

|  |
| --- |
| Data Validation using Elasticsearch |
|  |
| February 1  A joint paper by JP Tokyo & Co and MRIIRS  Authored by: Sonakshi Khosla  Reviewed by: Rajesh Nath, Charu Virmani |

Contents

[Elasticsearch 3](#_Toc536798082)

[What is elasticsearch? 3](#_Toc536798083)

Validating data………………………………………………………………………………………………………… 3

Searching for duplicare data………………………………………………………………………….. 4

Removing duplictes……………………………………………………………………………………….. 4

Data map and data plan…………………………………………………………………………………. 4

Conclusion……………………………………………………………………………………………………………… 8

# Elasticsearch

|  |
| --- |
| What is elasticsearch? **Elasticsearch** is a [search engine](https://en.wikipedia.org/wiki/Search_engine_(computing)) based on [Lucene](https://en.wikipedia.org/wiki/Lucene) library. It provides a distributed, [multitenant](https://en.wikipedia.org/wiki/Multitenancy)-capable [full-text search](https://en.wikipedia.org/wiki/Full-text_search) engine with an [HTTP](https://en.wikipedia.org/wiki/HTTP) web interface and schema-free [JSON](https://en.wikipedia.org/wiki/JSON) documents. Elasticsearch is developed in [Java](https://en.wikipedia.org/wiki/Java_(programming_language)) and is released as [open source](https://en.wikipedia.org/wiki/Open_source_software) under the terms of the [Apache License](https://en.wikipedia.org/wiki/Apache_License). Official clients are available in [Java](https://en.wikipedia.org/wiki/Java_(programming_language)), [.NET](https://en.wikipedia.org/wiki/.NET_Framework) ([C#](https://en.wikipedia.org/wiki/C_Sharp_(programming_language))), [PHP](https://en.wikipedia.org/wiki/PHP_(programming_language)), [Python](https://en.wikipedia.org/wiki/Python_(programming_language)), [Apache Groovy](https://en.wikipedia.org/wiki/Apache_Groovy), [Ruby](https://en.wikipedia.org/wiki/Ruby_(programming_language)) and many other languages.[[5]](https://en.wikipedia.org/wiki/Elasticsearch#cite_note-offizsite-5) According to the [DB-Engines ranking](https://en.wikipedia.org/wiki/DB-Engines_ranking), Elasticsearch is the most popular enterprise search engine followed by [Apache Solr](https://en.wikipedia.org/wiki/Apache_Solr), also based on Lucene.[[6]](https://en.wikipedia.org/wiki/Elasticsearch#cite_note-6)  Elasticsearch is developed alongside a data-collection and log-parsing engine called Logstash, and an analytics and visualisation platform called [Kibana](https://en.wikipedia.org/wiki/Kibana). The three products are designed for use as an integrated solution, referred to as the **"Elastic Stack"** (formerly the "ELK stack").  **Kibana** is an open source data visualization plugin for **Elasticsearch**. It provides visualization capabilities on top of the content indexed on an **Elasticsearch** cluster. Users can create bar, line and scatter plots, or pie charts and maps on top of large volumes of data.  Companies using Elasticsearch are: Ebay, IEEE, Citibank, Otto motors, NYU Library, Shopify, Godaddy, Fitbit, Microsoft, Volkswagen, Accenture, Dell, Facebook, Netflix, Walmart, Slack, BBC, Uber etc. |
|  |

VALIDATING DATA

Two kinds of validation:

* Structural validation of fields based on a regular expression for  
  example. Perhaps something can be configured in the mapping...
* Integrity validation of document. For example preventing from indexing  
  a document with a field value that already exists.

Data can be validated using Python and the Pandas library. While we validate data using pandas, it should be taken care that validation does ensure success but we need to re-check the data once to ensure correctness.

SEARCHING FOR DUPLICATE DATA:

First step for validation is removing duplicates. It is important to remove duplicates as their presence could slow down the algorithm and may give us false results outweighing our analytics. If an entry is repeated, the probability of its occurrence increases and it affects our analysis.

The following code in pandas would help us to find duplicare rows in a tabular data:

import pandas as pd

xml = objectify.parse(open(‘XMLData2.xml’))

root = xml.getroot()

dataf = pd.DataFrame(columns=(‘Number’, ‘String’, ‘Boolean’))

for i in range(0,4):

obj = root.getchildren()[i].getchildren()

row = dict(zip([‘Number’, ‘String’, ‘Boolean’],

[obj[0].text, obj[1].text,

obj[2].text]))

row\_s = pd.Series(row)

row\_s.name = i

dataf = dataf.append(row\_s)

search = pd.DataFrame.duplicated(dataf)

print (dataf)

print (search[search == True])

The code uses a modified version of the XMLData.xml file and XMLData2.xml, containing a simple repeated row in it. A real data file may contain thousands of records and likely hundreds of repeations. The example begins by reading the data file into memory and then places the data into a DataFrame.

At this juncture, the data is corrupted as it has a duplicated row. However, the duplicated row can be removed by searching it first. First a search object is created, to store the list of duplicated rows by calling pd.DataFrame.duplicated(). The duplicated rows have True next to their row number. This gives us an unordered list without any duplications. The duplicated once can be retrieved and stored separately by searching for the parameter TRUE, by search==True expression.

REMOVING DUPLICATES:

Pandas helps us to remove the duplications to get a clean dataset. The following code helps us to achieve our goal:

**import pandas as pd**

**xml = objectify.parse(open(‘XMLData2.xml’))**

**root = xml.getroot()**

**dataf = pd.DataFrame(columns=(‘Number’, ‘String’, ‘Boolean’))**

**for i in range(0,4):**

**obj = root.getchildren()[i].getchildren()**

**row = dict(zip([‘Number’, ‘String’, ‘Boolean’],**

**[obj[0].text, obj[1].text,**

**obj[2].text]))**

**row\_s = pd.Series(row)**

**row\_s.name = i**

**dataf = dataf.append(row\_s)**

**print( df.drop\_duplicates() )**

As done previously a DataFrame is created containing the duplicate record. To remove this record, drop\_duplicates() is called.

DATA PLAN AND DATA MAP

Data map is an overview of the dataset being processed. It helps to spot the problems which may exist in our dataset, such as:

1. Redundant variables
2. Possible errors
3. Missing values
4. Variable transformations

The results after checking for these problems go into the data plan, a list of tasks to be performed to ensure the integrity of our data.

A data map X with two data sets S and T, is shown in the following code:

**import pandas as pd**

**dataf = pd.DataFrame({‘X’: [0,0,0,0,0,1,1],**

**‘S’: [1,2,3,5,4,2,5],**

**‘T’: [5,3,4,1,1,2,3]})**

**a\_group\_desc = df.groupby(‘T’).describe()**

**print a\_group\_desc**

The data map uses 0s for the first series and 1s for the second series. The groupby() function places the datasets, S and T, into groups. Using describe(), statistics can be obtained to check the viability of the data map. The output of the above code would be similar to this:

**S T**

**X**

**0 count 5.000000 5.000000**

**mean 3.000000 2.800000**

**std 1.581139 1.788854**

**min 1.000000 1.000000**

**25% 2.000000 1.000000**

**50% 3.000000 3.000000**

**75% 4.000000 4.000000**

**max 5.000000 5.000000**

**1 count 2.000000 2.000000**

**mean 3.500000 2.500000**

**std 2.121320 0.707107**

**min 2.000000 2.000000**

**25% 2.750000 2.250000**

**50% 3.500000 2.500000**

**75% 4.250000 2.750000**

**max 5.000000 3.000000**

The break up of these two datasets is called data plan. The statistics show some cases which are relatively far away, as a result this plan might not be viable.

The following code can be used to break up the data:

**unstacked = a\_group\_desc.unstack()**

**print unstacked**

The output of the following code would be:

**S**

**count mean std min 25% 50% 75% max**

**X**

**0 5 3.0 1.581139 1 2.00 3.0 4.00 5**

**1 2 3.5 2.121320 2 2.75 3.5 4.25 5**

**T**

**count mean std min 25% 50% 75% max**

**X**

**0 5 2.8 1.788854 1 1.00 3.0 4.00 5**

**1 2 2.5 0.707107 2 2.25 2.5 2.75 3**

The data can be reduced in the following manner:

**B C**

**count mean count mean**

**A**

**0 5 3.0 5 2.8**

**1 2 3.5 2 2.5**

Using the following code:

**print unstacked.loc[:,(slice(None),[‘count’,’mean’]),]**

**CONCLUSION**

In the ways stated above, the data can be filtered using the Python and Pandas plug-in in Elasticsearch. In, Elasticsearch data can also be filtered using Nodejs and REST API. But Python and Pandas would be the best option to work on. It also gives us an option to visualise the analysed into bar and line graphs, pie charts etc by using the **matplot** library. It eases the graphics part of the analysis and validation and a graphical representation is better to explain the concepts to rhe business professionls, stake holders and customers who may not otherwise be comfortable enough with statistics.