Automating Efficient Data Collection through the Synergy of Agentic AI and Active Learning

Yael Einy
Tel Aviv University
Tel Aviv, Israel
yaeleiny@mail.tau.ac.il

Guy Dar
Tel Aviv University
Tel Aviv, Israel
guy.dar@cs.tau.ac.il

Slava Novgorodov
Tel Aviv University
Tel Aviv, Israel
slavanov@post.tau.ac.il

Tova Milo
Tel Aviv University
Tel Aviv, Israel
milo@post.tau.ac.il

Nave Frost
eBay Research
Netanya, Israel
nafrost@ebay.com

Abstract

In this paper, we present an implementation of *Sentence-to-Model*, a scalable automated data collection system. Data collection has been traditionally hard to automate, due to the unstructured nature of the task. Our system employs a synergetic approach that combines LLM agents with active learning. The input to the system is a natural language query specifying a data science need, *without an available dataset*, and the output is the outcome of the data science process – namely, a tabular dataset with an accompanying prediction model. The use of LLMs enables the handling of unstructured data, while active learning allows for prioritizing costly LLM calls. Data collection automation has the potential to empower data science research and empirical research more broadly. This work is a first step toward achieving this goal.

Keywords

Data Collection, Large Language Models, Agentic AI

1 Introduction

Data collection is the starting point of every data science project. Unfortunately, it requires expertise and reasoning capabilities [16], which classical algorithms struggle to provide [34]. The task might require arduous and repetitive work, and a semantic understanding in to identify the relevant sources. As a consequence, problems without a pre-existing structured dataset often remain unexplored.

At the same time, relevant information often exists - whether online or within organizations' internal network – but it is buried in large, unstructured sources that are ill-suited for predictive modeling. While LLMs can be pretrained on such raw input, they still fall short in performing robust relational reasoning between features and variables [44]. In contrast, specialized tabular models, such as XGBoost [13], excel at these tasks, consistently outperforming the alternatives in this setting [8, 22]. Because these models require structured tables as input, transforming fragmented and unorganized sources into tabular form is, therefore, essential for building reliable models.

Recent advances in large language models (LLMs) offer new opportunities for solving this non-trivial task [49]: their semantic understanding and reasoning capabilities make it possible to find relevant data sources and transform them into structured datasets with minimal case-specific engineering [1, 9].

Building on these advances, we introduce *Sentence-to-Model*, a modular architecture for automated data collection powered by LLM agents and complemented by a novel active learning–based algorithm to ensure cost-effective use of LLMs. By selectively guiding LLMs to focus only on data that is valuable for downstream model training, our system avoids redundant processing and reduces the expenses of LLM usage and the impact of rate limits. The architecture enables autonomous exploration of structured and unstructured sources on the web, in knowledge graphs (KGs), and/or by searching in an internal network of an organization. Our system aims to cover a broad family of data collection tasks (see Section 2.1) and is easily extendable.

Our system is applicable in the setup of large e-commerce companies, where huge amount of data is coming from different sources, both external and internal. The internal data includes combination of product data that include structured elements such as attributes and unstructured data such as title and description with user-generated content such as product reviews, together with user-behavior data (purchase history and co-views/clicks). The external sources include general market and industry trends, social media reviews, manufacturers information and more. All this requires systems that help to automate the data collection tasks in order to solve downstream tasks using machine learning models. Assume an example of a given a natural-language request (e.g., "*given the internal product data and purchase history, and publicly available sources create a dataset of compatible products that are trending and should be advertised together or sold as a bundle*"), the system outputs two artifacts: the final dataset and a machine learning model trained on the dataset. The proposed architecture allows the user to travel from a data science need to the end of a data science process with minimal human effort. A demonstration of Sentence-to-Model was recently presented in [18], but it provided only a brief description of the system, whereas this paper provides the theoretical foundations and algorithms underlying it. Toward this goal, we combine two powerful paradigms: LLM agents and active learning (AL).

The first pillar our design relies on is the LLM agent. Large language models (LLMs) have become a transformative technology across diverse domains, due to their ability to understand and generate human-like text [41]. To extend these capabilities and to enable autonomous interaction with external environments, LLM agents have been introduced [61]. In this paradigm, the LLM serves as a reasoning and planning core, while external *tools* - such as search, retrieval, and code execution – serve as its interface to the world.

EDBT '26, Tampere (Finland)

10.48786/edbt.2026.54

Recent multi-agent systems, such as *AutoData* [40], further demonstrate how coordinated LLM agents can automate large-scale data collection from the open web. Our architecture uses LLM agents to carry out data retrieval and extraction tasks on web sources, KGs, and organizational networks - converting raw inputs into structured tables of entities - with labels and features, based on the requested prediction task. This design combines the flexibility of LLM-driven exploration with cost-sensitive filtering in the rest of the pipeline.

The second pillar in our design is active learning. Automated data collection under budgetary constraints requires determining *which* data points to enrich and *what* information to acquire for them. This challenge arises both in classical active learning, where the missing information is a label [52], and in cost-aware feature acquisition, where one or more features are missing and costly to obtain [32, 58, 63]. While traditional feature acquisition methods focus on collecting features for test samples at inference time to improve prediction confidence, the recently proposed POCA framework [4] shifted the focus to acquiring features for training samples, aiming to improve model generalization under partial observability.

Our approach follows POCA's line of research but reinterprets it for the context of automated, LLM-driven data collection. We adopt POCA's formulation of the feature acquisition task—deciding which missing features to acquire to best enhance the model—but redefine the underlying cost model to reflect the real expense of invoking LLM agents for data retrieval. Unlike prior work that measures cost purely by the number of features acquired, we account for the computational and financial costs of accurate LLM-based enrichment. To ensure scalability, we combine principles from active learning [54] and coreset selection [26] to prioritize the most informative and representative samples for enrichment. This targeted, cost-aware prioritization enables efficient and generalizable data acquisition, forming a key component of our automated data collection pipeline and serving as a versatile module that can be seamlessly integrated into other pipelines seeking cost-effective and intelligent data enrichment.

1.1 Our Approach

Motivation. In this paper, we inquire into the way recent advancements in LLMs can be incorporated to streamline automated data collection. We believe that by considering the right tools for different parts of the pipeline, we can largely improve the utility of the proposed method. This task is of practical importance to data scientists to quickly engage with hypotheses about statistical correlations in the real world, with minimal friction. We identify two types of broad classes of sources of data, batched sources and entity sources (Section 2). Batched sources, such as knowledge graphs (or tables of entity features on websites, though we find those less common), and entity sources, sources where each entity is found on a different page. These data sources require different treatments. We propose a schematic approach to AI-assisted data collection. The batched sources can be used to bootstrap the table, while entity sources can be used to enrich them one by one, with AI extracting them one by one. The last component that we incorporate into the system is prioritization, to avoid natural problems that arise when we engage in AI-assisted scraping of data, rate limits (relevant for scraping in general) and LLM costs. We employ an active learning-based approach to decide on the right entities to choose, in order to reduce the uncertainty of the predictive power of the table we extract, given a pre-defined extraction budget.

System Design. Given a user query, our system requires the following four components, as depicted in Figure 1. Their roles are the following:

(1) *Explorer*: An agentic AI module whose role is to identify tabular data in the sources used to bootstrap the dataset. This initial dataset is populated with features found in batched sources. In addition, the Explorer generates instructions for the Extractor. In their advanced form, these instructions can be reusable scraping scripts.
(2) *Prioritizer*: Analyzes the retrieved data to rank entities (data rows) by their importance for effectively modeling the target task.
(3) *Extractor*: Iteratively enriches the prioritized entities with additional features, processing one entity at a time. To do this in a cost-effective manner, we rely on the prioritization in the Prioritizer. The Extractor relies on instructions generated in the Explorer to guide its work.
(4) *Modeling*: Automatically trains a machine learning model on the resulting structured dataset.

Our system marries the flexible nature of LLMs with algorithmic approaches from data science. Moreover, other LLM systems might also benefit from a similar marriage. Since prioritization is crucial for effective planning in general cases, variants of our approach can be beneficial for information-seeking agents in other contexts.

The artifacts of Sentence-to-Model can be used as-is if the output satisfies the user. Otherwise, the method can serve to test data hypotheses. The accelerated data collection process opens up new opportunities for data scientists to perform a more open-ended process – *data collection exploration* – a symbiotic human-machine process of back-and-forth experimentation, which helps identify the appropriate compromise between data needs and available data. We can map out where human effort is still required and/or refine user needs.

Contributions. To summarize our contributions:

- Sentence-to-model's demo paper [18] is the first, to our knowledge, alongside the contemporaneous [40], to explore the use of AI agents to automate tabular data collection in the open-domain context.
- We propose a generic system design suitable for the task that plays to the complementary benefits of LLMs and traditional data science algorithms. We implement a concrete workflow adhering to this design, and evaluate it on three data collection tasks.
- We propose a novel algorithm for active feature acquisition for tabular data, which we evaluate individually and as part of the entire pipeline. The implementation details and the evaluation of the algorithm are elaborated in our technical report [17].

2 Setup

This section formalizes the input query structure and describes the data collection tasks addressed by the system, along with the structured and unstructured data sources it utilizes.

2.1 Input Query

The query specifies a tabular data collection task in natural language. For concreteness, we concentrate on the family of data

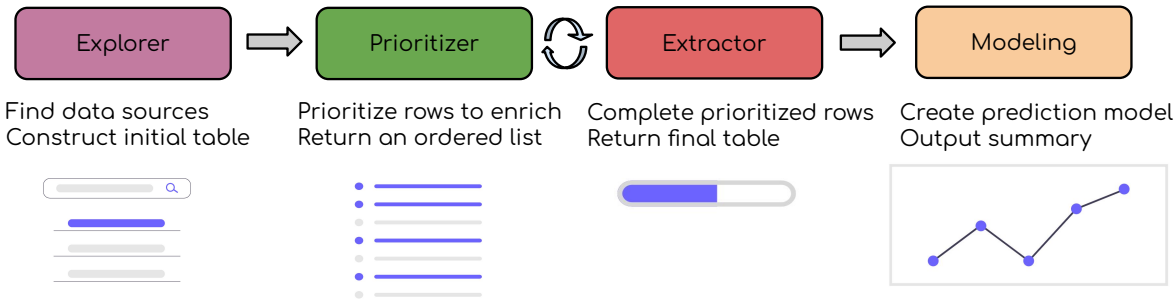

Figure 1: System Overview

collection tasks that can be described by a 2-tuple (D, T), where D is a domain of entities (e.g., *"Basketball players"*), and T is a target feature (e.g., *"salary"*). Each entity is represented by exactly one row in the dataset. The query is stated in natural language rather than a 2-tuple for user convenience at no cost (as the system operates in natural language regardless). Filtering criteria can be absorbed into the entity description (e.g., *"Basketball players born after 1980"*). Target transformations can be absorbed into the target description (*"Player's weight-to-height ratio"*). Our system is easily adaptable to data collection tasks beyond this scope.

2.2 Data Sources

Data sources are divided into two broad categories.

Batched (Table) Sources. Table Sources are sources where one page (one web page that can be scraped, one SPARQL query, one SQL query, a REST API endpoint, etc.) contains features for many entities *simultaneously*. These sources are the most economical because the retrieval cost is independent of the number of entities returned; a single request can yield thousands of data points without a proportional increase in overhead. When the Explorer finds tabular sources, it will extract them immediately programmatically to ensure robust extraction.

We can, for instance, query the DBPedia knowledge graph (KG) [5] to retrieve a comprehensive table of NBA players and their attributes. To identify which features to include, the KG can be traversed starting from a representative entity; this allows for the discovery and aggregation of properties and relations that are relevant for our task.

Entity Sources. In entity sources, on the other hand, information is spread across multiple pages, and each entity requires access to a separate page. For example, IMDB contains one page per movie. The Explorer does not extract directly from entity sources, as this is costly. Instead, it discovers site-level navigation patterns by anchoring on multiple sample entities. These patterns are used later by the Extractor to fetch entities one-by-one (according to prioritization).

3 Part I: System Design

In this section, we will elaborate on the proposed architectural design. The design complies with many traditional software engineering principles, such as modularity and encapsulation. The subsystems of the pipeline are loosely connected micro-components. Their implementation is encapsulated, and each component has a specific role, requiring only that it satisfies a contract about its inputs and outputs. With each component substitutable and extendable in isolation, this makes the design extremely modular, extendable, and even incorporable into other systems. In the following subsections, we start by describing each subsystem's role and motivation, and follow with the specifics of the implementation we examine in the current paper's experiments. The specific implementation is detailed in Figure 2.

3.1 Explorer

3.1.1 Role. The role of an Explorer component is to ingest the user's natural language query and output a dataset obtained from table sources, and an *optional* list of instructions for future extraction of entity sources. This task can benefit from LLM-based semantic skills and agentic capabilities. The Explorer operates through a streamlined pipeline with two parallel discovery paths: one for table sources and another for entity sources. At the end of the process, the Explorer outputs an initial dataset constructed from the table sources. The Explorer can also output a list of *instructions* that the Extractor can use. The format of the instructions is not constrained by our design, and it can be any of the following, or any other format:

- *Scraping Script*: A code that accepts an entity name and outputs the requested features.
- *Natural Language Instructions*: These are clear, actionable natural language instructions given to a simple ReAct agent [61] with a curated set of efficient tools. This allows for greater flexibility and fault-tolerance than a scraping tool, but it is bottlenecked by the costs of using an LLM, though simple instructions enable using smaller language models, reducing costs significantly.
- *Feature List*: This is a small list of strings, representing the features that the extractor needs to extract, without explicit instructions on how to do that. To extract these features, we use an LLM agent that can answer general questions via web search.

Instructions should be validated and tested for robustness. The Explorer is, generally speaking, expected to perform all the exploration of the external sources, including validating their performance, making the Extractor's work as predictable and mechanical as possible. For robustness, it is advisable that the Explorer check the instructions on a small representative set of entities, either by code execution (for scraping scripts), ReAct (for natural language instructions), or question answering over the web (for feature list "instructions").

3.1.2 Implementation. We find that many of the interesting table sources are contained in DBPedia [5], which is a relatively reliable data source. Therefore, we design a SPARQL agent with

access to SPARQL execution tools, inspired by the SPINACH agent for SPARQL [38]. The agent generates a SPARQL query that retrieves the requested entities, their target feature, and additional attributes, excluding entities with missing target values. For Extractor instructions, the LLM examines the existing columns and outputs a list of six missing features.

3.2 Enrichment

The enrichment process consists of two components with complementary roles. A prioritization algorithm that decides on an ordering of entities and an extraction agent that executes feature extraction based on the prioritized list.

The system operates under budgetary constraints, which raises the need for the prioritization stage. There are three main reasons for those constraints. First, certain websites have rate limits that prevent scrapers from sending unlimited queries. If the website contains one page per entity, we need to choose which entities to extract. Second, the extraction of unstructured data, such as text (for instance, an entity's Wikipedia page), is difficult to streamline and requires semantic understanding. To obtain these features robustly, it is generally recommended to use an LLM. To keep costs in check, we must set a cap on the number of calls allowed. Third, we want to minimize latency for the user. To promote use in practice, our system must be responsive and return useful artifacts in the minimal time possible.

Prioritizer. The Prioritizer is a prioritization algorithm that decides which entities are most urgent to explore further. The input to the Prioritizer is the current state of the dataset, and the output of a single Prioritizer step is the next entity to enrich. Each step of the Prioritizer is followed by an Extractor step that fills the relevant row of the dataset using data from external sources.

Intuitively, the more uncertain the model is about an entity, the more important it is. Our formal metric for quantifying the entity's importance is the gain in the accuracy of the model trained on the data with this entity's features. Since we cannot quantify this directly, we use uncertainty as a proxy, in a similar spirit to approaches from active learning. We defer a more detailed discussion of the algorithm to our technical report [17].

Extractor. An Extractor step consists of extracting the features of the entity assigned by the Prioritizer. The Extractor also takes in the Explorer's instructions and follows them on the requested entity and features. For example, if the instruction format is a script, this means running the script. In the case of natural language instructions, this means running a ReAct agent with these instructions. In the case of a list of features, a web-based question answering is performed. In our implementation, we use Tavily[1] for web-based question answering, to search information about each prioritized entity online, entity by entity, and then complete the list of features based on it. In case the value of a certain feature is not found, it is completed by the LLM's best guess.

Modeling. An automatic machine learning pipeline is applied to the dataset. Performance is evaluated using cross-validation. The modeling part takes as input the dataset and trains an XGBoost model on the dataset.

[1]https://www.tavily.com/

4 Part II: Prioritizer Architecture

In data-limited scenarios with partially observed features (e.g., data from our *Explorer*), the objective is to maximize predictive performance gains by strategically acquiring missing information under a constrained budget. This section formally defines the prioritizer problem and reviews existing solutions that serve as ablations in the development of our composite algorithm, *ExtendTab*. Full implementation details, the algorithms, and an evaluation of *ExtendTab* against these ablations and other relevant baselines are presented in the technical report [17].

4.1 Motivation

Data extracted from web sources and knowledge graphs (KGs) is inherently incomplete, leading to substantial gaps in feature availability. This problem intensifies when merging multiple data sources, where missing values accumulate, resulting in increased data sparsity. Additionally, certain data segments (subsets of data grouped by similar characteristics) are more challenging to classify than others. Some segments require richer information (i.e., additional features) to achieve satisfying classification accuracy, while others can be classified confidently based on a small set of features. Given a limited budget for acquiring additional features, our goal is to maximize predictive performance improvements by selectively enriching the dataset. Our proposed framework implements a cost-effective strategy for feature acquisition, selectively enhancing data based on predictive utility.

Entity Prioritization. Entity (row) prioritization is the first step in our enrichment process. The Prioritizer determines which entities should be enriched with additional features and ranks them by importance, using our prioritization algorithm, *ExtendTab*.

Prioritization Strategy. At a high level, *ExtendTab* trains a decision tree model and uses the model's prediction uncertainty to quantify the importance of enriching an entity with additional information. The more uncertain the model is about a given entity, the higher its priority for enrichment. This approach ensures that entities contributing most to classification difficulty receive additional feature augmentation. The process is further refined through an online feedback loop that dynamically adjusts prioritization as new data becomes available.

Due to our system's modularity, *ExtendTab* can be easily deployed independently in other systems that require cost-effective prioritization for tabular data enrichment. In addition, *ExtendTab*, as well as the other components of our framework, can be seamlessly replaced to incorporate future tools into the system.

4.2 Problem Statement

Partially Observable Cost-Aware Active Learning [4], POCA for short, is a recently proposed problem that focuses on improving the predictive performance of a supervised model $p_\phi(y|x)$ trained on a set of samples $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^I$ in a setting where sample features are *partially observed* (missing in the data) and features can be *acquired* (obtained) for a certain cost.[2] Each sample consists of a set of features $\boldsymbol{x}_i = \{x_{i,j}\}_{j=1}^J$ indexed by $j \in [J] = \{1, \ldots, J\}$ and a label y_i. We denote by $\boldsymbol{x}_{i,\mathbf{o}}$ the set of observed features with $\mathbf{o} \subseteq [J]$, where $\mathbf{o}$ potentially depends on i. In POCA, the goal is to choose the missing table cells to acquire in order to optimize a utility function $U_t(\cdot)$ subject to a cost constraint $r_t(\cdot)$ at iteration t. $U_t(\cdot)$ quantifies the tradeoff

[2]The definition also allows for samples with missing labels but this is less relevant to the Prioritizer as defined here.

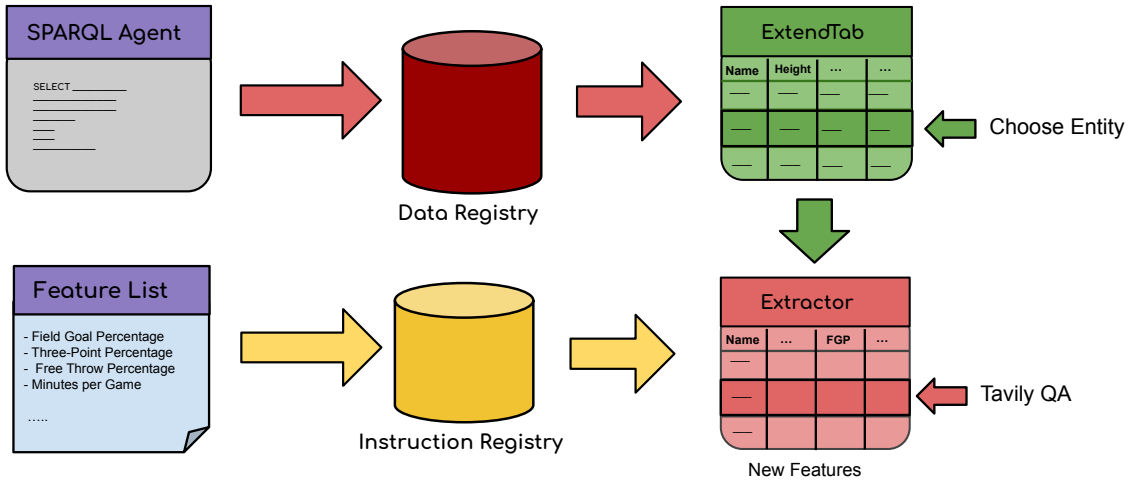

Figure 2: Sentence-to-Model implementation in this paper

between the costs of data acquisition and the increased generalization capabilities of the model p_ϕ. Thus, the POCA's objective is:

$$(i,j)^* = \underset{i\in[I],j\subseteq[J]}{\arg\max}\ U_t(i,j), \quad \text{s.t. } r_t(i,j),$$

where (i,j) indicates a set of sub-indices $i,j \subseteq [I,J]$. Exact optimization of the above objective is infeasible, as one would have to acquire and compare all sets of features. Therefore, one needs to adopt a heuristic that uses the available information $\boldsymbol{x}_{i,\mathbf{o}}$ to estimate the utility of the hypothetical acquisition of a specific set of features.

To the best of our knowledge, the recent work [4], which has introduced the POCA problem statement, is the only proposed solution addressing the problem defined above. The paper [4] operates under a well-justified but different cost model from ours. Their cost model suppresses LLM costs and considers only human labor costs. They estimate cell uncertainty as the variance between multiple LLM completions per cell.[3] Then, after estimating the uncertainty of all cells, they prioritize the completion of the most uncertain cells *by humans*, which constitute a far more expensive "resource". This approach is less natural in our setup, as in our fully automated pipeline, LLM calls are the main cost.

Cell vs. Entity Prioritization. A key distinction in prioritization strategies is between entity-based and cell-based enrichment. In our automated setup, the primary cost often involves identifying and accessing data sources related to a specific entity (sample or data row). Once accessed, it often contains multiple features. Hence, entity-based prioritization aligns naturally with our computational cost structure. Furthermore, entity-based prioritization better reflects some real-world scenarios where the incremental cost of obtaining comprehensive information for a single entity, such as requesting multiple medical tests from a patient, is often negligible. Thus, we focus our prioritization strategy on the entity-level ([4] approaches this objective with a cell-level-based cost model). The decision is to choose a subset of rows (entities) $I^* \subseteq [I]$ to extend, where the cost r is simply the number of rows chosen in I^*. This cost must be under a budget B. Thus our problem statement is:

$$I^* = \underset{I'\subseteq[I]}{\arg\max}\, U(I'), \quad \text{s.t. } r(I') = |I'| \le B.$$

Where I^* is the optimal set of sample indices chosen for acquisition, and $U(I')$ is the utility gained from acquiring all currently missing features for the set of samples I'.

4.3 Introduction to The Proposed Algorithm

We propose a new prioritization strategy based on approaches to similar problems: coreset selection [20] and active learning (AL) [52].

Coreset selection and AL improve data efficiency by selecting representative subsets of samples [20] or prioritizing the labeling of informative instances [52], respectively. They facilitate training with reduced subsets that retain essential structural properties. However, there is an important distinction between the AL and coreset selection problems and our POCA setting. While AL and coreset selection methods aim to efficiently choose samples from fully observed feature spaces, our objective differs: we seek samples to enrich by acquiring additional missing features in partially observed data.

- **Active Learning (AL) [52]**: A machine learning approach where the model iteratively queries for labels of samples from a batch of unlabeled samples, to enhance accuracy.
- **AL Uncertainty Sampling [54]**: A key strategy to perform AL, where the model selects samples for labeling based on uncertainty metrics.
- **Coreset Selection [20]**: Aims to maximize model generalization using a limited-sized training set.
- **CoreTab [26]**: A powerful coreset selection algorithm specifically designed for tabular data classification.

[3] LLM calls are non-deterministic when called in sampling mode.

While these approaches are designed for choosing a representative subset of samples to train a prediction model with fully-observed features, their algorithmic approach can be adapted to our scenario.

To address this, we propose adapting key elements from AL Uncertainty Sampling and CoreTab, formulating a simple yet effective prioritization algorithm.

Our proposed algorithm, *ExtendTab*, is a novel, two-phased prioritization strategy for the POCA problem under a row-based cost model. The algorithm first employs a tabular datamap [26], which offers a principled way to group structurally similar samples based on a classifier's decision boundaries, and assess each group's classification difficulty. Using this information, *ExtenTab* selects an initial sample set that preserves the dataset's representativeness while prioritizing hard-to-learn instances. It then transitions to an iterative Uncertainty Sampling loop to select subsequent samples based on predictive uncertainty.
The ExtendTab-FreeSize variant introduces an early stopping rule to terminate enrichment when marginal validation performance gains stagnate. Details and evaluation are in the technical report [17].

5 Experiments

We evaluate our system through the complete *Sentence-to-Model* pipeline in realistic data collection scenarios. This evaluation separately examines: (1) the *Explorer*, which identifies and retrieves relevant data sources, and (2) the *Extractor* that retrieves the required missing features, using a web search tool. The *Extractor* was tested both with and without the *Prioritizer* to measure the added benefit of budget-aware enrichment. We execute the system on an example set of user queries spanning different domains:

(1) Predict NBA All-Star selections.
(2) Identify top 10% box-office films.
(3) Classify endangered animal species.

Evaluation of our prioritization algorithm, *ExtendTab*, against ablations and baselines are provided in the technical report [17].

5.1 System Evaluation

We experiment the complete *Sentence-to-Model* system end-to-end, evaluating its ability to automatically construct task-specific tabular datasets from natural-language queries and train predictive models on the resulting data. To demonstrate the generality of our approach, we select three representative user queries spanning distinct domains:

- Q1: Create a dataset of NBA players to predict whether a player will be selected as an All-Star.
- Q2: Create a dataset of films to predict which will achieve top 10% box-office success.
- Q3: Create a dataset of animals to predict which species are endangered.

5.1.1 Explorer Experiments. One of the roles of the *Explorer* is to provide a usable starting point for the system by retrieving entities and labels from a reliable data source. In the experiments presented here, it retrieves these from DBpedia [5] (Wikipedia-derived), ensuring that the entities and labels underlying our results rely on verifiable data rather than LLM-generated content.

Even when only entities and labels are retrieved, the system can still complete its Sentence-to-Model task via feature collection by the Extractor. We therefore evaluate the *Explorer* according to two criteria: (1) the number of (*entity*, *label*) pairs successfully retrieved, and (2) the predictive performance (F_1 score) of models trained solely on the Explorer-retrieved dataset, before any feature extraction. Table 1 summarizes these results.

Table 1: Performance of the *Explorer*. For each query, we report the number of (*entity*, *label*) pairs retrieved and the F_1 score of a model trained on the dataset prior to enrichment.

Query	Entities + Targets	F_1 Score
Q1: NBA All-Star	10,000	0.1038
Q2: Film Box Office	4,761	0.6481
Q3: Animal Endangerment	3,168	0.0000

Retrieved features and completeness. The *Explorer* retrieved the following features by scraping DBpedia. For each feature, we report the proportion of existing (non-missing) values.

- **NBA:** *Player URI* (100%), *player name* (100%), *Has All-Star Selection* (100%), *dirth date* (98%), *Height* (78%), *Position* (100%), *awards* (68%), *teams* (37%).
- **Films:** *film* (100%), *film label* (100%), *box-office-class* (100%), *director-label* (89%), *cast-label* (97%), *budget* (64%), *genre* (100%), *runtime* (100%), *language-count* (100%), *production-company-count* (100%), *release-year* (10%).
- **Animals:** *animal* (100%), *endangered-status* (100%), *weight-range* (35%), *life-span* (11%).

These tables provide a usable yet highly incomplete foundation for modeling. The *Explorer*, implemented as an LLM agent, goes beyond data retrieval: leveraging both the information it has gathered in the exploration and its internal world knowledge, it reasons about which additional features are most likely to improve downstream model performance. As described in Section 2, the *Explorer* then selects six new features for each dataset—those expected to yield the highest predictive gain—and instructs the *Extractor* to obtain their values for each prioritized entity through web search using the Tavily API[4].

The selected features for each experimental task are:

- **NBA All-Star Prediction:** *Field Goal Percentage, Three-Point Percentage, Free Throw Percentage, Minutes per Game, Defensive Rebounds per Game, Steals per Game.*
- **Film Success Prediction:** *Average Critic Rating, Audience Rating, Number of Reviews, Marketing Budget, Award Nominations, Release Country.*
- **Animal Endangerment Prediction:** *Primary Predators, Reproductive Rate, Habitat Range, Breeding Habits, Human Impact, Climate-Change Effects.*

Through this process, the *Explorer* evolves from a data retriever into a strategic planner. It interprets its findings, identifies informative missing attributes, and delegates their acquisition to the *Extractor*, thereby converting an incomplete knowledge-graph snapshot into a targeted, reasoning-driven enrichment plan.

[4] https://www.tavily.com/

5.1.2 Extractor Experiments. The *Extractor* constitutes the final stage of the data-collection pipeline. Building on the feature-selection plan produced by the LLM-based *Explorer*, it completes all the features it is tasked to acquire for each prioritized entity.

The Extractor's performance reflects the effectiveness of the entire *Sentence-to-Model* framework in an end-to-end setting. Its success depends on both the availability of web-accessible information and the relevance of the features chosen by the *Explorer*. To quantify the Extractor's contribution, we evaluate the F1 score of predictive models trained on datasets enriched by the Extractor, and report the gain (Δ) over models trained on the Explorer-only dataset. This highlights the incremental value of feature acquisition via the Extractor.

We assess performance under two conditions: (1) with sample prioritization using the Prioritizer (i.e., ExtendTab), and (2) with full table completion, where the Extractor is applied to the entire initial dataset (represented by the last row in each block of Table 2). The first setting reflects realistic usage of the system with cost-sensitive enrichment, while the second isolates the Extractor's capabilities when enrichment regardless of prioritization.

As shown in Table 2, the Extractor consistently improves performance across datasets and core set sizes. The six extracted features are selected by the Explorer–Planner–LLM (Section 2), and populated using web-assisted LLM extraction.

Notably, ExtendTab-based selection outperforms random sampling in nearly all cases, underscoring the benefit of informed prioritization. These results confirm the utility of our system's modular architecture, demonstrating that reliable and meaningful model improvements can be achieved through LLM-driven retrieval, prioritization, and extraction.

Table 2: Extractor performance (F1 score) on core sets of — 10%, 20%, 33.3%, 50%, 75%, and 100% of the training data. Δ is the F1 gain of *ExtendTab* over the Explorer-only baseline (using all training samples of the *Explorer's* retrieved set). The last row in each block reports results for *ExtendTab-FreeSize* compared to a random subset of the same size.

Dataset	Size	ExtendTab	Random	Δ
Animals	237	0.3578	0.3616	+0.3578
	475	0.4589	0.3664	+0.4589
	792	0.4531	0.3964	+0.4531
	1188	0.4519	0.4252	+0.4519
	1782	0.4636	0.3529	+0.4636
	2376	0.4393	0.4393	+0.4393
FreeSize	2288	0.4275	0.4021	+0.4275
NBA	740	0.3548	0.1853	+0.2510
	1480	0.3938	0.2389	+0.2900
	2467	0.3730	0.2521	+0.2692
	3700	0.3598	0.3408	+0.2560
	5550	0.3670	0.3193	+0.3670
	7401	0.3917	0.3917	+0.2879
FreeSize	1388	0.4793	0.2021	+0.3755
Films	357	0.5985	0.5362	-0.0496
	714	0.6346	0.6301	-0.0135
	1190	0.8115	0.6891	+0.1634
	1785	0.8472	0.7705	+0.1991
	2677	0.8609	0.7854	+0.2128
	3570	0.8384	0.8384	+0.1903
FreeSize	2785	0.8993	0.7991	+0.2512

6 Future Work

The immediate purpose of Sentence-to-Model is data collection for a task. The greater vision is to become a tool in the data scientist's toolbox. To clarify this vision, we demonstrate potential venues we intend to extend and use our work.

Extending Toolbox. The current implementation serves as a proof of concept but can be readily extended. Its modular design allows seamless integration of new capabilities for unstructured and semi-structured data retrieval—such as web crawlers, domain-specific APIs, or extraction modules for organizational data. Future iterations may enable the same reasoning and prioritization framework to operate across a broader ecosystem of sources. By providing well-defined interfaces between its components, the framework establishes a flexible foundation for increasingly sophisticated and cost-aware automated data collection.

Prediction Needs. Prediction needs are ubiquitous even outside data science. Prediction markets, for example, embody the human drive to seek answers. Sentence-to-model takes a *dataset* question and outputs a prediction model. It can be easily extended to start from a *particular* question (*"What will be Michael Jordan's salary in 2030?"*). Behind the scenes, the system will transform the problem into a dataset problem (*"Create a dataset for predicting basketball players' salaries throughout their career."*), and then run the prediction model on the specific entity's features.

Prediction via Generalization. Our system's setup as-is is non-robust when data is unavailable or the requested target is missing. One way to remedy this is by casting the problem as a subset of a more general problem and using data that *is* available online to train a model on the general task. For example, assuming our task is to predict the results of an election in Narnia, where data is missing, the agent can drive a meta-exploration to find data for other election races. Then, based on the generalized task, a first-order prediction is provided automagically. An LLM can be used to automate the planning steps required for exploring potential generalizations.

The Waze of Data Collection. Sentence-to-Model helps to democratize the data collection process. Crowdsourcing can help us gain meta-insights on the process. We can request users to track the journeys taken by the framework on their requests. The journeys can be later analyzed with the help of manual or automated work. The goal of the analysis is to identify common pitfalls as well as strategies that proved useful across journeys and to compile a list of suggestions for the Planner to improve future journeys. Moreover, subgoals shared by multiple tasks can be cached for the future.

MCP Access. Sentence-to-model can be exposed as a tool usable by chatbots. A standardized way to expose tools in a easy-to-use way is called the Model Context Protocol (MCP [2]). Moreover, we can make the subsystems and even internal tools used by the subsystems available via MCP. This allows for a more granular access to the system and facilitates utilizing features of the system in isolation. For example, through MCP, translating a question about Michael Jordan, into a data science problem, as discussed in Prediction Needs above, becomes trivial.

7 Discussion and Limitations

The system implemented is still in its early stages. Fortunately, the design is modular and easy to extend, and each of its parts can easily be replaced. A major limitation of the current implementation arises when the required information is simply unavailable. While we discuss possible mitigations to this challenge in Section 6, further work is needed to enable the system to handle such cases seamlessly—without requiring prior knowledge of whether a task is feasible. At present, as our *Explorer* component supports only knowledge-graph–based retrieval, its applicability is confined to entities represented within such structured resources. Although DBpedia encompasses many entities, this still restricts the range of prediction problems the system can currently address.

Another limitation relates to temporal consistency. When features evolve over time, extracted values may reflect different temporal contexts, leading to datasets that mix attributes valid at distinct points in time. Because the data retrieved through the current *Explorer* is as up-to-date as Wikipedia, it may occasionally include outdated information. Our *Extractor*, which identifies relevant attributes across the entire web using Tavily's search API for each given entity, alleviates this restriction but introduces other risks—such as retrieving unreliable data. When information cannot be located on the web, the *Extractor* falls back on LLM-based completion, which may introduce hallucinated values. In the experiments presented in this paper, we focus on domains where factual data is abundant and reliable (e.g., animals, NBA players, films), reducing the practical impact of this issue. Nevertheless, for less prominent or frequently changing entities, ensuring data veracity remains an open challenge. Future versions could incorporate reasoning-based validation mechanisms to ensure temporal and factual consistency across features.

In order to increase the applicability of our method, some iterations should be made, but we believe the main ingredients are already accounted. The challenges above are paradoxically also the system's moat. The reason is that highly specialized challenges require tailor-made solutions, and generalist approaches (e.g., a universal agentic system [21]) fail, because many common pitfalls can be avoided and domain knowledge can be integrated.

8 Related Work

We address the challenge of automating the construction of task-specific tabular datasets under explicit budget constraints. While prior work automates parts of the data science process—such as data discovery, feature engineering, or model selection—none provide a unified, cost-aware pipeline that discovers entities, acquires features, and trains predictive models end-to-end.

Automated data science and LLM-driven systems. Recent frameworks aim to automate portions of data science using LLMs. Systems such as [40, 62] automate data retrieval and analysis from web corpora or preexisting sources. Other systems, including *Evaporate* [3] and *Symphony* [14], leverage LLMs to extract tabular views from semi-structured data lakes or multimodal repositories. However, these pipelines do not handle iterative discovery or prioritize data acquisition under cost or model performance constraints.

LLM-driven assistants such as [19, 28, 37] automate feature engineering and reasoning [24, 30, 45] over existing datasets, yet they presuppose data availability. Our approach instead constructs the dataset itself, bridging unstructured evidence and tabular modeling.

Data discovery and preparation. Structured web resources such as DBpedia [7, 36] and WebTables [10, 11] demonstrate that large-scale tabular data can be discovered on the web. Subsequent efforts, such as OpenCeres [39] and SWDE [27] extended extraction to semi-structured sources. These methods assume entities and attributes are readily available, whereas our system dynamically discovers and reconciles them across heterogeneous sources, generating missing features when none exist. Future work may integrate such extractors as complementary modules within our *Explorer*.

Data wrangling and weak supervision. Tools such as Wrangler [35] and OpenRefine [43] enable interactive transformation once data are collected. Weak-supervision frameworks like Snorkel [47] improve data quality programmatically. These methods operate *post hoc* on existing datasets, complementing our focus on pre-collection discovery and selective enrichment under budget constraints.

Active learning and feature acquisition. Active learning (AL) reduces labeling effort by labeling informative samples [6, 29, 50, 53, 54], while coreset-based methods compress datasets into representative subsets for efficient learning [12, 23, 26, 51, 60]. Active feature acquisition (AFA) extends this reasoning to the feature space, learning instance-wise policies that balance accuracy and cost [15, 31–33, 48, 56, 58, 63] at inference time, or during the training phase [4]. Our *Prioritizer* integrates these principles, to decide which data to acquire within a global budget.

Agentic LLM frameworks. Tool-augmented [46] and multi-agent systems [25, 55] (e.g., AutoGen [59], Magentic-One [21], enable orchestrated task execution via reasoning–acting loops [46, 57, 61]. These architectures demonstrate the value of multi-agent coordination but focus on conversational or reasoning tasks, not structured data construction. Our framework similarly decomposes complex objectives into modular agents but tailors each to data collection and model preparation, introducing cost-awareness and verifiable provenance—properties absent in existing generalist agents. We also evaluated OpenAI's *ChatGPT Deep Research* [42], a strong general-purpose research agent. While it can search and synthesize web content, it fails to produce structured, verifiable datasets: generated and retrieved information are intermingled, sample counts are insufficient for learning, and multi-stage pipelines (e.g., scraping, schema alignment, modeling) are not reliably executed. In contrast, our system decomposes the process into verifiable, cost-efficient stages. Future DR-like agents may augment these components, but current evidence underscores the need for specialized, decomposed architectures for automated data science.

Positioning. Existing works address isolated steps—discovery, cleaning, AutoML, AL/AFA, or LLM reasoning. Our contribution is their integration into a single, budget-aware pipeline that (1) autonomously discovers entities and sources, (2) prioritizes instances and features using AL and LLM-derived signals, (3) extracts verifiable features with tool-augmented LLMs, and (4) delivers a structured dataset and trained predictive model.

9 Conclusion

In this work, we proposed a new AutoML system with an emphasis on the data collection component of the system. We used LLM agents to enable the system to handle semantic tasks, and proposed a new algorithm for prioritizing LLM calls to enrich the initial dataset. In this work, we design and test a preliminary version of the system and identify future directions for extension.

References

[1] Anonymous. 2024. ByteScience: Bridging Unstructured Scientific Literature. *arXiv preprint arXiv:2411.12000* (2024).

[2] Anthropic. 2024. Model Context Protocol. Blog post. https://www.anthropic.com/news/model-context-protocol

[3] Simran Arora, Brandon Yang, Sabri Eyuboglu, Avanika Narayan, Andrew Hojel, Immanuel Trummer, and Christopher Ré. 2023. Language models enable simple systems for generating structured views of heterogeneous data lakes. *arXiv preprint arXiv:2304.09433* (2023).

[4] Nicolás Astorga, Tennison Liu, Nabeel Seedat, and Mihaela van der Schaar. 2024. Partially observable cost-aware active-learning with large language models. In *The Thirty-Eighth Annual Conference on Neural Information Processing Systems*.

[5] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. DBpedia: a nucleus for a web of open data. In *Proceedings of the 6th International The Semantic Web and 2nd Asian Conference on Asian Semantic Web Conference*. Berlin, Heidelberg, 722–735.

[6] Maria-Florina Balcan, Andrei Broder, and Tong Zhang. 2007. Margin based active learning. In *International Conference on Computational Learning Theory*. Springer, 35–50.

[7] Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. 2009. DBpedia—A Crystallization Point for the Web of Data. *Web Semantics* 7, 3 (2009), 154–165. doi:10.1016/j.websem.2009.07.002

[8] Vadim Borisov, Tobias Leemann, Kathrin Seßler, Johannes Haug, Martin Pawelczyk, and Gjergji Kasneci. 2022. Deep neural networks and tabular data: A survey. *IEEE transactions on neural networks and learning systems* 35, 6 (2022), 7499–7519.

[9] William Brach, Kristián Košťál, and Michal Ries. 2025. The Effectiveness of Large Language Models in Transforming Unstructured Text to Standardized Formats. *arXiv preprint arXiv:2503.02650* (2025).

[10] Michael J Cafarella, Alon Halevy, Hongrae Lee, Jayant Madhavan, Cong Yu, Daisy Zhe Wang, and Eugene Wu. 2018. Ten Years of WebTables. *PVLDB* 11, 12 (2018), 2140–2149. https://www.vldb.org/pvldb/vol11/p2140-cafarella.pdf

[11] Michael J Cafarella, Alon Halevy, Y Zhang, Z Wang, and Eugene Wu. 2008. WebTables: Exploring the Power of Tables on the Web. In *PVLDB*. https://sirrice.github.io/files/papers/webtables-vldb08.pdf

[12] Chengliang Chai, Jiabin Liu, Nan Tang, Ju Fan, Dongjing Miao, Jiayi Wang, Yuyu Luo, and Guoliang Li. 2023. GoodCore: Data-Effective and Data-Efficient Machine Learning through Coreset Selection over Incomplete Data. *Proceedings of the ACM on Management of Data* 1, 2 (June 2023), 157. doi:10.1145/3589302

[13] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *KDD*. Association for Computing Machinery, New York, NY, USA, 785–794. doi:10.1145/2939672.2939785

[14] Zui Chen, Zihui Gu, Lei Cao, Ju Fan, Samuel Madden, and Nan Tang. 2023. Symphony: Towards Natural Language Query Answering over Multi-modal Data Lakes.. In *CIDR*.

[15] Ian Connick Covert, Wei Qiu, Mingyu Lu, Na Yoon Kim, Nathan J White, and Su-In Lee. 2023. Learning to Maximize Mutual Information for Dynamic Feature Selection. In *Proceedings of the 40th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 202)*, Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (Eds.). PMLR, 6424–6447. https://proceedings.mlr.press/v202/covert23a.html

[16] Tijl De Bie et al. 2021. Automating Data Science: Prospects and Challenges. *arXiv preprint arXiv:2105.05699* (2021).

[17] Yael Einy, Guy Dar, Slava Novgorodov, Nave Frost, and Tova Milo. 2025. Automating Efficient Data Collection through the Synergy of Agentic AI and Active Learning (Technical Report). (2025). https://slavanov.com/research/edbt26-report.pdf

[18] Yael Einy, Guy Dar, Slava Novgorodov, and Tova Milo. 2025. Sentence to Model: Cost-Effective Data Collection LLM Agent. In *Companion of the 2025 International Conference on Management of Data, SIGMOD/PODS 2025, Berlin, Germany, June 22-27, 2025*. 83–86.

[19] Yael Einy, Tova Milo, and Slava Novgorodov. 2024. Cost-Effective LLM Utilization for Machine Learning Tasks over Tabular Data. In *GUIDE-AI*. Association for Computing Machinery, New York, NY, USA, 45–49. doi:10.1145/3665601.3669848

[20] Dan Feldman. 2020. Core-sets: Updated survey. *Sampling techniques for supervised or unsupervised tasks* (2020), 23–44.

[21] Adam Fourney, Gagan Bansal, Hussein Mozannar, Cheng Tan, Eduardo Salinas, Erkang (Eric) Zhu, Friederike Niedtner, Grace Proebsting, Griffin Bassman, Jack Gerrits, Jacob Alber, Peter Chang, Ricky Loynd, Robert West, Victor Dibia, Ahmed Awadallah, Ece Kamar, Rafah Hosn, and Saleema Amershi. 2024. *Magentic-One: A Generalist Multi-Agent System for Solving Complex Tasks*. Technical Report MSR-TR-2024-47. Microsoft. https://www.microsoft.com/en-us/research/publication/magentic-one-a-generalist-multi-agent-system-for-solving-complex-tasks/

[22] Léo Grinsztajn, Édouard Oyallon, and Gaël Varoquaux. 2022. Why Do Tree-Based Models Still Outperform Deep Learning on Tabular Data? *NeurIPS Datasets and Benchmarks* (2022). https://arxiv.org/abs/2207.08815

[23] Chengcheng Guo, Bo Zhao, and Yanbing Bai. 2022. Deepcore: A comprehensive library for coreset selection in deep learning. In *International Conference on Database and Expert Systems Applications*. Springer, 181–195.

[24] Siyuan Guo, Cheng Deng, Ying Wen, Hechang Chen, Yi Chang, and Jun Wang. 2024. DS-Agent: Automated Data Science by Empowering Large Language Models with Case-Based Reasoning. arXiv:2402.17453 [cs.LG] https://arxiv.org/abs/2402.17453

[25] Taicheng Guo et al. 2024. Large Language Model based Multi-Agents: A Survey of Progress and Challenges. *arXiv preprint arXiv:2402.01680* (2024). https://arxiv.org/abs/2402.01680

[26] Aviv Hadar, Tova Milo, and Kathy Razmadze. 2025. Datamap-Driven Tabular Coreset Selection for Classifier Training. *VLDB* (2025).

[27] Qiang Hao, Rui Cai, Yanwei Pang, and Lei Zhang. 2011. From one tree to a forest: a unified solution for structured web data extraction. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Beijing, China) *(SIGIR '11)*. Association for Computing Machinery, New York, NY, USA, 775–784. doi:10.1145/2009916.2010020

[28] Noah Hollmann, Samuel Müller, and Frank Hutter. 2023. Large Language Models for Automated Data Science: Introducing CAAFE for Context-Aware Automated Feature Engineering. In *NeurIPS*.

[29] Alex Holub, Pietro Perona, and Michael C Burl. 2008. Entropy-based active learning for object recognition. In *2008 IEEE computer society conference on computer vision and pattern recognition workshops*. IEEE, 1–8.

[30] Sirui Hong, Yizhang Lin, Bangbang Liu, Binhao Wu, Danyang Li, Jiaqi Chen, Jiayi Zhang, Jinlin Wang, Lingyao Zhang, Mingchen Zhuge, Taicheng Guo, Tuo Zhou, Wei Tao, Wenyi Wang, Xiangru Tang, Xiangtao Lu, Xinbing Liang, Yaying Fei, Yuheng Cheng, Zhibin Gou, Zongze Xu, Chenglin Wu, Li Zhang, Min Yang, and Xiawu Zheng. 2024. Data Interpreter: An LLM Agent For Data Science. *ArXiv* abs/2402.18679 (2024).

[31] Hung-Tien Huang, Dzung Dinh, and Junier B. Oliva. 2025. Information Templates: A New Paradigm for Intelligent Active Feature Acquisition. *arXiv preprint arXiv:2508.18380* (2025). https://arxiv.org/pdf/2508.18380

[32] Sheng-Jun Huang, Miao Xu, Ming-Kun Xie, Masashi Sugiyama, Gang Niu, and Songcan Chen. 2018. Active feature acquisition with supervised matrix completion. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1571–1579.

[33] Jaromír Janisch, Tomáš Pevnỳ, and Viliam Lisý. 2019. Classification with Costly Features as a Sequential Decision-Making Problem. *arXiv preprint arXiv:1909.02564* (2019). https://arxiv.org/abs/1909.02564

[34] Eun Seo Jo and Timnit Gebru. 2020. Lessons from Archives: Strategies for Collecting Sociocultural Data in Machine Learning. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*. ACM, 306–316.

[35] Sean Kandel, Andreas Paepcke, Joseph M Hellerstein, and Jeffrey Heer. 2011. Wrangler: Interactive Visual Specification of Data Transformation Scripts. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 3363–3372. doi:10.1145/1978942.1979444

[36] Jens Lehmann, Robert Isele, Max Jakob, and et al. 2015. DBpedia–A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia. *Semantic Web* 6, 2 (2015), 167–195. doi:10.3233/SW-140134

[37] Yin Lin, Bolin Ding, H. V. Jagadish, and Jingren Zhou. 2024. SMARTFEAT: Efficient Feature Construction through Feature-Level Foundation Model Interactions. In *CIDR*.

[38] Shicheng Liu, Sina J. Semnani, Harold Triedman, Jialiang Xu, Isaac Dan Zhao, and Monica S. Lam. 2024. SPINACH: SPARQL-Based Information Navigation for Challenging Real-World Questions. arXiv:2407.11417 [cs.CL] https://arxiv.org/abs/2407.11417

[39] Colin Lockard, Prashant Shiralkar, and Xin Luna Dong. 2019. OpenCeres: When Open Information Extraction Meets the Semi-Structured Web. 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)*, Jill Burstein, Christy Doran, and Thamar Solorio (Eds.). Association for Computational Linguistics, Minneapolis, Minnesota, 3047–3056. doi:10.18653/v1/N19-1309

[40] Tianyi Ma, Yiyue Qian, Zheyuan Zhang, Zehong Wang, Xiaoye Qian, Feifan Bai, Yifan Ding, Xuwei Luo, Shinan Zhang, Keerthiram Murugesan, Chuxu Zhang, and Yanfang Ye. 2025. AutoData: A Multi-Agent System for Open Web Data Collection. arXiv:2505.15859 [cs.IR] https://arxiv.org/abs/2505.15859

[41] Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, and Jianfeng Gao. 2025. Large Language Models: A Survey. arXiv:2402.06196 [cs.CL] https://arxiv.org/abs/2402.06196

[42] OpenAI. 2025. Introducing deep research. https://openai.com/index/introducing-deep-research/. Accessed Sep 27, 2025.

[43] OpenRefine Community. [n. d.]. OpenRefine. https://openrefine.org/. Accessed Sep 27, 2025.

[44] Anonymous (or use actual authors if known). 2025. Just Because You Can, Doesn't Mean You Should: LLMs .... *Preprint / arXiv* (2025). https://arxiv.org/html/2508.19563v1 arXiv preprint.

[45] Yichen Qian, Yongyi He, Rong Zhu, Jintao Huang, Zhijian Ma, Haibin Wang, Yaohua Wang, Xiuyu Sun, Defu Lian, Bolin Ding, and Jingren Zhou. 2024. UniDM: A Unified Framework for Data Manipulation with Large Language Models. *arXiv preprint arXiv:2405.06510* (2024).

[46] Yujia Qin, Shi Liang, Yining Ye, Kunlun Zhu, Lan Yan, Ya-Ting Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Runchu Tian, Ruobing Xie, Jie Zhou, Marc H. Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2023. ToolLLM: Facilitating Large Language Models to Master 16000+ Real-world APIs. *ArXiv* abs/2307.16789 (2023).

[47] Alexander Ratner, Stephen H Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. 2017. Snorkel: Rapid Training Data Creation with Weak Supervision. *arXiv preprint arXiv:1711.10160* (2017). https://arxiv.org/abs/1711.10160

[48] Maytal Saar-Tsechansky, Prem Melville, and Foster Provost. 2009. Active feature-value acquisition. *Management Science* 55, 4 (2009), 664–684.

[49] Andreas Schmitt et al. 2024. Challenges and opportunities of automated data pipelines. *Journal of Big Data* 11, 1 (2024), 45.

[50] Ozan Sener and Silvio Savarese. 2017. Active learning for convolutional neural networks: A core-set approach. *arXiv preprint arXiv:1708.00489* (2017).

[51] Ozan Sener and Silvio Savarese. 2018. Active Learning for Convolutional Neural Networks: A Core-Set Approach. In *International Conference on Learning Representations (ICLR)*. https://openreview.net/forum?id=H1aIuk-RW

[52] Burr Settles. 2009. Active learning literature survey. (2009).

[53] H Sebastian Seung, Manfred Opper, and Haim Sompolinsky. 1992. Query by Committee. *COLT* (1992). https://collaborate.princeton.edu/en/publications/query-by-committee

[54] Mel Silberman et al. 1996. Active learning. *Boston: Trustco* (1996).

[55] Khanh-Tung Tran, Dung Dao, Minh-Duong Nguyen, Quoc-Viet Pham, Barry O'Sullivan, and Hoang D. Nguyen. 2025. Multi-Agent Collaboration Mechanisms: A Survey of LLMs. *arXiv preprint arXiv:2501.06322* (2025). https://arxiv.org/abs/2501.06322

[56] Henrik von Kleist, Alireza Zamanian, Ilya Shpitser, and Narges Ahmidi. 2025. Evaluating Active Feature Acquisition Policies under Distribution Shift: A Semi-Offline Reinforcement Learning Perspective. *Journal of Machine Learning Research* 26 (2025), 1–84. https://www.jmlr.org/papers/volume26/23-1635/23-1635.pdf

[57] Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. 2023. Plan-and-Solve Prompting: Improving Zero-Shot Chain-of-Thought Reasoning by Large Language Models. In *Annual Meeting of the Association for Computational Linguistics*.

[58] Y. Wang, X. Zhang, J. Li, et al. 2025. A Survey on Active Feature Acquisition Strategies. *arXiv preprint arXiv:2502.11067* (2025). https://arxiv.org/html/2502.11067v1

[59] Qingyun Wu, Gagan Bansal, Jieyu Zhang, and et al. 2023. AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation. *arXiv preprint arXiv:2308.08155* (2023). https://arxiv.org/abs/2308.08155

[60] Xiaobo Xia, Jiale Liu, Jun Yu, Xu Shen, Bo Han, and Tongliang Liu. 2022. Moderate coreset: A universal method of data selection for real-world data-efficient deep learning. In *The Eleventh International Conference on Learning Representations*.

[61] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. ReAct: Synergizing Reasoning and Acting in Language Models. *ArXiv* abs/2210.03629 (2022).

[62] Yu Zhang, Junwei Liao, and Ning et al. Li. 2024. Agentic Information Retrieval: A New Paradigm for Agent-Driven Retrieval. *arXiv preprint arXiv:2410.09713* (2024). https://arxiv.org/abs/2410.09713

[63] Zhiqiang Zheng and Balaji Padmanabhan. 2002. On active learning for data acquisition. In *2002 IEEE International Conference on Data Mining, 2002. Proceedings.* IEEE, 562–569.