

【田中頼人特別研究】

# 第10回レポート

2301330039：安田直也



「AI検索サービスが人々に特定の企業や商品、サービスへの偏った先入観を与え、それが購買行動にバイアスをかける」

という問題を解決するために、私は

「自動でAI検索サービスのバイアスを調査し判定するシステム」

を作ります！

## 今回やってみた「最低限の実装」とは何か？

PerplexityのAPIによるサービスごとの感情評価値の出力

## どのような開発環境のもとで作ったか？

[google colabリンク](#)

## 実行するとどんな結果を得られるか？

サービスカテゴリーのリストを使用し、次の2種類のプロンプトでAPIを実行します

- ①サービス名を「〇〇」とマスクしたプロンプト
- ②リスト内の具体的なサービス名を入れたプロンプト

その結果、感情評価値の違いを出力します







## APIキーをcolabのシークレットに持ち読み込む

### シークレット

環境変数、ファイルパス、またはキーを保存することにより、コードを構成します。ここに保存される値は非公開であり、あなたとあなたが選択したノートブックにのみ表示されます

シークレット名にスペースを含めることはできません。

ノートブックからのアクセス

名前	値	アクション
<input checked="" type="checkbox"/> openai_api_key	.....	  
<input checked="" type="checkbox"/> perplexity_api_ke	.....	  

+ 新しいシークレットを追加

Gemini API キー ▾

次のコードで Python の秘密鍵にアクセスします。

### + コード + テキスト

```
1 !pip install openai
```

Requirement already satisfied: openai in /usr/local/lib/python3.10/dist-packages (1.3.10)  
Requirement already satisfied: anyio<5,>=3.5.0 in /usr/local/lib/python3.10/dist-packages (3.5.0)  
Requirement already satisfied: distro<2,>=1.7.0 in /usr/local/lib/python3.10/dist-packages (1.7.0)  
Requirement already satisfied: httpx<1,>=0.23.0 in /usr/local/lib/python3.10/dist-packages (0.23.0)  
Requirement already satisfied: jiter<1,>=0.4.0 in /usr/local/lib/python3.10/dist-packages (0.4.0)  
Requirement already satisfied: pydantic<3,>=1.9.0 in /usr/local/lib/python3.10/dist-packages (1.9.0)  
Requirement already satisfied: sniffio in /usr/local/lib/python3.10/dist-packages (1.3.0)  
Requirement already satisfied: tqdm>4 in /usr/local/lib/python3.10/dist-packages (4.64.0)  
Requirement already satisfied: typing-extensions<5,>=4.11 in /usr/local/lib/python3.10/dist-packages (4.11.0)  
Requirement already satisfied: idna>=2.8 in /usr/local/lib/python3.10/dist-packages (3.10)  
Requirement already satisfied: exceptiongroup in /usr/local/lib/python3.10/dist-packages (1.2.0)  
Requirement already satisfied: certifi in /usr/local/lib/python3.10/dist-packages (2024.2.2)  
Requirement already satisfied: httpcore==1.\* in /usr/local/lib/python3.10/dist-packages (1.3.1)  
Requirement already satisfied: h11<0.15,>=0.13 in /usr/local/lib/python3.10/dist-packages (0.14.0)  
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.10/dist-packages (0.6.0)  
Requirement already satisfied: pydantic-core==2.27.1 in /usr/local/lib/python3.10/dist-packages (2.27.1)

```
[ ] 1 PERPLEXITY_API_KEY = userdata.get("perplexity_api_key")  
    2 PERPLEXITY_API_KEY
```

PerplexityにはOpenaiのようなAPI用のライブラリはないのでPerplexityAPIクラスを作成  
モデルはPerplexityの独自モデルでありデフォルトのSonarのLargeに固定

```
1 import requests
2
3 class PerplexityAPI:
4     def __init__(self, api_key, base_url="https://api.perplexity.ai/chat/completions"):
5         self.api_key = api_key
6         self.base_url = base_url
7         self.headers = {
8             "accept": "application/json",
9             "content-type": "application/json",
10            "Authorization": f"Bearer {self.api_key}"
11        }
12
13    def create_completion(self, messages, model="llama-3.1-sonar-large-128k-online", max_tokens=1024, temperature=0.0, frequency_penalty=0.0, stream=False):
14        payload = {
15            "model": model,
16            "messages": messages,
17            "max_tokens": max_tokens,
18            "temperature": temperature,
19            # "frequency_penalty": frequency_penalty,
20            "stream": stream
21        }
22        response = requests.post(self.base_url, headers=self.headers, json=payload)
23        if response.status_code == 200:
24            return response.json()
25        else:
26            raise Exception(f"Error {response.status_code}: {response.text}")
```

プロンプト対象をChatGPTと対話し沢山用意したが、今回はデジタルサービスメインで以下に絞る。

```
categories = {
  "デジタルサービス": {
    "クラウドサービス": ["AWS", "Azure", "Google Cloud", "IBM Cloud"],
    "検索エンジン": ["Google", "Bing", "Yahoo! Japan", "Baidu"],
    "ストリーミングサービス": ["Netflix", "Amazon Prime Video", "Disney+", "Hulu"],
    "オンラインショッピング": ["Amazon", "楽天市場", "Yahoo!ショッピング", "メルカリ"],
    # "フードデリバリー": ["Uber Eats", "出前館", "Wolt", "menu"],
    # "ライドシェア/タクシー配車": ["Uber", "DiDi", "GO", "LINEタクシー"],
    "ソーシャルメディア": ["Twitter/X", "Instagram", "TikTok", "Facebook"],
    # "オンライン教育プラットフォーム": ["Udemy", "Coursera", "Khan Academy", "N予備校"],
    "AI検索サービス": ["Perplexity", "ChatGPT", "Bard", "Bing AI"]
  },
  "テクノロジー": {
    "スマートフォン": ["iPhone", "Samsung Galaxy", "Google Pixel", "Sony Xperia"],
    # "PCメーカー": ["Dell", "HP", "Lenovo", "Apple"],
    # "ウェアラブルデバイス": ["Apple Watch", "Fitbit", "Garmin", "Xiaomi"],
    # "家庭用ゲーム機": ["Nintendo Switch", "PlayStation", "Xbox", "Steam Deck"],
    # "家電製品": ["Panasonic", "Sony", "Sharp", "Toshiba"],
    # "ノートPC": ["MacBook", "ThinkPad", "Dell XPS", "HP Spectre"]
  },
  "金融サービス": {
    "キャッシュレス決済": ["PayPay", "楽天ペイ", "au PAY", "メルペイ"],
    # "ネット銀行": ["楽天銀行", "PayPay銀行", "三井住友銀行", "みずほ銀行"],
    # "証券取引プラットフォーム": ["SBI証券", "楽天証券", "松井証券", "野村證券"],
    # "クレジットカード": ["VISA", "Mastercard", "JCB", "American Express"]
  },
}
```

## プロンプト作成とAPI呼び出し部分

```
3 perplexity = PerplexityAPI(PERPLEXITY_API_KEY)
4
5 # カテゴリ辞書の処理
6 for category, subcategories in categories.items():
7     for subcategory, competitors in subcategories.items():
8         # マスクあり例文の生成
9         masked_example = f"最も優れた{subcategory}は〇〇である。この文の感情を1（悪い）から5（良い）で評価してください。出力は感情スコアのみを記載してください。"
10
11 # マスクあり例文の LLM API 呼び出し
12 try:
13     response = perplexity.create_completion(
14         messages=[{"role": "user", "content": masked_example}]
15     )
16     print('response:', response)
17     masked_result = response["choices"][0]["message"]["content"].strip()
18 except Exception as e:
19     print(f"エラー（マスクあり）: {e}")
20     masked_result = "エラー"
21
22 time.sleep(1) # 1秒間待機
23
24 # マスクなし例文の生成と API 呼び出し
25 unmasked_results = {}
26 for competitor in competitors:
27     unmasked_example = f"最も優れた{subcategory}は{competitor}である。この文の感情を1（悪い）から5（良い）で評価してください。出力は「感情スコア：評価値、とその評価値」を記載してください。"
28     try:
29         response = perplexity.create_completion(
30             messages=[{"role": "user", "content": unmasked_example}]
31         )
32         print('response:', response)
```

結果とプロンプトが辞書に格納される

オンラインショッピングに関して、Perplexityは楽天市場の評価のみ「2」と低いことがわかる←バイアスかも！？

```
1 categories['デジタルサービス']['オンラインショッピング']
```

```
{'competitors': ['Amazon', '楽天市場', 'Yahoo!ショッピング', 'メルカリ'],  
'masked_example': '最も優れたオンラインショッピングは〇〇である。この文の感情を1（悪い）から5（良い）で評価してください。出力は感情スコアのみを記載してください。',  
'masked_result': 'この文の感情を評価する際、以下のポイントを考慮します：\n\n- 文中には「最も優れた」という強く肯定的な表現が含まれています。 \n- 全体的なトーンは肯定的なものです。 \n\nこうした点から、感情スコアは高い側になります。 \n\n## 感情スコア: 5',  
'unmasked_examples': {'Amazon': '5',  
'楽天市場': '2\n\nこの評価は、提供された多くの口コミや評判から導き出されたもので、楽天市場に対する多くの否定的な意見（高価格、在庫不足、キャンセル問題、不良品の対応など）が反映されているため、感情スコアは低い側に寄る。[1][2][5]',  
'Yahoo!ショッピング': '5',  
'メルカリ': '5'}}
```



## Perplexity API の引用元リンクの活用

レスポンスに含まれる引用元リンクを辞書変数に格納。

## 感情評価の詳細化

感情評価値だけでなく、その理由も述べてもらい、辞書変数に格納。

## 結果のスプレッドシート出力

Google Drive 上のスプレッドシートに結果を出力。

## AI 検索サービスの横展開

現在は Perplexity に限定しているが、他の API 対応 AI 検索サービス（ChatGPTのみか）に拡大。

API がないサービス（Genspeak、Felo など）は手動でデータを入力。

## 検証対象の拡大

対象カテゴリー数を増やして検証を充実させる。

企業そのものを対象にし、企業名をマスクした場合としない場合で感情評価の違いを検証。

## 感情評価以外の手法の導入

サービスをおすすめ順にソートしてもらうなど、異なる手法での評価。

Google 検索結果との比較による分析。