〜だいたいできる〜ほぼできる、と濃淡があることが特徴の一つ

巷で言及される例の中には「できる時がある」の良い例だけ切り出される場合もある。自分で試してみることが重要

参考: GPT-4 System Card (https://cdn.openai.com/papers/gpt-4-system-card.pdf)

今回のハンズオンで学ぶ工程

この講義では下記工程の「２」の箇所の方法論を学ぶ

1. テキスト生成AIがこの業務効率改善に役立ちそう、というアイデアをもつ
2. テキスト生成AIへの入出力を具体的に設計する
3. 生成AIのコンポーネントを含めたシステムアーキテクチャを検討する
4. 実際に開発する

補足１

基本的な開発プロセスに対応させた説明

要求分析→要件定義→基本設計→詳細設計→コーディング

の流れの中で、「どのくらい実現できそうか」を**要求分析**段階で推定することで

（１）どこで（２）だれに（３）どのように使わせるか、の解像度を大幅に上げられる

※「どのくらい実現できそうか」の濃淡によって成果物が変わってくる可能性がある。利用者は一部の職員か、国民全般か、など

要求分析段階で解像度を上げる重要性については、例えば、AIプロダクト品質保証ガイドライン Guideline for Quality Assurance of AI-based products and services (https://www.qa4ai.jp/QA4AI.Guideline.202207.pdf) を参考

補足２

従来のAI開発プロセスとの「量的」な違い

入力文字列を工夫するだけで多彩な挙動を行うため、

試行錯誤が誰でも迅速に行える→**試行錯誤の回数をプロジェクト初期に大幅に増やせる**

データを用意して機械学習させて～という流れが不要になった ※

※fine tuning とよばれる過程がGPTにも存在するが、(1)そもそも機能が制限 (2)コスト増(3)モデルが不安定になる危険性増大、などの副作用も大きく、まずはfine tuningなしでできるかの検討を推奨

Brown, Tom, et al. "Language models are few-shot learners." *Advances in neural information processing systems* 33 (2020): 1877-1901.

技術概要

GPTモデル (Chat) のAPIとは

OpenAIの公式ドキュメントが読みやすいため、こちらに準拠して解説

Azure版もほとんど同じだが、細かく違う点に注意

インターネットを介してAzure上のGPTモデルを利用できる

ChatGPTで使っているものと同じモデルとされている

APIキー (シークレット)とよばれるパスワードのようなものを使って認証

決して流出させてはならない

入出力の文字の長さ（厳密にはtoken数）に比例する従量課金

https://platform.openai.com/docs/api-reference/chat/create

GPTモデル (Chat) のAPIで主に意識するパラメータ

model

どのモデルを使うか。GPT3.5や後発のGPT4があり、4の方が賢いが重くて高い (15~30倍)

そのため、まずは3.5で十分そうかを検討することを推奨

messages

GPT APIへの入力文。プロンプトとも呼ばれ、これの書き方にコツがいる

プロンプトエンジニアリングと呼ばれる分野が誕生したほど

ChatGPTでの書き方と差異がある。以降、解説

https://platform.openai.com/docs/api-reference/chat/create

GPTモデル (Chat) のAPIのmessages を構成するもの

1. 「このGPTは何をするものか」を指定する **system**
2. 「GPTへの入力」を指定する **user**
3. 「GPTへの出力」を指定する **assistant**

https://platform.openai.com/docs/api-reference/chat/create

一番シンプルなケース

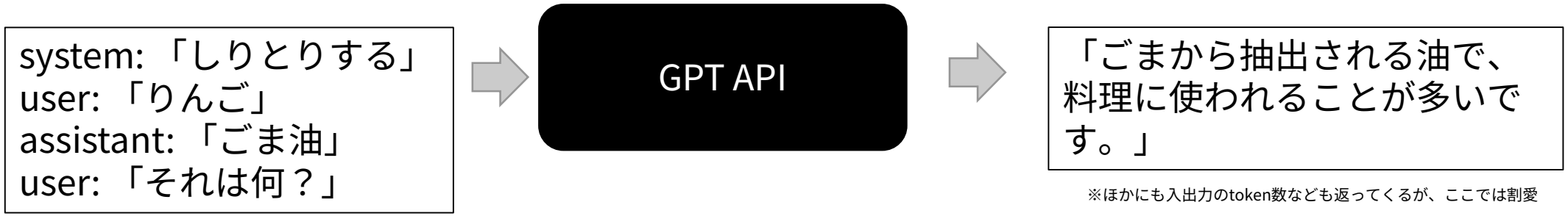
※ほかにも入出力のtoken数なども返ってくるが、ここでは割愛

https://platform.openai.com/docs/api-reference/chat/create

System prompt のLIVEデモ

※当日は実際に動く検証環境をもとにライブデモを行いました

会話履歴をもつケース (ChatGPT的)



※ほかにも入出力のtoken数なども返ってくるが、ここでは割愛

GPTのAPIは状態をもたないため、会話履歴を考慮した応答を実現するために、過去の会話履歴を入力する
APIの入出力の最大token数に制限があるため、保持できる会話履歴には限界がある
現在、GPTでは32kが最大token数。技術進歩により将来的には大幅に緩和されるかも？

Token数緩和のための研究として、例えば、Yu, Lili, et al. "Megabyte: Predicting million-byte sequences with multiscale transformers." *arXiv preprint arXiv:2305.07185* (2023).

しりとりのLIVEデモ（会話履歴についての解説）

「しりとり」を題材に過去の会話履歴を毎回GPT APIに送信して
いることを示しました

※当日は実際に動く検証環境をもとにライブデモを行いました

messages の解説

GPTの過去会話履歴のassistantを直接書き換えることが可能！

GPTの出力キャラクタ（口調など）や出力フォーマットを直示できる

MSのドキュメントではこれをfew shot learningと呼んでいる（参考）

理屈に合わないassistantを指定すると出力が不安定になり、最悪エラーが出るので注意

ルールを明示的に指定できる場合はsystemやuserでルールを指定した方が良い

例示から背後にある一般的なルールを学習するのは難しい課題

学習させているのではなく会話の流れを指定するだけなので、例えば user:「それは何？」で直前に指定したassistantの内容が返ってくる（意図した挙動ではないはず）

直示の具体的な例としてhttps://qiita.com/naosugi1987/items/059fb69a15a3dc06f33d　や　https://qiita.com/naosugi1987/items/1e5349424e4ed878c84d

「それは何？」による脆弱性については、例えばこのゲーム(https://prptinj.net/)　で体験可能

assistant 指定による口調調整とフォーマット指定のデモ

system: 「しりとりする」
user: 「りんご」
assistant: 「ごま油」
user: 「それは何？」

←この会話のassistant の「ごま油」を「ごま」に書き換えると、
「ごま」の解説が返ってくることを確かめました

※当日は実際に動く検証環境をもとにライブデモを行いました

system と user の間の自由度について

あるドキュメントAを要約させたい場合

system: 「以下についての質問を受け付ける。{ドキュメントAの内容}」
user: 「要約して」

system: 「与えられた入力を要約する」
user: 「{ドキュメントAの内容}」

のどちらの書き方もできる。プログラムとしてのあるべきだったら

「Aについて答えてくれるくん」なら左で、「文章要約してくれるくん」なら右

ただし、要約の品質にどのくらいの違いが出るかは未知

質疑応答 (2/4)

※当日はここで休憩を入れました

Tips (まだまだ発展途上です。絶対の正解ではない)

Tips (発展途上)

プロンプトの書き方のコツ

できる限りコンテキストを明確にして書くこと

Good: 候補を5つ挙げる。Bad: 候補をできる限り挙げる

Good: step by step で考える

相手はSFに出てくるロボットだと思って丁寧な会話(?)をする。ある程度の慣れが必要

GPTの理解度(?)を確認しながら進める

例: 法令文章をsystemに書き込んだ場合、その文章から想定QAや理解度確認テストを作らせ、それを専門家が確認する

最初はマニュアルを読むより、まず自分でやってみて感覚をつかみことを推奨

参考: **Prompt Engineering Guide** (https://www.promptingguide.ai/jp)

プロンプト エンジニアリングの手法 (https://learn.microsoft.com/ja-jp/azure/cognitive-services/openai/concepts/advanced-prompt-engineering)

候補だし & step by step のデモ

以下の２つのデモを行いました
1.「候補だし」をGPTで行わせるさいに「できるだけ」という指示(prompt)だと出力された候補が10個だったが、「20個」と指定することで20個（10個より多い）候補が出力されることの確認
2.「2025を7で割った余りは？」と聞くと誤った回答が返ってくるが、promptに「think it step by step」とつけることで正しい回答が返ってくることの確認

※当日は実際に動く検証環境をもとにライブデモを行いました

Tips (発展途上)

AI班で作成したサンプルの紹介

逐条解説より個人番号について答えるくん

プロンプトの工夫: 逐条解説を要約したもの(次ページで詳細)

入力例: 個人番号利用事務と個人番号関係事務の違いは？（〜第N条によると、の形式で回答）

課題: token数が多い。GPT4を利用しないと性能があやしいため利用料が高く反応速度も遅い

手続きをラベリングするくん

プロンプトの工夫: 出力フォーマットを指定。ラベル付けそのものではなく関連度を出力させる

入力: 手続きに関する文章

課題: 「ご不幸」といったハイコンテキストなラベルはそのまま使うと性能が出ない。ラベルの定義などを別途ちゃんと与えた方がよさそう

手続きを箇条書きで作成するくん

プロンプトの工夫: 出力フォーマットを指定

入力:手続きに関する文章

課題: 手続き文章には、申請書の具体項目が欠落している場合が多く、申請書の情報も入力に加える必要がある

※当日はこれらのライブデモを行いました

Tips (発展途上)

逐条解説をsystem promptに埋め込む (1/4)

GPTが備えている「知識」は古く (2021年9月まで)、ネット上に広く流布される誤った見解が反映されている場合がある

そこで新規制定された法令に対応すべく、プロンプトにその法令の内容を直接書き込む

GPTにとっては国語の読解問題を解くような形になる

法令の文章そのものだけでなく、実際の運用に関わる箇所も回答可能にするため、逐条解説を用いる。しかし、逐条解説はtoken数上限を大きく超えてしまう

英訳によってtoken数を軽減できるが、反応が機械翻訳的に不自然になったり専門用語が違ったりしてしまう

広く流布される誤った見解の例の参考: GPT-4 System Card (https://cdn.openai.com/papers/gpt-4-system-card.pdf)

逐条解説をsystem promptに埋め込む (2/4)

そこでGPTを使って文章を項目ごとに要約させる。

1　「行政機関」（第１項）

この法律における「行政機関」は、行政機関個人情報保護法第２条第１項に規定する行政機関とするものである。

なお、会計検査院については、憲法上の機関としての位置づけを持っており、かつ、内閣に対し独立の地位を有する機関であることを踏まえ、行政機関個人情報保護法上は、会計検査院の保有する個人情報ファイルは、総務大臣への事前通知の対象とされていない。また、会計検査院長が行った開示決定等、訂正決定等及び利用停止決定等に対する不服申立てについては、別途会計検査院に設置される審査会に諮問することとされており、総務大臣は、会計検査院の長に対し、資料・説明の要求及び意見の陳述は行わないこととされている。

本法における会計検査院の取扱いについては、行政機関として、原則として他の行政機関と同様の規制が及ぶこととしつつ、行政機関個人情報保護法上特例が認められる範囲でこれを追認するという考え方がとられている。

1.「行政機関」（第1項）
- 本法における「行政機関」は、行政機関個人情報保護法第2条第1項に規定する行政機関とする。
- 会計検査院は憲法上の機関であり、内閣に対し独立の地位を有する。
- 会計検査院の個人情報ファイルは、総務大臣への事前通知の対象外。
- 会計検査院長の決定に対する不服申立ては、審査会に諮問される。
- 本法における会計検査院の取扱いは、原則として他の行政機関と同様の規制が及ぶが、特例が認められる範囲で追認される。

508文字が204文字に削減

左記の文章は、行政手続における特定の個人を識別するための番号の利用等に関する法律の逐条解説から
(https://warp.da.ndl.go.jp/info:ndljp/pid/12165359/www.cao.go.jp/bangouseido/pdf/chikujou.pdf)

逐条解説をsystem promptに埋め込む (3/4)

なぜ文節ごとに要約させるのか？

1. GPTが回答するさいに「どこ」の情報を参照したか、を応えられるため
2. 人間がプロンプトを作るときに見落としの確認がしやすいため
3. 要約の精度をあげるため

一方、このやり方だと要約時に重要な情報が落ちてしまう可能性がある

実際に作った後に応答をさせ、おかしなところを逐次修正していく、が今のところよさそう

逐条解説をsystem promptに埋め込む (4/4)

おかしなところがないかの調べ方（user プロンプトの例）

1. 「XXXについて1000文字程度の解説記事を作ってください」
2. 「XXXについての想定QAを10個作ってください」
3. 「XXXについての理解度確認クイズを作ってください」
4. 実際にきた問い合わせを入力してみる

作るさいにXXXについての専門家が必要＆AIが専門家を上回ることはない、の2つ観点が重要

Tips (発展途上)

プロンプト開発の試行錯誤の手順例

1. modelをGPT3.5にする

2. systemとuserを書き、挙動を確認する

3. どうしても直示的に指定したい場合は assistant を指定する

4. 挙動に不満な場合はGPT4にしてみる

GPT4でダメならプロンプトが悪いかそもそも課題が難しすぎる（だいたい前者）

GPT4でOKなら、3.5でも動くようにもっとプロンプトを平易な書き方にできないか検討

例えば、プロンプトに埋め込んだルール文章が冗長で長い場合は要約する

それでも難しい場合はGPT4を採用

ここまでのまとめ

プロンプトの試行錯誤だけで結構多くのことができる

アイデアによっては、今回のデモ環境だけで実現できる場合もあります

→その場合、デモ環境のためのコードなどの資材をAI班から提供できます

しかし、それでは不十分な場合も多々あります

講義の最後で、発展的内容としてGPT API を1回たたくだけ以外のケースについて紹介します

質疑応答 (3/4)

発展的＆将来の内容

発展

GPTの入出力の書き換え

GPT APIをシステムに組み込むさいに、品質を安定させる＆不適切なコンテンツ生成に対して

頑健になる手法 **(絶対ではない点に注意)**

不適切の例: 虚偽 (Hallucination)、政治的に偏りのある意見、差別的な言動、犯罪を増長させる発言、ユーザーを危険にさらす手順、明確な著作権侵害などなど

手法１　エンドユーザからの入力文章を書き換えて GPT API におくる

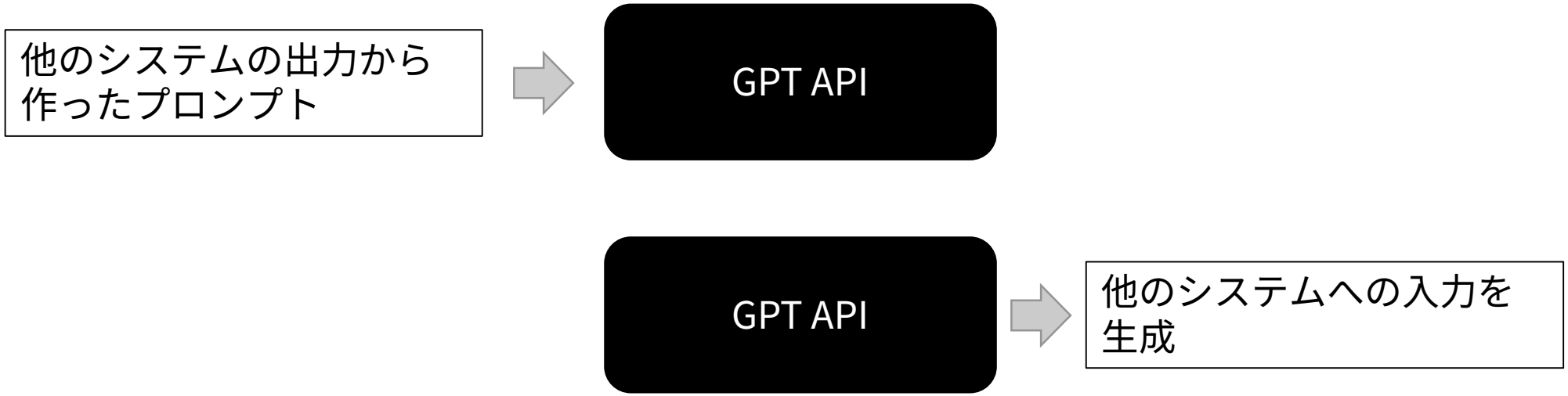
例: ユーザの入力に 「(第N条によると、の形式で回答)」を付け足す

手法２　GPT API の出力文章を書き換えてエンドユーザに返す

例: GPTの出力をjson指定し、そのjsonの一部キーの内容だけ返す。jsonが返ってない場合はエラーを返す

不適切なコンテンツ生成だけでなくシステムのプロンプトを表示させるプロンプトインジェクションについても頑健になる。入出力の書き換えと限界は、このゲーム(https://prptinj.net/)　で体験可能

他システムとGPT API の組み合わせパターン

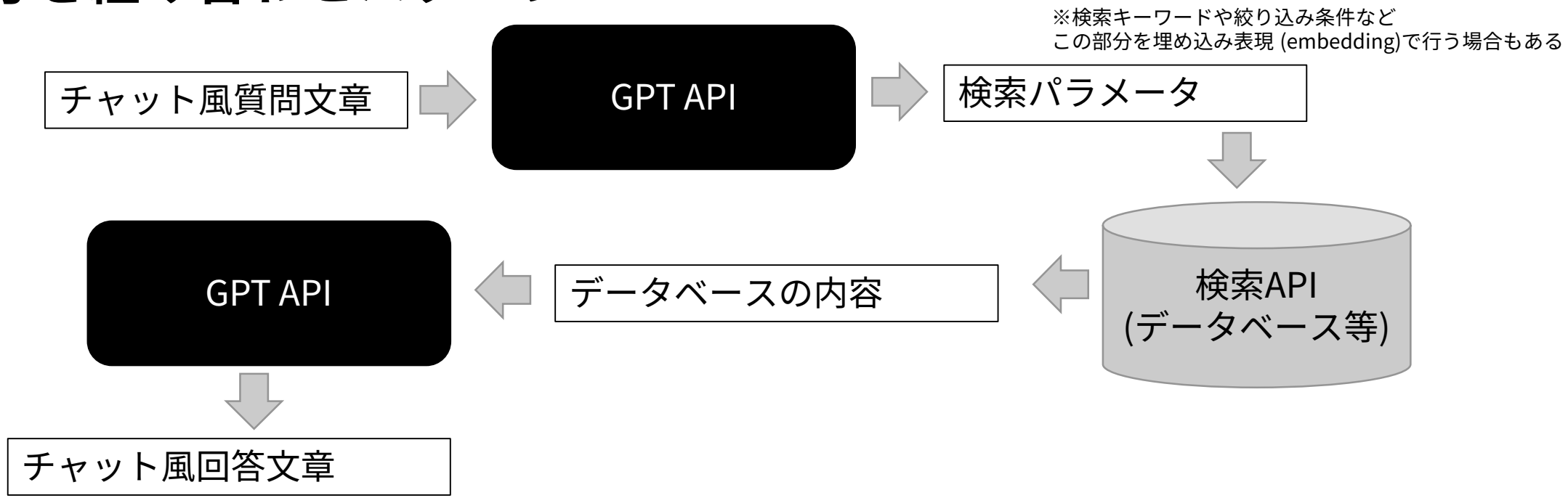


この２つのパターンの組み合わせで、さまざまな発展的ユースケースが作られる

前者の例: 法令検索APIの出力の法令文章をGPT API で要約

後者の例: GPTで表記ゆれ吸収＆フォーマットを成型した結果をDBに登録

両方を組み合わせパターン



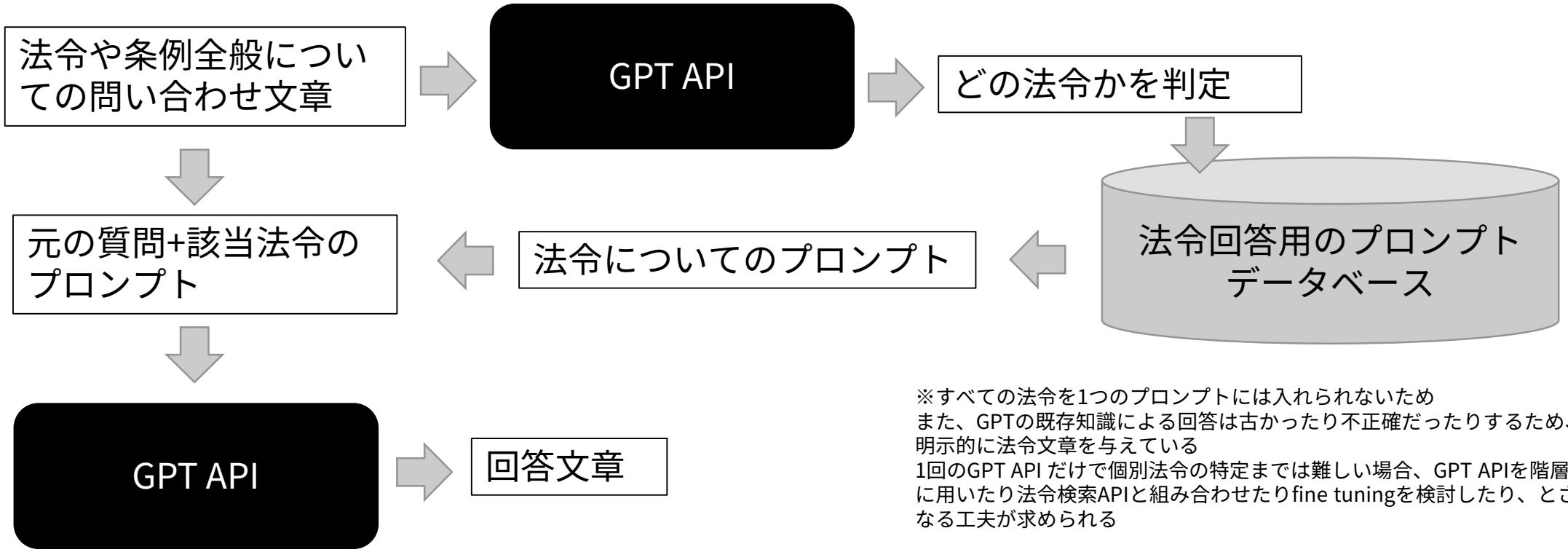
この２つのパターンの組み合わせで、さまざまな発展的ユースケースが作られる

前者の例: 法令検索APIの出力の法令文章をGPT API で要約

後者の例: GPTで表記ゆれ吸収＆フォーマットを成型した結果をDBに登録

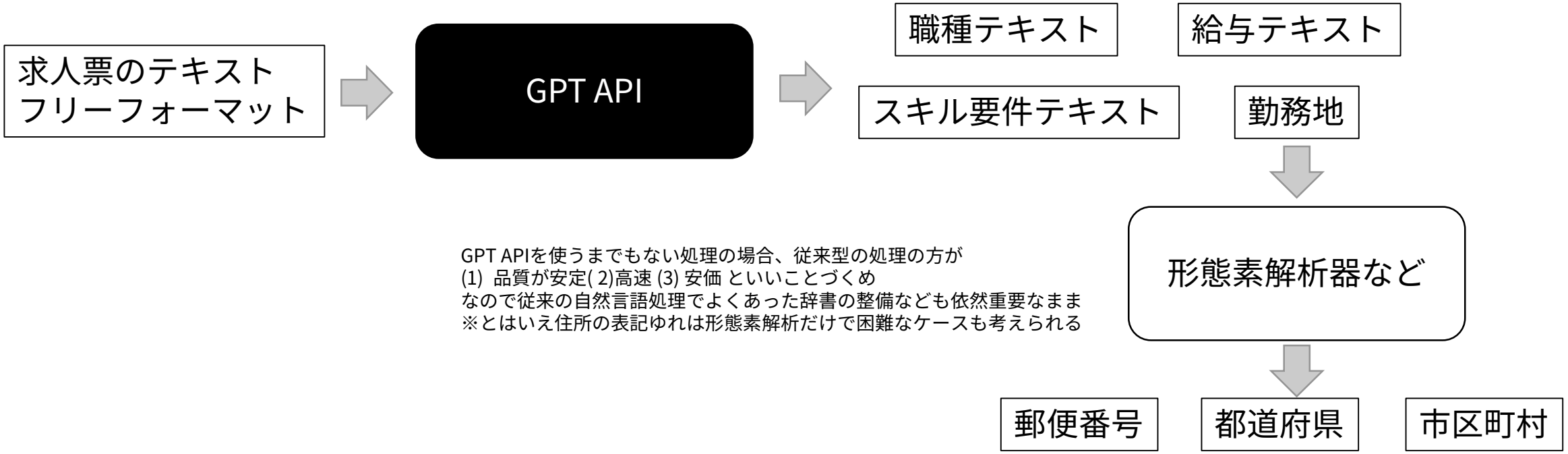
参考: 0421DS協会_ChatGPTによって描かれる未来とAI開発の変遷. ...

GPT APIの出力に応じて、GPT APIの入力をつくるパターンもあり



※すべての法令を1つのプロンプトには入れられないため
また、GPTの既存知識による回答は古かったり不正確だったりするため、明示的に法令文章を与えている
1回のGPT API だけで個別法令の特定までは難しい場合、GPT APIを階層的に用いたり法令検索APIと組み合わせたりfine tuningを検討したり、とさらなる工夫が求められる

GPT APIと従来の自然言語処理を組み合わせるパターン



GPT APIを使うまでもない処理の場合、従来型の処理の方が
(1) 品質が安定(2)高速 (3) 安価 といいことづくめ
なので従来の自然言語処理でよくあった辞書の整備なども依然重要なまま
※とはいえ住所の表記ゆれは形態素解析だけで困難なケースも考えられる

参考　とにかく日本の住所のヤバさをもっと知るべきだと思います　https://note.com/inuro/n/n7ec7cf15cf9c

将来

GPT以外のAI系APIの組み合わせ

6/15に英語版がGA。日本語はまだ

GoogleのGCP Vertex AIも生成系AIのAPIを近日リリース予定

https://cloud.google.com/generative-ai-studio

将来的には「どのモデル」かを意識しないで、大量のAPIの回答を統合したものが主流になるかもしれない

すでにその種のオープンソースも出始めている

https://github.com/ShishirPatil/gorilla　適切なAPIを選択するAI

※機械学習の歴史的には、「個別に作られた大量のモデルを統合する」アプローチは単独のモデルよりも良い結果になる傾向にある（Boostingなど）

質疑応答 (4/4)

100回の会議より、1回のAPI callの方が理解が進みます

まずは試してみましょう！

結び（再掲）

アイデアの価値は「検証」してみないとわからない

「Microsoft でテストされたアイデアのうち、改善を示すメトリクスを実際に改善できたのは3分の1にすぎない」 (Microsoft社 元Vice President)

「もしあなたが実験主導のチームにいるなら、70%の仕事が捨てられることに慣れてください。それに応じてプロセスを構築しましょう」(Slack社 Director)

A/Bテスト実践ガイド p14より

一方で

「アイデアの価値を見積もることは難しい。このケースでは、年間１億ドルの価値ある単純な変更が何か月も遅れていた。」(同著 p5より) こともあります

午前中のアイデアソンで出たアイデアはちゃんと検証するまで価値があるかは不明です。

ちゃんと検証するためには、ちゃんとアイデアを形にする必要があり、そのためには「どう作るか」「そもそもできそうか」を試すことが重要です

要するに、作って終わりではないです。が、作らないと話にならないので、さっさと作りましょう

ここからはGPT のAPIを実際に叩いてみて、自身のアイデアを形にするための手ごたえをつかむ時間となります

この時間だけで形にしきることは難しいと思いますが、明日以降に**独力で検証を進めるための方法論**を身につけられれば幸いです

（明日以降のデジ庁の何らかの支援が必要な場合は追って相談とさせてください）

A/Bテスト実践ガイド 真のデータドリブンへ至る信用できる実験とは https://www.kadokawa.co.jp/product/302101000901/