# Schema-Driven Information Extraction from Heterogeneous Tables

**Anonymous ACL submission**

## Abstract

In this paper, we explore the question of whether large language models can support cost-efficient information extraction from tables. We introduce *schema-driven information extraction*, a new task that transforms tabular data into structured records following a human-authored schema. To assess various LLM's capabilities on this task, we present a benchmark comprised of tables from four diverse domains: machine learning papers, chemistry literature, material science journals, and webpages. We use this collection of annotated tables to evaluate the ability of open-source and API-based language models to extract information from tables covering diverse domains and data formats. Our experiments demonstrate that surprisingly competitive performance can be achieved without requiring task-specific pipelines or labels, achieving $F_1$ scores ranging from 74.2 to 96.1, while maintaining cost efficiency. Moreover, through detailed ablation studies and analyses, we investigate the factors contributing to model success and validate the practicality of distilling compact models to reduce API reliance.[1]

## 1 Introduction

Vast quantities of experimental data are locked away in tables found in scientific literature. These tables are primarily designed for visual presentation, and the underlying data is typically not available in any structured format, such as a relational or graph database. Some table collections have simple or uniform structures (Cafarella et al., 2008), making them easy to convert to relational data, for example, Wikipedia tables (Lebret et al., 2016; Iyyer et al., 2017), however a lot of information is stored in tables with complex and varied layouts, such as tables of results in papers found on `arXiv.org`.

Prior work on extracting data from tables has focused on developing custom pipelines for each new table format or domain, for example extracting machine learning leaderboards from LaTeX result tables (Kardas et al., 2020). Importantly, the development of these specialized pipelines necessitates domain-specific labeled data, which not only incurs a significant cost in collection for every new extraction task but also constrains their applicability outside the originating domain.

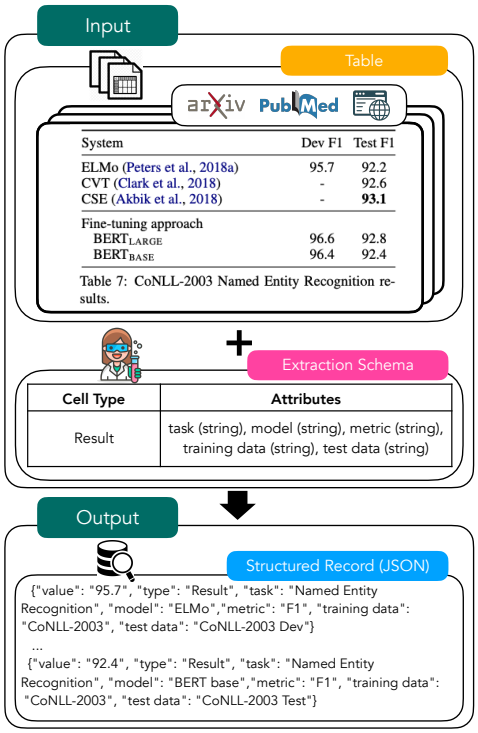


Figure 1: Overview of Schema-Driven Information Extraction. The input includes two elements: the source code of a table and a human-authored extraction schema, outlining the target attributes and their data types. The output consists of a sequence of JSON records that conform to the extraction schema.

In this paper, we show how LLMs can enable accurate domain-independent extraction of data from heterogeneous tables. We present a new formulation of the table extraction problem, which we refer to as Schema-Driven Information Extraction. In Schema-Driven IE, the only human supervision

[1] Code and data are available at an anonymous repository.

provided is a schema that describes the data model, including the target attributes and their data types, formulated in a JSON format.[2] Given an extraction schema, and a table as input, the model then outputs a sequence of JSON objects, each of which describes a table cell. For example, as demonstrated in Figure 1, a domain expert outlines the attributes of interest related to result cells in a machine learning table, and the model extracts JSON objects following this schema.

To evaluate the ability of LLMs to perform Schema-Driven IE, we introduce a new benchmark consisting of table extraction datasets in four diverse domains: machine learning papers, chemistry literature, material science journals, and webpages - each of which has a different data format (LaTeX, XML, CSV, and HTML, respectively). We curate and annotate new datasets for the first two domains, while adapting existing datasets for the latter two.

Using this newly developed benchmark, we analyze the performance of open-source and proprietary LLMs. We find that state-of-the-art proprietary models are capable of accurately extracting data from diverse domains and table formats without supervision. For example, GPT-4 (OpenAI, 2023) and `code-davinci` (Chen et al., 2021), are capable of accurate table extraction (ranging from 74.2 to 96.1 $F_1$), given only a relevant data schema as input to define the task. This performance is comparable to fully supervised models, which operate at an $F_1$ range of about 64.1 to 96.1. We also present a number of analyses on various factors that are key to achieving good performance while minimizing inference costs, including retrieving text from outside the table, in addition to an iterative error recovery strategy. Moreover, we demonstrate the utility of Schema-Driven IE by evaluating performance on the downstream task of leaderboard extraction from machine learning papers (Kardas et al., 2020).

## 2 Schema-Driven Information Extraction

We now describe Schema-Driven IE, a new task that extracts structured records from tables with minimal supervision. As shown in Figure 1, the task input contains two elements: 1) a table with numerous cells, optionally supplemented with contextual text, e.g., retrieved paragraphs from the same document; and 2) an extraction schema that outlines target attributes and their data types for various record types (implemented as JSON templates). Given the input, the model generates a sequence of JSON objects, where each object corresponds to a cell in the table and contains key-value pairs for the pre-defined attributes of a specific record type.

Consider a table in an ML paper that displays various models' results. Our proposed task enables the extraction of result records from each cell in the table. These records include relevant attributes such as the evaluation metric, task, etc, which are structured in corresponding JSON objects and could facilitate meta-analysis of experiments or support research on reproducibility.

To demonstrate the feasibility of Schema-Driven IE on tables, we introduce **INSTRUCTE**, a method to extract structured records from a broad range of semi-structured data, using only task-specific instructions. INSTRUCTE uses a template-based approach to information extraction (Chambers and Jurafsky, 2011; Chen et al., 2023), where the extraction schema is represented as a series of JSON templates. The underlying LLM is instructed to select the appropriate template and populate it with extracted values for each cell in an input table, following a specified cell traversal order. As illustrated in Figure 2 (left), the prompt used by INSTRUCTE consists of four key components: an input table (optionally) supplemented with contextual text, an extraction schema, task-specific instructions, and an initial record for starting the process.

Despite explicit instructions, we found that models often fail to generate JSON records for all the cells in a single inference pass. Instead, models often deviate from the instructed cell traversal order, leading to partial extraction of the input table's cells. To mitigate this, we use an iterative error recovery strategy. As shown on the right side of Figure 2, we detect deviations from the instructed *left-right, top-down* order by comparing predicted cell values with those from a rule-based cell detector. Then, we truncate the LLM's output to the point of deviation, and re-prompt the model with the truncated sequence, adding the value of the next target cell. This process is repeated until all records are generated. Using identified cells as a scaffold, this strategy helps the model adhere to the instructed order, significantly improving per-

[2] JSON is chosen as the output format for two main reasons: 1) its widespread use ensures a significant representation in the LLM's pre-training corpus, which is crucial for optimizing model performance; and 2) its simplicity in parsing and processing, especially its support for one-line output, makes it advantageous for outputs spanning multiple cells, offering a clear benefit over indent-based formats like YAML.

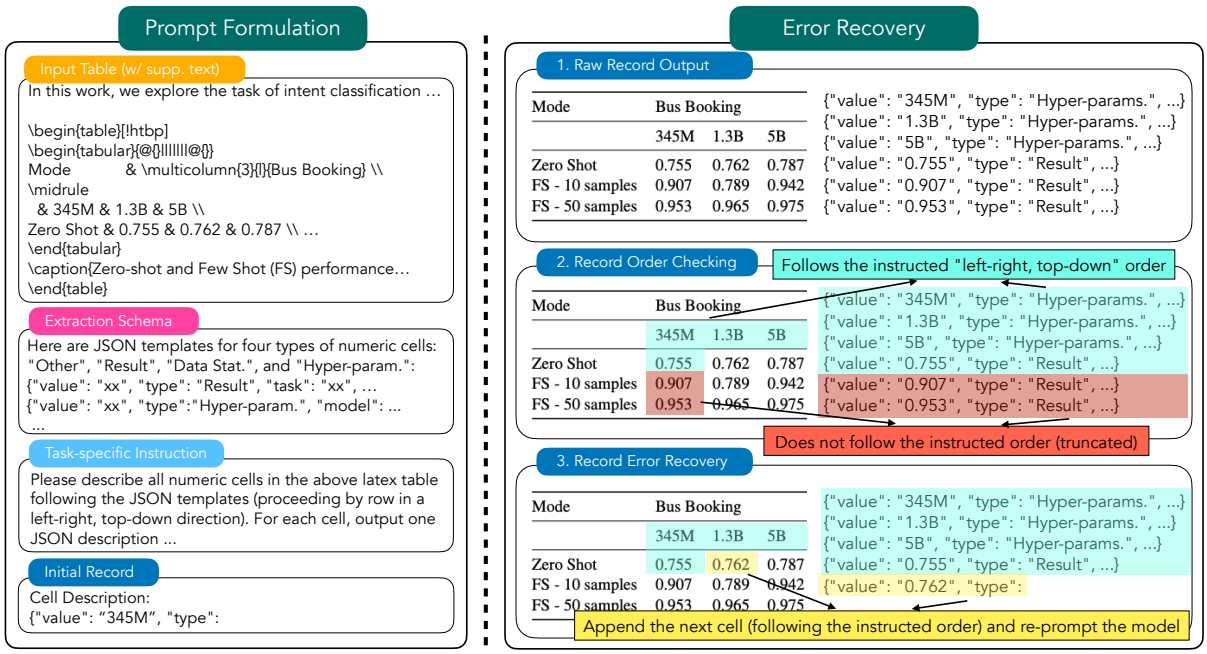


Figure 2: Left: Prompt formulation of our proposed method INSTRUCTE. Right: Illustration of our error-recovery strategy, which ensures the model compliance of the instructed cell traversal order and reduces inference costs.

formance despite potential propagated errors in cell identification. In Section 4.4, we show that our approach is much more cost-efficient than cell-by-cell prompting while achieving similar performance. For more details on INSTRUCTE, including prompt formulation and cell detectors, please refer to Appendix A.

## 3 The SCHEMA-TO-JSON Benchmark

We now present the details of our benchmark, SCHEMA-TO-JSON, which is designed to assess the capabilities of LLMs to extract data from tables, adhering to a predefined schema. This benchmark contains tables from four domains: machine learning papers, chemistry literature, materials science journals, and webpages. Each domain features a unique textual format, namely, LaTeX, XML, CSV, and HTML, requiring models to generalize across domains and formats. For ML tables, we add relevant paragraphs from the same documents to provide additional context, testing the models' capacity to jointly understand tabular and textual data. We manually annotate datasets for the first two domains and adapt pre-existing datasets into our unified format for the latter two. Statistics of the four datasets are summarized in Table 1.

**arXiv Machine Learning Tables** We create a manually annotated dataset focused on tables from arXiv ML papers, emphasizing numeric cells that are classified into four categories: Results, Hyperparameters, Data Statistics, or Other. Extraction attributes are pre-defined for the first three categories; for instance, result records incorporate textual attributes such as *evaluation metric* (e.g., $F_1$) and *dataset* (e.g., SQuAD), as shown in Figure 1. To avoid data contamination with top models like GPT-4 (0613),[3] we collected papers published after the knowledge cutoff (between October and November 2022) from three subfields: Machine Learning, Computer Vision, and Natural Language Processing. Five tables were randomly selected from each paper, including appendices. We employ computer scientists with ML backgrounds for annotation, and evaluate inter-annotator agreement (IAA) score by calculating $F_1$ (see Section 4.1 for details) on double-annotated tables, treating one set of annotations as gold labels and the other as predictions. This method yields an $F_1$ score of 96.6 when applying thresholded token-level $F_1$ for attribute matching. For additional information on ML tables, including predefined attributes and the annotation process, please refer to Appendix B.

**PubMed Chemistry Tables** We also annotate a new dataset of PubMed tables describing the physical properties of chemical compounds. The automated extraction of physical properties from such tables could provide substantial real-world benefits, for example collecting much-needed data for train-

[3] According to OpenAI website, GPT-4 (0613) was trained on data until Sep. 2021.

| | ML (ours) | Chemistry (ours) | DISCOMAT (2022) | SWDE (2011) |
|---|---|---|---|---|
| Textual format | LaTeX | XML | CSV | HTML |
| # cell types | 4 | 6 | 2 | 8 |
| # attr. types | 11 | 4 | 4 | 32 |
| # papers (web.) | 25 | 16 | 656 | 80 |
| # tables (pages) | 122 | 26 | 1,031 | 1,600 |
| # anno. records | 3,792 | 1,498 | 9,036 | 1,600 |
| # records / table | 31.1 | 57.6 | 8.8 | 1 |

Table 1: Dataset statistics of four datasets in our SCHEMA-TO-JSON benchmark.

ing ML models that can support inverse molecular design (Kim et al., 2018) and thus accelerating the drug design process (Fields, 2019; Stokes et al., 2020). Here, we focus on cells concerning five important physical properties identified by chemists: $IC_{50}$, $EC_{50}$, $GI_{50}$, $CC_{50}$, and MIC.[4] Three common attributes are manually extracted from tables for all properties: *unit*, *treatment* (experimental compound), and *target* (measured biological entity, e.g., a gene expression). Similar to the ML tables, domain experts annotate JSON records for relevant cells, and Table-$F_1$ calculated on double-annotated tables is used as the IAA score. A Table-$F_1$ score of 91.0 underscores the reliability of the dataset.

**DISCOMAT (Gupta et al., 2022)** We experiment with DISCOMAT, a dataset focusing on glass composition tables from Elsevier material science journals. The task is to extract tuples comprising (*material*, *constituent*, *percentage*, *unit*) from given tables. We adapt DISCOMAT to fit our Schema-Driven IE framework by grounding the *percentage* element to numeric cells in the table and considering the other elements as attributes. The model is tasked to identify numeric cells representing constituent percentages and predict the associated three attributes. We refer readers to Gupta et al. (2022) for more details of DISCOMAT.[5]

**SWDE (Hao et al., 2011)** Finally, we add SWDE (Structured Web Data Extraction) as a fourth dataset, aimed at extracting pre-defined attributes from HTML webpages. This dataset comprises roughly 124K pages gathered from eight distinct verticals, such as *Auto*, *Book*, and *Movie*. Each vertical includes ten unique websites and is associated with a set of 3 to 5 target attributes. For instance, the *Movie* vertical seeks to extract attributes such as `title`, `director`, and `genre`.

## 4 Experiments

We evaluate the capability of various LLMs to perform Schema-Driven IE, in addition to full fine-tuning using our benchmark. For ML and chemistry tables, we use a subset of 10 and 7 randomly sampled papers separately for model development, which facilitates the training of supervised baselines. For the two pre-existing datasets, we follow the data splits used in the original experiments.

### 4.1 Evaluation

To evaluate predicted JSON records we report $F_1$ score against gold cell attributes that are exhaustively labeled by human annotators for each table. We report results both using exact match (EM), in addition to a threshold based on token-level similarity. The token similarity threshold is tuned on dev data to maximize alignment between our estimated model performance and performance measured using human judgments (see Appendix C for more details). We report $F_1$ macro-averaged over tables, due to the wide variance in table sizes.

For DISCOMAT and SWDE, we use similar metrics specified in their original papers to support comparison with prior work. We report Tuple-$F_1$ (Gupta et al., 2022) for DISCOMAT, where a predicted 4-element tuple is considered correct only if it exactly matches the gold tuple. For SWDE, we report Page-$F_1$ (Hao et al., 2011), which measures the number of pages where the attributes are accurately predicted.[6]

To further validate our conclusions, we also present the results of full human evaluation of model outputs in §4.5.

### 4.2 Baselines & Implementation Details

We evaluate the capability of multiple LLMs to perform Schema-Driven IE, including API-based GPT-4 and GPT-3.5 models and open-source models, such as Llama2-Chat-13B (Touvron et al., 2023b), CodeLlama-instruct-13B (Rozière et al., 2023), StarCoder-15.5B (Li et al., 2023), LLaMA-7B (Touvron et al., 2023a), and Alpaca-7B (Taori et al., 2023). We also frame Schema-Driven IE as a TableQA problem, applying multi-choice and

[4] https://www.sciencedirect.com/topics/pharmacology-toxicology-and-pharmaceutical-science/ic50

[5] In the released corpus, tables are represented as matrices; we, therefore, transform them into CSV tables (using the pipe symbol "|" as the delimiter) prior to feeding them into LLMs.

[6] Notably, SWDE primarily focuses on identifying textual HTML nodes containing attribute values rather than exact text spans, so we use token-level $F_1$ to identify the most relevant HTML node for each extracted attribute.

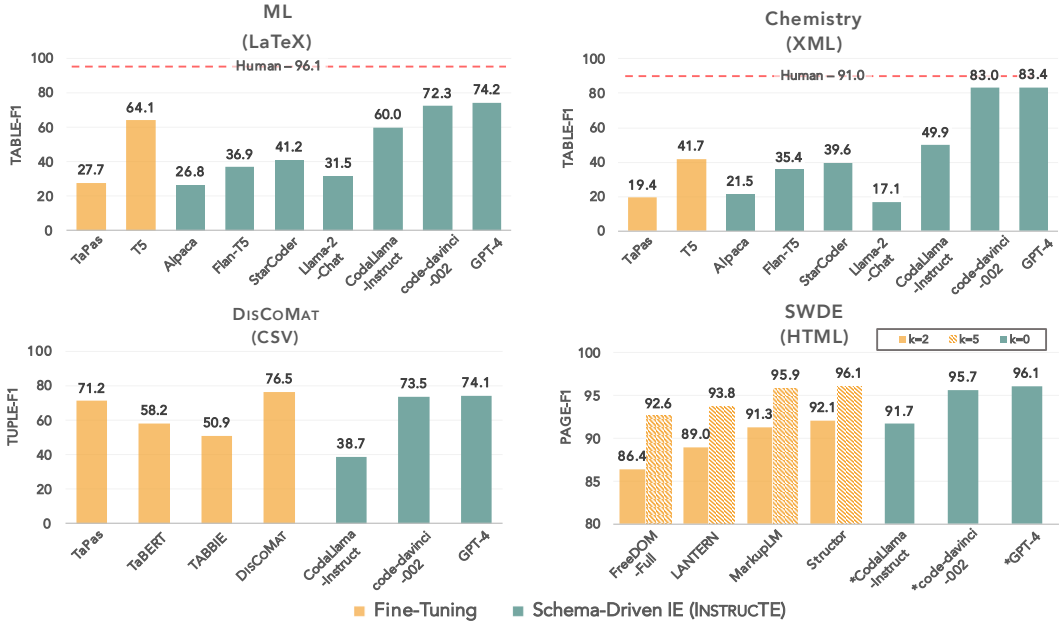


Figure 3: Capability of various LLMs to perform Schema-Driven IE, measured using the SCHEMA-TO-JSON benchmark. We employ Table-$F_1$ for our two newly annotated datasets and provide a measure of human performance. For DISCOMAT (Gupta et al., 2022) and SWDE (Hao et al., 2011), we adhere to their original evaluation metrics, i.e., Tuple-$F_1$ and Page-$F_1$ respectively, to support comparisons with established methods. In SWDE experiments, $k$ represents the number of trained websites from each vertical. Due to API cost constraints, *INSTRUCTE's results are computed on a 1,600 webpage sample, with bootstrap confidence intervals calculated to validate the reliability of these performance estimates (margin of error for 95% confidence interval with 1000 samples is 0.00995.)

extractive QA prompts for template selection and cell attribute prediction, respectively. Furthermore, we also evaluate T5-11B (Raffel et al., 2020) and TaPas (Herzig et al., 2020), a table-specialized LM. For implementation details of INSTRUCTE and other methods, see Appendix D.[7]

For DISCOMAT and SWDE, we compare INSTRUCTE with established baselines, which either design task-specific architectures, such as FreeDom (Lin et al., 2020) and LANTERN (Zhou et al., 2022), or use LMs pretrained on tables or web pages, like TaPas (Herzig et al., 2020), TaBERT (Yin et al., 2020), and MarkupLM (Li et al., 2022).

### 4.3 Main Results

Figure 3 presents the main results from the comparison between INSTRUCTE and other methods on our SCHEMA-TO-JSON benchmark. We observe that INSTRUCTE, in conjunction with API-based models, achieves strong performance across domains and input formats, without any domain-specific labels. With GPT-4, INSTRUCTE can outperform fine-tuned models on ML and chemistry tables. However, a substantial disparity remains compared to human performance, e.g., the Table-$F_1$ on double-annotated examples for ML tables stands at 96.6 when applying thresholded token-level $F_1$ for attribute matching, which is 22.4 $F_1$ points higher than GPT-4.

For DISCOMAT and SWDE, GPT-4 performs on par or slightly trails behind the fully supervised state-of-the-art methods, signifying the potential of LLMs to act as flexible, powerful tools for extracting information from tables across diverse data formats and domains.

Despite a noticeable gap when compared to API-based LLMs, open-source models, like CodeLlama-instruct-13B, show promising results in ML and web domains, achieving 60.0 Table-$F_1$ and 91.7 Page-$F_1$ on ML tables and SWDE, respectively.

### 4.4 Ablation Studies

We assess the impact of different components of INSTRUCTE, including task formulation and error recovery, using ML tables.

**LLMs & Task Formulation** In Table 2, we compare different LLMs, leading to two principal observations. First, code models show strong perfor-

[7]We developed a rule-based method for chemistry tables based on the training set, which only achieved a Table-$F_1$ score of 51.3, significantly lower than our proposed InstrucTE. Due to the substantial effort required to create specialized rule-based systems for each domain and the performance gap, we decided not to pursue this approach further.

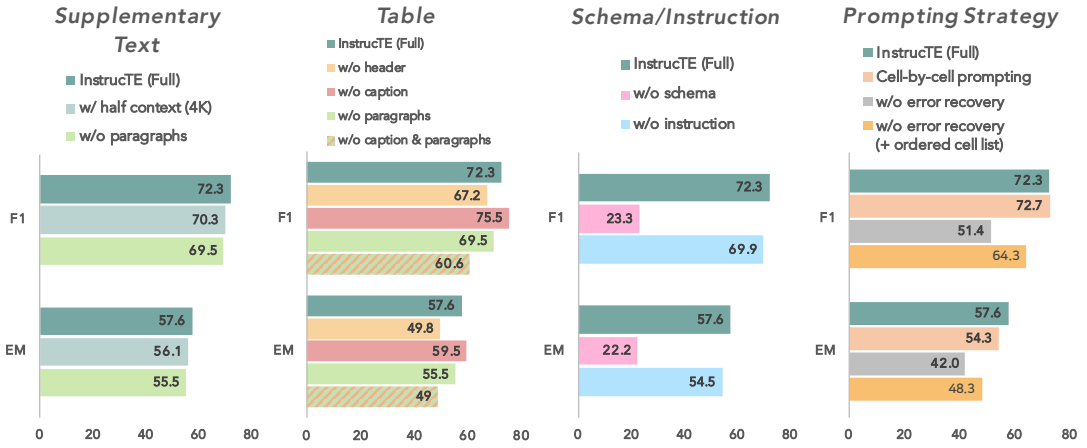


Figure 4: Ablation studies on various components of our INSTRUCTE (w/ `code-davinci-002`) on the ML tables. Interestingly, excluding the table caption improves performance. Our detailed analysis in Appendix E reveals that low-quality captions (e.g., lack of specificity) may confuse the model, leading to inaccurate predictions.

mance on Schema-Driven IE. This is evident from several key comparisons, such as the performance similarity between `code-davinci-002` and GPT-4, the superior performance of `code-davinci-002` compared to other GPT-3.5 models, and the fact that CodeLlama-instruct-13B significantly outperforms Llama2-chat-13B, approaching the performance of `gpt-3.5-turbo`. This superiority of code models might be attributed to their alignment with Schema-Driven IE, which involves converting table source code into JSON records. Second, non-code open-source models with similar sizes (for instance, those in the 6-7B range) tend to achieve comparable fine-tuning performance, though they might exhibit variations in prompting performance.

Subsequently, we compare three task formulations: SCHEMA-TO-JSON, TableQA, and Function Calling, which is a feature provided by the OpenAI API.[8] In Function Calling, the schema is formatted as function definitions with attributes serving as arguments. The LM is then tasked with selecting the function and generating JSON objects for extracted arguments on a cell-by-cell basis. From the T5-11B fine-tuning experiments, we observe that SCHEMA-TO-JSON attains better performance than TableQA, demonstrating the value of integrating task-specific instructions and extraction schema in the input. Function Calling with `gpt-3.5-turbo` shows limited effectiveness, and error analysis suggests that this shortfall primarily stems from the model's struggle in selecting the correct function.[9]

| Exp. Setup | Formulation | Model | Token-$F_1$ | EM |
|---|---|---|---|---|
| Fine-tuning (# Train=1169) | TableQA | TaPas (large) | 27.7 | 21.6 |
| | | T5 (11B) | 61.2 | 46.2 |
| | SCHE2JSON | GPT-J (6B) | 49.6 | 38.4 |
| | | LLaMA (7B) | 51.3 | 38.0 |
| | | Alpaca (7B) | 50.2 | 39.4 |
| | | T5 (11B) | 64.1 | 50.2 |
| No Fine-tuning | TableQA | Flan-T5 (11B) | 36.9 | 27.7 |
| | Func. Calling | `gpt-3.5-turbo (0613)` | 22.4 | 18.4 |
| | SCHE2JSON | GPT-J (6B) | 18.6 | 16.2 |
| | | LLaMA (7B) | 13.5 | 11.5 |
| | | Alpaca (7B) | 26.8 | 21.1 |
| | | Llama2-chat (13B) | 31.5 | 23.0 |
| | | StarCoder (15.5B) | 41.2 | 32.3 |
| | | CodeLlama-instruct (13B) | 60.0 | 44.0 |
| | | `gpt-3.5-turbo (0613)` | 64.1 | 47.9 |
| | | `text-davinci-003` | 67.4 | 50.4 |
| | | `code-davinci-002` | 72.3 | 57.6 |
| | | `gpt-4 (0613)` | **74.2** | **58.1** |

Table 2: TEST set performance on ML tables with different LLMs and task formulations.

**Prompt Components & Error Recovery** Figure 4 shows INSTRUCTE's performance subject to the exclusion of varying prompt components. We use `code-davinci-002` for these experiments considering API budget limitations and its resemblance to GPT-4 in terms of performance and context length. We observe that removing supplementary text degrades performance. Table headers contribute positively as expected, while captions surprisingly do not. Further analysis on table captions is provided in Appendix E, which suggests that unclear captions can sometimes mislead the model, resulting in inaccurate predictions. Notably, discarding the extraction schema, specifically JSON templates, causes a substantial performance decline, primarily due to attribute name mismatches in the evaluation. Lastly, we show that INSTRUCTE's performance

[8] https://platform.openai.com/docs/guides/function-calling

[9] This finding is supported by a marked performance increase to 63.8 Table-$F_1$ when the gold function is pre-specified. As each function call yields only one JSON object, this method requires cell-by-cell prompting, which is cost-intensive with GPT-4. Due to API budget constraints, our experiments are limited to `gpt-3.5-turbo`.

drops significantly without error recovery. Compared to cell-by-cell prompting, error recovery offers similar performance at a fraction of the API cost ($100 v.s. $670 on Azure).[10]

### 4.5 Performance Analysis

To further verify our main conclusions from automatic evaluation and gain deeper insights into INSTRUCTE's performance, we conduct a human evaluation and discuss a set of key questions.

**What errors are made by INSTRUCTE?** To understand where INSTRUCTE struggles, we conduct error analysis on GPT-4 predictions for ML tables. We sample 10 tables from the test set and 10 records for each table, comparing each attribute with the gold value. In total, we find 154 errors out of 591 attributes. We group errors into one of eight categories, listed in Table 3. For example, one type of false positive error is when the gold attribute value is present in the table caption, but the model is distracted by a table header. Table 3 provides a detailed breakdown and includes the top three affected attributes within each error category. We find that the most common error occurs when the model fails to identify attributes present in the table (31.2%), particularly for *experimental settings* like *5-shot* in Result records. Another major error is when attributes present in the accompanying text lead to either null predictions (14.9%) or incorrectly predicting a table header (19.5%). These errors highlight the challenges of Schema-Driven IE, where the model must understand nuances of table layouts and also effectively integrate information from surrounding text.

**How does the data format impact INSTRUCTE's performance?** The variation in model performance across datasets from different domains with unique formats raises questions about the influence of format differences. To address this, we conducted experiments converting ML tables from LaTeX to HTML and chemistry tables from XML to CSV, utilizing both commercial (`tableconvert`[11]) and open-source (`TeX4ht`[12]) tools, and selecting the one with the highest conversion accuracy. Despite `tableconvert` showing superior conversion quality, residual code from the original formats in the converted tables, e.g., LaTeX commands in HTML tables, presents a novel "code-switching" challenge for INSTRUCTE. Performance evaluation with GPT-4 reveals a minimal drop for ML tables (from 74.2 to 74.1 in Table-$F_1$) and a more significant decrease for chemistry tables (from 83.4 to 78.1 in Table-$F_1$). Both conversion noise and the model's format-specific processing capabilities could contribute to these differences. The optimal performance on original formats underlines the necessity of developing models adept at handling diverse data formats directly, rather than relying on format conversion tools.

**Human Evaluation** Similar to our error analysis on ML tables, we manually inspect attribute predictions for 100 cell records for chemistry tables and DisCoMat, as well as 160 pages for SWDE, and report the prediction precision. Half of the sampled data are double-annotated, with the inter-annotator agreement score calculated as the F1 score between the two annotations. The statistics and results are provided in Table 9 in the appendix. The results show that INSTRUCTE achieves high precision across different datasets, ranging from 73.9 to 96.4, aligning with the performance under automatic metrics. Additionally, the high inter-annotator agreement scores (all above 90) indicate that the human evaluation is reliable and consistent.

### 4.6 Knowledge Distillation

Considering the strong performance of API-based models on Schema-Driven IE, we now show that it is possible to use knowledge distillation (Le et al., 2022; Kang et al., 2023) to build a cost-efficient compact model, using ML tables as a demonstration. Specifically, this process first generates synthetic data by performing inference on unlabeled tables using `code-davinci-002`, followed by fine-tuning a smaller model (e.g., 7B parameters) using the synthetic data. We compile a collection of 979 arXiv ML papers, submitted between 2008 and 2019, yielding 3,434 tables (containing a total of 100K cells). In Table 4, we can see that LLaMA-7B and Alpaca-7B demonstrate similar performance as seen in the fine-tuning results (Table 2). While fine-tuning LLaMA with LoRA (Hu et al., 2022) presents noticeable computational efficiency, full-parameter fine-tuning of T5-11B matches the teacher model's performance.[13]

[10] The pricing for `code-davinci-002` on Azure is $0.1 per 1,000 tokens as of June 23rd, 2023.

[11] https://tableconvert.com/api/

[12] https://tug.org/tex4ht/

[13] The improvement over the teacher model is not significant (p-value is 42.3%, Berg-Kirkpatrick et al., 2012).

| Category | % (#) | Fine-grained Error Types | Top 3 Affected Attributes |
|---|---|---|---|
| False Negative | 31.2 (48) | gold answer in table, not predicted | "Result:experimental settings" (39.6%), "Result:training data" (39.6%), "Result:model settings" (20.8%) |
| | 14.9 (23) | gold answer in main text, not predicted | "Result:training data" (39.1%), "Hyper-parameter:model" (26.1%), "Hyper-parameter:dataset" (26.1%), |
| | 6.5 (10) | gold answer predicted, wrong attribute | "Result:experimental settings" (100%) |
| False Positive | 19.5 (30) | gold answer in main text, table header predicted | "Result:training data" (33.3%), "Result:task" (33.3%), "Result:model" (33.3%), |
| | 11.7 (18) | partial match, but misses important details | "Result:test data" (100%) |
| | 6.5 (10) | gold answer in table caption, table header predicted | "Result:metric" (100%) |
| | 6.5 (10) | complete mismatch | "Result:experimental settings" (100%) |
| Propagated Errors | 3.2 (5) | select wrong record template | "Other:type" (100%) |

Table 3: Error analysis of INSTRUCTE (w/ GPT-4) for ML tables, inspecting 591 attribute predictions from 100 cell records sampled from 10 tables. For each fine-grained error type, we provide the error percentage, detailed error sources, and the top three affected attributes.

| | Model (GPU hours) | Token-Level $F_1$ | | | EM | | |
|---|---|---|---|---|---|---|---|
| | | P | R | $F_1$ | P | R | $F_1$ |
| Teacher | code-davinci-002 | 74.1 | 71.8 | 72.3 | 59.4 | 56.9 | 57.6 |
| Student | LLaMA-7B (50h) | 74.1 | 67.6 | 69.1 | 56.8 | 53.4 | 54.3 |
| | Alpaca-7B (50h) | 72.7 | 64.8 | 67.5 | 56.1 | 50.0 | 52.0 |
| | T5-11B (380h) | 75.8 | 71.4 | 73.2 | 60.3 | 56.7 | 58.1 |

Table 4: Experimental results for knowledge distillation on the ML tables. Student models are trained on 3,434 tables labeled by the teacher model. GPU hours refers to the training time ($\times$ number of GPUs) of student models for one epoch.

### 4.7 Leaderboards and Image Extraction

To further validate INSTRUCTE's practicality, we integrate it with multi-modal models, like GPT4-V, for extracting data from table images. In an initial study with ML tables, it yields a Table-$F_1$ of 70.2, approaching the 74.2 Table-$F_1$ achieved with the original text inputs. Additionally, we explore INSTRUCTE's application to the task of Leaderboard Extraction, where it shows competitive performance against leading supervised systems. Due to space constraints, details on these explorations are provided in Appendix F.

## 5 Related Work

**Table Understanding in NLP Research** Recently there have been many research efforts involving tables, particularly, table-to-text generation (Parikh et al., 2020; Wang et al., 2022; Hu et al., 2023). For example, ToTTo (Parikh et al., 2020) introduced the task of open-domain table-to-text generation. In contrast, our work transforms tables into structured JSON records, where a data schema is the only supervision provided.

**Pre-training on Semi-structured Data** TaPas (Herzig et al., 2020) and TaBERT (Yin et al., 2020) pre-train on linearized tables with a specialized cell index embedding. TABBIE (Iida et al., 2021) employs dual transformers for separate row and column encoding. Similarly, TabLLM (Hegselmann et al., 2023) uses general-purpose LLMs to process tables, but we focus on schema-driven IE rather than table classification or question answering.

**IE from Semi-structured Data** Information extraction from semi-structured data has gained increasing interest (Carlson and Schafer, 2008; Dong et al., 2020; Gupta et al., 2022; Lou et al., 2023). OpenCeres (Lockard et al., 2019) and ZeroShotCeres (Lockard et al., 2020) highlight open-domain extraction from web data, while AxCell (Kardas et al., 2020) and TDMS-IE (Hou et al., 2019) focus on leaderboard extraction from ML tables. DisCoMat (Gupta et al., 2022) showcases material composition extraction from scientific tables. Unlike most existing methods requiring supervised datasets for fine-tuning, our approach stands out by using LLMs to accurately extract data across various domains using an extraction schema.

## 6 Conclusion

This paper explores the capabilities of LLMs for extracting structured data from heterogeneous tables. We introduce a new task, Schema-Driven Information Extraction, which converts tables into structured records guided by a human-authored data schema. To facilitate this task, we present a benchmark, comprised of tables from four diverse domains, and evaluate various LLMs through our proposed method INSTRUCTE. The experiments reveal that while API-based models excel across domains and formats, open-source models display significant potential in specific areas. Moreover, we conduct detailed ablation studies and analyses to investigate the factors for model success, and validate the feasibility of building compact models through distillation to reduce dependency on APIs.

## Limitations

While INSTRUCTE showcases strong performance as an instruction-based prompting approach, it encounters specific challenges. Firstly, similar to other prompting methods, its performance could be sensitive to the phrasing of the prompt. Despite offering guidelines for crafting prompts in Appendix A, such as emphasizing clear attribute names, developing robust extraction schemas for new domains often relies on iterative experimentation. Future work could explore automatic prompt optimization (Zhou et al., 2023; Wang et al., 2024) to reduce the need for human trial-and-error. Additionally, the model's varying performance across different domains and formats is difficult to interpret, possibly due to biases in the pretraining corpus, a factor we cannot fully analyze due to the opaque nature of the pre-training process. InstrucTE also faces difficulties with dataset-specific nuances, as it operates on general task descriptions without detailed examples, making it challenging to navigate boundary cases effectively.

Beyond the model's inherent limitations, the availability of specific API-based backbones like GPT-4 and `code-davinci-002` may change, impacting reliance on these resources. To reduce this dependency, we include results from open-source models and investigate knowledge distillation as a viable alternative, showing promising results. Our benchmark aims to facilitate future research focused on enhancing smaller, openly accessible models, recognizing the importance of such developments for practical application and broader accessibility.

## Ethical Considerations

Our use of OpenAI's API-based models to distill open-source table extractors complies with OpenAI's terms of service, as we do not *"use the output from the Services to develop models that compete with OpenAI"*. Regarding licenses of four datasets in our SCHEMA-TO-JSON benchmark, the arXiv ML tables align with the licenses of their original papers. The PubMed Chemistry tables, sourced from the PMC Open Access Subset, conform to Creative Commons or equivalent licenses. For the other two datasets, we adapt pre-existing datasets released by the NLP research community, abiding by their respective original licenses.

## References


Taylor Berg-Kirkpatrick, David Burkett, and Dan Klein. 2012. An empirical investigation of statistical significance in NLP. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 995–1005, Jeju Island, Korea. Association for Computational Linguistics.

Michael J Cafarella, Alon Halevy, Daisy Zhe Wang, Eugene Wu, and Yang Zhang. 2008. Webtables: exploring the power of tables on the web. *Proceedings of the VLDB Endowment*, 1(1):538–549.

Andrew Carlson and Charles Schafer. 2008. Bootstrapping information extraction from semi-structured web pages. In *ECML/PKDD*, page 16.

Nathanael Chambers and Dan Jurafsky. 2011. Template-based information extraction without the templates. In *Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies*, pages 976–986.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Yunmo Chen, William Gantt, Tongfei Chen, Aaron Steven White, and Benjamin Van Durme. 2023. A unified view of evaluation metrics for structured prediction. *arXiv preprint arXiv:2310.13793*.

Xin Luna Dong, Hannaneh Hajishirzi, Colin Lockard, and Prashant Shiralkar. 2020. Multi-modal information extraction from text, semi-structured, and tabular data on the web. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts*, pages 23–26, Online. Association for Computational Linguistics.

Gregg B. Fields. 2019. The rebirth of matrix metalloproteinase inhibitors: Moving beyond the dogma. *Cells*, 8(9):984.

Tanishq Gupta, Mohd Zaki, N. M. Anoop Krishnan, and Mausam. 2022. Discomat: Distantly supervised composition extraction from tables in materials science articles.

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*, SIGIR '11, page 775–784, New York, NY, USA. Association for Computing Machinery.

Stefan Hegselmann, Alejandro Buendia, Hunter Lang, Monica Agrawal, Xiaoyi Jiang, and David Sontag. 2023. Tabllm: Few-shot classification of tabular data with large language models. In *Proceedings of*

*The 26th International Conference on Artificial Intelligence and Statistics*, volume 206 of *Proceedings of Machine Learning Research*, pages 5549–5581. PMLR.

Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Eisenschlos. 2020. TaPas: Weakly supervised table parsing via pre-training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4320–4333, Online. Association for Computational Linguistics.

Yufang Hou, Charles Jochim, Martin Gleize, Francesca Bonin, and Debasis Ganguly. 2019. Identification of tasks, datasets, evaluation metrics, and numeric scores for scientific leaderboards construction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5203–5213, Florence, Italy. Association for Computational Linguistics.

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. In *International Conference on Learning Representations*.

Hanxu Hu, Yunqing Liu, Zhongyi Yu, and Laura Perez-beltrachini. 2023. Improving user controlled table-to-text generation robustness. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 2272–2279, Dubrovnik, Croatia. Association for Computational Linguistics.

Hiroshi Iida, Dung Thai, Varun Manjunatha, and Mohit Iyyer. 2021. Tabbie: Pretrained representations of tabular data. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3446–3456.

Mohit Iyyer, Wen-tau Yih, and Ming-Wei Chang. 2017. Search-based neural structured learning for sequential question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1821–1831.

Junmo Kang, Wei Xu, and Alan Ritter. 2023. Distill or annotate? cost-efficient fine-tuning of compact models. *Proceedings of ACL*.

Marcin Kardas, Piotr Czapla, Pontus Stenetorp, Sebastian Ruder, Sebastian Riedel, Ross Taylor, and Robert Stojnic. 2020. AxCell: Automatic extraction of results from machine learning papers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8580–8594, Online. Association for Computational Linguistics.

Kyungdoc Kim, Seokho Kang, Jiho Yoo, Youngchun Kwon, Youngmin Nam, Dongseon Lee, Inkoo Kim, Youn-Suk Choi, Yongsik Jung, Sangmo Kim, et al. 2018. Deep-learning-based inverse design model for intelligent discovery of organic molecules. *npj Computational Materials*.

Hugo Laurençon, Lucile Saulnier, Léo Tronchon, Stas Bekman, Amanpreet Singh, Anton Lozhkov, Thomas Wang, Siddharth Karamcheti, Alexander M. Rush, Douwe Kiela, Matthieu Cord, and Victor Sanh. 2023. Obelics: An open web-scale filtered dataset of interleaved image-text documents.

Nghia T. Le, Fan Bai, and Alan Ritter. 2022. Few-shot anaphora resolution in scientific protocols via mixtures of in-context experts. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2693–2706, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural text generation from structured data with application to the biography domain. *arXiv preprint arXiv:1603.07771*.

Junlong Li, Yiheng Xu, Lei Cui, and Furu Wei. 2022. MarkupLM: Pre-training of text and markup language for visually rich document understanding. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6078–6087, Dublin, Ireland. Association for Computational Linguistics.

Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, et al. 2023. Starcoder: may the source be with you! *arXiv preprint arXiv:2305.06161*.

Bill Yuchen Lin, Ying Sheng, Nguyen Vo, and Sandeep Tata. 2020. Freedom: A transferable neural architecture for structured information extraction on web documents. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '20, page 1092–1102, New York, NY, USA. Association for Computing Machinery.

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)*, pages 3047–3056.

Colin Lockard, Prashant Shiralkar, Xin Luna Dong, and Hannaneh Hajishirzi. 2020. ZeroShotCeres: Zero-shot relation extraction from semi-structured webpages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8105–8117, Online. Association for Computational Linguistics.

Yuze Lou, Bailey Kuehl, Erin Bransom, Sergey Feldman, Aakanksha Naik, and Doug Downey. 2023. S2abel: A dataset for entity linking from scientific tables.

OpenAI. 2023. Gpt-4 technical report. *ArXiv*, abs/2303.08774.

Ankur Parikh, Xuezhi Wang, Sebastian Gehrmann, Manaal Faruqui, Bhuwan Dhingra, Diyi Yang, and Dipanjan Das. 2020. Totto: A controlled table-to-text generation dataset. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1173–1186.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. 2023. Code llama: Open foundation models for code.

Jonathan M. Stokes, Kevin Yang, Kyle Swanson, Wengong Jin, Andres Cubillos-Ruiz, Nina M. Donghia, Craig R. MacNair, Shawn French, Lindsey A. Carfrae, Zohar Bloom-Ackermann, Victoria M. Tran, Anush Chiappino-Pepe, Ahmed H. Badran, Ian W. Andrews, Emma J. Chory, George M. Church, Eric D. Brown, Tommi S. Jaakkola, Regina Barzilay, and James J. Collins. 2020. A deep learning approach to antibiotic discovery. *Cell*, 180(4):688–702.e13.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023a. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023b. Llama 2: Open foundation and fine-tuned chat models.

Fei Wang, Zhewei Xu, Pedro Szekely, and Muhao Chen. 2022. Robust (controlled) table-to-text generation with structure-aware equivariance learning. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5037–5048, Seattle, United States. Association for Computational Linguistics.

Xinyuan Wang, Chenxi Li, Zhen Wang, Fan Bai, Haotian Luo, Jiayou Zhang, Nebojsa Jojic, Eric Xing, and Zhiting Hu. 2024. Promptagent: Strategic planning with language models enables expert-level prompt optimization. In *The Twelfth International Conference on Learning Representations*.

Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. 2020. Tabert: Pretraining for joint understanding of textual and tabular data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8413–8426.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

Yichao Zhou, Ying Sheng, Nguyen Vo, Nick Edmonds, and Sandeep Tata. 2022. Learning transferable node representations for attribute extraction from web documents. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, WSDM '22, page 1479–1487, New York, NY, USA. Association for Computing Machinery.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. 2023. Large language models are human-level prompt engineers. In *The Eleventh International Conference on Learning Representations*.

## A INSTRUCTE

**Prompt Formulation** Our proposed prompt consists of four components: 1) "Input Table (w/ supp. text)" includes the table source code paired with supplementary text from the document; 2) "Extraction Schema" defines the JSON formats for extracted records, encompassing the record type, attribute names, and associated data types; 3) "Task-specific Instructions" outline the task execution process, addressing both the extraction process from individual cells and the traversal strategy across cells, such as *"left-right, top-down"*; 4) "Initial Record" is used to jump-start the prompting process, including the partial record of the first cell.

For "Input Table (w/ supp. text)", we employ the BM25 algorithm to retrieve the most relevant paragraphs for each table. For "Extraction Schema", we propose two guidelines for schema design: 1) Attribute names should be specific, which decreases the probability of the model generating incorrect attributes, or hallucinations. For instance, when extracting relevant attributes about a movie from a movie webpage, it's advisable to use specific terms such as "`movie name`" or "`director name`", rather than the generic "`name`"; 2) Attributes should be strategically ordered, placing simpler attributes ahead of more complex ones as errors in preceding attributes can adversely affect the prediction of subsequent ones due to the autoaggressive nature of LMs. The exact INSTRUCTE prompts used in our experiments are shown in Table 6 and Table 7.

**Cell Detector** We develop a rule-based method to identify numeric cells for both the ML and chemistry tables. Specifically, for the ML tables, we use the row separator "\\" and the column separator "&" to divide the table into cells. We then loop over each cell, checking for numeric values after stripping away any stylized text. In cases, where a cell contains multiple numeric values, such as "$0 \pm 0$", we consistently choose the first numeric value. For the chemistry tables, the parsing process is more straightforward, owing to the structured XML format of the table. Here, we iterate over each cell, verifying if it contains a numeric value once stylized text has been removed. The performance of our rule-based cell detector on two datasets is presented in Table 5. In the case of DISCOMAT, we use the cell detector provided by the original paper Gupta et al. (2022).

| Dataset | Split | P | R | $F_1$ |
|---|---|---|---|---|
| ML Tables | Dev | 100.0 | 97.0 | 98.0 |
| | Test | 99.9 | 99.6 | 99.7 |
| Chem. Tables | Dev | 100.0 | 100.0 | 100.0 |
| | Test | 100.0 | 98.3 | 99.2 |

Table 5: Results of (numeric) cell detection on ML and chemistry tables.

## B arXiv Machine Learning Tables

**Extraction Attributes** We design a set of extraction attributes for each of the three primary types of numeric cells in ML tables: "Result", "Hyper-parameter", and "Data Statistics". These attributes are outlined in detail below.

- "Result" includes seven attributes: `training data`, `test data`, `task`, `metric`, `model`, `model settings` and `experimental settings`. The first five attributes are fixed, with answers being text spans in the paper. The last two attributes, `model settings` and `experimental settings`, are free-form attributes, with answers being JSON objects. For example, the `experimental settings` attribute may be *{"number of training examples": "0"}* for a zero-shot setting. This scheme is more detailed than previous approaches (Hou et al., 2019; Kardas et al., 2020) and can accommodate a broader range of ML paradigms and provide more granular information.
- "Hyper-parameter" includes optimization parameters like learning rate and batch size, as well as numeric descriptions of model architectures such as layer count. The three fixed attributes for this category are: `model`, `parameter/architecture`, and `dataset`.
- "Data Stat." covers four attributes: `dataset`, `dataset attribute`, `sub-set/group`, and `dataset features`. The `sub-set/group` specifies a dataset subset (e.g., "train" or "test"), while `dataset features`, a free-form attribute, captures various dataset characteristics like the language or domain.

**Annotation Process** We sample 10 papers from each of three pertinent arXiv fields: Machine Learning, Computer Vision, and Natural Language Processing. After removing papers without LaTeX source code or any tables, a total of 25 papers

| Dataset | Full Prompt |
|---|---|
| ML Tables | [Retrieve paragraphs]<br><br>[Input table]<br><br>Here are JSON templates for four types of numeric cells: “Other”, “Result”, “Data Stat.”, and “Hyper-parameter/Architecture”:<br>{“value”: “xx”, “type”: “Other”}<br>{“value”: “xx”, “type”: “Result”, “task”: “xx”, “metric”: “xx”, “training data/set”: “xx”, “test data/set”: “xx”, “model/method”: “xx”, “model/method settings”: {“xx”: “yy”}, “experimental settings”: {“xx”: “yy”}}<br>{“value”: “xx”, “type”: “Data Stat.”, “dataset”: “xx”, “attribute name”: “xx”, “sub-set/group name”: “xx”, “dataset features”: {“xx”: “yy”}}<br>{“value”: “xx”, “type”: “Hyper-parameter/Architecture”, “model”: “xx”, “parameter/architecture name”: “xx”, “dataset”: “xx”}<br><br>Please describe all numeric cells in the above latex table following the JSON templates (proceeding by row in a left-right, top-down direction). For each cell, output one JSON description per line. For any unanswerable attributes in the templates, set their value to the placeholder “xx” if it is of string type and {“xx”: “yy”} if it is of dictionary type.<br><br>Cell Description:<br>{“value”: “[Query cell]”, “type”: |
| Chem. Tables | [Input table]<br><br>Here are JSON templates for six types of numeric cells: “Other”, “IC50”, “EC50”, “CC50”, “MIC”, and “GI50”:<br>{“value”: “xx”, “type”: “Other”}<br>{“value”: “xx”, “type”: “IC50”, “unit”: “xx”, “treatment compound”: “xx”, “target compound”: “xx”}<br>{“value”: “xx”, “type”: “EC50”, “unit”: “xx”, “treatment compound”: “xx”, “target compound”: “xx”}<br>{“value”: “xx”, “type”: “CC50”, “unit”: “xx”, “treatment compound”: “xx”, “target compound”: “xx”}<br>{“value”: “xx”, “type”: “MIC”, “unit”: “xx”, “treatment compound”: “xx”, “target compound”: “xx”}<br>{“value”: “xx”, “type”: “GI50”, “unit”: “xx”, “treatment compound”: “xx”, “target compound”: “xx”}<br><br>Please describe all numeric cells in the above XML table following the JSON templates (proceeding by row in a left-right, top-down direction). For each cell, output one JSON description per line. For any unanswerable attributes in the templates, set their value to the placeholder “xx”.<br><br>Cell Description:<br>{“value”: “[Query cell]”, “type”: |

Table 6: INSTRUCTE prompts used for ML and chemistry tables.

are covered in our dataset. To optimize the annotation budget and the dataset diversity, we cap the number of annotated tables to five per paper. Recognizing the domain-specific expertise needed, we employ expert annotators with backgrounds in ML research, who are provided with tables in both LaTeX and PDF formats and encouraged to thoroughly read the paper before annotation. The annotation process comprises two steps: 1) identifying the numeric cells and their record types, and 2) filling in the slots of pre-determined attributes, forming a JSON record with keys as attribute names and values as extracted content, in a text editor. Consequently, the dataset contains 122 tables, with 3,792 cells and 21K attributes annotated.

## C Evaluation Metrics

Comparing an LLM-predicted JSON object with a gold JSON object is a non-trivial task, as those generative LLMs may produce text spans that do not exactly exist in the input table. Consequently, we devote substantial effort to examining various metrics to determine the one best suited for our task using ML tables. Here, we consider three metrics: the standard token-level $F_1$ to capture the level of lexical overlap between the predicted and

| Dataset | Full Prompt |
|---|---|
| DISCOMAT | [Input table]<br><br>Here are JSON templates for two types of numeric cells: "Other" and "Glass_Compound_Amount":<br>{"value": "xx", "type": "Other"}<br>{"value": "xx", "type": "Glass_Compound_Amount", "constituent compound name": "xx", "unit": "xx", "glass material/sample name/id/code": "xx"}<br><br>Please describe all numeric cells in the above table following the JSON templates (proceeding by row in a left-right, top-down direction). For each cell, output one JSON description per line. For any unanswerable attributes in the templates, set their value to the placeholder "xx".<br><br>Cell Description:<br>{"value": "[Query cell]", "type": |
| SWDE-auto | [Input webpage]<br><br>Here is the JSON template for automobile attribute extraction:<br>{"webpage title": "xx", "automobile model (year)": "xx", "price": "xx", "engine type": "xx", "fuel economy": "xx"}<br><br>Please extract the automobile's attributes from the HTML code above following the JSON template. For any unanswerable attributes in the template, set their value to the placeholder "<NULL>".<br>{"webpage title": "[webpage title]", "automobile model (year)": |

Table 7: INSTRUCTE prompts used for DISCOMAT and SWDE. For SWDE, we use the "Auto" vertical as an illustrative example, and the prompts for other verticals differ only in attribute names (refer to Table 8 for the attributes of each vertical).

| Vertical | # Sites | # Pages | Attributes |
|---|---|---|---|
| Auto | 10 | 17,923 | `model, price, engine, fuel-economy` |
| Book | 10 | 20,000 | `title, author, ISBN-13, publisher, publish-date` |
| Camera | 10 | 5,258 | `model, price, manufacturer` |
| Job | 10 | 20,000 | `title, company, location, date` |
| Movie | 10 | 20,000 | `title, director, genre, rating` |
| NBA Player | 10 | 4,405 | `name, team, height, weight` |
| Restaurant | 10 | 20,000 | `name, address, phone, cuisine` |
| University | 10 | 16,705 | `name, phone, website, type` |

Table 8: SWDE statistics.

| Dataset | # Records | # Attr. | Precision | IAA |
|---|---|---|---|---|
| ML Tables | 100 | 591 | 73.9 | 95.7 |
| Chem. Tables | 100 | 380 | 95.3 | 100 |
| DISCOMAT | 100 | 201 | 92.5 | 99.4 |
| SWDE | 160 | 640 | 96.4 | 98.2 |

Table 9: Statistics and results of attribute-level human evaluation on four datasets. The inter-annotator agreement score (IAA) is calculated as the F1 score between the two annotations

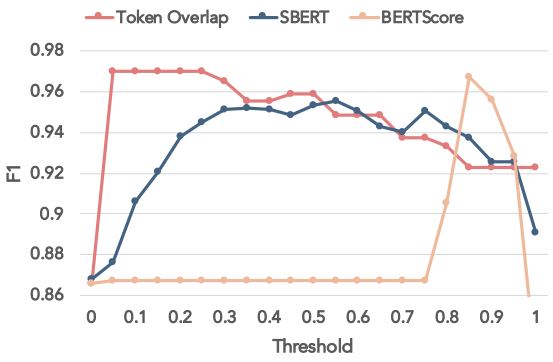


Figure 5: Results of comparing various metrics, including token-level $F_1$, SBERT, and BERTScore, to human judgment over different thresholds on ML tables. Numbers are computed over 677 sampled attributes that are paired with respective gold references.

gold attributes, and two semantic similarity metrics, SBERT (Reimers and Gurevych, 2019) and BERTScore (Zhang et al., 2020), to identify semantically similar expressions (e.g., # params vs. the number of parameters).

**Meta Evaluation** To assess how accurate each metric is compared to human evaluation, we manually annotated predicted-gold attribute pairs as to whether or not each pair matches. We consider a given pair to "match" if they are semantically equivalent, meaning they can be used interchangeably. For attributes that encapsulated multiple sub-attributes, we consider a pair to match if at least half of the sub-attributes are matched (i.e., $F_1$ score

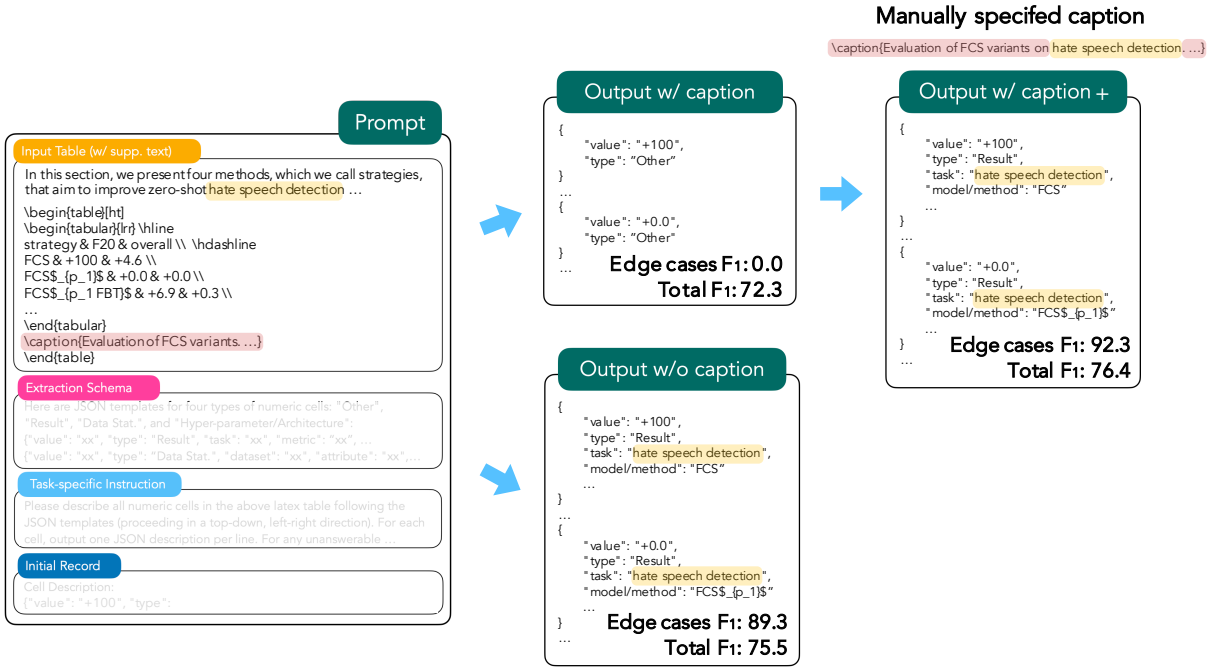


Figure 6: An error analysis of edge cases in which the predictions made by INSTRUCTE with captions default to "Other" (resulting in an 0 $F_1$). Our hypothesis that this issue may stem from the caption's lack of specificity is tested by manually expanding the caption (displayed on the right). This amendment significantly improves the performance on these edge cases, increasing the $F_1$ score to 92.3.

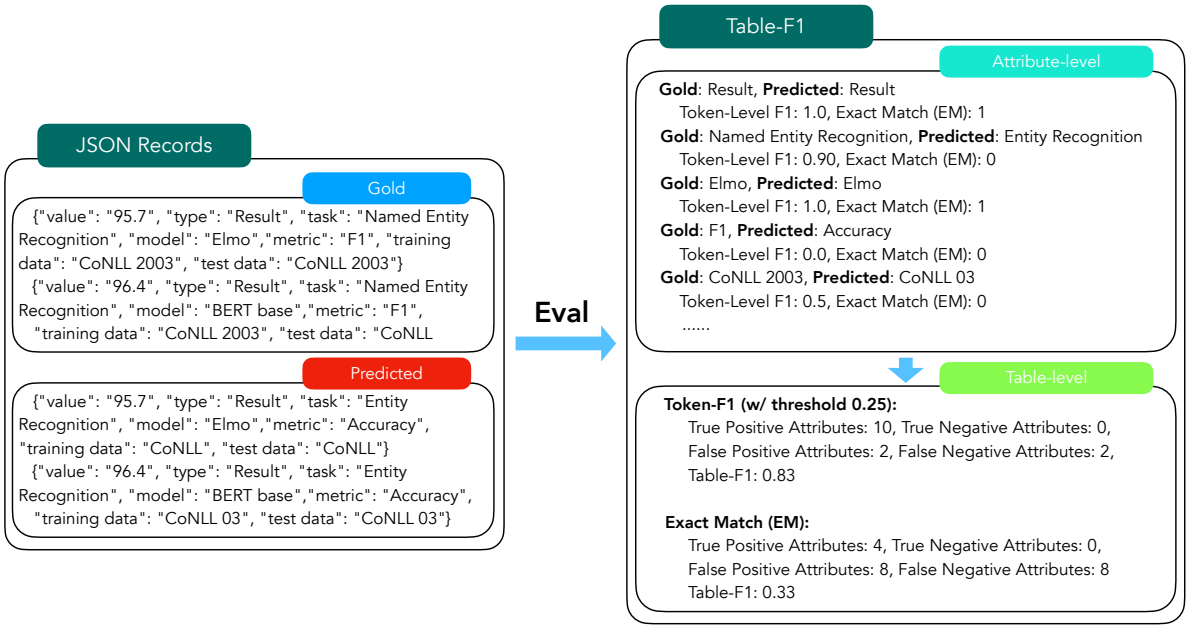


Figure 7: An example of Table-$F_1$ calculation, where two predicted records are compared against the two gold records.

$\geq 0.5$), with the decision for each sub-attribute being based on the same as in the text-span attributes. For the set of pairs to annotate and use as a test set, we sample a total of 100 cell pairs (i.e., 677 attribute pairs) according to the following process: 1) we first uniformly sample a table from the development set (containing 10 papers); and 2) we then sample a random cell from the table, ensuring there were no duplicate cells. For each pair of predicted-gold attributes, each metric's decision (1 or 0) is made using a specific threshold. For example, if the token-level $F_1$'s score for paired attributes is 0.4

| | token-level $F_1$ | SBERT | BERTScore |
|---|---|---|---|
| Meta Eval. $F_1$ | 97.0 | 95.6 | 96.7 |
| Threshold | 0.25 | 0.55 | 0.85 |

Table 10: Results of comparing various metrics, including token-level $F_1$, SBERT, and BERTScore, to human judgment on ML tables. Numbers are computed over 677 sampled attributes that are paired with gold references. The highest achieved $F_1$ scores are displayed alongside the thresholds. A complete illustration of results, sorted by thresholds, can be found in Figure 5 in Appendix.

and the threshold is 0.5, then the decision would be 0, indicating no match. The decisions over the test set containing 677 attribute pairs are then compared to human evaluation. In this binary classification problem, $F_1$ is used to evaluate the performance of the metrics.

In Table 10, we present the performances of each metric with the optimal threshold for each. Surprisingly, we find that the token-level $F_1$ (with a threshold of 0.25) decision aligns nearly perfectly with human judgment, and performs the best among all metrics for our task. This might suggest that discerning subtle differences is more crucial than identifying different phrases with the same meaning for this task. Based on these empirical findings, we opt for the token-level $F_1$ for automatic evaluation at the attribute level. This choice is highly desirable not only because of its high accuracy but also due to its simplicity.

## D Implementation Details

Considering the lengthy source code for tables, we employ different strategies to encode the input table and perform Schema-Driven IE, based on the context length of the chosen LLM. For LLMs with a larger context length, such as GPT-4, `code-davinci-002`, and CodeLlama, we input the full table and conduct the proposed error recovery process. For LLMs with a more limited context length, such as LLaMA and T5-11B, we query each target cell individually. The input table is condensed by rows, retaining the first two rows, typically containing headers, and the row with the query cell, with the token `<select>` pinpointing the position of the query cell. We use greedy decoding to maximize the reproducibility of our results.

For the TableQA setting, we divide the problem into two steps: selecting the record type and predicting the relevant attributes. For T5 and Flan-T5, the first step is modeled as a multi-choice QA problem, where the model chooses the type of the query cell from a list of provided options. The second step is modeled as an extractive QA task, asking the model to pinpoint the answer spans for the attributes associated with the selected type. For TaPas, the initial step is treated as a classification problem, whereas the latter one is handled as a cell selection problem. The hyper-parameters used for fine-tuning T5 and TaPas are presented in Table 11.

| | T5 (11B) | TaPas |
|---|---|---|
| learning rate | 1e-4 | 5e-5 |
| batch size | 8 | 32 |
| # epoches | 5 | 10 |

Table 11: Hyper-parameters used for fine-tuning T5 and TaPas.

## E Error Analysis of Caption

In Section 4.4, we observe an unexpected finding that table captions do not enhance performance, but rather seem to detract from it, which is counterintuitive. To delve deeper into this observation, we conduct an error analysis. This involves comparing the performances of our INSTRUCTE system with and without captions at the table level. This analysis uncovers a few outliers (3 out of 68) where including a caption leads to a 0 $F_1$ score, whereas the score is near perfect when the caption is excluded. For instance, as depicted in Figure 6, the predictions all fall into the "Other" category when a caption is included, leading to a 0 $F_1$ score in these outlier instances. Conversely, removing the caption results in an $F_1$ score of 89.3. This high score is due to the fact that retrieved paragraphs provide ample contextual information (e.g., "hate speech detection") without the presence of a caption.

We hypothesize that the model's inclination to predict "Other" in the presence of a caption may be a consequence of the captions' lack of specificity with respect to the attributes relevant to the table cells (for example, "hate speech detection"). This lack of explicit, relevant details could create confusion in associating the caption with the retrieved paragraphs, thereby misleading the model. To test our hypothesis, we manually adjust the captions to include more specific attributes, such as "hate speech detection" and "T5-Base." As a result, we observe an improvement in the model's performance with the revised caption, with the total $F_1$ score even exceeding that achieved without a cap-

tion. This outcome partially supports our hypothesis and suggests that carefully crafted captions could indeed be beneficial, aligning with our initial expectations. However, this investigation also points to the fact that the model currently lacks robustness in handling these outlier scenarios.

# F Extracting Leaderboards from Table Images

## F.1 Extraction from Table Images

One practical challenge with INSTRUCTE is the need for tables in a textual format, while many tables are available only as PDFs or images. To address this, we integrate INSTRUCTE with multi-modal models to extract structured data from table images. Specifically, we experiment with two strategies: 1) direct extraction from table images, and 2) a pipeline that first employs multi-modal models to transform table images into text, and then run INSTRUCTE on the textual tables.

In a preliminary study with ML tables, we use GPT-4V as the backbone for INSTRUCTE. We find that the pipeline method yields a Table-$F_1$ score of 70.2 from image inputs, approaching the 74.2 Table-$F_1$ achieved with the original text inputs. It outperforms direct extraction using GPT-4V, which attains only a Table-$F_1$ score of 46.4, as the pipeline can capitalize on INSTRUCTE's error recovery capabilities, resulting in more thorough and accurate extractions.

Additionally, we test `IDEFICS-80b-instruct` (Laurençon et al., 2023), a leading open-source multi-modal model, which unfortunately could not perform the table-text conversion or direct extraction.[14] This suggests a clear avenue for future research to enhance multi-modal models' ability to accurately process image-based tables.

## F.2 Leaderboard Extraction from ML Papers

**Task Definition & SOTA Methods** The task of leaderboard extraction (Hou et al., 2019; Kardas et al., 2020) entails extracting leaderboard tuples (`task`, `dataset`, `metric`, `score`) from tables in ML papers. Unlike our proposed Schema-Driven IE, which requires open-domain span identification, leaderboard extraction presumes prior knowledge of all leaderboards, represented as pre-defined (`task`, `dataset`, `metric`) tuples, and centers on linking numeric cells to these leaderboards.

The state-of-the-art leaderboard extraction method, AXCELL (Kardas et al., 2020), is a comprehensive pipeline system comprising four components: Table Type Classification, Table Segmentation, Cell Linking, and Filtering. For each component, except the last one, AXCELL employs a supervised model. It starts with table type classification to identify result-related tables, which are then passed to the table segmenter responsible for annotating the header cells of the table. Following this step, a retrieval model links numeric cells in the table to pre-defined leaderboards using human-engineered features. Lastly, AXCELL filters and selects the best record based on the leaderboard taxonomy criteria, such as retaining higher values for "Accuracy" and lower ones for "error rate".

**Application of INSTRUCTE** To extract leaderboards from an ML paper, we consider all tables that contain numeric cells, instead of selecting tables via a trained classifier as in AXCELL. For each table, we run INSTRUCTE using a customized leaderboard extraction JSON template. This template resembles the ML-table template with two additional fixed attributes: `eval split` and `eval class` in the "Result" cell template. We add the `eval split` attribute because the evaluated split is essential information for this task; for instance, *"dev $F_1$"* and *"test $F_1$"* are treated as different metrics in the leaderboard taxonomy. The `eval class` attribute is used to exclude sub-set or sub-class results that are typically present in analysis tables. After generating all predicted cell descriptions, we filter them based on three criteria: 1) the `type` attribute must be *"Result"*; 2) the `eval class` attribute must be *"all"* or *"Null"* as observed on the development set; and 3) the cell must be bolded in the table, as this usually indicates its superior performance and possible relevance to the leaderboard. For papers without any bolded cells, we experiment with two strategies: 1) include all the remaining cells in the table that meet the first two criteria; 2) use cells selected by AXCELL, as its engineered features for cell selection may be useful. This hybrid system is referred to as INSTRUCTE+. We then use the predicted `task`, `dataset`, and `metric` attributes in each JSON record to match with the pre-defined leaderboards using token-level $F_1$, and we select the leaderboard with the highest average score over three attributes. Finally, follow-

[14] The `IDEFICS-80b-instruct` model either produces unrelated content or simply output "I am sorry, but I cannot generate LaTeX code from the table."

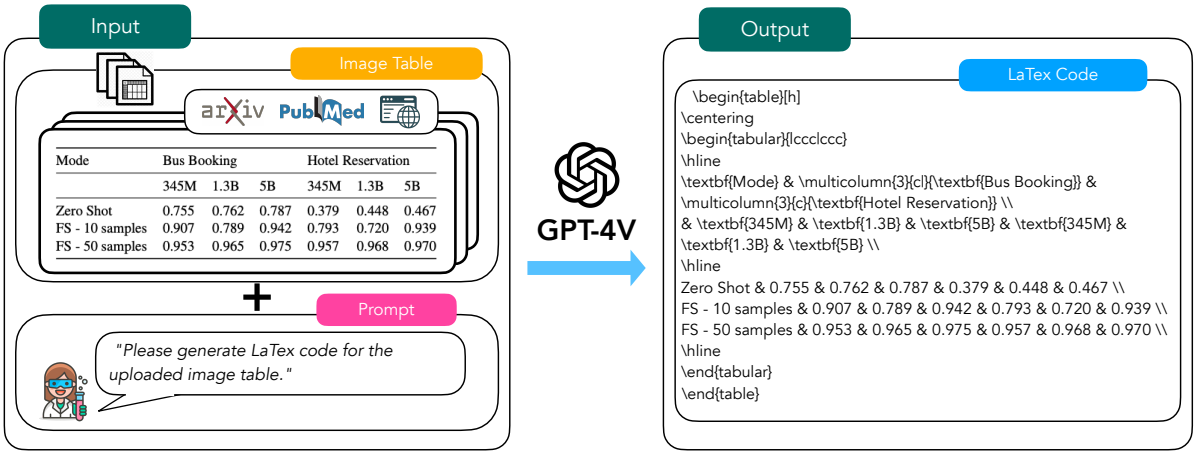


Figure 8: Generate LaTeX code for image tables using GPT-4V.

| Method | Micro-Average | | | Macro-Average | | |
|---|---|---|---|---|---|---|
| | P | R | $F_1$ | P | R | $F_1$ |
| AXCELL | **25.4** | 18.4 | 21.3 | **21.5** | 21.5 | 20.0 |
| INSTRUCTE | 20.1 | 20.8 | 20.5 | 20.3 | 23.1 | 19.6 |
| INSTRUCTE+ | 23.9 | **21.2** | **22.4** | 21.2 | **23.7** | **20.5** |

Table 12: Leaderboard extraction results on the PWC LEADERBOARDS dataset.

ing AXCELL, we choose the best record based on the leaderboard taxonomy criteria, e.g., retaining higher values for "Accuracy" and lower ones for "error rate".

**Results** We compare INSTRUCTE with AXCELL on PWC LEADERBOARDS (Kardas et al., 2020), the largest dataset for leaderboard extraction. For INSTRUCTE, we use `code-davinci-002` given its excellent performance on SCHEMA-TO-JSON. Table 12 presents the results of both methods. We can see that INSTRUCTE achieves competitive performance compared to the supervised AXCELL, highlighting the efficacy of our proposed approach. When we enhance INSTRUCTE with AXCELL's cell selection capabilities to create INSTRUCTE+, it outperforms AXCELL, demonstrating the promising potential of combining these two approaches.