2020/2021

LINKÖPING UNIVERSITY

Chatbot for a service company

732A92 Text Mining Final Project

Student
K.M. Mudith Chathuranga Silva (kogsi273)

Examiner
Prof. Marco Kuhlmann

Table of Contents

Abstract 3
1. Introduction 4
1.1 Motivation 4
1.2 Objective 4
2. Theory 5
3. Data 9
4. Method 13
4.1 Brainstorming 13
4.2 Customer Chat Analysis and Create Gold Label Class 14
4.3 Check ML models and parameter optimization 14
4.4 Identify Best Model 17
4.5 Using Best model and saving 18
4.6 Feature Importance 19
4.7 Python Helper Class 20
5. Results 30
5.1 Chatbot Flow Diagram 31
5.2 Chatbot Analytics 32
5.2.1 Importance of Chatbot Analytics 32
5.2.2 Chatbot Analytics Metrices 32
5.3 Chatbot Evaluation Summary 35
6. Discussion 36
6.1 Chatbot Scalability 40
6.2 Chatbot Limitations and Known Bugs 40
7. Conclusion 44
Bibliography 45

Abstract

Since the development of social media networks, more and more businesses have been using chat services to attract more customers by providing real-time information for their potential or existing customers. Medium or large scale businesses have a dedicated customer service person to cater to their online customers' needs. But for small-scale businesses are unable to compete with their rival companies since they have a limited amount of employees and unable to pay for an extra person for online customer handling. And also there are some special service-related businesses in which the owner needs to directly interact with customers. So they can't use an employee to handle their customer because of the business model or they can't expose customer information with another party.

Over the last few years, we could see that Text Mining and Natural language processing has been spreading to varies kind of industries. So we could use the Text mining knowledge to address the above situations and create an automated chatbot. We could extract existing customer's text messages from a specific industry and create a simple chatbot that could communicate with customers and solve their problems in real-time.

1. Introduction

1.1 Motivation

I have a friend in Sri-Lanka who is running a small-scale salon. And also he has unique customers such as Actresses, Businessman's … etc. Most of his customers make/cancel and remove appointments through WhatsApp and Facebook Messenger (Private profile and Business profile). By the way, he won't use the phone when he is providing a service to a client. Therefore sometimes he can't reach the phone to reply to his exiting customer or a new customer who is asking general questions or even ask appointments over the chat. Some of the important clients' required a reply instantly and new customers are also demanding a reply from the salon as soon as possible. Since I'm aware of this problem, I thought that Text Mining may help to resolve this problem somehow.

1.2 Objective

It is always recommended to have a look at the dataset before implementing the solution to the problem. So sample chat records were requested for initial analysis. Then several question groups were identified. Automated replies could be generated from the salon side by checking the client question type. But the tricky part is to extract customer information for the appointment placement. Then generate an appropriate text according to the customer's reply. The chatbot would be an ideal solution for this kind of scenario. A chatbot could be trained by using previous customer questions and provide a general reply. And also python libraries such as Spacy could be used to identify customer replies and store required data.

2. Theory

There are some frequently used terminologies in Text mining and it is required to get some idea about those terms and the theory behind them. Generally ‘Spacy’ and ‘sci-kit’ libraries used for the project and the below terms are corresponding with the library functionalities.

Tokenization

During processing, spaCy first tokenizes the text, i.e. segments it into words, punctuation and so on. This is done by applying rules specific to each language. For example, punctuation at the end of a sentence should be split off – whereas “U.K.” should remain one token. (Spacy - Tokenization, n.d.)

Stop Words

A “stop list” is a classic trick from the early days of information retrieval when search was largely about keyword presence and absence. It is still sometimes useful today to filter out common words from a bag-of-words model. To improve readability, STOP_WORDS are separated by spaces and newlines, and added as a multiline string. (Spacy - Stop Words, n.d.)

Count Vectorizer

Number of words appear in the given document and represent the count in a vector form. This approach may be ignore rare words when creating the model. To overcome this problem inverse document frequency method introduced. (scikit - CountVectorizer, n.d.)

Inverse Document Frequency

Overall weighted document of a word. This helps to detect most frequent words and penalize them. TfidfVectorizer weights the number of word count in the document by calculating the frequency of the word. In scikit-learn, the tf–idf weight is computed as:- (Wikipedia - tf-idf, n.d.)

$$tf - idf(t,d) = tf(t,d).\left(\log\frac{1+N}{1+df(t)}+1\right)$$

- N _denotes the number of documents in the collection.

Text Classification pipeline

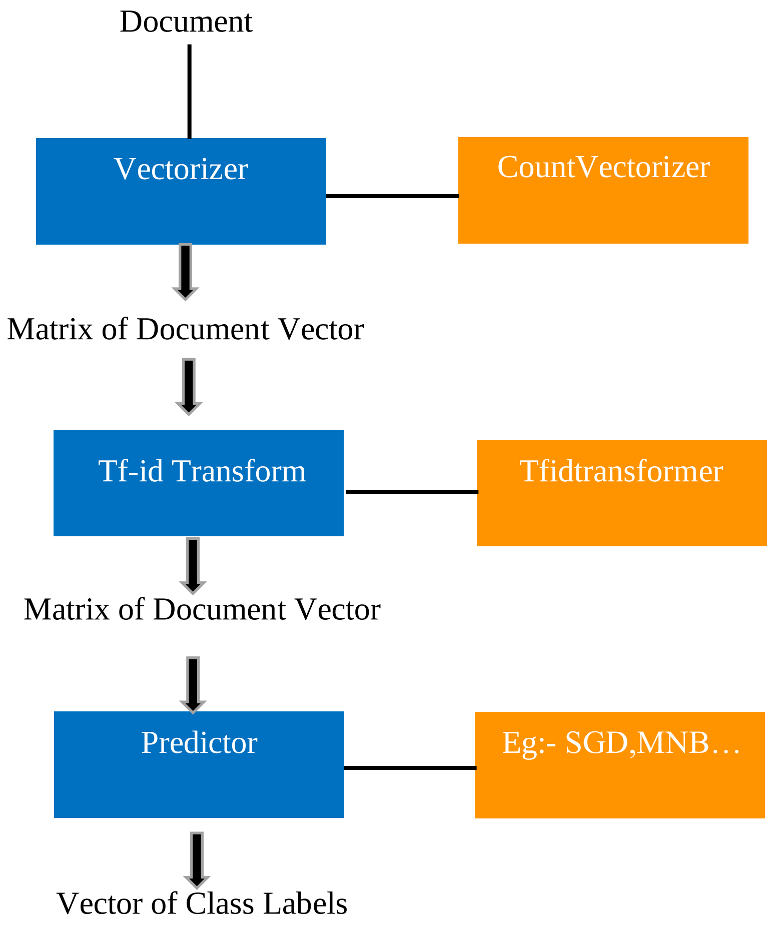


Figure 1.1

Accuracy, Precision, Recall and F1-Measure

The accuracy of a classifier is the proportion of documents for which the classifier predicts the gold-standard class.

$$\boldsymbol{Accuracy = \frac{Number\ of\ correctly\ classified\ documents}{Number\ of\ all\ documents}}$$

Precision is the proportion of correctly classified documents among all documents for which the system predicts specific class.

$$\boldsymbol{Precision = \frac{True\ poisitves}{True\ positives + false\ positive}}$$

Recall is the proportion of correctly classified documents among all documents with gold-standard class.

$$\boldsymbol{Recall = \frac{True\ positives}{True\ positives + False\ Negatives}}$$

The F1-measure is the harmonic mean of the two values. A good classifier should balance between precision and recall.

$$\boldsymbol{F1 = \frac{2 * precision * recall}{Precision + Recall}}$$

Naïve Bayes Classifier

C A set of possible classes

V A set of possible words; the model's vocabulary

$P(c)$ Probabilities that specify how likely it is for a document to belong to class c (one probability for each class)

$P(w|c)$ Probabilities that specify how likely it is for a document to contain the word w, given that the document belongs to class c (one probability for each class–word pair)

$\hat{C}$ Predicted class for the document.

W Count of the word W in the document

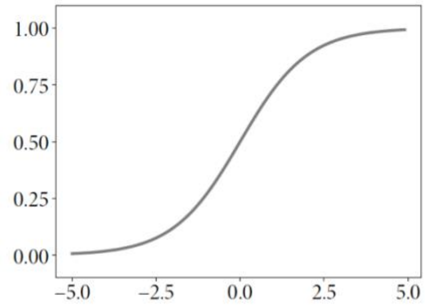
$$\hat{C} = \arg max_{c \in C} P(C). \prod P(W \mid C)^{\#(w)}$$

Logistic Regression Classifier

The logistic model extends the linear model by adding a pointwise logistic function f.

$$\hat{y} = f(z); \qquad z = xW + b$$

The term logistic regression refers to the procedure of learning the parameters of the logistic model. For Binary classifications :-



$$\hat{y} = \frac{1}{1 + \exp(-z)}$$

3. Data

All the chat messages were extracted from the messenger application(FB Messenger, WhatsApp, …) from the salon owner's mobile phone. Chat messages were copied to an excel sheet manually. Some of the messages were contained private information, such as names, phone numbers, general greetings….etc. By the way, it's essential to remove above information since we need to train the model to identify question groups in general and model do not need to be aware of different kind of names, phone numbers, …etc. Otherwise, the model may be overfitted by considering several names or user related entities. Here are the preprocessed steps on the chat dataset.

Any person name by introducing him/her self :- **username**

Eg:- Hi I'm Jason. Nice to meet you => Hi I'm **username**. Nice to meet you

Any greetings (eg:- Good Morning, Morning, Evening..) **usergreet**

Eg:- Good Morning! I need to know about general hair cut prices =>
usergreet! I need to know about general hair cut prices

Any welcome interjections (eg:- Hi, Hello …) **interjection**

Eg:- Hi! I need to make an appointment => **interjection**! I need to make an appointment

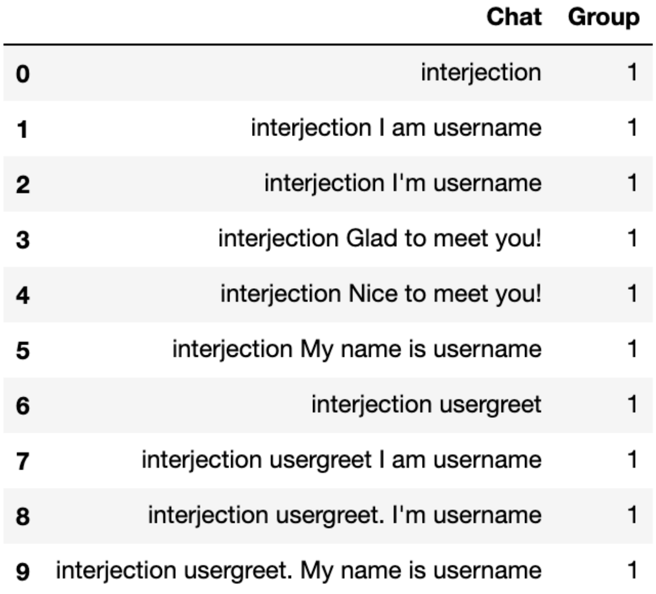
Any salon services (eg:- Hair cut, Coloring, ….) **gservice**

Eg:- I need to know about your salon hair coloring prices =>
I need to know about your salon **gservice** prices

Above words were used to train the model to remove redundant words and introduce the generic word to represent all the word groups. All the preprocessed chats were stored on ***UserChatData.xlsx.*** A brief summary of the chats and gold label classes could be generated from the data.

1 - General Greeting	Used Hi for all injection words (Hey, Hi,..)
2 - Greeting + Checkback	Replaced usergreetings -> Good Morning
3 - Ask Available Services	Eg:- All available services
4 - Ask General Services	Eg :- Hair cut, Colouring …
5 - Make Reservation	
6 - Change Reservation	
7 - Remove/Cancel Reservation	
8 - Close Chat / Close greeting	Eg :- Thank you, Okay Bye,….

Figure 3.1 – Gold Label Classes



	Chat	Group
0	interjection	1
1	interjection I am username	1
2	interjection I'm username	1
3	interjection Glad to meet you!	1
4	interjection Nice to meet you!	1
5	interjection My name is username	1
6	interjection usergreet	1
7	interjection usergreet I am username	1
8	interjection usergreet. I'm username	1
9	interjection usergreet. My name is username	1

Figure 3.2 – Data frame head(10)

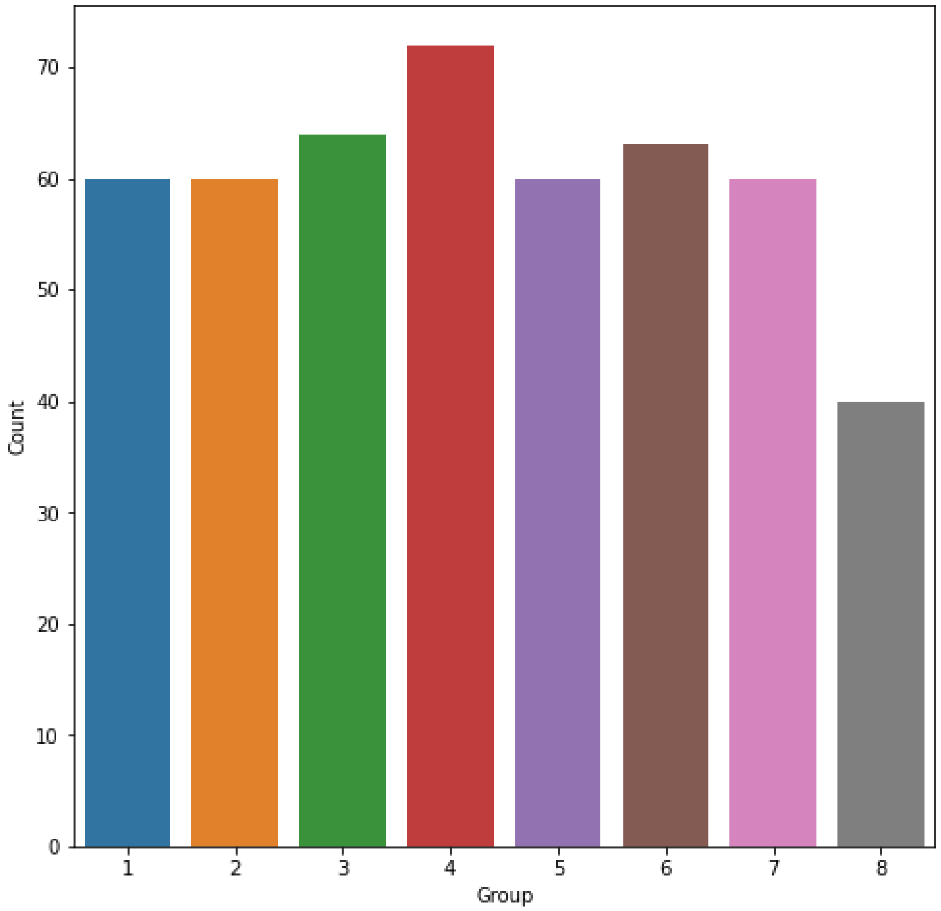
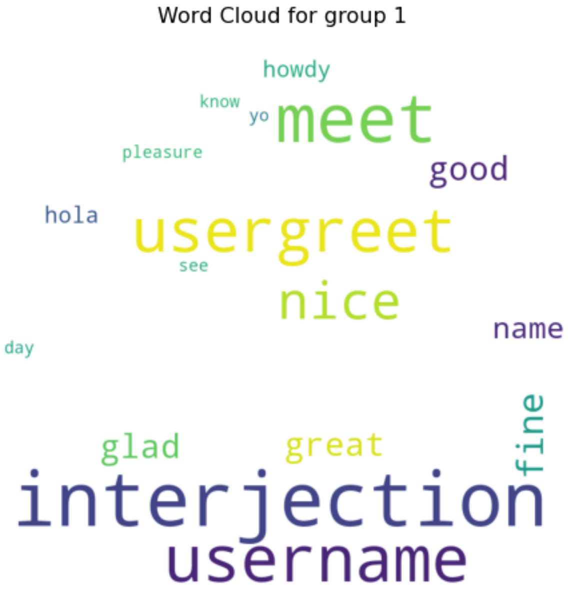
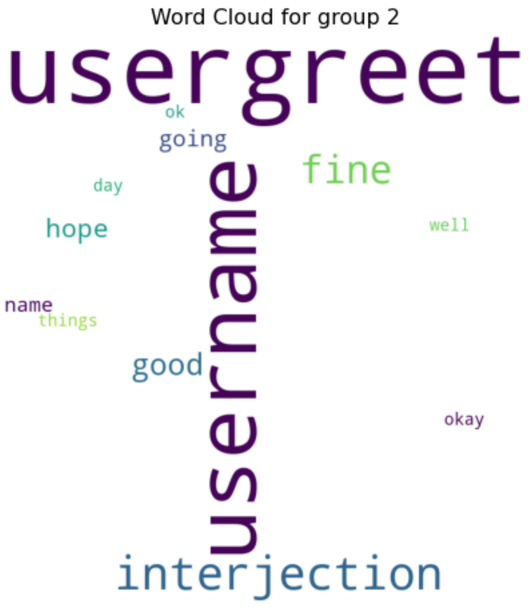
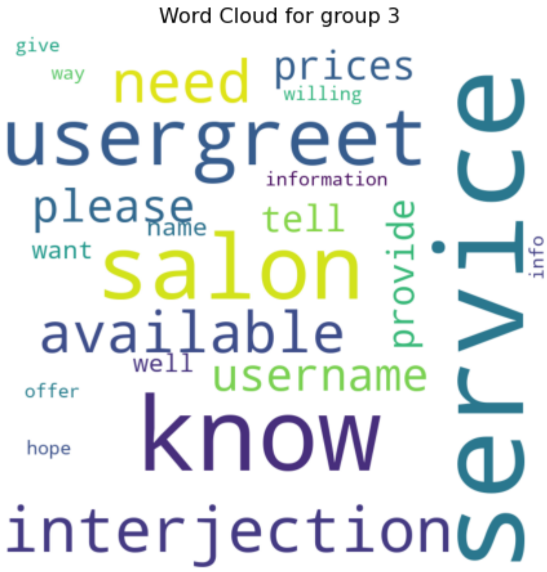


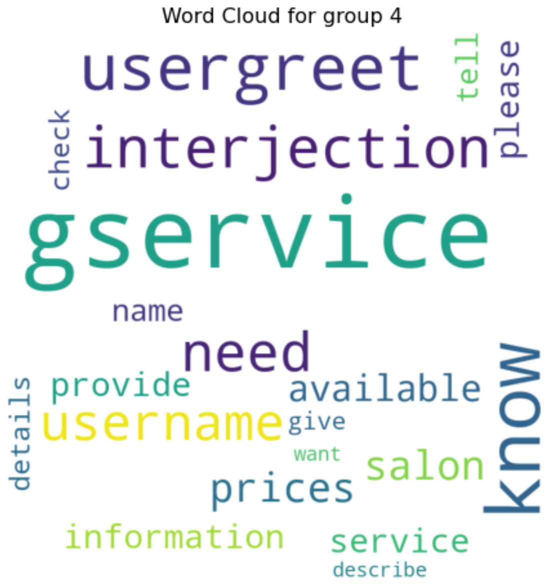
Figure 3.3 – Count Plot for Gold label classes

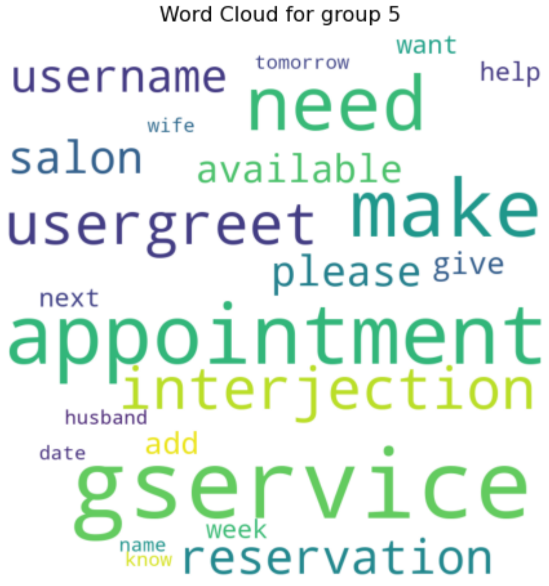
Word Cloud for Gold label classes.

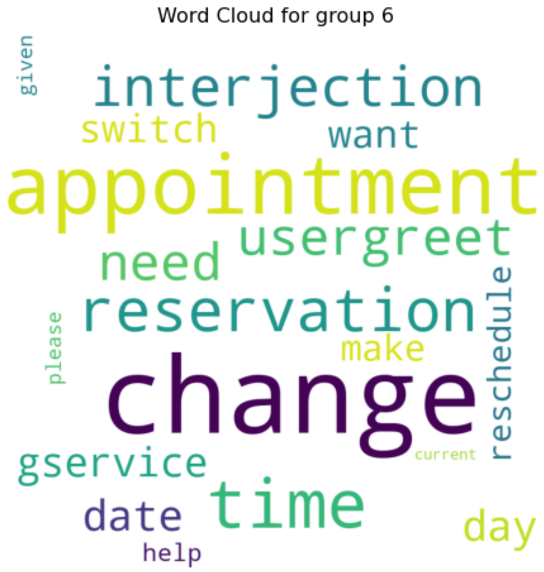


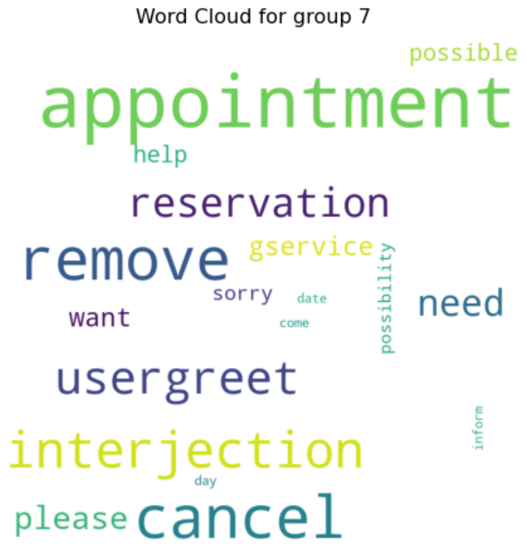


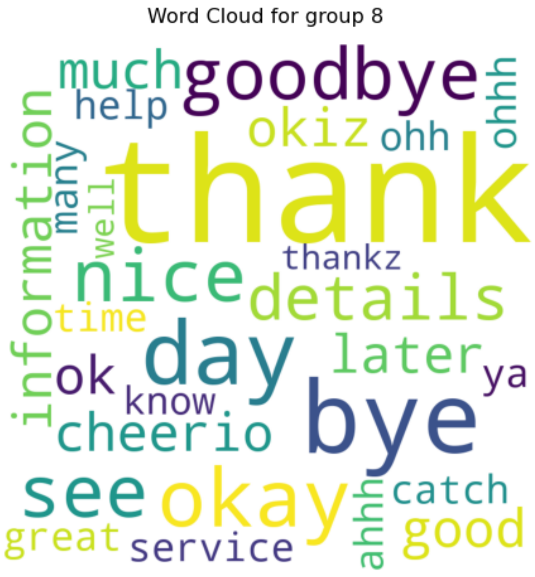

Word Cloud for group 3
service
know
salon
usergreet
interjection
available
need
prices
please
tell
username
provide
willing
information
name
want
well
give
way
offer
hope
info


Word Cloud for group 4
gservice
usergreet
interjection
know
need
username
available
prices
salon
provide
information
service
name
check
tell
please
details
give
want
describe


Word Cloud for group 5
appointment
gservice
need
make
usergreet
interjection
reservation
username
salon
available
please
give
help
want
tomorrow
wife
next
husband
add
date
week
name
know


Word Cloud for group 6
change
appointment
interjection
reservation
usergreet
time
need
gservice
date
day
switch
want
make
reschedule
please
given
current
help


Word Cloud for group 7
appointment
remove
reservation
cancel
interjection
usergreet
need
gservice
please
want
help
possible
sorry
possibility
date
come
day
inform


Word Cloud for group 8
thank
bye
goodbye
okay
day
nice
details
see
much
help
okiz
ohh
ohhh
information
many
well
thankz
time
later
ok
ya
know
cheerio
catch
good
great
service
ahhh

4. Method

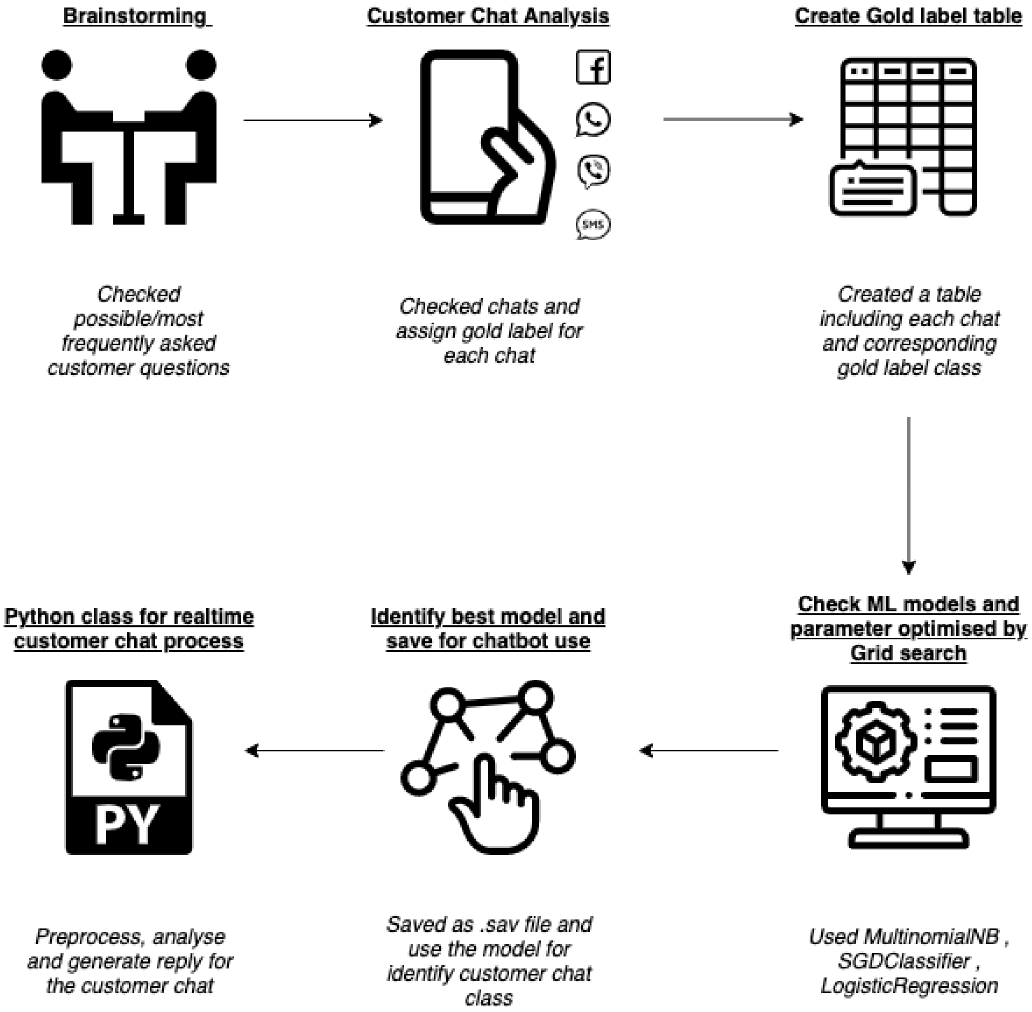

Brainstorming
Customer Chat Analysis
Create Gold label table
Checked possible/most frequently asked customer questions
Checked chats and assign gold label for each chat
Created a table including each chat and corresponding gold label class
Check ML models and parameter optimised by Grid search
Identify best model and save for chatbot use
Python class for realtime customer chat process
Used MultinomialNB , SGDClassifier , LogisticRegression
Saved as .sav file and use the model for identify customer chat class
Preprocess, analyse and generate reply for the customer chat
SMS
PY

Figure 4.1 – Method Phases

4.1 Brainstorming

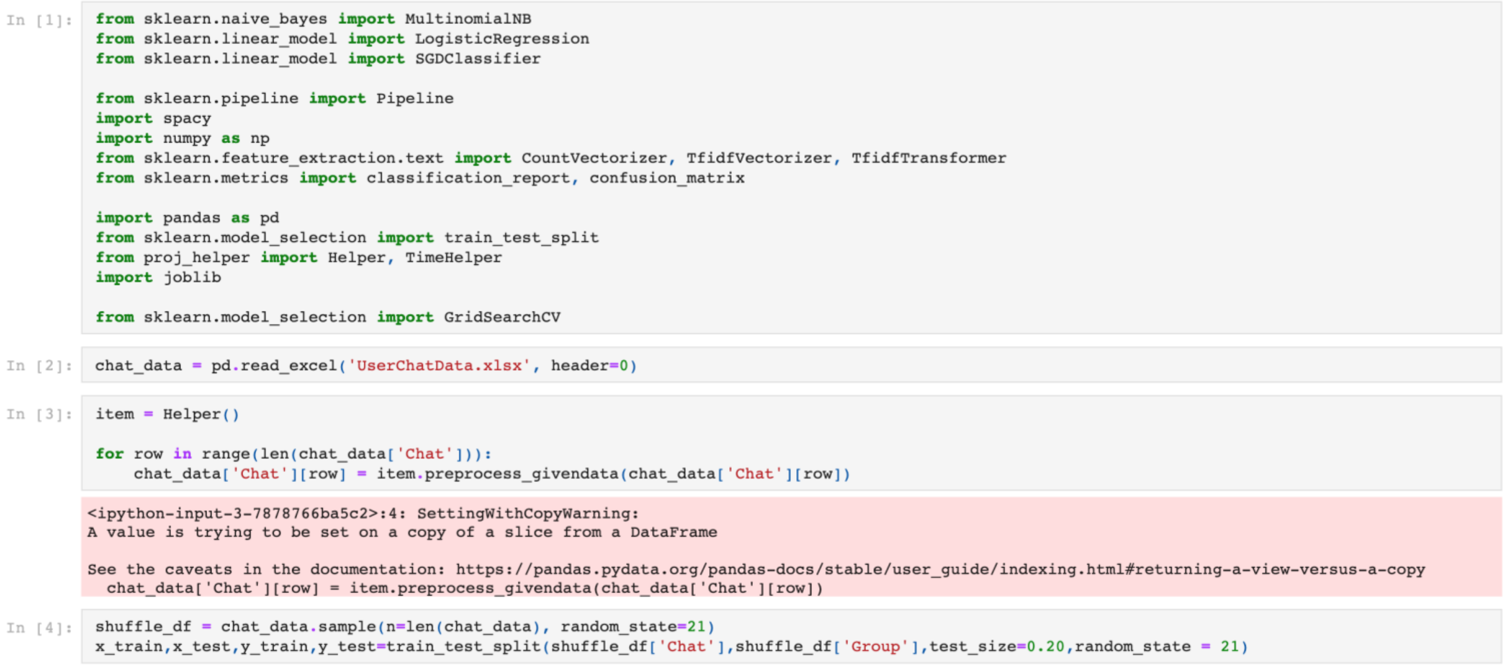
Frequently asked questions depend on the service. So for this particular scenario, the required questions are collected from the salon owner and experienced customer care personal. Then gold label classes were defined after cross-check with the previously asked questions by customers.

4.2 Customer Chat Analysis and Create Gold Label Class

Checked most of the chats from customers and assigned a gold label class for each chat message. Then evaluated data were stored on an excel sheet for model training and testing phases. Initial preprocessing steps were followed. Such as username, usergreet, interjection, and gservice. (Steps are explained in **‘3.Data’** Section) – File :- *Grid_Search.ipynb.*

4.3 Check ML models and parameter optimization

Data Preprocess



```
In [1]: from sklearn.naive_bayes import MultinomialNB
        from sklearn.linear_model import LogisticRegression
        from sklearn.linear_model import SGDClassifier

        from sklearn.pipeline import Pipeline
        import spacy
        import numpy as np
        from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer, TfidfTransformer
        from sklearn.metrics import classification_report, confusion_matrix

        import pandas as pd
        from sklearn.model_selection import train_test_split
        from proj_helper import Helper, TimeHelper
        import joblib

        from sklearn.model_selection import GridSearchCV

In [2]: chat_data = pd.read_excel('UserChatData.xlsx', header=0)

In [3]: item = Helper()

        for row in range(len(chat_data['Chat'])):
            chat_data['Chat'][row] = item.preprocess_givendata(chat_data['Chat'][row])

        <ipython-input-3-7878766ba5c2>:4: SettingWithCopyWarning:
        A value is trying to be set on a copy of a slice from a DataFrame

        See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
          chat_data['Chat'][row] = item.preprocess_givendata(chat_data['Chat'][row])

In [4]: shuffle_df = chat_data.sample(n=len(chat_data), random_state=21)
        x_train,x_test,y_train,y_test=train_test_split(shuffle_df['Chat'],shuffle_df['Group'],test_size=0.20,random_state = 21)
```

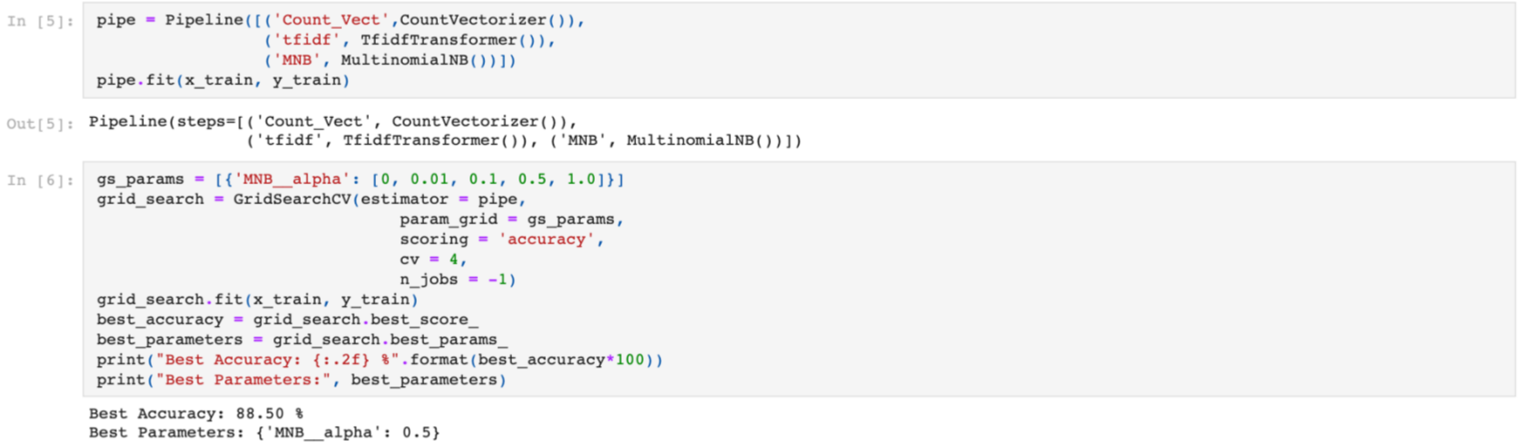
In [1] - Library Loading

ln [2] – Load chat table – Gold label table

ln [3] – Preprocess chats by using custom Helper class. (Remove names, greetings ,…..).

ln [4] – Shuffle chat table rows before train , test split

MultinomialNB



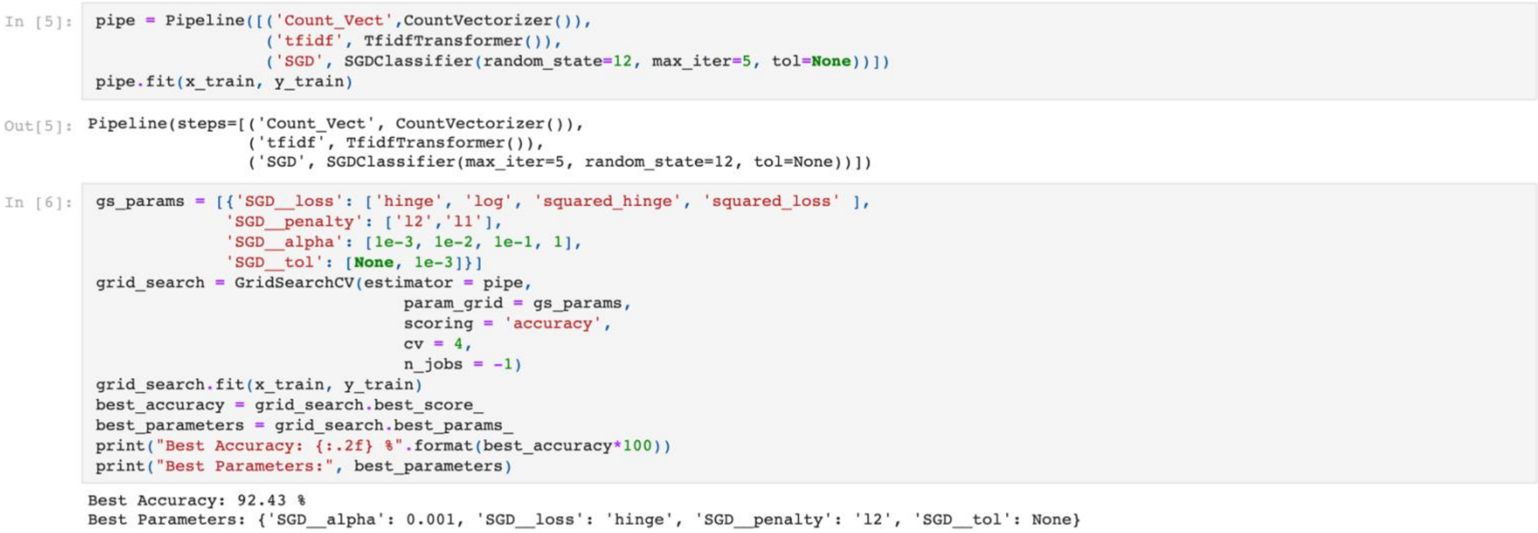
```
In [5]: pipe = Pipeline([('Count_Vect',CountVectorizer()),
                         ('tfidf', TfidfTransformer()),
                         ('MNB', MultinomialNB())])
        pipe.fit(x_train, y_train)

Out[5]: Pipeline(steps=[('Count_Vect', CountVectorizer()),
                        ('tfidf', TfidfTransformer()), ('MNB', MultinomialNB())])

In [6]: gs_params = [{'MNB__alpha': [0, 0.01, 0.1, 0.5, 1.0]}]
        grid_search = GridSearchCV(estimator = pipe,
                                   param_grid = gs_params,
                                   scoring = 'accuracy',
                                   cv = 4,
                                   n_jobs = -1)
        grid_search.fit(x_train, y_train)
        best_accuracy = grid_search.best_score_
        best_parameters = grid_search.best_params_
        print("Best Accuracy: {:.2f} %".format(best_accuracy*100))
        print("Best Parameters:", best_parameters)

        Best Accuracy: 88.50 %
        Best Parameters: {'MNB__alpha': 0.5}
```

ln [5] – Pipe for Count vectorizer (Edpresso - CountVectorizer in Python, n.d.), TfidTransformer (Wikipedia - tf-idf, n.d.) and MultinomialNB (Naive Bayes classifier, n.d.).

ln [6] – Grid Search for MNB (Hyperparameter optimization, n.d.)

Output :- Best Parameters for MultinomialNB – **alpha(0.5)**

SGDClassifier



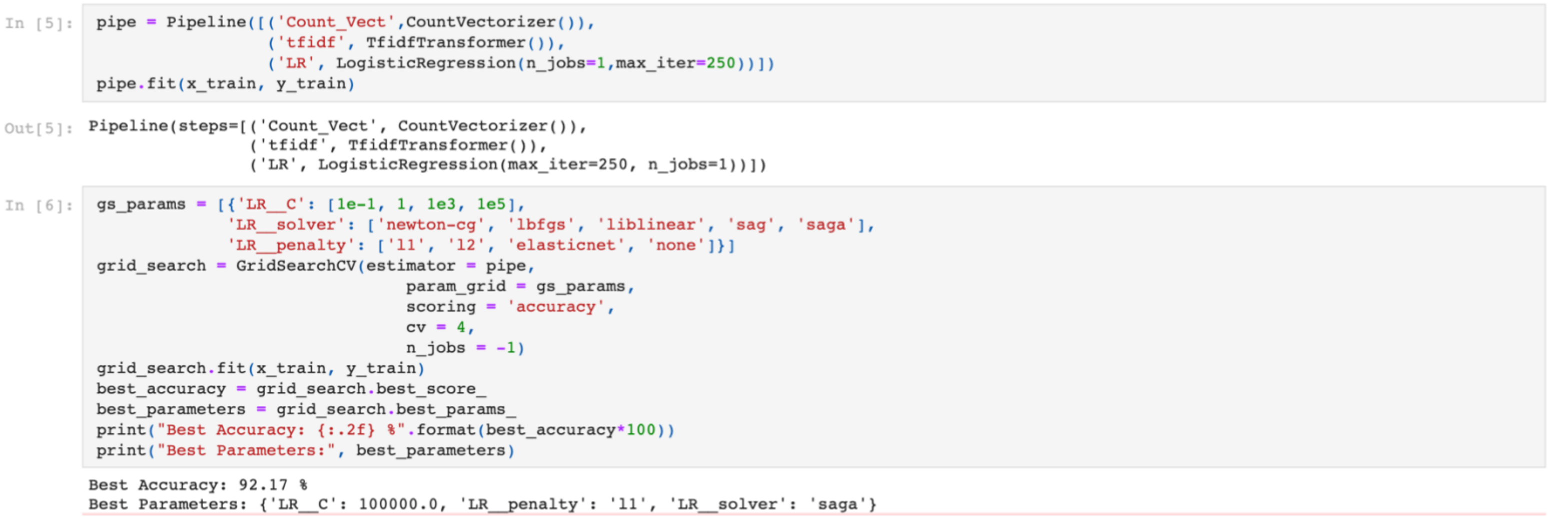
```
In [5]: pipe = Pipeline([('Count_Vect',CountVectorizer()),
                         ('tfidf', TfidfTransformer()),
                         ('SGD', SGDClassifier(random_state=12, max_iter=5, tol=None))])
        pipe.fit(x_train, y_train)

Out[5]: Pipeline(steps=[('Count_Vect', CountVectorizer()),
                        ('tfidf', TfidfTransformer()),
                        ('SGD', SGDClassifier(max_iter=5, random_state=12, tol=None))])

In [6]: gs_params = [{'SGD__loss': ['hinge', 'log', 'squared_hinge', 'squared_loss' ],
                      'SGD__penalty': ['l2','l1'],
                      'SGD__alpha': [1e-3, 1e-2, 1e-1, 1],
                      'SGD__tol': [None, 1e-3]}]
        grid_search = GridSearchCV(estimator = pipe,
                                   param_grid = gs_params,
                                   scoring = 'accuracy',
                                   cv = 4,
                                   n_jobs = -1)
        grid_search.fit(x_train, y_train)
        best_accuracy = grid_search.best_score_
        best_parameters = grid_search.best_params_
        print("Best Accuracy: {:.2f} %".format(best_accuracy*100))
        print("Best Parameters:", best_parameters)

        Best Accuracy: 92.43 %
        Best Parameters: {'SGD__alpha': 0.001, 'SGD__loss': 'hinge', 'SGD__penalty': 'l2', 'SGD__tol': None}
```

ln [5] – Pipe for Count vectorizer, TfidTransformer and SGDClassifier (Stochastic gradient descent, n.d.).

ln [6] – Grid Search for SGD

Output :- Best Parameters for MultinomialNB – **{'SGD__alpha': 0.001, 'SGD__loss': 'hinge', 'SGD__penalty': 'l2', 'SGD__tol': None}**

LogisticRegression



```
In [5]: pipe = Pipeline([('Count_Vect',CountVectorizer()),
                         ('tfidf', TfidfTransformer()),
                         ('LR', LogisticRegression(n_jobs=1,max_iter=250))])
        pipe.fit(x_train, y_train)

Out[5]: Pipeline(steps=[('Count_Vect', CountVectorizer()),
                        ('tfidf', TfidfTransformer()),
                        ('LR', LogisticRegression(max_iter=250, n_jobs=1))])

In [6]: gs_params = [{'LR__C': [1e-1, 1, 1e3, 1e5],
                      'LR__solver': ['newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'],
                      'LR__penalty': ['l1', 'l2', 'elasticnet', 'none']}]
        grid_search = GridSearchCV(estimator = pipe,
                                   param_grid = gs_params,
                                   scoring = 'accuracy',
                                   cv = 4,
                                   n_jobs = -1)
        grid_search.fit(x_train, y_train)
        best_accuracy = grid_search.best_score_
        best_parameters = grid_search.best_params_
        print("Best Accuracy: {:.2f} %".format(best_accuracy*100))
        print("Best Parameters:", best_parameters)

        Best Accuracy: 92.17 %
        Best Parameters: {'LR__C': 100000.0, 'LR__penalty': 'l1', 'LR__solver': 'saga'}
```

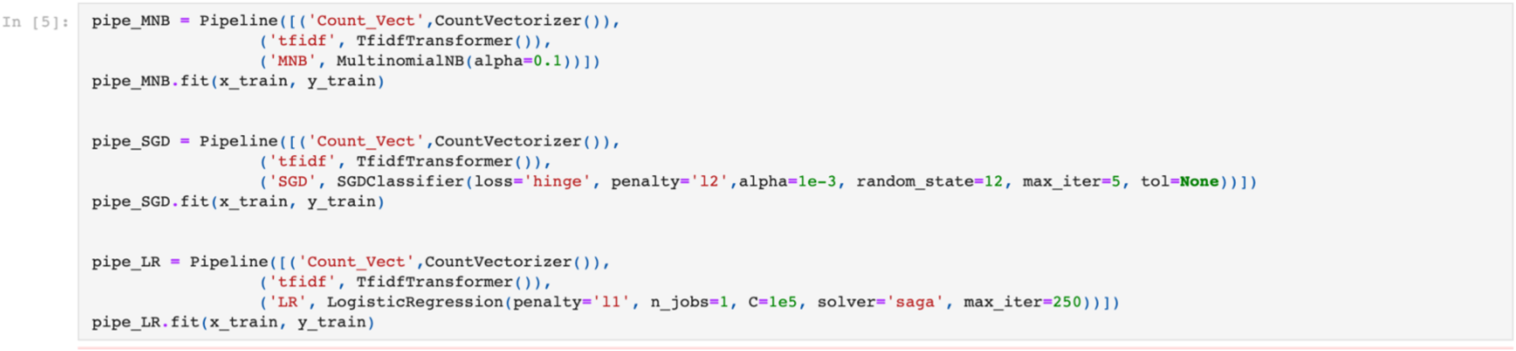
ln [5] – Pipe for Count vectorizer, TfidTransformer and LogisticRegression (Logistic regression, n.d.).

ln [6] – Grid Search for LR

Output :- Best Parameters for MultinomialNB – **{'LR__C': 100000.0, 'LR__penalty': 'l1', 'LR__solver': 'saga'}**

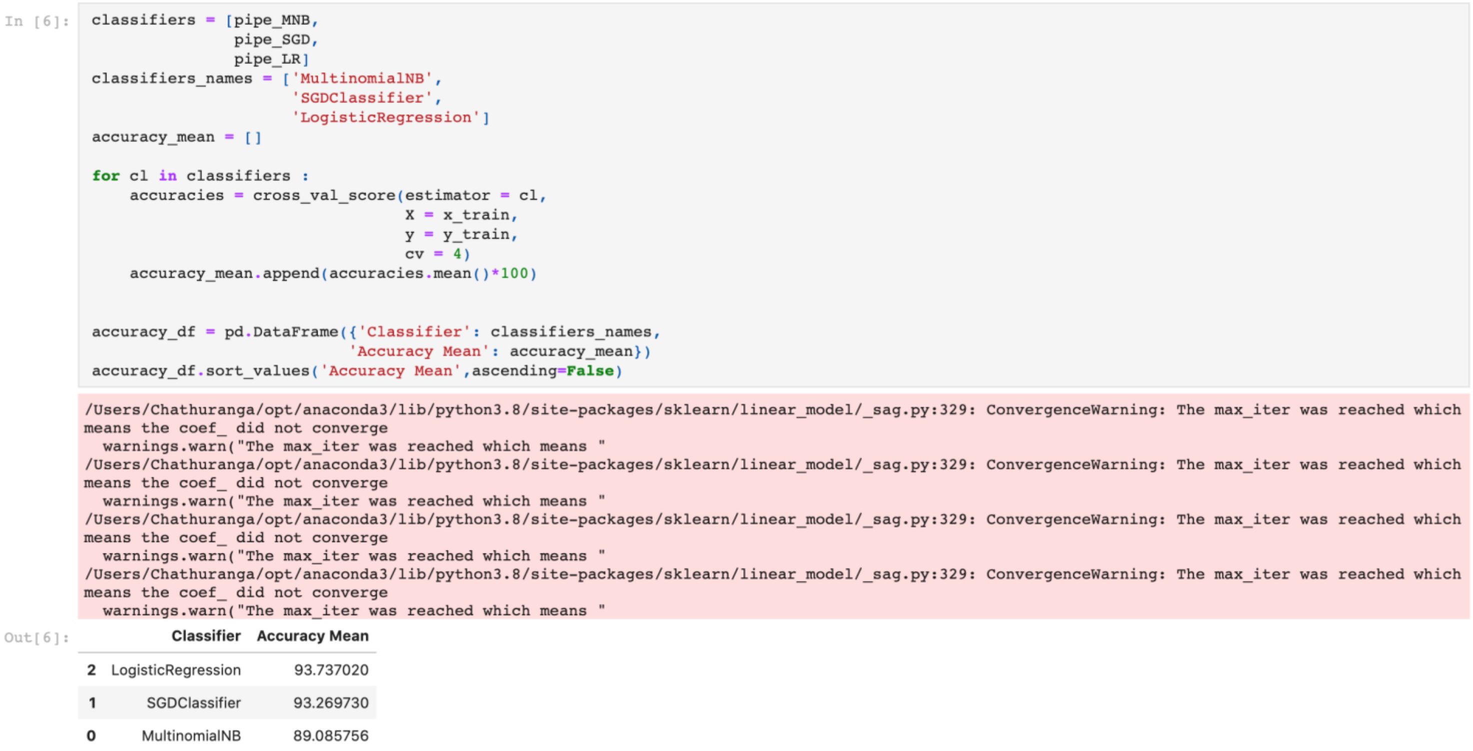
4.4 Identify Best Model

Best model evaluation was done by using the cross-validation score for each model with optimized parameters. Then the best model was chosen by considering the accuracy mean for the cross-validation. File - *Cross_Validation.ipynb*



```
In [5]: pipe_MNB = Pipeline([('Count_Vect',CountVectorizer()),
                         ('tfidf', TfidfTransformer()),
                         ('MNB', MultinomialNB(alpha=0.1))])
        pipe_MNB.fit(x_train, y_train)


        pipe_SGD = Pipeline([('Count_Vect',CountVectorizer()),
                         ('tfidf', TfidfTransformer()),
                         ('SGD', SGDClassifier(loss='hinge', penalty='l2',alpha=1e-3, random_state=12, max_iter=5, tol=None))])
        pipe_SGD.fit(x_train, y_train)


        pipe_LR = Pipeline([('Count_Vect',CountVectorizer()),
                         ('tfidf', TfidfTransformer()),
                         ('LR', LogisticRegression(penalty='l1', n_jobs=1, C=1e5, solver='saga', max_iter=250))])
        pipe_LR.fit(x_train, y_train)
```

Results :-



```
In [6]: classifiers = [pipe_MNB,
                       pipe_SGD,
                       pipe_LR]
        classifiers_names = ['MultinomialNB',
                             'SGDClassifier',
                             'LogisticRegression']
        accuracy_mean = []

        for cl in classifiers :
            accuracies = cross_val_score(estimator = cl,
                                         X = x_train,
                                         y = y_train,
                                         cv = 4)
            accuracy_mean.append(accuracies.mean()*100)


        accuracy_df = pd.DataFrame({'Classifier': classifiers_names,
                                    'Accuracy Mean': accuracy_mean})
        accuracy_df.sort_values('Accuracy Mean',ascending=False)
```

```
/Users/Chathuranga/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which
means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/Users/Chathuranga/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which
means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/Users/Chathuranga/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which
means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/Users/Chathuranga/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which
means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
```

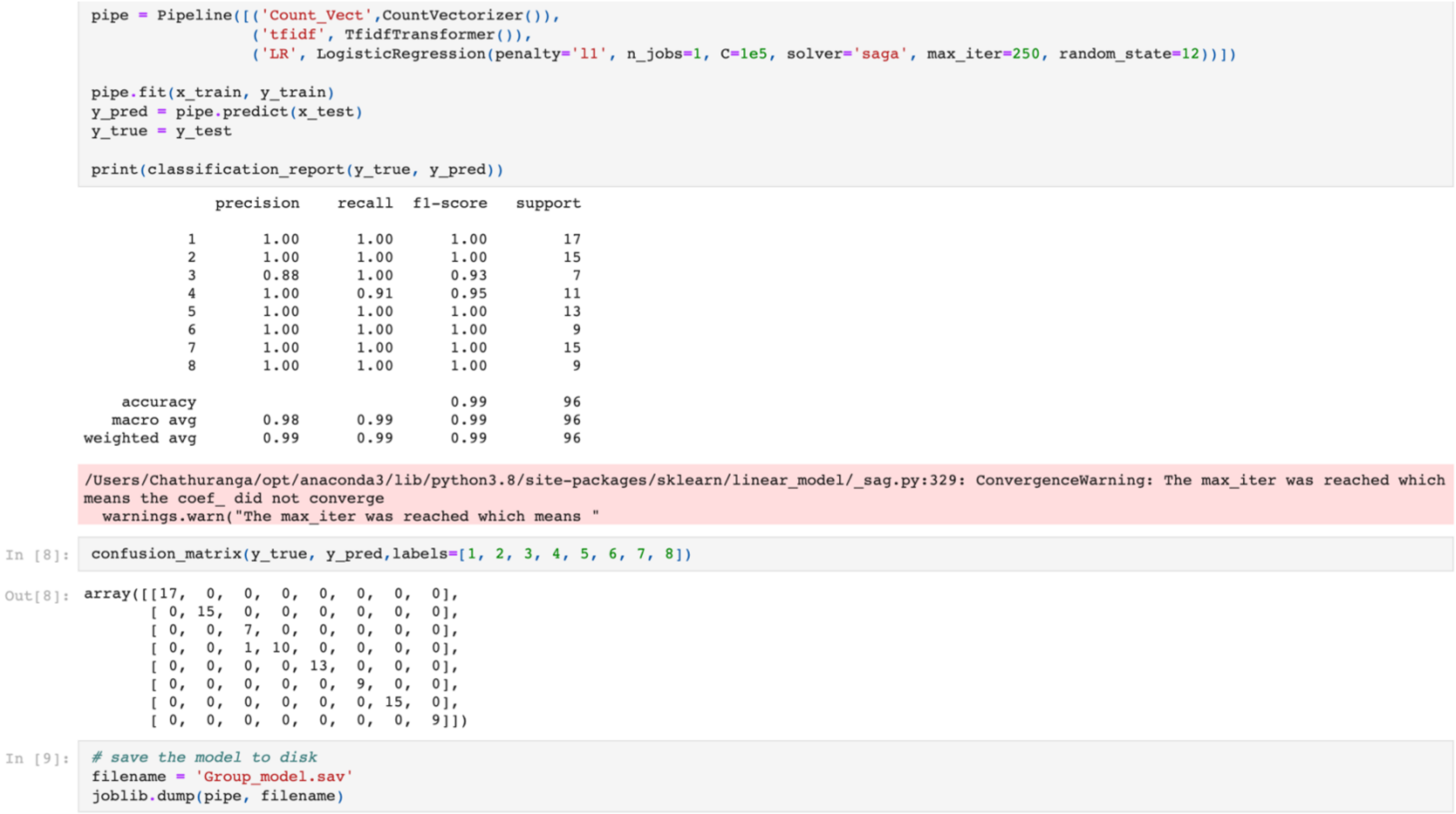
```
Out[6]:           Classifier  Accuracy Mean
        2  LogisticRegression      93.737020
        1       SGDClassifier      93.269730
        0       MultinomialNB      89.085756
```

Logistic Regressor was chosen as the best model since it has the best accuracy mean among the classifiers.

4.5 Using Best model and saving

Best model trained by using the train dataset and checked the F1 score and accuracy score. Model seems to be perform well on clustering chat messages.

File – *StarterChat_Evaluation.ipynb*



```
pipe = Pipeline([('Count_Vect',CountVectorizer()),
                 ('tfidf', TfidfTransformer()),
                 ('LR', LogisticRegression(penalty='l1', n_jobs=1, C=1e5, solver='saga', max_iter=250, random_state=12))])

pipe.fit(x_train, y_train)
y_pred = pipe.predict(x_test)
y_true = y_test

print(classification_report(y_true, y_pred))
```

```
              precision    recall  f1-score   support

           1       1.00      1.00      1.00        17
           2       1.00      1.00      1.00        15
           3       0.88      1.00      0.93         7
           4       1.00      0.91      0.95        11
           5       1.00      1.00      1.00        13
           6       1.00      1.00      1.00         9
           7       1.00      1.00      1.00        15
           8       1.00      1.00      1.00         9

    accuracy                           0.99        96
   macro avg       0.98      0.99      0.99        96
weighted avg       0.99      0.99      0.99        96
```

```
/Users/Chathuranga/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which
means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
```

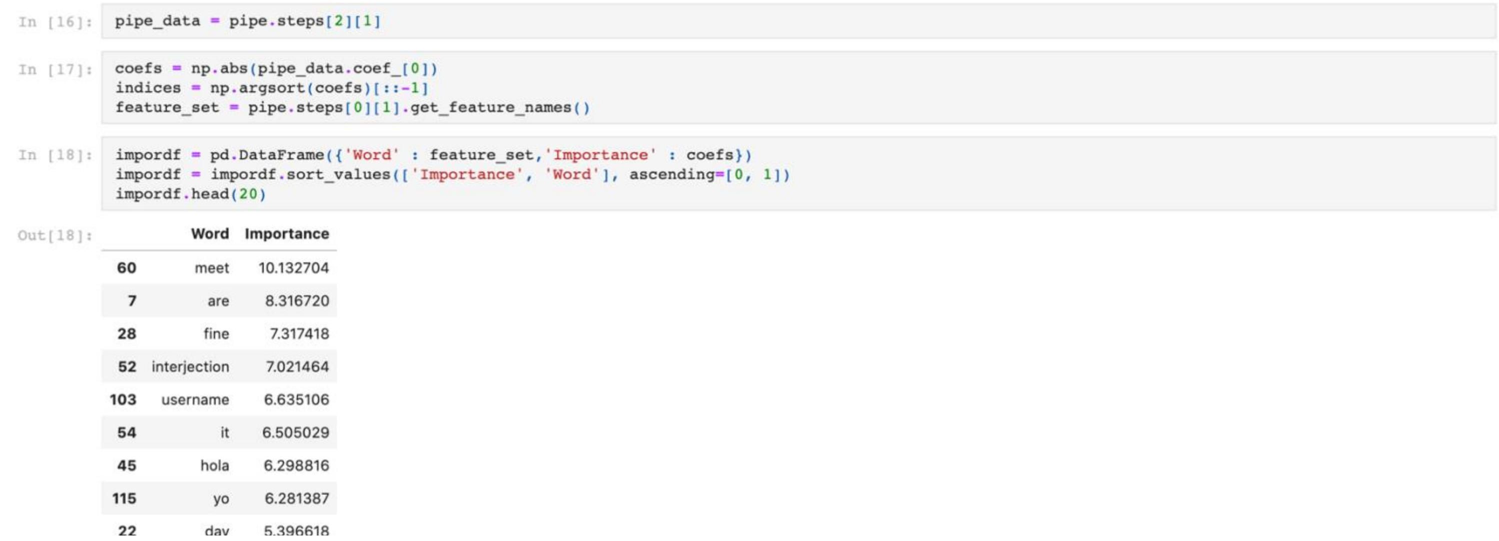
```
In [8]: confusion_matrix(y_true, y_pred,labels=[1, 2, 3, 4, 5, 6, 7, 8])
```

```
Out[8]: array([[17,  0,  0,  0,  0,  0,  0,  0],
               [ 0, 15,  0,  0,  0,  0,  0,  0],
               [ 0,  0,  7,  0,  0,  0,  0,  0],
               [ 0,  0,  1, 10,  0,  0,  0,  0],
               [ 0,  0,  0,  0, 13,  0,  0,  0],
               [ 0,  0,  0,  0,  0,  9,  0,  0],
               [ 0,  0,  0,  0,  0,  0, 15,  0],
               [ 0,  0,  0,  0,  0,  0,  0,  9]])
```

```
In [9]: # save the model to disk
        filename = 'Group_model.sav'
        joblib.dump(pipe, filename)
```

4.6 Feature Importance

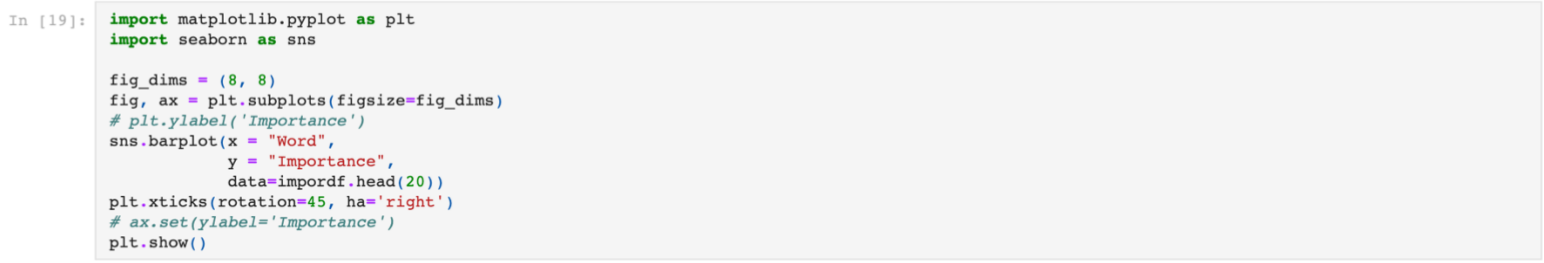
Feature importance refers to assign a score by analyzing how useful they are when predicting. For the given scenario, words can be categorized along with how useful they are when predicting classes. (How to Calculate Feature Importance With Python, n.d.)



```
In [16]: pipe_data = pipe.steps[2][1]

In [17]: coefs = np.abs(pipe_data.coef_[0])
         indices = np.argsort(coefs)[::-1]
         feature_set = pipe.steps[0][1].get_feature_names()

In [18]: impordf = pd.DataFrame({'Word' : feature_set,'Importance' : coefs})
         impordf = impordf.sort_values(['Importance', 'Word'], ascending=[0, 1])
         impordf.head(20)
```

Out[18]:

	Word	Importance
60	meet	10.132704
7	are	8.316720
28	fine	7.317418
52	interjection	7.021464
103	username	6.635106
54	it	6.505029
45	hola	6.298816
115	yo	6.281387
22	day	5.396618



```
In [19]: import matplotlib.pyplot as plt
         import seaborn as sns

         fig_dims = (8, 8)
         fig, ax = plt.subplots(figsize=fig_dims)
         # plt.ylabel('Importance')
         sns.barplot(x = "Word",
                     y = "Importance",
                     data=impordf.head(20))
         plt.xticks(rotation=45, ha='right')
         # ax.set(ylabel='Importance')
         plt.show()
```

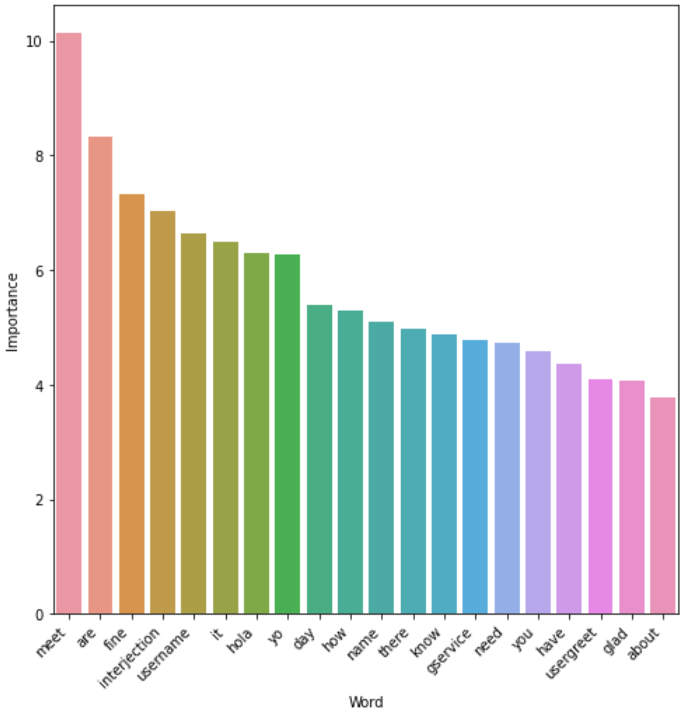


Figure 4.2 – Word Importance

4.7 Python Helper Class

There are some mentionable functions on Python helper class which is using for the chatbot.

```
def preprocess_cluster_data(self,text):
    doc = self.nlp(text)
    lemma_list = [t.lemma_ for t in doc if t.pos_ != 'PUNCT' if not t.is_stop if t.pos_ == 'NOUN' or t.pos_ == 'VERB' or t.pos_ == 'ADJ']
    return lemma_list

def load_cluster3_words(self):
    chat_data = pd.read_excel('UserChatData.xlsx', header=0)
    clust_3_df = chat_data[chat_data['Group'] == 3]
    ret_list = []

    for row in clust_3_df.itertuples():
        text_list = self.preprocess_cluster_data(row[1])
        ret_list.extend(text_list)
    self.cluster3_words = list(dict.fromkeys(ret_list))

def is_text_cluster3(self,text):
    text_word_list = self.preprocess_cluster_data(text)
    if len(list(set(text_word_list) - set(self.cluster3_words))) > 0:
        return False
    else:
        return True
```

Some new clients do not know about available services on the salon. Hence they would ask for some services that are not in the salon currently. There is a function that detects the salon services from the user chat and masks it as gservice. If it fails to identify the service the chat message will be categorized as group 3. So in here it checks NOUN, VERBS, and ADJECTIVES on the chat and detects whether it’s about on salon service.

```
def remove_names(self,text):
    if self.caller_name != None:
        new_set = text.lower().replace(self.caller_name.lower(), 'username')
        return new_set
    else:
        return text

def remove_greetings(self,text):
    if self.welcome_greeting != None:
        new_set = text.lower().replace(self.welcome_greeting.lower(), 'usergreet')

        return new_set
    else:
        return text

def remove_welcomeINTJ(self,text):
    if self.welcome_intj != None:
        new_set = text.lower().replace(self.welcome_intj.lower(), 'interjection')
        return new_set
    else:
        return text

```

```python
def remove_services(self,text):
    ret_text = text
    for serv in self.checked_services_tags:
        ret_text = ret_text.lower().replace(serv, 'gservice')
    return ret_text
```

Remove names, user greetings, interjections and salon services from the chat and masks as username, usergreet, interjection, gservice.

```python
def check_name(self,doc):
    self.matches = self.matcher(doc)

    if len(self.matches) != 0:
        for match_id, start, end in self.matches:
            string_id = self.nlp.vocab.strings[match_id]
            if string_id == "Caller_Name1":
                print('Pass')
                self.caller_name = str(doc[start+1:end]).title()
                if self.caller_name[len(self.caller_name) - 1] == ".":
                    self.caller_name = self.caller_name[0:len(self.caller_name) - 1]
                self.re_name = self.caller_name
                break
            elif string_id == "Caller_Name2":
                self.caller_name = str(doc[start]).title()
                if self.caller_name[len(self.caller_name) - 1] == ".":
                    self.caller_name = self.caller_name[0:len(self.caller_name) - 1]
                self.re_name = self.caller_name
                break
    else:
        pass

def check_greeting(self,doc):
    self.matches = self.matcher(doc)

    if len(self.matches) != 0:
        for match_id, start, end  in self.matches:
            string_id = self.nlp.vocab.strings[match_id]
            if string_id == "Greeting1" or string_id == "Greeting2" or string_id == "Greeting3" or string_id == "Greeting4":
                self.is_welcome_greeting = True
                self.welcome_greeting = str(doc[start:end])
                break
            else:
                pass
    else:
        pass

def check_intj(self,doc):
    self.matches = self.matcher(doc)

    if len(self.matches) != 0:
        for match_id, start, end  in self.matches:
            string_id = self.nlp.vocab.strings[match_id]
            if string_id == "INTJ1":
                self.welcome_intj = str(doc[start:end])
            else:
                pass
    else:
        pass

def check_services(self,doc):
    for ent in doc.ents:
        if ent.label_ == "GEN_SERVICE":
            ind_val = self.gen_service_tags_df.loc[self.gen_service_tags_df['Tag'] == ent.text.lower()]
            self.checked_services_codes.append(ind_val['Code'].values[0])
            self.checked_services_tags.append(ent.text.lower())
            self.is_newProd_check = True
        else:
            pass

def check_client_name(self,text):
    doc = self.nlp(text)
    name_list = [t.text for t in doc if t.pos_ == 'PROPN' if not t.is_stop]

    if len(name_list) != 0:
        det_name = self.listToString(name_list)
        # print("Name detected:- ", det_name)
        self.re_name = det_name.title()
    else:
        # print("Name not detected!")
        self.re_name = ""
```

Detect names, user greetings, interjection and salon available services. Spacy Matches was used to identify names, user greetings and interjections as below:-

```
def add_matchers(self):
    name_pattern_1 = [{'POS': 'AUX', 'OP': '+'},
                        {'POS': 'PROPN', 'OP': '+'}]
    name_pattern_2 = [{'POS': "PROPN"},
                        {'LOWER': 'here'}]

    # Add General Greeting Matchers
    greeting_pattern_1 = [{'LOWER': 'good', 'OP': '*'},
                            {'LOWER': 'morning'}]
    greeting_pattern_2 = [{'LOWER': 'good', 'OP': '*'},
                            {'LOWER': 'evening'}]
    greeting_pattern_3 = [{'LOWER': 'good', 'OP': '*'},
                            {'LOWER': 'afternoon'}]
    greeting_pattern_4 = [{'LOWER': 'good', 'OP': '*'},
                            {'LOWER': 'noon'}]

    # Detect Interjection
    intj1 = [{'POS': 'INTJ'}]




    self.matcher.add("Caller_Name1", None, name_pattern_1)
    self.matcher.add("Caller_Name2", None, name_pattern_2)
    self.matcher.add("Greeting1", None, greeting_pattern_1)
    self.matcher.add("Greeting2", None, greeting_pattern_2)
    self.matcher.add("Greeting3", None, greeting_pattern_3)
    self.matcher.add("Greeting3", None, greeting_pattern_4)
    self.matcher.add("INTJ1", None, intj1)
```

Spacy Entity was used to identify salon related keywords. Keywords could be added, delete and modify the created excel sheet('Services_tags.xlsx'). So the owner could add new services or modify them without changing the code.

```python
def add_service_entities(self):
    service_data = self.gen_service_tags_df[self.gen_service_tags_df['Type'] == 1]

    service_list = service_data['Tag'].tolist()

    service_entity = Entity(keywords_list=service_list, label='GEN_SERVICE')
    self.nlp.add_pipe(service_entity, last=True)
```

Services_tags.xlsx

Tag	Code	Type
haircut	G001	1
general hair cut	G001	1
cut	G001	1
children haircut	G001-1	1
childrens haircut	G001-1	1
children's haircut	G001-1	1
men's haircut	G001-2	1
mens haircut	G001-2	1
men haircut	G001-2	1
women's haircut	G001-3	1
womens haircut	G001-3	1
women haircut	G001-3	1
ladies haircut	G001-3	1
ladies haircut	G001-3	1
hair setup	G002	1
hair setups	G002	1
setup	G002	1
setups	G002	1
hair dressings	G003	1
hair dressing	G003	1
dressings	G003	1
dressing	G003	1
massage	G004	1
head massage	G004	1
scalp massage	G004	1

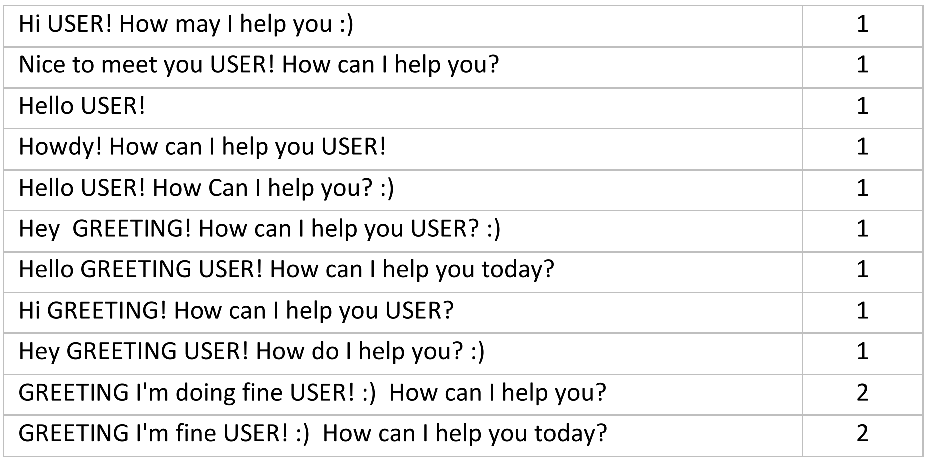
hair massage	G004	1
facial	G005	1
facials	G005	1
pedicure	G006	1
manicure	G007	1
hair colouring	G008	1
hair coloring	G008	1
colouring	G008	1
coloring	G008	1

Figure 4.3 – Service Tags xlsx

Chatbot Replies

```
def get_general_greeting(self):
    #greet_data = pd.read_excel('ReplyChat.xlsx', header=0)

    reply_chat = self.reply_data[self.reply_data['Type'] == 1]
    item = random.randint(len(reply_chat))
    reply_chat = reply_chat['Chat'].tolist()[item]

    if self.caller_name != None:
        reply_chat =  reply_chat.replace("USER", self.caller_name)
    else:
        reply_chat =  reply_chat.replace("USER", "")

    if self.time_greet != None:
        reply_chat = reply_chat.replace("GREETING", self.time_greet)
    else:
        reply_chat = reply_chat.replace("GREETING", "")

    return reply_chat

def get_checkback_greeting(self):
    #greet_data = pd.read_excel('ReplyChat.xlsx', header=0)

    reply_chat = self.reply_data[self.reply_data['Type'] == 2]
    item = random.randint(len(reply_chat))
    reply_chat = reply_chat['Chat'].tolist()[item]

    if self.caller_name != None:
        reply_chat =  reply_chat.replace("USER", self.caller_name)
    else:
        reply_chat =  reply_chat.replace("USER", "")

    if self.time_greet != None:
        reply_chat = reply_chat.replace("GREETING", self.time_greet)
    else:
        reply_chat = reply_chat.replace("GREETING", "")

    return reply_chat
```

Return prepopulated greeting reply if the chat group is 1 or 2. By the way, if the system already has the user name, username will be replaced by the customer name. And the same will applies to greeting.



Hi USER! How may I help you :)	1
Nice to meet you USER! How can I help you?	1
Hello USER!	1
Howdy! How can I help you USER!	1
Hello USER! How Can I help you? :)	1
Hey GREETING! How can I help you USER? :)	1
Hello GREETING USER! How can I help you today?	1
Hi GREETING! How can I help you USER?	1
Hey GREETING USER! How do I help you? :)	1
GREETING I'm doing fine USER! :) How can I help you?	2
GREETING I'm fine USER! :) How can I help you today?	2

Figure 4.4 – Reply Chat – Group 1&2 xlsx

Salon owner could add new service tags and also descriptions about available services. All the service-related information and service codes are saved on Service_details.xlsx file.

Code	Description
G000	We give you current hairstyle tips and treatments around ongoing hair trends, always with professional hair care products. We provide Hair cutting, Coloring, Dressing, Steups, Hair massages, Facials, Predicure and Manicures. Our prices are based on our experience and on time and material needs, and are from-prices. Exact price can be given after consultation with your hairdresser. We are a cashless salon, we take debit / credit cards. By the way you could make appointments through chat :)
G001	We are providing Hair cuts for Ladies, Gents and Children hair cuts. Ladies Hair cut 20$ upwards. Gents Hair cut 10$ upwards. And we have special children hair cuts 6$ upwards.
G001-1	Haircut for those a little younger. Applies to children up to 10 years. From $ 15
G001-2	We have special haircut styles for Men. From $ 20
G001-3	We have special haircut styles for Womens. From $ 20
G002	We provide hair setups with complementry face massage. From $ 30
G003	We have hair dressings with complementry facials. From $ 30
G004	30 luxurious minutes with care for your hair and scalp. A special treatment. Long durability that takes care of your hair. From $ 35
G005	We will take care all of you face skin treatments and makeups with quality prodcuts in the industry. From $ 40
G006	Kick up your heels and enjoy a spa staple with a pedicure. Your feet will be polished and massaged to pretty perfection, and you’ll leave a more relaxed and more comfortable person. During a spa pedicure you should expect your feet to be soaked in warm water

	and your nails to be cut and shaped. Your nail spa therapist will use a pumice stone to buff away dry skin, and will follow up with an exfoliation. Prices from $ 50
G007	With nail salon services ranging from gel, acrylics, paraffin dips, and nail art, there is something for everyone looking for a manicure. Whether you just want a basic grooming and polish or want to go all out with 3D art and gradients. They help ensure healthy nails and hands and are a quick and easy way to change or polish your look. Prices from $ 20
G008	Simple coloring or alternatively loops (not foil loops). Haircuts and lighter styling are included. NOTE! All prices are from-prices, exact price can be given after consultation with your hairdresser. In some cases, decolorization / deep cleaning may be needed before other treatment. Have you dyed your hair before or done any other type of color treatment? Contact the salon before booking.
ERROR	Sorry I'm not aware about this service. Hence I will inform this to the management and will reach back to you.

Figure 4.5 – Service details xlsx

If the predicted group number is 3, General information about the salon (G000) will be provided for the client. If the predicted group number is 4, a Product description will be provided for the client as the above. If the product is not found on service list, ERROR message will be provided for the client.

Appointment Add/Change/Cancel

For Groups 5,6,7 there is special criteria to follow.

Make Reservation :-

- User should provide appointment person name, and get name verification from the client.
- User should provide required appointment date, required service and phone number. Then again information will be get verified from client.
- Make appointment inquiry and show reply message.

Change Reservation :-

- User should provide appointment person name and get name verification from the client.
- User should provide new appointment date and phone number.

Cancel/Remove Reservation :-

- User should provide appointment person name and get name verification from the client.
- User should provide phone number.

NOTE :- All the chatbot replies connected with appointment were stored on ‘Appointment_Reply.xlsx’ file. It could be easily changed according to the owner preferences and independent from the code base.

When a chatbot needs confirmation from the user, it asks questions from the user. Such as:- (Questions could be found on Appointment_Reply)

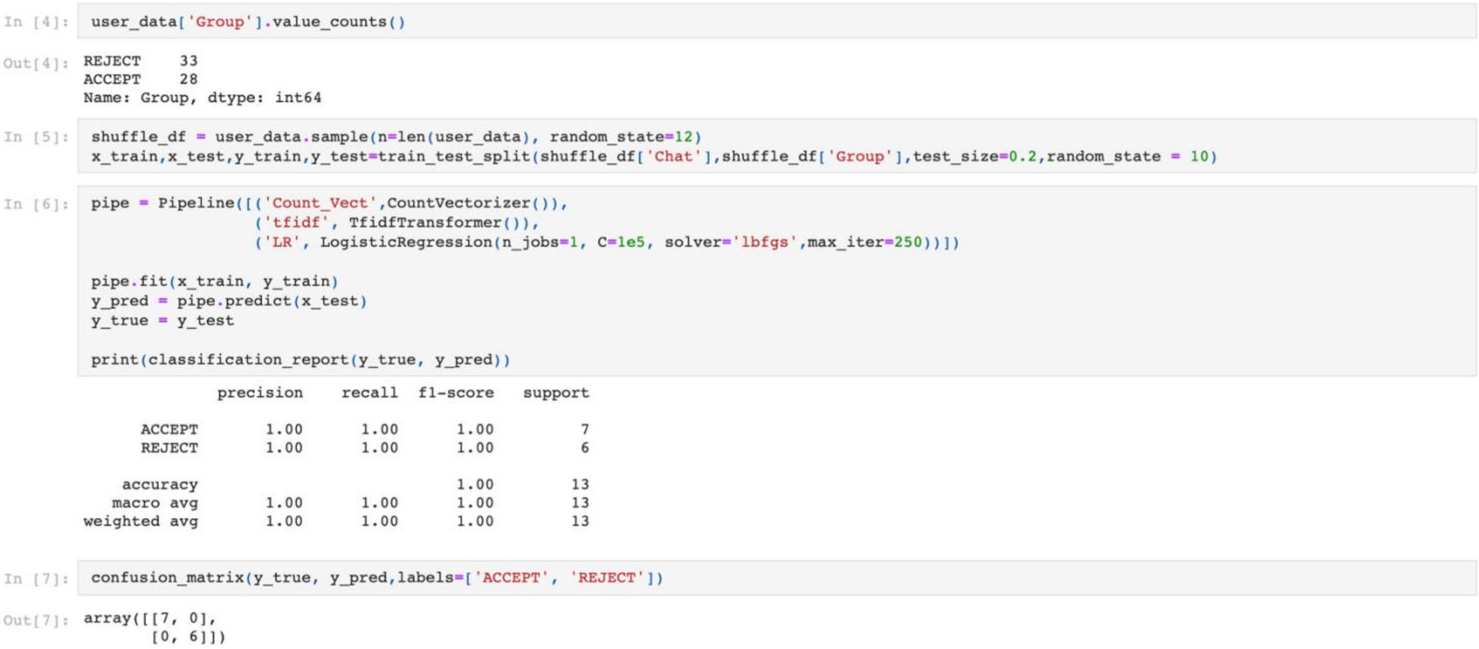
Chat	Type	Group
Is this appointment for you USER?	P	5
So is this for you USER?	P	5
So the appointment for USER? Isn't it ?	P	5
I'm making the appointment for USER. Okay ? :)	P	5
So could you please tell me the person name for the appointment?	NA	5
First I need the person name for the appointment :)	NA	5
Sorry I'm unable find an appointment person name. Could you please tell me the name ?	E	5
Ohh okay :) So the appointment for USER? Right ? :)	N	5
So this appointment for USER? Isn't it ?	N	5
Was this appointment made for you USER?	P	6
So was this for you USER?	P	6
So was the appointment for USER? Isn't it ?	P	6
So could you please tell me the person name for the appointment which you have placed?	NA	6
First I need the person name for the appointment :)	NA	6

Sorry I'm unable find an appointment person name. Could you please tell me the name ?	E	6
So the appointment was made for USER? Right ? :)	N	6
So this appointment was made for USER? Isn't it ?	N	6
Was this appointment made for you USER?	P	7
So was this for you USER?	P	7
So was the appointment for USER? Isn't it ?	P	7
So could you please tell me the person name for the appointment which you have placed?	NA	7
First I need the person name for the appointment :)	NA	7
Sorry I'm unable find an appointment person name. Could you please tell me the name ?	E	7
So the appointment was made for USER? Right ? :)	N	7
So this appointment was made for USER? Isn't it ?	N	7

Figure 4.6 – Appointment_reply. xlsx

There is a separate model that evaluates whether the user gives a positive or negative response to the above questions. If the user gives positive feedback for the above questions then the chatbot knows that the provided details are true. If not chatbot again asks the question.

User feedback data(UserAccpetReject.xlsx) has been collected from call center associate since there are no available data from the salon's side. The Trained model was saved as ' Acc_Rej_model.sav'.



```
In [4]:  user_data['Group'].value_counts()

Out[4]:  REJECT    33
         ACCEPT    28
         Name: Group, dtype: int64

In [5]:  shuffle_df = user_data.sample(n=len(user_data), random_state=12)
         x_train,x_test,y_train,y_test=train_test_split(shuffle_df['Chat'],shuffle_df['Group'],test_size=0.2,random_state = 10)

In [6]:  pipe = Pipeline([('Count_Vect',CountVectorizer()),
                          ('tfidf', TfidfTransformer()),
                          ('LR', LogisticRegression(n_jobs=1, C=1e5, solver='lbfgs',max_iter=250))])

         pipe.fit(x_train, y_train)
         y_pred = pipe.predict(x_test)
         y_true = y_test

         print(classification_report(y_true, y_pred))

                       precision    recall  f1-score   support

              ACCEPT       1.00      1.00      1.00         7
              REJECT       1.00      1.00      1.00         6

            accuracy                           1.00        13
           macro avg       1.00      1.00      1.00        13
        weighted avg       1.00      1.00      1.00        13

In [7]:  confusion_matrix(y_true, y_pred,labels=['ACCEPT', 'REJECT'])

Out[7]:  array([[7, 0],
                [0, 6]])
```

Figure 4.7 – Accpet_Reject_Grouping. xlsx

Close and exit chat

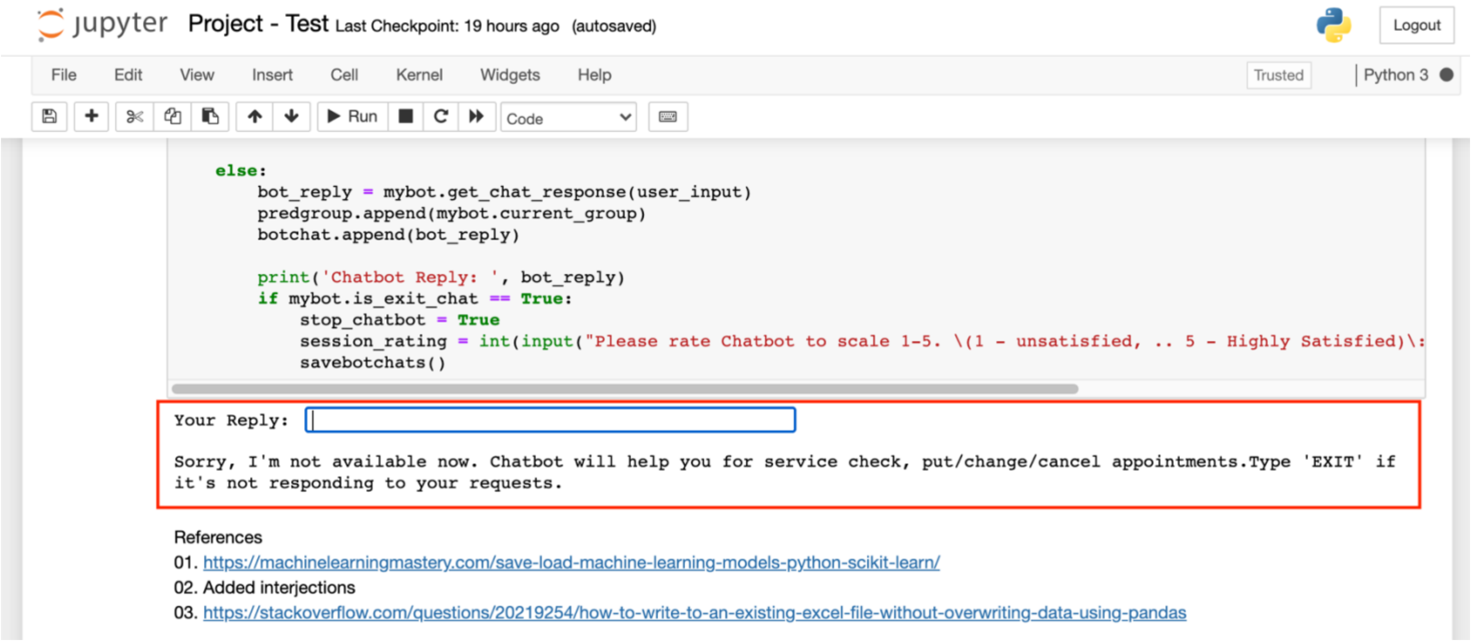
If the user gives feedback regarding the service (eg:- Thank you, Bye, Thank you for the details.) or end of the appointment process, chatbot closes automatically by predicting class 8. If the user does not get any required service or if the chatbot is not working properly user could enter ‘EXIT’ to force quite the chatbot.

Evaluation Step

After the successful conversation feedback message will be displayed and request for client feedback about chatbot service. Then customer feedback will be saved along with the history of the chat for later analysis. By the way we could use chat history for future training phases and increase the chatbot accuracy.

5. Results

The chatbot could launch on Project – Test.ipynb file. It could open on Jupyter notebook and it shows chatbot text on the below of the page.



```
    else:
        bot_reply = mybot.get_chat_response(user_input)
        predgroup.append(mybot.current_group)
        botchat.append(bot_reply)

        print('Chatbot Reply: ', bot_reply)
        if mybot.is_exit_chat == True:
            stop_chatbot = True
            session_rating = int(input("Please rate Chatbot to scale 1-5. \(1 - unsatisfied, .. 5 - Highly Satisfied)\:
            savebotchats()
```

```
Your Reply: 

Sorry, I'm not available now. Chatbot will help you for service check, put/change/cancel appointments.Type 'EXIT' if
it's not responding to your requests.
```

References
01. https://machinelearningmastery.com/save-load-machine-learning-models-python-scikit-learn/
02. Added interjections
03. https://stackoverflow.com/questions/20219254/how-to-write-to-an-existing-excel-file-without-overwriting-data-using-pandas

Figure 5.1 – Chatbot

Chatbot starts with an open message. It clearly shows the unavailability of the salon owner and requests to chat with the bot. Customers could check product availability and make/change/cancel appointments through the chat. Sometimes chatbots will give inappropriate answers or not responding to client's chats. For those cases, the customer could use the 'EXIT' keyword for the chatbot close.

5.1 Chatbot Flow Diagram

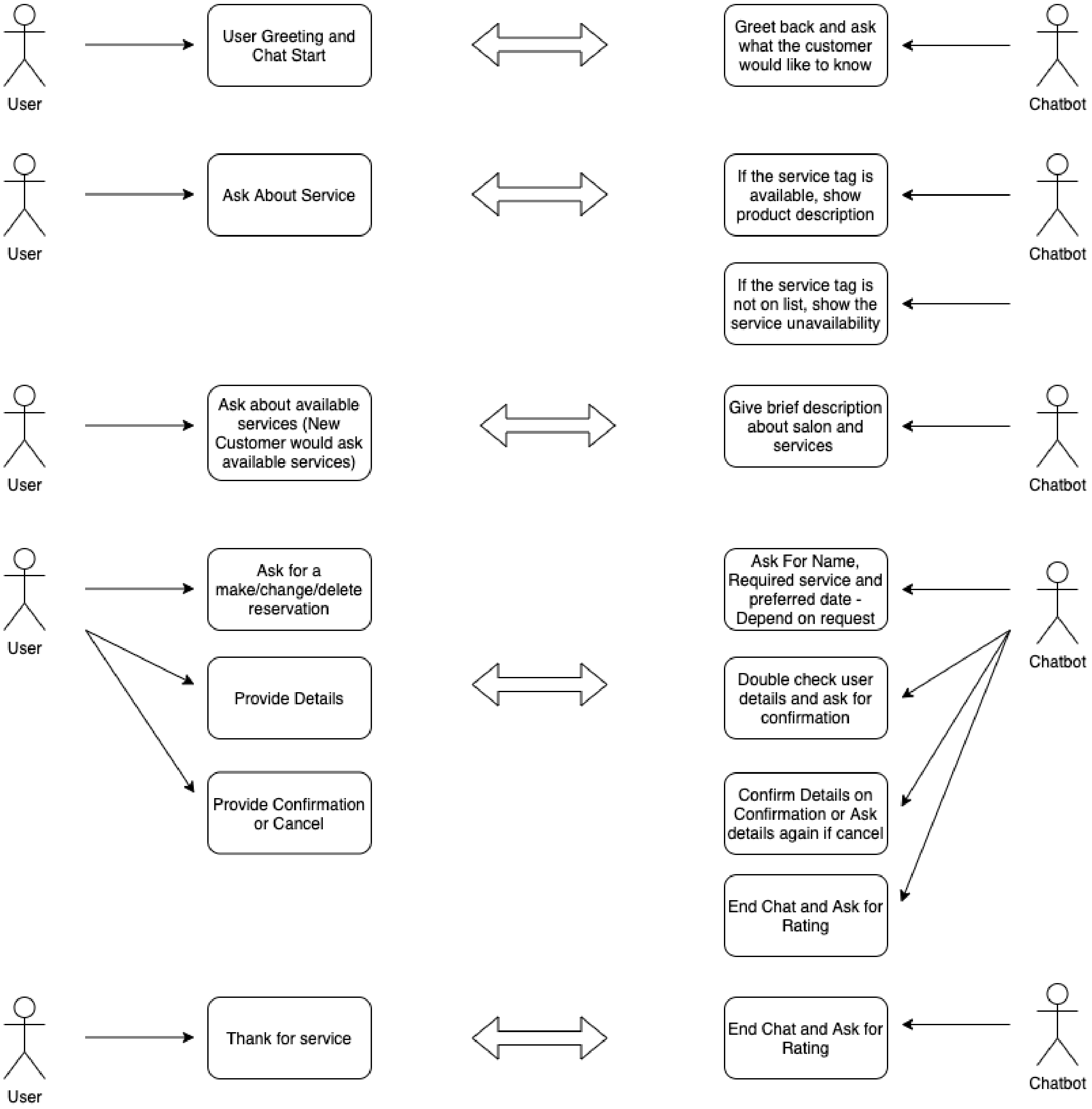

User
User Greeting and Chat Start
Greet back and ask what the customer would like to know
Chatbot
User
Ask About Service
If the service tag is available, show product description
Chatbot
If the service tag is not on list, show the service unavailability
User
Ask about available services (New Customer would ask available services)
Give brief description about salon and services
Chatbot
User
Ask for a make/change/delete reservation
Ask For Name, Required service and preferred date - Depend on request
Chatbot
Provide Details
Double check user details and ask for confirmation
Provide Confirmation or Cancel
Confirm Details on Confirmation or Ask details again if cancel
End Chat and Ask for Rating
User
Thank for service
End Chat and Ask for Rating
Chatbot

Figure 5.2 – Chatbot Flow Diagram

5.2 Chatbot Analytics

Chatbot analytics is the process of checking and evaluating chatbot historical data and generating useful insight details about the bot performances and customer experience. Chatbot history would be used to determine the performance and check the effectiveness of achieving the objectives of the bot. There are several ways to evaluate chatbot performance. Some of the matrices could be useful for the given problem. Since the chatbot is still in the testing phase, some of the user matrices could not be a check on this system.

5.2.1 Importance of Chatbot Analytics

Focus

Chatbot analytics could be used to enhance the chatbot performance. And also it enables to track breaking points where the chatbot fails due to various reasons including wrong cluster analysis. (Metrics for Chatbot Analytics, n.d.)

Customer insight generation

Chatbot historical data provide an idea about popular customer questions, paths, current trends, and exit or fallback points through historical data. Afterward, the company could analyze trends and patterns which was not noticed during the previous model training phases. This would help to better understand the customer's intentions and thoughts
(Metrics for Chatbot Analytics, n.d.)

5.2.2 Chatbot Analytics Metrices

There are several definitions for Metric. Simply Metric is a quantifiable measure that could be used to track and assess the status of a specific process. The chatbot will be evaluated through metrics. For a new chatbot like the implemented one, some of the metrics could influence dramatically and some of the metrics couldn’t be used. So these metrices could be used to monitor chatbot closely and evaluate it’s performance in several ways. There are several kind of chatbot metrices. Such as :-

1. User Metrices
2. Message Metrices
3. Bot Metrices

(Chatbot Analytics, n.d.) (Metrics for Chatbot Analytics, n.d.) (AI Chatbot Performance, n.d.)

User Metrices

- Total Users
- Active Users
- Engaged Users
- New Users

Since the chatbot is in the initial stage and currently, it does not interact with real customers. Therefore the above-mentioned metrics couldn’t be used for now. But these metrices could be used on latter stages.

Message Metrices

- Bot Messages Count
- In Messages Count
- Total Conversation Message Count

The above metrics would apply to an ongoing/live running chatbot. Since this is under the testing phase, these metrics couldn’t be checked for now.

Bot Metrices

- Fallback Rate (FBR)

  Every chatbot is not perfect. Sometimes chatbots fail to identify what the customer is talking about. So chatbots are expected to fail sometimes. But it’s essential to trace the fallback rate since chatbot is replaced as a customer service representative. So the fallback rate should be maintained as low as possible. If a high fallback rate observes, it’s better to re-evaluate the model and try to find a new data source for the training set for performance improvement.

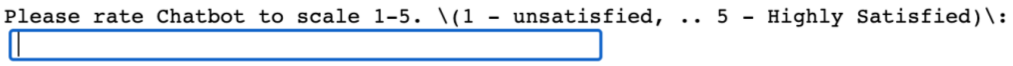
In this particular system, the user could use the “EXIT” keyword to force quit the chat. So the system automatically stores this instance and saves the chat history for inspection. Then FBR could be reviewed as:-

$$\boldsymbol{FBR = \frac{Number\ of\ force\ quit\ [\text{EXIT}]Chats}{Total\ Number\ of\ Chats}}$$

- User Satisfaction

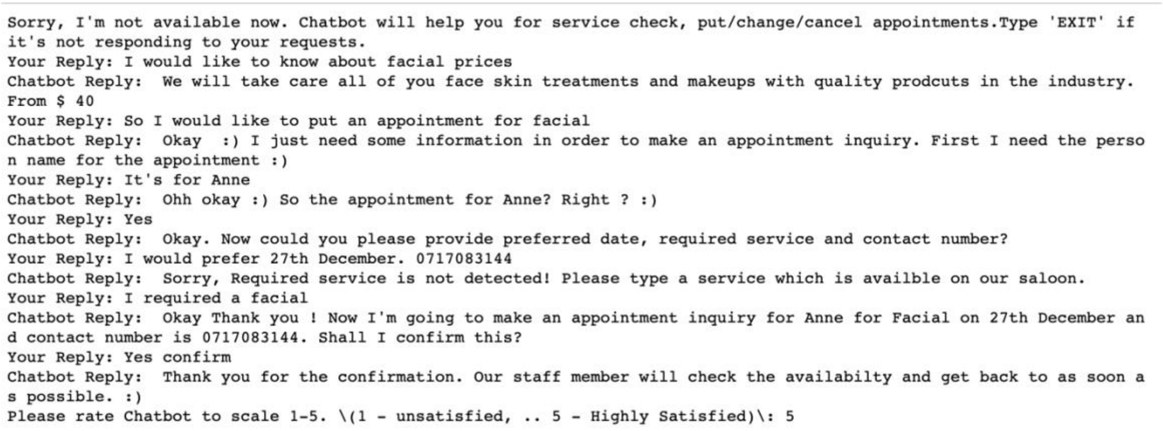
  User Feedback option could be introduced through the exit survey. People who engaged with the chatbot could rate their experience. This could be a binary value such as 1 – Good, 0 – Bad. Or a point system could be introduced. This metric could capture the overall effectiveness of the bot from the user experience point.

  In this system, a point system is introduced to track user satisfaction. Which is 1 – Bad 2 – Ok, but need lots of improvement, 3- Average, 4 – Good, Need some improvements, 5 – Excellent.



```
Please rate Chatbot to scale 1-5. \(1 - unsatisfied, .. 5 - Highly Satisfied)\:
```

Figure 5.3 – Chatbot Exit Survey – User Feedback



```
Sorry, I'm not available now. Chatbot will help you for service check, put/change/cancel appointments.Type 'EXIT' if
it's not responding to your requests.
Your Reply: I would like to know about facial prices
Chatbot Reply:  We will take care all of you face skin treatments and makeups with quality prodcuts in the industry.
From $ 40
Your Reply: So I would like to put an appointment for facial
Chatbot Reply:  Okay  :) I just need some information in order to make an appointment inquiry. First I need the perso
n name for the appointment :)
Your Reply: It's for Anne
Chatbot Reply:  Ohh okay :) So the appointment for Anne? Right ? :)
Your Reply: Yes
Chatbot Reply:  Okay. Now could you please provide preferred date, required service and contact number?
Your Reply: I would prefer 27th December. 0717083144
Chatbot Reply:  Sorry, Required service is not detected! Please type a service which is availble on our saloon.
Your Reply: I required a facial
Chatbot Reply:  Okay Thank you ! Now I'm going to make an appointment inquiry for Anne for Facial on 27th December an
d contact number is 0717083144. Shall I confirm this?
Your Reply: Yes confirm
Chatbot Reply:  Thank you for the confirmation. Our staff member will check the availabilty and get back to as soon a
s possible. :)
Please rate Chatbot to scale 1-5. \(1 - unsatisfied, .. 5 - Highly Satisfied)\: 5
```

Figure 5.4 – Chatbot Exit Survey – User Feedback example

Two Files were used for Chatbot evaluation.

- chatbot_chat_history.xlsx – All Testing Chat History
- chatbot_rating.xlsx – Chatbot Ratings

Different unique use-cases were tested on Chatbot and history each chat were stored on chatbot_chat_history file.

5.3 Chatbot Evaluation Summary

- No of tested unique uses cases - 25
- No of EXIT cases (0 -score on chat rating) - 02
- No of Rating 1 Chats - 00
- No of Rating 2 Chats - 03
- No of Rating 3 Chats - 03
- No of Rating 4 Chats - 05
- No of Rating 5 Chats - 12
- Average Rating - (12*5)+(5*4)+(3*3)+(3*2)+(1*0) / 25
  - 3.8
- Chatbot FBR Rate = 2/25 = **0.08**

At a glance, it’s clear that testcases are really small for that chatbot. But each chat was an unique use-case and in each case questions were asked from chatbot differently. In the initial testing phase 85 chats were recorded from testers.

But there were lots of chats which have a similar kind of approach and follows the same way of asking questions. So for most cases chatbot performed well and got 5/5. Even though this reflects the chatbot's good performance, it seems kind of misleading. Chatbot got really good rating because lots of users follow the same way of asking questions and chatbot provided good answers for them. Hence similar kind of use-cases (Almost all removed chats got 5/5 rating) were not considered for the evaluation. [Chat History available on file]

Generally, Chatbot seems to work well in the testing phase and sometimes chatbot failed to detect what the user needs or unable to detect what the user really talking about. By the way, some Spacy library bugs were behind on some of the chatbot bugs. Sometimes chatbot failed to identify phone numbers (Cardinality) on the given chat. It happens only for some numbers and unable to find a good explanation for this bug. And also due to the token conflict, the chatbot unable to detect some service tags. Finally, there are some limitations on the bot side and that information is explained in the Discussion section.

6. Discussion

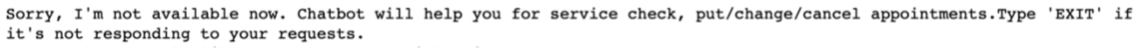
The project was shared between several individuals including the salon owner and an experienced customer service representative. The project repository was shared with people who have some software experience in using a Jupiter notebook on their machines. TeamViewer was used to demonstrate and test the chatbot for the salon owner and customer service representative.

Testing chat history and chat ratings were collected after the testing phase and used those data for chatbot evaluation. Some of the misclassified chats were identified and used some of the chats to re train the chatbot again. Some changes on the model after testing phase : -

Previous Model	New Model
Customer may use words like ‘thank you’ after the chatbot provide some information and user would like to ask an appointment. But chatbot fails to detect ‘Thank you’ and categories as cluster 8, then close chat.	Retrained the model to detect Thank you, okay, …. before asking for an appointment. Now it’ll not quite chat if the customer thanks to the chatbot for providing information in the middle of product check.
Some of the service tags were missing. Such as :- Waxing, Beard cut …	Added new service tags and product information on the excel sheet. (It’s easily could be changed on an excel sheet even without affecting the code)
Some chatbot replies are not courteous. Such as :- Please provide, Give details, …	Rephrase the chatbot replies. Such as :- Could you please provide ….

By the way, Some of the testers have requested to change the welcome note from the salon. Welcome note should direct customers, how to communicate with chatbot and provide instruction on how to chat with the bot.

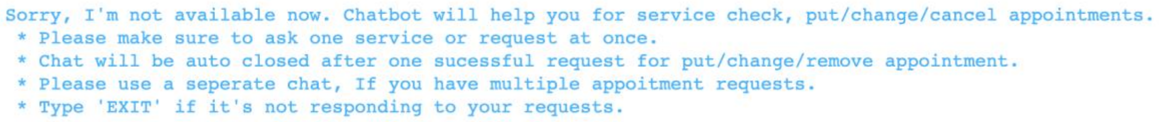
Previous Welcome Note :-



```
Sorry, I'm not available now. Chatbot will help you for service check, put/change/cancel appointments.Type 'EXIT' if
it's not responding to your requests.
```

Figure 6.1 – Previous Welcome Note

New Welcome Note :-

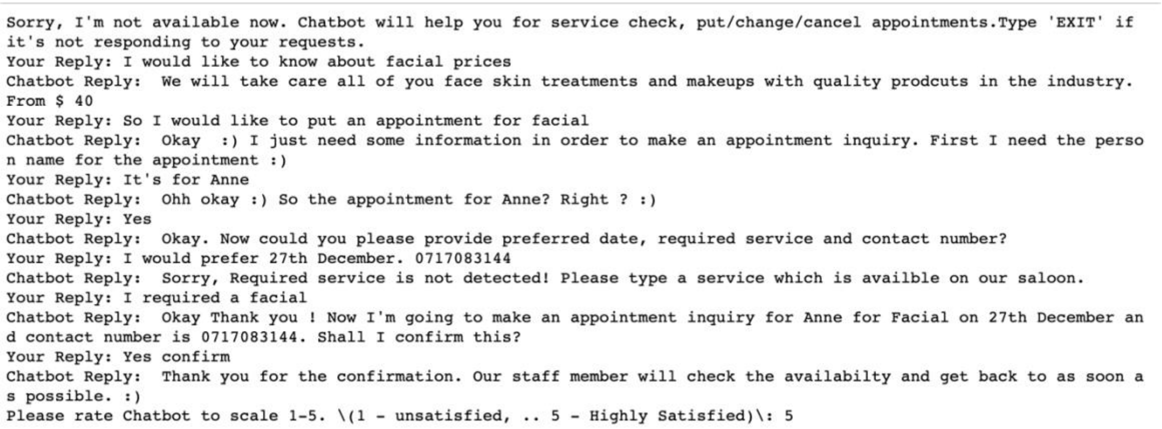


```
Sorry, I'm not available now. Chatbot will help you for service check, put/change/cancel appointments.
 * Please make sure to ask one service or request at once.
 * Chat will be auto closed after one sucessful request for put/change/remove appointment.
 * Please use a seperate chat, If you have multiple appoitment requests.
 * Type 'EXIT' if it's not responding to your requests.
```

Figure 6.2 – New Welcome Note

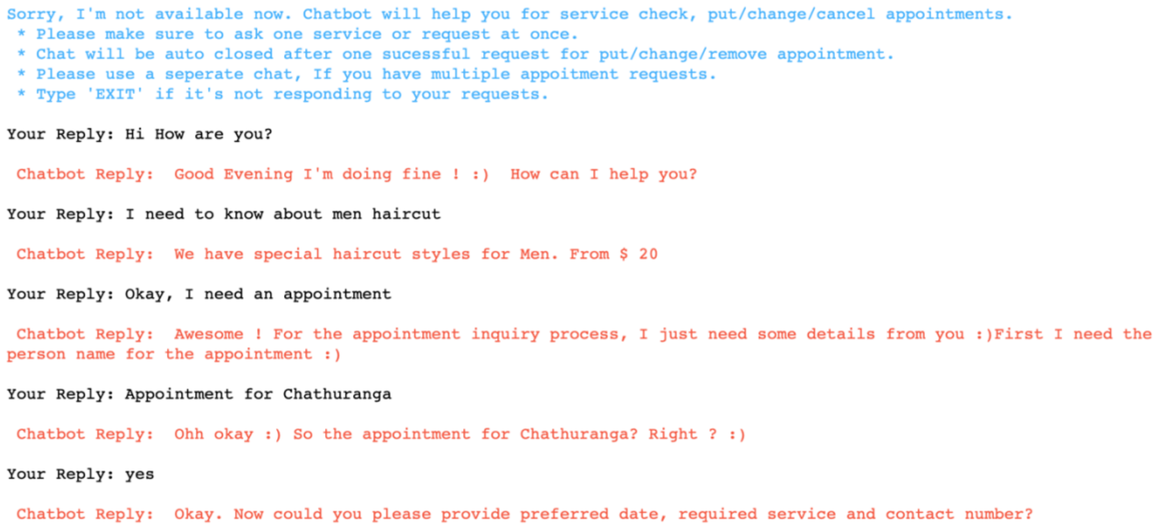
<u>Testers UI Changes</u>

And Also, testers complained about the chat UI. Since it was kind of irritating to the eyes and hard to see the chat for a long time while testing. This is kind of a testing phase for that chatbot and UI is the next level in the implementation. However, some of the changes have been made to make the chatbot easy to test and interactive for the tester.



```
Sorry, I'm not available now. Chatbot will help you for service check, put/change/cancel appointments.Type 'EXIT' if
it's not responding to your requests.
Your Reply: I would like to know about facial prices
Chatbot Reply:  We will take care all of you face skin treatments and makeups with quality prodcuts in the industry.
From $ 40
Your Reply: So I would like to put an appointment for facial
Chatbot Reply:  Okay  :) I just need some information in order to make an appointment inquiry. First I need the perso
n name for the appointment :)
Your Reply: It's for Anne
Chatbot Reply:  Ohh okay :) So the appointment for Anne? Right ? :)
Your Reply: Yes
Chatbot Reply:  Okay. Now could you please provide preferred date, required service and contact number?
Your Reply: I would prefer 27th December. 0717083144
Chatbot Reply:  Sorry, Required service is not detected! Please type a service which is availble on our saloon.
Your Reply: I required a facial
Chatbot Reply:  Okay Thank you ! Now I'm going to make an appointment inquiry for Anne for Facial on 27th December an
d contact number is 0717083144. Shall I confirm this?
Your Reply: Yes confirm
Chatbot Reply:  Thank you for the confirmation. Our staff member will check the availabilty and get back to as soon a
s possible. :)
Please rate Chatbot to scale 1-5. \(1 - unsatisfied, .. 5 - Highly Satisfied)\: 5
```

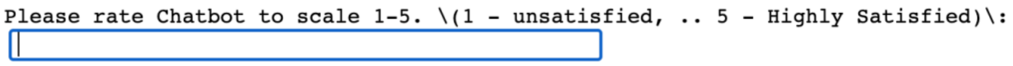
Figure 6.3 – Old UI on Jupyter Notebook



```
Sorry, I'm not available now. Chatbot will help you for service check, put/change/cancel appointments.
 * Please make sure to ask one service or request at once.
 * Chat will be auto closed after one sucessful request for put/change/remove appointment.
 * Please use a seperate chat, If you have multiple appoitment requests.
 * Type 'EXIT' if it's not responding to your requests.

Your Reply: Hi How are you?

 Chatbot Reply:  Good Evening I'm doing fine ! :)  How can I help you?

Your Reply: I need to know about men haircut

 Chatbot Reply:  We have special haircut styles for Men. From $ 20

Your Reply: Okay, I need an appointment

 Chatbot Reply:  Awesome ! For the appointment inquiry process, I just need some details from you :)First I need the
person name for the appointment :)

Your Reply: Appointment for Chathuranga

 Chatbot Reply:  Ohh okay :) So the appointment for Chathuranga? Right ? :)

Your Reply: yes

 Chatbot Reply:  Okay. Now could you please provide preferred date, required service and contact number?
```

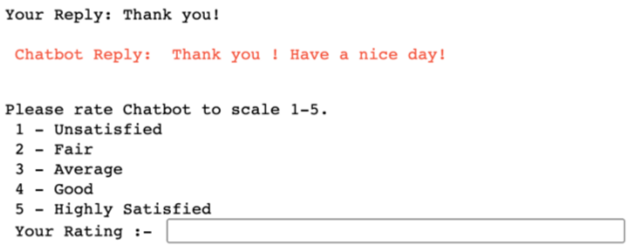
Figure 6.4 – New UI on Jupyter Notebook

Some of the testers were tried to give double value figures for user rating and the system expect an integer value. So the final user rating statement kind of misleading and some tester were tried to give double value figures and the system was unable to detect it. Then, clear instructions were shown to reduce the communication gap.



```
Please rate Chatbot to scale 1-5. \(1 - unsatisfied, .. 5 - Highly Satisfied)\:
```

Figure 6.5 – Old UI on Jupyter Notebook – User Rating



```
Your Reply: Thank you!

 Chatbot Reply:  Thank you ! Have a nice day!

Please rate Chatbot to scale 1-5.
 1 - Unsatisfied
 2 - Fair
 3 - Average
 4 - Good
 5 - Highly Satisfied
 Your Rating :-
```

Figure 6.6 – New UI on Jupyter Notebook – User Rating

6.1 Chatbot Scalability

There are so many chatbots available nowadays. But if the user needs any simple service addition, the chatbot needs to be modified on the code. So it tightly binds with the code. But this particular chatbot is for small-scale businesses and they do not spend more money on chatbot upgrades. So this issue has been solved up to a certain level on this chatbot and enhances the bot scalability.

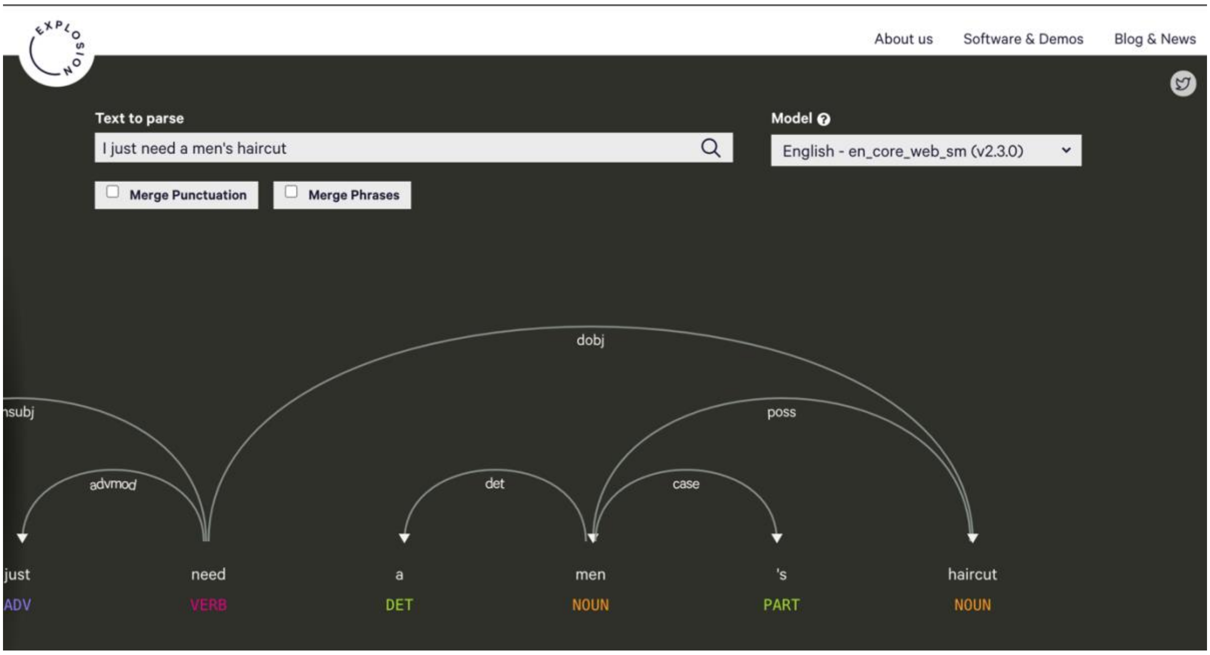
New services could be added, removed, or changed easily on the excel sheet, and also service tag description also needed to be added on another excel sheet. And also chatbot replies could be changed according to the owner's preferences.

6.2 Chatbot Limitations and Known Bugs

Since this is the initial stage of the chatbot, it has a limited amount of capabilities and trained only to capture several use-cases. As described in the earlier section, the chatbot is working fine with providing service related information and managing make/change and remove appointments. By the way, the chatbot could only record the appointment requests and push for human review. Then all requests should be manually checked by a human assistant. Currently, the chatbot does not have access to the Database system to make changes or add appointments.

Service Tag Bug

Sometimes chatbot fails to detect some of the service-related tags. The bug was discovered after several bug diagnostics sessions. It’ because of the apostrophe sign and spacy fails to detect as an entity.



Text to parse: I just need a men's haircut

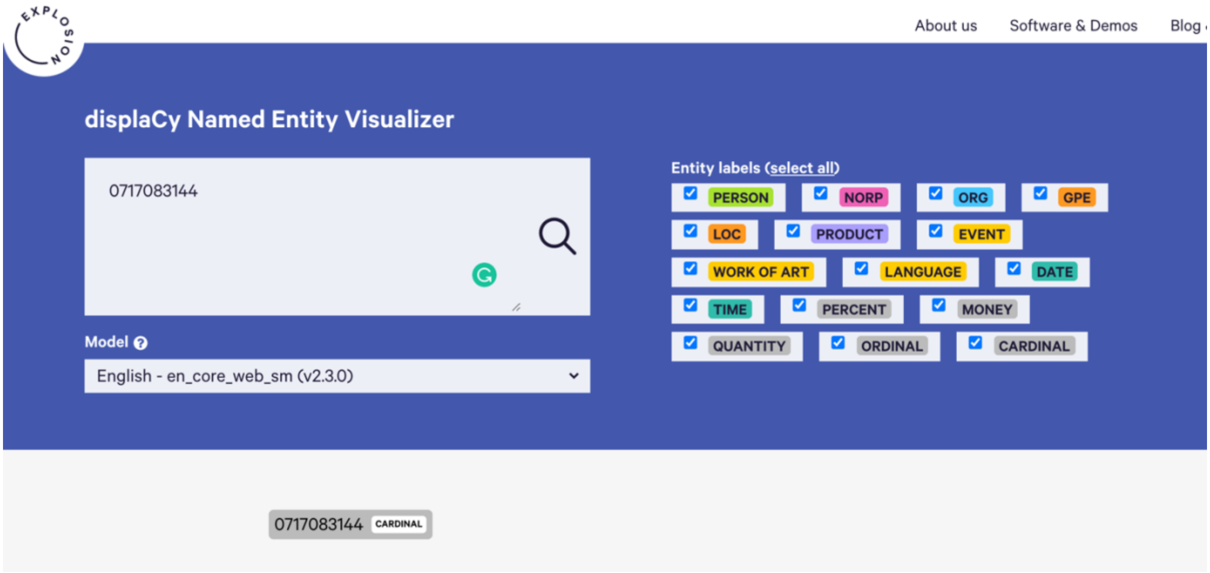
Model: English - en_core_web_sm (v2.3.0)

Figure 6.7 – Spacy Dependency Visualizer – Apostrophe sign

As the above, apostrophe sign + s detect as a separate word. So as a example if there is any product tag like men’s haircut, it won’t detect it as an entity. So this is kind of a known bug at this time. For as a workaround, tags which has apostrophe sign replaced without the sign or need to be removed on preprocess. So as an example now if any user type “mens haircut” is detectable.

Spacy Number (Cardinality) Detection Bug

In this project, Chatbot needs to detect chat phone numbers in order to save user appointment requests. During the initial development stages, phone numbers were detected successfully. But After the spacy update (v_2.3.2) now *some of the phone numbers* are not detectable. In order to demonstrate the issue, here are the Entity recognizer from the Spacy website(v_2.3.0) and from the Jupyter notebook. So there may be a version error on spacy.

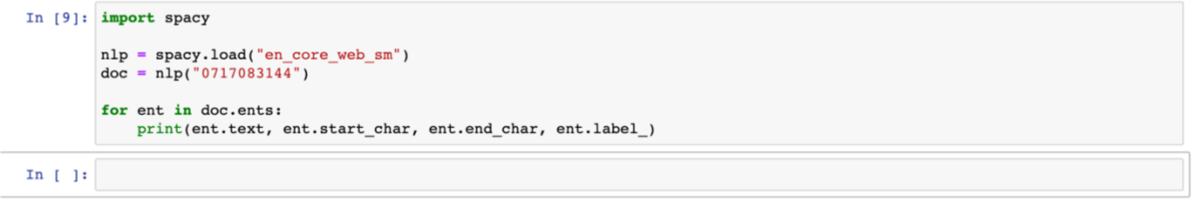


0717083144

Model: English - en_core_web_sm (v2.3.0)

0717083144 CARDINAL

Figure 6.8 – Entity Visualizer on Spacy for Phone number

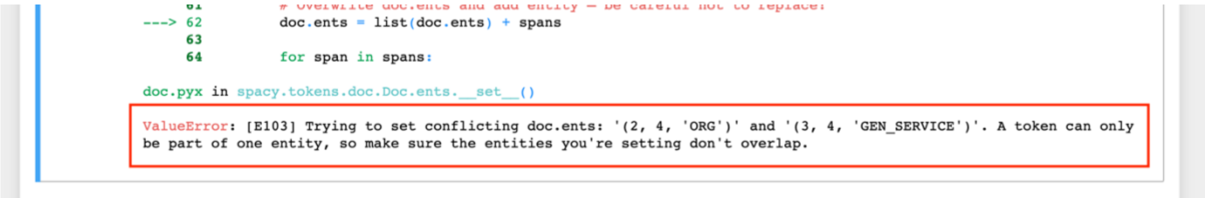


```
In [9]: import spacy

        nlp = spacy.load("en_core_web_sm")
        doc = nlp("0717083144")

        for ent in doc.ents:
            print(ent.text, ent.start_char, ent.end_char, ent.label_)

In [ ]:
```

Figure 6.9 – Entity Visualizer on Jupyter – Phone number undetectable

Spacy Token Conflict Bug

There are some services in which the bot needs to be detected on the chat. For that purpose, service tags were introduced and initialized as a custom **Entity**. In the Spacy, it’s not allowed to make multiple entities for a single word. Because of that some of the service tags have conflicts with other entity tags. Currently, it’s a known bug and will try to introduce service tags as an **EntityRuler** or **EntityLinker** on next phase.



```
     61         # overwrite doc.ents and add entity – be careful not to replace!
---> 62         doc.ents = list(doc.ents) + spans
     63
     64         for span in spans:

doc.pyx in spacy.tokens.doc.Doc.ents.__set__()

ValueError: [E103] Trying to set conflicting doc.ents: '(2, 4, 'ORG')' and '(3, 4, 'GEN_SERVICE')'. A token can only
be part of one entity, so make sure the entities you're setting don't overlap.
```

Figure 6.10 – Token Conflict Bug

7. Conclusion

I’ve gained so much experience in Natural language processing through this project. At the beginning of the course, I didn’t have any knowledge of Text Mining. Then gradually I learned about Text Mining techniques and NLP libraries. I got more information about the Spacy and how to use some of the powerful functionalities. So I hope to dig down more on the spacy model creation and custom model creation. Then I could make the chatbot more user-friendly.

It’s been an interesting project, which I feel that my small attempt to solve a real-world problem would help someone to solve an existing problem. Now I have the feeling that this chatbot project could expand to next level and have some space to make some performance improvements. Even though there are some existing bugs, the chatbot could perform well after several model training sessions and extracting more real customers for training.

UI and Database connectivity will be focused on the next phase. So then chatbot could directly check available services, put appointment and make changes to the existing appointment. Then chatbot could handle user requests without human supervision. However, the chatbot needs to be trained well on the real customer chats and new clusters could be introduced to give better service to the customers.

Project Repository :- https://github.com/mudithsilva/text-mining/tree/master/Final%20Project

Chatbot Live :- https://github.com/mudithsilva/text-mining/blob/master/Final%20Project/Project%20-%20Test.ipynb

Note :-

- Open **Project - Test.ipynb**
- Kernel -> Restart & Run All

Bibliography

Spacy - Tokenization. (n.d.). Retrieved from https://spacy.io/usage/linguistic-features

Spacy - Stop Words. (n.d.). From https://spacy.io/usage/adding-languages#stop-words

scikit - CountVectorizer. (n.d.). From https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html

Wikipedia - tf-idf. (n.d.). From https://en.wikipedia.org/wiki/Tf%E2%80%93idf

Stackoverflow-elksie5000. (n.d.). *How to extract feature importances from an Sklearn pipeline*. From Stackoverflow: https://stackoverflow.com/questions/38787612/how-to-extract-feature-importances-from-an-sklearn-pipeline

Edpresso - CountVectorizer in Python. (n.d.). From Educative: https://www.educative.io/edpresso/countvectorizer-in-python

Naive Bayes classifier. (n.d.). From Wikipedia: https://en.wikipedia.org/wiki/Naive_Bayes_classifier

Hyperparameter optimization. (n.d.). From Wikipedia: https://en.wikipedia.org/wiki/Hyperparameter_optimization

Stochastic gradient descent. (n.d.). From Wikipedia: https://en.wikipedia.org/wiki/Stochastic_gradient_descent

Logistic regression. (n.d.). From Wikipedia: https://en.wikipedia.org/wiki/Logistic_regression

How to Calculate Feature Importance With Python. (n.d.). From Machine Learning Mastery: https://machinelearningmastery.com/calculate-feature-importance-with-python/

Metrics for Chatbot Analytics. (n.d.). From AI Multiple: https://research.aimultiple.com/chatbot-analytics/

Chatbot Analytics. (n.d.). From REVE Chat: https://www.revechat.com/blog/chatbot-analytics-metrics/

AI Chatbot Performance. (n.d.). From Inbenta: https://www.inbenta.com/en/blog/10-key-metrics-to-evaluate-your-ai-chatbot-performance/#:~:text=Chatbot%20user%20experience%20metrics&text=Self%2Dservice%20rate%3A%20percentage%20of,action%20after%20using%20the%20bot.&text=Average%20number%20of%20interact