From Scores to Workflows: An Interactive Framework for Data Attribution in Foundation Models

Weizhen (Gary) Gao

CMU-CS-26-109

May 2026

Computer Science Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee

Chenyan Xiong, Chair
Alexander Rudnicky

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science.

Keywords: Large Language Model, Foundation Model, Data Attribution

To my friends and family.

Abstract

Training data attribution seeks to identify which training examples are most related to a model prediction or user query, yet applying attribution methods to modern foundation models practice often brings difficulties to users as such methods could be computationally expensive, method-specific, and hard to inspect within a single analysis process. This thesis presents an interactive framework for training data attribution that combines live query computation with precomputed training-side reference representations, enabling users to retrieve, compare, and interpret ranked training examples for live queries or selected validation examples within a unified workflow. The framework centers on gradient-based attribution methods, including gradient similarity, DataInf-style influence approximation over projected gradient features, LESS-style low-rank gradient matching, and LoGra-style features. It also supports multi-query analysis through score aggregation and comparative inspection across methods, allowing attribution to be analyzed as a workflow rather than as an isolated score. Demonstrations are conducted on public question-answering datasets with Pythia-family models, on a multimodal driving-video setting with Qwen3-VL, and on a medical-domain setting that extends the framework to private and domain-specific data. This thesis shows that training data attribution becomes substantially more useful when it is treated not only as a scoring method, but as a practical workflow for querying, comparing, and interpreting training examples across models, methods, and datasets.

Acknowledgments

Firstly, I would like to thank my research advisor, Professor Chenyan Xiong, for the guidance and support throughout the time I participated in research at CMU. His expertise and advice made this experience invaluable. I would like to thank Professor Alexander Rudnicky for joining my thesis committee and providing thoughtful feedback for my thesis. Their kind illumination and consideration has greatly helped me in improving this thesis.

I would like to thank the great group members of Professor Chenyan Xiong's research group. They have inspired this thesis project and provided vital help as I explore the ground of academic research; special thanks to Cathy Jiao for her mentorship.

I would like to thank the program directors and managers from the Fifth-Year CS Master's Program. Their coordination helped me navigate through many logistical difficulties.

Finally, I would like to thank my friends and family. Their support helped me go through the most difficult times in my academic life.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Motivation

Foundation models have changed how machine learning systems are developed and reused. A single pretrained model can support a wide range of downstream tasks, and closely related modeling ideas now span language, vision, multimodal reasoning, and domain-specific settings. As these models become more capable, they also become more difficult to analyze in terms of the training examples that shape their behavior. In many settings, users are not satisfied with only the output of a model. They also want to understand which training data appear most related to a prediction, a retrieved response, or a model behavior that deserves further inspection. This question appears naturally in model debugging, dataset curation, data valuation, and the analysis of factual or task-specific knowledge in large models [23, 13, 18, 1, 31].

Training data attribution provides one way to answer that question. At a high level, it connects a model output or query to the training examples that are most relevant to it under an attribution rule. Classical influence-based methods remain a central reference point in this literature because they formalize how a training example affects model parameters and downstream predictions [23]. Later work has expanded this direction toward methods that are more suitable for modern foundation models, including projected-gradient methods such as TRAK, efficient influence approximations such as DataInf, targeted data-selection methods such as LESS, and low-rank approaches such as LoGra [31, 26, 41, 8]. At the same time, recent work has shown that attribution is not confined to one algorithmic family. In-context probing, for example, can track gradient-based attribution under suitable similarity conditions, which broadens the methodological landscape and makes comparison between attribution methods especially important [21].

The importance of attribution also extends beyond post-hoc explanation. In data valuation and data pricing, attribution-style scores can influence concrete decisions about which data should be selected, purchased, or prioritized for training. Recent work on fairshare data pricing, for instance, uses valuation methods such as BM25, InflIP, and DataInf to guide purchasing decisions in simulated large-language-model data markets [43]. Such applications make the interpretability of attribution rankings more important. If attribution scores support data curation or valuation decisions, then users need to understand not only aggregate outcomes, but also which examples are being surfaced by each method and how those rankings differ.

These developments have made attribution more practical, but they have also exposed a different limitation. In many projects, attribution is still treated mainly as a backend computation. A method may produce a ranking over training examples, yet that ranking often lives in a method-specific script, notebook, or offline artifact store. Researchers may still need to rebuild workflows by hand in order to issue live queries, compare methods on the same input, inspect agreement near the top of a ranking, or move from a text setting to a multimodal or domain-specific one. The usefulness of attribution therefore depends not only on the scoring rule, but also on whether the result can be queried, compared, and inspected in a consistent way.

This practical gap becomes sharper for modern foundation models. Large models and large training sets make attribution expensive enough that the training side and the current input are often handled differently. Training-side representations are commonly precomputed, projected, and stored, while live inputs still need to be processed when the user asks a new question. Contemporary use cases also increasingly involve heterogeneous data. A framework that works only for short text examples is less useful when users need to inspect video clips or structured medical records. The challenge, then, is not only how to define an attribution score. It is how to turn attribution into a coherent workflow that supports live use, method comparison, and extension across settings.

This thesis is motivated by that challenge. It takes the view that training data attribution becomes substantially more useful when it is embedded in a framework for querying, comparison, and inspection across models, methods, and data types. This perspective shifts the emphasis from isolated rankings toward operational workflows. It also matches the way attribution is used in research practice: as a tool for understanding why particular training examples surface, how methods differ, and what those differences reveal about model behavior.

1.2 Overview

This thesis presents an interactive framework for training data attribution in foundation models. The central idea is straightforward: attribution should be exposed as a workflow rather than as a standalone score. The framework therefore combines live query computation with precomputed training-side reference representations, and uses that structure to support ranked retrieval of related training examples, side-by-side comparison of attribution methods, overlap analysis, multi-query aggregation, and inspection of results through a shared interface. Within this design, a user can issue a live query or select an existing validation example, compute attribution under a chosen method, and inspect the resulting training examples in a form that supports further reasoning rather than one-off score reporting.

The framework focuses on gradient-based attribution methods suitable for modern foundation models. Its method layer includes gradient-similarity approaches together with a DataInf-style influence approximation, LESS, and LoGra [26, 41, 8]. These methods are valuable not only because they produce rankings, but also because they reveal different notions of training-example relevance. Some highlight direct gradient alignment, some reflect low-rank or projected representations, and some behave more like efficient influence approximations. Bringing them into one framework makes their relationships visible: agreement can be inspected, disagreement can be interpreted, and method-specific behavior can be studied under the same user interaction

model.

A second focus of the thesis is breadth across settings. The framework is developed and demonstrated on public text question-answering datasets with Pythia-family models, on a multimodal driving-video setting with Qwen3-VL, and on a medical-domain pathway built around codified EHR data and the CatchFM-160M model [37]. Together these settings show that the same framework logic can support public text tasks, multimodal video records, and domain-specific assets. That breadth matters because attribution workflows often break when data formats, model interfaces, or result displays change. A framework that remains usable across these differences is more valuable than a method-specific pipeline tied to one benchmark.

The thesis evaluates the framework primarily through workflow demonstrations and qualitative analysis rather than through a claim that one attribution method is universally best. The results show how users can inspect single-method rankings, combine several queries through score averaging, compare methods using full-ranking and top-K agreement, and examine returned examples across text, video, and medical settings. Additional runtime estimates show that, once reference artifacts are prepared, the interactive workflow can be used on a single workstation-class GPU. Preliminary user feedback further supports the central design direction: users found example-level inspection and comparison mode useful, while also identifying concrete improvements for metric explanation, score calibration, validation search, and method onboarding.

Taken together, the thesis advances a workflow-oriented view of training data attribution. It shows that attribution becomes easier to use, compare, and interpret under a framework that supports querying, aggregation, comparison, and modality-aware inspection across settings that would otherwise remain disconnected.

Chapter 2

Related Work

Research on training data attribution spans several neighboring literatures, and those literatures do not always use the same language or pursue the same objective. In one line of work, the goal is *contributive* attribution: identifying which training examples shaped a model prediction, generation, or behavior. In another, the goal is data valuation: estimating the utility of a datum or subset for model performance, data selection, or data pricing. In language-model applications, similar questions also appear as fact tracing, knowledge provenance, and training-data selection. A useful starting point is therefore to separate evidence attribution for generated claims from contribution attribution over training data. Worledge et al. argue that the word *attribution* has become overloaded in the language-model literature, where it may refer either to external evidence for a statement or to the contribution of training examples to model behavior [39]. This thesis focuses on the latter problem: relating model behavior back to training examples and making that relationship inspectable through a shared workflow.

That framing matters because the practical demands of training data attribution differ from those of citation or evidence attribution. A contributive attribution method must define what it means for a training example to matter to a prediction, approximate that quantity at a tractable cost, and surface the resulting relationships in a form that users can inspect. For modern foundation models, these three requirements are tightly coupled. The core challenge is no longer only to derive an attribution score, but also to make that score usable across large models, large datasets, and heterogeneous inputs. The literature has responded to this challenge along several axes: classical influence-function analysis, scalable gradient-based approximations, data-selection methods, proxy methods such as in-context probing, provenance-oriented evaluation, and interactive attribution tools [23, 33, 31, 26, 41, 8, 27]. The present thesis sits at the intersection of these directions. It studies how attribution becomes more useful when several method families are exposed through a common interactive framework.

2.1 Training Data Attribution as a Problem

2.1.1 From local influence to data value

The classical formulation of training data attribution is instance influence. Koh and Liang adapt influence functions from robust statistics to modern supervised learning and ask how model parameters, and consequently a target prediction, would change if a training example were infinitesimally upweighted [23]. This framing remains foundational because it gives training data attribution a counterfactual interpretation: a point is influential when changing its role in training would change the target behavior. The attraction of this view is conceptual clarity. It ties attribution directly to training dynamics rather than to surface similarity or post-hoc plausibility.

A second strand broadens the problem from local influence on one prediction to the value of data more generally. Data Shapley defines the value of a training example through its marginal contribution across subsets of the dataset, importing the axiomatic language of cooperative game theory into machine learning [13]. Datamodels move in a different but related direction by asking whether one can learn a function that maps training-set composition to model behavior on a fixed target point [18]. These works are not interchangeable with influence functions, but they make two ideas explicit that remain important throughout the attribution literature. First, attribution is not only about explanation after the fact; it is also about understanding how data governs model behavior. Second, the same methodological machinery can support several downstream goals, including debugging, data curation, model criticism, and efficient retraining.

This broader view naturally connects training data attribution to data selection. The language-model community has increasingly treated data selection as a first-class problem, especially in settings where the available training pool is much larger than what can be used efficiently. Albalak et al. survey this space and organize methods ranging from rule-based filtering to model-aware selection [2]. Within that landscape, some methods estimate the utility of data through explicit interaction with the target model, while others learn proxies for those utilities. DsDm uses datamodels to cast dataset selection as an optimization problem over target tasks rather than as heuristic filtering by human notions of quality [10]. MATES goes further toward dynamic model-awareness by repeatedly estimating local data influence and training a small influence model to predict useful data for the next stage of pretraining [42]. These papers address different stages of the training lifecycle than the present thesis, but they share a common premise: useful data analysis depends on understanding how candidate examples affect later model behavior.

2.1.2 Attribution in foundation models

Foundation models changed the scale and meaning of attribution. In smaller supervised models, one can often speak about the effect of a training example on a single prediction without much ambiguity. In foundation models, the same training example may affect a wide range of behaviors, and the relevant target can be a generated sequence, an answer, a latent representation, or a downstream capability. The attribution problem therefore becomes less local and more behavioral. TRAK makes this point explicitly by defining the goal as tracing *model behavior* back to training data rather than only explaining one scalar loss value [31]. This broader framing is useful for a thesis on workflow-oriented attribution because it fits how practitioners actually

use attribution results: not as a final causal verdict, but as evidence for understanding model behavior.

Language models also sharpen the distinction between causal contribution and explicit support. A model may answer a factual question because it encountered a sentence that directly states the fact, but it may also rely on examples that reinforce a relation type, an entity pattern, or a distributional prior. Fact tracing work makes this issue concrete. Akyürek et al. formulate the task of tracing a model assertion back to the training examples that supplied its evidence and show that both gradient-based and embedding-based methods still leave substantial headroom on this problem [1]. Their benchmark is especially important because it reveals that retrieval of explicit supporting passages and attribution of causal contribution are related but not identical tasks. This distinction recurs throughout the literature and motivates the use of lexical retrieval as a contrast method rather than as a replacement for gradient-based attribution.

2.1.3 Training data provenance and evaluation difficulties

A large part of the modern language-model attribution literature is motivated by provenance rather than by standard supervised explanation. At pretraining scale, gradient-based influence can recover examples that appear causally relevant to a prediction, even when those examples do not literally state the fact in the most direct way [7]. This suggests that provenance has multiple layers. Some retrieved examples are evidential because they contain explicit supporting information. Others are influential because they reinforce a concept, entity, reasoning pattern, or relation that supports the model behavior indirectly.

Evaluation remains difficult for the same reason. A method may retrieve training examples that are useful for understanding a prediction even when those examples would not be judged as citations or evidence in the conventional sense. Conversely, examples that are lexically close to a target may not be the ones that most shaped the model's behavior. Recent attribution benchmarks emphasize this tension and show that automatic attribution evaluation remains challenging even when the target is framed as evidence support rather than contributive influence [28]. This motivates workflows that let users inspect competing notions of relevance instead of collapsing them into a single correctness label.

2.1.4 Attribution, selection, and data markets

The connection between attribution and pretraining data selection has become stronger in recent work. QuRating approaches data selection through model-assisted judgments of document quality [38], while DsDm and MATES use learned or approximated influence signals to select training subsets that improve downstream performance [10, 42]. These methods are not direct substitutes for per-query attribution. They optimize training corpora at a coarser level and usually target aggregate model quality or specific downstream tasks. Still, they demonstrate that influence-like signals can guide training decisions at scale. Attribution is therefore not only a post-hoc analytic instrument; it also participates in training-time decision making.

A related question is what kinds of training examples support emergent behaviors such as in-context learning. Studies on supportive pretraining data show that certain pretraining examples

appear especially helpful for later in-context learning ability [15]. This is not the same as per-example attribution for a target query, but it reinforces the broader claim that model behavior and training-data composition are tightly coupled. It also highlights why connecting behaviors back to training data is valuable even when the immediate goal is not to score one target output.

Data valuation has also been connected to economic mechanisms for LLM training data. Fairshare Data Pricing proposes a framework in which buyers use data valuation methods to decide which training examples to purchase, while sellers set prices based on anticipated buyer demand [43]. Its empirical studies use tasks such as MathQA, MedQA, and PIQA and compare valuation methods including BM25, InflIP, and DataInf. This line of work is relevant because it shows that attribution scores can influence concrete decisions about training-data acquisition. At the same time, such work typically reports aggregate outcomes such as buyer utility, model performance, or cost-performance tradeoffs. An interactive attribution framework provides a complementary view by making the underlying rankings inspectable at the level of individual queries and returned training examples.

2.2 Gradient-Based Attribution Methods

2.2.1 Classical and first-order methods

Influence functions

Influence functions remain the canonical starting point for gradient-based attribution. They estimate how a training point would change a target prediction through its effect on the learned parameters, typically via a Hessian-inverse–gradient product [23]. The conceptual strength of influence functions is that they approximate the leave-one-out or upweighting counterfactual that attribution would ideally answer. The practical weakness is that they depend on second-order information that is difficult to estimate accurately in large, non-convex neural networks. As a result, influence functions have often been more valuable as a conceptual template than as a directly deployable method at foundation-model scale.

That tension did not disappear in the era of large language models. Grosse et al. show that influence functions can still reveal substantial structure in large-model generalization, including sparsity in influence patterns and qualitative differences across behaviors, but doing so requires careful approximations to curvature and nontrivial engineering to make the computation feasible [14]. Their work is an important reminder that the limiting factor in attribution is often not whether the idea is meaningful, but whether the required approximations and infrastructure preserve enough fidelity to remain interpretable.

Gradient trajectories and checkpoint-based approximations

A major practical alternative to explicit Hessian inversion is to use the training trajectory itself. TracIn estimates the influence of a training example on a target by tracking how the target loss changes along gradient descent when that example is seen, approximated through gradient inner products at saved checkpoints [33]. This idea is important for two reasons. First, it shows

that useful influence estimates can be obtained without explicit second-order computation. Second, it frames attribution as something that can piggyback on ordinary training artifacts such as checkpoints and gradients, a design principle that later scalable methods continue to exploit.

The appeal of checkpoint-based approximations is especially clear in modern deep learning, where exact counterfactual retraining is infeasible but gradient information is already part of the training process. TracIn therefore occupies a key middle ground: it preserves a direct connection to optimization while remaining implementable with standard training pipelines. This lineage suggests a natural division of labor between expensive offline artifacts and lightweight scoring on a current query.

2.2.2 Scaling gradient attribution to foundation models

Projected-gradient methods

TRAK is a landmark in the large-scale attribution literature because it changes the practical unit of computation. Instead of operating directly on full per-example gradients, TRAK uses random projection and after-kernel approximations to obtain compressed training-side representations that can be reused across queries [31]. Conceptually, this is a major shift. It turns attribution from a computation that must be repeated from scratch into one that can be organized around a precomputed reference store.

TRAK is also important because it shows that random projection is not merely a memory trick. It is part of a broader rethinking of how attribution should be executed at scale. Once the reference side is compressed into reusable representations, attribution begins to look more like retrieval over a specialized feature store. That perspective has shaped much of the later work in scalable attribution.

DataInf and efficient influence under parameter-efficient tuning

DataInf develops a different route to scalability. Rather than building a general projected-representation framework, it derives a closed-form approximation to the influence function that is especially well suited to LoRA-tuned models [26]. The method exploits the structure induced by parameter-efficient fine-tuning, where the trainable space is dramatically smaller than the full model. This is a particularly important development for foundation models because LoRA-style adaptation is now standard in many downstream pipelines [17]. Once fine-tuning is confined to a low-rank parameter subspace, attribution can target that same subspace rather than the full dense model.

The significance of DataInf is both algorithmic and representational. Algorithmically, it offers an efficient approximation to influence. Representationally, it aligns attribution with the parameter geometry of modern fine-tuning. For a system that compares attribution methods under one interface, this means DataInf is not simply another similarity score; it encodes a distinct view of where influence should be measured.

LESS and targeted instruction tuning

LESS addresses attribution from the perspective of targeted instruction tuning. Its key question is not only which examples influenced a given prediction, but which instruction examples should be selected so that a model acquires a desired downstream capability [41]. The method combines a warmup model, optimizer-aware influence approximations, and a low-dimensional gradient datastore that can be searched using gradient similarity to a few-shot validation set. This makes reusable gradient stores central to the method rather than incidental to the implementation.

LESS also sharpens the connection between attribution and data selection. Many attribution papers stop at ranking training points for explanation. LESS uses the ranking to select a small subset of instruction data that can outperform training on the full set for targeted skills. That framing shows why access to attribution rankings is useful: rankings are often most valuable when they become part of a larger workflow of inspection, comparison, and downstream decision making.

LoGra and low-rank gradient projection

LoGra, introduced in the context of LLM-scale data valuation with influence functions, tackles the storage and throughput bottleneck of gradient-based attribution by exploiting low-rank structure in backpropagation [8]. Instead of treating projection as an external compression step alone, the method is motivated by the internal structure of gradients in large transformer layers. The accompanying LogIX software package is also notable because it lowers the implementation barrier for influence-style data valuation by helping convert ordinary training pipelines into data valuation pipelines.

This line of work makes clear that scalable attribution is partly a software problem. A method may be theoretically appealing, but it becomes practically useful only when its gradient collection, storage, and scoring can be integrated into a reusable system. LoGra therefore belongs not only to the literature on low-rank gradient methods, but also to the emerging literature on attribution tooling.

2.3 Proxy and Non-Gradient Approaches

2.3.1 Retrieval-based contrasts

BM25 and lexical support

BM25 remains an important reference point in the attribution literature. As a probabilistic retrieval baseline, it is model-agnostic and depends only on lexical matching between the target and candidate examples [34]. On the surface, that seems much weaker than gradient-based attribution. Yet in fact tracing and provenance settings it can perform competitively, or even outperform neural attribution methods, when the goal is to find explicit passages that state the relevant fact [1, 7]. This is not an argument against gradient-based attribution. It shows that lexical support and causal contribution are distinct, and that a useful framework should make their difference visible.

Why retrieval still matters

The persistence of retrieval baselines is especially important for a system thesis. If a framework only exposes one method family, users cannot tell whether a retrieved training example surfaces because of model-conditioned influence or because of ordinary lexical similarity. A retrieval baseline provides a strong non-gradient contrast for precisely that reason. It anchors interpretation. When BM25 and a gradient method agree, the result may reflect direct textual support. When they disagree, the disagreement itself becomes informative.

2.3.2 In-context and prompt-based proxies

Prompt retrieval and demonstration selection

Researchers have also studied how to choose demonstrations or candidate examples for in-context learning. Rubin et al. learn to retrieve prompts for in-context learning [35]. Iter et al. propose cross-entropy difference for in-context demonstration selection [19]. Nguyen and Wong connect demonstration selection more directly to influence by using influence-style ideas to guide example choice [30]. These works are not training data attribution in the strict contributive sense, but they are strongly adjacent. They all ask how a small set of examples should be matched to a target input so that the model behaves better on that target.

In-context probing as proxy attribution

The closest connection between this line and gradient-based attribution is the in-context probing literature. Jiao et al. show that in-context probing can act as a fast proxy for gradient-based data attribution when task type and content similarity between candidate data and the target task are sufficiently aligned [21]. Their analysis is especially relevant to this thesis because it focuses on ranking agreement rather than on replacing one method family with another in the abstract. The result is nuanced: in-context probing is not a universal substitute for gradient-based attribution, but it can track gradient-based rankings in appropriate domains and offers a computationally lightweight point of comparison. That is precisely the kind of relationship a comparison framework should be able to surface.

2.4 Attribution Beyond Standard Text Benchmarks

2.4.1 Multimodal foundation models

Vision-language and video models

Recent open multimodal models such as LLaVA and Qwen3-VL demonstrate that instruction-following and reasoning now extend beyond text into images and video [29, 4]. These models accept heterogeneous inputs, combine visual encoders with language backbones, and increasingly support long-context video understanding. From the perspective of attribution, this change is significant. The object being attributed is no longer always a short text sequence, and the

returned training examples may themselves be media-rich objects such as clips or interleaved multimodal records.

Implications for attribution workflows

Many attribution systems still assume a comparatively homogeneous interface: text input, text output, and ranked text examples. Multimodal models complicate every stage of the pipeline. Query construction may involve frame sampling or multimodal prompting. Reference artifacts may target only a parameter-efficient subset of the model. Retrieved examples may need to be rendered as media rather than read as text. This is one of the reasons why an attribution framework should be discussed not only in algorithmic terms, but also in workflow terms. A method that is conceptually valid but unusable for multimodal inspection remains of limited practical value.

2.4.2 Healthcare and EHR foundation models

From sequential risk models to EHR foundation models

Healthcare provides another setting where standard text assumptions break down. Earlier models such as RETAIN and StageNet showed that temporal structure, visit order, and event history are essential in longitudinal clinical prediction [9, 11]. More recent work has moved toward foundation-model-scale pretraining on structured or semi-structured patient timelines. NYUTron trains health-system-scale language models for diverse clinical prediction tasks [20]. Foresight models patient timelines with a generative transformer over biomedical concepts and free-text-derived coded events [25]. MOTOR adopts a time-to-event foundation-model perspective for structured records [36]. EHRSHOT adds a public benchmark and few-shot protocol for structured EHR foundation models together with a released pretrained model [40]. Taken together, these papers show that the foundation-model paradigm has already moved into longitudinal healthcare records.

Benchmarks and cancer risk prediction

Domain-specific prediction tasks then build on this broader EHR foundation-model landscape. Placido et al. show that disease trajectories in health records can be used to predict pancreatic cancer risk [32]. More recently, CATCH-FM studies large-scale EHR pretraining for cancer prescreening and shows that code-based healthcare foundation models can support strong cancer risk prediction and cross-system transfer [37]. These papers are not primarily about training data attribution, but they matter for the present thesis because they motivate why an attribution framework should support structured, coded, and domain-specific settings rather than only public text tasks. In healthcare, the usefulness of attribution depends heavily on whether retrieved examples preserve chronology, coding structure, and patient-history context in a readable form.

2.5 From Methods to Workflows

2.5.1 Interactive attribution tools

Compared with the large literature on attribution algorithms, the literature on interactive attribution tools is still small. LLM Attributor is a notable example: it provides interactive visualizations for training data attribution of language-model generations and lets users compare how attribution changes across selected phrases or edited outputs [27]. Chang et al. also release a web-based visualization tool for exploring influential examples at pretraining scale [7]. These works show growing recognition that attribution is most useful when it can be explored, compared, and inspected rather than only reported as a ranked list in a static table.

At the same time, many existing tools remain method-specific, model-specific, or text-only. They are valuable demonstrations, but they do not yet amount to a general workflow layer across attribution methods and heterogeneous data settings. This is where the present thesis is positioned. The literature already offers rich algorithmic ideas for gradient-based attribution, efficient approximations, proxy methods, provenance evaluation, data selection, and domain-specific foundation models. What is rarer is a framework that brings several of these ideas into one interactive environment and lets users compare them across public text tasks, multimodal video, and a medical record setting. Therefore, what remains underdeveloped is the workflow layer that connects these methods and settings in a shared interactive system. This thesis takes that layer as its focus.

Chapter 3

Design and Setup

3.1 Framework Formulation

Training data attribution is often introduced as a method for assigning a relevance or influence score to each training example with respect to a model output. In practice, however, attribution is most useful when it can be queried, compared, and inspected repeatedly under a common interaction model. The framework developed in this thesis is designed around that observation. Rather than treating attribution as an isolated offline computation, it exposes attribution as an interactive workflow in which a user selects a dataset, a model, and a method, submits one or more queries, and receives ranked training examples together with comparison-oriented summaries.

For a selected dataset–model pair, let the training set be $\mathcal{D}_{\text{train}} = \{z_i\}_{i=1}^{N}$. Given a query q and attribution method m, the framework assigns each training example a score

$$s_i^{(m)}(q) = \psi_m(\phi_m(q), \rho_m(z_i)), \qquad i \in \{1, \ldots, N\}, \tag{3.1}$$

where $\phi_m(q)$ denotes the representation computed from the current query, $\rho_m(z_i)$ denotes the stored reference representation for training example z_i, and ψ_m is the method-specific matching rule. The ranking returned to the user is the ordering induced by $\{s_i^{(m)}(q)\}_{i=1}^{N}$. This formulation is broad enough to cover lexical retrieval, gradient similarity, influence-style approximations, and low-rank methods while keeping the system interface stable.

The same formulation also accommodates the two main entry points of the framework. In live-query mode, q is constructed directly from user-provided input. In selected-point mode, q is derived from one or more validation examples that already exist in the dataset browser. Both routes lead to the same scoring pipeline: the current query is normalized into a method-compatible form, a score vector over the training set is computed, and the highest-ranked training examples are displayed for inspection.

A distinctive feature of the framework is support for multiple input queries within the same request. If a user provides a set of queries $Q = \{q_1, \ldots, q_M\}$, the system computes one score vector per query and averages them elementwise,

$$\bar{s}_i^{(m)}(Q) = \frac{1}{M} \sum_{j=1}^{M} s_i^{(m)}(q_j), \tag{3.2}$$

then ranks the training set by $\bar{s}_i^{(m)}(Q)$. The averaging takes place over scores rather than over rank positions. This detail matters because it makes multi-query attribution a genuine analytical operation: the resulting ranking reflects consensus at the level of the scoring rule itself.

The framework is equally centered on comparison. When two methods m_a and m_b are applied to the same query, the system compares their resulting rankings through agreement measures that emphasize both global rank structure and overlap among the most highly ranked training examples. Given the top-K sets $\mathrm{TopK}_{m_a}(q)$ and $\mathrm{TopK}_{m_b}(q)$, the framework reports

$$\mathrm{Jaccard@}K = \frac{\left|\mathrm{TopK}_{m_a}(q) \cap \mathrm{TopK}_{m_b}(q)\right|}{\left|\mathrm{TopK}_{m_a}(q) \cup \mathrm{TopK}_{m_b}(q)\right|}, \tag{3.3}$$

and it computes Spearman correlation over rank positions on the common training-point index set. These comparison operators are built into the framework because attribution is often most informative when different methods are applied to the same example under the same conditions.

A final principle underlying the formulation is the separation between the current query and the training corpus. Modern attribution methods, especially gradient-based ones, are too expensive to rebuild the training side from scratch for every interaction. The framework therefore computes the representation of the current query live, while storing training-side reference representations offline. This hybrid organization keeps the interface interactive while preserving the structure required by gradient-based and low-rank attribution methods.

3.2 System Architecture and Execution

The framework is implemented as three cooperating services: a browser frontend, an API backend, and a GPU attribution backend. This separation keeps interaction, orchestration, and model-specific scoring distinct, while preserving a single user-facing system. Figure 3.1 summarizes the main components. The figure is intentionally simple: the essential design is that the frontend collects user requests and displays results, the API backend coordinates requests and post-processing, and the GPU backend performs model-side attribution computation against precomputed reference artifacts.

The frontend is the interactive surface of the framework. It presents dataset, model, and method selectors; supports both live-query input and selected validation points; and renders ranked training examples together with comparison results, histograms, and modality-aware media. The frontend is intentionally thin. It does not load model weights, training-side reference stores, or method-specific attribution logic. Its role is to collect structured requests and display returned outputs in a readable way.

The API backend is the coordinating layer. It exposes the catalog of available datasets, models, and methods; handles validation-point pagination; transforms frontend inputs into normalized internal requests; sends scoring requests to the GPU backend; aggregates score vectors in multi-query mode; computes overlap and correlation summaries; and formats the results returned to the frontend. This layer is central to the framework design because it keeps the user interaction model stable even when the underlying models and attribution methods differ substantially. Without this middle layer, each modality and each method would require separate handling in the frontend.

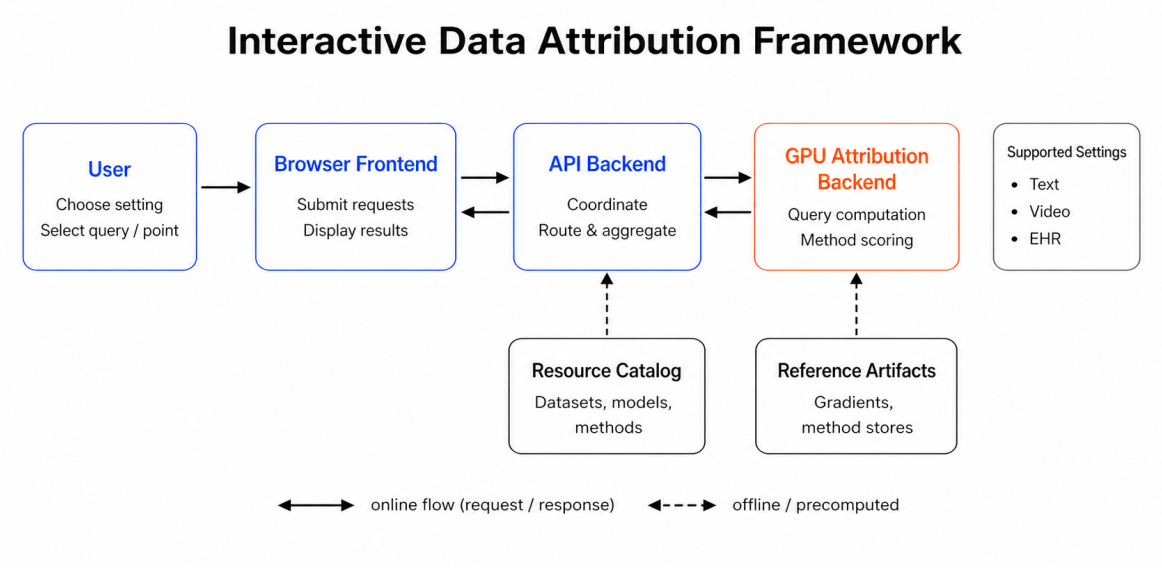


Figure 3.1: An illustration of the interactive data attribution framework.

The GPU attribution backend performs model-specific scoring. Given a normalized request, it loads the selected model family, constructs the representation for the current query, retrieves the appropriate training-side reference store, computes a full score vector over the training set, and returns that vector to the API backend. The computation required at this stage depends strongly on the selected method. BM25 requires only text statistics over the reference collection. Gradient-based methods require a live forward-backward pass for the current query together with stored training-side gradient representations. Low-rank methods rely on corresponding low-rank reference stores. The same execution contract nevertheless holds across these cases: the backend returns one scalar score per training example.

The runtime path through the system is therefore consistent. A request begins in the frontend with a dataset, a model, a method, and either a live query or a selected validation example. The API backend normalizes that request and dispatches it to the GPU attribution backend. The GPU backend computes the score vector over the training set and returns it. The API backend then derives ranked outputs and, when requested, comparison summaries such as top-K overlap and Spearman agreement. Finally, the frontend renders the returned ranking and any associated metadata or media.

This organization supports the main operational goal of the framework: heavy model computation remains server-side, while the interaction model remains lightweight and uniform. The browser does not need access to models or attribution artifacts. The API backend does not need to duplicate method-specific scoring logic. The GPU backend does not need to carry presentation logic. Each service is responsible for a well-defined slice of the framework.

3.3 Workflow Layer

The framework is built around a small set of stable workflows that remain available across settings. The first is live-query attribution. In this mode, a user enters a query directly, typically as text, and requests ranked training examples under a selected method. In text settings, the query can be a full question–answer pair or another task-specific input string. In the video setting, the query can be a selected validation example or a formatted multimodal record that contains both textual content and a linked clip. Regardless of modality, the system converts the user-facing input into the model-specific representation required by the selected method.

The second workflow begins from stored validation examples. Many attribution analyses start not from a newly composed query, but from an existing example whose behavior deserves inspection. The framework therefore allows the user to browse validation data, select one or more examples, and use those examples as the source query. From the backend perspective, this workflow is intentionally aligned with live-query attribution: the selected example is reformulated into the same normalized request structure, then processed by the same scoring pipeline. The advantage is methodological consistency. Live inputs and stored validation examples can be inspected under the same scoring rules and comparison tools.

The third workflow is multi-query aggregation. Both live-query mode and selected-point mode support multiple simultaneous inputs. This feature is valuable when a user wants to characterize a concept or behavior through several related examples rather than a single point. Because the framework averages score vectors as in Equation 3.2, the resulting ranking can differ from any single-query ranking. The system therefore exposes a natural way to move from pointwise attribution to group-wise attribution without introducing a separate method or interface.

The fourth workflow is direct comparison. Once a ranking has been produced for one method, the same query can be passed to a second method and the two results can be compared under the shared definitions of rank correlation and top-K overlap. The same comparison interface can also be used to compare models under a fixed dataset and input. In practice, comparison mode is one of the most important parts of the system because it turns attribution from a one-method report into a comparative analysis environment. Agreement across configurations suggests a stable attribution signal, while disagreement reveals that different scoring rules or models attend to different structural properties of the data.

A few implementation details support the usability of these workflows. Validation points are paginated so that the user can browse large sets without loading the full validation corpus at once. Result formatting is modality-aware: text results are displayed as question–answer records, video results render associated media, and medical records are displayed through field-aware formatting rather than raw internal representations. These design choices are part of what makes attribution inspectable as a workflow across heterogeneous datasets.

3.4 Attribution Methods in the Framework

Although the framework supports several methods, they can be described through a common representation-matching view. Each method computes or retrieves a query representation, compares it with a stored representation for every training example, and returns a ranking over the

training set. The main differences lie in what is represented and how the match is scored. In the framework, the central contrast is between a lexical retrieval method, BM25, and a family of gradient-based or low-rank methods rooted in influence-style reasoning and scalable representation matching [23, 31, 26, 41, 8, 33].

3.4.1 BM25

BM25 serves as the framework's main non-gradient contrast method. It is important in this role because it provides a ranking based on lexical evidence rather than model-internal representations. For a query q and training example z_i, BM25 scores the overlap between query terms and the text of z_i using inverse-document-frequency weighting and document-length normalization [34]. In standard notation, the score can be written as

$$s_i^{\mathrm{BM25}}(q) = \sum_{t \in q} \mathrm{idf}(t) \cdot \frac{\mathrm{tf}(t, z_i)(k_1 + 1)}{\mathrm{tf}(t, z_i) + k_1 \left(1 - b + b|z_i|/\mathrm{avgdl}\right)}. \tag{3.4}$$

Here $\mathrm{tf}(t, z_i)$ is the frequency of token t in the training example, $|z_i|$ is the length of the training example, and avgdl is the average training-example length in the corpus.

Within the framework, BM25 plays two roles. Methodologically, it anchors the comparison between lexical retrieval and model-based attribution. Operationally, it provides a lightweight scoring path that requires no model weights and no precomputed gradient artifacts. This makes it an especially useful contrast method in interactive settings, because its rankings can be produced quickly and interpreted directly in terms of textual similarity.

3.4.2 Gradient similarity: GradDot and GradSim

The core gradient methods in the framework are GradDot and GradSim. Both begin from the same representation: a gradient-based feature vector for the current query and a stored gradient-based feature vector for each training example. Let

$$g(q) = P\nabla_\theta \ell(q; \theta), \qquad g_i = P\nabla_\theta \ell(z_i; \theta), \tag{3.5}$$

where $\ell(\cdot; \theta)$ is the task loss under model parameters θ and P denotes the projection or representation operator used to store the gradients efficiently. The training-side vectors g_i are constructed offline and cached as reference artifacts, while $g(q)$ is computed for the current query when the request arrives.

GradDot uses the inner product

$$s_i^{\mathrm{GradDot}}(q) = g(q)^\top g_i, \tag{3.6}$$

which measures direct alignment between the current query and training example in gradient space. This scoring rule is closely related to inner-product influence approximations that drop explicit Hessian terms while preserving gradient-level alignment [33]. It is also the framework counterpart of the InflIP score used in prior attribution and data-valuation work, where training examples are ranked by the inner product between target and training gradients [21, 43].

GradSim uses cosine similarity,

$$s_i^{\mathrm{GradSim}}(q) = \frac{g(q)^\top g_i}{\|g(q)\|_2 \|g_i\|_2}, \tag{3.7}$$

which removes scale effects and emphasizes angular agreement. In practice, the two methods offer a clean contrast between magnitude-sensitive and magnitude-invariant gradient matching while preserving the same reference-store design. Their inclusion in the framework exposes a familiar and interpretable gradient-attribution surface.

3.4.3 DataInf

DataInf is an efficient approximation to influence-based data attribution for large models, with particular advantages in parameter-efficient fine-tuning settings such as LoRA [23, 26]. At a high level, it seeks to estimate how much a training example contributes to the loss of a target query or prediction. In an influence-style formulation, the attribution score between a training example z and a target example q can be written as

$$I(z, q) \approx - g(q)^\top (G + \lambda I)^{-1} g(z), \tag{3.8}$$

where $g(\cdot)$ denotes the gradient of the loss and G summarizes gradient second-moment information over the training set. The main contribution of DataInf is to derive a closed-form approximation to this quantity that is practical for modern large-model settings without requiring explicit Hessian inversion [26].

Within the framework, DataInf provides an influence-oriented view of training data relevance. This is conceptually different from direct gradient-similarity methods such as GradDot and GradSim. Whereas similarity-based methods emphasize gradient alignment, DataInf is intended to approximate the contribution of each training example to the behavior of the current query. Including DataInf therefore allows the comparison workflow to place direct similarity and influence-oriented scoring side by side.

3.4.4 LESS

LESS, short for *Low-rank gradiEnt Similarity Search*, was introduced as a method for targeted instruction tuning [41]. Its goal is to identify the subset of instruction data that is most useful for a desired downstream capability, rather than treating all candidate training data as equally relevant. The method begins from an influence-style view of training dynamics, but adapts that view to the practical setting of instruction tuning, where models are typically optimized with Adam and trained on variable-length sequences. In the original formulation, this leads to an optimizer-aware scoring rule defined over a sequence of warmup checkpoints:

$$s_{\mathrm{LESS}}(z, D_{\mathrm{val}}^{(j)}) = \sum_{i=1}^{N} \bar{\eta}_i \, \cos\Big(\bar{g}_i(D_{\mathrm{val}}^{(j)}), \widetilde{\Gamma}_i(z) \Big) \,, \tag{3.9}$$

where $\bar{g}_i(D_{\text{val}}^{(j)})$ is the aggregated gradient feature of a validation subtask $D_{\text{val}}^{(j)}$ at checkpoint i, $\widetilde{\Gamma}_i(z)$ is the corresponding low-dimensional feature of a candidate training example, and $\bar{\eta}_i$ reflects the epoch-wise learning-rate contribution [41]. In practice, LESS scores each candidate example against one or more target subtasks and selects the examples with the highest scores.

LESS is a useful method to include in a unified attribution framework because it broadens comparison beyond direct gradient similarity. It is still gradient-based, but its interpretation is capability-oriented: it asks which training examples support the behavior represented by a target validation set or query. This makes it a natural bridge between attribution and targeted data selection, and it shows how the same interactive workflow can support more than one notion of training-example relevance.

3.4.5 LoGra

LoGra is a low-rank gradient projection method introduced to scale influence-based data valuation to large language models [8]. The starting point is the observation that classical influence methods become difficult to apply at LLM scale because both the storage and manipulation of per-example gradients are prohibitively expensive. LoGra addresses this bottleneck by exploiting the structure of backpropagation in matrix-valued layers. For a linear layer with input activation h and backpropagated signal δ, the method projects the gradient into a compact space through separate low-rank projections of the forward and backward activations:

$$\phi(\nabla_W \ell) = (P_{\text{out}}^\top \delta) \otimes (P_{\text{in}}^\top h), \tag{3.10}$$

where P_{out} and P_{in} are low-dimensional projection matrices and $\otimes$ denotes the Kronecker product [8]. This formulation preserves the influence-oriented structure of the gradient while avoiding the cost of materializing and projecting the full high-dimensional gradient directly.

Within the method suite, LoGra represents the low-rank end of gradient-based attribution. It differs from direct gradient-similarity methods because its representation is shaped by a principled low-rank projection strategy, and it differs from closed-form influence approximations such as DataInf because it is built around scalable projected gradients rather than a separate approximation of the influence computation. Its inclusion broadens the framework from direct gradient matching to low-rank data valuation.

The framework also includes simple controls such as Random and Flat. These controls are useful for checking result-page behavior and sanity-testing ranking displays, but the scientific core of the method layer lies in BM25 and the model-based family described above.

3.5 Supported Settings and Data Processing

The framework currently spans three kinds of settings: public text datasets, a multimodal driving-video corpus, and a medical record setting built around structured EHR data. These settings differ markedly in input format and model interface, yet they all participate in the same ranking and comparison workflows.

3.5.1 Public text setting

The public text setting combines three established reasoning-oriented datasets with Pythia-family models [5]. The first is MathQA, which formulates math word-problem solving through operation-based representations and provides a natural source of structured reasoning examples [3]. The second is MedQA, an open-domain medical question answering dataset built from professional medical exam questions [22]. The third is PIQA, a dataset centered on physical commonsense reasoning [6]. Together, these datasets cover numerical reasoning, domain-specific factual reasoning, and physical commonsense, while still admitting a common text interface inside the framework.

In the system, each text example is normalized into a point representation that can be displayed and queried consistently. A typical record includes a prompt-like field and an answer or target field, which makes the text setting especially well suited to method comparison. Because all three datasets share the same broad question–answer style interface, the framework can expose live query input, selected validation points, multi-query averaging, and comparison mode without changing the interaction logic across datasets.

The underlying text models are Pythia-410M and Pythia-1B [5]. These models are well suited to a framework-oriented study because they are open, research-friendly, and available at multiple scales within the same model family. Their shared architecture also simplifies the offline construction of reference stores and makes method comparisons easier to interpret.

3.5.2 Driving-video setting

The driving-video setting extends the framework to multimodal attribution. It uses a local corpus of short driving clips organized through train and validation manifests, with sources that include clips derived from JAAD and A2D2 [24, 12]. Each record couples a video path with textual fields that describe or contextualize the clip. This combination is important for the thesis because it allows the same framework to move beyond text-only inputs while retaining a query–ranking workflow that is recognizable from the public text setting.

The underlying model is Qwen3-VL, a multimodal foundation model capable of processing images, video frames, and text within a shared instruction-following interface [4]. For a selected video example, the framework samples a fixed set of frames from the clip, rescales them for processing, combines them with the associated text fields, and feeds the resulting multimodal record through the Qwen3-VL processor. This produces the query representation required by the selected attribution method. The training side is handled through offline reference stores built for the same model and dataset.

The driving-video setting demonstrates two important properties of the framework. First, the method layer does not depend on text-only assumptions. Second, the workflow layer remains stable even when the underlying input becomes multimodal. A user can still select a validation point, compare methods, inspect top-ranked results, and browse returned examples through the same interactive structure used in the text setting.

3.5.3 Medical record setting

The medical setting extends the framework to structured longitudinal EHR data. Its model pathway follows the healthcare foundation-model design used in CATCH-FM, where patient history is represented directly in medical code space rather than converted into free text [37]. This choice is well aligned with longitudinal EHR data such as NHIRD, which provides large-scale de-identified medical records with standardized coding over long patient histories [16].

A patient record can be written as a sequence

$$x = [c_{\text{age}}, c_{\text{gender}}, v_1, t_1, v_2, \ldots, v_n, [\text{EOS}]], \qquad v_j = [c_j^1, c_j^2, \ldots, c_j^{m_j}], \tag{3.11}$$

where demographic tokens are followed by chronologically ordered visits, each visit contains one or more medical-event codes, and t_j denotes the time token between visits. This representation is particularly suitable for attribution because it preserves the structure of the medical history while remaining compatible with sequence models.

Within the framework, the medical setting is exposed through the CatchFM-160M model and a structured record interface that shows key attributes of patient records. The same system mechanisms used elsewhere still apply: the user selects the setting from the catalog, chooses an attribution method, issues a live query or selects a stored example, and inspects the resulting ranked training records. The framework can incorporate domain-specific structured data without abandoning its common workflow contract.

3.6 Implementation Choices and Extensibility

The framework is designed to support multiple datasets and models without rewriting core logic for each new setting. This is achieved through a resource catalog that specifies dataset entries, model entries, labels, split paths, point files, media roots, and method-specific reference stores. The catalog decouples the system inventory from the repository layout, allowing external and private assets to remain in their own storage locations while still appearing as ordinary options in the framework.

This catalog-driven organization works together with a dataset-formatting layer. Text datasets can be displayed directly as question–answer records. Video datasets require media handling through a backend endpoint and a video-aware renderer. Medical datasets require structured field formatting so that patient records remain readable. Because configuration and formatting are separated, adding a new setting has a clear and localized path: define the resource entry, prepare the required reference artifacts, and implement the formatting rules for the new record type.

The same principle governs the treatment of training-side artifacts. Gradient-based and low-rank methods rely on precomputed reference stores, which are constructed offline and then reused during interactive sessions. The current query is always processed live, but the training side is stored in method-specific form. This design makes interactive attribution feasible for large models while preserving the distinction between methods. Generic gradient matching, DataInf, LESS, and LoGra each depend on their own artifact family, yet all are exposed through the same workflow interface.

The implementation choices also clarify why the framework remains practical for interactive use. The browser remains lightweight because it communicates only with the API backend and never loads model weights or reference stores. The API backend remains method-agnostic because model-specific processing is delegated to the GPU backend. The GPU backend remains reusable because every method ultimately reduces to computing a query representation, reading a stored reference representation, and applying a scoring rule. The result is a framework whose operational surface is stable even as the underlying datasets, models, and attribution methods vary.

Taken together, the system maintains a common operational contract—dataset, model, method, query, ranking—across heterogeneous inputs and attribution methods. Architecture, catalog configuration, offline artifact construction, and modality-aware formatting all serve that contract. This is what turns training data attribution from a collection of method-specific scripts into a coherent framework for interactive analysis.

Chapter 4

Framework Demonstrations and Results

4.1 Overview

This chapter presents the framework in use. The goal is to show how attribution becomes an interactive process: a user selects a setting, issues a query or chooses an existing validation example, inspects ranked training examples, compares attribution configurations, and extends the same workflow across text, video, and medical records. The results therefore combine visual demonstrations, qualitative observations, approximate runtime measurements, and preliminary user feedback.

The chapter begins with the basic single-method workflow and the result visualization that accompanies it. This establishes the core interaction pattern: configuration, query submission, score-distribution inspection, and top-K / bottom-K example review. It then presents multi-query averaging as a second interaction pattern, where several related inputs are combined through elementwise score-vector averaging before ranking. The video and medical settings are shown next to demonstrate that the same workflow can be preserved while the presentation layer adapts to media clips and structured patient records.

The chapter then turns to comparison mode. This is the most important analytical workflow in the system because it places two attribution configurations under the same query and exposes both full-ranking agreement and top-region overlap. The comparison view makes it possible to inspect not only whether two methods agree numerically, but also which top-ranked examples are shared and which are method-specific.

The final subsections further extend the results. A qualitative insights section summarizes what one might learn about the models, methods, and data settings by using the framework. A runtime section gives approximate time and precomputation costs under a single-GPU setup, clarifying the practicality of our system and the distinction between offline reference construction and online interaction. Finally, a preliminary user feedback study conducted with several users reports how the framework supports users in a typical data attribution task. Together, these results support the main claim of the thesis: training data attribution becomes more useful when it is organized as a workflow for querying, comparing, and interpreting retrieved training examples.

4.2 General Workflow and Attribution Result Visualization

The framework is easiest to understand through its simplest use case. A user first chooses a dataset, a model, and an attribution method, then submits either a live query or a selected validation example, and finally receives a ranked list of training examples for inspection. The resulting page combines two complementary views. The first is local: a ranked panel of the most highly scored training examples, together with their associated metadata and content. The second is global: a score-distribution graphic constructed from the full score vector over the training set.

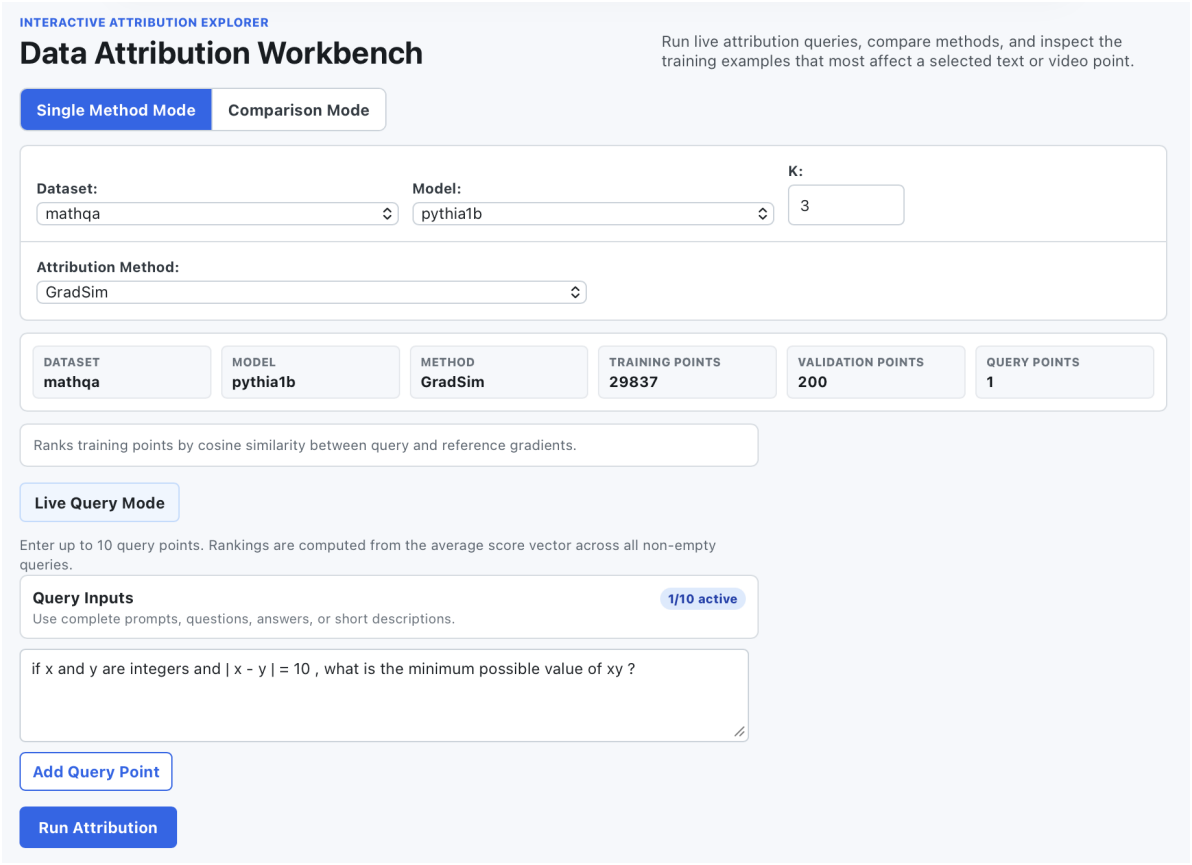
Figure 4.1 shows the selection-stage workflow in a representative public text setting. The upper panel illustrates live-query mode. The user selects a dataset, an underlying model, an attribution method, and a return size K, then enters a query directly into the interface. In the example shown here, the framework is configured with MathQA, Pythia-1B, and GradSim.

The page makes the main analytical choices visible before attribution is executed. The dataset, model, and method selectors determine the attribution setting explicitly, while the summary cards below them provide a compact view of the current scale of analysis, including the number of training points and validation points. The lower panel of Figure 4.1 shows the alternative validation-point workflow. Instead of writing a new query, the user may browse saved validation examples and choose one or more points of interest. This mode is especially useful when the goal is to inspect how the model relates a known held-out example to its training data. The paginated browser is an important part of that design. Rather than loading every validation point at once, the framework exposes them in manageable pages, allowing the user to inspect examples incrementally and select relevant cases without overwhelming the interface. The live-query and validation-point modes therefore differ only in how the input is specified. Once an input is submitted, both proceed through the same backend scoring pipeline.

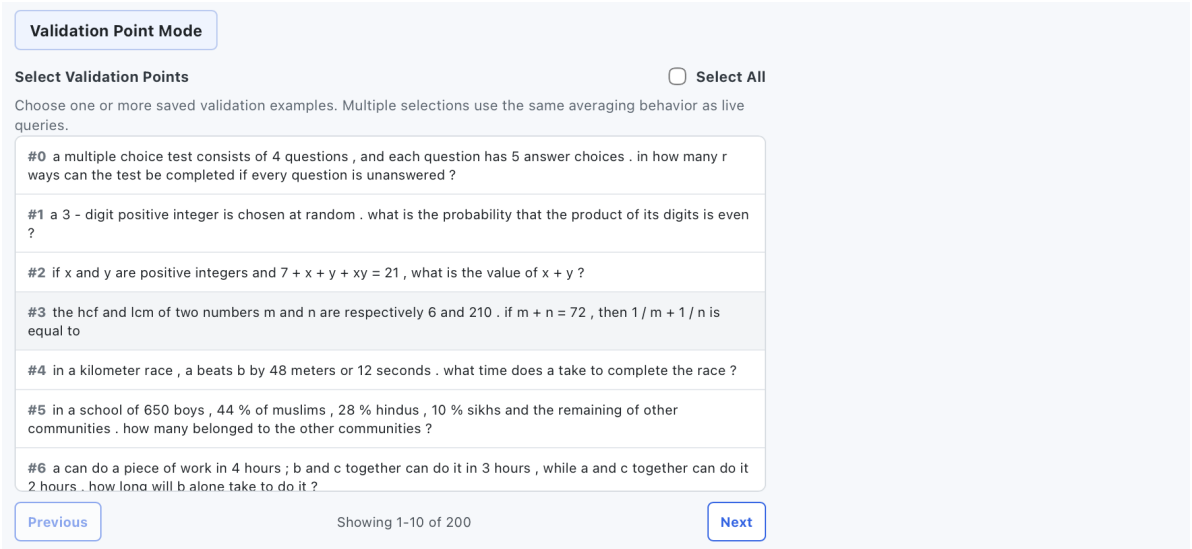
After submission, the framework returns a result page that is meant to support inspection rather than only score reporting. Figure 4.2 shows this page for the same single-method text setting. The top of the page records the submitted query together with the active dataset, model, and attribution method, so the context of the current run remains visible while the returned examples are inspected. This makes the result page self-contained: a reader can interpret the ranking without needing to return to the configuration screen.

A central element of the result page is the score-distribution graph. This graph is built from the full set of attribution scores assigned to the training set for the current run, not only from the small number of examples that are ultimately displayed in the ranking panels. In practice, this matters because a ranked list alone does not reveal whether the highest-scoring examples are sharply separated from the rest of the training set or whether they come from a much flatter score landscape. The graph therefore provides an overview of how concentrated or diffuse the current attribution pattern is. A narrow high-scoring tail suggests that only a small subset of training points strongly support the query, while a broader distribution suggests a weaker separation between returned examples and the rest of the corpus. This global view gives necessary context to the local ranking shown below it.

The lower half of the page presents two ranked panels: Top-K and Bottom-K. The top panel contains the training examples that the chosen method considers most supportive for the submitted query. The bottom panel shows the least supportive examples under the same scoring rule. Presenting both ends of the ranking is useful because it allows the user to inspect not only



(a) Live-query configuration page.



(b) Validation-point selection page.

Figure 4.1: Selection-stage workflow in single-method mode.

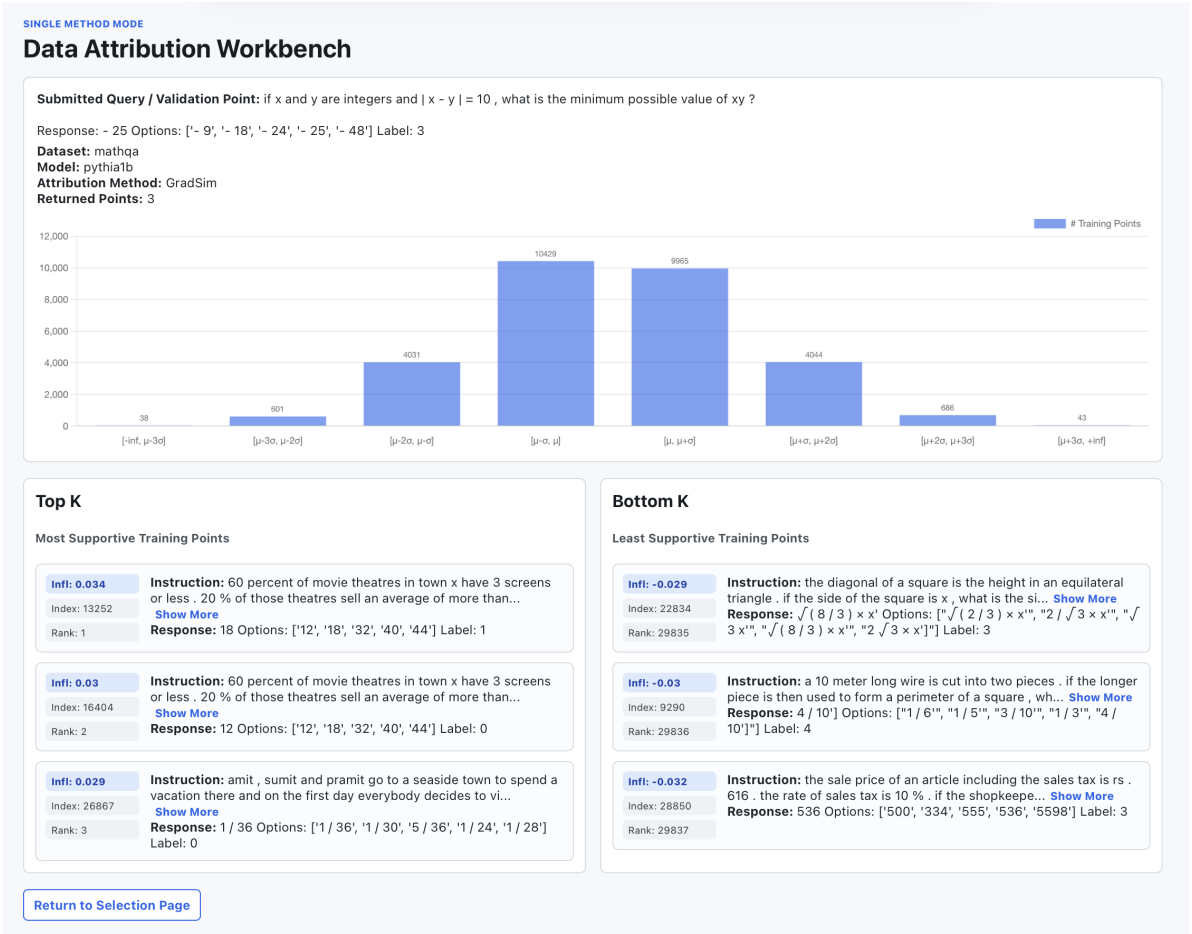
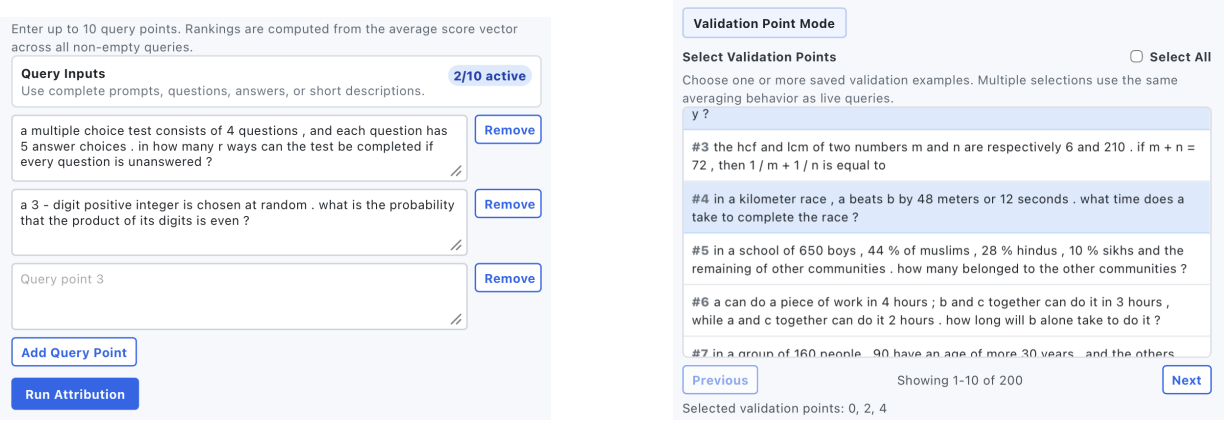


Figure 4.2: Single-method attribution result page.

what the framework retrieves as highly related, but also what it pushes furthest away. The user can therefore move directly from the abstract notion of attribution to concrete question–answer records. The result cards also make the ranking itself legible. Each card includes the score, the example index, and its rank position, together with the formatted example content. This small amount of metadata is enough to tie together the distribution graph and the returned records.

4.3 Multi-Query Averaging

In addition to single-input attribution, the framework supports multi-query analysis. This feature allows the user to submit more than one input within the same run and obtain a single aggregated ranking over the training set. Figure 4.3 shows the two ways in which this functionality appears in the interface. In live-query mode, the user may enter multiple free-form inputs under the same configuration. Each query occupies its own input field, and the interface keeps track of how many query slots are currently active. In validation-point mode, the same functionality is expressed through multi-selection of saved validation examples, and the system displays the indices of the current selected points for easier tracking.



(a) Live-query multi-input configuration.

(b) Validation-point multi-input configuration.

Figure 4.3: Multi-query configuration in the two input modes.

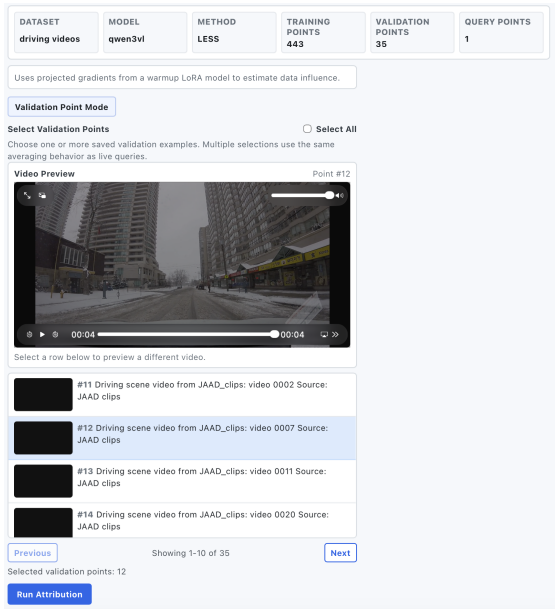
The backend treats these inputs through a shared aggregation rule. Each query produces its own attribution score vector over the training set, and the framework averages these score vectors elementwise before constructing the final ranking. As a result, the returned examples reflect the joint behavior of the selected inputs rather than the output of any one of them in isolation. This makes multi-query mode a natural extension of the single-query workflow introduced earlier: the same result page is used, but the ranking now summarizes a set of inputs instead of a single one.

From the user's perspective, multi-query support broadens the role of the framework. The system is no longer limited to analyzing one example at a time, but can also surface training examples that are repeatedly relevant across a small collection of related points. This makes the interface more flexible while preserving the same overall interaction pattern of configuration, submission, and inspection.

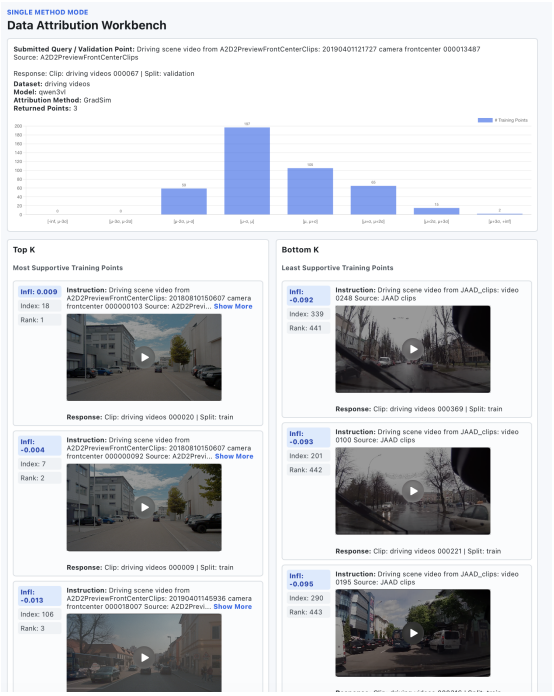
4.4 Video and Medical Settings

The same workflow extends beyond the text setting to additional data modalities. In particular, the framework supports a multimodal driving-video setting and a medical setting based on longitudinal EHR sequences. These two setups show that the system can preserve the same attribution workflow while adapting the display and input handling to non-text records. The interaction pattern remains recognizable across settings: the user selects a configuration, chooses or submits an input, runs attribution, and inspects returned training examples through a modality-aware result view.

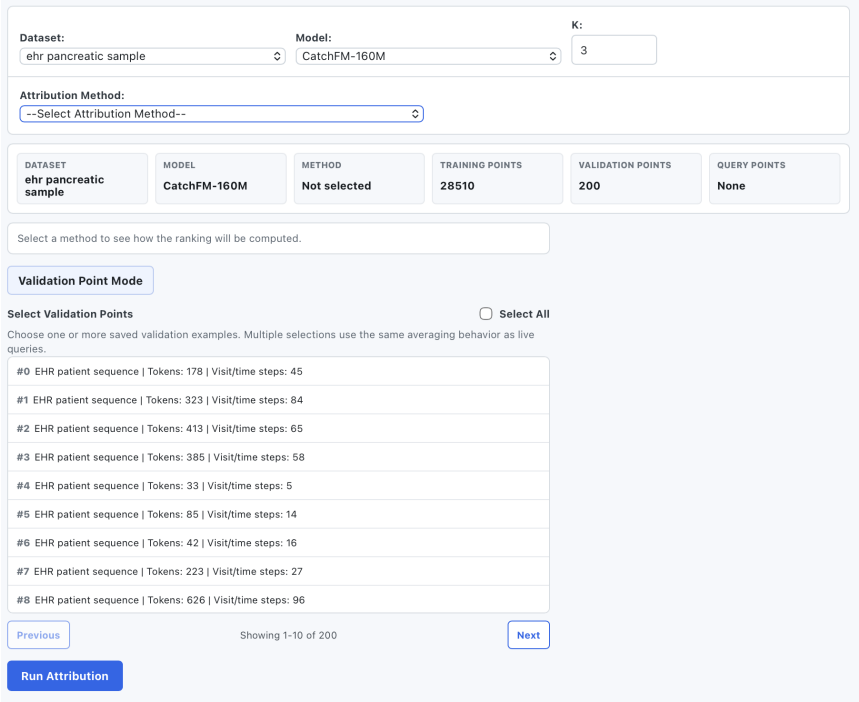
Figure 4.4 shows representative views of these two settings. The video workflow uses the driving_videos dataset together with Qwen3-VL, and presents the validation-point mode through a media-aware interface. Rather than listing only text descriptions, the configuration page includes a preview panel for the currently selected video and displays the surrounding validation entries in a selection list. This design supports multimodal inspection by allowing the user to identify the current query point visually before running attribution, rather than relying only on clip identifiers or metadata.



(a) Video configuration page.



(b) Video result page.



(c) Medical configuration page.

Figure 4.4: Video and medical workflow views.

The corresponding result view for the video setup follows the same overall logic introduced earlier. The page records the submitted point and the active dataset, model, and method, then summarizes the score distribution and displays the top-ranked and bottom-ranked training examples. The main difference is that the returned examples are now rendered as video items rather than as text records. This makes the ranking easier to interpret in a multimodal setting. A user can inspect not only which clips are retrieved, but also what visual content those clips contain.

The medical setup illustrates a different kind of extension. Here the framework is configured for the a sample EHR for pancreatic cancer dataset and the CatchFM-160M model. The configuration page presents validation examples as patient sequences, together with compact structural information such as the number of tokens and visit or time steps. This representation is helpful when individual points are not naturally viewed as short text prompts or media clips. The interface therefore surfaces the information needed to browse and select patient trajectories in a readable form without exposing the full record immediately on the configuration page.

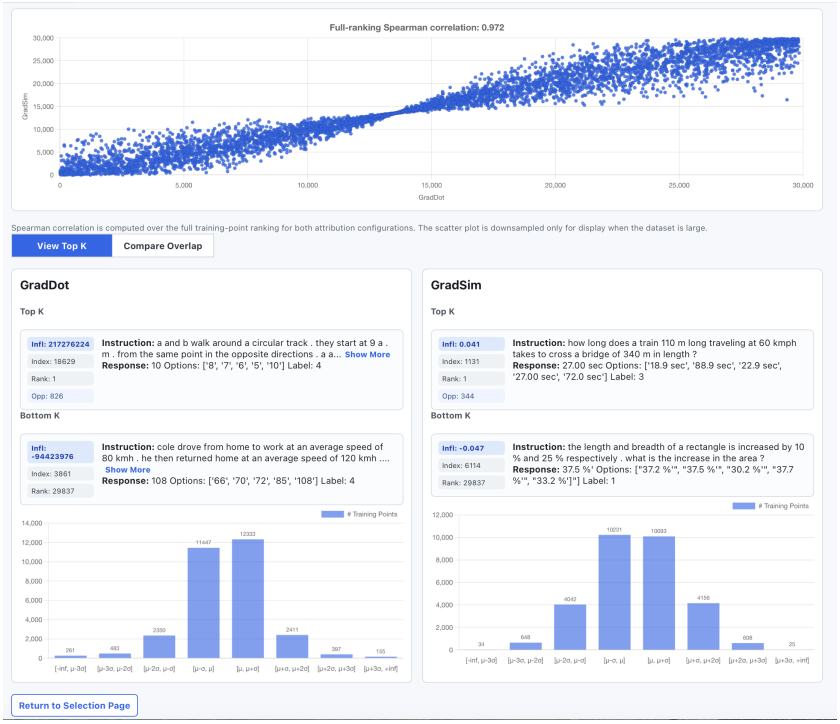
Taken together, these views illustrate the broader scope of the framework. The system preserves a common attribution workflow while adapting the presentation layer to different modalities, which is one of its practical strengths.

4.5 Comparison Mode

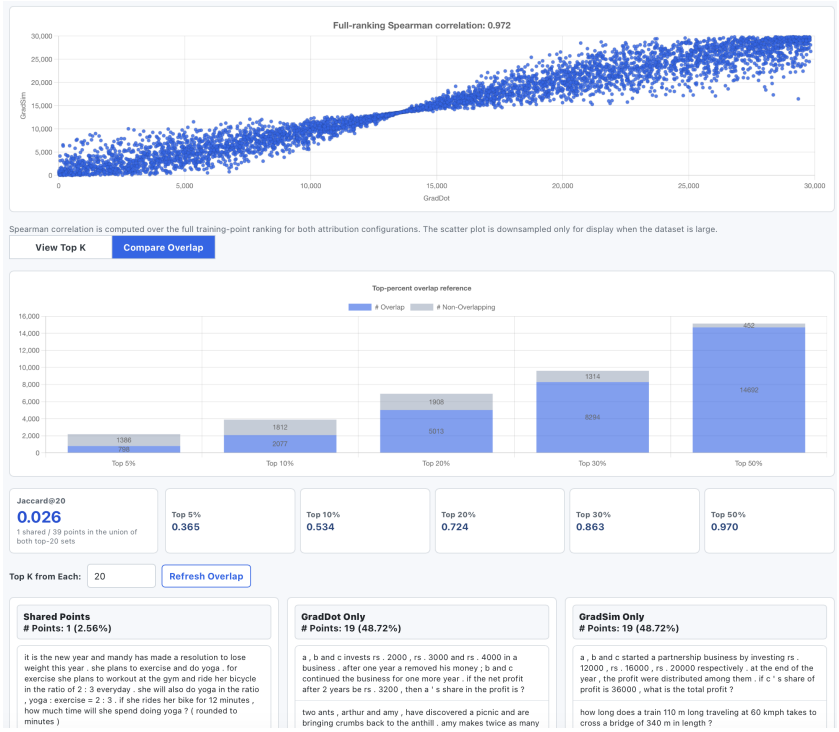
A single attribution run is useful for retrieving training examples under one chosen configuration, but many of the most informative questions arise only when two rankings are examined together. In practice, a user often wants to know whether two attribution methods identify similar evidence, whether a change of attribution configuration changes the retrieved examples, or whether agreement holds only globally or also among the highest-ranked points. The comparison mode is designed for this purpose. It keeps the dataset and the submitted query fixed, executes two attribution configurations under the same setting, and presents the relationship between the resulting rankings in a form that can be inspected directly.

The most common use of this interface is method comparison. In that setting, the framework compares two attribution methods on the same model and the same query or validation point. This makes the source of variation easy to interpret: differences in the returned ranking come from the attribution rule rather than from the input. The same interface can also be used to compare two models on the same dataset with the same attribution method. That second use is equally valuable because it reveals whether model choice changes which training examples appear most relevant even when the comparison logic itself is held fixed.

Figure 4.5 shows the two main views of this workflow. The left panel presents the ranking comparison page. At the top of the page, the framework reports the full-ranking Spearman correlation between the two configurations and visualizes the relationship through a scatter plot of rank positions. This view summarizes the agreement of the two rankings over the entire training set. Spearman is computed over rank positions rather than raw attribution scores, so it measures whether the two configurations impose a similar ordering on the training data. The figure gives the reader an immediate sense of whether the two compared rankings are globally aligned or structurally different. Below the scatter plot, the page continues with the same result components introduced in single-method mode, but now arranged in parallel. The user can therefore



(a) Comparison Top-K results page.



(b) Overlap analysis view.

Figure 4.5: Comparison mode views.

move naturally from a global summary of agreement to the concrete examples that each configuration promotes or suppresses. In the example shown here, this makes it possible to see not only that the compared rankings are strongly aligned overall, but also which examples appear at the top or bottom of each side and how similar those sets actually are.

The comparison mode also includes a dedicated overlap view, shown on the right side of Figure 4.5. This tab shifts the focus from the entire ordering to the shared structure of the upper-ranked region. The overlap view contains three layers of information. First, it reports top-percent overlap across several thresholds, which shows how much of the upper-ranked region is shared as the cutoff becomes more or less selective, making the difference between narrow and broad agreement immediately visible. Second, it reports compact summary statistics, most notably Jaccard@K, which measures the overlap between the two top-K sets relative to their union. Third, the page explicitly separates returned examples into shared points and points unique to each configuration, which reveals exactly which training examples are jointly emphasized and which are specific to one side of the comparison.

These components separate two kinds of agreement that are often conflated. Spearman summarizes the full ordering, while Jaccard and the shared/non-shared panels focus on the examples a user is most likely to inspect first. By exposing both views in one workflow, the framework makes attribution stability visible at the level of concrete training examples. A user can identify points that are shared across configurations, points that are specific to one method, and cases where broad rank agreement does not translate into identical top-ranked examples. This is the main value of comparison mode: it turns two score vectors into a comparative analytical view.

This comparison workflow also connects the system to prior data-valuation work. Fairshare Data Pricing uses valuation methods such as BM25, InflIP, and DataInf to guide purchasing decisions in simulated LLM training-data markets, including settings based on math, medical, and physical reasoning tasks [43]. Such studies evaluate attribution or valuation methods through aggregate outcomes, for example whether selected or purchased data improves model utility under a budget. The interface here provides a complementary local view. Since GradDot corresponds to the gradient inner-product idea used by InflIP, the same comparison mechanism can place a lexical valuation principle such as BM25 beside a gradient-based one and expose the retrieved examples behind the aggregate decision. The existing comparison view demonstrates this local interpretability layer: rankings can be inspected, their overlaps can be measured, and their shared and method-specific examples can be read directly.

4.6 Qualitative Insights

The demonstrations above and additional sample runs also suggest several qualitative insights about the models, methods, and data settings tested in the framework. These observations are not intended as a large-scale benchmark of attribution quality. Rather, they illustrate the kinds of questions that become easier to ask when attribution outputs are placed inside an interactive workflow.

In the text QA setting with Pythia-family models, gradient-based of GradDot and GradSim, typically show signs of high agreement including Spearman correlation of above 0.9. Since both methods use the same underlying gradient representation, their high full-ranking Spearman

correlation is expected. At the same time, high global agreement does not necessarily mean that the same examples appear in the small set a user actually inspects when using the system. GradDot and GradSim are strongly aligned in full-ranking terms, but their top-K overlap could be smaller for individual queries. This distinction is visible in the comparison workflow of Figure 4.5: Spearman summarizes broad rank structure, while top-K overlap and shared points show whether the inspected examples are actually the same for the interested query.

The returned text examples also suggest that gradient-based methods often retrieve examples with similar task form and question structure rather than only exact lexical overlap. In the MathQA setting, highly ranked examples often share properties such as comparable question length, multiple-choice formatting, and word-problem structure. Figure 4.2 shows why the result cards are useful for this kind of inspection: the user can see the problem text, answer format, rank, and score together. This makes it possible to notice when attribution reflects model-conditioned task structure rather than only keyword matching.

BM25 would provide a clear contrast. Because it is driven by lexical overlap, it often retrieves a different neighborhood from gradient-based methods. In the sampled text bank, BM25 is largely separated from GradDot, GradSim, and DataInf. This makes BM25 useful as a non-gradient reference point. When BM25 and a gradient method agree, the result may reflect both surface similarity and model-conditioned relevance. When they disagree, the framework helps the user inspect whether the gradient method is surfacing structurally similar examples even when lexical overlap is weak.

Influence-style approximation provides another source of method-specific behavior, with more differing attribution retrieval results suggesting that they can define a different attribution neighborhood from raw gradient alignment. The practical effect is especially visible at the level of individual queries: changing from a similarity-oriented method to an influence-oriented method can substantially change which training examples appear at the top. This is one reason the framework emphasizes shared and non-shared top points rather than only aggregate agreement.

Under the settings with video-based dataset, the framework also shows that method behavior can change across modalities. In the sampled video bank, GradDot and GradSim remain related, but they do not collapse as strongly as in the text setting, with Spearman correlation dropping to the range of below 0.75 typically. One plausible interpretation is that normalization matters more for multimodal examples, where video content, text metadata, and representation scale vary more substantially. BM25 is also mostly weakly aligned with gradient-based video attribution, which is expected because it mostly uses textual metadata associated with a clip.

The video consensus results further show that there is no single universal top-ranked neighborhood. Across the sampled video queries, a smaller number of top-ranked examples are shared by multiple methods, while many are method-specific. This makes comparison mode especially useful for multimodal attribution: shared clips suggest a small consensus signal, while method-specific clips reveal how different scoring rules emphasize different aspects of the same video collection.

The medical setting suggests a different kind of inspection. After closer examination of raw data, non-gradient methods often retrieve patient records with overlapping medical codes or similar sequence summaries, which is not the same as model-conditioned patient-history similarity. In addition, the EHR display also makes visible confounding factors such as token count, visit

count, and time-step structure. These quantities matter because a long patient history may contain more overlapping codes and more opportunities for model interaction. Thus, retrieved EHR examples should be interpreted not only by score, but also by whether similarity comes from clinically meaningful trajectory structure or from broader utilization patterns.

4.7 Runtime and Practicality of the Workflow

The runtime behavior of the framework follows the design principle used throughout the thesis: expensive training-side computation is moved offline, while the user-facing workflow remains interactive. The relevant question is not whether all compute work can be rebuilt instantly, but whether a user can browse examples, submit queries, compare configurations, and inspect results within a normal research workflow once the reference artifacts are available.

Under a single NVIDIA RTX 3090 GPU with sufficient CPU and network support, basic interface operations are effectively immediate from the user's perspective. Loading the catalog, changing dataset or model settings, and browsing paginated validation points happen at sub-second scale. The main visible delay begins when the user submits an attribution request that requires model-backed scoring.

For the full-sized text-based QA dataset settings, the runtime depends on model size and method family. With Pythia-410M, a single-query run using ordinary gradient-based methods is generally within 10 seconds, with a representative average around 6.2 seconds for the sampled runs. Low-rank gradient methods are faster in the same setting, typically around 2–4 seconds, with an average near 3.3 seconds. With Pythia-1B, the same workflow is slower but remains usable: ordinary gradient-based methods are expected to take within 20 seconds for warm query, with an average around 13.9 seconds, while low-rank gradient methods are closer to an average around 7.1 seconds. The first live request after backend startup can be slower because model and runtime state may need to be initialized, but repeated requests are closer to the warm-query estimates.

Comparison mode adds cost because two attribution configurations must be evaluated before the backend computes agreement and overlap summaries. For Pythia-410M, a single-query comparison between configurations typically 8–15 seconds, with a representative average around 11.7 seconds. The rank-correlation, Jaccard, and overlap computations are lightweight; most of the time is spent obtaining the two score vectors. This places comparison mode above the single-method in terms of latency, but still within a practical exploratory range: a user can run a comparison, inspect shared and method-specific examples, and adjust the method or input without rebuilding the training-side artifacts.

The driving-video setting has a different profile. Although with fewer samples, each query involves multimodal processing with Qwen3-VL. For the current 443-video sample collection, a warm single-query attribution run is expected to still take under 15 seconds typically, with a representative average around 9.8 seconds. This makes the video setting practical for inspection, especially because the returned examples are rendered as clips and can be judged visually rather than only through metadata.

Offline reference construction is more expensive and at a higher scale. As an example, for MathQA, MedQA, and PIQA datasets together, ordinary gradient-reference construction takes

approximately 12 hours for Pythia-410M under the same single-GPU setting. For the current Qwen3-VL driving-video collection, reference construction is around 5 hours for 443 videos. However, we would like to emphasize that these are one-time setup costs for the backend server, which does not affect per-query runtime for users who use the data attribution system and interact with the frontend. Furthermore, such backend computation times are still suitable for the typical research workflow for foundation models.

The runtime estimates of our system support the practicality of the framework as an exploratory attribution workbench. The hardware setup for our runtime measurements involves only a modest workstation-class GPU for foundation-model experiments, but once the reference artifacts are prepared, the interaction loop is in the seconds to low tens-of-seconds range. This is appropriate for the thesis goal: enabling users to efficiently query, compare, and inspect training-example attributions without rerunning expensive training-side computation during the session.

4.8 Preliminary User Feedback

To obtain some preliminary user feedback on our framework, we have asked 4 users with computer science and machine learning background to complete a short attribution-inspection task. Before the task, they are first provided with a short demonstration of single-method attribution using a sample question from another question-answering dataset. The demonstration showed how to select a dataset, model, and attribution method, run attribution, and inspect the returned training examples. After this demonstration, users were asked to use the system to find highly influential MathQA training examples for a given mathematical question-answering problem, compare results from two attribution methods, inspect the returned top examples, and explain what the comparison metrics indicated. The goal of this exercise was formative rather than statistical. It was intended to test whether the central workflow of the thesis—selecting a setting, running attribution, inspecting examples, and comparing methods—was understandable and useful to users. Table 4.1 reports the anonymous user feedback from this task.

The most consistent positive feedback concerned direct example-level inspection. Users found the returned training-example cards useful because the system displayed the original instruction or question directly on the result page. This made the attribution output immediately inspectable. Rather than seeing only an example index or a raw score, users could read the returned problem and judge whether the top examples were related to the query. This was especially important for the assigned task, because users were asked to find influential training examples and explain whether the returned examples made sense. The feedback therefore supports one of the main design choices of the framework: attribution results should be presented as readable training examples, not only as score vectors or ranked identifiers.

Several comments also support the top-K and bottom-K result design. One user noted that showing both ends of the ranking made it clearer that the method was not merely returning superficially similar questions, but was ranking examples in opposite directions. Another user found the small summary visualization for the top-K results helpful because it made it easier to see whether highly ranked examples were concentrated around a similar answer pattern or label. These comments suggest that users benefited from having both concrete examples and lightweight aggregate summaries on the same result page. The framework therefore helps users

move between two levels of interpretation: reading individual returned examples and observing summary patterns among the returned set.

The comparison workflow received particularly strong feedback. Users repeatedly identified the shared versus method-specific layout as one of the most useful parts of the system. In the comparison task, the Shared column gave users a direct way to see which examples were retrieved by both attribution configurations, while the method-only columns showed how the rankings diverged. This was useful both for experienced and less experienced users. One user interpreted the shared examples as "core supporters" that both methods considered important, and suggested that such examples could be candidates for data selection or dataset distillation. Another user, who had not used data attribution methods before, still found the comparison view understandable because it connected abstract method differences to actual returned question-answer examples. This confirms that comparison mode is not only a metric display. It is an example-level interface for understanding where attribution methods agree and where they differ.

Users also found the Spearman scatter plot useful as a visual summary of ranking agreement. The scatter plot helped users see whether two methods ranked the training set similarly, while the shared and method-specific panels helped them inspect the concrete examples behind that agreement or disagreement. One user explicitly noted that seeing full-ranking Spearman correlation and Jaccard overlap together helped distinguish broad ranking agreement from agreement among the top examples they would inspect first. This feedback aligns with the motivation for including both global and local comparison metrics.

At the same time, the feedback revealed several usability issues. The most common issue was that comparison metrics need clearer explanation at the point of use. Some users could not immediately distinguish Spearman from Jaccard when both were displayed together. Although the demonstration helped, the interface would benefit from short descriptive labels. This feedback suggests that the system could benefit from making the interpretation of its metrics more explicit. A second issue concerned score calibration. Users found raw influence scores difficult to interpret without additional context. A natural improvement would be to display percentiles, normalized values, or z-scores alongside raw scores. The existing score-distribution histogram indeed gives part of this context, but the user feedback suggests that the result cards themselves should also connect each score to the full distribution.

The feedback also identified onboarding and navigation improvements. Users suggested search or filtering over saved validation examples, especially when browsing a validation set by pages. One user needed a moment to distinguish Live Query Mode from Validation Point Mode, even though the demonstration made the difference clear. Another user wanted more in-context guidance when choosing attribution methods, such as recommended starting methods, labels distinguishing lexical baselines from gradient-based methods, or suggested method pairs in comparison mode. These are not changes to the core attribution logic, but they would make the workflow easier for users with less foundation in data attribution.

Overall, the preliminary user feedback supports the workflow framing of the thesis. Users valued the features that turned attribution scores into inspectable objects. The most important negative feedback concerned explanation and guidance rather than the core workflow itself. Such observations point to concrete future work: adding metric tooltips, score percentiles or normalization, validation-point search, guided task presets, and recommended comparison pairs. The main lesson is that attribution workflows are most useful when they make model-specific

training-data relationships visible through examples, while also helping users understand the metrics and method choices that structure those examples.

User	What worked	What did not work
User 1	I like that the system directly showed the original instruction or question for each returned training example. For the QA tasks, this makes the result immediately useful because I can read the top few examples and quickly judge whether they were related to the query. I don't have to open another dataset file or map an index back to the original data. The top and bottom layout also helps me understand that the method was ranking examples in opposite directions, not just returning a list of similar-looking questions. The comparison mode helps me understand which examples both methods agreed on and which examples came from only one method. The scatter plot is also intuitive because I can visually see whether the two methods ranked the training set in a similar way.	When I first look at the comparison page, Spearman and Jaccard were shown close together, and I can't immediately tell what each one measured. Short labels such as "full-ranking agreement" and "top-k overlap" will help. The raw influence score is hard to benchmark; a percentile, normalized score, or z-score makes it easier to judge whether a value is unusually high.
User 2	The shared column in comparison mode was the most useful part for completing the task. Since the task asked me to compare two attribution methods, the Shared column gave me a concrete set of examples that both methods considered important. I interpreted these shared examples as core supporters for the query. They felt more reliable than examples retrieved by only one method, and I could imagine using them as candidates for data selection or dataset distillation. I also liked that the page showed both full-ranking Spearman correlation and Jaccard overlap. Seeing both metrics together helped me understand that two methods might agree over the broad ranking but still disagree on the examples I would actually inspect first.	For the validation-point task, browsing saved examples would be easier with search or filtering. Pagination works, but finding a specific type of problem still requires manual browsing. I also wanted more context for raw influence scores, since their magnitudes are hard to compare across methods. A normalized score, percentile rank, or z-score would help. It would also be useful to save or export the shared examples from comparison mode for later inspection.
User 3	The system makes the task much easier than what I would normally do manually. If I want to know which similar training examples the model may have seen when answering a question, I usually need to combine embeddings, retrieval outputs, or dataset search by myself. With this system, I can select the dataset and model, run attribution, and get a ranked list in a few clicks. This makes the workflow feel much more direct. I also like that the returned examples were displayed on the page with their instructions, responses, scores, and ranks. For the task, this helped me compare the query with the top-ranked examples immediately. The small summary visualization for the top k results is useful, because it let me quickly see if the top examples were concentrated around a similar answer pattern or label, instead of requiring me to count them manually. The demonstration before the task was also helpful because it gave me a clear example of how to go from a question to returned training examples before I try the mathqa task myself.	At first, I need a moment to distinguish live query mode from validation point mode. I understand after the demonstration that one mode lets me type a new query and the other selects an existing validation example, but the toggle is not visually prominent enough. Larger labels or short descriptions such as "enter a new question" can help. A guided task preset will also make the first interaction smoother for new users.
User 4	Even though I had not used the data attribution methods before, the comparison view helped me understand the task. The most useful part was that the system connected abstract method comparison to actual returned examples. I could then look at the method-only columns and see what each method retrieved differently. This helped me reason about whether the methods were operating in a similar space, even without fully understanding the math behind the methods. Seeing the actual mathqa problems made it possible to decide whether the results made sense for the query. The initial demonstration also helped me understand the basic single-method workflow before using comparison mode.	The method choices were still somewhat difficult for me as a first-time attribution user. The demonstration and method descriptions helped, but I still wanted more in-context guidance near the selector, such as "recommended first method", "lexical baseline". It would also help if comparison mode suggested common method pairs and briefly explained what kind of difference to expect, such as lexical overlap versus model-based relevance.

Table 4.1: Preliminary user feedback from the attribution-inspection task.

Chapter 5

Conclusion

This thesis presented an interactive framework for training data attribution in foundation models. The central idea was that attribution becomes more useful when it is treated not only as a score assigned to training examples, but as a workflow for querying, comparing, and interpreting model behavior. The framework supports this view by combining live query computation with precomputed training-side reference artifacts and by exposing the resulting attribution outputs through a unified interface. Within the same system, a user can issue live queries, select existing validation points, aggregate multiple inputs, compare attribution configurations, inspect overlap among highly ranked examples, and apply the same workflow across several data settings.

The framework was demonstrated on public question-answering datasets, a multimodal driving-video setting, and a medical record setting based on structured EHR sequences. Across these settings, the same operational pattern remains intact even though the input formats and result displays differ substantially. Text examples are presented as readable question–answer records, video examples are rendered through media-aware browsing and result views, and medical records are surfaced through structured sequence summaries. This consistency is the main contribution of the thesis: it shows that training data attribution can be organized as a reusable workflow framework rather than as a collection of method-specific scripts.

The demonstrations also show the analytical value of comparison and aggregation. Comparison mode makes it possible to move between full-ranking agreement, top-region overlap, and shared versus configuration-specific retrieved examples. Multi-query support extends attribution beyond single-example inspection by allowing a small group of related inputs to define the ranking through elementwise score aggregation. The qualitative observations in Chapter 4 further show that this workflow can reveal method- and modality-specific behavior: gradient methods can agree globally while differing in top examples, lexical and model-based methods can surface different neighborhoods, video attribution requires visual inspection of clips, and EHR attribution requires attention to sequence structure.

The practical and user-facing additions also support the workflow framing. Runtime estimates show that, once reference artifacts are prepared, the system can support seconds-to-low-tens-of-seconds interaction on a single workstation-class GPU. Preliminary user feedback suggests that users especially valued the parts of the system that made attribution concrete: displayed training examples, top-K result cards, shared comparison columns, and rank-agreement visualizations. At the same time, the feedback identified interface improvements that would make the

workflow easier to use.

Limitations and Future Work

The current framework leaves several directions for future work. First, the system relies on precomputed training-side artifacts for gradient-based and low-rank methods. This design is essential for interactive use, but it also means that attribution behavior depends on the quality, dimensionality, and construction procedure of the stored reference representations. Future work could study these artifact choices more systematically, including how rankings change under different projection dimensions, gradient targets, or influence approximations.

Second, the demonstrations in this thesis emphasize framework behavior and workflow functionality rather than exhaustive benchmark-scale attribution evaluation. A broader empirical study across larger models, larger datasets, and more varied modalities would clarify how the framework behaves as the scale and heterogeneity of the attribution problem increase.

Third, preliminary user feedback suggests several concrete interface improvements. Users found the example-level and comparison views useful, but they wanted clearer metric labels, better score calibration, search or filtering over validation examples, stronger distinction between live-query and validation-point modes, and more guidance when choosing attribution methods. These improvements point toward a more guided attribution workbench with metric tooltips, score percentiles or normalized scores, saved comparison outputs, suggested method pairs, and example task presets.

Finally, the framework can be improved as an interactive system. Query caching, approximate retrieval over reference stores, batched processing for multimodal inputs, and more efficient server-side execution would improve responsiveness in larger deployments. User-facing studies would also be valuable: researchers and practitioners may use comparison mode, overlap analysis, and multi-query inspection in different ways when debugging models, curating datasets, or studying training-data behavior. These directions would extend the thesis contribution from a working framework to a more complete environment for practical data attribution research.

Bibliography

[1] Ekin Akyurek, Tolga Bolukbasi, Frederick Liu, Binbin Xiong, Ian Tenney, Jacob Andreas, and Kelvin Guu. "Towards Tracing Knowledge in Language Models Back to the Training Data". In: *Findings of the Association for Computational Linguistics: EMNLP 2022*. Association for Computational Linguistics, 2022, pp. 2429–2446. DOI: 10.18653/v1/2022.findings-emnlp.180. URL: https://aclanthology.org/2022.findings-emnlp.180/.

[2] Alon Albalak, Yanai Elazar, Sang Michael Xie, Shayne Longpre, Nathan Lambert, Xinyi Wang, Niklas Muennighoff, Bairu Hou, Liangming Pan, Haewon Jeong, Colin Raffel, Shiyu Chang, Tatsunori Hashimoto, and William Yang Wang. "A Survey on Data Selection for Language Models". In: *arXiv preprint arXiv:2402.16827* (2024). URL: https://arxiv.org/abs/2402.16827.

[3] Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. "MathQA: Towards Interpretable Math Word Problem Solving with Operation-Based Formalisms". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, 2019, pp. 2357–2367. DOI: 10.18653/v1/N19-1245. URL: https://aclanthology.org/N19-1245/.

[4] Shuai Bai, Yuxuan Cai, Ruizhe Chen, et al. "Qwen3-VL Technical Report". In: *arXiv preprint arXiv:2511.21631* (2025). URL: https://arxiv.org/abs/2511.21631.

[5] Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar van der Wal. "Pythia: A Suite for Analyzing Large Language Models Across Training and Scaling". In: *Proceedings of the 40th International Conference on Machine Learning*. Vol. 202. Proceedings of Machine Learning Research. PMLR, 2023, pp. 2397–2430. URL: https://proceedings.mlr.press/v202/biderman23a.html.

[6] Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. "PIQA: Reasoning about Physical Commonsense in Natural Language". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 5. 2020, pp. 7432–7439. DOI: 10.1609/aaai.v34i05.6239. URL: https://ojs.aaai.org/index.php/AAAI/article/view/6239.

[7] Tyler A. Chang, Dheeraj Rajagopal, Tolga Bolukbasi, Lucas Dixon, and Ian Tenney. "Scalable Influence and Fact Tracing for Large Language Model Pretraining". In: *The Thir-*

teenth International Conference on Learning Representations. 2025. URL: https://openreview.net/forum?id=gLa96FlWwn.

[8] Sang Keun Choe, Hwijeen Ahn, Juhan Bae, Kewen Zhao, Minsoo Kang, Youngseog Chung, Adithya Pratapa, Willie Neiswanger, Emma Strubell, Teruko Mitamura, Jeff Schneider, Eduard Hovy, Roger Grosse, and Eric Xing. "What is Your Data Worth to GPT? LLM-Scale Data Valuation with Influence Functions". In: *arXiv preprint arXiv:2405.13954* (2024). URL: https://arxiv.org/abs/2405.13954.

[9] Edward Choi, Mohammad Taha Bahadori, Jimeng Sun, Joshua Kulas, Andy Schuetz, and Walter Stewart. "RETAIN: An Interpretable Predictive Model for Healthcare Using Reverse Time Attention Mechanism". In: *Advances in Neural Information Processing Systems*. Vol. 29. 2016. URL: https://proceedings.neurips.cc/paper/2016/hash/231141b34c82aa95e48810a9d1b33a79-Abstract.html.

[10] Logan Engstrom, Axel Feldmann, and Aleksander Madry. "DsDm: Model-Aware Dataset Selection with Datamodels". In: *Proceedings of the 41st International Conference on Machine Learning*. Vol. 235. Proceedings of Machine Learning Research. PMLR, 2024, pp. 12491–12526. URL: https://proceedings.mlr.press/v235/engstrom24a.html.

[11] Junyi Gao, Cao Xiao, Yasha Wang, Wen Tang, Lucas M. Glass, and Jimeng Sun. "StageNet: Stage-Aware Neural Networks for Health Risk Prediction". In: *Proceedings of The Web Conference 2020*. 2020, pp. 530–540. DOI: 10.1145/3366423.3380187.

[12] Jakob Geyer, Yohannes Kassahun, Mentar Mahmudi, Xavier Ricou, Rupesh Durgesh, Andrew S. Chung, Lorenz Hauswald, Viet Hoang Pham, Maximilian M"uhlegg, Sebastian Dorn, Tiffany Fernandez, Martin J"anicke, Sudesh Mirashi, Chiragkumar Savani, Martin Sturm, Oleksandr Vorobiov, Martin Oelker, Sebastian Garreis, and Peter Schuberth. *A2D2: Audi Autonomous Driving Dataset*. 2020. arXiv: 2004.06320 [cs.CV]. URL: https://arxiv.org/abs/2004.06320.

[13] Amirata Ghorbani and James Zou. "Data Shapley: Equitable Valuation of Data for Machine Learning". In: *Proceedings of the 36th International Conference on Machine Learning*. Vol. 97. Proceedings of Machine Learning Research. PMLR, 2019, pp. 2242–2251. URL: https://proceedings.mlr.press/v97/ghorbani19c.html.

[14] Roger Grosse, Juhan Bae, Cem Anil, Nelson Elhage, Alex Tamkin, Amirhossein Tajdini, Benoit Steiner, Dustin Li, Esin Durmus, Ethan Perez, Evan Hubinger, Kamile Lukosiute, Karina Nguyen, Sam McCandlish, Jared Kaplan, and Samuel R. Bowman. "Studying Large Language Model Generalization with Influence Functions". In: *arXiv preprint arXiv:2308.03296* (2023). URL: https://arxiv.org/abs/2308.03296.

[15] Xiaochuang Han, Daniel Simig, Todor Mihaylov, Yulia Tsvetkov, Asli Celikyilmaz, and Tianlu Wang. "Understanding In-Context Learning via Supportive Pretraining Data". In: *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Toronto, Canada: Association for Computational Linguistics, 2023, pp. 12660–12673. DOI: 10.18653/v1/2023.acl-long.708. URL: https://aclanthology.org/2023.acl-long.708/.

[16] Cheng-Yang Hsieh, Chien-Chou Su, Shih-Chieh Shao, Sheng-Feng Sung, Swu-Jane Lin, Yea-Huei Kao Yang, and Edward Chia-Cheng Lai. "Taiwan's National Health Insurance

Research Database: Past and Future". In: *Clinical Epidemiology* 11 (2019), pp. 349–358. DOI: 10.2147/CLEP.S196293. URL: https://pubmed.ncbi.nlm.nih.gov/31118821/.

[17] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. "LoRA: Low-Rank Adaptation of Large Language Models". In: *arXiv preprint arXiv:2106.09685* (2021). URL: https://arxiv.org/abs/2106.09685.

[18] Andrew Ilyas, Sung Min Park, Logan Engstrom, Guillaume Leclerc, and Aleksander Madry. "Datamodels: Understanding Predictions with Data and Data with Predictions". In: *Proceedings of the 39th International Conference on Machine Learning*. Vol. 162. Proceedings of Machine Learning Research. PMLR, 2022, pp. 9525–9587. URL: https://proceedings.mlr.press/v162/ilyas22a.html.

[19] Dan Iter, Reid Pryzant, Ruochen Xu, Shuohang Wang, Yang Liu, Yichong Xu, and Chenguang Zhu. "In-Context Demonstration Selection with Cross Entropy Difference". In: *Findings of the Association for Computational Linguistics: EMNLP 2023*. Singapore: Association for Computational Linguistics, 2023, pp. 1150–1162. DOI: 10.18653/v1/2023.findings-emnlp.81. URL: https://aclanthology.org/2023.findings-emnlp.81/.

[20] Lavender Yao Jiang, Xujin Chris Liu, Nima Pour Nejatian, Mustafa Nasir-Moin, Duo Wang, Anas Abidin, Kevin Eaton, Howard Antony Riina, Ilya Laufer, Paawan Punjabi, et al. "Health System-Scale Language Models Are All-Purpose Prediction Engines". In: *Nature* 619.7969 (2023), pp. 357–362. DOI: 10.1038/s41586-023-06160-y. URL: https://www.nature.com/articles/s41586-023-06160-y.

[21] Cathy Jiao, Weizhen Gao, Aditi Raghunathan, and Chenyan Xiong. "On the Feasibility of In-Context Probing for Data Attribution". In: *Findings of the Association for Computational Linguistics: NAACL 2025*. Ed. by Luis Chiruzzo, Alan Ritter, and Lu Wang. Albuquerque, New Mexico: Association for Computational Linguistics, 2025, pp. 5155–5170. DOI: 10.18653/v1/2025.findings-naacl.286. URL: https://aclanthology.org/2025.findings-naacl.286/.

[22] Di Jin, Eileen Pan, Nassim Oufattole, Wei-Hung Weng, Hanyi Fang, and Peter Szolovits. "What Disease Does This Patient Have? A Large-Scale Open Domain Question Answering Dataset from Medical Exams". In: *Applied Sciences* 11.14 (2021), p. 6421. DOI: 10.3390/app11146421. URL: https://www.mdpi.com/2076-3417/11/14/6421.

[23] Pang Wei Koh and Percy Liang. "Understanding Black-box Predictions via Influence Functions". In: *Proceedings of the 34th International Conference on Machine Learning*. Vol. 70. Proceedings of Machine Learning Research. PMLR, 2017, pp. 1885–1894. URL: https://proceedings.mlr.press/v70/koh17a.html.

[24] Iuliia Kotseruba, Amir Rasouli, and John K. Tsotsos. *Joint Attention in Autonomous Driving (JAAD)*. 2016. arXiv: 1609.04741 [cs.CV]. URL: https://arxiv.org/abs/1609.04741.

[25] Zeljko Kraljevic, Dan Bean, Anthony Shek, Rebecca Bendayan, Harry Hemingway, Joshua Au Yeung, Alexander Deng, Alfie Baston, Jack Ross, Esther Idowu, James T. Teo, and

Richard J. B. Dobson. "Foresight – Generative Pretrained Transformer (GPT) for Modelling of Patient Timelines Using EHRs". In: *arXiv preprint arXiv:2212.08072* (2024). URL: https://arxiv.org/abs/2212.08072.

[26] Yongchan Kwon, Eric Wu, Kevin Wu, and James Zou. "DataInf: Efficiently Estimating Data Influence in LoRA-tuned LLMs and Diffusion Models". In: *The Twelfth International Conference on Learning Representations*. 2024. URL: https://openreview.net/forum?id=9m02ib92Wz.

[27] Seongmin Lee, Zijie J. Wang, Aishwarya Chakravarthy, Alec Helbling, ShengYun Peng, Mansi Phute, Duen Horng Chau, and Minsuk Kahng. "LLM Attributor: Interactive Visual Attribution for LLM Generation". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 39.28 (2025), pp. 29655–29657. DOI: 10.1609/aaai.v39i28.35357. URL: https://ojs.aaai.org/index.php/AAAI/article/view/35357.

[28] Yifei Li, Xiang Yue, Zeyi Liao, and Huan Sun. "AttributionBench: How Hard Is Automatic Attribution Evaluation?" In: *Findings of the Association for Computational Linguistics: ACL 2024*. Bangkok, Thailand: Association for Computational Linguistics, 2024, pp. 14919–14935. DOI: 10.18653/v1/2024.findings-acl.886. URL: https://aclanthology.org/2024.findings-acl.886/.

[29] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. "Visual Instruction Tuning". In: *arXiv preprint arXiv:2304.08485* (2023). URL: https://arxiv.org/abs/2304.08485.

[30] Tai Nguyen and Eric Wong. "In-Context Example Selection with Influences". In: *arXiv preprint arXiv:2302.11042* (2023). URL: https://arxiv.org/abs/2302.11042.

[31] Sung Min Park, Kristian Georgiev, Andrew Ilyas, Guillaume Leclerc, and Aleksander Madry. "TRAK: Attributing Model Behavior at Scale". In: *Proceedings of the 40th International Conference on Machine Learning*. Vol. 202. Proceedings of Machine Learning Research. PMLR, 2023, pp. 27074–27113. URL: https://proceedings.mlr.press/v202/park23c.html.

[32] Davide Placido, Bo Yuan, Jessica X. Hjaltelin, Chunlei Zheng, Amalie D. Haue, Piotr J. Chmura, Chen Yuan, Jihye Kim, Renato Umeton, Gregory Antell, et al. "A Deep Learning Algorithm to Predict Risk of Pancreatic Cancer from Disease Trajectories". In: *Nature Medicine* 29.5 (2023), pp. 1113–1122. DOI: 10.1038/s41591-023-02332-5.

[33] Garima Pruthi, Frederick Liu, Satyen Kale, and Mukund Sundararajan. "Estimating Training Data Influence by Tracing Gradient Descent". In: *Advances in Neural Information Processing Systems*. Vol. 33. 2020, pp. 19920–19930. URL: https://proceedings.neurips.cc/paper/2020/hash/e6385d39ec9394f2f3a354d9d2b88eec-Abstract.html.

[34] Stephen Robertson and Hugo Zaragoza. "The Probabilistic Relevance Framework: BM25 and Beyond". In: *Foundations and Trends in Information Retrieval* 3.4 (2009), pp. 333–389. DOI: 10.1561/1500000019.

[35] Ohad Rubin, Jonathan Herzig, and Jonathan Berant. "Learning to Retrieve Prompts for In-Context Learning". In: *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Seattle, United States: Association for Computational Linguistics, 2022, pp. 2655–2671.

DOI: 10.18653/v1/2022.naacl-main.191. URL: https://aclanthology.org/2022.naacl-main.191/.

[36] Ethan Steinberg, Jason Alan Fries, Yizhe Xu, and Nigam H. Shah. "MOTOR: A Time-to-Event Foundation Model for Structured Medical Records". In: *The Twelfth International Conference on Learning Representations*. 2024. URL: https://openreview.net/forum?id=NialiwI2V6.

[37] Liwen Sun, Hao-Ren Yao, Gary Gao, Ophir Frieder, and Chenyan Xiong. "Intercept Cancer: Cancer Pre-Screening with Large Scale Healthcare Foundation Models". In: *arXiv preprint arXiv:2506.00209* (2025). DOI: 10.48550/arXiv.2506.00209. URL: https://arxiv.org/abs/2506.00209.

[38] Alexander Wettig, Aatmik Gupta, Saumya Malik, and Danqi Chen. "QuRating: Selecting High-Quality Data for Training Language Models". In: *Proceedings of the 41st International Conference on Machine Learning*. Vol. 235. Proceedings of Machine Learning Research. PMLR, 2024, pp. 52915–52971. URL: https://proceedings.mlr.press/v235/wettig24a.html.

[39] Theodora Worledge, Judy Hanwen Shen, Nicole Meister, Caleb Winston, and Carlos Guestrin. "Unifying Corroborative and Contributive Attributions in Large Language Models". In: *arXiv preprint arXiv:2311.12233* (2024). URL: https://arxiv.org/abs/2311.12233.

[40] Michael Wornow, Rahul Thapa, Ethan Steinberg, Jason A. Fries, and Nigam H. Shah. "EHRSHOT: An EHR Benchmark for Few-Shot Evaluation of Foundation Models". In: *Advances in Neural Information Processing Systems*. Vol. 36. 2023. URL: https://proceedings.neurips.cc/paper_files/paper/2023/hash/d42db1f74df54cb992b
Abstract-Datasets_and_Benchmarks.html.

[41] Mengzhou Xia, Sadhika Malladi, Suchin Gururangan, Sanjeev Arora, and Danqi Chen. "LESS: Selecting Influential Data for Targeted Instruction Tuning". In: *Proceedings of the 41st International Conference on Machine Learning*. Vol. 235. Proceedings of Machine Learning Research. PMLR, 2024, pp. 54104–54132. URL: https://proceedings.mlr.press/v235/xia24c.html.

[42] Zichun Yu, Spandan Das, and Chenyan Xiong. "MATES: Model-Aware Data Selection for Efficient Pretraining with Data Influence Models". In: *arXiv preprint arXiv:2406.06046* (2024). URL: https://arxiv.org/abs/2406.06046.

[43] Luyang Zhang, Cathy Jiao, Beibei Li, and Chenyan Xiong. "Fairshare Data Pricing via Data Valuation for Large Language Models". In: *Advances in Neural Information Processing Systems*. 2025.