

ProdPrepAI

Matthew Torre, Tanaya Yadav, Arpit Ranasaria
{mtorre04, tanayayv, arpitr}@stanford.edu

This project is part of the CS230: Deep Learning course at Stanford (Fall '24), taught by Prof. Andrew Ng and Kian Katanforoosh.

Abstract

In a competitive job market, aspiring product managers face significant challenges in preparing for interviews. While existing AI-driven tools like AI for Interview and Interview Sidekick offer features such as automated feedback, customizable answers, and real-time transcription, they fall short in addressing critical gaps. These gaps include inconsistent evaluation metrics, the absence of real-time adaptability, and a lack of depth and focus tailored specifically for product-based interviews. **ProdPrepAI** fills these voids by offering a comprehensive, real-time, and dynamic interview preparation platform. Our system combines robust evaluation metrics, such as clarity, completeness, product thinking, and feasibility, with adaptable feedback mechanisms that align closely with real interview scenarios. We are focusing on providing valuable feedback to our potential users to improve their product interview skills.

1 Introduction

ProdPrepAI is an AI tool designed to assist aspiring product managers with comprehensive interview preparation. ProdPrepAI leverages question-response pairs as input, combined with evaluation labels—clarity, completeness, product thinking, and feasibility—to assess and enhance interview responses. Using a BERT-based model, the platform processes text inputs to generate embeddings and predict scores for these attributes. Built on top of this is a reinforcement learning agent that provides adaptive feedback by selecting actions such as “Ask for details” or “Reframe” or “Ask new question” guided by real-time rewards that improve response quality dynamically. This tool bridges supervised and reinforcement learning to provide tailored feedback that demystifies the product management interview preparation process and provides the candidate more insight into their current interview performance.

2 Related work

Numerous AI-driven tools for interview preparation have been developed, which can be broadly categorized into real-time coaching systems, automated analysis tools, and evaluation metric frameworks. Real-time coaching systems like AI for Interview and Interview Sidekick offer features such as real-time transcription, continuous support, question detection, and instant feedback on grammar and clarity. While these platforms provide helpful general-purpose feedback, they lack domain-specific adaptability and fail to provide tailored evaluation metrics for more dynamic contexts such as product management interviews. (1, 2). Similarly, automated analysis tools like Mock Interviewer AI enable mock interviews with feedback powered by large language models and voice-to-text systems. These tools can be effective in simulating generic interview scenarios and predicting outcomes using ML techniques. However, they focus all too much on surface-level analysis of a candidate response and does not effectively perform with the specificity and structure so vital to a successful product interview. (3)

Evaluation metric frameworks such as ComOn Check, designed to assess communication skills through structured scales, and the study by Naim et al.(2015), which uses AI to analyze both verbal and nonverbal signals to predict traits such as excitement and engagement, provide valuable insights into communication assessment. Yet, these frameworks are predominantly geared toward educational or medical contexts, where the primary focus is on improving general communication rather than addressing the nuanced needs of job interviews, especially for specialized roles like product management (5, 7).

Our work builds upon these prior approaches by addressing their key limitations. Specifically, we introduce adaptive evaluation metrics tailored to the unique demands of product management interviews. Unlike general-purpose tools, our tool provides real-time feedback that dynamically adjusts based on the context of the interview. Additionally, it offers in-depth, domain-specific insights, bridging the gap between generic evaluation systems and the specialized requirements of candidates preparing for product management roles.

3 Dataset and Features

Our dataset comprises 200 labeled examples, each consisting of a product management-related question, a corresponding response, four evaluation metrics: clarity, completeness, product thinking, feasibility, and an expected decision label for the RL Agent. These metrics are scored on a scale of 1 to 5, with higher scores representing better performance. The expected ground truth labels for each response were manually determined by our team with 0 signifying “Ask for details”, 1 as “Reframe”, and 2 as “Ask new question”. The data was sourced using web scraping from Daily Product Prep, a comprehensive website hosting product management questions and answers from various companies. The scraping process enabled us to extract a diverse set of real-world scenarios to form a robust foundation for our dataset. The data is structured as rows with features including the question and response text as well as numerical labels for the evaluation metrics.

Limitations of Our Current Data: Although the dataset is rich in quality, its size—200 examples limits its capacity for more broad generalization. Given the limitations of available data for low-scoring responses across our evaluation metrics, synthetic data generation through GPT-4o were used to fill out our dataset for completeness. Data augmentation, including synthetic data generation and question diversification do remain as viable strategies for future work to enhance dataset variability and model robustness.

To prepare the data for training, responses and questions were concatenated and tokenized using the BERT tokenizer, with inputs padded, truncated to a maximum length of 128 tokens, and converted into tensor representations. The dataset was split into an 80% training set and a 20% test set to evaluate the model’s performance effectively.

```
{
  "Question: ...." ,
  "Response: ...." ,
  "labels": {
    "clarity": 5,
    "completeness": 5,
    "product_thinking": 5,
    "feasibility": 5
  }
}
```

Semantic of the dataset

4 Methods

To access the Google Colab notebook, visit the following link: [Google Colab Notebook](#).

4.1 Model Architecture

Our implementation consists of two primary components: a BERT-based multi-label classification model and a Reinforcement Learning (RL) agent designed to provide dynamic feedback and improve user interaction during mock interviews.

4.2 BERT-based Multi-Label Classifier

The BERT-based component is a multi-label classification model that evaluates interview responses based on four metrics: *clarity*, *completeness*, *product thinking*, and *feasibility*. The architecture integrates a pre-trained BERT model (8) as the feature extractor and a fully connected feedforward layer to classify each metric into five discrete levels. Some of the key design choices are as follows:

- **Input Representation:** The input to the model is a concatenation of the interview question and response, tokenized using the BERT tokenizer. Padding and truncation ensure fixed-length sequences of 128 tokens.
- **Embedding Extraction:** Sentence embeddings are generated from the [CLS] token of the last hidden state output by BERT. These embeddings serve as input to the classification head.
- **Output:** The classifier outputs a tensor of logits, reshaped to predict scores for each of the four metrics simultaneously.

During training, the model uses a multi-class CrossEntropy loss function to optimize predictions across the four metrics. The inclusion of dropout layers mitigates overfitting, and the training loop fine-tunes BERT weights using the AdamW optimizer.

4.3 Reinforcement Learning Agent

The RL agent employs a Deep Q-Network (DQN) to help determine the next action the interviewer (the agent) should take in the interview process based on the question and response. It utilizes a Deep Neural Network to approximate the Q-value of each state/action pair in our environment, in this case the outputs of our BERT classifier being the states, and the potential next steps being the actions.

- **Observation Space:** The state vector comprises BERT-generated sentence embeddings for the question-response pair concatenated with the predicted evaluation scores for the four metrics. This multimodal representation allows the RL agent to make context-aware decisions.
- **Action Space:** The agent can choose from three actions: *Ask for Details (0)*, *Reframe (1)*, or *Next Question (2)*. These actions would then be useful in guiding the user in refining their responses and progressing them through the interview.
- **Reward Function:** Rewards are computed based on the improvement of evaluation scores after taking an action. Penalties are applied for repetitive actions or minimal improvement.

The RL training process utilizes the stable-baselines3 DQN implementation, which optimizes the policy using experience replay and exploration-exploitation trade-offs. The agent learns to maximize cumulative rewards, enabling dynamic adaptability in real-time scenarios.

4.4 Integration and Insights

The BERT classifier and RL agent work synergistically. The classifier provides quantitative evaluations of responses, while the RL agent leverages this information to iteratively guide the user toward higher-quality responses. Together, these components work together to help provide a proof of concept for a full product management interview cycle - where the model can ask a question, receive a response, evaluate the response on a variety of different scales, and utilize its knowledge to either probe for more information or move on to the next question in the interview.

5 Experiments/Results/Discussion

5.1 Training and Validation Performance

5.2 Hyperparameter Choices and Setup

The environment for training the Interview Environment (`InterviewEnv`) was initialized with a BERT model for generating embeddings and label predictions. Key hyperparameters include:

- Learning rate for BERT: $2e^{-5}$
- Total training timesteps: 10,000
- Mini-batch size: 16
- Learning rate for the DQN agent: 0.0001
- Total training timesteps: 10,000
- Exploration rate decay: 0.05
- Mini-batch size: 16

We systematically explored various hyper parameters for small amounts of epochs and saw how this affected the training loss, and arrived at these values. The training data consisted of questions and responses from a custom dataset (`data.csv`), shuffled to avoid bias.

5.3 Performance Metrics

The primary metrics used to evaluate the model’s performance include:

- Episode reward mean (ep_rew_mean)
- Average episode length (ep_len_mean)
- Loss during training
- Exploration rate

These metrics provide insight into both the learning process and the quality of the trained agent’s decision-making.

5.4 Quantitative Results

The training process was evaluated over multiple episodes. Table 3 summarizes the key results at different training stages.

Table 1: MultiClassifier Performance Metrics

Metric	Precision	Recall	F1 Score
Clarity	0.7981	0.8750	0.8274
Completeness	0.8308	0.9000	0.8591
Product Thinking	0.8839	0.8500	0.8550
Feasibility	0.7256	0.8250	0.7658

Table 2: MultiClassifier Performance Metrics

Metric	Precision	Recall	F1 Score
RL Model	0.7757	0.7350	0.7486

'Precision': 0.7757454267413008, 'Recall': 0.735, 'Accuracy': 0.735, 'F1 Score': 0.748624464540695

Table 3: Training Metrics at Various Stages

Episodes	Average Reward	Loss	Exploration Rate	Episode Length
4	-34.7	0.00552	0.495	133
12	12.2	0.00516	0.05	133
24	29.8	0.00533	0.05	133
44	37.6	0.00466	0.05	133
72	41.3	0.00532	0.05	133

5.5 Qualitative Results Analysis

The Multiclassifier model performs well at predicting extreme scores (e.g., 1 and 5) across all metrics (Figures 2-5), but it struggles with intermediate labels (3 and 4) which can likely be attributed to overlapping feature representations or insufficient training data. Some future improvements could involve diversifying our dataset, refining feature engineering, or incorporating advanced architectures to capture subtle distinctions between intermediate scores more effectively.

The learning progression of the agent demonstrated clear improvement over time. Figure 1 shows the reward trends across episodes, highlighting the model’s ability to maximize rewards effectively while maintaining stability. We note that while the RL agent was able to consistently identify when to move on to the next question, it struggled a bit more in determining when to ask for clarification versus elaboration, which is likely evident in their similar function within an interview.

Additionally, the confusion matrix in Figure 6 indicates that the classifier performs well for class 0 and class 2, as evidenced by high true positive rates. However, it struggles with class 1, where many samples are misclassified as class 0 or 2.

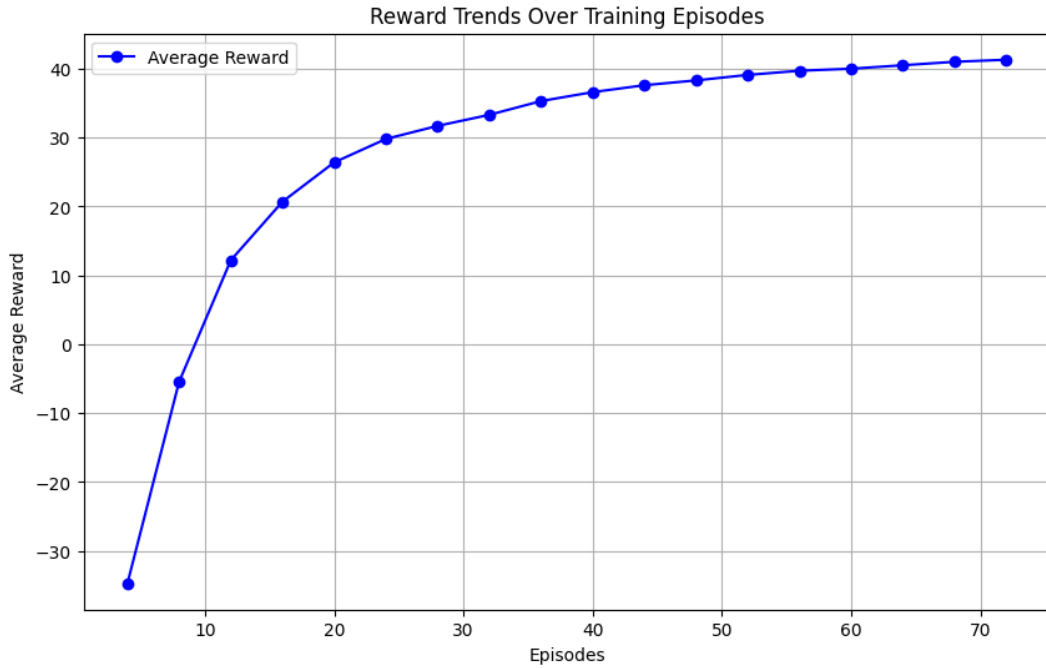


Figure 1: Reward Trends Over Training Episodes

5.6 Discussion

The model achieved a consistent improvement in rewards, as shown in the increasing `ep_rew_mean`. The use of reinforcement learning allowed the agent to learn optimal responses effectively and is reflected in the near-linear loss reduction.

Challenges included balancing exploration and exploitation during training, which was mitigated by tuning the exploration rate. The learning curves suggest that the model did not overfit the training data, as the loss values remained stable.

Future Improvements:

- Expanding the dataset to include a more diverse range of questions and responses.
- Experimenting with alternative reward functions to better reflect response quality.
- Utilizing more advanced RL algorithms, such as PPO or SAC, for improved convergence.

6 Conclusion/Future Work

It was our goal throughout the course of this project to develop a practical, AI-driven tool that is tailored to the breadth of skills that aspiring product managers must have in order to prepare for and be successful in interviews. By introducing adaptive evaluation metrics, domain-specific feedback, and real-time adaptability, our tool addresses limitations in existing platforms, bridging the gap between general-purpose tools and the specialized demands of professional interviews. BERT-based multi-label classification emerged as the highest-performing algorithm due to its transformer architecture, pretrained contextual understanding, and ability to capture semantic relationships in text, outperforming simpler models like RNNs or logistic regression. Its success is attributed to its robust handling of sequential data and the ability to generalize well on multi-label tasks with fine-tuning. For future work, exploring lightweight transformers like DistilBERT, domain-specific fine-tuning, or reinforcement learning approaches could further enhance performance and adaptability. Despite promising results, challenges such as limited training data, particularly negative responses, and constraints in computational resources and multimodal analysis hindered the model's robustness and generalizability.

For future work, expanding the dataset to include diverse positive and negative examples would strengthen the model's evaluation capabilities. One initial idea that would be a natural next step would be to incorporate a speech-to-text component, such as Whisper, would enable real-time transcription and improve accessibility, though resource limitations and dataset availability prevented its implementation in this iteration.

7 Contributions

Tanaya: Webscraping, Synthetic Data Generation, Data Collection, Research

Matthew: Model fine-tuning, hyper parameter optimization, report writing, and research.

Arpit: Multiclassifier, Reinforcement Learning Agent, Hyperparameter tuning

References

- [1] AI for Interview. (2024). AI-Driven Question Detection and Real-Time Feedback. Available at: https://www.aiforinterview.com/?utm_source=chatgpt.com.
- [2] Interview Sidekick. (2024). Customized Interview Practice and Evaluation. Available at: https://interviewsidekick.com/?utm_source=chatgpt.com.
- [3] Mock Interviewer AI. (2024). *AI-Powered Mock Interview Platform*. Available at: <https://www.mockinterviewer.ai/>
- [4] Wizco. (2024). *Tailored Interview Preparation Platform*. Available at: <https://www.wizco.io/ava/b3nkcqhqs1=1>
- [5] Weidner, N., Freudenthaler, M. (2020). *How to assess communication skills? Development of the rating scale ComOn Check*. Available at: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7011842/>
- [6] Zhang, J., Li, H., Wong, A. (2020). *AI for Nonverbal Signal Analysis in Interviews: Personality and Communication Prediction*. Available at: <https://hcis-journal.springeropen.com/articles/10.1186/s13673-020-0208-3>
- [7] Naim, I., Tanveer, M. I., Gildea, D., Hoque, M. E. (2015). *Automated prediction and analysis of job interview performance: The role of what you say and how you say it*. Proceedings of the IEEE Conference on Automatic Face and Gesture Recognition (FG), 1-8. Available at: https://roc-hci.com/wp-content/uploads/2018/06/naim_et_al_FG2015_final_version.pdf
- [8] Devlin, Jacob. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018). Available at: <https://arxiv.org/abs/1810.04805>
- [9] PyTorch. (2024). Deep Learning Framework for Python. Available at: <https://pytorch.org/>
- [10] Hugging Face Transformers. (2024). Pre-trained Transformer Models for NLP. Available at: <https://huggingface.co/transformers/>
- [11] scikit-learn. (2024). Machine Learning in Python. Available at: <https://scikit-learn.org/>
- [12] Stable-Baselines3. (2024). Reinforcement Learning Made Easy. Available at: <https://stable-baselines3.readthedocs.io/>
- [13] OpenAI Gym. (2024). Toolkit for Developing and Comparing RL Algorithms. Available at: <https://www.gymnasium.dev/>
- [14] Shimmy. (2024). Gym Wrapper for Stable-Baselines3. Available at: <https://github.com/Farama-Foundation/Shimmy>
- [15] Matthew Torre, Arpit Ranasaria, Tanaya Yadav (2024). ProdPrepAI: AI-Driven Multi-Label Classification and Reinforcement Learning for Interview Preparation. Available at: <https://colab.research.google.com/drive/1baCoMkW2M5SUI0iQaRn2hr0aLqE6MQpE?usp=sharing>

Figures BERT CLASSIFIER

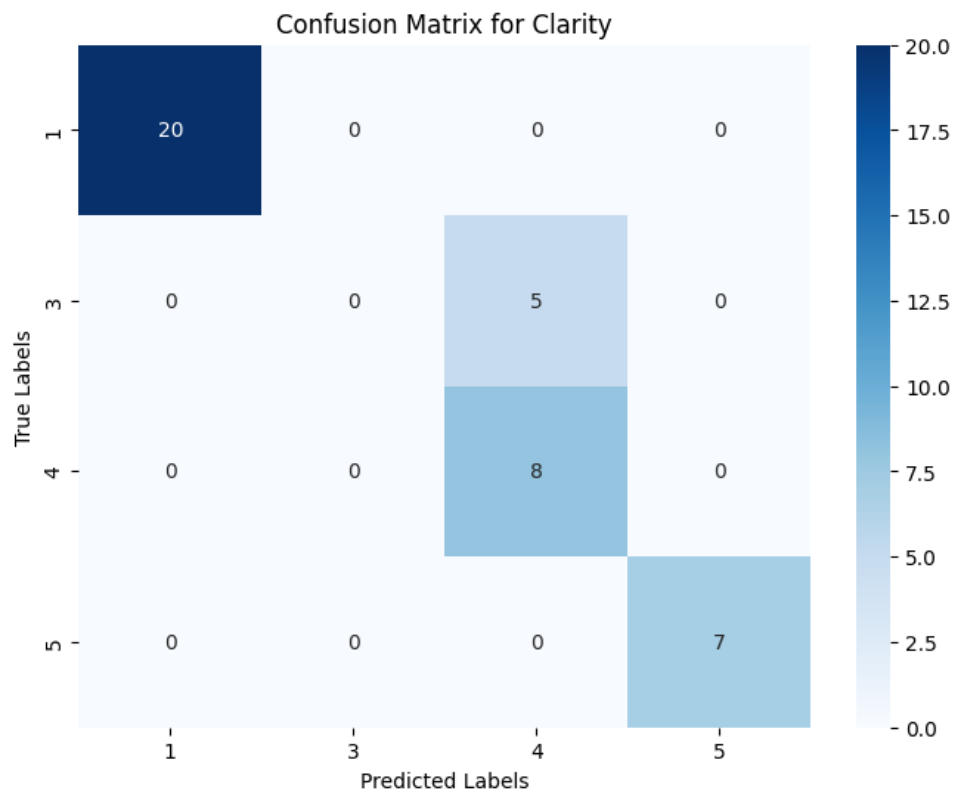


Figure 2: Confusion Matrix for Clarity.

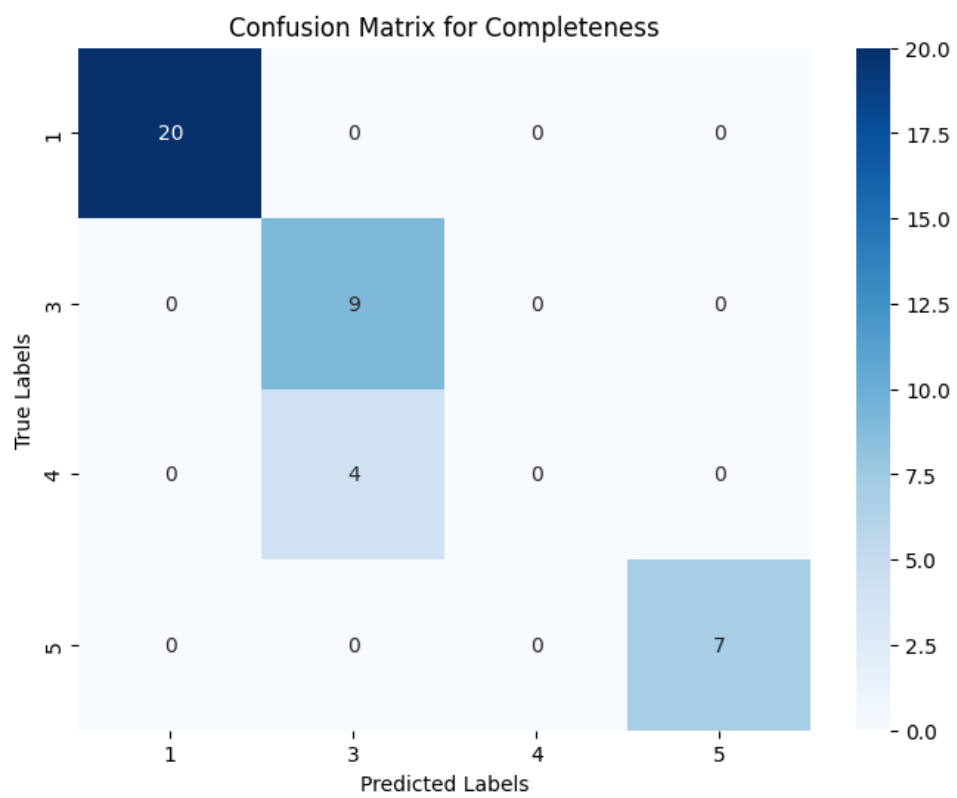


Figure 3: Confusion Matrix for Completeness.

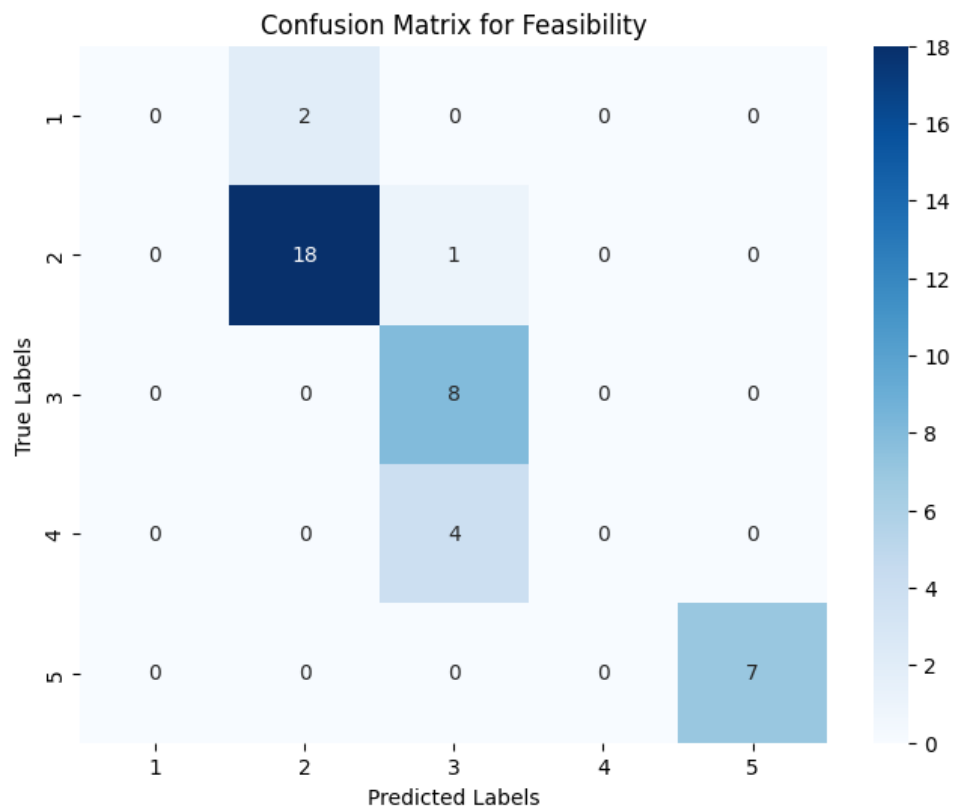


Figure 4: Confusion Matrix for Feasibility.

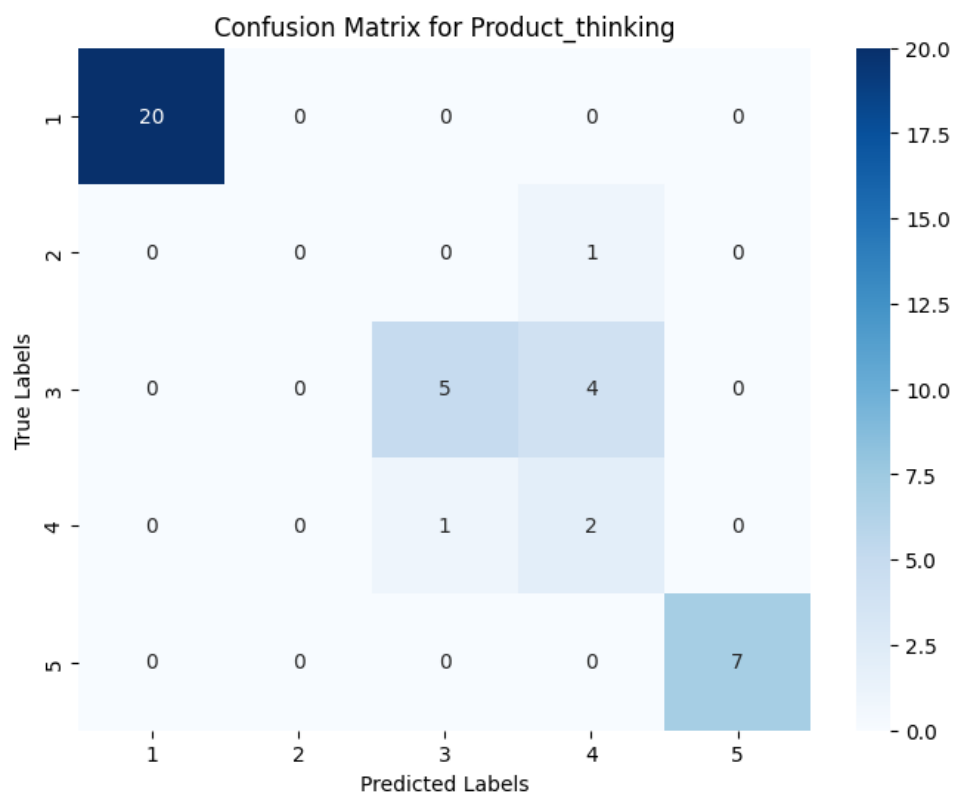


Figure 5: Confusion Matrix for Product Thinking.

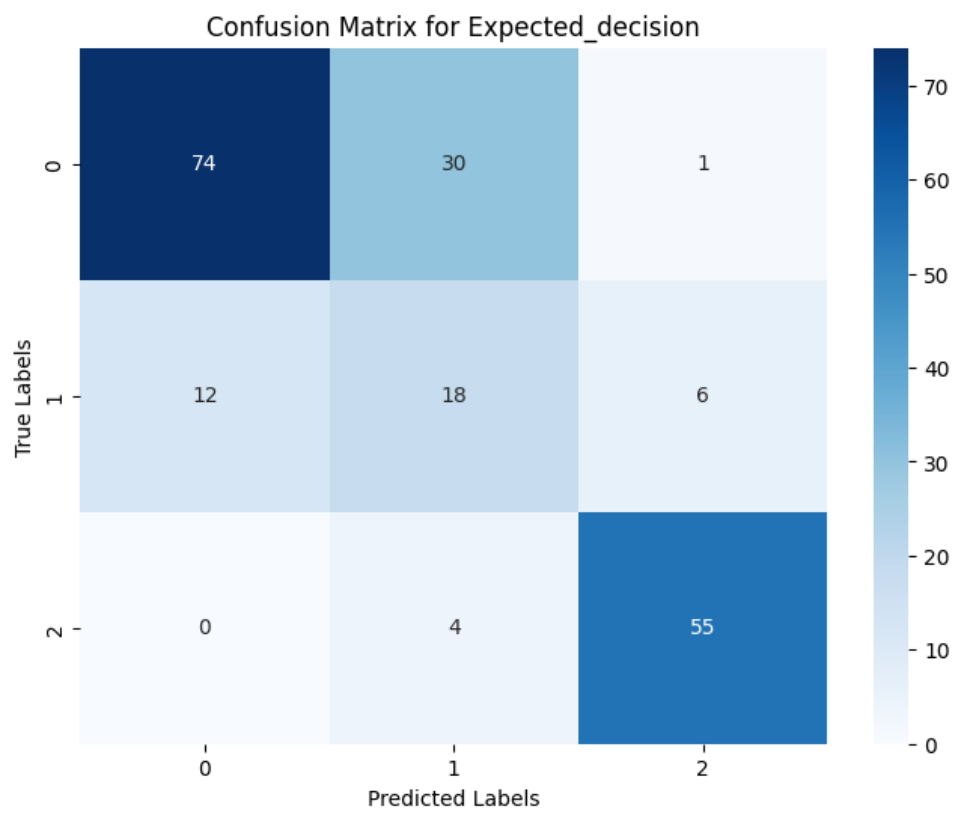


Figure 6: Confusion Matrix for RL Agent.