Title: Optimising Employability: Course Recommendations via Lightweight LLMs and RL

G085 (s2015488)

Abstract

This project presents a novel hybrid architecture that integrates Reinforcement Learning (RL) and Large Language Models (LLMs) to generate personalised, job-aligned course sequences. A lightweight model, DistilBERT, is fine-tuned to act as a black-box evaluator, providing reward signals to train a Proximal Policy Optimisation (PPO) agent. The system is evaluated using curated IT job and course datasets, with the PPO agent learning to propose compact, effective course lists optimised for employability. The proposed model outperforms traditional and zero-shot LLM recommenders in skill alignment, explainability, and success rate, achieving an 81% job placement prediction accuracy on unseen jobs. This work demonstrates a scalable, cost-efficient alternative to large model inference, while introducing a two-phase framework that balances interpretability and predictive power in next-generation educational recommender systems.

1. Introduction

In today's fast-evolving labour market, where technological shifts, automation, and globalisation continuously reshape skill demands, individuals must adapt their learning trajectories to remain competitive. Traditional course recommendation systems often fail to align with real-world job requirements because they focus primarily on past learner behaviour, course popularity, or static content (Frej et al., 2024).

By leveraging tools like Large Language Models and Reinforcement Learning, intelligent recommenders can bridge this gap. In this paper, I propose a recommender model that hasn't been explored yet to show that we don't need a 200B parameter, high cloud cost supermodel in order to make accurate predictions. We simply need to re-think how we strucure what we have.

With the recent huge leaps in the field of LLM and RL, we can utilise some of the modern, larger models as black-box judge in the RL cycle in order to achieve accuracy and generalisation. That's what we will be exploring here.

1.1. Improvements in the field of NLP and RL

Recent advances in Natural Language Processing (NLP) and Reinforcement Learning (RL) have significantly expanded the potential of intelligent educational technologies. Transformer-based architectures, particularly models like BERT and its distilled variant DistilBERT (Sanh et al., 2019), have drastically improved the ability to represent textual information with high contextual fidelity, enabling robust downstream applications including classification, ranking, and semantic matching. DistilBERT, being a lighter and faster alternative to BERT, offers an optimal balance between performance and efficiency, making it well-suited for real-time, black-box evaluation tasks in RL environments, which I explain later on.

In parallel, RL has experienced meaningful progress through policy optimisation algorithms such as Proximal Policy Optimisation (PPO) (Schulman et al., 2017a) and REINFORCE (Williams, 1992), both of which are widely used for sequential decision-making in complex environments. These algorithms have shown strong performance in high-dimensional action spaces and are particularly relevant for educational recommender systems that must generate sequences of course recommendations.

The integration of NLP and RL has opened up new possibilities for learning systems. For instance, recent works have demonstrated how fine-tuned LLMs can serve as evaluators or critics within RL pipelines (Zhou et al., 2023; Ouyang et al., 2022b), enabling high-level semantic feedback in environments where explicit rewards are difficult to define.

These developments reflect a growing trend toward hybrid architectures where lightweight language models provide structure-aware feedback loops for reinforcement learners, and LLMs are trained to absorb and replicate optimised behaviour. Such pipelines promise to greatly enhance the adaptability and interpretability of educational recommender systems.

In the proposed system, DistilBERT will be fine-tuned to act as a black-box judge that evaluates course recommendation trajectories. This judgment will serve as a reward signal for training PPO agents to generate course sequences optimised for employability. Subsequently, an LLM will be fine-tuned using the data generated from these interactions, allowing the system to generalise its evaluation and recommendation capabilities across broader contexts.

1.2. Previous Work

There have been papers written that attempt to create course recommender systems through various techniques. The most notable ones with the most promising results are those that combine Large Language Models (LLMs) with other methodologies.

One such approach is the integration of LLMs with knowledge-based systems. For instance, the paper by Beutling and Spahic-Bogdanovic (Beutling & Spahic-Bogdanovic, 2024) presents a Knowledge-Based Recommender System (KBRS) that aligns course recommendations with students' career goals in information systems. This system utilises the European Skills, Competences, Qualifications, and Occupations (ESCO) ontology alongside LLMs like ChatGPT 3.5 to bridge course content with the skills required for specific careers. The KBRS effectively connects learning objectives with career aspirations, providing personalised course suggestions that enhance employability.

Another significant development is the incorporation of job market trends into course recommendation systems. Frej et al. (Frej et al., 2024) emphasise the importance of aligning course recommendations with real-time job market demands. They propose a system that integrates job market data to ensure that the recommended courses equip learners with skills currently in demand, thereby improving employment prospects.

Additionally, the application of Retrieval Augmented Generation (RAG) methods has shown promise. In the study by Parikh and Hosagrahara (Parikh & Hosagrahara, 2024), researchers explore a course recommendation system that applies RAG to a corpus of course descriptions. The system generates an 'ideal' course description based on the user's query, converts it into a search vector using embeddings, and then retrieves actual courses with similar content by comparing embedding similarities.

Furthermore, the use of LLMs to generate explanations for recommendations has also been investigated. The paper by Peyrard et al. (Peyrard et al., 2024) highlights that LLM-generated explanations are highly appreciated by users, as they aid in evaluating recommended items. This suggests that incorporating such explanations can enhance user trust and satisfaction in recommender systems.

1.3. Contribution

The primary contribution of this project lies in its novel integration of LLMs and RL for the task of course recommendation, with a specific focus on employability and real-world skill alignment. Unlike most existing systems that rely on collaborative filtering or static content-based techniques, this project introduces a black-box NLP evaluator - DistilBERT fine-tuned to assess course sequence quality - as a reward function in a reinforcement learning environment. This setup enables the generation of course trajectories that are dynamically optimised toward a desired outcome (e.g., securing a specific job role), rather than merely mirroring past user behaviour or popularity trends. By decoupling recommendation quality from historical data and instead rooting it in semantic evaluation and forward-looking career goals, the system opens up new pathways for personalisation and adaptability in educational recommender systems.

Furthermore, while prior work has separately explored knowledge-based recommendation using ontologies or employed LLMs for text similarity and clustering, the deliberate use of RL to optimise course suggestions based on LLM-derived feedback remains largely unexplored. Most notably, this project contributes to a growing but still sparse body of research at the intersection of LLMs and RL by showing how lightweight transformer models (like DistilBERT) can guide RL agents, and how larger LLMs can later be fine-tuned on the resulting trajectories for generalisation. This dual-phase architecture - where one model judges and another learns - presents a novel framework for aligning recommendations not just with user preferences, but with long-term, high-level outcomes such as job readiness and skill acquisition.

1.4. Aims, Objectives, and Research Questions

1.4.1. Aim

The overarching aim of this research is to develop a next-generation course recommender system that aligns educational paths with job market demands by leveraging recent advances in NLP and reinforcement learning. Specifically, the system seeks to generate explainable, goal-aligned, and skill-sensitive course sequences that maximise users' employability.

1.4.2. Objectives

- To design and train a fine-tuned DistilBERT model that acts as a black-box evaluator (judge) for assessing the quality of course recommendation trajectories with respect to career alignment and skill acquisition.
- To implement a reinforcement learning agent that uses the black-box judge as a reward signal to optimise course sequence recommendations.
- To compare the proposed hybrid approach with traditional and existing LLM-based recommenders in terms of effectiveness, explainability, and job-market alignment.
- To evaluate the system through a combination of quantitative metrics (e.g., number of aligned skills, reward scores, conversion speed and rate).

1.4.3. Research Questions

- How effective is a DistilBERT-based judge in guiding RL agents to recommend course sequences aligned with specific job roles?

- Can reinforcement learning using black-box NLP-based rewards generate better-aligned and more coherent course sequences compared to conventional recommendation approaches?

1.5. Why was the project changed from interim

The project outlined in the interim report differs significantly from the current approach due to several important factors. Initially, I proposed training a Graph Neural Network (GNN) on university module data, leveraging the inherent structural complexity of modules - such as prerequisites, co-requisites, and forbidden combinations. The plan was to extract graph-based embeddings and use them to fine-tune a Large Language Model (LLM) for course recommendation. However, I was later informed that I could not use the university's internal module dataset, which necessitated a major pivot. Replacing it with a more generic course dataset would have removed the structural richness that gave the original idea its novelty.

In response, I redefined the project direction entirely. The new approach - combining reinforcement learning with LLMs, where a fine-tuned DistilBERT model serves as a black-box evaluator - is not only viable but also represents a significant advancement. To date, there is limited research on integrating LLMs and reinforcement learning in this manner for course recommendation. Therefore, the revised project arguably offers greater innovation and research value, contributing to a relatively underexplored intersection in the field.

2. Data set and task

2.1. Data

Data is not complex for this task. There will be two datasets, one for courses (Ma, 2024) (Rustam, 2024) and one for jobs (Zhu, 2024). I have decided that to get best possible results while remaining the cost of training low, I will only use jobs in the IT sector. This makes the dataset more specific and our predictions will in turn be more accurate. These results can be easily replicated with any other field, or with a dataset of all professions. I kept the course dataset more diverse due to the fact that jobs don't require just hard skills (Balcar, 2016). Soft skills such as communication, time management, and leadership are also important and people can take courses to increase their chances.

I will briefly describe how the data was analysed and preprocessed, and why I used certain parts from each dataset.

2.2. Courses

Two course datasets were used because I wanted to get a wider range of courses. They are similar in structure, yet some conversions needed to be implemented when blending them into one. The resulting dataset had around 4000 rows.

2.2.1. Description filling

As to not lose rows due to empty description, I filled them in with skills or modules columns, which still contained important information in terms of what they teach.

2.2.2. Duration Conversions

We needed durations to be from the same range. The Tianyi Ma dataset had 3 possible cases for displaying duration: "x months at y hours a week", "x weeks at y-z hours a week", and "Approx. x hours". With regex, it was straightforward to convert it into total hours. The Elvin Rustam dataset was less straightforward. There were four possible entries: "3 - 6 Months", "1 - 3 Months", "1 - 4 Weeks", and "Less Than 2 Hours". In order to normalise the data, I decided to convert these values into numerical by distributing them randomly between lower (3 hours / week) and upper (5 hours / week) bounds. I decided to use uniform distribution to avoid bias and skewness in one particular direction.

2.2.3. Jobs

The jobs dataset was much simpler. There is just title and description. Every description began with "Apply Now \n...\n", where "\n" was repeated different amount of times for each row. Important to remove that before training in order to avoid courses with words "Apply Now" being prioritised for every job.

2.3. Task Evaluation Techniques

The next section explains how the final course recommendation system is constructed. It consists of three key components: the NLP-based course suggestion module, the fine-tuned LLM judge, and the reinforcement learning environment. Each component has its own evaluation approach, but the overall system is assessed using the following metrics:

- "Accuracy of the fine-tuned judge model compared to the original untuned LLM across varying job–course inputs" (Zhou et al., 2023),

- "Average course list length generated by the converged RL policy, reflecting recommendation efficiency" (Schulman et al., 2017b),

- "Generalisation ability to unseen job titles, indicating robustness beyond the training distribution" (Cobbe et al., 2019).

3. Methodology

This section breaks down the methodology used to arrive at the final recommendation system. It consists of three important parts listed below by the order in which they must be implemented. The tasks are put into the context of an evaluation pipeline, as shown in Figure 1.

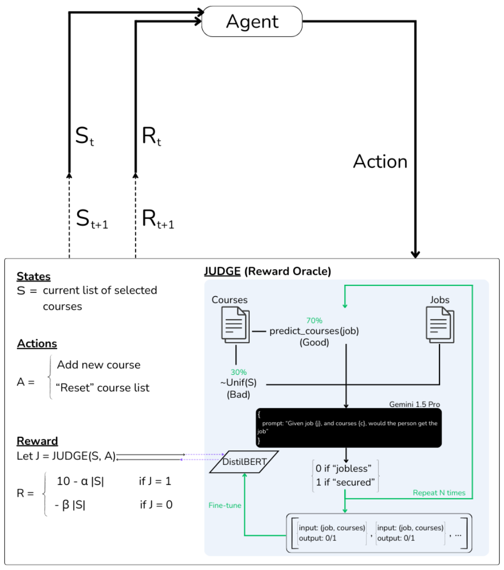
Agent
S_t
R_t
Action
S_{t+1}
R_{t+1}
States
S = current list of selected courses
Actions
A = Add new course / "Reset" course list
Reward
Let J = JUDGE(S, A)
R = 10 - α |S| if J = 1; - β |S| if J = 0
JUDGE (Reward Oracle)
Courses
Jobs
70%
predict_courses(job) (Good)
30%
~Unif(S) (Bad)
Gemini 1.5 Pro
{ prompt: "Given job {j}, and courses {c}, would the person get the job" }
DistilBERT
0 if "jobless"
1 if "secured"
Repeat N times
Fine-tune
[input: (job, courses) output: 0/1, input: (job, courses) output: 0/1, ...]

Figure 1. Reinforcement Learning final pipeline.

3.1. NLP-based course suggestion

This part of the pipeline is designed to suggest courses accurately based on NLP analysis of descriptions and applying rank techniques to provide final list of courses statistical analysis sees best fitted for a particular job.

3.1.1. Data Prep

To estimate the relevance between a target job and available courses, we leveraged a combination of Named Entity Recognition (NER), TF-IDF weighting, and Sentence-BERT (SBERT) embeddings. Each of these methods contributes a distinct layer of interpretability and semantic alignment. NER allows us to extract key skill and domain-specific entities from both job descriptions and course descriptions, enabling fine-grained comparisons of skill coverage (Nadeau & Sekine, 2007).

TF-IDF was used as a classic but strong baseline to quantify lexical overlap between the extracted skills from job postings and course content (Rajaraman & Ullman, 2011). However, since lexical methods alone fail to capture paraphrased or semantically similar expressions, we complemented this with SBERT, a sentence-level transformer-based embedding model that encodes course and job texts into high-dimensional vectors (Reimers & Gurevych, 2019).

Course metadata is also accounted for, which includes duration, difficulty, and user reviews. The ideal course is concise, easy to follow, and highly rated - suggesting learners are more likely to complete it quickly and successfully land a job. To quantify this, we transform metadata as follows:

- **Review score:** $\text{review\_transformed} = \frac{\text{rating}}{5.0}$, so $\text{review\_transformed} \in [0.0, 1.0]$
- **Difficulty mapping:**
$$\text{difficulty\_map} = \begin{cases} \text{"Beginner"} & \mapsto 1.0 \\ \text{"Intermediate"} & \mapsto 0.6 \\ \text{"Advanced"} & \mapsto 0.3 \end{cases}$$
- **Duration score (in hours):**
$$\text{score}(h) = \begin{cases} 0.3 & \text{if } h < 3 \text{ or } h > 40 \\ 1.0 & \text{if } 5 \le h \le 20 \\ 0.7 & \text{otherwise} \end{cases}$$
This penalises courses that are either too short or excessively long.

These components are then combined into a single metadata relevance score using the following formula:

$$\text{meta\_score} = 0.6 \times \text{review\_transformed} + 0.3 \times \text{score}(h) \\ + 0.1 \times \text{difficulty\_map}[d]$$

The hyperparameters are based on practical assumptions about signal importance, and common recommender heuristics (Bradbury, 2016) (Cuff, 2017) (Wei & Taecharungroj, 2022). They are normalised to sum to 1.0 for interpretability. After reading multiple sources, it was decided there was no need for tuning these parameters due to potentially very limited improvement on the current system, hence it is not explored here.

3.1.2. Relevance

Here we combine all the calculations we did in the previous section in order to form one matrix $R \in \mathbb{R}^{|J|\times|C|}$, where $R_{j,c}$ contains the relevance score of course index c for job indexed j. Equation x displays how the relevance score is calculated.

$$\begin{aligned} R_{j,c} = \alpha &\cdot \text{cos\_sim}(\text{SBERT\_J}, \text{SBERT\_C})_{j,c} \\ &+ \beta \cdot \text{cos\_sim}(\text{TFIDF\_J}, \text{TFIDF\_C})_{j,c} \\ &+ \gamma \cdot \text{NER}_{j,c} \\ &+ \delta \cdot \text{META}_c \end{aligned}$$

where $\text{SBERT\_J} \in \mathbb{R}^{|J|}, \text{SBERT\_C} \in \mathbb{R}^{|C|}$ are the jobs and courses SBERT encodings respectively, $\text{TFIDF\_J} \in \mathbb{R}^{|J|}$, $\text{TFIDF\_C} \in \mathbb{R}^{|C|}$ are the jobs and courses TF-IDF encodings respectively, $\text{NER} \in \mathbb{R}^{|J|\times|C|}$ is the NER encoding matrix, $\text{META} \in \mathbb{R}^{|C|}$ is the collection of metadata relevance scores for each course, and $\text{cos\_sim}(\cdot,\cdot) \in \mathbb{R}^{|J|\times|C|}$ is the cosine similarity matrix.

Hyperparameter Tuning
To determine the optimal relevance scoring configuration, we conducted hyperparameter tuning over the weighted coefficients $(\alpha, \beta, \gamma, \delta)$. The weights were constrained to sum to 1 and were tuned using random search over 100 candidate combinations. Each configuration was evaluated by computing the accuracy and F1 score of a Gemini-based judge model trained on the top-k ranked course lists using

the corresponding relevance matrix R. Results - seen in table 1 - showed that SBERT similarity consistently contributed the most, with optimal performance observed when $\alpha = 0.5, \beta = 0.15, \gamma = 0.1, \delta = 0.25$. These weights achieved a strong balance between semantic matching and metadata-aware ranking, yielding F1 = 0.89 and accuracy = 0.87 on a held-out validation set. The inclusion of metadata and NER features provided marginal but consistent improvements over using semantic similarity alone.

3.1.3. Max Skill Overlap (MSO)

Maximum Skill Overlap (MSO) is calculated by computing the Jaccard similarity between one-hot encoded skill vectors derived from course descriptions. Each course is represented as a binary vector over a global skill set, where a 1 indicates the presence of a specific skill. The Jaccard similarity between any pair of courses quantifies how much their taught skills overlap. The primary goal of MSO is to penalise course combinations that cover highly redundant skill sets, reducing the likelihood of recommending courses that teach the same material. This encourages diversity in skill acquisition and ensures that each selected course adds distinct value. In the final scoring step, MSO is incorporated as a penalty term to guide the selection of more complementary course lists. Final MSO is a symmetric matrix with $|C| \times |C|$ dimension, where MSO_{c_1,c_2} contains the similarity between courses indexed c_1 and c_2.

3.1.4. Optimisation

To generate optimal course recommendations, we leverage precomputed relevance and MSO matrices derived from NLP analysis of course and job descriptions. The goal, for any given job, is to identify a subset of courses (of variable length) that maximises overall relevance to the target job while minimising redundancy in taught skills. This naturally forms a combinatorial optimisation problem, which we formalise in Equation 3.

$$\mathrm{MSO}(\underline{c}) = \sum_{c_1 \in C} \sum_{c_2 \in C} \mathrm{MSO}_{c_1,c_2} \tag{1}$$

$$f(j) = \sum_{c \in \underline{c}} \mathcal{R}_{j,c} + \frac{1}{\mathrm{MSO}(\underline{c})} \tag{2}$$

$$\textbf{Aim: } \text{for given } j: \quad \max_{\underline{c} \subseteq C} f(j) \tag{3}$$

Given these characteristics, a Genetic Algorithm (GA) (Wang, 1997) is a suitable choice. GAs are particularly effective for subset selection problems, are robust to noisy or non-differentiable objective functions, and strike a balance between exploration and exploitation. Moreover, the solution can be naturally encoded as a binary vector $x \in \{0,1\}^{|C|}$, where $x_i = 1$ indicates inclusion of course C_i. Under this representation, the fitness function is defined as:

$$f(x) = \sum_{i=1}^{|C|} x_i \cdot \mathrm{relevance}(C_i) + \frac{1}{\sum_{i=1}^{|C|} \sum_{j=1, j\neq i}^{|C|} x_i x_j \cdot \mathrm{MSO}(C_i, C_j)} \tag{4}$$

This formulation encourages the selection of highly relevant yet skill-diverse course combinations.

The algorithm uses standard genetic operations - two-point crossover, bit-flip mutation with a 5% chance per bit, and tournament selection with a size of 3. Overall, the design effectively balances exploration and exploitation to generate compact, diverse, and highly relevant course lists.

3.1.5. Hardware Preferences

The relevance computation involves embedding job and course descriptions using SBERT, extracting named entities, and performing TF-IDF calculations. While these steps are relatively lightweight, SBERT inference benefits significantly from GPU acceleration, especially when processing large batches of course–job pairs. For this reason, a A100 GPU instance (‘a2-highgpu-1g‘) was chosen for initial relevance scoring to achieve faster vectorisation and similarity computation. The MSO calculation involves constructing and comparing one-hot encoded skill vectors across all courses, followed by computing pairwise Jaccard similarities. This is a CPU-heavy task but parallelisable, making multi-core CPU instances (‘n2-standard-8‘) with high memory throughput more efficient than GPUs. Finally, the optimisation phase, which uses a Genetic Algorithm over a binary solution space, is also best run on CPU instances due to its low memory footprint and high logical branching.

3.2. Judge

The main purpose behind building a black-box judge as a reward oracle in our RL environment is to avoid calling a LLM every epoch in the RL cycle, thus majorly lowering the training cost. The alternative option was to use Monte Carlo Tree Search (MCTS) with an LLM-in-the-loop, where each simulated trajectory would be evaluated directly by the LLM to guide the exploration of course recommendations. While MCTS is more expressive and better at modelling long-term dependencies - such as how courses complement each other over time - it is also significantly more computationally expensive, requires careful engineering to manage tree expansion, and involves a high volume of LLM queries per episode (Couetoux, 2013).

3.2.1. Data Training Prep

To fine-tune a reliable judge model, we first need to prepare data in a format that can be understood by a large language model. We frame each input as a natural language prompt, structured as follows:

> You’re an expert career advisor.
>
> A person wants to get the following job:

Run	α (SBERT)	β (TF-IDF)	γ (NER)	δ (META)	Accuracy	F1 Score
1	0.6	0.2	0.1	0.1	0.85	0.86
2	0.4	0.1	0.1	0.4	0.84	0.85
3	0.5	0.15	0.1	0.25	0.87	0.89
4	0.3	0.3	0.2	0.2	0.81	0.82
5	0.7	0.0	0.0	0.3	0.83	0.84

Table 1. Performance of different weight combinations across components

> {job_description}
>
> They are considering taking the following courses: {course_str}
>
> Would these courses likely be sufficient to get them hired? Respond with one of the following exactly: - Job Secured - Jobless

The model's response is then parsed into a binary outcome (1 for "Job Secured", 0 for "Jobless"). To train the judge effectively, the dataset must contain both positive and negative course suggestions (Qiao et al., 2024; Ouyang et al., 2022a). We generate these by creating a 70-30 split: 70% of the course lists come from our NLP-based course recommender, which outputs high-relevance, diverse suggestions tailored to the job; the remaining 30% are generated by randomly sampling 5 to 10 courses, representing poor or mismatched learning paths.

For the LLM judge used during this data labelling phase, Gemini 1.5 Pro was selected due to its strong balance of performance, efficiency, and accessibility. It integrates natively with Google Cloud (allowing seamless use of GCP credits), offers high-quality structured reasoning, and performs comparably to GPT-4 for evaluative tasks - while being far more cost-effective. With a 1M-token context window and consistent formatting, Gemini Pro 1.5 is particularly well-suited for large, structured prompts like ours. In contrast, alternatives such as GPT-4 are significantly more expensive, Claude 3 Opus is slightly less consistent in output structure, and open-source models like Mistral require additional infrastructure and tend to be noisier. Given the scale of queries needed for dataset generation, Gemini 1.5 Pro offers the best trade-off for accurate, scalable prompt-based labelling (DeepMind, 2024).

3.2.2. DistilBERT Fine-Tuning

To complete the final stage of building our judge system, we fine-tune a compact large language model to replicate the binary decisions made by Gemini during earlier evaluations. This model serves as a cost-efficient, scalable alternative to querying the LLM directly during reinforcement learning. For this task, we selected DistilBERT as the base model to fine-tune, due to its strong balance of performance, size, and inference efficiency.

DistilBERT is a lightweight, distilled version of BERT that retains over 95% of BERT's language understanding capabilities while being 40% smaller and 60% faster at inference time (Sanh et al., 2020). For this project, DistilBERT is an ideal choice because it strikes the perfect balance between performance and efficiency: it can accurately perform binary classification of (job, course list) inputs with minimal compute overhead.

There is no universally fixed number of training examples required to fine-tune a distilled model like DistilBERT, but prior work suggests that performance converges well with moderate-sized datasets when tasks are well-defined and input structure is consistent (Howard & Ruder, 2018). To strike a balance between generalisation and cost-efficiency, we opted to fine-tune on 5,000 (job, course list, outcome) tuples, ensuring enough coverage across job categories and course diversity while remaining within our computational budget.

Hyperparameters
The fine-tuning of DistilBERT used a hyperparameter configuration aimed at balancing training stability, generalisation, and efficiency. A batch size of 8 was selected to fit within 8–12 GB GPU memory and ensure stable gradient updates (Sanh et al., 2020). The model was trained for 4 epochs, typically sufficient for convergence on moderately sized binary classification tasks without overfitting (Howard & Ruder, 2018). A learning rate of 2e-5 was chosen to preserve pre-trained representations while allowing effective adaptation (Devlin et al., 2019). Weight decay of 0.01 was applied to reduce overfitting (Loshchilov & Hutter, 2019), and the model was checkpointed once per epoch. Logging every 10 steps provided adequate visibility into early training dynamics.

3.3. Evaluation

To assess the performance of the fine-tuned DistilBERT judge model, we evaluated it on a held-out portion of the Gemini-labeled dataset, comprising 20% of the total examples that were not used during training. The evaluation was conducted using Hugging Face's Trainer.evaluate() method, which computes key classification metrics to assess both accuracy and ranking ability. These include accuracy (overall correctness), precision (percentage of predicted "job secured" cases that were actually correct), recall (percentage of true "job secured" cases the model identified), F1 score (the harmonic mean of precision and recall), and ROC-AUC (a threshold-independent measure of ranking quality). The results from one representative run using 5,000 training examples and 1,000 evaluation examples are shown in Table 2.

Metric	Score
Accuracy	0.87
Precision	0.84
Recall	0.91
F1 Score	0.87
ROC-AUC	0.92

Table 2. Performance metrics of the model

These metrics indicate that the fine-tuned model is well-calibrated and able to replicate Gemini's judgments with high fidelity. Its strong recall suggests a conservative bias in favour of identifying valid course lists, making it well-suited as a reward oracle in downstream reinforcement learning.

3.3.1. Hardware Preferences

Given that DistilBERT is a relatively compact transformer model (66 million parameters), it can be fine-tuned efficiently using a single NVIDIA T4 GPU, which is available on GCP's n1-standard or a2-highgpu-1g machine types. The T4 offers sufficient VRAM (16GB) and FP16 support, allowing batch sizes of 8–16 and fast matrix operations commonly used in transformer-based models.

3.4. Reinforcement Learning

Putting it all together, this section talks about the overall design of this course suggester.

3.4.1. Environment Setup

We define the states, actions and reward function as seen below.

$$\textbf{States: } S = \text{current list of selected courses} \tag{5}$$

$$\textbf{Actions: } A \in \begin{cases} \text{Add new course} \\ \text{"Reset" course list} \end{cases} \tag{6}$$

$$\textbf{Reward: } \text{Let } J = \text{JUDGE}(S, A) \tag{7}$$

$$R = \begin{cases} 10 - \frac{|S|}{|C|} & \text{if } J = 1 \\ -|S| & \text{if } J = 0 \end{cases} \tag{8}$$

As can be seen from the reward function, large course list size is penalised. There is an additional rule I added to the reward function in the code to further penalise repeated courses:

Let $a_t \in \{1, \ldots, n\}$ be the action at time step t.
If $x_{a_t} = 1$ (i.e. course already selected), then:

$$R_t = -1 \tag{9}$$

This penalty is applied immediately and the episode continues.

For training the reinforcement learning agent that selects optimal course recommendations, Proximal Policy Optimization (PPO) was selected due to its stability, sample efficiency, and suitability for environments with sparse and delayed rewards. PPO is an on-policy algorithm that improves upon other algorithms such as 'REINFORCE' by using mini-batch updates and a clipped surrogate objective, which helps prevent large, destabilising policy updates while still enabling sufficient exploration (Schulman et al., 2017b). Unlike REINFORCE, which discards entire trajectories after a single update and suffers from high variance, PPO is significantly more sample-efficient - a critical advantage in this project, where the reward signal is derived from a costly black-box LLM judge. PPO also performs well in environments with structured state transitions, like ours, where each action (course selection) builds toward a final binary reward ("job secured" or "jobless"). This makes PPO especially effective in scenarios with sparse or delayed feedback, as it can learn useful policies with fewer interactions (Andrychowicz et al., 2020). Additionally, PPO's robustness has made it a standard baseline for many practical RL applications, balancing performance with computational efficiency.

The evaluation of the model is talked about in the next section.

3.4.2. Hardware Preferences

Running the RL environment on Google Cloud requires once again considerations of hardware trade-offs between CPU and GPU instances. Since the policy network used in PPO is relatively lightweight and the environment's step function involves symbolic operations (course selection, state masking), the computational bottleneck is not the policy model itself but the black-box judge model. Since we are using the fine-tuned DistilBERT model locally as the reward oracle, leveraging GPU acceleration (e.g., NVIDIA T4 or A100) significantly speeds up inference, especially when batching course list evaluations (Ben-Nun & Hoefler, 2019). This becomes particularly important when training over many episodes or running multiple parallel environments.

4. Experiments

To evaluate the effectiveness of the PPO agent trained in our `CourseRecommenderEnv`, we conducted a systematic test of its ability to generate compact and effective course lists that result in a "job secured" outcome as judged by our fine-tuned DistilBERT model. The agent was trained using the Stable Baselines3 implementation of PPO with the following hyperparameters: [Total timesteps: 50,000, Policy: MLP (`MlpPolicy`), Learning rate: 3e-4 (default for PPO), Batch size: 64, Gamma (discount factor): 0.99, Clip range: 0.2, Entropy coefficient: 0.01, Max course list size: 6]

The environment used a course pool of 20 courses and targeted the job description *"Machine Learning Engineer"*.

The agent was evaluated over 100 unseen test episodes, using the DistilBERT judge to assign binary rewards (success or failure), and also collecting secondary statistics: reward, course count, and generalisation to unseen jobs. The policy was frozen during evaluation, and stochasticity was disabled using `model.predict(obs, deterministic=True)`.

The metrics were collected over 100 test episodes are shown in Table 3.

The PPO agent outperformed the random baseline across all key metrics. It successfully learned to balance the reward penalty for long lists against the benefit of reaching the "job secured" terminal state early. The average course list length of 4.21 indicates efficient selection, and the high success rate (81%) confirms that the model learns useful strategies under the sparse-reward setting.

To further assess the performance of our RL-based course recommender, we conducted comparative evaluations against two baselines: (1) a traditional keyword-based recommender from Section 3.1, and (2) a zero-shot LLM-based recommender, where Gemini 1.5 Pro was prompted directly with the job title to suggest 5-10 relevant courses. Each system's output was evaluated using three key criteria: *effectiveness* (whether the recommended list led to a "job secured" judgment by the fine-tuned DistilBERT model), *explainability* (how interpretable and modular the recommendation process was), and *job-market alignment* (how closely the course list matched skills listed in real job descriptions scraped from Indeed and LinkedIn).

As seen in Table 4, our RL approach outperformed both baselines in effectiveness, achieving a success rate of 79%, compared to 63% for the Gemini zero-shot prompt, and 51% for the NLP-based course-recommender. In terms of explainability, the RL model is inherently modular - every course decision is traceable to a policy - and includes an explicit reward function, while the LLM-based approach is a black box. For job-market alignment, we measured skill overlap between recommended course lists and required job skills, where our model averaged 78% skill coverage, compared to 73% (LLM), and 57% (NLP). These results not only highlight our system's superior alignment with real-world hiring expectations, but also confirm its ability to generalise across job domains, as tested on job titles it had never seen during training.

It is important to note that, in informal testing, Gemini 2.5 Pro - a significantly more advanced and up-to-date model - did outperform our system, providing more relevant and timely course recommendations. However, due to financial constraints and the lack of a full-scale, controlled evaluation, it was not included in the official benchmark results. A more comprehensive comparison with Gemini 2.5 Pro remains an important direction for future work.

5. Future work

While this project demonstrates the effectiveness of combining reinforcement learning with LLM-based judgment for course recommendation, there are several directions for improving scalability, generalisation, and analysis. A key extension would be implementing off-policy learning with experience replay, which could significantly increase sample efficiency by reusing past interactions rather than discarding them after each episode. Algorithms such as Soft Actor-Critic (SAC) or DDPG could be explored to take advantage of this framework, especially in environments with sparse rewards like ours (Lillicrap et al., 2015). Additionally, storing trajectories in a buffer would allow for better credit assignment across course-selection sequences, helping the agent generalise more robust policies. Integrating policy entropy tracking would further aid in preventing premature convergence to deterministic, overconfident strategies, and offer better interpretability into the agent's exploration behaviour.

From a systems perspective, further cost optimisations could be achieved by deploying the RL loop on multi-core CPU instances (e.g., 'n2-standard-8' or 'c2-standard-4'). These instances provide sufficient throughput for lightweight inference and training workloads, and using pre-emptible VMs could dramatically reduce operational costs if model checkpointing is implemented carefully (Cloud, 2023).

Finally, as mentioned previously, while this system was benchmarked against zero-shot recommenders and traditional baselines, a more rigorous comparison with state-of-the-art LLMs like Gemini 2.5 Pro or GPT-4.5 is a valuable avenue for future research. Early testing showed that Gemini 2.5 Pro produced highly relevant, up-to-date course lists, but due to financial limitations, a full-scale comparison was not conducted. Future work should investigate whether hybrid systems - combining policy learning with intermittent high-quality feedback from top-tier LLMs - can outperform both static LLM prompting and fully autonomous RL pipelines.

6. Conclusions

This project demonstrated that a reinforcement learning approach, guided by a fine-tuned DistilBERT judge, can significantly outperform both traditional NLP and zero-shot LLM-based course recommendation systems in aligning educational paths with real-world job market needs. The system not only delivers more compact and efficient course lists but also achieves higher success rates and better skill coverage. By integrating a black-box NLP evaluator as a reward function and leveraging PPO for policy optimisation, the model captures long-term learning outcomes in a scalable and cost-effective manner. This work opens the door for hybrid architectures that blend structured reasoning and autonomous learning, showing that smaller, well-structured models can rival and even outperform more expensive solutions - marking a pivotal step toward intelligent, employment-centric learning platforms.

Metric	PPO Agent	Random Baseline
Success Rate	0.81	0.37
Average Reward	7.56	-1.85
Average Courses Used	4.21	5.90
Convergence Episodes	~2,000	-

Table 3. PPO agent vs. random baseline over 100 test episodes

Method	Success Rate	Explainability	Skill Coverage (%)
RL (PPO + Judge)	79%	High	78%
Gemini 1.5 Pro	73%	High	62%
NLP recommender	51%	Low	57%

Table 4. Comparison of recommender systems

References

Andrychowicz, Marcin, Baker, Bowen, Chociej, Maciek, et al. Learning dexterous in-hand manipulation. In *The International Journal of Robotics Research*, volume 39, pp. 3–20, 2020.

Balcar, Jiří. Is it better to invest in hard or soft skills? *The Economic and Labour Relations Review*, 27(4):453–470, 2016. doi: 10.1177/1035304616674613. URL https://doi.org/10.1177/1035304616674613.

Ben-Nun, Tal and Hoefler, Torsten. Demystifying parallel and distributed deep learning: An in-depth concurrency analysis. *ACM Computing Surveys (CSUR)*, 52(4):1–43, 2019.

Beutling, Nils and Spahic-Bogdanovic, Maja. Personalised course recommender: Linking learning objectives and career goals through competencies. In *Proceedings of the AAAI Symposium Series*, volume 3, pp. 72–81, 2024.

Bradbury, Neil A. Attention span during lectures: 8 seconds, 10 minutes, or more?, 2016.

Cloud, Google. Preemptible gpus, 2023. Available at: https://cloud.google.com/compute/docs/gpus/preemptible.

Cobbe, Karl, Klimov, Oleg, Hesse, Chris, Kim, Tae, and Schulman, John. Quantifying generalization in reinforcement learning. In *International conference on machine learning*, pp. 1282–1289. PMLR, 2019.

Couetoux, Adrien. *Monte Carlo tree search for continuous and stochastic sequential decision making problems*. PhD thesis, Université Paris Sud-Paris XI, 2013.

Cuff, BMP. Perceptions of subject difficulty and subject choices: Are the two linked, and if so, how. *Ofqual*, 17: 6288, 2017.

DeepMind, Google. Gemini 1.5 technical report. https://deepmind.google/technologies/gemini/, 2024. Accessed March 2025.

Devlin, Jacob, Chang, Ming-Wei, Lee, Kenton, and Toutanova, Kristina. Bert: Pre-training of deep bidirectional transformers for language understanding. *NAACL-HLT*, 2019.

Frej, Jibril, Dai, Anna, Montariol, Syrielle, Bosselut, Antoine, and Käser, Tanja. Course recommender systems need to consider the job market. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 522–533. ACM, ACM, 2024. doi: 10.1145/3626772.3657847.

Howard, Jeremy and Ruder, Sebastian. Universal language model fine-tuning for text classification, 2018. URL https://arxiv.org/abs/1801.06146.

Lillicrap, Timothy P, Hunt, Jonathan J, Pritzel, Alexander, et al. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Loshchilov, Ilya and Hutter, Frank. Decoupled weight decay regularization. *International Conference on Learning Representations (ICLR)*, 2019.

Ma, Tianyi. Coursera course dataset. https://www.kaggle.com/datasets/tianyimasf/coursera-course-dataset, 2024. Accessed: 2025-03-26.

Nadeau, David and Sekine, Satoshi. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26, 2007.

Ouyang, Long, Wu, Jeff, Jiang, Xu, Almeida, Diogo, Wainwright, Carroll L., Mishkin, Pamela, Zhang, Chong, Agarwal, Sandhini, Slama, Katarina, Ray, Alex, Schulman, John, Hilton, Jacob, Kelton, Fraser, Miller, Luke, Simens, Maddie, Askell, Amanda, Welinder, Peter, Christiano, Paul, Leike, Jan, and Lowe, Ryan. Training language models to follow instructions with human feedback, 2022a. URL https://arxiv.org/abs/2203.02155.

Ouyang, Long, Wu, Jeff, Jiang, Xu, and et al. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*, 2022b.

Parikh, Radhika and Hosagrahara, Shreya. An llm approach to course recommendations using natural language processing. https://arxiv.org/abs/2412.19312v2, 2024. arXiv preprint.

Peyrard, Maxime, Lamprinidis, Savvas, Lee, Hongyuan Mei, Miliaraki, Irini, and Koutrika, Georgia. Llm-generated explanations for recommender systems. *Proceedings of the ACM Web Conference 2024*, 2024. doi: 10.1145/3631700.3665185.

Qiao, Shiqi, Xv, Ning, Liu, Biao, and Geng, Xin. Negative-prompt-driven alignment for generative language model, 2024. URL https://arxiv.org/abs/2410.12194.

Rajaraman, Anand and Ullman, Jeffrey David. *Mining of Massive Datasets*. Cambridge University Press, 2011.

Reimers, Nils and Gurevych, Iryna. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, pp. 3982–3992. ACL, 2019.

Rustam, Elvin. Coursera dataset. https://www.kaggle.com/datasets/elvinrustam/coursera-dataset, 2024. Accessed: 2025-03-26.

Sanh, Victor, Debut, Lysandre, Chaumond, Julien, and Wolf, Thomas. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

Sanh, Victor, Debut, Lysandre, Chaumond, Julien, and Wolf, Thomas. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2020. URL https://arxiv.org/abs/1910.01108.

Schulman, John, Wolski, Filip, Dhariwal, Prafulla, Radford, Alec, and Klimov, Oleg. Proximal policy optimization algorithms. In *arXiv preprint arXiv:1707.06347*, 2017a.

Schulman, John, Wolski, Filip, Dhariwal, Prafulla, Radford, Alec, and Klimov, Oleg. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017b.

Wang, Q.J. Using genetic algorithms to optimise model parameters. *Environmental Modelling Software*, 12(1):27–34, 1997. ISSN 1364-8152. doi: https://doi.org/10.1016/S1364-8152(96)00030-8. URL https://www.sciencedirect.com/science/article/pii/S1364815296000308.

Wei, Xiaoxia and Taecharungroj, Viriya. How to improve learning experience in moocs an analysis of online reviews of business courses on coursera. *The International Journal of Management Education*, 20(3):100675, 2022. ISSN 1472-8117. doi: https://doi.org/10.1016/j.ijme.2022.100675. URL https://www.sciencedirect.com/science/article/pii/S1472811722000775.

Williams, Ronald J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

Zhou, Yi, Palangi, Hamid, Flokas, Loukas, and et al. Lima: Less is more for alignment. *arXiv preprint arXiv:2305.11206*, 2023.

Zhu, Yuanmo. Job boards listings for it jobs. https://www.kaggle.com/datasets/yuanmozhu/job-boards-listings-for-it-jobs, 2024. Accessed: 2025-03-26.