SCHOLARLY REVIEW ONLINE

SRO - Late Summer 2023 - 1

Examining the Runtime of NLTK and Tensorflow Algorithms for a Chatbot Based on intents.json Length

By Rishi Hariharaprasad

AUTHOR BIO

Rishi Hariharaprasad is a student programmer at Brandeis High School. He worked on the app "Disabled Health," which helps provide tech and health information to disabled people for this app. He was named a Congressional Award Winner. Rishi was born and raised in Texas, while his parents were born in India. Currently, he is working on creating a website to showcase Disabled Health and working on a UI/UX design for the application itself. He seeks to better understand the machine learning (ML) algorithm that is used in his app and find the most efficient way to read data from an intents.json file.

EXPANDED ABSTRACT

Today, over 1.4 billion users worldwide are using chatbots. (1) The chatbots that we use today are made by large companies with funding. People who want to create chatbots using their own system have issues compiling neural networks. Currently, the most popular module to use in Python for pre-made machine modules is TensorFlow. (2) This research paper also utilizes NLTK for neural networks; although it is not the industry standard, it is suitable for academic purposes. (3) I created a chatbot that provides health assistance to the disabled, and I observed a huge issue with epochs and how long it takes to train models. I wanted my system to be dynamic, but to be dynamic you would have to retrain each new tag in the intents.json file, and this takes a long time based on the byte size of your addition. Thus, I wanted to find out if the relationship between byte size and training length was linear or exponential. The purpose of this work is to find whether the runtime is linear or exponential, but also to find more efficient ways to train epochs based on byte size. The least squares regression line ended up being $\hat{y} = 914.90212X - 1291.32081$; the relationship between byte size and training runtime is linear, and thus, after calculating, it has an O(N) runtime. In this study, we sought to answer the question "what is the relationship between intents.json length and the tensorflow/nltk epoch runtime?" to find ways to maximize the efficiency of chatbot algorithms based on different usage. Hypotheses tested were all 1000 epochs, as large intents.json files take incredibly long to run after certain byte sizes. The present information is expanded upon in a forthcoming research paper.

Keywords: Computer Science; Algorithms; Runtimes; NLTK; TensorFlow; ChatBot; intents.json; Epochs; Byte-Analyzation; Relationships in CS; Machine Learning; AI; Deep Learning; Deep Speech.

GitHub: https://github.com/Rishi-prog731/disabledhealth

Code and Data

Code

Import Statements

va2.py

```
import nltk
from nltk.stem.lancaster import LancasterStemmer
stemmer = LancasterStemmer()

import numpy
import tflearn
import tensorflow
import random
import json
import pickle
```

ML/ TF Code

```
with open("intents.json") as file:
    data = json.load(file)

try:
    with open("data.pickle", "rb") as f:
        words, labels, training, output = pickle.load(f)
except:
    words = []
    labels = []
    docs_x = []
    docs_y = []

    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"])

    words = [stemmer.stem(w.lower()) for w in words if w != "?"]
    words = sorted(list(set(words)))

    labels = sorted(labels)

    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)


    training = numpy.array(training)
    output = numpy.array(output)

    with open("data.pickle", "wb") as f:
        pickle.dump((words, labels, training, output), f)

tensorflow.compat.v1.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)

try:
    model.load("model.tflearn")
except:
    model.fit(training, output, n_epoch=50, batch_size=8, show_metric=True)
    model.save("model.tflearn")


def bag_of_words(s, words):
    bag = [0 for _ in range(len(words))]

    s_words = nltk.word_tokenize(s)
    s_words = [stemmer.stem(word.lower()) for word in s_words]

    for se in s_words:
        for i, w in enumerate(words):
            if w == se:
                bag[i] = 1

    return numpy.array(bag)
```

Chat Function

```
while True:
    inp = input("You: ")
    if inp.lower() == "quit":
        break

    results = model.predict([bag_of_words(inp, words)])
    results_index = numpy.argmax(results)
    tag = labels[results_index]

    for tg in data["intents"]:
        if tg['tag'] == tag:
            responses = tg['responses']

    print(random.choice(responses))
```

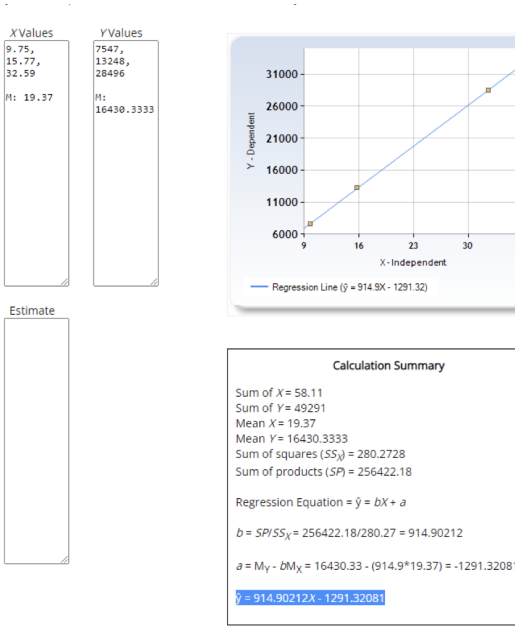
Data

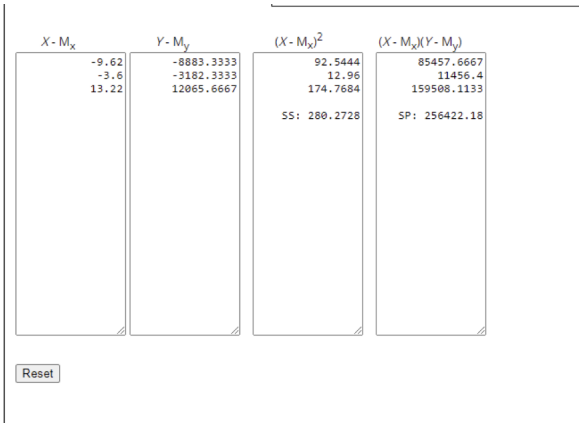
Table 1

intents.json Byte SIze	Runtime
7547 Bytes	9.75 s
13,248 Bytes	15.77 s
28,496 Bytes	32.59 s

Note: I ran 3 different experiments with these specific byte sizes, and all of them came out to this runtime; thus, I did not feel the need to copy-paste this same chart for simplicity.

Graph with Calculation Summary:





REFERENCES

1. Jovic, Danica. "The Future Is Now - 37 Fascinating Chatbot Statistics." *SmallBizGenius*, www.smallbizgenius.net/by-the-numbers/chatbot-statistics/#:~:text=1.4%20billion%20people%20are%20using%20chatbots.&text=Chatbot%20growth%20has%20been%20prominent,on%20a%20fairly%20regular%20basis.
2. MSV, Janakiram. "Tensorflow Turns 5 - Five Reasons Why It Is the Most Popular ML Framework." *Forbes*, Forbes Magazine, 27 Nov. 2020, https://www.forbes.com/sites/janakirammsv/2020/11/27/tensorflow-turns-5five-reasons-why-it-is-the-most-popular-ml-framework/?sh=60ca8ea7e67e.
3. "Is NLTK Outdated, and What Is the Best Alternative for It in Python?" *Quora*, https://www.quora.com/Is-NLTK-outdated-and-what-is-the-best-alternative-for-it-in-Python.