An Automatic Collection System for Open Data in the Public Sector

Juhong Namgung*, Myeong-Seon Gil, and Yang-Sae Moon

Feb. 27, 2019

Dept. of Computer Science, Kangwon National University, Korea

DATA AND KNOWLEDGE ENGINEERING

Data & Knowledge Engineering Lab.

Department of Computer Science, Kangwon National University

Contents

- Background
- Motivation and Goals
- An Automatic Collection System for Open Data
  - System architecture
  - Input JSON file format
- Experimental Evaluation
- Conclusions

Background

Open Data

- “Open data is data that can be freely used, shared and built-on by anyone, anywhere, for any purpose.” – The Open Definition, 2005, The Open Knowledge Foundation

- Nowadays, many organizations are introducing open big data.

  ex) Green Button, Real-time prediction of energy consumption, and so on.

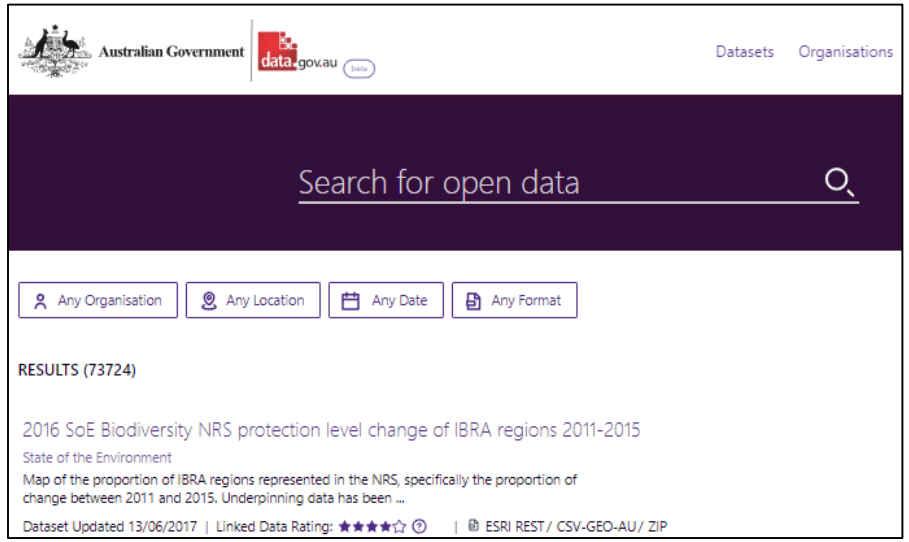
- Attention and demands on open data collection and deployment are fast growing.

Open Data Portals (1/3)

- Open Data Portal acts as a counter for the open big data, making it easy for anyone to collect and use the public data.
- Many government agencies provide public sector data through their Open Data Portals.
- United States (http://www.data.gov)
  - To build federal data repositories between government agencies.
  - Applications: ‘Roadify’, “Where are the jobs”

Open Data Portals (2/3)

- Australian (http://www.data.gov.au)
  - The government launched a government 2.0 task force team to pursue an open government.
  - Applications: “Suburban Trends”, “Aubiz.net”



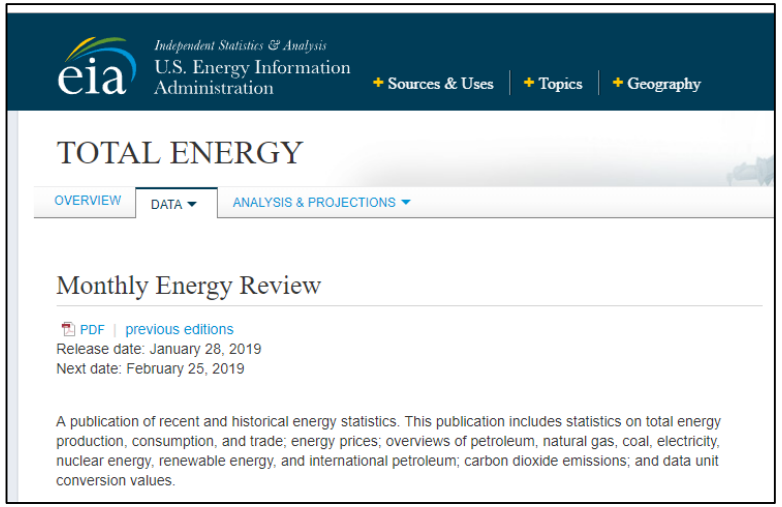
- Korea (http://www.data.go.kr)
  - Allows people to use the public data freely through communication channels.
  - Applications: “Seoul Bus”, “Public Culture and Art Information”

서울버스

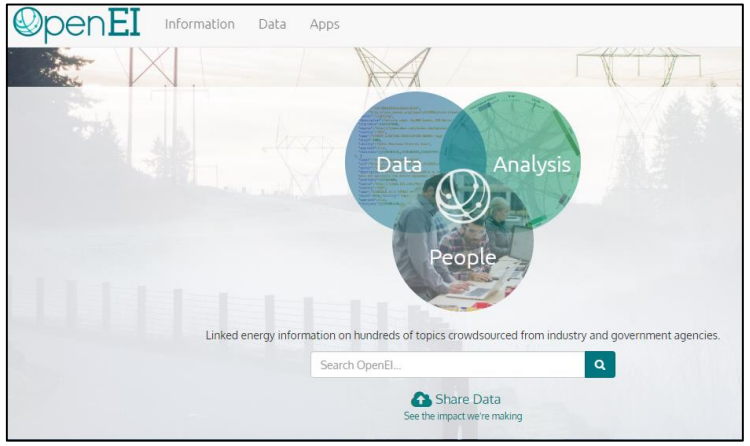
Open Data Portals (3/3)

- In the field of energy, Open Data Portals provide energy-related big data and show interesting use cases.

- EIA (Energy Information Administration)
  - Analyzes the energy data to predict short-term and long-term trends in the energy market.
  - Application: U.S. Green Button



- OpenEI (Open Energy Information)
  - Provides energy data and energy applications for data analysis.



Open Data Deployment Methods

- File-based
  - Most common way of providing the open data in Open Data Portals.
  - Downloads the open data directly as a file format.
  - Update frequency is very low → It is used to deploy a large amount of **accumulated static data**.
- API-based
  - Data transmission through Open API(Application Programming Interface).
  - It provides **frequently updated data in real time**.
- Visualization, statistics, etc.

❖ We focus on the File- and API-based data collection.

Motivation and Goals

Problems on Open Data Collection (1/2)

❖ If we want to collect the open data continuously from different Open Data Portals?

Problem 1 It needs manual operations and repeated jobs.

- File-based
  - Click download URL.
  - We need to **periodically download** files.
- API-based

1) We need to request API Key in Open Data Portal.

2) We define service URL and request parameters.

3) Based on the API Key and URL, we perform API calls.

ex) http://data.ekape.or.kr/openapi-data/service/cattleNO=4100&ServiceKey=APIKEY

4) We saves the result data of API calls in file format.

5) We have to **repeat the above process**.

> 오픈 API 인증키와 URL을 기반으로 API 호출 가능

http://data.ekape.or.kr/openapi-data/service/user/mtrace/breeding/cattleMove?cattleNo=410002042894485&ServiceKey=서비스키

제공기관 : 축산물품질평가원
오픈API명 : 쇠고기이력정보서비스
상세기능명 : 소 이동정보


```xml
<?xml version="1.0" encoding="UTF-8" standalone="true"?>
- <response>
  - <header>
      <resultCode>00</resultCode>
      <resultMsg>NORMAL SERVICE.</resultMsg>
    </header>
    <notice/>
    <body>
      - <items>
        - <item>
            <cattleNo>410002042894485</cattleNo>
            <farmAddr>경상남도 산청군 삼장면</farmAddr>
            <farmerNm>정재진</farmerNm>
            <movePlace>전산등록</movePlace>
            <moveYmd>2009-06-10</moveYmd>
          </item>
        - <item>
            <cattleNo>410002042894485</cattleNo>
            <farmAddr>경상남도 진주시 명석면</farmAddr>
            <farmerNm>이재생</farmerNm>
            <movePlace>양수</movePlace>
            <moveYmd>2012-09-13</moveYmd>
          </item>
        - <item>
            <cattleNo>410002042894485</cattleNo>
            <farmAddr>경남 진주시 명석면 외율리</farmAddr>
            <farmerNm>이재생</farmerNm>
            <movePlace>도축출하</movePlace>
            <moveYmd>2013-02-23</moveYmd>
          </item>
        </items>
    </body>
</response>
```

Problems on Open Data Collection (2/2)

❖ If we want to collect the open data continuously in different Open Data Portals?

Problem 2 → APIs require different parameters for each Open Data Portals.

- EIA
  - API Key is necessary.
  - Saves the data in a JSON format.
- U.S. Open Data Portal
  - API Key is not required.
  - Resulting data supports various formats.
    ex) XML, CSV etc.
- Australian Data Portal
  - Include the header information in the request URL.

API Query Browser

EIA Data Sets > U.S. Nuclear Outages

API CALL TO USE

http://api.eia.gov/series/?
api_key=YOUR_API_KEY_HERE&series_id=NUC_STATUS.OUT.US.D

<An API request URL example in EIA>

```
curl -X GET --header 'Accept: text/plain'
--header 'Authorization: apikey 3vqs****'
'https://api.transport.nsw.gov.au/v1/gtfs/realtime/
buses?debug=true'
```

<An API request URL example in Australian Open Data Portal>

Motivation and Goals

- In order to continuously collect the big data from Open Data Portals,

  1) We need to periodically download files or perform API calls.

  2) In API-based collection, we need to make appropriate parameter settings and perform API calls.

  3) We need to save and manage the result data.

- The goal is to collect open data automatically from Open Data Portals.

We propose a **data collection system that automatically gathers open data.**

An Automatic Collection System

Automatic Collection System

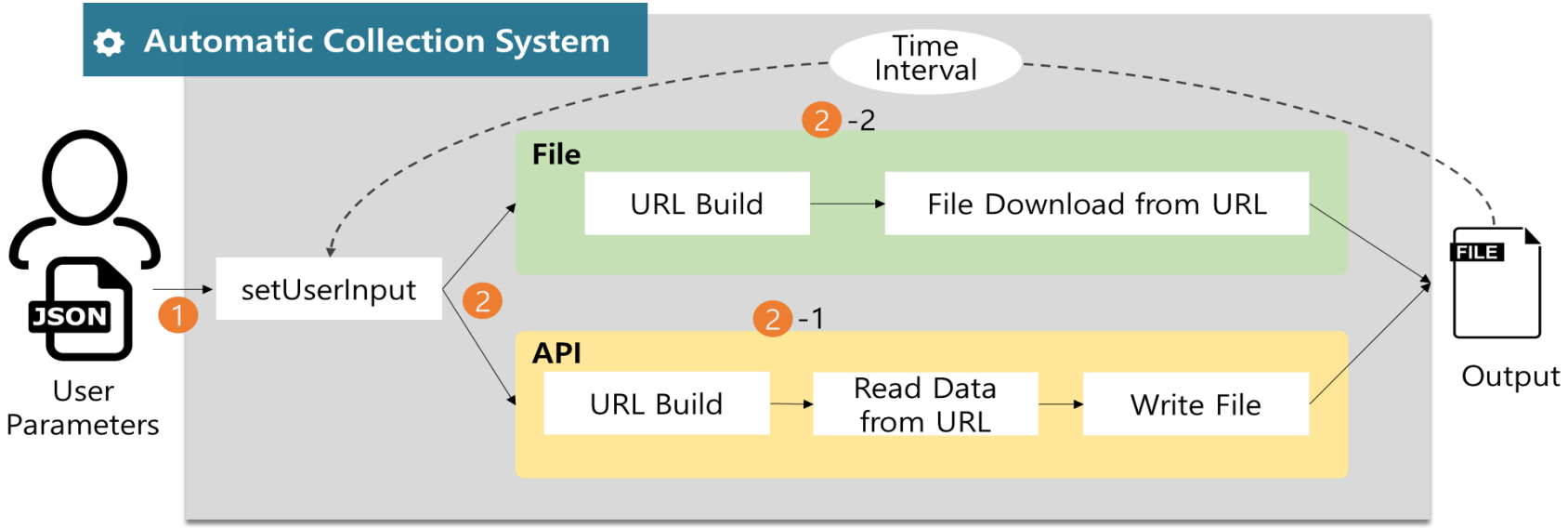
① Our system receives an input JSON file and saves the parameters.

– In order to support different file formats and APIs of Open Data Portals.

② It decide to file gathering operations.

1) File-based: downloads the file using a URL in user parameters.

2) API-based: builds a URL based on user parameters, then reads the required data and save the data.



❖ It saves file using a naming convention to distinguish and manage the collected data.

– Combines the file name with the saved time (timestamp)

Input JSON File Format (1/3)

- File-based data collection input JSON file format
  - ① File download URL (`URL`)
  - ② Time period (`timeInterval`)
  - ③ File path to store the results (`directory`)
  - ④ File name to store the results (`filename`)

```
{
  "URL": " Download URL " ,
  (Option)"timeInterval": "time",
  "directory": "user_directory",
  "filename": "user_filename"
}
```

<User parameter format for **file-based** data collection>

Input JSON File Format (2/3)

```
{
  "URL": " Service URL " ,
  (Option)"serviceKey": "****" |
              { user_key:user_value },
  (Option)"parameter": {
       "key1": "value1",
       "key2": "value2",
        …
        },
  "timeInterval": "time",
  "directory": "user_directory",
  "filename": "user_filename",
  "input_format: " xml | json | csv | txt ",
  "output_format: " xml | json | csv | txt "
}
```

<User parameter format for **API-based** data collection>

- API-based data collection input JSON file format
  - API call message definition
    - ① Service URL (URL)
    - ② API Key (serviceKey)
    - ③ User request parameters (parameter)
  - User output definition
    - ④ Time period (timeInterval)
    - ⑤ File path to store the results (directory)
    - ⑥ File name to store the results (filename)
    - ⑦ File format (input&output format)

❖ Our system supports type conversion function

Input JSON File Format (3/3)

- Our system can collect **file- and API-based data** according to the final input JSON file format.

```
{
  "service": " file | api ",
  "URL": " Download or Service URL " ,
  (Option)"serviceKey": "****" | { user_key:user_value },
  (Option)"parameter": {
       "key1": "value1",
       "key2": "value2",
        …
        },
  (Option)"timeInterval": "time",
  "directory": "user_directory",
  "filename": "user_filename",
  (Option)"input_format: " xml | json | csv | txt ",
  (Option)"output_format: " xml | json | csv | txt "
}
```

<Final format of user parameters in file- and API-based data collections>

Experimental Evaluation

Experimental Environment

Evaluation Scenario

- An automatic data collection using different methods.
  - File- and API-based data collection.
- An automatic data collection from different Open Data Portals.
- Data collection using type conversion.

System Implementation Environment

- Hardware: Intel(R) Core™ i3-4150 CPU, 4GB RAM
- Software: CentOS 7.3 Linux operating system
- Develop Language: Java

File-based Data Collection (1/2)

- OpenEI file-based data collection
  - Hourly energy emission amount for electricity generation in the United States data
  - Method: File-based collection
  - Time period: 1 min

Data and Resources

Monthly average hourly CO2, NOx, and SO2 emission ... Monthly average hourly CO2, NOx, and SO2 emission factors for each U.S. eGRID...	More info Download
Monthly average hourly charts for each mainland U Monthly average hourly charts for each mainland U.S. eGRID subregion showing...	Preview Download
Monthly average hourly CO2, NOx, and SO2 emission ... Monthly average hourly CO2, NOx, and SO2 emission factors for the AZNM eGRID...	Preview Download
Monthly average hourly CO2, NOx, and SO2 emission ... Monthly average hourly CO2, NOx, and SO2 emission factors for the CAMX eGRID...	Preview Download
Monthly average hourly CO2, NOx, and SO2 emission ... Monthly average hourly CO2, NOx, and SO2 emission factors for the ERCT eGRID...	Preview Download

OpenEI Wiki Apps Datasets Search Login Sign Up
Find data Add data About CKAN

License

Open Data Commons Attribution License 1.0

Author

National Renewable Energy Laboratory

Contact

Daniel Studer

Share on Social Sites

Google+
Twitter
Facebook

Dataset Activity Stream

Harvested, read orignal on DOE Opendata

Hourly Energy Emission Factors for Electricity Generation in the United States

Emissions from energy use in buildings are usually estimated on an annual basis using annual average multipliers. Using annual numbers provides a reasonable estimation of emissions, but it provides no indication of the temporal nature of the emissions. Therefore, there is no way of understanding the impact on emissions from load shifting and peak shaving technologies such as thermal energy storage, on-site renewable energy, and demand control.

This project utilized GridViewTM, an electric grid dispatch software package, to estimate hourly emission factors for all of the eGRID subregions in the continental United States. These factors took into account electricity imports and exports across the eGRID subregion boundary, and included estimated transmission and distribution (T) losses. Emission types accounted for included carbon dioxide (CO2), nitrogen oxides (NOx), and sulfur dioxide (SO2).Data reported as part of this project include hourly average, minimum, and maximum emission factors by month; that is, the average, minimum, and maximum emission factor for the same hour of each day in a month. Please note that the data are reported in lbs/MWh, where the MWh value reported is site electricity use (the actual electricity used at the building) and the pounds of emissions reported are the emissions created at the generator to meet the building load, including transmission and distribution losses. The demand profiles used to generate the data pertain to the following years: eastern interconnect - 2005; Electricity Reliability Council of Texas (ERCOT) - 2008; Western Electricity Coordinating Council (WECC) - 2008.

File-based Data Collection (2/2)

- OpenEI file-based data collection result
  - We can confirm that the proposed system gathers the data using **file-based method** every minute periodically.

```
{
    "service": "file",
    "URL": "https://openei.org/doe-opendata/dataset/download/usmonthlyaverageh
        ourlyemissionfactorsforaznmegridsubregion.csv",
    "timeInterval" : "1",
    "dir": "/home/hadoop/paper/data/"
    "filename": "hourly_emission_factor"
}
```

<Input JSON for file-based data collection>

```
-,Hour,Min,Average,Max,Standard Deviation,Min,Average,Max,Standar
January,Monthly Average,"1,035","1,219","1,410",70.6,1.15,1.61,2.
January,1,"1,185","1,292","1,395",51.3,1.62,1.82,2.11,0.13,2.13,2
January,2,"1,200","1,306","1,407",50.0,1.65,1.85,2.11,0.13,2.18,2
January,3,"1,197","1,306","1,404",53.8,1.63,1.84,2.12,0.15,2.14,2
January,4,"1,185","1,303","1,405",53.9,1.58,1.83,2.11,0.14,2.12,2
January,5,"1,189","1,285","1,410",51.7,1.57,1.77,2.12,0.14,2.12,2
January,6,"1,159","1,247","1,397",51.3,1.47,1.66,2.06,0.14,2.01,2
January,7,"1,089","1,197","1,368",57.3,1.28,1.54,1.96,0.16,1.85,2
January,8,"1,075","1,181","1,346",58.1,1.22,1.50,1.96,0.18,1.83,2
January,9,"1,078","1,177","1,332",49.7,1.28,1.50,1.92,0.14,1.82,2
```

<Hourly energy emission data>

Name	Size (KB)	Last modified
..		
hourly_emission_factor-2018-12-04_00_36_40.csv	23	2019-12-04 ...
hourly_emission_factor-2018-12-04_00_37_41.csv	23	2019-12-04 ...
hourly_emission_factor-2018-12-04_00_38_43.csv	23	2019-12-04 ...
hourly_emission_factor-2018-12-04_00_39_44.csv	23	2019-12-04 ...
hourly_emission_factor-2018-12-04_00_40_45.csv	23	2019-12-04 ...
hourly_emission_factor-2018-12-04_00_41_46.csv	23	2019-12-04 ...
hourly_emission_factor-2018-12-04_00_42_48.csv	23	2019-12-04 ...
hourly_emission_factor-2018-12-04_00_43_49.csv	23	2019-12-04 ...

<Saved result file>

API-based Data Collection (1/2)

- EIA API-based data collection
  - Hourly demand data for Arizona Public Service Company
  - Method: API-based collection
  - Time period: 5 min

eia + Sources & Uses + Topics + C

Demand for Arizona Public Service Company (AZPS), Hourly	20181204T05Z	H	3348
Demand for Arizona Public Service Company (AZPS), Hourly	20181204T04Z	H	3438
Demand for Arizona Public Service Company (AZPS), Hourly	20181204T03Z	H	3454
Demand for Arizona Public Service Company (AZPS), Hourly	20181204T02Z	H	3451
Demand for Arizona Public Service Company (AZPS), Hourly	20181204T01Z	H	3222
Demand for Arizona Public Service Company (AZPS), Hourly	20181204T00Z	H	2817

API Query Browser

EIA Data Sets > U.S. Electric System Operating Data > Demand

API CALL TO USE	http://api.eia.gov/series/?api_key=YOUR_API_KEY_HERE&series_id=EBA.AZPS-ALL.D.H
SERIES NAME	Demand for Arizona Public Service Company (AZPS), Hourly
SERIES ID:	EBA.AZPS-ALL.D.H Show me how to embed a chart of this series
GEOSET ID:	EBA.D.H Show me how to embed a map of this set

API-based Data Collection (2/2)

- EIA API-based data collection result
  - We confirm that the system collects the data using **API-based method**.

```
{
        "service": "api",
        "URL" :
"http://api.eia.gov/series/?api_key=303ab55f5ca64005c07
c2df4892e9ee5&series_id=EBA.AZPS-ALL.D.H"
        "timeInterval" : "5",
        "dir": "/home/hadoop/data/eia/",
        "filename": "demand_AZPS_hourly",
}
```

<Input JSON for API-based data collection>


```
{
  "request": {
    "command": "series",
    "series_id": "EBA.AZPS-ALL.D.H"
  },
  "series": [
    {
      "series_id": "EBA.AZPS-ALL.D.H",
      "name": "Demand for Arizona Public Service Company (AZPS), Hourly",
      "units": "megawatthours",
      "f": "H",
      "description": "Timestamps follow the ISO8601 standard (https://en.w
rovided in Universal Time.",
      "start": "20150701T08Z",
      "end": "20181203T13Z",
      "updated": "2018-12-03T09:30:54-0500",
      "data": [
        [
          "20181203T13Z",
          3142
        ],
        [
          "20181203T12Z",
          2870
        ],
```


<Hourly demand data>

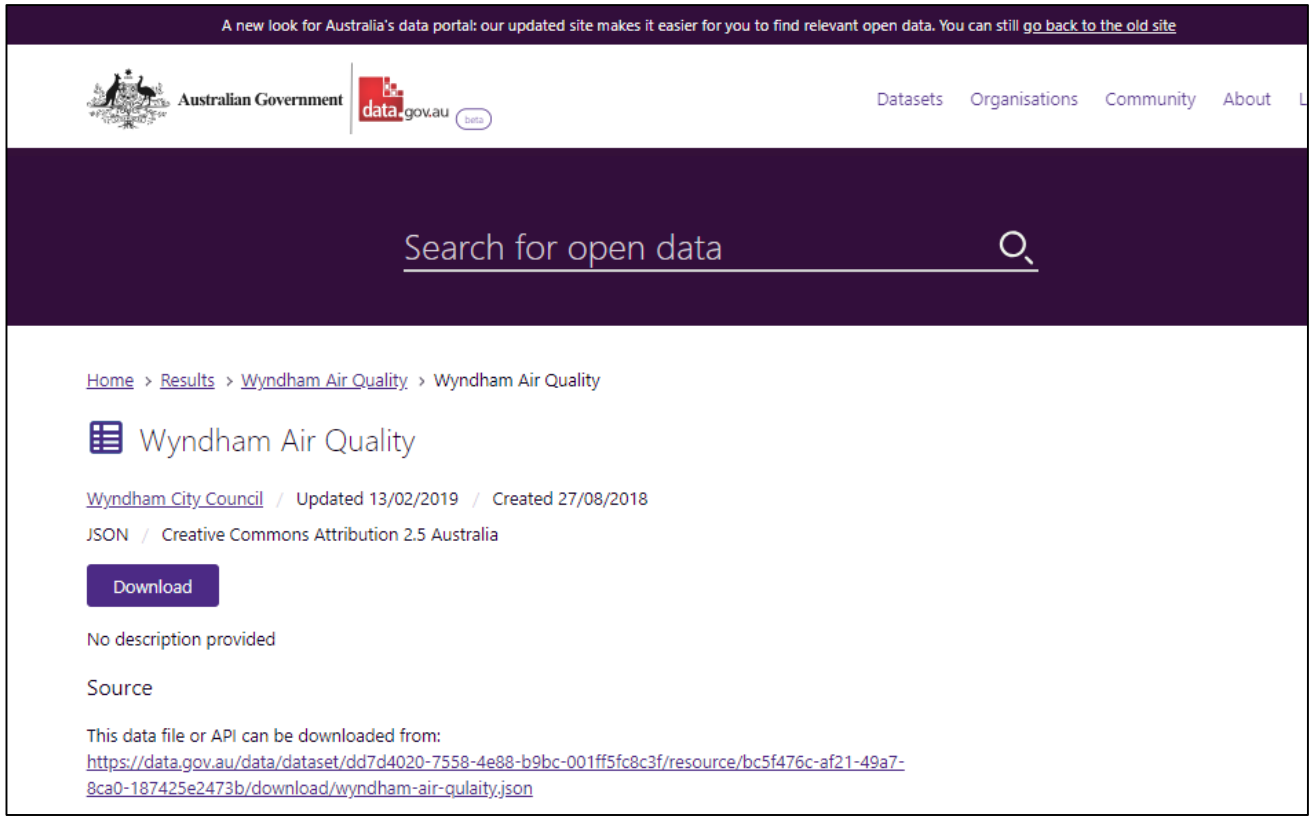
Name	Size (KB)	Last modified
..		
demand_AZPS_hourly-2018-12-04_01_27_30.json	1 815	2019-12-04 ...
demand_AZPS_hourly-2018-12-04_01_32_32.json	1 815	2019-12-04 ...
demand_AZPS_hourly-2018-12-04_01_37_34.json	1 815	2019-12-04 ...
demand_AZPS_hourly-2018-12-04_01_42_36.json	1 815	2019-12-04 ...
demand_AZPS_hourly-2018-12-04_01_47_38.json	1 815	2019-12-04 ...

<Saved result file>

Type Conversion Data Collection (1/2)

- Australian Open Data Portal
  - Wyndham's air quality data
  - Method: API-based collection
  - Input data format: JSON
  - Output data format: CSV

  → **Using type conversion**



Type Conversion Data Collection (2/2)

- Australian Open Data Portal result
  - We confirm that the system stores the data automatically in the CSV format according to the **type conversion settings**.

```
{
        "service": "api",
        "URL" : "https://data.gov.au/dataset/dd7d4020-7558-4e88-b9bc-001ff5fc8c3f/resource/bc5f476c-af21-49a7-8ca0-187425e2473b/download/wyndham-air-qulaity.json"
        "timeInterval" : "1",
        "dir": "/home/hadoop/data/au/",
        "filename": "Wyndham_air_quality",
        “input_format": "json",
        "output_format":"csv"
}
```

<Input JSON for type conversion data collection>


```
[
  {
    "json_featuretype":"wyndham-air-qulaity.json",
    "timestamp":"2019-02-13T15:02",
    "AQI":"0",
    "AQIrealtime":"0",
    "AQI30":"0",
    "AQI60":"0",
    "AQI6h":"0",
    "AQI24h":"4",
    "AQI1w":"25",
    "AQI10":"3",
    "lattitude":"-37.900868",
    "longitude":"144.66433"
  },
```


<API call result data>


```
AQI,AQI10,AQI1w,AQI24h,AQI30,AQI60,AQI6h,AQIrealtime,json_featuretype,lattitude,
0,4,8,4,0,0,4,2,wyndham-air-qulaity.json,-37.900868,144.66433,2018-12-04T05:12
0,1,8,4,0,0,0,0,wyndham-air-qulaity.json,-37.894849,144.648308,2018-12-04T05:12
```


<Stored data using the type conversion function of system>

Conclusions

Conclusions

- **We proposed an automatic collection system that periodically gathers open data.**
  - Our system collects file-and API-based open data continuously from Open Data Portals.
  - It supports the type conversion function.
- We showed that the proposed system successfully collected the open data.
- The proposed system makes naïve users collect and use various open data of public sectors more easily and practically.

Thank you!