

回答根拠確認チュートリアル（推論を用いた回答根拠確認）

はじめに

本書は、本サービスが提供する回答根拠確認APIによって、LLMの生成結果の回答根拠を確認するためのチュートリアルです。

本書の対象読者は以下を想定しています。

- 本サービスと連携したシステムや製品開発を行う開発者

なお、本書ではPythonのサンプルコードを記載しておりますが、本サービスのAPIはREST形式のため、他の言語からのご利用いただけます。

サポートしているAPIとリクエスト・レスポンスの詳細については、APIリファレンスを参照してください。

- 本ガイドに記載する「<https://<サーバのドメイン名>/>」はGenerative AI FWがインストールされているサーバのドメイン名に置き換えてAPIを実行してください。
- 本サービスではHTTPSに既定では自己証明書を使用しています。そのためAPI利用時に考慮が必要です。詳細は「スタートアップマニュアル（導入準備編）」をご確認ください。

回答根拠確認APIとは

LLM が生成したテキストと、テキスト生成時に情報源としたテキストを入力とすることで、LLMが生成したテキストが情報源に含まれるか・どの部分が根拠となっているのか・その判断理由を確認するAPIです。

推論を用いない回答根拠確認では、紐づく箇所・関連度のみを確認できますが、推論を用いた回答根拠確認では「情報源に含まれない」という判定や「情報源に含まれない」・「根拠あり」の判定に対する根拠を推論によって出力することができます。

回答根拠確認を使用することでLLMの要約結果の回答根拠の確認、検索対話の回答根拠の確認を行うことができます。

チュートリアルの流れ

本チュートリアルの流れは以下です。

1. 要約結果の回答根拠確認
2. 検索対話の回答根拠確認
3. (応用) requestsライブラリを利用したpythonプログラムで回答根拠確認を利用する

要約結果の回答根拠確認

要約結果と要約前のテキストを入力とすることで、要約結果の回答根拠を確認することができます。

APIの実行方法

以下のcurlコマンドを用いて、APIを呼び出します。APIの詳細は「Generative AI FW APIリファレンス」を参照してください。

```
1 curl -X POST https://<サーバのドメイン名>/genai-api/v1/entailment ¥
2 -H "Content-Type: application/json" ¥
3 -d "<リクエストパラメータ>" ¥
4 -H "Authorization: <API Key>"
```

以下はリクエストパラメータの例です。answerには要約後のテキスト、sourcesには要約前のテキストを指定します。

```
1 {
2   "answer": "NECは、日本語大規模言語モデルを開発した。このLLMは業務特化モデルである。",
3   "sources": [
4     {
5       "text": "NECは、Generative AI(生成AI)における日本語大規模言語モデル(Large Language Model、以下LLM)を開発"
6     }
7   ]
8 }
```

応答がJson形式で返却されます。explainContentsには入力したテキストとそのテキストに紐づけされた回答根拠となるテキストが返却されます。

```
1 {"explainContents": [{"answer": "NECは、日本語大規模言語モデル...", "...モデルです。"}]}
```

回答根拠の見方

```
1 {
2   "explainContents": [
3     {
4       "answer": "NECは、日本語大規模言語モデルを開発した。",
5       "label": "entailment",
6       "reasoning": "生成文の内容は、NECが日本語大規模言語モデルを開発したという点で、ドキュメントの内容と一致"
7       "evidence": "NECは、Generative AI(生成AI)における日本語大規模言語モデル(Large Language Model、以下LLM)を"
8     },
9     {
10      "answer": "このLLMは業務特化モデルである。",
11      "label": "contradiction",
12      "reasoning": "生成文はLLMが業務特化モデルであると述べているが、ドキュメントでは汎用的なモデルであると述"
13      "evidence": "NECが開発した汎用的なモデル、いわゆるファウンデーションモデルです。"
14    }
15  ]
16 }
```

各パラメータの詳細は以下の通りです。

パラメータ	説明
-------	----

explainContents	answer	LLMの回答内容。リクエストパラメータのanswerを文単位に分けた内容。
	label	含意か矛盾かを表す。 含意：entailment 矛盾：contradiction 情報がない場合はなし
	reasoning	判断理由。 labelがある場合このパラメータが存在する。（ただし、Explainer側で空文字が返ってきた場合は空文字となる。） 情報がない場合はなし
	evidence	根拠情報。 labelがある場合このパラメータが存在する。（ただし、Explainer側で空文字が返ってきた場合は空文字となる。） 情報がない場合はなし
	metadata	根拠となるソースドキュメントのメタデータ。根拠情報がない、もしくは根拠情報に紐づくソースドキュメントのメタデータがない場合はなし リクエストで与えた値と同値

検索対話の回答根拠確認

検索対話の回答のテキストと、検索対話に使われた情報源のテキストを入力とすることで検索対話の回答根拠を確認することができます。

APIの実行方法

以下のcurlコマンドを用いて、APIを呼び出します。APIの詳細は「Generative AI FW APIリファレンス」を参照してください。

```
1 curl -X POST https://<サーバのドメイン名>/genai-api/v1/entailment ¥
2 -H "Content-Type: application/json" ¥
3 -d "<リクエストパラメータ>" -H "Authorization: <API Key>"
```

以下はリクエストパラメータの例です。answerには検索対話の回答、sourcesのtextに検索対話の応答のsourceDocumentsのdocContent、metadataにsourceDocumentsのmetadataを指定します。

```
1 {
2   "answer": "NECは、高い日本語性能を有する軽量なLLM(Large Language Model : 大規模言語モデル)「cotomi」を...",
3   "sources": [
4     {
5       "text": "「NEC Digital Platform」に「cotomi」は含まれます。",
```

```

6     "metadata": {
7         "url": "https://jpn.nec.com/..."
8     }
9 },
10 {
11     "text": "NECは、高い日本語性能を有する軽量なNEC開発のLLM(Large Language Model : 大規模言語モデル)...",
12     "metadata": {
13         "url": "https://jpn.nec.com/..."
14     }
15 }
16 ]
17 }

```

応答がJson形式で返却されます。

```

1 {explainContents:{"answer":"","...}}

```

requestsライブラリを利用したpythonプログラムで回答根拠確認を利用する

requestsライブラリのインストール

インストールコマンドの例

```

1 pip install requests

```

Pythonコードを実装する

Pythonコードの例

Pythonのエディタを開き、下記のコードを記述します。ファイル名を「entailment_main.py」として保存します。

```

1 import requests
2
3 # KEYはAPIキー
4 KEY = 'abcdefg1234567890'
5 # BASEはサービスのURL
6 BASE = 'https://<サーバのドメイン名>/genai-api/v1'
7
8 def func():
9     url = BASE + '/entailment'
10    key = 'Bearer ' + KEY
11    payload = { "answer": "NECは、日本語大規模言語モデルを開発した。このLLMは業務特化モデルである。",
12                "sources": [
13                    { "text": "NECは、Generative AI(生成AI)における日本語大規模言語モデル(Large Language Model)"
14                }]
15    headers = { 'content-type': 'application/json',
16                'Authorization': key }
17    # リクエスト
18    response = requests.post(url, json=payload, headers=headers)
19    print(response.status_code)
20    print(response.text)
21    #print(response.headers)
22
23 if __name__ == '__main__':
24     func()
25

```

Pythonの実行

実行コマンド

```
1 python entailment_main.py
```

実行結果の例

```
1 200  
2 {explainContents:[{"answer":"", ...}]}
```