LLM-assisted Labeling Function Generation for Semantic Type Detection

Chenjie Li
Illinois Institute of Technology
United States
cli112@hawk.iit.edu

Dan Zhang
Megagon Labs
United States
dan_z@megagon.ai

Jin Wang
Megagon Labs
United States
jin@megagon.ai

ABSTRACT

Detecting semantic types of columns in data lake tables is an important application. A key bottleneck in semantic type detection is the availability of human annotation due to the inherent complexity of data lakes. In this paper, we propose using programmatic weak supervision to assist in annotating the training data for semantic type detection by leveraging labeling functions. One challenge in this process is the difficulty of manually writing labeling functions due to the large volume and low quality of the data lake table datasets. To address this issue, we explore employing Large Language Models (LLMs) for labeling function generation and introduce several prompt engineering strategies for this purpose. We conduct experiments on real-world web table datasets. Based on the initial results, we perform extensive analysis and provide empirical insights and future directions for researchers in this field.

VLDB Workshop Reference Format:
Chenjie Li, Dan Zhang, and Jin Wang. LLM-assisted Labeling Function Generation for Semantic Type Detection. VLDB 2024 Workshop: The 1st International Workshop on Data-driven AI (DATAI).

1 INTRODUCTION

Semantic type detection is an important task in many data preparation applications, such as data cleaning, schema matching, entity resolution and data discovery [11, 14–16]. Given a table and a set of semantic labels, semantic type detection aims at identifying a type label for each column in the table so that each cell in the column has the same semantic types. This task has attracted significant attention from the database community, and many solutions based on deep learning techniques, especially pre-trained Language Models (PLMs) [7, 11, 16], have been developed to improve overall performance.

Although such PLM-based solutions are effective, they have a high requirement of labeled training instances to perform fine-tuning. Due to the large scale and complex structure of data lake tables, it is rather challenging to acquire high-quality human annotation for semantic type detection [3]. We argue that a weak supervision approach, such as data programming [10], is a good solution to reduce the burdens of training data annotation. In the data programming paradigm, users are asked to design label functions (LF) that provide labels to a subset of data at a much lower cost rather than manually label instances one by one. Then a label model is learned to denoise and aggregate the weak labels from each LF. Finally, the label model could predict labels over unlabeled corpus to provide annotated training data.

Over the past decade, significant efforts have been made in the field of data programming. Snorkel [9] proposed a probabilistic model to aggregate the user-written LFs. Snuba [13] aimed at proposing explainable LF while Nemo [5] and WITAN [2] focused on the problem of interactive data programming. The recent advances in the era of Large Language Model (LLM), such as GPT-4 [8] and LLaMA [12], have shown powerful capability in various tasks in different fields. Some recent efforts [4, 19] have been made to harness LLMs for automate the generation of LFs for NLP tasks. However, it is non-trivial to extend them to support the task of semantic type detection. Compared with the tasks supported in the previous studies, semantic type detection usually has a much larger labeling space and cardinality of datasets. For example, the number of semantic labels in the Gittable [6] and TURL [1] corpus is 835 and 255, respectively. Meanwhile, the task with the largest number of labels in the WRENCH benchmarking [18] only has 18 class labels. The large number of class labels brings two extra challenges: on the one hand, it is rather difficult for users to manually write enough LF for each class; on the other hand, it brings new challenges for the scalability of label model such as Snorkel to handle such large number of LFs and seeding instances for weak supervision.

In this paper, we propose an end-to-end framework to conduct weakly supervision to generate LFs for semantic type detection with the help of LLM. We systematically explore the strategies to construct LLM prompt for generating LFs given the seed instances of each label class. Specifically, we find that it is essential to include both the contents and the ground truth label of the seed instance in the prompt so as to provide sufficient context for LLM to produce effective LFs. To improve the scalability of Snorkel for semantic type detection, we develop a stacked label model to split the label space and allow the sub-models to run in parallel. We conduct experiments on widely-used tabular datasets and evaluate both the quality of the generated LFs and the effect of training end models with the datasets annotated by such LFs. Finally, we make an in-depth analysis of the preliminary results and provide some directions for the future work.

2 METHODOLOGY

2.1 Overview

The overall architecture of our proposed pipeline is shown in Figure 1. Similar to previous weak supervision works, it starts with some seed instances that help provide signals to generate LFs. Then we ask LLM to generate LFs based on the provided seed instances.

Proceedings of the VLDB Endowment. ISSN 2150-8097.

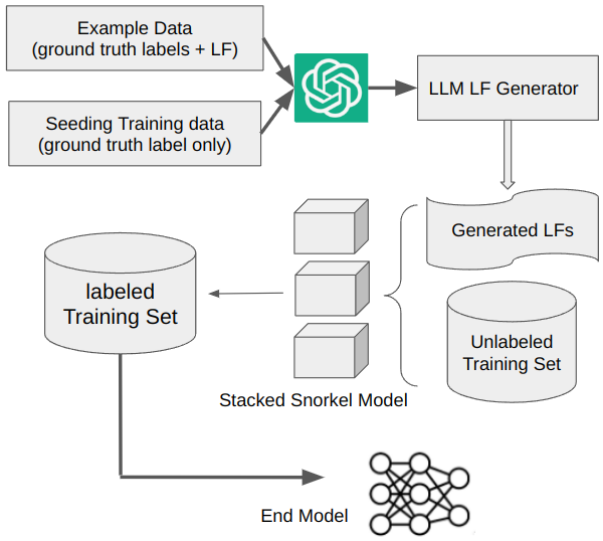

Figure 1: Overall Workflow of Labeling Pipeline with LLM

To this end, we conduct few-shot learning by providing some examples of the pairs of instances and LFs generated from them. After obtaining the set of LFs, we use them along with the seed instances to train the label model and obtain the aggregated results. In the current pipeline, we choose the well-known Snorkel [9] framework as the label model. Then given an unlabeled instance, i.e. column from a table, we will feed it to the label model and obtain the labels. Here we could evaluate the quality of the label model by considering the accuracy of labeling a set of unlabeled instances. Finally, we regard the instances obtained from the label model as the training set for an end model.

2.2 Labeling Function Generation with LLM

Next we introduce how to generate LFs by leveraging the LLMs. We follow previous studies [5, 9] to select the LFs based on the following aspects:

- Keyword: This kind of LF assumes that for each given semantic type L, there is a list of common keywords. Once the overlap between the unlabeled column and the list reaches a certain threshold, the column should be given the label L.
- Statistical: This kind of LF decides the label of a column based on the pattern of value distributions. The label is assigned if the value distribution of a column satisfies some pre-defined statistical patterns.
- Regular Expression: The LF could also be expressed with regular expressions. The label will be assigned if the column values match a provided regular expression.

The prompt template for generating is shown in Figure 2. It consists of two components: the system prompt and the user prompt. The system prompt is the general description of the background as well as some necessary instructions for the LLM. For example, our task of semantic type detection needs to specify the input as the given column values and the output as the semantic type of the column. The user prompt aims to provide the necessary contextual information to generate the labeling functions. In our prompt template, we employ the few-shot learning approach and provide some selected examples as demonstrations with the following information: (i) the examples with pairs of column values and ground truth semantic type; (ii) the LF template where the three kinds of LFs introduced above will have different templates; (iii) The question to ask for labeling functions. We randomly selected 5 semantic types and manually wrote a few labeling functions with the 3 kinds of LFs mentioned before. Specifically, within each group, we randomly selected 5 cell values delimited by a special character from that column to be used in the demonstration.

Compared with previous studies that try to use LLM to generate LFs [4, 19], we improve in the process of prompt construction. Specifically, we include the ground truth of the semantic type label of the given column in the prompt in the above item (i). The reason is that since the task is to generate the labeling function instead of predicting the label, it does not result in the risk of ground truth leakage. Meanwhile, providing the ground truth label could help LLM obtain more information to produce the labeling function as it does not need an additional step to predict the label.

2.3 Stacked Labeling Model

With the LFs generated by LLMs, the next step is to filter out the LFs with low quality. We use the idea of accuracy and redundancy filter introduced in the previous study [4]. After that, we fit the remaining LFs into the Snorkel label model and aggregate the labeling results. To reach this goal, there is still another challenge to overcome: the space complexity of Snorkel is $O(N * M * d)$, where N is the number of seed instances, M is the number of semantic types, d is the number of remaining LFs. Compared with those in previous studies of data programming [18], the target label set of semantic type detection is up to an order of magnitude larger. As a result, there will also be a much larger number of LFs which will bring significant overhead to the label model.

In this work, we use a stacking-based solution to solve this problem. The basic idea is to split the set of semantic type labels into several disjoint groups and train a Snorkel label model for each group. Then the overall computation cost will be significantly reduced. We will also have a routing model stacked on top of the set of label models: when a new unlabeled instance arrives, it sends it to all the label models and decides the label based on the results of all label models. In the current implementation, we choose the label with the highest probability as the result. The next issue to be resolved is how to split the set of labels. Some datasets, such as TURL WikiTables [1], provide the hierarchical structure of the label set which can be directly utilized to create groups. For other general cases, we propose to first get the word embedding of each label and then perform a K-means clustering over the word embeddings, where K target group count.

3 EXPERIMENTS

3.1 Experiment Setup

We evaluate the proposed framework on widely-used benchmark for semantic type detection. The Viznet dataset [1] is processed in

[1]https://github.com/megagonlabs/sato/tree/master/table_data

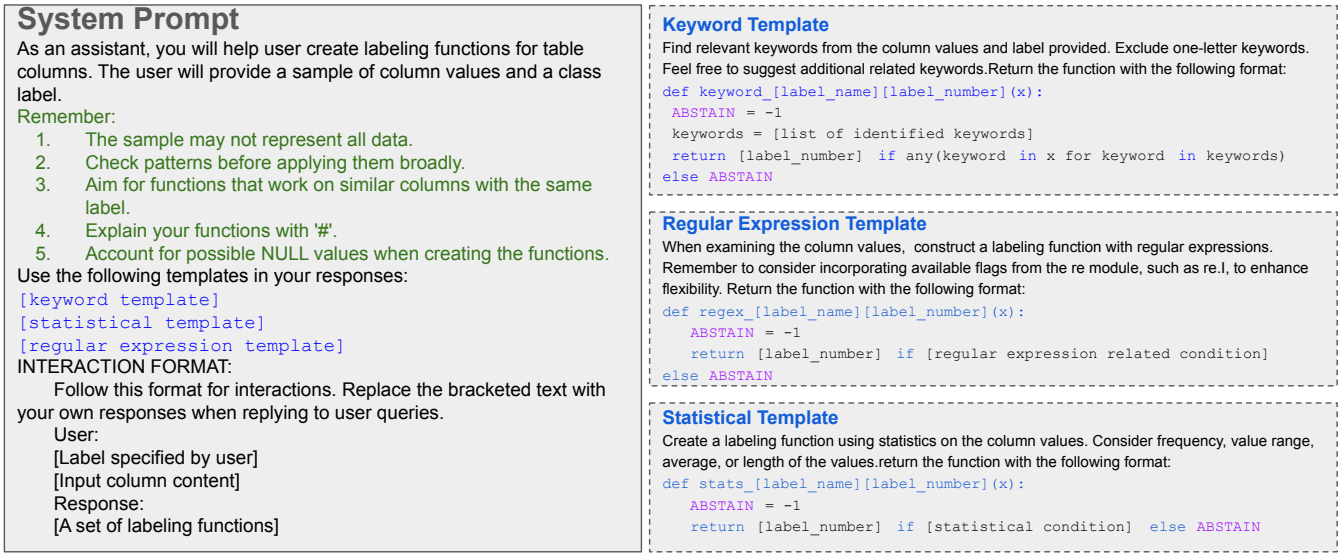

Figure 2: The Prompt Template for LF Generation

the previous study [16] on the basis of the Viznet corpus. There are 78 column types and 119,360 columns from 78,733 tables in total. We also explore a more challenging WikiTable dataset [2] proposed in [1], consisting of 570,171 tables with 255 semantic types.

To construct prompt for labeling function generation, for each semantic type in each dataset, we randomly select 10 columns that share the same semantic type and randomly select 5 column values from each column as the seed instance. We use GPT-4 as the LLM for labeling function generation. With the set of labeling functions, we stack 5 smaller snorkel models to generate the augmented training set. Finally, we fine-tune DoSolo, the single task version of Doduo [11] as the end model. For the F_1 score, we report results of both Micro and Macro F_1 scores.

3.2 Results

Table 1: Step by step evaluation results for the Viznet dataset.

Evaluation Step	Metric	Value(%)
Labeling Function	Avg. F1	18
Label Model	Micro F_1	22
	Macro F_1	28
End Model	Micro F_1	43
	Macro F_1	31

The step-by-step evaluation results for the Viznet data obtained for the proposed pipeline are shown in Table 1. We directly evaluate the quality of the LLM-generated labeling functions, the output of snorkel inference and final prediction of the end model fine-tuned on the noisy labels obtained from our augmentation process. Although there is a gap in performance between the end model and ones from previous studies that performs fine-tuning over pre-trained language models [1, 7, 11], it is worth noting that we only use a very limited set of labeled instances, which is up to 1.3% of the full training set as demonstration for the LLMs. We also evaluated the pipeline on a more challenging Wikitables dataset with 255 candidate classes and the fine-tuned end model has micro-F1 of 0.065 and macro-F1 of 0.094.

Compared to popular tasks (e.g., binary classification, NLI) in previous data programming works [18], semantic type detection tasks have much more complicated label spaces. Therefore, the task of finding explicit labeling functions (in the format of three kinds introduced before) for semantic type detection itself is rather challenging. One takeaway from our initial results would be that while explicit labeling functions works well for simple tasks such as text classification introduced in previous studies of data programming [9, 10, 18], for more complicated tasks like semantic type detection, we might either need to collect a much larger number of labeling functions and developing more scalable and efficient methods to train the label model; or develop new kinds of labeling functions that can trade the transparency for effectiveness.

3.3 Case Study

Finally, we conduct a case study to show some specific labeling functions generated by our proposed pipeline in Figure 3. Generally, we observe that the LFs generated by LLM could express reasonable semantics and are comparable with those written by humans. For example, Figure 3a illustrates a keyword-based LF for semantic type ISBN from Viznet dataset. As this column has a simple format (most of the column values are ISBN followed by the product identification number), it provides a very accurate rule to identify columns that belong to the ISBN type. Figure 3b is a statistic-based LF for semantic type year. As shown in the function, LLM could incorporate its internal knowledge with the given semantic type. Specifically, in the return statement it added the frequently mentioned year

[2] https://github.com/sunlab-osu/TURL#data

range (1700-2023) as the way to decide if a string is a year or not. Meanwhile, there are also some examples that LF has good coverage but is not so accurate. The LF in Figure 3c is a function created for the semantic type "name". The function looks for words that start with a capital letter followed by lowercase letters, optionally followed by another similar word. This function can cover most of the names, but at the same time, it can also cover a lot of values that are from the other semantic types such as location or address. Thus it provides limited insights in labeling the instances.

```
def isbn40_keyword(x):
    ABSTAIN = -1
    list_values = x.split('|x|')
    keywords = ['isbn', 'ISBN']
    return 40 if any((k in n for n in list_values for k in keywords))
else ABSTAIN
```

(a) Keyword LF for semantic type ISBN

```
def year77_stats(x):
    ABSTAIN = -1
    list_values = x.split('|x|')
    year_values=[n for n in list_values if n.isdigit()
                 and 1700 <= int(n) <= 2023]
    return 77 if len(year_values) / len(list_values) > 0.5 else ABSTAIN
```

(b) Statistics LF for semantic type year

```
def name46_regex(x):
    ABSTAIN = -1
    list_values = x.split('|x|')
    name_pattern = re.compile(r'\b[A-Z][a-z]+(?:\s[A-Z][a-z]+)?\b')
    matched_values = [n for n in list_values if name_pattern.search(n)]
    return 46 if len(matched_values) / len(list_values) > 0.5 else ABSTAIN
```

(c) Regular Expression LF for semantic type name

Figure 3: Examples of Generated Labeling Functions

4 CONCLUSION AND FUTURE WORK

In this paper, we studied the problem of generating labeling functions for the task of semantic type detection. We proposed an end-to-end pipeline that adopted LLM to automatically generate labeling functions for semantic type detection via prompt engineering and train a label model based on Snorkel. We make an extensive set of explorations on the design space and conduct initial experiments on two popular benchmark datasets.

Based on our initial efforts, we recognize several essential directions for further exploration of this topic. As illustrated by our experimental results, the current explicit labeling functions commonly used in previous data programming studies may not be sufficient for handling more complex tasks like semantic type detection. For this task, as well as related table understanding tasks such as relationship extraction and column population, we need to develop new types of labeling functions that are more powerful yet still explainable. A promising starting point could be building labeling functions based on simple machine learning models, such as logistic regression and decision trees. It is also crucial to improve the scalability of the label models concerning the number of class labels and labeling functions. Our idea of splitting the label space provides a reasonable solution to this problem. It is necessary to explore this approach further by generalizing the problem and developing an efficient algorithm that can identify high-quality splits.

Moreover, it is beneficial to consider advanced label models developed in recent efforts from the machine learning community as introduced in [17]. Last but not least, although the effectiveness of labeling functions generated in this work is limited, the labeling functions could still provide some useful insights for the semantic labels of columns. So it is also worth investigating how to use the generated labeling function to explain the results of existing solutions for semantic type detection, such as those based on pre-trained language models.

REFERENCES

[1] Xiang Deng, Huan Sun, Alyssa Lees, You Wu, and Cong Yu. 2020. TURL: Table Understanding through Representation Learning. *Proc. VLDB Endow.* 14, 3 (2020), 307–319.

[2] Benjamin Denham, Edmund M.-K. Lai, Roopak Sinha, and M. Asif Naeem. 2022. Witan: Unsupervised Labelling Function Generation for Assisted Data Programming. *Proc. VLDB Endow.* 15, 11 (2022), 2334–2347.

[3] Grace Fan, Jin Wang, Yuliang Li, and Renée J. Miller. 2023. Table Discovery in Data Lakes: State-of-the-art and Future Directions. In *Companion of SIGMOD*. 69–75.

[4] Naiqing Guan, Kaiwen Chen, and Nick Koudas. 2023. Can Large Language Models Design Accurate Label Functions? *CoRR* abs/2311.00739 (2023).

[5] Cheng-Yu Hsieh, Jieyu Zhang, and Alexander J. Ratner. 2022. Nemo: Guiding and Contextualizing Weak Supervision for Interactive Data Programming. *Proc. VLDB Endow.* 15, 13 (2022), 4093–4105.

[6] Madelon Hulsebos, Çagatay Demiralp, and Paul Groth. 2023. GitTables: A Large-Scale Corpus of Relational Tables. *Proc. ACM Manag. Data* 1, 1 (2023), 30:1–30:17.

[7] Zhengjie Miao and Jin Wang. 2023. Watchog: A Light-weight Contrastive Learning based Framework for Column Annotation. *Proc. ACM Manag. Data* 1, 4 (2023), 272:1–272:24.

[8] OpenAI. 2023. GPT-4 Technical Report. *CoRR* abs/2303.08774 (2023).

[9] Alexander Ratner, Stephen H. Bach, Henry R. Ehrenberg, Jason Alan Fries, Sen Wu, and Christopher Ré. 2017. Snorkel: Rapid Training Data Creation with Weak Supervision. *Proc. VLDB Endow.* 11, 3 (2017), 269–282.

[10] Alexander J. Ratner, Christopher De Sa, Sen Wu, Daniel Selsam, and Christopher Ré. 2016. Data Programming: Creating Large Training Sets, Quickly. In *NeurIPS*. 3567–3575.

[11] Yoshihiko Suhara, Jinfeng Li, Yuliang Li, Dan Zhang, Çagatay Demiralp, Chen Chen, and Wang-Chiew Tan. 2022. Annotating Columns with Pre-trained Language Models. In *SIGMOD*. 1493–1503.

[12] Hugo Touvron and et al. 2023. LLaMA: Open and Efficient Foundation Language Models. *CoRR* abs/2302.13971 (2023).

[13] Paroma Varma and Christopher Ré. 2018. Snuba: Automating Weak Supervision to Label Training Data. *Proc. VLDB Endow.* 12, 3 (2018), 223–236.

[14] Jin Wang and Yuliang Li. 2022. Minun: evaluating counterfactual explanations for entity matching. In *DEEM '22: Proceedings of the Sixth Workshop on Data Management for End-To-End Machine Learning*. 7:1–7:11.

[15] Jin Wang, Yuliang Li, Wataru Hirota, and Eser Kandogan. 2022. Machop: an end-to-end generalized entity matching framework. In *aiDM '22: Proceedings of the Fifth International Workshop on Exploiting Artificial Intelligence Techniques for Data Management*. 2:1–2:10.

[16] Dan Zhang, Yoshihiko Suhara, Jinfeng Li, Madelon Hulsebos, Çagatay Demiralp, and Wang-Chiew Tan. 2020. Sato: Contextual Semantic Type Detection in Tables. *Proc. VLDB Endow.* 13, 11 (2020), 1835–1848.

[17] Jieyu Zhang, Cheng-Yu Hsieh, Yue Yu, Chao Zhang, and Alexander Ratner. 2022. A Survey on Programmatic Weak Supervision. *CoRR* abs/2202.05433 (2022).

[18] Jieyu Zhang, Yue Yu, Yinghao Li, Yujing Wang, Yaming Yang, Mao Yang, and Alexander Ratner. 2021. WRENCH: A Comprehensive Benchmark for Weak Supervision. In *NeurIPS*.

[19] Rongzhi Zhang, Yue Yu, Pranav Shetty, Le Song, and Chao Zhang. 2022. Prompt-Based Rule Discovery and Boosting for Interactive Weakly-Supervised Learning. In *ACL*. 745–758.