A Comparative Analysis between Human-in-the-loop Systems and Large Language Models for Pattern Extraction Tasks

Maeda F. Hanafi, Yannis Katsis, Ishan Jindal, Lucian Popa
IBM Research
{maeda.hanafi, yannis.katsis, ishan.jindal}@ibm.com, lpopa@us.ibm.com

Abstract

Building a natural language processing (NLP) model can be challenging for end-users such as analysts, journalists, investigators, etc., especially given that they will likely apply existing tools *out of the box*. In this article, we take a closer look at how two complementary approaches, a state-of-the-art human-in-the-loop (HITL) tool and a generative language model (GPT-3) perform out of the box, that is, without fine-tuning. Concretely, we compare these approaches when end-users with little technical background are given pattern extraction tasks from text. We discover that the HITL tool performs with higher precision, while GPT-3 requires some level of engineering in its input prompts as well as post-processing on its output before it can achieve comparable results. Future work in this space should look further into the advantages and disadvantages of the two approaches, HITL and generative language model, as well as into ways to optimally combine them.

1 Introduction

Creating custom AI models for natural language processing (NLP) tasks is not an easy feat: it typically involves labeling large datasets, selecting appropriate ML architectures/models, and training them. To lower the barrier of entry in NLP model creation, the scientific and industrial community has proposed human-in-the-loop (HITL) systems (Wu et al., 2022; Hanafi et al., 2017; Monarch, 2021). While they differ in the NLP tasks they target (e.g., classification, extraction, question answering) and the techniques they employ (e.g., active learning, custom algorithms), their operation from the perspective of the end user is similar: The user labels a few examples, which are then used by the system to build a first version of the model and then to ask back the user for additional targeted feedback. This feedback is in turn used to iteratively refine the model and continue the loop by asking for further feedback. By asking for targeted feedback at each iteration, the goal is to lower the effort required to build a model, enabling fast convergence to a performant model while using a small number of labeled examples.

While such tools are gaining popularity, the NLP community has also recently proposed several pretrained generative language models, such as GPT-3. The largest versions of such models have shown incredible few-shot performance on several tasks (Brown et al., 2020). Thus the question arises: Given the out-of-the-box performance of such models, is it possible to feed them with a few examples and get similar performance to that of more complex HITL systems?

In this paper, we answer this question for the task of text pattern extraction. In this task, the goal is to extract text instances that follow a similar pattern. Examples include extracting crime incidents from crime reports, e.g. 4,556,123 incidents or 5,193,927 incidents, or revenue from financial press releases, e.g. revenue was $5.5 billion or revenue of $1.3 million.

To answer the question, we compare *Pattern Induction* (Hanafi et al., 2022), an HITL tool tailored to pattern extraction tasks, against *GPT-3* (Brown et al., 2020). We evaluate the two approaches based on how an *end-user*, who does not necessarily have a technical background, would use these tools to accomplish an NLP task, in this case pattern extraction. Unlike NLP experts, end-users typically use tools out-of-the-box, that is, without writing code and without fine-tuning parameters or modifying and re-training the layers of the AI model. Our preliminary results show that the use of simple techniques to prompt GPT-3 does not yet lead to the same performance as that of the tailored HITL tool. In this work, we describe these results and present further research directions on the relationship of models such as GPT-3 to HITL tools.

The following summarizes the contributions of this paper:

- A preliminary comparative analysis between an HITL system and large language model (GPT-3) in an information extraction (IE) context and a discussion of the pros and cons of the two approaches.
- A description of different techniques to leverage GPT-3 in this setting and evaluation of their impact on model performance.
- A discussion of interesting future directions that emanate from this preliminary study.

We start with literature review in Section 2, provide empirical study in Section 3, discuss each approach pros and cons in Section 4, and conclude in Section 5.

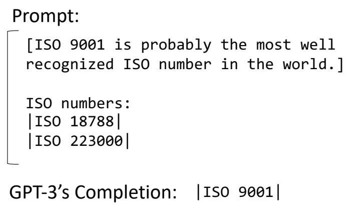
2 Related Works

In this work, we focus on few-shot learning systems (FSL) (Lake et al., 2016; Wang et al., 2020), which can learn from a handful of examples. This paradigm appeals to non-technical users given the small number of required examples to fine-tune a system, thus removing the need to label and maintain large labeled training datasets.

2.1 Human-In-The-Loop (HITL) Systems

HITL systems utilize human-interaction as opposed to systems that are fully automated, e.g. distant supervision, unsupervised, or semi-supervised methods (Ratner et al., 2017; Rühling Cachay et al., 2021). Automated methods leverage external data sources or seed inputs or rely on patterns or structures present in the dataset. They prove to be quite popular due to their ability to cover cases that a human would otherwise overlook. On the other hand, HITL systems integrate a human component in the relevant target task, and in this case an HITL information extraction would extract relevant texts with a human more involved in the process compared to a fully automated method (Monarch, 2021; Wu et al., 2022).

One such HITL tool for IE is Pattern Induction (Hanafi et al., 2017), and given an IE task, a user would do the following: (1) Highlights a minimum of two examples of text to extract, (2) The system learns a rule-based model, where each rule captures all of the examples, (3) The user provides "Yes" and "No" feedbacks to candidate extractions, which in turn refines the rules, i.e. saying "No" to revenue of 2013 would inform Pattern Induction to filter out rules capturing such extractions, (4) The user is either satisfied with the set of extractions or further refines the rule-based model with additional highlights of positive examples on the document. Additional examples and feedbacks further refines the learned rule-based model. Pattern Induction showcases an HITL system that provides the human interactions the IE tasks needs to ensure accuracy in the underlying learned model.



Prompt:

```
[ISO 9001 is probably the most well
recognized ISO number in the world.]

ISO numbers:
|ISO 18788|
|ISO 223000|
```

GPT-3's Completion: `|ISO 9001|`

Figure 1: A naive way of prompting GPT-3 to complete the text pattern extraction task.

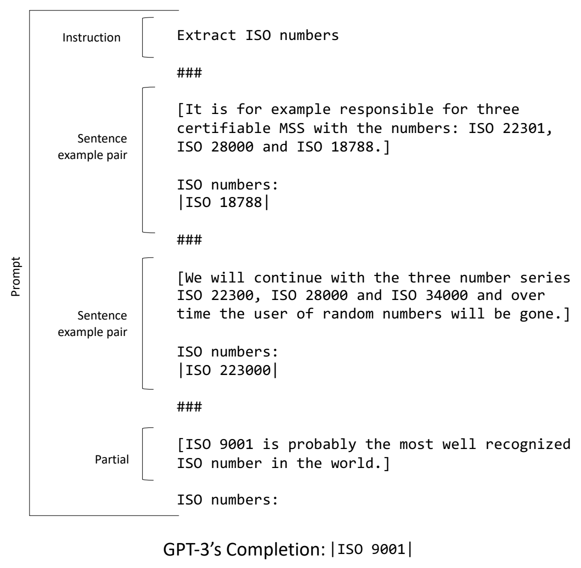
2.2 Generative Language Models

Unlike HITL models, large language models (LLMs), e.g. BERT (Devlin et al., 2019) and GPT (Brown et al., 2020), are pre-trained with large unlabeled datasets, which enables the LLMs to understand contextual information in the input text. While both BERT and GPT are transformer-based models, enabling their few-shot abilities is done in different ways; templates with masking are built to take advantage of BERT while one has to prompt GPT with text, such that it generates relevant text (Wang et al., 2021). GPT falls under the category of generative language models, and GPT-3 has a larger number of parameters (175 billion parameters in its largest *davinci* version), making it one of the most powerful generative language models compared to its predecessors. Performing IE with GPT often entails constructing and engineering well-structured *prompts* (Schick and Schütze, 2021a). Prompts contain examples of extractions where a desired extracted text is paired with the sentence where it occurs.

3 Experiments

Since HITL models help end-users with little to no technical background perform IE tasks, we want to understand how they compare against popular generative language models out of the box.

Since Pattern Induction is an HITL system, its output depends on the sequence of user actions. To



```
Extract ISO numbers

###

[It is for example responsible for three
certifiable MSS with the numbers: ISO 22301,
ISO 28000 and ISO 18788.]

ISO numbers:
|ISO 18788|

###

[We will continue with the three number series
ISO 22300, ISO 28000 and ISO 34000 and over
time the user of random numbers will be gone.]

ISO numbers:
|ISO 223000|

###

[ISO 9001 is probably the most well recognized
ISO number in the world.]

ISO numbers:
```

GPT-3's Completion: `|ISO 9001|`

Figure 2: Structured Prompts containing an instruction, sentence example pairs, and a partial of the document to extract texts from. For some sentence example pairs, not all of the desired extractions will appear in the example extractions, e.g. ISO 22301 , ISO 28000 (to mimic how an end-user might prompt GPT-3).

evaluate the system at scale, we perform user simulation in lieu of a real user that interacts directly with the system. As an added benefit, the user simulation also automates the manual aspects of evaluating the underlying model, namely recording the performance of the underlying models.

We recreate the use cases and user simulation framework described in (Hanafi et al., 2022). The use cases are shown in Table 1. The 7 pattern extraction use cases are based on 5 collections of unstructured texts documents covering a variety of topics, such as food recipes (Recipes), reports on the financial performance of a particular company (Financial Press Releases), and crime statistics (FBI Press Releases). Each dataset contains no more than 10 documents, where each document varies in length. Each use case has no more than 100 extractions in the groundtruth. We then compare how a generative language model, specifically GPT-3, performs on the same set of use cases.

The use cases supported in Pattern Induction can be described as syntactic text patterns. How well does a generative language model extract text patterns such as the ones in Pattern Induction?

3.1 User Simulation of an HITL System

Evaluating Pattern Induction for the IE use cases involves running a user simulation, constructed by the core *user actions* it supports: (a) highlighting examples and (b) providing "Yes" or "No" feedbacks to *candidate extractions* generated by Pattern Induction. Specifically, a user simulation, r_i, provides 2 seed examples, $E = \{e_1, e_2\}$, and provides answers to all of Pattern Induction's candidate extractions, $F = \{f_1, ..., f_n\}$. The user simulation's answers to the candidate extractions are determined by the groundtruth, which is provided as input to the user simulation framework. We ran the user simulation 100 times, $R = \{r_1, ..., r_{100}\}$. In each run r_i, $e \in E$ was randomly selected from the ground truth. At the end of each run, the precision (P), recall (R), and F1 (F1) score of the model learned by Pattern Induction was evaluated on the same set of ground truth. The final P/R/F1 of Pattern Induction for each use case was computed as the average value of the respective metric over all 100 runs. The resulting performance of Pattern Induction for each of the seven use cases in Table 1, is shown in Figure 3 (see HITL line).

3.2 Prompting GPT-3 for IE with Example Extractions

We performed a similar set of experiments using GPT-3, where we prompted GPT-3 using OpenAI's completion API, which is recommended for entity extraction [1].

The prompts were constructed with the same 2 seed examples, E, from the user simulation in Pattern Induction. While Pattern Induction does not require any sort of prompting, constructing a prompt requires some level of engineering in order for GPT-3 to understand the intent of our extraction task. GPT-3 is sensitive to the context and the structure of the prompt's text (Shin et al., 2020; Gao et al., 2021; Jiang et al., 2020; Schick and Schütze, 2021b). Moreover, GPT-3 has a limit of about 4,000 tokens for its most powerful model, the *davinci*. We thus split the documents d in the dataset D into *partials*, $p \subset d, d \in D$.

We experimented with the following prompt structures:

- Baseline, Naive Prompts, $GPT3_E$: We experimented with a rather naive approach to the prompt's structure, where the example extractions E are appended to the document's partial p (see Figure 1). This prompt structure aims to mimic an end-user's intuition when using Pattern Induction: directly highlighting

[1]https://beta.openai.com/docs/api-reference/completions

ID	Use Cases	Dataset	Representative Examples
U1	Covid Cases by Country	Disease Fatality Reports	Spain (239 932) , Malta (620)
U2	Crime Incident Count	FBI Press Releases	4,927,535 incidents , 6,572,870 incidents
U3	Crime Percentages by Type	FBI Press Releases	62.9 percent involved crimes against property , 24.6% were crimes against persons
U4	Cups Multiple Forms	Recipes	2 cup , 1/4 cup , 1 1/2 cup
U5	Earnings Time Period Multiple Forms	Financial Press Releases	2014 First-Quarter , fourth-quarter of 2013
U6	ISO Numbers	ISO Number Articles	ISO 639 , ISO 22300
U7	ISO Numbers Multiple Forms	ISO Number Articles	ISO 639 , TC 292 , ISO/IEC 40180 , ISO/TC 28

Table 1: Summary of Text Pattern Extraction Tasks in the Experiments.

or specifying example extractions in the document itself.

- Structured Prompts, $GPT3_{C(E)}$: We experimented with a more *structured* format of prompting. For each example e, we paired it with a sentence $s \in D$ that e appears in. When multiple examples, e.g. e_i and e_j, appear in the same sentence s, we combined the multiple examples with the same s in the prompt. Any additional desired extractions in s that are not in the example set E were not indicated in the prompt (to mimic how an end-user might prompt GPT-3). Moreover, to provide GPT-3 with a better contextual understanding of the pattern extraction task, we prepended each prompt with an instruction describing the task, such as "Extract crime percentages by type". We then added the sentence example pairs and appended the partial p that GPT-3 must extract text patterns from (see Figure 2). Note that in baseline prompting, an example e may not appear in the partial p, but in structured prompting, each e will appear in its paired sentence s.

The above methods of prompting GPT-3 may not be enough to take advantage of the contextual understanding capabilities of GPT-3. But the focus of our study is to use methods similar to how an end-user might prompt GPT-3, assuming the end-user does not have much technical background.

Given a prompt, GPT-3 returns a string containing a *completion*. The completion is usually delimited by the characters it learns from the prompt, e.g. "|". To calculate the precision and recall scores of what GPT-3 extracts, we split the completion text according to the delimiters.

Given the 100 runs along with each run's associated seed examples E from the user simulation on Pattern Induction, we constructed a prompt with each E as described above and fed the prompts into GPT-3. The results of prompting GPT-3 for text pattern extraction are shown in Figure 3, along with the results for HITL. Our preliminary investigation seems to suggest that GPT-3 is not a right choice. However, we believe that extensive experiments to find more suitable prompts will need to be conducted.

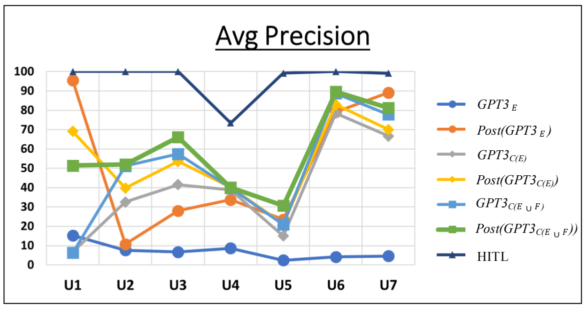
The results for $GPT3_E$ and $GPT3_{C(E)}$ have lower precision scores compared to Pattern Induction's precision scores. We observed that low-precision runs are due to the fact that GPT-3's extracted text is not always a part of the document dataset. We thus added a post-processing step over a set of runs, R:

- $Post(R)$: removes outputs that are not part of the document. So $Post(GPT3_E)$ indicates that the post-processing step is applied to the set of runs in $GPT3_E$.

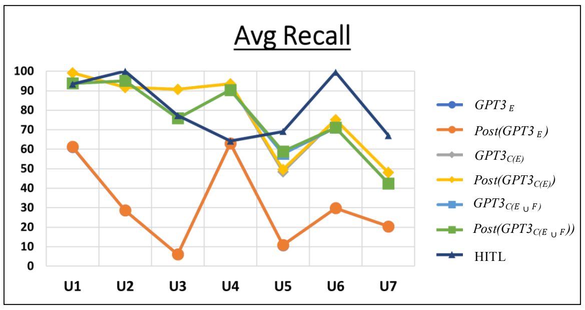
3.2.1 Insights: Better-Structured Prompts Improve Recall

As we move from naive prompting, $GPT3_E$, to structured prompting, $GPT3_{C(E)}$, the recall scores improve. Constructing well-structured prompts poses a limitation when using GPT, whereas Pattern Induction has no such requirement for an end-user.

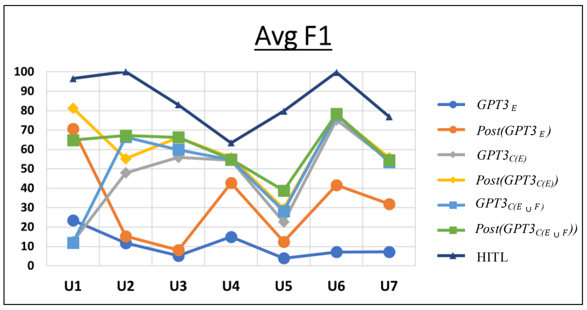
The post-processing step improved the precision scores, but not the recall scores, as the lines in the chart (see Figure 3b) of the post-processing step's



(a) Precision



(b) Recall



(c) F1

Figure 3: Experimental Results: HITL refers to Pattern Induction's user simulation results.

recall scores do not deviate much from the lines of the corresponding recall scores of raw output extractions.

The recall scores for U1, U3, and U4 (see task descriptions in Table 1) were higher with $GPT3_{C(E)}$ than in Pattern Induction (HITL). These tasks also happen to have *variations* in their extractions. For instance U1 requires the extraction of a country, U3 requires extraction regarding different crime types, e.g. "crimes against property", "crimes against persons", and U4 requires extractions of measuring cups of both integer and fractional types. GPT-3 seems to be able to understand that some of the words refers to countries, crime types, or integer and fractional quantities, where all the expected extractions may not necessarily abide to some syntactic pattern. In contrast, for the other tasks U2, U5, U6, U7 (see Table 1), the extractions follow

Substring of the prompt with F_N

```
###

[Part 1]

ISO numbers:
|###|

###
```

Figure 4: Substring of a prompt that demonstrates Part 1 should not be extracted, since Part 1 was rejected by the user simulation in Pattern Induction..

strict pattern extractions, e.g. in U2, the literal "incidents" constantly appears at the end of a 6 digit integer for all expected extractions, and in U5, the literal "quarter" and a 4 digit year appears in all extractions.

3.3 Prompting GPT-3 for IE with Additional Example Extractions

One may argue that the above method for prompting GPT-3 did not include the same set of inputs we provided to Pattern Induction. Namely, in Pattern Induction, the evaluation method also allows the user to provide feedback to candidate extractions, F, to further refine Pattern Induction's model, but no such additional inputs were added to the prompt for GPT-3. In this round, we reran the experiments with additional extraction examples:

- Prompts with Additional Extraction Examples, $GPT3_{C(E \cup F)}$: The structured prompts include the same set of candidate extractions F in addition to the set of highlighted seed examples E from the Pattern Induction user simulation.

We denote the set of extractions the user simulation accepted, i.e. answered "Yes" to fourth-quarter of 2012 , as $F_Y \subseteq F$, and the set of extractions the user simulation rejected, i.e. answered "No" to revenue of 2013 , as $F_N \subseteq F$. We integrated accepted extractions to the prompt in the same structured format as the seed examples (see Figure 2). Integrating rejected extractions to the prompt was done in a slightly different manner: for each $n \in F_N$, we append an end of sequence token in between the extraction delimiters (see Figure 4). While we have explored different ways of adding rejected extractions to the prompt such as using empty strings in between extraction delimiters, we found better performance with the end of sequence token.

3.3.1 Insights: Additional Examples in GPT-3 Improve Recall

The results of prompting GPT-3 with additional extraction examples are shown in Figure 3.

The precision scores are lower for $Post(GPT3_{C(E\cup F)})$ in comparison to HITL, with the exception of U6 and U7. The recall scores for both raw $GPT3_{C(E\cup F)}$ and post-processed $Post(GPT3_{C(E\cup F)})$ results are on par with HITL (U1, U2, U3), and they both even beat HITL in U4. Oddly enough $GPT3_{C(E)}$'s recall scores are very similar to $GPT3_{C(E\cup F)}$, except in U3 where the additional examples counter-intuitively dropped the recall scores by more than 10 points.

We observed that GPT-3 outputs creative texts regardless of whether there are additional examples or not ($GPT3_{C(E)}$ and $GPT3_{C(E\cup F)}$). Creative text outputs include "The race was unknown for 15.3 percent of reported known offenders." when given the U3 task. In addition to creative outputs, GPT-3, prompted with and without additional examples, also generates texts that are somewhat similar to the context in the examples but not exactly found in the text. In the U3 task, incorrect generated outputs would often contain percentage phrases that refer to statistics of other topics aside from crimes such as gender or race, e.g. "0.6 percent were American Indian or Alaska Native".

While the current set of experiments are only seeded with 2 examples, future experiments should look into the impact of increasing seed examples.

4 Discussion

Our experiments show that the HITL method for IE results in higher precision while in the large generative model, given a structured prompting and post-processing step, GPT-3 gives higher recall. GPT-3 is able to contextualize the prompts and learns a more general model.

Yet, the downside of GPT-3 for IE is that in of itself does not perform the IE tasks. To get comparable results to the HITL model, we had to (1) engineer and design the structure of the prompts to leverage GPT-3's powerful language abilities and (2) post-process the string output from GPT-3.

Additionally, the HITL model (1) elicits targeted user feedback and (2) allows for an iterative approach to building the underlying rule-based model. These two aspects are not found in GPT-3.

In light of these results, how do we then combine the advantages of each approach, human-in-the-loop and traditional AI models? How do we leverage the human-machine interactions to help increase both metrics? We leave these questions for future work.

5 Conclusions & Future Work

In this short work, we evaluate information extraction tasks on a human-in-the-loop, rule-based approach against a state-of-the-art generative language model, GPT-3. Our results show that the rule-based model outperforms GPT-3, when used out-of-the-box similar to how an end-user might use it to perform an NLP task. There are potentially better ways of constructing the prompts for GPT-3, but we wanted to better understand the performance of both the rule-based models and generative language model out-of-the-box.

Future work in this area should look into enabling end-users with little to no technical expertise to accurately and quickly build NLP models. For instance, one possibility is to go beyond user simulations and study how actual end-users create prompts. Another possibility is to leverage previous works in automatically generating prompts and then designing ways to elicit user input to craft better prompts (Shin et al., 2020; Jiang et al., 2020). An important future direction is to identify disadvantages and advantages of each approach, traditional large language models and rule-based models, and look into how combining such approaches would better enable end-users in NLP.

References

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics* *and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3816–3830, Online. Association for Computational Linguistics.

Maeda F. Hanafi, Azza Abouzied, Laura Chiticariu, and Yunyao Li. 2017. Seer: Auto-generating information extraction rules from user-specified examples. CHI '17, page 6672–6682, New York, NY, USA. Association for Computing Machinery.

Maeda F. Hanafi, Yannis Katsis, Martín Santillán Cooper, and Yunyao Li. 2022. A simulation-based evaluation framework for interactive ai systems and its application. 36:12658–12664.

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*, 8:423–438.

Brenden M. Lake, Tomer David Ullman, Joshua B. Tenenbaum, and Samuel J. Gershman. 2016. Building machines that learn and think like people. *Behavioral and Brain Sciences*, 40.

Robert Munro Monarch. 2021. *Human-in-the-Loop Machine Learning: Active learning and annotation for human-centered AI*. Simon and Schuster.

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

Salva Rühling Cachay, Benedikt Boecking, and Artur Dubrawski. 2021. End-to-end weak supervision. In *Advances in Neural Information Processing Systems*, volume 34, pages 1845–1857. Curran Associates, Inc.

Timo Schick and Hinrich Schütze. 2021a. Exploiting cloze-questions for few-shot text classification and natural language inference. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 255–269, Online. Association for Computational Linguistics.

Timo Schick and Hinrich Schütze. 2021b. It's not just size that matters: Small language models are also few-shot learners. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2339–2352, Online. Association for Computational Linguistics.

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 *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4222–4235, Online. Association for Computational Linguistics.

Chenguang Wang, Xiao Liu, Zui Chen, Haoyun Hong, Jie Tang, and Dawn Song. 2021. Zero-shot information extraction as a unified text-to-triple translation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1225–1238, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Yaqing Wang, Quanming Yao, James T. Kwok, and Lionel M. Ni. 2020. Generalizing from a few examples: A survey on few-shot learning. *ACM Comput. Surv.*, 53(3).

Xingjiao Wu, Luwei Xiao, Yixuan Sun, Junhang Zhang, Tianlong Ma, and Liang He. 2022. A survey of human-in-the-loop for machine learning. *Future Generation Computer Systems*.

A Appendix

Table 2 shows the exact numbers for the average precision, recall, and F1 scores across the different experiments.

ID	Tool	Avg Precision	Avg Recall	Avg F1
U1	$GPT3_{\mathrm{E}}$	15.3	61.1	23.5
	$Post(GPT3_E)$	95.4	61.3	70.7
	$GPT3_{C(E)}$	6.7	99.2	12.5
	$Post(GPT3_{C(E)})$	69.1	**99.2**	81.2
	$GPT3_{C(E \cup F)}$	6.3	93.8	11.9
	$Post(GPT3_{C(E \cup F)})$	51.4	93.8	64.8
	HITL	**100.0**	93.4	**96.5**
U2	$GPT3_E$	7.6	28.6	11.7
	$Post(GPT3_E)$	10.7	28.6	15.3
	$GPT3_{C(E)}$	32.6	91.7	47.9
	$Post(GPT3_{C(E)})$	39.8	91.7	55.2
	$GPT3_{C(E \cup F)}$	51.2	95.1	66.4
	$Post(GPT3_{C(E \cup F)})$	52.0	95.1	67.1
	HITL	**100.0**	**100.0**	**100.0**
U3	$GPT3_E$	6.7	6.1	5.2
	$Post(GPT3_E)$	28.0	6.1	8.1
	$GPT3_{C(E)}$	41.5	90.7	55.9
	$Post(GPT3_{C(E)})$	53.5	**90.7**	66.1
	$GPT3_{C(E \cup F)}$	57.4	75.9	59.8
	$Post(GPT3_{C(E \cup F)})$	66.1	75.9	66.2
	HITL	**100.0**	77.3	**83.0**
U4	$GPT3_E$	8.6	63.0	14.9
	$Post(GPT3_E)$	33.7	62.9	42.7
	$GPT3_{C(E)}$	38.8	93.5	54.5
	$Post(GPT3_{C(E)})$	40.1	**93.5**	55.7
	$GPT3_{C(E \cup F)}$	39.8	90.4	54.6
	$Post(GPT3_{C(E \cup F)})$	40.1	90.4	54.9
	HITL	**73.4**	64.2	**63.4**
U5	$GPT3_E$	2.4	10.8	3.9
	$Post(GPT3_E)$	23.5	10.9	12.4
	$GPT3_{C(E)}$	15.0	48.4	22.6
	$Post(GPT3_{C(E)})$	21.2	49.6	29.4
	$GPT3_{C(E \cup F)}$	20.9	57.6	28.2
	$Post(GPT3_{C(E \cup F)})$	30.6	58.9	38.8
	HITL	**99.2**	**69.1**	**79.7**
U6	$GPT3_E$	4.2	29.8	7.1
	$Post(GPT3_E)$	79.1	29.8	41.6
	$GPT3_{C(E)}$	78.5	75.1	75.2
	$Post(GPT3_{C(E)})$	82.9	75.1	77.6
	$GPT3_{C(E \cup F)}$	88.7	71.1	77.9
	$Post(GPT3_{C(E \cup F)})$	89.6	71.1	78.3
	HITL	**100.0**	**99.5**	**99.7**
U7	$GPT3_E$	4.6	20.4	7.2
	$Post(GPT3_E)$	89.1	20.4	31.9
	$GPT3_{C(E)}$	66.6	48.0	54.5
	$Post(GPT3_{C(E)})$	70.0	48.0	55.7
	$GPT3_{C(E \cup F)}$	77.8	42.5	53.5
	$Post(GPT3_{C(E \cup F)})$	81.2	42.5	54.5
	HITL	**99.1**	**67.0**	**76.9**

Table 2: Experimental Results: ID refers to the use case ID. HITL refers to Pattern Induction's user simulation results.