Statistical NLP Part-2 - Chatbot

- **DOMAIN: Customer support**

- **CONTEXT:**

Great Learning has a an academic support department which receives numerous support requests every day throughout the year. Teams are spread across geographies and try to provide support round the year. Sometimes there are circumstances where due to heavy workload certain request resolutions are delayed, impacting company's business. Some of the requests are very generic where a proper resolution procedure delivered to the user can solve the problem. Company is looking forward to design an automation which can interact with the user, understand the problem and display the resolution procedure [if found as a generic request] or redirect the request to an actual human support executive if the request is complex or not in it's database.

- **DATA DESCRIPTION:**

A sample corpus is attached for your reference. Please enhance/add more data to the corpus using your linguistics skills

- **PROJECT OBJECTIVE:**

Design a python based interactive semi - rule based chatbot which can do the following:

1. Start chat session with greetings and ask what the user is looking for
2. Accept dynamic text based questions from the user. Reply back with relevant answer from the designed corpus
3. End the chat session only if the user requests to end else ask what the user is looking for. Loop continues till the user asks to end it

In [1]:

```
# imports

import os
import json
import random
import warnings
from time import time
from math import floor
from pathlib import Path
from random import shuffle
import pandas as pd, numpy as np
from pprint import pprint
import matplotlib.pyplot as plt
import seaborn as sns
from tqdm import tqdm
from collections import defaultdict
import tensorflow as tf
tqdm.pandas()
warnings.filterwarnings('ignore')
%matplotlib inline
```

In [2]:

```
# reproducibility
seed = 7
random.seed(seed)
```

- **Import the corpus.**

In [3]:

In [3]:

```python
with open('./data/GL Bot.json', 'r') as f:
    intents = json.load(f)
```

In [4]:

```python
pprint(intents, compact=True)
```

```
{'intents': [{'context_set': '',
              'patterns': ['hi', 'how are you', 'is anyone there', 'hello',
                           'whats up', 'hey', 'yo', 'listen', 'please help me',
                           'i am learner from', 'i belong to', 'aiml batch',
                           'aifl batch', 'i am from', 'my pm is', 'blended',
                           'online', 'i am from', 'hey ya',
                           'talking to you for first time'],
              'responses': ['Hello! how can i help you ?'],
              'tag': 'Intro'},
             {'context_set': '',
              'patterns': ['thank you', 'thanks', 'cya', 'see you', 'later',
                           'see you later', 'goodbye', 'i am leaving',
                           'have a Good day', 'you helped me', 'thanks a lot',
                           'thanks a ton', 'you are the best', 'great help',
                           'too good', 'you are a good learning buddy'],
              'responses': ['I hope I was able to assist you, Good Bye'],
              'tag': 'Exit'},
             {'context_set': '',
              'patterns': ['olympus', 'explain me how olympus works',
                           'I am not able to understand olympus',
                           'olympus window not working', 'no access to olympus',
                           'unable to see link in olympus',
                           'no link visible on olympus',
                           'whom to contact for olympus',
                           'lot of problem with olympus',
                           'olypus is not a good tool',
                           'lot of problems with olympus', 'how to use olympus',
                           'teach me olympus'],
              'responses': ['Link: Olympus wiki'],
              'tag': 'Olympus'},
             {'context_set': '',
              'patterns': ['i am not able to understand svm',
                           'explain me how machine learning works',
                           'i am not able to understand naive bayes',
                           'i am not able to understand logistic regression',
                           'i am not able to understand ensemble techb=niques',
                           'i am not able to understand knn',
                           'i am not able to understand knn imputer',
                           'i am not able to understand cross validation',
                           'i am not able to understand boosting',
                           'i am not able to understand random forest',
                           'i am not able to understand ada boosting',
                           'i am not able to understand gradient boosting',
                           'machine learning', 'ML', 'SL',
                           'supervised learning', 'knn', 'logistic regression',
                           'regression', 'classification', 'naive bayes', 'nb',
                           'ensemble techniques', 'bagging', 'boosting',
                           'ada boosting', 'ada', 'gradient boosting',
                           'hyper parameters'],
              'responses': ['Link: Machine Learning wiki '],
              'tag': 'SL'},
             {'context_set': '',
              'patterns': ['what is deep learning',
                           'unable to understand deep learning',
                           'explain me how deep learning works',
                           'i am not able to understand deep learning',
                           'not able to understand neural nets',
                           'very diffult to understand neural nets',
                           'unable to understand neural nets', 'ann',
                           'artificial intelligence',
                           'artificial neural networks', 'weights',
                           'activation function', 'hidden layers', 'softmax',
                           'sigmoid', 'relu', 'otimizer', 'forward propagation',
                           'backward propagation', 'epochs', 'epoch',
                           'what is an epoch', 'adam', 'sgd'],
              'responses': ['Link: Neural Nets wiki'],
              'tag': 'NN'},
             {'context_set': '',
              'patterns': ['what is your name', 'who are you', 'name please',
                           'when are your hours of opertions',
                           'what are your working hours', 'hours of operation',
                           'working hours', 'hours'],
              'responses': ['I am your virtual learning assistant'],
              'tag': 'Bot'},
             {'context_set': '',
              'patterns': ['what the hell', 'bloody stupid bot',
                           'do you think you are very smart', 'screw you',
                           'i hate you', 'you are stupid', 'jerk',
                           'you are a joke', 'useless piece of shit'],
              'responses': ['Please use respectful words'],
              'tag': 'Profane'},
             {'context_set': '',
              'patterns': ['my problem is not solved', 'you did not help me',
                           'not a good solution', 'bad solution',
                           'not good solution', 'no help', 'wasted my time',
                           'useless bot', 'create a ticket'],
              'responses': ['Tarnsferring the request to your PM'],
              'tag': 'Ticket'}]}
```

Some terminology for the corpus:

- **Pairs: Collection of all transactions [Input and Output] to be used for training the chatbot.**
- **Read/patterns: Patterns which are or could be expected as inputs from end-users.**
- **Response: Patterns which are or could be delivered as outputs from the chatbot to end-users.**
- **Regular Expressions: Patterns which can be used to generalise patterns for read and response. This is mainly used to optimise the corpus by making it more generic and avoid generating static read and write responses.**
- **Tag: To group similar text instances and use the same as targeted outputs to train neural networks.**

In [5]:

```python
# enhanced intents corpus
with open('./data/Enhanced_GL_Bot_intents.json', 'r') as f:
    intents = json.load(f)['intents']
```

In [6]:

```python
pprint(intents, compact=True)
```

```
[{'context_set': '',
  'patterns': ['good day', 'hello', 'hey', 'hey i have a doubt', 'hi',
               'i need some help', 'is anyone there', 'is anyone there?',
               'whats up', "a'ight", 'afternoon, boss.',
               'ahoy matey how are you', 'ahoy matey how are you?',
               'aifl batch', 'aiml batch', 'aiml batch 10', 'aiml batch 11',
               'aiml batch 8', 'aiml batch 9', 'aloha.', 'anyone there?',
               'appreciate it', 'are you alright', 'are you having a good day',
               'are you ok?', 'are you okay', 'asante', 'blended', 'bonjour!',
               'brother', 'cheers', 'dear friend', 'do you feel good',
               'do you have a great day', 'evening', 'fist bump', "g'day",
               'good afternoon', 'good afternoon.', 'good day', 'good day.',
               'good evening', 'good morning', 'greetings',
               'greetings, earthling', 'heeyyyyyyyyyy', 'hello', 'hello.',
               'hey', 'hey buddy', 'hey man', 'hey mister', 'hey there',
               'hey ya', 'hey!', 'hey, sonny.', 'hey, you!', 'heyy', 'hi',
               'hi there', 'hi there.', 'high five!', 'hiya',
               'how are things going', 'how are things with you?',
               'how are things with you', 'how are things?', 'how are u',
               'how are you', 'how are you doing',
               'how are you doing this morning',
               'how are you doing today my sweet friend', 'how are you doing?',
               'how are you feeling', 'how are you today', 'how are you?',
               'how do you do.', 'how do you do?', 'how have you been',
               'how is it going', 'how is your day', 'how is your day going',
               'how is your evening', 'how was your day?', 'how you doin',
               "how's it going", "how's it hanging?", "how's it hanging",
               "how's life", "how's life been treating you",
               "how's life been treating you?",
               "how's life treating you friend",
               "how's life treating you friend?", "how's your day going",
               "how've you been?", 'howdy', 'howdy, partner.', 'hullo',
               'i am from', 'i am learner from', 'i belong to',
               "i'm fine and you", 'is anyone there', 'is anyone there?',
               'is everything all right', 'is everything ok?',
               'is everything okay', "it's a beautiful day.",
               "it's good to see you", "it's good to see you.",
               "it's great to see you!", "it's nice to meet you.",
               "it's so nice to hear from you.", 'ladies and gentlemen',
               'listen', 'little wave', 'morning', 'morning, sweetie',
               'my pm is', 'namaste', 'oh hello',
               "oh hello, i didn't see you there before!", 'oi', 'online',
               'pgaiml', 'please help me', 'rise and shine!', 'smile', 'sup',
               'suuuuuuuuuuppp', 'talking to you for first time', 'thank you',
               'thanks', 'thanx', 'thnx', 'watchadoing',
               'what a lovely morning!', 'what about your day',
               "what wonderful weather we're having.", "what's going on?",
               "what's good", "what's new?", "what's up", "what's up man",
               'whats up', 'why hello', 'yeehaw', 'yo', 'yo!', 'yoohoo'],
  'responses': ['Hello there! How can i help?'],
  'tag': 'Intro'},
 {'context_set': '',
  'patterns': ['accept my gratitude', 'all i can say is thanks!',
               'all i can say is thanks', 'appreciate it', 'bye', 'goodbye',
               'how can i show you how grateful i am?',
               'how can i show you how grateful i am', 'i appreciate it',
               'i appreciate your help', "i can't thank you enough",
               'i cannot express my appreciation', 'i humbly thank you',
               'i thank you', "i'll forever be grateful", "i'm thankful",
               'many thanks', 'please accept my deepest thank', 'see you later',
               'thank you', 'thank you for helping me',
               'thank you for the help', 'thank you so much',
               'thank you very much', 'thanks', 'thanks a lot', 'thanks a ton',
               'thanks for everything', 'thanks for the help', "that's helpful",
               'you have my gratitude', 'appreciate it', 'awesome', 'bye bye',
               'c ya', 'cee you later', 'cu', 'cya', 'good afternoon',
               'good by', 'good call', 'good night', 'goodbye', 'gracias',
               'great help', 'have a good day', 'have a nice day',
               'i am leaving', 'later', 'many thanks', 'nice', 'see ya',
               'see ya later', 'see you', 'see you around', 'see you later',
               'thank u', 'thank you', 'thank you for the help', 'thanks',
               'thanks a lot', 'thanks a ton', 'thanks again',
               'thanks for the help', 'thanks!', 'thanx', 'thnx', 'thnx a lot',
               'thnx for answering my questions', 'thnx for the help',
               'thnx for your time', 'thnx!', 'too good',
               'you are a good learning buddy', 'you are the best',
               'you helped me', 'youre a life saver'],
  'responses': ['I hope I was able to assist you, Good Bye'],
  'tag': 'Exit'},
 {'context_set': '',
  'patterns': ['olympus', 'explain olympus', 'explain olympus working',
               'explain working of olympus', 'explain how olympus works',
               'explain me how olympus works',
               'i am not able to understand olympus',
               'i dont understand olympus', 'i do not understand olympus',
               'olympus window not working', 'no access to olympus',
               'unable to see link in olympus', 'no link visible on olympus',
               'whom to contact for olympus', 'lot of problem with olympus',
               'olypus is not a good tool', 'lot of problems with olympus',
               'lots of problems with olympus', 'how to use olympus',
               'teach me olympus', 'bug in olympus', 'dashboard',
               'explain dashboard', 'explain dashboard working',
               'explain working of dashboard', 'explain how dashboard works',
               'explain me how dashboard works',
               'i am not able to understand dashboard',
               'i dont understand dashboard', 'i do not understand dashboard',
               'dashboard window not working', 'no access to dashboard',
               'unable to see link in dashboard',
               'no link visible on dashboard', 'whom to contact for dashboard',
               'lot of problem with dashboard', 'olypus is not a good tool',
               'lot of problems with dashboard',
               'lots of problems with dashboard', 'how to use dashboard',
               'teach me dashboard', 'bug in dashboard', 'learner dashboard',
               'explain learner dashboard', 'explain learner dashboard working',
               'explain working of learner dashboard',
               'explain how learner dashboard works',
               'explain me how learner dashboard works',
               'i am not able to understand learner dashboard',
               'i dont understand learner dashboard',
               'i do not understand learner dashboard',
               'learner dashboard window not working',
               'no access to learner dashboard',
               'unable to see link in learner dashboard',
               'no link visible on learner dashboard',
               'whom to contact for learner dashboard',
               'lot of problem with learner dashboard',
               'lots of problems with learner dashboard',
               'olypus is not a good tool',
               'lot of problems with learner dashboard',
               'how to use learner dashboard', 'teach me learner dashboard',
               'bug in learner dashboard'],
  'responses': ['Link: Olympus wiki'],
  'tag': 'Olympus'},
 {'context_set': '',
  'patterns': ['explain me how machine learning works',
               'i am not able to understand svm',
               'i am not able to understand naive bayes',
               'i am not able to understand logistic regression',
               'i am not able to understand ensemble techb=niques',
               'i am not able to understand knn',
               'i am not able to understand knn imputer',
               'i am not able to understand cross validation',
               'i am not able to understand boosting',
               'i am not able to understand random forest',
               'i am not able to understand ada boosting',
               'i am not able to understand gradient boosting', 'explain svm',
               'explain naive bayes', 'explain logistic regression',
               'explain ensemble techb=niques', 'explain knn',
               'explain knn imputer', 'explain cross validation',
               'explain boosting', 'explain random forest',
               'explain ada boosting', 'explain gradient boosting',
               'understand svm', 'understand naive bayes',
               'understand logistic regression',
               'understand ensemble techb=niques', 'understand knn',
               'understand knn imputer', 'understand cross validation',
               'understand boosting', 'understand random forest',
               'understand ada boosting', 'understand gradient boosting',
               'help svm', 'help naive bayes', 'help logistic regression',
               'help ensemble techb=niques', 'help knn', 'help knn imputer',
               'help cross validation', 'help boosting', 'help random forest',
               'help ada boosting', 'help gradient boosting',
               'machine learning', 'machine learning algo',
               'machine learning algorithm', 'machine learning algorithm wiki',
               'machine learning algorithms wiki',
               'machine learning algorithm help',
               'machine learning algorithms help', 'ml', 'ml algo',
               'ml algorithm', 'ml algorithm wiki', 'ml algorithms wiki',
               'ml algorithm help', 'ml algorithms help', 'sl', 'sl algo',
               'sl algorithm', 'sl algorithm wiki', 'sl algorithms wiki',
               'sl algorithm help', 'sl algorithms help', 'supervised learning',
               'supervised vs unsupervised', 'supervised ml vs unsupervised ml',
               'supervised alogs vs unsupervised algos',
               'supervised learning vs unsupervised learning',
               'supervised learning algo', 'supervised learning algorithm',
               'supervised learning algorithm wiki',
               'supervised learning algorithms wiki',
               'supervised learning algorithm help',
               'supervised learning algorithms help', 'bagging', 'boosting',
               'bagging and boosting', 'bagging & boosting', 'knn', 'lr',
               'linear regression', 'logistic regression',
               'multiple logistic regression', 'regression', 'regression algo',
               'regression task', 'regression alogrithm',
               'regression alogrithms', 'regression ml alogrithms',
               'regression machine learning alogrithms', 'classification',
               'classification algo', 'classification task',
               'classification alogrithm', 'classification alogrithms',
               'classification ml alogrithms',
               'classification machine learning alogrithms', 'naive bayes',
               'nb', 'ensemble techniques', 'bagging', 'boosting',
               'decision tree', 'rf', 'random forest', 'svm',
               'support vector machine', 'kmeans', 'kmeans clustering',
               'clustering', 'unsupervised', 'unsupervised ml',
               'unsupervised learning', 'unsupervised algo',
               'unsupervised alg1orithm', 'unsupervised ml alg1orithm',
               'unsupervised machine learning alg1orithm', 'ada', 'ada boost',
               'xgboost', 'lightgbm', 'catboost', 'xgboost algo',
               'ada boosting', 'adaptive gradient boosting',
               'gradient boosting', 'dimensionalirty reduction', 'svd', 'pca',
               'principal component analysis', 'hyper parameters',
               'hyper parameter turning'],
  'responses': ['Link: Machine Learning wiki '],
  'tag': 'SL'},
 {'context_set': '',
  'patterns': ['what is deep learning', 'dl', 'explain dl',
               'explain how dl works', 'deep learning', 'explain deep learning',
               'explain how deep learning works',
               'unable to understand deep learning',
               'explain me how deep learning works',
               'i am not able to understand deep learning',
               'not able to understand neural nets',
               'very diffult to understand neural nets',
               'unable to understand neural nets', 'ann',
               'artificial intelligence', 'artificial neural networks',
               'neural net', 'feed forward networks', 'forward prop',
               'forward propagation', 'backprop', 'backward propagation',
               'explain backward propagation', 'help backward propagation',
               'weights', 'activation function', 'hidden layers', 'softmax',
               'sigmoid', 'tanh', 'relu', 'adaptive gradient descent',
               'rmsprop', 'gradient descent', 'gradient descent with momentum',
               'stochastic gradient descent', 'otimizer', 'forward propagation',
               'backward propagation', 'epochs', 'epoch', 'train',
               'how to train', 'how to test', 'how to evaluate',
               'how to train nn', 'how to test nn', 'how to evaluate nn',
               'how to train neural net', 'how to evaluate neural net',
               'how to test neural net', 'training', 'what is an epoch', 'adam',
               'sgd', 'explainability', 'rnn', 'recurrant neural net',
               'recurrant neural network', 'lstm', 'long short term memory',
               'long short term memory net', 'long short term memory network',
               'cnn', 'convolutional neural network', 'cv', 'computer vision',
               'face recogntion', 'face detection', 'text to speech',
               'gpu training', 'ner', 'named entity recognition',
               'speech recogntion', 'nlp', 'natural language processing',
               'bert', 'albert', 'xlnet', 'imagenet', 'image net', 'yolo',
               'yolov2', 'yolov3', 'attention', 'all you need is attention',
               'transformer models', 'gan models', 'gan',
               'general adverserial nets', 'general adverserial networks'],
  'responses': ['Link: Neural Nets wiki'],
  'tag': 'NN'},
 {'context_set': '',
  'patterns': ['what is your name', 'whats your name', 'what should i call you',
               'who are you', 'what are you', 'why are you', 'who u', 'who r u',
               'name', 'are you a person', 'are you a bot',
               'are you a bot or person', 'robot', 'bot', 'name please',
               'when are your hours of opertions',
               'what are your working hours', 'hours of operation',
               'working hours', 'hours'],
  'responses': ['I am your virtual learning assistant'],
  'tag': 'Bot'},
 {'context_set': '',
  'patterns': ['what the hell', 'bloody stupid bot',
               'do you think you are very smart', 'screw you', 'i hate you',
               'you are stupid', 'shit', 'piss', 'jerk', 'you are a joke',
               'useless piece of shit'],
  'responses': ['Kindly use respectful words'],
  'tag': 'Profane'},
 {'context_set': '',
  'patterns': ['my problem is not solved', 'you did not help me',
               'not a good solution', 'bad solution', 'not good solution',
               'no help', 'wasted my time', 'such a waster', 'not resolved',
               'you did not resolve', 'you did not resolve my problem',
               'unsatisfactory', 'can you even understand me',
               'unsatisfactory solution', 'useless bot', 'create a ticket',
               'not satisfied', 'more help required', 'connect to human',
               'connect to person', 'connect me to human',
               'connect me to an actual human',
               'connect me to an actual person', 'talk to human',
               'talk to customer suppport', 'talk to customer suppport exec',
               'talk to customer suppport executive'],
  'responses': ['Tarnsferring the request to your PM'],
  'tag': 'Ticket'}]
```

- **Data Preprocessing**

In [7]:

```
intents[0].keys()
```

Out[7]:

```
dict_keys(['context_set', 'patterns', 'responses', 'tag'])
```

In [8]:

```
# create dataset
intents_df = pd.DataFrame()

for intent in intents:
    print(intent['tag'])
    patterns = intent['patterns']
    response = intent['responses'][0]
    intent_df = pd.DataFrame({'pattern': patterns,
                              'reponse':[response]*(len(patterns)),
                              'intent': [intent['tag']]*(len(patterns))})
    intents_df = pd.concat([intents_df, intent_df])
```

```
Intro
Exit
Olympus
SL
NN
Bot
Profane
Ticket
```

In [9]:

```
intents_df.sample(25)
```

Out[9]:

	pattern	reponse	intent
114	unsupervised ml	Link: Machine Learning wiki	SL
132	thanks	Hello there! How can i help?	Intro
54	see you around	I hope I was able to assist you, Good Bye	Exit
112	clustering	Link: Machine Learning wiki	SL
6	shit	Kindly use respectful words	Profane
4	i hate you	Kindly use respectful words	Profane

	pattern	reponse	intent
4	I hate you	Kindly use respectful words	Profane
26	understand ensemble techb=niques	Link: Machine Learning wiki	SL
35	stochastic gradient descent	Link: Neural Nets wiki	NN
115	unsupervised learning	Link: Machine Learning wiki	SL
1	all i can say is thanks!	I hope I was able to assist you, Good Bye	Exit
14	name please	I am your virtual learning assistant	Bot
22	thank you so much	I hope I was able to assist you, Good Bye	Exit
55	whom to contact for learner dashboard	Link: Olympus wiki	Olympus
28	sigmoid	Link: Neural Nets wiki	NN
35	cee you later	I hope I was able to assist you, Good Bye	Exit
67	how are u	Hello there! How can i help?	Intro
40	good day.	Hello there! How can i help?	Intro
64	convolutional neural network	Link: Neural Nets wiki	NN
37	forward propagation	Link: Neural Nets wiki	NN
39	how to use dashboard	Link: Olympus wiki	Olympus
29	that's helpful	I hope I was able to assist you, Good Bye	Exit
100	naive bayes	Link: Machine Learning wiki	SL
27	blended	Hello there! How can i help?	Intro
47	how to evaluate nn	Link: Neural Nets wiki	NN
98	classification ml alogrithms	Link: Machine Learning wiki	SL

In [10]:

```
# !pip install spacy
# !python -m spacy downlaod en_core_web_sm

import spacy
import nltk

nltk.download('punkt')
nltk.download('stopwords')
# Initialize spacy 'en_core_web_sm' model
nlp = spacy.load('en_core_web_sm', disable=['parser'])
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\surya\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\surya\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

In [11]:

```
# utility functions for text preprocesing
import re
import string
import unicodedata
import contractions
from bs4 import BeautifulSoup
from nltk.tokenize import word_tokenize, sent_tokenize
from nltk.stem.snowball import SnowballStemmer

stemmer = SnowballStemmer('english')
stop_words = set(nltk.corpus.stopwords.words('english'))

def replace_accented_chars(review_text):
    '''normalizes and replaces accented characters'''
    unaccented_text = unicodedata.normalize('NFKD', review_text).encode('ascii', 'ignore
').decode('utf-8', 'ignore')
    return unaccented_text
```

```python
def strip_html_tags(review_text):
    '''strips html tags like <h4> ..etc'''
    soup = BeautifulSoup(review_text, "html.parser")
    [s.extract() for s in soup(['iframe', 'script'])]
    stripped_text = soup.get_text()
    stripped_text = re.sub(r'[\r|\n|\r\n]+', '\n', stripped_text)
    return stripped_text


def expand_contractions(review_text):
    review_text = contractions.fix(review_text)
    return review_text


def remove_special_characters(review_text):
    '''
    Remove special characters but preserve digits and excalamation marks
    as they indicate emotionally charged review '''
    review_text = re.sub(r"[^A-Za-z0-9!?\'\`]", " ", review_text)
    return review_text


def strip_stops(text, is_lower_case=False, stop_words=stop_words):
    '''strip stopwrds'''
    tokens = word_tokenize(text)
    tokens = [token.strip() for token in tokens]
    if is_lower_case:
        filtered_tokens = [token for token in tokens if token not in stop_words]
    else:
        filtered_tokens = [token for token in tokens if token.lower() not in stop_words]
    filtered_text = ' '.join(filtered_tokens)
    return filtered_text


def tokenize(text):
    '''tokenize using spaCy'''
    doc = nlp(text)
    return " ".join([t.text for t in doc])

# Stemming/Lemmatization
def lemmatize(text):
    '''lemmatize using spaCy'''
    doc = nlp(text)
    return " ".join([t.lemma_ for t in doc])


def snowball_stem(text, stemmer=stemmer):
    '''stemming using snowball stemmer'''
    words = text.split()
    stemmed_words = [stemmer.stem(word) for word in words]
    review_text = " ".join(stemmed_words)
    return review_text
```

In [12]:

```python
def preprocess_text(text: str, lower=True,
                    strip_stops=False) -> str:
    text = replace_accented_chars(text)
    text = strip_html_tags(text)
    text = expand_contractions(text)
    text = remove_special_characters(text)
    if lower:
        text = text.lower()
    if strip_stops:
        text = strip_stops(text)
    text = tokenize(text)
    text = lemmatize(text)
    return str(text.strip())
```

```
sentence = "<p>How are you doing? 😊</p>"
cleaned = preprocess_text(sentence)
cleaned
```

Out[12]:

```
'how be you do ?'
```

In [13]:

```
def preprocess(row):
    text = row.pattern
    if isinstance(text, str):
        text = preprocess_text(text)
    else:
        text = np.nan
    row['cleaned_pattern'] = text
    return row
```

In [14]:

```
intents_df = intents_df.progress_apply(preprocess, axis=1)
```

```
100%|████████████████████████████████████████████████████████████████| 574
/574 [00:03<00:00, 159.00it/s]
```

In [15]:

```
intents_df.isna().any()
```

Out[15]:

```
pattern            False
reponse            False
intent             False
cleaned_pattern    False
dtype: bool
```

In [16]:

```
# encode the target column
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
intents_df['labels' ] = le.fit_transform(intents_df['intent'])
```

In [17]:

```
intents_df.sample(25)
```

Out[17]:

	pattern	reponse	intent	cleaned_pattern	labels
47	machine learning algorithm	Link: Machine Learning wiki	SL	machine learn algorithm	6
0	my problem is not solved	Tarnsferring the request to your PM	Ticket	my problem be not solve	7
144	whats up	Hello there! How can i help?	Intro	what be up	2
31	appreciate it	I hope I was able to assist you, Good Bye	Exit	appreciate it	1
104	boosting	Link: Machine Learning wiki	SL	boost	6
63	how are things going	Hello there! How can i help?	Intro	how be thing go	2
74	supervised learning algorithms wiki	Link: Machine Learning wiki	SL	supervise learn algorithms wiki	6
2	all i can say is thanks	I hope I was able to assist you, Good Bye	Exit	all I can say be thank	1
11	are you a bot or person	I am your virtual learning assistant	Bot	be you a bot or person	0

	pattern	reponse	intent	cleaned_pattern	labels
11	are you a bot or person	I am your virtual learning assistant	Bot	be you a bot or person	0
46	how to test nn	Link: Neural Nets wiki	NN	how to test nn	3
15	explain ensemble techb=niques	Link: Machine Learning wiki	SL	explain ensemble techb nique	6
28	sigmoid	Link: Neural Nets wiki	NN	sigmoid	3
80	bagging & boosting	Link: Machine Learning wiki	SL	bag boost	6
6	i am not able to understand olympus	Link: Olympus wiki	Olympus	I be not able to understand olympus	4
89	how's life been treating you	Hello there! How can i help?	Intro	how be life be treat you	2
122	xgboost	Link: Machine Learning wiki	SL	xgboost	6
74	how are you today	Hello there! How can i help?	Intro	how be you today	2
146	yeehaw	Hello there! How can i help?	Intro	yeehaw	2
64	thanks!	I hope I was able to assist you, Good Bye	Exit	thank !	1
107	random forest	Link: Machine Learning wiki	SL	random forest	6
44	great help	I hope I was able to assist you, Good Bye	Exit	great help	1
3	appreciate it	I hope I was able to assist you, Good Bye	Exit	appreciate it	1
76	bert	Link: Neural Nets wiki	NN	bert	3
29	understand cross validation	Link: Machine Learning wiki	SL	understand cross validation	6
70	thnx for your time	I hope I was able to assist you, Good Bye	Exit	thnx for your time	1

In [18]:

```
vocab = []
for text in intents_df.cleaned_pattern.tolist():
    vocab.extend(tokenize(text).split())
print(len(vocab))
```

```
1817
```

In [19]:

```
num_classes = len(le.classes_)
```

In [20]:

```
le.classes_
```

Out[20]:

```
array(['Bot', 'Exit', 'Intro', 'NN', 'Olympus', 'Profane', 'SL', 'Ticket'],
      dtype=object)
```

In [21]:

```
intent_to_idx = {i: j for i, j in zip(le.classes_, range(num_classes))}
idx_to_intent = {v: i for i, v in intent_to_idx.items()} # inverse lookup
intent_to_idx
```

Out[21]:

```
{'Bot': 0,
 'Exit': 1,
 'Intro': 2,
 'NN': 3,
 'Olympus': 4,
 'Profane': 5,
 'SL': 6,
 'Ticket': 7}
```

In [22]:

```
idx_to_intent
```

Out[22]:

```
{0: 'Bot',
 1: 'Exit',
 2: 'Intro',
 3: 'NN',
 4: 'Olympus',
 5: 'Profane',
 6: 'SL',
 7: 'Ticket'}
```

In [23]:

```
dataset = []
for text, intent in zip(intents_df.cleaned_pattern, intents_df.intent):
    bow = []
    text_tokens = text.split()
    for w in vocab:
        if w in text_tokens:
            bow.append(1)
        else:
            bow.append(0)

    one_hot = list([0]*(num_classes))
    one_hot[intent_to_idx[intent]] = 1  # one hot (1) at the specified index

    dataset.append([bow, one_hot])
```

In [24]:

```
SHUFFLE = True
if SHUFFLE:
    # shuffle our features and turn into np.array
    shuffle(dataset)

dataset = np.array(dataset)
dataset.shape
```

Out[24]:

```
(574, 2)
```

In [25]:

```
X_train = dataset[:,0]
y_train = dataset[:,1]
X_train.shape, y_train.shape
```

Out[25]:

```
((574,), (574,))
```

In [26]:

```
# pprint(X_train[0], compact=True)
y_train[0]
```

Out[26]:

```
[0, 0, 0, 0, 0, 0, 0, 1]
```

- **Design a neural network to classify the queries with INTENTS as target outputs**

In [27]:

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.python.keras.callbacks import LambdaCallback, EarlyStopping
from tensorflow.keras.layers import *
from tensorflow.keras.preprocessing import text
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.optimizers import SGD
```

In [28]:

```
simple_log = LambdaCallback(
    on_epoch_end = lambda e, l: print(f" ~| Epoch: {e+1} | Validation Loss: {l['val_loss
']:.5f}", end =" >|> \n" ))

early_stop = EarlyStopping(monitor='val_loss',
                              min_delta=0,
                              patience=1,
                              verbose=0,
                              restore_best_weights=True)
sns.set()
def plot_learning_curve(hist):
    plt.figure(figsize=(5,5))
    train = hist.history['loss']
    val = hist.history['val_loss']
    epochs_run = range(1,len(train) + 1)
    sns.lineplot(epochs_run, train, marker = 'o', color = 'coral', label = 'Training Los
s')
    sns.lineplot(epochs_run, val,  marker = '>', color = 'green', label = 'Validation Lo
ss')
    plt.title("Loss vs. Epochs", fontsize = 20)
    plt.legend()
    plt.show()
```

In [29]:

```
model = Sequential([
    Dense(128, input_shape=(len(X_train[0]), ), activation='relu'),
    Dropout(0.5),
    Dense(64, activation='relu'),
    Dropout(0.5),
    Dense(num_classes, activation='softmax')
])

# Compile model. Stochastic gradient descent with Nesterov accelerated gradient gives goo
d results for this model
sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense (Dense)                (None, 128)               232704    
_________________________________________________________________
dropout (Dropout)            (None, 128)               0         
_________________________________________________________________
dense_1 (Dense)              (None, 64)                8256      
_________________________________________________________________
dropout_1 (Dropout)          (None, 64)                0         
_________________________________________________________________
dense_2 (Dense)              (None, 8)                 520       
=================================================================
Total params: 241,480
Trainable params: 241,480
Non-trainable params: 0
_________________________________________________________________
```

In [30]:

```
X_train = np.array([np.array(i) for i in X_train])
y_train = np.array([np.array(i) for i in y_train])
```

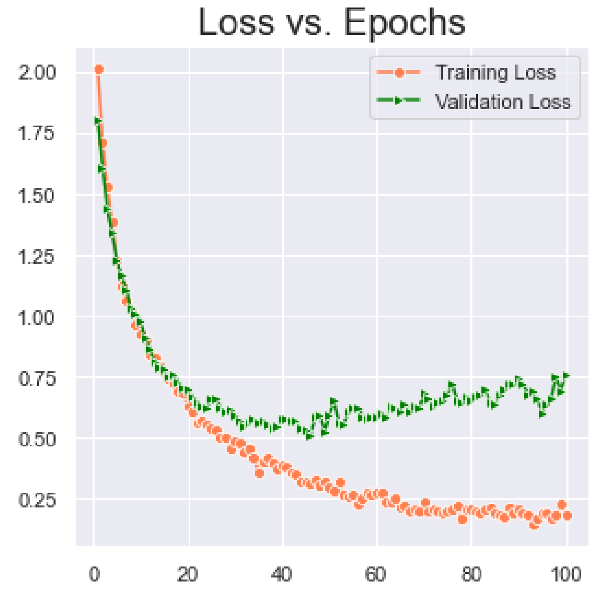
In [31]:

```
epochs = 100

h = model.fit(
    X_train, y_train,
    validation_split = 0.2,
    epochs = epochs,
    callbacks = [simple_log],
    verbose = False)

print("\nDone.")
```

```
 ~| Epoch: 1 | Validation Loss: 1.80175 >|>
 ~| Epoch: 2 | Validation Loss: 1.60664 >|>
 ~| Epoch: 3 | Validation Loss: 1.44113 >|>
 ~| Epoch: 4 | Validation Loss: 1.33685 >|>
 ~| Epoch: 5 | Validation Loss: 1.22956 >|>
 ~| Epoch: 6 | Validation Loss: 1.16458 >|>
 ~| Epoch: 7 | Validation Loss: 1.10524 >|>
 ~| Epoch: 8 | Validation Loss: 1.02818 >|>
 ~| Epoch: 9 | Validation Loss: 1.00700 >|>
 ~| Epoch: 10 | Validation Loss: 0.97976 >|>
 ~| Epoch: 11 | Validation Loss: 0.91114 >|>
 ~| Epoch: 12 | Validation Loss: 0.86507 >|>
 ~| Epoch: 13 | Validation Loss: 0.80688 >|>
 ~| Epoch: 14 | Validation Loss: 0.79017 >|>
 ~| Epoch: 15 | Validation Loss: 0.77656 >|>
 ~| Epoch: 16 | Validation Loss: 0.75201 >|>
 ~| Epoch: 17 | Validation Loss: 0.75824 >|>
 ~| Epoch: 18 | Validation Loss: 0.73035 >|>
 ~| Epoch: 19 | Validation Loss: 0.70634 >|>
 ~| Epoch: 20 | Validation Loss: 0.69931 >|>
 ~| Epoch: 21 | Validation Loss: 0.66998 >|>
 ~| Epoch: 22 | Validation Loss: 0.64194 >|>
 ~| Epoch: 23 | Validation Loss: 0.62776 >|>
 ~| Epoch: 24 | Validation Loss: 0.61994 >|>
 ~| Epoch: 25 | Validation Loss: 0.65826 >|>
 ~| Epoch: 26 | Validation Loss: 0.66154 >|>
 ~| Epoch: 27 | Validation Loss: 0.62058 >|>
 ~| Epoch: 28 | Validation Loss: 0.60942 >|>
 ~| Epoch: 29 | Validation Loss: 0.61046 >|>
 ~| Epoch: 30 | Validation Loss: 0.58914 >|>
 ~| Epoch: 31 | Validation Loss: 0.56548 >|>
 ~| Epoch: 32 | Validation Loss: 0.54199 >|>
 ~| Epoch: 33 | Validation Loss: 0.55655 >|>
 ~| Epoch: 34 | Validation Loss: 0.57247 >|>
 ~| Epoch: 35 | Validation Loss: 0.56182 >|>
 ~| Epoch: 36 | Validation Loss: 0.56451 >|>
 ~| Epoch: 37 | Validation Loss: 0.55642 >|>
 ~| Epoch: 38 | Validation Loss: 0.53718 >|>
 ~| Epoch: 39 | Validation Loss: 0.54405 >|>
 ~| Epoch: 40 | Validation Loss: 0.57291 >|>
 ~| Epoch: 41 | Validation Loss: 0.57314 >|>
 ~| Epoch: 42 | Validation Loss: 0.57016 >|>
 ~| Epoch: 43 | Validation Loss: 0.56434 >|>
 ~| Epoch: 44 | Validation Loss: 0.53544 >|>
 ~| Epoch: 45 | Validation Loss: 0.52745 >|>
 ~| Epoch: 46 | Validation Loss: 0.50759 >|>
 ~| Epoch: 47 | Validation Loss: 0.58344 >|>
 ~| Epoch: 48 | Validation Loss: 0.59066 >|>
 ~| Epoch: 49 | Validation Loss: 0.52587 >|>
 ~| Epoch: 50 | Validation Loss: 0.59051 >|>
 ~| Epoch: 51 | Validation Loss: 0.65048 >|>
 ~| Epoch: 52 | Validation Loss: 0.55783 >|>
 ~| Epoch: 53 | Validation Loss: 0.55178 >|>
 ~| Epoch: 54 | Validation Loss: 0.60211 >|>
 ~| Epoch: 55 | Validation Loss: 0.62406 >|>
 ~| Epoch: 56 | Validation Loss: 0.62321 >|>
 ~| Epoch: 57 | Validation Loss: 0.58671 >|>
 ~| Epoch: 58 | Validation Loss: 0.57220 >|>
 ~| Epoch: 59 | Validation Loss: 0.58632 >|>
 ~| Epoch: 60 | Validation Loss: 0.58517 >|>
 ~| Epoch: 60 | Validation Loss: 0.56517 >|>
 ~| Epoch: 61 | Validation Loss: 0.59908 >|>
 ~| Epoch: 62 | Validation Loss: 0.58309 >|>
 ~| Epoch: 63 | Validation Loss: 0.63043 >|>
 ~| Epoch: 64 | Validation Loss: 0.62200 >|>
 ~| Epoch: 65 | Validation Loss: 0.60824 >|>
 ~| Epoch: 66 | Validation Loss: 0.63717 >|>
 ~| Epoch: 67 | Validation Loss: 0.60373 >|>
 ~| Epoch: 68 | Validation Loss: 0.62840 >|>
 ~| Epoch: 69 | Validation Loss: 0.62245 >|>
 ~| Epoch: 70 | Validation Loss: 0.67838 >|>
 ~| Epoch: 71 | Validation Loss: 0.65584 >|>
 ~| Epoch: 72 | Validation Loss: 0.62673 >|>
 ~| Epoch: 73 | Validation Loss: 0.64272 >|>
 ~| Epoch: 74 | Validation Loss: 0.65160 >|>
 ~| Epoch: 75 | Validation Loss: 0.67540 >|>
 ~| Epoch: 76 | Validation Loss: 0.72006 >|>
 ~| Epoch: 77 | Validation Loss: 0.65085 >|>
 ~| Epoch: 78 | Validation Loss: 0.64394 >|>
 ~| Epoch: 79 | Validation Loss: 0.66466 >|>
 ~| Epoch: 80 | Validation Loss: 0.64973 >|>
 ~| Epoch: 81 | Validation Loss: 0.66823 >|>
 ~| Epoch: 82 | Validation Loss: 0.67119 >|>
 ~| Epoch: 83 | Validation Loss: 0.69649 >|>
 ~| Epoch: 84 | Validation Loss: 0.64133 >|>
 ~| Epoch: 85 | Validation Loss: 0.63242 >|>
 ~| Epoch: 86 | Validation Loss: 0.67542 >|>
 ~| Epoch: 87 | Validation Loss: 0.69314 >|>
 ~| Epoch: 88 | Validation Loss: 0.72292 >|>
 ~| Epoch: 89 | Validation Loss: 0.72033 >|>
 ~| Epoch: 90 | Validation Loss: 0.74421 >|>
 ~| Epoch: 91 | Validation Loss: 0.71598 >|>
 ~| Epoch: 92 | Validation Loss: 0.67650 >|>
 ~| Epoch: 93 | Validation Loss: 0.68821 >|>
 ~| Epoch: 94 | Validation Loss: 0.65890 >|>
 ~| Epoch: 95 | Validation Loss: 0.59797 >|>
 ~| Epoch: 96 | Validation Loss: 0.63804 >|>
 ~| Epoch: 97 | Validation Loss: 0.66182 >|>
 ~| Epoch: 98 | Validation Loss: 0.75123 >|>
 ~| Epoch: 99 | Validation Loss: 0.69061 >|>
 ~| Epoch: 100 | Validation Loss: 0.75738 >|>

Done.
```

In [32]:

```
plot_learning_curve(h)
```



- **Design a chat utility as a function to interact with the user till the user calls a “quit”**

- **If the user does not understand or finds the bot’s answer irrelevant, the user calls a “*” asking the bot to re-evaluate what the user has asked**

In [33]:

```
from collections import defaultdict

class IntentClassifier():

    def __init__(self, intents, vocab, idx_to_intent, model):
        self.intents = intents
        self.vocab = vocab
        self.model = model
        self.idx_to_intent = idx_to_intent

        responses = defaultdict()
        for intent in intents:
            responses[intent['tag']] = intent['responses']
        self.responses = dict(responses)

        intents_lookup = defaultdict()
        for intent in intents:
            tag = intent['tag']
            for text in intent['patterns']:
                intents_lookup[text] = tag
        self.intents_lookup = dict(intents_lookup)

    def search_intent(self, text):
        return self.intents_lookup.get(text.lower().strip(), 'na')

    def get_bow(self, text):
        text = preprocess_text(text)
        text_tokens = text.split()
        # bag of words - matrix of N words, vocabulary matrix
        bag = np.array([0]*len(self.vocab))
        for tok in text_tokens:
            for idx, word in enumerate(self.vocab):
                if word == tok:
                    # assign 1 if current word is in the vocabulary position
                    bag[idx] = 1
        return(np.array(bag))

    def predict_intent(self, text):
        # filter out predictions below a threshold
        bow = self.get_bow(text)
        pred = self.model.predict(np.array([bow]))[0]
        results = [[intent, prob] for intent, prob in enumerate(pred) if prob>0.25]
        # sort by strength of probability
        results.sort(key=lambda x: x[1], reverse=True)
        return_list = []
        for r in results:
            return_list.append({"intent": self.idx_to_intent[r[0]], "probability": str(r
[1])})
        return return_list

    def classify(self, text):
        intent = self.search_intent(text)
        if intent != 'na':
            return intent
        intents = self.predict_intent(text)
        if len(intents):
            return intents[0]['intent']

    def generate_response(self, text):
        intent = self.classify(text)
        default_msg = "I am sorry! I don't understand you. Can you rephrase your query?"
        response  = self.responses.get(intent, default_msg)
        return response
```

In [34]:

```
ic = IntentClassifier(intents, vocab, idx_to_intent, model)
ic = IntentClassifier(intents, vocab, idx_to_intent, model)
ic
```

Out[34]:

```
<__main__.IntentClassifier at 0x23457425520>
```

In [35]:

```
ic.classify('Hey there!')
```

Out[35]:

```
'Intro'
```

In [36]:

```
ic.generate_response('Hey there!')
```

Out[36]:

```
['Hello there! How can i help?']
```

In [37]:

```
ic.classify('please explain Deep Learning')  # out of sample text
```

Out[37]:

```
'NN'
```

In [38]:

```
ic.generate_response('please explain Deep Learning')  # out of sample text
```

Out[38]:

```
['Link: Neural Nets wiki']
```

In [39]:

```
# Chatbot Utility
def chat(ic):
    print("Hi there! I am Groot! (type your query or 'quit' to exit the chat)")
    print("If the response to the query doesn't make sense, type '*'")
    default_msg = "I am sorry! I don't understand you. Can you rephrase your query?"

    prev_query = 'na'
    user = input("Hi! WHat's your name?")
    errors = 0
    # chat loop
    while True:
        query = str(input())
        print(f'{user}: {query}')
        prev_query = query
        if query.lower().strip() == 'quit':
            break
        if query.lower().strip() == '*':
            errors += 1
            query = prev_query
        if errors == 2:
            print(f'Groot: {default_msg}')
            errors = 0
            continue
        if not isinstance(query, str):
            print(f'Groot: {default_msg}')
            continue
        response = ic.generate_response(query)
        print(f'Groot: {response[0]}')
        print()
```

In [40]:

```
chat(ic)
```

```
Hi there! I am Groot! (type your query or 'quit' to exit the chat)
If the response to the query doesn't make sense, type '*'

Pradeep: Anyone there?
Groot: Hello there! How can i help?


Pradeep: Who are you
Groot: I am your virtual learning assistant


Pradeep: I have a problem with my olympus dashboard
Groot: Link: Olympus wiki


Pradeep: Can you connect me to a human
Groot: Tarnsferring the request to your PM


Pradeep: Can you explain Machine Learning? □
Groot: I hope I was able to assist you, Good Bye


Pradeep: *
Groot: Link: Neural Nets wiki


Pradeep: *
Groot: I am sorry! I don't understand you. Can you rephrase your query?

Pradeep: explain ML ☹
Groot: Link: Machine Learning wiki


Pradeep: Can you explain Naive Bayes Classifier?
Groot: I hope I was able to assist you, Good Bye


Pradeep: *
Groot: Link: Neural Nets wiki


Pradeep: *
Groot: I am sorry! I don't understand you. Can you rephrase your query?

Pradeep: explain naive bayes classifier
Groot: Link: Machine Learning wiki


Pradeep: explain deep learning architecture
Groot: Link: Neural Nets wiki


Pradeep: ok thank you very much groot
Groot: I hope I was able to assist you, Good Bye


Pradeep: quit
```

- **This chatbot can be improved by further training as our intent classifier was trained on a very limited dataset although it was extended, we can train it using a dataset of converstations between users and customer support execs. For e.g, https://www.kaggle.com/thoughtvector/customer-support-on-twitter/data can beused for training a chatbot. This dataset was created by collecting publicly available conversations between customer supports and users on Twitte**
- **We can also use a better profanity filter by using packages like profanity-filter or building a ML/DL model to detect it ourselves so that the responses are filterd out properly**
- **We can also add an NER model to understand the language more and train better models to make our chatbot smarter.**
- **Also, we could use pre-built frameworks like dialogflow, rasa , etc. to utilize the**

- **Also, we could use pre-built frameworks like dialogflow, rasa ...etc., to utilize the language models that are in-built instead of doing it from scratch and customize the chatbot as per the requirements of the clients/users.**