Nama : Maria Enjelina Laru

NIM : 21220024

Matkul : Praktikum Sistem Cerdas

1. Hal yang harus di persiapkan:

* Anaconda3
* Install tensorflow
* Install pickle
* Install nltk

1. Setelah menginstall tools yang di perlukan, langkah selanjutnya buat nama file sebagai train\_chatbot.py. Mengimpor paket yang diperlukan untuk chatbot dan menginisialisasi variabel yang akan kami gunakan dalam proyek Python.

File data dalam format JSON jadi kami menggunakan paket json untuk mengurai file JSON ke dalam Python.

Program dari train\_chatboot.py.

import nltk

from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()

import json

import pickle

import numpy as np

from keras.models import Sequential

from keras.layers import Dense, Activation, Dropout

from keras.optimizers import SGD

import random

words=[]

classes = []

documents = []

ignore\_words = ['?', '!']

data\_file = open('intents.json').read()

intents = json.loads(data\_file)

for intent in intents['intents']:

    for pattern in intent['patterns']:

        #tokenize each word

        w = nltk.word\_tokenize(pattern)

        words.extend(w)

        #add documents in the corpus

        documents.append((w, intent['tag']))

        # add to our classes list

        if intent['tag'] not in classes:

            classes.append(intent['tag'])

# lemmaztize and lower each word and remove duplicates

words = [lemmatizer.lemmatize(w.lower()) for w in words if w not in ignore\_words]

words = sorted(list(set(words)))

# sort classes

classes = sorted(list(set(classes)))

# documents = combination between patterns and intents

print (len(documents), "documents")

# classes = intents

print (len(classes), "classes", classes)

# words = all words, vocabulary

print (len(words), "unique lemmatized words", words)

pickle.dump(words,open('words.pkl','wb'))

pickle.dump(classes,open('classes.pkl','wb'))

# create our training data

training = []

# create an empty array for our output

output\_empty = [0] \* len(classes)

# training set, bag of words for each sentence

for doc in documents:

    # initialize our bag of words

    bag = []

    # list of tokenized words for the pattern

    pattern\_words = doc[0]

    # lemmatize each word - create base word, in attempt to represent related words

    pattern\_words = [lemmatizer.lemmatize(word.lower()) for word in pattern\_words]

    # create our bag of words array with 1, if word match found in current pattern

    for w in words:

        bag.append(1) if w in pattern\_words else bag.append(0)

    # output is a '0' for each tag and '1' for current tag (for each pattern)

    output\_row = list(output\_empty)

    output\_row[classes.index(doc[1])] = 1

    training.append([bag, output\_row])

# shuffle our features and turn into np.array

random.shuffle(training)

training = np.array(training)

# create train and test lists. X - patterns, Y - intents

train\_x = list(training[:,0])

train\_y = list(training[:,1])

print("Training data created")

# Create model - 3 layers. First layer 128 neurons, second layer 64 neurons and 3rd output layer contains number of neurons

# equal to number of intents to predict output intent with softmax

model = Sequential()

model.add(Dense(128, input\_shape=(len(train\_x[0]),), activation='relu'))

model.add(Dropout(0.5))

model.add(Dense(64, activation='relu'))

model.add(Dropout(0.5))

model.add(Dense(len(train\_y[0]), activation='softmax'))

# Compile model. Stochastic gradient descent with Nesterov accelerated gradient gives good 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'])

#fitting and saving the model

hist = model.fit(np.array(train\_x), np.array(train\_y), epochs=200, batch\_size=5, verbose=1)

model.save('chatbot\_model.h5', hist)

print("model created")

1. Lalu buat file intents.jason dan isi kan code berikut.

Program dari intents.jason.

1. {"intents": [
2. {"tag": "greeting",
3. "patterns": ["Hi there", "How are you", "Is anyone there?","Hey","Hola", "Hello", "Good day"],
4. "responses": ["Hello, thanks for asking", "Good to see you again", "Hi there, how can I help?"],
5. "context": [""]
6. },
7. {"tag": "goodbye",
8. "patterns": ["Bye", "See you later", "Goodbye", "Nice chatting to you, bye", "Till next time"],
9. "responses": ["See you!", "Have a nice day", "Bye! Come back again soon."],
10. "context": [""]
11. },
12. {"tag": "thanks",
13. "patterns": ["Thanks", "Thank you", "That's helpful", "Awesome, thanks", "Thanks for helping me"],
14. "responses": ["Happy to help!", "Any time!", "My pleasure"],
15. "context": [""]
16. },
17. {"tag": "noanswer",
18. "patterns": [],
19. "responses": ["Sorry, can't understand you", "Please give me more info", "Not sure I understand"],
20. "context": [""]
21. },
22. {"tag": "options",
23. "patterns": ["How you could help me?", "What you can do?", "What help you provide?", "How you can be helpful?", "What support is offered"],
24. "responses": ["I can guide you through Adverse drug reaction list, Blood pressure tracking, Hospitals and Pharmacies", "Offering support for Adverse drug reaction, Blood pressure, Hospitals and Pharmacies"],
25. "context": [""]
26. },
27. {"tag": "adverse\_drug",
28. "patterns": ["How to check Adverse drug reaction?", "Open adverse drugs module", "Give me a list of drugs causing adverse behavior", "List all drugs suitable for patient with adverse reaction", "Which drugs dont have adverse reaction?" ],
29. "responses": ["Navigating to Adverse drug reaction module"],
30. "context": [""]
31. },
32. {"tag": "blood\_pressure",
33. "patterns": ["Open blood pressure module", "Task related to blood pressure", "Blood pressure data entry", "I want to log blood pressure results", "Blood pressure data management" ],
34. "responses": ["Navigating to Blood Pressure module"],
35. "context": [""]
36. },
37. {"tag": "blood\_pressure\_search",
38. "patterns": ["I want to search for blood pressure result history", "Blood pressure for patient", "Load patient blood pressure result", "Show blood pressure results for patient", "Find blood pressure results by ID" ],
39. "responses": ["Please provide Patient ID", "Patient ID?"],
40. "context": ["search\_blood\_pressure\_by\_patient\_id"]
41. },
42. {"tag": "search\_blood\_pressure\_by\_patient\_id",
43. "patterns": [],
44. "responses": ["Loading Blood pressure result for Patient"],
45. "context": [""]
46. },
47. {"tag": "pharmacy\_search",
48. "patterns": ["Find me a pharmacy", "Find pharmacy", "List of pharmacies nearby", "Locate pharmacy", "Search pharmacy" ],
49. "responses": ["Please provide pharmacy name"],
50. "context": ["search\_pharmacy\_by\_name"]
51. },
52. {"tag": "search\_pharmacy\_by\_name",
53. "patterns": [],
54. "responses": ["Loading pharmacy details"],
55. "context": [""]
56. },
57. {"tag": "hospital\_search",
58. "patterns": ["Lookup for hospital", "Searching for hospital to transfer patient", "I want to search hospital data", "Hospital lookup for patient", "Looking up hospital details" ],
59. "responses": ["Please provide hospital name or location"],
60. "context": ["search\_hospital\_by\_params"]
61. },
62. {"tag": "search\_hospital\_by\_params",
63. "patterns": [],
64. "responses": ["Please provide hospital type"],
65. "context": ["search\_hospital\_by\_type"]
66. },
67. {"tag": "search\_hospital\_by\_type",
68. "patterns": [],
69. "responses": ["Loading hospital details"],
70. "context": [""]
71. }
72. ]
73. }

4. **Praproses data**

Saat bekerja dengan data teks, kita perlu melakukan berbagai preprocessing pada data tersebut sebelum kita membuat machine learning atau model deep learning. Berdasarkan persyaratan, kami perlu menerapkan berbagai operasi untuk memproses data terlebih dahulu.

Tokenisasi adalah hal paling mendasar dan pertama yang dapat Anda lakukan pada data teks. Tokenizing adalah proses memecah seluruh teks menjadi bagian-bagian kecil seperti kata-kata.

Di sini kita mengulangi pola dan membuat token pada kalimat menggunakan fungsi nltk.word\_tokenize() dan menambahkan setiap kata dalam daftar kata. Kami juga membuat daftar kelas untuk tag kami.

5. **Membuat data pelatihan dan pengujian**

Sekarang, kami akan membuat data pelatihan di mana kami akan memberikan input dan output. Input kita akan menjadi pola dan output akan menjadi kelas milik pola input kita. Tapi komputer tidak mengerti teks jadi kami akan mengubah teks menjadi angka.

# create our training data

training = []

# create an empty array for our output

output\_empty = [0] \* len(classes)

# training set, bag of words for each sentence

for doc in documents:

# initialize our bag of words

bag = []

# list of tokenized words for the pattern

pattern\_words = doc[0]

# lemmatize each word - create base word, in attempt to represent related words

pattern\_words = [lemmatizer.lemmatize(word.lower()) for word in pattern\_words]

# create our bag of words array with 1, if word match found in current pattern

for w in words:

bag.append(1) if w in pattern\_words else bag.append(0)

# output is a '0' for each tag and '1' for current tag (for each pattern)

output\_row = list(output\_empty)

output\_row[classes.index(doc[1])] = 1

training.append([bag, output\_row])

# shuffle our features and turn into np.array

random.shuffle(training)

training = np.array(training)

# create train and test lists. X - patterns, Y - intents

train\_x = list(training[:,0])

train\_y = list(training[:,1])

print("Training data created")

6. **Jalankan chatbot**

Untuk menjalankan chatbot, kami memiliki dua file utama; **train\_chatbot.py** dan **chatapp.py** .

Pertama, kami melatih model menggunakan perintah di terminal:

python train\_chatbot. py

Jika kami tidak melihat kesalahan apa pun selama pelatihan, kami telah berhasil membuat model. Kemudian untuk menjalankan aplikasi, kami menjalankan file kedua.

chatgui python. py

Program akan membuka jendela GUI dalam beberapa detik. Dengan GUI Anda dapat dengan mudah mengobrol dengan bot.

Perintah menjalankan chatboot



