Active Learning and Human Feedback for Large Language Models

By IntuitionLabs • 8/5/2025 • 35 min read

active learning | human-in-the-loop | llm | data labeling | model alignment | rlhf | machine learning

Active Learning with Human-in-the-Loop for Large Language Models (LLMs)

Introduction

Active learning is a machine learning paradigm where the model **actively selects the most informative data points to label**, aiming to achieve high performance with minimal labeled data deepai.org deepai.org. In traditional settings, an initial model is trained on a small labeled set, then iteratively improved by querying an oracle (often a human) for labels on carefully chosen examples deepai.org. This **human-in-the-loop (HITL)** process is crucial when labeling is costly or data is abundant but unlabeled. Large Language Models (LLMs), with their massive scale and zero/few-shot capabilities, introduce new challenges and opportunities for active learning. On one hand, LLMs can often generalize from few examples and even **generate** text, potentially reducing the need for exhaustive labeled data. On the other hand, aligning LLMs with nuanced human preferences (e.g. factuality, helpfulness, safety) often *requires* extensive human feedback (such as preference comparisons or demonstrations). In this report, we delve into **core principles of active learning** and how they are adapted to LLMs, discuss HITL frameworks (like reinforcement learning from human feedback), and examine real-world use cases and implementation considerations from 2022–2025 literature.

Core Principles of Active Learning

Classical active learning provides a framework for selecting what data to label next. Below we outline its fundamental strategies:

Pool-Based Active Learning

In **pool-based** active learning, the algorithm has access to a large pool of unlabeled examples and can query labels for the most informative ones. The typical cycle is: (1) train a model on a small labeled set, (2) use it to evaluate all unlabeled data, (3) select the top candidates according to some *acquisition function*, (4) get those labeled by humans, and (5) retrain the model deepai.org. This loop repeats until a label budget is exhausted or performance converges. Pool-based sampling is the dominant setting in deep learning, as it allows the model to **choose data from anywhere in the pool** rather than being restricted to a stream. The advantage is focusing human effort on the most valuable data, often yielding significant label savings encord.com. However, it assumes a representative unlabeled pool is available and that we can effectively identify which samples will most benefit the model encord.com.

Uncertainty Sampling

Uncertainty sampling is one of the most common query strategies sagacify.com. The model queries those instances about which it is least confident deepai.org. The intuition is that labeling an already well-understood example is less useful than labeling one the model is currently unsure about. In practice, uncertainty can be quantified in various ways. For a classifier, popular metrics include: (a) *lowest predicted probability* (least confident) for the top class, (b) *smallest margin* between the most and second-most likely class, or © *highest entropy* of the predicted class distribution lilianweng.github.io lilianweng.github.io. For instance, if an LLM-based classifier assigns class A 51% probability and B 49%, it has a small margin and high uncertainty, making that sample a good candidate for labeling. Uncertainty sampling tends to reduce the model's confusion rapidly by targeting ambiguous cases. That said, modern deep networks can be **overconfident** in their predictions lilianweng.github.io, so raw probability scores might not perfectly reflect true uncertainty. Techniques like **calibration** or using an ensemble of models (Query-by-Committee) can help address this gap lilianweng.github.io lilianweng.github.io.

Query-by-Committee

Query-by-Committee (QBC) leverages multiple models (a "committee") to decide what to label deepai.org. Each model in the committee is trained (often on the current labeled set) and then all models are run on the unlabeled pool. The idea is to select examples where the committee members **most disagree** in their predictions deepai.org. If one model thinks an unlabeled text is positive sentiment while another thinks it's negative, that conflict indicates the sample is informative. Common ways to measure disagreement include *vote entropy* (how evenly the committee votes are split among classes) and Kullback–Leibler divergence among the models' predicted probability distributions lilianweng.github.io lilianweng.github.io. QBC can be more robust than single-model uncertainty because it captures **model uncertainty due to different decision boundaries**. In practice, committees might be formed by training multiple models on different random initializations or subsets of data, or by using one model with different dropout masks to simulate an ensemble (as in Monte Carlo dropout) lilianweng.github.io lilianweng.github.io. The challenge with QBC is the computational cost of maintaining several large models, especially for LLMs. However, approximate ensembles (dropout, snapshot ensembles, etc.) have been used to make this feasible lilianweng.github.io lilianweng.github.io.

Expected Model Change

Instead of looking only at model predictions, **expected model change** strategies choose samples that would maximally change the model's parameters or outputs if the true label were known deepai.org. In other words, it estimates how much *learning progress* labeling a given example would yield. One formulation is to compute the gradient that the model would take on an unlabeled sample for each possible label, and quantify the norm of that gradient (expected change in weights). A high gradient norm or loss reduction indicates the sample is likely to

significantly update the model ai-terms-glossary.com. By labeling such a sample, we expect a large **model improvement**. Expected model change can be computationally intensive (it may require simulating model updates for each candidate). Nonetheless, it aligns closely with the active learning goal: prioritize data that most reduces error. In deep learning, researchers have explored proxies for this, like "loss prediction" modules that predict how much the loss would drop if a sample were labeled lilianweng.github.io. The approach explicitly tries to **maximize training impact** per label lilianweng.github.io. While powerful in concept, it requires careful implementation; approximating model change reliably for LLMs remains nontrivial due to their high complexity.

Diversity (Diversity/Density) Sampling

Another core principle is **diversity sampling**, which seeks to label a set of examples that are as informative *collectively* as they are individually lilianweng.github.io. The goal is to avoid redundancy and cover the data distribution broadly. A model could be very uncertain about 10 very similar sentences – labeling all 10 might only teach it the same information repeatedly. Instead, diversity-based methods would pick perhaps one or two from that cluster and then other examples from different regions of the input space teaching.ieee.org openreview.net. Common techniques include clustering the unlabeled data (and picking representatives from each cluster), or selecting points that **maximize the coverage** of the pool's feature space. These are sometimes called *density-weighted* methods, since they ensure high-density regions (common patterns in data) are represented while also including outliers for breadth lexunit.ai. As Weng (2022) notes, "selected samples should be representative of the underlying distribution" and many approaches quantify similarity to achieve this lilianweng.github.io. In practice, diversity is often combined with uncertainty into hybrid strategies – e.g. choose the most uncertain samples that are also dissimilar to those already selected lilianweng.github.io. This guards against the model querying a bunch of nearly identical, uncertain examples. Diversity sampling is especially important for LLMs, since their **input space is high-dimensional and rich**; ensuring a diverse training set can mitigate biases and improve generalization across topics, styles, or domains.

Human-in-the-Loop Frameworks for LLMs

Scaling active learning to LLMs requires carefully designed human-in-the-loop frameworks. LLMs are used in complex tasks (conversation, summarization, coding assistance, etc.), so human feedback can take many forms beyond simple class labels. Here we discuss how HITL pipelines are structured for LLM training and alignment, including **task decomposition, feedback collection, annotation processes,** and the prominent approach of **Reinforcement Learning from Human Feedback (RLHF)**.

Task Decomposition and Annotation Pipelines

Large language model tasks can often be *decomposed* to make human feedback more effective. **Task decomposition** means breaking a complex task into smaller parts that either a human or a model can handle more easily. For instance, instead of asking a human annotator to directly label a long, complex prompt-response pair as “good” or “bad”, the process might be decomposed: the human might first check factual accuracy, then appropriateness, then completeness, etc., before giving an overall judgment. Decomposition is also used in prompt design: an LLM might be guided to tackle a multi-step problem step-by-step (and a human overseer can verify each step). By inserting checkpoints where humans verify intermediate results, errors can be caught earlier and feedback becomes more targeted. For example, NineTwoThree Studio (2024) describes breaking down an insurance claim processing task for an LLM into subtasks with human review at each stage, which boosted accuracy into the high 90% after iterative human corrections.

Human feedback collection for LLMs often goes beyond binary labels. Annotators might rank multiple model outputs, provide edited corrections, or supply reference answers. To manage this, organizations develop detailed **annotation pipelines**. These pipelines typically include a user interface for labelers to interact with model outputs (for example, a web app showing two responses and asking which is better, or a form to edit the model’s text). There are notable examples of such pipelines: OpenAI’s InstructGPT project built a system for contractors to **compare pairs of model-generated responses** and select the preferred one, as well as to write high-quality answers for prompts (for a supervised fine-tuning stage). Anthropic’s HH-RLHF (Harmlessness & Helpfulness) work involved crowdworkers providing feedback on model outputs and even *writing explanations* for their preferences to help the model learn. A good pipeline enforces **annotation consistency** by providing guidelines and training for humans – this is crucial, since different annotators may otherwise have varying preferences or interpretations. Automation can assist here too: researchers explore having LLMs **assist annotators** by pre-labeling data or suggesting edits, with humans only correcting when the LLM is likely wrong. Such semi-automated pipelines can increase throughput and reduce cost, but they require careful quality control to avoid reinforcing model biases. Overall, task decomposition and robust pipelines ensure that human expertise is used efficiently, focusing on aspects that LLMs can’t (yet) handle on their own (fact-checking, nuanced judgments, etc.).

Reinforcement Learning from Human Feedback (RLHF)

RLHF has emerged as a cornerstone framework for aligning LLMs with human preferences. In an RLHF pipeline, humans are in the loop to supply a *reward signal* that guides the model’s learning, rather than directly providing target outputs. A prototypical RLHF process (as used for training models like OpenAI’s ChatGPT and InstructGPT) consists of multiple stages:

- **Supervised Fine-Tuning (SFT)**: First, a subset of human demonstrators may be asked to provide ideal responses to various prompts. The base LLM is fine-tuned on this high-quality **demonstration data** to create a model that is already better aligned to user instructions huyenchip.com. SFT can be seen as a warm-up stage before RLHF proper, and not all pipelines include it, but it often improves subsequent training stability. For instance, OpenAI reported that ChatGPT's precursor was fine-tuned on human-written answers to make it follow instructions better huyenchip.com.

- **Preference Data Collection**: Humans are presented with outputs generated by the current model (or several models) and asked to **rank or choose the best output**. Typically, for a given prompt, two or more responses are sampled, and the human oracle selects which response they prefer (based on criteria like helpfulness, correctness, harmlessness). This yields a dataset of pairwise comparisons. Each comparison is a subtle form of annotation – it says "output A is better than output B for prompt X." These comparisons are especially useful because they collapse many aspects of quality into an **implicit scalar feedback** (a preference). Compared to training on direct numeric scores, pairwise preferences tend to be more consistent between annotators.

- **Reward Model (RM) Training**: The collected preference data is then used to train a **reward model** – typically a neural network that takes in (prompt, candidate output) and predicts a score that should align with human preferences. In practice, the reward model is often a smaller or duplicate version of the LLM, fine-tuned to predict which output in a pair was human-preferred. The training objective assumes a Bradley-Terry or logistic model form, where the probability of preferring output A over B is calibrated to the difference in their reward scores arxiv.org. Once trained, this reward model serves as a **proxy for human judgment**.

- **RL Fine-Tuning**: Finally, the original LLM is fine-tuned using reinforcement learning (commonly Proximal Policy Optimization, PPO) to maximize the reward model's score. Essentially, the LLM acts as a policy generating text, and the reward model's output is treated as the reward signal to optimize. Over many episodes, the LLM learns to produce outputs that the reward model (and thus, presumably humans) favor. This stage is where the term "RLHF" is most literal: the human feedback (via the RM) is used as the reinforcement reward to **polish the LLM's behavior**. Notably, this process can be unstable – tuning a massive policy with a learned reward can lead to issues like reward hacking or divergence if not carefully regularized arxiv.org. Techniques like PPO with KL-control (to keep the new policy close to the original model) are used to stabilize training.

Figure 1: A typical human-in-the-loop training pipeline for aligning an LLM, consisting of supervised fine-tuning on human-written demonstrations (to get an initial aligned model), followed by iterative preference collection and reinforcement learning from human feedback to further refine the model. In essence, a pretrained model ("untamed monster") is made more helpful and safe through fine-tuning on quality data, then "given a smiley face" via RLHF using human preference comparisons huyenchip.com.

RLHF has been remarkably effective. It was a key to OpenAI's **ChatGPT** success in 2022, enabling the model to follow instructions and adhere to human-desired behavior far more reliably than a pretrained model. Anthropic's Claude model similarly uses RLHF (and related techniques like Constitutional AI) to align with human values. By 2023, RLHF (or its variations) became a *de facto* strategy for industry LLM deployments huyenchip.com. However, RLHF is resource-intensive: it requires training additional models (the reward model, possibly value

models) and a supply of skilled human annotators. Research has been ongoing to simplify this pipeline. For example, **Direct Preference Optimization (DPO)** is an alternative that skips the RL step by incorporating the preference signal directly into a modified loss for the language model arxiv.org. DPO has shown promise in stability, though it still requires the same comparison data. Another line of work uses **AI feedback (RLAIF)** where another strong LLM simulates the role of the human in providing preference labels. AI feedback can be cheaper and faster, but often lacks the nuanced judgment of humans and may propagate model biases. As a result, hybrids are emerging: for instance, **RL from targeted human feedback (RLTHF)** uses an LLM or heuristic to label easy cases and reserves human effort for the most *hard-to-judge* cases. This approach achieved the same alignment quality as full human labeling with only ~6% of the data being human-annotated – a huge efficiency gain. Such innovations underscore a theme: the future of HITL for LLMs will likely combine human expertise with clever model-based assistance to maximize the value of each human decision.

Adapting Active Learning Strategies to LLMs

Applying active learning in the context of large language models requires rethinking traditional strategies to accommodate LLMs' unique properties: extremely high-dimensional representations, the ability to perform tasks with little data (zero/few-shot learning), and the fact that LLM training or fine-tuning can be very expensive. Researchers from 2022–2025 have explored various adaptations and novel techniques to make active learning effective for LLMs:

- **Uncertainty Estimation for LLMs**: Unlike a simple classifier that outputs a well-defined probability for a single label, an LLM often produces *free-form text*. So what does *uncertainty* mean for an LLM? One way is to focus on *subtasks* or model subcomponents. For example, if using an LLM for classification via prompting, one can extract a probability distribution over labels (from either the model's token probabilities or a learned classifier head) and then apply classical uncertainty metrics lilianweng.github.io. Indeed, many LLM applications still reduce to classification or scoring tasks (e.g. is this output acceptable or not). In generative settings, uncertainty can be gauged by the **entropy of the next-token predictions** or by generating multiple outputs and seeing how much they vary. The work *Active Preference Learning (APL)* by Muldrew et al. (2024) uses the LLM's **predictive entropy** over its continuation as part of an acquisition function for RLHF data. In their setup, the model would prefer to ask for human comparison labels on prompts where its current policy has high entropy in the predicted reward – essentially where it's unsure which completion is better. Another example is using an *ensemble of LLMs or prompts* as a committee: e.g., prompt the model in different ways or use multiple model sizes (GPT-3, GPT-4, etc.) and identify queries for which they disagree on the response. This could flag ambiguous instructions that need human clarification. However, fully training multiple large models is usually impractical – instead, techniques like **Monte Carlo dropout** or multi-head ensembles within one model have been used to approximate committees for LLMs lilianweng.github.io lilianweng.github.io.

- **Leveraging Zero/Few-Shot Capabilities**: A striking property of modern LLMs is that they often perform tasks *reasonably well with zero or few examples*, thanks to their vast pretraining. This somewhat reduces the classic need for many labeled examples, but it **doesn't eliminate the value of active learning**. Instead, the role shifts: active learning for LLMs often targets *edge cases or new domains* where even a strong pretrained model might struggle. Few-shot learning itself can be seen as an "active" process at inference time – the user provides a few exemplars in the prompt. Some researchers have combined this idea with active learning: e.g., selecting which exemplars to include in a prompt is analogous to selecting which data to label. *Active-Prompt* (Diao et al. 2023) and *AL-ICL* methods explore choosing demonstration examples that maximize model performance. These effectively do a form of active learning *within* the prompt (no parameter update needed). Additionally, the **cold start problem** in active learning (the initial model has no basis to estimate uncertainty) is mitigated by LLMs – a powerful pretrained model has a decent prior, so even the first selection can be informed. Bayer et al. (2024) introduced **ActiveLLM**, which flips the script by using an LLM (like GPT-4) to guide active learning for a smaller model. GPT-4 can evaluate unlabeled candidates and predict which would be most useful to label, overcoming cold start by injecting its prior knowledge. In experiments, this significantly improved few-shot fine-tuning of a BERT classifier, outperforming traditional uncertainty sampling and even strong few-shot methods. This highlights a new paradigm: using an LLM "oracle" to drive the selection of data – effectively **LLM-guided active learning**.

- **Instruction Tuning and Human Feedback**: Many LLM adaptations involve instruction tuning – fine-tuning the model on datasets of task instructions and ideal responses (often pulled from human demonstrations). Active learning can play a role here by determining *which instructions* would most benefit from human-provided answers. Rather than randomly labeling a bunch of tasks, one could use the current model to identify instructions it fails at or is uncertain about, and have humans provide high-quality examples for those. This approach was used implicitly in some alignment projects – for instance, Anthropic's red-teaming of language models involved using one model to generate tricky questions or adversarial prompts, which humans then answered or flagged, creating new training data arxiv.org. In safety-focused instruction tuning, researchers often enumerate potential problematic prompts (e.g. "How do I make a weapon?") and ensure those are addressed via human feedback. Active learning helps find such *blind spots*. An example is **adaptive data collection for reward models**: Shen et al. (2025) proposed an *Active Reward Modeling* approach where they selected comparison queries that balanced exploring new model behaviors and refining known borderline cases. They used metrics derived from Fisher Information to pick which pairs of outputs would be most informative to have humans compare. This way, during RLHF they focus labelers on comparisons that teach the model the most, rather than redundant ones. The result was a more label-efficient alignment process that maintained stability and performance.

- **High-Dimensional Representations and Diversity**: LLMs operate in an extremely rich feature space (billions of parameters, embedding spaces of thousands of dimensions). Ensuring diversity in selected data is thus both more challenging and more crucial. Simple clustering or distance metrics might be less effective in these complex spaces. Recent work suggests using the LLM's own embeddings or hidden states to measure diversity. For example, if fine-tuning GPT-3 on a new domain, one could embed all candidate texts using GPT-3's pretrained encoder and then run a clustering or *core-set selection* algorithm in that embedding space. This was the idea behind some *core-set active learning* approaches adapted to deep models (Sener & Savarese 2018) and remains relevant to LLMs. Moreover, **diversity of generated outputs** is an aspect: if an LLM tends to give repetitive answers, humans might proactively intervene to supply more varied rephrases or alternative styles, effectively enlarging the output distribution. Ensuring a diverse training set for LLMs also ties into **bias and fairness** – active learning should avoid only selecting data that reinforces the model's existing bias. Recent surveys note that active learning must be carefully managed to not over-focus on one subpopulation and miss others. In practice, techniques like stratified sampling (ensuring each category or group is represented in queries) or adding *fairness constraints* to the acquisition function are being explored to address this.

- **Cost-Awareness and Stopping Criteria**: With LLMs, the cost per annotation can be very high – labeling a single data point might involve writing a long answer or having multiple annotators reach consensus. So active learning strategies have evolved to be **cost-aware**. This means they might factor in not just how much a sample would help the model, but also how long it takes a human to label it. For instance, a very complex prompt might be informative but take an annotator 15 minutes to answer, whereas another question could be almost as informative and take 1 minute – a cost-aware strategy might favor the latter to optimize efficiency. Additionally, the stopping criterion for active learning in LLMs may include plateauing of a specific metric or exhaustion of budget, but also *diminishing returns in model improvement per cost*. Modern studies emphasize tracking the "return on investment" of human labels in real time arxiv.org arxiv.org. When fine-tuning a huge model, each training step is expensive, so one cannot afford an excessively long iterative loop. Batch-mode active learning (querying labels in batches rather than one by one) is almost always used with LLMs to amortize training costs. Researchers also consider *one-shot* active learning approaches for LLMs: doing a single round of selection and fine-tuning if iterative retraining is too costly, even though it sacrifices some of the adaptivity of the classic loop.

In summary, active learning for LLMs is a rapidly evolving area. Techniques like **LLM-based annotators** (LLM labels some data to save human effort) and **hybrid human-AI annotation** are particularly promising. As noted in a 2023 survey, the majority of deep active learning methods were still designed for task-specific models, not foundation models. But customizing active learning for LLMs is necessary to maximize their performance and alignment with minimal human labels. We are seeing the first wave of such custom methods now – from *ActiveLLM* that uses GPT-4 to guide selection, to *RLTHF* that smartly allocates human effort, to *coactive learning* that learns from implicit user edits instead of explicit labels cs.cornell.edu openreview.net. These approaches all adapt the core idea of querying the right data to the scale and abilities of LLMs.

Applications and Use Cases

Active learning with HITL has been applied in various contexts to improve LLMs efficiently. We highlight a few key use cases from recent years:

- **Safety Alignment and Moderation**: One critical application is aligning LLM behavior with safety norms – avoiding toxic, biased, or harmful outputs. Given the vast search space of prompts that could trigger bad behavior, active learning is used to *discover and label problematic cases*. For example, **red teaming** an LLM involves generating adversarial prompts (often using another AI or heuristic) to make the model produce undesired outputs, then having humans label or fix those outputs arxiv.org. Redwood Research's 2022 work on *adversarial training for high-stakes reliability* followed this loop: an adversary model generated scenarios where a language model might mention violent content, humans labeled whether the output was acceptable, and the model was fine-tuned on these targeted examples aisafety.info news.apartresearch.com. This is effectively active learning where the *model itself proposes queries* (via adversarial generation) and humans provide judgments. The result was a model much less likely to produce violent completions, achieved with far fewer training examples than if one tried to foresee and label all such cases upfront. Similarly, OpenAI's content moderation systems use an active learning style approach: they continually update the moderation model by labeling new kinds of undesirable outputs that the LLM produces in deployment, thereby **adapting the safety filter over time**. This ensures that as the LLM evolves or users find new loopholes, the human feedback loop closes those gaps efficiently. In summary, active learning is a powerful tool in *safety alignment*, focusing human oversight on the model's worst failures and iteratively making the model safer.

- **Data Efficiency and Fine-Tuning**: Active learning shines when data is scarce or labeling is expensive – a scenario common in domain adaptation of LLMs. Consider fine-tuning a general LLM to work in a specific domain (legal, medical) or task (classifying customer inquiries). Instead of labeling a huge random sample of domain data, practitioners can use active learning to identify a *small, valuable subset* for annotation. For example, an LLM might be used to bootstrap a classifier for legal case documents: it could preprocess and cluster thousands of documents, and then humans label a few from each cluster (diversity sampling) and a few the model is unsure about (uncertainty sampling). This approach was seen in a 2023 industry case where an LLM-based system helped select only 1,000 samples to label out of 100,000, yet achieved nearly the same accuracy on a legal document classification as labeling everything – resulting in substantial cost savings. In the research realm, *ActiveLLM* (2024) demonstrated that one can improve a smaller model's few-shot learning by querying labels for just a handful of critical examples chosen by a larger LLM. Another example is **Beyond-Labels** (Yao et al. 2023), which asked annotators not only for labels but also for a short rationale during active learning. Those rationales helped a QA model learn with fewer question-answer pairs by providing extra signal per example. Across the board, these use cases show active learning making fine-tuning and adaptation of LLMs *more data-efficient*, often reaching desired performance with only a fraction of the data labeled. This is extremely valuable for organizations that cannot afford to label millions of examples – e.g. a biotech company fine-tuning an LLM on lab reports might label only the 5% most informative entries and get excellent results.

- **Domain Adaptation and Personalization**: LLMs often need to be specialized to particular domains or even personalized to individual users. Active learning can guide what additional data to gather or annotate for these purposes. For instance, to adapt a general LLM to clinical text, one could actively select medical records or dialogues that the base model finds confusing, and have medical experts annotate them (for truthfulness, or to provide the correct conclusion). This targeted approach was investigated by Shu et al. (2023) who applied active learning for **clinical domain adaptation**, showing that selecting notes with uncommon vocabulary or ambiguous phrasing for doctor review led to a better medical language model with far fewer annotations than random selection. Another emerging area is personalization, where an AI assistant learns a user's preferences. Here, the system might actively ask the user for feedback in cases where it's unsure of the user's preference. This is a form of active learning where the user themselves is the oracle. For example, a personalized email-writing assistant might occasionally ask, "Do you prefer a more formal tone in situations like this?" – by querying the user actively, it learns faster about that user's style. While this goes slightly beyond traditional pool-based AL, it's conceptually similar: identify uncertainty (the model is unsure how the user would want something) and query the human for clarification. Research on **coactive learning** for LLMs explores this idea of learning from implicit user feedback like edits, which is essentially the user correcting the model's output and the model updating itself from that correction cs.cornell.edu openreview.net. Early results show this can personalize or adapt an LLM with very few explicit labels by using the stream of natural interactions as feedback.

Overall, these examples underscore that active learning with humans in the loop is a versatile strategy in the LLM era. Whether the goal is **safer AI**, **cheaper fine-tuning**, or **customized models**, intelligently choosing what to label – and leveraging both human expertise and model power – leads to big wins in efficiency and performance. Many of these approaches were conceptual or in early deployment around 2023–2025; we expect broader adoption as tool support and understanding of LLM-specific active learning mature.

Implementation Considerations

Designing and deploying an active learning pipeline for LLMs requires careful attention to practical considerations. Some key factors include:

- **Labeling Cost and Human Effort:** LLM-related annotations can be complex and time-consuming. Writing a high-quality answer or comparing subtle differences between two long texts is harder than labeling a single image or short sentence. This amplifies the need to **prioritize queries wisely**. Studies have emphasized selecting examples that give the "biggest bang for the buck" in terms of model improvement per human-minute. Budgeting is crucial: define a label budget up front and use cost-aware acquisition functions if possible. It's also important to consider annotator expertise – for specialized domains like medicine or law, your oracle may be a highly paid expert. In such cases, active learning's promise of reducing data needs by (say) 50% translates directly into huge cost savings.

- **Annotation Quality and Consistency:** With multiple humans in the loop, **inter-annotator variability** is a challenge. Disagreements introduce label noise that can confuse training. To combat this, clear guidelines and training are a must. Many LLM projects have annotation guides detailing exactly how to rate outputs, with examples. Some pipelines use consensus labeling (multiple annotators per example, then aggregate) to improve reliability, though this increases cost. Another tactic is **calibration of human raters**: periodically check agreement on a gold set or have discussions to align on borderline cases. Research has also looked at methods to detect and correct label noise post-hoc; for example, by having the model identify when a human-provided label seems inconsistent with its prior knowledge and flagging it for review. In LLM alignment, bias in human feedback is an emerging concern – if your annotators share a similar background, their preferences might tilt the model in a particular direction. Ensuring diversity among human annotators or explicitly instructing them to consider alternative perspectives can help reduce inadvertent bias reinforcement.

- **Iterative Loop and Model Retraining:** A classical active learning loop retrains the model from scratch (or fine-tunes further) after each batch of new labels deepai.org. With LLMs, retraining even a few times can be extremely computationally expensive. This leads to practical modifications: one might use **larger batch sizes** for querying (to do fewer retraining cycles), or use a smaller proxy model to decide what to label (as a stand-in for the big model, to save computation). For example, one could train a smaller BERT on a classification proxy task and use it to select data for a larger GPT-based model. Infrastructure-wise, it's critical to have an automated pipeline: data flows from the model's predictions to a labeling interface, then back into the training process. MLOps tooling for this is still nascent. Some teams build custom pipelines with task queues (for unlabeled samples) and databases to track labeled data and model versions. There are also emerging tools (e.g. **Label Studio** or enterprise platforms) that integrate with model inference APIs to support active learning loops. Whatever the setup, **monitoring** is vital. You should track not just the model's performance on a validation set each iteration, but metrics like annotator agreement, labeling throughput, and how often the same data is requested (to avoid duplicates). Deciding a stopping point can be tricky; common rules include when the model's validation accuracy plateaus or when the cost reaches the budget. Some use adaptive stopping: if the last N iterations gave minimal improvement, stop early to save effort deepai.org.

- **Tooling and Integration:** Incorporating human feedback for LLMs often means integrating with annotation tools or platforms (for example, a web app for contractors to rank model outputs). These tools should be designed to handle the length and richness of LLM outputs – showing multi-paragraph texts, highlighting differences between outputs for comparison, etc. Some organizations have built internal tooling for RLHF data collection, but there are also open-source frameworks coming up. OpenAI's **training GUI** (used in earlier preference studies) and Google's **ViewFinder** (for evaluating multiple model responses side by side) are examples. There is also interest in making use of **user feedback at scale**: if you deploy an LLM system, you could have a feedback button for end-users ("Was this response helpful?"). Aggregating such feedback (which is noisy and not as curated as labelling) and feeding it into model updates is a form of active learning on live data. This bridges into **continuous learning**: the system constantly retrains on new feedback. While promising, this raises safety concerns (you need safeguards so the model doesn't drift in undesirable ways based on a few users' inputs). Infrastructure that supports *shadow deployments* (testing a retrained model on the side before full release) and *version control for models and data* becomes important in such a scenario. A 2023 review highlighted the necessity of developing efficient **ML Ops systems** to coordinate data selection, annotation, and model updating for foundation models. In effect, as LLMs are deployed, active learning blurs into **model maintenance** – continually curating data and feedback to keep the model optimal and aligned. Having the right tools and processes in place will be key for organizations to manage this ongoing human-in-the-loop refinement cycle.

Conclusion

Active learning with humans in the loop offers a powerful approach to training and aligning large language models efficiently. By intelligently selecting which data points get human attention, we can curb the otherwise prohibitive demands of labeling at LLM scale. We reviewed core active learning strategies – uncertainty sampling, query-by-committee, expected model change, diversity sampling – and saw that while these principles still apply, they need adaptation for the LLM era. Techniques like using LLMs themselves to guide selection, focusing on preference feedback (as in RLHF), and balancing uncertainty with diversity are enabling orders-of-magnitude reductions in required human labels. Human-in-the-loop frameworks such as RLHF exemplify how carefully curated human feedback can steer giant models toward desired behaviors that pure self-supervised training cannot achieve arxiv.org. At the same time, the role of the human is evolving: from merely providing ground-truth labels to collaborating with AI (providing preferences, corrections, or high-level guidance). The research and examples from 2022–2025 – from safer AI training to domain-specific fine-tuning – demonstrate that incorporating human insight remains essential for LLMs to truly serve human needs. Implementing these systems in practice comes with challenges in cost, quality control, and engineering complexity, but solutions are emerging as the community gains experience.

Looking ahead, we can expect more seamless integration of active learning in LLM development. For instance, future LLMs might have built-in uncertainty estimators or explanations for when to ask for help, making the human-proactive querying more natural. We may also see **meta-active-learning**, where models learn *how to actively learn* (optimizing their own query strategies via

reinforcement learning or meta-learning). Foundation model research suggests that new strategies, perhaps learned automatically, will be needed to fully capitalize on active learning for such complex models. Nonetheless, the fundamental promise remains: by keeping humans in the loop in a principled way, we can train AI systems that are not only more efficient with data, but also more aligned with the subtle and evolving objectives that we care about. Active learning thus acts as a critical bridge between raw AI capability and human expertise, ensuring that even as models grow ever more powerful, they do so under thoughtful human guidance.

Sources: The information and examples in this report are drawn from recent literature and surveys on active learning and LLM alignment, including academic papers (e.g. on active preference learning, reward modeling, and coactive learning cs.cornell.edu), industry case studies (e.g. NineTwoThree Studio's HITL strategies), and comprehensive surveys on deep active learning in the era of foundation models. These sources, listed below, provide further technical details and results supporting the points discussed.

IntuitionLabs - Industry Leadership & Services

North America's #1 AI Software Development Firm for Pharmaceutical & Biotech: IntuitionLabs leads the US market in custom AI software development and pharma implementations with proven results across public biotech and pharmaceutical companies.

Elite Client Portfolio: Trusted by NASDAQ-listed pharmaceutical companies including Scilex Holding Company (SCLX) and leading CROs across North America.

Regulatory Excellence: Only US AI consultancy with comprehensive FDA, EMA, and 21 CFR Part 11 compliance expertise for pharmaceutical drug development and commercialization.

Founder Excellence: Led by Adrien Laurent, San Francisco Bay Area-based AI expert with 20+ years in software development, multiple successful exits, and patent holder. Recognized as one of the top AI experts in the USA.

Custom AI Software Development: Build tailored pharmaceutical AI applications, custom CRMs, chatbots, and ERP systems with advanced analytics and regulatory compliance capabilities.

Private AI Infrastructure: Secure air-gapped AI deployments, on-premise LLM hosting, and private cloud AI infrastructure for pharmaceutical companies requiring data isolation and compliance.

Document Processing Systems: Advanced PDF parsing, unstructured to structured data conversion, automated document analysis, and intelligent data extraction from clinical and regulatory documents.

Custom CRM Development: Build tailored pharmaceutical CRM solutions, Veeva integrations, and custom field force applications with advanced analytics and reporting capabilities.

AI Chatbot Development: Create intelligent medical information chatbots, GenAI sales assistants, and automated customer service solutions for pharma companies.

Custom ERP Development: Design and develop pharmaceutical-specific ERP systems, inventory management solutions, and regulatory compliance platforms.

Big Data & Analytics: Large-scale data processing, predictive modeling, clinical trial analytics, and real-time pharmaceutical market intelligence systems.

Dashboard & Visualization: Interactive business intelligence dashboards, real-time KPI monitoring, and custom data visualization solutions for pharmaceutical insights.

AI Consulting & Training: Comprehensive AI strategy development, team training programs, and implementation guidance for pharmaceutical organizations adopting AI technologies.

Contact founder Adrien Laurent and team at https://intuitionlabs.ai/contact for a consultation.

DISCLAIMER

The information contained in this document is provided for educational and informational purposes only. We make no representations or warranties of any kind, express or implied, about the completeness, accuracy, reliability, suitability, or availability of the information contained herein.

Any reliance you place on such information is strictly at your own risk. In no event will IntuitionLabs.ai or its representatives be liable for any loss or damage including without limitation, indirect or consequential loss or damage, or any loss or damage whatsoever arising from the use of information presented in this document.

This document may contain content generated with the assistance of artificial intelligence technologies. AI-generated content may contain errors, omissions, or inaccuracies. Readers are advised to independently verify any critical information before acting upon it.

All product names, logos, brands, trademarks, and registered trademarks mentioned in this document are the property of their respective owners. All company, product, and service names used in this document are for identification purposes only. Use of these names, logos, trademarks, and brands does not imply endorsement by the respective trademark holders.

IntuitionLabs.ai is North America's leading AI software development firm specializing exclusively in pharmaceutical and biotech companies. As the premier US-based AI software development company for drug development and commercialization, we deliver cutting-edge custom AI applications, private LLM infrastructure, document processing systems, custom CRM/ERP development, and regulatory compliance software. Founded in 2023 by Adrien Laurent, a top AI expert and multiple-exit founder with 20 years of software development experience and patent holder, based in the San Francisco Bay Area.

This document does not constitute professional or legal advice. For specific guidance related to your business needs, please consult with appropriate qualified professionals.