3 Definitions & Background

This chapter includes all the definitions, symbols, and operations frequently used in the RLHF process and with a quick overview of language models (the common optimization target of this book).

3.1 Language Modeling Overview

The majority of modern language models are trained to learn the joint probability distribution of sequences of tokens (words, subwords, or characters) in a autoregressive manner. Autoregression simply means that each next prediction depends on the previous entities in the sequence. Given a sequence of tokens $x = (x_1, x_2, ..., x_T)$, the model factorizes the probability of the entire sequence into a product of conditional distributions:

$$P_\theta(x) = \prod_{t=1}^{T} P_\theta(x_t \mid x_1, ..., x_{t-1}). \tag{1}$$

In order to fit a model that accurately predicts this, the goal is often to maximize the likelihood of the training data as predicted by the current model. To do so we can minimize a negative log-likelihood (NLL) loss:

$$\mathcal{L}_{\text{LM}}(\theta) = -\mathbb{E}_{x\sim\mathcal{D}}\left[\sum_{t=1}^{T} \log P_\theta\left(x_t \mid x_{<t}\right)\right]. \tag{2}$$

In practice, one uses a cross-entropy loss with respect to each next-token prediction, computed by comparing the true token in a sequence to what was predicted by the model.

Implementing a language model can take many forms. Modern LMs, including ChatGPT, Claude, Gemini, etc., most often use **decoder-only Transformers** [48]. The core innovation of the Transformer was heavily utilizing the **self-attention** [49] mechanism to allow the model to directly attend to concepts in context and learn complex mappings. Throughout this book, particularly when covering reward models in Chapter 7, we will discuss adding new heads or modifying a language modeling (LM) head of the transformer. The LM head is a final linear projection layer that maps from the models internal embedding space to the tokenizer space (a.k.a. vocabulary). Different heads can be used to re-use the internals of the model and fine-tune it to output differently shaped quantities.

3.2 ML Definitions

- **Kullback-Leibler (KL) divergence ($D_{KL}(P||Q)$)**, also known as KL divergence, is a measure of the difference between two probability distributions. For discrete probability distributions P and Q defined on the same probability space $\mathcal{X}$, the KL distance from Q to P is defined as:

$$D_{KL}(P||Q) = \sum_{x\in\mathcal{X}} P(x) \log\left(\frac{P(x)}{Q(x)}\right) \tag{3}$$

3.3 NLP Definitions

- **Prompt (x)**: The input text given to a language model to generate a response or completion.
- **Completion (y)**: The output text generated by a language model in response to a prompt. Often the completion is denoted as $y|x$.
- **Chosen Completion (y_c)**: The completion that is selected or preferred over other alternatives, often denoted as y_{chosen}.
- **Rejected Completion (y_r)**: The disfavored completion in a pairwise setting.
- **Preference Relation ($\succ$)**: A symbol indicating that one completion is preferred over another, e.g., $y_{chosen} \succ y_{rejected}$.
- **Policy (π)**: A probability distribution over possible completions, parameterized by θ: $\pi_\theta(y|x)$.

3.4 RL Definitions

- **Reward (r)**: A scalar value indicating the desirability of an action or state, typically denoted as r.
- **Action (a)**: A decision or move made by an agent in an environment, often represented as $a \in A$, where A is the set of possible actions.
- **State (s)**: The current configuration or situation of the environment, usually denoted as $s \in S$, where S is the state space.
- **Trajectory (τ)**: A trajectory τ is a sequence of states, actions, and rewards experienced by an agent: $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, ..., s_T, a_T, r_T)$.
- **Trajectory Distribution ($(\tau|\pi)$)**: The probability of a trajectory under policy π is $P(\tau|\pi) = p(s_0) \prod_{t=0}^{T} \pi(a_t|s_t) p(s_{t+1}|s_t, a_t)$, where $p(s_0)$ is the initial state distribution and $p(s_{t+1}|s_t, a_t)$ is the transition probability.
- **Policy (π)**, also called the **policy model** in RLHF: In RL, a policy is a strategy or rule that the agent follows to decide which action to take in a given state: $\pi(a|s)$.
- **Discount Factor (γ)**: A scalar $0 \leq \gamma < 1$ that exponentially down-weights future rewards in the return, trading off immediacy versus long-term gain and guaranteeing convergence for infinite-horizon sums. Sometimes discounting is not used, which is equivalent to $\gamma = 1$.
- **Value Function (V)**: A function that estimates the expected cumulative reward from a given state: $V(s) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s]$.
- **Q-Function (Q)**: A function that estimates the expected cumulative reward from taking a specific action in a given state: $Q(s, a) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s, a_0 = a]$.
- **Advantage Function (A)**: The advantage function $A(s, a)$ quantifies the relative benefit of taking action a in state s compared to the average action. It's defined as $A(s, a) = Q(s, a) - V(s)$. Advantage functions (and value functions) can depend on a specific policy, $A^\pi(s, a)$.

- **Policy-conditioned Values ($[]^{\pi(\cdot)}$)**: Across RL derivations and implementations, a crucial component of the theory and practice is collecting data or values conditioned on a specific policy. Throughout this book we will switch between the simpler notation of value functions et al. (V, A, Q, G) and their specific policy-conditioned values (V^π, A^π, Q^π). Crucial is also in the expected value computation is sampling from data d, that is conditioned on a specific policy, d_π.
- **Expectation of Reward Optimization**: The primary goal in RL, which involves maximizing the expected cumulative reward:

$$\max_\theta \mathbb{E}_{s\sim\rho_\pi, a\sim\pi_\theta}[\sum_{t=0}^{\infty}\gamma^t r_t] \tag{4}$$

  where ρ_π is the state distribution under policy π, and γ is the discount factor.
- **Finite Horizon Reward ($J(\pi_\theta)$)**: The expected finite-horizon discounted return of the policy π_θ, parameterized by θ is defined as: $J(\pi_\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T}\gamma^t r_t\right]$ {#eq:finite_horizon_return} where $\tau \sim \pi_\theta$ denotes trajectories sampled by following policy π_θ and T is the finite horizon.
- **On-policy**: In RLHF, particularly in the debate between RL and Direct Alignment Algorithms, the discussion of **on-policy** data is common. In the RL literature, on-policy means that the data is generated *exactly* by the current form of the agent, but in the general preference-tuning literature, on-policy is expanded to mean generations from that edition of model – e.g. a instruction tuned checkpoint before running any preference fine-tuning. In this context, off-policy could be data generated by any other language model being used in post-training.

3.5 RLHF Only Definitions

- **Reference Model (π_{ref})**: This is a saved set of parameters used in RLHF where outputs of it are used to regularize the optimization.

3.6 Extended Glossary

- **Synthetic Data**: This is any training data for an AI model that is the output from another AI system. This could be anything from text generated from a open-ended prompt of a model to a model re-writing existing content.
- **Distillation**: Distillation is a general set of practices in training AI models where a model is trained on the outputs of a stronger model. This is a type of synthetic data known to make strong, smaller models. Most models make the rules around distillation clear through either the license, for open weight models, or the terms of service, for models accessible only via API. The term distillation is now overloaded with a specific technical definition from the ML literature.
- **(Teacher-student) Knowledge Distillation**: Knowledge distillation from a specific teacher to student model is a specific type of distillation above and where the term originated. It is a specific deep learning method where a neural network loss is modified to learn from the log-probabilites of the teacher model over multiple potential tokens/logits, instead of learning directly from a chosen output [50]. An example

of a modern series of models trained with Knowledge Distillation is Gemma 2 [51] or Gemma 3. For a language modeling setup, the next-token loss function can be modified as follows [52], where the student model P_θ learns from the teacher distribution P_ϕ:

$$\mathcal{L}_{\text{KD}}(\theta) = -\,\mathbb{E}_{x\sim\mathcal{D}}\left[\sum_{t=1}^{T} P_\phi(x_t \mid x_{<t}) \log P_\theta(x_t \mid x_{<t})\right]. \tag{5}$$

- **In-context Learning (ICL)**: In-context here refers to any information within the context window of the language model. Usually, this is information added to the prompt. The simplest form of in-context learning is adding examples of a similar form before the prompt. Advanced versions can learn which information to include for a specific use-case.
- **Chain of Thought (CoT)**: Chain of thought is a specific behavior of language models where they are steered towards a behavior that breaks down a problem in a step by step form. The original version of this was through the prompt "Let's think step by step" [53].