ArcheType: A Novel Framework for Open-Source Column Type Annotation using Large Language Models

Benjamin Feuer
New York University
bf996@nyu.edu

Yurong Liu
New York University
yurong.liu@nyu.edu

Chinmay Hegde
New York University
chinmay.h@nyu.edu

Juliana Freire
New York University
juliana.freire@nyu.edu

ABSTRACT

Existing deep-learning approaches to semantic column type annotation (CTA) have important shortcomings: they rely on semantic types which are fixed at training time; require a large number of training samples per type; incur high run-time inference costs; and their performance can degrade when evaluated on novel datasets, even when types remain constant. Large language models have exhibited strong zero-shot classification performance on a wide range of tasks and in this paper we explore their use for CTA. We introduce ArcheType, a simple, practical method for context sampling, prompt serialization, model querying, and label remapping, which enables large language models to solve CTA problems in a fully zero-shot manner. We ablate each component of our method separately, and establish that improvements to context sampling and label remapping provide the most consistent gains. ArcheType establishes a new state-of-the-art performance on zero-shot CTA benchmarks (including three new domain-specific benchmarks which we release along with this paper), and when used in conjunction with classical CTA techniques, it outperforms a SOTA DoDuo model on the fine-tuned SOTAB benchmark.

PVLDB Reference Format:
Benjamin Feuer, Yurong Liu, Chinmay Hegde, and Juliana Freire. ArcheType: A Novel Framework for Open-Source Column Type Annotation using Large Language Models. PVLDB, 17(9): 2279 - 2292, 2024.
doi:10.14778/3665844.3665857

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/penfever/ArcheType.

1 INTRODUCTION

The goal of semantic column type annotation (CTA) is to associate each column of a relational table with one among several pre-defined semantic types that go beyond atomic types such as string, integer, or Boolean. CTA is a useful computational primitive in numerous settings, including data cleaning, where detection, correction, and transformation are performed using rules based on data types [24, 40], and schema matching for data discovery, where the semantic type can be used to constrain the search for matching attributes [22, 26]. Beyond being useful from a computational standpoint, efficient methods for CTA can also enable democratization of access to large, well-curated datasets by reducing labeling costs.

Learning-Based CTA. Recent approaches to CTA have increasingly been based on learning-based techniques. Deep learning approaches rely on the availability of large training corpora of columns annotated with their semantic types to train a deep neural network from scratch that can perform CTA on new, unseen columns of relational tables [21, 53]. Fine-tuned models, on the other hand, rely on pre-trained transformer-based language models (LMs) such as BERT [51] and fine-tune them for the specific task of CTA [10, 45]. Learning-based approaches have been shown to be effective for identifying generic types for which there exists sufficient training data. However, these approaches exhibit important limitations. First and foremost, their performance degrades substantially when evaluated against test datasets that have been acquired from different sources *even when their column types match closely*. This problem is sometimes called *distribution shift* [38]. An important desideratum of deep learning models is that they exhibit predictable model behavior under natural distribution shifts, i.e., when evaluation data which differs from the data on which a model was trained due to natural factors. However, recent works show that the vast majority of standard deep models for image classification perform significantly worse under natural shifts [18, 29, 41].

We posit that the same phenomenon occurs in closed-set deep learning models for CTA. Suppose we fix a given column type `location` and that our pre-training distribution is sourced from NYC Open Data [32]. Then we might see entries like `Broadway`, `SoHo`, `Jamaica`, which are locations in New York City. But if we use this model to perform CTA on a dataset from the Brazilian Dados Abertos [16], it is unlikely to assign the `location` label to `Corcovado` and `Lapa`, which are locations in Rio de Janeiro. As a simple empirical validation of this problem, we compared the performance of the fine-tuned DoDuo CTA model [45], on the Schema.Org Table Annotation Benchmark (SOTAB) [28]. We use the DoDuo variant pretrained on the similar VizNet dataset [19], reusing CTA labels from that benchmark wherever possible. We find that performance declines over 60% (from 84.8% to 23.8%).

Even if existing models did not struggle under shift, their utility is still constrained by the fact that *label sets have to be specified at training time*. However, real-world data is vast, and pre-trained

Proceedings of the VLDB Endowment, Vol. 17, No. 9 ISSN 2150-8097.
doi:10.14778/3665844.3665857

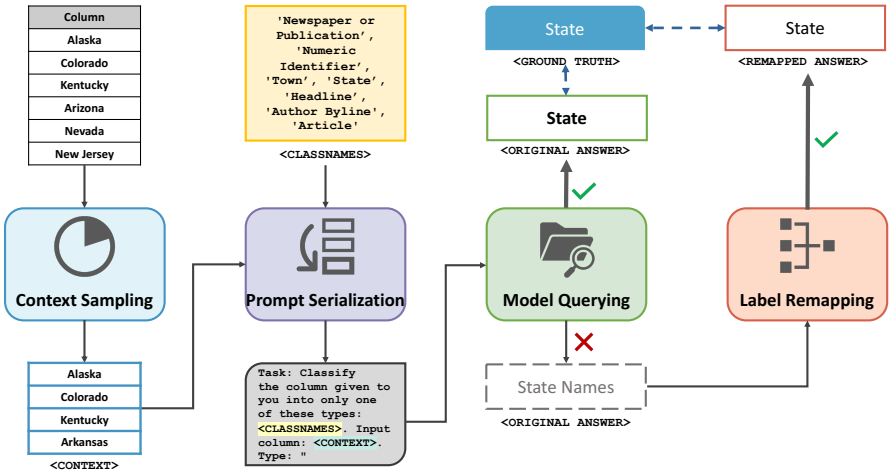

Figure 1: *ArcheType: a four-stage method for column type annotation.* *(1) In the Context Sampling stage, an algorithm selects a few representative samples from a column. (2) In the Prompt Serialization stage, the context and instruction string are serialized in a model-specific, token-efficient manner. (3) The prompt is input to a LLM in the Model Querying stage. (4) If the output of the LLM is not one of the allowable categories, the Label Remapping stage assigns the model output to a class.*

type labels rarely map cleanly to categories of interest in newly-encountered datasets; in many scenarios datasets do not have a schema which fit neatly into these pre-trained types.

Consider the NYC Open Data repository [32] which contains thousands of datasets published by NYC agencies and includes NYC-specific semantic types such as public schools, agencies, parks, and boroughs. As point of reference regarding the specificity of this collection, Ota et al. [34] computed the overlap between the contents of datasets in NYC Open Data and word vectors trained with GloVe (which uses Wikipedia as a source) and found that GloVe covers only 8% of the terms in the collection. We note that existing ontologies and taxonomies such as DBpedia [2] define generic types that encompass the NYC-specific types, gor example, a high school can be classified as `EducationalInstitution`. However, this type includes many institution types that are not public schools, such as colleges, medical centers and libraries. If we use this semantic type to find tables to augment information about NYC high schools, many irrelevant tables would be retrieved.

Further, *training a model to recognize new types is both time-consuming and costly* as it requires the acquisition of labeled data and the training of new deep models. This can severely limit the applicability of learning-based approaches [10, 21, 45] to long-tail and rare types, which can negatively affect downstream applications.

Moreover, the *volume of training data required by modern CTA models is substantial.* For example, Sherlock [21] was trained on over 675,000 columns retrieved from the VizNet corpus to recognize 78 semantic types from DBpedia [2]. Over 397,000 tables were used for training versions of the current state-of-the-art DoDuo [45]. This imposes high data cleaning and labeling costs which can be oppressive, particularly for infrequent classes.

Using LLMs for CTA. As a silver lining, the recent dramatic advances in generative large language models (LLMs) open the opportunity to address these challenges and create robust models for a broad set of semantic types without requiring large volumes of labeled data. LLMs are trained over a very large and diverse corpus and they are thus able to *accumulate knowledge that covers a plethora of semantic types.* Furthermore, these modes have the capability to perform *in-context learning*, where the label set can be specified as user-defined context during inference time, making it possible to *perform open-set classification even for rare types.* For example, when presented with the text `Stuyvesant`, GPT-3.5-Turbo learns in-context that it is being asked to perform classification and asserts it is a `High School in New York City`. This capability enables *zero-shot* CTA as well as the generation of labels that can be used to fine-tune models for domain-specific types. LLMs have also been shown to perform much better than other learning-based models under distribution shift [39], opening the possibility for the creation of robust CTA models.

Our contributions. In this paper, we take several steps towards establishing the effectiveness and limitations of LLMs for CTA. We discuss the challenges involved in using LLMs for CTA and systematically delineate the different components required to perform CTA using LLMs: sampling the data context, prompt serialization, model querying, and label remapping (illustrated in Fig. 1). We propose novel methods for these components and assess their effectiveness.

We also explore the impact of these components on two different modes of operation: (a) using existing LLMs for zero-shot CTA and (b) fine-tuning LLMs for CTA based on a training set of labeled column types. For both modes of operation, we report a series of results for *open-source LLMs.* As a basis of comparison we also study and report the performance of a closed-source LLM (the GPT family). However, we emphasize open-source LLMs in our work, since closed-source models are not transparent: since we do not know how they were constructed, it may be difficult to understand their behavior; and since closed-source LLMs are constantly updated, reported results cannot be reproduced [7].

We perform a detailed evaluation of our approach against state-of-the-art learning-based CTA systems [10, 21, 45] as well as a new zero-shot approach [25]. We use established benchmarks [6, 11, 19] and the SOTAB benchmark, which was designed for comparing the

performance of annotation systems on CTA tasks [28]. However, we observe that these benchmarks are primarily composed of well-known semantic types drawn from widely-used ontologies and taxonomies. To explore the breadth of LLM subject knowledge as well as how LLM-based CTA performs for a wide range of types (including rare, domain-specific types with novel characteristics), we also introduce three new benchmark datasets for CTA, described in Sec. 4.

Our main contributions can be summarized as follows:

(1) We introduce **ArcheType**, an open-source CTA framework centered around large language models, which leverages their strengths, adapts to their limitations, and is compatible with both open-source and closed-source LLMs.
(2) We enumerate four essential components for any LLM-based CTA (LLM-CTA) approach: sampling, serialization, querying, and label remapping. We propose new approaches for context sampling and label remapping, and demonstrate their importance to the overall accuracy of LLM-CTA (Sec. 3).
(3) We introduce three new zero-shot CTA benchmarks that cover a range of domain-specific schemas and attribute types (Sec. 4).
(4) Through a detailed experimental evaluation (Sec. 5), we show that ArcheType achieves strong fine-tuned performance and state-of-the-art zero-shot performance on a large and diverse suite of benchmarks, while requiring far less tabular data for both training and inference than existing methods (Sec. 5.2).

2 BACKGROUND: FOUNDATION MODELS

The term **foundation model** applies to large machine learning models that are pre-trained on vast amounts of raw data to capture a wide range of knowledge, and then fine-tuned on more specific tasks or datasets [3]. In the case of large language models (LLMs), the pre-training objective is autoregressive; the model is tasked with predicting the next word in a sequence based on the context provided by the preceding words. The scale of LLMs results in new emergent capabilities, and their effectiveness across a multitude of tasks incentivizes the use of foundation models as a starting point (or replacement) for fine-tuning task-specific models. However, this last step must be done with care since the defects of the foundation model are inherited by all the adapted models downstream [3].

2.1 LLMs and Tabular Data

The development of LLMs has largely been driven in the context of NLP tasks as question-answering, logical inference, and word disambiguation. Recent efforts based on instruction-following, such as [35] and [8], have demonstrated that fine-tuning foundational LLMs on a carefully curated corpus of prompt-response pairs is an effective strategy for more generic classification tasks. However, these approaches focus on natural language datasets that have small label sets, clean labels, and balanced classes.

There have been only a handful of attempts to apply LLMs to tasks that are germane to tabular data. Recently, Hegselmann et al. [17] proposed a LLM-based framework for few-shot classification of tabular data and experimented with different strategies to design the prompt. They showed that their approach can outperform state-of-the-art (SOTA) neural models both in the zero- and few-shot settings. Narayan et al. [31] outline a vision for leveraging LLMs for data management tasks and show that LLMs using few-shot and zero-shot approaches can achieve SOTA performance for entity matching, data imputation, and error detection.

Table 1: *Cost of CTA benchmarking with GPT.* *Approximate cost to perform CTA over the 15,040 column test set of the SOTAB dataset varying the table serialization* Method *(column for column-at-once or table for table-at-once); the number of context samples* #Smp. *drawn per column; the percentage* % k *of serialized prompts whose tokenized length is estimated to exceed a context window of size k.*

Method	# Smp.	% >1k	% >4k	% >16k	App. USD Cost
column	3	01.0%	00.1%	00.0%	$7.85
column	10	06.5%	00.3%	00.0%	$12.54
column	20	13.0%	01.3%	00.1%	$19.97
column	100	93.0%	15.1.%	01.0%	$90.38
column	1000	100.0%	100.0%	56.5%	$1,072.06
table	10	89.8%	58.8%	25.8%	$763.28

2.2 LLMs for Zero-Shot CTA

As discussed in Sec. 1, LLMs present new opportunities to derive robust models for CTA that can handle a broad set of classes at a much lower cost than existing learning-based methods. Two recent approaches have been proposed that leverage OpenAI's GPT for zero-shot and few-shot CTA [25, 27]. These methods do not require model training, and apply open-vocabulary labels either from parametric memory [25], or from options provided at test time [27].

The promise of such a direction is clear, but existing implementations have important limitations. Both [25, 27], which are to the best of our knowledge the only existing works on zero-shot CTA, rely on closed-source models (see discussion below). They also require access to the entire table at test time to achieve their best performance, which in practice can be expensive for private models. Tab. 1 demonstrates that the cost to evaluate the SOTAB test set (assuming sampling with replacement) scales poorly for table-at-once methods, and over 25% of the prompts exceed the maximum possible context window. The cost is also high for column-at-once methods when a large sample is used. Since these methods are highly sensitive to sample size, it is important to devise strategies that are sample-efficient. However, only simple random sampling and first-k-rows sampling methods have been explored for LLM-based CTA. Note that while these methods are costly on closed-source models, they can be impractical on open-source models, owing to their limited context windows.

A distinct line of inquiry studied by Tu et al. [50] treats CTA as one example of a family of matching tasks in data integration, and is able to perform zero-shot binary matching on CTA instances.

2.3 Open vs. Closed-Source LLMs

We consider a model **open-source** if, and only if, sufficient specifics of model design have been published to reproduce the architecture, checkpoints with pre-trained weights have been released and the contents of the pre-training corpus are available for inspection. The advantages of utilizing open-source models are explainability, reproducibility, and reduced cost, while the drawbacks are performance and limited context length.

Explainability. The architectures of most closed-source models are not known to the public; nor is it known how much prompt engineering and behind-the-scenes modification of the model output is being conducted. The specifics of the data on which these models

are trained is also unknown. These facts make it difficult to provide rigorous explanations for the behavior of closed-source models.
Reproducibility. As noted recently, results from closed-source models are non-reproducible, non-deterministic, and cannot be ablated with respect to the model architecture or dataset, all of which makes them unreliable for reproducible research [37, 42].
Cost. As closed-source models charge by the token, the cost incurred by any solution which relies on them can be considerable. Open-source models, by contrast, require computational resources to host and expertise to maintain.
Performance. As of this writing, the best open-source models underperform the best closed-source models across a wide range of benchmarks [4]. The causes of this performance gap are not fully understood, as large language models tend to exhibit unpredictable phase transitions as a function of scale. These transitions can lead to sudden leaps in performance on standard benchmarks [36].
Context length. The open-source large language models in common use at the time of this paper have context windows ranging from 512 to 2048 tokens [8, 48] (typically between 375 and 1500 words, if the string is English). If the string is in a different language or is largely numeric, however, the tokenization process tends to be approximately 2-4x times less efficient, since standard tokenization schemes employed by such models tend to handle unicode inefficiently [44]. Both phenomena are common in real-world tabular data. Closed-source models are less constrained (GPT-3.5 allows over 16,000 tokens at the time of this writing).

3 ARCHETYPE: METHODS AND SYSTEM

Formal Model of LLM-CTA. Consider a table T with t columns and r rows. We denote each column $C \in T$ as a function which maps row indices to strings; i.e., for $0 <= i < t$, we have $C_i : \mathbb{N} \rightarrow \Sigma_*$, where i is the column index. Here, Σ_* is the set of all possible strings, Σ_{C_i} is the set of all strings found in column C_i, $\Sigma_{C_i} \subset \Sigma_* \; \forall i$, with any individual string $\sigma \in \Sigma_{C_i}$. We make no further assumptions; C_i may include a column name, and T may contain an additional metadata field. However, neither of these properties is required to exist, and so we do not include them in our analysis. Many of our methods rely on a sample of *unique* values sampled from the column, $U_i := \texttt{unique}(|\Sigma_{C_i}|)$. We explore two LLM-based approaches for CTA: fine tuned and zero shot.

Definition 3.1 (Fine-tuned LLM-CTA). Let $L \subseteq \Sigma_*$ denote a label set; these are our column types to be annotated. Standard CTA assumes a fixed cardinality for this label set, indexed by a variable we call j.[1] Given the above definitions, we define fine-tuned single-label $CTA \subset T \times L$ as a relation between tables and labels:

$$\forall C, \exists l_j \mid (C_i, l_j) \in CTA \tag{1}$$

We seek a generative method $M : \Sigma_* \rightarrow \Sigma_*$ that comes closest to satisfying the following properties:

$$M(\sigma, L) \in L, \forall C \in T, M(\sigma, L) \in CTA \tag{2}$$

i.e., the model requires a single string as input and generates a label in L that correctly represents the type of C.

[1] In existing benchmarks, j can be anywhere from 10 to 300 [27]

Definition 3.2 (Zero-shot LLM-CTA). The definition of zero-shot LLM-CTA is identical to that of fine-tuned, except that: in a zero-shot setting, the number of rows r is presumed to be small enough to preclude the possibility of fine-tuning a model; L is chosen at test-time; and it is possible to define multiple values of L for one T.

3.1 Elements of LLM-CTA Methods

We observe that any LLM-CTA method must provide solutions to four problems: context sampling, prompt serialization, model querying, and label remapping. Individually, each is necessary for LLM-CTA; collectively, they are sufficient. By considering and ablating approaches to each of these problems separately, we designed ArcheType, a LLM-CTA framework which generalizes to a wide range of architectures, including popular open-source models. Fig. 1 provides an overview of ArcheType and in the remainder of this section, we describe its components in detail.

Context Sampling. As of this writing, all SOTA large language models (LLMs) are transformer-based [51]. By design, transformers have a hard scaling limit over which their dense attention can be applied, sometimes called a *context window*, W. Given a context C and a set of labels L, if $|C| + |L| > W$, a representative sample must be selected. From a practical standpoint, the context window sizes of contemporary LLMs are small enough that this event takes place quite frequently, e.g., [25] and [27] use simple random sampling and first-k-rows sampling, respectively. We introduce a new sampling method in Sec. 3.2 and provide ablation studies in Sec. 5.4.1.

Prompt Serialization. SOTA LLMs require prompts, or priors, to complete. Prompt serialization (or prompt engineering) is the process of transforming raw context into a prompt. Of the four components we consider here, this one has received the most attention in the existing literature; the methods introduced by [25, 27] are largely focused on improvements to prompt serialization. In Sec. 5.4.2, we ablate prompt serialization, independent of other components, and conclude prompt engineering should be treated as a hyperparameter rather than as a methodological contribution – we describe this approach in Sec. 3.3. When considering a range of model architectures, we find that any reasonable serialization method is about as likely to produce a good result as any other.

Model Querying. Model selection and querying is another important element of LLM-CTA. The method must correctly submit a query to some large language model(s) chosen in advance, and it must retrieve and process the response. This query may be processed on a local machine or via an API. This, too, has not been ablated in prior work. While future work may attempt to train a generative large language model from scratch specifically for this task, [25, 27] use GPT, and only GPT. As part of our study, we present ablations on architectures across a range of open-source models as well as GPT (Sec. 5.4.3) and find that no model dominates.

Label Remapping. All LLMs sometimes produce responses which do not match with any of the labels provided in the prompt, i.e., $\sigma_L \notin L$. Label remapping is a form of error correction which remaps an unbounded LLM output space to a limited set of labels. Kayali et al. [25] use an embedding-based method called anchoring to remap labels, whereas Korini and Bizer [27] use a dictionary lookup. As the latter approach is not compatible with zero-shot LLM-CTA, we ablate only the former approach, along with two other baselines,

and develop CONTAINS+RESAMPLE (Sec. 3.5), an algorithm which outperforms the baselines across model architectures. We ablate our choice of remapping method in Sec. 5.4.4.

3.2 Context Sampling

CTA approaches using deep learning face severe data requirement challenges in settings that require (very) large tables and open label sets. To address these challenges, we introduce a new approach which we call **context sampling** and outline in Algo. 1. Given the unique values of a target column U_i and a target sample size ϕ, we seek to construct the representative sample S that best summarizes the column. While it is possible in LLM-CTA to have ϕ vary by column, in this paper we consider the setting where ϕ is fixed in advance and consistent across all columns.

In the simplest case, we have $|U_i| \geq \phi$, and S is drawn without replacement from a distribution whose construction is described below. If $|U_i| < \phi$, then S is drawn with replacement instead.

In the fine-tuned setting, we find it is beneficial to add features to the context window, affecting both sampling and serialization. The features we utilize are described later in this section, and are sampled as described in Algo. 1. The context sample is then serialized and embedded into a prompt which is passed to the LLM, the format of which follows from recent works such as [30] and [8].

Context Sampling in ArcheType. The probability distribution over U_i from which we sample is weighted according to an importance function f. The probability of selecting an element σ from U_i under P_{U_i} is given by:

$$P(\sigma) = \frac{f(\sigma)}{\sum_{j \in U_i} f(\sigma_j)}.$$

We utilize two importance functions in ArcheType. For the American Stories (amstr) benchmark described in Sec. 4, we find that an importance function which prioritizes unique samples that include any target class name is most effective; $f(\sigma) = 1$ if, for any $l_j \in L$, $l_j \subset \sigma$, else $f(\sigma) = 0.1$. Note that this function does not require the ground truth label of any particular sample, only the entire label set, which is a required input for CTA.

For all other benchmarks, the importance function f is string length – our experiments showed that long strings lead to better results. One possible reason is that longer strings are more likely to contain useful information than shorter ones. While an extensive ablation of the choice of importance function is beyond the scope of this paper, we note that ArcheType users (subject matter experts) can define importance functions suitable for their applications.

There are challenges in the implementation of context sampling, including *low variance (degenerate) data* $U_i \ll o(1)$ and *high variance data* $U_i \gg \phi$. Each of these situations merits discussion.

High variance. In this case, helpful context may be lost in a limited sample. This phenomenon may explain why increasing the size of the context sample tends to improve model performance. However, the improvements are slight, suggesting an exponential scaling of data demands.

Low variance. CTA can easily become unsolvable for *low-variance* or, in the extreme case, *degenerate columns*. Consider a column C_d such that

$$\forall k \in U_i, \Sigma_{U_{i_k}} = \text{“0”}$$

Algorithm 1 ***Context sampling.*** *Given a table T, a valid probability function P, and optional additional features, produce a context sample S of the appropriate size. If $|U_i| \to \infty$, methods like [15] can be used to derive a finite-size U_i.*

procedure CONTEXT-SAMPLE($T, i, S, \phi, P, SS, TN, E$) ▷
T: A table, i, a target column index, S, a context sample to be returned, ϕ: A hyperparameter (number of samples), P: a valid probability function, SS: summary statistics, TN: table name, OC: other columns, E: extended context flag
  $U_i \leftarrow \text{UNIQUE}(T_i)$, $S \leftarrow \emptyset$
  if E **then** $S \leftarrow \text{SS}(T_i) + \text{TN}(T)$
  while $|S| \leq \phi$ **do**
    $S \leftarrow S + \sigma \sim P_{U_i}$ ▷ Drawn without replacement
  if $(|S| < \phi) \wedge E$ **then**
    for $j \in T, j \neq i$ **do**
      $S \leftarrow S + T_j[0]$
      if $|S| \geq \phi$ **then** BREAK
  else
    while $|S| \leq \phi$ **do**
      $S \leftarrow S + \sigma \sim P_{U_i}$ ▷ Drawn with replacement
  return S

and a label set `L = number, integer, quantity`. There exists no unique σ_{L_j} such that $CTA(C_d, L_j) = \sigma_{L_j}$.

In some cases, we find that incorporating additional metadata (such as the filename of the table) can help with the classification task, but in other cases, we found that it simply biases the LLM to parrot back portions of the input string.

Feature Selection. In context sampling, *feature selection* refers to what aspects of the original data we choose to include in the context. In all of our experiments, our first feature is the context sample itself (CS). We also experiment with including the *file name (FN)* of the table, used by [10], *summary statistics (SS)*, used by [21], and samples from *other columns (OC)*, used by [45].

Summary statistics (SS). SS feature selection proceeds as follows:

- We select statistics which support fast, accurate sketching.
- We select measures of center and spread which can provide additional information about missing column values.

The list of summary statistics included in our fine-tuned models was: standard deviation, average, mode, median, max, min. When the summary statistic is a floating-point value, we round it to two decimal places. When it is an integer, we exclude the decimal place. When all sampled values are numeric, the statistics are computed with respect to the individual *column values*. When any sampled value is non-numeric, the statistics are computed with respect to *column value lengths*.

We postulate that these statistics are useful because they help the model disambiguate between numeric column samples by preserving information about overall trends in the column. However, we focused on simple-to-calculate statistics and did not extensively ablate our choices; in future work we plan to explore this aspect.
Other columns. First, we take as many unique samples as are available from the target column. Then, we fill the remaining context length with an equal number of samples from each other column.

INSTRUCTION: Select the category which best matches the input.
INPUT: 'TABLE NAME: diaridegirona', SAMPLES: Partit: Armenia - Liechtenstein, Partit: Israel - Austria, Partit: Shakhtar Donetsk - Atalanta 'std: 4.76','mean: 27.52', 'mode: 25','median: 25.0'

CATEGORY: "sporting event" **Fine tuned**

INSTRUCTION: Select the option which best describes the input.
INPUT: 550mm, 608mm, 600mm, 520mm, 595mm OPTIONS: text, date, age, telephone, jobposting, currency, event, product, streetaddress, category, number, time, zipcode, person, url, gender, country, email, price, creativework, weight, language, boolean, company, organization, sportsteam, coordinates

ANSWER: number **Zero shot**

Figure 2: ***Examples of ArcheType fine-tuned (top) and zero-shot (bottom) prompting.***

We label samples from other columns with an index number in order to identify from which column they originated. Performing this improves fine-tuned performance, but has a negative effect on zero-shot performance; see Fig. 6. This is likely because the LLM cannot distinguish inter-column from intra-column values without the presence of learned special characters as provided in [10, 45].

3.3 Prompt Serialization

The *prompt serialization* stage transforms the context sample S into a prompt format suitable for querying an LLM; this includes modification of prompts that exceed the maximum allowable length of the context window and how to reformat the table.

Fig. 2 shows examples of prompts for both fine-tuned and zero-shot regimes of ArcheType. We style our fine-tuned prompt after the instruction-following method described in [46]. We treat the semantics of the *INSTRUCTION* field as a hyperparameter, and fix it at training time. The extended context includes the samples, the table name, and computed summary statistics including standard deviation, median and mode. In zero-shot, we again treat *INSTRUCTION* as a hyperparameter, sweeping over a space of possible semantic structures. *INPUT* is handled identically to fine-tuned. In zero-shot, the prompt also includes *OPTIONS*, or allowable column names, from which the model is expected to choose. The suffix *ANSWER:* cues the LLM to supply the label (in this case,"number").

The heuristic optimization of this process is sometimes referred to as *prompt engineering*, and is treated as an important contribution by existing zero-shot CTA methods [25, 27]. However, recent phenomenological studies of foundation models have raised significant doubts as to the near-term stability and long-term viability of prompt engineering as a method [43]. In fine-tuned ArcheType, we fix a single prompt serialization strategy, as the prompt is learned during the fine-tuning process and has little impact on the model output, as long as it is consistent. In zero-shot ArcheType, unlike

C: "For the following table column, select a schema.org type annotation from <CLASSNAMES>. Input column: <CONTEXT>. Output: "

K: "Answer the question based on the task below. If the question cannot be answered using the information provided, answer with "I don't know". Task: Classify the column given to you into only one of these types: <CLASSNAMES>. Input column: <CONTEXT>. Type: "

I: "Here is a column from a table: <CONTEXT>. Please select the class from that best describes the column, from the following options. Options: <CLASSNAMES> Response: "

S: "Pick the column's class. Column: <CONTEXT>. Classes: <CLASSNAMES>. Output: "

N: "Pick the column's class. I mean if you want to. It would be cool, I think. Anyway, give it a try, I guess? Here's the column itself! <CONTEXT>. And, um, here are some column names you could pick from ... <CLASSNAMES>. Ok, go ahead! "

B: "INSTRUCTION: Select the option | category which best describes | matches the input. INPUT: <CONTEXT> OPTION: <CLASSNAMES> ANSWER:"

Figure 3: ***Six prompt variations.*** *In zero-shot ArcheType, we treat prompting as a hyperparameter, and sweep over six distinct prompts, each chosen according to a conceptual serialization strategy. <CLASSNAMES> stands in for the label set, <CONTEXT> for the output of the context sampling step. We use two variants of the "B" prompt, with semantic differences denoted by "|".*

previous methods, we treat the choice of prompt as a hyperparameter. We provide experimental support for this idea in Sec. 5.4.2.

Serialization strategies. We explore six distinct serialization strategies, illustrated in Fig. 3. The strategies labeled "C" and "K" were proposed in [25] and [27], respectively. The remaining serialization strategies are designed to test the effect of varying prompt length, position, and tone; "N" adopts a casual, conversational tone and uses simple language, "I" inverts the position of prompt and context, compared to the other strategies, and "S" is designed to be as short as possible while remaining clear. Our "B" prompt is written in a technical and formal tone, similar to prompt "S", but more verbose. We use a minor variant of our "B" prompt in our fine-tuned experiments; the semantic differences are shown in [12].

Prompt Serialization in ArcheType Zero Shot (ZS). We have evaluated ArcheType ZS using all six prompts in [12]; we report performance on the best-performing configuration. Note that we

Algorithm 2 ***Fine-tuned ArcheType.*** *Fine-tuning procedure for ArcheType-LLAMA; the serialized prompts generated by ArcheType are tokenized and passed to the model in batches. The autoregressive objective during training is for the model to generate the appropriate class token, given the prompt.*

```
procedure FineTuneLM(M,D,H)▷ M, an LLM, D, a fine-tuning dataset, H, hyperparameters
    Tokenizer ← LoadTokenizer()        ▷ tokenizer for the LM
    D ← Tokenizer.Tokenize(D) ▷ Token. the fine-tuning data
    for epoch = 1, Hyperparameters.Epochs do
        for each B ∈ D do
            loss ← M.Forward(B) ▷ Compute the forward pass
            loss.backward()             ▷ Backpropagate the loss
            M ← optimizer.step(M)       ▷ Update parameters
    return Fine-tuned Model
```

include the label set L in the prompt. In order to simplify the label space further for open-source models, we attempt to detect using simple type testing whether all elements of the context are numeric; if so, we limit L to labels which are numeric (selecting which labels are exclusively numeric is a one-time optimization per dataset – on SOTAB-27, it required about five minutes).

Prompt Serialization in ArcheType Fine Tuned (FT). We follow the Alpaca instruction format described in [46] and omit the label set L to make more efficient use of the context window.

Column-at-once Serialization. Both [27] and [25] use *table-at-once* serialization; the entire table is presented to the LLM at inference time, and all columns in that table are classified together. ArcheType uses *column-at-once* serialization; only a single column to be classified is passed to the LLM. [27] provides ablation studies indicating that table-at-once outperforms column-at-once on their test set, a very small subset of SOTAB.

Table-at-once serialization, however, is impractical to implement on open-source models with small context windows, and inefficient in that it requires classification of *all columns*, whether or not the classes for all columns are required.

Handling Overflow. Using the length of each prompt, we produce a conservative estimate of whether the tokenized prompt might overflow the context window. If so, we tokenize the prompt, truncate it, add the classnames and response cue to the end of the prompt, and pass it through. Examples of serialized prompts can be found in Fig. 2.

3.4 Model Querying

The third stage of ArcheType involves passing the serialized prompt as input to the LLM, a process which we refer to as *model querying*. The key variable here is, naturally, the choice of model and, in the case of fine-tuned CTA, the approach to training said model.

Fine-Tuned Models. In the fine-tuning regime, our model is a LLAMA-7B, the smallest in a batch of LLMs from [48]. All models in the LLAMA family were pre-trained on the standard unsupervised language modeling task of next-token prediction, but had no instruction tuning as part of pre-training. In order to improve performance on instruction-following tasks, we apply the Alpaca method of [46] prior to applying ArcheType. See Algo. 2 for an overview of the fine-tuning procedure utilized to train our model. Fig. 2 contains an example of a single data point in the training set.

Our results for fine-tuning are reported using a fine-tuned LLAMA-7B trained on the SOTAB-full training dataset, using our context sampling and label remapping algorithms. Following [46], we fine-tune LLAMA-7B for 3 epochs, with a learning rate of 2e-5. Fine tuning took 8-12 hours on 4x A100-80GB GPUs.

Zero-Shot Models. In the zero-shot regime, we consider the recent open-source OPT-IML and LLAMA-2 models from [23, 49] as well as FLAN models introduced in [8, 47]. We also present results on the closed-source, private GPT family of models from OpenAI [35]. As zero-shot ArcheType is model-agnostic, we report results from the three best-performing architectures in our experiments (Tab. 4).

3.5 Label Remapping

The fourth stage of ArcheType is *label remapping*; mapping the generative output of the LLM to the space of allowed labels. A key drawback of using standard LLMs for classification tasks (based on instruction tuning alone) is that their outputs are not guaranteed to only belong to the provided label set. In our experiments, we found small decoder-only LLMs, such as LLAMA-7B, were particularly susceptible to this behavior.

Previous works such as [8] have proposed simply discarding all answers which are not an exact match for a label in the set, and measuring performance with respect to exact matches only. Another naïve solution is to simply map all non-matching answers to a default `null` class.

However, we find that such approaches tend to underrate what the model actually provides, particularly in the CTA context. Often, the LLM's 'best guess' can be reasonably remapped to an answer in the provided label set. Formally, we frame label remapping as a function $REMAP(\sigma_L) : \Sigma_* \rightarrow L$. In other words, the REMAP function is responsible for mapping arbitrary output strings (that are outputs of the LLM) to some specific label in the label set $\sigma_L \in L$. We explore multiple approaches, described below, and find that the optimal approach varies depending on the LLM and whether we are in a fine-tuned or zero-shot domain.

Remap-contains employs the simplest strategy of checking for intersections: $\forall L_j \in L, (\sigma \subseteq L_j \vee L_j \subseteq \sigma) \rightarrow (\sigma_L := L_j)$. In the case of multiple matches, we accept the longest match. This is computationally efficient but has a high rate of failure; it can therefore be used in conjunction with other label remapping strategies.

Remap-resample (Algo. 3) utilizes the probabilistic nature of LLM outputs. We fix a hyperparameter k setting both how many times we attempt the problem *and* how we adjust the hyperparameters on each subsequent call. The parameter k can be utilized as either an additive or a multiplicative factor; we find that additive k is suitable for adjusting top_p and repetition_penalty, while a multiplicative factor works well for temperature. For more details on these hyperparameters, please refer to [52].

Remap-similarity (Algo. 4) employs a similarity-search strategy. Using an encoder-only transformer model, the input σ is converted to a vector embedding v_σ, as are all the strings in L. $\forall j \in L$, we then compute the vector cosine similarity $\texttt{COSSIM}(v_\sigma, v_{L_j})$. The `ARGMAX` result becomes the model's predicted class. For our experiments, we used the S3Bert model introduced in [33]. This method has the advantage of always returning a solution. However, this solution may

Algorithm 3 ***Remap-resample.*** *Remap-resample calls the LLM up to k times with permuted hyperparameters in order to generate increasingly diverse responses.*

procedure REMAP(σ, σ_L, L, k) ▷ σ: A text string, σ_L, the label assigned to σ, L: A label set, k: A hyperparameter (number of retries)
  if $\sigma_L \in L$ **then**
    return σ_L
  for $i \leftarrow 1$ to k **do** $\sigma_L \leftarrow$ LLM(prompt, k) ▷ We call the LLM with k-permuted hyperparameters
    if $\sigma_L \in L$ **then** ▷ Here, we can also call CONTAINS
      return σ_L

Algorithm 4 ***Remap-similarity.*** *Remap-similarity maps the embedded LLM response which is not in L to the embedded response in L which maximizes embedding cosine similarity.*

procedure REMAP(σ, σ_L, L) ▷ σ: A text string, σ_L, the label assigned to σ, L: A label set, M, a sentence embedding model
  if $\sigma_L \in L$ **then**
    return σ_L
  else
    $\forall j \in L, E_{L_j} := M(L_j)$
    $\sigma_{L_*} := \text{ARGMAX}_{j \in L}\left(\frac{\langle \text{E}_{\sigma_{L_j}}, \text{E}_{L_*} \rangle}{\|\text{E}_{\sigma_{L_j}}\| \|\text{E}_{\sigma_{L_*}}\|}\right)$
    return σ_L

be not always the desired one; moreover, introducing an additional model adds to overall computational complexity.

Rule-Based Label Remapping. We find that in many CTA datasets, certain types are straightforward to detect or correct using simple algorithmic approaches. Therefore, in order to provide a more realistic picture of how our method would perform in a real-world setting, we supplement both our baselines and ArcheType with *rule-based label remapping* functions, applied both prior to and after model querying. These rules do not always lead to performance improvements, but they can save considerable time and some space in the context window; therefore, we predict they will be a valuable component of deployed CTA systems, and devote some time to studying their effects. To conserve the zero-shot nature of the problem, we limited ourselves to two hours per dataset for devising these functions. As this is a one-time cost per label set, we consider this a reasonable time budget.

In Tab. 2, we list the number of labels for which rules led to performance improvements, and the average amount of the improvement across all models and methods. The rules lead to a moderate improvement for the different benchmarks.

ArcheType+. To separate the effects of rule-based remapping from other elements of the ArcheType method, we report F1 scores with and without rule-based remapping in Tab. 3 and Tab. 4. In both tables, results with rules applied are denoted with a "+" symbol.

4 NEW ZERO-SHOT BENCHMARKS

Existing CTA benchmarks [5, 10, 20, 28] are useful sources of real-world tabular data, but they were designed to evaluate methods that perform CTA on a fixed set of labels that belong classes in well-known ontologies and taxonomies. In order to probe the breadth of LLM subject knowledge and assess the effectiveness of LLM-CTA methods over rare classes with different characteristics, we create *three new zero-shot column type annotation benchmarks*: D4Tables (D4-20), derived from the D4 dataset [34] and [32], AmstrTables (Amstr-56), derived from the American Stories dataset [9], and PubchemTables (Pubchem-20), derived from the Pubchem dataset [14].

Table 2: ***Manual label remapping complements LLM-CTA.*** *Certain labels are faster and more reliable to solve using traditional methods, rather than LLMs. We document the gains from manual label remapping on our zero-shot benchmarks.*

Dataset	Num labels	Avg. Pct. Gain
SOTAB	5	1.3%
D4	9	7.2%
Amstr	2	3.2%
Pubchem	5	9.9%

Each of our benchmarks is constructed using the same general approach: we reprocess the dataset so that classes of data can be interpreted as columns, fix a random seed, and sample from the data pool to produce synthetic columns of a wide range of lengths, treating all columns as independent. This approach to CTA benchmarking stands in contrast with existing benchmarks and methods, which leverage relationships at the level of a table. However, the definition of CTA does not *guarantee* the existence of such informative metadata. Furthermore, in some real-world settings, such information is not available. We therefore regard these new benchmarks as a distinct, but valuable, way to measure progress in CTA.

We follow the approach used in [28] and attempt to replicate, as closely as possible, the distributions encountered in *real-world data*. This results in some column types that are extremely low-variance (such as `ethnicity` in D4Tables, with only 5 unique values). In other types, the set of potential unique entries in one type is entirely subsumed by another type, e.g., `us-state, other-states` in D4-Tables. Others can be addressed model free with regex pattern matching (such as `Journal ISSN` in Pubchem). As noted in Sec. 3.5, when such solutions are possible, we utilize them in both our baseline approaches and the ArcheType method itself.

D4, Amstr and Pubchem are generated from existing data distributions – it is therefore possible to produce an arbitrary number of tables using them. Balancing time constraints with the desire to test a significant sample size, we heuristically select a sample size of 2000 columns, and apply this consistently to each benchmark. The complete class names for each dataset can be found in [12].

D4Tables. Ota et al. [34] clustered data from NYC Open Data in an unsupervised manner, and the most coherent clusters (representing semantic types) were assigned labels; in total, 20 clusters were labeled. For more information on the clustering method and the complete label list, please refer to our repository. For our paper, we convert the clusters to columns and sample accordingly.

The classes in D4 are representative of open and public data sources, including 2 classes which correspond to city agencies, 4 classes which relate to public schools, and 5 classes which correspond to neighborhoods, streets or regions located in specific New

York City Boroughs. This dataset aims to assess the model's understanding of regional information and fine-grained semantic types relevant to governments and NGOs.

AmstrTables. The American Stories dataset consists of 20 million OCR scans from the Library of Congress's public domain Chronicling America collection. Each scan contains an article written between 1774 and 1963. We adapt this dataset for CTA by: dividing the articles in the dataset according to the state in which they were originally published; and creating additional column types for author bylines, newspaper names, and subheadings. Because this dataset was published in 2023, it is unlikely that any of the models evaluated in this study have trained on this data before, reducing concerns of potential data contamination [9]. Another advantage is that for the majority of column types, individual row entries are quite long, corresponding to entire newspaper articles. This phenomenon is commonplace in real-world data, but rare among academic CTA benchmarks. The classes in AmstrTables mostly pertain to journalism and history.

PubchemTables. Pubchem is the world's largest collection of freely accessible chemical information. Chemicals are identified according to their name, molecular formula, structure, biological activities, safety and toxicity information, and more. The database also contains extensive information on patents related to chemistry, such as patent abstracts and author names, as well as the names of scientific journals. We convert the RDF triple format provided by Pubchem to a columnar format suitable for CTA, and sample from the resulting distributions to produce our target columns. Correct classification requires specialist domain knowledge of chemistry.

SOTAB-27. The original SOTAB (SOTAB-91) is an unbalanced, 91-class classification problem where the task is to match each unlabeled column name with its ground-truth label. We created a zero-shot, simplified 27-class version of the benchmark (SOTAB-27) to reduce the semantic overlap among SOTAB labels. The tables in this dataset are identical to the original SOTAB benchmark; however, we remap the 91 labels in the full SOTAB benchmark to a smaller set of 27 labels. The exact details of the class remapping can be found in our github repository [12].

5 EXPERIMENTS

5.1 Experimental Setup

Fine-tuned Baselines. For our fine-tuned experiments, we compare our ArcheType LLAMA-7B (Sec. 3.4) to DoDuo [45], the state-of-the-art model for column type annotation, as well as TURL [10].

We report DoDuo and TURL results following the approach described in [28], which passes the entire table to the model at inference time; we limit our own method to 15 samples per table.

Zero-shot Baselines. To the best of our knowledge, there exist no *open-source* CTA models that can operate in a zero-shot manner; therefore, we design strong baselines derived from zero-shot CTA methods which have been introduced specifically for use with GPT: *C-Baseline*, based on the method in [25], utilizes similarity label remapping and simple random sampling, and our C-prompt.
K-Baseline, derived from [27], utilizes our K-prompt, no-op label remapping and first-k-columns sampling. We omit the method described in [27], which requires a custom hash table for each problem, as this invalidates the zero-shot nature of the problem we consider here.

For all methods, we fix 5 samples per column and provide model inputs a column-at-once manner. The prompt includes class names.

To evaluate the robustness of the methods to variations in architecture, we evaluate each method using three different architectures: the closed-source GPT-3.5-Turbo model from OpenAI (October 2023 version) denoted GPT and GPT-4.0-Turbo model (gpt-4-turbo-preview, February 2024) denoted GPT4, and the open-source T5 and UL2 encoder/decoder LLMs from Google [47].

Benchmarks. A variety of realistic and challenging CTA benchmarks have been developed in the last few years. Prominent among these are GitTables from [20], WikiTables as modified in [10], and WebTables from [5]. However, these are usually pre-processed in an ad-hoc fashion and compared against some, but not all existing methods, making it difficult to truly measure progress in the field. For this reason, we use the recent SOTAB benchmark [28]. SOTAB was independently tested on both state-of-the-art CTA approaches, TURL and DoDuo, making it an ideal testing ground for new CTA methods. Furthermore, it is, to the best of our knowledge, the most challenging CTA benchmark in the literature; the strongest method to date, DoDuo, achieves a Micro-F1 score of 84.8 on SOTAB-91, while for WikiTables and VizNet it attains Micro-F1 scores between 91.47 and 96.4 [45].

For the zero-shot regime, we also use the benchmarks introduced in Sec. 4 as well as established benchmarks: T2D [6], Efthymiou [11], and VizNet [25].

5.2 ArcheType Effectiveness

Following [45], we report performance using the weighted micro-F1 score–the weighted average of F1 scores based on the sample size of each class. We provide 95% confidence intervals for all results using the normal approximation interval method. **Boldface** in tables indicates the best-performing method(s) within the error bounds.

Tab. 3 summarizes our key results in fine-tuned CTA and Tab. 4 shows our zero-shot findings using SOTAB and the zero-shot benchmarks (Sec. 4). We observe that: 1) in the fine-tuned regime, our ArcheType-LLAMA model is competitive with DoDuo, despite training on less than 1% data; and 2) in the zero-shot regime, ArcheType outperforms or matches baselines on all dataset/architecture pairings we evaluate. These results underscore the effectiveness of ArcheType and serve as evidence that, LLMs can enable CTA methods that are not just robust to distribution shift, but that handle open-label sets defined at inference time, including rare types.

We also compare our zero-shot ArcheType to prior CTA approaches on established benchmarks, specifically: TURL, fine-tuned on the T2D [6] and Efthymiou [11] benchmarks; CHORUS [25], zero-shot on T2D and a stratified sample of the VizNet dataset (VizNet-CHORUS); DoDuo, fine-tuned on VizNet (VN) and WikiTables (WT) and evaluated on VizNet-CHORUS; and Sherlock, fine-tuned on VizNet and evaluated on VizNet-CHORUS. In all cases, we follow as closely as possible the methodology of the aforementioned authors, adopting their metrics.

As Tab. 5 shows, ArcheType's performance is comparable to that of the other systems (both fine-tuned and zero-shot) even when using the smallest (T5) backbone.

Table 3: *ArcheType achieves strong performance on the SOTAB benchmark.* *Without rule-based remapping, our method (ArcheType-LLAMA) achieves performance close to the best available pre-trained model (DoDuo), while requiring far less tabular pretraining data. With rule-based remapping (ArcheType-LLAMA+), our method improves upon it.*

Model Name	Dataset (Train)	Dataset (Eval)	Micro-F1
ArcheType-LLAMA+	LLAMA + SOTAB-91	SOTAB-91	**85.97** ±0.6
DoDuo	VizNet + SOTAB-91	SOTAB-91	84.82 ±0.6
ArcheType-LLAMA	LLAMA + SOTAB-91	SOTAB-91	82.9 ±0.6
TURL	TURL-Tables + SOTAB-91	SOTAB-91	78.96 ±0.7

Table 4: *ArcheType achieves state-of-the-art performance on zero-shot CTA benchmarks.* *ArcheType is the among the best-performing methods across all zero-shot CTA benchmarks and model architectures in our suite. With respect to architectures, we find that neither open-source model dominates. Surprisingly, closed-source models do not dominate either; GPT wins two benchmarks, ties one and loses one. In order to ablate the effect of rule-based remapping, we separately report the performance of our models on all labels (denoted +) and on labels without rules. We also indicate the number of labels remaining in each dataset after the change. All scores are weighted Micro-F1, scale 0-100.*

Method	Arch.	SOTAB-27+	SOTAB-27	D4-20+	D4-11	Amstr-56+	Amstr-54	Pubchem-20+	Pubchem-15
OPEN-SOURCE									
ArcheType	UL2	**60.9** ±0.8	58.0 ±0.9	**82.4** ±1.7	**70.8** ±2.7	**35.8** ±2.1	**32.8** ±2.1	**70.9** ±2.0	**61.1** ±2.5
C-Baseline	UL2	52.2 ±0.8	51.3 ±0.9	78.0 ±1.8	**69.4** ±2.7	13.5 ±1.5	11.4 ±1.4	61.7 ±2.1	50.3 ±2.5
K-Baseline	UL2	52.8 ±0.8	52.5 ±0.9	76.7 ±1.9	67.6 ±2.7	22.4 ±1.8	20.1 ±1.8	64.8 ±2.1	54.7 ±2.5
ArcheType	T5	**62.5** ±0.8	**60.8** ±0.9	**84.6** ±1.6	**74.5** ±2.6	29.2 ±2.0	25.6 ±2.0	**72.0** ±2.0	**63.3** ±2.4
C-Baseline	T5	51.0 ±0.8	50.0 ±0.9	81.2 ±1.7	**75.0** ±2.6	11.4 ±1.4	08.5 ±1.3	**68.3** ±2.0	**59.0** ±2.5
K-Baseline	T5	52.6 ±0.8	52.1 ±0.9	81.2 ±1.7	**74.5** ±2.6	19.2 ±1.7	15.1 ±1.6	62.3 ±2.1	51.3 ±2.5
CLOSED-SOURCE									
ArcheType	GPT	**66.0** ±0.8	**64.3** ±0.9	**87.3** ±1.5	**83.0** ±2.2	**27.2** ±2.0	**22.5** ±1.9	**65.9** ±2.1	**60.2** ±2.5
C-Baseline	GPT	59.3 ±0.8	58.5 ±0.9	77.7 ±1.8	70.8 ±2.7	09.0 ±1.3	04.9 ±0.9	56.0 ±2.2	43.0 ±2.5
K-Baseline	GPT	59.3 ±0.8	57.2 ±0.9	81.8 ±1.7	**80.9** ±2.3	10.0 ±1.3	07.9 ±1.2	**65.8** ±2.1	55.8 ±2.5

Table 5: *ArcheType zero-shot is competitive with state-of-the-art models on well-established CTA benchmarks.* *Where results were unavailable in the literature, we write n/a.*

Dataset	Metric	TURL-FT	Archetype-ZS-T5	Archetype-ZS-GPT4
T2D	Unbal. Acc.	**96.2** ±3.3	**90.4** ±3.4	**95.8** ±3.3
Efthymiou	Unbal. Acc.	74.6 ±3.8	78.5 ±3.8	**95.7** ±3.3

Dataset	Metric	DoDuo-VN-FT	DoDuo-WT-FT	Sherlock-FT	Chorus-ZS-GPT	Archetype-ZS-T5	Archetype-ZS-GPT4
T2D	Weighted F1	65.4 ±3.9	75.7 ±3.8	n/a	**92.3** ±3.4	88.9 ±3.4	**95.3** ±3.3
VizNet-Chorus	Weighted F1	**90.0** ±2.4	81.5 ±2.5	**93.0** ±2.4	86.5 ±2.5	**88.5** ±2.5	**90.5** ±2.5

5.3 Observations

A detailed analysis of our results has both confirmed our hypotheses regarding LLMs as well as uncovered insights into some of their limitations. We summarize these below.

LLMs contain sufficient world knowledge to perform zero-shot CTA on domain-specific classes. We find that LLM performance is consistently strong across datasets and across benchmarks, emphasizing the generality of LLM-CTA, compared to fine-tuned methods such as DoDuo. In PubchemTables, we observe that models are consistently able to disambiguate challenging classes such as *disease, chemical, taxonomy, patent, SMILES (simplified molecular input line entry system)*, and *molecular formula*. On D4Tables, they are able to disambiguate the names of NYC public schools and NYC governmental agencies, as well as identify locations. With $\phi = 5$, we find that ArcheType-T5 and UL2 are able to correctly identify whether the addresses are in Queens, the Bronx, Brooklyn or Manhattan more than 50% of the time, on average. ArcheType-GPT is even more impressive; it is able to accurately classify regions in all five boroughs more than 87% of the time, on average. Class-specific accuracies for our zero-shot models can be found in the extended version [13].

Model error tends to be patterned and predictable when the prompt space is fixed. When zero-shot CTA fails, it tends to do so in ways that are patterned and predictable, making it easier to correct errors. The most common failure mode is class bias in favor of certain dataset classes over others. For any given prompt/model/dataset triple, this results in certain columns with near-perfect accuracy and others with near-zero accuracy, with the confusion matrix heavily concentrated in a few classes. We provide examples of this phenomenon in the extended version of this paper [13].

Simple factors can be used to estimate zero-shot CTA performance. Zero-shot performance is stronger on datasets such as PubchemTables and D4Tables; we attribute this to smaller label spaces, smaller

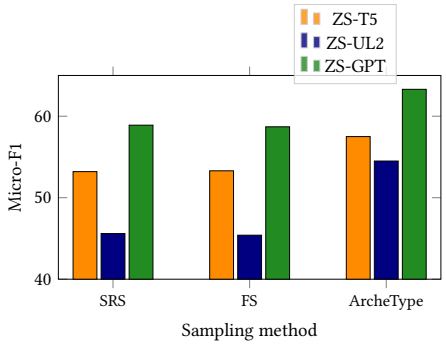

Figure 4: *ArcheType sampling outperforms baseline methods.* *The sampling method used by Zero-shot ArcheType using different architectures (GPT, UL2, and T5) on the SOTAB-27 dataset, substantially outperforms simple random sampling (SRS) and first-k-entries sampling (FS), as used in [25, 27].*

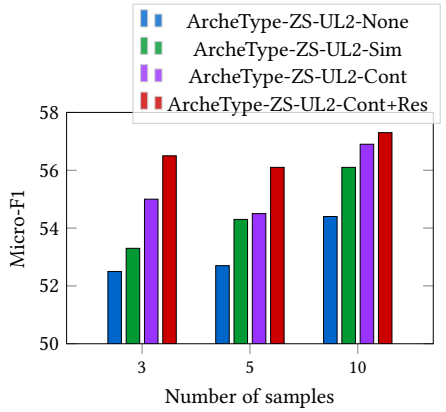

Figure 5: *ArcheType performance is affected by context size and label remapping.* *The model benefits from increasing the context size from 3 to 10 samples. All methods outperform a baseline no-op method. CONTAINS+RESAMPLE performs best at every context scale.*

individual sample sizes, and a high degree of intra-column similarity and a low degree of inter-column similarity. Amstr, which has more than twice as many labels as the next-largest dataset and a high degree of inter-column similarity (because the vast majority of the labels in the dataset correspond to newspaper articles drawn from the same general distribution), is the most challenging dataset in our benchmark.

ArcheType using open-source models is highly competitive with closed-source models. ArcheType CTA works well with a range of LLMs, small and large, open-source and closed-source, indicating that CTA benefits from flexibility in the model querying phase. Although GPT tends to have the strongest performance, the difference is not very large, and on PubChem and Amstr, GPT underperforms compared to the open-source models.

5.4 Ablation Studies

5.4.1 Ablations on Context Sampling. In Fig. 4, we ablate our choice of strategy using the SOTAB dataset, and find that ArcheType sampling consistently outperforms baseline methods.

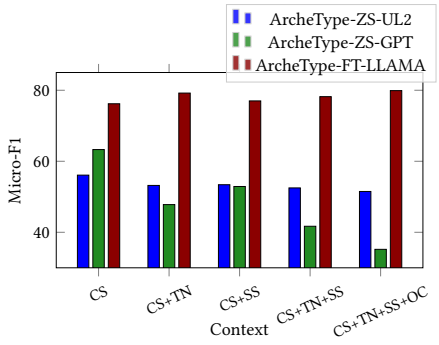

Figure 6: *Expanding feature selection during context sampling improves fine-tuned CTA performance, but degrades zero-shot performance.* *A fine-tuned ArcheType-LLAMA model is able to learn helpful associations from features such as summary statistics (SS), table filenames (TN), and other columns (OC), but that same information is not helpful when serialized in a zero-shot prompt, even when the prompt is customized to explain what each feature is.*

Sample size. The sample size $0 < \phi \leq c$ is a hyperparameter fixed at training time (in the case of fine-tuned) or inference time (in the case of zero-shot). In general, we observe in Fig. 5 that larger values of ϕ tend to result in better model performance, with the trade-off of slower inference and a larger number of truncated prompts.

Feature selection. In Fig. 6, we ablate our feature selection method, and find that ArcheType-FT benefits from each feature added, and ArcheType-ZS exhibits the opposite trend, even when we clearly identify the different types of incoming context:

```
TABLE NAME: " sourced from the
table named " + <TABLE_NAME>
OTHER COLUMNS: "For additional
context, here are some entries
from other columns in the table:
" + <OTHER_COLUMNS>
```

We consider the effective use of additional features an important area for future zero-shot CTA research.

5.4.2 Ablations on Prompt Serialization. We observe that improvements based on prompt serialization are quite sensitive to small changes in prompts; furthermore, the effects of these small changes differ depending on the LLM used. We explore six different prompts, labeled C(horus-style), K(orini-style), I(nverted), S(hort), N(oisy), B(aseline) (Sec. 3.3). The first two prompt styles are adapted from [25, 27], respectively. We test these prompts on SOTAB-27, holding other factors constant, across three architectures. As Tab. 6 shows: (1) All models are very sensitive to the choice of prompt; and (2) No prompt is a top-two performer on all three models. This supports our choice of using prompt serialization strategy as a hyperparameter. We also experimented with changing the label associated with a class and the *position* of a label in the string, and observed that these can have unpredictable effects on performance; namely, performance of relabeled class may not change, while performances of classes with the *same* labels *does* change. See [13] for details.

Prompt serialization as a hyperparameter. Our method treats prompt serialization and classname selection as *tunable hyperparameters* to be optimized and reported alongside experimental results. With the

Prompt	T5	GPT	UL2
C	49.4	57.6	56.4
K	54.0	53.2	53.4
I	52.1	62.5	55.2
S	53.0	64.6	54.5
N	48.6	61.4	47.2
B	47.1	63.4	52.1

Table 6: *Prompt serialization has unpredictable effects across models.* *A particular prompt can be engineered to perform well on a given model and fail to reproduce on others. Results shown are zero-shot Micro-F1 scores on the SOTAB-27 dataset. The best-performing prompt is highlighted in green, the second-best in yellow, and the lowest-performing in red.*

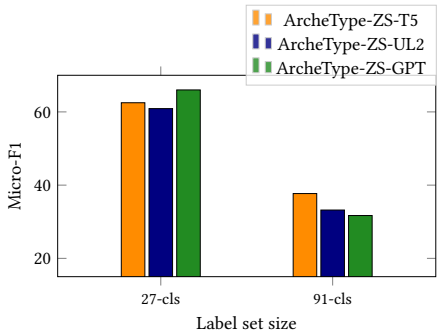

Figure 7: *Zero-shot performance degrades with large label sets.* *Both open and closed-source LLMs for zero-shot CTA struggle when the size of the label set grows large, compared to fine-tuned CTA.*

understanding that *any reasonable prompt is as likely to succeed as any other* [43], for each model-dataset pair, we conduct a grid search over our six prompt styles, each of which is stylistically distinct but similar in content and meaning. All prompts follow general best practices as described in [48], using capital letters, colons and line breaks to delineate instructions, label sets and context, but otherwise vary widely.

5.4.3 Ablations on Model Querying. The space of both open and closed LLMs has exploded of late, and the performance of these models on benchmarks can vary considerably. Rather than attempt an exhaustive comparison which would quickly grow out-of-date, we select strong representative models to stand for different categories of LLM which are frequently encountered in the literature. We find that *parameter count is not predictive of CTA performance*, and that *encoder-decoder architectures outperform decoder-only architectures* on this task. Due to space limitations, we include further details and experimental support in [13].

5.4.4 Ablations on Label Remapping. The choice of label remapping algorithm can substantially impact model performance; however, the number of remapped labels depends considerably on the selections made in the other three elements of the LLM-CTA method, as well as the dataset itself. We found a positive correlation between the number of remapped labels and model accuracy. As Fig. 5 shows, CONTAINS+RESAMPLE (Cont+Res) outperforms the other remapping strategies for all sample sizes.

5.5 Limitations

Like [31] and [17], we find that there is good reason to be optimistic about the potential for large language models to dramatically impact CTA and downstream data integration and discovery applications. Despite their strong performance, we note some limitations.

Context window lengths. The ArcheType-LLAMA method requires only 15 samples per column to reach parity with DoDuo, but it is difficult to exceed 15 samples without truncating individual examples. For that same reason, it is difficult to present large numbers of classes to zero-shot models. This limitation may be short-lived, as context windows are already reaching 200k tokens [1].

High parameter counts. Despite generalizing very well to distribution shifts, ArcheType models have very high parameter counts when compared to previous deep learning solutions. We find that increased parameter counts are likely necessary in order for the model to contain sufficient world knowledge to be applicable for CTA "in-the-wild"; however, the value added via zero-shot CTA methods will have to be weighed against their higher latency, energy, and carbon costs when they are deployed.

Context sampling. As noted in Sec. 5.4.1, zero-shot ArcheType models struggle when new features are added during context sampling. We consider this an important area of future work.

Numeric attributes. Although we benchmark ArcheType on all data types, we see the system as being primarily useful for semantic types (categorical or textual columns). Simpler approaches are likely work just as well (or perhaps even better) for purely numeric or alphanumeric columns.

Label set size. As Fig. 7 shows, all model architectures studied in this paper struggle to maintain their performance as the label set grows large, even when the context window is not exceeded. A possible reason for this is the difficulty in disambiguating several similar semantic concepts given only a brief label.

6 CONCLUSIONS AND FUTURE WORK

We introduce ArcheType, a novel CTA approach centered around LLMs. We show that with effective context sampling and label remapping, (a) LLMs can be made highly competitive with SOTA CTA models in the fine-tuned setting, and (b) LLMs are both easier to apply, and more accurate than existing deep models in the zero-shot domain. Using newly curated benchmarks (Sec. 4), we show that LLM-based CTA can generalize to considerable distribution shifts, making them ideally suited for real-world tasks.

We anticipate that methods building upon ArcheType can be useful in a variety of downstream dataset creation, curation, and processing tasks. In the future, we will explore the possibility of extending our methods to novel data tasks, such as semantic joinability, column property annotation, and dataset synthesis.

ACKNOWLEDGMENTS

This work was supported by NSF awards IIS-2106888, CMMI-2146306, and CCF-2046235; the AI Research Institutes program supported by NSF and USDA-NIFA under Award No. 2021-67021-35329; and DARPA D3M and ASKEM programs. Opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not reflect the views of NSF, USDA, or DARPA.

REFERENCES

[1] Anthropic. 2024. Introducing the next generation of Claude.

[2] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. Dbpedia: A nucleus for a web of open data. In *international semantic web conference*. Springer, 722–735.

[3] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. 2021. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258* (2021).

[4] Rishi Bommasani, Percy Liang, and Tony Lee. 2023. Holistic evaluation of language models. *Annals of the New York Academy of Sciences* 1525, 1 (2023), 140–146.

[5] Michael J. Cafarella, Alon Halevy, Daisy Zhe Wang, Eugene Wu, and Yang Zhang. 2008. WebTables: Exploring the Power of Tables on the Web. *Proc. VLDB Endow.* 1, 1 (aug 2008), 538–549.

[6] Jiaoyan Chen, Ernesto Jimenez-Ruiz, Ian Horrocks, and Charles Sutton. 2019. Learning Semantic Annotations for Tabular Data. http://arxiv.org/abs/1906.00781 arXiv:1906.00781 [cs].

[7] Lingjiao Chen, Matei Zaharia, and James Zou. 2024. How Is ChatGPT's Behavior Changing Over Time? *Harvard Data Science Review* 6, 2 (mar 12 2024). https://hdsr.mitpress.mit.edu/pub/y95zitmz.

[8] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Y. Zhao, Yanping Huang, Andrew M. Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling Instruction-Finetuned Language Models. *CoRR* abs/2210.11416 (2022). arXiv:2210.11416

[9] Melissa Dell, Jacob Carlson, Tom Bryan, Emily Silcock, Abhishek Arora, Zejiang Shen, Luca D'Amico-Wong, Quan Le, Pablo Querubin, and Leander Heldring. 2024. American stories: A large-scale structured text dataset of historical us newspapers. *Advances in Neural Information Processing Systems* 36 (2024).

[10] Xiang Deng, Huan Sun, Alyssa Lees, You Wu, and Cong Yu. 2022. TURL: Table Understanding through Representation Learning. *SIGMOD Rec. Association for Computing Machinery* 51, 1 (June 2022), 33–40.

[11] Vasilis Efthymiou, Oktie Hassanzadeh, Mariano Rodriguez-Muro, and Vassilis Christophides. 2017. Matching web tables with knowledge base entities: from entity lookups to entity embeddings. In *International Semantic Web Conference*. Springer, 260–277.

[12] Benjamin Feuer and Yurong Liu. 2023. The ArcheType System. https://github.com/penfever/ArcheType.

[13] Benjamin Feuer, Yurong Liu, Chinmay Hegde, and Juliana Freire. 2023. ArcheType: A Novel Framework for Open-Source Column Type Annotation using Large Language Models. *arXiv preprint arXiv:2310.18208* (2023).

[14] Gang Fu, Colin Batchelor, Michel Dumontier, Janna Hastings, Egon Willighagen, and Evan Bolton. 2015. PubChemRDF: towards the semantic annotation of PubChem compound and substance databases. *Journal of Cheminformatics* 7 (July 2015), 34.

[15] Phillip B Gibbons. 2016. Distinct-values estimation over data streams. In *Data Stream Management: Processing High-Speed Data Streams*. Springer, 121–147.

[16] Governo Brasileiro. 2024. Portal Brasileiro de Dados Abertos. https://dados.gov.br.

[17] 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 International Conference on Artificial Intelligence and Statistics*, Vol. 206. 5549–5581.

[18] Dan Hendrycks and Thomas G. Dietterich. 2019. Benchmarking Neural Network Robustness to Common Corruptions and Perturbations. In *International Conference on Learning Representations, ICLR*. OpenReview.net. https://openreview.net/forum?id=HJz6tiCqYm

[19] Kevin Hu, Neil Gaikwad, Michiel Bakker, Madelon Hulsebos, Emanuel Zgraggen, César Hidalgo, Tim Kraska, Guoliang Li, Arvind Satyanarayan, and Çağatay Demiralp. 2019. VizNet: Towards a large-scale visualization learning and benchmarking repository. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI)*. ACM, 1–12.

[20] Madelon Hulsebos, Çagatay Demiralp, and Paul Groth. 2023. Gittables: A large-scale corpus of relational tables. *Proceedings of the ACM on Management of Data* 1, 1 (2023), 1–17.

[21] Madelon Hulsebos, Kevin Hu, Michiel Bakker, Emanuel Zgraggen, Arvind Satyanarayan, Tim Kraska, Çagatay Demiralp, and César Hidalgo. 2019. Sherlock: A Deep Learning Approach to Semantic Data Type Detection. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 468–479.

[22] Ihab F. Ilyas and Xu Chu. 2019. *Data Cleaning*. ACM.

[23] Srinivasan Iyer, Xi Victoria Lin, Ramakanth Pasunuru, Todor Mihaylov, Daniel Simig, Ping Yu, Kurt Shuster, Tianlu Wang, Qing Liu, Punit Singh Koura, Xian Li, Brian O'Horo, Gabriel Pereyra, Jeff Wang, Christopher Dewan, Asli Celikyilmaz, Luke Zettlemoyer, and Ves Stoyanov. 2022. OPT-IML: Scaling Language Model Instruction Meta Learning through the Lens of Generalization. *CoRR* abs/2212.12017 (2022). https://doi.org/10.48550/ARXIV.2212.12017 arXiv:2212.12017

[24] Sean Kandel, Andreas Paepcke, Joseph 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*. ACM, 3363–3372.

[25] Moe Kayali, Anton Lykov, Ilias Fountalis, Nikolaos Vasiloglou, Dan Olteanu, and Dan Suciu. 2023. CHORUS: foundation models for unified data discovery and exploration. *arXiv preprint arXiv:2306.09610* (2023).

[26] Aamod Khatiwada, Grace Fan, Roee Shraga, Zixuan Chen, Wolfgang Gatterbauer, Renée J Miller, and Mirek Riedewald. 2023. SANTOS: Relationship-based Semantic Table Union Search. *Proceedings of the ACM on Management of Data* 1, 1 (2023), 1–25.

[27] Keti Korini and Christian Bizer. 2023. Column type annotation using chatgpt. *arXiv preprint arXiv:2306.00745* (2023).

[28] Keti Korini, Ralph Peeters, and Christian Bizer. 2022. SOTAB: The WDC Schema.org table annotation benchmark. In *CEUR Workshop Proceedings*, Vol. 3320. RWTH Aachen, Sun SITE Central Europe, 14–19.

[29] John P Miller, Rohan Taori, Aditi Raghunathan, Shiori Sagawa, Pang Wei Koh, Vaishaal Shankar, Percy Liang, Yair Carmon, and Ludwig Schmidt. 2021. Accuracy on the line: on the strong correlation between out-of-distribution and in-distribution generalization. In *International Conference on Machine Learning*. PMLR, 7721–7735.

[30] Niklas Muennighoff, Thomas Wang, Lintang Sutawika, Adam Roberts, Stella Biderman, Teven Le Scao, M Saiful Bari, Sheng Shen, Zheng Xin Yong, Hailey Schoelkopf, Xiangru Tang, Dragomir Radev, Alham Fikri Aji, Khalid Almubarak, Samuel Albanie, Zaid Alyafeai, Albert Webson, Edward Raff, and Colin Raffel. 2023. Crosslingual Generalization through Multitask Finetuning. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 15991–16111.

[31] Avanika Narayan, Ines Chami, Laurel J. Orr, and Christopher Ré. 2022. Can Foundation Models Wrangle Your Data? *Proc. VLDB Endow.* 16 (2022), 738–746.

[32] NYC Office of Technology and Innovation (OTI). 2024. NYC Open Data.

[33] Juri Opitz and Anette Frank. 2022. SBERT studies meaning representations: Decomposing sentence embeddings into explainable semantic features. In *Proceedings of the Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing*. Association for Computational Linguistics, 625–638.

[34] Masayo Ota, Heiko Müller, Juliana Freire, and Divesh Srivastava. 2020. Data-Driven Domain Discovery for Structured Datasets. *Proc. VLDB Endow.* 13, 7 (mar 2020), 953–967.

[35] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, Vol. 35. 27730–27744.

[36] Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin, and Vedant Misra. 2022. Grokking: Generalization beyond overfitting on small algorithmic datasets. *arXiv preprint arXiv:2201.02177* (2022).

[37] Ronak Pradeep, Sahel Sharifymoghaddam, and Jimmy Lin. 2023. Rankvicuna: Zero-shot listwise document reranking with open-source large language models. *arXiv preprint arXiv:2309.15088* (2023).

[38] Joaquin Quinonero-Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D Lawrence. 2008. *Dataset shift in machine learning*. Mit Press.

[39] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning Transferable Visual Models From Natural Language Supervision. In *ICML*. 1090–1094.

[40] Vijayshankar Raman and Joseph M Hellerstein. 2001. Potter's wheel: An interactive data cleaning system. In *VLDB*, Vol. 1. 381–390.

[41] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. 2019. Do imagenet classifiers generalize to imagenet?. In *International conference on machine learning*. PMLR, ICML, 5389–5400.

[42] Anna Rogers, Niranjan Balasubramanian, Leon Derczynski, Jesse Dodge, Alexander Koller, Sasha Luccioni, Maarten Sap, Roy Schwartz, Noah A Smith, and Emma Strubell. 2023. Closed ai models make bad baselines. *Hacking Semantics* 3 (2023).

[43] Melanie Sclar, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. 2023. Quantifying Language Models' Sensitivity to Spurious Features in Prompt Design or: How I learned to start worrying about prompt formatting. *arXiv preprint arXiv:2310.11324* (2023).

[44] Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 1715–1725.

[45] Yoshihiko Suhara, Jinfeng Li, Yuliang Li, Dan Zhang, Çağatay Demiralp, Chen Chen, and Wang-Chiew Tan. 2022. Annotating Columns with Pre-Trained Language Models. In *Proceedings of the International Conference on Management of Data (SIGMOD)*. ACM, 1493–1503.

[46] 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.

[47] Yi Tay, Mostafa Dehghani, Vinh Q. Tran, Xavier Garcia, Jason Wei, Xuezhi Wang, Hyung Won Chung, Dara Bahri, Tal Schuster, Huaixiu Steven Zheng, Denny Zhou, Neil Houlsby, and Donald Metzler. 2023. UL2: Unifying Language Learning Paradigms. In *The Eleventh International Conference on Learning Representations, ICLR*. OpenReview.net. https://openreview.net/pdf?id=6ruVLB727MC

[48] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971* (2023).

[49] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288* (2023).

[50] Jianhong Tu, Ju Fan, Nan Tang, Peng Wang, Guoliang Li, Xiaoyong Du, Xiaofeng Jia, and Song Gao. 2023. Unicorn: A Unified Multi-tasking Model for Supporting Matching Tasks in Data Integration. *Proceedings of the ACM on Management of Data* 1, 1 (2023), 1–26. https://doi.org/10.1145/3588938

[51] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All You Need. In *Proceedings of the International Conference on Neural Information Processing Systems (NEURIPS)*. 5998–6008.

[52] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771* (2019).

[53] Dan Zhang, Yoshihiko Suhara, Jinfeng Li, Madelon Hulsebos, Çağatay Demiralp, and Wang-Chiew Tan. 2020. Sato: Contextual Semantic Type Detection in Tables. *Proc. VLDB Endow.* 13, 12 (2020), 1835–1848.