



リアル業務でChatGPT APIを使うコツ

—

株式会社Helpfeel @teramotodaiki

ChatGPT APIがついに出了たね！

と書いたのが4週間前

もはや遠い昔の話に思える

Helpfeel社ではどんな風に使ってきたのか？



Generative Writer

 Helpfeel

Generative Writer

AK

Generative Writer

質問・回答テキスト

単語やメモ

サンプルをロード

入力ソース

衣類 タグを外していなければ 返金OK
食品 未開封なら 返金可能
それ以外はsupport@example.com

FAQの記事案を生成

✓ 生成しました

出力結果

タイトル

返金について

本文

衣類の場合、タグを外していなければ返金可能です。食品の場合、未開封であれば返金可能です。それ以外の商品については、support@example.comにお問い合わせください。

Intention prediction supporter

Intention prediction supporter

▼ 本文を理解する

重要な記述

- ☒ IvySearchは、画像の属性情報やOCRスキャンされた情報から自然言語処理を活用して、現在注目している画像に近い画像を連想的にたどっていくことによって、画像やデータを捜したり眺めたりする連想検索手法である。
- ☐ IvySearchは、ブラウザ上で実行できる。
- ☒ 類似度が高い画像は、「関連画像」として画像ページに表示される。
- ☒ IvySearchに使われるデータの取り扱いについては、画像の属性情報を参照することができる。
- ☐ IvySearchの開発思想や具体的な活用事例については、Gyazo BlogやTechCrunch Japanの記事を参照することができる。

連想される質問文

- ☐ 関連画像の表示方法
- ☐ 画像の属性情報の取り扱い
- ☒ 自然言語処理を活用した連想検索手法

▼ 質問文を言い換える

? 自然言語処理を活用した連想検索手法

Generated expressions

72 patterns

(言語処理を | 自然言語処理を | プログラム言語処理を) (使った | 用いた | 利用した | 活用した) (検索手法 | 連想検索技法 | 連想検索方法 | 連想検索手法 | イメージ検索手法 | キーワード検索手法)

言語処理を	使った	検索手法
自然言語処理を	用いた	連想検索技法

Contact Sense AI

メールでのお問い合わせ

お問い合わせ内容

お世話になっております。〇〇商事の山田と申します。
弊社はセキュリティに厳しく、外部のサーバーを利用することができません。
そういった場合でも御社の検索システムを利用可能でしょうか？
よろしくお願いします。

もしかして...

- ☐ Helpfeelのセキュリティを確認するためサービス構成を知りたい
- ☐ Helpfeelの検索ボックスだけ運用中のウェブサイトに取り込みますか？
- ☐ 運営会社のプライバシー・ポリシーを確認したい
- ☐ Helpfeelは自社のサーバーに対応していますか？
- ☐ FAQを検索エンジンの対象外にしたい



OpenAI社の技術を活用

問い合わせフォームの検索精度を30%向上

「Contact Sense AI」を公開

 Helpfeel

The graphic features a light blue background with a faint, circular, gear-like pattern. It is framed by a thick border that is orange on the left and top, and blue on the right and bottom.

本日リリース!! (技術的な詳細は後述)

作って学んだ知見を共有します

表に出ているもの・まだ出ていないもの・社内でこっそり使っているもの

色々作った結果、知見が増えてきた

- プロンプトとの向き合い方
- インフラ
- コスト最適化
- Hallucination対策

これらの知見を一斉放出します

細かく話してる余裕がないので、気になるところはsli.doで聞いてください↑



ボーナスタイムが終わっても役に立つ話

今は「ChatGPTを使った」って言うだけで注目が集まるボーナスタイム

てっとり早く成果を出せる手段(ライブラリ,プロンプト)に注目が集まりやすい

そこは既に多くの資料が上がっている(真偽は知らない)し、そもそもOpenAIの公式資料が非常に分かりやすいので、今回はあまり触れません

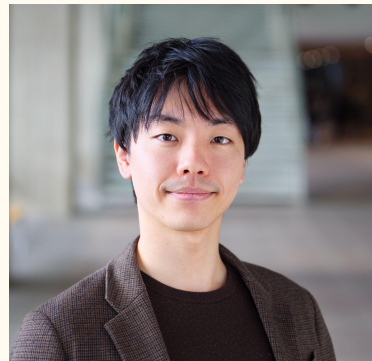
ボーナスタイムが終わったら、「顧客に価値を届けることが大事ですよ」って話に戻ってくる。そのとき技術的にはWebのベストプラクティスが重要になる

そのときにGPT-5や6が来ても(きっと)役に立つ、息の長い資料にしたい

Who am I?

@teramotodaiki

Webエンジニア(機械学習エンジニアではない)



ChatGPTに対するスタンス:

- ユーザーインターフェースが大きく変わる可能性にワクワクしている
- UI/UXを考えてプロダクトを作るのが好きなWebエンジニアとして、こいつを取り込んでとにかく良いものをたくさん作りたい
- 毎日アイデアが湧いてきて、実装する時間が足りない

そもそもChatGPTのAPIとは

「ChatGPTにこんな文章を投げたら、こういう出力が返ってくる」のAPI版

厳密に等価ではないが、ChatGPTに出来ることはAPIでも出来る

APIを使うメリット

- 自動化できる
- 従量課金(とても安い)
- 再現性がある
- プロンプトの自由度が高い
- 出力されたテキストを使って別のシステムを動かせる

ChatGPT APIの使い方

基本的な話はChatGPTに聞けば分かるので、省略

Sign upすると\$18のクレジットが貰えるので、とにかく試そう

重要なパラメータは以下

- messages: Chat ML形式のプロンプト(後述)
- model: モデルの名前。gpt-3.5-turbo など
- temperature: ランダム性のようなもの。0なら常に同じ回答を返す
- max_token: 出力トークンの上限
- stop_sequences: この文字が来たら出力をやめる
- stream: パパパッと出力を出せるやつ(後述)

Chat ML?

Playground

Load a preset... Save

SYSTEM
あなたは天使AIです。
天使として正しい発言をします。|

USER 財布を拾ったんだけど、どうしたらいいかな

⊕ Add message

Submit ↺

これを・・・こうじゃ

```
{
  "messages": [
    {
      "role": "system",
      "content": "あなたは天使AIです。\\n天使として正しい発言をします。"
    },
    {
      "role": "user",
      "content": "財布を拾ったんだけど、どうしたらいいかな"
    }
  ]
}
```

チャットボットを作るためのAPIなのか？ - No.

チャットボットを作るためのChat MLではない

ここから歴史の話(なんと1ヶ月も昔の話である！)

以前のAPI(text-davinci-003など)はCompletionタスクと言って、文章の続きを書かせるものだった。

Playground

Load a preset... ▾

SaveView codeShare...

teramotodaikiの所属は、

Submit



13

チャット形式が便利というだけに過ぎない

Completionタスクで指示をするにはコツが必要で、「指示」に対する「回答」を得るためのテクニックとして、会話のような形式が用いられた。



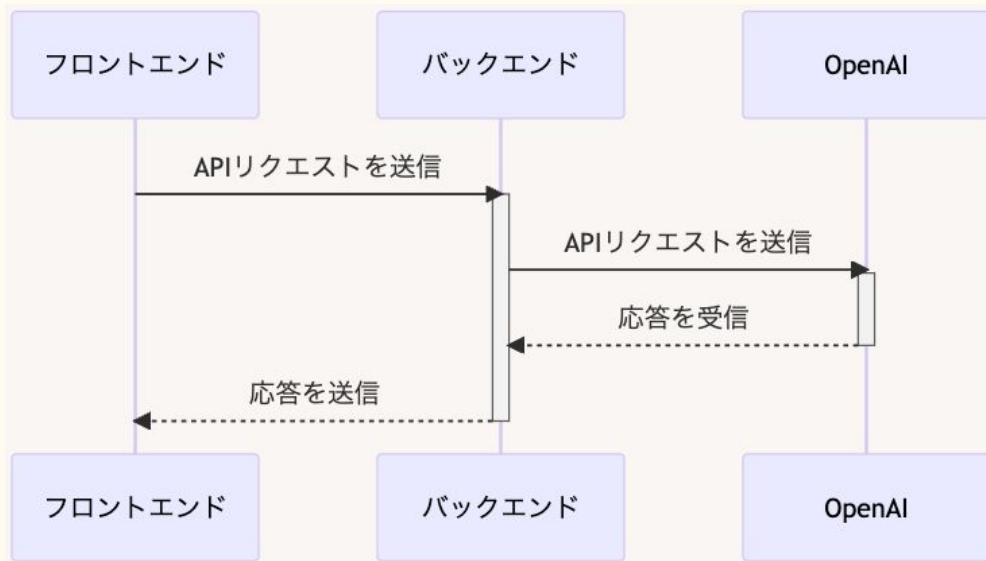
このように本質的でない文字列処理がLLMsを扱う障壁になっていた

→ OpenAIがChat MLという道筋を示してくれた

(素直に受けとれば) systemが意図、userが入力、assistantが出力

ChatGPT API
完全に理解した

実際の利用フロー

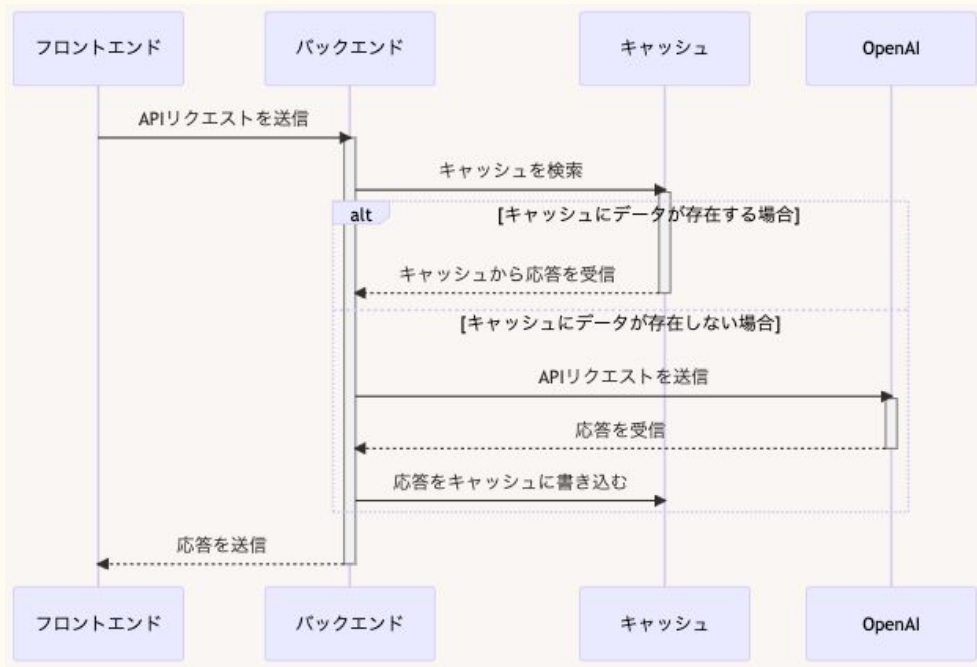


API Keyはフロントエンドに書き込めないで、別のサーバを経由する

プロンプトも秘匿できる

※リクエストあたりの実行時間に制限のあるサービス (GAS、AWS Lambda、Cloud Function gen 1など) は注意。生成に1分以上かかって時間切れになることがある

実際の利用フロー (if temperature == 0)



ユースケースによっては入出力のペアをキャッシュできる

- キャッシュで返せれば超高速
- コスト削減も可能

キャッシュキーはAPI request payloadのMD5ハッシュとかで良いと思う

脱線

ていうか

temperature : 0

じゃないユースケース

ってあるんですか？ 

👉なんでこれがデフォルト
じゃないのか謎過ぎる

ユーザーが結果を予測できないツールってUX悪くないですか？

創造的なケースもpromptを変えたり候補を複数出す方が現実的に使い勝手良さそう

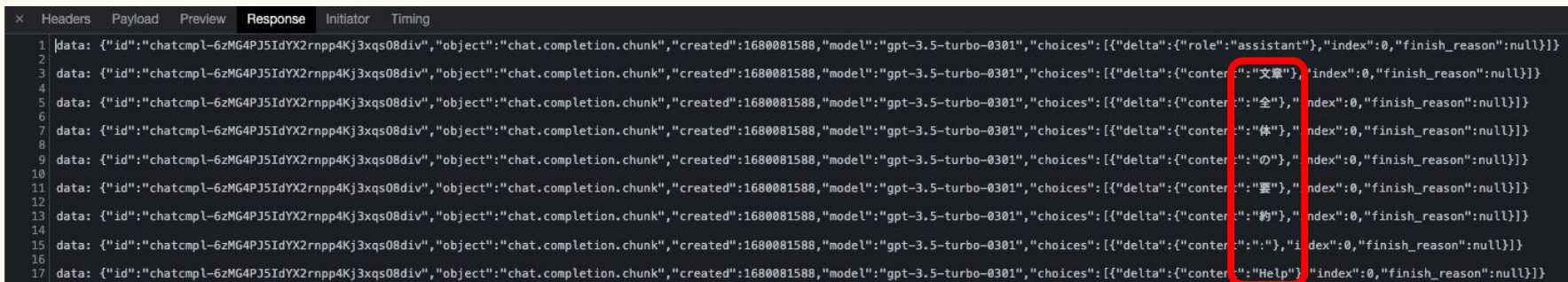
temperature > 0の方が向いてるユースケースがあればコメントで教えてください 

実際の利用フロー (if stream == true)

本家ChatGPTみたいに、パパパパッと文字が出るやつ

Server Sent Eventsというプロトコル

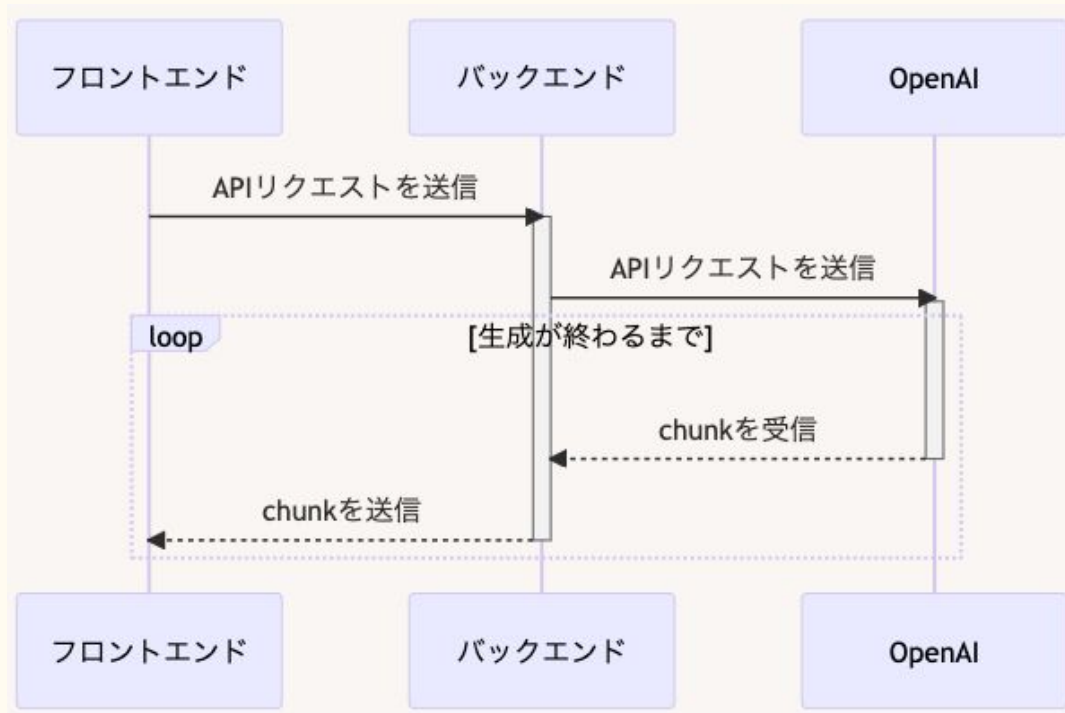
MIME Typeは text/event-stream で、bodyはこうになっている



```
1 data: {"id":"chatcmpl-6zMG4PJ5IdYX2rnpp4Kj3xqs08div","object":"chat.completion.chunk","created":1680081588,"model":"gpt-3.5-turbo-0301","choices":[{"delta":{"role":"assistant"},"index":0,"finish_reason":null}}]
2
3 data: {"id":"chatcmpl-6zMG4PJ5IdYX2rnpp4Kj3xqs08div","object":"chat.completion.chunk","created":1680081588,"model":"gpt-3.5-turbo-0301","choices":[{"delta":{"conter":"文","index":0,"finish_reason":null}}]
4
5 data: {"id":"chatcmpl-6zMG4PJ5IdYX2rnpp4Kj3xqs08div","object":"chat.completion.chunk","created":1680081588,"model":"gpt-3.5-turbo-0301","choices":[{"delta":{"conter":"全","index":0,"finish_reason":null}}]
6
7 data: {"id":"chatcmpl-6zMG4PJ5IdYX2rnpp4Kj3xqs08div","object":"chat.completion.chunk","created":1680081588,"model":"gpt-3.5-turbo-0301","choices":[{"delta":{"conter":"体","index":0,"finish_reason":null}}]
8
9 data: {"id":"chatcmpl-6zMG4PJ5IdYX2rnpp4Kj3xqs08div","object":"chat.completion.chunk","created":1680081588,"model":"gpt-3.5-turbo-0301","choices":[{"delta":{"conter":"の","index":0,"finish_reason":null}}]
10
11 data: {"id":"chatcmpl-6zMG4PJ5IdYX2rnpp4Kj3xqs08div","object":"chat.completion.chunk","created":1680081588,"model":"gpt-3.5-turbo-0301","choices":[{"delta":{"conter":"雲","index":0,"finish_reason":null}}]
12
13 data: {"id":"chatcmpl-6zMG4PJ5IdYX2rnpp4Kj3xqs08div","object":"chat.completion.chunk","created":1680081588,"model":"gpt-3.5-turbo-0301","choices":[{"delta":{"conter":"約","index":0,"finish_reason":null}}]
14
15 data: {"id":"chatcmpl-6zMG4PJ5IdYX2rnpp4Kj3xqs08div","object":"chat.completion.chunk","created":1680081588,"model":"gpt-3.5-turbo-0301","choices":[{"delta":{"conter":"の","index":0,"finish_reason":null}}]
16
17 data: {"id":"chatcmpl-6zMG4PJ5IdYX2rnpp4Kj3xqs08div","object":"chat.completion.chunk","created":1680081588,"model":"gpt-3.5-turbo-0301","choices":[{"delta":{"conter":"Help","index":0,"finish_reason":null}}]
18
```

いわゆるJSON Linesの行頭にdata: が付いただけ

実際の利用フロー (if stream == true)



chunk単位で届く

JSON LinesをStreamで捌く

※Cloud FunctionなどServer Sent Events非対応のPaaSに注意

実際の利用フロー (if stream == true) ブラウザ側

```
const response = await fetch("/api/completion", {
  method: "POST",
  headers: {
    "Content-Type": "application/json",
  },
  body: JSON.stringify({ content: text.trim() + "\n" }),
});
if (!response.body) throw new Error("No body");

// Server Sent Eventsで受け取る
const reader = response.body.getReader();
const decoder = new TextDecoder("utf-8");
const delimiter = "data:";

sse: while (true) {
  const { value, done } = await reader.read();
  if (done) break;
  const text = decoder.decode(value);
  const payloads = text.split(delimiter).map((s) => s.trim());
  for (const payload of payloads) {
    if (payload === "") continue;
    if (payload === "[DONE]") break sse;
    try {
      const json = JSON.parse(payload);
      // console.log(json);
      const current = json.choices[0].delta.content || "";
      setData(prev => prev + current);
    } catch (error) {
      console.error("JSON parse error:", payload);
    }
  }
}
```

Server Sent EventsのMDNに載ってる方法
(HTML Standard/EventSource)は仕様の
向いてないので、素直にfetch APIを使う

response.body.getReader() でStreamを読める

[DONE] が終了の合図

data: 以降をJSON Parseして中身を取り出す

setDataはUIに反映する関数

??く わかったわかった！ ChatGPTの話に戻してくれ！

プロダクト開発のどこにChatGPTを使うかが重要

まずはこの話をします

プロンプトエンジニアリングは避けて通れない

ただし、細かなテクニックは非本質的で、得た知見はすぐに陳腐化する

向き合い方が重要

どこにChatGPTを使うか

ChatGPTも万能ではない。微妙な出力が出たり、明らかに間違っている時もある

正確性と創造性のバランス

- 正確性: ユーザーは間違った回答を許してくれるだろうか？
- 創造性: あっと驚くような回答でユーザーを興奮させたい？

どちらに重きをおくにせよ、ユーザーとの期待値調整が重要

正確性と創造性どちらも百点満点のプロダクトは作れない

Hallucination (ハルシネーション)

いわゆるAIがつく嘘の問題

現時点において、ChatGPTをビジネスに使う上で一番の壁と思われる

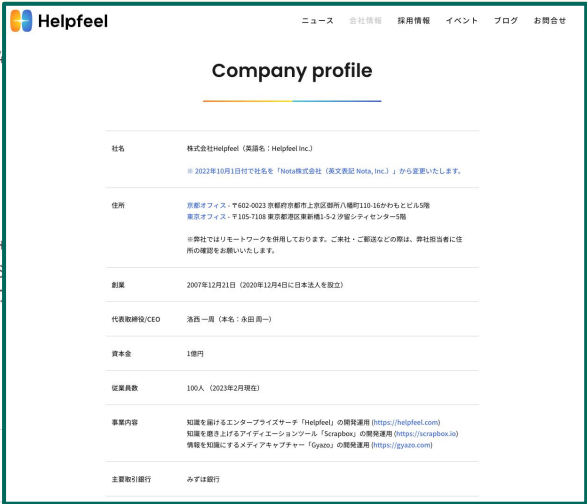
Hallucinationを確実に防ぐ方法はないが、リスクを軽減する方法はある

- 正しい情報を与える(創作を許さない)
- 問題を分割する
- Human in the loop

どのアプローチが良いかはユースケースによって異なる

正しい情報を与える(創作を許さない)

↓ HPの会社情報



USER	株式会社Helpfeelの代表は？
ASSISTANT	田中 雄介 (Yusuke Tanaka) さんが株式会社Helpfeelの代表を務めています。

	2007年12月21日 (2020年12月4日) 代表取締役/CEO 洛西 一周 資本金 1億円 従業員数 100人 (2023年2月現在) 事業内容 知識を繋げるエンタープライズサーチ「Helpfeel」の開発運用 知識を磨き上げるアイディエーションツール「Scrapsbox」の開発運用 情報を知覚にするメディアキャプチャー「Gyazo」の開発運用 主要取引銀行 みずほ銀行
USER	株式会社Helpfeelの代表は？
ASSISTANT	株式会社Helpfeelの代表取締役/CEOは洛西一周氏です。

正しい情報を持ってきて、文章だけ生成させる

※それでもHallucinationを完全に防ぐことはできない

問題を分割する

例) 株式会社Helpfeelに詳しいAIを作りたい

1. 本当に株式会社Helpfeelに関係した質問なのかを確認する
→無関係な質問に対しては「それはBingにでも聞いてください」と回答する
2. 個人情報や機密情報を聞き出そうとしていないかを確認する
→プライバシーポリシーのURLをご案内する
3. 質問に関連しそうな情報をなんらかのDBから検索して取り出す
→検索にヒットしない場合は「勉強します」と言って社内Slackに投げる
4. ヒットした文書とユーザーの質問文が本当に関連しているか確認する
→関係していそうな文書を優先
5. 複数の正しい情報源をもとに回答を作文させ、出典と共に見せる

問題を分割する

エンジニアもステップバイステップで考えよう

Q→A を一発で出せる魔法のプロンプトは今のところ見つかっていない

ChatGPTが(ほぼ)確実に解ける簡単な小タスクに分解し、積み上げよう

問題を分割するメリット

- 上手く分割すれば正答率を向上できる
- コンプライアンス的なsomething
- どこで躓いたのか原因分析しやすい
- テストしやすい

分類問題にすることで他のシステムを連携しやすい

単純なコマンドに落とし込めれば古典的プログラミングで処理できる

ChatGPTにコマンドを出力させて、そのコマンドを別のシステムに入力する

言語を媒介として、まったく別の非AIシステムを動かせる

入出力の境界ができるので、テストしやすい

ChatGPTを使ったモジュールのテスト

<https://github.com/openai/evals>

```
$ pip install oaeval
```

テストケースをjsonl形式で作成し、API Keyを入れて実行

全てのテストケースをAPI経由で実行し、正答率を算出してくれる(富豪的！)

評価ロジックは、Pythonで記述できるなら何でもよい

→「この文字列を含むか？」でテストできるプロンプトにすると良い

これがLLMs時代のTDD？(適当)

Human in the loop

ChatGPTの入出力をプロンプトだけで完全にコントロールすることは出来ない。
Hallucinationが発生しても人間がリカバリーできる仕組みにする

抑制ではなくフェイルセーフのアプローチ(ビジネスで受け入れられやすい?)

例えば…

- ユーザーが次に入力する文章をサジェストする
- 顧客対応をChatGPTで自動化しつつ、難しいケースだけ人間が巻き取る
- 意思決定の材料となる候補をChatGPTに出させて、判断は人間が行う

脱線

当事者がプロンプトを書くことでHallucinationに立ち向かう

自分で書いたプロンプトなら、多少不調でも「不調だな、改良しよ」と思う

プロンプトが分からないと、ユーザーは何がダメだったのかすら分からない

→エンジニアも何がダメなのかよく分からない。つらい

応用先は限られるし教育も必要だが、社内でこれが出来れば強いと思う

全員プロンプトエンジニア

プロンプトを改善する

- 良いプロンプトとは何か？
- どのように改善のサイクルを回せばよいのか？
- 最新のプロンプトはどうやってキャッチアップすればよいのか？ …etc

その前に、プロンプトとの向き合い方を考えたい

プロンプトエンジニアリングだけですべてを解決するという発想は捨てました

サンプル数1でプロンプトをいじるとこうなる

「なんとなく生成結果がイマイチだな～」

「この一文足してみよ(思いつき)」

「なんか上手くいった！なるほど、指定条件が足りなかったのか！」

「もっと条件足してみるか」

「あれ、なんかさっきより悪くなってる・・・」

そのプロンプト、たまたま上手くいっただけです

と自分に言い聞かせ続けよう

人間は上手くいったら自分の功績だと思い込みがち

別のサンプルを試すと、再現性がないことに気づく

無意味な空行を1個入れただけで出力が大きく変わったり

(GPT-3.5時代の)プロンプトエンジニアリングは諸行無常

常に複数のテストケースで試そう

N=1でプロンプトの細部をいじらない

数個でいいので、まずはテストケースを書く

判断基準が定性的になるのはしょうがない

「N個のサンプル中、M個が良くなった」という改善を積み重ねよう

こういうSaaSありそう

弊社ではGASとSpreadsheetで実験室を作っておりオススメです(後述)

Twitterに書いてあることは(ほぼ)生存バイアス

とくにChatGPTのスクショ・動画は、まず疑った方が良くと思う

フェイクではなく、切り取り方の問題

スクショに載せていない大量の会話ログや、試したけどイマイチだった別のプロンプトが山のようにあって、上手くいった部分だけ切り取っているかも知れない

それはプロダクトに組み込めない一期一会の自然現象なので・・・

GASとSpreadsheetでプロンプト実験室を作ろう

指示	ユーザー入力	天使AIの回答
あなたは天使AIです。 天使として正しい回答をしてください。	財布を拾っちゃったんだけど、どうしたらいいかな？	拾った財布は、近くの警察署や駅の忘れ物センターに届けることが望ましいです。もし、財布に身分証明書や連絡先があれば、その情報を元に直接持ち主に届けることもできます。ただし、財布の中身を勝手に使ったり、持ち主に連絡をしないで自分で使うことは避けましょう。
	ダイエット中なんだけど、お腹すいちゃったよ～	ダイエット中は、食事制限があるためお腹が空きやすくなりますね。そんなときは、低カロリーで満腹感のある食品を選ぶと良いですよ。例えば、野菜や果物、豆類、鶏肉や魚などがおすすめです。また、水分をたくさん摂ることも大切です。水やお茶、スープなどを飲んで、お腹を満たしてくださいね。
	友達の彼氏を好きになっちゃった...どうしたらいいと思う？	まず、友達の彼氏を好きになってしまったことを友達に正直に話すことが大切です。友達には自分の気持ちを理解してもらい、相談に乗ってもらうことができます。 ただし、友達の彼氏を好きになってしまった場合、友達の気持ちや関係性を壊さないように注意が必要です。友達の彼氏に対してアプローチすることは避け、友達との関係を大切にすることが大切です。 また、自分自身の気持ちを整理するために、友達の彼氏との接触を避けることも考えてみてください。時間をかけて自分の気持ちを落ち着かせ、友達との関係を優先することが大切です。

GASをシュッと書くとできる。キャッシュの有無でコストが大きく変わるので、なるべく中間サーバを立てよう。実行時間制限の問題も回避できる

ChatGPT APIのコスト

ぶっちゃけ赤字でもいいんじゃない？ (他で採算が取れれば) という人は、今は気にしないでいいかも知れないけど、将来的に採算とれるかは気になりますよね

コストの見積もり方法

- モデルによって単価が異なる (gpt-3.5-turboは \$0.002/ 1k tokens)
- 雑な見積もりでもいいなら、日本語1文字1トークン
- これが入力と出力の両方にかかる
- GPT-4は出力の方が高単価

例: 1回の利用で 1,000 tokens × 10,000回利用 = \$20

ターン制の会話はコストが高くなりやすい

本家ChatGPTのようなターン制の会話を愚直に実装するとこのようになる

回答①=GPT(前置き+質問①)

回答②=GPT(前置き+質問①+回答①+質問②)

回答③=GPT(前置き+質問①+回答①+質問②+回答②+質問③)

...

合計トークン数は加速度的に増えていく(どこかで上限に達する)

上手く「過去のやりとりを忘れる」工夫が必要

コストを節約するテクニック

英語を使う

日本語がおよそ1文字1トークンであるのに対し、

英語ではおよそ1単語1トークンなので、入力トークン数を節約できる

プロンプトに「in Japanese」と付けるだけで、回答が日本語なる

(今はできないが)ファインチューニング

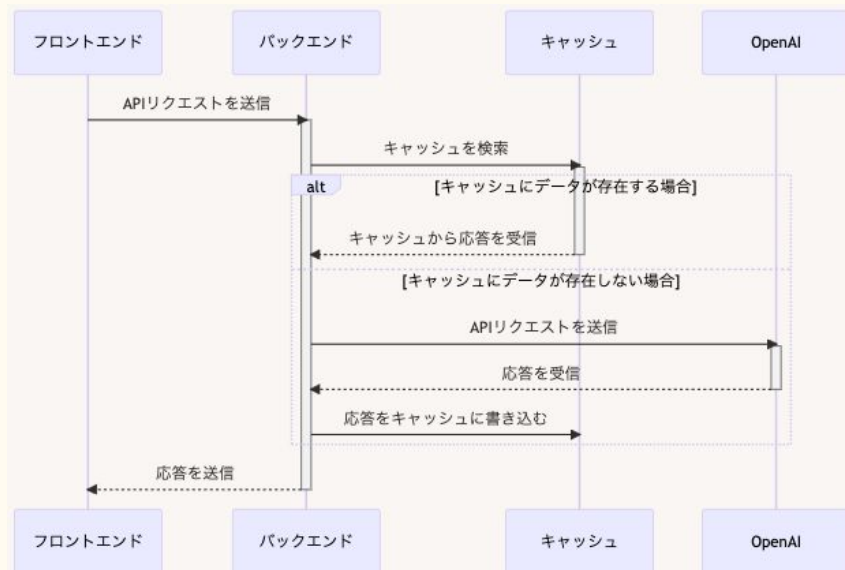
前置き部分なしで、入力→出力のペアでファインチューニングする

前置きがいない分、入力トークンを節約できる

コストを節約するテクニック(キャッシュ)

前半で解説した通り

入力と出力のペアをキャッシュすれば、同じ入力を2度行う必要がなくなる



コストを節約するテクニック (Batch Prompting)

Standard Prompting

```
# K-shot in-context exemplars
Q: {question}
A: {answer}

Q: {question}
A: {answer}

...

# One sample to inference
Q: Ali had $21. Leila gave him half of her
  $100. How much does Ali have now?
-----
# Response
A: Leila gave 100/2=50 to Ali. Ali now has
  $21+$50 = $71. The answer is 71.
```

<https://arxiv.org/abs/2301.08721> 2301.08721

Batch Prompting: Efficient Inference with Large Language Model APIs

Batch Prompting

```
# K-shot in-context exemplars in K/b batches
Q[1]: {question}
Q[2]: {question}
A[1]: {answer}
A[2]: {answer}
} b(=2) samples
  in one batch

...

# b samples in a batch to inference
Q[1]: Ali had $21. Leila gave him half of her
  $100. How much does Ali have now?
Q[2]: A robe takes 2 bolts of blue fiber and
  half that white fiber. How many bolts?
-----
# Responses to a batch
A[1]: Leila gave 100/2=50 to Ali. Ali now has
  $21+$50 = $71. The answer is 71.
A[2]: It takes 2/2=1 bolt of white fiber. The
  total amount is 2+1=3. The answer is 3.
```

固定部1個に対して可変部を複数書けるため、トークンを節約できる

※精度は保証されない

ハードリミットとソフトリミット

リミットが設けられているためうっかりバズってクラウド破産ということはない

- ハードリミット:これを超えるとAPIが止まる
- ソフトリミット:これを超えるとメールが来る

リミットを下げるのは自由だが、デフォルトより高くしたい場合はOpenAI社にリクエストを送る必要がある(クレジットカードの利用枠みたいですね)

リミットは組織単位で適用される(API Key単位やユーザー単位ではない)

組織 (Organization) を分ける

誰かが発行したAPI Keyでうっかりハードリミット分を使い切ったら、その組織のAPIリクエストがすべて失敗する

そこで、例えばこのように分けることができる

- 本番環境で使う組織
- 開発環境で使う組織 (デフォルト)

本番環境でAPIを使う場合は、Request headerにOrganization IDを付与する

リアル業務で使う際の
コツが掴めてきましたね

ここからは心機一転
ベクトル検索の話です

ベクトル検索

検索対象とクエリをベクトルに変換し、ベクトル間の距離で検索する手法

たとえば似た単語同士が近くなるようにベクトル化できれば、単語の意味で検索できるようになる(例:Word2Vec)

ChatGPT以前から存在し、これ自体は特に新しくはない

OpenAI社が出しているEmbedding(文章をベクトル化する)APIが面白い

Embedding API → 実用レベルの日本語全文検索

任意の長さの文章をベクトル化できる。このとき、意味的に近い文章はベクトルの距離も近くなる

ここで「意味とは？」という疑問が浮かぶ。しかし、僕にはよくわからない

確かなのは、OpenAIのEmbedding APIがかなり「意味っぽい」ベクトルを作れて、料金も安く、まあまあの速度(1～2秒)でレスポンスが返るので、検索が大きく変わるということ

全文検索への応用

1. あらかじめ文書を分割してベクトル化し、DBに保存しておく
2. 検索クエリを同じモデルでベクトル化し、DBに保存されたベクトルとの距離が近いものを検索結果とする

以上！全文検索実装完了！（ホンマか？）

ヤバいところ

- 日本語も英語も関係ない
- 表記揺れ/typo/言い換えにもある程度対応できる
- 実用に足る検索精度

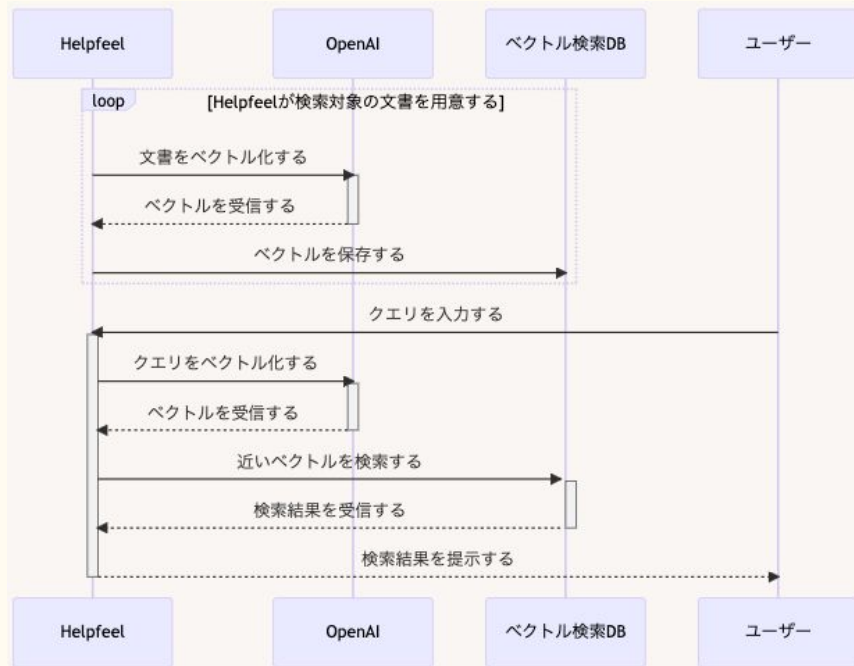
ベクトル検索を本番環境で利用する

その前に注意

- OpenAIのAPIは可用性が低いことを理解する。顧客とSLAを結んでいる場合はAzureを使うか、ベクトル検索を例外として定めるか、fallbackを用意する
- コストが安いとはいえ従量課金なので、サービスインする前に試算すべき
- Embedding APIのコストだけでなく、ベクトルDBなどの周辺システムにもコストがかかる

ベクトル検索のアーキテクチャ

基本的なアーキテクチャはこうなる



ベクトルDB

多次元ベクトルを保存し、与えたベクトルと距離の近いものを取り出せるDB

OpenAIのEmbeddingモデルであれば1536次元

近似最近傍探索を備えたDBもある。データ量が多い場合に有効

ユースケースに応じて様々な利用形態がある

- Pythonのライブラリ(例:faiss、annoy)
- Dockerで利用できるAPIを備えた検索サーバ(例:milvus、Qdrant)
- 従量課金制のSaaS(例:Pincone、Zilliz Cloud、Qdrant Cloud)

実験やプロトタイピングにはライブラリ、

本番環境にデプロイする場合はインフラを用意しなくて済むSaaSがオススメ

Embedding APIのコツ

1回のリクエストに1秒ちかくかかるため、
逐次で1万回叩くと1万秒ちかくかかって遅い(503が返ることもしばしば)

そういう場合は input に配列を入れると良い

具体例) 4時間かかっていたバッチ処理を500件ずつbulkで入れたら20分で完了

Q&Aの壁と検索精度をLLMsで向上するテクニック

HyDE (Hypothetical Document Embeddings)

質問から文書を検索するとき、あらかじめChatGPTに回答を聞いて、その回答をクエリとして文書をベクトル検索すると精度が上がる。それっぽい文章があればよいので、ChatGPTの回答は間違っているても良い

デメリット

- 遅い(文章の生成が終わるまで待つ必要がある)
- Embedding APIだけでなく、ChatGPT APIの使用料金もかかる

Helpfeelでは上記とは全く異なるアプローチで検索精度を向上しています

検索精度を向上するテクニック (Helpfeel方式)

回答 (FAQ) に対する想定質問を事前に作っておく

質問文から質問文を検索しているので、高い精度が出る

Helpfeel社内では「文書拡張」と呼ばれているテクニックで、ベクトル検索以前から検索のコア技術として採用。これをベクトル検索にも転用した

専門のテクニカルライターチームを作ること、質の高い質問文を用意できる

これもHuman in the loopの一種 (ビジネス的には「堀」と言える)

デモあるの？

<https://helpfeel.com/help/contact-us> で試せます！！！！

Helpfeelのお問い合わせフォームにはHelpfeelのことしか書かれていないので、みなさん「Helpfeel導入に興味がある人」というていで質問を書いてください

※フォームは送信しないでください(営業の人が混乱してしまうので)

「うちの会社に欲しいな」って思った方はぜひお問い合わせください👍

まとめ

この資料はTwitterに放流します

俺たちの戦いはまだ始まったばかりだ！！！！

sli.doの質問 に答える

ChatGPT使ってて面白いアプリは？本家が一番面白いみたいな状態になっており

いや～ほんとそれですよね😭 Webエンジニアとしては、せめて自分が使うツールくらいは自分で作りたい！！！！と思いながら日々本家を使っており()

めっちゃ宣伝ですみませんが、僕が副業?でやってる会社でもめっちゃ面白い使い方してるのでぜひ！

<https://prtimes.jp/main/html/rd/p/000000001.000118926.html>

プロンプトをうまく扱うフレームワークある？

フレームワークと呼べるかは微妙ですが、Completionタスクの時代はLangChainがその役目だったのかな～と思います。テンプレートエンジンだったり、Toolsだったり

でもChat MLがその立ち位置をかつさらっていった印象があって、決められたroleとcontentの中でプロンプトを書くことが強制された結果、それが事実上のベストプラクティスになったと思います

ToolsもプロンプトエンジニアリングではなくPlugin的なアプローチになっていくのだとしたら、単純な文字列処理やモデリングの枠に収まらないので、今後どうなっていくのかは要注目ですね

GPT4遅い割には賢いようには見えない

遅いのは同意です(贅沢な話なんですけどね…w)

賢いかどうかでいうとベンチマークが出てるのでまあそうなのかなという感じなのと、GPT-3.5には答えられない質問がGPT-4では答えられることも稀にあるので、多分賢いんでしょうね。もはやパッと判断つかないレベルになっているな～と思います

あと、GPT-4の目玉はマルチモーダルらしいので、そこに期待しましょう

日本語処理は高いと聞いたのですが

スライド39をどうぞ

破産防止はしていますか？

スライド42をご覧ください

利用事例の分析に関しての取り組みの可能性はありますか？

あると思います。ペルソナを想定して感想を言わせたり、みたいなのを100人とか1000人でガッツと試せるので、ある種定量的な結果を出せるかもですね

ScrapboxとChatGPTがつながる？

めっちゃ可能性あると思います！

多分ユーザーさんがやってそう

個人的には色々試したりしてます(公開する予定は今のところありませんが)

「丁寧な言葉で質問したり褒めたりすると回答の質が上がる」のような言説は眉唾だなと思いつつ、「学習ソースの中で丁寧に質問された場合の回答は質が高いことが多いので理にかなっている」と説明されていて、「そういうものなのか…？」となんとも判断つかない状態です。こういう眉唾の情報は皆さんどのように受け止めていますか？

—

眉唾ですね。ただ、それを棄却するための実験を行うのは骨が折れるので、眉唾だな〜と心で思いながらスルーしています。

こういう情報を鵜呑みにしてプロンプトを書くべきではないですが、サンプルケースを一定数用意して実験して正答率があがるならそれでいいと思います。

ChatGPTがデータベースだったら他のデータリソースを元に回答するみたいなことはすでにできるのでしょいか？

—

他のデータリソースから文章を持ってきて、それをプロンプトに与えて回答させるみたいなことは可能ですね。スライド24をご覧ください

「『perplexity AI』ってどう思う？」ってCGPTに聞いたらめちゃくちゃ早口で「perplexity」について語り出して草

—

ChatGPT、稀にヲタク特有の早口みたいな喋り方しますよねw

勝手に親近感湧いてます

来年度の業計のアイデア壁打ちにCGPTを使ってるが、CGPTの回答ママで業計作ってそれに従って企業が動けば、疑似シンギュラリティか？

—

appendix

向いてそうなユースケース

非リアルタイムかつ大規模のユースケース

例えば…

- 大量の書類ファイルにインデックスを付けたい
- 大量の議事録から自分に関係のある記述を見つけない
- 新人教育に使えるケーススタディを自動生成したい

