# Progressive Fine-Tuning for Cost-Effective Structured Attribute Generation in E-commerce

**Lakshman Kolasani**
Amazon
kolasl@amazon.com

**Fatemeh Taheri Dezaki**
Amazon
fatimaat@amazon.com

## Abstract

Large language models (LLMs) excel at structured information generation but face cost and latency challenges when deployed at scale in user-facing products. We present a parameter-efficient supervised fine-tuning pipeline for adapting a small language model (SLM) to structured attribute generation in e-commerce product listing, enabling continuous model improvement with implicit user feedback without expensive manual annotation. Our approach involves *completeness-deficit guided curation*, which ranks samples by divergence between model predictions and catalog listing attributes, selecting the highest completeness gap examples for progressive fine-tuning. Our system is deployed on a large-scale product listing service, reducing inference costs by 98% and p90 latency by 70% using a fine-tuned SLM relative to the baseline LLM while preserving an 86.4% user acceptance rate, translating to significant monthly infrastructure savings.

## 1 Introduction

Structured attribute generation is a fundamental task in e-commerce, where product listings require accurate population of hundreds of attributes ranging from structured fields (size, color, material) to unstructured content (titles, descriptions). As e-commerce services process millions of requests daily, the need for efficient and cost-effective solutions becomes paramount (Licardo and Tankovic, 2024). While large language models (LLMs) have shown strong performance on this task through in-context learning (Brown et al., 2020), deploying them at scale presents significant cost and latency challenges.

Modern product listing systems process multimodal inputs, combining product images with textual descriptions, to generate structured attributes that must conform to predefined schemas (Figure 1). Using state-of-the-art models like Claude

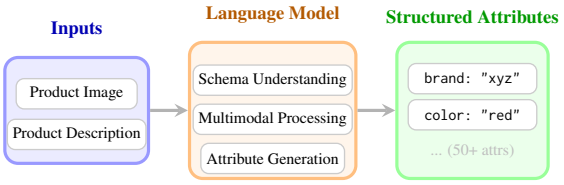



Figure 1: Structured attribute generation task. Multimodal inputs (product images and descriptions) are processed by a language model to generate schema-compliant structured attributes.

Sonnet 4.5 (Anthropic, 2025) achieves high quality but incurs substantial inference costs. The fundamental question becomes: *Can we adapt smaller, more efficient models to match the quality of larger models for domain-specific industrial applications through targeted fine-tuning?*

We address this challenge with two key contributions:

1. **Completeness-Deficit Guided Curation**: Data curation is critical for effective fine-tuning. We introduce a domain-specific sample selection method for structured attribute generation that leverages implicit user feedback. We select training examples where the model's predictions have the largest "completeness gap" compared to catalog listing attributes, focusing training on areas of weakness.

2. **Progressive Fine-Tuning**: We demonstrate that progressive model adaptation, where each training iteration initializes from the preceding model state rather than the original base model, outperforms cumulative data aggregation approaches, achieving higher completeness with less data and fewer training steps.

Our experiments on a large-scale product listing system show that a fine-tuned small language model (SLM), Amazon Nova 2 Lite (Amazon

Web Services, 2025), achieves comparable quality to Claude Sonnet 4.5 (selected as the highest-performing model for our use case at the time of experimentation) while reducing inference costs by 98% and p90 latency by 70%, maintaining an 86.4% user acceptance rate. We validate these results through both offline evaluation and online A/B testing.

## 2 Related Work

**Parameter-Efficient Fine-Tuning** Low-Rank Adaptation (LoRA) (Hu et al., 2022) enables efficient fine-tuning by training low-rank adapter matrices while keeping base model weights frozen. This approach has been successfully applied to various NLP tasks (Dettmers et al., 2023; Liu et al., 2024b), including structured output generation.

**Structured Output Generation** LLMs have been applied to structured information extraction through prompting (Wei et al., 2022) and fine-tuning (Wang et al., 2023; Dagdelen et al., 2024). Recent work focuses on schema-constrained generation (Li et al., 2024) and structured attribute extraction for e-commerce (Yang et al., 2023). Fine-tuning best practices have been extensively documented (Parthasarathy et al., 2024), though domain-specific adaptations remain challenging.

**Active and Continual Learning** Our completeness-deficit guided curation relates to active learning (Settles, 2009), where informative samples are selected to maximize learning efficiency. Progressive fine-tuning connects to continual learning (Parisi et al., 2019), which addresses how models can learn from sequential data without catastrophic forgetting (Kirkpatrick et al., 2017). Unlike traditional continual learning that focuses on preventing forgetting, our approach leverages checkpoint initialization to accelerate convergence on new hard examples.

## 3 Methodology

Figure 2 illustrates our progressive fine-tuning pipeline for structured attribute generation.

### 3.1 Problem Formulation

As illustrated in Figure 1, given a product with input features $x$ (images, product description), the task is to generate a set of structured attributes $y = \{a_1, a_2, ..., a_n\}$ where each attribute $a_i$ must

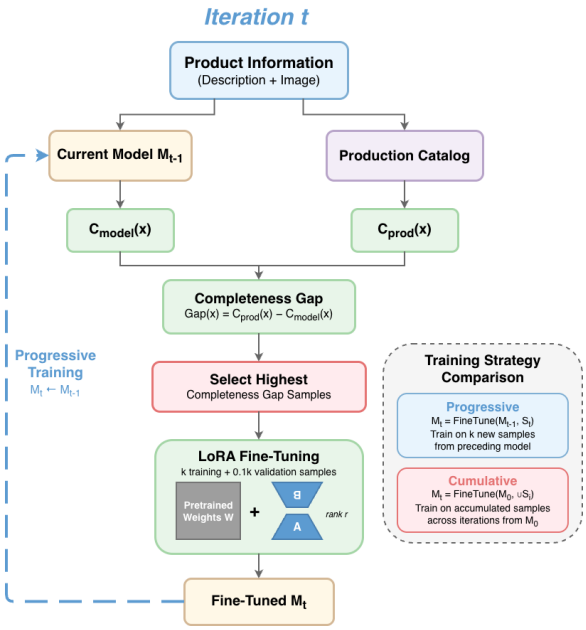



Figure 2: Progressive fine-tuning pipeline. Samples with the highest completeness gap between model predictions and catalog listing attributes are selected for LoRA fine-tuning ($k$ new samples per iteration). Dashed arrow indicates progressive training where $M_t$ initializes from $M_{t-1}$, unlike cumulative training which restarts from base model $M_0$.

conform to a predefined schema including allowed values, formats, and constraints.

We define *completeness* as the percentage of requested attributes for which the model generates a valid value:

$$\text{Completeness}(y) = \frac{|\{a_i : a_i \neq \emptyset \wedge \text{valid}(a_i)\}|}{|A|} \quad (1)$$

where $A$ is the set of all attributes requested for the product category. Importantly, completeness measures *schema-valid, non-empty* predictions rather than exact match against ground truth. For example, if a "color" attribute has allowed values including "Navy" and "Dark Blue," a prediction of "Navy" when the catalog listing value is "Dark Blue" counts as complete, since both are schema-valid. Exact match is ill-defined for such attributes with multiple valid values. This design reflects the deployment context: users review all model suggestions before submission, so generating any valid prediction reduces manual effort. Values outside the schema's allowed set (e.g., a new brand not yet registered) are not counted as complete; schemas are updated periodically to incorporate emerging values. We measure *correctness* separately through manual audits (Section 5.2).

### 3.2 Completeness-Deficit Guided Curation

While various data selection strategies exist for fine-tuning, many approaches do not explicitly target model weaknesses on the specific downstream task. We propose *completeness-deficit guided curation*, which leverages implicit user feedback by selecting samples where model predictions diverge most from catalog listing attributes. Given the same input $x$, we compute completeness for both the catalog listing data $C_{\text{cat}}(x)$ (representing existing attribute values in the product catalog) and the model's predictions $C_{\text{model}}(x)$. The *completeness gap* captures this divergence:

$$\text{Gap}(x) = C_{\text{cat}}(x) - C_{\text{model}}(x) \tag{2}$$

For each training iteration $t$, we select the $k$ samples with the highest completeness deficit:

$$S_t = \underset{|S|=k, S \subseteq D}{\arg\max} \sum_{x \in S} \left( C_{\text{cat}}(x) - C_{M_{t-1}}(x) \right) \tag{3}$$

where $M_{t-1}$ is the model from iteration $t-1$ (or base model for $t = 1$). Many catalog listing attributes are optional and frequently left blank. The completeness gap identifies products where the catalog contains attribute values that the model fails to generate, revealing learnable patterns. This approach focuses training on examples where the model underperforms relative to existing catalog listings, effectively using implicit feedback to guide sample selection without requiring explicit annotations.

### 3.3 Fine-Tuning Strategies

We compare two fine-tuning strategies. Let $k$ denote the number of new samples selected per iteration.

**Cumulative Training** Train from base model $M_0$ with accumulated data:

$$M_t = \text{FineTune}(M_0, \bigcup_{i=1}^{t} S_i) \tag{4}$$

**Progressive Training** Train from the preceding model state:

$$M_t = \text{FineTune}(M_{t-1}, S_t) \tag{5}$$

In cumulative training, each iteration adds $k$ new samples to the training set, resulting in progressively larger datasets ($k$ in iteration 1, $2k$ in iteration 2, etc.), all trained from the base model $M_0$. In progressive training, each iteration trains on only the $k$ new samples from the current iteration, using the preceding model state as initialization. Both methods select the same number of new samples per iteration, but differ in whether data accumulates and whether training resumes from the preceding state or the original base model.

Cumulative training is the standard protocol in active learning (Settles, 2009), where newly labeled data is added to the existing training pool, and the natural baseline in continual learning (Kirkpatrick et al., 2017). It resembles curriculum learning (Bengio et al., 2009), where the model is exposed to progressively larger datasets. However, restarting from the base model each iteration means previously learned patterns must be re-acquired, and with fixed training steps, larger accumulated datasets receive fewer effective epochs, limiting convergence. In contrast, progressive fine-tuning offers a key advantage: each iteration refines model weights based on newly curated hard examples while building on previously learned representations. This allows the model to progressively specialize on difficult cases without re-learning patterns already captured in earlier iterations.

### 3.4 Fine-Tuning Dynamics

Each training iteration follows a three-phase cycle: (1) sample selection using completeness-deficit guided curation on the current model, (2) LoRA fine-tuning with the selected $k$ samples, and (3) evaluation on a held-out benchmark. For progressive training, each iteration initializes from a checkpoint of the previous iteration. Importantly, both training and validation datasets change between iterations as new samples are selected based on the current model's completeness gaps.

We compare strategies under a *fixed compute budget* constraint, where each iteration uses the same number of training steps regardless of dataset size. This reflects practical production scenarios where training time and cost are limited.

### 3.5 Batched Attribute Generation

Rather than generating all attributes in a single prompt, we partition the attribute set $A$ into $B$ batches $\{A_1, A_2, ..., A_B\}$ and generate each batch separately. This *batched generation* approach improves completeness because models perform better with shorter output contexts (Liu et al., 2024a). The trade-off is increased inference cost: more batches require more API calls and repeated input

tokens.

## 4 Experimental Setup

### 4.1 Training Configuration

We use LoRA (Hu et al., 2022) for parameter-efficient fine-tuning with rank 32. Each iteration uses $k$=1,000 new training samples and 100 validation samples, trained for 100 steps with learning rate $1 \times 10^{-5}$, global batch size 32, and sequence length 8,192 tokens. Training is performed on 4× NVIDIA A100 GPUs using distributed training. We use the same prompt template across all experiments to ensure fair comparison between training strategies. For batched attribute generation, we use $B = 4$ batches, determined empirically to balance completeness improvement against inference cost. Batch assignment is randomized to avoid systematic biases from attribute ordering.

### 4.2 Dataset

We construct our dataset from a production product listing system. The source consists of public-facing catalog listing attributes that have been reviewed and verified, serving as ground truth for training. Inputs are multimodal, combining product images and text descriptions. The schema contains more than 50 structured attributes per product on average. Each training iteration samples 1,000 new examples based on completeness-deficit guided curation.

For offline evaluation, we use a held-out benchmark dataset of approximately 1,000 products, completely separate from training and validation data. The benchmark spans over 1,000 product categories with a long-tail distribution (the most frequent category represents under 3% of products), covering diverse domains including jewelry, apparel, electronics, home decor, personal care, and toys. The benchmark comprises three input modalities reflecting real-world listing scenarios: text+image (37.6%), text-only (45.9%), and image-only (16.5%). This distribution mirrors production traffic patterns where users provide varying combinations of product information. Multimodal inputs (text+image) generally yield the highest completeness, while image-only inputs present the greatest challenge due to limited textual signal for attribute inference.

### 4.3 Models

We compare three configurations: (1) Claude Sonnet 4.5 as the baseline, (2) Amazon Nova 2 Lite with LoRA adapters as the fine-tuned model, and (3) Amazon Nova 2 Lite without fine-tuning as the base comparison. While our experiments use Amazon Nova 2 Lite, the methodology (completeness-deficit guided curation and progressive fine-tuning) is model-agnostic and applicable to any LoRA-compatible base model.

### 4.4 Evaluation Metrics

We evaluate models using both offline benchmarks and online metrics.

**Offline Evaluation (Fixed Benchmark)** We measure three metrics on the held-out benchmark dataset:

- **Completeness**: Percentage of valid attribute predictions (as defined in Section 3.1)
- **Latency**: p90 response time (90th percentile)
- **Cost**: Inference cost based on publicly available model pricing

**Online Evaluation (Production A/B Test)** We measure real-world performance through production deployment:

- **Completeness**: Attribute coverage on live production traffic
- **User Acceptance Rate**: Percentage of model-generated suggestions accepted without modification, reflecting real-world utility as perceived by end users

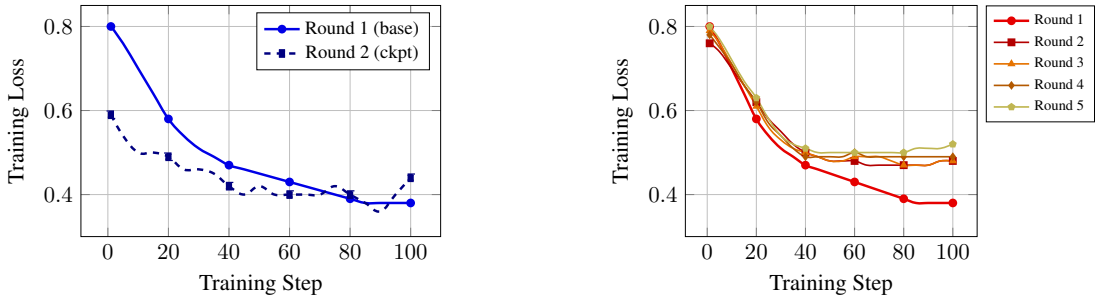
## 5 Results

### 5.1 Training Dynamics Analysis

Figure 3 illustrates the training dynamics for both strategies under our fixed compute budget constraint (100 training steps per iteration).

In progressive training (Figure 3a), Round 2 begins at lower initial loss (0.59 vs 0.80), reflecting a combination of checkpoint initialization and the characteristics of newly selected samples. The model builds on learned representations from Round 1 while adapting to new hard examples.

In cumulative training (Figure 3b), each iteration restarts from the base model with accumulated data. Under the fixed compute budget, this results in progressively fewer effective epochs as data accumulates (Round 1: ∼3 epochs over 1K samples; Round 5: ∼0.6 epochs over 5K samples). This epoch dilution contributes to incomplete convergence, with final loss *increasing* across rounds (0.38 → 0.52) despite more training data.




**(a)** Progressive Training (2 rounds)

**(b)** Cumulative Training (5 rounds)

Figure 3: Training loss curves. **(a) Progressive:** Round 2 initializes from R1 checkpoint, starting at significantly lower loss (0.59 vs 0.80), a 26% reduction that demonstrates knowledge transfer from prior training. **(b) Cumulative:** All rounds restart from base model with accumulated data. Final loss increases across rounds (R1→0.38, R5→0.52) because with fixed 100 training steps, larger datasets result in fewer effective epochs (R1: ∼3 epochs over 1K samples; R5: ∼0.6 epochs over 5K samples), leading to incomplete convergence.

| **Model** | **Iter.** | **Compl.** | **CC**† |
|---|---|---|---|
| Claude Sonnet 4.5 | – | 70.40% | – |
| Nova 2 Lite (base) | 0 | 66.03% | 17.70% |
| *Shared Iteration 1** | | | |
| Custom (iter 1) | 1 | 71.09% | – |
| *Progressive Training* | | | |
| Custom Prog (iter 2) | 2 | **74.20%** | **20.50%** |
| *Cumulative Training* | | | |
| Custom Cumul (iter 2) | 2 | 73.72% | – |

Table 1: Progressive vs. cumulative training with $B$=4 batches. All Custom models are Nova 2 Lite with LoRA. Compl. = Completeness (schema-valid predictions). CC = Correct Completeness from manual audits (completeness × correctness). *Both strategies are equivalent at $t$=1.

### 5.2 Offline Evaluation

Table 1 compares the two training strategies at iteration 2. Progressive training exceeded baseline completeness in just 2 iterations with higher completeness than cumulative training on the same data volume.

At iteration 2, progressive training outperforms cumulative training (74.2% vs 73.7%) despite using the same data volume, with progressive using only the 1K newly curated samples while cumulative trains on all 2K accumulated samples from the base model. In extended experiments (not shown), cumulative training continued to decline through 5 iterations (72.4%) despite accumulating 5× more data, confirming that progressive training better addresses model weaknesses through targeted learning on the current model state.

To validate that completeness gains translate to real quality improvements, we conducted separate manual audits measuring *correct completeness* (completeness × correctness), where correctness is human-judged accuracy per attribute. As shown in Table 1, progressive fine-tuning improves correct completeness by 16% relative (17.70% → 20.50%), confirming that the model generates valid predictions for substantially more attributes after fine-tuning, reducing manual effort to fill attributes. Note that catalog listing attributes were used as proxy labels for training, while manual audits served as the independent ground-truth benchmark for evaluation.

To validate that progressive training's advantage is not merely due to epoch dilution in cumulative training, we ran an additional ablation: doubling cumulative iteration 2 to 200 training steps, matching progressive iteration 2's effective epochs on the same data volume. The scaled cumulative model showed slightly lower completeness than the 100-step version, ruling out insufficient training as the explanation and confirming that the advantage stems from checkpoint-based progressive learning rather than compute allocation.

Beyond completeness improvements, the fine-tuned SLM offered substantial efficiency gains. Based on publicly available model pricing at the time of experimentation, Nova 2 Lite reduced inference costs by approximately 98% compared to Claude Sonnet 4.5. Additionally, the smaller model architecture resulted in approximately 70% lower p90 latency, enabling faster response times in user-facing applications.

| Metric | Claude Sonnet 4.5 | Nova 2 Prog |
|---|---|---|
| User Accept. Rate | **88.3%** ±0.2 | 86.4% ±0.5 |
| Completeness | **63.0%** ±0.3 | 59.1% ±0.7 |

Table 2: Production A/B test results with 95% confidence intervals. All differences are statistically significant ($p < 0.001$, two-proportion z-test).

### 5.3 Production A/B Test Results

We deployed Nova 2 Lite Custom Prog (iter 2) in a production A/B test with thousands of users randomly assigned to treatments. Table 2 shows the results comparing against Claude Sonnet 4.5, the highest-performing model for our use case at the time of experimentation.

The fine-tuned model maintained comparable user acceptance (1.9 percentage point reduction) despite using a much smaller model. While the SLM outperforms Claude offline (74.2% vs 70.4%), the ordering reverses in production (59.1% vs 63.0%) due to distribution shift between the curated benchmark and live traffic. The SLM has been fine-tuned on only 2K samples from the offline distribution and does not yet generalize as broadly as the larger model. Production completeness also differs methodologically, measuring attributes generated *and accepted* by users rather than schema-valid generation alone. Our progressive pipeline addresses this by curating subsequent iterations from current production weaknesses.

## 6 Conclusion

We presented a progressive fine-tuning approach for adapting LLMs to structured attribute generation in e-commerce. Our completeness-deficit guided curation leverages implicit user feedback to select training examples, and our progressive training strategy achieves 98% cost reduction and 70% p90 latency improvement while maintaining an 86.4% user acceptance rate, comparable to larger models. The key insight is that progressive model adaptation with targeted sample selection converges faster than cumulative data aggregation from the base model.

The core contributions, completeness-deficit curation and progressive fine-tuning, generalize to any structured generation task with implicit feedback signals, such as form filling, data entry assistance, or configuration generation. Future work includes continuing progressive fine-tuning iterations with fresh data to address data drift, as well as extending the approach to reinforcement learning with verifiable rewards for attributes with deterministic validation rules.

## Limitations

Our experiments focus on a single product listing domain and a single base model; generalization to other structured generation tasks and model families requires validation. As noted in the production results, the fine-tuned SLM shows sensitivity to distribution shift, with relative performance reversing between offline and production settings. Progressive training was evaluated through only two iterations; longer-horizon behavior, including potential compounding of distribution bias from iterative curation, remains an open question.

## Ethics Statement

This work uses only publicly available product catalog information (e.g., size, color, material) with no personally identifiable information. All data used for training and evaluation consists of public-facing product attributes from catalog listings. Human oversight is maintained through submission workflows, where all model-generated suggestions require explicit user approval before publication.

## References


Amazon Web Services. 2025. Amazon nova 2 lite – AWS AI service cards. Technical report, Amazon Web Services. Service Card current as of December 2, 2025.

Anthropic. 2025. System card: Claude sonnet 4.5. Technical report, Anthropic.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 41–48. ACM.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901.

John Dagdelen, Alexander Dunn, Sanghoon Lee, Nicholas Walker, Andrew S Rosen, Gerbrand Ceder, Kristin A Persson, and Anubhav Jain. 2024. Structured information extraction from scientific text with large language models. *Nature Communications*, 15(1):1418.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. QLoRA: Efficient finetuning of quantized LLMs. *arXiv preprint arXiv:2305.14314*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, and 1 others. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526.

Jinheon Li, Wenyue Zhao, Jiaping Zhang, and Yizhe Zhang. 2024. ResearchAgent: Iterative research idea generation over scientific literature with large language models. *arXiv preprint arXiv:2404.07738*.

Josip Tomo Licardo and Nikola Tankovic. 2024. Performance trade-offs of optimizing small language models for e-commerce. *arXiv preprint arXiv:2410.21970*.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024a. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. 2024b. DoRA: Weight-decomposed low-rank adaptation. *arXiv preprint arXiv:2402.09353*.

German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. 2019. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71.

Venkatesh Balavadhani Parthasarathy, Ahtsham Zafar, Aafaq Khan, and Arsalan Shahid. 2024. The ultimate guide to fine-tuning LLMs from basics to breakthroughs: An exhaustive review of technologies, research, best practices, applied research challenges and opportunities. *arXiv preprint arXiv:2408.13296*.

Burr Settles. 2009. Active learning literature survey. Technical Report 1648, University of Wisconsin–Madison.

Xiao Wang, Weikang Zhou, Can Zu, Han Xia, Tianze Chen, Yuan Zhang, Rui Zheng, Junjie Ye, Qi Zhang, Tao Gui, and 1 others. 2023. InstructUIE: Multi-task instruction tuning for unified information extraction. *arXiv preprint arXiv:2304.08085*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837.

Kai Yang, Xuewen Cheng, Junming Shu, Jianhui Ye, and Haining Gu. 2023. Extreme multi-label classification for product attribute value extraction. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1872–1876.