Draft 2 - Text Data Processing Techniques on Trouble Ticket for Enhancing Resolution Knowledge

Fauzy bin Che Yayah

December 28, 2015

Abstract -Trouble ticket system always includes very rich data, but how to data-mining from is a very challenging task.In this paper, we applied the new technique of data processing for the ease for the next stage of classification and prediction.In another word, we also improvised the data mining technique by combining with the Bigdata technology for faster and achievable results.We realize that this method will provide luxuriant decision information for any organization or management and become a new fulfillment the field of data mining. The present report contains the results of the recommended techniques on how to process trouble ticket dataset and data discovery within it.

1 Keywords

Data mining; data processing ; Hadoop ; tickets data ; unstructured ; text processing

2 Introduction

Executing research that related to the agnostic problems within the network operator often faces the problems to get the most reliability dataset from the real system. The third party like vendors is interesting to get the dataset, but the motive is against due to confidential information inside it.Some research instituted or education oriented such as the university, do treat this kind of confidentiality in a different way.

The dataset needs to be masked and anonymized until the originality of the data set transformed into more quantitative, flexible, and adaptable to the education environment for further processing and analytics such as machine learning techniques and statistical approach. This organization more likely did not interest with the sensitive information, but they are targeting the uniqueness of the dataset and insights that can discovered within it.

The trouble ticket system for telecommunication industries produced several thousand of records every day. These records provide rich, accurate and complete for the system diagnostics, network operation practice and enhancing the customer experience by solving the problems.

3 Data sources

The dataset is originally from the one of the telecommunication company.From this source, the dataset contains millions of trouble tickets that have been downloaded via `sqoop` operation starting events from 2010 until the latest. Sqoop will periodically schedule the events to download any latest dataset available. Sqoop is the Hadoop components that are designed as a tool to transfer data between Hadoop and most of the relational database server.The main source of the dataset called as Enterprise Data Warehouse. The Enterprise Data Warehouse is a centralization location of the database that consolidates data from multiple sources that used before as the source of information for reporting, analysis, and planning.

The current records stored inside the database of all levels of servers and replicated into the Enterprise Data Warehouse (EDWH) for data staging and shared among the departments via the credential and permission for different points of usage or use case. The data are coordinated and conserved for at least for five years,

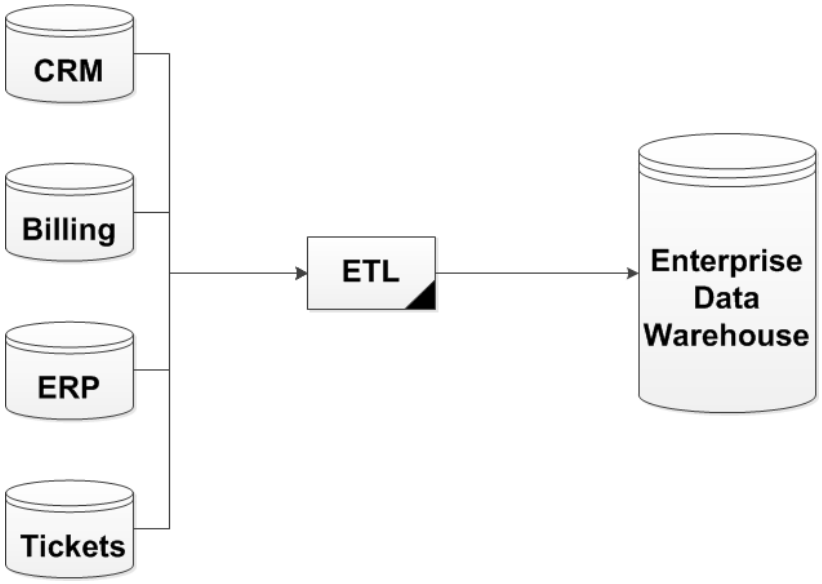


Figure 1: The Enterprise Data Warehouse

which is enough for the analytics and data discovery purpose. The remaining more than five years of data will not be discarded, and the archiving process will be applied for future reference. EDWH is optimized for database storage that any heavily queries for analytics purpose via this system is not recommended. This activity will jeopardize overall of the system performance.

4 Methodology

4.1 Trouble Ticket System (TTS)

The trouble ticket system considered as an assistant software that implements the Decision Support System (DSS) that manage and keep up lists of issues recorded by the customer support call center. The system has a knowledge-based containing the vital information for each subscribed customer, a common resolution for most of the problems, and other such data. The ticket is the details of the particular problem that has the ticket status and other relevant data. The trouble ticket can be so useful for the predictive analytics framework to feed the current dataset from it, and real-time computation can be done using pattern recognition such machine learning and data mining. Machine learning will explore and make the prediction from the dataset while data mining will generate new useful information in different perspective.

Due to the rapidly changes records operation such as update and insert from the source , EDWH also reflects this changes. The strategy to identify these changes is by scheduling it every day and get the records minimum three days before . Three days is the average restoration time targeted by the network operation for most of restoration work. This process will enrich the Bigdata platform such as Hadoop and ready for the next analytics work.

The operator defined as the location where the trouble tickets are created into the trouble tickets system via call or third party messaging system such as email. Tickets are the failure or problems records details from the customer. The trouble tickets system handle the storage of the trouble tickets case, and predictive analytics frameworks work as the external system that attached to trouble ticket system. The predictive analytics framework is the backends engine that facilitates the decision making such as prediction. Time is defined by $\mathrm{t}(n)$ that is n as the numbers of days of total trouble tickets dataset.

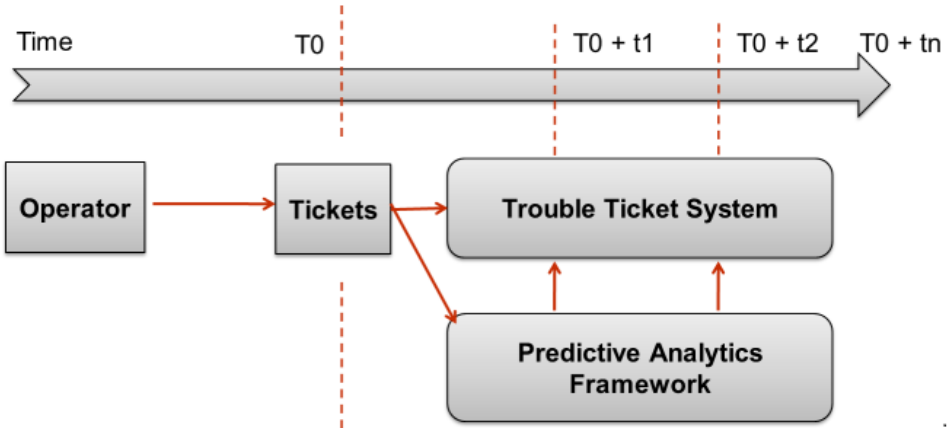


Figure 2: Trouble ticket system with predictive analytics framework

4.2 General structure of trouble ticket

The dataset of customer trouble ticket mainly comes from the trouble ticket system (TTS). The data set, then distributed into different levels of staging servers and other type database system at various levels of credential. Generally, it should contains

Properties	Notes
id	Generated ID by the system
created_date	Date of the trouble ticket creation
closed_date	Date of the trouble ticket closed
cause_category	Category Code define by the system
cause_code	Cause Code define by the system
resolution_code	Resolution Code define
description	Detail textual information such as customer information, session status , system status, network status , customer history , etc.
....	....

To interpret the downloaded dataset, it is necessary to overview the lifecycle of the tickets which also helps to understand more on how the operation work.

A ticket is created whenever equipment or service failure that is discovered by the operation team and have impacted to the network or complain from the customer site. The opening time varies depending on the time of problems discovery or the time of record entry into the database by the customer services. Moreover , the time of closing has multiple possible conditions. It may be the time when the failure was solved or batch periodically check when the tickets are outdated.

In general, when the ticket raised, the status is set to `Open`. This event recorded and required action from the team in field or network operator. This activity recorded into preliminary info that called as Service Request (SR). If the issue is solved via the telephone conversation, the general trouble tickets or network trouble ticket

will not be created. If this issue required more explanation and troubleshooting in details, new trouble ticket would be raise. The `open` status of trouble ticket requires more attentive from the operation team on the field. The tickets remain open until it states explicitly change to `closed` after the problems solved.

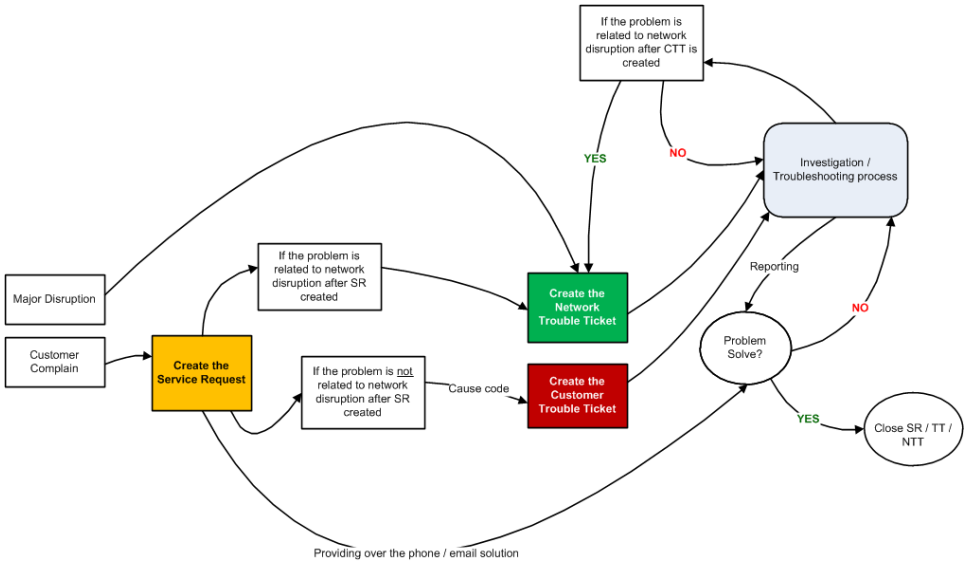


Figure 3: Trouble ticket system workflow

The purpose of the cause category fields contains the primary group of the problems defined by the operator. More details category is defined in the cause code. All the code listed in predefined by the system and the selection method is chosen manually based on the symptom analyzed and understood by the system operator.For more details regarding the symptom, the information received via the phone conversation is summarized in the free text section that is `description` section.

For more details regarding the symptom, the information received via the phone conversation is summarized in the free text section that is `description` section which written in the human language. Thus, the rational this particular part is extra information can be described, and the technical person in charge can more understand the real situation of the problem. Information is transfer from stage to stage, and this is needed to track back the history of the problems. The description part required interpretations and text mining operation, and this might be challenging if the language is different or the word used is not within the general dictionary.[1] Thus, most of the data is not explicitly structured, it is also highly noisy (e.g, inconsistent formatting) and also very heterogenous (e.g., multiple language, such as Bahasa Melayu or English. System-generated data, specific telecommunication term, abbreviation, acronyms, and jargons), making it hard to analyze. We introduced an approach for automatically search for specific knowledge or topics from the collection of the free form text section later.

The table below shows the example of the data type and the records inside the trouble tickets dataset.

Column Type	Example	Field Name
Integer	[1001] , [1002] , [1003]	ID
DateTime	[2012-09-05 19:39:01.0],[2012-09-05 19:45:39.0]	created_date
Text	*"Troubleshooting Step: -check outages -check account status"*	description
String	[Performance] , [Failure], [Open], [Closed]	status
....	....	....

[1]Ye, Yangdong, Jing Zhang, Junwei Gao, and Limin Jia. "The Application of Decision Tree Induction of Classification in Train Tickets System."Proceedings. International Conference on Machine Learning and Cybernetics (2002): n. pag. Web. [?].

4.3 Processing Trouble Tickets contents

Our effective methodology automatically structuring free-from heterogeneous textual data helps to identify the structural pattern specific pattern known to the call center or technician that helps them to speed up the tedious work manually searching and understanding the relevant data inside it . The category field inside the trouble tickets dataset contains relevant information between the description field. To categorize the keywords inside the description requires full contents of the records. Words that which belong to particular category must have at least one or more than association. Analysis of the categorization of words by the genre such as events, hardware, software, nodes and links listed below

Category	Keywords
Event	*run,show,through,job,day,backup,shuttle..*
Hardware	*home,fibers,power,ups,modem,reboot,shutdown..*
Software	*windows7,reboot,configuration,shutdown,slow ..*
Nodes	*routers,modem,switches,power,equipment..*
Links	*connection,cards,interface,waves,ping..*

The categories are crucial for the network operation to learn about the significant impact when the data set transformed into statistical representation. This point will be explained in the next section with the presentation of the results.

4.3.1 Acquiring trouble tickets dataset into Hadoop

To acquire trouble tickets dataset from the EDWH, we need special operation using `Sqoop` components inside Hadoop into Hive compatible format. Hadoop is an open-source software framework for storing data and running applications on clusters out of commodity hardware. Hive defined as an infrastructure built on top of Hadoop for providing data summarization, query, and analysis. Sqoop works by generating Java defination class to deserialize data from the data source for use with native Hadoop input format connectors such as JDBC. It creates and launch MapReduce job to automate the import process.MapReduce is a framework that makes the distributed processing inside the cluster. [2] . This paper also shows more details of the sqoop operation workflow inside the Figure 4.

[2]Chen, Changai, and Shan Jiang. "Research of the Big Data Platform and the Traditional Data Acquisition and Transmission Based on Sqoop Technology." TOAUTOCJ The Open Automation and Control Systems Journal 7.1 (2015): 1174-180. Web. [**?**].

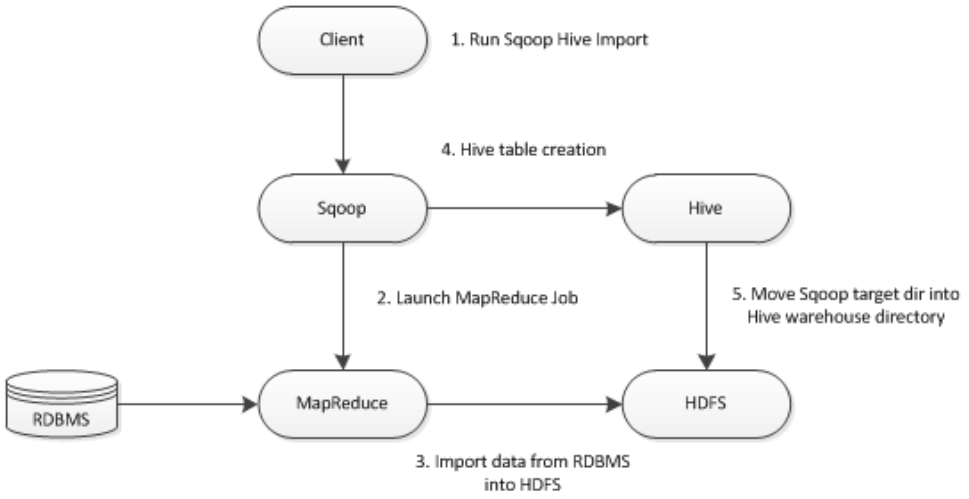


Figure 4: Hadoop - Sqoop - Hive Operation

To execute this operation, basic syntax for sqoop operation below needed for importing trouble ticket dataset from data warehouse as follows

```
sqoop import --connect jdbc:[rdbms driver]:@[ip_address]:[port]/[db_name]
--username [username] --password [password]  --target-dir [temp_target]  --verbose
-e "select [field[1:n],REGEXP_REPLACE([desc_field_name],
'[^a-zA-Z0-9\\:\\@\\/\\#\\-]', ' ') " --m 1 --hive-table
[dbname_in_hdfs][tablename_inhdfs] --hive-import
--append --fields-terminated-by '\001'
--lines-terminated-by '\n' --hive-delims-replacement '';
```

The description of the syntax shows more explanation for every parameter passed to the sqoop component

Syntax	Description
sqoop import	Import a table from a database to HDFS
–connect jdbc:[rdbms driver]	Specify JDBC connect string
[ip_address]:[port]/[db_name]	Provides ip address , port no and rdbms database name
–username [username]	Provides the username
–password [password]	Provides the password
–target-dir [temp_target]	Specify the target path for the output of the merge job
–verbose	Print information while execute
-e	specified query statement
select	statement returns a result set of records from one or more tables
[field[1:n]	use select * for selecting all fields available
REGEXP_REPLACE([desc_field_name],)	search a string for a regular expression pattern
‘[^a-zA-Z0-9\\:\\@\\/\\#\\-]’, ‘’)	remove all special characters and numbers
[-m 1]	Use n map tasks to import in parallel
[–hive-table	Sets the table name to use when importing to Hive
[[dbname_in_hdfs]	Sets the db name in Hive/HDFS
[tablename_inhdfs]	Sets the db name in Hive/HDFS
–hive-import	Import tables into Hive format

Syntax	Description
–append	Append data to an existing dataset in HDFS
–fields-terminated-by '\001'	Sets the field separator character
–lines-terminated-by '\n'	Sets the end-of-line character
–hive-delims-replacement ''	Replace delimiter from string fields with user defined string

A safer techniques to reside trouble ticket dataset into the Hive table by converting any special character using a special sqoop built-in function like `REGEXP_REPLACE` while acquiring the dataset. By replacing with default delimiter with `NULL` or empty char. To achieve this , applying the `--hive-delims-replacement '' ` and `REGEXP_REPLACE [^a-zA-Z0-9\\:\\@\\/\\#\\-]` inside the sqoop command line . The comparison images below shows the clean up results before and after applying the regular expression filtering while sqoop the dataset from the EDWH.

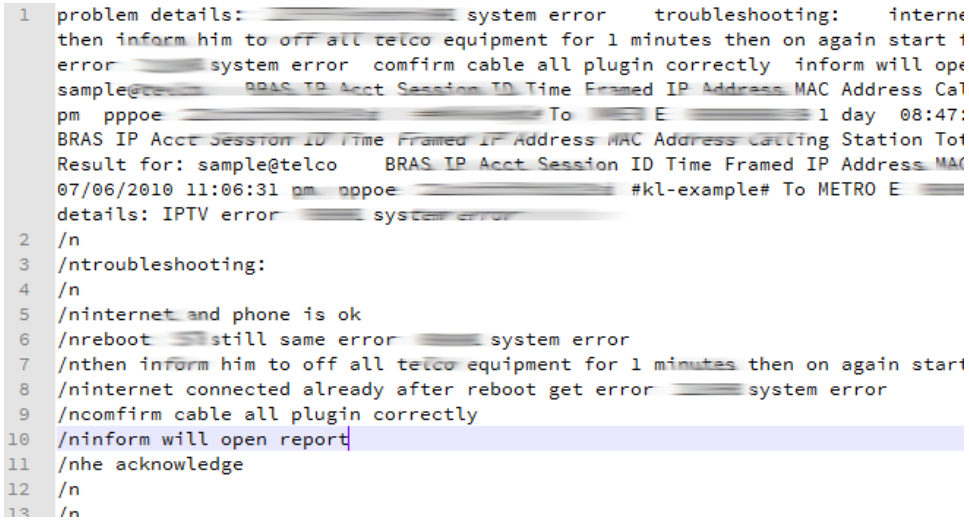


Figure 5: Original results - description field

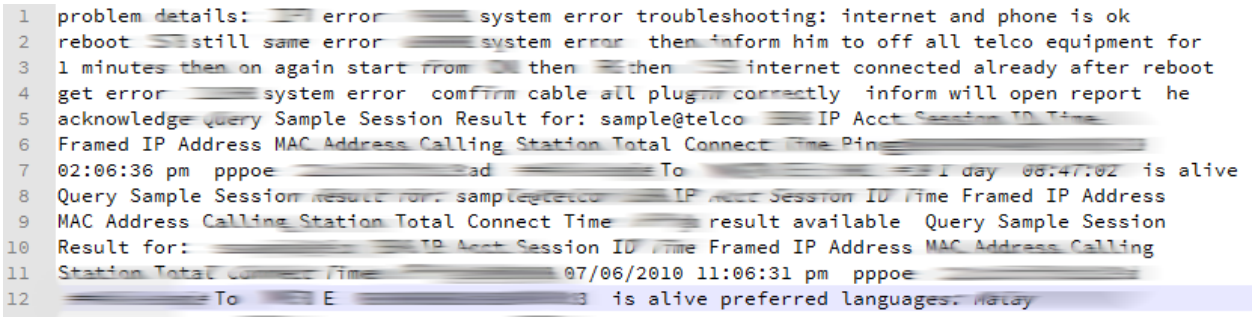


Figure 6: Cleaned results - description field after REGEXP_REPLACE via sqoop

Figure 7: Sqoop operation in console

4.3.2 Applying index number for TTS dataset inside Hive

Default Hive dataset does not have any `auto number` data type. The `auto number` is very useful for primary key operation or filter the records. We applied the auto number to the TTS dataset after the Sqoop operation finished. Because of the data set that resides inside Hadoop is in Hive format, we need to use external jar file that is `hive-contrib-0.12.0-cdh5.1.2.jar` to create auto number for the entire records. The jar file can be executed via JAVA by creating custom function as follows

```
drop function IF EXISTS row_sequence();
create function IF NOT EXISTS row_sequence()
returns BIGINT location 'lib/hive-contrib-0.12.0-cdh5.1.2.jar'
symbol='org.apache.hadoop.hive.contrib.udf.UDFRowSequence';
```

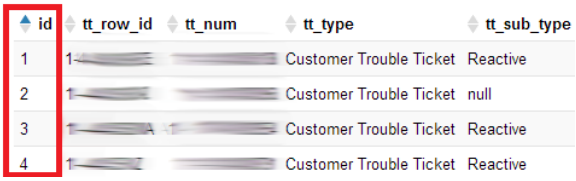
Drop function applies as to drop any defined function while create will new user defined function.

The rules applying the auto number to the TTS dataset are to set the records limits inside the Hive-SQL. This operation is to ensure the number of dataset creation is decrease into a single file inside the HDFS. Single HDFS file is faster to access by Hadoop. Without the limit statement, the Hive dataset files are chunked into the multiple split and the auto number sequence is set in 0 for each split or `single reducer`. This method will cause the each record filled with unique auto number ID. The sample of the Hive-SQL‘ command for this operation can be achieved by this example

SQL-Hive	Description
`drop table IF EXISTS tts_stage1;`	Create table statement if exist
`create table IF NOT EXISTS tts_stage1`	Create table and add autonumber as the first field
`as select row_sequence() as id ,`	before the TTS fields
`* from [tts_table] limit 10000;`	Select all the fields from original TTS and limits for `single reducer`

tt_row_id	tt_num	tt_type	tt_sub_type
[redacted]	[redacted]	Customer Trouble Ticket	Reactive
[redacted]	[redacted]	Customer Trouble Ticket	null
[redacted]	[redacted]	Customer Trouble Ticket	Reactive
[redacted]	[redacted]	Customer Trouble Ticket	Reactive

Figure 8: Initial Hive trouble ticket dataset after sqoop operation



id	tt_row_id	tt_num	tt_type	tt_sub_type
1	[redacted]	[redacted]	Customer Trouble Ticket	Reactive
2	[redacted]	[redacted]	Customer Trouble Ticket	null
3	[redacted]	[redacted]	Customer Trouble Ticket	Reactive
4	[redacted]	[redacted]	Customer Trouble Ticket	Reactive

Figure 9: Following Hive trouble ticket dataset after applying autonumber in Hive-SQL

The advantages of doing this method are it makes the queries much faster. Uniques number indexes guarantee identifiable records inside Hadoop. Limiting the records now is possible by adding a condition to the HiveSQL statement. Example like `id > 2` and `id < 5` will fetch records that have id 3 and 4.

4.3.3 Second filtering of TTS dataset

Since the dataset now are filled by autonumber `id`, now the record can be filtered by the `id` range . Tools such as RStudio which implements the R language statistical is used in this process. RStudio is capable to connect to various datasources including Hive. In this section , Open Database Connectivity(ODBC) for Hive need to be installed and configured. ODBC for Hive can be downloaded from the Cloudera website (http://www.cloudera.com/content/www/en-us/downloads/connectors/hive/odbc/2-5-12.html).

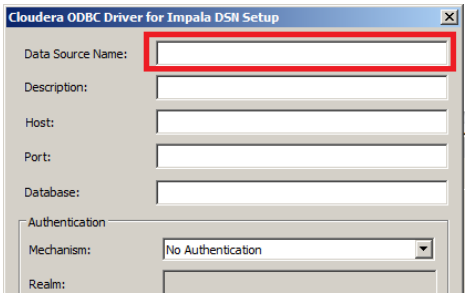


Figure 10: Hive ODBC Datasource Administrator

Set the ODBC data source name *[dsn_name]* , the host address , the port number (default for *Impala* is 21050 ; *Hive* is 10000) , and the database name *[db_name]*. After that , test the ODBC connection.

After the ODBC System DSN name have been set , use RStudio create new `RScript` and implements the R code below for connnecting to Hive ODBC via RODBC package. Package RODBC implements ODBC database connectivity. (https://cran.r-project.org/web/packages/RODBC/vignettes/RODBC.pdf).The Rstudio can be downloaded from the Rstudio website. (https://www.rstudio.com/products/rstudio/download/).

- Sample RScript to implements RODBC as follows :-

```
library(RODBC)
conn <- odbcConnect(dsn="[dsn_name]", uid="[username]", pwd="[username]") #
queryResult <- sqlQuery(conn, "select desc from [db_name].[hive_table_with_id]
where id > 0 and id <= 10000 ")
odbcClose(conn)
```

R Script	Description
`library(RODBC)`	load the RODBC instance components
`conn <- odbcConnect(dsn="[dsn_name]"),`	Set the DSN name
`uid="[username]", pwd="[username]"`	set username and password
`sqlQuery(conn, "select desc from [hive_table_with_id]"`	Hive-SQL to filter the records
`"where id > 0 and id <= 10000"`	Condition to select the records
`odbcClose(conn)`	Cleans up and frees resources

Five stages of data filteration need to be applied to clean up description text from the TTS information

#	Description	R Script with Regular Expression filter
1	Lowercase all the charactors	`df_desc <- as.data.frame(tolower(df_desc$review))`
2	Remove all the numbers	`df_desc <-as.data.frame (sapply(df_desc,gsub,pattern="[[:digit:]]",replacement=""))`
3	Remove all the punctuation	`df_desc<-as.data.frame`

#	Description	R Script with Regular Expression filter
	and symbols	`(sapply(df_desc,gsub,pattern="[[:punct:]]",replacement=""))`
4	Trim the unnecessary spaces	`df_desc <-as.data.frame (sapply(df_desc,gsub,pattern="@\\w+",replacement=""))`
5	Trimming the whitespace	`df_desc <-as.data.frame (sapply(df_desc,gsub,pattern="^\\s+|\\s+$",replacement=""))`

The image below shows state of text description before applying the regular expression by the RScript. The description text contains irrelevant character types like numbers , special symbols , spaces , etc.

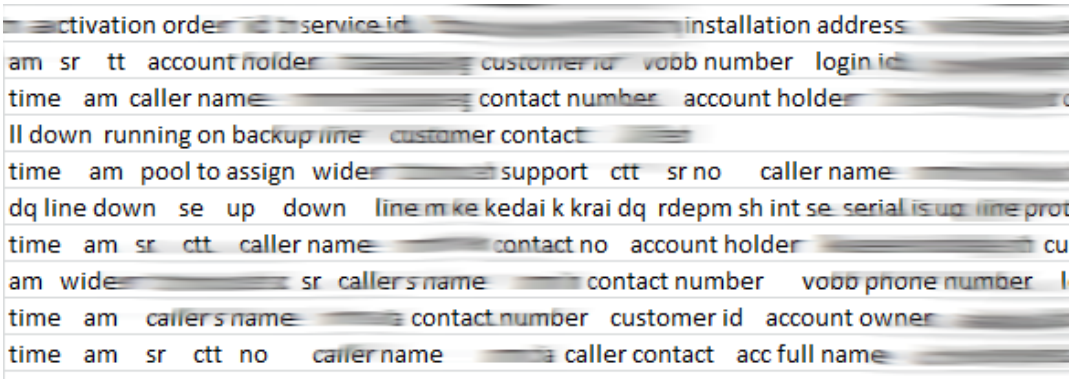


Figure 11: description text on before filteration

After appying the five steps of filteration , now the text description is suitable or the next stage of transformation which is Document Term Matrix (DTM) . DTM is a mathematical matrix that describes the frequency of terms that occur in a collection of documents. In this case the DTM will tied to each of the TTS records.

4.3.4 Create a Document Term Matrix (DTM)

Document Term Matrix (DTM) is a matrix based dataset which describes the frequency of terms that occur inside collection of dataset especially the text based types. The DTM was spesifically designed to have rows and columns. Rows is correspond as related unique rows records (*documents*) and columns as the unique terms which exist inside the selective column dataset.

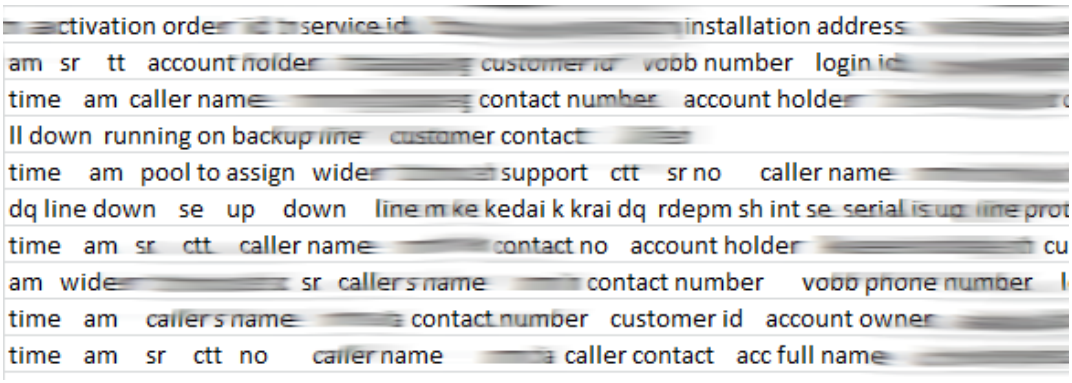
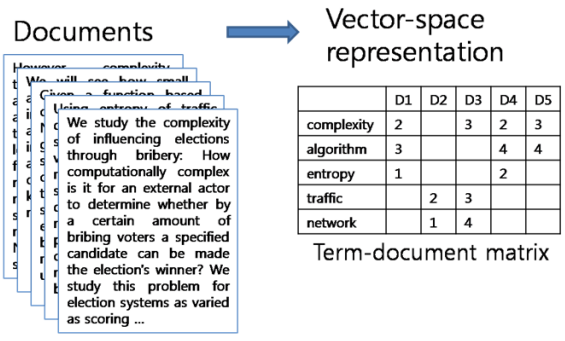


Figure 12: description text on after filteration



	D1	D2	D3	D4	D5
complexity	2		3	2	3
algorithm	3			4	4
entropy	1			2	
traffic		2	3		
network		1	4		

Figure 13: Document Term Matrix

The RScript below implements the Document Term Matrix function :-

```
mach_corpus = Corpus(VectorSource(df_review$review))
tdm = TermDocumentMatrix(mach_corpus,control = list(removePunctuation = TRUE,
stopwords = c(stopwords("english")),
removeNumbers = TRUE, tolower = TRUE))
m <- as.matrix(tdm)
```

The description of R Script as follows :-

R Script	Description
`Corpus(VectorSource(df_review$review))`	Representing and computing on `corpora`.
`TermDocumentMatrix`	Constructs or coerces to document-term matrix.
`c(stopwords("english"))`	Return various kinds of stopwords
`removeNumbers = TRUE`	Remove numbers from a text document
`as.matrix(tdm)`	Turn argument into a matrix

Example of generated Document Term Matrix from TTS dataset as follows :-

Table 9: Document Term Matrix of TTS

tt_num	access	alarm	already	atpmgitn
dfcc-1-12314522618	0	0	0	0
dfcc-1-12314390988	0	0	0	0
dfcc-1-12313258208	0	0	0	1
dfcc-1-12312911038	0	0	0	0
dfcc-1-12311810828	0	0	0	0
dfcc-1-12311581758	0	0	0	0
dfcc-1-12311375368	0	0	1	0
dfcc-1-12310229828	0	1	1	0
dfcc-1-12307977728	1	0	1	0
dfcc-1-12306912058	0	0	0	0

```
## <<DocumentTermMatrix (documents: 20, terms: 103)>>
## Non-/sparse entries: 393/1667
## Sparsity           : 81%
## Maximal term length: 31
## Weighting          : term frequency (tf)
```

The information above summarize that the sparsity level is above 80% and removal of sparse term is needed for the next process . Some term like `access` , `alarm` , `already` , `atpmgitn` is low in frequency which proves all TTS `desc` sparsity level.

4.3.5 Sparse Terms Removal

Sparse Terms is the term which has low frequency . This can help the generalization process and prevent overfitting into the model. The example show the R Script with removal of 60% of the sparsity level inside the TTS `desc` field.

```
tdm <- removeSparseTerms(tdm, sparse= 0.6 )
```

Sample of generated Document Term Matrix after removal of 60% (0.6) sparsity term below :-

Table 10: DTM removeSparseTerms

tt_num	check	contact	downno	end
dfcc-1-12314522618	3	1	1	2
dfcc-1-12314390988	1	0	1	1
dfcc-1-12313258208	0	0	0	0
dfcc-1-12312911038	3	1	0	2
dfcc-1-12311810828	0	0	0	0
dfcc-1-12311581758	2	1	0	2
dfcc-1-12311375368	0	0	1	1
dfcc-1-12310229828	0	0	1	1
dfcc-1-12307977728	0	0	0	1
dfcc-1-12306912058	2	1	0	2

```
## <<DocumentTermMatrix (documents: 20, terms: 17)>>
## Non-/sparse entries: 220/120
## Sparsity           : 35%
## Maximal term length: 14
## Weighting          : term frequency (tf)
```

After applying the remove sparse term filter , the DTM sparsity level was reduced to 35% . Now the DTM dimension is applicable as part of the predictive model.

4.3.6 Vectorization of TTS (VTTS)

To enable dataset for prediction data processing, we need to convert all the independent variables inside the TTS which is in factor format (*code status*) into numeric format. In this sample we select only one of the fields which is the `cause code` as the sample for vectorizing the dataset. This technique will be applied to all non-numeric field formats inside the TTS dataset.

Table 11: Sample TTS Cause Code

tt_num	cause_code
dfcc-1-12315710868	* disconnect in internal wirin
dfcc-1-12315710637	nt mat6d faulty
dfcc-1-12315707678	* sub system / cpe

tt_num	cause_code
dfcc-1-12315693158	* sub system / cpe
dfcc-1-12315693058	others pcm

To change the cause code into a vector , we need to apply the R Script code below to convert all the factor into numerical format.

```
for(i in names(df_all)){
  num <- as.numeric(as.factor(df_all[,i]))
  df_all <- cbind(df_all,num)
  names(df_all)[names(df_all)=="num"] <- paste(names(df_all[i]),"_factor",sep = "")
}
```

`as.numeric` is a function which creates the objects into numeric and `as.factor` encodes a vector as a factor. A table below shows the result of the encoding from the cause code factor `status` into the `numerical` format.

Table 12: Sample TTS Cause Code Vectorization

tt_num	cause_code	cause_code_factor
dfcc-1-12315710868	* disconnect in internal wirin	1
dfcc-1-12315710637	nt mat6d faulty	110
dfcc-1-12315707678	* sub system / cpe	9
dfcc-1-12315693158	* sub system / cpe	9
dfcc-1-12315693058	others pcm	126

5 Results

5.1 VTTS with DTM

The final steps of this techniques is to combine all the vectorization of TTS with the DTM. All the standard name of each selected column will be rename into `[fieldname]_factor` format. Example shows below :-

Original column name	Vector column name
created_by	created_by_*factor*
account_num	account_num_*factor*
product_factor	product_*factor*
sub_product	sub_product_*factor*
symp_err	symp_err_*factor*
...	...

The construction design of the table structure as follows :-

- Select the `tt_num` (*ticket number*) columns. Tt_num is the `key` of the TTS dataset.

```
df_tt_num <- as.data.frame(queryResult$tt_num)
```

- Selecting all the factor columns

```
df <- as.data.frame(queryResult)
names(df)[1] <- "created_by_factor"
names(df)[2] <- "account_num_factor"
names(df)[..n] <- "[..x]_factor"
....
replace [..n] with column number and [..x] is the column name
....
```

- Binding all the factor columns with the `tt_num`

```
df <- cbind(df_tt_num,df)
```

- The results after binding `tt_num` with some of the `factor` columns and the `tdm` matrix

Table 14: Ticket No , TTS Vectorization and TDM table (continued below)

tt_num	nova_account_num_factor	account_num_factor
1-8418913314	49	16
1-8418913153	49	16
1-8419016970	57	8
1-8419108779	9	19
1-8418913441	88	38

Table 15: Table continues below

[x]_factor	account	acct	cable	call	check
...	1	0	0	0	1
...	1	0	0	0	1
...	0	0	0	0	0
...	1	1	0	1	0
...	0	0	0	0	0

cust	customer	hsi	name	next	session
0	2	1	2	2	1
0	2	1	2	2	1
0	0	0	0	0	0
1	0	1	0	0	1
0	0	0	0	0	0

The table above show that combines three elements which is `TT_NUM` , `TTS` vector and the `DTM` matrix. Now all the field is ready to utilize for further analysis such as classification and prediction .

6 References

N.p., n.d. Web.
Ye, Yangdong, Jing Zhang, Junwei Gao, and Limin Jia.
"The Application of Decision Tree Induction of Classification in Train
Tickets System." Proceedings. International Conference on Machine Learning and
Cybernetics (2002): n. pag. Web.

Ye, Yangdong, Jing Zhang, Junwei Gao, and Limin Jia. "The Application of Decision
Tree Induction of Classification in Train Tickets System."Proceedings. International
Conference on Machine Learning and Cybernetics (2002): n. pag. Web.

Marcu, Patricia, Genady Grabarnik, Laura Luan, Daniela Rosu, Larisa Shwartz, and Chris Ward.
"Towards an Optimized Model of Incident Ticket Correlation."2009 IFIP/IEEE
International Symposium on Integrated Network Management(2009): n. pag. Web.

Shao, Qihong, Yi Chen, Shu Tao, Xifeng Yan, and Nikos Anerousis. "EasyTicket." Proc.
VLDB Endow. Proceedings of the VLDB Endowment 1.2 (2008): 1436-439. Web.