Studio Ousia QA ENGINE / アンサーロボ API仕様書

1.1. 概要

Studio Ousia のQA ENGINE およびアンサーロボ WebAPIの仕様について記述します。

QA ENGINEは「回答候補」と過去にどのような質問に対してどの回答候補を用いて回答したかという「教師データ(アノテーション)」の二種類のデータに基づき、未知の質問に対して蓄積された回答候補の合致度を算出するエンジンです。

アンサーロボは「FAQデータ」に基づき、未知の質問に対して登録されたFAQデータの合致度を算出するエンジンです。

各WebAPIのレスポンス本文(body)については特に記載がない場合JSONオブジェクトを返却するものとします。レスポンスのJSONデータの例の記述に際してはわかりやすいようにスペースとインデントを加えて記載しておりますが実際にWebAPIから返却されるレスポンス本文はスペースやインデントのない形式となります。

1.2 API エンドポイント

(QA ENGINEの場合) QA ENGINE管理画面の「モデル管理」ページにおいて、「学習/本番コピー」を行っていただくと自動的に作成されます。 ご利用いただけるAPIエンドポイントはステージングAPIと本番APIの2つがあります。 最初の学習でステージングAPIのAPIエンドポイントが割り当てられ、最初の本番コピーで本番APIのエンドポイントが割り当てられます。

学習/本番コピーが完了しますと、「モデル管理」ページの上部に「エンドポイント」が表示されます。(ドメイン名)

(アンサーロボの場合) アンサーロボ管理画面の「Model Manager」の「アンサーロボ　モデル反映管理」ページにおいて、「反映する」ボタンを押下いただくと自動的に作成されます。 反映が完了しますと、「アンサーロボ　モデル反映管理」ページの上部に「API エンドポイント」が表示されます。

1.3 API キー

(QA ENGINEの場合) 管理画面の「モデル管理」ページにおいて、学習/本番コピーを行っていただくと自動的に作成されます。 学習が完了しますと、「モデル管理」ページの上部に「APIキー」が表示されます。

(アンサーロボの場合) アンサーロボ管理画面の「Model Manager」の「アンサーロボ　モデル反映管理」ページにおいて、「反映する」ボタンを押下いただくと自動的に作成されます。 反映が完了しますと、「アンサーロボ　モデル反映管理」ページの上部に「APIキー」が表示されます。

(共通) HTTPリクエストを行う際に、"X-API-Key” というHTTPヘッダの値として送信ください。

1.4 WebAPI: QAクエリに関するAPI

QA ENGINE / アンサーロボのメインのAPIです。クエリ文字列を入れると人工知能がクエリに対してふさわしい回答をデータベースにある回答候補の中からランキングし、返します。

URLエンドポイント: `/api/query` 対応HTTPメソッド： `GET` , `POST`

必須HTTPヘッダー：

- `X-API-Key` : APIキー認証に用います。このヘッダの値が正しくない場合、403 Forbiddenエラーを返します。
  - セクション1.2 "API キー” でご案内させて頂いているAPIキーをご利用ください。

必須パラメータ：

- `query` : UTF-8エンコードされた文字列、回答候補をクエリする質問文(例: `〝退会をしたいのですがやり方がわかりません〞`)

オプションパラメータ：

- `top_n`：数字、ランキング上位何件を返すかを指定。指定しない場合、10件になります。(例: `"5"`, `"20"`)
- `include_title`：`true` でオプションをオンにします。このオプションをオンにすると、レスポンスにFAQ IDで紐づいたFAQタイトルが含まれるようになります。指定しない場合にはオプションはオフになります。また、明示的にオプションをオフにする場合には `false` を指定します。
- `include_query_uuid`：`true` でオプションをオンにします。このオプションをオンにすると、レスポンスにリクエストのクエリのUUIDが含まれるようになります。指定しない場合にはオプションはオフになります。また、明示的にオプションをオフにする場合には `false` を指定します。
- `tags`：タグによる絞り込みでレスポンス中に出現するFAQを制限することができます。半角スペース区切りで複数指定できます。複数指定した場合はデフォルトで `or` の挙動になります(指定したタグのどれかが紐付けられているFAQのみ出現するようになります)。後述の `tag_filter_mode` パラメータで `and` を指定することもできます。 `and` を指定した場合は指定した複数のタグすべてに紐付けられたFAQのみ出現するようになります。指定しない場合はすべてのFAQがスコアによるランキング対象になります。(注意: 2022/04/01以降に学習(QA ENGINE)/FAQ反映(アンサーロボ)されたAPIのみでサポートされている機能になります)
- `tag_filter_mode`：セットできる値は `and` または `or` のみ有効です。 `tags` パラメータを使う場合のみ意味があります。

レスポンスHTTPステータスコード：

- `200`：正常に回答候補を算出できた場合
- `400`：入力したパラメータに問題がある場合
  - `query` の入力が空、もしくは指定されていない
  - `top_n` にパラメータがセットされていたが、数字として認識できない入力だった
  - `include_title` の入力が許容されている入力以外の物が与えられている。
  - `include_query_uuid` の入力が許容されている入力以外の物が与えられている。
  - 2022/04/01以前に学習(QA ENGINE) / FAQ反映(アンサーロボ)されたAPIに対して `tags` オプションを使おうとした
- `403`：X-API-Keyによる認証が行えない場合
- `500`：サーバサイドで予期せぬエラーが発生した場合。(DBがダウンしていた場合など)

ステータスコード200時レスポンス：

- `result` オブジェクトには `top_n` 属性と `answer_candidates` 属性がセットされます。
- `top_n` 属性は `top_n` パラメータで指定した数が入ります。指定がない場合はデフォルト値の `10` がセットされます。
- `answer_candidates` 属性はAPIのメインの返り値である、ランキングされた回答候補がスコアの高い順でリストになったものがセットされます。
- リストの各要素はJSONオブジェクトになっており、それぞれの要素は `score` 属性と `answer_candidate` 属性を持ちます。
- `score` 属性は浮動小数点の数値が入ります。
  - AIが算出したクエリに対して該当の回答候補に対する合致度です。 `answer_candidates` 属性のリストはスコアが高い順に並んでいます。
- `answer_candidate` 属性は `answer_candidate_id` と `text` の2つの属性が必ず含まれています。
  リクエスト時の `include_title` オプションがオンの場合、加えて `faq_title` 属性が追加されます。
  リクエスト時の `include_query_uuid` オプションがオンの場合、加えて `query_uuid` 属性が追加されます。
  - `anser_candidate_id`：回答候補のIDが入ります。例: `"回答番号0001"`
  - `text`：回答候補の本文が入ります。

    - `faq_title`：`answer_candidate_id` とFAQ IDで紐づいたFAQタイトルが入ります。リクエスト時に `include_title` がオフになっていた場合、この属性はレスポンスには含まれません。FAQタイトルの用途の一例として、エンドユーザーへ適切な質問をサジェスチョンするなどがあります。
      FAQタイトルはリクエストを受けてから1時間キャッシュされますので、FAQタイトルを変更した場合には、1時間後にレスポンスに反映されます。
      また、クエリがリクエストされた時のFAQ IDが `answer_candidate_id` と違っていた場合、機械学習実行時の回答候補のFAQタイトルが入ります。この機械学習実行時の回答候補のFAQタイトルは、こちらのオプションがリリースされた日時以降に学習を実行していない場合には含まれません。その際はFAQタイトルには空文字が入ります。
      運用上はFAQを減らす、FAQ IDを変更するなどの操作を行っていなければ、あまり影響はございません。
    - `query_uuid`：リクエストのクエリのUUIDが入ります。
- `tags` 属性はオプションパラメータ `tags` で指定された絞り込みに使われた `tags` の値がセットされます。(呼び出し時に `tags` パラメータがセットされた場合のみセットされます)
- `tag_filter_mode` 属性は呼び出し時に `tags` パラメータがセットされた場合のみセットされます。オプションパラメータ `tag_filter_mode` で指定された場合はその値が、指定されなかった場合は `"or"` がセットされます。

ステータスコード200時レスポンス例（デフォルト、 `include_title` 、 `include_query_uuid` オプションがオフ）

```
{
     "status": "ok",
     "result": {
          "top_n": (top_nパラメータで指定した数, 指定しない場合10),
          "answer_candidates": [
               {
                    "score": 0.25,
                    "answer_candidate": {
                         "answer_candidate_id": (回答候補ID, 例: "回答番号0001"),
                         "text": (回答候補本文)
                    }
               },
               {
                    "score": 0.12,
                    "answer_candidate": {
                         ...
                    }
               },
               ...
          ]
     }
}
```

ステータスコード200時レスポンス例（ `include_title` オプションがオン）

```
{
     "status": "ok",
     "result": {
          "top_n": (top_nパラメータで指定した数, 指定しない場合10),
          "answer_candidates": [
               {
```

```
                    "score": 0.25,
                    "answer_candidate": {
                         "answer_candidate_id": (回答候補ID, 例: "回答番号0001"),
                         "faq_title": (FAQ タイトル),
                         "text": (回答候補本文)
                    }
               },
               {
                    "score": 0.12,
                    "answer_candidate": {
                         ...
                    }
               },
               ...
          ]
     }
}
```

ステータスコード200時レスポンス例（ `include_query_uuid` オプションがオン）

```
{
     "status": "ok",
     "result": {
          "top_n": (top_nパラメータで指定した数, 指定しない場合10),
          "answer_candidates": [
               {
                    "score": 0.25,
                    "answer_candidate": {
                         "answer_candidate_id": (回答候補ID, 例: "回答番号0001"),
                         "text": (回答候補本文)
                    }
               },
               {
                    "score": 0.12,
                    "answer_candidate": {
                         ...
                    }
               },
               ...
          ],
          "query_uuid": (クエリのUUID)
     }
}
```

ステータスコード200時レスポンス例（ `tags` オプションに `tag1` を, `tag_filter_mode` に `and` を指定した場合）

```
{
     "status": "ok",
     "result": {
          "top_n": (top_nパラメータで指定した数, 指定しない場合10),
          "answer_candidates": [
```

```
                {
                     "score": 0.25,
                     "answer_candidate": {
                          "answer_candidate_id": (回答候補ID, 例: "回答番号0001"),
                          "text": (回答候補本文)
                     }
                },
                {
                     "score": 0.12,
                     "answer_candidate": {
                          ...
                     }
                },
                ...
           ],
           "tags": ["tag1"],
           "tag_filter_mode": "and",
           "query_uuid": (クエリのUUID)
     }
}
```

ステータスコード400時レスポンス：

```
{ "status": "error", "result": null, "message": <エラーに関する情報> }
```

ステータスコード403時レスポンス：

```
{ "status": "error", "result": null, "message": <エラーに関する情報> }
```

ステータスコード500時レスポンス：

```
{ "status": "error", "result": null, "message": <エラーに関する情報> }
```

(付録)サンプル

GETメソッドの動作を確認

```
$ curl -H 'X-API-Key: QAENGINE_API_KEY' 'https://END_POINT/api/query?query=退会したいです。'
```

POSTメソッドの動作を確認

```
$ curl -H 'X-API-Key: QAENGINE_API_KEY' -F 'query=退会します。'
'https://END_POINT/api/query'
```

GETメソッドの動作を確認（include_titleオプションを使用する）

```
$ curl -H 'X-API-Key: QAENGINE_API_KEY' 'https://END_POINT/api/query?query=退会したいです。&include_title=true'
```

POSTメソッドの動作を確認（include_titleオプションを使用する）

```
$ curl -H 'X-API-Key: QAENGINE_API_KEY' -F 'query=退会します。' -F 'include_title=true'
'https://END_POINT/api/query'
```

GETメソッドの動作を確認（include_query_uuidオプションを使用する）

```
$ curl -H 'X-API-Key: QAENGINE_API_KEY' 'https://END_POINT/api/query?query=退会したいです。&include_query_uuid=true'
```

POSTメソッドの動作を確認（include_query_uuidオプションを使用する）

```
$ curl -H 'X-API-Key: QAENGINE_API_KEY' -F 'query=退会します。' -F 'include_query_uuid=true' 'https://END_POINT/api/query'
```

GETメソッドの動作を確認(tagsオプションとtag_filter_modeオプションを使用する)

```
$ curl -H 'X-API-Key: QAENGINE_API_KEY' 'https://END_POINT/api/query?tags=tag1&tag_filter_mode=or&query=退会したいです。'
```

POSTメソッドの動作を確認(tagsオプションとtag_filter_modeオプションを使用する)

```
$ curl -H 'X-API-Key: QAENGINE_API_KEY' -F 'query=退会します。' -F 'tags=tag1' -F 'tag_filter_mode=or' 'https://END_POINT/api/query'
```

2. バッチ予測API: 複数のクエリを一括分類するAPI

バッチ予測APIは複数のクエリ文字列に対して、まとめて回答候補の予測を行うAPIです。メインのAPI（ `api/query` ）は一つのクエリに対して回答候補を返却するので、たくさんのクエリを予測したい場合はクエリの数だけAPIを呼び出す必要があります。これに対して、バッチ予測APIは一回のAPI呼び出しで最大1000個までのクエリの分類を行うことができます。

URLエンドポイント: `/api/batch` 対応HTTPメソッド： `POST`

必須HTTPヘッダー：

- `X-API-Key` : APIキー認証に用います。このヘッダの値が正しくない場合、403 Forbiddenエラーを返します。
  - セクション1.2 "API キー” でご案内させて頂いているAPIキーをご利用ください。
- `Content-Type` : `application/json` に設定します。APIに送るデータは `json` で記述し、結果も `json` で返却されます。

入力JSONデータの構造：

```
{
    "queries": [
        "文字列", ...
    ],
    "top_n": 123,
    "tags": ["tag1", ...],
    "tag_filter_mode": "(andまたはor)"
}
```

必須パラメータ：

- `queries` : 分類したいクエリの配列。リストの長さは最大で1000まで。

オプションパラメータ：

- `top_n` : 数字、ランキング上位何件を返すかを指定。指定しない場合、10件になります。(例: `“5”` , `“20”`)

- `tags` : タグによる絞り込みでレスポンス中に出現するFAQを制限することができます。半角スペース区切りで複数指定できます。複数指定した場合はデフォルトで `or` の挙動になります(指定したタグのどれかが紐付けられているFAQのみ出現するようになります)。後述の `tag_filter_mode` パラメータで `and` を指定することもできます。 `and` を指定した場合は指定した複数のタグすべてに紐付けられたFAQのみ出現するようになります。指定しない場合はすべてのFAQがスコアによるランキング対象になります。(注意: 2022/04/01以降に学習(QA ENGINE)/FAQ反映(アンサーロボ)されたAPIのみでサポートされている機能になります)
- `tag_filter_mode` : セットできる値は `and` または `or` のみ有効です。 `tags` パラメータを使う場合のみ意味があります。

レスポンスHTTPステータスコード：

- `200` : 正常に回答候補を算出できた場合
- `400` : 入力したパラメータに問題がある場合
  - `Content-Type` が `application/json` ではない
  - `queries` の入力が空、もしくは指定されていない、配列の長さが最大長を超えている
  - `top_n` にパラメータがセットされていたが、数字として認識できない入力だった
  - 2022/04/01以前に学習(QA ENGINE) / FAQ反映(アンサーロボ)されたAPIに対して `tags` オプションを使おうとした
- `403` : X-API-Keyによる認証が行えない場合
- `500` : サーバサイドで予期せぬエラーが発生した場合。(DBがダウンしていた場合など)

ステータスコード200時レスポンス：

- `answer_candidate_texts` 回答候補ID(`answer_candidate_id`) から回答候補のテキストへのマッピングです。
- `predicted` 各質問に対する推論結果のリストです。 `top_n` が指定されていて、回答候補全体の長さよりも小さい場合、各質問の推論結果の長さは `top_n` となります。
- `tags` 属性はオプションパラメータ `tags` で指定された絞り込みに使われた `tags` の値がセットされます。(呼び出し時に `tags` パラメータがセットされた場合のみセットされます)
- `tag_filter_mode` 属性は呼び出し時に `tags` パラメータがセットされた場合のみセットされます。オプションパラメータ `tag_filter_mode` で指定された場合はその値が、指定されなかった場合は `"or"` がセットされます。
- 各質問に対する推論結果の各要素はJSONオブジェクトになっており、それぞれの要素は `answer_candidate_id` （回答候補ID）属性と `score` （回答候補のスコア）属性を持ちます。

ステータスコード200時レスポンス例

```
{
    "result": {
        "answer_candidate_texts": {
            "string": "回答候補のテキスト", ...
        },
        "predicted": [
            [
                {
                    "answer_candidate_id": "回答候補ID",
                    "score": 0.12
                }, ...
            ], ...
        ],
        "top_n": (top_nパラメータで指定した数, 指定しない場合10)
    },
```

```
    "status": "ok"
}
```

ステータスコード200時レスポンス例(`tags` パラメータに `tag1` を, `tag_filter_mode` パラメータに `and` を指定した場合)

```
{
    "result": {
        "answer_candidate_texts": {
            "string": "回答候補のテキスト", ...
        },
        "predicted": [
            [
                {
                    "answer_candidate_id": "回答候補ID",
                    "score": 0.12
                }, ...
            ], ...
        ],
        "top_n": (top_nパラメータで指定した数, 指定しない場合10),
        "tags": ["tag1"],
        "tag_filter_mode": "and"
    },
    "status": "ok"
}
```

ステータスコード400時レスポンス：

```
{ “status”: “error”, “result”: null, “message”: <エラーに関する情報> }
```

ステータスコード403時レスポンス：

```
{ “status”: “error”, “result”: null, “message”: <エラーに関する情報> }
```

ステータスコード500時レスポンス：

```
{ “status”: “error”, “result”: null, “message”: <エラーに関する情報> }
```

(付録)サンプル：バッチAPI呼び出しの入力と対応する出力の例を以下に示します。

```
$ curl -X POST \
       -H 'Content-Type: application/json' \
       -H 'X-API-Key:<API_KEY>' \
       https://<ENDPOINT_PREFIX>.qaengine.ai/api/batch \
       -d @input.json
```

`input.json`

```
{
    "queries":[
            "質問のデータ形式を教えて",
            "データをどれくらい準備すればいいか",
            "アノテーションのデータ形式を教えてください"
```

```
        ],
    "top_n": 2
}
```

output.json

```
{
    "result": {
        "answer_candidate_texts": {
            "2": "もしも、同じタイプの質問に対して回答の内容だけを変更したい場合には、FAQ ManagerからFAQ管理画面に行っていただき、右の編集ボタンをクリックして回答内容を変更します。",
            "3": "質問のデータは、アップロードしていただきます。",
            "4": "質問データはヘルプデスクの過去の質問データがテキストファイルで存在する場合は、そのデータを使っていただくことができます。もしも過去の質問のデータがない場合には、想定される質問を記述していただくことでも代用できます。",
            "5": "アノテーションは各回答候補に対して最低１０個の質問データのアノテーションが付いていることが望ましいです。",
            "9": "回答候補のデータは、CSVでご用意いただき、FAQ Managerページの FAQデータアップロードのところからアップロードしていただきます。"
        },
        "predicted": [
            [
                {
                    "answer_candidate_id": "3",
                    "score": 0.9987479188192485
                },
                {
                    "answer_candidate_id": "4",
                    "score": 0.0005502702688846855
                }
            ],
            [
                {
                    "answer_candidate_id": "4",
                    "score": 0.6726147780707615
                },
                {
                    "answer_candidate_id": "5",
                    "score": 0.3272546378035233
                }
            ],
            [
                {
                    "answer_candidate_id": "3",
                    "score": 0.7956248204653308
                },
                {
                    "answer_candidate_id": "2",
                    "score": 0.12696617790203357
                }
            ]
        ],
```

```
        "top_n": 2
    },
    "status": "ok"
}
```

3. フィードバックAPI (アンサーロボのみ)

フィードバックAPIはクエリAPIの結果に対して役に立った, または役に立たなかったというフィードバックを記録することができるAPIです。このフィードバックAPIで投入したフィードバック情報は管理画面の集計画面の集計に反映されます。

URLエンドポイント: `/api/feedback` 対応HTTPメソッド: `GET` または `POST`

必須HTTPヘッダー：

- `X-API-Key`：APIキー認証に用います。このヘッダの値が正しくない場合、403 Forbiddenエラーを返します。
  - セクション1.2 "API キー” でご案内させて頂いているAPIキーをご利用ください。

必須パラメータ:

- `query_uuid`：フィードバックを記録したい対象のクエリのUUIDを文字列で指定します。クエリAPIの `include_query_uuid` オプションを有効にして取得した値をセットします。詳細はセクション 1.4 QAクエリに関するAPIをご覧ください。
- `is_worked`：クエリが役に立ったかどうかを "true" または "false" の文字列で指定します。クエリ結果が役に立ったというフィードバックを記録する場合は "true" を, 役に立たなかったというフィードバックを記録する場合は "false" を指定します。

オプションパラメータ:

- `faq_id`：役に立ったと思われるFAQ IDを文字列で指定します。

レスポンスHTTPステータスコード:

ステータスコード200時レスポンス:

```
{
     "status": "ok",
     "result": {
          "query_uuid": "(パラメータquery_uuidで入力した値, 文字列)",
          "is_worked": (パラメータis_workedで入力した値, boolean),
          "faq_id": "(パラメータfaq_idで入力した値, 文字列, パラメータfaq_idを指定しなかった場合はセットされない)"
     }
}
```

ステータスコード400時レスポンス:

```
{
     "status": "error",
     "message": "(エラーの原因)"
}
```

以下のような `message` がセットされることがあります:

- `"missing parameter: query_uuid"`：パラメータ `query_uuid` の値がセットされていない, または空文字列だった場合

- `"missing parameter: is_worked"` : パラメータ `is_worked` の値がセットされていない, または空文字列だった場合
- `"malformed input: query_uuid"` : パラメータ `query_uuid` の値が不正な値だった場合(32文字の小文字のUUID文字列を想定)
- `"malformed input: is_worked"` : パラメータ `query_uuid` の値が不正な値だった場合("true" または "false" を想定)
- `"invalid parameter: faq_id (too long)"` : パラメータ `faq_id` の値が 255文字より長かった場合
- `"invalid parameter: faq_id (empty)"` : パラメータ `faq_id` の値が空文字列だった場合

改訂履歴

- 2017-07-12 初版
- 2020-08-07 エンドポイント/APIキーの記述について現行システムに合わせて修正, バッチ予測API、`include_title` オプション、 `include_query_uuid` オプションの仕様を記載
- 2022-04-01 FAQに紐付けられたタグによる絞り込み機能についての説明を記述
- 2022-05-09 フィードバックAPIの仕様を追加