Data Harvesters

* [Microsoft Academic Graph (MAG) harvester](#_1oda32pq9r00)
* Stackoverflow Harvester
* Github Harvester

### [MAG Harvester](#_1oda32pq9r00)

This is a harvester for getting research data from MAG, i.e., <https://msr-apis.portal.azure-api.net/products/project-academic-knowledge>. It consists of two scripts:

1. MAG\_Conference\_harvester - <https://colab.research.google.com/drive/1NfV8ahz39ylm5DkM3ZQQw07JullsAf9a?usp=sharing>

* Class MAG():

Arguments:

-----------

- conference: Name of the conference we need to fetch data for

- data\_fetch: What kind of Data to fetch from MAG (We have three options right now Publication data, Field of Study data and Author data )

- start\_date: start date of data

- end\_date: end date of data

- retry: Number of attempts to connect to MAG

Returns:

---------

- self.data: A cleaned data frame containing MAG data.

* Member functions
  + \_\_init\_\_(self,\*\*kwargs)
* Here we define the data members of the class:
  + self.conference: the conference name provided by the user.
  + self.start\_date: the start date provided by the user.
  + self.end\_date: The end date provided by the user.
  + self.retry: Number of attempts to connect to MAG.
  + self.s\_d: stores the year of the start date provided.
  + self.e\_d: stores the year of the end date provided.
  + Self.data\_fetch: The type of data we need to fetch is defined here according to user input.
  + self.conference\_id: Mapping conference name to its id and then storing it in this file. (Mapping: <https://drive.google.com/file/d/1lqLp8UUqSmdDaXQ6begbRFpct3Q3uXNh/view?usp=sharing> )
  + getData(self)
* In this function, we write queries for MAG and request some attributes based on the query and get the response from MAG.
* First, we define three variables which are standard format in MAG.
  + Endpoint = ‘<https://api.labs.cognitive.microsoft.com/>’
  + Api\_version = 'academic/v1.0/evaluate?'
  + Headers = {'Ocp-Apim-Subscription-Key':’Your subscription key to Project academic knowledge’}
  + To generate your own key subscribe to <https://msr-apis.portal.azure-api.net/products/project-academic-knowledge>.
* Here, we are using the Evaluate REST API Method from Project Academic knowledge to get the data we want. For details: <https://docs.microsoft.com/en-us/academic-services/project-academic-knowledge/reference-evaluate-method>.
* Based on the value given for self.data\_fetch we define searchstring, attributes and URL.
  + Searchstring: It contains the MAG query. These are defined to get particular data. For query expression: <https://docs.microsoft.com/en-us/academic-services/project-academic-knowledge/reference-query-expression-syntax>.
  + Attributes: Attributes are the values we need to fetch based on the queries. Each attribute in an entity has a particular name <https://docs.microsoft.com/en-us/academic-services/project-academic-knowledge/reference-entity-attributes>.
  + URL: It is the request url.
* Publication Count
  + We use Conference Instance Entity of MAG to get the publication count of each year of any of the conference listed in <https://drive.google.com/file/d/1lqLp8UUqSmdDaXQ6begbRFpct3Q3uXNh/view?usp=sharing>.
  + Searchstring = And(Composite(PCS.CN==self.conference),CISD=['self.start\_date','self.end\_date'])
  + Attributes are CISD,PC,CIL,PCS.CN.
  + Here, PCS.CN is the conference name and CISD is the Start date of the conference instance, PC is the publication count, CIL Location of the conference instance.
  + For more information on Entities visit <https://docs.microsoft.com/en-us/academic-services/project-academic-knowledge/reference-entity-attributes>.
* Field of Study
  + We use Paper entities to get the Field of study data.
  + In this we fetch all the papers that are published in a particular conference by querying the conference id and the year and returning all the papers with their Affiliations, Title, Fields of study mapped to this paper.
  + Searchstring = And(Composite(PCS.CId==self.conference\_id),Y=['self.s\_d','self.e\_d'])
  + Attributes are AA.DAfN,VSN,Y,F.FN,Ti.
  + Here, PCS.CId is the conference Id and Y is the year the paper was published, AA.DAfN is the Affiliation of the paper, F.FN is the Fields of study marked to that paper, Ti is Title of that paper and VSN conference normalized name.
  + For other attributes in paper entities visit: <https://docs.microsoft.com/en-us/academic-services/project-academic-knowledge/reference-paper-entity-attributes>.
* Author
  + We use paper entities to get the author data. We fetch all the papers of a particular conference and query the conference id and the year and return all the papers with their Affiliations, Title, and the Author names of that paper.
  + Searchstring = And(Composite(PCS.CId==self.conference\_id),Y=['self.s\_d','self.e\_d'])
  + Attributes are Ti,Y,AA.DAuN,AA.DAfN,VSN.
  + Here, PCS.CId is the conference Id and Y is the year the paper was published, AA.DAfN is the Affiliation of the paper, AA.DAuN is the Authors of the paper, Ti is Title of that paper and VSN conference normalized name.
* Once the URL is defined we generate a get request to the server,i.e, response = requests.get(url, headers=headers)
* And store the output in response.
* Then we store the output into a variable query:

query = response.json()

* And then we call json\_to\_csv() function to convert this json to csv.
* Then we call basicPreprocess() function for preprocessing the data.
* Finally, we call grid function for mapping Affiliations to countries.
  + json\_to\_csv()
* Standard json to csv converter.
* Input: json file
* Output: csv file
  + basicPreprocess()
* Here we preprocess the data based on the data fetched by the user.
* Field of study
  + Here we drop unnecessary fields and group by Date, Field\_of\_study, Affiliation and aggregate count to get the publication count of each affiliation in each field of study over the years.
* Author
  + Here we just drop unnecessary fields.
  + grid()
* This function is used to map affiliations with the city, state, countries, latitude and longitude.
* We use [GRID - Global Research Identifier Database](https://www.grid.ac/downloads).
* Here, first we merge address.csv and grid.csv to get the Latitude and longitude data with affiliation and country data into one dataframe and then we merge it with our dataframe.