Exploring Disaster Data

Author Name: Bharath Kumar Devakumar

Content

Table of Contents

Introduction

Aim to wrangle the provided data in a complete clean form and help to use this data for cleansing process.The data provided is large and messy so step by step process completion will help us to complete the task.

Input data file	Data.dat
Output data file	27559807_parsed_data.csv
Program Used	Python 2.7.11 and Jupyter notebook Anaconda version 2
Python Library	pandas, numpy , re,datetime, maketrans

Task1: Examining and loading data

To complete this task, data has to be observed and to be loaded into pandas data frame

Data Observation

Before loading the provided data into the data frame, its always better to view it. The given data can be viewed using text editor or text editor plus but its hard to have a close observation throughout the data since it is a large data though its better to have a look at the data so it can give you an idea overall.

From the data observation, I can see this data is belong to details of disaster effects across different countries where attributes like Type, Sub_Type, Killed, Affected, Cost confirms that it is a data related to disaster effects.

Loading Data

In overall Python has many different loading methods but its recommended to use pandas loading method because it has some inbuilt functionality which help us to load the day easy.

Used ‘read_table ’ function to load the data into the data frame.Parameters like

error_bad_lines = False	Helps to avoid lines with too many fields by setting it to ‘False’
header = None	To skip the first line of data in considering the values as header
sep=','	To separate the values using comma (,)
names=('Start','End','Country', 'Location','Type','Sub_Type','Names', 'Killed','Affected','Cost','ID')	List of attribute names for the separated columns
skipinitialspace=True	To skip the initial space in the dataset

By using the above parameters the data has been separated into individual columns with appropriate column names but values need to be parsed.

Task2:Parsing all the columns in the DataFrame

- The values inside each column is combined with its column name (i.e) Countries column has values like 'Countries:Australia'.
- To extract the values of each row of the column,I have used replace function with regular expression.
- Used Strip function to remove both space in the beginning and end of the string.

Parsing Individual Columns

From the above, the values extracted can be used to parse individual columns.

Start Date and End Date

Observation:

- The values in this column are not in date format i.e '07052012'.
- Some values of this column have invalid date form like '00102010' where '00' day is invalid so date is invalid though months and year are valid.
- Even though some day are invalid in the start date column, month and year data can be used. Likewise day and month may be invalid but year can be used for analysis.

Approach:

- To validate the date, I have written a custom function 'valdate' with one parameter.
- This Function can validate day based on the following month i.e '31-02-2011' – Feb month doesn't have 31 days so it is invalid.
- Function 'valdate' will return the complete date if the whole date is valid .
- It will return month and year if the day is not valid.
- Whereas, it will return year alone if day,month are not valid
- This kind of approach, helps to retrieve minimal range of valid data even though some are invalid.
- Suppose, if we perform data analysis on basis of each year where some retrieved year value can support the analysis. It is always good to have something rather nothing.

Country

Observation:

- It doesn't have missing values.
- Observed some typo based error in the column.

Approach:

- Assigned data type to be string.
- Typo errors can be rectified in data cleansing process.

Location

Observation:

- Some values in this column has special characters, continuous pull stops which makes the values meaning less.
- Missing values in this column is filled with the string 'nan'.
- Values with inappropriate case.

Approach:

- Used replace function to replace ‘:’ with space between the string.
- Translate function to replace some special characters with None.
- Followed by title() function to convert all string values with initcaps.
- Replace the string ‘nan’ to np.nan so that we can extract missing values easily.
- Assigned data type as string.

Type

Observation:

- This column doesn’t have much issues but still need care in values case.
- Need to assign data type for the column.
- Doesn’t have missing values

Approach:

- Used title function to convert the string value case into initcap.
- Assigned data type as string.

Sub Type

Observation:

- This column also doesn’t have much issues but required care in some minor issue like missing values.
- Need to assign data type for the column.

Approach:

- Replace the string ‘nan’ to np.nan so that we can extract missing values easily.
- Assigned data type as string.

Names

Observation:

- Basically name values need some additional care. Need to restrict some like
- Numbers at the start of the name
- Any uppercase without lowercase after. i.e. BBBBBJosh
- Anything that is all uppercase.
- Trailing and leading whitespace.

Approach:

- Used replace function to remove numbers.
- Used translate to remove any special character.
- Title function to change the values to initcap.
- Replace function replace the string 'Nan' to None or np.nan.

Killed and Affected

Observation:

- Space after decimal point in most of the values.
- Decimal values cannot be logically valid in this column.
- Need to assign data type.

Approach:

- Replace function to replace the decimal value with None because logically killed count may not be 0.5 or 1.5 and its not logically valid.
- Convert to numeric using 'to_numeric' handled error using 'coerce'.
- Used fillna function to replace null values with 0 because it is a integer attribute.
- Assigned data type as int64 for this attribute.

Cost

Observation:

- This column doesn’t have issues in overall.
- Need to assign data type.

Approach:

- Assigned datatype as float64 for this attribute.

ID

Observation:

- This column doesn’t have any missing values.
- Need to remove ‘-’ between the string so that it can converted to integer.

Approach:

- Removed ‘-’ symbol using replace function.
- Assigned datatype as float64 for this attribute.

Task 3:Saving Data

- The data frame has been converted into csv file for further cleansing process.

Conclusion

- I think each individual column has been parsed in a appropriate way.
- The output csv file is ready to be take for data cleansing process.