Data Mining and Visualzation For Material Design Via R Language

Amir Mostaghimi

15/04/2021

Introduction

More than a century ago, Fritz Haber proposed a catalytic process for sustainable ammonia synthesis which today is known as Haber-bush process. This method utilizes earth abundant nitrogen and hydrogen in presence of an Fe based catalyst to produce NH^3 which is a key component for fertilizers. Figure 1 (Erisman et al. (2008)), shows the impact of this catalytic process on human population. Since then, heterogeneous catalysis became one of the center pillars in almost all the industrial sectors including transportation, sustainable energy, food industry, etc.(Friend and Xu (2017)) A key aspect in heterogeneous catalysis is design ideal active, cost-efficient, stable, and environmentally friendly materials.

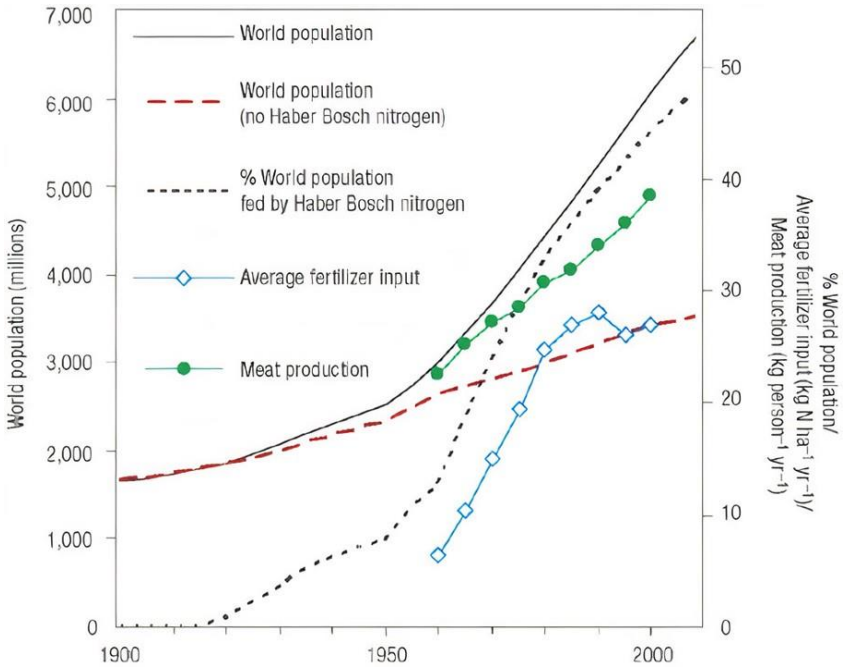


Figure1) The dramatic effect of Haber bush process on the world's population increase. The dashed red line shows the projected population without ammonia synthesis.

With the aid of tremendous increase in the computational power and resources, advanced simulation methods such as Density functional theory (DFT) and molecular dynamics (MD) became accessible. As a result, the chemical and mechanical properties of a vast number of

materials have been examined and reported.(Yang, Fidelis, and Sun (2019)) On the other hand, an interdisciplinary method known as machine learning (ML) has been perfected and entered to many fields of science. The purpose of ML methods is primarily to analyze existing data clusters and identify trends, rules or functions. To produce reliable predictions, ML methods require sufficient amount of featured data which is processable for the currently existing algorithms.(Ma and Liu (2020)) However, despite being a powerful data analysis tool, the data gathering and analysis for material design purposes are yet to be developed for r language. In this report, we design an Application Programming Interface (API) for data collection and processing. An API can be defined as a software intermediary that allows communication across multiple applications. We used "httr" package to create a comprehensive function for data collection form an online material data base. The collected data is converted to r objects for data visualization. Finally, the data will be processed and returned in the format of a dataframe suitable for machine learning purposes. The goal of this report is the demonstrate a technique to facilitate the cumbersome procedure of data collection, visualization and engineering from an online data source.

Computational method and tools

We used Rstodio as IDE to interpret r programming language and codes. The software was installed and tested on Windows 10 operating system (OS). The data transfer with the data base were performed using Hyper Text Transfer Protocol (HTTP) using "httr" package.(Wickham (2020)) This package provides a convenient way to request and get response form most of the data servers.

```
install.packages("httr") #Packages in R are a collection of functions which f
acilitate the coding process. Each package needs to be installed once and can
be called in any code. This line is to install "httr" package which is descri
bed above.

library(httr) #To use the functionality of packages, they must be called in e
ach script. This line shows how "httr" package is called
```

This library comes with most common HTTP verbs. Get() function, can be used for collecting data from an online server. This function requires a correct format of address to the data in question. The data must be a Uniform Resource Locator (URL) in string format. Every database would provide the instruction on the structure of the URL. In this study we will be using Materials project which the benchmark database for materials science. In the following we show the general format of URL and store the string in a variable called url.(Jain et al. (2013))

```
url=  paste0 ("https://www.materialsproject.org/rest/v2/materials/",request_t
ype,"/vasp?API_KEY=",API_key)
```

As shown in the code, the URL takes two arguments namely "API_Key" and "request_type." An "API_Key" is a unique code to identify the user, developer, or programmer. The code can be obtained from the data provider and utilized in an application. The second argument "request_type" specifies the type of data to be collected. Materials project gives the option to download the information based on 1) material ID, 2) system type, or 3) chemical formula.

Therefore, it is essential to generate url variable flexible enough to operate with any type of request. The following section will be dedicated to build a function that automatically detects the request type and collect the data.

Data collection

We prepared a function called "get_data" to automate the process of requesting and downloading data. The function takes the format of request as the only argument and combines it with a set of string to generate the url to the target data. The target data is then downloaded and formatted via "httr" functionalities.

```
API_key= "JIHJm9DeH3vZJxhDgk9"
request_type= NA #Making an object with empty entry. This object will determi
ne what information will be downloaded form the data base.
get_data <- function(request_type){ # Creates a function to automate sending
request and downloading data form the data base. The arguments the data type
to be downloaded.
  if (is.na(request_type)){ # A simple condition ensure the request type is s
pecified.
    print ('please specify the request type')
  }
  else {
    request_url=  paste0 ("https://www.materialsproject.org/rest/v2/materials
/",request_type,"/vasp?API_KEY=",API_key) # Makes a unify URL in the format o
f string.
    raw_results <<- GET(request_url) # Sends an access and download request t
o the data base with the specified authentication key.
    text_results <<- content(raw_results, as= "text") # Extracts the content
of the downloaded data and saves it as text object.
  }
}
```

The generated function is tested as following:

```
get_data("*1*102") # We test the request function to download data for perovs
kite structures.

## No encoding supplied: defaulting to UTF-8.

class (raw_results) # Checks the type of data in "raw_results" object.

## [1] "response"

class (text_results)

## [1] "character"
```

The data requested in get_data function is "* 1* 103" in which stands for any atom. In general, this is a call for a class of materials known as "Perovskites" with general formula of ABO^3. The function will return a response object (raw_results) which has all the communication details. It is not possible to read or modify data as a response object hence, it is converted to

a character object (text_results). The obtained text object needs to be converted to a dataframe for data analysis. One must note the character object is in json format. We used "jsonlite" and "dplyr" libraries to read and convert the Json data. "jsonlite" library is a package used for converting json objects to r objects whilst the other library (dplyr) is a dataframe manipulation package. The following code demonstrates the conversion of json object to a dataframe (my.df).

```
install.packages(c("jsonlite","dplyr"))

library(jsonlite)
library(dplyr)

my.df <- fromJSON(text_results, flatten = T) %>% data.frame() # The data we h
ave is in the text JSON format. This line will read the Json file and convert
the content into a dataframe.
class(my.df) #Checks the class of the my.df to make sure the

## [1] "data.frame"
```

The argument "flatten = T" indicates that the initial data consists of several nested lists. Therefore, the json format is unlisted to have easier access to all the 185 variables in the dataframe.

Data visualization

To start data manipulation, it is essential to establish the necessary knowledge on the structure of the data and the type of information it contains. This goal can be achieved via data visualization methods that are available in r. As mentioned earlier, the initial json response is consist of several nested lists which are cumbersome to navigate through. To address this issue, we used "listviewer" package which converts a json data into an interactive window (Figure2a). A "tree" type representation is used in the interactive window to display the parent and child lists in the json object. The figure shows the "response" tab has 1119 child nodes which corresponds to 1119 materials (observations). Each of the 1119 materials contains physical, chemical, and structural properties that can be used for training ML models. The 185 rows in the "my.df" dataframe corresponds to the material properties. To gain insight on the existing properties, we mapped the information via "NetworkD3" package. The results are summarized in the format of an interactive radial tree diagram in figure 2b. The Lives on the diagram correspond to the rows of the dataframe (Material property).

```
install.packages(c("data.tree","networkD3","webshot"))

library(data.tree) # a data organization library
library(networkD3) # A graphing library
library(webshot)

## Warning: package 'webshot' was built under R version 4.0.5

webshot::install_phantomjs() #The majority of the data visualization tools ar
e available as html objects. Packages "phantomjs" and "webshot" generate an s
```

```
creen shot of the html output that can be used as figures in PDF and Word doc
uments.

install.packages("listviewer")

listviewer::jsonedit(text_results) # List viewer generates an interactive win
dow of the Json file content.
```

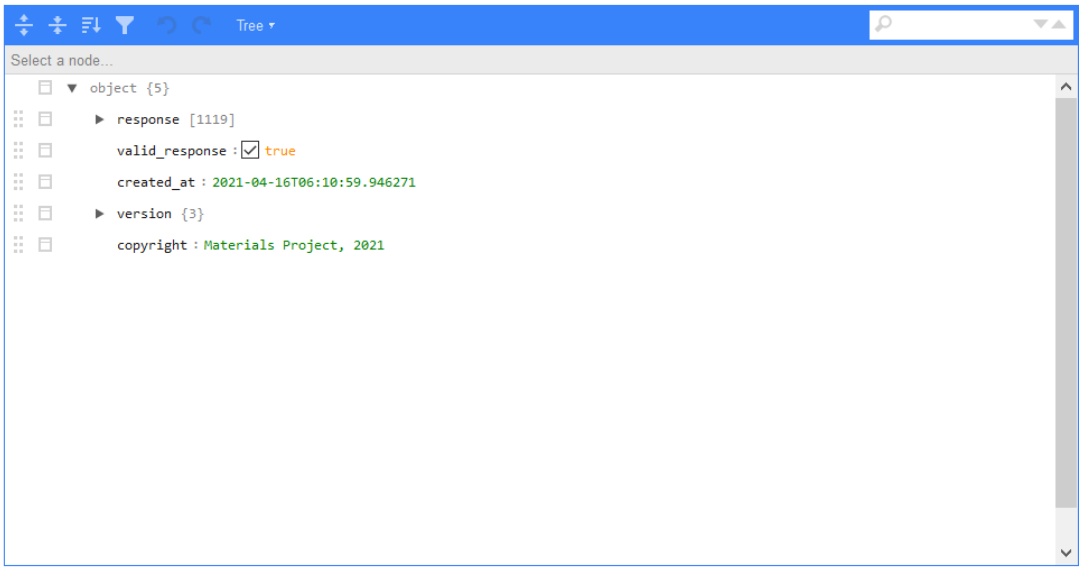


Figure 2a) A detailed structure of the json file can be viewed in the interactive tree diagram.

```
material_properties<- colnames(my.df) # We call for the columns which corresp
ond to the  material properties in the data frame.
all.proporties <- Node$new("information") # Creates a node object from "data.
tree library and call it information"
for (i in material_properties) { # We need all the properties to be displayed
in a graph. So this loop will call them from "Materials properties object and
put them add them as child nodes into the "information"node.
  information <- all.proporties$AddChild(i) # The material property (i) is ad
ded as a child (via $addchild command from data.tree library) to the "all.pro
perties data tree under "information" node.
}
tree.list <-ToListExplicit(all.proporties, unname = TRUE) # The produced Node
object is converted to list a list object which is the only acceptable data f
orm for the following ploting function.
radialNetwork(tree.list) # Plotting radial tree diagram via NetworkD3 package
.
```

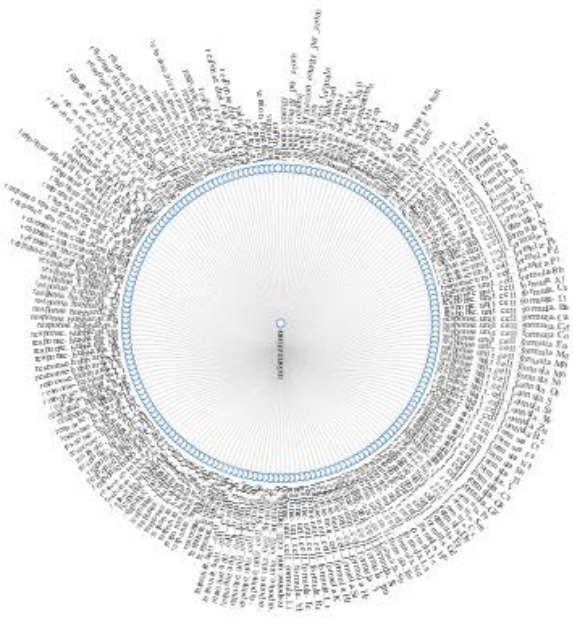


Figure 2b) Radial tree diagram shows the material properties in the dataframe.

We acknowledge the labels on the figures are small and somewhat illegible. However, the purpose of this report is to show the possible methods for data visualization. In a realistic scenario, both diagrams in this section are interactive and can be conveniently used for data mapping in r environment.

Material filtering

Training a model on a large number of unprocessed data is time consuming and often results in unreliable predictions. Therefore, it is essential to analyze the generated material dataframe based on the practicality of materials. We will use safety and the stability of compositions as filtering credentials. First, we focus on the safety of the compounds. the elements in each compound can be the indicator of material safety. For example, there are several elements such as Pb or Hg which are well known as hazardous materials. We can investigate the dataframe by generating a list of hazardous elements called "poison." A close look at figure 2b reveals that the dataframe contains a list of elements for each material. By storing the elements row in a new list (Element.list) and comparing the elements to the poison list in a 'nested for loop' the safety of the composition is evaluated.

```
Element.List<-my.df$response.elements #grabbing the elements in each compound
and saving them in an list object

poison<- c("Zn","Tl","Pb","As", "Cd","Hg","H", "Tc","Ra", "Rf","Db", "Ho", "S
g","Bh","Hs","Mt","Ds","Rg","Cn", "Nh","Fl", "Mc", "Lv", "Ts", "Og", "Ac", "P
m"," Th", "Pa","U","Np","Pu","Am","Cm","Bk","Cf","Es","Fm","Md", "No","Lr") #
The list of hazardous elements.
```

The nested for loop returns a list of elements called filter which contains all the hazardous compositions. By subletting my.data based on the elements in filter list, we generate a new dataframe called “Safe.materails.”

```
# a nested for loop is defined below t go over the element list in each row a
nd compare the elements to the elements in the "poison" list. This will final
ly will return a list of lists called "filter".
filter<-c() # This is an empty list to save the compounds that do not have  a
ny hazardous material
for (i in Element.List){ # A nested for loop is required to loop through all
the element list from the original data
  for (j in poison){ # and a second loop to go over all the elements in eleme
nt list i.
    if (j %in% i){ # there is any of the hazardous materials is in the elemen
t list then add that element list (i)to the filter object
      filter[[length(filter) + 1]] <- i
    }
  }
}
Safe.dataset <- my.df[!my.df$response.elements %in% filter,] # Subsetting the
dataset based on the values in the filter list. The (!) means "NOT". So, if t
he element combination is not in filter then the materials is considered to b
e safe.
length(Safe.dataset$response.pretty_formula) # Gets the new length of the val
ues.

## [1] 846
```

Now a dataset of clean materials generated, we focus on the stability of the compositions. Energy above hull can be defined as the stability of material in solid state respective to its decomposed form. Thus, this value can be considered as a representation of material stability. In the following, we map the stability of the initial database on a heat map. The blue points on the graph show the unstable materials and as the (e_above_hull) value approaches to zero the materials are more stable.

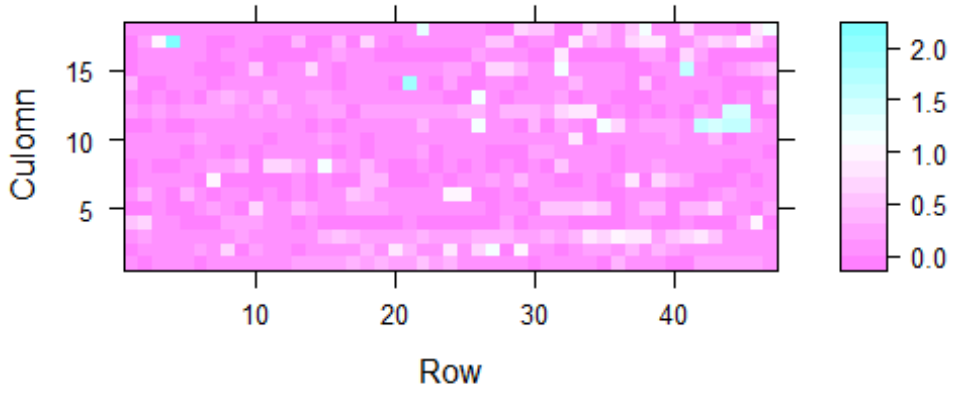
By determining a stability threshold of 0.2 eV, a new dataset “Final.dataset” is generated. We used max() function to test the stability of the materials in the dataset and a value of ~0.2 eV is returned for the most unstable material.

```
install.packages("lattice") #This package provides a convenient way for plott
ing heat maps

library(lattice)
length(Safe.dataset$response.e_above_hull) # The length of data in energy abo
ve hull column.

## [1] 846

data <- matrix(Safe.dataset$response.e_above_hull ,47  , 18) # Creating a 47x
18 matrix to cover all the 846 data points.
levelplot(data, xlab= "Row", ylab="Culomn")
```




```
Final.dataset <- Safe.dataset[Safe.dataset$response.e_above_hull <0.2,]# If t
he e_above_hull of the compound is below 0.2 eV then it will be considered st
able.

print (paste0("The maximum energy above hull in the dataset is: ", max(Final.
dataset$response.e_above_hull), " eV"))

## [1] "The maximum energy above hull in the dataset is: 0.19983416875 eV"
```

Throughout this report, data collection and processing via R programming language is discussed. The Materials project is the benchmark materials database which provides necessary information for machine learning methods. Currently, R language lacks an API for downloading data from MP database. We Designed an API which successfully is used to collecting data of perovskite structures. The collected data subsequently formatted, visualized and processed to construct an R dataframe. The generated dataframe is indeed can be used for machine learning purposes.

References

Erisman, Jan Willem, Mark A Sutton, James Galloway, Zbigniew Klimont, and Wilfried Winiwarter. 2008. "How a Century of Ammonia Synthesis Changed the World." *Nature Geoscience* 1 (10): 636–39.

Friend, Cynthia M, and Bingjun Xu. 2017. "Heterogeneous Catalysis: A Central Science for a Sustainable Future." *Accounts of Chemical Research* 50 (3): 517–21.

Jain, Anubhav, Shyue Ping Ong, Geoffroy Hautier, Wei Chen, William Davidson Richards, Stephen Dacek, Shreyas Cholia, et al. 2013. "Commentary: The Materials Project: A Materials Genome Approach to Accelerating Materials Innovation." *APL Materials* 1 (1): 011002.

Ma, Sicong, and Zhi-Pan Liu. 2020. "Machine Learning for Atomic Simulation and Activity Prediction in Heterogeneous Catalysis: Current Status and Future." *ACS Catalysis* 10 (22): 13213–26.

Wickham, Hadley. 2020. *Httr: Tools for Working with URLs and HTTP*. https://CRAN.R-project.org/package=httr.

Yang, Wenhong, Timothy Tizhe Fidelis, and Wen-Hua Sun. 2019. "Machine Learning in Catalysis, from Proposal to Practicing." *ACS Omega* 5 (1): 83–88.