



International Journal of Innovative Research in Science Engineering and Technology (IJIRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)



Impact Factor: 8.699

Volume 14, Issue 4, April 2025

Software Behaviour Prediction Utilizing Multi-Label Classification and Long Short-Term Memory Networks

Palla Siva, Sanapathi Sai Sameera, Kalagarla Dileep Kumar, Madagala Nava Bulli Sai Ganesh,
Mittireddi Dhananjaya Rao

UG Scholar, Dept. of Computer Science Engineering (Artificial Intelligence and Data Science), Satya Institute of
Technology and Management, Vizianagaram, India

R. Swetha

Assistant Professor, Dept. of Computer Science Engineering (Artificial Intelligence and Data Science), Satya Institute
of Technology and Management, Vizianagaram, India

ABSTRACT: Program behaviour prediction is essential for confirming program attributes and guaranteeing system dependability. The interleaving of execution pathways caused by features like concurrency and resource restrictions makes it especially difficult to understand the behaviour of large software systems. As a result, it might be quite difficult to predict how such software would behave. We provide a Software Behaviour Prediction (SBP) model that fully captures and precisely forecasts software behaviour to overcome this difficulty. The SBP model increases the precision and effectiveness of recognising target states or software behaviours by utilising the machine learning ideas of Multi-label Classification (MLC) and Long Short-Term Memory (LSTM) networks. In addition, we present a threshold determination technique intended to increase MLC task accuracy. In real-world applications, our SBP model can comprehensively and iteratively forecast the states in subsequent execution path moments, doing away with the requirement to analyse the software's starting state. This feature makes it possible for our method to forecast software behaviour more accurately and effectively than the baseline models. The outcomes of the experiment show how well the SBP model predicts several simultaneous behaviours in intricate software systems. Furthermore, the SBP model outperforms the baseline approaches in terms of resource efficiency. These results demonstrate the SBP model's applicability and potential influence in real-world situations, providing encouraging paths for strengthening software design methodologies and boosting system dependability.

KEYWORDS: Software behaviour prediction, multi-label classification, long short-term memory, software design model

I. INTRODUCTION

To ensure accuracy and dependability, software design is the process of analysing, diagnosing, and comprehending software systems in order to obtain insight into their behaviour [1]. A key component of this procedure is recording software behaviour under varied conditions. Software behaviour prediction is essential for maintaining liveness, safety, and preventing deadlocks [2]. Features like concurrency and resource limitations make it harder to understand behaviour in software systems that are getting more complex, which can result in flaws, race situations, and deadlocks [3].

Formal approaches [4] define, create, and validate the software systems using mathematical models. Expert judgement uses domain knowledge to forecast software behaviour, whereas probabilistic modelling approaches use models like Bayesian networks or Markov chains to depict software behaviour, efficiently incorporating uncertainties and variances. To simulate and analyse behaviour in various contexts, a software behaviour model may use formal models like reachability graphs, Petri nets, and finite state machines.

More complex structures like concurrency and loops can be handled via the Petri net (PN) paradigm [5]. As seen in a reachability marking graph (RMG), a reachable marking of the PN acts as a snapshot of the program behaviour at a particular moment in time. However, the first step in analysing the states on the RMG is to start with an initial marking, represented by M_0 , which represents the software's initial state. To find the target states, designers then follow each of the RMG's sub-paths. Sequences of marks, beginning with M_0 and moving through M_1, M_2, \dots, M_t , where t indicates the precise time of the state, are used to depict these goal behaviours. Notwithstanding the usefulness of reachability graphs, structural complexity makes it difficult to find target states in an RMG. It might be challenging to identify consistent marking sequences when concurrent pathways lead to interleaved transitions and dependencies. The search method is further complicated by branching behaviours and cycle non-linearities brought on by the RMG's high number of fan-out nodes. As a result, identifying the goal states in the RMG necessitates navigating a complex network of potential software behaviours, which frequently causes delays and inefficiencies in the research procedure.

The adaptability and efficiency of multi-label classification (MLC) [6] have been demonstrated by its widespread use in a variety of fields, such as text classification [7-8] and sequence creation [9-12]. However, since a single input instance may be linked to several output labels, it becomes difficult to anticipate the correctness of every output label in the MLC task. A greater anticipated probability value [13], score-based and rank-based approaches [14], and a default threshold of 0.5 [15-16] are some of the approaches that have been put forth to choose the decision threshold for the MLC problem.

Machine learning methods that can handle time-series data are necessary to predict the sequential state of behaviour using past data. Sequential predictions have historically been made using recurrent neural networks (RNNs); nevertheless, they have drawbacks including the vanishing gradient issue. To forecast the sequential states of a system based on previous data, Long Short-Term Memory (LSTM) [17] networks are favoured due to their capacity to retain information across longer sequences. Furthermore, forecasting the software system's next state produces a collection of possible future states rather than a single output. Therefore, to correctly record various possible states, the MLC technique is required.

The urgent need for accurate sequential state prediction in complex systems, where each succeeding state may have several possible outputs, is what inspired this work. These difficulties are made worse by the structural complexity brought about by fan-out nodes in the RMG, which result in non-linearities that make finding target states more difficult. As a result, finding these states in the RMG requires negotiating a complex and multifaceted space of possible software behaviours, which usually causes inefficiencies and delays in the analysis procedure.

The SBP model is a software behaviour prediction model that we provide in this work with the goal of capturing and forecasting all software behaviours. To improve the accuracy and dependability of the MLC job, our suggested SBP model combines the MLC and LSTM networks and adds a threshold determination technique. The use of the J-index in the MLC framework, which offers a reliable indicator for assessing and improving the performance of the MLC models, is a distinctive feature of our methodology. This combination helps to efficiently explore software behaviour by predicting software behaviour more accurately. Petri nets are useful for modelling and organising system behaviours when using the PN model to assess program behaviour. However, an RMG is necessary in order to analyse the behaviour of complex systems. Using machine learning techniques, the SBP model was created to mimic the RMG's capabilities. SBP might forecast software behaviours without requiring direct access to the original RMG by training machine learning algorithms to anticipate and analyse software behaviours. By allowing predictions to be made at $O(1)$ speed, this method improves efficiency, especially in situations when the system does not explicitly define certain behaviours.

The SBP model, which is intended to capture and forecast all potential software behaviours and improve the effectiveness of finding target states or behaviours inside the program, starting with any state, is one of the paper's primary contributions.

- To correctly categorise states, we included machine learning methods, particularly multi-label classification (MLC). To identify lengthy data sequences in the reachability marking graph (RMG), a Long Short-Term Memory (LSTM) network is also used.

- We presented a brand-new technique for figuring out the classification threshold needed to correctly categorise states in the suggested SBP model. This approach improves the MLC task's precision and dependability, which helps produce more accurate software behaviour predictions.
- When applied to two software benchmark datasets, the experimental results show that the SBP model outperforms baseline models in terms of performance and resource efficiency. These findings demonstrate the SBP model's potential for useful applications in real-world situations by highlighting its ability to reliably forecast software behaviour and its benefit in resource-constrained contexts.

II. METHODOLOGY

In this section, we describe our proposed methodology in detail. First, we present the data preparation. We then explain the details of SBP model construction and present the threshold determination method to construct our Software Behaviour Prediction SBP, we divided our methodology into three phases, as shown in Fig. 1.

In the first phase, the dataset was prepared using target software modelling. In the second phase, an SBP model was constructed and trained using the prepared dataset. During the training process, we evaluated the learning performance of the SBP model by considering poor model performance or potential problems such as overfitting or underfitting. If problems arise, we return to the previous step of preparing a new dataset and repeat the training process until the SBP model is well-trained. Subsequently, in the MLC task, the classifying threshold was computed for the well-trained SBP model, and its performance was evaluated according to the evaluation metrics. If the model does not perform well, we would reconstruct both the dataset and the model. If the model performs well, it becomes the SBP model that is ready to use.

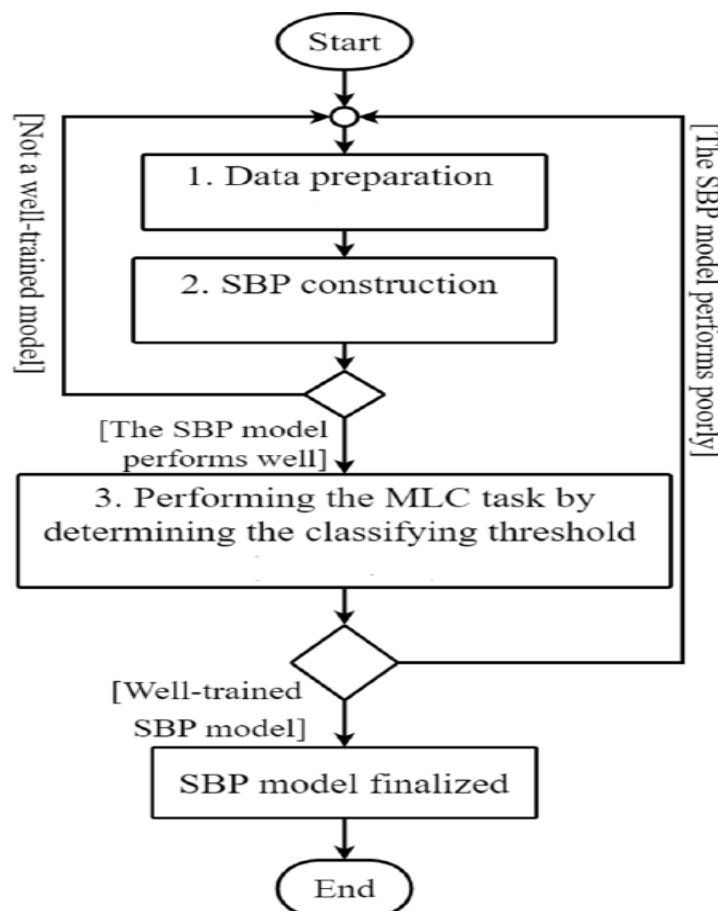


Figure 1. Proposed Framework

A. Data Preparation

In this section, we present software modelling using the RMG of the PN model, which is a fundamental tool for capturing and analyzing software behavior. We then describe the preparation process for our dataset, which involves collecting sub-paths from execution paths to create a comprehensive dataset for the machine learning model.

B. SBP Model Construction

In this section, we introduce the construction process of the SBP model, which aims to capture the intricate behavior of the software and enhance the efficiency of identifying target states in the RMG within them, leveraging principles from machine learning, specifically the MLC and LSTM networks.

III. EXPERIMENTAL FINDINGS

A. System Environment

The proposed model is developed using said hardware system configuration and software required to implement the design is shown below.

Hardware System Configuration:

- Processor: Intel i5
- RAM : 8 GB
- Hard Disk : 256 GB
- Key Board : Standard Windows Keyboard
- Mouse : Two or Three Button Mouse
- Monitor : SVGA

Software Requirements:

- Operating system : Windows 7 Ultimate.
- Coding Language: Python.
- Front-End : Python.
- Back-End : Django-ORM
- Designing : Html, css, javascript.
- Data Base : MySQL (WAMP Server).

B. Experimental Results

The shortcomings of the baseline techniques, especially BR and ML-KNN, were blamed for their poor performance. Because the BR model assumes label independence, it may ignore important label interactions that affect prediction accuracy, especially in complex datasets. Scalability and performance are impacted by ML-KNN's inability to handle high-dimensional data and its sensitivity to the number of (K) closest neighbours. Label dependencies are not captured by BR-KNN, which combines label independence with KNN instance-based learning, leading to higher complexity and processing costs. These models are less accurate even though they often use less memory. In particular, both models need substantially higher runtime for the STM dataset, even though they utilise less memory than our SBP model. In order to improve these models, it is necessary to address their inherent limitations by using techniques that increase their capacity to manage high-dimensional data for the ML-KNN model, handle label dependencies for the BR and BR-KNN models, and optimise model parameters for increased predictive accuracy in multi-label classification tasks. Our model continuously outperformed the baseline models on the FTL and STM datasets by using the temporal relationships and context information included in LSTM networks. Furthermore, by utilising a threshold to choose explicit output class labels and transforming the predicted probabilities into binary predictions for each label, our threshold determination technique improves the MLC accuracy. The processed results and user interface results are shown in below figures.



Machine learning, multi-label classification, software behavior, threshold determination...


REGISTER NOW!

REGISTER YOUR DETAILS HERE !!!

Enter Username	<input type="text" value="User Name"/>	Enter Password	<input type="password" value="Password"/>
Enter EMail Id	<input type="text" value="Enter Email"/>	Enter Address	<input type="text" value="Enter Address"/>
Enter Gender	----Select Gender ----	Enter Mobile Number	<input type="text" value="Enter Mobile Number"/>
Enter Country Name	<input type="text" value="Enter Country Name"/>	Enter State Name	<input type="text" value="Enter State Name"/>
Enter City Name	<input type="text" value="Enter City Name"/>	REGISTER	

Figure 2. Interface for New user registration

Machine learning, multi-label classification, software behavior, threshold determination....



Login Using Your Account:

user

.....


LOGIN

Are You New User !!! REGISTER

[Home](#) | [Remote User](#) | [Service Provider](#)

Figure 3. Login Interface

[Prediction Of Software Behavior Type](#) | [VIEW YOUR PROFILE](#) | [LOGOUT](#)



YOUR PROFILE DETAILS !!!

Username	siva	Email Id	sivalisivali143@gmail.com
Mobile Number	6300090126	Gender	Male
Address	vizianagaram	Country	India
State	ANDHRA PRADESH	City	Vizianagaram

Figure 4. Register user details

PREDICTION OF SOFTWARE BEHAVIOR TYPE III

ENTER ALL NETWORK DATA SET DETAILS !!!

Enter Sid	<input type="text" value="S12345"/>	Enter Software_Name	<input type="text" value="BugTracker Pro"/>
Enter Package	<input type="text" value="bugtracker_pro_v2.3.zip"/>	Enter Category	<input type="text" value="Bug Tracking"/>
Enter Description	<input type="text" value="module, and integration with JIRA."/>	Enter Rating	<input type="text" value="4.5"/>
Enter Numberofratings	<input type="text" value="1250"/>		
<input type="button" value="Predict"/>			

Predicted Software Behavior Type::

Figure 5. Prediction of software behaviour type

IV. CONCLUSION

The goal of the Software Behaviour Prediction (SBP) model put out in this study is to fully capture and forecast all software behaviour. Our model successfully captures and forecasts software behaviour by combining the MLC and LSTM networks. The approach explicitly addresses the difficulties presented by complex software systems by improving the accuracy of the MLC job using a threshold determination mechanism. In comparison to baseline models like BR, ML-KNN, and BR-KNN, the experimental findings indicated that our SBP model offered noteworthy improvements, including large reductions in Hamming Loss (HL) and notable gains in Micro-F1 (MF1) scores across two sophisticated software benchmark datasets. For sequence prediction tasks, the LSTM-ATT model works quite well, although it uses more processing power. In contrast, the SBP model maintains its competitive edge and offers a resource efficiency advantage, showcasing its ability to match computational restrictions with performance and its resilience in accurately determining thresholds. For situations where accuracy and resource efficiency are crucial, the SBP model is therefore the recommended option. Finally, the SBP model offers an approach that can be tailored to software systems and represents a notable advance in forecasting individual software behaviours. This adaptability guarantees that the behaviour of diverse software systems is accurately captured by the SBP model. In further research, we will investigate more uses and expansions of the SBP model to improve its capacity to identify the characteristics of software systems and validate software behaviour. Furthermore, to better forecast complex interaction dependencies across sequences, we want to use sophisticated deep learning models, like transformers, to understand the software behaviour of large-scale systems. These initiatives will improve our comprehension and modelling of complex software behaviours, leading to more reliable and expandable solutions.

REFERENCES

1. Sommerville, *Software Engineering*, vol. 137035152, 9th ed. London, U.K.: Pearson, 2011.
2. J. Hatcliff, M. Dwyer, and H. Zheng, "Slicing software for model construction," *Higher-Order Symbolic Comput.*, vol. 13, pp. 315–353, Jun. 2000.
3. G. R. Andrews, *Concurrent Programming: Principles and Practice*. San Francisco, CA, USA: Benjamin-Cummings, 1991.
4. E. M. Clarke, O. Grumberg, D. Peled, and D. A. Peled, *Model Checking* (The Cyber-Physical Systems Series). Cambridge, MA, USA: MIT Press, 1999.
5. B. W. Choi, *Petri Net Approaches for Modeling, Controlling, and Validating Flexible Manufacturing Systems*. Ames, IA, USA: Iowa State Univ., 1994.

6. A. C. de Carvalho and A. A. Freitas, “A tutorial on multi-label classification techniques,” *Found. Comput. Intell.*, vol. 5, pp. 177–195, Jul. 2009.
7. G. Chen, D. Ye, Z. Xing, J. Chen, and E. Cambria, “Ensemble application of convolutional and recurrent neural networks for multi-label text categorization,” in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, 2017, pp. 2377–2383.
8. H. Ma, Y. Li, X. Ji, J. Han, and Z. Li, “MsCoa: Multi-step coattention model for multi-label classification,” *IEEE Access*, vol. 7, pp. 109635–109645, 2019.
9. P. Yang, X. Sun, W. Li, S. Ma, W. Wu, and H. Wang, “SGM: Sequence generation model for multi-label classification,” 2018, *arXiv:1806.04822*.
10. S. Song, H. Huang, and T. Ruan, “Abstractive text summarization using LSTM-CNN based deep learning,” *Multimedia Tools Appl.*, vol. 78, no. 1, pp. 857–875, Jan. 2019.
11. Z. Wang, J. Poon, and S. Poon, “TCM translator: A sequence generation approach for prescribing herbal medicines,” in *Proc. IEEE Int. Conf. Bioinf. Biomed. (BIBM)*, Nov. 2019, pp. 2474–2480.
12. Z. Yang and G. Liu, “Hierarchical sequence-to-sequence model for multilabel text classification,” *IEEE Access*, vol. 7, pp. 153012–153020, 2019.
13. Marella, B. C. C., & Palakurti, A. (2025). Harnessing Python for AI and Machine Learning: Techniques, Tools, and Green Solutions. In *Advancing Social Equity Through Accessible Green Innovation* (pp. 237-250). IGI Global Scientific Publishing.
14. Z. Chen and J. Ren, “Multi-label text classification with latent word-wise label information,” *Int. J. Speech Technol.*, vol. 51, no. 2, pp. 966–979, Feb. 2021.
15. O. Gharroudi, H. Elghazel, and A. Aussem, “Ensemble multi-label classification: A comparative study on threshold selection and voting methods,” in *Proc. IEEE 27th Int. Conf. Tools with Artif. Intell. (ICTAI)*, Nov. 2015, p. 29.
16. Y. Hua, L. Mou, and X. X. Zhu, “Recurrently exploring class-wise attention in a hybrid convolutional and bidirectional LSTM network for multi-label aerial image classification,” *ISPRS J. Photogramm. Remote Sens.*, vol. 149, pp. 188–199, Mar. 2019.
17. J. Read, B. Pfahringer, and G. Holmes, “Multi-label classification using ensembles of pruned sets,” in *Proc. 8th IEEE Int. Conf. Data Mining*, Dec. 2008, pp. 995–1000.
18. S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.



INTERNATIONAL
STANDARD
SERIAL
NUMBER
INDIA



INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

 9940 572 462  6381 907 438  ijirset@gmail.com



www.ijirset.com

Scan to save the contact details