Optimizing Strategic Release Planning using NLP and Reinforcement Learning

Sai Krishna Naraharasetti
College of Engineering and Applied Science
University of Cincinnati
narahasa@mail.uc.edu

Avinash S Nandikatti
College of Engineering and Applied Science
University of Cincinnati
nandikam@mail.uc.edu

Uday Surya Deveswar Reddy Kovvuri
College of Engineering and Applied Science
University of Cincinnati
kovvuruy@mail.uc.edu

***Abstract*—This study takes a novel approach to strategic release planning by combining Q-learning with sentiment analysis, resulting in a revolutionary technique in requirements engineering. The main goal is to optimize the selection of software requirements for different release plans, guaranteeing a balanced inclusion of features based on their sentiment value. The process entails utilizing the VADER sentiment analysis tool to categorize requirements into positive and negative sentiment categories. A Q-learning model is created to meet the unique requirements of separate release plans, each with a set of features. By utilizing a multidimensional Q-table, the model is able to strategically learn and adapt its decision-making policy. The model's performance is thoroughly assessed against numerous measures, including sentiment balance and the variety of specified requirements. These measures indicate the model's capacity to pick requirements that are balanced and diversified. The findings of this study highlight the effectiveness of merging Machine Learning and Natural Language Processing in release planning, laying the groundwork for more data-driven, adaptive methods in the software development lifecycle.**

***Index Terms*—Reinforcement Learning, Q-Learning, VADER Sentiment, Natural Language Processing**

I. Introduction

In this blooming phase of the world, efficiently incorporating customer requirements into valuable software products is imperative in the dynamic realm of software development. Developing a strategic plan for the release of a widely utilized information system poses numerous challenges. Constantly evolving requirements at various levels of abstraction frequently arise and affect the inclusion of long-term selected features [1]. At the heart of this strategy lies a comprehensive process that surpasses the mere creation of a program and encompasses the formulation of a business plan as well as addressing resource and customer needs.

Startup projects without a well-defined strategic plan resemble a boat navigating turbulent waters without a compass, resulting in underwhelming outcomes that affect the efficiency and the excellence of the end product and stakeholders. An unstructured launch could cause an uneven distribution of resources, a failure to prioritize effectively, and a considerable risk of project downfall. As the software development landscape grows increasingly dynamic, the necessity for flexible and strategic planning becomes increasingly apparent. Strategic release planning, or road mapping, is a crucial part of the requirements engineering process at the product level [2]. Its objective is to choose and allocate requirements in a series of releases, ensuring that essential technical and resource limitations are met.

Engineers would be able to make more informed decisions about implementing changes, particularly in large-scale projects, if they could analyze and anticipate the development of requirements and assess their impact early on in the design process. For a decade, machine learning techniques have played a huge role in addressing issues of requirements engineering such as finding uncertainties [3]. In machine learning, reinforcement learning is the most popular type to model the release plans for a product enhancement. Reinforcement learning is very effective in dealing with the traceability of requirements and modeling an operation to make a strategic release plan [4]. Natural language processing (NLP) is showing its typical touch in requirements engineering for last two decades. The NLP techniques have been employed in identifying the importance of the requirement, recognizing important concepts within a particular field, and creating connections between requirements and these concepts [5]. The method of organizing a product's development efficiently is through the use of operative release planning in an agile environment [6].

Developing software systems that meet stakeholders' needs and expectations is the ultimate goal of any software provider. For a software system to succeed, quality must be maximized, cost minimized, and time-to-delivery be as short as possible [7]. Release planning constitutes the development of a broad plan detailing the timeline for making an accessible version of a software product and specifying the functionalities to be encompassed in that release. The ability to execute a well-defined and practical release plan can be the determinant factor between the success and failure of a software project. A release plan can prove extremely valuable to an organization. It can be used to guide and appraise the work progress but also aid in strategic planning. Crafting a plan that is both clear and realistic poses a significant challenge due to the inherent uncertainty associated with factors like the time and cost of developing specific functionalities and the anticipated returns [8].

One potential strategy for release planning involves setting a predetermined release date and then allocating functionality based on the achievable workload within that timeframe. An alternative approach is selecting the desired functionalities and trying to determine a release date accordingly. Regardless of the chosen approach, the software organization must assess the anticipated value of each requirement in comparison to the expected cost and development time. Failing to account for uncertainty in these values heightens the risk of exceeding budgetary or scheduling constraints [9].

From the above methods that are employed in the various works to enhance the requirements traceability and release planning, one can understand the importance of requirements engineering in the cycle of various fields especially, in software development. Although there are distinct approaches, the studies do not focus on experimentation of the proposed plan, instead, they just represent the ideas to make enhancements in strategic release plans of requirements. Hence, any kind of experimentation and findings with technology integration for optimal tracing and release planning of requirements will be very useful for different sectors in the background of requirements engineering.

II. Related Work

In Parmeet Kaur's seminal work [10], the focus on using a reinforcement learning approach serves as a cornerstone for understanding the context of adaptive release planning in agile software development environments. The agile method which involves iterative development with evolving requirements makes the release planning challenging. Existing release planning methods follow simple statistical approaches or risk analysis. However, they do not automatically learn and adapt plans between iterations. Reinforcement learning allows an agent (the development team) to learn an optimal policy for release planning through trial-and-error interactions with the environment. The method uses Q-learning to learn state-action values and update the policy after each iteration. Over time, it converges to an optimal release planning policy. The limitations of the author's approach could include computational complexity in modeling large projects and a lack of techniques to provide human-interpretable explanations behind learned policies. This study is a primary reference to the research that was performed in this research work.

Reinforcement learning is used to dynamically set lead times for order release in production planning that adapts based on system feedback [11]. Order release planning decisions affect key objectives like balancing workloads, and minimizing inventory, and backorders. The Markov decision process (MDP) provides a sound framework, with states capturing system conditions, and actions for lead time updates. The work demonstrates machine learning can support complex dynamic planning tasks for production systems.

The paper, "Dynamic job-shop scheduling using reinforcement learning agents" [12], presents an intelligent agent-based dynamic job-shop scheduling system utilizing reinforcement learning. The agent dynamically selects dispatching rules based on current shop conditions for real-time scheduling of incoming jobs. The proposed Q-III learning algorithm enhances the generalization of experiences, training the agent to improve scheduling policies over time. The simulation environment generates jobs, sequences operations, and manages events, with the agent making decisions based on three rules (SPT, COVERT, CR). The integration of ML models for automated time estimation and resource allocation, to provide insights to the managers to guide them in release planning. This helps in taking time-critical actions and preventive measures for future expected risks [13].

The modern Machine learning approach is utilized in optimizing order lead time planning, challenging traditional static lead time assumptions. Utilizing an artificial neural network (ANN), the study demonstrates superior flow time forecast accuracy compared to traditional methods. [14] The ANN approach significantly reduces total costs, particularly in high-utilization scenarios, by minimizing backorder and finished goods holding costs. While some scenarios show higher WIP costs, implementing constraints on release quantities helps mitigate this issue. Overall, the research highlights the potential of machine learning in optimizing dynamic lead time planning.

Reinforcement learning [15] is predominantly recognized in machine learning research as a framework where agents acquire the ability to select the optimal action for each situation or state they encounter. The agent is assumed to be in a specific state, undertaking actions in each step, leading to a transition to another state. Following each transition, the agent is rewarded with R(s). The objective is for the agent to learn the actions to execute in each state, maximizing the cumulative reward on its path to the goal state. The collection of actions chosen in each state constitutes the agent's policy. A variant of this approach is Q-Learning, wherein the agent does not calculate explicit values for each state but computes a value function Q(s,a), denoting the value of performing action 'a' in state 's'. Formally, the value of Q(s,a) is the discounted sum of future rewards attainable by executing action a in s and subsequently selecting optimal actions. This method is primarily concerned with estimating an evaluation of performing specific actions in each state, known as Q-values. To address the challenges with Q-Learning, it is necessary to define states and actions appropriately, establish a reward function suitable for the problem, and design a procedure to train the system.

The motivation for the discussed approach arises from the need to create a robust, flexible, and responsive strategic release planning. Leveraging the adaptability of Reinforcement Learning (RL), particularly the Q-learning algorithm, forms the core of this study. RL allows for dynamic prioritization of features based on real-time feedback and interactions, ensuring that release plans remain agile in the face of evolving project goals and user preferences. Integrating sentiment analysis provides an initial understanding of user sentiments associated with each requirement. This sentiment awareness is invaluable for ensuring that the release plans align with user expectations and preferences.

Without sentiment analysis, the system might overlook the nuanced preferences and sentiments of users, potentially leading to suboptimal release plans. A crucial facet is the assimilation of compound sentiment scores into the RL algorithm. These scores serve as a key input to the Q-learning process, enriching the learning mechanism unlike just prioritizing the features based on length or textual familiarity of the feature descriptions in release plans. This combined approach not only adapts to the changes but also captures the subjective user experience more comprehensively. The solution aims to handle unforeseen changes and novel features by allowing RL to explore new possibilities. Striking a balance between user satisfaction and project objectives, the combined approach seeks to deliver strategic release plans that are not only technically sound but also align closely with user expectations.

III. Proposed Methodology

A substantial difficulty comes in ensuring that the specified requirements are not only well-defined but also strategically planned and released. The lack of a strong strategic release planning process frequently results in several challenges that threaten a project's overall success. This issue statement intends to throw emphasis on the vital relevance of strategic release planning in requirements engineering.

Reinforcement learning is a subset of machine learning that focuses on discovering the best actions to maximize rewards in a certain scenario. It is commonly used in software and computers to determine the best effective behavior or course of action in particular scenarios. It entails an agent interacting with its environment, gaining rewards for actions that contribute to a goal, and receiving negative consequences for actions that move away from the goal. Unlike supervised learning, which uses labeled data with explicit answers to train models, reinforcement learning works without a predetermined response key. Instead, the reinforcement agent decides its activities to complete a particular job. Reinforcement learning, which does not require a training dataset, depends on experiential learning to improve its performance.

Natural language processing is a technique that is used to translate words in a way that is understandable for the computer, where it recognizes the meaning of the sentence or a word. Natural Language Processing (NLP) is a computational technology that promotes computer-human communication by allowing machines to comprehend and may be used to identify the relevance of phrases within a given text. The purpose of extractive summarizing is to locate and extract the most relevant and crucial lines from a material to construct a summary.

In this study, the combination of Natural Language Processing(NLP) and Reinforcement algorithm is employed to build the Strategic Release Planner (SRP) of features driven by diverse factors of which Sentiment analysis is one of the primary foundations. Firstly, leveraging the Vader lexicon algorithm, the significance of each feature is identified by analyzing the keywords in a particular feature deriving the nature of sentiment, and assigning a compound score which is calculated for each feature provided, which enables efficient release planning based on derived insights related their value using sentiment analysis. Furthermore, our feature inclusion technique utilizes the capabilities of a Q-learning algorithm to choose actions (0 for non-selection, 1 for selection) for each plan and requirement, achieving a balance between random exploration and exploitation of the greatest Q-value. A complex reward mechanism offers positive incentives for satisfying criteria and negative rewards otherwise, impacting the inclusion of features in the release plan. During training, the Q-table is constantly updated depending on actions made and rewards observed. The Q-table directs the selection process after training, offering a balanced release plan by minimizing repetition and preserving sentiment equilibrium with a mix of highly essential, operational, and good-to-have features.

A. *VADER Sentiment Analysis*

In this study, the VADER Lexicon method is utilized to find the importance of a specific requirement and to use the feature importance score to allocate features in a specific release plan. The sentiment lexicon used by the Valence Aware Dictionary and sentiment Reasoner (VADER) is referred to as the Vader Lexicon. This lexicon is essentially a dictionary that has a complete collection of terms and their corresponding sentiment polarity scores, which range from extremely negative to extremely positive. VADER, which was created for social media text analysis, combines this vocabulary with a set of rules to capture sentiment details, taking into account elements such as capitalization, punctuation, and negation. For a given text, the system computes an overall sentiment intensity score, allowing it to identify the sentiment as positive, negative, or neutral. The Vader Lexicon is critical to VADER's capacity to swiftly and efficiently assess feelings in a variety of textual material, making it especially ideal for applications such as sentiment analysis in social media.

B. *Q-Learning Algorithm*

Q-learning is a reinforcement learning strategy, that selects actions at random to maximize rewards to find the next ideal action based on the current state. The model aims to determine the most advantageous course of action given its current situation. It can make its regulations or go beyond the authorized policy since it needs to properly adhere to a set of principles. The Q-learning algorithm comprises five major elements as shown in Fig. 1.

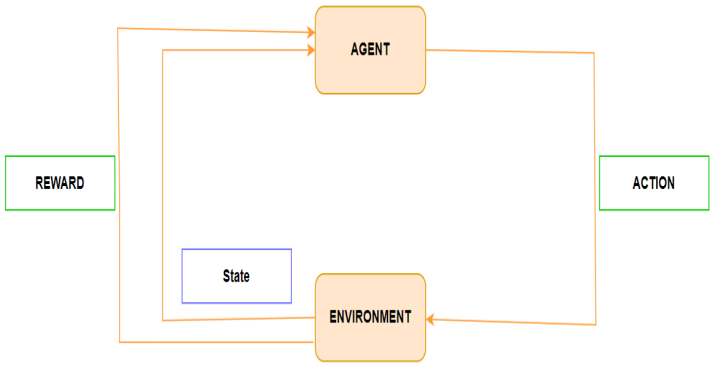
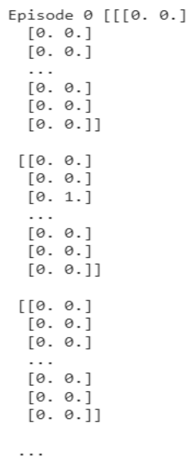


Fig. 1. Elements of Q-Learning

The Q-learning method begins by demonstrating the current state of the system and identifying the set of feasible actions inside that state. This state-action space has significance for decision-making by the agent. A vital data structure is the Q-table, which stores Q-values that estimate the predicted cumulative rewards associated with each action in a given condition. The agent's decision between known actions with high Q-values and investigating new options is influenced by the agent's ability to balance exploration and exploitation. The learning rate and discount factor refine the Q-values further, influencing the model's flexibility and consideration of future rewards.

$$Q(state, action) = Q(state, action) + \alpha \times (reward + \gamma \times \max(Q(next_{s}tate, all_{a}ctions)) - Q(state, action)) \quad (1)$$

By providing incentives for each action, the reward function gives rapid feedback, altering the learning process. Exploration strategies, such as epsilon-greedy, encourage the agent to explore at random with a particular probability, enhancing the learning experience. The stage is created via proper initialization, and an update rule refines the Q-values depending on observed rewards and the Q-values of succeeding states. These interrelated pieces form a logical foundation for the Q-learning algorithm, allowing it to develop optimum policies in reinforcement learning repeatedly.



```
Episode 0 [[[0. 0.]
  [0. 0.]
  [0. 0.]
  ...
  [0. 0.]
  [0. 0.]
  [0. 0.]]

 [[0. 0.]
  [0. 0.]
  [0. 1.]
  ...
  [0. 0.]
  [0. 0.]
  [0. 0.]]

 [[0. 0.]
  [0. 0.]
  [0. 0.]
  ...
  [0. 0.]
  [0. 0.]
  [0. 0.]]

 ...
```

Fig. 2. Initial State of Q-Table

C. *State Space, Action Space, and Reward*

In our research, the state space serves as the entire decision-making environment in our study scenario for deciding the inclusion of features, or needs, in release plans. Each distinct stage inside this space corresponds to the consideration of a particular feature for inclusion in a certain release plan. Given the "n" features and "x" release plans, the state space contains a wide range of possibilities, resulting in n * x multiplied by "x" possible states when each feature for each plan is considered independently.

The action space, which is naturally linked to decision-making, consists of two unique actions for each state: deciding not to include the current feature in the current release plan (marked as 0) or incorporating the feature in the present release plan (marked as 1). The reward function, precisely designed to motivate feature acquisition while maintaining a fragile equilibrium of sentiments within each release plan, further guides the complexities of decision-making. This function works by awarding positive incentives when the introduction of a feature adds to or maintains a harmonic balance of positive and negative thoughts. The penalty for imbalance, on the other hand, is applied if the addition of a feature upsets this equilibrium, resulting in a negative reward. This organized framework not only outlines the complicated dynamics of decision-making in our study scenario but also lays the groundwork for an in-depth investigation of the interaction between features and sentiments in the release planning process.

D. *Q-table Learning Policy*

The development of the state space, action space, and reward functions is an important component of our research work. Following the basic structure, the subsequent focal point converges on model training, with particular importance given to the role of the Q-table. This matrix, which is a key component of reinforcement learning, captures the interaction between states and actions, with each entry indicating the predicted utility of a single action inside a given state. The learning process of the algorithm is represented in Fig. 3.

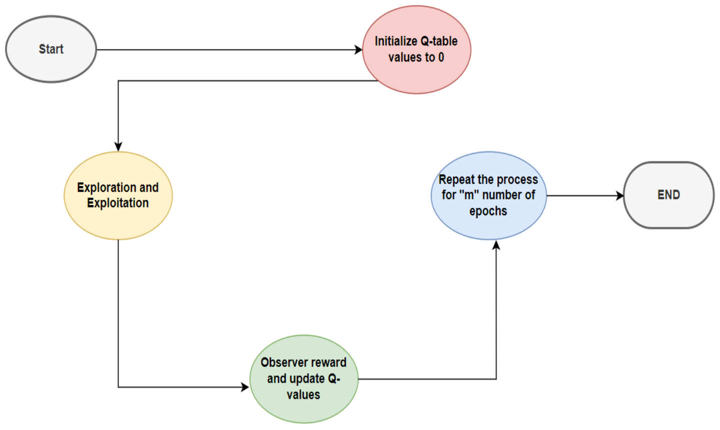

Start
Initialize Q-table values to 0
Exploration and Exploitation
Repeat the process for "m" number of epochs
END
Observer reward and update Q-values

Fig. 3. Training Approach for SRP

Initially, the Q-table is initialized, with values set to zero to indicate a lack of past knowledge. The iterative training takes place over epochs, and the algorithm navigates the decision-making environment strategically, seamlessly merging exploration and exploitation techniques where it randomly picks up the features from a set driven by current Q-values. The Q-values are dynamically changed after each action, reflecting perceived immediate rewards and aligning with the Bellman equation(Formula 1) integrating critical factors like the learning rate and the discount factor. This recurrent refinement culminates in the convergence of Q-values across several epochs, providing the model with a nuanced and optimal approach for feature selection that is perfectly linked with the given reward function.

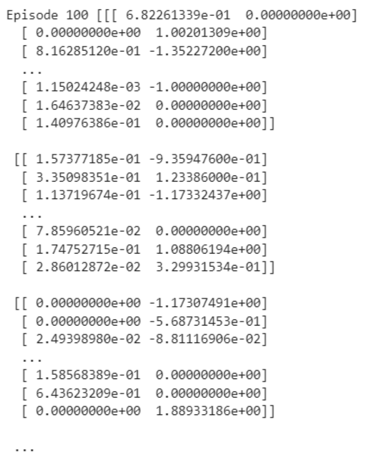
IV. Results and Analysis

A. Data Characteristics

In this study, the data referenced was of three variants of requirements about Feature Releases made by diverse hosting platforms like Zoom, Webex, and Discord. The primary source of this data was an Excel file that comprised a summary page documenting the overall number of requirements issued in each plan, as well as additional sheets listing the requirements published in each cycle throughout the year. All these requirement descriptions are provided in Natural Language which formed the basis of input for our Reinforcement Learning Algorithm along with Sentiment Analysis.

B. System Authenticity

Some of the critical factors that make the proposed Strategic Release Planner solution stand out from the other research are Integration of Natural Language Processing to compute Compound Sentiment scores using Sentiment Analysis, Training (Updating) the data in Q-Table based on formulating the Equilibrium for each release plan employing Homogenous extraction of requirements ranging from the ones that mark on great customer experience to selecting good-to-have requirements in each plan to build a holistic, concrete release plan that impacts on overall selection.



```
Episode 100 [[[ 6.82261339e-01  0.00000000e+00]
  [ 0.00000000e+00  1.00201309e+00]
  [ 8.16285120e-01 -1.35227200e+00]
  ...
  [ 1.15024248e-03 -1.00000000e+00]
  [ 1.64637383e-02  0.00000000e+00]
  [ 1.40976386e-01  0.00000000e+00]]

 [[ 1.57377185e-01 -9.35947600e-01]
  [ 3.35098351e-01  1.23386000e-01]
  [ 1.13719674e-01 -1.17332437e+00]
  ...
  [ 7.85960521e-02  0.00000000e+00]
  [ 1.74752715e-01  1.08806194e+00]
  [ 2.86012872e-02  3.29931534e-01]]

 [[ 0.00000000e+00 -1.17307491e+00]
  [ 0.00000000e+00 -5.68731453e-01]
  [ 2.49398980e-02 -8.81116906e-02]
  ...
  [ 1.58568389e-01  0.00000000e+00]
  [ 6.43623209e-01  0.00000000e+00]
  [ 0.00000000e+00  1.88933186e+00]]

 ...
```

Fig. 4. Intermediate State of Q-Table (after 100 episodes)

C. Performance Evaluation Callouts

Several major indicators of performance are important in the evaluation of a Q-learning model used for optimizing software release plans based on requirements selection. The first measure, Balance, is essential for evaluating the model's capacity to maintain a balance of positive and negative sentiment needs within each release plan. It is measured by comparing the number of favorably and negatively indicated emotion needs by two, with a lower difference suggesting a better balanced plan. The second measure, Diversity, assesses the model's efficacy in preventing the same needs from being repeated across multiple release plans. This is measured by calculating the number of distinct needs chosen across all plans, with a higher number indicating greater variety.

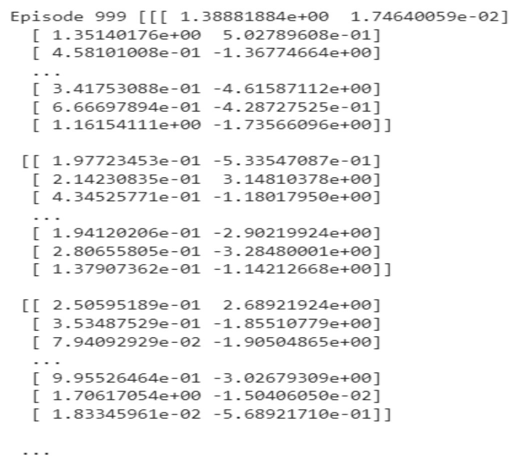
Furthermore, the Coverage measure is used to guarantee that each release plan satisfies its needs boundaries. This is derived by comparing the number of requirements chosen for each plan to the plan's specified requirement count threshold, represented as a percentage of plans that meet the required number of features. Finally, the Reward Convergence measure is useful for tracking the Q-learning model's learning process. It entails recording cumulative rewards throughout subsequent training sessions, with the convergence of these values suggesting a stabilization in the model's decision-making process. Together, these metrics provide a complete framework for evaluating the Q-learning model's efficacy and efficiency in optimizing software release planning through balanced and diversified requirement selection. The hyperparameters used in building the Q-Table are mentioned in Table 1. The

TABLE I
VALUES OF LEARNABLE PARAMETERS

S.no	Hyperparameters	Range
1	Alpha	0.1
2	Gamma	0.6
3	Epsilon	0.1
4	Number of episodes	1000

Q-table is just a multi-dimensional array, with each dimension representing a different facet of the learning environment. The various release plans are represented by one dimension. Each release plan is handled in a different decision-making context. Another dimension is associated with the needs (or features). This dimension takes into consideration all of the requirements that might be included in a release strategy. The last dimension shows each state's possible actions (requirement and plan combination). In this scenario, the actions are binary: picking or not selecting a release plan need.

The Q-Table outputs will capture the dynamic, demonstrating how the Q-values vary initially - indicating the model's exploration of alternative actions and learning from their results - then subsequently converge towards more stable values for each state-action combination. According to Formula 1, the modifications to the Q-Table for each episode are determined by the observed and predicted feature rewards. The intermediate transitions of the Q-Table shown below (fig. 2, fig. 4, fig. 5) indicate the transition of various Q-Values being updated in various episodes.



```
Episode 999 [[[ 1.38881884e+00  1.74640059e-02]
  [ 1.35140176e+00  5.02789608e-01]
  [ 4.58101008e-01 -1.36774664e+00]
  ...
  [ 3.41753088e-01 -4.61587112e+00]
  [ 6.66697894e-01 -4.28727525e-01]
  [ 1.16154111e+00 -1.73566096e+00]]

 [[ 1.97723453e-01 -5.33547087e-01]
  [ 2.14230835e-01  3.14810378e+00]
  [ 4.34525771e-01 -1.18017950e+00]
  ...
  [ 1.94120206e-01 -2.90219924e+00]
  [ 2.80655805e-01 -3.28480001e+00]
  [ 1.37907362e-01 -1.14212668e+00]]

 [[ 2.50595189e-01  2.68921924e+00]
  [ 3.53487529e-01 -1.85510779e+00]
  [ 7.94092929e-02 -1.90504865e+00]
  ...
  [ 9.95526464e-01 -3.02679309e+00]
  [ 1.70617054e+00 -1.50406050e-02]
  [ 1.83345961e-02 -5.68921710e-01]]

 ...
```

Fig. 5. End-State of Q-Table

Developing proper baselines is critical for a full evaluation when utilizing a Q-learning model for strategic release planning based on requirement sentiments. This is a distinct approach to employing Q-learning in requirements engineering, particularly with sentiment analysis. The given source of requirements and their release plans don't constitute any additional factors that influence their order of release such as requirement priority or sentiment associated etc. Having such a mechanism that monitors the overall sentiment using a robust Natural Language Processing Technique such as Sentiment Analysis gives the strategic release planner a better understanding of the requirements in the queue when combined with the Reinforcement Learning process, better planning can be executed where RL can maintain a homogenous execution of requirements in each release which are complex, highly critical and good to have customer asks and maintain utmost equilibrium in executing them. Therefore, sentiment precheck and RL training will elevate the quality of planning in the longer run.

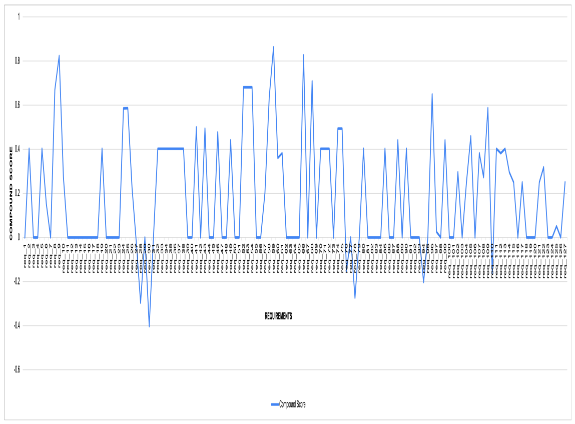


Fig. 6. Before Learning Process

From the above 6, the insights derived affirm that the produced data points before leveraging Reinforcement Learning (RL) are static and not balanced. The pattern observed is the formation of continued identical compound scores across certain groups of requirements in the given plan. This gives rise to a homogenous flat line group in the resultant visualization. After applying the learning algorithm, the resulting 7 shows a varied sinusoidal pattern with fewer consistent flat lines, indicating a mix of requirements with distinct compound scores in each plan. This suggests an amalgamation of essential, functional, and desirable requirements has been achieved.

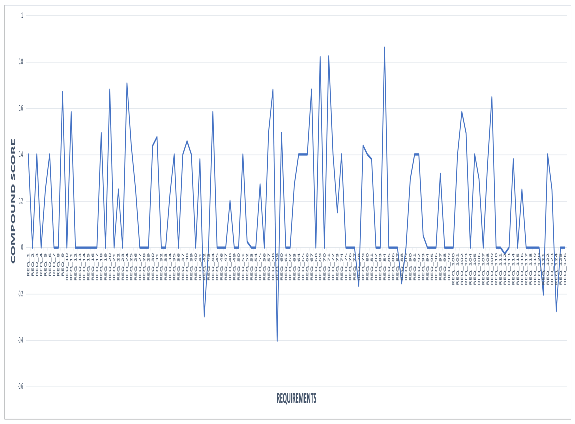


Fig. 7. After Learning process

V. Limitations

The study combining Q-learning and sentiment analysis for software release planning has limited constraints. The model's reliance on high-quality data is crucial to fully grasp and prioritize complicated features beyond disclosed sentiment. The computational needs of the Q-learning method also offer substantial scaling issues, particularly in complex development environments with a huge volume of requirements and interdependencies. Such computational difficulties become more obvious in instances where customers or stakeholders impose explicit prioritization mandates for specific qualities, forcing the model to adapt to these limits while still optimizing the selection process.

VI. Conclusion

This research demonstrates a creative integration of Q-learning with sentiment analysis in optimizing software release planning, a significant step in requirements engineering. By analyzing requirements and applying a Q-learning model across multiple release plans, the study showcases the potential of AI in complex decision-making scenarios. The model's capacity to balance sentiments and ensure diversity among requirements, as demonstrated by its developing approach across 1000 episodes, demonstrates its efficacy and flexibility. This strategy not only corresponds with stakeholder interests and project objectives, but it also opens the door to employing AI in comparable complicated environments. As a result, this study adds to the dynamic landscape of software development by highlighting the revolutionary influence of machine learning and natural language processing in requirements engineering and beyond.

References

[1] G. Zorn-Pauli, B. Paech, T. Beck, H. Karey, and G. Ruhe, "Analyzing an industrial strategic release planning process–a case study at roche diagnostics," in *Requirements Engineering: Foundation for Software Quality: 19th International Working Conference, REFSQ 2013, Essen, Germany, April 8-11, 2013. Proceedings 19*. Springer, 2013, pp. 269–284.

[2] S. Saad-Bin and S. Muhammad-Usman, "A study on strategic release planning models of academia and industry," 2008.

[3] K. Zamani, D. Zowghi, and C. Arora, "Machine learning in requirements engineering: A mapping study," in *2021 IEEE 29th International Requirements Engineering Conference Workshops (REW)*. IEEE, 2021, pp. 116–125.

[4] H. Sultanov and J. H. Hayes, "Application of reinforcement learning to requirements engineering: requirements tracing," in *2013 21st IEEE International Requirements Engineering Conference (RE)*. IEEE, 2013, pp. 52–61.

[5] L. Zhao, W. Alhoshan, A. Ferrari, K. J. Letsholo, M. A. Ajagbe, E.-V. Chioasca, and R. T. Batista-Navarro, "Natural language processing for requirements engineering: A systematic mapping study," *ACM Computing Surveys (CSUR)*, vol. 54, no. 3, pp. 1–41, 2021.

[6] T. Şahin, T. Huth, J. Axmann, and T. Vietor, "A methodology for value-oriented strategic release planning to provide continuous product upgrading," in *2020 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*. IEEE, 2020, pp. 1032–1036.

[7] J. Karlsson and K. Ryan, "A cost-value approach for prioritizing requirements," *IEEE software*, vol. 14, no. 5, pp. 67–74, 1997.

[8] K. Logue and K. McDaid, "Agile release planning: Dealing with uncertainty in development time and business value," in *15th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems (ecbs 2008)*. IEEE, 2008, pp. 437–442.

[9] K. Logue, K. McDaid, and D. Greer, "Allowing for task uncertainties and dependencies in agile release planning," in *4th Proceedings of the Software Measurement European Forum*, 2007, pp. 275–284.

[10] P. Kaur, "Reinforcement learning based approach for adaptive release planning in an agile environment," in *2010 International Conference on Computational Intelligence and Software Engineering*. IEEE, 2010, pp. 1–4.

[11] M. Schneckenreither, S. Haeussler, and J. Peiro, "Average reward adjusted deep reinforcement learning for order release planning in manufacturing," *Knowledge-Based Systems*, vol. 247, p. 108765, 2022.

[12] M. E. Aydin and E. Öztemel, "Dynamic job-shop scheduling using reinforcement learning agents," *Robotics and Autonomous Systems*, vol. 33, no. 2-3, pp. 169–178, 2000.

[13] E. Ebrahim, M. Sayed, M. Youssef, H. Essam, S. Abd El-Fattah, D. Ashraf, O. Magdy, and R. ElAdawi, "Ai decision assistant chatbot for software release planning and optimized resource allocation," in *2023 IEEE International Conference On Artificial Intelligence Testing (AITest)*. IEEE, 2023, pp. 55–60.

[14] M. Schneckenreither, S. Haeussler, and C. Gerhold, "Order release planning with predictive lead times: a machine learning approach," *International Journal of Production Research*, vol. 59, no. 11, pp. 3285–3303, 2021.

[15] R. S. Sutton and A. G. Barto, "Reinforcement learning: An introduction," *Robotica*, vol. 17, no. 2, pp. 229–235, 1999.