The ALCHEmist: Automated Labeling 500x CHEaper Than LLM Data Annotators

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

Large pretrained models can be used as annotators, helping replace or augment crowdworkers and enabling distilling generalist models into smaller specialist models. Unfortunately, this comes at a cost: employing top-of-the-line models often requires paying thousands of dollars for API calls, while the resulting datasets are static and challenging to audit. To address these challenges, we propose a simple alternative: rather than directly querying labels from pretrained models, we task models to *generate programs that can produce labels*. These programs can be stored and applied locally, re-used and extended, and cost orders of magnitude less. Our system, **Alchemist**, obtains comparable to or better performance than large language model-based annotation in a range of tasks for a fraction of the cost: on average, improvements amount to a **12.9%** enhancement while the total labeling costs across all datasets are reduced by a factor of approximately **500**×.

1 Introduction

One of the most exciting developments in machine learning is the use of large pretrained models to act as *annotators* or *labelers* [1, 2, 3, 4, 5, 6, 7, 8]. This includes the use of large language models (LLMs) like GPT-4 [9] and Claude 3 [10]. This process offers multiple benefits. First, pretrained models are an efficient way to annotate and have the potential to partially or fully replace expensive human crowdworkers [2, 6]. Second, this approach allows for *distilling* large models into smaller, task-specific models that can be deployed locally at lower cost [3, 11, 7, 8]. This is additionally important in settings like healthcare and finance where privacy laws require the use of local models.

Despite this promise, pretrained model-based annotation has several drawbacks that stymie its adoption. These drawbacks include

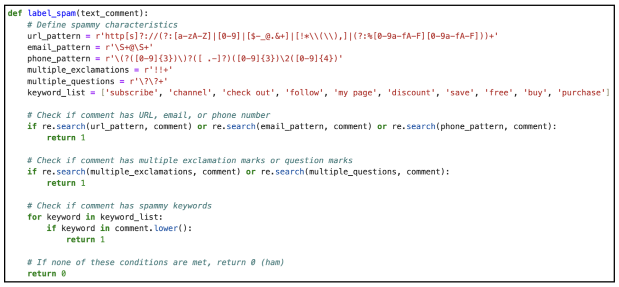
- **High Cost**: Labeling a dataset can be expensive. This is particularly so in cases where each data point consists of many tokens. For example, we find that labeling a moderately-sized dataset [12] with 7,569 data points using GPT-4 costs over $1,200.
- **Lack of Extensibility**: Making even small changes to specifications necessitates re-running the entire pipeline to obtain new labels. This inflexibility means the resulting labels are static.
- **Inability to Audit**: API access to pretrained models does not permit inspecting most aspects of the model. Users must simply accept the provided labels with only minimal additional information. Techniques that ask the model for explanations for its decisions may not be reliable [13, 14, 15].

We address these obstacles through a simple but surprisingly powerful notion. Rather than having pretrained models label data, we task language models to ***generate programs that can output labels***. These synthesized programs serve as annotators, capturing the underlying logic used by the models when annotating. In other words, instead of distilling a powerful model to label a dataset (and subsequently training a smaller model on the labeled data), we *distill directly into code* (Figure 1).

Prompt Template in Alchemist

[Task Description]	‣ Write a bug-free and executable function in python to label comment on Youtube as spam or ham.
[Labeling Instruction]	‣ If the comment is spam, function returns 1. If the comment is ham, function returns 0. ‣ If the comment cannot be determined as spam or ham, function returns -1.
[Function Signature]	‣ def label_spam (text_comment):

[Task Description]	‣ Write a bug-free and executable function in python to label the topic of biomedical document.
[Labeling Instruction]	‣ If document is about colon cancer, returns 0. If document is about lung cancer, returns 1. ‣ If document is about thyroid cancer, returns 2. If document cannot be categorized, returns -1.
[Function Signature]	‣ def label_cancer (document):



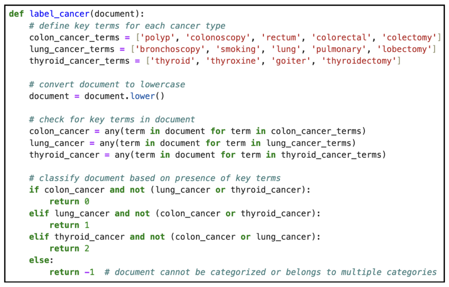
```python
def label_spam(text_comment):
    # Define spammy characteristics
    url_pattern = r'http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|[!*\\(\\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))+'
    email_pattern = r'\S+@\S+'
    phone_pattern = r'\(?([0-9]{3})\)?([ .-]?)([0-9]{3})\2([0-9]{4})'
    multiple_exclamations = r'!!+'
    multiple_questions = r'\?\?+'
    keyword_list = ['subscribe', 'channel', 'check out', 'follow', 'my page', 'discount', 'save', 'free', 'buy', 'purchase']

    # Check if comment has URL, email, or phone number
    if re.search(url_pattern, comment) or re.search(email_pattern, comment) or re.search(phone_pattern, comment):
        return 1

    # Check if comment has multiple exclamation marks or question marks
    if re.search(multiple_exclamations, comment) or re.search(multiple_questions, comment):
        return 1

    # Check if comment has spammy keywords
    for keyword in keyword_list:
        if keyword in comment.lower():
            return 1

    # If none of these conditions are met, return 0 (ham)
    return 0
```



```python
def label_cancer(document):
    # define key terms for each cancer type
    colon_cancer_terms = ['polyp', 'colonoscopy', 'rectum', 'colorectal', 'colectomy']
    lung_cancer_terms = ['bronchoscopy', 'smoking', 'lung', 'pulmonary', 'lobectomy']
    thyroid_cancer_terms = ['thyroid', 'thyroxine', 'goiter', 'thyroidectomy']

    # convert document to lowercase
    document = document.lower()

    # check for key terms in document
    colon_cancer = any(term in document for term in colon_cancer_terms)
    lung_cancer = any(term in document for term in lung_cancer_terms)
    thyroid_cancer = any(term in document for term in thyroid_cancer_terms)

    # classify document based on presence of key terms
    if colon_cancer and not (lung_cancer or thyroid_cancer):
        return 0
    elif lung_cancer and not (colon_cancer or thyroid_cancer):
        return 1
    elif thyroid_cancer and not (colon_cancer or lung_cancer):
        return 2
    else:
        return -1  # document cannot be categorized or belongs to multiple categories
```

Figure 1: Examples of generated programs and their prompts. These are synthesized by GPT-4 for spam detection and cancer identification tasks. Programs use regular expressions (left program) and keyword matching (right program) as their labeling logic to classify data points.

These resulting programs can either make predictions directly or can label training dataset then train a downstream model using it[1].

This simple notion resolves all of the challenges related to pretrained model-based annotation. First, ***API calls scale with the number of programs instead of the number of data points.*** That is, since we generate programs that can themselves make any number of predictions locally at no cost, we can reduce the number of API calls by orders of magnitude. For example, for the dataset described above [12], the number of GPT-4 calls was reduced from 7,569 (the size of the dataset) to 10 (the number of generated programs), resulting in a massive cost reduction from $1,200 to $0.70, a 1,700-fold decrease. Moreover, code can be easily inspected, corrected, and extended, allowing seamless adaptation when prediction classes or labeling rules change.

While a powerful idea, distilling model into code presents several challenges. First, any particular program may be inaccurate, fail to compile, or may otherwise be flawed, resulting in noisy program outputs. We address this obstacle by applying *weak supervision*, a framework for dataset construction from multiple noisy sources of signal [16, 17, 18, 19]. Next, operating on non-text modalities is challenging. We handle this via a simple two-step approach that first extracts high-level concepts and then uses them in concert with a local feature extractor to enable tractable program generation.

Contributions. We propose an alternative approach to replace expensive annotation processes that require repetitive prompting for labels. We developed a system called ***Alchemist*** that implements this idea. Empirically, Alchemist improves performance five out of eight datasets, with an average enhancement of 12.9%—while reducing total costs by a factor of approximately $500\times$. Finally, we introduce and validate extensions that address non-text modalities.

2 Related Work

Our work relates to LLM-based annotation, prompting, and the weak supervision framework.

Using Large Pretrained Models for Data Annotation. Large pretrained models have demonstrated powerful capabilities using zero-shot prompting across a wide range of tasks [1]. One promising development is their potential to serve as data labelers, which can reduce the cost and human effort in data labeling [1, 2]. Existing research in this area mainly focuses on approaches that allow

[1]The latter option is preferable, as these models can often generalize beyond their source of supervision [11]

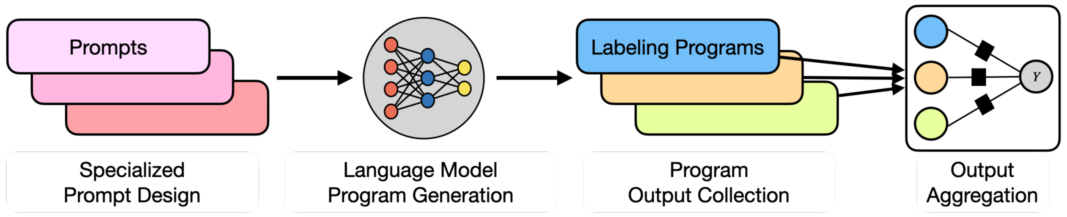

Prompts
Labeling Programs
Y
Specialized Prompt Design
Language Model Program Generation
Program Output Collection
Output Aggregation

Figure 2: Overall workflow for Alchemist.

for more efficient inference, enhanced label generation, and distilling into smaller but specialized labelers [3, 4, 5, 6, 7, 8]. However, *scalability* is the main limitation in these approaches, as making inferences via querying an API for data examples can be cost-prohibitive. To tackle this challenge, rather than prompting for labels repetitively, we propose prompting pretrained models for programs that use synthesized labeling logic and can thus serve as alternative data labelers.

Prompt Engineering & In-Context Learning. In-context learning adapts pretrained models to new tasks without additional fine-tuning [1]. It involves providing relevant examples as demonstrations to solve the task, such as pairs of languages for translation [20]. By including task-specific examples, models can better understand the task at hand. Adding a few data points as demonstrations [21] is commonly suggested when models act as data annotators. Moreover, they can be selected [22, 23], retrieved [24], or more efficiently, generated [25]. We explore various types of supplementary information that can be added to Alchemist to help improve program generation and permit more control over the labeling logic used in the programs.

Weak Supervision Framework. Weak supervision enables the rapid creation of large training datasets by aggregating cheap-but-noisy signals derived from various labeling sources [16, 17, 19, 26]. These sources can be crafted by domain expertise, using labeling heuristics, or even trained on smaller, weaker classifiers [27, 28, 29, 30]. Recent advancements in code generation open up the potential to automate the heuristic design process. Frameworks such as ScriptoriumWS [31], and DataSculpt [32] have been developed to take advantage of code-generating models [33, 9, 34] to craft weak supervision sources through prompting. While similar in spirit to our approach, these have several drawbacks: ScriptoriumWS requires more human effort in prompt engineering to better guide code-generation models. Both ScriptoriumWS and DataSculpt can perform poorly in tasks requiring specific domain expertise and, most importantly, they do not handle modalities beyond text—unlike Alchemist.

3 Alchemist System

We begin by presenting a general annotation workflow in Alchemist, followed by a detailed discussion of each key step.

General Workflow. The process is depicted in Fig. 2. First, users select an unlabeled dataset and create simple prompts to instruct language models to generate programs that incorporate labeling logic. These prompts can integrate relevant information and may vary in their design, allowing for the synthesis of multiple programs. Next, given a set of generated programs and their outputs, we apply weak supervision techniques to obtain a set of aggregated labels. Finally, the labeled points can be used to train a distilled model that can be stored and used locally.

3.1 Prompting Strategy

We propose a general and extensible prompt template for querying language models to generate annotator programs. This general template consists of three key components:

- **Task Description**: Provides the model an overview of generated program's desired objectives.
- **Labeling Instructions**: Specifies classes and the expected structure of the program's output.
- **Function Signature**: Describes the function's name and the input types to be used.

This simple but general template allows for flexible incorporation of various types of information, enabling the model to generate programs that are tailored to specific requirements. Two sample prompt templates in Alchemist are displayed in Fig 1.

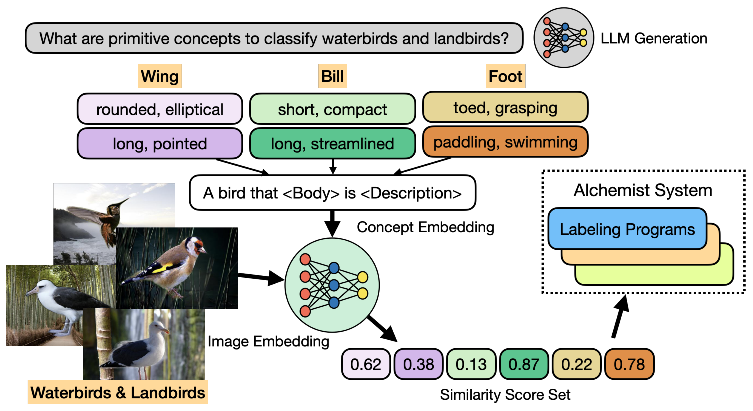


Figure 3: Alchemist can handle rich modalities through a simple extension. First, a language model identifies task-specific concepts (top). Then, a local multimodal model is used as a feature extractor for these concepts, producing low-dimensional feature vectors that can be ingested by generated labeling programs.

Using Supplementary Information. Drawing inspiration from few-shot prompting [35, 1], where users provide demonstrations (i.e., data points with their labels) to enhance generated responses, we explore various types of supplementary information that can be integrated to assist models in synthesizing programs. This approach is particularly useful for scenarios where language models may lack the expertise to generate effective programs, or where specific adaptations in labeling logic are required. Such information can be crafted by users themselves, domain experts or, more efficiently, generated by language models themselves. Additionally, it can be combined with retrieval-augmented generation (RAG) systems [36, 37] to access external knowledge.

We explore various types of supplementary information to assist in code generation, starting with high-level concepts and then progressively looking into more practical details to control programs.

Dataset and Prediction Class Description. First, supplementary information can include relevant background details about the purpose for which the dataset was built and high-level information about the dataset, such as definitions for each label class. By providing this context, the language model can better understand the task at hand.

Data Exemplars. Furthermore, we recommend including a small number of labeled data examples in the prompt. This can help language models better comprehend the specific problem. Examples act as concrete illustrations of the task, offering a clearer understanding of the expected output. This can be particularly beneficial when dealing with a complex problem.

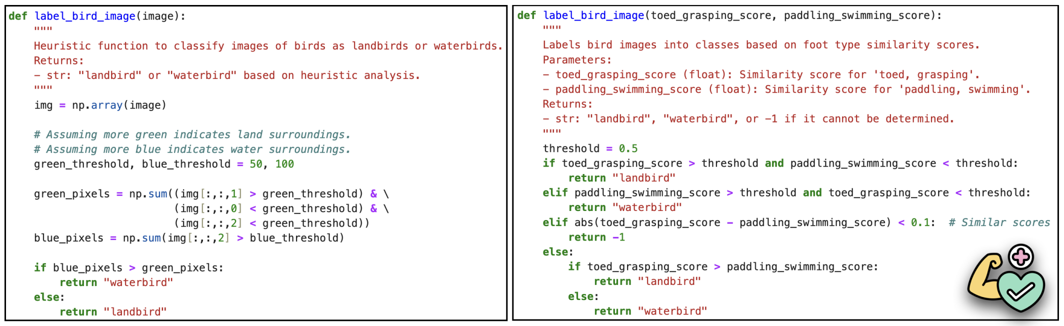
Keywords. Next, labeling logic in programs can make use of keyword-searching techniques (e.g., Fig 1). For instance, in situations such as spam detection or topic classification, certain words or phrases may have a strong correlation with specific classifications. Providing several keywords in the prompt may lead models to create labeling programs that explicitly search for the presence or absence of these keywords. This allows for more targeted and precise labeling.

Specialized Labeling Rules. Finally, more prior knowledge such as heuristics, specialized labeling rules, guidance, and domain-specific knowledge can be integrated into the prompt. This information can provide concrete labeling steps on how to label specific classes and offer greater control over the logic implemented in the generated programs.

Overall, supplementary context is provided before the task description to enhance language models' understanding of the task. This, in turn, enables models to generate programs that are more effective and tailored to the specific requirements of user needs.

3.2 Dataset Synthesis

While generated programs can efficiently annotate data, these programs may produce outputs that are noisy or inaccurate. However, as such programs may employ different techniques, such as pattern-matching, heuristic rules, or other approaches—each with its own strengths and limitations—there



```python
def label_bird_image(image):
    """
    Heuristic function to classify images of birds as landbirds or waterbirds.
    Returns:
    - str: "landbird" or "waterbird" based on heuristic analysis.
    """
    img = np.array(image)

    # Assuming more green indicates land surroundings.
    # Assuming more blue indicates water surroundings.
    green_threshold, blue_threshold = 50, 100

    green_pixels = np.sum((img[:,:,1] > green_threshold) & \
                          (img[:,:,0] < green_threshold) & \
                          (img[:,:,2] < green_threshold))
    blue_pixels = np.sum(img[:,:,2] > blue_threshold)

    if blue_pixels > green_pixels:
        return "waterbird"
    else:
        return "landbird"
```

```python
def label_bird_image(toed_grasping_score, paddling_swimming_score):
    """
    Labels bird images into classes based on foot type similarity scores.
    Parameters:
    - toed_grasping_score (float): Similarity score for 'toed, grasping'.
    - paddling_swimming_score (float): Similarity score for 'paddling, swimming'.
    Returns:
    - str: "landbird", "waterbird", or -1 if it cannot be determined.
    """
    threshold = 0.5
    if toed_grasping_score > threshold and paddling_swimming_score < threshold:
        return "landbird"
    elif paddling_swimming_score > threshold and toed_grasping_score < threshold:
        return "waterbird"
    elif abs(toed_grasping_score - paddling_swimming_score) < 0.1:  # Similar scores
        return -1
    else:
        if toed_grasping_score > paddling_swimming_score:
            return "landbird"
        else:
            return "waterbird"
```

Figure 4: Program examples generated by GPT4o on Waterbirds dataset. The left program is synthesized by directly asking for a labeling program when the input is an image (raw pixels), while the right program uses Alchemist's extension. The former labels birds using the dominant color in the image, which can be predicted incorrectly due to spurious correlations (e.g., background).

may be *complementary* signal in their outputs. This means we can aggregate them to mitigate the impact of noise. To do so, we apply weak supervision techniques [16, 17, 18, 19]. This process starts by learning a model of the reliabilities of the programs. Once learned, this model enables aggregating label outputs from different programs into high-quality *pseudolabels*.

Alchemist is compatible with a variety of weak supervision aggregation models, called *label models*, providing flexibility in the choice of the weak supervision approach. For simplicity, in this work, we focus on using the Snorkel framework [17], which is a standard and widely-used approach in the weak supervision community.

3.3 Extensions: Handling Complex Modalities.

Crafting programs that operate over text is relatively easy for large language models. More complex data modalities, however, can be far more challenging. Consider images as an illustrative example. Even employing state-of-the-art multimodal models, e.g., GPT-4o [38] and GPT-4V [9], to seek programs operating over sample images may not produce satisfactory results.

To address this challenge, we extend Alchemist's pipeline to include an intermediate step. Specifically, we convert the raw data (i.e., in our example, image pixels) into a set of features representing high-level concepts. These concepts are obtained by prompting a language model (or, potentially, a multimodal model) to identify task-relevant notions. For example, for a bird categorization task, models may identify "wing shape," "beak shape," or "foot type" as informative concepts for distinguishing between bird species. Next, we use any open-source local multimodal model, like CLIP [39], as a feature extractor for the identified concepts, producing low-dimensional feature vectors that can be easily ingested by generated programs. As such models are free, this does not increase our cost.

Fig. 3 and Fig. 4 present examples of generated high-level concepts and the corresponding programs used for the Waterbirds dataset, where the task is to distinguish between landbird and waterbird species [40]. This simple approach can be applied to any data modality where we have access to a local multimodal model (i.e., a model operating on the modality of interest and text).

4 Experiments

We study the capability of Alchemist empirically. Our goals are to validate the following claims:

- **Cost Reduction and Improved Performance (Sec. 4.1):** Alchemist can reduce cost by orders of magnitude, while producing labels of similar or better accuracy.
- **Extendibility to Other Modalities (Sec. 4.2):** Alchemist can operate with modalities beyond text.
- **Use of Supplementary Information (Sec. 4.3):** Incorporating relevant information into prompts enables the generation of better programs, yielding more accurate pseudolabels.

	YouTube		SMS		Yelp		IMDb	
	Est. Cost	Accuracy	Est. Cost	F1-score	Est. Cost	Accuracy	Est. Cost	Accuracy
Zero-shot Prompting	0.096	0.871	0.240	**0.907**	3.873	**0.845**	3.400	**0.737**
Alchemist with GPT-3.5	0.004	**0.891**	0.004	0.900	0.005	0.575	0.004	0.662

	MedAbs		Cancer		Finance		French	
	Est. Cost	Accuracy	Est. Cost	Accuracy	Est. Cost	Accuracy	Est. Cost	Accuracy
Zero-shot Prompting	1.944	0.311	15.925	0.716	0.201	0.641	0.641	0.611
Alchemist with GPT-3.5	0.006	**0.346**	0.003	**0.968**	0.007	**0.660**	0.006	**0.690**

Table 1: Testing performance of the distilled model is reported for each combination of method and dataset. The estimated cost is obtained by calculating the number of input and output tokens associated with GPT-3.5's pricing table [47]. Other models may be even more expensive.

- **More Diverse Programs Can Help (Sec. 4.4):** Increasing the diversity of generated programs created by different labeling logic enables better pseudo labels.
- **Comparing to Human-crafted Programs (Sec 4.5):** Synthesized programs may be more effective in comparison to human-crafted ones.

Datasets. We include diverse datasets covering text and image modalities. For text, we include eight datasets that span three different types of language tasks. These include the YouTube [41], SMS [42] datasets for spam classification, IMDb [43], Yelp [43], Finance [44], and French [45] datasets for sentiment analysis, and the MedAbs [46] and Cancer [12] datasets for topic classification. We note that the Finance, French, MedAbs, and Cancer datasets are relatively challenging, with points that require a degree of domain expertise for accurate labeling. For example, the French dataset requires a good understanding of the language. These may pose challenges for pretrained models.

For our extensions to richer modalities, we focus on image tasks. Our evaluation uses the Waterbirds dataset [40]. This dataset is designed to assess models' robustness to spurious correlations and ability to handle distribution shifts. More details are in Appendix A.

4.1 Cost Reduction and Improved Performance

Setup. We open our evaluation of Alchemist with text domain datasets and use GPT-3.5 to generate programs. For each dataset, we input pure prompts without supplementary information into GPT-3.5 and generate 10 programs to use. We construct training datasets by aggregating the programs' outputs into pseudolabels with the weak supervision framework Snorkel [17]. We then train a two-layer MLP as a distilled model. We run five times with different random seeds and report their average performance. As our main baseline, we directly use language models to produce annotations per point. The resulting labels are used to train a distilled model for comparison. The prompt template used in our baseline approach and our training settings are provided in Appendix A.

Expected Results. We anticipate that Alchemist can generate programs that can produce accurate labels while substantially reducing the expense of API calls.

Results. Table 1 presents the distilled model's performance on each testing dataset. We observe that label accuracy is improved on five out of eight datasets, particularly in challenging settings such as the MedAbs, Cancer, and French datasets, outperforming the baseline zero-shot prompting approach. We also report the estimated costs of building training datasets. The costs for zero-shot prompting depend on the number of tokens for the dataset. In contrast, Alchemist only prompts 10 programs for each task, resulting in a significant reduction in the costs—by orders of magnitude. This efficiency is the main advantage of Alchemist, ***as it allows for the creation of high-quality datasets with minimal expense.*** We include ablation studies with other weak supervision models within the Alchemist framework in Appendix C. They successfully demonstrate the flexibility and robustness of using Alchemist.

4.2 Extending Alchemist to Other Modalities

Setup. Next, we validate the extension of Alchemist to richer modalities. We consider our approach, where we prompt a multimodal model such as GPT4o and Claude 3, to generate high-level task-

Feature Extractor	Method	Average Accuracy (↑)	Worst Group Accuracy (↑)	Gap (↓)
—	**Vanilla Alchemist with GPT4o**	0.395	0.367	0.028
	Vanilla Alchemist with Claude 3	0.781	0.022	0.759
CLIP ViT-B/32	Zero-shot Prompting	0.820	0.318	0.502
	Group Prompting	**0.823**	0.383	0.440
	Alchemist with GPT4o	0.805	0.283	0.522
	Alchemist with Claude 3	0.774	**0.463**	**0.410**
CLIP ViT-L/14	Zero-shot Prompting	**0.904**	0.335	0.569
	Group Prompting	0.791	0.240	0.551
	Alchemist with GPT4o	0.802	**0.467**	**0.335**
	Alchemist with Claude 3	0.737	0.346	0.391

Table 2: Alchemist on non-text modalities. We experiment with standard Alchemist (top), our proposed extension with two CLIP-based local models as feature extractors, and CLIP prompting baselines. Alchemist achieves comparable performance on average accuracy while improving robustness to spurious correlations.

specific concepts. We extract features for these concepts by employing CLIP as our local feature extractor. This converts raw pixels into feature vectors for the extracted high-level concepts, producing a set of similarity scores. Armed with these scores, we describe scores associated with their concepts in prompts and ask GPT4o and Claude 3 for 10 programs. As before, we use Snorkel as our aggregation procedure.

Baselines. We study two baselines. The first is the vanilla version of Alchemist, where we directly ask GPT4o and Claude 3 to produce code that can operate on images (see left program in Fig. 4). The second is simple zero-shot prompting using CLIP, along with a variant, a group prompting approach that assumes access to spurious information and adds it to the given prompt[2].

Expected Results. We expect employing our two-step process can enable tractable program generation. In addition, we hypothesize that programs generated in this way are beneficial in targeting salient concepts and reducing the impact of irrelevant or shortcut features, thereby enhancing robustness.

Results. We present results in Table 2. Our evaluation focuses on three key metrics: average accuracy, worst group accuracy, and the gap between these two measures. Ideally, a robust model should achieve high average accuracy and high worst group accuracy while minimizing the disparity between the two. First, we see that directly asking programs to use may have very low performance (GPT4o) or may hugely suffer from spurious correlations, destroying worst group performance (Claude 3, CLIP zero-shot). Our method addresses both cases. Compared to baseline methods, Alchemist demonstrates increased worst group accuracy and a reduced gap between the average and worst group accuracies. This is a key strength of Alchemist: ***targeting salient concepts to be used as features may help move models away from spurious shortcuts found in the data***. This validates Alchemist's ability to handle complex modalities while improving robustness.

4.3 Use of Supplementary Information

Setup. We test how integrating relevant information into the prompt context can augment generated programs. Instead of manually crafting supplementary information, we harness the power of language models to generate and integrate. This approach is useful for challenging datasets where users may not have the necessary knowledge or expertise to start. We evaluate the effectiveness of this approach, by comparing label model performance using programs generated by two different methods: pure prompting and in-context prompting. In-context prompting involves supplementary information, while pure prompting relies solely on the task description without any additional guidance. We employ GPT-4 and Claude 3 as our program sources and synthesize ten for each strategy.

[2]the group prompts are "waterbird on water background", "waterbird on land background", "landbird on water background", and "landbird on land background".

	YouTube			SMS			Yelp			IMDb		
	GPT-3.5	GPT-4	Claude 3	GPT-3.5	GPT-4	Claude 3	GPT-3.5	GPT-4	Claude 3	GPT-3.5	GPT-4	Claude 3
General Prompt	0.92	0.92	0.66	0.64	0.62	0.75	0.65	0.82	0.78	0.71	0.77	0.77
+ Dataset Description	0.64	**0.93**	**0.71**	0.63	**0.63**	**0.76**	**0.72**	0.82	**0.79**	0.70	**0.79**	0.73
+ 5 Data Exemplars	0.91	0.86	**0.76**	0.46	**0.66**	0.62	**0.72**	0.82	**0.82**	0.68	0.75	0.73
+ Keywords	0.76	**0.93**	0.53	0.40	0.42	0.64	**0.69**	0.81	0.78	0.69	**0.78**	0.72
+ Labeling Rules	0.74	0.82	0.56	**0.67**	**0.67**	0.58	**0.75**	0.81	**0.79**	0.71	0.77	0.74
	MedAbs			Cancer			Finance			French		
	GPT-3.5	GPT-4	Claude 3	GPT-3.5	GPT-4	Claude 3	GPT-3.5	GPT-4	Claude 3	GPT-3.5	GPT-4	Claude 3
General Prompt	0.52	0.53	0.55	0.71	0.73	0.59	0.66	0.49	0.56	0.65	0.55	0.56
+ Dataset Description	0.49	0.50	0.51	0.59	0.62	**0.60**	0.61	**0.63**	**0.62**	0.39	**0.58**	**0.67**
+ 5 Data Examples	**0.53**	**0.54**	0.55	0.55	0.57	**0.63**	0.60	**0.50**	**0.60**	0.40	**0.69**	0.44
+ Keywords	**0.55**	**0.55**	0.55	0.55	0.55	0.46	0.66	**0.62**	**0.65**	**0.69**	**0.66**	**0.67**
+ Labeling Rules	0.52	**0.55**	**0.56**	0.61	0.59	**0.63**	0.66	**0.56**	**0.67**	0.65	**0.66**	0.33

Table 3: Testing performance of the label model is reported for each combination of prompting strategy and dataset. We observe that GPT-4 and Claude 3 (that may possess better comprehension capabilities) exhibit greater enhancements when provided with supplementary information.

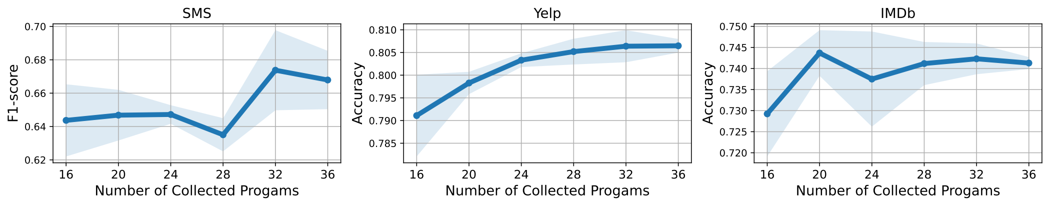


Figure 5: Performance is reported using their average performance and standard deviations. Results indicate that the label model is improved when the number of diverse programs increases.

Expected Results. We hypothesize that providing supplementary information can enhance task understanding, demonstrate specific labeling logic, and offer concrete steps, ultimately leading to better programs for use.

Results. Table 3 presents this comparative analysis on label model performance using different type of information. We observe that by incorporating supplementary information into pure prompts, Alchemist can guide language models to generate more effective programs, which in turn produce more accurate pseudolabels. Improvements are particularly evident in the challenging datasets such as Finance and French. Moreover, this approach can be combined with RAG systems to include external knowledge bases and customize the relevant information. Such flexibility compared to zero-shot prompting is another key strength of Alchemist, as ***programs can easily be adapted, augmented, and specialized.***

4.4 More Diverse Programs Can Help

Setup. As shown in Table 3, incorporating different supplementary information results in varying degrees of additional improvement. Potentially, certain sets of supplementary information allow the model to specialize better on certain data points than others. We seek to achieve these performance improvements *without* the need to re-prompt the model with each set of supplementary information. Instead, we collect previously generated programs to obtain a set of programs with greater diversity. We ask: *can Alchemist achieve better performance by modeling more diverse programs?*

We randomly select a set of programs from each category, collect them, and train the label model with their program outputs. Additionally, we increase the number of sampled programs in each category from 4 to 9. We test this approach on the datasets where Alchemist gives comparable or lower performance than zero-shot prompting in our initial experiments in Table 1, namely the SMS, Yelp, and IMDb datasets.

Expected Results. By obtaining more diverse programs to use, Alchemist can capture a wider range of perspectives and labeling logic, potentially leading to more accurate pseudolabels.

	YouTube				SMS				Yelp				IMDb			
	Human crafted	Synthesized Programs			Human crafted	Synthesized Programs			Human crafted	Synthesized Programs			Human crafted	Synthesized Programs		
		GPT-3.5	GPT-4	Claude 3		GPT-3.5	GPT-4	Claude 3		GPT-3.5	GPT-4	Claude 3		GPT-3.5	GPT-4	Claude 3
Num. of Programs	10	10	10	10	73	10	10	10	8	10	10	10	5	10	10	10
Coverage	0.89	1.00	1.00	1.00	0.41	1.00	1.00	1.00	0.83	0.78	0.99	0.88	0.88	0.89	1.00	0.98
Performance	0.85	**0.89**	**0.89**	0.72	0.89	**0.90**	**0.93**	0.89	0.76	0.57	**0.82**	**0.83**	0.73	0.66	**0.75**	0.70

Table 4: Analysis showing that Alchemist can achieve comparable or better accuracy and higher coverage while using fewer programs to label the data.

Results. Fig. 5 visualizes the effect on the label model's performance when we increase the diversity in collected programs. It demonstrates a trend and indicates that involving a more diverse set of programs can help to mitigate the impact of individual strategy biases or limitations, leading to the production of better labels.

Overall, results in Sec. 4.3 and in Sec. 4.4 underscore that ***the use of supplementary information and involving diverse types of programs can help achieve better performance.***

4.5 Comparing to Human-crafted Programs

Setup. Lastly, we compare synthesized programs in Alchemist and manually crafted labeling functions in WRENCH [48], which is a widely-used benchmark for evaluating weak supervision methods. We focus on the datasets that overlap between Alchemist and WRENCH. For each dataset, we use pure prompts to query GPT-3.5, GPT-4, and Claude 3 for 10 programs. We then evaluate the performance of the distilled model for both methods. We also include the label model's coverage in our comparison. Higher coverage means that label model can produce more pseudolabels, yielding a larger size of training dataset to use.

Expected Results. We expect that synthesized programs may offer some advantages in terms of efficiency and effectiveness compared to human-designed ones.

Results. Table 4 presents their comparison. By leveraging the knowledge and capabilities of language models, we find that generated programs offer several advantages, including better coverage (i.e., the ability to label more data points) and comparable, or even better, performance. Generated programs can reduce the need for laborious engineering, which can be time-consuming and often requires a tedious design process to fine-tune labeling logic, such as thresholds and keyword usage. This design process may lead to many undiscovered rules, resulting in lower performance on coverage and potentially limiting the effectiveness of the labeling functions—unlike synthesized programs.

This is particularly evident in the SMS dataset, where WRENCH requires 73 manually crafted labeling functions to obtain high-quality labels, while Alchemist only needs 10 generated programs to obtain comparable performance and higher coverage. This significant reduction highlights the potential of Alchemist to ***assist humans in designing labeling functions and make it more accessible to users without extensive domain expertise.***

5 Conclusion

We propose an alternative approach to costly annotation procedures that require repeated API requests for labels. Our solution introduces a simple notion of prompting programs to serve as annotators. We developed an automated labeling system called Alchemist to embody this idea. Empirically, our results indicate that Alchemist demonstrates comparable or even superior performance compared to language model-based annotation, improving five out of eight datasets with an average enhancement of 12.9%. Notably, Alchemist reduces total costs by a factor of approximately 500. Furthermore, we showcase the system's extensibility to handle more complex modalities while enhancing the robustness of predicted labels. Finally, we confirm that incorporating relevant information can generate better programs, and increasing diversity leads to obtaining higher-quality labels.

References

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

[2] Shuohang Wang, Yang Liu, Yichong Xu, Chenguang Zhu, and Michael Zeng. Want to reduce labeling cost? gpt-3 can help. *arXiv preprint arXiv:2108.13487*, 2021.

[3] Jiacheng Ye, Jiahui Gao, Qintong Li, Hang Xu, Jiangtao Feng, Zhiyong Wu, Tao Yu, and Lingpeng Kong. Zerogen: Efficient zero-shot learning via dataset generation. *arXiv preprint arXiv:2202.07922*, 2022.

[4] Jiacheng Ye, Jiahui Gao, Zhiyong Wu, Jiangtao Feng, Tao Yu, and Lingpeng Kong. Progen: Progressive zero-shot dataset generation via in-context feedback. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 3671–3683, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics.

[5] Jiahui Gao, Renjie Pi, LIN Yong, Hang Xu, Jiacheng Ye, Zhiyong Wu, WEIZHONG ZHANG, Xiaodan Liang, Zhenguo Li, and Lingpeng Kong. Self-guided noise-free data generation for efficient zero-shot learning. In *International Conference on Learning Representations*, 2023.

[6] Xingwei He, Zhenghao Lin, Yeyun Gong, Alex Jin, Hang Zhang, Chen Lin, Jian Jiao, Siu Ming Yiu, Nan Duan, Weizhu Chen, et al. Annollm: Making large language models to be better crowdsourced annotators. *arXiv preprint arXiv:2303.16854*, 2023.

[7] Ruoyu Zhang, Yanzeng Li, Yongliang Ma, Ming Zhou, and Lei Zou. Llmaaa: Making large language models as active annotators. *arXiv preprint arXiv:2310.19596*, 2023.

[8] Yu Meng, Martin Michalski, Jiaxin Huang, Yu Zhang, Tarek Abdelzaher, and Jiawei Han. Tuning language models as training data generators for augmentation-enhanced few-shot learning. In *International Conference on Machine Learning*, 2023.

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

[10] Anthropic. Introducing the next generation of claude, Mar 4, 2024.

[11] Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alexander Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. *arXiv preprint arXiv:2305.02301*, 2023.

[12] Tetsuya Sasaki, Firoz Chowdhury, and Sunil Thite. Medical text dataset: Cancer doc classification, 2023.

[13] Shiyuan Huang, Siddarth Mamidanna, Shreedhar Jangam, Yilun Zhou, and Leilani H Gilpin. Can large language models explain themselves? a study of llm-generated self-explanations. *arXiv preprint arXiv:2310.11207*, 2023.

[14] Andreas Madsen, Sarath Chandar, and Siva Reddy. Can large language models explain themselves? *arXiv preprint arXiv:2401.07927*, 2024.

[15] Chirag Agarwal, Sree Harsha Tanneru, and Himabindu Lakkaraju. Faithfulness vs. plausibility: On the (un) reliability of explanations from large language models. *arXiv preprint arXiv:2402.04614*, 2024.

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

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

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

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

[20] Sweta Agrawal, Chunting Zhou, Mike Lewis, Luke Zettlemoyer, and Marjan Ghazvininejad. In-context examples selection for machine translation. *arXiv preprint arXiv:2212.02437*, 2022.

[21] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):1–35, 2023.

[22] Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. What makes good in-context examples for gpt-3? *arXiv preprint arXiv:2101.06804*, 2021.

[23] Stephanie Lin, Jacob Hilton, and Owain Evans. Teaching models to express their uncertainty in words. *arXiv preprint arXiv:2205.14334*, 2022.

[24] Ohad Rubin, Jonathan Herzig, and Jonathan Berant. Learning to retrieve prompts for in-context learning. *arXiv preprint arXiv:2112.08633*, 2021.

[25] Hyuhng Joon Kim, Hyunsoo Cho, Junyeob Kim, Taeuk Kim, Kang Min Yoo, and Sang-goo Lee. Self-generated in-context learning: Leveraging auto-regressive language models as a demonstration generator. *arXiv preprint arXiv:2206.08082*, 2022.

[26] Daniel Y. Fu, Mayee F. Chen, Frederic Sala, Sarah M. Hooper, Kayvon Fatahalian, and Christopher Ré. Fast and three-rious: Speeding up weak supervision with triplet methods. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

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

[28] Nilaksh Das, Sanya Chaba, Renzhi Wu, Sakshi Gandhi, Duen Horng Chau, and Xu Chu. Goggles: Automatic image labeling with affinity coding. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 1717–1732, 2020.

[29] Benedikt Boecking, Willie Neiswanger, Eric Xing, and Artur Dubrawski. Interactive weak supervision: Learning useful heuristics for data labeling. In *International Conference on Learning Representations*, 2021.

[30] Nicholas Roberts, Xintong Li, Tzu-Heng Huang, Dyah Adila, Spencer Schoenberg, Cheng-Yu Liu, Lauren Pick, Haotian Ma, Aws Albarghouthi, and Frederic Sala. AutoWS-bench-101: Benchmarking automated weak supervision with 100 labels. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022.

[31] Tzu-Heng Huang, Catherine Cao, Spencer Schoenberg, Harit Vishwakarma, Nicholas Roberts, and Frederic Sala. Scriptoriumws: A code generation assistant for weak supervision. *ICLR Deep Learning for Code Workshop*, 2023.

[32] Naiqing Guan, Kaiwen Chen, and Nick Koudas. Can large language models design accurate label functions?, 2023.

[33] 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. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

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

[35] Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, and Zhifang Sui. A survey on in-context learning. *arXiv preprint arXiv:2301.00234*, 2022.

[36] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020.

[37] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2023.

[38] Open AI. Hello gpt-4o, Mar 13, 2024.

[39] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021.

[40] Shiori Sagawa, Pang Wei Koh, Tatsunori B Hashimoto, and Percy Liang. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. *arXiv preprint arXiv:1911.08731*, 2019.

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

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

[43] Wendi Ren, Yinghao Li, Hanting Su, David Kartchner, Cassie Mitchell, and Chao Zhang. Denoising multi-source weak supervision for neural text classification. *arXiv preprint arXiv:2010.04582*, 2020.

[44] P. Malo, A. Sinha, P. Korhonen, J. Wallenius, and P. Takala. Good debt or bad debt: Detecting semantic orientations in economic texts. *Journal of the Association for Information Science and Technology*, 65, 2014.

[45] Abir ELTAIEF. french book reviews, 2023.

[46] Tim Schopf, Daniel Braun, and Florian Matthes. Evaluating unsupervised text classification: Zero-shot and similarity-based approaches. In *Proceedings of the 2022 6th International Conference on Natural Language Processing and Information Retrieval*, NLPIR '22, page 6–15, New York, NY, USA, 2023. Association for Computing Machinery.

[47] OpenAI. Openai pricing table.

[48] Jieyu Zhang, Yue Yu, Yinghao Li, Yujing Wang, Yaming Yang, Mao Yang, and Alexander Ratner. Wrench: A comprehensive benchmark for weak supervision. *arXiv preprint arXiv:2109.11377*, 2021.

[49] A. P. Dawid and A. M. Skene. Maximum likelihood estimation of observer error-rates using the em algorithm. *Applied Statistics*, 28(1):20–28, 1979.

The appendix is organized as follows. First, we provide details about datasets, training settings, and computation resources in Appendix A. Next, in Appendix B we list prompts that we use to query language models. Then, we present ablation studies in Appendix C using other models in weak supervision to work with Alchemist. Lastly, we discuss limitations and broader impacts of our work in Appendix D.

A Datasets and Implementation Details

Dataset	Task Type	Prediction Classes	# of Classes	# of Train
YouTube [41]	spam comment detection	{“spam”, “ham”}	2	1686
SMS [42]	spam text detection	{“spam”, “ham”}	2	4571
Yelp [43]	restaurant review sentiment classification	{“postive”, “negative”}	2	30400
IMDb [43]	movie review sentiment classification	{“postive”, “negative”}	2	20000
MedAbs [46]	medical abstract topic classification	{“neoplasms”, “digestive system diseases”, “nervous system diseases”, “cardiovascular diseases”, “general pathological conditions”}	5	10395
Cancer [12]	biomedical document topic classification	{“colon cancer”, “lung cancer”, “thyroid cancer”}	3	5450
Finance [44]	finance news sentiment classification	{“positive”, “neutral”, “negative”}	3	3488
French [45]	book review sentiment classification	{“positive”, “neutral”, “negative”}	3	6953
Waterbirds [40]	bird species classification	{“landbird”, “waterbird”}	2	5794

Table 5: Dataset Table.

We place more details about our datasets and experimental setups here. First, in Table 5 we show task type, prediction classes, and number of training data points in each dataset. MedAbs, Cancer, Finance, and French are considered to be more challenging settings, where these datasets typically need domain expertise to provide labels. Waterbirds is considered to test for a more complex modality.

We employ Snorkel as our label model to aggregate program outputs and report results in the main paper. We show more results using different choices of label model in Appendix C. All the distilled models use the MLP model that is trained with 2 hidden layers, each comprising 32 units, using ReLU activations between layers and no normalization. We run 5 times with different random seeds and report their average performance. We use a A6000 NVidia GPU to run all experiments.

B Used Prompts

Dataset	Zero-shot Prompting (Baseline)
YouTube	what is the category of this youtube comment: [text]
SMS	what is the category of this sms text: [text]
Yelp	what is the sentiment of this restaurant review: [text]
IMDb	what is the sentiment of this movie review: [text]
MedAbs	what is the topic of this abstract: [text]
Cancer	what is the topic of this document: [text]
Finance	what is the sentiment of this news: [text]
French	what is the sentiment of this book review: [text]

Table 6: Prompts for baseline approach are presented.

Dataset	Task Description (Alchemist)
YouTube	Write a bug-free and executable function in python to label comment on Youtube as spam or ham.
SMS	Write a bug-free and executable function in python to label SMS text as spam or ham.
Yelp	Write a bug-free and executable function in python to label the sentiment of restaurant review on Yelp as postive or negative.
IMDb	Write a bug-free and executable function in python to label the sentiment of movie review on IMDB as postive or negative
MedAbs	Write a bug-free and executable function in python to label the topic of medical abstract.
Cancer	Write a bug-free and executable function in python to label the topic of biomedical document.
Finance	Write a bug-free and executable function in python to label the sentiment of financial news as postive, neutral, or negative
French	Write a bug-free and executable function in python to label the sentiment of book review written in French as postive, neutral, or negative.

Table 7: Task descriptions in Alchemist’s prompt are presented.

Next, we present the prompts used to query language models in the baselines and Alchemist.

	Youtube		SMS		Yelp		IMDB	
	Est. Cost	Accuracy	Est. Cost	F1-score	Est. Cost	Accuracy	Est. Cost	Accuracy
Zero-shot Prompting	0.096	0.871	0.240	0.907	3.873	**0.845**	3.400	**0.737**
Weighted Majority Vote	0.004	**0.874**	0.004	0.886	0.005	0.705	0.004	0.520
Dawid-Skene	0.004	0.864	0.004	0.895	0.005	0.682	0.004	0.507
FlyingSquid	0.004	0.863	0.004	**0.915**	0.005	0.678	0.004	0.500
Snorkel	0.004	**0.891**	0.004	0.900	0.005	0.575	0.004	0.662
	MedAbs		**Cancer**		**Finance**		**French**	
	Est. Cost	Accuracy	Est. Cost	Accuracy	Est. Cost	Accuracy	Est. Cost	Accuracy
Zero-shot Prompting	1.944	0.311	15.925	0.716	0.201	0.641	0.641	0.611
Weighted Majority Vote	0.006	**0.354**	0.003	**0.968**	0.007	**0.650**	0.006	0.221
Dawid-Skene	0.006	0.262	0.003	**0.957**	0.007	**0.661**	0.006	0.221
FlyingSquid	0.006	**0.323**	0.003	**0.967**	0.007	**0.661**	0.006	**0.690**
Snorkel	0.006	**0.346**	0.003	**0.968**	0.007	**0.660**	0.006	**0.690**

Table 8: Testing performance of the distilled model is reported for each combination of label model and dataset.

First, we show the prompts used for the baseline approach of zero-shot prompting on text datasets in Table 6. In these prompts, the placeholder "[text]" is replaced with individual data points and sent via API calls to obtain labels for each data point.

Next, we present the prompts used in Alchemist in Table 7. The table displays the task description component of each prompt. These descriptions outline the objective of the generated program and are associated with the prediction classes. For the labeling instructions, we directly map the prediction classes to their corresponding class indices and query the language models to output the appropriate class index.

For the image task, we use the prompts ["an image of landbird", "an image of waterbird"] to perform zero-shot prompting using CLIP. In Alchemist, we first query high-level concepts and then combine them with computed scores to prompt LLMs to generate programs. The first step involves the following prompt: "What are the visual primitive concepts to classify "landbird" and "waterbird"? Please organize the primitive concepts by name and use comparisons for the classes. Parse the results into JSON format."

Once we have obtained a set of similarity scores, we use the following prompt: "I have measured similarity scores for the following descriptions as float numbers. If a score is close to 1, it is highly related to the description. If a score is close to 0, it is less related to the description. The descriptions are: ["A bird's foot type is toed, grasping"]; ["A bird's foot type is paddling, swimming"]. Generate a labeling function with input scores to classify landbirds and waterbirds. If it cannot be determined, the function should return -1, but use this cautiously." Descriptions will be replaced by different generated concepts.

C Ablation Studies

Alchemist is compatible with a variety of weak supervision aggregation approaches. We report additional results with different choices of label models. Besides Snorkel, we consider three more widely-used label models: Weighted Majority Vote, Dawid-Skene [49], and FlyingSquid (FS) [26]. We reuse our experimental setup from Sec. 4.1 and in Sec. 4.5 and present the performance of the distilled models in Table 8 and in Table 9, respectively.

In Table 8, we observe that the label accuracy is enhanced or achieves comparable performance with different label models, showcasing Alchemist's flexibility in working with various label models. In Table 9, we include compare them with human-crafted labeling functions developed in WRENCH [48]. Similarly, Alchemist obtains higher coverage and achieves comparable or even better label accuracy while reducing the need to craft a large number of programs manually.

		Number of Programs	Coverage	Weighted Majority Vote	Dawid-Skene	FlyingSquid	Snorkel
Youtube	Human-crafted	10	0.89	0.88	0.84	0.87	0.85
	GPT-3.5	10	1.00	0.87	0.86	0.86	**0.89**
	GPT-4	10	1.00	0.85	**0.88**	**0.87**	**0.89**
	Claude 3	10	1.00	0.77	0.71	0.73	0.72
SMS	Human-crafted	73	0.41	0.90	0.86	0.00	0.89
	GPT-3.5	10	1.00	0.89	0.90	0.90	0.90
	GPT-4	10	1.00	**0.91**	0.90	**0.92**	**0.93**
	Claude 3	10	1.00	**0.91**	**0.92**	**0.92**	0.89
Yelp	Human-crafted	8	0.83	0.75	0.83	0.77	0.76
	GPT-3.5	10	0.78	0.70	0.68	0.68	0.57
	GPT-4	10	0.99	0.73	**0.81**	0.72	0.82
	Claude 3	10	0.88	**0.77**	0.78	**0.81**	**0.83**
IMDb	Human-crafted	5	0.88	0.72	0.73	0.68	0.73
	GPT-3.5	10	0.89	0.52	0.51	0.50	0.66
	GPT-4	10	1.00	0.54	0.55	0.54	**0.75**
	Claude 3	10	0.98	0.59	0.64	0.60	0.70

Table 9: We offer a comparison between a wider range of label model options for synthesized programs and those designed by humans.

D Discussion

Limitations. There are two primary limitations in Alchemist. First, the performance of the datasets we test is still dependent on the capabilities of the language model. If the language model's ability to comprehend the given task and generate effective programs is subpar, the labeling performance may suffer. The second limitation arises when dealing with extremely complex tasks. As the complexity of the task increases, the generated code may become longer, more intricate, and harder to understand, posing challenges for developers who take time to validate correctness.

Broader Impacts. We do not see explicit negative impacts in Alchemist's annotation process. However, generated programs from language models may contain biased labeling logic, toxic content, or malicious functions. To mitigate this, auditing and guardrails may be necessary.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and precede the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes] , [No] , or [NA] .
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes] " is generally preferable to "[No] ", it is perfectly acceptable to answer "[No] " provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No] " or "[NA] " is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading “NeurIPS paper checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper’s contributions and scope?

   Answer: [Yes]

   Justification: Yes. Our claims are accurately reflect our contributions in data annotation and its scope.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

Justification: Yes. We discuss the limitations of the work in Appendix D.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification: This paper is an empirical work. It does not apply to this paper.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Yes. We provide code, data, and results in supplementary files. We report our training settings, and used prompts in the Appendix (See Appendix A and Appendix B).

   Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Yes. We attach our code and data in the submission file. We will release our implementation soon.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.

- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We discuss training and testing details in our experiment sections and in Appendix A.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [NA]

   Justification: Re-prompting labels for data points may create a huge expense.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We place the information about computer resources in the Appendix A.

   Guidelines:

   - The answer NA means that the paper does not include experiments.

- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: It follows NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss them and place in Appendix D.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: This doesn't apply to this paper.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Yes. We properly credited data, paper, and ideas that we used in this paper.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We document well about the asset.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: This doesn't apply to this work.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: This paper doesn't involve crowdsourcing and research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.