2020/2021

A picture containing shape

Description automatically generated

**732A92 Text Mining Final Project**

**Chatbot for a service company**

Student

K.M. Mudith Chathuranga Silva (kogsi273)

Examiner

Prof. Marco Kuhlmann

Table of Contents

[Abstract 3](#_Toc59653675)

[1. Introduction 4](#_Toc59653676)

[1.1 Motivation 4](#_Toc59653677)

[1.2 Objective 4](#_Toc59653678)

[2. Theory 5](#_Toc59653679)

[3. Data 9](#_Toc59653680)

[4. Method 13](#_Toc59653681)

[4.1 Brainstorming 13](#_Toc59653682)

[4.2 Customer Chat Analysis and Create Gold Label Class 14](#_Toc59653683)

[4.3 Check ML models and parameter optimization 14](#_Toc59653684)

[4.4 Identify Best Model 16](#_Toc59653685)

[4.5 Using Best model and saving 17](#_Toc59653686)

[4.6 Feature Importance 18](#_Toc59653687)

[4.7 Python Helper Class 19](#_Toc59653688)

[5. Results 27](#_Toc59653689)

[Bibliography 27](#_Toc59653690)

# 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

## 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.

## 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.)

* 𝑁 \_denotes the number of documents in the collection.

Text Classification pipeline

Document

Vectorizer

CountVectorizer

Matrix of Document Vector

Tfidtransformer

Tf-id Transform

Matrix of Document Vector

Eg:- SGD,MNB…

Predictor

Vector of Class Labels

*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.

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

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

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

Predicted class for the document.

W Count of the word W in the document

Logistic Regression Classifier

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

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

Chart, line chart, histogram

Description automatically generated

# 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** (Stackoverflow-elksie5000, n.d.)

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*

Table

Description automatically generated

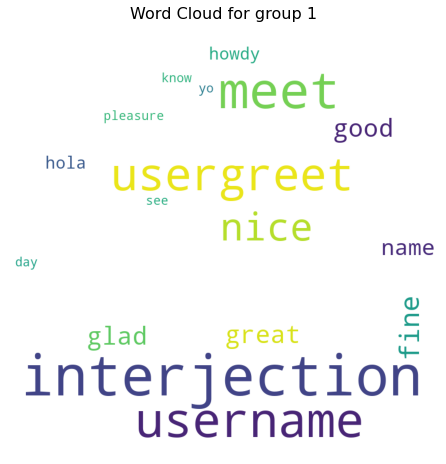
*Figure 3.2 – Data frame head(10)*

Chart, bar chart

Description automatically generated

*Figure 3.3 – Count Plot for Gold label classes*

Word Cloud for Gold label classes.

A picture containing pie chart

Description automatically generated

Text

Description automatically generatedText

Description automatically generated

Text

Description automatically generatedText, application, chat or text message

Description automatically generated

Text

Description automatically generatedGraphical user interface, text, application, chat or text message

Description automatically generated

# 4. Method

A picture containing shape

Description automatically generated

*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

Graphical user interface, text, application, email

Description automatically generatedData Preprocess

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

**MultinomialNBGraphical user interface, text, application, email

Description automatically generated**

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)**

**Graphical user interface, text, application

Description automatically generatedSGDClassifier**

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}**

Graphical user interface, text, application

Description automatically generated**LogisticRegression**

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*

Graphical user interface, text, application

Description automatically generated

Text

Description automatically generatedResults :-

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.

Table

Description automatically generatedFile – *StarterChat\_Evaluation.ipynb*

## 4.6 Feature Importance

A picture containing shape

Description automatically generatedGraphical user interface, text, application, email

Description automatically generatedFeature 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.)

Chart, bar chart

Description automatically generated

*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.

1. **def** preprocess\_cluster\_data(self,text):
2. doc = self.nlp(text)
3. 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']
4. **return** lemma\_list
6. **def** load\_cluster3\_words(self):
7. chat\_data = pd.read\_excel('UserChatData.xlsx', header=0)
8. clust\_3\_df = chat\_data[chat\_data['Group'] == 3]
9. ret\_list = []
11. **for** row **in** clust\_3\_df.itertuples():
12. text\_list = self.preprocess\_cluster\_data(row[1])
13. ret\_list.extend(text\_list)
14. self.cluster3\_words = list(dict.fromkeys(ret\_list))
16. **def** is\_text\_cluster3(self,text):
17. text\_word\_list = self.preprocess\_cluster\_data(text)
18. **if** len(list(set(text\_word\_list) - set(self.cluster3\_words))) > 0:
19. **return** False
20. **else**:
21. **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.

1. **def** remove\_names(self,text):
2. **if** self.caller\_name != None:
3. new\_set = text.lower().replace(self.caller\_name.lower(), 'username')
4. **return** new\_set
5. **else**:
6. **return** text
8. **def** remove\_greetings(self,text):
9. **if** self.welcome\_greeting != None:
10. new\_set = text.lower().replace(self.welcome\_greeting.lower(), 'usergreet')
11. **return** new\_set
12. **else**:
13. **return** text
15. **def** remove\_welcomeINTJ(self,text):
16. **if** self.welcome\_intj != None:
17. new\_set = text.lower().replace(self.welcome\_intj.lower(), 'interjection')
18. **return** new\_set
19. **else**:
20. **return** text
22. **def** remove\_services(self,text):
23. ret\_text = text
24. **for** serv **in** self.checked\_services\_tags:
25. ret\_text = ret\_text.lower().replace(serv, 'gservice')
26. **return** ret\_text

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

1. **def** check\_name(self,doc):
2. self.matches = self.matcher(doc)
4. **if** len(self.matches) != 0:
5. **for** match\_id, start, end **in** self.matches:
6. string\_id = self.nlp.vocab.strings[match\_id]
7. **if** string\_id == "Caller\_Name1":
8. **print**('Pass')
9. self.caller\_name = str(doc[start+1:end]).title()
10. **if** self.caller\_name[len(self.caller\_name) - 1] == ".":
11. self.caller\_name = self.caller\_name[0:len(self.caller\_name) - 1]
12. self.re\_name = self.caller\_name
13. **break**
14. **elif** string\_id == "Caller\_Name2":
15. self.caller\_name = str(doc[start]).title()
16. **if** self.caller\_name[len(self.caller\_name) - 1] == ".":
17. self.caller\_name = self.caller\_name[0:len(self.caller\_name) - 1]
18. self.re\_name = self.caller\_name
19. **break**
20. **else**:
21. **pass**
23. **def** check\_greeting(self,doc):
24. self.matches = self.matcher(doc)
26. **if** len(self.matches) != 0:
27. **for** match\_id, start, end  **in** self.matches:
28. string\_id = self.nlp.vocab.strings[match\_id]
29. **if** string\_id == "Greeting1" **or** string\_id == "Greeting2" **or** string\_id == "Greeting3" **or** string\_id == "Greeting4":
30. self.is\_welcome\_greeting = True
31. self.welcome\_greeting = str(doc[start:end])
32. **break**
33. **else**:
34. **pass**
35. **else**:
36. **pass**
38. **def** check\_intj(self,doc):
39. self.matches = self.matcher(doc)
41. **if** len(self.matches) != 0:
42. **for** match\_id, start, end  **in** self.matches:
43. string\_id = self.nlp.vocab.strings[match\_id]
44. **if** string\_id == "INTJ1":
45. self.welcome\_intj = str(doc[start:end])
46. **else**:
47. **pass**
48. **else**:
49. **pass**
51. **def** check\_services(self,doc):
52. **for** ent **in** doc.ents:
53. **if** ent.label\_ == "GEN\_SERVICE":
54. ind\_val = self.gen\_service\_tags\_df.loc[self.gen\_service\_tags\_df['Tag'] == ent.text.lower()]
55. self.checked\_services\_codes.append(ind\_val['Code'].values[0])
56. self.checked\_services\_tags.append(ent.text.lower())
57. self.is\_newProd\_check = True
58. **else**:
59. **pass**
61. **def** check\_client\_name(self,text):
62. doc = self.nlp(text)
63. name\_list = [t.text **for** t **in** doc **if** t.pos\_ == 'PROPN' **if** **not** t.is\_stop]
65. **if** len(name\_list) != 0:
66. det\_name = self.listToString(name\_list)
67. # print("Name detected:- ", det\_name)
68. self.re\_name = det\_name.title()
69. **else**:
70. # print("Name not detected!")
71. 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:-

1. **def** add\_matchers(self):
2. name\_pattern\_1 = [{'POS': 'AUX', 'OP': '+'},
3. {'POS': 'PROPN', 'OP': '+'}]
4. name\_pattern\_2 = [{'POS': "PROPN"},
5. {'LOWER': 'here'}]
7. # Add General Greeting Matchers
8. greeting\_pattern\_1 = [{'LOWER': 'good', 'OP': '\*'},
9. {'LOWER': 'morning'}]
10. greeting\_pattern\_2 = [{'LOWER': 'good', 'OP': '\*'},
11. {'LOWER': 'evening'}]
12. greeting\_pattern\_3 = [{'LOWER': 'good', 'OP': '\*'},
13. {'LOWER': 'afternoon'}]
14. greeting\_pattern\_4 = [{'LOWER': 'good', 'OP': '\*'},
15. {'LOWER': 'noon'}]
17. # Detect Interjection
18. intj1 = [{'POS': 'INTJ'}]



23. self.matcher.add("Caller\_Name1", None, name\_pattern\_1)
24. self.matcher.add("Caller\_Name2", None, name\_pattern\_2)
25. self.matcher.add("Greeting1", None, greeting\_pattern\_1)
26. self.matcher.add("Greeting2", None, greeting\_pattern\_2)
27. self.matcher.add("Greeting3", None, greeting\_pattern\_3)
28. self.matcher.add("Greeting3", None, greeting\_pattern\_4)
29. 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.

1. **def** add\_service\_entities(self):
2. service\_data = self.gen\_service\_tags\_df[self.gen\_service\_tags\_df['Type'] == 1]
3. service\_list = service\_data['Tag'].tolist()
5. service\_entity = Entity(keywords\_list=service\_list, label='GEN\_SERVICE')
6. self.nlp.add\_pipe(service\_entity, last=True)

Services\_tags.xlsx

|  |  |  |
| --- | --- | --- |
| Tag | Code | Type |
| hair cut | G001 | 1 |
| general hair cut | G001 | 1 |
| cut | G001 | 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 |
| body 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

1. **def** get\_general\_greeting(self):
2. #greet\_data = pd.read\_excel('ReplyChat.xlsx', header=0)
4. reply\_chat = self.reply\_data[self.reply\_data['Type'] == 1]
5. item = random.randint(len(reply\_chat))
6. reply\_chat = reply\_chat['Chat'].tolist()[item]
8. **if** self.caller\_name != None:
9. reply\_chat =  reply\_chat.replace("USER", self.caller\_name)
10. **else**:
11. reply\_chat =  reply\_chat.replace("USER", "")
13. **if** self.time\_greet != None:
14. reply\_chat = reply\_chat.replace("GREETING", self.time\_greet)
15. **else**:
16. reply\_chat = reply\_chat.replace("GREETING", "")
18. **return** reply\_chat
19. **def** get\_checkback\_greeting(self):
20. #greet\_data = pd.read\_excel('ReplyChat.xlsx', header=0)
22. reply\_chat = self.reply\_data[self.reply\_data['Type'] == 2]
23. item = random.randint(len(reply\_chat))
24. reply\_chat = reply\_chat['Chat'].tolist()[item]
26. **if** self.caller\_name != None:
27. reply\_chat =  reply\_chat.replace("USER", self.caller\_name)
28. **else**:
29. reply\_chat =  reply\_chat.replace("USER", "")
31. **if** self.time\_greet != None:
32. reply\_chat = reply\_chat.replace("GREETING", self.time\_greet)
33. **else**:
34. reply\_chat = reply\_chat.replace("GREETING", "")
36. **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.

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

# 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