Predict Resolution set for Technical Tickets

1 Project Scope

Scope of this project is to explore and analyse the Problem-Abstracts from tickets reported by clients/customers. Customer/clients raise tickets whenever they have issues or interruptions in service offered through various applications/systems across organization.
Each Ticket has a "Problem-Abstract" that explains about the technical issue faced by customer. Based on Technical issue, these tickets need to be routed to concerned teams for fixing issues. Each fix provided by technician has a "Resolution_set" associated with it based on root_cause of the issue.

Our Aim is to train a model that reads the problem abstract from customer and predict resolution_set name for each ticket before routing to a technician. This will help reducing delays in routing the tickets from one team to other, reduce the time for identifying the root_cause and enables faster resolution of the issues.

2 Data Understanding

Ticket_Resolution data:

This dataset contains the data of all the historical tickets that include Problem_Abstract, root_cause of the ticket, open date, resolution_set, organization(department that resolved the ticket) and other details.

Ticket_Logs:

This dataset contains the logs for all historical tickets with respective organisation names that have taken action on ticket and log_messages and data of action.

3 Data Cleaning & Preparation

The datasets are pretty much clean and did not require much cleaning except for few missing values. We have filled missing values in categorical data with a class “Unknown”.

Consolidated logs for each ticket from ticket_logs into a single log with “date”, “organization” that created log and “message”.

Calculated, ticket life-time as “fix-days” from the dates available in ticket_logs.

Created a single data-frame ticket_db with below listed key attributes:

```
'Ticket', 'Ticket_Opened', 'Ticket_State', 'last_modified_date',
'Problem_Abstract', 'fix_days', 'ticket_log_message',
'Root_Cause_Desc_Tx', 'Resolution_Org', 'Resolution_Set_Name','Dispatch
_Text'
```

4 Exploratory Data Analysis Network graphs

After preparing dataset with key attributes for ticket-analysis, We created Bipartite-Multi digraph that represents all tickets as "ticket-nodes" and all organization that take actions on tickets as "org-nodes". Each ticket-node is connected to one or more org-nodes. Each edge represents an action taken by corresponding "org-node" on the "ticket-node".

Then we have calculated bipartite-degree centrality for all the nodes in the ticket_graph.

Sample ticket-node = *189785518: {'bipartite': 'tkt', 'open_date': Timestamp ('2014-09-04 15:45:31'), 'closure_date': Timestamp('2015-01-28 19:44:13 '), 'fix_days': 146, 'Root_Cause': 'Application', 'Resolution_Set_Name' : 'HD - Workaround', 'Dispatch_Text': 'No', 'Resolution_Org': 'UnKnown' , 'dc': 1.0}*

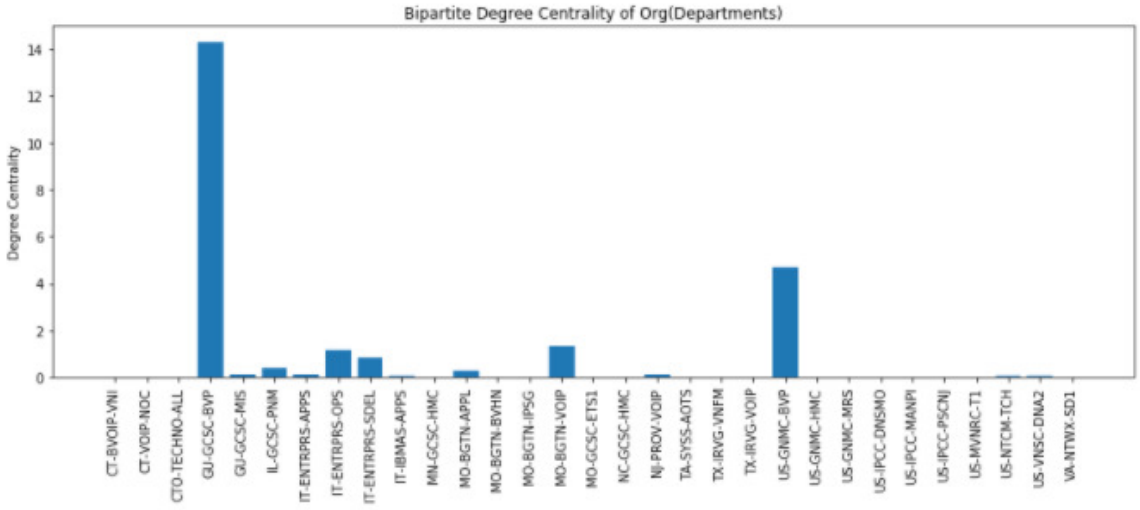
Sample org-node = *'TX-IRVG-VOIP': {'bipartite': 'org', 'dc': 0.002400384061 449832}*

Sample edge = *(192805683, 'GU-GCSC-BVP', {'Ticket': 192805683, 'Ticket_Log _Parse.Active_Org': 'GU-GCSC-BVP', 'Ticket_Last_Modified_Date': Timesta mp('2015-01-28 00:08:36'), 'Ticket_Message': ' Automation successfully synced state to Closed from IE 000000192810745'})*

We have defined new custom functions to override base functions in "NxViz" package to color and group the nodes as per attribute values.

Top 5 departments to work on tickets with degree centrality:

	org_name	degree_centrality
0	GU-GCSC-BVP	14.262136
1	US-GNMC-BVP	4.699029
2	MO-BGTN-VOIP	1.305825
3	IT-ENTRPRS-OPS	1.150943
4	IT-ENTRPRS-SDEL	0.849057

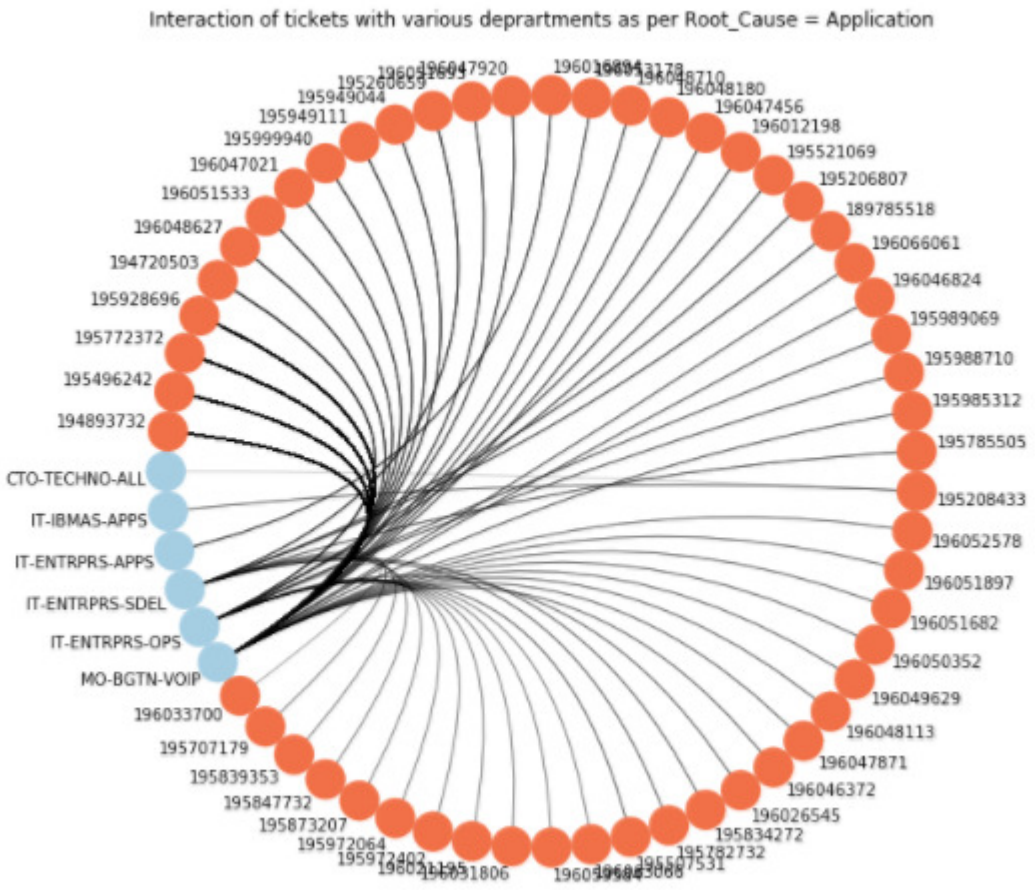


From above bar-chart you can notice that organization that servers most number of tickets is "GU-GCSC-BVP" with bipartite degree-centrality of 14.26. The next department that servers most number of tickets is "US-GNMC-BVP" with bipartite degree-centrality of 4.69.

We have further analysed to know what are the departments that work on most number of tickets with root-cause = "Application". From below table we can see "MO-BGTN-VOIP" works on most number of tickets with "Application" as root-cause.

list of departments that work on tickets with Root_Cause= Application :

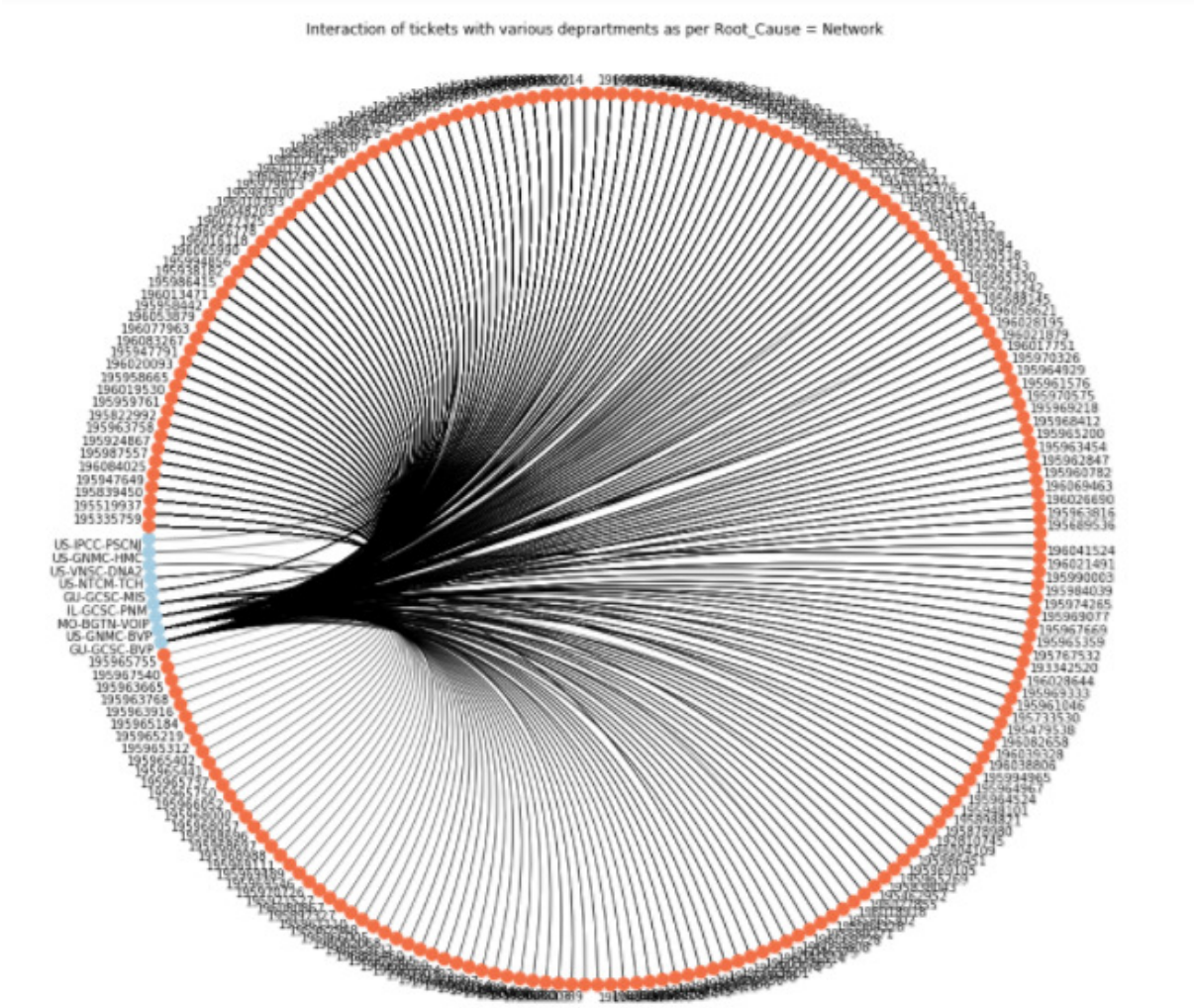
	org_name	dc_in_subgraph
0	MO-BGTN-VOIP	40.094340
1	IT-ENTRPRS-OPS	1.150943
2	IT-ENTRPRS-SDEL	0.849057
3	IT-ENTRPRS-APPS	0.132075
4	IT-IBMAS-APPS	0.075472
5	CTO-TECHNO-ALL	0.018868



Similarly “GU-GCSC-BVP” department works most number of tickets with root-cause “Network”.

list of departments that work on tickets with Root_Cause = Network :

	org_name	dc_in_subgraph
0	GU-GCSC-BVP	14.262136
1	US-GNMC-BVP	4.699029
2	MO-BGTN-VOIP	1.305825
3	IL-GCSC-PNM	0.368932
4	GU-GCSC-MIS	0.111650
5	US-NTCM-TCH	0.067961
6	US-VNSC-DNA2	0.043689
7	US-GNMC-HMC	0.029126
8	US-IPCC-PSCNJ	0.024272



When you compare above Circos Plot with bar-chart we can correlate the fact that most number of tickets are identified with root-cause = "Network" .

5 Modeling & Evaluation

We have started our modelling with tokenization of the problem_abstract from sample records.

Tokenization is a process of building a corpus that consists all the words from entire set of sample documents. After tokenizing the sample documents, we calculate TF-IDF (Term Frequency-Inverse Document Frequency) vector from Word-Corpus and individual sample documents.

Tf-idf formula:

$$w_{i,j} = tf_{i,j} * \log(\frac{N}{df_i})$$

$w_{i,j}$ = tf-idf weight for token i in document j

$tf_{i,}$ = number of occurences of token i in document j

df_i = number of documents that contain token i

N = total number of documents

Word-Count vectorizer, simply calculates the frequency of each word in a given document and does not take other language and context aspects into consideration.

Tf-idf vectorizer is much more efficient than a simple word-count vectorizer as tf-idf down-weighs the importance most common words(eg: stop words) and document specific words will be weighted high.

After calculating Tf-idf vectors for sample documents, we have tried modelling the vectors to predict given predictor variables(eg: Root_Cause of problem, Resolution_set for given problem.).

We used, Naïve-Bayes and Random Forest classifiers from scikit-learn python libraray for modelling.

It is observed that Random-Forest classifier has given better prediction results compared to that of Naïve-Bayes classifier across various predictor variables.

Predicting Resolution_set_name:

Model Name	**Accuracy Score**
Naive Bayes with tfidf_vector_model	0.6
RandomForest with tfidf_vector_model	0.63

Random-Forest top-20 word-feature weights:

features	**weight**
ruby	0.029073
voip	0.028905
test	0.024606
description	0.014564
1	0.013172
2015	0.012532
28	0.012352
router	0.008913
internet	0.007991
critical	0.007752
express	0.007324
interface	0.007135
ticketing	0.00686
subcomponent	0.006558
att	0.006523
fault	0.006451
els	0.006344
component	0.006272
working	0.006142
net	0.006102

Predicting Root_cause:

Model Name	Accuracy Score
Naive Bayes with tfidf_vector_model	0.66
RandomForest with tfidf_vector_model	0.67

Random-Forest top-20 word-feature weights:

features	weight
test	0.030168
voip	0.02959
ruby	0.026691
description	0.014987
2015	0.012663
1	0.012419
28	0.010485
internet	0.008932
router	0.008438
critical	0.007844
express	0.007765
traps	0.007559
fault	0.007322
ticketing	0.007114
interface	0.007028
att	0.007022
net	0.007017
subcomponent	0.006972
component	0.006902
els	0.006309

6 Areas of Application

- Predicting Resolution_set_name for quick fix. Can further identify steps for self help to advise end-user in chat-bots.
- Network Map: based on text in problem abstract we can predict which “Org” or department can fix the ticket quickly. This helps in automated routing reducing overall turnaround times.
- Predict estimated time to fix an issue and close the ticket.
- Predict if a dispatch will be required for fixing ticket-issue, this will further help in forecasting work-force requirement for field-technician teams.
- Predict Root-Cause for the given problem abstract.