Language Models in the Loop: Incorporating Prompting into Weak Supervision

RYAN SMITH*, Snorkel AI, USA

JASON A. FRIES*, Stanford University and Snorkel AI, USA

BRADEN HANCOCK, Snorkel AI, USA

STEPHEN H. BACH, Brown University and Snorkel AI, USA

We propose a new strategy for applying large pre-trained language models to novel tasks when labeled training data is limited. Rather than apply the model in a typical zero-shot or few-shot fashion, we treat the model as the basis for labeling functions in a weak supervision framework. To create a classifier, we first prompt the model to answer multiple distinct queries about an example and define how the possible responses should be mapped to votes for labels and abstentions. We then denoise these noisy label sources using the Snorkel system and train an end classifier with the resulting training data. Our experimental evaluation shows that prompting large language models within a weak supervision framework can provide significant gains in accuracy. On the WRENCH weak supervision benchmark, this approach can significantly improve over zero-shot performance, an average 19.5% reduction in errors. We also find that this approach produces classifiers with comparable or superior accuracy to those trained from hand-engineered rules.

CCS Concepts: • **Computing methodologies** → **Machine learning approaches**; **Natural language processing**.

Additional Key Words and Phrases: weak supervision, zero-shot learning

1 INTRODUCTION

Large pre-trained language models [10, 17, 23, 39, 40] have shown remarkable zero-shot and few-shot performance on a range of natural language tasks. By prompting them to answer queries, users can tap vast knowledge acquired through large-scale self-supervised pre-training. Prompting [31] refers to the emerging practice of conditioning a language model on an input representing a query and interpreting the output as a solution to the task. For example, in a web spam classification task, we could give the prompt "The following comment is spam. Yes or No? Subscribe to my channel! example.com/12345" and compute whether the continuation "Yes" or "No" is more probable to make a prediction. Remarkably, large pre-trained models can generalize in non-trivial ways to unseen tasks [10, 36, 46, 57]. Beyond being useful for solving tasks directly, pre-trained language models are instances of foundation models [7], large pre-trained models that can be used as the foundation for new models that are better suited to specialized tasks, either because they are more accurate, less computationally expensive, or both. Building on top of foundation models is an
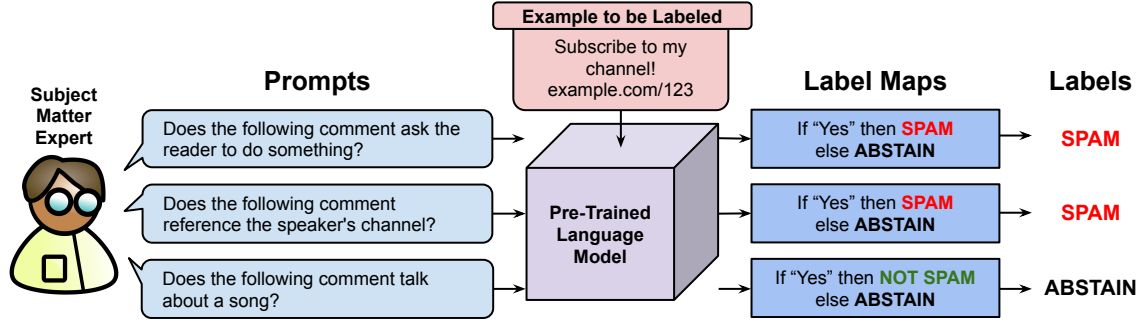

Fig. 1. An overview of how a subject matter expert (SME) can use prompting to create weak supervision sources. The SME expresses tests for signifiers of the class of interest as natural language prompts. The prompts are combined with unlabeled examples and given to a pre-trained language model. The model's responses are mapped to votes on the true label for the example.

important challenge for data science, as data scientists often need to create predictive models, particularly from limited labeled training data. In this work, we investigate how to direct the knowledge contained in pre-trained language models toward the creation of labeled training data for models that generalize beyond the performance of the source language model.

Limited labeled training data is a major bottleneck in many areas of supervised machine learning. In recent years, the area of *programmatic weak supervision* [64] has emerged to address this bottleneck. There are a range of techniques, but generally they use multiple noisy heuristic labelers called *labeling functions*, such as hand-written code and other models, to create training data for new tasks. These labelers are applied to abundant unlabeled data, and they either vote on the correct label or abstain. Then, a label modeling stage attempts to resolve the conflicts among the labelers without access to much or any ground truth labels. The resulting labels are finally used to train an end model that generalizes beyond the labelers. This approach has seen many practical successes in areas such as information extraction [12, 21, 41] and medical imaging [18, 20]. Programmatic weak supervision has also been deployed at major technology companies [5, 9, 28, 50]. Large pre-trained language models are an untapped resource as a potentially complementary source of heuristic labels. In addition to the ease of specifying heuristics with natural language, we show that they can effectively capture a wide range of fuzzy concepts that can be hard to express as traditional labeling functions written in code.

Despite this potential, naively prompting pre-trained models to label training data has several potential pitfalls. First, language models are sensitive to the wording of prompts [26, 49]. Even models that have been fine-tuned on a variety of prompt wordings can still be sensitive to phrasing [46, 56, 57]. Second, prompted language models are limited in the complexity of the instructions they can follow [36, 56]. Tasks can have nuanced decision boundaries based on context. For example, a link to a music video might be more likely to be spam on a news website but not spam on a video site. A single prompt, even paraphrased into multiple variants to address model sensitivity, is often insufficient to capture the full specification of a task. For these reasons, a framework for incorporating pre-trained language models into weak supervision is needed that can incorporate significant amounts of subject matter expertise in a manner efficient for users.

Prompting is an emerging area in natural language processing, and recent related works have explored using prompted models as sources of supervision. Several works use pre-trained models to generate or modify text examples conditioned on a desired label that can be used for training [8, 47, 59, 60]. Other recent works use pre-trained models

to aid in labeling unlabeled examples. Concurrently, Lang et al. [30] use co-training to iteratively generate training data for variations of the same prompt. Also concurrently, Zhang et al. [67] use prompting and labeled training data to suggest new labeling functions. Also concurrently, Chen et al. [13] propose using embeddings from foundation models to capture which examples are best labeled by which labeling functions. Across these methods, there remains a need for a framework that allows users to refine the contours of a decision boundary with multiple prompts, particularly when labeled data is scare.

In this work, we propose a framework for incorporating prompting into programmatic weak supervision, in order to address the above challenges and realize potential benefits from pre-trained language models (Figure 1). We model prompts as labeling functions by adding additional metadata that maps possible completions to target labels or abstentions. For example, if a task is to classify spam comments, a prompt could be "Does the following comment ask the user to click a link?" If the language model responds positively, then this is an indication that the comment is spam. On the other hand, if the model responds negatively then that might be mapped to an abstention because both spam and non-spam comments can lack that property. We then model the outputs of the these labeling functions as usual: using a label model to reason about the accuracies of the different prompts and create training data for an end model. This approach is novel because it exploits pre-trained language models not just as zero- or few-shot learners, but as rich sources of knowledge that can be queried in many complementary ways to create training data.

We conduct an extensive experimental study of this approach. Using the WRENCH [66] benchmark as a starting point, we first demonstrate that many existing types of labeling functions expressed as code can be effectively translated into natural language prompts. We show on a range of GPT-3 [10] and T0 [46] models that using these prompts for zero-shot querying and using the resulting prompted predictions as labeling functions leads to end models that are more accurate than those trained on the original labeling functions. Surprisingly, we find that using these translated labeling functions works better in many cases than simply prompting the model to solve the task of interest. This result suggests that pre-trained models contain more useful information than can be easily accessed by a single zero-shot prompt. The additional domain knowledge provided by expressing complementary heuristics as prompts and describing how they relate to the task of interest is a key ingredient for improved accuracy. We show empirically that these prompt-based labeling functions usually make complementary, i.e. only weakly correlated mistakes, suggesting that the pre-trained language is actually applying different heuristics based on different prompts.

In summary, our main contributions are:

- We propose expressing wide ranges of data-labeling heuristics as zero-shot prompts for pre-trained language models, and using a label model to resolve their conflicts.
- We demonstrate the effectiveness of this new approach as a zero-shot learning approach, showing that prompting pre-trained models with multiple heuristic tasks can significantly outperform directly prompting the model to solve the task of interest, with an average improvement of 20.2 percentage points.
- We also show that translating labeling functions expressed as code into prompts can lead to significantly improved weakly supervised models, with an average improvement of 7.1 percentage points, when using our best language model, T0++ [46]

2 RELATED WORK

This work builds on both weakly supervised machine learning and prompting with large pre-trained language models. In this section, we overview the most closely related work.

2.1 Weakly Supervised Machine Learning

The difficulty of obtaining large amounts of labeled training data has long motivated alternatives to traditional supervised machine learning. *Weak supervision* refers to a broad family of techniques that attempts to learn from data that is noisily or less precisely labeled than usual. Our focus is on *programmatic weak supervision*, in which the sources of supervision are heuristic labelers, often called *labeling functions* that vote on the true labels of unlabeled examples [64]. Labeling functions can be hand-written programs, models trained for related tasks, or even human annotators if available. Labeling functions have their roots in work on *distant supervision* [15, 35], in which a single heuristic is used to label data and the resulting labels are assumed to be noise-free. Ratner et al. [42] proposed the *data programming* paradigm for weak supervision, in which multiple labeling functions that can disagree or abstain are available.

Using multiple labeling functions gives rise to the key technical challenge in programmatic weak supervision: resolving their disagreements without access to ground truth, in order to create training data. The original formulation of data programming uses a probabilistic generative model that assumes the ground truth label for each example is a latent random variable that generates the outputs of the labeling functions. The parameters of the model are learned by maximizing the likelihood of the observed outputs of the labeling functions. This model generalizes the classic Dawid-Skene model [16] for *crowdsourcing*, i.e., learning from multiple human annotators. In the simplest case, the label sources can be assumed to be conditionally independent given the true label. In practice, this approach often works well. However, since programmatic heuristics might exhibit biases and correlations in more systematic ways than human annotators, it is often advantageous to model more complex dependencies among the labeling functions. Multiple methods for learning such dependencies from the labeling function outputs have been proposed [4, 43, 53]. Many of these techniques for data programming are integrated in the Snorkel system [41].

Programmatic weak supervision has been extended in many directions. Using adversarial learning instead of maximum likelihood estimation can provide strong theoretical guarantees without assumptions on the distribution of labels and labeling function outputs, but requires either a small amount of labeled data or other assumptions to constrain the accuracy of the labeling functions [2, 33, 34]. Weak supervision can be applied to other settings like structured prediction [44, 45, 48]. Labeling functions can incorporate additional forms of supervision beyond individual labels, such as hierarchical multi-task supervision [43], partial labels [61], labels from misaligned spaces [65], or constraints [1]. Labeling functions can also be automatically constructed using a small amount of labeled data [52]. Another line of work has extended the label modeling stage to incorporate features of the underlying data, in order to model which types of examples each labeler is best at labeling [51]. Concurrent with our work, Chen et al. [13] proposed using large pre-trained models to create representations for the label model. Our work differs in that we use large pre-trained models to directly implement labeling functions as zero-shot predictors.

Finally, programmatic weak supervision is complementary to many other techniques for learning with limited labeled data. It can be combined with semi-supervised learning [27], self-supervised learning [62], and active learning [6, 11]. Since our work creates labeling functions that can be modeled in the same way as traditional ones, they can also be incorporated into all of these related frameworks.

2.2 Language Models and Prompting

Language models are trained to predict the next or missing words conditioned on a partial sequence of natural language text. Neural-network-based language models have become ubiquitous in recent work in natural language processing because they learn useful vector representations of text that can be incorporated into models for other tasks. Most

recently developed language models are based on transformer architectures [54]. Recently, there has been increasing interest in *prompting*, an alternative way of exploiting language models [31]. Instead of using language models only as feature encoders, prompting uses a language model's ability to predict words to directly solve tasks. Tasks are posed as natural language text called prompts, and the language model's predictions for missing or subsequent words are interpreted as task solutions. The language model can either be fine-tuned on specific prompts using labeled examples, or it can be queried in a zero-shot fashion, i.e., prompted to solve tasks it has never been explicitly trained to solve.

Brown et al. [10] demonstrated that large pre-trained language models can solve zero-shot tasks. Other works showed that the zero-shot abilities of large language models can be improved by further fine-tuning the language model on a large mix of prompted tasks [36, 46, 57]. Despite these successes, there are still many challenges when using prompting for zero-shot or few-shot learning. Models can be sensitive to the wording of the prompt [26, 46, 49, 56, 57], and many works have tried to reduce this sensitivity and boost accuracy [36, 46, 57].

Several recent works have investigated other ways of creating or augmenting supervision using pre-trained language models. Schick and Schütze [47] prompt language models to generate examples of a certain label, e.g., generating documents with a specific topic. Ye et al. [60] generate data in an unsupervised way and then label them for training using a simple classification rule. Chia et al. [14] generate examples expressing relations among entities to create training data for relation extraction. Wu et al. [59] fine-tune language models to modify datasets so that they exhibit fewer biases, and Bonifacio et al. [8] fine-tune them to modify datasets for different information retrieval tasks. Several works use language models to generate "chains of thought" that can improve reasoning and be used for self-training [55, 58, 63]. In concurrent work, Lang et al. [30] use co-training to fine-tune language models, where the different views of the data come via different prompts. Like other work on enforcing consistency among prompted outputs [3, 19], they consider alternative wordings of the same task, whereas we focus on prompting multiple tasks to create supervision. Also in concurrent work, PRBoost [67] uses labeled data and labeling function templates to prompt language models to suggest additional labeling functions to human annotators. In contrast, we show that no modification of existing weak supervision pipelines are needed to achieve good performance, and that sufficiently large pre-trained language models are powerful sources of weak supervision.

3 WEAK SUPERVISION VIA PROMPTING

In this section we describe our proposed approach to incorporating large pre-trained language models into weakly supervised machine learning. The goal is to enable data scientists and other subject matter experts to leverage these resources more effectively. We focus on scenarios where users are not necessarily machine learning experts, meaning that fine-tuning large models with gradient updates is either infeasible because of the size of the model or impossible because they do not have access to the underlying model. Instead, they might only have API access and want to exploit the large pre-trained model to create a new one that is higher quality and servable in production (i.e., not prohibitively large to work with). Our presentation and experiments in Section 4 focus on the case where all supervision comes via a language model, but this approach also naturally integrates with other forms of weak supervision, such as hand-engineered programs.

3.1 Workflow

We first describe the workflow in our approach (Figure 2). In the the scenarios we consider, the user is a subject matter expert (SME) who wants to create a classifier for unlabeled data. Continuing our running example, this could be a classifier for detecting spam comments on a video website. They have access to a large amount of unlabeled data that
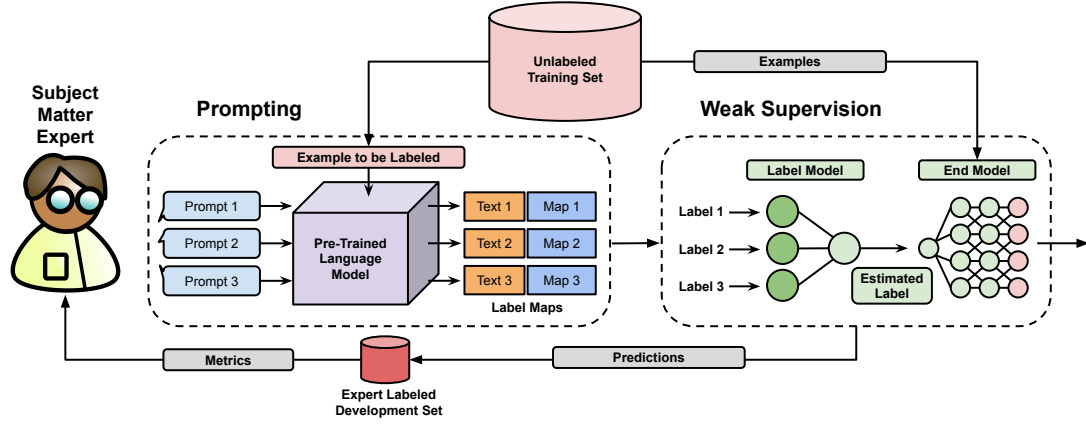
Fig. 2. Language models in the loop: the overall framework for developing and applying prompted labeling functions. The subject matter expert (SME) expresses their domain knowledge via prompts that are combined with unlabeled examples and given to a pre-trained language model. The model's responses are interpreted with label maps to produce votes on the true label. These votes are denoised with a label model, and the resulting estimated labels are used to train an end model. Throughout the process, the SME can refine their prompts by inspecting unlabeled examples and evaluating with a small labeled development set.

can be used for training. They also have access to a small (dozens up to hundreds of examples) development data set that has been manually labeled. That development set will be used to evaluate modeling decisions like the choice of prompts and tuning hyperparameters.

The SME then develops heuristics for labeling examples by inspecting unlabeled examples. These heuristics are expressed as natural language prompts that capture some aspect or feature of the data that is likely to indicate the true label. For example, in the case of labeling spam comments, the SME might notice by browsing comments that many spam examples contain some call to action, such as asking the reader to click or visit a link. Enumerating all the ways that a call to action could be expressed in natural language is challenging to do accurately, requiring the SME to curate many keywords and regular expressions that are sufficiently precise. Alternatively, a simple prompt like "Does the following comment ask the reader to do something?" has the potential to better capture this heuristic while requiring less effort from the SME.

The SME's heuristic prompts are encapsulated as *prompted labeling functions*. Prompted labeling functions consist of a prompt template and a label map. The prompt template defines how the SME's prompt is applied to unlabeled examples. Unlabeled examples consist of one or more fields of text. In this work, we focus on Yes/No question answering-style prompt templates. However our method generalizes to many prompt template and label map formats. In the case of website comments, the text could be represented as a single field `[TEXT]` and the entire prompt template for a labeling function could be

```
Does the following comment ask the reader to do something? [TEXT]
```

The label map then defines how responses by the pre-trained language model are mapped to votes on the true label for the example. Our framework focuses on generative language models like T0 [46] and GPT-3 [10], so the responses can be arbitrary text strings. The label map $M : \mathcal{S} \rightarrow \mathcal{Y} \cup \{\emptyset\}$ is a function from the set $\mathcal{S}$ of strings composed from the

pre-trained language model's vocabulary to the set of labels $\mathcal{Y}$ and a special symbol $\emptyset$, which indicates that the labeling function abstains, i.e., has no vote on that example. In the case of the above example prompt, a corresponding label map would map positive responses like "Yes" and "True" to the spam label, and all other responses to abstentions. SMEs can also refine their prompts by evaluating their labeling functions on the unlabeled data and the small labeled development data set. In this way, the SME enters a feedback loop, in which they can reword prompts and construct additional ones to add complementary information. We discuss the development of prompted labeling functions further in Section 3.2.

After the SME has developed their prompted labeling functions, they can be plugged into many standard weak supervision frameworks, such as Snorkel [41]. In such frameworks, the labeling functions are executed on all the available unlabeled data to produce votes on what the correct label is. These votes are aggregated in the *label model* that produces probabilistic estimates of the correct label. Finally, an appropriate end model, such as a deep neural network, is trained for the classification task of interest by minimizing the expected empirical risk with respect to the probabilistic estimates of the true labels. The resulting classifier can be used outside of this weak supervision framework and independently from the underlying pre-trained language model. In this way, language models in the loop enable SMEs to distill information locked away in large foundation models into smaller, more servable models. As we show in Section 4, these resulting models can also often significantly improve over the accuracy obtained by using the pre-trained model alone.

3.2 Developing Prompted Labeling Functions

We now discuss the advantages of writing prompted labeling functions, and how it differs from writing labeling functions in code. Prompted labeling functions are a mechanism by which a large pre-trained model can be adapted with limited labeled training data to new tasks. We find that large pre-trained models such as T0++ and GPT-3 exhibit a phenomenon wherein they "know more than they realize," in the sense that they can solve many other tasks that provide useful signals about the task of interest, even if they do not know how to integrate those signals.

Weakly supervised machine learning is a natural paradigm for integrating these signals effectively. For example, in the spam comment task, the zero-shot approach is to prompt the pre-trained language model with a prompt like "Is the following comment spam?" In contrast, we propose using prompting to collect multiple signals related to the task of interest. Examples from our experimental study (Section 4) are

(1) "Does the following comment ask the reader to do something?"
(2) "Does the following comment reference the speaker's channel?"
(3) "Does the following comment contain the words 'check out'?"

Each of these prompts, along with the associated label map, provides additional domain knowledge about the definition of spam in this particular application. Task supervision is often multifaceted and difficult to summarize in a single prompt. Pre-trained language models can have difficulty with long, nuanced instructions [36, 56]. Our approach breaks down task supervision into salient components, expressed as multiple prompts capturing different aspects of labeling.

The above example prompts also illustrate the advantages that pre-trained language models can offer weakly supervised machine learning. Standard rule-based labeling function expressed in code or via resources like term dictionaries are brittle. In contrast, prompts can handle significant amounts of ambiguity. The three example prompts above are arranged in order of decreasing ambiguity. Prompt (1) covers a wide range of scenarios that would be difficult to enumerate with rules. Answering the prompt accurately likely requires an understanding of intent. Prompt (2) is in the middle, in that it asks for references to a specific entity (the speaker's channel), but that entity can be referred to in

Examples of PLM True Positives:

"Rap from Belarus, check my channel:)"

"Hey everyone. Watch this trailer!!!!!!!!"

"Please look at my channel"

Examples of PLM False Positives:

"Came here to check the views, goodbye."

"Love this song makes me wanna dance!"

"This Song is AWESOME!!!!"

PLM and RegEx both flag 40% of spam

PLM also flags another 18%

Examples of PLM False Negatives:

"Check me out I'm all about gaming "

"Check out Melbourne shuffle, everybody!"

Flagged by RegEx (n=371)

Flagged by PLM (n=645)

Actual Spam (n=831)
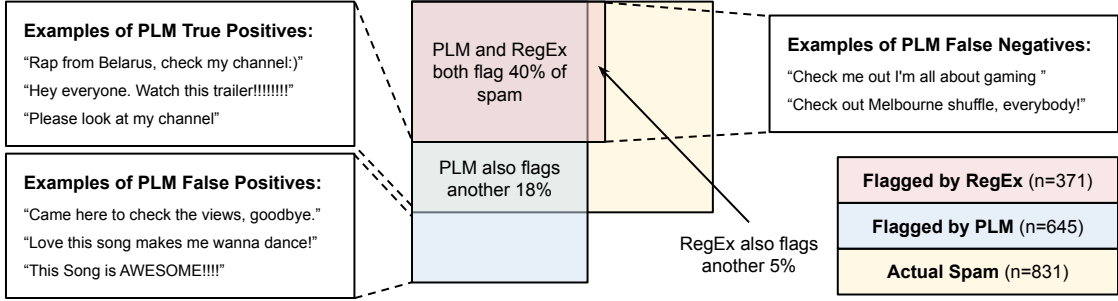
RegEx also flags another 5%

Fig. 3. A comparison of a regular expression (RegEx) labeling function from the WRENCH benchmark [66] with the corresponding prompted labeling function using the T0++ [46] pre-trained language model (PLM). The regular expression looks for variations of the phrase "check out" and the prompted labeling function uses the prompt "Does the following comment contain the words 'check out'?" RegEx has 100% precision and 45% recall, while PLM has 76% precision and 58% recall. This comparison shows that even simple labeling functions can be made more general while maintaining acceptable precision by using prompting.

many ways, including indirectly, e.g., a comment like "Like and subscribe!" Prompt (3) is the most specific, asking if the comment contains a specific phrase.

Surprisingly, even prompted labeling functions asking for a specific phrase have interesting, useful properties that differ from traditional labeling functions. Figure 3 compares a prompted labeling function using prompt (3) with the corresponding, traditional labeling function from the WRENCH benchmark for weak supervision [66] on the Youtube comment spam dataset. The traditional labeling function is a regular expression that also checks for the phrase "check out." It is very precise, with 100% precision and 45% recall. The prompted labeling function has 76% precision and 58% recall. The tradeoff is that the prompted labeling function finds many true positives that say something with a meaning similar to "check out," but also misfires on some false positives. This example illustrates that even with seemingly straightforward heuristics like a simple regular expression, pre-trained language models can provide useful additional flexibility. Our experiments in Section 4 show that this can be a favorable tradeoff for developers.

3.3 Calibration

We find that it is useful to improve the calibration of prompted labeling functions. Calibration is a measurement of how strongly a model's predicted probabilities correlate with observed accuracy, i.e., a predicted probability of $\hat{p}$ should be correct $\hat{p} \cdot 100\%$ of the time. Current language models are not well-calibrated, with predicted probabilities subject to several forms of biasing, e.g., favoring tokens observed more during pretraining or tokens that appear near the end of a prompt [26, 68]. Miscalibration creates challenges in prompting, which requires choosing the most likely answer from a set of candidate text completions. When using prompts as labelers, we may also want to threshold predictions to select only the most confident answers. Popular recalibration methods such as Platt and vector scaling [25, 38] require labeled data to learn a transformation of the model's predicted probabilities, creating challenges to directly applying these methods in zero-shot settings. Instead, we use *contextual calibration* [68], where scaling weights are estimated from the predicted token probabilities of a prompt queried using "content-free" or null input instances. Contextual calibration has demonstrated empirical performance gains when used in prompt-based, few-shot classification. We use the tokens $\{$ N/A, ϵ, [MASK], NULL, <|endoftext|> $\}$ as our null inputs, using the average predicted probabilities per

token to estimate our scaling weights for each prompt. The resulting transformation is then applied to each prompted labeling function's predictions.

4 EXPERIMENTAL STUDY

We conduct an experimental study to evaluate how incorporating prompted labeling functions compare with two alternatives: (1) distilling pre-trained language models in a zero-shot fashion, and (2) hand-written labeling functions. We use the WRENCH benchmark [66] for weak supervision in order to control the choice of labeling functions. WRENCH provides traditional labeling functions that we translate into corresponding prompted labeling functions for comparison. We find that

(1) Creating models via prompted labeling functions can significantly outperform directly prompting the model to solve the task of interest, with an average improvement of 20.2 percentage points, and

(2) Translating labeling functions expressed as code into prompts can lead to significantly improved weakly supervised models, with an average improvement of 7.1 percentage points, when using our best language model, T0++ [46].

4.1 Datasets

The WRENCH benchmark includes 22 diverse datasets for evaluating weakly supervised learning [66]. Datasets include labeling function sets for programmatically creating labeled training data and corresponding manually curated gold labels for evaluation. We focus on a subset of text classification tasks: YouTube, SMS, and Spouse. Note that 4 WRENCH datasets (IMDB, Yelp, AG News, TREC) were used as part of T0++ training, thus we exclude them from our analysis. Dataset summary statistics are outlined in Table 1.

Name	Task	#Labels	Class Labels	P(positive)	#LFs	Train	Valid.	Test
YouTube	Spam Detection	2	HAM,SPAM	0.488 (0.02)	10	1,586	120	250
SMS	Spam Detection	2	HAM,SPAM	0.132 (<0.01)	73	4,571	500	500
Spouse	Relation Extraction	2	NOT_SPOUSE,SPOUSE	0.074 (<0.01)	9	22,254	2,801	2,701

Table 1. Summary statistics for our WRENCH text classification datasets. P(positive) is the class balance of the positive label (SPAM or SPOUSE depending on the task) calculated as the mean and standard error of relative frequency for all gold labeled splits.

4.2 Translating WRENCH Labeling Functions into Prompts

Labeling functions are developed by SMEs via data exploration, which entails iteratively designing labeling rules by inspecting unlabeled examples and a small, hand-labeled development set. For WRENCH datasets, this process has already occurred, so our experiments focus on translating existing labeling rules into prompted form. We note this is a more restricted setting than if SMEs developed prompts initially, as WRENCH labeling functions are biased towards rules that are easy to express in code while prompts have more flexibility. All labeling function prompts are formulated as Yes/No questions and a label map that transforms text completions into class labels or abstains (i.e., not emitting a label).

For example, consider a WRENCH labeling function written in Python for the Spouse task, which uses keywords occurring between person mentions to label negative training examples by identifying likely family members.

```
def lf_familial_relationship(x):
    family = {"father", "mother", "sister", "brother", "son", "daughter", "uncle", "aunt"}
```

```
    return NOT_SPOUSE if len(family.intersection(set(x.between_tokens))) > 0 else ABSTAIN}
```

Instead of enumerating an incomplete list of keywords describing family relationships, our prompt focuses on the general insight conveyed by the labeling function.

```
 Context: [TEXT]\n\nAre [PERSON1] and [PERSON2] family members? ↦ {yes:NOT_SPOUSE, no:ABSTAIN}
```

Prompts were developed for GPT-3 and T0++ separately by iteratively querying each language model with unlabeled training instances, performing an ad hoc performance assessment, and then selecting a single prompt to use per labeling function. This mirrors the process by which a SME might query a language model to guide prompt development. The complete list of WRENCH prompts used in this work are found in Appendix §6.4.

4.3 Comparing Programmatic Labelers

Dataset	Model	Prompt
YouTube	T0++	`Is the following comment spam?\n\n"[TEXT]"`
SMS	T0++	`Is the following text message spam?\n\n"[TEXT]"`
Spouse	T0++	`Context: "[TEXT]"\n\nAre [PERSON2] and [PERSON1] married?`
YouTube	GPT-3	`Q: Is the following comment "[TEXT]" spam? \nA:`
SMS	GPT-3	`Q: Is the following text message "[TEXT]" spam? \nA:`
Spouse	GPT-3	`Context: "[TEXT]"\nQ: Are [PERSON1] and [PERSON2] married? \nA:`

Table 2. Zero-shot prompts for all datasets and language model families. [TEXT], [PERSON1], [PERSON2] are populated with text from the target example. Label maps are {no:HAM, yes:SPAM} for YouTube/SMS and {no:NOT_SPOUSE, yes:SPOUSE} for Spouse.

We compare three approaches for programmatically generating training labels, following the typical workflow used for weakly supervised learning. For each dataset in our analysis, we assume the original training split is unlabeled. All *labelers*, here prompted labeling functions and code-based labeling functions, are applied to the unlabeled training split to generate votes for the true label of each example. All prompts are calibrated using contextual calibration. All labeler votes, unless otherwise noted, are combined and denoised using the FlyingSquid [22] label model to estimate a single, probabilistic consensus label per example. The resulting labels are used to train a RoBERTa [32] end model, which provides a smaller, more servable classification model tailored to our task of interest. All model performance measures are then evaluated using gold labeled test splits. The three approaches we compare are:

(1) *WRENCH Benchmark*: The original WRENCH labeling functions released as part of the benchmark. Here *majority vote* (i.e., the mode of all labeling function outputs per example) is used as the label model since it performed the best when used with RoBERTa for all three of our tasks.

(2) *Zero Shot*: A zero-shot baseline where training data is labeled by one prompt that queries a language model for an example's class label. Prompts are outlined in Table 2 and were designed to align with prompts commonly used in zero shot learning by providing a simple, but often underconstrained, task description.

(3) *Prompted Weak Supervision*: The prompted versions of the WRENCH labeling functions. These labelers reflect the prototypical weakly supervised workflow, except we have replaced manually coded labeling functions with prompted versions.

4.4 Large Language Models

All prompts are evaluated using two different language model families: GPT-3 and T0++. We use the InstructGPT [37] family of GPT-3 engines, evaluating Ada, Babbage, and Curie since different engines are claimed to be better suited to specific tasks.[1] DaVinci was not used due to cost constraints (see complete pricing for all GPT-3 queries in Appendix §6.1). All queries were submitted via the OpenAI API between 01/24/2022–03/01/2022. Queries were restricted by the API to include only the top 100 most likely text completions.

T0++ [46] is an open, publicly available 11B parameter model based on the T5 architecture [40]. T0++ is trained using a large dataset of supervised tasks transformed into prompted training data. This explicit, multitask formulation of prompted training data results in better zero-shot classification performance that often matches or exceeds the much larger GPT-3. The model requires 42 GB of GPU memory to efficiently run locally without parameter offloading. We used a p3.8xlarge AWS EC2 instance with 4 Tesla V100 GPUs for inference.

4.5 Evaluation Metrics

We evaluate all models using precision, recall, F1, and accuracy. Performance metrics are reported as the mean and standard error of six training runs using different random seeds. Standard error is calculated using the sample standard deviation. For direct comparisons with WRENCH, we report accuracy or F1 based on the default metric reported in WRENCH benchmarks.

4.6 Results

	Youtube (Accuracy)		SMS (F1)		Spouse (F1)	
	Zero Shot	Prompted WS	Zero Shot	Prompted WS	Zero Shot	Prompted WS
WRENCH Benchmark	-	94.9 (0.5)	-	92.4 (0.5)	-	37.9 (2.8)
T0++	58.7 (2.4)	**92.0 (0.5)**	83.2 (2.4)	**91.8 (1.6)**	41.5 (13.1)	**62.9 (0.8)**
InstructGPT Curie	52.8 (0.0)	77.7 (1.9)	0.0 (0.0)	65.7 (5.8)	49.6 (1.0)	41.0 (0.9)
InstructGPT Babbage	78.5 (3.0)	85.1 (1.3)	32.2 (3.0)	23.6 (0.0)	40.9 (0.9)	34.9 (1.7)
InstructGPT Ada	51.7 (2.4)	52.9 (0.1)	26.3 (2.6)	28.3 (1.8)	19.1 (0.8)	17.7 (6.2)

Table 3. Performance metrics for Zero Shot and Prompted Weak Supervision (Prompted WS) using four large language models and calibrated prompts. Scores are the mean/standard error of 6 training replicates with the best prompted model performance in bold.

4.6.1 Prompted Weak Supervision. Table 3 outlines the performance of Zero Shot and Prompted Weak Supervision using four language models (T0++, InstructGPT family) compared against the WRENCH benchmark. Prompted weak supervision outperforms the zero-shot baseline by an average of 18.2% (-26.7 to 100%) across all language models and datasets. T0++ consistently demonstrated strong performance, outperforming InstructGPT in all datasets when using Prompted Weak Supervision. Considering only T0++ performance, Prompted Weak Supervision outperforms Zero Shot by an average of 39.5% (10.3 to 56.7%). In the InstructGPT models, Prompted Weak Supervision largely negatively impacted performance, with performance gains consistently observed only in the YouTube dataset. Overall, the InstructGPT family performed substantially worse than T0++, which outperformed InstructGPT Curie by an average of 37.2% (18.4 to 53.4%).

---

[1]See https://beta.openai.com/docs/engines/gpt-3

Using the T0++ model, prompted performance approaches or exceeds models trained using the WRENCH Benchmark labeling functions. In the case of Spouse, T0++ significantly outperformed WRENCH labeling functions, improving performance by 25 F1-score points when using Prompted Weak Supervision.

Dataset	Language Model	CC	Precision	Recall	F1	±F1	Acc.	±Acc.
YouTube	T0++	✓	92.6 (0.5)	91.7 (0.5)	91.9 (0.5)	-3.5 (0.6)	92.0 (0.5)	-3.4 (0.6)
			95.7 (0.4)	95.2 (0.5)	95.4 (0.4)	–	95.4 (0.4)	–
SMS	T0++	✓	95.9 (2.5)	88.1 (1.1)	91.8 (1.6)	**+0.3 (2.5)**	97.9 (0.4)	+0.2 (0.7)
			91.6 (3.2)	91.5 (0.8)	91.4 (1.6)	–	97.7 (0.5)	–
Spouse	T0++	✓	54.2 (1.8)	75.4 (1.2)	62.9 (0.8)	**+18.0 (1.7)**	92.8 (0.3)	**+10.0 (1.4)**
			30.7 (1.6)	86.0 (2.8)	44.9 (1.3)	–	82.8 (1.2)	–
YouTube	InstructGPT Curie	✓	80.1 (1.0)	77.1 (2.1)	76.7 (2.3)	**+0.8 (2.0)**	77.7 (1.9)	-0.1 (1.6)
			84.8 (0.6)	76.4 (1.2)	75.9 (1.3)	–	77.7 (1.1)	–
SMS	InstructGPT Curie	✓	60.6 (11.5)	83.8 (4.4)	65.7 (5.8)	**+65.7 (5.8)**	86.2 (3.9)	-0.4 (3.9)
			0.0 (0.0)	0.0 (0.0)	0.0 (0.0)	–	86.6 (0.0)	–
Spouse	InstructGPT Curie	✓	29.5 (0.9)	67.5 (2.0)	41.0 (0.9)	**+41.0 (0.9)**	84.3 (0.7)	-7.7 (0.7)
			0.0 (0.0)	0.0 (0.0)	0.0 (0.0)	–	91.9 (0.0)	–

Table 4. The impact of contextual calibration (CC) on performance metrics for T0++ and InstructGPT Curie, the best performing GPT-3 model when using calibrated prompts. Scores are the mean/standard error of 6 training replicates. Overall improvements due to calibration are in bold.

4.6.2 Prompt Calibration. Calibration had significant performance impact on all language models. Table 4 contains the overall benefit, in F1-score and accuracy, from using contextual calibration for T0++ and InstructGPT Curie. Complete pre- and post-calibration performance scores for all models are reported in the Appendix §6.3. In many cases, calibration provides significant performance improvements, with the largest increases seen in cases where the uncalibrated model had pathological performance. Figure 4 provides additional insight into calibration, where prompts evaluated with InstructGPT Curie and Ada often resulted in zero or extremely low coverage, causing training failures. Comparing coverage and accuracy of the original WRENCH labeling functions against their prompted versions shows how prompts result in much higher coverage than the same rule as expressed in code. For SMS, WRENCH keyword labeling functions (the blue points) are high precision, low coverage and highly tailored to the SMS task. Despite this low coverage, an end model trained with data generated by these labeling functions performs quite well, with 92.4 F1. For T0++ models, prompts are noisier, with higher coverage and lower accuracy especially in the positive class. Despite this, by combining and denoising signal across multiple prompts, T0++ achieves end model scores of 91.8 F1, only a 0.6 point drop.

Figures 5 shows how contextual calibration, at the level of individual prompts, can result in an unclear trade-off between accuracy and coverage. This plot presents the absolute change in accuracy and coverage between an uncalibrated prompt its calibrated equivalent. Recalibration generally increases a prompt's coverage, i.e., the number of labeled points, often at the cost of decreased accuracy. For T0++ models, accuracy decreased an average of 1.5 points while coverage increased by 2.4 points. For the InstructGPT models, the change is more substantial, with decreases in accuracy of 2.0 to 10.5 points while coverage increased by 40 to 69.7 points. For the Babbage and Ada engines, many prompts are driven to nearly 100% coverage, i.e., labeling the entire training set, due in part to a prompt responding with the same answer for every example. Only T0++ and InstructGPT Curie consistently improve prompt accuracy in

Fig. 4. SMS prompted labeling function coverage (x-axis) vs. accuracy (y-axis). The top figure is calibrated using contextual calibration and the bottom is uncalibrated. WRENCH Benchmark labeling function performance is in blue in every subfigure, which in SMS favors high precision, extremely low-coverage (<2%).

the positive (minority) class. The negative class in T0++ had very little change in accuracy, with calibration increasing coverage at little-to-no change in accuracy. T0++ is the only language model where calibration consistently resulted in more conservative labelers, i.e., prompts where accuracy increased and coverage decreased. Class-conditional views of these figures are available in the Appendix §6.3.

4.6.3 Diversity Measures. A key factor influencing labeling function performance is how they interact with other labeling functions. As in ensembling, we want labelers that provide complimentary information and have low correlated error rates, which improves ensemble efficiency and enables combining many weak classifiers to achieve stronger classification performance. To gain insight into the diversity of prompted labeling functions, we compute metrics informed by ensemble diversity measures [29]. Given a pair of labelers, i and j, we construct a 2x2 contingency table of vote counts for pairs of unlabeled examples. In binary classification, where N^{ij} is the total number of label pairs emitted by labelers i and j, this table contains $N^{00} + N^{10} + N^{01} + N^{11}$ covered instances. We consider the following diversity measures defined using these counts, normalizing all measures by the total size of the unlabeled training set.

(1) *Agreement* $:= N^{00} + N^{11}$
(2) *Disagreement* $:= N^{10} + N^{01}$
(3) *Double Fault* $:= N^{00}$
(4) *Double Correct* $:= N^{11}$

Agreement and disagreement provide measures of correlation between two labeling function prompts and enable characterizing the degree to which prompts provide complimentary label information.
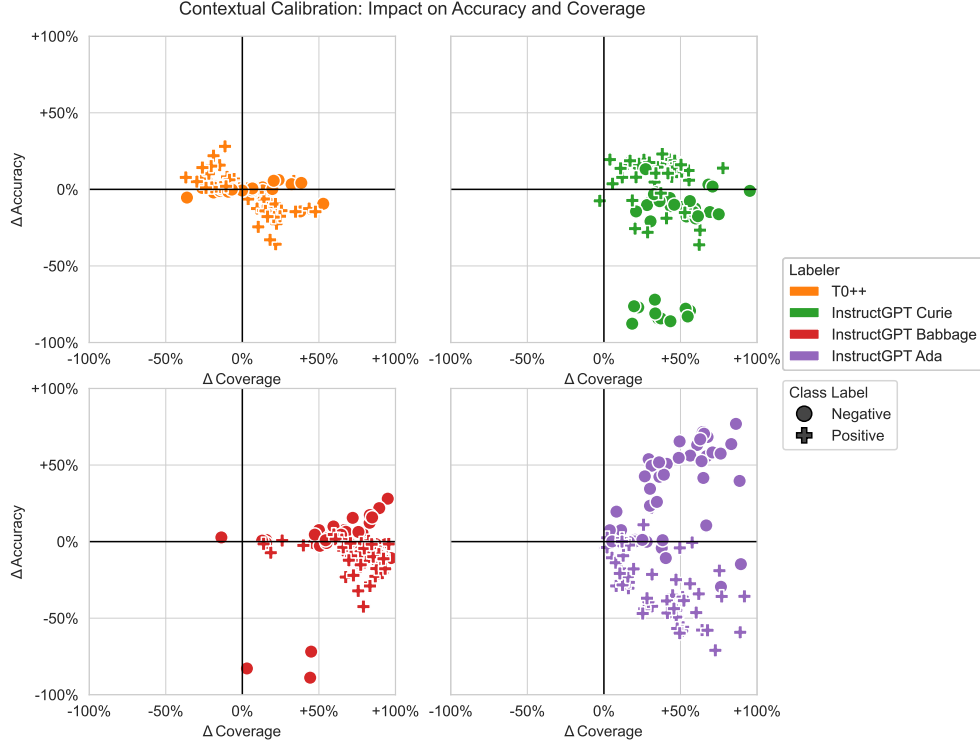
Fig. 5. Absolute change in accuracy and coverage after contextual calibration for all prompted labeling functions and language models. Each subfigure contains points from all datasets. The x-axis is change in coverage, the y-axis is change in accuracy, and each point reflects the change in that prompt's labeling performance after calibration.

Figure 6 shows a heatmap view of pairwise diversity of the YouTube dataset. Note there is more variation (disagreement) in the T0++ models and less agreement (double fault and double correct) compared to the InstructGPT family of models. The Babbage model, for example, generates strongly correlated labels and less variation in label signal. T0++ has higher variation in labels and less correlated errors across both classes. Lower correlated errors suggests that prompts evaluated using T0++ are providing complimentary label information, resulting in greater ensemble efficiency and improving overall model performance [24]. Similar patterns are observed in the other datasets (see Appendix §6.5).

5 DISCUSSION AND CONCLUSION

Developing flexible methods to query and adapt large-scale foundation models for downstream tasks is emerging as a critical component of machine learning systems. Our work demonstrates several benefits of using prompted weak supervision to query and repurpose information found in language models. Combining multiple prompted labeling functions provides significant improvements over underspecified prompts commonly used for zero-shot classification. By formulating tasks using multiple prompts, prompted weak supervision provides an inspectable mechanism for contextualizing task insight and querying knowledge found in large language models.

Prompts provide several advantages that compliment traditional code-based labeling functions. Unlike code, which is static and potentially expensive to refine, prompts are interpreted by an underlying language model, meaning the
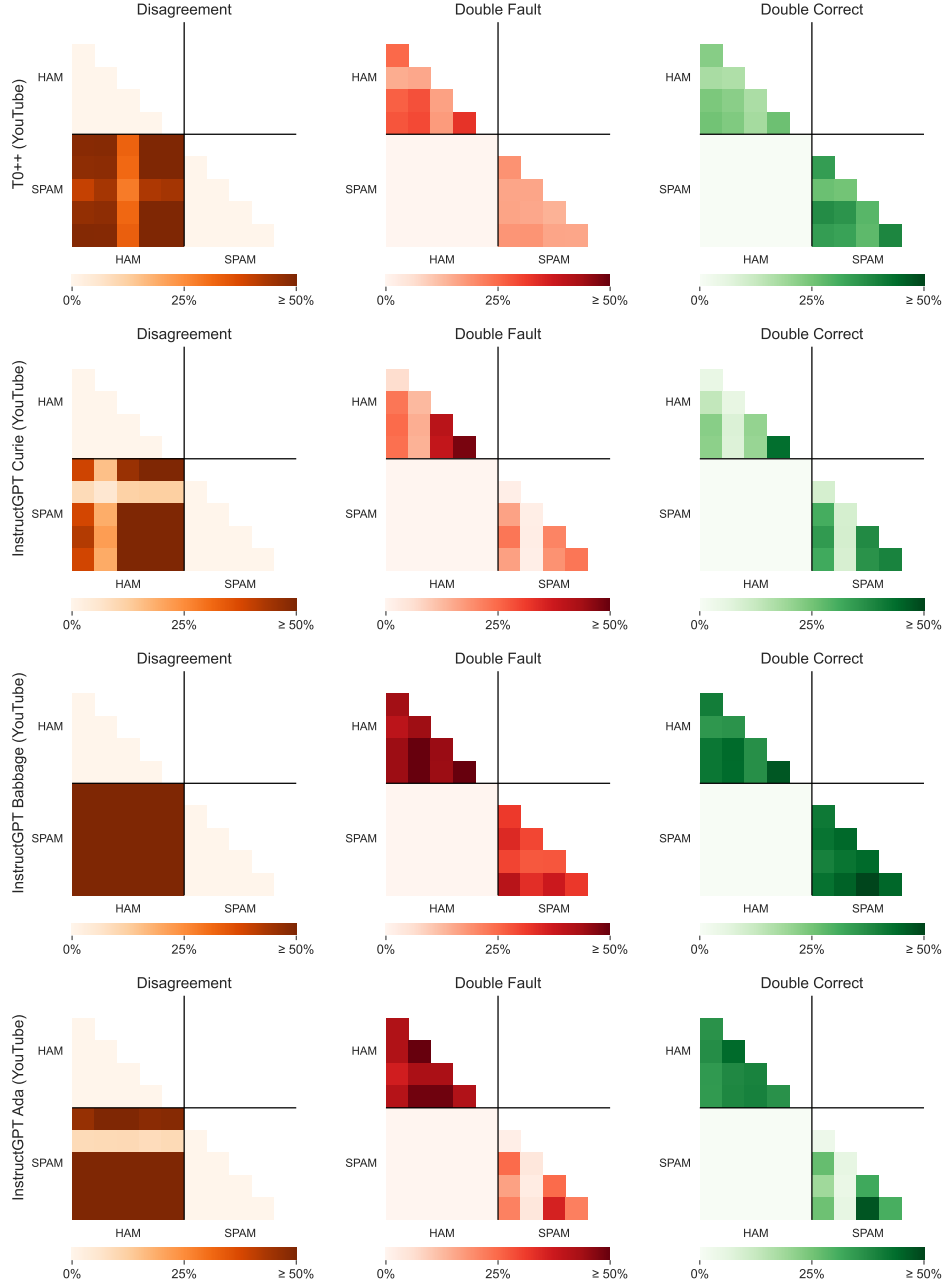
Fig. 6. YouTube prompted labeling function pairwise diversity measures: *disagreement* (left), *double fault* (center), *double correct* (right). Each matrix cell represents the percentage of training examples, indicated by color intensity, where prompts i, j both label an example. Rows are sorted by class label (one per-prompt) to emphasize block structure. Note some blocks are zero by definition, e.g., double fault measures when two prompts both emit the same incorrect label so the SPAM/HAM block is zero.

labels generated by prompts may improve as language models themselves continue improving. Moreover, the prompts explored in this work likely underestimate the potential performance of our approach, as we focused on translating existing labeling functions rather than developing and refining new prompts.

In our experiments, T0++, which was pretrained with multi-task prompted examples, consistently outperforms the InstructGPT family of language models when used for prompted weak supervision. Future work may consider methods of generating additional prompted pretraining data that aligns more closely with how SMEs approach prompt design in weakly supervised workflows. This is a particularly exciting use of data exhaust, as the process of querying and interacting with a language model can be used to directly improve the quality of the underlying model [37].

Finally, the success of contextual calibration underscores the benefits and current limitations of recalibration methods for prompt-based zero-shot learning. Performance gains, while consistent at the level of collections of prompts, is inconsistent and brittle at the level of an individual prompt. As new methods continue to improve language model calibration, we expect prompted weak supervision to benefit by increasing the ability of SMEs to refine the operating threshold of individual labeling functions.

ACKNOWLEDGMENTS

REFERENCES

[1] Chidubem Arachie and Bert Huang. 2021. Constrained Labeling for Weakly Supervised Learning. In *Uncertainty in Aritficial Intelligence (UAI)*.

[2] Chidubem Arachie and Bert Huang. 2021. A general framework for adversarial label learning. *Journal of Machine Learning Research* 22, 118 (2021), 1–33.

[3] Anonymous Authors. 2022. Prompt Consistency for Zero-Shot Task Generalization. Submitted to ACL Rolling Review. https://openreview.net/pdf?id=Ig8xeTpEmHf

[4] Stephen H. Bach, Bryan He, Alexander Ratner, and Christopher Ré. 2017. Learning the Structure of Generative Models without Labeled Data. In *International Conference on Machine Learning (ICML)*.

[5] Stephen H. Bach, Daniel Rodriguez, Yintao Liu, Chong Luo, Haidong Shao, Cassandra Xia, Souvik Sen, Alexander Ratner, Braden Hancock, Houman Alborzi, Rahul Kuchhal, Christopher Ré, and Rob Malkin. 2019. Snorkel DryBell: A Case Study in Deploying Weak Supervision at Industrial Scale. In *ACM SIGMOD Conference on Management of Data (SIGMOD) Industry Track*.

[6] Samantha Biegel, Rafah El-Khatib, Luiz Otavio Vilas Boas Oliveira, Max Baak, and Nanne Aben. 2021. Active WeaSuL: Improving Weak Supervision with Active Learning. In *ICLR Workshop on Weakly Supervised Learning*.

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

[8] Luiz Bonifacio, Hugo Abonizio, Marzieh Fadaee, and Rodrigo Nogueira. 2022. InPars: Data Augmentation for Information Retrieval using Large Language Models. *arXiv preprint arXiv:2202.05144* (2022).

[9] Eran Bringer, Abraham Israeli, Yoav Shoham, Alex Ratner, and Christopher Ré. 2019. Osprey: Weak supervision of imbalanced extraction problems without code. In *International Workshop on Data Management for End-to-End Machine Learning (DEEM)*.

[10] 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. *Neural Information Processing Systems (NeurIPS)* (2020).

[11] Clemens-Alexander Brust, Christoph Käding, and Joachim Denzler. 2020. Active and incremental learning with weak supervision. *KI-Künstliche Intelligenz* 34, 2 (2020), 165–180.

[12] Alison Callahan, Jason A Fries, Christopher Ré, James I Huddleston, Nicholas J Giori, Scott Delp, and Nigam H Shah. 2019. Medical device surveillance with electronic health records. *NPJ digital medicine* 2, 1 (2019), 1–10.

[13] Mayee F Chen, Daniel Y Fu, Dyah Adila, Michael Zhang, Frederic Sala, Kayvon Fatahalian, and Christopher Ré. 2022. Shoring Up the Foundations: Fusing Model Embeddings and Weak Supervision. *arXiv preprint arXiv:2203.13270* (2022).

[14] Yew Ken Chia, Lidong Bing, Soujanya Poria, and Luo Si. 2022. RelationPrompt: Leveraging Prompts to Generate Synthetic Data for Zero-Shot Relation Triplet Extraction. In *Findings of the Association for Computational Linguistics*.

[15] Mark Craven, Johan Kumlien, et al. 1999. Constructing biological knowledge bases by extracting information from text sources.. In *Intelligent Systems for Molecular Biology (ISMB)*.

[16] A. P. Dawid and A. M. Skene. 1979. Maximum Likelihood Estimation of Observer Error-Rates Using the EM Algorithm. *Journal of the Royal Statistical Society C* 28, 1 (1979), 20–28.

[17] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Meeting of the North American Association for Computational Linguistics (NAACL)*.

[18] Jared A Dunnmon, Alexander J Ratner, Khaled Saab, Nishith Khandwala, Matthew Markert, Hersh Sagreiya, Roger Goldman, Christopher Lee-Messer, Matthew P Lungren, Daniel L Rubin, et al. 2020. Cross-modal data programming enables rapid medical machine learning. *Patterns* 1, 2 (2020).

[19] Yanai Elazar, Nora Kassner, Shauli Ravfogel, Abhilasha Ravichander, Eduard Hovy, Hinrich Schütze, and Yoav Goldberg. 2021. Measuring and improving consistency in pretrained language models. *Transactions of the Association for Computational Linguistics* 9 (2021), 1012–1031.

[20] Sabri Eyuboglu, Geoffrey Angus, Bhavik N Patel, Anuj Pareek, Guido Davidzon, Jin Long, Jared Dunnmon, and Matthew P Lungren. 2021. Multi-task weak supervision enables anatomically-resolved abnormality detection in whole-body FDG-PET/CT. *Nature communications* 12, 1 (2021), 1–15.

[21] Jason A Fries, Ethan Steinberg, Saelig Khattar, Scott L Fleming, Jose Posada, Alison Callahan, and Nigam H Shah. 2021. Ontology-driven weak supervision for clinical entity classification in electronic health records. *Nature communications* 12, 1 (2021), 1–11.

[22] 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.

[23] Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. 2020. The Pile: An 800GB Dataset of Diverse Text for Language Modeling. *arXiv preprint arXiv:2101.00027* (2020).

[24] Raphael Gontijo-Lopes, Yann Dauphin, and Ekin Dogus Cubuk. 2022. No One Representation to Rule Them All: Overlapping Features of Training Methods. In *International Conference on Learning Representations*. https://openreview.net/forum?id=BK-4qbGgIE3

[25] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. 2017. On Calibration of Modern Neural Networks. In *ICML (Proceedings of Machine Learning Research, Vol. 70)*. PMLR, 1321–1330.

[26] Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. 2020. How Can We Know What Language Models Know? *Transactions of the Association for Computational Linguistics* 8 (2020), 423–438. https://doi.org/10.1162/tacl_a_00324

[27] Giannis Karamanolakis, Subhabrata Mukherjee, Guoqing Zheng, and Ahmed Hassan Awadallah. 2021. Self-Training with Weak Supervision. In *Meeting of the North American Association for Computational Linguistics (NAACL)*.

[28] Zhaobin Kuang, Chidubem Arachie, Bangyong Liang, Pradyumna Narayana, Giulia DeSalvo, Michael Quinn, Bert Huang, Geoffrey Downs, and Yang Yang. 2021. Firebolt: Weak Supervision Under Weaker Assumptions. In *Artificial Intelligence and Statistics (AISTATS)*.

[29] Ludmila I Kuncheva and Christopher J Whitaker. 2003. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine learning* 51, 2 (2003), 181–207.

[30] Hunter Lang, Monica Agrawal, Yoon Kim, and David Sontag. 2022. Co-training Improves Prompt-based Learning for Large Language Models. *arXiv preprint arXiv:2202.00828* (2022).

[31] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *arXiv preprint arXiv:2107.13586* (2021).

[32] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Ro{BERT}a: A Robustly Optimized {BERT} Pretraining Approach. https://openreview.net/forum?id=SyxS0T4tvS

[33] A. Mazzetto, C. Cousins, D. Sam, S. H. Bach, and E. Upfal. 2021. Adversarial Multiclass Learning under Weak Supervision with Performance Guarantees. In *International Conference on Machine Learning (ICML)*.

[34] A. Mazzetto, D. Sam, A. Park, E. Upfal, and S. H. Bach. 2021. Semi-Supervised Aggregation of Dependent Weak Supervision Sources With Performance Guarantees. In *Artificial Intelligence and Statistics (AISTATS)*.

[35] M. Mintz, S. Bills, R. Snow, and D. Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Meeting of the Association for Computational Linguistics (ACL)*.

[36] Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. 2022. Cross-Task Generalization via Natural Language Crowdsourcing Instruction. In *Meeting of the Association for Computational Linguistics (ACL)*.

[37] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. *CoRR* abs/2203.02155 (2022).

[38] John Platt et al. 1999. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers* 10, 3 (1999), 61–74.

[39] Jack W Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, et al. 2021. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446* (2021).

[40] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research* 21, 140 (2020), 1–67.

[41] A. J. Ratner, S. H. Bach, H. E. Ehrenberg, J. Fries, S. Wu, and C. Ré. 2020. Snorkel: Rapid Training Data Creation with Weak Supervision. *The VLDB Journal* 29, 2 (2020), 709–730.

[42] Alexander J Ratner, Christopher M De Sa, Sen Wu, Daniel Selsam, and Christopher Ré. 2016. Data Programming: Creating large training sets, quickly. In *Neural Information Processing Systems (NeurIPS)*.

[43] Alexander J Ratner, Braden Hancock, Jared Dunnmon, Frederic Sala, Shreyash Pandey, and Christopher Ré. 2019. Training Complex Models with Multi-Task Weak Supervision. In *AAAI Conference on Artificial Intelligence (AAAI)*.

[44] Esteban Safranchik, Shiying Luo, and Stephen H. Bach. 2020. Weakly Supervised Sequence Tagging from Noisy Rules. In *AAAI Conference on Artificial Intelligence (AAAI)*.

[45] Frederic Sala, Paroma Varma, Shiori Sagawa, Jason Fries, Daniel Fu, Saelig Khattar, Ashwini Ramamoorthy, Ke Xiao, Kayvon Fatahalian, James Priest, et al. 2019. Multi-resolution weak supervision for sequential data. In *Neural Information Processing Systems (NeurIPS)*.

[46] Victor Sanh, Albert Webson, Colin Raffel, Stephen H. Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Fevry, Jason Alan Fries, Ryan Teehan, Stella Biderman, Leo Gao, Tali Bers, Thomas Wolf, and Alexander M. Rush. 2022. Multitask Prompted Training Enables Zero-Shot Task Generalization. In *International Conference on Learning Representations (ICLR)*.

[47] Timo Schick and Hinrich Schütze. 2021. Generating Datasets with Pretrained Language Models. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

[48] Changho Shin, Winfred Li, Harit Vishwakarma, Nicholas Roberts, and Frederic Sala. 2022. Universalizing Weak Supervision. In *International Conference on Learning Representations (ICLR)*.

[49] Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh. 2020. AutoPrompt: Eliciting knowledge from language models with automatically generated prompts. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

[50] Sahaana Suri, Raghuveer Chanda, Neslihan Bulut, Pradyumna Narayana, Yemao Zeng, Peter Bailis, Sugato Basu, Girija Narlikar, Christopher Ré, and Abishek Sethi. 2020. Leveraging Organizational Resources to Adapt Models to New Data Modalities. *Proc. VLDB Endow.* 13, 12 (2020), 3396–3410.

[51] Paroma Varma, Bryan He, Dan Iter, Peng Xu, Rose Yu, Christopher De Sa, and Christopher Ré. 2016. Socratic learning: Augmenting generative models to incorporate latent subsets in training data. *arXiv preprint arXiv:1610.08123* (2016).

[52] Paroma Varma and Christopher Ré. 2018. Snuba: Automating weak supervision to label training data. *Proceedings of the VLDB Endowment* 12, 3 (2018), 223.

[53] Paroma Varma, Fred Sala, Ann He, Alex Ratner, and Christopher Ré. 2019. Learning Dependency Structures for Weak Supervision Models. In *International Conference on Machine Learning (ICML)*.

[54] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Neural Information Processing Systems (NeurIPS)*.

[55] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, and Denny Zhou. 2022. Self-Consistency Improves Chain of Thought Reasoning in Language Models. *arXiv preprint arXiv:2203.11171* (2022).

[56] Albert Webson and Ellie Pavlick. 2021. Do Prompt-Based Models Really Understand the Meaning of their Prompts? *arXiv preprint arXiv:2109.01247* (2021).

[57] Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2022. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations (ICLR)*.

[58] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain of Thought Prompting Elicits Reasoning in Large Language Models. *arXiv preprint arXiv:2201.11903* (2022).

[59] Yuxiang Wu, Matt Gardner, Pontus Stenetorp, and Pradeep Dasigi. 2022. Generating Data to Mitigate Spurious Correlations in Natural Language Inference Datasets. In *Meeting of the Association for Computational Linguistics (ACL)*.

[60] Jiacheng Ye, Jiahui Gao, Qintong Li, Hang Xu, Jiangtao Feng, Zhiyong Wu, Tao Yu, and Lingpeng Kong. 2022. ZeroGen: Efficient Zero-shot Learning via Dataset Generation. *arXiv preprint arXiv:2202.07922* (2022).

[61] P. Yu, T. Ding, and S. H. Bach. 2022. Learning from Multiple Noisy Partial Labelers. In *Artificial Intelligence and Statistics (AISTATS)*.

[62] Yue Yu, Simiao Zuo, Haoming Jiang, Wendi Ren, Tuo Zhao, and Chao Zhang. 2021. Fine-Tuning Pre-trained Language Model with Weak Supervision: A Contrastive-Regularized Self-Training Approach. In *Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

[63] Eric Zelikman, Yuhuai Wu, and Noah D Goodman. 2022. STaR: Bootstrapping Reasoning With Reasoning. *arXiv preprint arXiv:2203.14465* (2022).

[64] 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).

[65] Jieyu Zhang, Bohan Wang, Xiangchen Song, Yujing Wang, Yaming Yang, Jing Bai, and Alexander Ratner. 2022. Creating Training Sets via Weak Indirect Supervision. In *International Conference on Learning Representations (ICLR)*.

[66] Jieyu Zhang, Yue Yu, Yinghao Li, Yujing Wang, Yaming Yang, Mao Yang, and Alexander Ratner. 2021. WRENCH: A Comprehensive Benchmark for Weak Supervision. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*. https://openreview.net/forum?id=Q9SKS5k8io

[67] Rongzhi Zhang, Yue Yu, Pranav Shetty, Le Song, and Chao Zhang. 2022. PRBoost: Prompt-Based Rule Discovery and Boosting for Interactive Weakly-Supervised Learning. In *Meeting of the Association for Computational Linguistics (ACL)*.

[68] Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate Before Use: Improving Few-shot Performance of Language Models. In *ICML (Proceedings of Machine Learning Research, Vol. 139)*. PMLR, 12697–12706.

6 APPENDIX

6.1 GPT-3 API Costs

Dataset	Supervision	#Queries	InstructGPT Language Models			
			Ada	Babbage	Curie	DaVinci
YouTube	Zero Shot	1,586	$0.04	$0.06	$0.28	$2.76
YouTube	Prompted WS	10,586	$0.43	$0.65	$3.24	$32.40
SMS	Zero Shot	4,571	$0.11	$0.16	$0.82	$8.24
SMS	Prompted WS	333,683	$9.72	$14.59	$72.93	$729.31
Spouse	Zero Shot	22,254	$1.52	$2.28	$11.40	$113.97
Spouse	Prompted WS	200,286	$16.02	$24.03	$120.16	$1,201.62

Table 5. OpenAI API estimated query costs for labeling WRENCH training sets with InstructGPT family of language models. See https://openai.com/api/pricing/ (accessed 03/01/2022).

6.2 Zero Shot Prompt Baseline

6.2.1 End Model Generalization. Table 6 contains performance of Zero Shot (ZS) prompts directly evaluated on test data compared to the same prompts used for prompted weak supervision, where we programmatically label the training split, train a RoBERTa end model, and evaluate on test data (ZS+End Model). All prompts are contextually calibrated. The RoBERTa end model provides consistent improvements.

	Youtube (Accuracy)		SMS (F1)		Spouse (F1)	
	ZS	ZS+End Model	ZS	ZS+End Model	ZS	ZS+End Model
T0++	55.6 (0.0)	**58.7 (2.4)**	34.0 (0.0)	**83.2 (2.4)**	**63.0 (0.0)**	41.5 (13.1)
Curie	**54.4 (0.0)**	52.8 (0.0)	0.0 (0.0)	0.0 (0.0)	38.3 (0.0)	**49.6 (1.0)**
Babbage	55.6 (0.0)	**78.5 (3.0)**	20.6 (0.0)	**32.2 (3.0)**	26.9 (0.0)	**40.9 (0.9)**
Ada	44.8 (0.0)	**51.7 (2.4)**	25.1 (0.0)	**26.3 (2.6)**	17.2 (0.0)	**19.1 (0.8)**

Table 6. Comparing the Zero Shot (ZS) prompt as a direct classification model for test data versus the same prompt when used as a labeler to programmatically generate training data for a RoBERTa model (ZS+End Model). The best performing prompt performances are in bold.

6.2.2 Zero Shot Labeling Function. Table 7 contains results for prompted weak supervision models that add the Zero Shot prompt as an additional labeling function. Performance benefits were mixed, with models generally negatively impacted by incorporating the Zero Shot labeler. Here T0++ had an average improvement of 0.2 F1 points, while InstructGPT Curie and Babbage has an average drop of 2.8 and 0.8 F1 points respectively. InstructGPT Ada improved by 2.6 F1 points on average.

6.3 Prompt Calibration

6.3.1 Contextual Calibration. We find calibration improves performance of prompted labeling functions, with the largest gains found in settings where uncalibrated prompts display pathological performance. We observed that the

Dataset	Prompts	Language Model	Precision	Recall	F1	Accuracy
YouTube	PWS+ZS	T0++	92.3 (0.5)	90.8 (0.9)	91.0 (0.8)	91.2 (0.8)
YouTube	PWS	T0++	92.6 (0.5)	91.7 (0.5)	**91.9 (0.5)**	**92.0 (0.5)**
SMS	PWS+ZS	T0++	96.5 (0.8)	92.8 (1.4)	**94.5 (0.4)**	**98.6 (0.1)**
SMS	PWS	T0++	95.9 (2.5)	88.1 (1.1)	91.8 (1.6)	97.9 (0.4)
Spouse	PWS+ZS	T0++	52.6 (1.0)	75.4 (2.4)	61.8 (0.8)	92.5 (0.2)
Spouse	PWS	T0++	54.2 (1.8)	75.4 (1.2)	**62.9 (0.8)**	**92.8 (0.3)**
YouTube	PWS+ZS	InstructGPT Curie	80.5 (1.1)	70.5 (1.1)	68.9 (1.4)	72.0 (1.0)
YouTube	PWS	InstructGPT Curie	80.1 (1.0)	77.1 (2.1)	**76.7 (2.3)**	**77.7 (1.9)**
SMS	PWS+ZS	InstructGPT Curie	53.4 (5.8)	80.6 (2.9)	63.0 (4.9)	86.0 (3.7)
SMS	PWS	InstructGPT Curie	60.6 (11.5)	83.8 (4.4)	**65.7 (5.8)**	**86.2 (3.9)**
Spouse	PWS+ZS	InstructGPT Curie	35.6 (2.2)	58.9 (5.6)	**43.2 (0.8)**	**87.5 (1.2)**
Spouse	PWS	InstructGPT Curie	29.5 (0.9)	67.5 (2.0)	41.0 (0.9)	84.3 (0.7)
YouTube	PWS+ZS	InstructGPT Babbage	83.7 (0.6)	83.0 (0.7)	83.0 (0.6)	83.1 (0.6)
YouTube	PWS	InstructGPT Babbage	85.8 (1.2)	84.8 (1.4)	**84.9 (1.4)**	**85.1 (1.3)**
SMS	PWS+ZS	InstructGPT Babbage	13.4 (0.0)	100.0 (0.0)	23.6 (0.0)	13.4 (0.0)
SMS	PWS	InstructGPT Babbage	13.4 (0.0)	100.0 (0.0)	23.6 (0.0)	13.4 (0.0)
Spouse	PWS+ZS	InstructGPT Babbage	25.0 (2.7)	59.5 (4.0)	34.3 (2.2)	**80.9 (2.4)**
Spouse	PWS	InstructGPT Babbage	24.2 (2.2)	67.7 (5.6)	**34.9 (1.7)**	79.3 (1.9)
YouTube	PWS+ZS	InstructGPT Ada	54.0 (9.5)	50.6 (0.3)	**36.0 (0.7)**	**53.3 (0.3)**
YouTube	PWS	InstructGPT Ada	34.8 (8.4)	50.1 (0.1)	34.7 (0.2)	52.9 (0.1)
SMS	PWS+ZS	InstructGPT Ada	13.4 (0.0)	100.0 (0.0)	23.6 (0.0)	13.4 (0.0)
SMS	PWS	InstructGPT Ada	16.6 (1.3)	99.5 (0.5)	**28.3 (1.8)**	**30.4 (6.2)**
Spouse	PWS+ZS	InstructGPT Ada	20.0 (0.4)	53.1 (1.7)	**29.0 (0.3)**	79.0 (0.8)
Spouse	PWS	InstructGPT Ada	16.8 (5.6)	20.6 (8.1)	17.7 (6.2)	**88.7 (1.4)**

Table 7. Incorporating the Zero Shot prompt as an additional labeling function in Prompted Weak Superision.

InstructGPT family of language models performed very poorly in many zero shot and prompted weak supervision experiments, as shown in Table 8. The performance benefits of contextual calibration for all language models and datasets are outlined for the Zero Shot baseline in Table 9 and for prompted weak supervision in Table 10.

Figures 12 and 13 show the class conditional view of calibration changes vs. accuracy changes for all datasets and language models. Note that for T0++, prompts labeling the negative class have little-to-no change in accuracy after calibration.

6.4 WRENCH Labeling Function Prompts

The complete set of translated WRENCH labeling functions are show in Tables 11, 12, and 13.

6.5 Labeling Function Diversity

Figure 9 shows a heatmap view of diversity metrics for the original WRENCH labeling functions. Figures 10 and 11 show diversity measures for the SMS and Spouse datasets respectively.

	Youtube (Accuracy)		SMS (F1)		Spouse (F1)	
	Zero Shot	Prompted WS	Zero Shot	Prompted WS	Zero Shot	Prompted WS
WRENCH Benchmark	-	94.9 (0.5)	-	92.4 (0.5)	-	37.9 (2.8)
T0++	54.1 (0.6)	**95.4 (0.4)**	84.1 (1.7)	91.4 (1.6)	**60.6 (0.8)**	44.9 (1.3)
InstructGPT Curie	52.8 (0.0)	77.7 (1.1)	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)
InstructGPT Babbage	52.8 (0.0)	69.2 (3.0)	0.0 (0.0)	40.5 (10.3)	33.8 (7.2)	0.0 (0.0)
InstructGPT Ada	52.8 (0.0)	67.3 (1.2)	0.0 (0.0)	**94.7 (0.5)**	26.1 (1.2)	0.0 (0.0)

Table 8. The same performance metrics presented in Table 3 but with uncalibrated prompts.

Dataset	Language Model	CC	Precision	Recall	F1	±F1	Acc.	±Acc.
YouTube	T0++	✓	61.5 (7.3)	56.5 (2.6)	48.0 (4.9)	**+10.5 (5.0)**	58.7 (2.4)	**+4.7 (2.4)**
			50.6 (10.9)	51.3 (0.6)	37.5 (1.3)	–	54.1 (0.6)	–
SMS	T0++	✓	77.5 (3.9)	90.3 (1.1)	83.2 (2.4)	-0.9 (2.1)	95.0 (0.8)	-0.5 (0.7)
			82.4 (4.5)	88.1 (4.3)	84.1 (1.7)	–	95.5 (0.5)	–
Spouse	T0++	✓	37.3 (11.8)	46.7 (14.8)	41.5 (13.1)	-19.1 (13.6)	92.7 (0.3)	**+0.1 (0.7)**
			54.3 (2.2)	69.7 (3.1)	60.6 (0.8)	–	92.6 (0.5)	–
YouTube	InstructGPT Curie	✓	26.4 (0.0)	50.0 (0.0)	34.6 (0.0)	0.0 (0.0)	52.8 (0.0)	0.0 (0.0)
			26.4 (0.0)	50.0 (0.0)	34.6 (0.0)	–	52.8 (0.0)	–
SMS	InstructGPT Curie	✓	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)	86.6 (0.0)	0.0 (0.0)
			0.0 (0.0)	0.0 (0.0)	0.0 (0.0)	–	86.6 (0.0)	–
Spouse	InstructGPT Curie	✓	37.9 (1.8)	74.5 (4.8)	49.6 (1.0)	**+49.6 (1.0)**	87.7 (1.0)	-4.2 (1.0)
			0.0 (0.0)	0.0 (0.0)	0.0 (0.0)	–	91.9 (0.0)	–
YouTube	InstructGPT Babbage	✓	81.4 (2.2)	77.6 (3.2)	77.2 (3.7)	**+42.6 (3.7)**	78.5 (3.0)	**+25.7 (3.0)**
			26.4 (0.0)	50.0 (0.0)	34.6 (0.0)	–	52.8 (0.0)	–
SMS	InstructGPT Babbage	✓	20.8 (2.7)	79.4 (5.4)	32.2 (3.0)	**+32.2 (3.0)**	52.0 (8.0)	-34.6 (8.0)
			0.0 (0.0)	0.0 (0.0)	0.0 (0.0)	–	86.6 (0.0)	–
Spouse	InstructGPT Babbage	✓	28.5 (1.1)	75.3 (6.0)	40.9 (0.9)	**+7.1 (7.1)**	82.5 (1.3)	-6.1 (0.6)
			28.1 (5.8)	43.7 (10.0)	33.8 (7.2)	–	88.5 (0.8)	–
YouTube	InstructGPT Ada	✓	40.6 (7.7)	53.4 (1.9)	43.4 (5.3)	**+8.9 (5.3)**	51.7 (2.4)	-1.1 (2.4)
			26.4 (0.0)	50.0 (0.0)	34.6 (0.0)	–	52.8 (0.0)	–
SMS	InstructGPT Ada	✓	15.3 (1.9)	99.8 (0.2)	26.3 (2.6)	**+26.3 (2.6)**	21.1 (7.7)	-65.5 (7.7)
			0.0 (0.0)	0.0 (0.0)	0.0 (0.0)	–	86.6 (0.0)	–
Spouse	InstructGPT Ada	✓	10.6 (0.5)	97.9 (0.6)	19.1 (0.8)	-7.0 (1.7)	32.2 (3.9)	-23.2 (5.8)
			15.1 (0.8)	96.1 (0.9)	26.1 (1.2)	–	55.4 (2.9)	–

Table 9. Performance impact of contextual calibration (CC) on all Zero Shot baseline models. Scores are the mean/standard error of 6 training replicates. Overall improvements due to calibration are in bold.

Dataset	Language Model	CC	Precision	Recall	F1	±F1	Acc.	±Acc.
YouTube	T0++	✓	92.6 (0.5)	91.7 (0.5)	91.9 (0.5)	-3.5 (0.6)	92.0 (0.5)	-3.4 (0.6)
YouTube	T0++		95.7 (0.4)	95.2 (0.5)	95.4 (0.4)	–	95.4 (0.4)	–
SMS	T0++	✓	95.9 (2.5)	88.1 (1.1)	91.8 (1.6)	**+0.3 (2.5)**	97.9 (0.4)	+0.2 (0.7)
SMS	T0++		91.6 (3.2)	91.5 (0.8)	91.4 (1.6)	–	97.7 (0.5)	–
Spouse	T0++	✓	54.2 (1.8)	75.4 (1.2)	62.9 (0.8)	**+18.0 (1.7)**	92.8 (0.3)	**+10.0 (1.4)**
Spouse	T0++		30.7 (1.6)	86.0 (2.8)	44.9 (1.3)	–	82.8 (1.2)	–
YouTube	InstructGPT Curie	✓	80.1 (1.0)	77.1 (2.1)	76.7 (2.3)	**+0.8 (2.0)**	77.7 (1.9)	-0.1 (1.6)
YouTube	InstructGPT Curie		84.8 (0.6)	76.4 (1.2)	75.9 (1.3)	–	77.7 (1.1)	–
SMS	InstructGPT Curie	✓	60.6 (11.5)	83.8 (4.4)	65.7 (5.8)	**+65.7 (5.8)**	86.2 (3.9)	-0.4 (3.9)
SMS	InstructGPT Curie		0.0 (0.0)	0.0 (0.0)	0.0 (0.0)	–	86.6 (0.0)	–
Spouse	InstructGPT Curie	✓	29.5 (0.9)	67.5 (2.0)	41.0 (0.9)	**+41.0 (0.9)**	84.3 (0.7)	-7.7 (0.7)
Spouse	InstructGPT Curie		0.0 (0.0)	0.0 (0.0)	0.0 (0.0)	–	91.9 (0.0)	–
YouTube	InstructGPT Babbage	✓	85.8 (1.2)	84.8 (1.4)	84.9 (1.4)	**+18.2 (4.3)**	85.1 (1.3)	**+15.9 (3.6)**
YouTube	InstructGPT Babbage		74.2 (3.4)	68.2 (3.0)	66.7 (3.5)	–	69.2 (3.0)	–
SMS	InstructGPT Babbage	✓	13.4 (0.0)	100.0 (0.0)	23.6 (0.0)	-16.9 (10.3)	13.4 (0.0)	-72.1 (2.4)
SMS	InstructGPT Babbage		48.9 (12.2)	48.5 (15.1)	40.5 (10.3)	–	85.5 (2.4)	–
Spouse	InstructGPT Babbage	✓	24.2 (2.2)	67.7 (5.6)	34.9 (1.7)	**+34.9 (1.7)**	79.3 (1.9)	-12.6 (1.9)
Spouse	InstructGPT Babbage		0.0 (0.0)	0.0 (0.0)	0.0 (0.0)	–	91.9 (0.0)	–
YouTube	InstructGPT Ada	✓	34.8 (8.4)	50.1 (0.1)	34.7 (0.2)	-27.6 (1.9)	52.9 (0.1)	-14.4 (1.2)
YouTube	InstructGPT Ada		77.5 (1.4)	65.5 (1.3)	62.3 (1.9)	–	67.3 (1.2)	–
SMS	InstructGPT Ada	✓	16.6 (1.3)	99.5 (0.5)	28.3 (1.8)	-66.4 (1.9)	30.4 (6.2)	-68.2 (6.2)
SMS	InstructGPT Ada		98.7 (0.7)	91.0 (0.8)	94.7 (0.5)	–	98.6 (0.1)	–
Spouse	InstructGPT Ada	✓	16.8 (5.6)	20.6 (8.1)	17.7 (6.2)	**+17.7 (6.2)**	88.7 (1.4)	-3.3 (1.4)
Spouse	InstructGPT Ada		0.0 (0.0)	0.0 (0.0)	0.0 (0.0)	–	91.9 (0.0)	–

Table 10. Performance impact of contextual calibration (CC) on all Prompted Weak Supervision models. Scores are the mean/standard error of 6 training replicates. Overall improvements due to calibration are in bold.

Model	Prompt Template	Label
T0++	Does the following comment reference the speaker's channel or video?\n\n[TEXT]	SPAM
	Does the following comment ask you to subscribe to a channel?\n\n[TEXT]	SPAM
	Does the following comment have a URL?\n\n[TEXT]	SPAM
	Does the following comment ask the reader to do something?\n\n[TEXT]	SPAM
	Does the following comment talk about a song?\n\n[TEXT]	HAM
	Does the following comment contain the words "check out"? \n\n[TEXT]	SPAM
	Is the following comment fewer than 5 words?\n\n[TEXT]	HAM
	Does the following comment mention a person's name?\n\n[TEXT]	HAM
	Does the following comment express a very strong sentiment?\n\n[TEXT]	HAM
	Does the following comment express a subjective opinion?\n\n[TEXT]	HAM
GPT-3	Q: Does the following comment "[TEXT]" reference the speaker's channel or video?\nA:	SPAM
	Q: Does the following comment "[TEXT]" ask you to subscribe to a channel?\nA:	SPAM
	Q: Does the following comment "[TEXT]" have a URL?\nA:	SPAM
	Q: Does the following comment "[TEXT]" ask the reader to do something?\nA:	SPAM
	Q: Does the following comment "[TEXT]" talk about a song?\nA:	HAM
	Q: Does the following comment "[TEXT]" contain the words "check out"?\nA:	SPAM
	Q: Is the following comment "[TEXT]" fewer than 5 words?\nA:	HAM
	Q: Does the following comment "[TEXT]" mention a person's name?\nA:	HAM
	Q: Does the following comment "[TEXT]" express a very strong sentiment?\nA:	HAM
	Q: Does the following comment "[TEXT]" express a subjective opinion?\nA:	HAM

Table 11. YouTube labeling function prompts with class labels HAM = 0, SPAM = 1. A label map transforms text completions to class labels, where "yes" emits the value denoted in the label column and "no" emits ABSTAIN.

Model	Prompt Template	Label
T0++	Does the following text message contain the words "[KEYWORDS]"?\n\n[TEXT]	
GPT-3	Q: Does the following text message "[TEXT]" contain the words "[KEYWORDS]"?\nA:	
[KEYWORDS]	*??1.50, ??500, ??5000, call for offer, cash prize, chat date, chat to, childporn, credits, dating call, direct, expires now, fantasies call, free phones, free price, free ringtones, free sex, free tone, guaranteed free, guaranteed gift, hard live girl, important lucky, inviting friends, latest, latest offer, message call, new mobiles, no extra, password, please call, sms reply, unlimited calls, urgent award guaranteed, urgent prize, voucher claim, welcome reply, win shopping, winner reward, won call, won cash, won cash prize, won claim*	SPAM
	I, I can did, I it, I miss, I used to, adventuring, amrita, can't talk, did u got, do you, fb, goodo, hee hee, i'll, jus, link, maggi, mine, my kids, noisy, praying, shit, should I, thanks, that's fine, thats nice, u how 2, we will, where are, wtf, your I	HAM

Table 12. SMS Labeling function prompts with class labels HAM = 0, SPAM = 1 that are defined by individual [KEYWORDS]. A label map transforms text completions to class labels, where "yes" emits the value denoted in the label column and "no" emits ABSTAIN.

Model	Prompt Template	Label
T0++	Context: [TEXT]\n\nIs there any mention of "spouse" between the entities [PERSON1] and [PERSON2]?	SPOUSE
	Context: [TEXT]\n\nIs there any mention of "spouse" before the entity [PERSON1]?	SPOUSE
	Context: [TEXT]\n\nIs there any mention of "spouse" before the entity [PERSON2]?	SPOUSE
	Context: [TEXT]\n\nDo [PERSON1] and [PERSON2] have the same last name?	SPOUSE
	Context: [TEXT]\n\nDid [PERSON1] and [PERSON2] get married?	SPOUSE
	Context: [TEXT]\n\nAre [PERSON1] and [PERSON2] family members?	NOT_SPOUSE
	Context: [TEXT]\n\nIs [PERSON1] said to be a family member?	NOT_SPOUSE
	Context: [TEXT]\n\nIs [PERSON2] said to be a family member?	NOT_SPOUSE
	Context: [TEXT]\n\nAre [PERSON1] and [PERSON2] dating?	NOT_SPOUSE
	Context: [TEXT]\n\nAre [PERSON1] and [PERSON2] co-workers?	NOT_SPOUSE
	Are [PERSON1] and [PERSON2] married?	SPOUSE
GPT-3	Context: "[TEXT]"\nQ: Is there any mention of "spouse" between the entities [PERSON1] and [PERSON2]?\nA:	SPOUSE
	Context: "[TEXT]"\nQ: Is there any mention of "spouse" before the entity [PERSON1]?\nA:	SPOUSE
	Context: "[TEXT]"\nQ: Is there any mention of "spouse" before the entity [PERSON2]?\nA:	SPOUSE
	Context: "[TEXT]"\nQ: Do [PERSON1] and [PERSON2] have the same last name?\nA:	SPOUSE
	Context: "[TEXT]"\nQ: Did [PERSON1] and [PERSON2] get married?\nA:	SPOUSE
	Context: "[TEXT]"\nQ: Are [PERSON1] and [PERSON2] family members?\nA:	NOT_SPOUSE
	Context: "[TEXT]"\nQ: Is [PERSON1] said to be a family member?\nA:	NOT_SPOUSE
	Context: "[TEXT]"\nQ: Is [PERSON2] said to be a family member?\nA:	NOT_SPOUSE
	Context: "[TEXT]"\nQ: Are [PERSON1] and [PERSON2] dating?\nA:	NOT_SPOUSE
	Context: "[TEXT]"\nQ: Are [PERSON1] and [PERSON2] co-workers?\nA:	NOT_SPOUSE
	Q: Are [PERSON1] and [PERSON2] married?\nA:	SPOUSE

Table 13. Spouse labeling function prompts with class labels NOT_SPOUSE = 0, SPOUSE = 1. A label map transforms text completions to class labels, where "yes" emits the value denoted in the label column and "no" emits ABSTAIN.

Calibrated Labeling Function Prompts (YouTube)

Uncalibrated Labeling Function Prompts (YouTube)

Fig. 7. YouTube prompted labeling function accuracy vs. coverage scatter plots. The top figure is calibrated using contextual calibration and the bottom is uncalibrated. Colors correspond to the language models used for labeling and marker style indicates class label.

Calibrated Labeling Function Prompts (Spouse)

Uncalibrated Labeling Function Prompts (Spouse)

Fig. 8. Spouse prompted labeling function accuracy vs. coverage scatter plots. The top figure is calibrated using contextual calibration and the bottom is uncalibrated. Colors correspond to the language models used for labeling and marker style indicates class label.
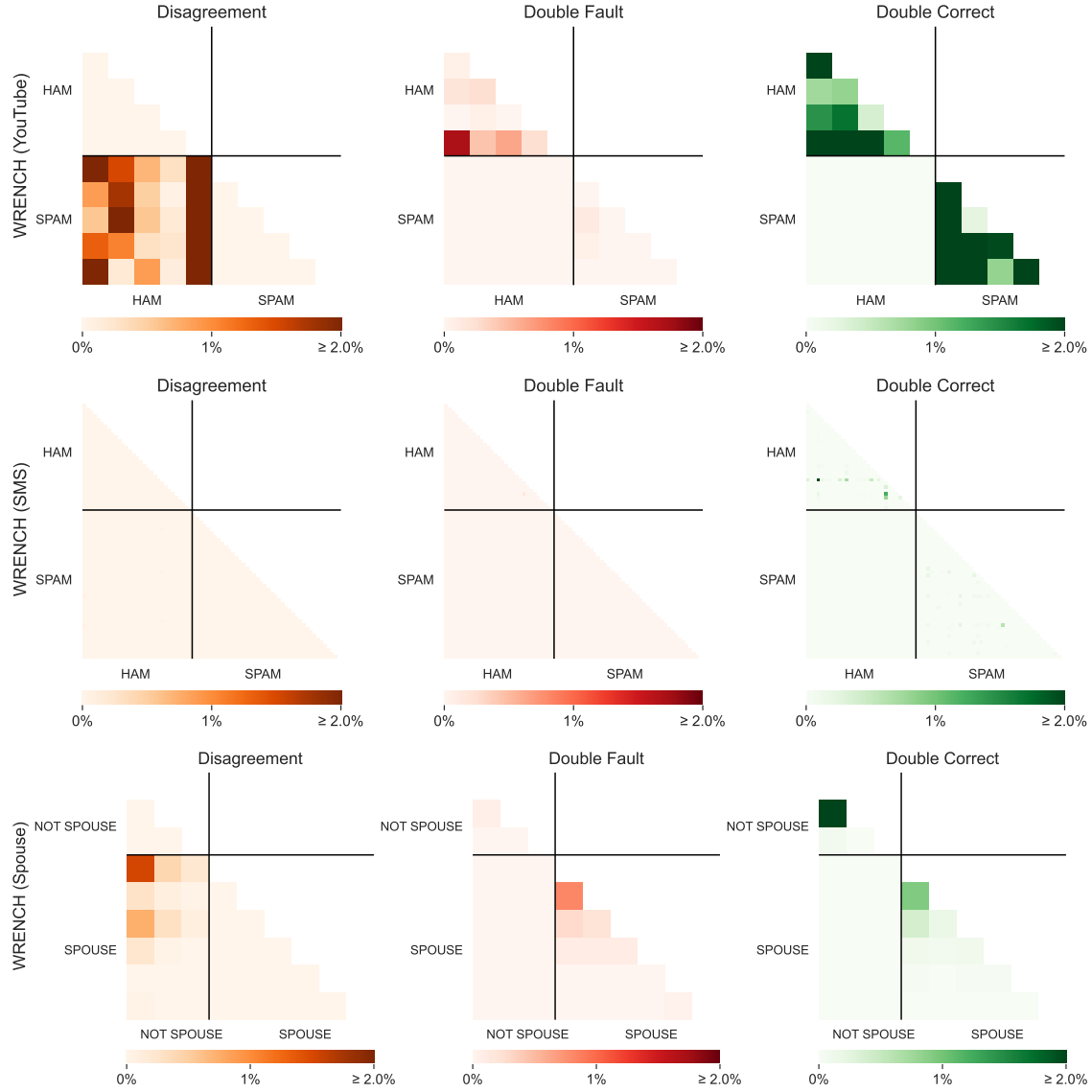
Fig. 9. Diversity measures for the WRENCH Benchmark labeling function set. Here rules have very low coverage (i.e., rules typically vote on less that 2% of the training set) but have high precision. SMS and Spouse have very low overall disagreement levels. YouTube has higher disagreement, but only limited cases where both labeling functions make correlated errors (double fault).
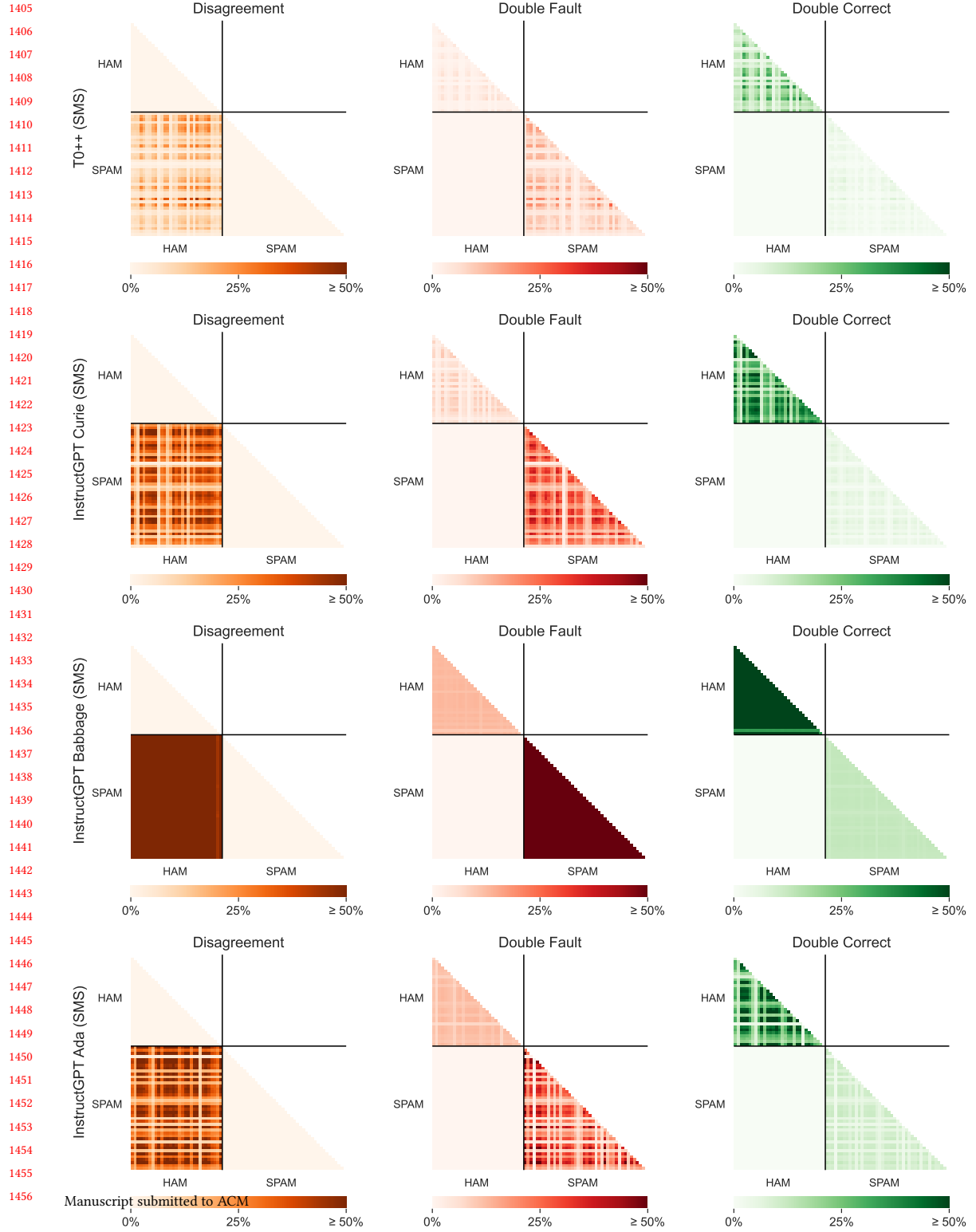
Fig. 10. SMS prompted labeling function diversity measures. Color intensity represents the percentage of training examples labeled by a pair of prompts.
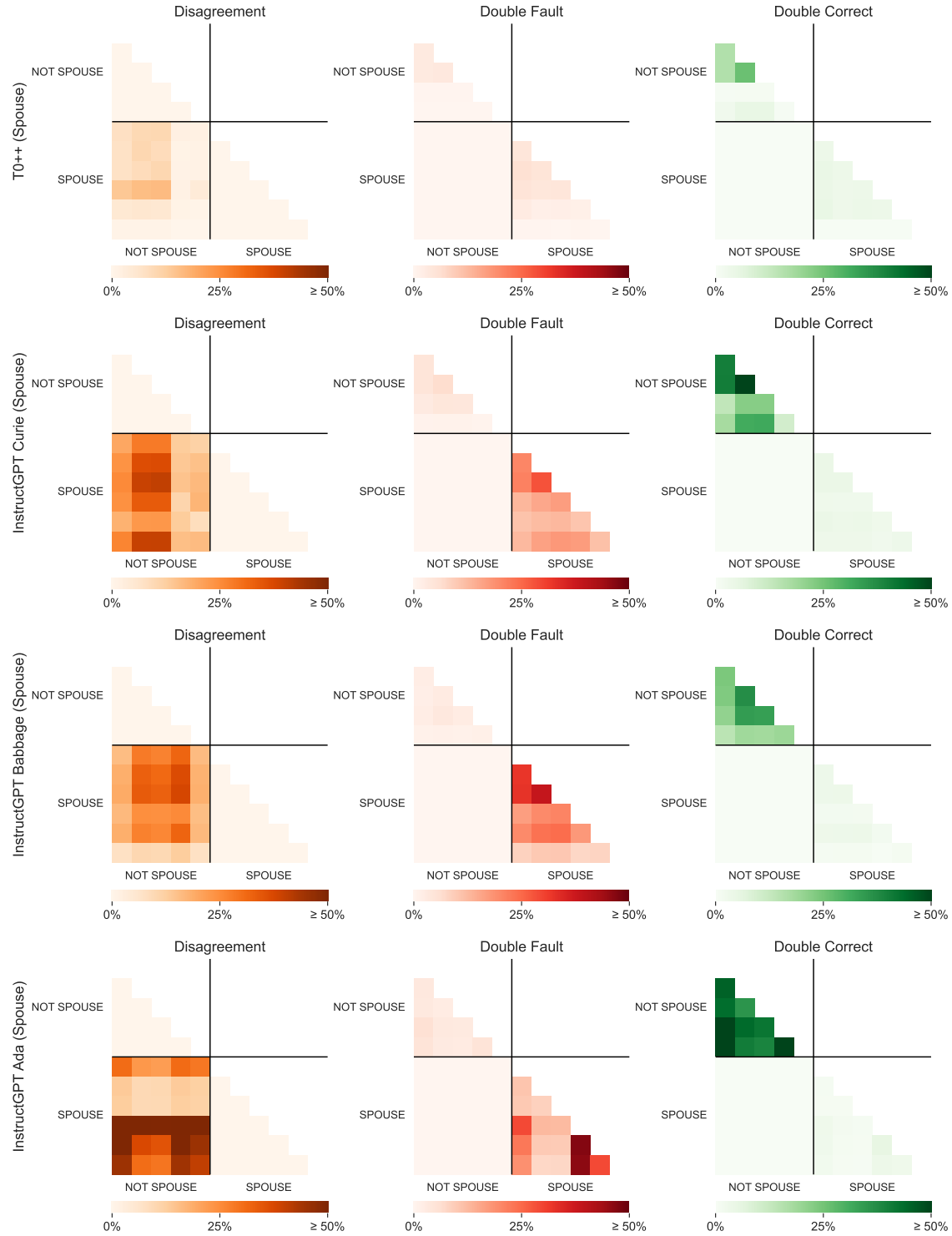
Fig. 11. Spouse prompted labeling function diversity measures. Color intensity represents the percentage of training examples labeled by a pair of prompts.
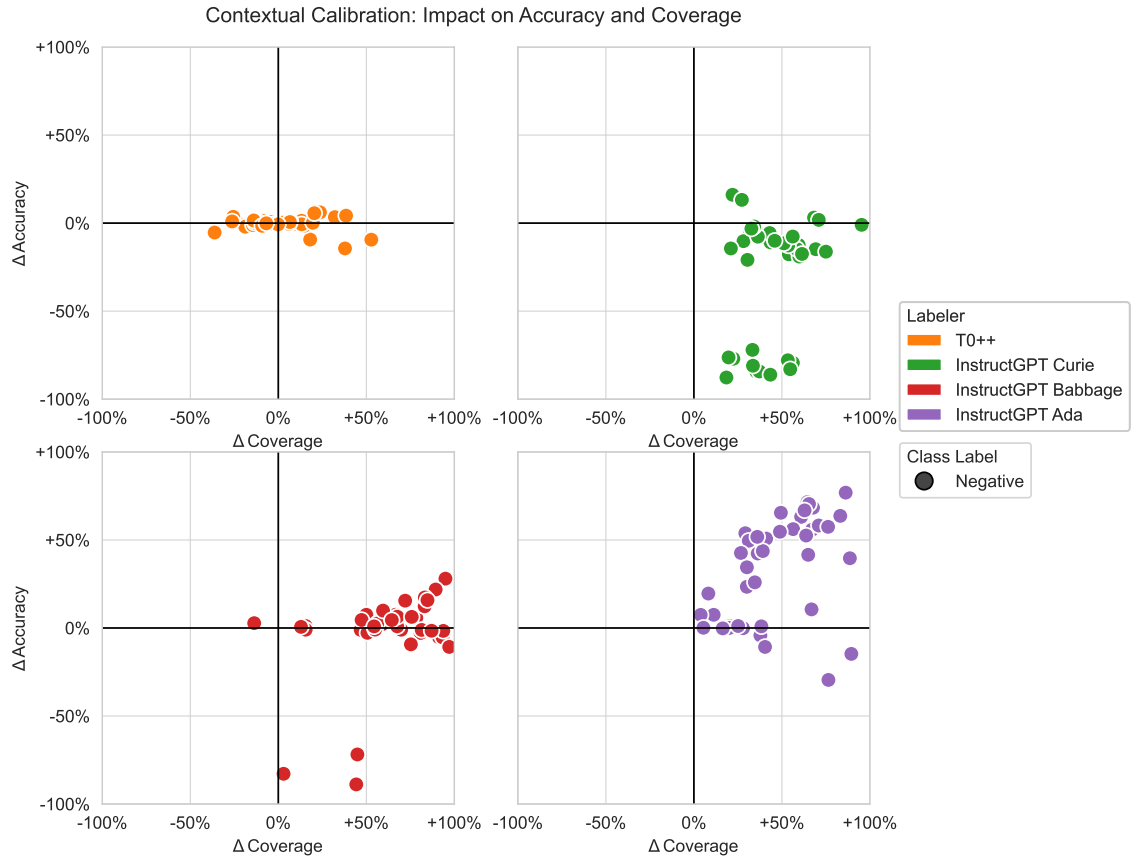
Fig. 12. Accuracy and coverage changes as a result of contextual calibration, broken down by the negative class label.
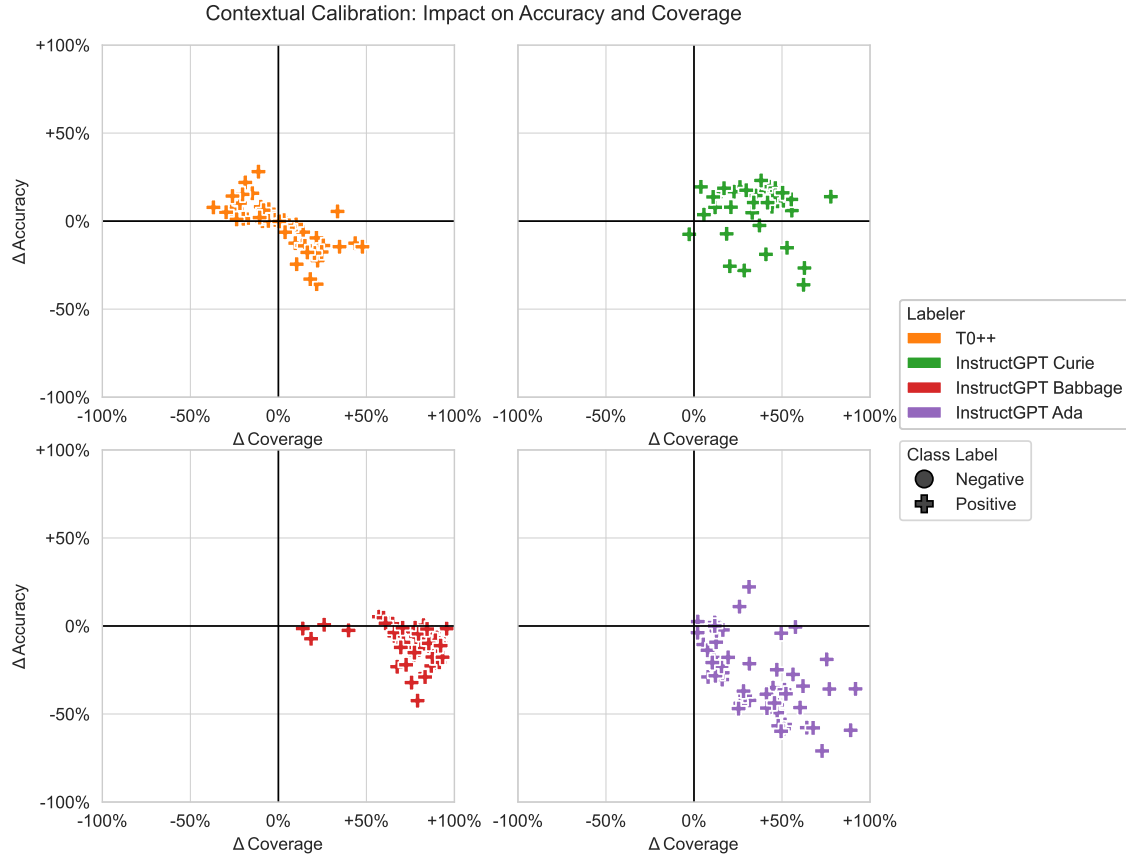
Fig. 13. Accuracy and coverage changes as a result of contextual calibration, broken down by the positive class label.