Hariharaprasad, R. International Journal of Computer Science and Mobile Applications, Vol 11 Issue 6, June-2023, pg. 01-08.

ISSN: 2321-8363
Impact Factor: 6.308

(An Open Accessible, Fully Refereed and Peer Reviewed Journal)

Examining the Runtime of NLTK and Tensor Flow Algorithms for Chatbot Based on intents.json Length

Rishi Hariharaprasad*
Louis D. Brandeis High School,13011 Kyle Seale Pkwy, San Antonio, Texas, United States
E-mail: rishi.hhp@gmail.com

Received date: 23 June 2023**,** Manuscript No. ijcsma-23-103697; **Editor assigned:** 25 June 2023**,** Pre QC No ijcsma-23-103697 (PQ); **Reviewed:** 09 July 2023, QC No. ijcsma-23-103697 (Q)**; Revised:** 15 July 2023**,** Manuscript No. ijcsma-23-103697 (R); **Published date:** 21 July 2023
doi. 10.5281/zenodo.8116501

Abstract

This research paper aims to explain an empirical investigation concerning the correlation between the length of an intents.json file and the runtime of NLTK (Natural Language Toolkit) and Tensor Flow algorithms in Python, tailored towards chatbot development. By examining the runtime efficiency based on various intents.json file sizes, the study aims to ascertain the impact of file length on algorithm performance through the utilization of a least squares regression line. Experimental analysis reveals a compelling linear relationship between intents.json length and training runtime, indicative of an O (N) runtime complexity. This research provides practical implications for developers seeking to enhance the runtime performance of chatbot systems by providing them with a baseline of runtime for dynamic chatbot creation.

Keywords*:* Computer science; Artificial intelligence; Machine learning; NLTK, Tensor flow; Runtime complexity; Algorithm runtime.

Hariharaprasad, R. International Journal of Computer Science and Mobile Applications, Vol 11 Issue 6, June-2023, pg. 01-08.

ISSN: 2321-8363
Impact Factor: 6.308

(An Open Accessible, Fully Refereed and Peer Reviewed Journal)

1. Introduction

Algorithms are paramount to any digital instrument. With the contemporary deluge of chatbots - with over 1.4 billion users worldwide - how humans interact with technology has been revolutionized [1]. While extensive companies with considerable resources frequently develop these chatbots, aspiring computer scientists have begun to dabble in assembling their chatbot systems.

Although a plethora of aspiring teens and adults run into issues compiling neural networks and consequently turn to pre-made Python modules that make compiling neural networks much faster.

Python offers prevalent and efficient machine learning models, with Tensor Flow as the industry standard [2]. However, for academic purposes, NLTK, a natural language processing module, is a suitable alternative for neural network development [3].

Similar to neural network development, there are artificial intelligence and machine learning. Although artificial intelligence and machine learning are used interchangeably, some distinct differences exist.

Artificial intelligence refers to the broader way of viewing a "simulation" of human intelligence onto computers. It utilizes various techniques: computer vision, natural language processing models, machine learning, etc. In short, it tries to emulate human intelligence in order to interact with their environment through the usage of algorithms.

Machine learning is a subset of artificial intelligence that deals with the development of both algorithms and models that allow computers to make predictions and decisions based on those predictions. Once the model is trained, it can be used for the entirety of the program without fail or explicit programming. They are dynamic systems: improving their performance through data and processing.

Like machine learning, natural language processing models are another subset of artificial intelligence. This subset encompasses computers being able to "speak" like a human: interpretation and generation of the human language. It combines techniques from linguistics and artificial intelligence to process and analyze text data in order to commonly create question-answering systems such as chatbots [4].

An algorithm is a step-by-step set of instructions designed to solve specific problems. In the case of this study, it is used to process and analyze data before making predictions based on the given data. Given the increased importance of chatbots in the real world with new technologies such as Chat GPT and slews of innovation within the field, this paper will study the relationship between NLTK (Natural Language Processing Module) and Tensor Flow algorithms based on the data that it processes.

This study investigates the relationship between intents.json length and the Tensor Flow/NLTK runtime in order to create a baseline for the efficiency of chatbot information creation. An intents.json file is utilized in the context of NLP systems in the field of virtual assistants. It is a configuration file that defines the various intents/ purposes behind user inputs or queries that the system can respond to. Each intent in the file is defined as an object with specific properties. Specifically, it has a unique identifier, examples (training data), actions, parameters, and contextual information. It is used in conjunction with dialogue management systems to build conversational AI systems in order to respond to user inputs more effectively. An "intents.json" is simply the standard naming convention.

The main significance of this paper lies in the fact that it can provide insights into the runtime required for training chatbot models based on machine learning. This can provide organizations with the ability to properly allocate their resources (processing, power, memory, money, etc.) based on the size of the data that they are processing. This can help optimize their infrastructure and improve their resource management for testing chatbot development. This research can serve as a baseline to help provide guidance on potential bottlenecks in the runtime of chatbot implementations - aspiring to enable more robust and efficient chatbot systems.

Hariharaprasad, R. International Journal of Computer Science and Mobile Applications, Vol 11 Issue 6, June-2023, pg. 01-08.

ISSN: 2321-8363
Impact Factor: 6.308

(An Open Accessible, Fully Refereed and Peer Reviewed Journal)

2. Methods

This controlled observational research design utilized a plethora of Python modules: NLTK, numpy, TFLearn, Tensor Flow, random, JSON, and pickle. Controlled observational research was the most optimal approach for the research objectives as it allowed for higher variable control, such as parameters, patterns, and byte size, and observes the output runtime in order to determine the relationship between the byte size and the runtime.

In the Natural Language Toolkit library, the program to test the runtimes utilized the Lancaster Stemmer, which reduces words to their root form. Creating and utilizing an instance of the Lancaster Stemmer is vital for the natural language processing model, which simplifies text classifications and information retrieval. It allows the machine learning algorithms to run without "worrying" about the core meaning of words. Utilizing training data from a version that removes the dimensionality of the user input makes the program more efficient by treating similar words as identical, thus removing the generalization and developing the efficiency of the models.

Numpy holds a plethora of important mathematical operations that were crucial in the design of this program. Array creation for training data and output labels was completely taken care of by numpy. Numpy was also employed in the manipulation of these arrays, transforming the output data and providing efficient computation and matrix operation. The study integrated numpy, mainly its diverse set of arrays, with Tensor Flow to serve as input data for training the machine learning models.

TFLearn is a deep learning library that is built on top of Tensor Flow that provides abstractions and utilities to both build and train neural networks. Tensor Flow, the crux of this research, on top of being a popular deep learning framework, has provided the design for this program with the ability for neural network implementation. The Lancaster Stemmer has four steps: initialization, suffix removal, stemming rules, and termination condition. It reduces words to their base form [5]. It defined the structure of the neural network model and trained it with data from the arrays, allowing for efficient computation. Since it utilizes both the CPU and GPU, Tensor Flow algorithms accelerate training processing by utilizing concepts such as parallel processing and optimization techniques to handle large-scale data potentially. In short, Tensor Flow allows the program to use incredibly large file lengths with the intents.json input for testing without causing additional errors.

Python's random module was a small but significant factor in this experiment, as it added a way for the chatbot to provide varied responses during its interactions, which is crucial for testing different responses. The JSON library was utilized for reading data from the intents.json file, which contained structured data defining intents, patterns, and responses for the chatbot. This allowed the program to extract and utilize the necessary information for training and using the chatbot.

Pickle sped up the process of training data. First, it allowed for object serialization - forcing Python objects into files. It provided me with the opportunity to load data from files in a binary format. It would read and write data in an efficient manner by reprocessing code over and over for each implementation.

By opening the intents.json file, which contains the intents and patterns for the chatbot, it loads the content into a variable. If there is no preprocessed data available, the model reprocesses the data from the intents file and serializes it into a pickle file. Next, process the data from the file by extracting the words, labels, training and output data, and patterns provided.

We then load the processed data into numpy arrays and integrate it with Tensor Flow through the binaries in the pickle file. To reset the runtime, the program would reset Tensor Flow's default graph, which clears any previously defined operations and allows for complete retraining of the data. The program also defines the neural network

Hariharaprasad, R. International Journal of Computer Science and Mobile Applications, Vol 11 Issue 6, June-2023, pg. 01-08.

ISSN: 2321-8363
Impact Factor: 6.308

(An Open Accessible, Fully Refereed and Peer Reviewed Journal)

architecture using TFLearn with a softmax function and a DNN model. The model utilized one-thousand epochs (amount of iterations) and a batch size of eight [6]. The epochs determined the runtime, and 1000 is a common baseline for epochs in machine learning literature [6]. The batch size refers to the number of data points processed through a neural network during training. The dataset is divided into a smaller subset of batches, and the batch size determines the size of the subsets. Eight is relatively small, which helps with memory and computational efficiency. The model was trained based on the data and saved in a TFLearn index, Meta, and data. Finally, the program converted the user input into a bag-of-words representation of the code in the form of a numerical vector from a predefined vocabulary [7]. The program tokenizes the sentence and stems the words in the sentence, comparing them to the predefined vocabulary and checking for the presence of specific words to be fed into the chatbot model to predict a suitable response based on learned patterns.

The study design was built around the described program. Other than the downloaded Python modules, materials utilized included the lightweight data interchange format with a pre-made corpus of intents and patterns in the Python software's ecosystem. Furthermore, in the environmental setup, the experiments were run on Python 3.10.5 on an Alien ware Aurora R7 with an Intel(R) Core(™) i7-8700 CPU @ 3.20 GHz (3.19 GHz usable), 16GB (15.8GB usable) RAM with Windows 10 Home, and a GTX 1080 Ti (11121 MB VRAM). The Python code was run in Sublime Text.

For measurement, the study tested three different byte sizes in three different trials, finding the runtime of how long it took to create and initialize the newly processed data into the chatbot model. The IDE Sublime Text has an inbuilt function that outputs the runtime of a program, which is what the study utilized to measure timing.

I utilized a linear regression model that examines the relationship between a set of independent and dependent variables that is represented in the form y=bx+c where y is the dependent variable, b is the slope, c is the constant, and x is the independent variable. The model examines the ability of the independent variables to predict the set of dependent variables - providing a good indication of which variables are significant in relation to impacting the other. This was important as it was possible that the entirety of this paper was built on a false pretense that the JSON length had nothing to do with the runtime, meaning that utilizing this model would tell me both the falsehood of this claim and if the claim was true, it would quantify it. Using a linear regression model helps identify if JSON length has an impact on run time. If so, it helps quantify the impact.

To create this linear regression model, the experiments were run on three different data sets of varying sizes: small, regular, and large. The small dataset was created as there are a lot of limitations to runnable byte-size maxes in the real world, and it would deal more with utilizing chatbots for more niche purposes. The "regular" data set included an average amount of data and was used to deal more with the application interface as it provided a byte size that was consistent with highly-specific application chatbots. The larger byte size included an abnormally large amount of data and was tested to check if the runtime complexity was parabolic or linear and if the linearity of the byte size had an effect on runtime performance.

3. Results

After compiling the code and the results, this study produced a least squares regression line of $\hat{y}$=914.90212, x-1291.32081. The y-value, in this case, is the byte size of the intents.json file; the x-value is the runtime in seconds. To confirm this data, the study tried the following numerical values in the training. The program compiled three different byte sizes (7547 bytes, 13248 bytes, and 28496 bytes) and got three different runtimes (9.75s, 15.77s, and 32.59s, respectively). Repeated trials with these byte sizes resulted in the same runtimes since the program is a

Hariharaprasad, R. International Journal of Computer Science and Mobile Applications, Vol 11 Issue 6, June-2023, pg. 01-08.

ISSN: 2321-8363
Impact Factor: 6.308

(An Open Accessible, Fully Refereed and Peer Reviewed Journal)

deterministic algorithm.

Deterministic algorithms produce the same result given the same input and conditions, providing a consistent runtime [8]. The algorithm follows a fixed set of steps with no elements of probability or randomness, allowing for the same execution time. In a more nuanced sense, the process of training the chatbot using TFLearn and Tensor Flow followed a fixed set of steps.

The model training process involved processing the input data, creating/configuring the neural network architecture, specifying the number of epochs (1000) and batch size of eight, and fitting the model to the training data. These deterministic steps were combined with the bag-of-words representation, which simply converted user input into numerical vectors, which is another deterministic process. It tokenized the input sentence, applied stemming (using Lancaster Stemming in NLTK), and compared the words to a predefined vocabulary, ultimately producing the same results given the same input and conditions. In the **Table 1** and **Figure 1**, the numbers are completely consistent for any trial.

Table 1. Runtime analysis for varying byte size in the computational model.

intents.json	Runtime (Trial 1)	Runtime (Trial 2)	Runtime (Trial 3)
7,547 bytes	9.75s	9.75s	9.75s
13,248 bytes	15.77s	15.77s	15.77s
28,496 bytes	32.59s	32.59s	32.59s

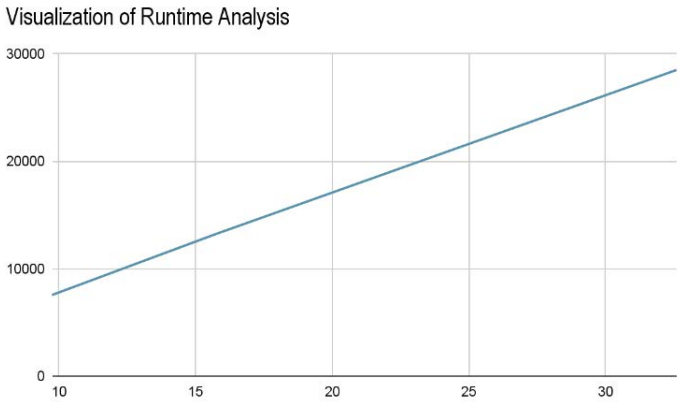

Figure 1. Graphs Generated by Google Drawings Graph Function that Show the Linear Regression Result of the Data.

ISSN: 2321-8363
Impact Factor: 6.308
(An Open Accessible, Fully Refereed and Peer Reviewed Journal)

I plotted one of each trial of each byte size respectively as the data was very monotonous. The sum of the runtimes (58.11), the sum of the byte sizes (49,291), mean of the runtimes (19.37), the mean of the byte sizes (16430.33), the sum of the squares for the x (ssx = 280.2728), and the sum of the products (sp = 256422.18) were all utilized to calculate the LSRL (least squares regression line). The slope value is SPISSx = 256422.18/280.27, which nets us a value of 914.90212, and a value is My-bMx = 16430.33 - (914.9*19.37) = -1291.32081

4. Discussion

The study sought to investigate and quantify the affinity between the byte size of the intents.json file and the machine learning compilation runtime. This examination gauged the compilation runtime of three different data sets of deviating sizes to determine whether the byte size particularly influenced the runtime period.

The results of the analysis indicate that while the data does not fit a strictly linear behavior, assuming a linear relationship between the byte size and the runtime due to the program's O (N) runtime complexity is allowed.

O (N) implies that the runtime of the algorithm grows approximately linearly with the size of the input, and in this case, the data processing steps, stemming of words and bag-of-words representation contribute to this complexity.

The analysis further indicates a possibility of an error in the runtime check in Sublime text, as the slope-intercept line equation from two points - that being (9.75,7547) and (32.59,28496) is y=917.2066549912433x-1395.76488616.

This is nearly identical to the least squares regression line, which means that there is a possibility that it is entirely linear, and there was some regulation error. Either way, the research found the data to be non-linear, so external factors must be considered.

Note that the non-linearity observed in the data could be attributed to heterogeneous data. While O (N) provides an upper-bound estimate for runtime complexity, external factors such as nested loops or recursive calls in the code may introduce variations and impact the actual runtime [9].

One limitation of this study is the relatively small sample size of both the computers used and the intents.json file sizes. Generalizing these findings would benefit from larger and more diverse samples in future studies. Additionally, exploring the effects of CPU and GPU overclocking, upgrading, or downgrading on timing could provide valuable insights into the performance of the chatbot model.

It is entirely possible that the amount of time it takes to train each model in the Tensor Flow DNN function is dependent on the specifications of the clone's computer. Replicating this research would require the user to utilize the same computer specifications used in this study, and further research could be taken to find whether or not the claim of the runtime being dependent on specifications is true. Furthermore, this study relied on deterministic algorithms and a bag-of-words representation for data processing. However, contextual data and quality issues in natural language processing were not thoroughly studied, which may introduce additional complexities and variations in the runtime performance.

Despite these limitations, the present study serves as a baseline for understanding the runtime complexity of the

ISSN: 2321-8363
Impact Factor: 6.308
(An Open Accessible, Fully Refereed and Peer Reviewed Journal)

chatbot model in the context of artificial intelligence and machine learning algorithms. It contributes to the existing body of knowledge and highlights the importance of considering algorithm runtime in the development and optimization of natural language processing systems.

In conclusion, the findings of this study suggest a linear relationship between the byte size of the intents.json file and the runtime of the chatbot model, as indicated by the O (N) runtime complexity. However, the non-linearity observed in the data underscores the influence of heterogeneous data and external factors on the actual runtime performance. Future research should aim to address the limitations of this study by incorporating larger sample byte sizes, exploring hardware-related factors, and considering the impact of contextual data on the performance of chatbot models. By expanding on these different research directions, people can further enhance the understanding and optimization of chatbot models in real-world applications [10].

5. Conclusion

As is evident from the data and the graphs, the data does not fit a strictly linear behavior, but the assumption of linearity due to the program being of O (N) runtime is accredited the non-linearity to heterogeneous data. In the field of computer science, O (N) refers to the runtime complexity of an algorithm, and in this case, specifically the stemming of the words, the bag of words, and the data processing.

O (N) indicates that the runtime of the algorithm grows approximately linearly with the size of the input. O (N) is a worst-case runtime complexity (upper-bound), and some external factors in the code through nested loops or recursive calls obviously have some other unforeseen impact on the runtime. A limitation of the present study was its relatively small sample size of both computers to use and intents.json sizes.

Attempting to generalize these findings, the program and thus study design had to opt for more utilized Python models. Hence, larger and more varied samples for future studies would be very beneficial, and it would also be productive to see if CPU and GPU overclocking, upgrading, or downgrading have any effect on the timing.

Sadly, the only cross-validation technique used was running the simulation multiple times, which means there is a caveat in the breadth and depth of the data utilized to answer and confirm the hypothesis. Same-sized files also might perform differently depending on contextual data - which was not studied in this research, which potentially botches the data. Quality issues, natural language processing bias, and computational constraints plagued this study, somewhat limiting the scope. However, this study can stand on its own to serve as a baseline for natural language process development. These results add to the body of knowledge of algorithm runtime for artificial intelligence and highlight how machine learning algorithms run.

6. Biography

Rishi Hariharaprasad is a 12th-grade programmer at Brandeis High School. He developed the app “eHealth,” which helps provide tech and health information to disabled people for this app. Rishi was born and raised in Texas, while his parents were born in India. He garnered recognition as an international QBH Hackathon winner, Rise Finalist, and Congressional App winner. His scholastic excellence is exemplified by his placing at NSF’s Computer Science Bee at the national and regional levels. Rishi is not just a coding enthusiast but is also involved in his community, curriculum development, school clubs, academics, and debate. He is currently writing this research paper to better understand the machine learning (ML) algorithm that is used in his app and find the most efficient way to read data

ISSN: 2321-8363
Impact Factor: 6.308
(An Open Accessible, Fully Refereed and Peer Reviewed Journal)

from an intents.json file

References

[1] Jovic, D. "The future is now–37 fascinating chatbot statistics, Smallbizgenius." (2020).
[2] Janakiram, M. S. V. "Tensor Flow Turns 5 - Five Reasons Why It Is the Most Popular ML Framework." *Forbes Forbes Mag.*, (2020).
[3] Swaathi K. G., blogger: Swaathi Kakarla is the co-founder and CTO at Skcript. She enjoys talking and writing about code efficiency. "*Nat Lang Process: NLTK Spacy, Act"* (2021).
[4] Natural Language Processing (NLP) - A Complete Guide." *(NLP) A Complete Guid.*
[5] Paice, C. D. "Another stemmer." *ACM Sigir Forum.* 24(3), (1990).
[6] Brownlee, J. "What is the Difference Between a Batch and an Epoch in a Neural Network." *Mach Learn Mastery.* 20, (2018).
[7] Brownlee, J. "A Gentle introduction to the bag-of-words model." *Mach Learn Mastery.* 7 (2019).
[8] "Difference between Deterministic and Non-Deterministic Algorithms." *GeeksforGeeks.* (2023).
[9] Olawanlet J. "Big O Cheat Sheet – Time Complexity Chart." *Free Org.*, (2023).
[10] "Learning from Heterogeneous Data." *Parietal*.