

Performance Evaluation of Machine Learning App Approach to Modular Arrangement of Predetermined Time Standard



Emmanuel Basitere, Ilesanmi Daniyan , Khumbulani Mpofu ,
and Adefemi Adeodu

Abstract Merging Modular Arrangement of Predetermined Time Standard (MODAPTS) and techniques used in the fourth industrial revolution (4IR) such as Machine Learning (ML) can start to improve the user experience of time standards. This study used Artificial Neural Networks (ANN) applied as chatbots to see whether ML could indeed improve MODAPTS in terms of ease, pace, and accessibility. The conventional and ANN methods were compared with assistance from logistics engineers, and the ANN approach. A chatbot using ANN was created and packaged on an html page presented to the research participants making use of a mobile device. The experiment used five material handling written scenarios to emulate the observation process, looking at the traditional approach when conducting a MODAPTS time study then followed by the ANN solution making use of the chatbot. ANN was found to be 0.25 min faster at a prediction rate of over 90% when the chatbot was in use. The result showed that machine learning could indeed be used with MODAPTS to equal performance and potentially improve the use of the time standard. The neural network was able to accurately predict the MODAPTS code of 94.7% of the 262 activities entered by the research participants. The potential to add other ML learning techniques and time study methods exists, the template is flexible enough to be moulded into a tool that all engineers can adapt in their different working environment.

Keywords ANN · Chatbot · 4IR · MODAPTS

E. Basitere · I. Daniyan (✉) · K. Mpofu
Department of Industrial Engineering, Tshwane University of Technology, Pretoria, South Africa
e-mail: daniyan.ia@achievers.edu.ng

A. Adeodu
Department of Mechanical Engineering, University of South Africa, Florida, South Africa

© The Author(s), under exclusive license to Springer Nature Singapore Pte Ltd. 2024
G. L. Conte and O. Sename (eds.), *Proceedings of the 11th International Conference on Mechatronics and Control Engineering*, Lecture Notes in Mechanical Engineering,
https://doi.org/10.1007/978-981-99-6523-6_9

117

1 Introduction

Conducting a MODAPTS analysis usually involves several components that would allow for a successful feasibility study, this includes but not limited to understanding an activity by video analysis or continuous observation, analysing possible efficiencies from that observation, and most importantly an in-depth knowledge of the MODAPTS time standard and its principles to prove those efficiencies. Mastering the MODAPTS time standard can be a bit challenging and rather complicated for the engineer [1]. The total time it takes to conduct a MODAPTS study may take too long due the nature of the observed activity and converting those activity tasks to MODAPTS code. MODAPTS is rather an old-fashioned time standard when looking at its inception and application, but efforts were made to simplify the time standard in multiple environments. Applications such as clerical, transit, janitorial and sewing are some of the options an engineer may explore. These applications are dated and too wordy, specifically when looking to solve a problem as an engineer today, in this era of 4IR. Suffice to say the world has changed drastically today from the 1960's era to the era of the 4IR. Furthermore, the adoption of Artificial Intelligence (AI) has made it easier to solve difficult problems experienced in the past [2–4].

Merging the MODAPTS time standard with solutions birthed from 4IR such as machine learning, can help reduce the complexity and improve efficiency when conducting feasibility studies. A notable example is research conducted by Wu et al. [5], who proposed incorporating motion analysis to the MODAPTS time standard by adding an extensive ergonomics template called Principle Components Analysis (PCA). The study found that the PCA-based approach was rated 80.08% amongst its participants and was more efficient at three minutes as opposed to the one hour that it took using the traditional MODAPTS approach. Another notable inclusion was a wearable sensory glove proposed by Mallembakam [1], the research focused on using a Bluetooth module mounted onto a glove that translated hand movements into MODAPTS code, this was compared to the traditional MODAPTS approach and it was found that the glove showed reliable and reputable results with that of using MODAPTS the conventional way.

An untapped field that would be an impactful inclusion is the merger of MODAPTS and machine learning. Today machine learning is more prevalent making it easier to solve difficult problems and improve livelihoods [6–8]. A massive advantage to researching machine learning techniques is that they are largely open sourced when looking at big data [9], and can be applied to the MODAPTS time standard.

The significance of this study is that a consolidation of multiple MODAPTS process keywords was collected for continual neural network training. Furthermore, a potential pattern recondition system to assist the engineer during observations may emerge through continues use of the ANN research template. Hence, the research looks at a form of Machine Learning called Artificial Neural Networks (ANN) as an application to MODAPTS with the aim to improve the activities involved during time study observations whilst making use of the MODAPTS time standard. Hence,

this research will look to solve the issue of rapidity and the ease of use in conducting a MODAPTS study making use of machine learning.

2 Methodology

A chatbot using ANN was created and packaged on an html page presented to the research participants making use of a mobile device. The experiment used five material handling written scenarios to emulate the observation process, looking at the traditional approach when conducting a MODAPTS time study then followed by the ANN solution making use of the chatbot.

The chatbot measured each research participant against several key attributes that defined the user experience toward the chatbot application. The experiment used a simple NN chatbot to employ a MODAPTS conversation with the research participants. The focus was on the predicted MODAPTS code produced by the chatbot, each time a research participant presented an activity. Primary data was collected from MODAPTS time studies produced by DSV and was used to train the chatbot. This was mostly dialect in the form of keywords showing how DSV described their process activities.

The chatbot was build making use of the python programming language, it used a ML module called Keras to train and implement the NN. The python code was executed using an integrated development environment called PyCharm.

2.1 *The Chatbot Intent File*

Constructing a chatbot consisted of several programming files working together to create the application. The chatbot was first initialised with a JavaScript Object Notation file called “intents.js”. This file defined the possible intentions that could take place during interactions with the chatbot [9–13]. Figure 1 shows a snippet of the “intents.js” file.

The intents file consisted of a tag element i.e., a subcategory that a user query might fall into. Each terminal class “get” and “put” were paired with a movement class element and the degree of difficulty to form the tag. Any query for example that used the “hand” movement to “get” (terminal) an object with an “easy” degree of difficulty would fall under the “hand_get_easy” tag or subcategory.

The next element in the intents.js file was the patterns, a group of keywords that a user query must contain to fall under a specific tag or subcategory. The patterns keywords were created using synonyms of the terminal class (get, put) paired up with the movement class and the degree of difficulty. A user query “Pick box using your hand at an easy stance” will fall under the “hand_get_easy” category as the keywords “pick”, “hand” and “easy” are present in the query.

```
{  
  "tag": "hand_get_easy",  
  "patterns": [  
    "obtain hand easy",  
    "collect hand easy",  
    "get hand easy",  
    "take hand easy",  
    "acquire hand easy",  
    "pick hand easy",  
    "hold hand easy",  
    "gain hand easy",  
    "attain hand easy"  
  ],  
  "responses": [  
    "M2G1"  
  ]  
}
```

Fig. 1 Code snippet of the “intents.js” file

Once a user query falls under a tag, then the chatbot will issue a response found under the responses keyword. The response of the user query was the MODAPTS code designated for the tag [14].

Table 1 shows an example of a general query made to the chatbot NN.

The chatbot was able to break down sentences if the user was able to split activities of the sentence using a period, comma or the “And” keyword. Each research participant was given this example including a list of rules and conversion table to use when querying the chatbot.

Table 2 shows the MODAPTS symbol conversion when querying the Chatbot NN.

Table 1 Example of a chatbot query

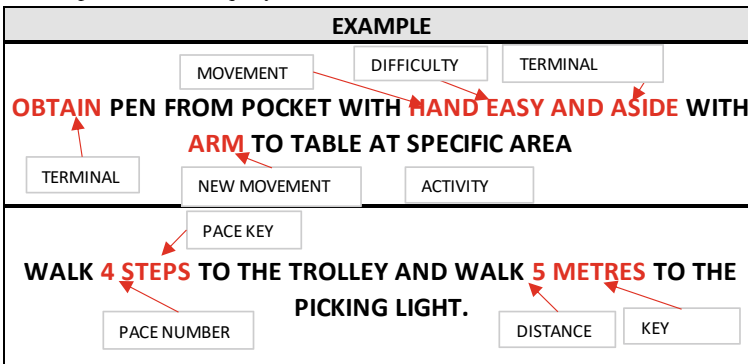




Table 2 Conversion table

	SYMBOL	DESCRIPTION
	M1	Fingers
	M2	Hand
	M3	Arm
	M4	Shoulder
	M5	Extended Shoulder
	M7	Trunk
	DIFFICULTY	G1
G3		Difficult
P0		General
P2		Specific
P5		Exact
	W5	Steps
	W7.75	Metres

2.2 Deployment of the Experiment

The chatbot was deployed to the research participants making use of a webpage using a mobile device. The code for the chatbot was packaged and stored on a server and accessed on request by the web page using the flask micro web framework. A user request would be captured on webpage, then sent to the server to query the chatbot NN, then perform some MODAPTS calculations and then finally returns the results to the user on the webpage.

The front-end used a styling toolkit called Bootstrap to position and place the elements onto the webpage. The backend contained a database to hold user actions, MODAPTS data and user information while the application was being used.

Each research participant was first given a questionnaire asking basic experiential questions relating to MODAPTS. Once the questionnaire had been completed then

the participants were given the five scenarios to read and conduct a MODAPTS study with the acquired information. A stopwatch was used to measure the total duration that each research participant took to complete the MODAPTS study for each scenario and was recorded.

The next step involved giving each participant an android device with the machine learning webpage. They each were required to sign up to use the application which was helpful as their tracked user data could be stored and accessed anytime and anywhere with aid of a local database. They each assessed the MODAPTS scenarios again this time making use of the ML App. The scenarios were built into the application and were accessed through a touch of a button. The application was able to record MODAPTS durations from the time each research participant logged onto the application until the time they had signed off.

The data generated from the application by the participants were collected and simultaneously sent to a database preparing them for analytical studies.

Acquiring this information required building an ANN chatbot to interact with the user, developing a web interface for that chatbot and finally, employing elements in the application that would keep record of the statistics needed to define that potential improvement.

There were setbacks in acquiring some primary data but ultimately scenarios were created to facilitate that shortfall. The prerequisites of the experiment took time to compile but was necessary to extract crucial information about the research participants. All these elements individually contributed to finding the total time to compare MODAPTS and the ML approach.

2.3 Testing the Chatbot Neural Network

Testing the chatbot involved creating a python file that would access the saved model, submit the user query, and give the prediction. Each prediction was assessed based on the intents file, if an error occurred then the intents file was adjusted, and the model was re-trained. This repeated action was done on the terminal of the IDE and then applied to the UI once the testing was completed and chatbot acceptable for use. Figure 2 shows the testing code snippet.

3 Results and Discussion

The section is divided into two parts, first part focused on the results and analysis of the Machine Learning App approach of conducting the same MODAPTS study while the last part compared the two approaches in relation to the final duration with aid of a t-test analysis.

```
def get_response(intents_list, intents_json):
    tag = intents_list[0]
    list_of_intents = intents_json["intents"]
    for i in list_of_intents:
        if i["tag"] == tag:
            results = random.choice(i["responses"])
            break
    return results

def get_predictions(message):
    intents = pred_class(message, words, classes)
    results = get_response(intents, data)
    # need to extract message to get number
    number = re.findall(r'[0-9]?[0-9]\.[0-9]+\.[0-9]+', message)
    if "W" in results or "H" in results or "E" in results or "S" in results or "R" in results:
        if number:
            return f"{number[0]}{results}"
        else:
            return f"!{results}"
    return results
```

Fig. 2 Code snippet for testing the NN chatbot

3.1 Error Accumulation

Error accumulation monitors all errors generated by the research participants while making use of the application. The frequency of each error correlates to a difficulty in use of the application or a particular deficiency in understanding the MODAPTS language from the research participant. Errors monitored included: editing an activity, deleting an activity, input length error and MODAPTS code error.

Table 3 depicts the error entries made by each research participant throughout all scenarios.

There was a total of thirty-two errors done by the research participants. None of these errors contributed significantly to the functionality of the application but impacted the total time. The error for “description length” was encountered the most, a total of nineteen times appearing in four of the five scenarios. The error was a timestamp recorded each time the prediction button was pressed with no activity added. This mistake was expected as the prediction button needed to be pressed to activate the chatbot. Figure 3 shows a clustered column graph of the accumulated application errors.

Table 3 Error entities by scenario

Activity	SC1	SC2	SC3	SC4	SC5	TTL
Code			3		1	4
Delete			2	1	2	5
Description length	5	7	2	5		19
Activity		2	1	1		4
Total	5	9	8	7	3	32



Fig. 3 Application errors

3.2 Total Duration

The total duration is a measure that analyses the total time it takes to conduct a MODAPTS study using the NN application. A timestamp is added each time the research participant log’s in or sign’s up into the NN application. Another timestamp is added each time a research participant logs out of the application. The two timestamps are the subtracted from each other to gain the total duration through by scenario.

Table 4 shows a snippet of the entries entered each time a user log’s in, sign’s in and log’s out of the application.

The total average time each research participant took to perform a study using the chatbot NN was 11.2 min. Scenario 5 displayed longest time, but this was caused by an outlier from research participant 4 with the higher duration of 21 min. The lowest recorded total average and duration was scenario 1, this was expected as the scenario had the least number of activities, and its aim was to gently introduce the participants to the chatbot application.

Table 4 Activity durations

Participant	SC1	SC2	SC3	SC4	SC5	TTA	TTD
1	6	9	10	19	11	11	55
2	6	9	7	13	10	9	45
3	7	9	6	6	11	7,8	39
4	9	22	20	13	21	17	85
Total AVR	7	12,25	10,75	12,75	13,25	11,2	
Total duration	28	49	43	51	53		224

3.3 Discussion of the Traditional and ML Approach Results

The research participants reacted well to conducting a MODAPTS study under traditional means, this was expected as it is the norm. Introducing the ML application was at first a learning curve but gradually became easier through each scenario as shown by a 41.2% decrease in App navigation time from each participant. Hence, the notion mentioned by Wu et al. [5] with regards to a learning deficiency that new engineers face when presented with MODAPTS becomes less challenging and more manageable during study creation. A novice engineer can focus on describing the activity rather than the complex steps involved in MODAPTS code formulation.

The neural network was able to accurately predict the MODAPTS code of 94.7% of the 262 activities entered by the research participants. This is in line with the findings of existing studies that have indicated the suitability of the neural network for predictive purpose [15–17] This aspect saved a considerable amount of time that the research participants would have used to formulate the MODAPTS code. The total average time each research participant took to perform a study using the chatbot NN was 11.2 min. Furthermore, when looking at phase 3 of the framework developed by Kumar et al. [18], activity formulation becomes simpler due to the library of keywords that initially and consistently get added through the continuous use of the ANN.

The App alternative proved to be 0.25 min faster than the traditional approach when used under the same conditions. The difference may be small, but the ML approach performed well to produce equal results under first time users as compared to the already established excel template used by an adept participant. This was echoed by the approach followed by Mallembakam [1] and Wu et al. [5] who implemented innovative tools along with the MODAPTS time standard to match and even reduce the duration of conducting MODAPTS time studies. This proves that MODAPTS can be paired external solutions to better the performance of the time standard.

The ML chatbot however had challenges understanding some activities as it had a deficiency when deciphering a particular group of MODAPTS code. The group consisted of the element's "R", "E" and the finger movement, which caused the application to record an error or forced the research participant to manually input the code forfeiting the prediction.

Furthermore, the application had no method of handling repetition when inputting an activity. This drastically increased the time that each research participant spent when using the ML alternative during the experiment. All code related errors have been fixed and the potential of decreasing the final ANN time is evident when the consistent repetition is removed from the application.

4 Conclusion

The objective of this study was to test the neural network using operational scenarios against the traditional MODAPTS observation approach. This was achieved by packaging the created ANN chatbot template into an html page and handing that to the research participants to use and test. Five written scenarios were also used to traditionally create the MODAPTS time study using four activities. These two approaches were compared with each other to find the most effective and fastest method.

Furthermore a user-friendly and workable ANN MODAPTS template was developed. This was achieved as the chatbot was packaged onto an HTML page and deployed onto a mobile device. The template was built to be user friendly geared for the research participant as it made use of a styling framework called bootstrap. The results obtained indicated that the ANN was found to be 0.25 min faster at a prediction rate of over 90% when the chatbot was in use. The result also showed that machine learning could indeed be used with MODAPTS to equal performance and potentially improve the use of the time standard. The neural network was able to accurately predict the MODAPTS code of 94.7% of the 262 activities entered by the research participants. This saved a considerable amount of time that the research participants would have used to formulate the MODAPTS code. The total average time each research participant took to perform a study using the chatbot NN was 11.2 min. The activity formulation becomes simpler due to the library of keywords that initially and consistently get added through the continuous use of the ANN.

The time difference may be small, but the ML approach performed well to produce equal results under first time users as compared to the already establish excel template used by an adept participant.

The ANN approach makes it far easier to implement MODAPTS in multiple industries due to the chatbots customisable keywords and dialect recognition. A wealth of MODAPTS data is far more easily reachable as the template would be online to be used. The results show that MODAPTS ML template replaces tedious MODAPTS analysis done on the computer with a mobile alternative that consistently records user and study analytics for the betterment of overall analysis. The template has great potential for improvement and refinement when looking at efficiency, it makes data sharing amongst engineers easy due to its descriptive database.

The potential to add other ML learning techniques and time study methods exists, the template is flexible enough to be moulded into a tool that all engineers can adapt in their different working environment not just only automotive. Future works can test the neural network against the traditional MODAPTS observation approach using other operational scenarios for more performance evaluation.

Acknowledgements Funding: “The authors disclosed receipt of the following financial support for the research: Technology Innovation Agency (TIA) South Africa, Gibela Rail Transport Consortium (GRTC), National Research Foundation (NRF grant 123575) and the Tshwane University of Technology (TUT).”

References

1. Mallembakam VR (2020) Incorporating modular arrangement of predetermined time standard with a wearable sensing glove. University of Windsor, Canada
2. Aissani N, Beldjilali B, Trentesaux D (2008) Use of machine learning for continuous improvement of the real time heterarchical manufacturing control system performances. *Int J Ind Syst Eng* 3:474–497
3. Alpaydin E (2020) Introduction to machine learning. MIT Press
4. Brownlee J (2017) Deep learning for natural language processing: develop deep learning models for your natural language problems. *Machine Learning Mastery*
5. Wu S, Wang Y, Bolabola JZ, Qin H, Ding W, Wen W, Niu J (2016) Incorporating motion analysis technology into modular arrangement of predetermined time standard (MODAPTS). *Int J Ind Ergon* 53:291–298
6. Donkers T, Loepp B, Ziegler J (2017) Sequential user-based recurrent neural network recommendations. In: Proceedings of the eleventh ACM conference on recommender systems, pp 152–160
7. Manaswi NK (2018) Developing chatbots. In: Deep learning with applications using Python. Springer
8. Zaremba W, Sutskever I, Vinyals O (2014) Recurrent neural network regularization. arXiv preprint [arXiv:1409.2329](https://arxiv.org/abs/1409.2329)
9. Landset S, Khoshgoftaar TM, Richter AN, Hasanin T (2015) A survey of open source tools for machine learning with big data in the Hadoop ecosystem. *J Big Data* 2:24
10. Brandtzaeg PB, Følstad A (2017) Why people use chatbots. In: International conference on internet science. Springer, pp 377–392
11. Lishchynska D (2017) What are bots, how do chat bots work? What are bots, how do chat bots work? [Online]. Available from: [https://botscrew.com/blog/what-are-bots/#:~:text=Chatbot%20or%20bot%20%E2%80%93%20is%20a,an%20instant%20pre%2Dset%20answer](https://botscrew.com/blog/what-are-bots/#:~:text=Chatbot%20or%20bot%20%E2%80%93%20is%20a,an%20instant%20pre%2Dset%20answer.). [Accessed 02/01 2021]
12. Nelson D (2021) Text generation with Python and Tensor/Flow. Available from: <https://stackabuse.com/text-generation-with-python-and-tensorflow-keras> [Accessed 07/30 2021]
13. Pykes K (2021) Build a simple chatbot in Python with deep learning. Available from: <https://towardsdatascience.com/a-simple-chatbot-in-python-with-deep-learning-3e8669997758> [Accessed 31/03/2021 2022]
14. Stewart JR (2002) Applying MODAPTS standards. *Soc Work Sci (SWS)* 1:1–4
15. Adesina OT, Jamiru T, Daniyan IA, Sadiku ER, Ogunbiyi OF, Adesina OS, Beneke LW (2020) Mechanical property prediction of SPS processed GNP/PLA polymer nanocomposite using artificial neural network. *Cogent Eng* 7(1720894):1–17
16. Daniyan IA, Mpofu K, Tlhabadira I, Ramatsetse BI (2021) Process design for milling operation of titanium alloy (Ti₆Al₄V) using artificial neural network. *Int J Mech Eng Rob Res* 10(11):601–611
17. Daniyan IA, Bello EI, Ogedengbe TI, Mpofu K (2020) Use of central composite design and artificial neural network for predicting the yield of biodiesel. *Procedia CIRP* 89:59–67
18. Kumar R, Charak A, Thakur G (2020) Productivity improvement of an automotive assembly line using modular arrangement of predetermined time standards (MODAPTS). *i-Manager's J Future Eng Technol*, 16, 32