OPENALT V2.0

User Manual

Document Version *1.0*

04/20/2021

Prepared by:

Darpan Shah

Mohammad Tahmid

Rihat Rahman

Salsabil Bakth

Tabish Shaikh

VERSION HISTORY

Version #	Implemented By	Revision Date	Approved By	Approval Date	Reason
1.0	*Team*	04/20/2021	*Team*	04/20/2021	Initial User Manual

Table of Contents

1. Install the Dependencies

1.1 Windows

1. Install Python 3.8.6 and add it to your PATH.
2. Install MySQL using the Windows Installer. Be sure to install the Python connector and workbench.
3. Use pip to install the needed Python modules. This command will install them all at once:
    - *pip install schedule crossrefapi flask virtualenv python-dateutil flask-paginate pytz*

1.2 Ubuntu

1. Install Python 3:
    - *sudo apt install Python3*
2. Python for Windows includes pip3 but on Ubuntu we need to install it with:
    - *sudo apt install python3-pip*
3. Install the first set of needed Python modules:
    - *pip3 install schedule crossrefapi flask virtualenv python-dateutil flask-paginate pytz*
4. Add the mysql apt repository (*https://dev.mysql.com/downloads/repo/apt/*) to your sources. You can use *dkpg -i* or just use Gnome Software Center to install it by double clicking it.
5. Update and install mysql-community-server:
    - *sudo apt update && sudo apt install mysql-community-server*
6. Check systemd to ensure the MySQL daemon is enabled and active:
    - *sudo systemctl status mysql ---- check that daemon is active and enabled*
7. Install the config will get the config you need for the last two pip modules:
    - *sudo apt-get install libmysqlclient-dev*
8. Install the last of the Python modules:
    - *pip3 install flask-mysqldb mysql-connector-python*
9. Install the mysql-connector:
    - *sudo apt install mysql-connector-python-py3*
10. Install MySQL shell:

    - *sudo apt install mysql-shell*
11. Go into mysql shell by executing this command: *mysql --user root -p* and then execute:
    - *SET GLOBAL sql_mode=(SELECT REPLACE(@@sql_mode,'ONLY_FULL_GROUP_BY',''));*

2. Setting Up the Databases

The Event data will be ingested into a MySQL database titled `crossRefEventDataMain`. The script to create it can be found at (https://github.com/darpanshah-wsu/openAlt_W2021/blob/master/SQL/CrossrefeventdataWithMain/crossrefeventdataWithMain.sql).

The journal, publisher, author, title, and date information is stored in a seperate MySQL database titled `doidata`. The script to create it can be found at (https://github.com/darpanshah-wsu/openAlt_W2021/blob/master/SQL/DOI_Author_Database/doidataSchema.sql).

Anyone can use our scripts and database schemas to create and fill in `crossRefEventDataMain`, but you will need to use other methods to fill in the needed fields for `doidata`. This GitHub repository (https://github.com/fabiobatalha/crossrefapi) is a good place to start.

The `OpenCitations` database can be created using the opencitationsSchema.sql that can be found at (https://github.com/darpanshah-wsu/openAlt_W2021/blob/master/SQL/OpenCitations/opencitationsSchema.sql).

The 'BulkSearchStats' database is necessary for the bulk search limitation to avoid user abuse of the system. The schema to create this database can be found at (https://github.com/darpanshah-wsu/openAlt_W2021/tree/master/SQL/BulkSearchStats).

3. Update Configs

1. After creation of the SQL tables, edit the config file to match your database credentials. The config file can be found in
    - (openAlt/*config/openAltConfig.json*)
2. In the project, update the config_path variable to match the directory of your config file. The variable can be found in the following files:
    - content_domain_ingest.py
    - fetchCitations.py
    - fetchOpenCitations.py
    - fetchEventBuffer.py
    - ingestMongoEvents.py
    - app.py
    - dbQuery.py
    - emailAdmin.py
    - uploadAuthor.py
    - uploadDOI.py
    - uploadUni.py

3. This can be easily done in VS Code by hitting Ctrl + Shift + F and searching and replacing the variable.

4. Collecting and Organizing the Events

Before we can run the web server, we also need to collect the data from the Crossref API. This will take some time, as there are millions of events to collect. We highly recommend reading Crossref's guide (https://www.eventdata.crossref.org/guide/) before continuing.

Our Python script `openAlt/pythonScripts/fetchEventBuffer.py` grabs new Event data from the Crossref API. This data is retrieved in a JSON format and then ingest into `crossrefeventdatamain` database.

Author metadata for publications are also retrieved from the Crossref API. Metada can be fetched and ingested by running the `metadataThroughMongoDB.py` script which can be found at (https://github.com/darpanshah-wsu/openAlt_W2021/blob/master/pythonScripts/metadataThroughMongoDB.py).

Citation data is retrieved from the OpenCitations API. This takes a longer duration than fetching the event data as a publication can have upwards of thosands of citations. We also recommend reading the manual for OpenCitations API at (https://opencitations.net/index/coci/api/v1).

`openAlt/pythonScripts/fetchOpenCitations.py` script can be run to fetch citation data for all of the publications of doidata database and store them in OpenCitations database.

4.1 Example JSON Format

Downloaded JSON files will look similar to this. Each of the 13 Event Types has a unique format.


```
{
        "license": "https:\/\/creativecommons.org\/publicdomain\/zero\/1.0\/",
        "obj_id": "https:\/\/doi.org\/10.1370\/afm.1970",
        "source_token": "a6c9d511-9239-4de8-a266-b013f5bd8764",
        "occurred_at": "2016-09-13T14:16:37Z",
        "subj_id": "https:\/\/reddit.com\/r\/UPFORFUN\/comments\/52koe6\/science_recommending_oral_probiotic
        "id": "c831d209-a955-4e61-903d-40ef35b0e454",
        "evidence_record": "https:\/\/evidence.eventdata.crossref.org\/evidence\/201702226e03dbb4-bc2e-46e3-
        "terms": "https:\/\/doi.org\/10.13003\/CED-terms-of-use",
        "action": "add",
        "subj": {
          "pid": "https:\/\/reddit.com\/r\/UPFORFUN\/comments\/52koe6\/science_recommending_oral_probiotics_
          "type": "post",
          "title": "[Science] Recommending Oral Probiotics to Reduce Winter Antibiotic Prescriptions in Peop
          "issued": "2016-09-13T14:16:37.000Z"
        },
        "source_id": "reddit",
        "obj": {
          "pid": "https:\/\/doi.org\/10.1370\/afm.1970",
          "url": "http:\/\/www.annfammed.org\/content\/14\/5\/422.full"
        },
        "timestamp": "2017-02-22T16:15:50Z",
        "relation_type_id": "discusses"
}
```

5. Ingesting the Data

These files will need to be ingested into the database by the following script: *openAlt/pythonScripts/Ingest/main.py*.

This script reads each JSON in the data directory, and inserts the events into the MySQL database. Again, the Event data does not contain the journal, publisher, authors, or titles for the DOI's. We utilized Dr. Bowman's database which was already populated with this data when we started this project. If you are cloning the repository, *you will need to source this data yourself*. This GitHub repository is a good place to start:
https://github.com/fabiobatalha/crossrefapi

5.1 Ingesting From JSON Files

1. Change the data directory for your JSON folder to suit your system (line 28).
2. Run python *ingestJSONMain.py* in your preferred terminal.

5.2 Ingesting from SciELO

We received a dump of data for articles from SciELO (*scielo.org*) that are from Brazil. This dump is located at *openAlt/pythonScripts/SciELOPID* and includes PIDs for over 400,000+ records. These records are not for the public at this time and will take time to gather all the information for each of the articles. It is recommended to run the following steps for about 10,000 records instead of the full 400,000 to see how the data is ingested.

How to Ready PIDs for Insertion:

SciELO data is inserted by using CSV files filled with a list of PID numbers from SciELO for the various articles. The program only accepts these lists in CSV format and the file itself should be one column with no header. In the one column there can be multiple rows for every PID that is to be entered into the database. An example with over 400,000 PIDs is located in the *openAlt/pythonScripts/SciELOPID* folder.

Step By Step Guide:

1. Open up MySQL Workbench.
    - Connect to your Local MySQL Connection.
2. Execute this SQL command *SELECT * FROM doidata._main_*
3. Go to this directory: *openAlt/pythonScripts* in the project files and locate the *getSciELO.py* file .
4. Navigate to the *openAlt/config* folder and locate *OpenAltConfig.json*. Change MySQL username and password to suit your device.
5. Run *python getSciELO.py* in your preferred terminal from *openAlt/pythonScripts* to ingest data from Crossref for the SciELO articles.
6. Run *python fetchEventBuffer.py* in your preferred terminal from *openAlt/pythonScripts* to ingest event data for the articles.
7. Run *python metadataThroughMongoDB.py* in your preferred terminal from *openAlt/pythonScripts* to ingest in metadata for the articles.
8. Run *python fetchOpenCitations.py* in your preferred terminal from *openAlt/pythonScripts* to ingest citation and reference data for the articles.

6. Quirks of the Crossref API

- Some Events give a DOI(objectID) of simply *https://doi.org*. For example, the event with ID: *5c83ca20-d4a1-471b-a23f-f21486cefb5c*

- Some DOI's in the Crossref Event data are malformed.
  This appears to be an inability of the Crossref agent to process Arabic text. We have only observed this for Twitter events so far.
  For example, the event with ID *5dd6719b-8981-4712-988c-8c01f7ad760b* has a DOI(objectID) of:

  "obj_id":
  "https://doi.org/www.kau.edu.sa/content.aspx?pg=%d9%88%d8%b8%d8%a7%d8%a6%d9%81-%d8%b4%d8%a7%d8%ba%d8%b1%d8%a9-%d8%b9%d9%84%d9%89-%d8%a8%d9%86%d8%af-%d8%a7%d9%84%d8%a7%d8%ac%d9%88%d8%b1-%d8%a8%d9%88%d9%83%d8%a7%d9%84%d8%a9-%d8%b4%d8%b7%d8%b1-%d8%a7%d9%84%d8%b7%d8%a7%d9%84%d8%a8%d8%a7%d8%aa"

- Many Twitter Events do not provide a link to the tweet as their subjectID. Instead, they have only *http://twitter.com* as their link. Since these events do not contain useful links, we have designed the website to hide these events

from the "Latest Events" section on the article dashboard. These events are still counted towards the total number of events for the author/article.

7. Initiating CRON Jobs

The purpose of the CRON Jobs is to gather new data and allow it to be fetched, sorted, and ingested from third-party APIs so that the databases are updated frequently. To set up the CRON Jobs, it must be run on Linux Terminal. To begin, you must first locate the path files of the following files:

- fetchEventBuffer.py
- fetchOpenCitations.py
- metadataThroughMongoDB.py

In addition to locating the path files of these files, you must also find the path file of the python library module. Without locating the python library, the CRON Jobs will not be able to execute the python scripts. Once these are found, you can begin to initiate the CRON Jobs.

The command to access the CRON Job scheduler through the Linux Terminal is: *`crontab -e`*. This will allow you to set up the CRON Jobs. To do this, you must first indicate how frequently you would like to run the script, then you specify the path file of the python library module, and finally you specify the path file of the python script. After indicating the tasks to be run, you will initiate the CRON Jobs by entering this command: *`sudo cron`*. To check if the CRON Jobs are running, you can check the status of it by entering this command on the terminal: *`sudo service cron status`*.

8. How to Run the Web Server

8.1 Before We Start

This guide assumes you are using Python 3.8, and have established the *`crossrefeventdatamain, doidata, opencitations,`* and *`bulksearchstats`* databases in MySQL. Check *`openAlt/SQL/`* for the relevant scripts.

If you have Python 2 installed, you will need to substitute Python3 for Python below.

8.2 Step by Step Guide

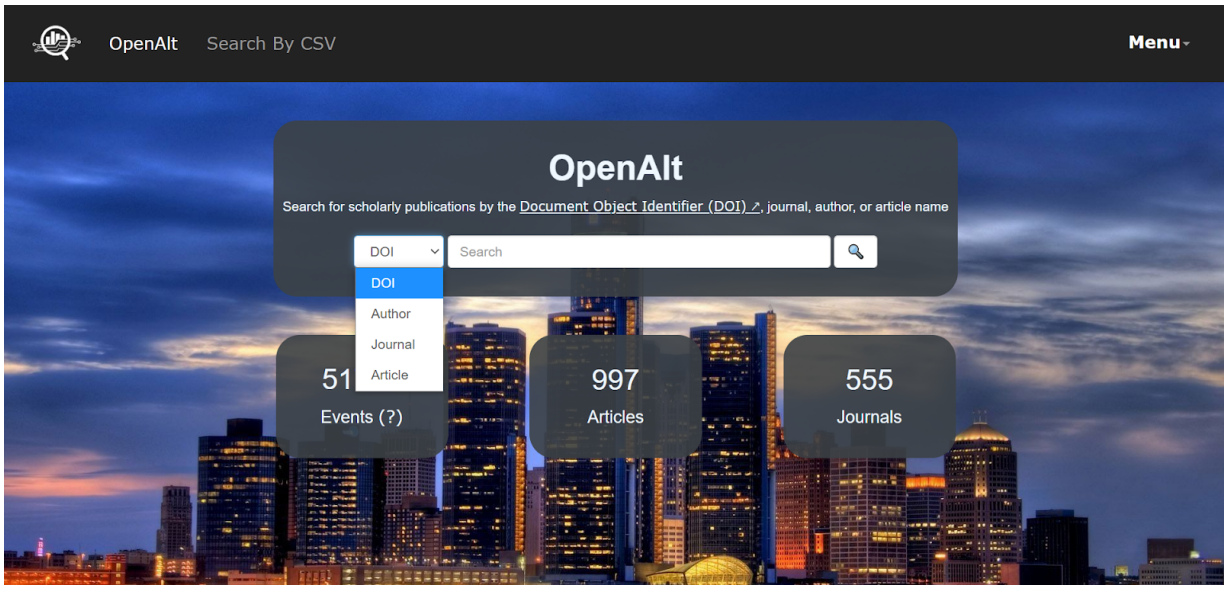
These actions should be performed inside the *openAlt/web/* folder.

1. Install virtualenv:
    - *pip install virtualenv*
2. Create a virtual environment:
    - *python -m virtualenv venv*
3. Activate the environment:
    - Windows: *./venv/scripts/activate*
    - Linux/Mac: *./venv/bin/activate*
4. Install Flask and our dependencies to this virtual environment:
    - *pip install flask mysql-connector-python flask-mysqldb python-dateutil flask-paginate*
5. Create a new file named *passwd.txt*
    - Open the file and type only your MySQL user password.
    - Save and close.
    - This file is ignored by git but used by *app.py* to access your local MySQL server.
6. Navigate to the config folder and locate *OpenAltConfig.json*. Change MySQL username and password to suit your device.
7. If you are not using the root MySQL user account, you will need to change the user on line 19 in *app.py*.
8. Start the web server using *python app.py*.
9. When the web server starts, navigate to *127.0.0.1:5000*.

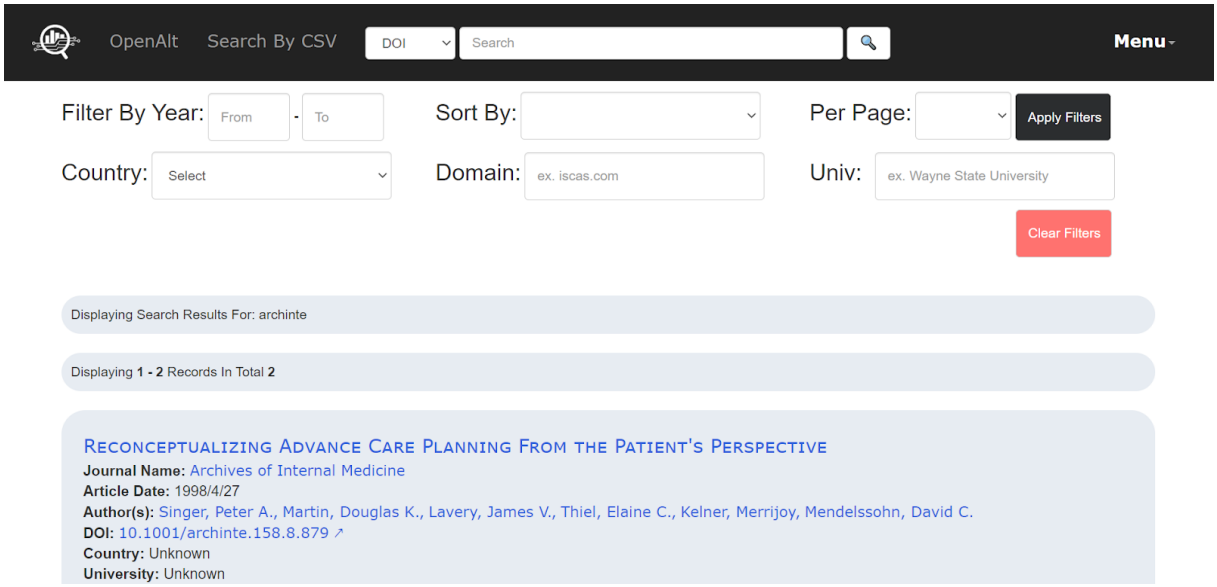
9. Using the App

9.1 Regular Search

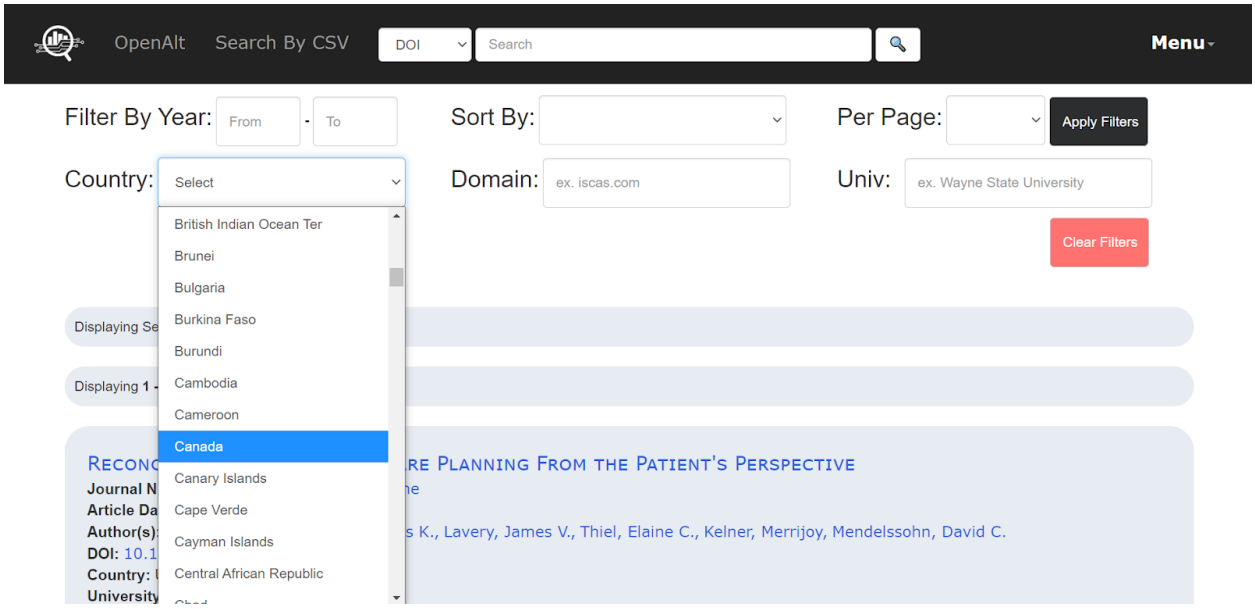
1. Select the type of search attribute, enter text on the search bar, and click the search icon on the right of search bar.



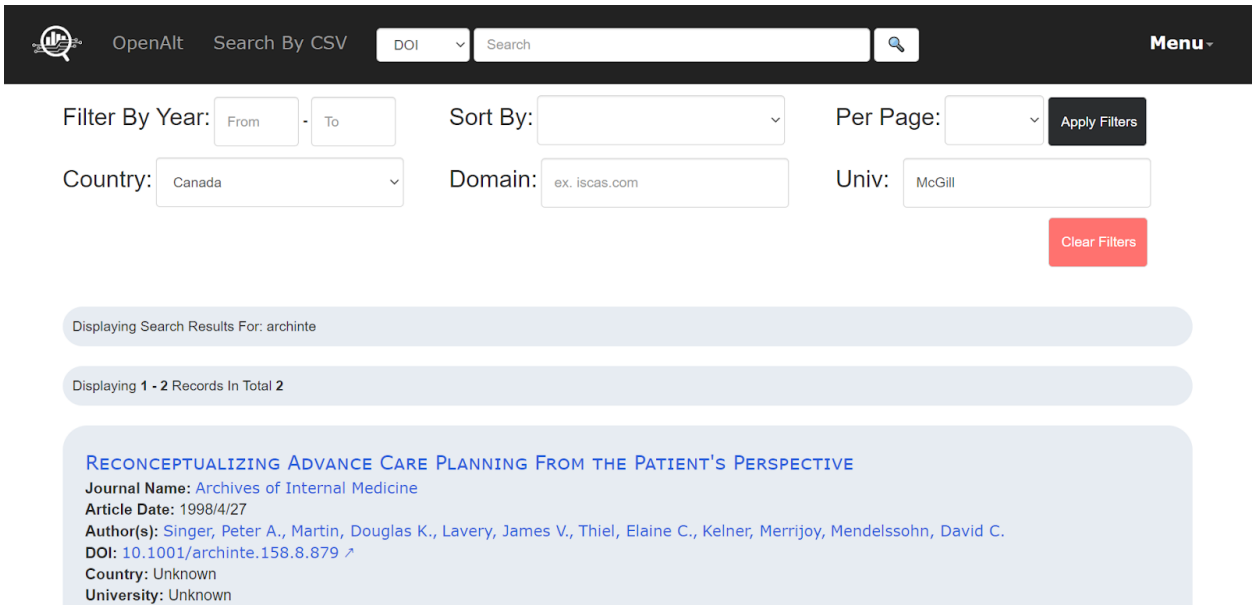
2. After hitting the search icon, you will be redirected to the search results page which looks like the following.



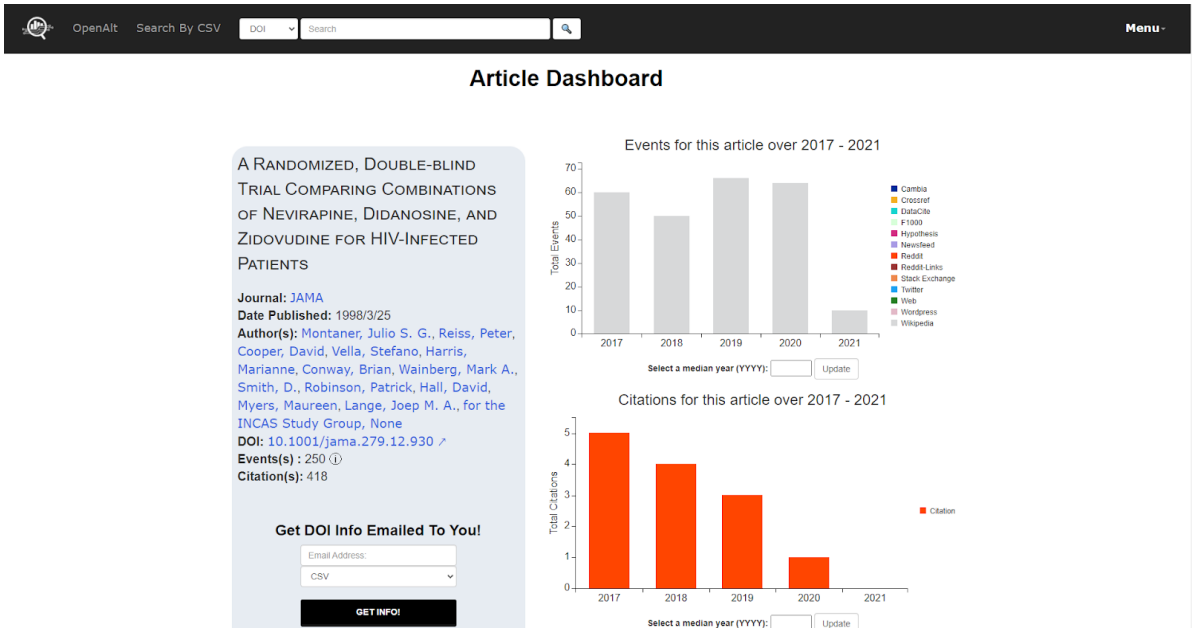
3. Click on the country dropdown menu and select a country from the list.



4. Enter domain or/and university name on the corresponding field. Click “Apply Filter”.

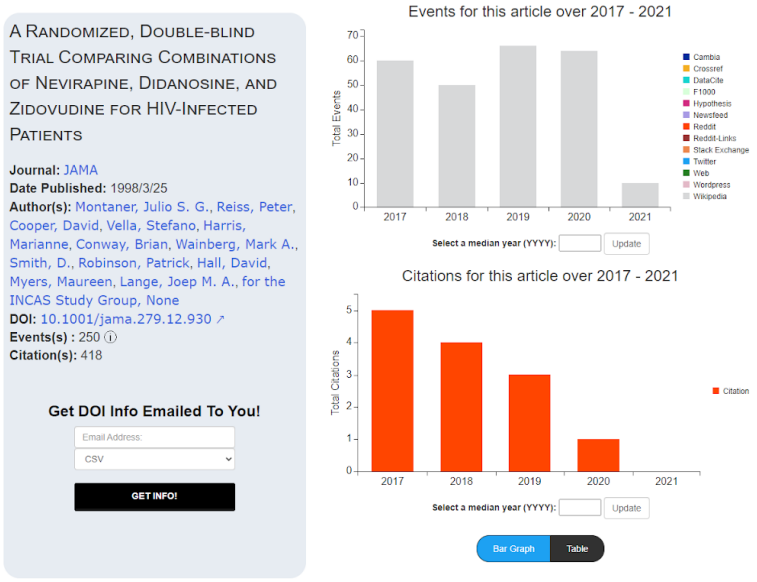


5. After clicking “Apply Filter”, you will be displayed publications associated with the selected filters. Click one of the articles from search results. You will be redirected to the “Article Dashboard Page” which looks like the following.

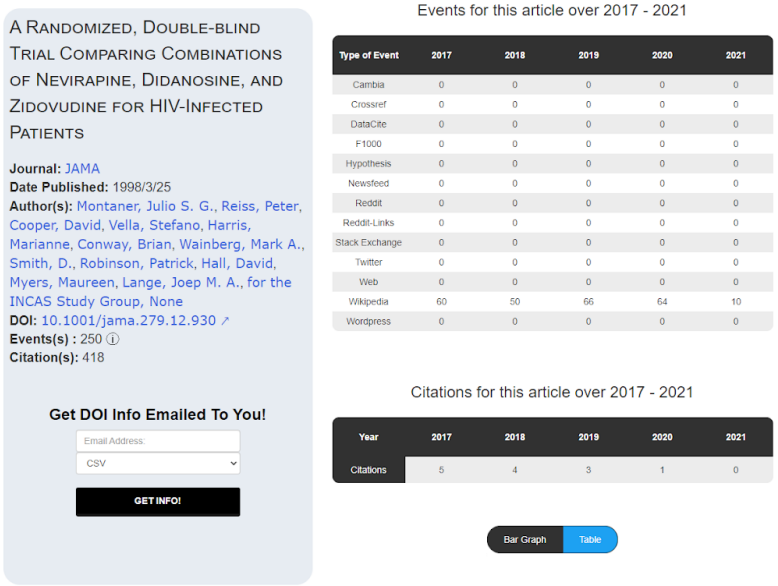


6. This page contains event and citation bar graphs. To update year for either graph, enter a year in the “Select a median year” field, and click update.

7. To view bar graph data in a tabular format, scroll down and click the “Table” button.

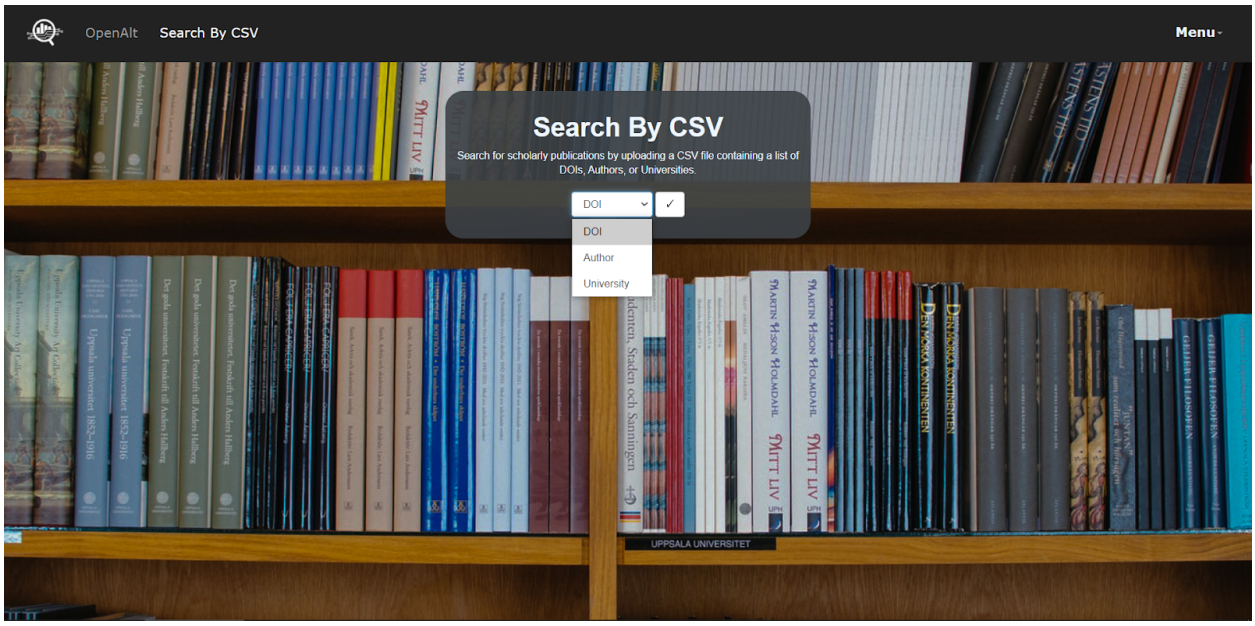


8. After clicking the table button, the page will look like the following. To revert back to bar graphs, click the “Bar Graph” button.

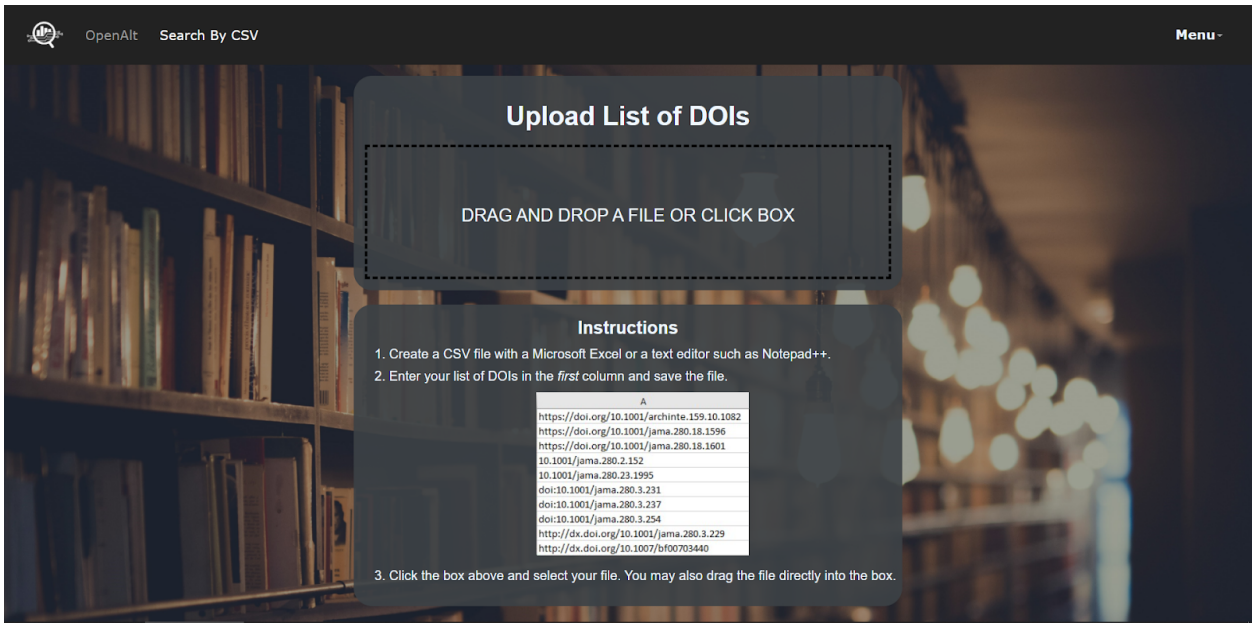


9.2 Search By CSV

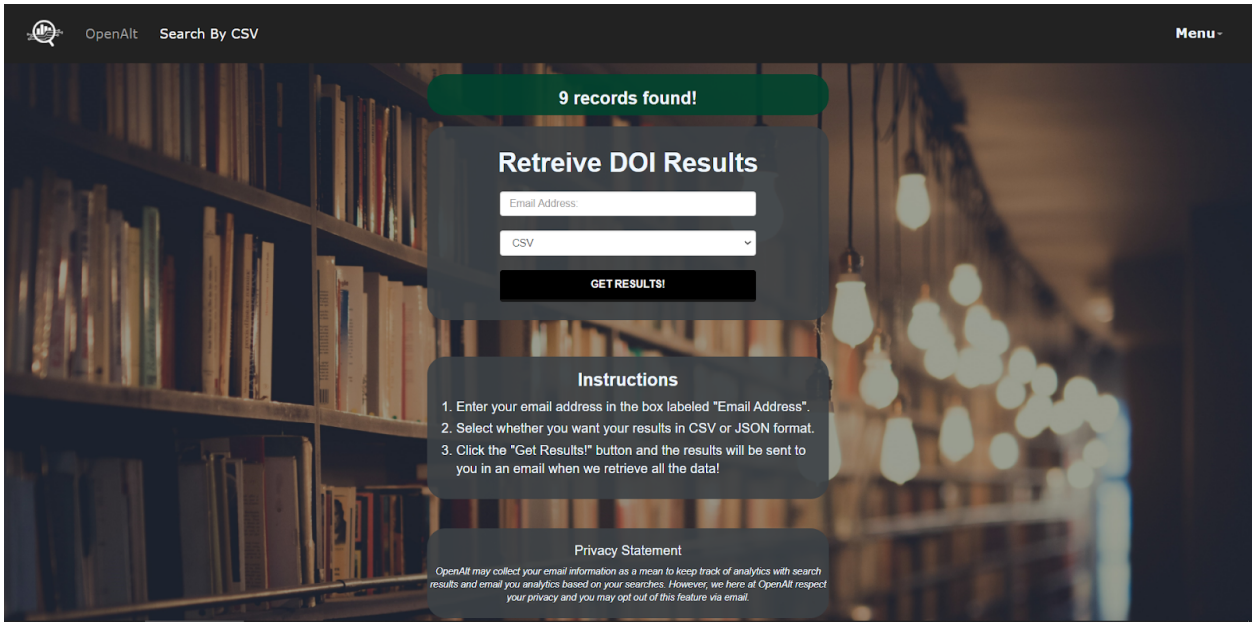
1. Click “Search By CSV” in the navbar to be lead to (/searchByOptions)



2. After selecting an option, you will be lead to a page similar to this:



3. Upload a CSV file containing a list of search entries as specified in the Instructions. Then you will be lead to a page similar to this:



4. Enter a valid email address and select the format you wish your results to be in and click “Get Results!”
5. You should receive the results in a zip file to the entered email address shortly.