Short Paper

DataSculpt: Cost-Efficient Label Function Design via Prompting Large Language Models

Naiqing Guan
University of Toronto
Toronto, Canada
naiqing.guan@mail.utoronto.ca

Kaiwen Chen
University of Toronto
Toronto, Canada
kckevin.chen@mail.utoronto.ca

Nick Koudas
University of Toronto
Toronto, Canada
koudas@cs.toronto.edu

ABSTRACT

Programmatic weak supervision methodologies expedite the labeling of extensive datasets using label functions (LFs) that encapsulate heuristic data sources. Nonetheless, the creation of precise LFs necessitates domain expertise and substantial endeavors. Recent advances in large language models (LLMs) have exhibited substantial potential across diverse tasks, including synthesizing LFs automatically. However, previous works either fall short of creating accurate LFs due to a lack of specificity or require a substantial monetary budget for exhaustive querying.

In this research, we propose a novel framework, DataSculpt, that harnesses LLMs for automated LF generation. DataSculpt leverages few-shot learning in an iterative PWS framework, creating a massive and diverse LF set with a single prompt template. We evaluated DataSculpt on six real-world datasets. Our results show that DataSculpt can generate accurate LFs and significantly reduce costs. For instance, using GPT-3.5, DataSculpt achieved downstream model accuracy comparable to exhaustive querying methods at only a fraction of the cost, with total expenses around \$0.06 compared to over \$250 for exhaustive querying. DataSculpt demonstrates that efficient LF generation with high accuracy is achievable, making it a practical solution for large-scale data labeling.

1 INTRODUCTION

Modern machine learning models require large amounts of labeled data to achieve high accuracy, but manual labeling is expensive, particularly when domain expertise is needed. The programmatic weak supervision (PWS) paradigm [26, 28] offers a solution by enabling rapid labeling of large datasets through the use of label functions (LFs). LFs are heuristic rules that generate noisy labels for subsets of data, reducing the cost and effort compared to manual labeling. In the PWS paradigm, the user designs LFs to automatically provide weak labels, which are then denoised and aggregated by a label model. Finally, the aggregated labels are used to train the downstream machine learning model (or end model).

While PWS reduces the manual labeling burden, significant effort is still required to design accurate LFs. To mitigate this, researchers have developed frameworks like interactive weak supervision [5] and interactive data programming [14], which assist users in LF design. However, these methods require substantial user input and do not fully automate the LF creation process.

Recent advancements in large language models (LLMs) such as GPT-3 [6] have led researchers to explore their use in automating LF design. ScriptoriumWS [15] employs the OpenAI Codex model [7] to generate code-based LFs automatically, while PromptedLF [34] uses multiple prompts in a zero-shot manner to annotate data, treating these annotations as weak labels. Although these methods advance LF automation, they present trade-offs between accuracy and cost. ScriptoriumWS often produces less accurate LFs due to a lack of specificity in its broad prompts, while PromptedLF achieves high accuracy but at significant cost by annotating every unlabeled instance with multiple prompts.

To address these challenges, we introduce DataSculpt[1], a novel framework that leverages few-shot prompting to design highly accurate and cost-effective labeling functions. Figure 1 illustrates the iterative workflow of DataSculpt. In each iteration, the framework selects a query instance from the unlabeled dataset, constructs a prompt for the LLM, and uses in-context examples from the labeled validation set. The LLM-generated LFs are verified for validity, accuracy, and redundancy before being added to the LF set. DataSculpt advances automated weak supervision techniques, offering several key innovations:

(1) **Few-shot Prompting for iterative LF Generation**: Unlike previous approaches [15, 34], DataSculpt uniquely applies few-shot learning principles in an iterative workflow to create highly specific and accurate LFs. By incorporating demonstration examples from a labeled dataset and selecting specific query instances from the unlabeled dataset, DataSculpt enables generation of a large and diverse LF set with minimal manual input, significantly reducing the need for multiple prompt templates.
(2) **Integration of Advanced Prompting Strategies**: We are the first to explore and adapt cutting-edge prompting techniques specifically for LF development. By incorporating chain-of-thought prompting [39], in-context example selection [19] and self-consistency [38], DataSculpt encourages LLMs to break down the LF creation process into logical steps and generate multiple LFs for each query instance, enhancing the interpretability and reliablity of synthesized LFs.
(3) **Balanced Accuracy and Cost-effectiveness**: To the best of our knowledge, DataSculpt is the first work that extensively evaluates and optimizes the trade-off between LF accuracy and cost-effectiveness. By focusing on selected query instances and using few-shot learning, our framework achieves high LF accuracy while only requiring a sample from the unlabeled dataset to query LLMs. This approach significantly reduces costs compared to methods that annotate every instance, while maintaining comparable downstream model accuracy.

DataSculpt follows a recent line of research [16, 22, 36] exploring prompting techniques for the solution of problems that necessitated human effort. While DataSculpt builds upon existing techniques, its contributions to the field of automated weak

[1] Code available at https://github.com/Gnaiqing/DataSculpt

supervision are significant and multifaceted. Our work provides the first comprehensive evaluation of various prompting strategies for LF development, offers crucial insights into the trade-offs between accuracy and cost in LF generation, and demonstrates how the integration and optimization of existing techniques can lead to substantial improvements in weak supervision pipelines. Through extensive empirical analysis across multiple datasets, DataSculpt not only advances our understanding of LLM-based weak supervision but also establishes important benchmarks and reproducible frameworks for future research in this rapidly evolving field.

2 RELATED WORK

Programmatic Weak Supervision. Programmatic Weak Supervision (PWS) [26, 28, 40] is an emerging framework that leverages various weak supervision sources to label training data automatically. In the PWS framework, users first design label functions to provide weak labels to the data, and then the label model aggregates noisy, weak labels to provide probabilistic labels to the unlabeled data. Researchers have proposed various label models [11, 26, 27] to learn the accuracy and correlations between label functions.

Efficient label function design is another important research direction for PWS. Snuba [37] proposes deriving label functions from simple machine learning models. IWS [5], WITAN [9], and Darwin [12] interactively ask users to verify candidate label functions. Nemo [14] proposes the interactive data programming framework, which interactively selects candidate instances for users to observe and develop label functions. With the advance of pre-trained language models, researchers have also begun exploring prompting these models to design LFs automatically [15, 34].

Large Language Models. Large language models (LLMs) developed recently, such as GPT-3 [6], GPT-4 [1], and Llama 2[35], can perform various tasks by prompting. Finetuning LLMs using human feedback improves their performance in following human instructions [25]. This has motivated researchers to design efficient prompting methods, including chain-of-thought [39], self-consistency [38] and efficient in-context example selection methods [19]. Our paper adopts a few-shot learning-based framework [6] for leveraging these LLMs to design LFs.

Active Learning. Active learning [31, 32] focuses on improving the learning process by strategically selecting the most informative data points for the model to learn from. Various strategies for active learning have been explored within the machine learning community, including uncertainty sampling [18], query-by-committee [33], core-set methods [30], etc. For those interested in a more detailed overview of active learning techniques, recent surveys [20, 29] provide comprehensive coverage. Researchers have also investigated active learning strategies to improve PWS performance, such as guiding LF development [5, 14] or improving label quality [13, 23]. Our paper leverages active learning strategies for query instance selection.

3 FRAMEWORK DESCRIPTION

DataSculpt is tailored for text classification tasks, which involve categorizing text passages into various groups, such as topics or sentiments. It also supports relationship classification between entities within a passage, such as determining if two individuals are spouses. This section describes the LF space and prompt template used in DataSculpt, followed by strategies for selecting query instances, in-context exemplars, and filtering LFs.

3.1 LF space

DataSculpt focuses on text classification, creating keyword-based LFs from queried instances. A keyword-based LF, denoted as $\lambda_{k,c}$, labels a passage as class c if it contains a specific keyword or phrase k. DataSculpt restricts keywords to unigrams, bigrams, and trigrams.

A single keyword may be insufficient for relationship classification tasks (e.g., determining if A and B are spouses in the Spouse [8] dataset). For instance, the keyword *marry* cannot distinguish between *A marry B* and *A marry C*. To address this, DataSculpt extends keyword-based LFs to include target entities, such as *[A] marry [B]*, which activates if both entities match the query.

While the keyword-based LFs only support inclusion (i.e., including a keyword) by themselves, the label model can aggregate the weak supervision sources and support more complicated criteria like exclusion or conjunction. The downstream model in PWS can generalize beyond keyword-based decision boundaries by considering the text features (extracted by BERT [10] in our experiments).

3.2 Prompt Template

DataSculpt employs few-shot learning to generate LFs using a single template per task, which is at the heart of our proposal. Figure 2 illustrates the Base and CoT templates for DataSculpt respectively. Each template begins with a task description and class definitions, followed by in-context examples to help the LLM understand the task and output format. The in-context examples are selected from the validation set as detailed in Section 3.3. The template concludes with a user query, prompting the LLM to generate LFs based on the query instance. This approach leverages the advantages of few-shot learning, enabling DataSculpt to design accurate and diverse LFs. In the DataSculpt-CoT prompt, the template includes a step-by-step reasoning process in the prompt (italicized in Figure 2), further enhancing the output accuracy using chain-of-thought [39]. The LLM produces a set of keywords and a class label based on the queried instance. We convert the keywords and class labels to Python programs that detect whether the keywords exist in the text, which serves as the LFs in the PWS framework.

3.3 In-context Example Selection

Providing a few in-context examples before the query helps the LLM better understand the task, enhancing its effectiveness. DataSculpt evaluates two in-context example selection methods: *class-balanced*, which randomly selects examples balanced across classes, and *KATE* [19], which selects examples closest to the test input in a feature space. These examples are drawn from a labeled validation set.

Examples must include indicative keywords and, if applying CoT, a step-by-step reasoning process that requires manual annotation. We select ten examples per dataset from the validation set for class-balanced selection and manually provide keywords and explanations. For KATE, manual annotation is impractical due to varying query instances. Inspired by previous works on automatically generating CoT [17, 43], we use the LLM to generate indicative keywords and CoT for KATE-selected examples automatically. These examples are labeled in advance, and the LLM

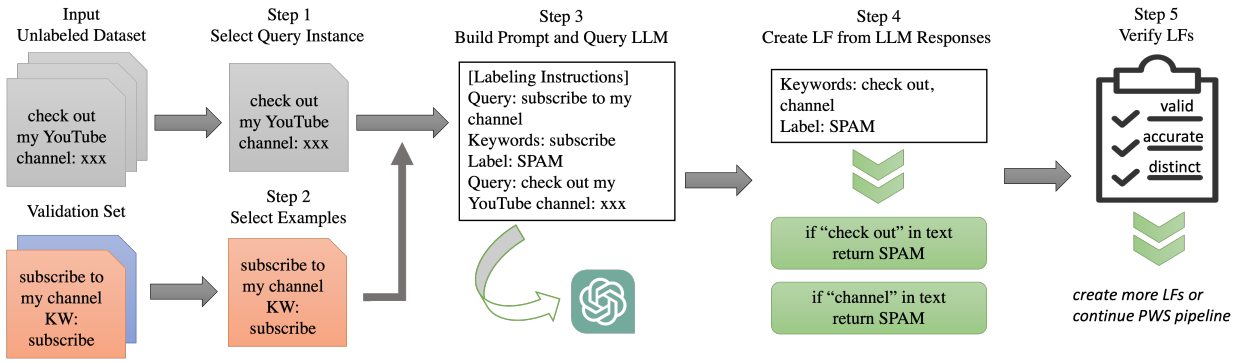

Figure 1: Illustration of the DataSculpt framework for generating LFs. In each iteration, the framework selects query instance and in-context examples from the unlabeled dataset and labeled validation set, respectively, and queries the LLM to create LFs.

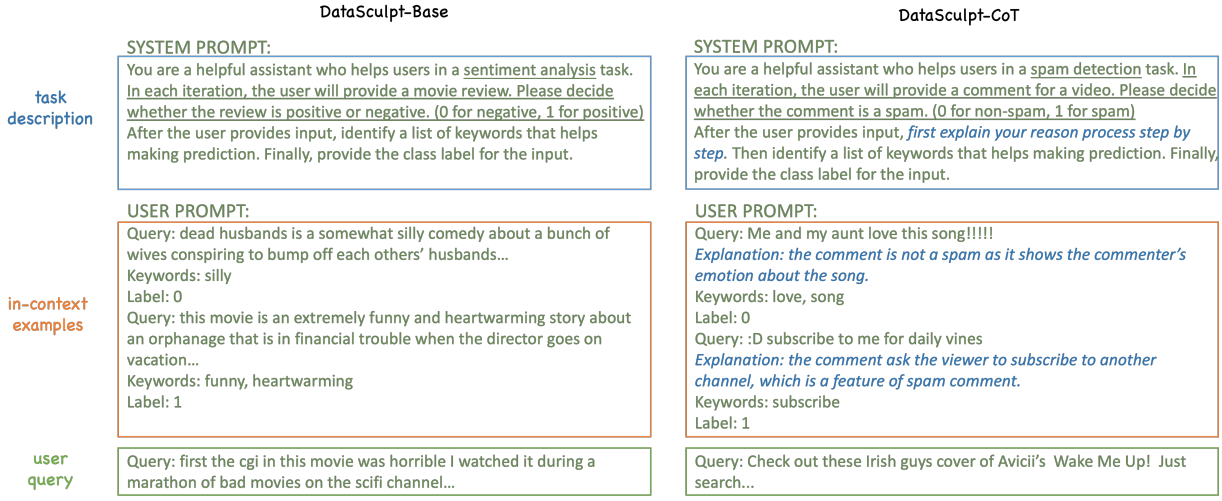

Figure 2: Prompt template for the DataSculpt framework. The underlined instructions are dataset-specific.

provides reasoning to justify the labels, similar to the prompts in Figure 2, with labels included in the user input.

3.4 Query Instance Selection

DataSculpt iteratively selects instances for the LLM to generate LFs, significantly influencing LF quality. We evaluate several selection methods:

- **Random Sampling**: Randomly selects instances from the unlabeled dataset (default strategy).
- **Uncertain Sampling** [18]: Selects instances where the current downstream model is most uncertain, measured by predictive entropy.
- **Select by Expected Utility (SEU)** [14]: This approach first statistically measures the utility of potentially generated LFs, then proposes a user model that measures the conditional probability of the user returning an LF when observing specific instances. The user model estimates the accuracy of candidate LFs and assumes that more accurate LFs are prone to be selected by the user. Finally, SEU selects the instance that leads to the highest expected utility of LFs.

In DataSculpt, we provide one query instance per iteration. We also experimented with providing multiple query instances in the prompt per iteration, but we did not observe clear advantages in the accuracy and coverage of curated LFs. We hypothesize that this is because multiple query instances do not frequently have keywords in common for designing better LFs; thus, querying multiple instances in an iteration has a similar performance to querying them separately in multiple iterations in terms of LF design.

3.5 LF Filters

To ensure quality, DataSculpt applies several filters to the generated LFs:

- **Validity Filter**: Checks if the LF keywords are unigrams, bigrams, or trigrams and if the labels are in candidate classes.
- **Accuracy Filter**: Evaluates LFs on the validation set, pruning those with an accuracy below a threshold (0.6 by default). If an LF is inactive on any validation instance, it passes this filter.
- **Redundancy Filter**: Removes LFs with high similarity to existing ones, using an intersection over union metric. An LF is pruned if its consensus with an existing LF exceeds 95% over active instances.

Table 1: Datasets used in evaluation.

Task	Domain	Dataset	#Class	#Train	#Valid	#Test
Spam Cls.	Review	Youtube [2]	2	1586	120	250
	Text Message	SMS [3]	2	4571	500	500
Sentiment Cls.	Movie	IMDB [21]	2	20000	2500	2500
	Review	Yelp [42]	2	30400	3800	3800
Tpoic Cls.	News	Agnews [42]	4	96000	12000	12000
Relation Cls.	News	Spouse [8]	2	22254	2811	2701

3.6 Default Class

In some classification tasks, the positive class refers to the existence of some pattern, and the negative class refers to its absence. In these tasks, it is usually easier to design LFs for the positive class than the negative class. Our evaluation shows that LLMs often fail to provide LFs for the negative class, leading to class imbalance and reduced performance. We define a *default class* for datasets with these characteristics to address this. If any LFs do not cover an instance, it is assigned to the default class before training the downstream model. In our evaluations, the default class is defined for the spouse dataset (default is no spouse relationship).

4 EXPERIMENTS

In this section, we comprehensively evaluate DataSculpt's performance in generating label functions (LFs) and compare it with several baselines that leverage large language models (LLMs) for LF design.

4.1 Experiment Setup

We evaluate DataSculpt on six real-world text classification datasets from the WRENCH benchmark [41]. Table 1 summarizes the datasets used in our evaluation. To ensure a fair comparison, we exclude the pre-existing LFs associated with these datasets in the WRENCH benchmark and instead generate LFs using DataSculpt's pipeline.

For each run of DataSculpt, we iteratively select 50 query instances to design LFs. We repeat each run with different random seeds five times and report the average LF statistics and downstream model performance after the LF development process. We use BERT [10] (110M parameters) to extract features from the text, MeTaL [27] as the label model, and logistic regression as the downstream model, consistent with the WRENCH benchmark configurations. Note that ground truth labels for the training data are unavailable for the Spouse dataset, so we only report metrics that do not require these labels.

DataSculpt offers various configuration options that affect its performance. By default, we use the gpt-3.5-turbo-0613 model from the OpenAI API [24] as the LLM, set temperature as 0.7, apply random sampling for query instance selection, and use all three LF filters (validity, accuracy, redundancy). We compare the following versions of DataSculpt against baseline methods:

- **DataSculpt-Base**: Uses a basic few-shot learning prompt.
- **DataSculpt-CoT**: Uses a few-shot learning prompt with the chain-of-thought strategy [39].
- **DataSculpt-SC**: Builds on DataSculpt-CoT by leveraging self-consistency [38] to aggregate multiple responses. Specifically, the LLM generates 10 responses, and the predicted label is determined by majority voting.
- **DataSculpt-KATE**: Extends DataSculpt-SC by incorporating the KATE method [19] to select in-context examples similar to the queried instance.

Table 2: Statistics of synthesized LFs and end model accuracy.

Metric	Method	Youtube	SMS	IMDB	Yelp	Agnews	Spouse	AVG
#LFs	WRENCH	10	73	5	8	9	9	19.0
	ScriptoriumWS	9	73	6	11	8	8	19.2
	PromptedLF	10	73	7	7	4	11	18.7
	DataSculpt-Base	76	164	88	127	180	***14***	108.2
	DataSculpt-CoT	69	115	*104*	118	158	10	95.7
	DataSculpt-SC	***108***	**201**	***225***	***247***	***225***	**43**	***174.8***
	DataSculpt-KATE	**117**	***200***	**329**	**321**	**236**	13	**202.7**
LF Acc.	WRENCH	**0.832**	**0.973**	0.699	0.731	0.817	–	***0.810***
	ScriptoriumWS	0.646	0.897	0.595	0.736	0.565	–	0.688
	PromptedLF	***0.810***	***0.923***	**0.953**	**0.882**	0.674	–	**0.848**
	DataSculpt-Base	0.739	0.913	0.718	0.769	**0.847**	–	0.797
	DataSculpt-CoT	0.757	0.861	0.728	***0.775***	***0.826***	–	0.789
	DataSculpt-SC	0.735	0.884	0.726	0.772	0.824	–	0.788
	DataSculpt-KATE	0.726	0.870	***0.731***	0.761	0.811	–	0.780
LF Cov.	WRENCH	0.163	***0.710***	0.236	0.183	0.103	0.038	0.239
	ScriptoriumWS	**0.592**	**0.815**	**0.894**	***0.716***	***0.305***	**1.000**	**0.720**
	PromptedLF	***0.219***	0.011	***0.369***	**0.723**	**0.334**	***0.200***	***0.309***
	DataSculpt-Base	0.025	0.007	0.038	0.024	0.002	0.026	0.020
	DataSculpt-CoT	0.026	0.007	0.034	0.023	0.003	0.021	0.019
	DataSculpt-SC	0.021	0.009	0.024	0.018	0.003	0.034	0.018
	DataSculpt-KATE	0.020	0.008	0.017	0.014	0.003	0.006	0.011
Total Cov.	WRENCH	0.877	0.405	0.876	0.828	0.691	0.907	0.764
	ScriptoriumWS	**1.000**	**1.000**	**0.998**	***0.991***	***0.692***	**1.000**	**0.947**
	PromptedLF	***0.974***	0.530	0.970	**1.000**	**0.916**	***0.940***	***0.888***
	DataSculpt-Base	0.808	0.637	0.932	0.900	0.323	0.305	0.651
	DataSculpt-CoT	0.782	0.529	0.942	0.871	0.350	0.175	0.608
	DataSculpt-SC	0.820	***0.754***	0.977	0.947	0.480	0.774	0.792
	DataSculpt-KATE	0.832	0.716	***0.980***	0.948	0.438	0.061	0.663
EM Acc/F1	WRENCH	0.887	**0.865**	0.761	0.847	**0.835**	0.181	0.729
	ScriptoriumWS	0.832	0.720	0.740	0.750	0.806	0.157	0.668
	PromptedLF	**0.908**	0.832	**0.831**	0.858	0.663	**0.462**	0.759
	DataSculpt-Base	***0.896***	0.821	0.791	0.869	***0.813***	0.412	***0.767***
	DataSculpt-CoT	0.885	0.817	0.793	0.854	0.795	0.329	0.746
	DataSculpt-SC	0.879	0.829	***0.794***	***0.874***	0.758	***0.455***	0.765
	DataSculpt-KATE	0.862	***0.836***	0.788	**0.879**	0.800	0.440	**0.768**

4.1.1 Baselines. We compare DataSculpt with the following baselines. For a fair comparison of end model accuracy, we use MeTaL as the label model and logistic regression as the end model across all baselines:

- **WRENCH** [41]: The WRENCH benchmark includes manually designed LFs by human experts. This baseline provides an intuitive comparison of the quality of automatically generated LFs versus manually designed ones.
- **ScriptoriumWS** [15]: ScriptoriumWS uses code generation models to create small programs that function as LFs. Since ScriptoriumWS was evaluated on the same datasets we use, we present the results reported by the authors.
- **PromptedLF** [34]: PromptedLF requires the user to design multiple prompt templates, which are then combined with unlabeled instances to query the LLM, and the responses are treated as weak labels. The original paper provides prompt templates for YouTube, SMS, and Spouse datasets. We use these provided templates and create additional templates for other datasets by translating the WRENCH benchmark LFs into prompt templates. The templates can be found in our code repository.

4.2 Performance Overview

Table 2 presents the statistics of LFs synthesized by the evaluated frameworks, including the number of synthesized LFs (#LFs), the average accuracy of LFs on the training set (LF Acc.), the average coverage of LFs on the training set (LF Cov.), the total fraction of data covered by any LF (Total Cov.), and the end model accuracy or F1 score for imbalanced datasets on the test set (EM Acc/F1). LF coverage measures the percentage of data where an LF is activated (i.e., returns a weak label). LF accuracy is measured on the subset of data where the LF is activated. To help the readers navigate through the figures, we highlight the top 3 results using

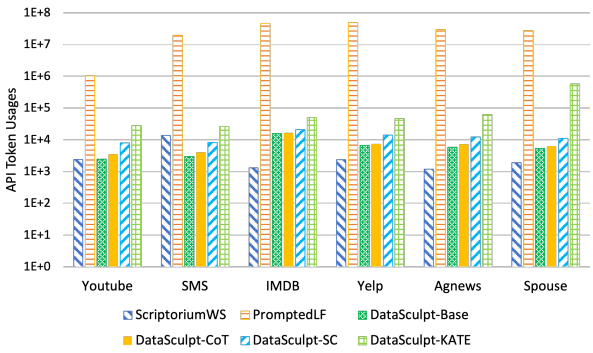

Figure 3: Token usage for synthesizing LFs.

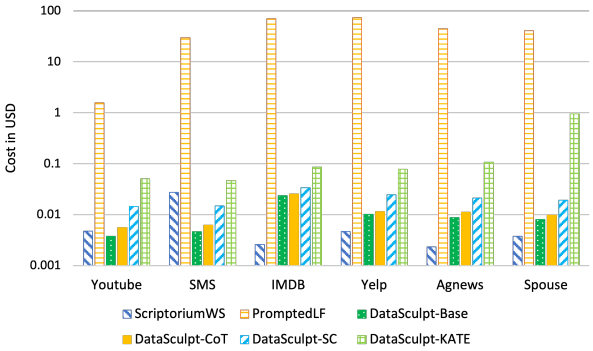

Figure 4: API cost for synthesizing LFs.

bold, italic with bold, and underline tags, respectively, in Table 2. We also report the average results on all evaluated datasets.

From Table 2, we observe that DataSculpt generates a significantly more extensive set of LFs than baseline methods. This is because, in baseline methods, the human effort required to design LFs scales linearly with the size of the LF set, making it impractical to create a large LF set with limited human resources. Conversely, DataSculpt requires the user to design only a single prompt template for each dataset, with the diversity of synthesized LFs arising from the selection of queried instances. This allows DataSculpt to scale to hundreds of LFs without increasing the human effort.

DataSculpt is also highly cost-efficient. Figures 3 and 4 illustrate the different methods' token usage and API costs, respectively. DataSculpt-Base consumes 38,992 tokens across six datasets, costing only \$0.06 in total[2]. In contrast, PromptedLF consumes over 170 million tokens and costs over \$250 to label the same datasets, with only minor improvements in LF accuracy. This efficiency arises because DataSculpt requires only $\Theta(m)$ queries (where m is the number of LFs to create). In contrast, PromptedLF must query every unlabeled instance using every prompt template. Thus, DataSculpt offers a cost-efficient solution for synthesizing LFs with good accuracy.

Regarding the accuracy of synthesized LFs, DataSculpt achieves accuracy levels on par with human-designed LFs in the WRENCH benchmark and outperforms LFs synthesized by ScriptoriumWS by 10.9 points on average. This indicates that LLMs can produce

[2]the API cost for gpt-3.5-turbo-0613 was \$1.50 / 1M tokens for input and \$2.0 / 1M tokens for output: https://openai.com/api/pricing/

Table 3: Ablation study using different LLMs.

Metric	LLM	Youtube	SMS(F1)	IMDB	Yelp	Agnews	Spouse(F1)	AVG
#LFs	GPT-3.5	108	201	225	247	225	43	174.8
	GPT-4	**115**	232	242	248	257	**66**	193.3
	Llama2-CHAT-7b	95	**243**	**363**	**309**	250	32	**215.3**
	Llama2-CHAT-13b	74	192	215	167	**266**	33	157.8
	Llama2-CHAT-70b	98	204	283	277	239	10	185.2
LF Acc.	GPT-3.5	0.735	0.884	0.726	0.772	0.824	–	0.788
	GPT-4	**0.856**	**0.923**	**0.752**	**0.801**	**0.847**	–	**0.836**
	Llama2-CHAT-7b	0.628	0.833	0.669	0.735	0.744	–	0.722
	Llama2-CHAT-13b	0.568	0.784	0.672	0.750	0.784	–	0.712
	Llama2-CHAT-70b	0.716	0.853	0.715	0.766	0.833	–	0.777
LF Cov.	GPT-3.5	0.021	0.009	0.024	**0.018**	0.003	0.034	0.018
	GPT-4	0.023	**0.011**	0.020	0.015	**0.004**	0.010	0.014
	Llama2-CHAT-7b	0.021	**0.011**	0.019	0.016	**0.004**	**0.062**	**0.022**
	Llama2-CHAT-13b	0.013	0.008	**0.021**	0.015	**0.004**	0.031	0.015
	Llama2-CHAT-70b	**0.024**	0.009	0.020	0.014	**0.004**	0.005	0.013
Total Cov.	GPT-3.5	0.820	0.754	0.977	0.947	0.480	**0.774**	**0.792**
	GPT-4	**0.838**	0.860	0.972	0.930	**0.590**	0.326	0.753
	Llama2-CHAT-7b	0.743	**0.869**	**0.988**	**0.956**	0.560	0.610	0.788
	Llama2-CHAT-13b	0.776	0.725	0.952	0.855	0.570	0.711	0.765
	Llama2-CHAT-70b	0.788	0.761	0.980	0.935	0.576	0.048	0.681
EM Acc/F1	GPT-3.5	0.879	0.829	0.794	**0.874**	0.758	**0.455**	0.765
	GPT-4	**0.898**	0.852	**0.801**	0.871	**0.818**	0.438	**0.780**
	Llama2-CHAT-7b	0.895	0.813	0.786	0.869	0.810	0.074	0.708
	Llama2-CHAT-13b	0.826	0.808	0.775	0.868	0.719	0.364	0.727
	Llama2-CHAT-70b	0.871	**0.855**	0.786	0.864	0.817	0.239	0.739

high-quality LFs by focusing on specific query instances rather than following general instructions. Note that as we are evaluating the average LF accuracy, this advantage is independent of the LF set size of DataSculpt or ScriptoriumWS. While PromptedLF generally achieves the best accuracy due to its instance-specific weak labels, it is also significantly more costly due to its exhaustive querying strategy.

Next, we examine the end model accuracy using LFs synthesized by different frameworks. DataSculpt-Base outperforms WRENCH on 4 out of 6 datasets and surpasses ScriptoriumWS on all evaluated datasets, consistent with the comparison of average LF accuracy. Specifically, DataSculpt excels in sentiment analysis tasks (IMDB and Yelp). DataSculpt-Base outperforms PromptedLF on 2 of 6 datasets. Averaging over all six datasets, DataSculpt-Base outperforms PromptedLF by 0.9 points regarding downstream model accuracy. Overall, while DataSculpt creates LFs based on only a subset of instances, its downstream model performance rivals expensive prompting strategies like PromptedLF.

Regarding coverage, a single LF curated by DataSculpt generally has lower coverage than other evaluated methods as they are based on a single keyword. However, DataSculpt compensates for this with larger LF sets, resulting in overall coverage comparable to the WRENCH baseline.

There is no significant difference in the accuracy or coverage of generated LFs among the evaluated versions of DataSculpt. However, the self-consistency methods (DataSculpt-SC and DataSculpt-KATE) help create a larger LF set by leveraging multiple responses from the LLM, thus enhancing LF diversity.

4.3 Comparative Studies

Next, we conduct comparative studies to evaluate the impact of different design choices in DataSculpt, including the choice of LLM, query instance selection methods, and LF filtering techniques. We use the DataSculpt-SC configuration in this section.

Pre-trained Language Models. We assess the performance of different pre-trained language models in DataSculpt. For GPT-3.5 and GPT-4, we use the *gpt-3.5-turbo-0613* and *gpt-4-0613* models from the OpenAI API [24]. For Llama2-CHAT models, we use the Anyscale AI API [4] and evaluate three variants of Llama2-CHAT models with 7 billion, 13 billion, and 70 billion parameters, respectively. The results are shown in Table 3.

Regarding LF accuracy, GPT-4 performs best, followed by GPT-3.5 and Llama2-CHAT-70b. Averaging across all datasets (except

Table 4: Ablation study using different samplers.

Metrics	Sampler	Youtube	SMS (F1)	IMDB	Yelp	Agnews	Spouse (F1)	AVG
#LFs	random	108	**201**	**225**	**247**	**225**	43	**174.8**
	uncertain	**118**	192	**225**	232	193	**79**	173.2
	SEU	25	62	112	77	126	23	70.8
LF Acc.	random	0.735	0.884	0.726	0.772	**0.824**	–	0.788
	uncertain	0.693	0.811	0.727	**0.776**	0.738	–	0.749
	SEU	**0.788**	**0.944**	**0.733**	0.772	0.754	–	**0.798**
LF Cov.	random	0.021	0.009	0.024	0.018	0.003	**0.034**	0.018
	uncertain	0.018	0.008	0.022	0.018	0.002	0.013	0.014
	SEU	**0.034**	**0.018**	**0.033**	**0.025**	**0.004**	0.006	**0.020**
Total Cov.	random	0.820	**0.754**	**0.977**	**0.947**	**0.480**	**0.774**	**0.792**
	uncertain	**0.832**	0.692	0.971	0.942	0.358	0.643	0.740
	SEU	0.501	0.652	0.937	0.798	0.362	0.090	0.557
EM Acc/F1	random	**0.879**	**0.829**	**0.794**	0.874	0.758	0.455	**0.765**
	uncertain	0.878	0.823	0.793	**0.877**	0.738	**0.461**	0.762
	SEU	0.810	0.730	0.787	0.835	**0.786**	0.452	0.733

Table 5: Ablation study using different LF filters.

Metrics	Filter	Youtube	SMS (F1)	IMDB	Yelp	Agnews	Spouse (F1)	AVG
#LF	all	108	201	225	247	225	43	174.8
	no accuracy	129	234	**368**	**371**	**325**	**53**	**246.7**
	no redundancy	**210**	**259**	294	337	261	**53**	235.7
LF Acc.	all	0.735	**0.884**	0.726	0.772	**0.824**	–	0.788
	no accuracy	0.690	0.773	0.640	0.685	0.678	–	0.693
	no redundancy	**0.822**	**0.884**	**0.732**	**0.776**	0.822	–	**0.807**
LF Cov.	all	0.021	0.009	0.024	0.018	0.003	0.034	0.018
	no accuracy	0.026	**0.011**	0.033	0.024	**0.005**	0.029	0.021
	no redundancy	**0.045**	0.008	**0.042**	**0.031**	0.004	**0.056**	**0.031**
Total Cov.	all	0.820	0.754	0.977	0.947	0.480	**0.774**	0.792
	no accuracy	**0.864**	**0.809**	**0.998**	**0.984**	**0.743**	0.771	**0.862**
	no redundancy	0.823	0.756	0.974	0.941	0.482	0.714	0.782
EM Acc/F1	all	**0.879**	0.829	0.794	**0.874**	0.758	**0.455**	**0.765**
	no accuracy	0.801	0.744	0.691	0.834	0.748	0.258	0.679
	no redundancy	0.851	**0.832**	**0.796**	0.855	**0.802**	0.284	0.737

Spouse), GPT-4 outperforms GPT-3.5 by 4.8 points and Llama2-CHAT-70b by 5.9 points in LF accuracy. The smaller Llama2 models generally exhibit lower accuracy and sometimes generate artificial examples instead of addressing the query directly. The same trend is observed in end model accuracy, where GPT-4 outperforms GPT-3.5 by 1.5 points and Llama2-CHAT-70b by 4.1 points. Although GPT-4 offers the highest accuracy, the gap between GPT-4 and the more cost-effective models like GPT-3.5 and Llama2-CHAT-70b is not substantial. Considering the balance between accuracy and cost, GPT-3.5 and Llama2-CHAT-70b are practical choices for LF design.

Query Instance Selection. Table 4 evaluates the impact of different query instance selection methods on LF statistics and end model accuracy. Random sampling and uncertain sampling produce LF sets of similar size, while SEU results in smaller LF sets. This is because SEU tends to select similar query instances, leading to redundancy pruned by LF filters. Regarding LF accuracy, random sampling and SEU are comparable, while uncertain sampling performs worse. Averaging across all datasets (except Spouse), random sampling outperforms uncertain sampling by 3.9 points, and SEU outperforms uncertain sampling by 4.9 points in LF accuracy. Uncertain sampling often selects difficult instances, which LLMs may not effectively label. For end model accuracy, random sampling generally performs best, outperforming uncertain sampling by 0.3 points and SEU by 3.2 points on average. Random sampling selects a diverse set of instances, helping DataSculpt generate a more comprehensive LF set, which enhances downstream model accuracy.

LF Filtering. We compare three LF filtering approaches: applying all filters, removing the accuracy filter, and removing the redundancy filter. Table 5 presents the results. Removing any filter increases the LF set size. Averaging across all datasets, omitting the accuracy filter reduces LF accuracy by 9.5 points and end model accuracy by 8.6 points, highlighting the importance of filtering out inaccurate LFs. The impact of removing the redundancy filter varies by dataset. In 3 out of 6 datasets, end model accuracy improves when the redundancy filter is omitted, likely because high-quality LFs are retained. Our findings suggest always using the accuracy filter to ensure downstream model accuracy, while the redundancy filter's use may depend on the specific dataset.

5 DISCUSSIONS

Based on our evaluation, we provide a few takeaways for leveraging LLMs to design LFs:

- **T1:** LLMs perform well in designing keyword-based LFs for text classification. Still, verifying the accuracy of auto-generated LFs (e.g., using a labeled validation set) is crucial for enhancing downstream model accuracy.
- **T2:** While prompting methods like chain-of-thought and *KATE* improve the LLM's accuracy in some datasets, their performance is dataset-dependent, and more accurate predictions do not always lead to more accurate LFs.
- **T3:** Current active query instance selection methods do not work well in prompting LLMs to design LFs because they do not consider the imperfect nature of LLMs or learn from LLM feedback. Further research is required to create more effective selection methods for LLM prompting.

The current framework has a few limitations. First, it focuses on text classification tasks and design keyword-based label functions; thus, further investigation is required to verify how well the approach can generalize to other label functions and tasks. Second, as the framework requires a labeled validation set for pruning out the inaccurate LFs, the user must manually label a small set of data. Lastly, our work does not revise the LFs developed by LLMs. Future works could consider an iterative prompting strategy to enhance LF quality further.

6 CONCLUSIONS

In this paper, we proposed DataSculpt, a novel framework that leverages large language models to design label functions automatically for programmatic weak supervision. DataSculpt leverages few-shot learning to create a massive number of label functions cost-efficiently. Experiments on six real-world datasets show that DataSculpt can create LFs with high accuracy at only a fraction of the cost of exhaustive querying methods.

REFERENCES

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774* (2023).

[2] Túlio C Alberto, Johannes V Lochter, and Tiago A Almeida. 2015. Tubespam: Comment spam filtering on youtube. In *2015 IEEE 14th international conference on machine learning and applications (ICMLA)*. IEEE, 138–143.

[3] Tiago A Almeida, José María G Hidalgo, and Akebo Yamakami. 2011. Contributions to the study of SMS spam filtering: new collection and results. In *Proceedings of the 11th ACM symposium on Document engineering*. 259–262.

[4] Anyscale. [n.d.]. *Anyscale API.* https://www.anyscale.com/endpoints#hosted

[5] Benedikt Boecking, Willie Neiswanger, Eric Xing, and Artur Dubrawski. 2020. Interactive weak supervision: Learning useful heuristics for data labeling. *arXiv preprint arXiv:2012.06046* (2020).

[6] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.

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

[8] David PA Corney, Dyaa Albakour, Miguel Martinez-Alvarez, and Samir Moussa. 2016. What do a million news articles look like?. In *NewsIR@ ECIR*. 42–47.

[9] Benjamin Denham, Edmund MK Lai, Roopak Sinha, and M Asif Naeem. 2022. Witan: unsupervised labelling function generation for assisted data programming. *Proceedings of the VLDB Endowment* 15, 11 (2022), 2334–2347.

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[11] Daniel Fu, Mayee Chen, Frederic Sala, Sarah Hooper, Kayvon Fatahalian, and Christopher Ré. 2020. Fast and three-rious: Speeding up weak supervision with triplet methods. In *International Conference on Machine Learning*. PMLR, 3280–3291.

[12] Sainyam Galhotra, Behzad Golshan, and Wang-Chiew Tan. 2021. Adaptive rule discovery for labeling text data. In *Proceedings of the 2021 International Conference on Management of Data*. 2217–2225.

[13] Naiqing Guan and Nick Koudas. 2024. ActiveDP: Bridging Active Learning and Data Programming. *arXiv preprint arXiv:2402.06056* (2024).

[14] Cheng-Yu Hsieh, Jieyu Zhang, and Alexander Ratner. 2022. Nemo: Guiding and Contextualizing Weak Supervision for Interactive Data Programming. *Proceedings of the VLDB Endowment* 15, 13 (2022), 4093–4105.

[15] Tzu-Heng Huang, Catherine Cao, Spencer Schoenberg, Harit Vishwakarma, Nicholas Roberts, and Frederic Sala. 2023. ScriptoriumWS: A Code Generation Assistant For Weak Supervision. In *ICLR Deep Learning for Code Workshop*.

[16] Moe Kayali, Anton Lykov, Ilias Fountalis, Nikolaos Vasiloglou, Dan Olteanu, and Dan Suciu. 2024. Chorus: Foundation Models for Unified Data Discovery and Exploration. *Proceedings of the VLDB Endowment* 17, 8 (2024), 2104–2114.

[17] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems* 35 (2022), 22199–22213.

[18] David D Lewis. 1995. A sequential algorithm for training text classifiers: Corrigendum and additional data. In *Acm Sigir Forum*, Vol. 29. ACM New York, NY, USA, 13–19.

[19] Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2021. What Makes Good In-Context Examples for GPT-3? *arXiv preprint arXiv:2101.06804* (2021).

[20] Peng Liu, Lizhe Wang, Rajiv Ranjan, Guojin He, and Lei Zhao. 2022. A survey on active deep learning: from model driven to data driven. *ACM Computing Surveys (CSUR)* 54, 10s (2022), 1–34.

[21] Andrew Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*. 142–150.

[22] Avanika Narayan, Ines Chami, Laurel Orr, and Christopher Ré. 2022. Can Foundation Models Wrangle Your Data? *Proceedings of the VLDB Endowment* 16, 4 (2022), 738–746.

[23] Mona Nashaat, Aindrila Ghosh, James Miller, Shaikh Quader, Chad Marston, and Jean-Francois Puget. 2018. Hybridization of active learning and data programming for labeling large industrial datasets. In *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, 46–55.

[24] OpenAI. [n.d.]. *OpenAI API*. https://platform.openai.com/

[25] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems* 35 (2022), 27730–27744.

[26] Alexander Ratner, Stephen H Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. 2017. Snorkel: Rapid training data creation with weak supervision. In *Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases*, Vol. 11. NIH Public Access, 269.

[27] Alexander Ratner, Braden Hancock, Jared Dunnmon, Frederic Sala, Shreyash Pandey, and Christopher Ré. 2019. Training complex models with multi-task weak supervision. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 4763–4771.

[28] Alexander J Ratner, Christopher M De Sa, Sen Wu, Daniel Selsam, and Christopher Ré. 2016. Data programming: Creating large training sets, quickly. *Advances in neural information processing systems* 29 (2016), 3567–3575.

[29] Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Brij B Gupta, Xiaojiang Chen, and Xin Wang. 2021. A survey of deep active learning. *ACM computing surveys (CSUR)* 54, 9 (2021), 1–40.

[30] Ozan Sener and Silvio Savarese. 2018. Active Learning for Convolutional Neural Networks: A Core-Set Approach. In *International Conference on Learning Representations*.

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

[32] Burr Settles. 2012. Active learning. *Synthesis lectures on artificial intelligence and machine learning* 6, 1 (2012), 1–114.

[33] H Sebastian Seung, Manfred Opper, and Haim Sompolinsky. 1992. Query by committee. In *Proceedings of the fifth annual workshop on Computational learning theory*. 287–294.

[34] Ryan Smith, Jason A Fries, Braden Hancock, and Stephen H Bach. 2022. Language models in the loop: Incorporating prompting into weak supervision. *ACM/JMS Journal of Data Science* (2022).

[35] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. LLaMA: Open and Efficient Foundation Language Models. arXiv:cs.CL/2302.13971

[36] Immanuel Trummer. 2023. Can Large Language Models Predict Data Correlations from Column Names? *Proceedings of the VLDB Endowment* 16, 13 (2023), 4310–4323.

[37] Paroma Varma and Christopher Ré. 2018. Snuba: Automating weak supervision to label training data. In *Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases*, Vol. 12. NIH Public Access, 223.

[38] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171* (2022).

[39] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems* 35 (2022), 24824–24837.

[40] Jieyu Zhang, Cheng-Yu Hsieh, Yue Yu, Chao Zhang, and Alexander Ratner. 2022. A survey on programmatic weak supervision. *arXiv preprint arXiv:2202.05433* (2022).

[41] Jieyu Zhang, Yue Yu, Yinghao Li, Yujing Wang, Yaming Yang, Mao Yang, and Alexander Ratner. 2021. WRENCH: A Comprehensive Benchmark for Weak Supervision. *arXiv preprint arXiv:2109.11377* (2021).

[42] Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *Advances in neural information processing systems* 28 (2015).

[43] Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2022. Automatic chain of thought prompting in large language models. *arXiv preprint arXiv:2210.03493* (2022).