Dedupe.io web API documentation

Release 1.0

DataMade

Mar 18, 2020

Contents

1 Overview 1

2 When to use the web API 3

3 API access 5

HTTP Routing Table 13

CHAPTER 1

Overview

Dedupe.io is a a software as a service platform for quickly and accurately identifying clusters of similar records across one or more files or databases.

CHAPTER 2

When to use the web API

Once you have completed the de-duping process for a project, you can continue to incrementally check, match and add to it via API calls.

By posting a chunk of data to the API (described in the **match** endpoint), Dedupe.io can compare it to your dataset and return one or more potential matches. In the case where more than one result is returned, you can optionally tell Dedupe.io which one is correct and it will update the training for your dataset based on it (described in the **train** endpoint).

CHAPTER 3

API access

API access is available for Dedupe.io Enterprise customers. If you are interested in an Enterprise account, contact us at dedupe@datamade.us.

All methods are accessed via: **https://app.dedupe.io/api/v1/ENDPOINT**

3.1 projects/

POST /api/v1/projects/

Get a list of Dedupe.io projects associated with an account.

Query Parameters

- **api_key** – user API key
- **order_by** – field to sort on. Supports **name** and **date_added**. Defaults to **date_added**.
- **descending** – **true** or **false**. If true, sorts descending, otherwise sorts ascending.

Example request:

```
POST /api/v1/projects/ HTTP/1.1
Host: app.dedupe.io
Accept: application/json, text/javascript

{
  "api_key": "50b400ed-cc7f-4bbb-b16f-13dbdc022e91",
  "order_by": "date_added",
  "descending": "true"
}
```

Example response:

```
HTTP/1.1 200 OK
Vary: Accept
Content-Type: text/javascript
```

```
{
  "status": "ok",
  "projects": [
    {
      "id": "2ce57916-1d34-4ad5-bddb-f043ef701e45",
      "name": "Early Childhood Deduplication",
      "description": "",
      "date_added": "2020-01-09T19:11:48.656659+00:00",
      "date_updated": null,
      "datasets": [
        {
          "id": "db387e58-18b1-4663-b0b1-70f21cf9ad4f",
          "name": "Early Childhood Programs",
          "status": "canonical",
          "record_count": 3337,
          "entity_count": 1234,
          "review_count": 108,
          "description": "",
          "processing": false,
          "date_added": "2020-01-09T19:11:48.656659+00:00",
          "date_updated": "2020-01-09T19:18:54.001089+00:00",
          "field_mapping": {
            "abs_phone": [
              "phone"
            ],
            "abs_address": [
              "address",
              "zip"
            ],
            "abs_site_name": [
              "site_name"
            ]
          },
          "status_info": {
            "step": 5,
            "machine_name": "canonical",
            "human_name": "Dataset is de-duped",
            "next_step_name": "Dataset is de-duped!",
            "next_step_url": "/entity-browser/?project_id=db387e58-18b1-4663-b0b1-
↪70f21cf9ad4f",
            "next_step": null
          }
        },
        {
          "id": "05458c7d-a639-4532-9933-fdf8b97c4e66",
          "name": "File 2",
          "status": "model defined",
          "record_count": 800,
          "entity_count": null,
          "review_count": null,
          "description": "",
          "processing": false,
          "date_added": "2020-01-09T19:39:25.134078+00:00",
          "date_updated": "2020-01-09T19:40:38.591402+00:00",
          "field_mapping": {
            "abs_phone": [
              "phone"
```

```
            ],
            "abs_address": [
              "address",
              "zip"
            ],
            "abs_site_name": [
              "site_name"
            ]
          },
          "status_info": {
            "step": 1,
            "machine_name": "model defined",
            "human_name": "Model defined",
            "next_step_name": "Train model",
            "next_step_url": "/train-model/?dataset_id=05458c7d-a639-4532-9933-
↪fdf8b97c4e66",
            "next_step": 2
          }
        }
      ]
    },
    {
      "id": "49e21187-f091-496e-8b99-e92c24d4d332",
      "name": "Restaurant matching",
      "description": "Merge restaurant lists",
      "date_added": "2017-10-09T19:24:43.491660+00:00",
      "date_updated": null,
      "datasets": [
        {
          "id": "2cdf3d1d-ad2d-4f56-9cba-caf22d3e5b78",
          "name": "Restaurants 2",
          "status": "linked",
          "record_count": 752,
          "entity_count": 779,
          "review_count": 23,
          "description": "A second list of restaurants",
          "processing": false,
          "date_added": "2017-10-09T19:25:36.670140+00:00",
          "date_updated": "2017-10-09T19:29:58.523173+00:00",
          "field_mapping": {
            "abs_city": [
              "city_2"
            ],
            "abs_name": [
              "name_2"
            ],
            "abs_address": [
              "address_2"
            ],
            "abs_cuisine": [
              "cuisine_2"
            ]
          },
          "status_info": {
            "step": 4,
            "machine_name": "linked",
            "human_name": "Dataset is linked",
            "next_step_name": "Linked!",
```

```
            "next_step_url": "/entity-browser/?project_id=2cdf3d1d-ad2d-4f56-9cba-
↪caf22d3e5b78",
            "next_step": null
          }
        },
        {
          "id": "cb4b2c5d-8485-4395-9783-4cb69edb9bfa",
          "name": "Restaurants 1",
          "status": "canonical",
          "record_count": 112,
          "entity_count": null,
          "review_count": null,
          "description": "Names, addresses and cuisines for a list of restaurants
↪",
          "processing": false,
          "date_added": "2017-10-09T19:24:43.491660+00:00",
          "date_updated": "2017-10-09T19:24:46.103786+00:00",
          "field_mapping": {
            "abs_city": [
              "city"
            ],
            "abs_name": [
              "name"
            ],
            "abs_address": [
              "address"
            ],
            "abs_cuisine": [
              "cuisine"
            ]
          },
          "status_info": {
            "step": 5,
            "machine_name": "canonical",
            "human_name": "Dataset is de-duped",
            "next_step_name": "Dataset is de-duped!",
            "next_step_url": "/entity-browser/?project_id=cb4b2c5d-8485-4395-9783-
↪4cb69edb9bfa",
            "next_step": null
          }
        }
      ]
    }
  ]
}
```


3.2 match/

POST /api/v1/match/

Send one record to check for matches against a Dedupe.io project.

This endpoint is currently only available for completed (de-duplicated) projects with one uploaded file.

Query Parameters

- **api_key** – user API key

- **project_id** – identifier for project to match against
- **object** – dictionary of field values for one record. This must match the fields you selected when setting up your project. All field names will be lower cased and with no spaces.
- **num_results** – number of results to return (default: 5)
- **threshold** – minimum matching confidence score of results returned

Example request:

```
POST /api/v1/match/ HTTP/1.1
Host: app.dedupe.io
Accept: application/json, text/javascript

{
  "api_key": "50b400ed-cc7f-4bbb-b16f-13dbdc022e91",
  "project_id": "ebfc2317-7050-4e89-992c-56bcab13f1a1",
  "object": {
    "site_name": "Korean American Community Services",
    "address": "4300 North California Ave.   60618",
    "phone": "5838281"
  },
  "threshold": 0.8
}
```

Example response:

```
HTTP/1.1 200 OK
Vary: Accept
Content-Type: text/javascript

{
  "matches": [
    {
      "confidence": "1.0",
      "processed_record": {
        "abs_site_name": "korean american community services",
        "abs_address": "4300 north california ave. 60618",
        "abs_phone": "5838281",
        "record_id": 92
      },
      "cluster_id": "f44df274-3055-4aae-b8d2-f3680df37b4c",
      "raw_record": {
        "site_name": " Korean American Community Services ",
        "zip": "60618",
        "record_id": 92,
        "source": "NAEYC_accreditation.csv",
        "address": "4300 North California Ave.  ",
        "phone": "5838281",
        "fax": null
      }
    }
  ],
  "status": "ok"
}
```

The user will want to act based on the response of this API call in one of three ways.

1. **none** of the matches returned is correct - investigate and potentially add new product to canonical dataset

2. **one** of the matches returned is correct - the product should be associated with the proper ID
3. **more than one** of the matches returned is correct - the canonical database is not canonical and products should be merged

3.3 train/

POST /api/v1/train/

Send a tagged record to a Dedupe.io project for training.

This API call should only get zero or one positive matches. If more than one positive match is provided, it means the canonical database of products is not canonical and should be corrected on the client's side.

Query Parameters

- **api_key** – customer API key
- **project_id** – identifier for project to train
- **object** – original object to match
- **matches** – list of objects with a match flag attribute flagged by a human reviewer

Example request:

```
POST /api/v1/train/ HTTP/1.1
Host: app.dedupe.io
Accept: application/json, text/javascript

{
  "api_key": "50b400ed-cc7f-4bbb-b16f-13dbdc022e91",
  "project_id": "ebfc2317-7050-4e89-992c-56bcab13f1a1",
  "object": { "site_name": "Carole Robertson Center for Learning", "address":
↪"2929 w. 19th st. 60623", "phone": " "},
  "matches": [
    { "site_name": "Carole Robertson", "address": "2929 w. 19th st. 60623", "phone
↪": "5211600", "match": 1 },
    { "site_name": "Rob Robertson", "address": "2920 w. 19th st. 60623", "phone":
↪"5211600", "match": 0 },
    { "site_name": "Joseph Robertson", "address": "2929 w. 17th st.", "phone":
↪"5211600", "match": 0 }
  ]
 }
```

Example response:

```
HTTP/1.1 200 OK
Vary: Accept
Content-Type: text/javascript
```

3.4 Python example

This example sends one record to the *match* endpoint and prints out the resulting matches.

```
import requests
import json
```

```
# set your session IDs
API_KEY = 'YOUR API KEY'
PROJECT_ID = 'YOUR PROJECT ID'

# the field names in the match_object must match the field names in your session
match_object = {
  "name": 'john smith',
  "address": '222 W Merchandise Mart Plz, Chicago IL',
  "phone": "(555) 725-0195"
}

# post the
post_data = {
  'api_key': API_KEY,
  'project_id': PROJECT_ID,
  'threshold': 0.5, # set this to a value between 0 and 1 for how conservative the␣
↪returned matches should be
  'object': match_object
}

r = requests.post('https://app.dedupe.io/match/',
data=json.dumps(post_data))

# print the response from Dedupe.io
print(r.json())
```

HTTP Routing Table

/api

```
POST /api/v1/match/, 8
POST /api/v1/projects/, 5
POST /api/v1/train/, 10
```