VIETNAM GENERAL CONFEDERATION OF LABOR

**TON DUC THANG UNIVERSITY**

**FACULTY OF INFORMATION TECHNOLOGY**



**PHAN ANH TUAN - 521H0516**

**PHAM TRAN TIEN PHAT– 521H0285**

**NGUYEN VAN TRUONG – 521H0324**

**FINAL REPORT**

**INTRODUCTION TO  
NATURAL LANGUAGE PROCESSING**

**HO CHI MINH CITY, 2025**

VIETNAM GENERAL CONFEDERATION OF LABOR

**TON DUC THANG UNIVERSITY**

**FACULTY OF INFORMATION TECHNOLOGY**



**PHAN ANH TUAN - 521H0516**

**PHAM TRAN TIEN PHAT– 521H0285**

**NGUYEN VAN TRUONG – 521H0324**

**FINAL REPORT**

**INTRODUCTION TO  
NATURAL LANGUAGE PROCESSING**

Instructor

**Assoc. Prof. Dr. Le Anh Cuong**

**HO CHI MINH CITY, 2025**

**ACKNOWLEDGEMENT**

We would like to thank Ton Duc Thang University, Faculty of Information Technology, for including this project in the training chart of Computer Science and for creating the best learning conditions for us as well as all students of the faculty. Our report is a product of what we have learned in the semester.

We would like to give **Mr. Le Anh Cuong**, who enthusiastically taught and worked tirelessly, to give me enough tools and skills to complete this report. He played an important role in improving my mathematical logic and knowledge. The second thanks I would like to give to the teachers of the Department of Information Technology of Ton Duc Thang University for giving me the opportunity to do this report, because it is not only an information technology project but also a very important experience for me in the next following years.

*Ho Chi Minh City, 29th April 2025*

*Author*

|  |
| --- |
|  |
|  |  |

*(Sign and write full name)*

*Phan Anh Tuan*

*Pham Tran Tien Phat*

*Nguyen Van Truong*

**DECLARATION OF AUTHORSHIP**

I hereby declare that this thesis was carried out by myself under the guidance and supervision of **Mr. Le Anh Cuong**. and that the work and the results contained in it are original and have not been submitted anywhere for any previous purposes. The data and figures presented in this thesis are for analysis, comments, and evaluations from various resources by my own work and have been duly acknowledged in the reference part.

In addition, other comments, reviews and data used by other authors, and organizations have been acknowledged, and explicitly cited.

**I will take full responsibility for any fraud detected in my thesis**. Ton Duc Thang University is unrelated to any copyright infringement caused on my work (if any).

*Ho Chi Minh City, 29th April 2025*

*Author*

|  |
| --- |
|  |
|  |  |

*(Sign and write full name)*

*Phan Anh Tuan*

*Pham Tran Tien Phat*

*Nguyen Van Truong*

**INSTRUCTOR VERIFICATION AND EVALUATION SECTION**

**Confirmation from the instructor**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Ho Chi Minh City, 19th March 2025*

|  |
| --- |
|  |
|  |  |

*(Sign and write full name)*

**The teacher's evaluation part marks the test**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Ho Chi Minh City, 19th March 2025*

|  |
| --- |
|  |
|  |  |

*(Sign and write full name)*

**TÓM TẮT**

Báo cáo này bao gồm hai phần:

- Phần đầu tiên khảo sát các phương pháp Học tăng cường (RL) để tinh chỉnh các Mô hình ngôn ngữ lớn (LLM). Phần này giới thiệu các nguyên tắc và thuật toán RL chính, sau đó thảo luận về những thách thức trong thiết kế phần thưởng cho việc tạo văn bản. RLHF được trình bày như một giải pháp thực tế, với RLAIF và DPO là các giải pháp thay thế có thể mở rộng. PPO cũng được đánh giá về tính ổn định của nó. Bài báo kết thúc bằng các ví dụ thực tế, bao gồm một quy trình đào tạo RLHF đầy đủ cho GPT-2.

- Phần thứ hai này trình bày một nghiên cứu so sánh về hai phương pháp tiếp cận để phát triển hệ thống dịch máy từ tiếng Anh sang tiếng Việt. Phương pháp tiếp cận đầu tiên bao gồm tinh chỉnh mô hình dịch thuật envit5 sử dụng 4-bit quantization và Low-Rank Adaptation (LoRA) trên 1% tập dữ liệu opus-100. Phương pháp tiếp cận thứ hai xây dựng kiến ​​trúc Bộ mã hóa-Giải mã Biến áp từ đầu với bộ mã hóa Mã hóa cặp byte (BPE) tùy chỉnh được đào tạo trên 10% cùng một tập dữ liệu. Cả hai phương pháp đều được đánh giá bằng các số liệu BLEU và ROUGE. Kết quả của chúng tôi chứng minh sự đánh đổi giữa hiệu quả tính toán và chất lượng dịch thuật, với mô hình được đào tạo trước cho thấy hiệu suất vượt trội ở hầu hết các số liệu mặc dù sử dụng ít dữ liệu đào tạo hơn. Chúng tôi cung cấp phân tích chi tiết về cả hai phương pháp tiếp cận và tính phù hợp của chúng đối với các tác vụ dịch thuật từ tiếng Anh sang tiếng Việt.

**ABSTRACT**

This report consists of two parts:

- The first part surveys Reinforcement Learning (RL) methods for fine-tuning Large Language Models (LLMs). It introduces key RL principles and algorithms, then discusses challenges in reward design for text generation. RLHF is presented as a practical solution, with RLAIF and DPO as scalable alternatives. PPO is also reviewed for its stability. The paper ends with practical examples, including a full RLHF training pipeline for GPT-2.

- This second part presents a comparative study of two approaches to develop an English-to-Vietnamese machine translation system. The first approach consists of fine-tuning the envit5 translation model using 4-bit quantization and Low-Rank Adaptation (LoRA) on a 1% opus-100 dataset. The second approach builds a Transformer Encoder-Decoder architecture from scratch with a custom Byte Pair Encoding (BPE) tokenizer trained on 10% of the same dataset. Both methods are evaluated using BLEU and ROUGE metrics. Our results demonstrate trade-offs between computational efficiency and translation quality, with the pretrained model showing superior performance in most metrics despite using less training data. We provide a detailed analysis of both approaches and their suitability for English-to-Vietnamese translation tasks.

**INDEX**

[LIST OF FIGURES xii](#_Toc198131088)

[LIST OF TABLES xiv](#_Toc198131089)

[LIST OF ABBREVIATIONS xv](#_Toc198131090)

[CHAPTER 1: REINFORCEMENT LEARNING 1](#_Toc198131091)

[1.1 What is Reinforcement Learning? 1](#_Toc198131092)

[1.2 Principle of Reinforcement Learning 3](#_Toc198131093)

[1.2.1 Reinforcement Learning Example: Navigating a Maze 4](#_Toc198131094)

[1.3 Mathematical modeling of Reinforcement Learning 6](#_Toc198131095)

[1.4 Q-learning 10](#_Toc198131096)

[1.4.1 Overview 10](#_Toc198131097)

[1.4.2 How does the agent choose to act? 11](#_Toc198131098)

[1.4.3 How are rewards measured? 12](#_Toc198131099)

[1.4.4 How to update Q table 12](#_Toc198131100)

[1.4.5 Deep Q-Networks (DQN), SARSA and policy-based algorithms 13](#_Toc198131101)

[1.5 Reinforcement Learning algorithms used in large language models 14](#_Toc198131102)

[1.5.1 Initial training background of LLMs 14](#_Toc198131103)

[1.5.2 Reinforcement Learning A New Direction 15](#_Toc198131104)

[1.5.3 Challenge - Definition of rewards for LLMs 16](#_Toc198131105)

[1.6 Reinforcement Learning from Human Feedback (RLHF) 16](#_Toc198131106)

[1.6.1. What is RLHF? 16](#_Toc198131107)

[1.6.2. Importance of RLHF 17](#_Toc198131108)

[1.6.3. How does RLHF work? 18](#_Toc198131109)

[1.6.4. Formulas of RLHF (Reinforcement Learning from Human Feedback) 19](#_Toc198131110)

[1.6.5. RLHF Applications in Generative AI 21](#_Toc198131111)

[1.7 Reinforcement Learning with AI Feedback (RLAIF) 21](#_Toc198131112)

[1.7.1. What is AI Feedback Reinforcement Learning (RLAIF)? 21](#_Toc198131113)

[1.7.2. How does RLAIF work? 22](#_Toc198131114)

[1.7.3. RLAIF Formulas 23](#_Toc198131115)

[1.7.4. Comparison of RLAIF & RLHF 23](#_Toc198131116)

[1.7.5. Applications of RLAIF 25](#_Toc198131118)

[1.8 Direct Preference Optimization – DPO 26](#_Toc198131119)

[1.8.1. What is Direct Interest Optimization (DPO)? 26](#_Toc198131120)

[1.8.2. How does DPO work? 27](#_Toc198131121)

[1.8.2. DPO loss function: Elimination of reward model 28](#_Toc198131122)

[1.8.3. Can DPO scale to real-world interest datasets? 29](#_Toc198131123)

[1.8.4. The effectiveness and potential of DPO in language model tuning 30](#_Toc198131124)

[1.9 Proximal Policy Optimization - PPO 30](#_Toc198131125)

[1.9.1. Proximity Policy Optimization – PPO 30](#_Toc198131126)

[1.9.2. Main formulas 31](#_Toc198131127)

[1.9.3. Exploration vs Exploitation: 32](#_Toc198131128)

[1.10 Kullback-Leibler divergence - (KL divergence) or relative entropy 33](#_Toc198131129)

[1.10.1. Definitions 33](#_Toc198131130)

[1.10.2. Main properties 33](#_Toc198131131)

[1.10.3. Loss function for neural network 34](#_Toc198131132)

[1.10.4. Optimization of variational autoencoder 35](#_Toc198131133)

[1.10.5. Reproductive antagonist network 35](#_Toc198131134)

[1.11 Compare the algorithms with each other 38](#_Toc198131135)

[1.12 DEMO 39](#_Toc198131136)

[1.12.1 Dataset 39](#_Toc198131137)

[1.12.2. Demo code for Q-learning 39](#_Toc198131138)

[1.12.3. Demo code for RLHF 41](#_Toc198131139)

[CHAPTER 2: MACHINE TRANSLATION MODEL 43](#_Toc198131140)

[2.1 Introduction 43](#_Toc198131141)

[2.2 Dataset Description 44](#_Toc198131142)

[2.2.1 Overview 44](#_Toc198131143)

[2.2.2 Data Preprocessing 44](#_Toc198131144)

[2.3 Approach 1: Fine-Tuning Pretrained Model 47](#_Toc198131145)

[2.3.1 Pretrained Model Selection 47](#_Toc198131146)

[2.3.2 Quantization Implementation 47](#_Toc198131147)

[2.3.3 LoRA Fine-tuning 47](#_Toc198131148)

[2.3.4 Results 48](#_Toc198131149)

[2.4 Approach 2: Building From Scratch 48](#_Toc198131150)

[2.4.1 Transformer Encoder-Decoder Architecture 48](#_Toc198131151)

[2.4.2 Tokenization with BPE 49](#_Toc198131152)

[2.4.3 Training Methodology 50](#_Toc198131153)

[2.4.4 Results 50](#_Toc198131154)

[2.5 Evaluation Metrics 51](#_Toc198131155)

[2.5.1 BLEU (Bilingual Evaluation Understudy) 51](#_Toc198131156)

[2.5.2 ROUGE Metrics 51](#_Toc198131157)

[2.6 Compare and Evaluate 52](#_Toc198131158)

[2.6.1 Comparison 52](#_Toc198131159)

[2.6.2 Evaluation and Analysis 53](#_Toc198131160)

[REFERENCES 54](#_Toc198131161)

# LIST OF FIGURES

[Figure 1. 1 Reinforcement Learning simulates the way humans learn by trial and error to achieve goals. 1](#_Toc197197927)

[Figure 1. 2 The Reinforcement Learning Loop with an Interpreter Component 2](#_Toc197197928)

[Figure 1. 3 Taxonomy of Reinforcement Learning Algorithms 3](#_Toc197197929)

[Figure 1. 4 Reinforcement Learning works on Markov Decision Process 4](#_Toc197197930)

[Figure 1. 5 Navigating a Maze 5](#_Toc197197931)

[Figure 1. 6 Example of Agent 6](#_Toc197197932)

[Figure 1. 7 The Agent's surrounding environment 7](#_Toc197197933)

[Figure 1. 8 The Agent takes action 7](#_Toc197197934)

[Figure 1. 9 The changes state for the agent 7](#_Toc197197935)

[Figure 1. 10 The situation that agent perceives. 8](#_Toc197197936)

[Figure 1. 11 Feedback 8](#_Toc197197937)

[Figure 1. 12 Interaction Loop Between Agent and Environment 9](#_Toc197197938)

[Figure 1. 13 Total Reward Accumulation 9](#_Toc197197939)

[Figure 1. 14 Discounted Total Reward in Reinforcement Learning 10](#_Toc197197940)

[Figure 1. 15 Q-Table 11](#_Toc197197941)

[Figure 1. 16 Value-based algorithms applied 14](#_Toc197197942)

[Figure 1. 17 An overview of the RLHF learning process 18](#_Toc197197943)

[Figure 1. 18 RLAIF: Scaleing Reinforcement Learning from Human Feedback with AI Feedback 24](#_Toc197197944)

[Figure 1. 19 RLAIF produces the highest percentage of harmless conversations compared to RLHF and SFT 25](#_Toc197197945)

[Figure 1. 20 Direct Preference Optimization 26](#_Toc197197946)

[Figure 1. 21 Direct Preference Optimization Process 27](#_Toc197197947)

[Figure 1. 22 Comparison between RLHF and DPO Training Pipelines 27](#_Toc197197948)

[Figure 1. 23 TL; DR summary win rate vs human-written summaries, using GPT-4 as the evaluator. DPO outperforms PPO's best summary performance 29](#_Toc197197949)

[Figure 1. 24 Pseudocode 33](#_Toc197197950)

[Figure 1. 25 Architecture Comparison between Autoencoder (AE) and Variational Autoencoder (VAE) 35](#_Toc197197951)

[Figure 1. 26 Architecture of a Generative Adversarial Network (GAN) 36](#_Toc197197952)

[Figure 1. 27 The original maze 40](#_Toc197197953)

[Figure 1. 28 Results 41](#_Toc197197954)

[Figure 1. 29 Fine-tuning 42](#_Toc197197955)

[Figure 1. 30 The Comparison result 42](#_Toc197197956)

[Figure 2. 1 Machine translation 43](#_Toc198125218)

[Figure 2. 2 Check “Null” values 45](#_Toc198125219)

[Figure 2. 3 Check duplicated values 45](#_Toc198125220)

[Figure 2. 4 Check lengths 45](#_Toc198125221)

[Figure 2. 5 Check outlier 45](#_Toc198125222)

[Figure 2. 6 Text Length 46](#_Toc198125223)

[Figure 2. 7 Check character sets 46](#_Toc198125224)

[Figure 2. 8 Clean dataset 47](#_Toc198125225)

[Figure 2. 9 Results of Fine-Tuning Pretrained Model 48](#_Toc198125226)

[Figure 2. 10 The results of translation 50](#_Toc198125227)

# LIST OF TABLES

[Table 1. 1 Comparison of RLAIF & RLHF 24](#_Toc197198012)

[Table 1. 2 Compare the algorithms with each other 39](#_Toc197198013)

[Table 2. 1 Comparison 51](#_Toc198125228)

# LIST OF ABBREVIATIONS

|  |  |
| --- | --- |
| RL | Reinforcement Learning |
| **RLHF** | Reinforcement Learning from Human Feedback |
| **RLAIF** | Reinforcement Learning from AI Feedback |
| DPO | Direct Preference Optimization |
| PPO | Proximal Policy Optimization |
| **KL-Divergence** | Kullback-Leibler divergence |
| LLMs | Large-Language Models |
| AE | AutoEncoder |
| VAE | Variational AutoEncoder |
| LoRa | Low-Rank Adaptation |

# CHAPTER 1: REINFORCEMENT LEARNING

## What is Reinforcement Learning?

A robot holding a tablet

AI-generated content may be incorrect.Reinforcement Learning (RL) is a type of artificial learning that teaches programs to take decisions in the hope of reaching the best result. The method replicates human beings' application of trial and error learning to reach a specified goal. RL assists the software in reinforcing actions towards the goal and eliminate actions away from the goal.

Figure 1. 1 Reinforcement Learning simulates the way humans learn by trial and error to achieve goals.

Reinforcement Learning is a technique in which an agent (the learner or decision-maker) interacts with an environment to achieve a goal. The agent takes actions and receives feedback in the form of rewards or penalties, thereby adjusting its strategy to optimize decisions over time.

Reinforcement Learning accepts short-term sacrifices to gain long-term benefits, enabling AI to effectively adapt in complex, unknown environments. It is widely applied in areas such as autonomous robotics, video games, financial trading, supply chains, and many other automated systems.

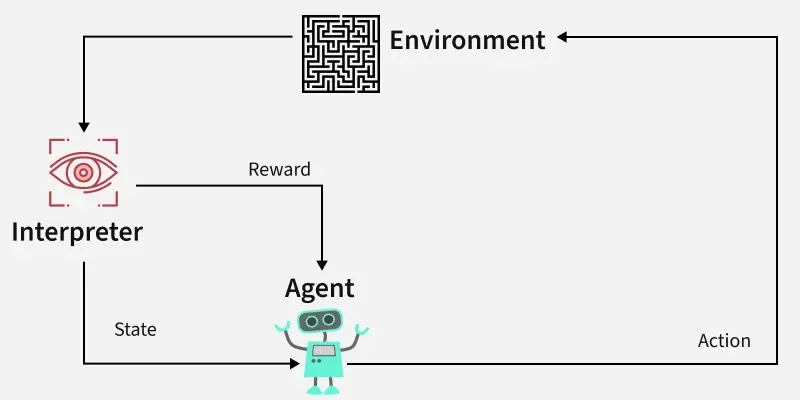


Figure 1. 2 The Reinforcement Learning Loop with an Interpreter Component

**\* Main components of RL:**

* ***Agent:*** The person who decides to take action.
* ***Environment:*** The world or system in which the agent operates.
* ***State:*** The situation or condition the agent is facing.
* ***Actions:*** The moves or decisions the agent can take.
* ***Rewards:*** Feedback from the environment based on the agent's actions.

**\* Note:**

Some studies classify Reinforcement Learning (RL) into two main groups: ***Model-Based*** and ***Model-Free***.

However, in some other papers, Reinforcement Learning is divided into four groups: ***Value-Based***, ***Policy-Based***, ***Model-Based***, and ***Hybrid Approaches*** (such as Actor-Critic Methods).

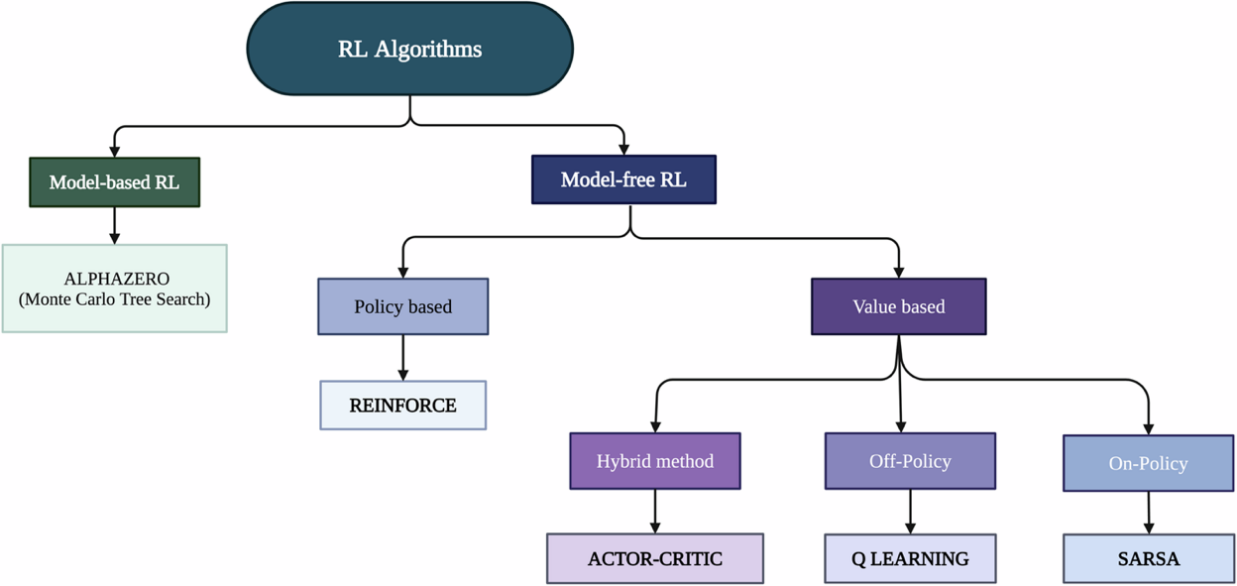
In this paper, we describe that the separation of RL into two types (***Model-Based*** and ***Model-Free***) is a general framework, emphasizing whether the agent is using a model of the world or not while learning. But the separation into four types is technically a more precise subdivision in the Model-Free type, where the model does not need to have prior understanding or predicting the environment but learns through direct experience.

Figure 1. 3 Taxonomy of Reinforcement Learning Algorithms

## Principle of Reinforcement Learning

RL is equivalent to the agent operating within a world, learning from rewards and penalties since its actions are influenced, and the agent adapting as a result of this loop and enhancing decision making as time progresses. Reinforcement Learning depends on several underlying aspects:

* Agent
* Environment
* Action
* State
* Reward
* Cumulative Reward

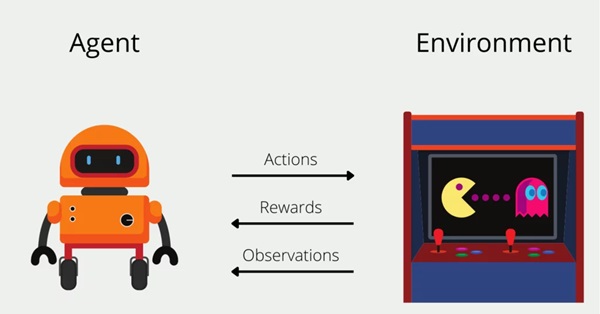


Figure 1. 4 Reinforcement Learning works on Markov Decision Process

The agent is trained to optimize its action by trial and error learning, similar to human learning. RL is based on the concept of a Markov Decision Process (MDP): the agent takes an action, the environment responds with a new state and reward, and the agent updates its policy.

Learning process involves a balance between exploration (experimenting with new actions) and exploitation (selecting best-known actions) to find the optimal policy which maximizes the total reward.

In particular, the process of operation in Reinforcement Learning is based on the Markov Decision Process (MDP), with the agent taking sequential steps one after another. For every step:

* The agent takes an action based on the current state of the environment.
  + The environment responds by updating to a new state and providing a corresponding reward.
  + The agent learns from the feedback and adjusts its action policy to optimize future rewards.

### Reinforcement Learning Example: Navigating a Maze

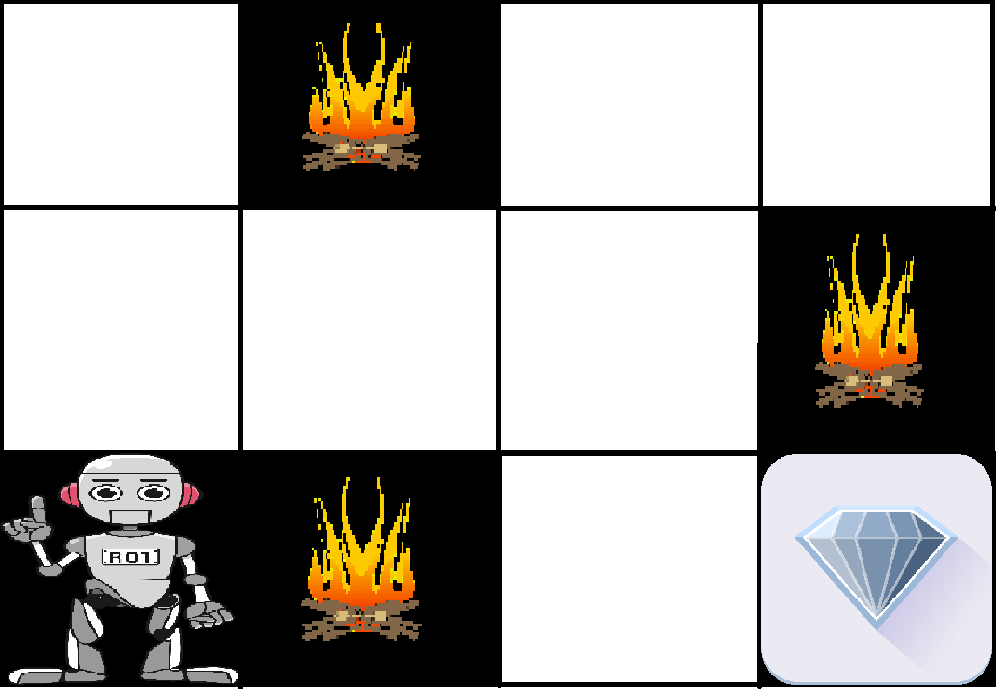
- ***Description:*** The robot navigates a maze to find the optimal path to the diamond, by trying multiple paths, receiving rewards for correct paths and penalties for incorrect paths, thereby learning to choose the action that yields the highest cumulative reward.

Figure 1. 5 Navigating a Maze

**The robot's learning process is as follows:**

**Step 1:** Exploration: The robot begins to explore the paths in the maze, performing different actions (moving left, right, up or down).

**Step 2:** Feedback: After each move, the robot receives feedback from the environment:

Positive rewards when approaching the diamond.

Penalties when entering a dangerous area (fire).

**Step 3:** Behavior adjustment: Based on the feedback received, the robot adjusts its actions to prioritize the safe path and increase the accumulated reward.

**Step 4:** Finding the optimal path: Through the learning process, the robot determines the optimal path, avoiding danger and achieving the highest reward by applying accumulated experience.

**Two main types of algorithms:**

* **Model-based Reinforcement Learning:**  
  The agent builds an internal model that simulates the environment to plan its actions.  
  Advantages: Requires fewer trials to learn.

Example: Dyna-Q (combines model-based and model-free learning).

* **Model-free Reinforcement Learning:**  
  The agent learns directly from real interactions without needing a model of the environment. Suitable for dynamic environments but requires more exploration.  
  Notable algorithms include:
  + **Q-Learning:** Learns Q-values for each state-action pair.
  + **SARSA:** Similar to Q-Learning but updates based on the actual chosen action.
  + **Policy Gradient Methods:** Learn the policy directly, e.g., REINFORCE and PPO.
  + **Deep Q-Network (DQN):** Combines Q-learning with deep neural networks.

## Mathematical modeling of Reinforcement Learning

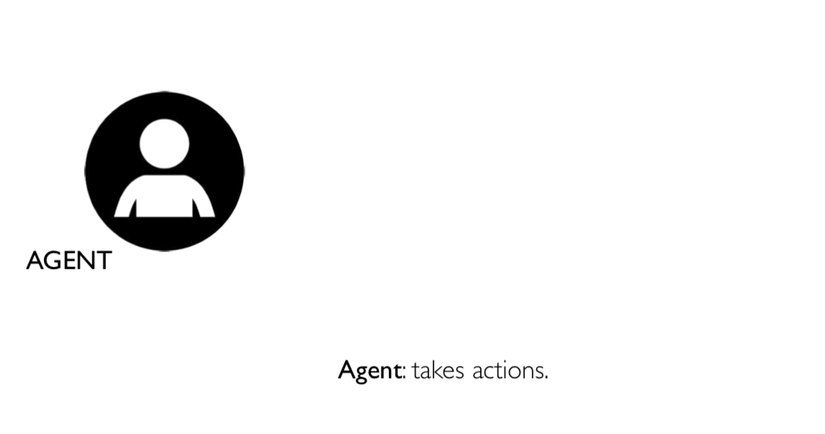
* Visual example of Agent:

Figure 1. 6 Example of Agent

* The Agent's surrounding environment, where the agent exists and interacts



Figure 1. 7 The Agent's surrounding environment

* A black and white image of a tree

  AI-generated content may be incorrect.Based on the State S(t) of the current environment, the agent will take action a(t)

Figure 1. 8 The Agent takes action

* A diagram of a tree

  AI-generated content may be incorrect.After receiving interaction from the agent, the environment changes state for the agent

Figure 1. 9 The changes state for the agent

* The current state of the environment is S(t+1), that is, at time t+1

A diagram of a tree

AI-generated content may be incorrect.

Figure 1. 10 The situation that agent perceives.

* A diagram of a tree

  AI-generated content may be incorrect.At this time, the agent receives a reward r(t). This reward depends on the agent's action a(t) and the State S(t) of the environment at the previous time, that is, at time t:

Figure 1. 11 Feedback

* Since we do not know the end time of this iteration, the total reward will be an infinite series of component rewards at different times since time t (initially):

A diagram of a diagram of a diagram

AI-generated content may be incorrect.

Figure 1. 12 Interaction Loop Between Agent and Environment

* We can expand this infinite series as follows

A red line with black text

AI-generated content may be incorrect.

Figure 1. 13 Total Reward Accumulation

* Since the chain does not converge by itself, researchers add a discount factor to ensure that the chain converges. Convergence is a mandatory requirement to successfully train an agent or Neural Network.

A diagram of a mathematical equation

AI-generated content may be incorrect.

Figure 1. 14 Discounted Total Reward in Reinforcement Learning

## Q-learning

### Overview

Q-Learning is a model-free reinforcement learning algorithm that enables an agent to learn an optimal policy through intelligent interactions with the environment.

In a discrete environment with a limited set of actions (e.g., in a video game), the algorithm uses a Q-table:

* Rows correspond to states.
* Columns correspond to actions.
* The values in the table represent the expected reward of taking a specific action in a particular state.

By updating the Q-values in the table after each interaction, the agent gradually learns to prioritize actions based on their expected long-term rewards.

**A white grid with black lines

AI-generated content may be incorrect.The way Q-Table works:**

Figure 1. 15 Q-Table

**- Step 1:** Initialize the Q table and fill it with initial values.

**- Step 2:** Start a learning session.

**- Step 3:** The agent performs the action.

**- Step 4:** Measure the reward received.

- **Step 5:** Move to a new state.

- **Step 6:** New Q calculates the value for the new state.

**- Step 7:** The learning session ends with a failure, victory, or timeout.

- **Step 8:** The environment is reset.

- **Step 9:** Repeat steps 2 through 8 for the desired number of learnings.

### How does the agent choose to act?

Initially, since all Q values ​​are zero, the first action is chosen randomly. In subsequent steps, the agent balances exploration and exploitation using the hyperparameter epsilon (usually 0.1). If the random number is less than epsilon, the agent chooses a random action; otherwise, it chooses the action with the highest Q value. This approach is called an epsilon-greedy policy, which decides on the action and updates the Q value.

### How are rewards measured?

The rules for calculating rewards are set by the environment designer, such as rewards for performing the right action and penalties for performing the wrong action. However, most of the time there will be no reward unless a special action occurs.

### How to update Q table

Review and explain step by step:

**A group of black text

AI-generated content may be incorrect.**

**A close up of a text

AI-generated content may be incorrect.**The first part of the formula:

This part says, “Given the previous state and action, the new Q value is computed as (...).”

A white rectangular sign with black text

AI-generated content may be incorrect.The lower part is the current (soon to be old) Q value that the agent used to perform the action.

Final part:

A black text on a white background

AI-generated content may be incorrect.

When the action is performed, a new state Sₜ₊₁ appears, and the new Q-value is calculated by taking the largest Q-value in the new state, multiplying it by the gamma discount factor (γ), plus the reward received.

Gamma adjusts the priority between current and future rewards. In addition, the learning rate alpha (α) controls how quickly the Q-value is updated: a high α value makes learning faster, a low α makes it more stable.

Balancing gamma and alpha is an important factor for effective Q-learning.

### Deep Q-Networks (DQN), SARSA and policy-based algorithms

* + - * 1. **Deep Q-Networks (DQN**):

It is an extended version of Q-Learning, using neural networks instead of a Q-table to estimate Q-values. This method is suitable for environments with large state spaces, helping the agent generalize and make good decisions even for new states.

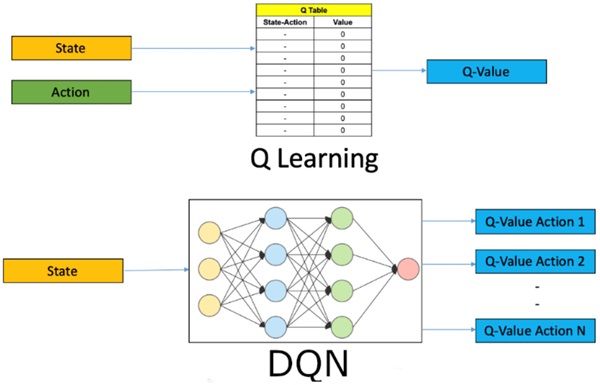


Figure 1. 16 Value-based algorithms applied

* + - * 1. **SARSA**:

It is an On-Policy algorithm in reinforcement learning. It updates the Q-value based on the actual action the agent chooses according to the current policy, rather than optimizing based on the best action like Q-Learning. As a result, SARSA is more suitable for problems that require safe and stable behavior, especially in high-risk environments.

* + - * 1. **Policy-Based Algorithms:**

Instead of learning the value of each state, this group of algorithms focuses directly on optimizing the policy – that is, the rule for the agent to choose actions in each state.

These algorithms update the policy directly to maximize the reward, with notable examples such as REINFORCE, PPO, TRPO, Actor-Critic, A2C, DDPG, and TD3.

## Reinforcement Learning algorithms used in large language models

### Initial training background of LLMs

Large Language Models (LLMs) such as GPT-3 and GPT-4 are initially trained using supervised learning and self-supervised learning methods. During this phase, the model's tasks typically include:

* Predicting the next token based on context.
* Optimizing the probability of generating responses that resemble the training data.

Although this approach helps the model learn vast linguistic knowledge, it has several limitations:

* Not optimized for usefulness: A response may be syntactically correct but not meet the user's needs.
* Safety concerns: The model may generate harmful, toxic, or misleading content.
* Does not reflect social norms: Responses may lack ethics, be biased, or controversial.
* Lacks understanding of user intent: The model might give mechanical answers, far from the actual user intent.

### Reinforcement Learning A New Direction

To overcome the above limitations, Reinforcement Learning (RL) is introduced into the model refinement phase. Because in RL:

* The model (agent) performs the action (e.g., generates an answer).
* The environment (environment) provides feedback in the form of a reward.
* The goal of the model: Learn to act in a way that maximizes the total reward over time.

Apply to LLMs:

* The model generates a prompt-based response.
* A rating system (human or AI) scores the response.
* The model updates the weights based on that “satisfaction” level.

**What it means:** Reinforcement Learning (RL) helps the model not only generate natural text, but also generate text that is more consistent with human values ​​and expectations.

### Challenge - Definition of rewards for LLMs

The application of RL in LLMs faces a major challenge: “How to determine the appropriate reward?”

* In games or robots, rewards can be clearly defined (scoring points, winning matches, etc.).
* But in text generation, "good answers" are abstract, multidimensional, and subjective.

For Example:

* An answer can be correct but boring.
* An answer is creative but a bit lacking in rigor.
* An answer is polite but lacking in depth.

🡺 Therefore, a mechanism is needed to incorporate human evaluation criteria into the reward system. To solve this problem, two main methods have emerged:

* Reinforcement Learning from Human Feedback (RLHF): Using evaluations from real humans to guide the model.
* Reinforcement Learning from AI Feedback (RLAIF): Using evaluations from pre-trained AI models (reward models).

## 1.6 Reinforcement Learning from Human Feedback (RLHF)

### 1.6.1. What is RLHF?

Reinforcement Learning from Human Feedback (RLHF) is a machine learning (ML) technique that leverages human feedback to optimize ML models, enabling more effective self-learning.

Reinforcement learning (RL) techniques train software to make decisions that maximize rewards, resulting in more accurate outcomes. RLHF incorporates human feedback into the reward function, allowing ML models to perform tasks that better align with human goals, preferences, and needs.

RLHF is used in generative artificial intelligence (generative AI) applications and is integrated into large language models (LLMs).

### 1.6.2. Importance of RLHF

AI is widely applied with the goal of simulating human reactions and behaviors. To achieve this, AI needs to learn from data and human feedback.

Reinforcement Learning from Human Feedback (RLHF) is a technique that helps AI become more human-like by using human evaluations to improve the model's responses.

RLHF is especially prominent in the field of natural language processing (NLP) and various other generative AI applications.

**Role of RLHF:**

* ***Enhancing AI performance:***
* RLHF makes ML models more accurate and natural by incorporating user feedback into the training process, going beyond learning from available data alone.
* Example: improving language translations to sound more natural.
* ***Handling complex parameters:*** For hard-to-quantify factors like "mood" in music, RLHF enables faster learning through human-provided labels.
* ***Improving user satisfaction:*** RLHF helps models generate responses that are not only accurate but also more friendly, natural, and pleasant for users.

### 1.6.3. How does RLHF work?

RLHF consists of four main stages to fine-tune an AI model (e.g., a chatbot using an internal knowledge base):

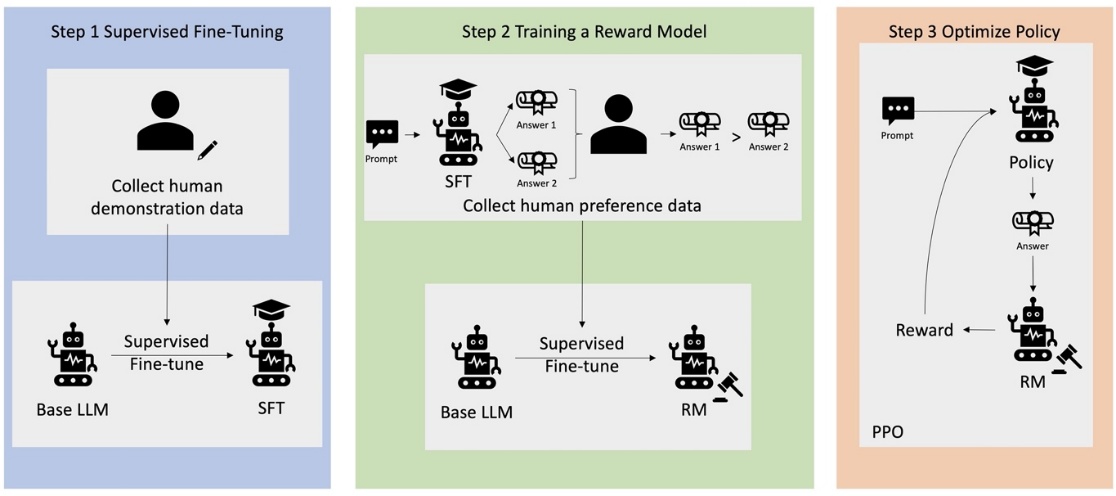
* ***Data collection:*** Generate a set of human-written prompts and responses, which serve as initial training data for the model.
* ***Supervised language model fine-tuning:***
* Take a pre-trained commercial model, and further fine-tune it using internal data (e.g., using RAG techniques).
* Compare machine-generated responses to human responses and assign a similarity score.
* ***Reward model building:***
* Train a separate reward model based on human ratings of responses.
* This model estimates the score a human would give each response.
* ***Language model optimization:***The language model uses the reward model to evaluate the responses and select the best one - that is, optimize to match human responses and expectations.

Figure 1. 17 An overview of the RLHF learning process

### 1.6.4. Formulas of RLHF (Reinforcement Learning from Human Feedback)

The standard approach to RLHF is to construct a reward function and maximize (objective):

In the formula, there are two main parts that guide how to fine-tune the model:

* ***Reward Maximization:*** The part of the formula is about maximizing the expected reward. The model is motivated to choose the response that receives the highest reward according to the function r, which evaluates the fit between the response y and the input x.
* ***Deviation Minimization:*** The is used to control the behavior of the model. This term calculates the difference or divergence between the model's current response strategy and the reference strategy . The model will be penalized if its strategy strays too far from this reference, ensuring that as it learns to improve, it doesn't start generating strange or unexpected responses.
* The goal is to fine-tune the model to both maximize the reward for good responses, while keeping the responses within safe limits by penalizing standard deviations, often optimized using reinforcement learning (PPO).

**RLHF has a 3-step process that covers the entire RLHF pipeline:**

1. Supervised Fine-tuning (SFT):

* First, train the model using supervised learning on the instruction-following dataset. Loss function:
* : The dataset consists of (prompt, answer) pairs.
* : Policy (generative model) with parameter .
* Meaning: Given prompt x, maximize the probability of producing the correct answer y.

1. Reward Model (RM) Training:

* Suppose there are 2 answers , for the same prompt x, the evaluator chooses as better. Reward model is trained to assign higher score to the good answer.
* Loss to train reward model:
* Where:

is sigmoid function.

* Meaning:

If > then the loss is small 🡪 reward model is encouraged to assign higher score to good answer.

1. Optimizing Policy using RL (usually using PPO):

* Objective: Optimize the generative model to maximize the total reward according to the reward model.
* RL objective function:
* : Reward score for output y.
* : Original policy before RL (e.g. after SFT).
* KL divergence: Keep the new policy from deviating too much from the old policy.
* : Parameter to adjust the penalty.

d) Process:

* Train the initial model using supervised learning (SFT).
* Train the reward model from human feedback.
* Use RL (PPO) to optimize the reward-based model.

### 1.6.5. RLHF Applications in Generative AI

RLHF is a standard technique for large language models (LLMs) to generate truthful, safe, and useful content.

Since human communication is subjective and creative, different models (trained from different sets of user feedback) will produce different outputs, depending on the values ​​and choices of the trainer.

RLHF is not limited to LLMs, but also applies to:

* AI image generation: assessing authenticity, expertise, mood.
* Music composition: creating music to match mood or for specific activities.
* Voice assistants: making feedback more friendly, private, and trustworthy.

## 1.7 Reinforcement Learning with AI Feedback (RLAIF)

### 1.7.1. What is AI Feedback Reinforcement Learning (RLAIF)?

RLAIF is a machine learning technique where AI models provide feedback to other AI models during reinforcement learning.

Instead of relying solely on data from humans, RLAIF leverages existing AI systems (e.g. large language models) to evaluate and guide other agents.

AI feedback can be in the form of generating rewards, ratings, or suggesting improvements.

RLAIF helps automate training, reduce costs, and improve AI system performance.

### 1.7.2. How does RLAIF work?

The RLAIF implementation process consists of four main stages, each of which plays an important role in helping AI models learn from AI-generated feedback.

**Step 1:** Training the feedback model (optional)

* An “off-the-shelf” LLM is used as the feedback model for preference labeling.
* In some cases, especially with specialized terminology (finance, healthcare), the LLM can be fine-tuned on relevant data to improve its understanding capabilities.
* This fine-tuning improves the quality and reliability of the AI ​​system in specialized applications.

**Step 2:** Generating the AI ​​response:

* Using the LLM feedback model to generate a preference label.
* The feedback model is put into context with two candidate responses and determines the preferred response.
* The input prompt structure includes: instructions, examples, input context, response pairs, and optional labels.

**Step 3:** Training the preference model:

* After collecting the preference labels from the LLM feedback model, the next step is to use these labels to train a preference model.
* The preference model learns the mapping between the context, the candidate's feedback, and the scalar reward signal, to simulate human preferences. This model acts as the evaluation component, integrated into the reinforcement learning cycle.

**Step 4:** Reinforcement learning with AI feedback:

* After obtaining the preference model, reinforcement learning is conducted by using the AI ​​feedback as the reward signal.
* In this, the agent (base LLM) will interact with the environment, generating responses. Each response is evaluated by the preference model and assigned a reward based on how well it matches the desired preference learned from the previous feedback data.

### 1.7.3. RLAIF Formulas

RLAIF is similar to RLHF, but instead of human evaluation, uses an AI model (Judge Model) to automatically score the output.

The formula RLAIF is very similar to RLHF, with only one change:

Instead of being trained by humans, it will be taken from an AI scoring model.

The RL loss in RLAIF is also:

Where:

is a reward model trained by AI, without human labeling.

### 1.7.4. Comparison of RLAIF & RLHF

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | | **RLHF** | **RLAIF** |
| **Source of Feedback** | | Human annotators | Existing AI models (e.g., LLMs) |
| **Scalability** | | Limited by availability and cost of human labor | Highly scalable due to automation |
| **Feedback Quality** | High potential for capturing nuanced human preferences | | Depends on the feedback quality of the AI model |
| **Cost** | Potentially expensive due to reliance on human labor | | More cost-effective due to automation |
| **Speed** | Slower due to time required for human annotation | | Faster with automated feedback generation |

Table 1. 1 Comparison of RLAIF & RLHF

### A diagram of a diagram AI-generated content may be incorrect.

Figure 1. 18 RLAIF: Scaleing Reinforcement Learning from Human Feedback with AI Feedback

Differences between RLHF and RLAIF:

* Both RLHF and RLAIF start from an initial model (SFT) that generates sample responses.
* RLHF: humans evaluate and rank the responses.
* RLAIF: an off-the-shelf large language model (LLM) performs the evaluation.
* These evaluations are used to train a reward model (RM).
* The RL model then learns and improves based on the rewards from the RM, depending on the feedback from humans (RLHF) or AI (RLAIF).

A graph with red and blue bars

AI-generated content may be incorrect.Potential for performance improvement: Initial experimental results show that RLAIF performs on par with RLHF and in some cases outperforms it. This suggests that RLAIF not only simplifies the feedback collection process, but also has the potential to improve the overall performance of large language models.

Figure 1. 19 RLAIF produces the highest percentage of harmless conversations compared to RLHF and SFT

### 1.7.5. Applications of RLAIF

RLAIF is applied to many natural language processing tasks, including:

* **Text summarization:** RLAIF helps models prioritize important information, ensuring brevity, relevance, and accuracy in summaries.
* **Dialogue generation:** AI feedback guides the dialogue model to respond in a useful, safe, and engaging way, while avoiding generating harmful or inappropriate content.
* **Question answering:** RLAIF helps train systems to answer questions with accurate, concise answers and minimize the risk of misunderstanding.
* **Creative content generation:** RLAIF adjusts the generation of stories, articles, poems, etc. to match the desired style and genre conventions.

After learning about RLHF and RLAIF, we see that one of the key factors in the model training process is how to optimize the model behavior based on rewards and control the behavior deviation. To understand this mechanism in more depth, we need to delve into the main optimization techniques such as Proximal Policy Optimization (PPO), Direct Preference Optimization (DPO) and the concept of KL divergence — core tools that help balance between improving the quality of feedback and maintaining stability and safety in the model behavior.

## 1.8 Direct Preference Optimization – DPO

### 1.8.1. What is Direct Interest Optimization (DPO)?

Direct Preference Optimization (DPO) is a novel approach that helps large language models (LLMs) better match human preferences using simple classification, rather than complex processes such as RLHF.

DPO uses a binary cross-entropy objective to guide the model to generate preferred responses, eliminating the need to build and tune separate reward models. As a result, DPO simplifies the training process, reduces computational power, and increases development efficiency.

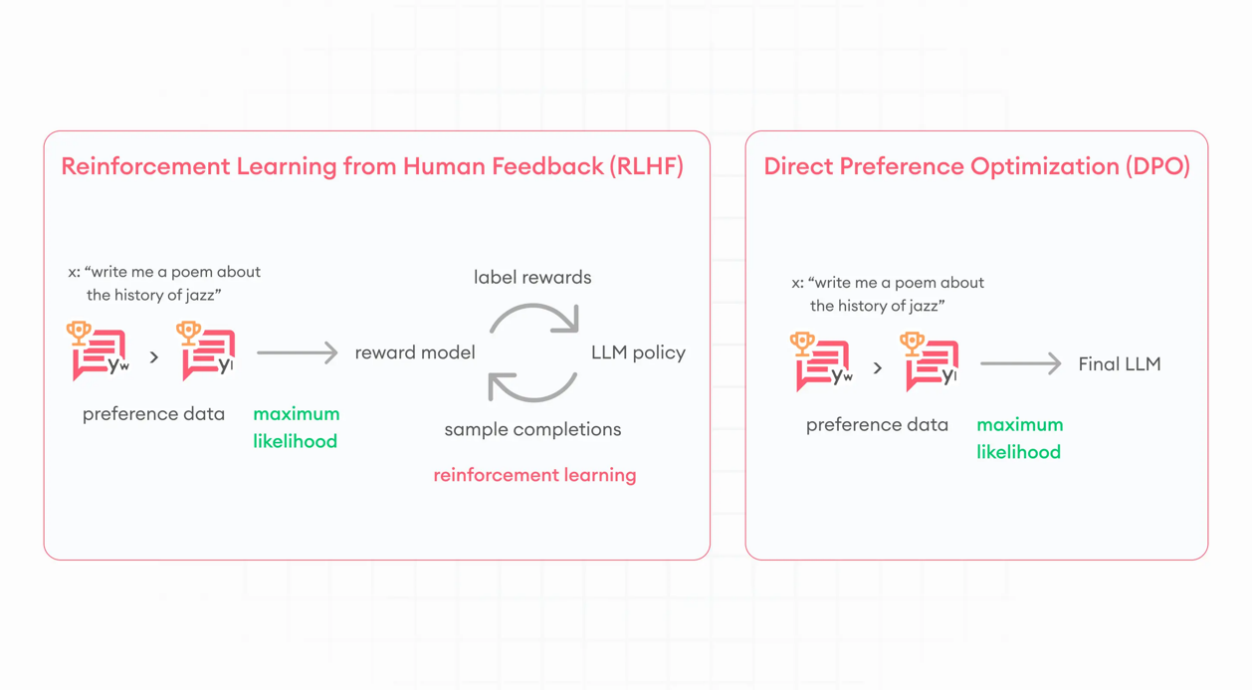


Figure 1. 20 Direct Preference Optimization

### 1.8.2. How does DPO work?

DPO treats the language model as a reward model, eliminating the need for a separate reward model and complex reinforcement learning algorithms. The process involves:

(1) The model generates response pairs for prompts;

(2) Humans label the more desirable responses;

(3) The model is fine-tuned using a cross-entropy loss function, to favor positive responses while preserving the original model's ability.

Figure 1. 21 Direct Preference Optimization Process

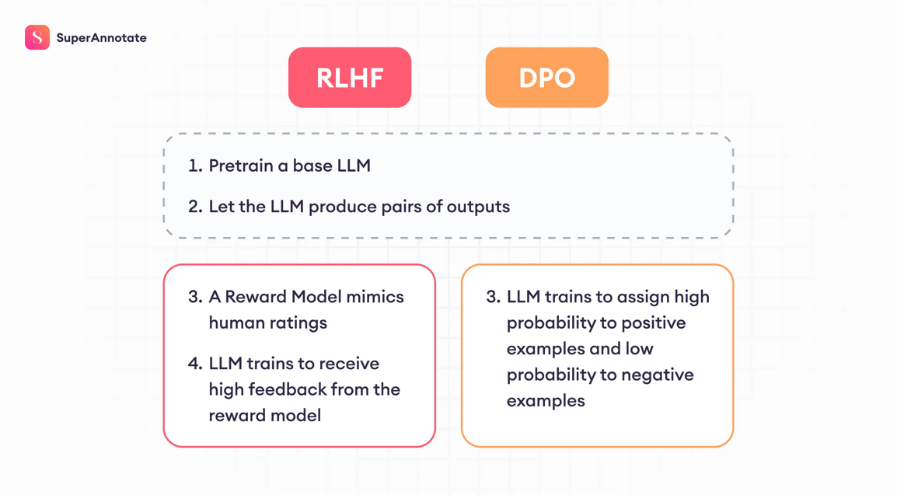
In DPO, the first two steps are similar to RLHF, but the difference lies in the last step: constructing the loss function. Here, the model is optimized to assign high probability to positive feedback and low probability to negative feedback.

Figure 1. 22 Comparison between RLHF and DPO Training Pipelines

### 1.8.2. DPO loss function: Elimination of reward model

Equation:

* + **và** These terms represent the probabilities that the model, parameterized by θ, ssigns to specific responses ( for preferred or higher response, for preferred or lower response) given the input x.
  + **và** These parameters refer to the probabilities given by the policy or the reference model (which can be the baseline or untrained model) for the same response given the same input. This helps assess how much the current model's predictions deviate from the baseline.
  + **:** This is where the optimization takes place. The model updates itself to increase the probability of the preferred response () relative to the baseline while decreasing the probability of the less preferred response (). The coefficient β here acts as a balancing factor, determining how much the model should adhere to or deviate from the reference behavior.
  + E[...]: This notation indicates that we take the expected value of the internal expression over all possible inputs and outputs. This means we are averaging the adjustments across all scenarios the model encounters to find the optimal overall behavior.

🡺 DPO helps the model learn to prioritize human-preferred responses and avoid undesirable responses, while keeping the output within the bounds of the reference policy. Instead of using a reward model like RLHF, DPO applies classification directly via a binary cross-entropy loss function. The main innovation of DPO is the reparameterization of the loss function, which allows for simple and efficient fine-tuning of the model policy and avoids problems such as reward hacking or behavioral bias.

### 1.8.3. Can DPO scale to real-world interest datasets?

Automated metrics like ROUGE do not accurately reflect human judgment in summarization tasks. When compared on the TL;DR set, DPO achieved a win rate of 61%, slightly better than PPO (57%), and maintained more stable performance across various temperature settings, demonstrating superior stability over PPO.

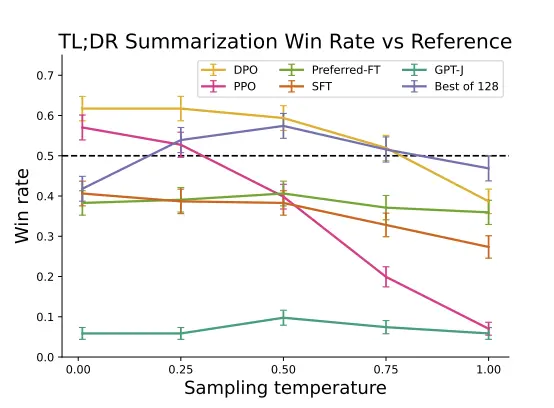


Figure 1. 23 TL; DR summary win rate vs human-written summaries, using GPT-4 as the evaluator. DPO outperforms PPO's best summary performance

### 1.8.4. The effectiveness and potential of DPO in language model tuning

* **One-way dialogue performance with DPO:** DPO is evaluated on the Anthropic HH set and shows results that are equal to or better than other methods, including Best of 128. DPO not only improves continuously but also converges quickly to optimal performance.
* **Why DPO is important:** DPO enables faster, cheaper, and more stable LLM tuning by eliminating the need to build a separate reward model, which improves the interoperability of AI in the real world.

🡺 DPO represents a major step forward in language model tuning, promising to become the new standard due to its simplicity, efficiency, and potential to reduce technical barriers

### 1.9 Proximal Policy Optimization - PPO

### 1.9.1. Proximity Policy Optimization – PPO

Proximal Policy Optimization (PPO) is a family of first-order policy optimization methods aimed at efficiently updating policies while controlling changes to avoid performance degradation. PPO is simple to implement and yields very good experimental results.

**There are two main variants of PPO:**

* + PPO-Penalty: Directly penalizes the KL-divergence deviation in the objective function and automatically adjusts the penalty coefficient during training to ensure the new policy is not too different from the old policy.
  + PPO-Clip: Does not use KL-divergence or any constraints. Instead, it applies clipping in the objective function to prevent the new policy from deviating too far from the old one. (In practice, PPO-Clip is more commonly used, for example, at OpenAI).

**Key points about PPO:**

* + It is an on-policy algorithm (using only data from the current policy).
  + It can be applied to environments with either discrete or continuous actions.
  + It supports parallelization with MPI in libraries such as Spinning Up.

### 1.9.2. Main formulas

PPO-clip policy updates via:

,

Often perform multiple (usually minibatch) steps of SGD to maximize the objective.

L is given by

,

Where ϵ is a (small) hyperparameter that roughly represents how far the new policy is allowed to deviate from the old policy.

The original expression is quite complex and it is difficult to understand how it keeps the new policy close to the old policy. However, there is a simpler version which is easier to understand and is also used in real implementations.

, )

With:

G( =

To understand the "intuition" behind this, let's consider a single state-action pair (v,v) and think about the cases.

**Positive Advantage:** Suppose the advantage for that state-action pair is positive. In that case, its contribution to the objective is reduced.

)

When the advantage is positive, the objective will increase if the action probability increases, but the formula limits the increase by clipping at the threshold (1+ϵ). This ensures that the new policy is not encouraged to deviate too much from the old policy.

**Negative Advantage:** Suppose the advantage for that state-action pair is negative. In that case, its contribution to the objective is reduced.

)

When the advantage is negative, the objective will increase if the action probability decreases, but the formula limits the increase when the action probability decreases too much, with the clipping threshold set at (1-ϵ). The new policy is not encouraged to deviate too far from the old policy.

Clipping helps adjust the policy by removing the incentive for large changes, with the hyperparameter 𝜖 determining the maximum distance the new policy can differ from the old policy while still achieving the objective.

### 1.9.3. Exploration vs Exploitation:

PPO trains policies in an on-policy manner, discovering random actions based on the latest policy. During training, the policy becomes less random and more biased towards exploiting the found reward, which can lead to getting stuck in a local A black and white text with black text

AI-generated content may be incorrect.optimum.

Figure 1. 24 Pseudocode

## 1.10 Kullback-Leibler divergence - (KL divergence) or relative entropy

### 1.10.1. Definitions

Given two discrete probability distributions P and Q on the same sample space X, the Kullback–Leibler difference is defined as:

For the continuous case, the analogous formula is the integral:

.

### 1.10.2. Main properties

* Non-negative:

,

and is zero if and only if P = Q almost everywhere.

* Asymmetric:

,

so it is not a standard metric distance.

-Relation to cross entropy:

H(P, Q) = H(P) + ,

Where:

* H(P) = is the entropy of P.
* is the cross-entropy betwee P và Q.

### 1.10.3. Loss function for neural network

In regression, we use MSE, while in classification, KL-divergence is used as the loss function to compare the model's output distribution with the true label distribution.

For example, a binary cat classifier predicts the probability for an image as p(cat) = 0.8 and p(no cat) = 0.2. If the true label of the image is that it is a cat, then the true distribution becomes p(cat) = 1 and p(no cat) = 0. Now we have two different distributions modeling the same variable. The neural network aims to bring these predicted distributions as close to the true label as possible.

Taking the values from the example above, P(X) = {1, 0} and Q(X) = {0.8, 0.2}, the divergence will be:

Substitute the corresponding values:

= - = 0.0969

When using KL-divergence as the loss, the network will optimize to minimize the divergence towards 0.

### 1.10.4. Optimization of variational autoencoder

Autoencoders compress the input data into an embedding layer, while variational auto-encoders (VAEs) project the data into a probability distribution, usually a Gaussian distribution.

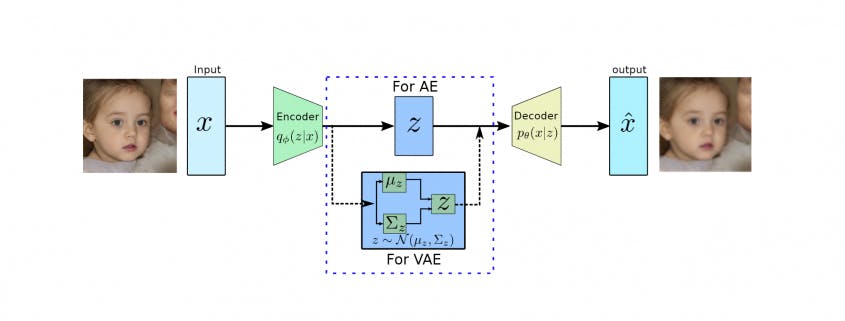


Figure 1. 25 Architecture Comparison between Autoencoder (AE) and Variational Autoencoder (VAE)

VAE uses two loss functions: MSE to measure the difference between the output image and the real image, and KL divergence to measure the distance between the real distribution and the approximate distribution.

### 1.10.5. Reproductive antagonist network

GANs consist of two adversarial networks: one that generates fake data that looks like real data, and one that distinguishes between fake and real data

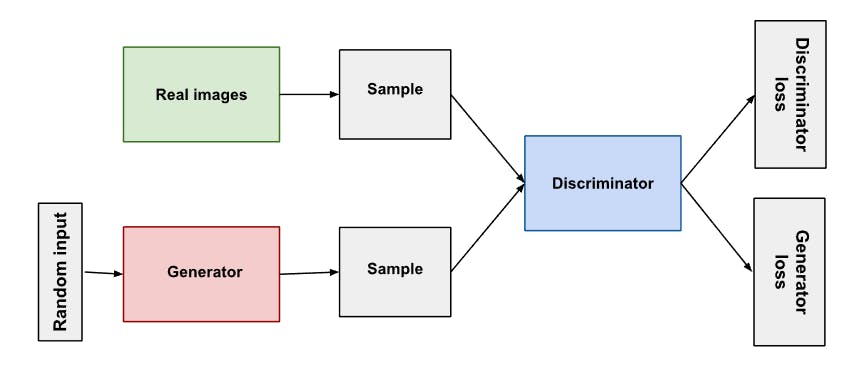


Figure 1. 26 Architecture of a Generative Adversarial Network (GAN)

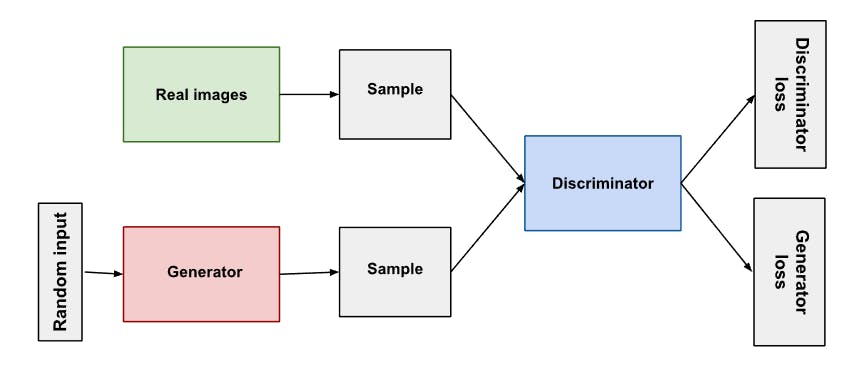
In GANs, real and fake data are compared using KL divergence to evaluate the learning process. The discriminator tries to maximize the divergence, while the generator tries to minimize it. However, due to the disadvantages of KL such as asymmetry and training instability, Jensen-Shannon divergence is often preferred.

**Example:**

* A close up of words

  AI-generated content may be incorrect.As training begins, the generator generates obvious fake data and the discriminator quickly learns to recognize fake data:
  + A red and green text

    AI-generated content may be incorrect.As training progresses, the generator gets closer to producing output that can fool the discriminator:
* Eventually, if the generator training goes well, the discriminant value will become less effective at distinguishing between real and fake. The model will start classifying fake data as real data and the model's A close-up of green text

  AI-generated content may be incorrect.accuracy will decrease.
  + Whole system:

## 1.11 Compare the algorithms with each other

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Algorithm** | **Purpose** | **Advantages** | **Disadvantages** | **Applications** |
| **RLHF** | Fine-tuning the model with human feedback. | Brings the model closer to the real values and expectations of users. | Expensive and slow due to the need for extensive human feedback. | Fine-tuning LLMs (e.g., ChatGPT, Claude). |
| **RLAIF** | Fine-tuning the model with automated feedback from another AI model. | Cost-effective, better scalability than RLHF. | Dependent on the quality of the evaluation model (Judge Model). | Large-scale automatic fine-tuning of LLMs (e.g., Gemini, Claude 3). |
| **DPO** | Direct optimization from preference labels, no intermediate reward model required. | Simple, fast, and cheap. No need to train a reward model. | Not suitable for extremely complex tasks or multi-objective optimization. | Fast, low-cost LLM optimization (e.g., summarization, dialogue). |
| **PPO** | Safely and stably updates the policy in RL. | Easy to implement, stable learning. Popular in practice. | Requires many mini-batch update rounds, can be slow to learn. | Core in RLHF for training LLMs; used in games (Atari), robotics. |
| **KL-Divergence** | Measures the deviation between two probability distributions (old and new policies). | Easy to compute, helps control the policy from changing too drastically. | Asymmetric, can be hard to optimize if the distributions differ too much. | Constraints between old and new policies in PPO, DPO, RLHF. |

Table 1. 2 Compare the algorithms with each other

## 1.12 DEMO

### 1.12.1 Dataset

"DailyDialog" is a high-quality English conversation dataset consisting of approximately 13,000 short conversations set in everyday life, designed to train and evaluate conversation models.

Each segment consists of a sequence of turns between two people, which is educational, polite, and natural. The data is organized as a list of conversations, where each element is a sequence of turns.

It covers a wide range of topics such as daily communication, relationships, and social interactions, making it useful for building emotionally intelligent AI systems. Furthermore, the dataset includes labeled emotion and act tags, which help in tasks like emotion recognition and dialogue act classification.

### 1.12.2. Demo code for Q-learning

**Description:** The code uses the Q-learning algorithm to find a path in a 15x15 maze, with values: 0 (path), 1 (wall), 2 (start), 3 (end). The Q-learning parameters include actions (up, down, left, right), Q-table (q\_table), discount factor (gamma), learning rate (alpha), and exploration probability (epsilon). The code trains the robot through multiple episodes, updates the Q-table, and finds the optimal path from the start point to the end point. After training, the code executes the optimal path and displays the maze with the path marked by color. and updates the Q value while exploring the environment.

A maze with a red square in center

AI-generated content may be incorrect.**The original maze:**

Figure 1. 27 The original maze

**Result:**

A screenshot of a game

AI-generated content may be incorrect.

Figure 1. 28 Results

### 1.12.3. Demo code for RLHF

Description: Build a conversational language model (GPT‑2) that is refined not only by supervised learning but also based on human feedback, making the quality of the response more natural and suitable for the request. Through the processing steps:

* SFT ensures a solid conversational foundation.
* RM provides a human-perspective quality measure.
* PPO combines the above two parts to create a conversational model that both understands the context and optimally adapts to real-world feedback.

**Result:**

A screen shot of a computer

AI-generated content may be incorrect.A screenshot of a computer program

AI-generated content may be incorrect.

Figure 1. 30 The Comparison result

Figure 1. 29 Fine-tuning

# CHAPTER 2: MACHINE TRANSLATION MODEL

## 2.1 Introduction

A hand writing on a piece of paper

AI-generated content may be incorrect.Machine translation has witnessed remarkable advancements with the development of neural networkbased approaches. The introduction of transformer architectures has established new performance benchmarks across numerous language pairs. However, translation between typologically distinct languages like English and Vietnamese presents unique challenges requiring specialized solutions.

Figure 2. 1 Machine translation

**English and Vietnamese differ substantially in their linguistic properties:**

* English belongs to the Germanic branch of the Indo-European language family
* Vietnamese is an Austroasiatic language with distinctive characteristics:
* A tonal system with six tones that change word meanings
* Monosyllabic morphology with complex compounding
* Isolating grammatical structure with minimal inflection
* Classifier-based noun system without grammatical gender or number
* Pragmatic pronoun system based on social hierarchies

This report explores the development of an English-Vietnamese neural machine translation system through two complementary approaches:

1. Utilizing pretrained transformer-based models

2. Implementing fine-tuning techniques to enhance performance:

Our evaluation employs **BLEU,** **ROUGE-L, ROUGE-1, ROUGE-2** and metrics to provide a comprehensive assessment of translation quality

## 2.2 Dataset Description

### 2.2.1 Overview

We chose OPUS-100 – a large-scale multilingual corpus developed as part of the ***OPUS project***. This project focuses on creating open parallel corpora for tasks related to **Machine Translation** and **Multilingual Natural Language Processing**. OPUS-100 stood out to us because of its wide language coverage, practical scale, and open-access format, which are highly suitable for training and evaluating multilingual translation models in our project.

The OPUS-100 dataset is constructed from a variety of corpora, including ***movie subtitles***, ***GNOME*** documentation, ***TED*** talks, and even the ***Bible***. Instead of curating or balancing the dataset across domains, the creators chose a straightforward approach: they downloaded and concatenated all available corpora for each language pair. While this results in domain imbalance, it helps maximize language coverage and volume, which is critical for many multilingual NLP tasks.

The dataset is divided into three parts: ***training***, ***development***, and ***tests***. For each language pair, up to 1 million sentence pairs were randomly selected for training, and 2,000 pairs for both dev and test.

And in this part, we used **English – Vietnamese** pair for Machine Translation problem:

- 1% randomly of dataset was used for Fine-Tuning Pretrained Model (Approach 1).

- 10% of dataset was used for Building From Scratch (Approach 2).

### 2.2.2 Data Preprocessing

**Step 1:** Check “Null” values

A black background with white text

AI-generated content may be incorrect.

Figure 2. 2 Check “Null” values

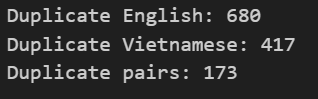
**Step 2:** Check duplicated values

Figure 2. 3 Check duplicated values

**Step 3:** Check lengths

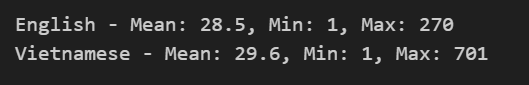


Figure 2. 4 Check lengths

A black background with white text

AI-generated content may be incorrect.**Step 4:** Check outlier

Figure 2. 5 Check outlier

A black background with white text

AI-generated content may be incorrect.A graph with numbers and a number of text

AI-generated content may be incorrect.

Figure 2. 6 Text Length

Figure 2. 7 Check character sets

**Step 5:** Remove extra whitespace

**Step 6:** Remove special characters but keep punctuation

**Step 7:** Normalize Unicode (important for Vietnamese)

**Step 8:** Lowercase

**Step 9:** Remove entries with empty strings after cleaning

**Step 10:** Remove duplicates

**Step 11:** Remove extremely long or short entries

**Step 12:** Check for mismatched lengths (possible alignment issues)

A black background with white text

AI-generated content may be incorrect.

Figure 2. 8 Clean dataset

## 2.3 Approach 1: Fine-Tuning Pretrained Model

### 2.3.1 Pretrained Model Selection

In our project, we chose **VietAI/envit5-translation** as the base model. This model is built on the **T5 encoder-decoder architecture** and has been pre-trained specifically on **English Vietnamese parallel datasets**, making it highly suitable for our translation task. It uses a shared vocabulary of **32,000 tokens** and consists of around **220 million parameters**, which balances performance and computational efficiency quite well for our experiments.

### 2.3.2 Quantization Implementation

To make the fine-tuning process more efficient, we applied **4-bit quantization** using the bitsandbytes library. This method helps reduce the model's weight precision from standard **32-bit floating-point** to **4-bit integers**, which greatly lowers memory usage. Despite the reduced precision, the model still maintains a good level of performance, making it a practical choice for training on limited hardware resources.

### 2.3.3 LoRA Fine-tuning

We employed LoRA for parameter-efficient fine-tuning [8], which introduces trainable low-rank matrices to adapt the pretrained weights. This approach significantly reduces the number of trainable parameters while still allowing effective model adaptation.

We applied LoRA adapters to the attention layers with rank r = 16 and scaling factor α = 32, resulting in only 2.4% of the original parameters requiring gradient updates during training.

### A graph of different colored bars AI-generated content may be incorrect.2.3.4 Results

Figure 2. 9 Results of Fine-Tuning Pretrained Model

## 2.4 Approach 2: Building From Scratch

### 2.4.1 Transformer Encoder-Decoder Architecture

In this project, we also built a **Transformer-based encoder-decoder architecture from scratch**. However, we made several adjustments to better suit our task and computational constraints:

* **Embedding dimension (d\_model):** 256
* **Number of encoder/decoder layers:** 4
* **Number of attention heads:** 8
* **Feedforward network dimension:** 512
* **Dropout rate:** 0.1

To ensure the model handles **variable-length inputs** correctly, we added **source padding masks** in the encoder. In the decoder, we used **causal masks** to prevent the model from attending to future tokens during training, which is essential for autoregressive generation.

### 2.4.2 Tokenization with BPE

To better handle the differences between the two languages, we developed **separate BPE (Byte Pair Encoding) tokenizers** for English and Vietnamese. This approach allows us to perform **language-specific subword segmentation**, which is especially important for languages like Vietnamese that use diacritical marks and compound words.

The BPE algorithm works by iteratively merging the most frequent pairs of bytes or characters to form new subword units. Our implementation followed these main steps:

1. **Initialize the vocabulary** with all individual characters in the corpus, along with special tokens like <PAD>, <UNK>, <SOS>, <EOS>, and a word boundary marker (\_).
2. **Count the frequency of adjacent byte pairs** in the training corpus.
3. **Merge the most frequent pair**, update the vocabulary and recalculate frequencies.
4. **Repeat** this merging process until reaching the target vocabulary size of **10,000 tokens**.

During encoding, each word is segmented by **greedily applying the learned merge operations**, which allows the model to:

* Effectively handle Vietnamese characters with tone and diacritics.
* Split rare or unknown words into more common subword units, improving generalization.
* Preserve the core **semantic meaning** of words while significantly reducing the vocabulary size.

### 2.4.3 Training Methodology

The model was trained using the following configuration:

* ***Optimizer:*** AdamW with learning rate 5e-5 and weight decay 0.01
* ***Learning rate scheduler***: ReduceLROnPlateau with patience of 3 epochs
* ***Loss function:*** Cross-entropy with padding token masking
* ***Early stopping:*** With patience of 3 epochs
* ***Maximum epochs:*** 20

### 2.4.4 Results

Evaluated on 5918 test samples

* ***BLEU Score:*** 0.0918
* ***ROUGE-1 Score:*** 0.4403
* ***ROUGE-2 Score:*** 0.2123
* ***ROUGE-L Score:*** 0.3911

A screenshot of a computer screen

AI-generated content may be incorrect.

Figure 2. 10 The results of translation

## 2.5 Evaluation Metrics

We employed two primary metrics to evaluate translation quality

### 2.5.1 BLEU (Bilingual Evaluation Understudy)

- ***BLEU (Bilingual Evaluation Understudy)*** is a metric that checks how many n-grams from the model’s translation appear in reference translations. To avoid rewarding overly short outputs, it also applies a brevity penalty when the translation is much shorter than the reference.

**- Algorithm:**

1. Tokenize both the candidate and reference sentences.
2. Compute precision for each n-gram (n = 1 to 4):

* Count how many n-grams in the candidate appear in the reference.
* Precision = (number of matched n-grams) / (total n-grams in candidate).

1. Calculate the Brevity Penalty (BP):

* BP = 1 if candidate length ≥ reference length.
* BP = exp (1 - reference length / candidate length) if candidate is shorter.

1. Compute the final BLEU score:

BLEU =

### 2.5.2 ROUGE Metrics

- ***ROUGE Metrics:***

* ROUGE - L: Longest Common Subsequence between candidate and reference
* ROUGE - 1: Overlap of unigrams
* ROUGE - 2: Overlap of bigrams

**- Algorithm:**

**ROUGE-L (Longest Common Subsequence):**

1. Identify the Longest Common Subsequence (LCS) between the candidate and reference.
2. Compute Precision = LCS length / candidate length
3. Compute Recall = LCS length / reference length
4. Compute F1-score =

**ROUGE-1 & ROUGE-2 (Overlap of unigrams & bigrams):**

1. Tokenize both candidate and reference sentences.
2. Count the number of unigrams or bigrams shared between candidate and reference.
3. Compute Precision = (matched n-grams) / (total n-grams in candidate)
4. Compute Recall = (matched n-grams) / (total n-grams in reference)
5. Compute F1-score in the same way as ROUGE-L.

## 2.6 Compare and Evaluate

### 2.6.1 Comparison

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Bleu** | **Rouge-L** | **Rouge-1** | **Rouge-2** |
| **W/o FT- EnviT5** | 0.22 | 0.653 | 0.426 | 0.623 |
| **FT - EnviT5** | 0.32 | 0.66 | 0.44 | 0.63 |
| **Encoder - Decoder** | 0.09 | 0.44 | 0.21 | 0.39 |

Table 2. 1 Comparison

### 2.6.2 Evaluation and Analysis

To assess the performance of different translation models, we used common evaluation metrics in Natural Language Processing: **BLEU**, **ROUGE-L**, **ROUGE-1**, and **ROUGE-2**. These metrics capture both lexical overlap and sequence similarity between model outputs and reference translations.

The results demonstrate several key findings:

1. **Fine-tuning improves performance:** The FT-EnviT5 model, which is fine-tuned on the task-specific dataset, outperforms the W/o FT-EnviT5 baseline across all metrics. Specifically, the BLEU score increases from 0.22 to 0.32, indicating better n-gram overlap with reference translations. ROUGE scores also show slight improvements, confirming better phrase and sentence-level similarity.
2. **Pretrained models significantly outperform from-scratch models:** The custom Encoder–Decoder model trained from scratch shows much lower scores, especially in BLEU (0.09) and ROUGE-1 (0.21), suggesting that training without pretrained weights is not effective for this task, likely due to limited data or model capacity.
3. **Model stability and generalization:** The ROUGE-L scores remain relatively stable across the two EnviT5 settings (0.653 vs. 0.66), suggesting that the model’s ability to preserve longer sequence structure benefits from both pretraining and fine-tuning.

# REFERENCES

[1]. <https://www.geeksforgeeks.org/what-is-reinforcement-learning/>

[2].<https://machinelearningmastery.com/principles-of-reinforcement-learning-an-introduction-with-python/>

[3]. <https://reinforcement-learning-math.readthedocs.io/en/latest/>

[4]. <https://www.geeksforgeeks.org/q-learning-in-python/>

[5]. <https://www.nature.com/articles/s41598-024-83625-8>

[6]. <https://www.ibm.com/think/topics/rlhf>

[7]. <https://www.datacamp.com/blog/rlaif-reinforcement-learning-from-ai-feedback>

[8]. <https://www.unite.ai/direct-preference-optimization-a-complete-guide/>

[9]. <https://www.geeksforgeeks.org/a-brief-introduction-to-proximal-policy-optimization/>

[10]. <https://en.wikipedia.org/wiki/Kullback%E2%80%93Leibler_divergence>

[11]. [https://aclanthology.org/P02-1040/](https://aclanthology.org/P02-1040/?utm_source=chatgpt.com)

[12]. <https://aclanthology.org/W04-1013/>