ESCALATIONS – EXPERT SUPPORT CALCULATOR

THOMAS EAPEN

2018

Contents

List of Figures

1 REPORTING & ANALYTICS

1.1 ESCALATION – EXPERT SUPPORT CALCULATOR

To generate a count of all transfers that were done by an Agent, I had to sort all the records for each Interaction ID by time-stamp. Generating a list of Agents for that Interaction ID and observing the order of the interested Agent in the list indicates the position of the Agent in the interaction and indicates if this was the Agent that initiated the transfer or not. With this information along with the fact that we know if the Escalations/Expert Support Queue was involved in the interaction, we can determine if the interested Agent was in fact the Agent that made the call to the respective queues.

1.2 CORE ALGORITHM

```python
for index, agent_row in df_summary_subset.iterrows():
    #set_trace()
    df_agent = df_cleaned.loc[df_cleaned['AGENT_USERNAME'] == agent_row['Agent']]
    df_agent = df_agent.drop_duplicates(subset='INTERACTION_ID')
    esc_count = expert_count = trf_count = 0
    for _, row in df_agent.iterrows():
        trf_happened = False
        df_iid_trf = df_cleaned[df_cleaned['INTERACTION_ID'] == row['INTERACTION_ID']].sort_values(by='IRF_START_DATE_TIME')
        df_iid_trf = df_iid_trf.dropna(subset=['USER_EMAIL'])
        agent_list = df_iid_trf.USER_EMAIL.unique()
        if len(agent_list) > 1:
            #set_trace()
            if agent_list[-1] != agent_row['Agent']+"@tangerine.ca" and agent_list[-2] == agent_row['Agent']+"@tangerine.ca":
                trf_happened= True
                trf_count += 1
        #set_trace()
        df_iid_esc = df_cleaned.loc[(df_cleaned['INTERACTION_ID'] == row['INTERACTION_ID']) & (df_cleaned['QUEUE_QUEUE'].isin(esc_list))]
        if not df_iid_esc.empty:
            if trf_happened == True:
                set_trace()
                esc_count += 1
        df_iid_expert = df_cleaned.loc[(df_cleaned['INTERACTION_ID'] == row['INTERACTION_ID']) & (df_cleaned['QUEUE_QUEUE'].isin(expert_list))]
        if not df_iid_expert.empty:
            if trf_happened == True:
                expert_count += 1
    #set_trace()
    #df_summary.at[index, 'Escalation Count'] = esc_count
    #df_summary.at[index, 'Expert Count'] = expert_count
    df_summary_subset.at[index, 'Escalation Count'] = esc_count
    df_summary_subset.at[index, 'Expert Count'] = expert_count
    df_summary_subset.at[index, 'Transfer Count'] = trf_count
```

Listing 1: Code Listing → Transfer Call Count

1.3 CORE ALGORITHM – V3

In the updated design, we generate a list of participants by only removing consecutive duplicate Agents rather than all duplicate Agents. This removal of only consecutive duplicates makes sure that the call flow is maintained with the right order which can be inspected to account for cases where multiple transfers by the Agent to different CSLs or multiple transfers to the same CSL are accounted for. Once the list of CSL Agents are obtained from the call flow, we look at array indices comparisons to view if the call flowed from the interested Agent to the CSL.

```
for index, agent_row in df_summary_subset.iterrows():
    df_agent = df_cleaned.loc[df_cleaned['AGENT_USERNAME'] == agent_row['Agent']]
    df_agent = df_agent.drop_duplicates(subset='INTERACTION_ID')
    interested_email = agent_row['Agent']+'@tangerine.ca'
    esc_count = expert_count = trf_count = 0
    for _, row in df_agent.iterrows():
        df_iid_trf = df_cleaned[df_cleaned['INTERACTION_ID'] == row['INTERACTION_ID']].sort_values(by='IRF_START_DATE_TIME')
        df_iid_trf = df_iid_trf.dropna(subset=['USER_EMAIL'])
        agent_list = df_iid_trf.USER_EMAIL
        agent_list = agent_list.tolist()
        agent_list = [x[0] for x in groupby(agent_list)]
        if len(agent_list) > 1:
            agent_list_trf = list(OrderedDict.fromkeys(agent_list))
            if agent_list_trf[-1] != interested_email:
                trf_count += 1
        df_iid_esc = df_cleaned.loc[(df_cleaned['INTERACTION_ID'] == row['INTERACTION_ID']) & (df_cleaned['QUEUE_QUEUE'].isin(esc_list))]
        if not df_iid_esc.empty:
            df_iid_esc = df_iid_esc.dropna(subset=['USER_EMAIL'])
            esc_names = df_iid_esc['USER_EMAIL'].tolist()
            esc_names = list(OrderedDict.fromkeys(esc_names))
            for name in esc_names:
                indices = [i for i, x in enumerate(agent_list) if x == name]
                for i in indices:
                    if agent_list[i - 1] == interested_email:
                        iid_list.append(df_iid_esc['INTERACTION_ID'].iloc[0])
                        esc_count += 1
                        break

    #set_trace()
    df_summary.at[index, 'Escalation Count'] = esc_count
```

Listing 2: Code Listing → Core Algorithm – v3