Chat Bot –Tekoaly

Install Packages

Before starting to work on our chatbot we need to download a few python packages.

- numpy  
- nltk  
- tensorflow  
- tflearn

Training Data

open intentschat.json file and add whatever you need to add in the same format given .whenever you edit the file you need to retrain it ie run the train.py before running bot.py . The tag on each dictionary in the file indicates the group that each message belongs too. With this data we will train a neural network to take a sentence of words and classify it as one of the tags in our file. Then we can simply take a response from those groups and display that to the user. The more tags, responses, and patterns you provide to the chatbot the better and more complex it will be.

Loading our JSON Data

We will start by importing some modules and loading in our json data. Make sure that your .json file is in the same directory as your python script!

**import** json **with** open**(**'intents.json'**)** **as** file**:** data **=** json**.**load**(**file**)**

Extracting Data

Now its time to take out the data we want from our JSON file. We need all of the patterns and which class/tag they belong to. We also want a list of all of the unique words in our patterns ,so lets setup some blank lists to store these values.

words **=** **[]**

labels **=** **[]**

docs\_x **=** **[]**

docs\_y **=** **[]**

Now its time to loop through our JSON data and extract the data we want. For each pattern we will turn it into a list of words using nltk.word\_tokenizer, rather than having them as strings. We will then add each pattern into our docs\_x list and its associated tag into the docs\_y list.

**for** intent **in** data**[**'intents'**]:**

**for** pattern **in** intent**[**'patterns'**]:**

wrds **=** nltk**.**word\_tokenize**(**pattern**)**

words**.**extend**(**wrds**)** docs\_x**.**append**(**wrds**)** docs\_y**.**append**(**intent**[**"tag"**])**

**if** intent**[**'tag'**]** **not** **in** labels**:**

labels**.**append**(**intent**[**'tag'**])**

Word Stemming

We will use this process of stemming words to reduce the vocabulary of our model and attempt to find the more general meaning behind sentences.

words **=** **[**stemmer**.**stem**(**w**.**lower**())**

**for** w **in** words **if** w **!=** "?"**]**

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

labels **=** sorted**(**labels**)**

This code will simply create a unique list of stemmed words to use in the next step of our data preprocessing.

Bag of Words

We need some way to represent our sentences with numbers and this is where a bag of words comes in. What we are going to do is represent each sentence with a list the length of the amount of words in our models vocabulary. Each position in the list will represent a word from our vocabulary. If the position in the list is a 1 then that will mean that the word exists in our sentence, if it is a 0 then the word is nor present. We call this a bag of words because the order in which the words appear in the sentence is lost, we only know the presence of words in our models vocabulary.

training **=** **[]**

output **=** **[]**

out\_empty **=** **[0** **for** \_ **in** range**(**len**(**labels**))]**

**for** x**,** doc **in** enumerate**(**docs\_x**):**

bag **=** **[]**

wrds **=** **[**stemmer**.**stem**(**w**.**lower**())**

**for** w **in** doc**]** **for** w **in** words**:**

**if** w **in** wrds**:**

bag**.**append**(1)**

**else:** bag**.**append**(0)**

output\_row **=** out\_empty**[:]**

output\_row**[**labels**.**index**(**docs\_y**[**x**])]** **=** **1** training**.**append**(**bag**)**

output**.**append**(**output\_row**)**

Developing a Model

or our purposes we will use a fairly standard feed-forward neural network with two hidden layers. The goal of our network will be to look at a bag of words and give a class that they belong too (one of our tags from the JSON file). Keep in mind that you can mess with some of the numbers here and try to make an even better model.

tensorflow**.**reset\_default\_graph**()**

net **=** tflearn**.**input\_data**(**shape**=[**None**,** len**(**training**[0])])**

net **=** tflearn**.**fully\_connected**(**net**,** **8)**

net **=** tflearn**.**fully\_connected**(**net**,** **8)**

net **=** tflearn**.**fully\_connected**(**net**,** len**(**output**[0]),** activation**=**"softmax"**)**

net **=** tflearn**.**regression**(**net**)** model **=** tflearn**.**DNN**(**net**)**

Training & Saving the Model

Once we are done training the model we can save it to the file **model.tflearn** for use in other scripts

model**.**fit**(**training**,** output**,** n\_epoch**=1000,** batch\_size**=8,** show\_metric**=**True**)** model**.**save**(**"model.tflearn"**)**

Loading a Model

model**.**load**(**"model.tflearn"**)**

Making Predictions

we need to remember that our model does not take string input, it takes a bag of words. We also need to realize that our model does not spit out sentences, it generates a list of probabilities for all of our classes

The bag\_of\_words function will transform our string input to a bag of words using our created words list. The chat function will handle getting a prediction from the model and grabbing an appropriate response from our JSON file of responses.