DEFINING AND BENCHMARKING ACTIVE DATA LABELING

Anonymous authors
Paper under double-blind review

ABSTRACT

The availability of LLMs and VLMs enables the study of a new kind of machine learning setup, which we call Active Data Labeling (ADL): given a fixed set of data points, a novel classification task, and an oracle providing labels to this task, the goal is to annotate the entire set with oracle quality, but using as few calls to the oracle as possible. ADL is related to Active Learning, but also substantially differs from it: changing the goal from creating a model to annotating a fixed data set gives rise to a new set of affordances, methods and tradeoffs that could be explored (e.g., deferring certain classes of hard samples to the oracle while focusing on the easier, manageable ones). The ADL setup arises naturally with LLMs: a user asks to classify a set of documents or records by applying an LLM prompt to each one, and our goal is to provide a high quality annotation while minimizing actual LLM calls. We formalize the ADL learning setup, propose standardized evaluation metrics, and provide TeLa-Bench, a benchmark of cases on which it could be studied. We also present baseline algorithms that establish a lower bound on what can be achieved in the black-box ADL setup: a reduction of 43% in LLM calls while maintaining 98% agreement with the oracle.

1 INTRODUCTION

Consider an analyst faced with thousands of rows of textual data, to which they want to assign a label (e.g., categorizing user feedback into company-specific themes). To do this, a common solution these days is to design an LLM prompt and apply it to all the examples. While this is a highly common use case, its straightforward application is expensive: thousands of LLM calls cost both time and money. Can we reduce the number of LLM calls by replacing them with online trained classifiers, or some other labeling function? Note that in this case we are not interested in producing a general purpose classifier, but rather in classifying a closed set of items according to an ad-hoc criteria, once. We formalize this scenario in a new machine learning setup which we call ADL: Active Data Labeling.

Formally, the ADL setup assumes a fixed, finite set of unlabeled data points, a novel classification task, and access to a computationally expensive oracle. The objective is to annotate the entire dataset such that the final labels exhibit oracle-level quality, but using as few actual calls to the oracle as possible. Today, Large Language Models (LLMs) and Vision-Language Models (VLMs) act as highly capable, instantaneous algorithmic oracles. From the practical angle, the ADL setup provides the opportunity of immense cost-reductions. More interestingly for us, it also unlocks a rich, unexplored algorithmic research space: How can a system dynamically calibrate its confidence without static validation sets? How do we build mechanisms that systematically toggle between an algorithmic oracle and models during continuous inference, where obtained oracle labels are also used for validating, calibrating and improving the prediction models?

Superficially, ADL resembles Active Learning (AL) setup. However, while AL shares the goal of reducing oracle queries, its objective is fundamentally different: the goal of AL is producing a standalone model capable of generalizing to future, unseen data. In AL, the oracle acts as a temporary teacher during a distinct training phase and is eventually discarded. In contrast, ADL focuses exclusively on assigning labels to the current dataset. Any intermediate models generated during the process are disposable components of the pipeline. An ADL system can collapse the training and inference phases, treating the oracle as a permanent fallback. This allows an ADL

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

system affordances which are not available for AL: for example, they can partition the data and devise/learn a separate predictor for each partition, or completely abstain from learning difficult, rare, or out-of-distribution data types by routing them directly to the oracle and not attempting to generalize from them. Consequently, while traditional AL algorithms can be repurposed for ADL, they are inherently sub-optimal because they optimize for future generalization rather than instance-specific decisions. We believe there is rich theory and practice to be discovered by studying the ADL setup.

To formally ground this setup, we define the ADL problem space and introduce standardized evaluation metrics that quantify the trade-off between computational efficiency and annotation quality. Furthermore, to facilitate the empirical study of this paradigm, we provide a concrete benchmark for the multi-class classification ADL setting: TeLa-Bench, sourced from 30 diverse public datasets (Hugging Face and Kaggle) and encompassing over a million text classification instances, TeLa-Bench provides pre-computed oracle annotations across varying domains, text lengths, and class counts.

Using this benchmark, we explore simple algorithmic solutions by implementing four baseline systems. These baselines establish a lower bound on what can be achieved in the challenging black-box ADL setup, demonstrating that even foundational approaches can reduce LLM calls by up to 43% while maintaining a strict 98% agreement with the oracle, indicating significant potential for future research.

2 Related Work

Cost-Reduction and Hybrid Annotation Frameworks Recent works have addressed the high costs of LLM deployment by proposing various hybrid routing and distillation strategies. For instance, *FrugalGPT* (Chen et al., 2023) introduces an inference-time cascading framework that routes queries through sequentially stronger, static LLMs based on learned confidence thresholds. While sharing the goal of cost reduction, cascading is designed for standalone, independent queries rather than dynamically training a model to exploit the semantic redundancy of a specific dataset.

Moving towards dynamic systems, *Online Cascade Learning* (Nie et al., 2024) proposes a streaming architecture that routes instances between a lightweight local model and an LLM, continuously updating the local model online based on the LLM's feedback. This setup closely mirrors the configuration of our Active Data Labeling (ADL) formulation in the streaming variant setup. However, it is presented as a specific algorithmic solution, whereas our work formalizes the overarching ADL paradigm into a standardized, large-scale benchmark to evaluate any algorithmic approach.

Finally, works like *LLM on a Budget* (Luccioli et al., 2025) combine active learning with knowledge distillation to selectively query an LLM and train a small, local classifier for large text corpora. While highly relevant, this approach fundamentally differs from ADL in its end objective: it aims to produce a robust, standalone model for future, independent deployment. In contrast, ADL focuses strictly on the immediate annotation of the current dataset, treating the model as a disposable component and maintaining a permanent fallback to the oracle for complex edge cases.

Connections to AI Efficiency Methods The ADL paradigm intersects with several established efficiency and training techniques, yet maintains fundamentally distinct objectives:

- **Knowledge Distillation (KD):** KD aims to transfer the generalization capabilities of a large "teacher" model to a smaller "student" model (Hinton et al., 2015; Gou et al., 2021). While ADL systems implicitly perform a form of dataset-specific distillation by training a classifier on oracle-generated labels (Hsieh et al., 2023), traditional KD is designed to produce a robust, standalone model for future deployment. In ADL, the student model operates with a permanent fallback to the teacher during the labeling process, explicitly abstaining from learning complex edge cases by routing them back to the oracle.
- **Model Routing:** Dynamic model routing dispatches inputs to the most appropriate expert model based on complexity, cost, or task domain (Shnitzer et al., 2023; Ong et al., 2025). ADL employs a specialized, data-centric variation of model routing. Crucially, whereas standard routing frameworks typically toggle between fixed, pre-trained destination models, an ADL system actively learns both the model and the routing threshold simultane-

ously. These components are continuously calibrated against a specific dataset's topology, rather than relying on a generalized router for open-ended inference.

- **Active Learning (AL):** Traditional AL selects the most informative samples for an oracle to label, aiming to minimize the annotation budget required to train a generalized classifier (Settles, 2009; Ren et al., 2021). While ADL systems (including several of our baselines) leverage AL algorithms as a sampling strategy to select which instances to route to the oracle, the end goal differs entirely. AL optimizes for the model's future predictive accuracy on unseen data; ADL optimizes for the immediate, accurate labeling of the current data pool. It leverages the oracle to cover the out-of-distribution long tail directly, rather than forcing a small model to master it.

3 Active Dataset Labeling (ADL)

The fundamental goal of an ADL system is to minimize expensive oracle usage during large-scale data annotation while maintaining oracle-level output performance. To ensure this formulation accommodates any algorithmic approach, we define the ADL problem strictly in terms of system inputs, outputs, and optimization goals, remaining agnostic to the specific labeling mechanisms, whether oracle and learned.

3.1 Problem Statement (ADL for Multi-Class Labeling)

Let $\mathcal{X}$ be the input space of textual documents and $\mathcal{Y} = \{1, \ldots, K\}$ be a discrete label space. We are given an unlabeled set of examples $\mathcal{U} = \{x_1, x_2, \ldots, x_N\}$, and an oracle function $\mathcal{O} : \mathcal{X} \to \mathcal{Y}$, where oracle invocations involve a sample-dependent cost $\mathcal{C}_\mathcal{O} : \mathcal{X} \to R$.

The goal is to assign labels $y_1, y_2..., y_N$ such that the maximal number of assigned labels y_i will match the oracle labels $y_i^\mathcal{O} = \mathcal{O}(x_i)$, while incurring minimal cost.[1]

A trivial way to reduce cost is to replace the oracle invocation with a cheaper labeling procedure, for as many samples as possible. Thus, an ADL algorithm should be able to identify informative samples to expose their oracle labels, as well as to assess its ability to assign an oracle-matching label to a yet-unlabeled samples, based on the exposed oracle-label samples. It should also strive to minimize its own computational cost.

We note that unlike other setups, like active-learning or online-learning, the goal of ADL is strictly to label the input set. It is specifically not concerned with generalization to samples outside of the given input set. Unlike online-learning, the oracle label is not exposed after making a prediction.

3.2 Variants

We now describe variants of the ADL setting:

3.2.1 Data Arrival Configurations: Batch vs. Streaming

Batch ADL: In the batch setting, the entire unlabeled pool $\mathcal{U}$ is available *a priori*. This allows the system to perform a global analysis over the entire dataset, such as density estimation, global clustering, or manifold mapping, before making any oracle queries or assigning any model labels. This allows for globally-informed partitioning strategies.

Streaming ADL: In the streaming setting, data instances arrive sequentially over time: $x_1, x_2, \ldots, x_T$. At each time step t, the system receives x_t and must assign its label before proceeding. This requires constant on-the-fly evaluation to determine whether to resolve its final label

[1]Note that the underlying assumption is that the oracle defines the true labeling: the goal of the ADL labeling is to match the oracle's predictions, and not some supposed underlying "truth" which may differ from it. This definition holds also for the real world use-cases where the oracle is realized by an ad-hoc classifier like a prompted-LLM which may err on some examples. The ADL system will be judged on its ability to replicate the LLM labels, and it will be penalized for predictions that differ from LLM predictions, even if these are "more correct".

by the model or escalate it to the oracle. The system has no prior knowledge of the overall data distribution or future instances, forcing it to dynamically update its internal representation and decision boundaries while processing the stream. After consulting the oracle with respect to an instance x_t, the system may use the label to improve its assessment of future labels. However, because model updating incurs additional computational time, the system must continuously balance the frequency of these updates against the overall inference latency (e.g., retraining a model from scratch at each step incurs a greater cost than doing it periodically, or performing a small update to an existing model). The streaming is superficially similar to online-learning, but with a crucial difference: the learner has to decide if they want to reveal the oracle label or not, revealing the oracle label incurs a cost, and not revealing it makes the final loss only partially observable.

3.2.2 Black Box vs. White Box Oracle

The oracle itself can also come in two variants: black-box and white-box.

Black-Box Oracle: The oracle acts as an opaque mapping function $\mathcal{O} : \mathcal{X} \to \mathcal{Y}$. and the system can only access the final output (the discrete label) $y_i^{\mathcal{O}}$. The system cannot observe internal model states or oracle-assigned confidence scores. This configuration is mirrors querying proprietary commercial prediction services via restrictive APIs.

White-Box Oracle: The system has full access to the oracle's inner working. For an LLM-based oracle this includes architecture, weights, intermediary activations, attention scores, logit-prediction scores, etc. A system could theoretically leverage the oracle's internal representations to assess confidence of predictions or similarity between examples, and a classifier may share parts or all of the oracle's computation (shared flops), e.g. in early-exit or learned layer-skipping schemes.

Thus, each of the setups exposes different challenges, with the black-box setup being more concerned with effective estimation and calibrartion based on few examples and making the most out of few bits of information, while the white-box setup encourages deep understanding of the underlying model's internals and inner working.

3.3 Quality Metrics

The goal of ADL balances two competing objectives: achieving the best labeling quality with respect to the oracle labeling (performance), and reducing the overall computation cost (efficiency). We present concrete metrics for these two criteria.

Performance Metrics Performance assesses the labeling capabilities of the system.

Accuracy. The overall accuracy evaluated across the entire dataset ($\mathcal{U}$), including both oracle-revealed labels and those assigned by the ADL system. Since oracle annotations are treated as the ground truth, this metric effectively measures the accuracy of the labeler scaled by the effectiveness, penalizing only the mistakes made by the model. The accuracy is the fraction of labels y_i that differ from $\mathcal{O}(x_i)$.

Macro Recall. For label-unbalanced datasets, accuracy can be misleading, as high accuracy can be obtained while heavily erroring on rare classes. To account for this, we use macro-recall: unweighted mean of the class-recall across the set of classes L assigned by the labeler:

$$\text{Macro Recall} = \frac{1}{|L|} \sum_{\ell \in L} \frac{TP_\ell}{N_\ell}$$

we define Macro Recall $= 0$ if $|L| = 0$ (i.e., the model did not predict any instances for any class).

Efficiency Metrics Efficiency measures how effectively the system reduces reliance on the oracle and optimizes computational resources. If we assume the oracle is significantly more expensive than any computation performed by the ADL algorithm, we could use the number of oracle calls as an indicator of efficiency. Metrics derived on the number of oracle calls are informative, and lend themselves to more clean, theoretical analyses. In the real world, the ADL computations also incur a cost which can be non-negligible, and it may be useful to compare ADL algorithms also on the computation cost of the algorithm itself. This gives rise to two metrics, *effectiveness* and *cost-reduction*:

Effectiveness. is based on the number of oracle calls (n_{oracle}), and measures the fraction of samples in $\mathcal{U}$ that were labeled without consulting the oracle. A good ADL algorithm will maximize effectiveness.

$$\text{effectiveness} = 1 - \frac{n_{\text{oracle}}}{|\mathcal{U}|}$$

Cost reduction. To quantify the computational efficiency, we consider the *cost ratio* between labeling using the ADL system and consulting the oracle for all the samples: $(\mathcal{C}_{\mathcal{O}}(\mathcal{U}^*) + \mathcal{C}_{ADL})/\mathcal{C}_{\mathcal{O}}(\mathcal{U})$, where $\mathcal{U}^*$ is the subset for which the oracle was consulted. A good ADL system should minimize the cost-ratio, with values above 1 indicating an ineffective ADL system. For consistency with the other metrics, whose values should be maximized rather than minimized, we move from cost-ratio to cost-reduction:

$$\text{cost-reduction} = 1 - \text{cost-ratio} = 1 - \frac{\mathcal{C}_{\mathcal{O}}(\mathcal{U}^*) + \mathcal{C}_{ADL}}{\mathcal{C}_{\mathcal{O}}(\mathcal{U})}$$

This quantity should be maximized, with inefficient systems having negative cost-reduction values.

Measuring cost. How do we measure cost in practice? metrics like latency or financial costs are time, implementation and hardware dependent. Instead we adopt FLOPs (Floating-Point Operations) as our cost metric. FLOPs strikes a balance between reflecting real-world cost and being reproducible, objective, hardware agnostic and stable over time. Our cost-reduction metric then changes to *FLOPs-reduction* (FR).

A remaining practical concern is FLOPs computation. For $\mathcal{C}_{ADL}$, FLOPs can be computed using software libraries such as `fvcore`, and the same for white-box oracles. For the oracle cost $\mathcal{C}_{\mathcal{O}}$, this is trickier, as this value is often opaque (e.g., when utilizing proprietary LLMs via APIs). In cases where the true oracle FLOPs-cost is unknown, one may resort to reporting only $\mathcal{C}_{ADL}$ rather than FLOPs-reduction. A better choice is to estimate the oracle FLOP costs of the proprietary model, and report the estimation together with the cost-reduction metric. (For convenience, our benchmark packages both oracle predictions and their FLOPs-estimates for each of its samples).

Quality by Accuracy Threshold To evaluate the system's ability to maintain performance constraints, it is informative to report the quality metrics across a range of strict mean accuracy requirements (quality@acc). For a given accuracy threshold, we measure the achievable effectiveness, macro-recall, and FLOPs/FLOPs reduction on each task.

3.4 Scoring ADL Systems

The above quality metrics assess the quality of an ADL system on a specific instance. However, a high-quality ADL system should work well on a range of instances (tasks). An *ADL evaluation set* consists of a set of *tasks* where each task is an ADL instance: a pair $(\mathcal{U}_i, \mathcal{O}_i)$ where $\mathcal{U}_i$ is a set of data points, and $\mathcal{O}_i$ is the labeling oracle for that specific data.

For each individual task t_i, we compute a set of evaluation metrics independently, yielding a distinct task-level score for each metric. These individual task scores are then aggregated into corresponding global scores. Specifically, each global score is calculated as the average of its respective metric across the results of all N tasks in the evaluation set (a system scored on k distinct metrics per task, will have k global scores for the evaluation set).

4 TeLa-Bench

To facilitate the empirical study of Active Data Labeling (ADL), we introduce TeLa-Bench, an ADL evaluation set comprising of a diverse set of ADL tasks, coupled with oracle-predictions and oracle cost estimates. Tele-Bench focuses on text-classification tasks for LLM-based oracles.

4.1 Desiderata for an ADL Benchmark

To be effective, an ADL evaluation set should include a *diverse range* of *realistic* ADL tasks. Unlike ML benchmarks that artificially balance classes on the one hand or filter out "easy" samples to test

deep reasoning on the other, an ADL benchmark must reflect the structural realities of real-world data streams. Our evaluation set is thus based on the following assumptions:

- **Naturally occuring classification tasks:** Each task should be naturally occuring, in the sense that a human defined it as an classification problem as interest.
- **Semantic Redundancy and Natural Imbalance:** Real-world data is inherently noisy, non-uniform, and highly repetitive. Preserving natural class imbalances and semantic redundancy is crucial, as the primary objective of an ADL system is to identify and locally process these predictable clusters, reserving the oracle strictly for the complex "long tail."
- **Domain and Scale Heterogeneity:** To rigorously test an ADL algorithm adaptability across different regimes, the benchmark must encompass high variance in dataset sizes, input text lengths, and label cardinalities (ranging from simple binary decisions to fine-grained multi-class taxonomies).

To be effective, an evaluation set should also be sufficiently small to allow researchesr to experimentation with many algorithms under a reasonable compute budget.

For the current (and first) ADL benchmark, we additionally chose to exclude tasks with fuzzy or subjective class boundaries (e.g., ordinal rating tasks like 1-5 star reviews, as they degrade the oracle's annotation consistency and impair the model's ability to accurately estimate its own confidence. They also makes it harder to compare ADL algorithms. We leave such tasks for future benchmarks.

Data Collection. We constructed TeLa-Bench by aggregating 30 diverse text classification datasets from Hugging Face and Kaggle. The selection process was guided by strict inclusion criteria: each task was required to contain a minimum of 1,000 samples to support meaningful model training and reliable evaluation. Furthermore, we prioritized high variance in dataset characteristics to reflect the heterogeneity of real-world enterprise data. Consequently, the benchmark encompasses a wide spectrum of text lengths (ranging from concise social media posts to extensive academic paragraphs), diverse domains (e.g., medicine, finance, cyber-security), and varying label cardinalities (from standard binary classification to fine-grained 28-class categorization).

Oracle Annotation. To establish the oracle's upper bound and provide a standardized testbed, we pre-computed the ground-truth labels for the entire benchmark using a typical model for such tasks (GPT-5-mini). For each task, we designed and iteratively optimized specific zero-shot prompts to elicit strict instruction-following.

During the generation process, we encountered isolated instances where the LLM returned conversational text rather than the requested discrete label. We mitigated these structural inconsistencies through an automated retry mechanism using the same prompt. In cases of persistent formatting failures, the prompts were further refined to strictly enforce the output schema, ensuring a valid labels. Note that while we provide these cached oracle responses to eliminate API costs during evaluation, a general ADL system is expected to interact with a live oracle.

Due to the extensive labeling rules, negative constraints, and contextual definitions provided to the oracle LLM, the optimized prompts are too lengthy for the main manuscript. Consequently, the complete set of prompts for all 30 tasks is provided in the supplementary material and will be released in our public codebase upon publication.

The resulting dataset. TeLa-Bench encompasses a corpus of over million annotated instances across its 30 constituent tasks, providing a scale that closely mirrors enterprise-level machine learning workloads. The datasets range in size from specialized, low-resource domains (e.g., Climate detection with $\sim$1,000 samples) to large-scale corpora (e.g., AG News and SST-2 with 50,000 samples each). The labeling criteria spans from standard binary classification (e.g., Spam or Phishing detection) to highly complex, fine-grained taxonomies containing up to 28 distinct classes (e.g., GoEmotions and Genre Classification). Furthermore, the input text lengths vary dramatically, encompassing short-form social media constraints (tweets and headlines) as well as long-form, domain-specific literature (PubMed paragraphs and BBC articles). The datasets in TeLa-Bench feature varying degrees of semantic redundancy and class imbalance, hallmarks of real-world data streams. Oracle labeling accuracy with respect to the gold-truth dataset labelling also varies across tasks, as is expected in

real world ADL usage. We remind that the goal of ADL is to match the oracle judgments, and not an external "gold truth".

Each task is comprised of classification instances, a prompt, label predictions by a gpt5-mini oracle, and per-instance estimated oracle cost. Table 3 in the Appendix characteristics summarizes the different tasks and the datasets they were derived from.

5 BASELINES

To establish a robust starting point for future research on ADL, we provide results over four baseline architectures. These range from standard heuristic and active-learning approaches to more advanced propagation methods, spanning a range of the efficiency-quality trade-off. The included baselines establish an empirical lower bound for the black-box ADL regime. Baselines B3 and B4 operate in the batch setting, whereas baselines B1 and B2 can also be applied in the streaming environment. Pseudocode for each algorithm are provided in the appendix.

5.1 B1: FIXED SAMPLE BASELINE

This baseline method operates as a static, single-shot routing framework (Appendix Algorithm 2).

The system initializes by randomly sampling a predefined percentage of the unlabeled pool and dispatching it to the oracle for annotation. The resulting labeled data is partitioned into training and validation splits. A sequence classifier is then fully trained from scratch on the training split.

Following classifier training, the system trains a confidence router using the hold-out validation split. The router is a binary classifier that takes the sequence classifier's softmax probability distributions as input. It is trained to predict whether the classifier's prediction matches the ground truth (1.0) or not (0.0). Because raw router outputs can be uncalibrated, they are subsequently transformed into reliable confidence scores using a calibration algorithm.

In the final auto-labeling phase, the model and router evaluate the entirety of the remaining unlabeled pool. Samples that achieve a calibrated router score exceeding a strict predefined threshold are assigned the classifier's labels and permanently removed from the pool. Any remaining instances that fail to meet this confidence threshold are subsequently routed to the oracle for manual annotation.

5.2 B2: TRAIN UNTIL CONFIDENT BASELINE

This baseline method operates as a sequential, all-or-nothing switching framework (Algorithm 3). The system fine-tunes a sequence classifier by querying the oracle in discrete batches from the stream until a global performance threshold, τ, is achieved.

In each iteration, the system queries the oracle to label a sequential batch of samples from the incoming stream. The resulting labeled data is partitioned into training and test splits. The classifier is then finetuned on the training set and evaluated on the hold-out test set to compute its current accuracy.

The distinguishing feature of this baseline is its global auto-labeling trigger. If the classifier's test accuracy meets or exceeds the predefined target threshold τ, the system immediately halts all further oracle querying. It then transitions entirely to an auto-labeling phase, assigning the classifier's labels to the entirety of the remaining data stream.

5.3 B3: ACTIVE LEARNING AND LEARNED SELECTOR (ALLS) BASELINE

This baseline method operates in an active learning loop (Algorithm 4), progressively expanding the model's capabilities while dynamically routing high-confidence predictions to the classifier and reserving uncertain samples for the oracle.

The pipeline initializes by querying the oracle for a diverse seed set, identified via clustering on the data representations.

Following initialization, ALLS enters an iterative active learning loop. In each iteration, the system performs standard entropy-based uncertainty sampling to select the most informative samples for oracle annotation, after which the sequence classifier is retrained.

The distinguishing feature of ALLS is its routing mechanism, which filters the unlabeled pool in each iteration to bypass the oracle when the classifier is sufficiently confident. The router is a binary classifier that takes the sequence classifier's softmax distributions as input. It is trained on a validation set to predict whether the classifier's prediction matches the ground truth (1.0) or not (0.0). Because raw outputs can be uncalibrated, they are transformed into reliable confidence scores using a calibration step. During the auto-labeling phase, any remaining unlabeled samples with a calibrated router score exceeding a strict threshold are assigned the classifier's pseudo-label and permanently removed from the active learning pool.

5.4 B4: Active Label Propagation Routing (ALPR) Baseline

This baseline method operates in an iterative loop (Algorithm 5), progressively expanding the model's capabilities while dynamically routing high-confidence predictions to the system and reserving uncertain samples for the oracle.

As we do in B3, the pipeline initializes by querying the oracle for a diverse seed set, identified via clustering on the data representations.

Following initialization, ALPR enters an iterative loop. In each iteration, the system performs uncertainty sampling to select the most informative samples for oracle annotation, after which the sequence classifier is retrained. However, ALPR enhances standard entropy sampling by utilizing density-weighted uncertainty, which combines the model's predictive entropy with graph centrality (node degree) to select batches.

The distinguishing feature of ALPR is its graph-based routing mechanism, which filters the unlabeled pool in each iteration to bypass the oracle when the system is sufficiently confident. Unlike B3, which routes based directly on the classifier's softmax outputs, ALPR leverages a Label Propagation algorithm (Zhu & Ghahramani, 2002), a semi-supervised technique that iteratively diffuses label distributions from annotated to unannotated nodes along the edges of a similarity graph. The system constructs an affinity graph over the entire dataset, weighting edges by semantic similarity, and propagates the known labels through the graph to the unlabeled pool.

To evaluate confidence, the system extracts the maximum probability from the label propagation distributions. These raw confidence scores are transformed into reliable statistical probabilities using a calibration model fitted on a hold-out validation set. During the auto-labeling phase, any remaining unlabeled samples with a calibrated router score exceeding a strict threshold are assigned the propagation's label and permanently removed from the pool.

5.5 Implementation Details

In our implementation across all baselines, the sequence classifier is a pre-trained `ModernBERT-base` model. For B1 and B3, the router classifier is a lightweight binary Multi-Layer Perceptron (MLP). The confidence scores, both from the MLP routers in B1 and B3, and the raw propagation probabilities in B4, are calibrated using Isotonic Regression. Additionally, where diverse seed sets are required (B3 and B4), they are identified via K-Means clustering on the mean-pooled representations of the data, and the similarity graph utilized in B4 is constructed using K-Nearest Neighbors over the dense dataset embeddings.

5.6 Results

To ensure robust and reliable results, all experiments were executed three independent times across the entire dataset, with the reported metrics reflecting the average outcomes. Table 1 presents the performance across the 30 tasks at accuracy thresholds (92%, 95%, and 98%). Note that for the strict 98% accuracy threshold, baseline B3 failed to maintain the required performance (indicated by dashes), while baseline B4 resulted in a negative FLOPs reduction (-16.1%), indicating that its estimated computational cost actually exceeded that of using an oracle.

Table 1: Performance comparison across different accuracy thresholds (in %). FR denotes FLOPs reduction: the percentage of estimated FLOPs saved relative to the estimated FLOPs required for an oracle.

Method	**Acc** $> 92\%$				**Acc** $> 95\%$				**Acc** $> 98\%$			
	Acc	Eff.	Rec.	FR	Acc	Eff.	Rec.	FR	Acc	Eff.	Rec.	FR
B1: Fixed sample	92.4	69.6	68.3	64.8	95.8	57.2	69.3	49.2	98.8	33.6	47.0	26.1
B2: Train until con.	92.6	58.0	58.0	58.8	95.7	49.2	49.1	43.8	98.6	24.6	24.5	22.6
B3: ALLS	92.2	66.3	73.2	51.8	95.6	55.4	28.2	–	–	–	–	–
B4: Active LP rout.	92.7	72.0	80.7	44.0	95.2	59.4	82.6	25.2	98.6	43.3	81.7	−16.1

5.7 The Challenge of Achieving Target Accuracy

During our experiments across the benchmark, we observed a persistent phenomenon: predictably achieving a desired target accuracy is highly challenging. In our baselines, the system's final accuracy and efficiency are controlled via a static confidence threshold. Because setting an a-priori threshold fails to predictably control the final performance, it becomes necessary to empirically evaluate multiple thresholds to reach the desired accuracy results.

This instability occurs for three primary reasons, ordered by their fundamental impact on the ADL setup. Most importantly, the overall accuracy metric evaluates the entire dataset, blending the model's predictions with the oracle's perfect ground truths. Consequently, the threshold heavily dictates the routing ratio (effectiveness); raising it can artificially inflate overall system accuracy simply by offloading more data to the oracle, completely masking potential degradations in the model's internal accuracy. Second, statistical calibration in the ADL setup is inherently weak due to extreme data scarcity. Because the overarching goal is to minimize oracle calls, the confidence router must be fitted on a very small validation set, causing the model to struggle with calibrating extreme confidence bounds. Finally, a threshold acts as a strict lower bound on the model's *estimated* confidence rather than a target average, inherently causing the empirical mean of the passing subset to over-shoot the threshold itself.

Table 2: Impact of the threshold parameter on internal and overall metrics for Baseline B2.

Threshold (τ)	**0.83**	**0.85**	**0.88**	**0.90**	**0.95**
Acc	92.6	93.5	95.7	96.7	98.6
Eff.	62.4	59.7	49.2	43.4	24.6

Table 2 illustrates this dynamic. For instance, applying a threshold of $\tau = 0.83$ yields an overall system accuracy of 92.6% while maintaining a robust effectiveness of 62.4%. However, attempting to enforce stricter quality by raising τ to 0.95 causes the system's effectiveness to collapse to just 24.6%. Because the router is poorly calibrated on such a small validation set, it fails to efficiently isolate the most accurate predictions. Instead, the overall system accuracy reaches 98.6% solely because the threshold acts as an extreme fail-safe, rejecting the vast majority of the workload and relying almost entirely on the perfect oracle to compensate. Because the threshold operates exclusively on these unreliable internal scores, there is no way to forecast the expected efficiency (i.e., cost savings) for a specific target accuracy on a novel task. Consequently, determining how to dynamically guarantee a target accuracy constraint on-the-fly, rather than relying on manual threshold evaluations, emerges as a critical open problem for future ADL research.

6 Conclusions

We introduced Active Data Labeling (ADL), a novel machine learning setup facilitated by prompted generative AI, which we believe to be of both practical and theoretical interets. We provide an evaluation set for ADL, as well as demonstrate a set of possible algorithms and their performance. In future work, we intend an in-depth study of improved ADL algorithms, and hope the research community will join us.

Reproducibility Statement

To ensure the full reproducibility of our empirical results and to facilitate future research on the Active Data Labeling (ADL) setup, we are committed to fully release all relevant materials needed for reproduction as open-source. First, we will provide the complete source code for the evaluation framework and the four baseline algorithms (B1–B4), accompanied by formal algorithmic pseudocode detailed in Appendix B. The codebase extensively documents all experimental hyperparameters, model architectures, and training configurations required to replicate the results presented in Table 1. Furthermore, we will provide the complete TeLa-Bench dataset, which encompasses the 30 text classification tasks detailed in Appendix 4.1. This includes the pre-computed oracle annotations and the exact, task-specific optimized LLM prompts utilized during the data creation phase. We commit to fully open-sourcing the benchmark, datasets, and framework repository upon publication.

Use of AI

The first author used an LLM to help polish the writing by consulting with them on specific paragraphs or phrasing. The first author also used LLM to help in literature review, by using them as additional source for locating relevant papers, beyond traditional search engines which were also used. Naturally, we also used an LLM to obtain LLM-derived labels for the benchmark, as described in the section on its creation.

References

Tiago A. Almeida, Jose Maria Gomez Hidalgo, and Akebo Yamakami. Contributions to the study of sms spam filtering: New collection and results. In *Proceedings of the 2011 ACM Symposium on Document Engineering (DOCENG'11)*, 2011.

Julia Bingler, Mathias Kraus, Markus Leippold, and Nicolas Webersinke. How cheap talk in climate disclosures relates to climate initiatives, corporate emissions, and reputation risk. Working paper, Available at SSRN 3998435, 2023.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. In Lluís Màrquez, Chris Callison-Burch, and Jian Su (eds.), *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 632–642, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1075. URL https://aclanthology.org/D15-1075.

Lingjiao Chen, Matei A. Zaharia, and James Y. Zou. Frugalgpt: How to use large language models while reducing cost and improving performance. *ArXiv*, abs/2305.05176, 2023. URL https://api.semanticscholar.org/CorpusID:258564349.

Dorottya Demszky, Dana Movshovitz-Attias, Jeongwoo Ko, Alan Cowen, Gaurav Nemade, and Sujith Ravi. GoEmotions: A Dataset of Fine-Grained Emotions. In *58th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2020.

Jianping Gou, Baosheng Yu, Stephen J. Maybank, and Dacheng Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129(6):1789–1819, March 2021. ISSN 1573-1405. doi: 10.1007/s11263-021-01453-z. URL http://dx.doi.org/10.1007/s11263-021-01453-z.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015. URL https://arxiv.org/abs/1503.02531.

Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alexander Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes, 2023. URL https://arxiv.org/abs/2305.02301.

Di Jin and Peter Szolovits. Hierarchical neural networks for sequential sentence classification in medical scientific abstracts. *arXiv preprint arXiv:1808.06161*, 2018.

Viviana Luccioli, Rithika Iyengar, Ryan Panley, Flora Haberkorn, Xiaoyu Ge, Leland Crane, Nitish Sinha, and Seung Jung Lee. Llm on a budget: Active knowledge distillation for efficient classification of large text corpora, 2025. URL https://arxiv.org/abs/2511.11574.

Julian McAuley and Jure Leskovec. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM conference on Recommender systems*, pp. 165–172, 2013.

Lun-Yiu Nie, Zhi-Min Ding, Erdong Hu, Christopher Jermaine, and Swarat Chaudhuri. Online cascade learning for efficient inference over streams. *ArXiv*, abs/2402.04513, 2024. URL https://api.semanticscholar.org/CorpusID:267522781.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M Waleed Kadous, and Ion Stoica. Routellm: Learning to route llms with preference data, 2025. URL https://arxiv.org/abs/2406.18665.

Bo Pang and Lillian Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the ACL*, 2005.

Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Brij B. Gupta, Xiaojiang Chen, and Xin Wang. A survey of deep active learning, 2021. URL https://arxiv.org/abs/2009.00236.

Burr Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009. URL http://axon.cs.byu.edu/~martinez/classes/778/Papers/settles.activelearning.pdf.

Tal Shnitzer, Anthony Ou, Mírian Silva, Kate Soule, Yuekai Sun, Justin Solomon, Neil Thompson, and Mikhail Yurochkin. Large language model routing with benchmark datasets, 2023. URL https://arxiv.org/abs/2309.15789.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1631–1642, Seattle, Washington, USA, October 2013. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D13-1170.

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, volume 28, 2015.

Xiaojin Zhu and Zoubin Ghahramani. Learning from labeled and unlabeled data with label propagation. In *Learning from Labeled and Unlabeled Data with Label Propagation*, 2002. URL http://www.scholar.google.com/url?sa=U&q=http://www.cs.cmu.edu/~zhuxj/pub/propagate.ps.gz.

A Detailed Dataset Statistics

Table 3: Summary of the 30 datasets comprising our evaluation benchmark

DATASET NAME	CLASSES	SIZE ($\times 10^3$)	SOURCE / CITATION
PubMed Paragraph Classification	5	29.5	Hugging Face
SMS Spam	2	5.6	(Almeida et al., 2011)
AG News Articles	4	50.0	(Zhang et al., 2015)
Market Sentiment	3	9.5	Hugging Face
Fake News Detection	2	44.2	Hugging Face
Climate Related	2	1.7	(Bingler et al., 2023)
Tweet Sentiment	2	50.0	Hugging Face
SST-2 Movie Review Sentiment	2	50.0	(Socher et al., 2013)
TREC Question Classification	6	6.0	Hugging Face
Hate Speech Detection	2	32.0	Hugging Face
Clickbait Headline	2	30.3	Hugging Face
Grammarly Correctness	2	50.0	Hugging Face
Rotten Tomatoes Sentiment	2	10.7	(Pang & Lee, 2005)
Physics vs. Chemistry vs. Biology	3	10.3	Kaggle
Yahoo Answers Topics	10	50.0	(Zhang et al., 2015)
Completed Sentence Detection	2	50.0	Kaggle
PubMed 20k RCT	5	50.0	(Jin & Szolovits, 2018)
Spotify User Sentiment	2	50.0	Kaggle
Natural Language Inference (SNLI)	3	50.0	(Bowman et al., 2015)
Ecommerce Description	4	50.0	Kaggle
Apple Twitter Sentiment	3	1.6	Kaggle
Phishing URL Detection	2	50.0	Hugging Face
GoEmotions	28	48.8	(Demszky et al., 2020)
Genre Classification IMDb	27	50.0	Kaggle
Name to Gender	2	50.0	Kaggle
Amazon Polarity	2	50.0	(McAuley & Leskovec, 2013)
Sexism Detection	2	20.0	Kaggle
Social Media Stress Detection	2	5.6	Kaggle
Goodreads Book Genres	18	39.6	Hugging Face
Content Articles Topics	6	21.0	Hugging Face

B Baseline Algorithms Pseudocode

To facilitate reproducibility, we provide the formal pseudocode for the four baseline routing strategies evaluated on TeLa-Bench.

In the algorithms presented below, we utilize some auxiliary functions:

Algorithm 1: Auxiliary functions

Function *TrainClassifier*($\mathcal{D}$)
- **Input:** Labeled dataset $\mathcal{D}_{train}$;
- **Output:** Trained classification model f;
- Initialize base model parameters;
- Optimize f to minimize classification loss on $\mathcal{D}$;
- **return** f;

Function *TrainCalibratedRouter*($f, \mathcal{D}_{calib}$)
- **Input:** Trained classifier (or probability model) f, Labeled calibration set $\mathcal{D}_{calib}$;
- **Output:** Calibrated router g;
- Extract predictions $f(x)$ for all $(x, y) \in \mathcal{D}_{calib}$;
- Fit router g to estimate true prediction confidence based on $f(x)$ and true labels y;
- **return** g;

Function *KMeansSample*($\mathcal{U}, n$)
- **Input:** Unlabeled pool $\mathcal{U}$, Number of samples to select n;
- **Output:** Diverse subset $\mathcal{S}_{seed}$ of size n;
- Extract mean-pooled feature representations for all $x \in \mathcal{U}$;
- Obtain n cluster centroids $C = \{c_1, \ldots, c_n\}$ using K-Means clustering;
- $\mathcal{S}_{seed} \leftarrow \{\arg\min_{x \in \mathcal{U}} \text{distance}(x, c_i) \mid c_i \in C\}$;
- **return** $\mathcal{S}_{seed}$;

Function *LabelPropagation*($f, \mathcal{U}, \mathcal{D}_{train}$)
- **Input:** Trained classifier f, Entire dataset $\mathcal{U}$, Labeled source data $\mathcal{D}_{train}$;
- **Output:** Label probability distributions P_{LP};
- Extract deep feature embeddings for all $x \in \mathcal{D}_{all}$ using f;
- Construct a K-Nearest Neighbors (KNN) similarity graph over the embeddings;
- Iteratively propagate labels from $\mathcal{D}_{train}$ through the graph to estimate class probabilities;
- **return** P_{LP};

Algorithm 2: Fixed sample baseline

Input: Unlabeled pool $\mathcal{U}$, oracle $\mathcal{O}$;
Hyper-parameters: Threshold τ, Initial training size n, Calibration size k;
Output: Fully labeled dataset $\mathcal{D}_{labeled}$;

$\mathcal{S}_{train} \leftarrow \text{Sample}(\mathcal{U}, n)$;
$\mathcal{S}_{calib} \leftarrow \text{Sample}(\mathcal{U} \setminus \mathcal{S}_{train}, k)$;
$\mathcal{D}_{train} \leftarrow \mathcal{O}(\mathcal{S}_{train})$;
$\mathcal{D}_{calib} \leftarrow \mathcal{O}(\mathcal{S}_{calib})$;
$\mathcal{U} \leftarrow \mathcal{U} \setminus (\mathcal{S}_{train} \cup \mathcal{S}_{calib})$;
$f \leftarrow TrainClassifier(\mathcal{D}_{train})$;
$g \leftarrow TrainRouter(f, \mathcal{D}_{calib})$;
$\mathcal{U}_{auto} \leftarrow \{x \in \mathcal{U} \mid g(f(x)) > \tau\}$;
$\mathcal{D}_{auto} \leftarrow \{(x, \text{argmax}(f(x))) \mid x \in \mathcal{U}_{auto}\}$;
$\mathcal{D}_{labeled} \leftarrow \mathcal{D}_{train} \cup \mathcal{D}_{calib} \cup \mathcal{D}_{auto} \cup \mathcal{O}(\mathcal{U} \setminus \mathcal{U}_{auto})$;
return $\mathcal{D}_{labeled}$;

Algorithm 3: Train until confident baseline

Input: Unlabeled pool $\mathcal{U}$, oracle $\mathcal{O}$;
Hyper-parameters: Batch size n, Test size ratio b, Threshold τ;
Output: Fully labeled dataset $\mathcal{D}_{labeled}$;

$\mathcal{S}_{init} \leftarrow$ Sample$(\mathcal{U}, n)$;
$\mathcal{D}_{train}, \mathcal{D}_{test} \leftarrow$ Split$(\mathcal{O}(\mathcal{S}_{init}), b)$;
$\mathcal{U} \leftarrow \mathcal{U} \setminus \mathcal{S}_{init}$;

while $\mathcal{U} \neq \emptyset$ **do**
  $f \leftarrow$ TrainClassifier$(\mathcal{D}_{train})$;
  if *Accuracy*$(f, \mathcal{D}_{test}) \geq \tau$ **then**
    $\mathcal{D}_{auto} \leftarrow \{(x, \text{argmax}(f(x))) \mid x \in \mathcal{U}\}$;
    break;
  else
    $\mathcal{S}_{batch} \leftarrow$ Sample$(\mathcal{U}, n)$;
    $\mathcal{D}_{trainBatch}, \mathcal{D}_{testBatch} \leftarrow$ Split$(\mathcal{O}(\mathcal{S}_{batch}), b)$;
    $\mathcal{D}_{train} \leftarrow \mathcal{D}_{train} \cup \mathcal{D}_{trainBatch}$;
    $\mathcal{D}_{test} \leftarrow \mathcal{D}_{test} \cup \mathcal{D}_{testBatch}$;
    $\mathcal{U} \leftarrow \mathcal{U} \setminus \mathcal{S}_{batch}$;

$\mathcal{D}_{labeled} \leftarrow \mathcal{D}_{train} \cup \mathcal{D}_{test} \cup \mathcal{D}_{auto}$;
return $\mathcal{D}_{labeled}$;

Algorithm 4: Active learning and learned selector (ALLS) baseline

Input: Unlabeled pool $\mathcal{U}$, oracle $\mathcal{O}$;
Hyper-parameters: Batch size n, Calibration size ratio b, Threshold τ;
Output: Fully labeled dataset $\mathcal{D}_{labeled}$;

$\mathcal{S}_{init} \leftarrow$ KMeansSample$(\mathcal{U}, n)$;
$\mathcal{D}_{labeled} \leftarrow \mathcal{O}(\mathcal{S}_{init})$;
$\mathcal{U} \leftarrow \mathcal{U} \setminus \mathcal{S}_{init}$;
$f \leftarrow$ TrainClassifier$(\mathcal{D}_{labeled})$;

while $\mathcal{U} \neq \emptyset$ **do**
  $\mathcal{S}_{batch} \leftarrow \text{argmax}^{n}_{x \in \mathcal{U}}$Entropy$(f(x))$;
  $\mathcal{D}_{labeled} \leftarrow \mathcal{D}_{labeled} \cup \mathcal{O}(\mathcal{S}_{batch})$;
  $\mathcal{D}_{train}, \mathcal{D}_{calib} \leftarrow$ Split$((\mathcal{D}_{labeled}), b)$;
  $\mathcal{U} \leftarrow \mathcal{U} \setminus \mathcal{S}_{batch}$;
  $f \leftarrow$ TrainClassifier$(\mathcal{D}_{train})$;
  $g \leftarrow$ TrainCalibratedRouter$(f, \mathcal{D}_{calib})$;
  $\mathcal{U}_{auto} \leftarrow \{x \in \mathcal{U} \mid g(f(x)) > \tau\}$;
  $\mathcal{D}_{auto} \leftarrow \{(x, \text{argmax}(f(x))) \mid x \in \mathcal{U}_{auto}\} \cup \mathcal{D}_{auto}$;
  $\mathcal{U} \leftarrow \mathcal{U} \setminus \mathcal{U}_{auto}$;

$\mathcal{D}_{labeled} \leftarrow \mathcal{D}_{labeled} \cup \mathcal{D}_{auto}$;
return $\mathcal{D}_{labeled}$;

Algorithm 5: Active Label Propagation Routing (ALPR) baseline

Input: Unlabeled pool $\mathcal{U}$, oracle $\mathcal{O}$;
Hyper-parameters: Batch size n, Calibration size ratio b, Threshold τ;
Output: Fully labeled dataset $\mathcal{D}_{labeled}$;

$\mathcal{S}_{init} \leftarrow$ KMeansSample$(\mathcal{U}, n)$;
$\mathcal{D}_{labeled} \leftarrow \mathcal{O}(\mathcal{S}_{init})$;
$\mathcal{U} \leftarrow \mathcal{U} \setminus \mathcal{S}_{init}$;
$f \leftarrow$ TrainClassifier$(\mathcal{D}_{labeled})$;

while $\mathcal{U} \neq \emptyset$ **do**
  $\mathcal{S}_{batch} \leftarrow \text{argmax}^{n}_{x \in \mathcal{U}}$Entropy$(f(x))$;
  $\mathcal{D}_{labeled} \leftarrow \mathcal{D}_{labeled} \cup \mathcal{O}(\mathcal{S}_{batch})$;
  $\mathcal{D}_{train}, \mathcal{D}_{calib} \leftarrow$ Split$(\mathcal{D}_{labeled}, b)$;
  $\mathcal{U} \leftarrow \mathcal{U} \setminus \mathcal{S}_{batch}$;
  $f \leftarrow$ TrainClassifier$(\mathcal{D}_{train})$;
  $P_{LP} \leftarrow$ LabelPropagation$(f, \mathcal{D}_{labeled} \cup \mathcal{U}, \mathcal{D}_{train})$;
  $g \leftarrow$ TrainCalibratedRouter$(P_{LP}, \mathcal{D}_{calib})$;
  $\mathcal{U}_{auto} \leftarrow \{x \in \mathcal{U} \mid g(P_{LP}(x)) > \tau\}$;
  $\mathcal{D}_{auto} \leftarrow \mathcal{D}_{auto} \cup \{(x, \text{argmax}(P_{LP}(x))) \mid x \in \mathcal{U}_{auto}\}$;
  $\mathcal{U} \leftarrow \mathcal{U} \setminus \mathcal{U}_{auto}$;

$\mathcal{D}_{labeled} \leftarrow \mathcal{D}_{labeled} \cup \mathcal{D}_{auto}$;
return $\mathcal{D}_{labeled}$;