Chatbot

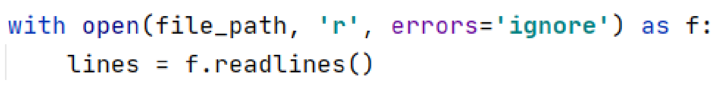
1. Train on multiple datasets

- **Dataset : Twitter chat log (courtesy of Marsan Ma) -Chat.txt https://github.com/Marsan-Ma/chat_corpus**

Description : Chat file contains data with question and answer pairs, where odd lines are questions, even lines are answers.

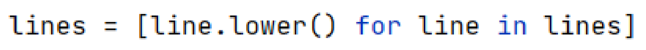
Data Preprocessing :

1. **Load the data :**



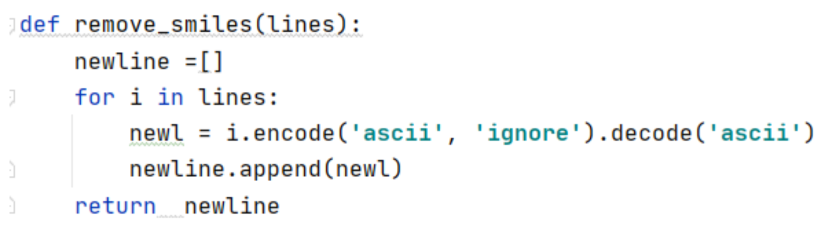
```
with open(file_path, 'r', errors='ignore') as f:
    lines = f.readlines()
```

2. **Convert into lower case:**



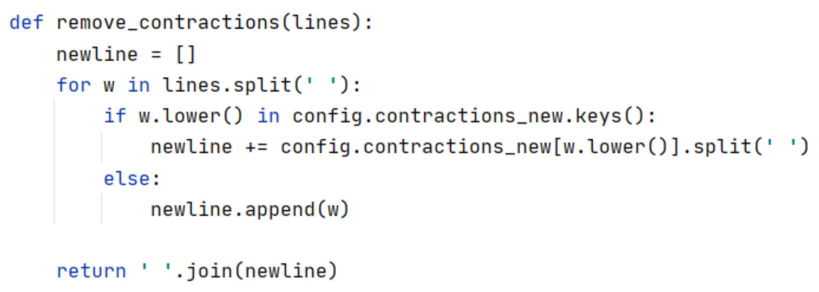
```
lines = [line.lower() for line in lines]
```

3. **Remove emoji :**



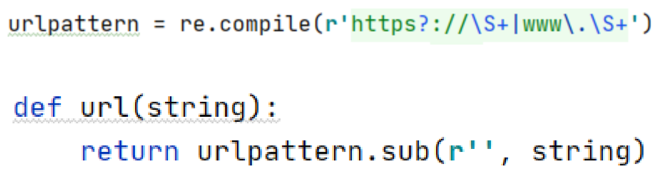
```
def remove_smiles(lines):
    newline =[]
    for i in lines:
        newl = i.encode('ascii', 'ignore').decode('ascii')
        newline.append(newl)
    return  newline
```

4. **Remove contractions :**



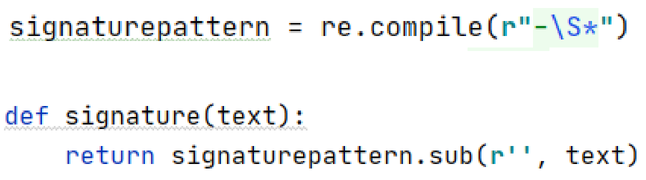
```
def remove_contractions(lines):
    newline = []
    for w in lines.split(' '):
        if w.lower() in config.contractions_new.keys():
            newline += config.contractions_new[w.lower()].split(' ')
        else:
            newline.append(w)

    return ' '.join(newline)
```

5. **Remove URL :**



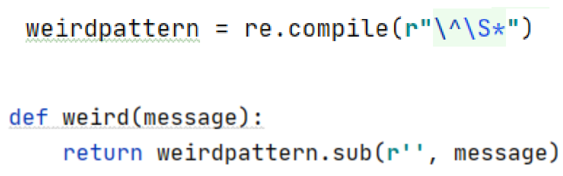
```
urlpattern = re.compile(r'https?://\S+|www\.\S+')

def url(string):
    return urlpattern.sub(r'', string)
```

6. **Remove Signature :**



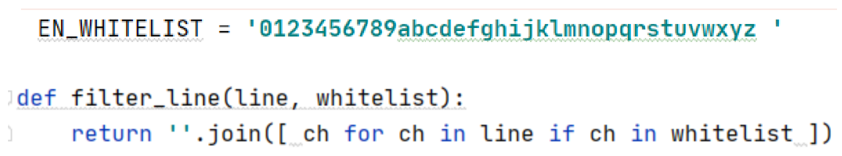
```
signaturepattern = re.compile(r"-\S*")

def signature(text):
    return signaturepattern.sub(r'', text)
```

7. **Remove weird things :**



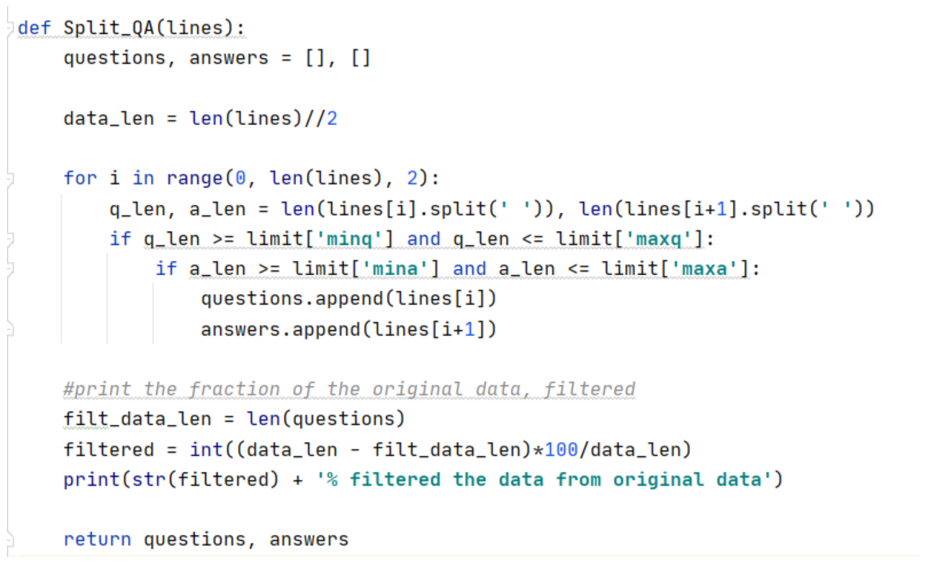
```
weirdpattern = re.compile(r"\^\S*")

def weird(message):
    return weirdpattern.sub(r'', message)
```

8. **Remove unnecessary characters :**



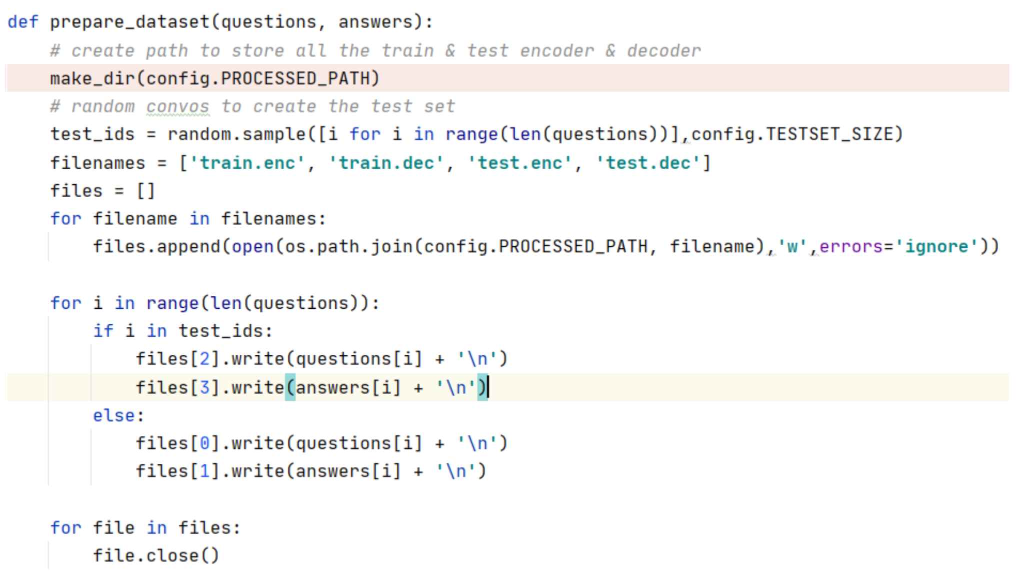
```
 EN_WHITELIST = '0123456789abcdefghijklmnopqrstuvwxyz '

def filter_line(line, whitelist):
    return ''.join([ ch for ch in line if ch in whitelist ])
```

9. **Filter and split data into questions and answers**



```
def Split_QA(lines):
    questions, answers = [], []

    data_len = len(lines)//2

    for i in range(0, len(lines), 2):
        q_len, a_len = len(lines[i].split(' ')), len(lines[i+1].split(' '))
        if q_len >= limit['minq'] and q_len <= limit['maxq']:
            if a_len >= limit['mina'] and a_len <= limit['maxa']:
                questions.append(lines[i])
                answers.append(lines[i+1])

    #print the fraction of the original data, filtered
    filt_data_len = len(questions)
    filtered = int((data_len - filt_data_len)*100/data_len)
    print(str(filtered) + '% filtered the data from original data')

    return questions, answers
```

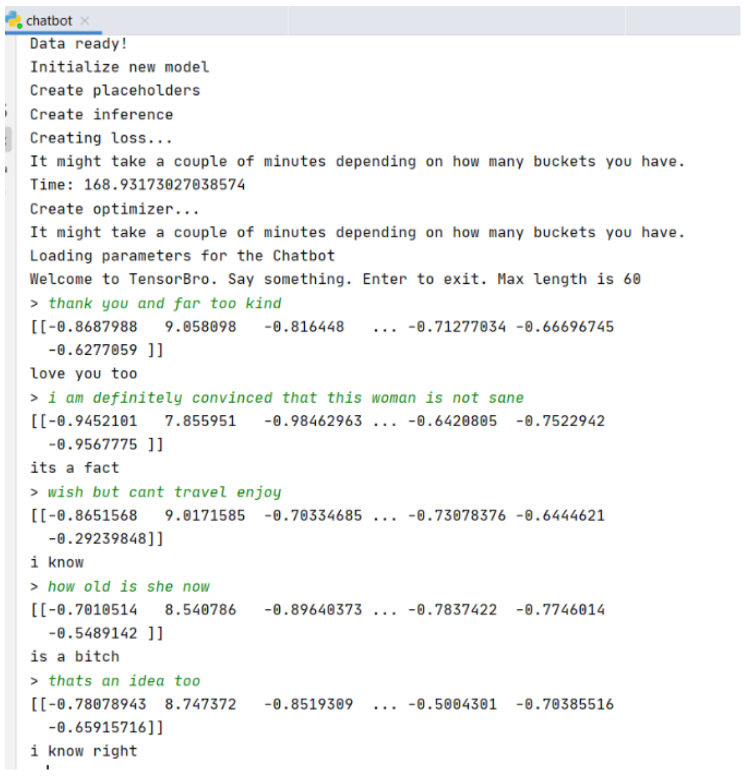
10. Once your data is ready then split into the train and test



```
def prepare_dataset(questions, answers):
    # create path to store all the train & test encoder & decoder
    make_dir(config.PROCESSED_PATH)
    # random convos to create the test set
    test_ids = random.sample([i for i in range(len(questions))],config.TESTSET_SIZE)
    filenames = ['train.enc', 'train.dec', 'test.enc', 'test.dec']
    files = []
    for filename in filenames:
        files.append(open(os.path.join(config.PROCESSED_PATH, filename),'w',errors='ignore'))

    for i in range(len(questions)):
        if i in test_ids:
            files[2].write(questions[i] + '\n')
            files[3].write(answers[i] + '\n')
        else:
            files[0].write(questions[i] + '\n')
            files[1].write(answers[i] + '\n')

    for file in files:
        file.close()
```

11. Train the model : run the chatbot.py file and train the model

Output for the Twitter chat bot



```
Data ready!
Initialize new model
Create placeholders
Create inference
Creating loss...
It might take a couple of minutes depending on how many buckets you have.
Time: 168.93173027038574
Create optimizer...
It might take a couple of minutes depending on how many buckets you have.
Loading parameters for the Chatbot
Welcome to TensorBro. Say something. Enter to exit. Max length is 60
> thank you and far too kind
[[-0.8687988   9.058098   -0.816448    ... -0.71277034 -0.66696745
  -0.6277059 ]]
love you too
> i am definitely convinced that this woman is not sane
[[-0.9452101   7.855951   -0.98462963 ... -0.6420805  -0.7522942
  -0.9567775 ]]
its a fact
> wish but cant travel enjoy
[[-0.8651568   9.0171585  -0.70334685 ... -0.73078376 -0.6444621
  -0.29239848]]
i know
> how old is she now
[[-0.7010514   8.540786   -0.89640373 ... -0.7837422  -0.7746014
  -0.5489142 ]]
is a bitch
> thats an idea too
[[-0.78078943  8.747372   -0.8519309  ... -0.5004301  -0.70385516
  -0.65915716]]
i know right
```

- **Dataset : More movie subtitles**
  https://github.com/Marsan-Ma/chat_corpus/

- **Description** : movie_subtitles_en file contains data with question and answer pairs, where odd lines are questions, even lines are answers.

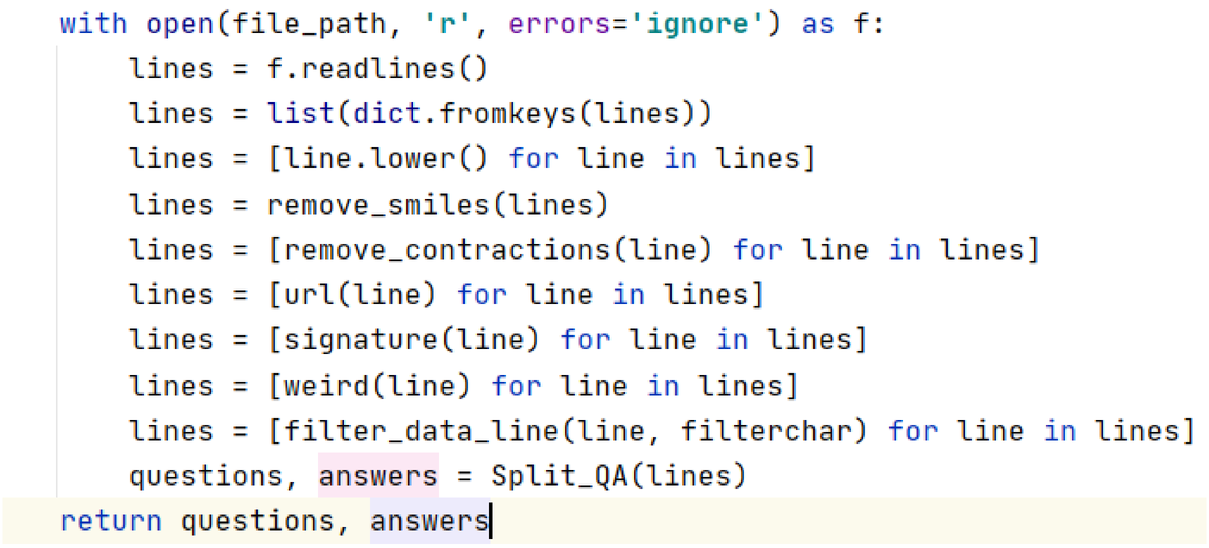
  - **Data Preprocessing :**

1. **Remove Duplicate sentence** :

```
lines = list(OrderedDict.fromkeys(lines))
```

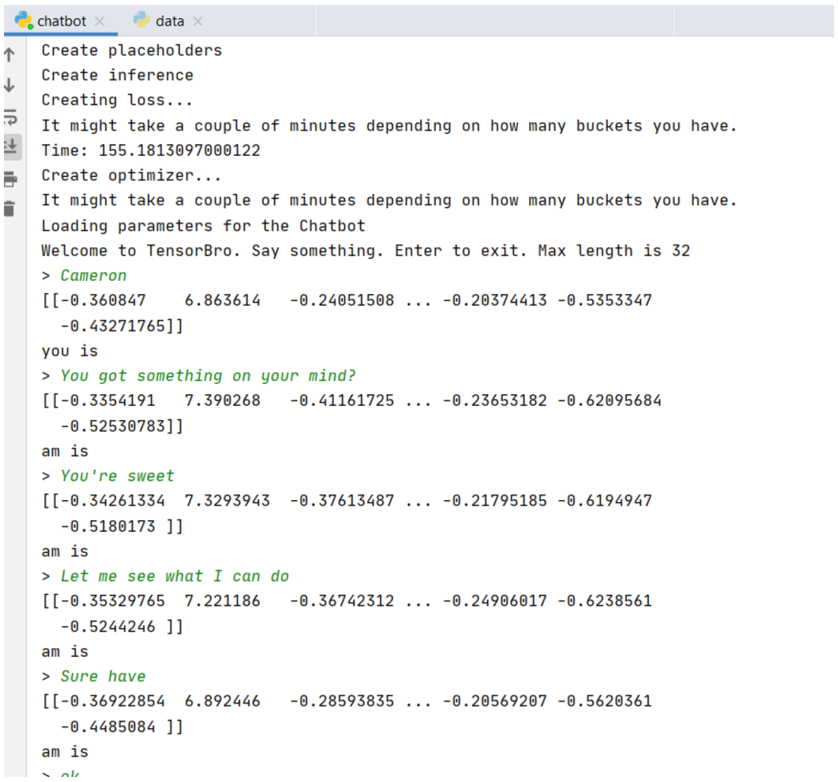
   And the same above process is followed for the data preprocessing

2. **Convert into lower case**
3. **Remove emoji**
4. **Remove contractions**
5. **Remove URL**
6. **Remove Signature**
7. **Remove weird things**
8. **Remove unnecessary characters**
9. **Filter and split data into questions and answers**
10. **Once your data is ready then split into the train and test**
11. **Run the chatbot.py file which is to train the model**



```
with open(file_path, 'r', errors='ignore') as f:
    lines = f.readlines()
    lines = list(dict.fromkeys(lines))
    lines = [line.lower() for line in lines]
    lines = remove_smiles(lines)
    lines = [remove_contractions(line) for line in lines]
    lines = [url(line) for line in lines]
    lines = [signature(line) for line in lines]
    lines = [weird(line) for line in lines]
    lines = [filter_data_line(line, filterchar) for line in lines]
    questions, answers = Split_QA(lines)
return questions, answers
```

Output for the Movie subtitle dataset



```
Create placeholders
Create inference
Creating loss...
It might take a couple of minutes depending on how many buckets you have.
Time: 155.1813097000122
Create optimizer...
It might take a couple of minutes depending on how many buckets you have.
Loading parameters for the Chatbot
Welcome to TensorBro. Say something. Enter to exit. Max length is 32
> Cameron
[[-0.360847    6.863614   -0.24051508 ... -0.20374413 -0.5353347
  -0.43271765]]
you is
> You got something on your mind?
[[-0.3354191   7.390268   -0.41161725 ... -0.23653182 -0.62095684
  -0.52530783]]
am is
> You're sweet
[[-0.34261334  7.3293943  -0.37613487 ... -0.21795185 -0.6194947
  -0.5180173 ]]
am is
> Let me see what I can do
[[-0.35329765  7.221186   -0.36742312 ... -0.24906017 -0.6238561
  -0.5244246 ]]
am is
> Sure have
[[-0.36922854  6.892446   -0.28593835 ... -0.20569207 -0.5620361
  -0.4485084 ]]
am is
```

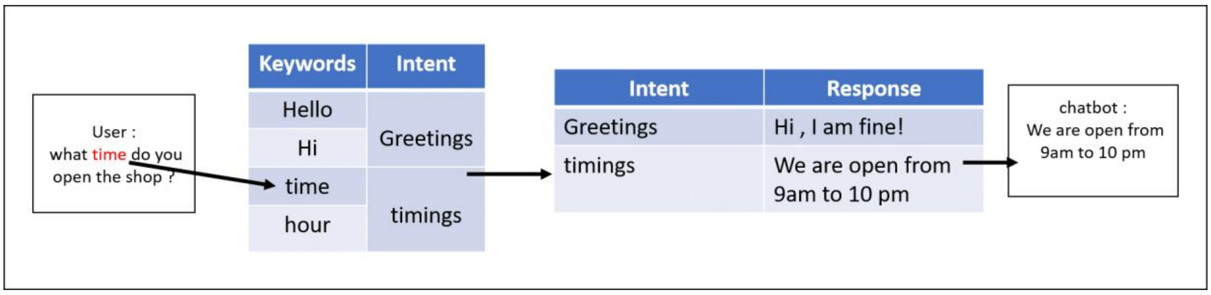
2. Make your chatbot remember information from the previous conversation

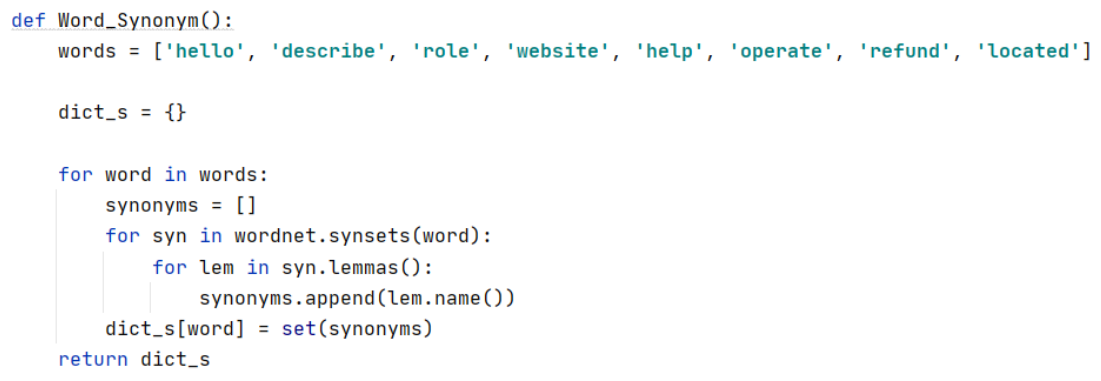
The Rule based approach :

- Check for particular keywords in a user's input. To understand what action the user needs to take, the keywords will be used (user's intent).
- The bot will then choose a response relevant to the intent until the intent is identified.

The Input and output of the response save in one file.

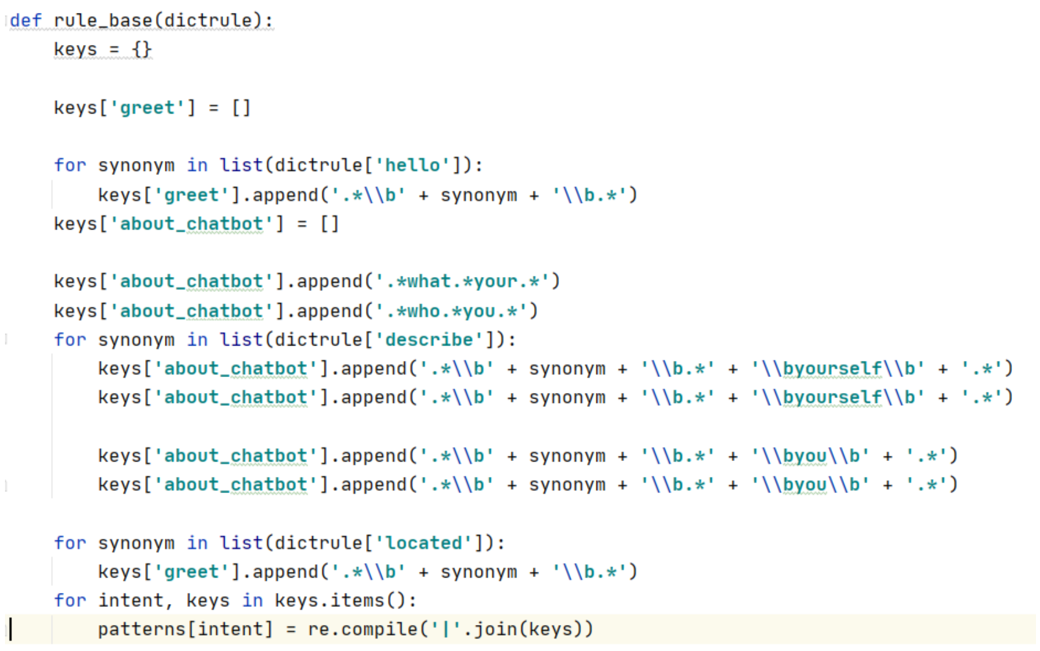
1. **Building the Keyword list** : Create a keyword list such as hello,describe and get the synonym of that keywords and save it in the dictionary.





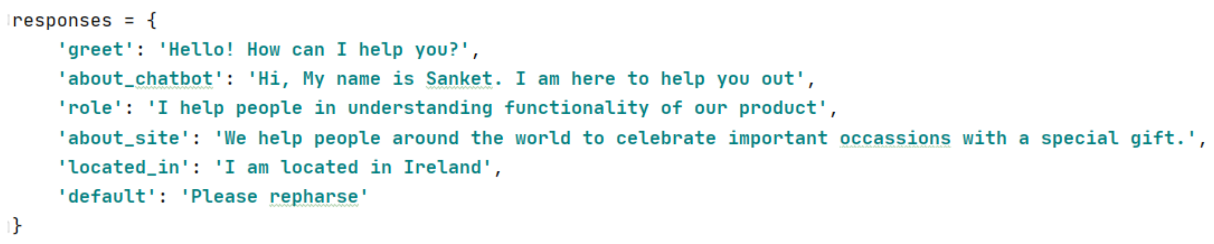
```python
def Word_Synonym():
    words = ['hello', 'describe', 'role', 'website', 'help', 'operate', 'refund', 'located']

    dict_s = {}

    for word in words:
        synonyms = []
        for syn in wordnet.synsets(word):
            for lem in syn.lemmas():
                synonyms.append(lem.name())
        dict_s[word] = set(synonyms)
    return dict_s
```

2. **Building the dictionary intents** : add keywords and its responses in the dictionary.



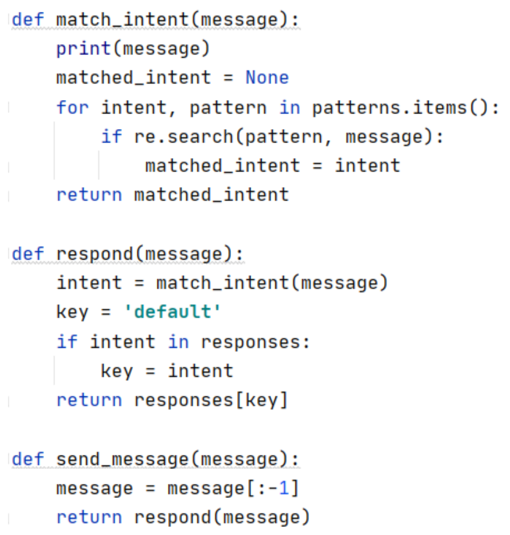
```python
def rule_base(dictrule):
    keys = {}

    keys['greet'] = []

    for synonym in list(dictrule['hello']):
        keys['greet'].append('.*\\b' + synonym + '\\b.*')
    keys['about_chatbot'] = []

    keys['about_chatbot'].append('.*what.*your.*')
    keys['about_chatbot'].append('.*who.*you.*')
    for synonym in list(dictrule['describe']):
        keys['about_chatbot'].append('.*\\b' + synonym + '\\b.*' + '\\byourself\\b' + '.*')
        keys['about_chatbot'].append('.*\\b' + synonym + '\\b.*' + '\\byourself\\b' + '.*')

        keys['about_chatbot'].append('.*\\b' + synonym + '\\b.*' + '\\byou\\b' + '.*')
        keys['about_chatbot'].append('.*\\b' + synonym + '\\b.*' + '\\byou\\b' + '.*')

    for synonym in list(dictrule['located']):
        keys['greet'].append('.*\\b' + synonym + '\\b.*')
    for intent, keys in keys.items():
        patterns[intent] = re.compile('|'.join(keys))
```

3. **Defining the Dictionary response** :



```python
responses = {
    'greet': 'Hello! How can I help you?',
    'about_chatbot': 'Hi, My name is Sanket. I am here to help you out',
    'role': 'I help people in understanding functionality of our product',
    'about_site': 'We help people around the world to celebrate important occassions with a special gift.',
    'located_in': 'I am located in Ireland',
    'default': 'Please repharse'
}
```

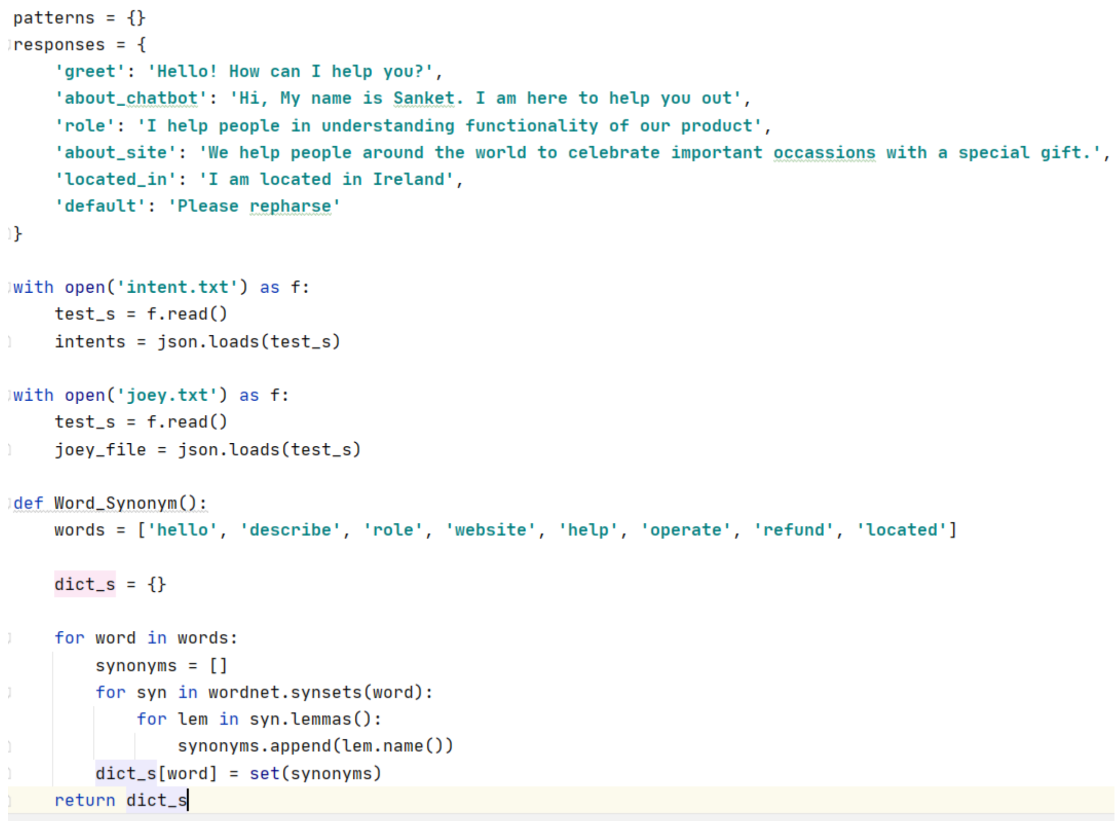
4. **Matching intents and giving response** : The input and the keyword matches and it gives the responses according to the input



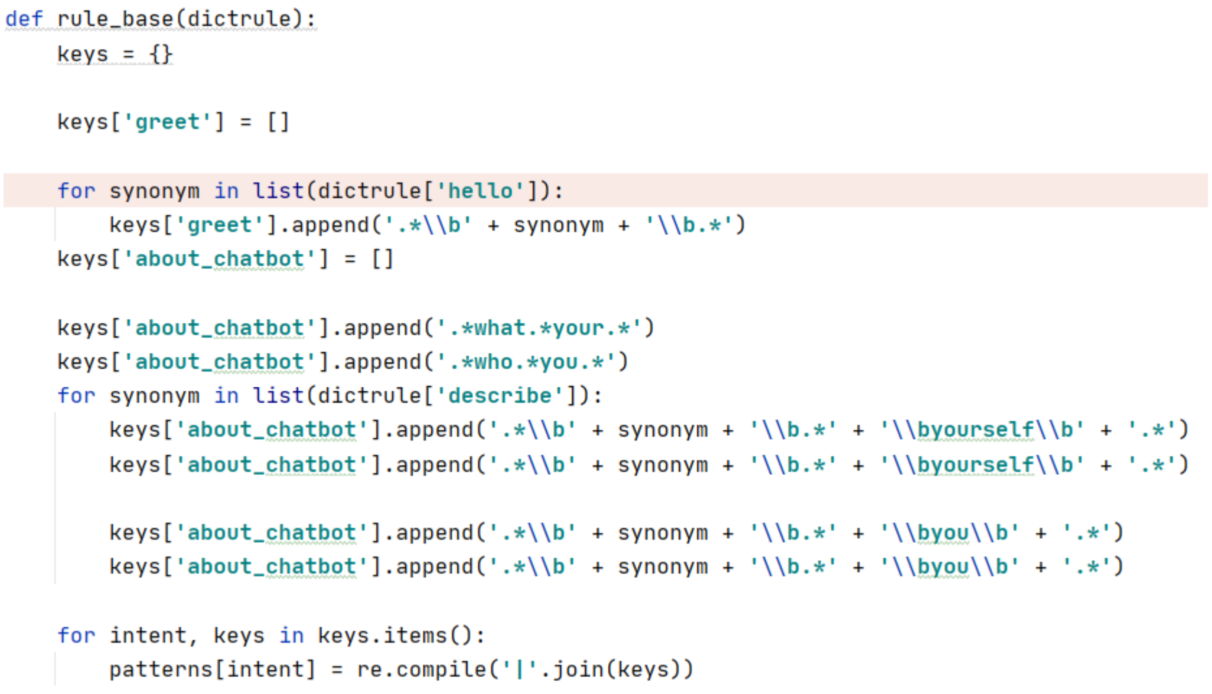
```python
def match_intent(message):
    print(message)
    matched_intent = None
    for intent, pattern in patterns.items():
        if re.search(pattern, message):
            matched_intent = intent
    return matched_intent

def respond(message):
    intent = match_intent(message)
    key = 'default'
    if intent in responses:
        key = intent
    return responses[key]

def send_message(message):
    message = message[:-1]
    return respond(message)
```

This all chat saved in Output_convo file and the chatbot scans the questions in the Output convo file first, if he goes the same then he will respond to the response previously given by the bot.

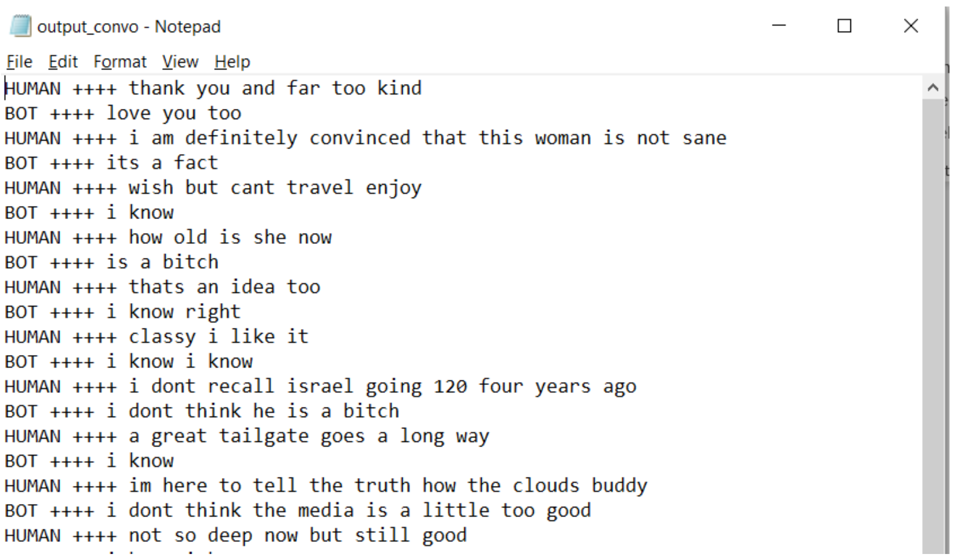
Following code is used for the rule based approach :



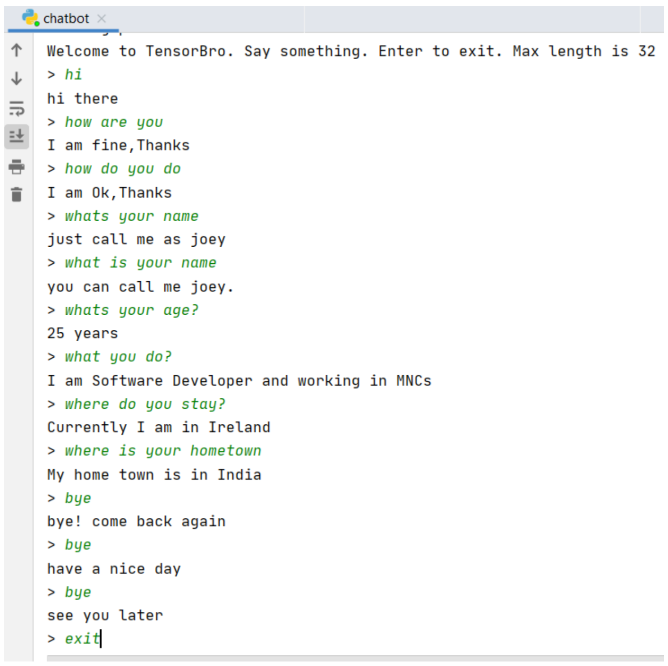
```python
patterns = {}
responses = {
    'greet': 'Hello! How can I help you?',
    'about_chatbot': 'Hi, My name is Sanket. I am here to help you out',
    'role': 'I help people in understanding functionality of our product',
    'about_site': 'We help people around the world to celebrate important occassions with a special gift.',
    'located_in': 'I am located in Ireland',
    'default': 'Please repharse'
}

with open('intent.txt') as f:
    test_s = f.read()
    intents = json.loads(test_s)

with open('joey.txt') as f:
    test_s = f.read()
    joey_file = json.loads(test_s)

def Word_Synonym():
    words = ['hello', 'describe', 'role', 'website', 'help', 'operate', 'refund', 'located']

    dict_s = {}

    for word in words:
        synonyms = []
        for syn in wordnet.synsets(word):
            for lem in syn.lemmas():
                synonyms.append(lem.name())
        dict_s[word] = set(synonyms)
    return dict_s
```



```python
def rule_base(dictrule):
    keys = {}

    keys['greet'] = []

    for synonym in list(dictrule['hello']):
        keys['greet'].append('.*\\b' + synonym + '\\b.*')
    keys['about_chatbot'] = []

    keys['about_chatbot'].append('.*what.*your.*')
    keys['about_chatbot'].append('.*who.*you.*')
    for synonym in list(dictrule['describe']):
        keys['about_chatbot'].append('.*\\b' + synonym + '\\b.*' + '\\byourself\\b' + '.*')
        keys['about_chatbot'].append('.*\\b' + synonym + '\\b.*' + '\\byourself\\b' + '.*')

        keys['about_chatbot'].append('.*\\b' + synonym + '\\b.*' + '\\byou\\b' + '.*')
        keys['about_chatbot'].append('.*\\b' + synonym + '\\b.*' + '\\byou\\b' + '.*')

    for intent, keys in keys.items():
        patterns[intent] = re.compile('|'.join(keys))
```



```
HUMAN ++++ thank you and far too kind
BOT ++++ love you too
HUMAN ++++ i am definitely convinced that this woman is not sane
BOT ++++ its a fact
HUMAN ++++ wish but cant travel enjoy
BOT ++++ i know
HUMAN ++++ how old is she now
BOT ++++ is a bitch
HUMAN ++++ thats an idea too
BOT ++++ i know right
HUMAN ++++ classy i like it
BOT ++++ i know i know
HUMAN ++++ i dont recall israel going 120 four years ago
BOT ++++ i dont think he is a bitch
HUMAN ++++ a great tailgate goes a long way
BOT ++++ i know
HUMAN ++++ im here to tell the truth how the clouds buddy
BOT ++++ i dont think the media is a little too good
HUMAN ++++ not so deep now but still good
```



```
Welcome to TensorBro. Say something. Enter to exit. Max length is 32
> hi
hi there
> how are you
I am fine,Thanks
> how do you do
I am Ok,Thanks
> whats your name
just call me as joey
> what is your name
you can call me joey.
> whats your age?
25 years
> what you do?
I am Software Developer and working in MNCs
> where do you stay?
Currently I am in Ireland
> where is your hometown
My home town is in India
> bye
bye! come back again
> bye
have a nice day
> bye
see you later
> exit
```

3. *Create a chatbot with personality*

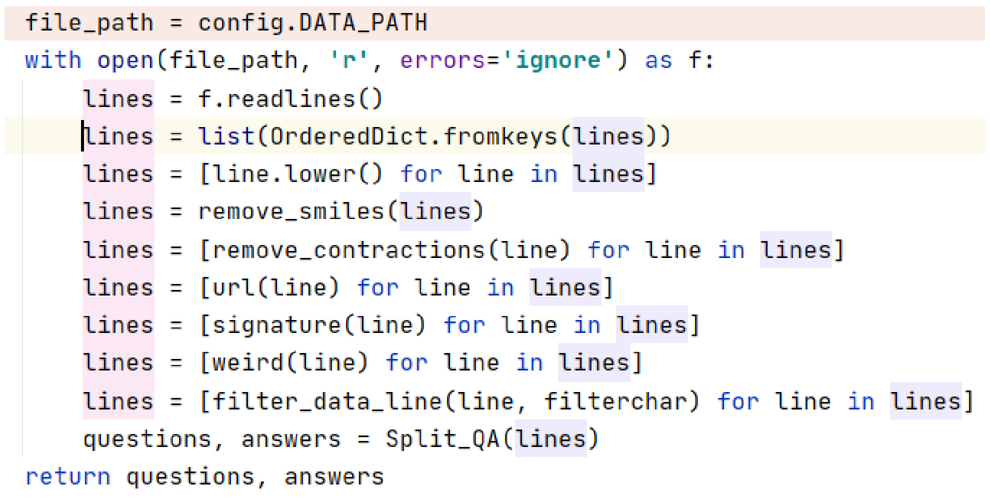
- **Dataset : res_joey** https://github.com/shbhmbhrgv/Personality-Chatbot/tree/master/data/lightweight

- **Description** : For this project we used the [Friends TV Corpus](https://fangj.github.io/friends/) We preprocessed the data to get conversations between main character(Joey)

- **Approach 1: At the decoder phase, inject consistent information about the bot such as name, age, hometown, current location, job.**

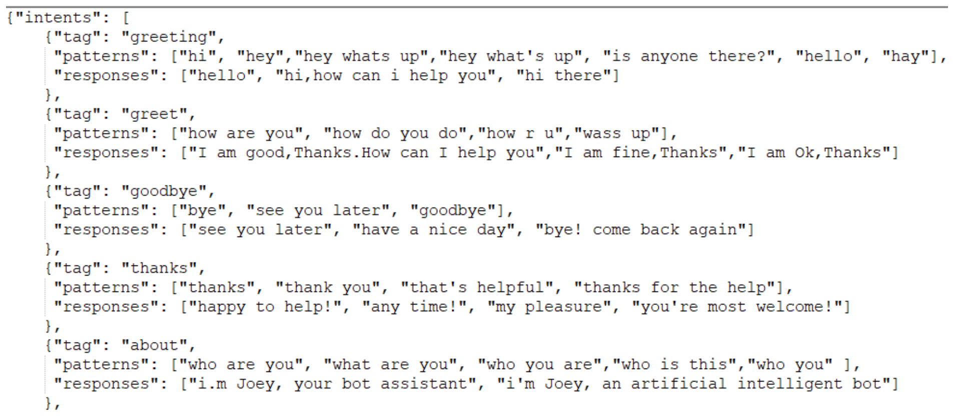
  - **Data Preprocessing** :

1. **Convert into lower case :**
2. **Remove emoji**
3. **Remove contractions**
4. **Remove URL**
5. **Remove Signature**
6. **Remove weird things**
7. **Remove unnecessary characters**
8. **Filter and split data into questions and answers**
9. **Once your data is ready then split into the train and test**
10. **Train the model**



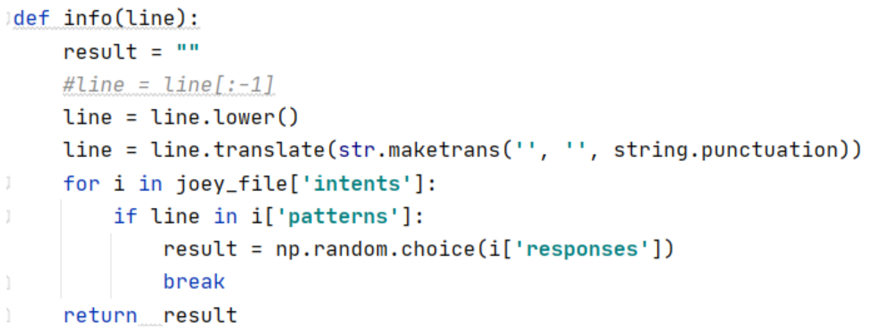
```python
file_path = config.DATA_PATH
with open(file_path, 'r', errors='ignore') as f:
    lines = f.readlines()
    lines = list(OrderedDict.fromkeys(lines))
    lines = [line.lower() for line in lines]
    lines = remove_smiles(lines)
    lines = [remove_contractions(line) for line in lines]
    lines = [url(line) for line in lines]
    lines = [signature(line) for line in lines]
    lines = [weird(line) for line in lines]
    lines = [filter_data_line(line, filterchar) for line in lines]
    questions, answers = Split_QA(lines)
return questions, answers
```

Name,age,hometown,curentlocation,job all consistent information are stored in the below joey file

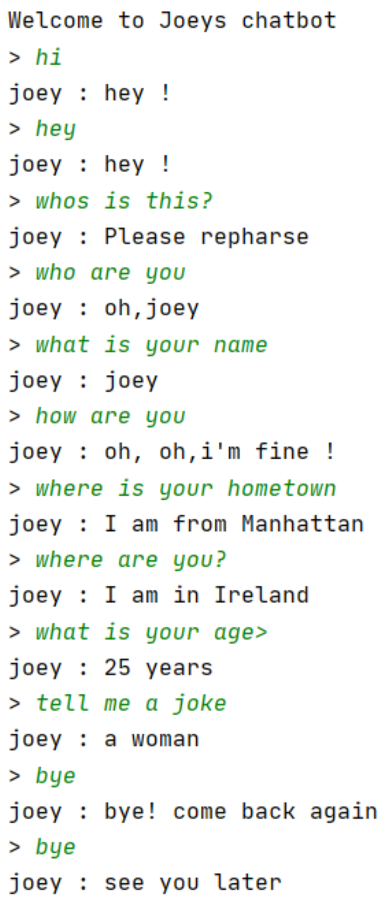


```json
{"intents": [
    {"tag": "greeting",
     "patterns": ["hi", "hey","hey whats up","hey what's up", "is anyone there?", "hello", "hay"],
     "responses": ["hello", "hi,how can i help you", "hi there"]
    },
    {"tag": "greet",
     "patterns": ["how are you", "how do you do","how r u","wass up"],
     "responses": ["I am good,Thanks.How can I help you","I am fine,Thanks","I am Ok,Thanks"]
    },
    {"tag": "goodbye",
     "patterns": ["bye", "see you later", "goodbye"],
     "responses": ["see you later", "have a nice day", "bye! come back again"]
    },
    {"tag": "thanks",
     "patterns": ["thanks", "thank you", "that's helpful", "thanks for the help"],
     "responses": ["happy to help!", "any time!", "my pleasure", "you're most welcome!"]
    },
    {"tag": "about",
     "patterns": ["who are you", "what are you", "who you are","who is this","who you" ],
     "responses": ["i.m Joey, your bot assistant", "i'm Joey, an artificial intelligent bot"]
    },
```

Code to get the information from the intent file



```python
def info(line):
    result = ""
    #line = line[:-1]
    line = line.lower()
    line = line.translate(str.maketrans('', '', string.punctuation))
    for i in joey_file['intents']:
        if line in i['patterns']:
            result = np.random.choice(i['responses'])
            break
    return result
```



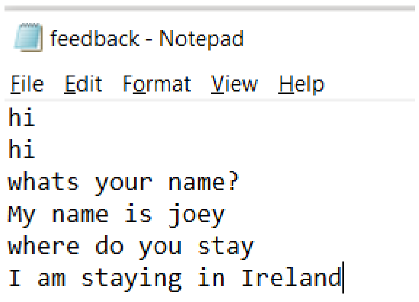
```
Welcome to Joeys chatbot
> hi
joey : hey !
> hey
joey : hey !
> whos is this?
joey : Please repharse
> who are you
joey : oh,joey
> what is your name
joey : joey
> how are you
joey : oh, oh,i'm fine !
> where is your hometown
joey : I am from Manhattan
> where are you?
joey : I am in Ireland
> what is your age>
joey : 25 years
> tell me a joke
joey : a woman
> bye
joey : bye! come back again
> bye
joey : see you later
```

4. Create a feedback loop that allows users to train your chatbot (30%)

If the user says that "That's wrong. You should have said xyz" so we are following the below steps

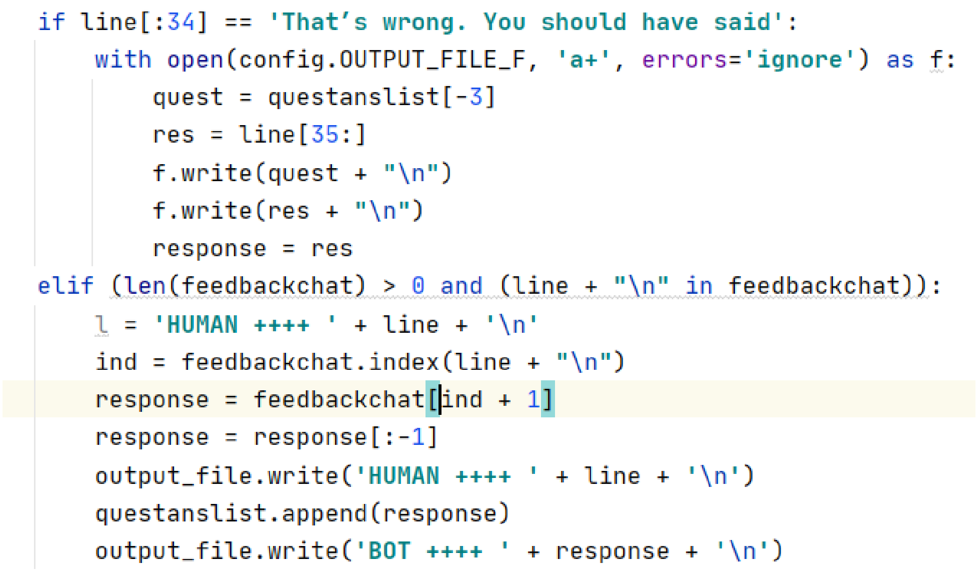
- Create a list which stores the questions and answer list
- If the user says "That's wrong. You should have said " then we take previous question from the list and the answer which user just said to change it.
- Save that question and the new answer in feedback file
- When user ask the same question again then first we will search that question into our file, if he finds then it gives new updated answer
- Next time when we train the model we will combine the feedback and train data.

Feedback file



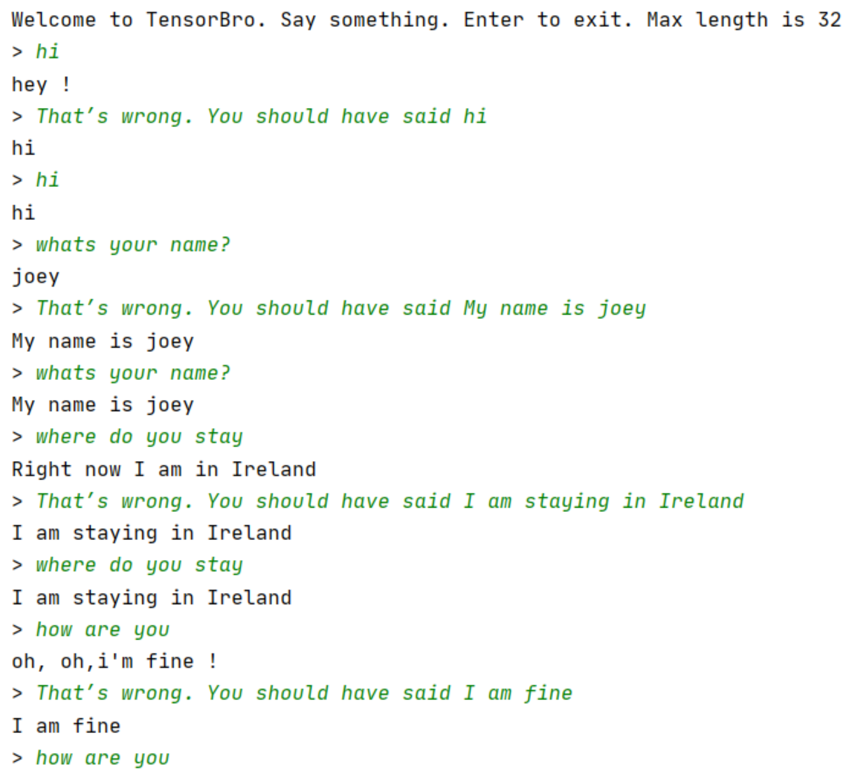
```
hi
hi
whats your name?
My name is joey
where do you stay
I am staying in Ireland
```

Following are the conditions use for the feedback loop:



```
if line[:34] == 'That’s wrong. You should have said':
    with open(config.OUTPUT_FILE_F, 'a+', errors='ignore') as f:
        quest = questanslist[-3]
        res = line[35:]
        f.write(quest + "\n")
        f.write(res + "\n")
        response = res
elif (len(feedbackchat) > 0 and (line + "\n" in feedbackchat)):
    l = 'HUMAN ++++ ' + line + '\n'
    ind = feedbackchat.index(line + "\n")
    response = feedbackchat[ind + 1]
    response = response[:-1]
    output_file.write('HUMAN ++++ ' + line + '\n')
    questanslist.append(response)
    output_file.write('BOT ++++ ' + response + '\n')
```

Output for feedback chat



```
Welcome to TensorBro. Say something. Enter to exit. Max length is 32
> hi
hey !
> That’s wrong. You should have said hi
hi
> hi
hi
> whats your name?
joey
> That’s wrong. You should have said My name is joey
My name is joey
> whats your name?
My name is joey
> where do you stay
Right now I am in Ireland
> That’s wrong. You should have said I am staying in Ireland
I am staying in Ireland
> where do you stay
I am staying in Ireland
> how are you
oh, oh,i'm fine !
> That’s wrong. You should have said I am fine
I am fine
> how are you
```

References:

https://chatbotslife.com/introducing-conversational-chat-bots-using-rule-based-approach-c8840aeaad07

https://github.com/shbhmbhrgv/Personality-Chatbot

https://blog.datasciencedojo.com/building-a-rule-based-chatbot-in-python/