Capstone project on NLP – Automatic Ticket Assignment

Great Lakes AIML January 2020

Team Members	Krupa
	Utkarsh Khamgaonkar
	Sai Tejaswi
	Manish Raj

Introduction

In this capstone project, the goal is to build a classifier that can classify the tickets by analysing text. This report summarizes all the details of the project and analysis performed on the simulated dataset. The following sections will give more insight into the use case of building an Automated Ticket Assignment System and the corresponding business value of the solution.

Problem Statement

One of the key activities of any IT function is to "Keep the lights on" to ensure there is no impact to the Business operations. IT leverages the Incident Management process to achieve the above Objective. An incident is something that is an unplanned interruption to an IT service or reduction in the quality of an IT service that affects the Users and the Business. The main goal of the Incident Management process is to provide a quick fix / workarounds or solutions that resolves the interruption and restores the service to its full capacity to ensure no business impact. In most of the organizations, incidents are created by various Business and IT Users, End Users/ Vendors if they have access to ticketing systems, and from the integrated monitoring systems and tools. Assigning the incidents to the appropriate person or unit in the support team has critical importance to provide improved user satisfaction while ensuring better allocation of support resources. The assignment of incidents to appropriate IT groups is still a manual process in many of the IT organizations. Manual assignment of incidents is time consuming and requires human efforts. There may be mistakes due to human errors and resource consumption is carried out ineffectively because of the misaddressing. On the other hand, manual assignment increases the response and resolution times which result in user satisfaction deterioration / poor customer service

Business Value

In the support process, incoming incidents are analysed and assessed by the organization's support teams to fulfil the request. In many organizations, better allocation and effective usage of the valuable support resources will directly result in substantial cost savings. Currently the incidents are created by various stakeholders (Business Users, IT Users and Monitoring Tools) within the IT Service Management Tool and are assigned to Service Desk teams (L1 / L2 teams). This team will review the incidents for right ticket categorization,

priorities and then carry out initial diagnosis to see if they can resolve. Around ~54% of the incidents are resolved by L1 / L2 teams. In the case L1 / L2 is unable to resolve, they will then escalate / assign the tickets to Functional teams from Applications and Infrastructure (L3 teams). Some portions of incidents are directly assigned to L3 teams by either Monitoring tools or Callers / Requestors. L3 teams will carry out detailed diagnosis and resolve the incidents. Around ~56% of incidents are resolved by Functional / L3 teams. In the case if vendor support is needed, they will reach out for their support towards incident closure.
L1 / L2 needs to spend time reviewing Standard Operating Procedures (SOPs) before assigning to Functional teams (Minimum ~25-30% of incidents needs to be reviewed for SOPs before ticket assignment). 15 min is being spent for SOP review for each incident. Minimum of ~1 FTE effort needed only for incident assignment to L3 teams.

During the process of incident assignments by L1 / L2 teams to functional groups, there were multiple instances of incidents getting assigned to wrong functional groups. Around ~25% of Incidents are wrongly assigned to functional teams. Additional effort needed for Functional teams to re-assign to right functional groups. During this process, some of the incidents are in queue and not addressed timely resulting in poor customer service.
Guided by powerful AI techniques that can classify incidents to right functional groups can help organizations to reduce the resolving time of the issue and can focus on more productive tasks.

Objective

Build a Multi-Class classifier that can classify the tickets by analysing text. Guided by powerful AI techniques that can classify incidents to right functional groups can help organizations to reduce the resolving time of the issue and can focus on more productive tasks. We will learn to use different classification models and learn to set the optimizers, loss functions, epochs, learning rate, batch size, checkpointing, early stopping etc.

Steps for Data Science Model Building:

Importing the necessary Libraries:

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import pickle
from time import time
import re
# ftfy (Fixes Text For You) to check the given text for weird characters
!pip install ftfy
from ftfy import fix_text, badness
# goslate to translate non-English text into English
!pip install goslate
from goslate import Goslate
# spacy to lemmatize the english description
!pip3 install spacy
!python3 -m spacy download en_core_web_sm
import spacy
# nltk for tokenization
import nltk
nltk.download('punkt')
from wordcloud import WordCloud, STOPWORDS
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer
from sklearn.naive_bayes import MultinomialNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import LinearSVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier
import xgboost as xgb
```

Reading and Understanding Dataset: Given a dataset consisting of various incidents reported and are classified to various L1/L2 groups. The dataset consists of the following columns

1. Short Description – Describes the incident in brief
2. Description – Detail description of the incident
3. Caller – Details/Userid of the caller
4. Assignment Group – L1/L2 group to which incident was assigned.

Here we do not need the Caller group as it does not have any impact on the modelling.

Short Description, Description would be the X Variables and Assignment Group is the Y/Target Variable. Column names are renamed to eliminate the spaces.

```python
data = pd.read_excel('input_data.xlsx')
data.head()
```

	Short description	Description	Caller	Assignment group
0	login issue	-verified user details.(employee# & manager na...	spxjnwir pjlcoqds	GRP_0
1	outlook	\r\n\r\nreceived from: hmjdrvpb.komuaywn@gmail...	hmjdrvpb komuaywn	GRP_0
2	cant log in to vpn	\r\n\r\nreceived from: eylqgodm.ybqkwiam@gmail...	eylqgodm ybqkwiam	GRP_0
3	unable to access hr_tool page	unable to access hr_tool page	xbkucsvz gcpydteq	GRP_0
4	skype error	skype error	owlgqjme qhcozdfx	GRP_0

```python
data.rename(columns = {'Short description': 'short_description', 'Description': 'description', 'Caller': 'caller', 'Assignment gr
data.head()
```

	short_description	description	caller	assignment_group
0	login issue	-verified user details.(employee# & manager na...	spxjnwir pjlcoqds	GRP_0
1	outlook	\r\n\r\nreceived from: hmjdrvpb.komuaywn@gmail...	hmjdrvpb komuaywn	GRP_0
2	cant log in to vpn	\r\n\r\nreceived from: eylqgodm.ybqkwiam@gmail...	eylqgodm ybqkwiam	GRP_0
3	unable to access hr_tool page	unable to access hr_tool page	xbkucsvz gcpydteq	GRP_0
4	skype error	skype error	owlgqjme qhcozdfx	GRP_0

EDA and Data Pre-processing:

Data Cleansing:

As part of EDA, we shall perform below checks and perform necessary actions as needed:

1) Check for Duplicates: We have found that duplicates exist and have dropped them.
2) Check for Null Values: Since there are 8 null records that have missing data, the null values were replaced with empty strings to avoid any data loss.
3) Dropping the Caller column.

In [8]:
```
dup_rows = data[data.duplicated()]
dup_rows
```

Out[8]:

	short_description	description	caller	assignment_group
51	call for ecwtrjnq jpecxuty	call for ecwtrjnq jpecxuty	olckhmvx pcqobjnd	GRP_0
229	call for ecwtrjnq jpecxuty	call for ecwtrjnq jpecxuty	olckhmvx pcqobjnd	GRP_0
493	ticket update on inplant_872730	ticket update on inplant_872730	fumkcsji sarmtlhy	GRP_0
512	blank call //gso	blank call //gso	rbozivdq gmlhrtvp	GRP_0
667	job bkbackup_tool_powder_prod_full failed in j...	received from: monitoring_tool@company.com\r\n...	bpctwhsn kzqsbmtp	GRP_8
...	...	...	...	...
7836	probleme mit erpgui \tmqfjard qzhgdoua	probleme mit erpgui \tmqfjard qzhgdoua	tmqfjard qzhgdoua	GRP_24
8051	issue on pricing in distributor_tool	we have agreed price with many of the distribu...	hbmwlprq ilfvyodx	GRP_21
8093	reset passwords for prgthyuulla ramdntythanjes...	the	boirqctx bkijgqry	GRP_17
8347	blank call // loud noise	blank call // loud noise	rbozivdq gmlhrtvp	GRP_0
8405	unable to launch outlook	unable to launch outlook	wjtzrmqc ikqpbflg	GRP_0

83 rows × 4 columns

In [9]:
```
data.drop_duplicates(keep='first',inplace=True,ignore_index=True)
print(data.info())
data[data.duplicated()]
```

```
data.fillna(str(), inplace=True)
data[pd.isnull(data).any(axis=1)]
```

short_description	description	caller	assignment_group

```
data.isnull().sum()
```

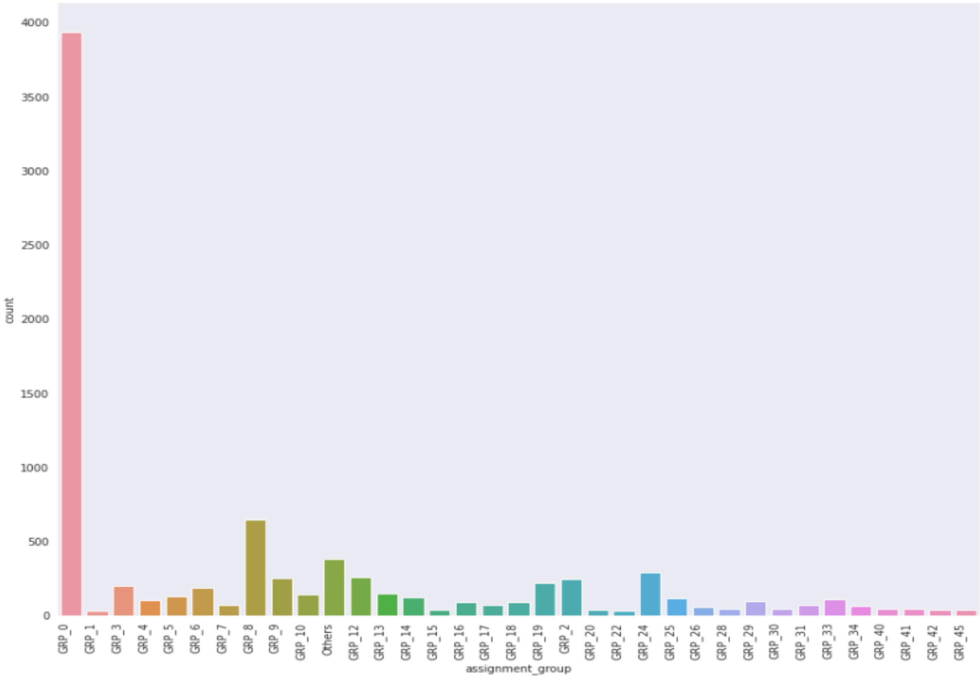
```
short_description    0
description          0
caller               0
assignment_group     0
dtype: int64
```

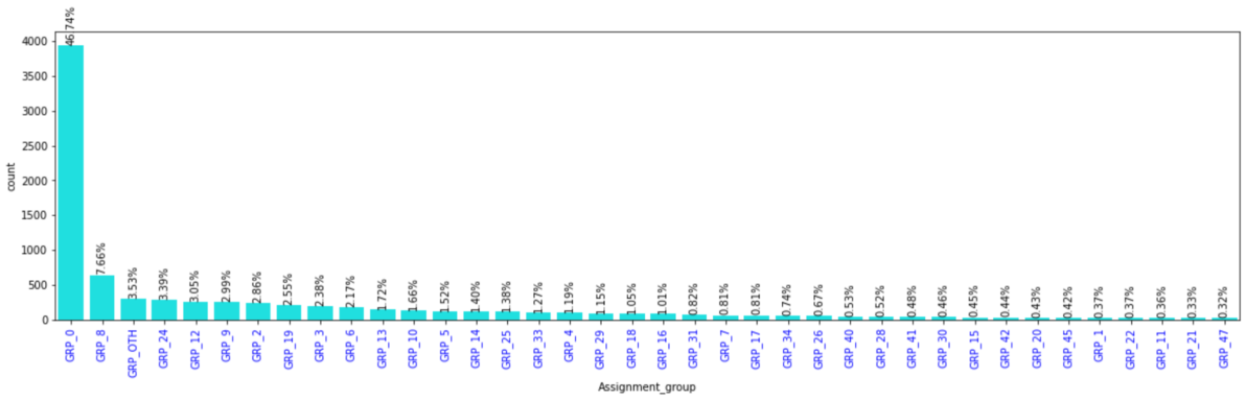
```
del data['caller']
data.head()
```

	short_description	description	assignment_group
0	login issue	-verified user details.(employee# & manager na...	GRP_0
1	outlook	\r\n\r\nreceived from: hmjdrvpb.komuaywn@gmail...	GRP_0
2	cant log in to vpn	\r\n\r\nreceived from: eylqgodm.ybqkwiam@gmail...	GRP_0
3	unable to access hr_tool page	unable to access hr_tool page	GRP_0
4	skype error	skype error	GRP_0

Data Visualization:

Understanding the distribution of incidents over the Groups: As we observed most of the incidents are across few groups and few groups have as less as 1 incident. Since this is an imbalanced data set, we considered only the top 10 groups for which the incidents are contributing to maximum percentage.






```
dfTicketPrep['Assignment_group'].value_counts().nlargest(10)
```

```
GRP_0      3934
GRP_8       645
GRP_OTH     297
GRP_24      285
GRP_12      257
GRP_9       252
GRP_2       241
GRP_19      215
GRP_3       200
GRP_6       183
Name: Assignment_group, dtype: int64
```

Concatenating Short Description and Description: As both are contributing to the processing of text, we concatenated the short description and description.

```
#Concatenate Short Description and Description columns as they contribute to same descriptions necessary for classification
dfTicketPrep['Full_Description'] = dfTicketPrep['Short description'] + ' ' +dfTicketPrep['Description']
```

Word Cloud Distributions:

Text Pre-processing to make it ready for model processing:

Data Cleansing:

1.We have identified few junk/weird characters in the data and necessary actions are taken to treat the data.

2.We have cleansed data for removing email patterns, number patterns, stop words, punctuations, additional space and lines and other patterns which are not required for text classification.

```
]: # Regex patterns
EMAIL_PATTERN = r"([\w.+-]+@[a-z\d-]+\.[a-z\d.-]+)"
PUNCTUATION_PATTERN = r"[,|@|\|?|\\|$&*|%|\r|\n|.:|\s+|/|//|\\|/|\||-|<|>|;|(|)|=|+|#|-|\"|[-\]]|{|}]" # also remove - followed
NUMBER_PATTERN = r"(\d+(?:\.\d+)?)"

# Function to process the ticket text
def cleanText(text):
    # Make the text unicase (lower)
    text = str(text).lower()
    # Remove email adresses
    text = re.sub(EMAIL_PATTERN, '', text, flags=re.IGNORECASE) # check for duplicates after this
    # Remove all numbers
    text = re.sub(NUMBER_PATTERN, '', text)
    # Replace all punctuations with blank space
    text = re.sub(PUNCTUATION_PATTERN, " ", text, flags=re.MULTILINE)
    # Replace multiple spaces from prev step to single
    text = re.sub(r' {2,}', " ", text, flags=re.MULTILINE)
    text = text.replace('`',"'")
    return text.strip()
```

Lemmatization with Spacy:

We have made use of Spacy - en_core_web_sm model to lemmatize the date

```
In [31]: # Initialize spacy 'en' medium model, keeping only tagger component needed for lemmatization
nlp = spacy.load('en_core_web_sm', disable=['parser', 'ner'])

# Define a function to lemmatize the descriptions
def lemmatizer(sentence):
    # Parse the sentence using the loaded 'en' model object `nlp`
    doc = nlp(sentence)
    return " ".join([token.lemma_ for token in doc if token.lemma_ !='-PRON-'])
```

```
In [32]: # Take an example of row# 43 Description and lemmatize it
print('\033[1mOriginal text:\033[0m')
print(data['description'][100])
print('_'*100)
print('\033[1mLemmatized text:\033[0m')
print(lemmatizer(data['description'][100]))
```

```
Original text:
unable to access mails
____________________________________________________________________________________________________
Lemmatized text:
unable to access mail
```

Adding Additional Features: Additional features for the data to add the words counts and length of the incident text is added to the dataset.

```
: # Create new features of length and word count for both of the description columns
data.insert(1, 'sd_len', data['short_description'].astype(str).apply(len))
data.insert(2, 'sd_word_count', data['short_description'].apply(lambda x: len(str(x).split())))
data.insert(4, 'desc_len', data['description'].astype(str).apply(len))
data.insert(5, 'desc_word_count', data['description'].apply(lambda x: len(str(x).split())))
data.head()
```

Converting the Assignment Groups to Category Code: As we need to pass the target values in category codes for the model to process during training, performing the steps

[43]:
```
# Create two new columns - word_collection and target
data['word_collection'] = data['short_description'].str.strip() + ' ' + data['description'].str.strip()
data['target'] = data['assignment_group'].astype('category').cat.codes
data.head()
```

[43]:

	short_description	sd_len	sd_word_count	description	desc_len	desc_word_count	Language	assignment_group	word_collection	target
0	login issue	11	2	-verified user detail employee manager name -c...	183	32	en	GRP_0	login issue - verified user detail employee man...	0
1	outlook	7	1	receive from hello team meeting skype meeting ...	135	23	en	GRP_0	outlook receive from hello team meeting skype ...	0
2	can not log in to vpn	21	6	receive from hi i can not log on to vpn best	44	11	en	GRP_0	can not log in to vpn receive from hi i can no...	0
3	unable to access hr_tool page	29	5	unable to access hr_tool page	29	5	en	GRP_0	unable to access hr_tool page unable to access...	0
4	skype error	11	2	skype error	11	2	en	GRP_0	skype error skype error	0

Creating the Train and Test Splits: We are splitting the data in 80:20 ratio

]:
```
# Create train and test datasets with 80:20 ratio
X_train, X_test, y_train, y_test = train_test_split(data.word_collection,
                                                    data.target,
                                                    test_size=0.20,
                                                    random_state=42)
print('Shape of the training set:', X_train.shape, X_test.shape)
print('Shape of the test set:', y_train.shape, y_test.shape)
```

```
Shape of the training set: (5504,) (1377,)
Shape of the test set: (5504,) (1377,)
```

Model Building: Since this is a multi-class classification , we have created models with following algorithms and the corresponding accuracy achieved as below, also we have created a LSTM neural network using Keras and TensorFlow.

Below are the layers and the corresponding activation functions used:

]:
```
# Sequential Modeling
from keras.models import Sequential, Model
from keras.layers import Input, Dropout, Flatten, Dense, Embedding, LSTM, GRU,Activation
from keras.layers import BatchNormalization, TimeDistributed, Conv1D, MaxPooling1D
from keras.preprocessing.text import Tokenizer, text_to_word_sequence
from keras.preprocessing.sequence import pad_sequences
from keras.callbacks import EarlyStopping, ModelCheckpoint
from keras import optimizers
```

]:
```
tokenizer = Tokenizer()
tokenizer.fit_on_texts(dfTicketsClean.Description)
word_counts = tokenizer.word_counts
num_words = len(word_counts)
```

In [ ]:
```
from sklearn.preprocessing import LabelBinarizer
num_labels = 38
vocab_size = 75000
batch_size = 64
# define Tokenizer with Vocab Size
tokenizer = Tokenizer(num_words=vocab_size)
tokenizer.fit_on_texts(X_train)

x_train = tokenizer.texts_to_matrix(X_train, mode='tfidf')
x_test = tokenizer.texts_to_matrix(X_test, mode='tfidf')

encoder = LabelBinarizer()
encoder.fit(y_train)
y_train = encoder.transform(y_train)
y_test = encoder.transform(y_test)

model = Sequential()
model.add(Dense(50, input_shape=(vocab_size,)))
model.add(Activation('relu'))
model.add(Dropout(0.3))
model.add(Dense(30))
model.add(Activation('relu'))
model.add(Dropout(0.3))
model.add(Dense(num_labels))
model.add(Activation('softmax'))
model.summary()
```

```
Model: "sequential_8"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_28 (Dense)             (None, 50)                3750050
_________________________________________________________________
activation_18 (Activation)   (None, 50)                0
_________________________________________________________________
dropout_20 (Dropout)         (None, 50)                0
_________________________________________________________________
dense_29 (Dense)             (None, 30)                1530
_________________________________________________________________
activation_19 (Activation)   (None, 30)                0
_________________________________________________________________
dropout_21 (Dropout)         (None, 30)                0
_________________________________________________________________
dense_30 (Dense)             (None, 38)                1178
_________________________________________________________________
activation_20 (Activation)   (None, 38)                0
=================================================================
Total params: 3,752,758
Trainable params: 3,752,758
Non-trainable params: 0
```

Validations: Currently we are doing validations on the different models used. We have below table for accuracies attained.

```
modelLogs.sort_values(by='TrainScore')
```

	Classifier	TrainScore	TestScore
0	SVC	43.193673	44.248234
0	AdaboostClassifier-5kEstimators	48.679118	49.545913
0	AdaboostClassifier	48.695945	49.495459
0	MultinomialNB	55.359246	55.903128
0	KNeighborsClassifier	57.294296	58.274470
0	LogisticRegression-lbfgs	65.017668	62.260343
0	LogisticRegression-sag	65.017668	62.260343
0	LogisticRegression	65.017668	62.260343
0	DeepNN-4HiddenLayer	80.077404	57.769930
0	SGDClassifier	85.192664	66.044400
0	DeepNN-1HiddenLayer	85.630155	64.581233
0	LinearSVC	87.767121	65.691221
0	BaggingClassifier	89.651691	61.453078
0	GradientBosstingClassifier	89.718997	60.998991
0	DeepNN-2HiddenLayer	92.242974	63.017154
0	RandomForestClassifier	92.781424	64.026236
0	DecisionTreeClassifier	92.781424	59.989909