Data Cleaning DSL
Report

Hao Wang, Jingyuan Xu, Changshuang Kou
School of EECS
Oregon State University

June 9, 2016

1 Introduction

A DSL language in data cleaning DSL, which can help programmers save their time in data preprocessing steps.Data cleaning is a perivious step of data analysis. Basically it will help users turn a messy datasource into more clean one. Currently, There are some tools can be utilised for data cleaning. Such like Google Refine; also, some programming language can be used in this area, such like R and Python. However, there are some limitations when users use ready- made tools/languages.

Tools: For example in Google refine, users only have the opinions which already build-in the tool. If the wrong data type is unusual, the suggestions may hard to understand. Even users can follow the suggestions, and they also need to edit the error data manually. If this happens to a large dataset, users will lose the confidence with the data already cleansed, and worry about their analysis result.

Programming Language: R and Python have libraries for data analyzing. But first users need to make sure the packages are well written and provide methods for their host language. For example, R has dplyr package. But dplyr can not provide methods for R summary functions (e.g. mean(), or sum()) [1]. Also, users need to think about package encapsulation. If users want to add a feature to deal with a new wrong data type, they may face problems.

2 Users

The DSL can be used by programmers who knows Haskell and have a general idea in data cleaning. The primary usage for the DSL is data cleaning. If users not familiar with data cleaning, or they need visual control in the clean processing, they may need other tools to help them.

3 Outcomes

The DSL may help a programmer finish these steps in his data preprocessing:
He can use DSL to deal with the messy dataset with different rules (filter, rename, distinct). These various rules can random combine each other. For example, the programmer wants to remove the repeating data

lines, and check the specifical data is in the reasonable range (e.g. age can not be a negative number), or spelling errors, etc. Finally, he can output the cleaning result as a standard data file, or connect to a database, then show in the tables directly.
Also, our DSL is can reduce difficulty when users learn host language, and it can generate visualize result

4 Use Cases / Scenarios

Common data cleaning problems:

1. Error spelling data
2. missing data:
3. invalid data
4. Repeating data
5. Synonyms data
6. Consistency

If the input data source is single data source, then No.1 to No.3 may have more chances to happen than No.4 to No.6.

The last three problems always happen in combine multiple data sources. Such that we need some algorithm (join, duplicate, group, filter) to help us in order to correct these problems.

After we know the content of data cleaning, we need to build a cleaning model for specific cleaning data sources. Data cleaning in different industries may have different operate way, here we choose a real world case: domain information data that we download from Whois database, to demo the cleaning step.

Table 1: 2016-2-8.biz domin registration informtion in Australia (part 1).

domainName	registrarName	contactEmail
cairnsent.biz	PLANETDOMAIN PTY LTD	reception@cairnsent.com
cairnsearnosethroatcentre.biz	PLANETDOMAIN PTY LTD	reception@cairnsent.com
localmenu.biz	GODADDY.COM, INC.	grant@sitesuite.com.au
alink247.biz	TPP WHOLESALE PTY LTD.	domains@alink.com.au
flashprojects.biz	NETWORK SOLUTIONS INC.	petermclean76@gmail.com
earnosethroatcairns.biz	PLANETDOMAIN PTY LTD	reception@cairnsent.com

Table 2: 2016-2-8.biz domin registration informtion in Australia (part 2).

domainName	registrant_street1	registrant_street2	registrant_street3
cairnsent.biz	Level 1, 207 Lake Street		
cairnsearnosethroatcentre.biz	Level 1, 207 Lake Street		
localmenu.biz	Level 3	1 Bay St.	
alink247.biz	Unit 6 Trade Central 37 Keilor Park Drive		
flashprojects.biz	36 Pinkwood Drive		
earnosethroatcairns.biz	Level 1, 207 Lake Street		

Once we have the data source, then we need tools can help us do the work. In data cleaning DSL, the grammar rules are our useful tools. In the paper “Declarative Data Cleaning: Language, Model, and Algorithm“[2], the author generalize five operators to cover all cleaning steps in a data clean model. Thus, we also have similar five main operators to help us do the work.

Here are some operators we defined for datacleaning DSL. The basic operators in data cleaning DSL are:
1. delete operator
2. change operator
3. join operator
4. duplicate operator
5. filter operator

(1) Single data source: Back to the example, in table 2, which we do not need is column “registrantstreet3“, so for this single data source, we can delete column 3 with delete operator. Continue look at table 2, we found the “localmenu.biz“ domain address seems wrong, it should be “Level 3 1 Bay St.“, not “Level“, so we can use change operator in here. If we want to check if table 2 has two same domain information, we can use duplicate operator to keep one record. Now the single data source table 2 clean enough.
We can use data cleaning DSL as an example to show how to do these steps:
1. input a data source:
$d1 \leftarrow readFile$ “bizdomin1.csv“
let dset1 = input “bizdomin2.csv“
2. delete column 3
let delc= deleteColumn [3] dset1
3. output result:
output ”result.csv” delc

(2) Multiple data source: Table 1 has more important domain information, for future usage, we need to combine them together. Join operator can help us in this part. In this process, we can have other minor operators like filter, finally we can output the dataset which we want.
In here, after the three steps we shows above, we can use join operator in here to join two tables:
join two tables:
let j = joinForKey[(dset1,1), (dset2,1)]
output ”joinresult.csv” j
Such that, we can get the result dataset which contains data from two dataset.

(3) data problems we can fix with this dataCleaning DSL:
1. Estimation: change operator supports users update the wrong data and missing data. Users can use statistic method to update the data
2. Casewise deletion: delete operator and filter operator supports users filter out bad data and delete them
3. Pairwise deletion: delete operator also can use in this case– If a data pair happens issues, and it is not useful for data analysis, then remove the data pair.
4. Simple replacement policy: change operator can be used in here. e.g. use zip code instead of Postal Code, postcode

Table 3: 2016-2-8.biz domin registration informtion in Australia (part 1).

domainName	registrarName	contactEmail	registrant_street1
cairnsent.biz	PLANETDOMAIN PTY LTD	reception@cairnsent.com	Level 1, 207 Lake St
cairnsearnosethroatcentre.biz	PLANETDOMAIN PTY LTD	reception@cairnsent.com	Level 1, 207 Lake St
localmenu.biz	GODADDY.COM, INC.	grant@sitesuite.com.au	Level 3 ,1 Bay St.
alink247.biz	TPP WHOLESALE PTY LTD.	domains@alink.com.au	Unit 6 Trade Central
flashprojects.biz	NETWORK SOLUTIONS INC.	petermclean76@gmail.com	36 Pinkwood Drive
earnosethroatcairns.biz	PLANETDOMAIN PTY LTD	reception@cairnsent.com	Level 1, 207 Lake St

5 Basic Objects

For this input spreadsheet, we define type class for them. For one row, it equals to datalist, so the data type is :
type D = String
the whole spreadsheet, it equals to list of list.
type Dset = [[D]]
Every content in the cell is string type. After we have these basic types, we can create our operators.

6 Operators and Combinators

We have 5 operators in here. Delete, update, duplicate, search, and join.

(1) Search operator: The search operator give a element as key word, an input dataset, then output the search result. The result is a dataset which just contains key word row.

```
search ::  D ->Dset ->Dset
search d [] = []
search d (dlist:dset) = case elem d dlist of
                           True -> dlist : search d dset
                           False->search d dset
```

For example, we have a spreadsheet, called input.csv. Then we apply search operator with keyword 2 to this spreadsheet, then we can get the datalist contains 2.
E.g. : let s= search ”2” dset1

(2) Delete operator: We design this operator have two usages: delete row or delete column.
1. deleteColumn
The main idea in this function design is move the column out of the dataset. So we set a helper function. moveEleout. This function takes the column number of the element a, and a datalist [a], then output the final result as a pair of (a, [a]).

```
moveEleOut ::Int-> [a] ->(a,[a])
moveEleOut 1 (x:xs) = (x,xs)
moveEleOut i (x:xs) = (fst (moveEleOut (i-1) xs), [x] ++ snd (moveEleOut (i-1) xs))
```

Then we can pick second part of the pair then pattern matching to get the result dataset:

```
deleteColumn1 :: Int -> Dset->Dset
deleteColumn1 _ []      = []
deleteColumn1 i (s:ss)  = snd (moveEleOut i s) : deleteColumn1 i ss
```

2. deleteRow deleteRow is similar idea with deleteColumn.

```
deleteRow1 :: Int ->  Dset -> Dset
deleteRow1 i s= snd (moveEleOut i s)
```

(3) Duplicate operator:
In GHCI, we have nub can duplicate same elements from a list, except empty element. So if the dataset not contains empty element, we can use nub. So we can duplicate [1,2,3], not [1,2,3,]. However it is normal that a spreadsheet contains empty cells or rows. If we want to duplicate this dataset, we need a new function.
So we design dupForAll this function which fit our requirements. It need to follow 2 steps:
Step 1. check if two datalists can be duplicate
Step 2 duplicate the whole dataset.
For step 1, we have a helper function called canBeDup, this is used for checking if the input datalist satisy with the requirement. If it is true, then we can apply a dataset to dupForAll to get the result.

```
canBeDup :: [D]-> [D]  ->Bool
canBeDup [] _          = True
canBeDup _ []          = True
canBeDup (x:xs) (y:ys) = (x==ys 11 x=="" 11y=="") \&\& canBeDup xs ys
```

If canBeDup returns True, then we can go to next step. We extend nub function, and named it dupForAll. It can deal with empty cells and lists, and won't return errors.

```
dupForAll ::  Dset -> Dset
dupForAll [] = []
dupForAll x  = if isEmptyList (head (nub x))
                   then dupForAll (tail (nub x))
                  else (head (nub x)) : dupForAll (tail (nub x))
```

(4) Update value operator.
In here we have two ways to update value.
1. change value by target value, we call change by value
We have two helper functions in here to assist change value by position The first one is moveEleIn, it is similar with moveEleOut. The second one is changeHelperByPos. the case show in the slide can clear explain how it works. It takes an element, and replace it into the specific position.

```
moveEleIn :: Int-> (a,[a]) ->[a]
moveEleIn _ (x,[])     = [x]
moveEleIn 1 (x,ys)     = x:ys
moveEleIn i (x,(y:ys)) = y:moveEleIn (i-1) (x,ys)

changeHelperByPos :: Int -> D ->[D] -> [D]
changeHelperByPos i d dlist = moveEleIn i (d, snd (moveEleOut i dlist))

changeValueByPos1 :: (Int,Int) -> D -> Dset ->Dset
changeValueByPos1 (1,i) d (dlist:dset)  = changeHelperByPos i d dlist : dset
changeValueByPos1 (i0,i) d (dlist:dset) = dlist : changeValueByPos1 ((i0-1),i) d dset
```

For example, In the input.csv, we want to change cell A1s value from 1 to a, so our position in here is (1,1), and the new value is a.

2. Change value by column and row number. we call the function change value by position
Change value by search is similar ideas, It takes a Dset, an original string, and a target string as input, and output a Dset that change the original string to the target string.

Above 4 operators can be used in single or multiple data sources

(5) Join operator
The fifth operator join is used in multiple datasource environment.
Before we combine two datalist together, we need to keep the values in key column are unique:

```
dupForKey :: Int -> Dset ->Dset
dupForKey _ []       = []
dupForKey i (d:dset) = map (dupForKey2 i d) dset ++dupForKey i dset
```

Then we create function called joinForKey. This function takes two dataset with key column number, then join them together. With dupForKey, we can keep the keyvalue unique, so we can use this function safely.

(6) Main function
we use haskell library Text.csv to parse the input csv files, and transform to dataset. users have two ways to use our DSL for their data cleaning task. They can choose write line by line in the command console, or they can program the whole DSL into one file, then run the file. Here is an example to write the DSL in .hs file:
E.g. :

```
sample :: IO ()
sample = do
         $d1 \xleftarrow{}  readFile$ "w1.csv"
         let dset1 = input "w1.csv" d1
         $ d2 \xleftarrow{}  $  "w2.csv"
         let dset2 = input "w2.csv" d2
        let join = joinForKey [(dset1,1), (dset2,1)]
         output "showJoin.csv" join
         let dcol = deleteColumn [26,27] join
         output "showDeleteColumn.csv" dcol
         let cval = changeValueBySearch ["4217","2218","TPP WHOLESALE PTY LTD."] ["4219","4217
         print cval
         output "showChangeValue.csv" cval
```

7 Design Decisions

This DSL only use shallow embedding. which can make the function more clear to understand, if later users want to update our basic functions, they can easily change it. Also, because the data type we define is very easy, and we don't need deep embedding here.
To make our DSL usable in real world issues, we use IO manad and IO functions researved in Haskell prelude module as described before, and define our own input and output function to read from files and write to files.

8 Related DSLs

In the paper "Declarative Data Cleaning Language, Model, and Algorithm", the author introduce their DSL, which implement by Java, and also define operators like update, join (merge), search (view)...This gives us idea about how to design our operators.

9 Future Work

Every tools contain different limitations, like SQL, it just runs on database platform, and hardly to generate visual analysis. Excel vba need users have programming skill, same as python, R. In our DSL, we just have basic operators, and we didnt create analystic functions. Data cleaning is a very deep area, it can be combine with data mining, so machine learning method can be applied in here.Our future work will focus on using statistic method to create analysic functions, Another limitation is when the input file is very large, the performance is not very well, and in future we will try to rewrite the readFile function, in order to speed up the performance.

References

[1] "Introduction to dplyr," Introduction to dplyr. [Online]. Available at: https://cran.rstudio.com/web/packages/dplyr/vignettes/introduction.html.

[2] Helena Galhardas, Daniela Florescu, Dennis Shasha, Eric Simon, and Cristian-Augustin Saita. 2001. Declarative Data Cleaning: Language, Model, and Algorithms. In Proceedings of the 27th International

Conference on Very Large Data Bases (VLDB '01), Peter M. G. Apers, Paolo Atzeni, Stefano Ceri, Stefano Paraboschi, Kotagiri Ramamohanarao, and Richard Thomas Snodgrass (Eds.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 371-380.