A Logic-based Microtasking Approach for LLMs and Human Processing

Tomoya Kanda, Hiroyoshi Ito, Nobutaka Suzuki, Atsuyuki Morishima

University of Tsukuba
Kasuga 1-2, Tsukuba, Ibaraki, Japan
{akei.tomoya@klis, Ito@slis, nsuzuki@slis, mori@slis}.tsukuba.ac.jp

Abstract

LLMs are rapidly increasing their power and serving as important information sources. Some existing work suggests that they show comparable performance with human workers for some tasks. However, human intervention is still required to obtain high-quality outputs. This demo intends to raise discussion on whether microtasking is an effective approach for the purpose. In our demo, instead of submitting a whole task to LLM, we divide the task into microtasks, to be generated by a predefined workflow on Crowd4U, a microtask-based human-in-the-loop data platform that deals with LLMs as AI workers to complete microtasks. Then, tasks are automatically assigned to human and AI workers, according to the logic-based workflow associated with microtasks. Through this demo, we show that the microtasking approach is a promising way to combine the power of LLM and humans, to improve the class of answerable queries, the quality of answers, and their explainability.

Introduction

Large Language Models (LLMs) are now serving as important information sources for many applications. The services based on the state-of-the-art LLMs, such as ChatGPT, answer correctly some factual questions (Qin et al. 2023).

Although LLMs are expected to be able to answer more complex questions correctly in the near future, the user would need to join to obtain the high-quality results, because LLMs often output logically inconsistent hallucinating answers (van Dis et al. 2023).

This demo will demonstrate that microtasking is a promising approach to address this problem. In our demo, instead of submitting a whole task to LLM, we divide the task into microtasks, to be generated by the predefined logic-based workflow on Crowd4U, which is a crowdsourcing platform that deals with LLMs as *AI workers* that perform microtasks along with humans(Morishima et al. 2012; Kobayashi, Wakabayashi, and Morishima 2021). We assume that the human workers are expert workers who do better works than LLMs. Therefore, they are complementary to each other in terms of scalabirity and speed (LLMs are better), and ability of quality check of other's results (humans are better).

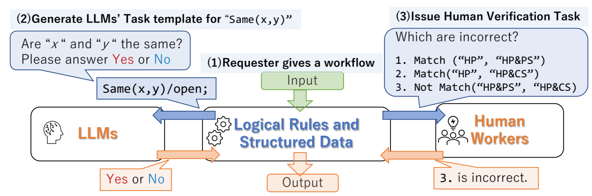

Figure 1: Workflow for the entity matching scenario. The processor uses a task template to generate LLMs tasks when encountering open predicates. After receiving the results, it checks consistency among other facts and generates human computation tasks to update prompts or directly fix the results if it finds inconsistency.

This demo shows that microtasking and techniques developed for human crowdsourcing can be extended and applied to take more advantages of both workers; microtasking can be used to logically decompose the queries to describe clear semantics, complete a task that consists of exact steps, and automatically assign tasks to humans for verifying outputs.

Focus of the Demo. Figure 1 overviews how our workflow controls task assignment (with an example scenario to be explained later) (1) The requester gives a workflow in a set of logical rules, in which some of the predicates are connected to microtasks. (2) When we need to evaluate predicates connected to tasks, the tasks are generated and assigned to LLMs to evaluate them[1]. Then, (3) we ask humans to perform verification tasks on the LLM results. Our question here in the demo is how to design *good interaction with humans in the verification tasks*. In this demo, we show a variety of interaction designs for human verification tasks to provide an opportunity to discuss the issue. The interaction designs have to be evaluated in terms of the number of interactions, the user load and the quality improvement. We focus on the verification tasks because efficient verification of AI results is the key to deal with AI workers; the result acts as a *trigger* to select plausible answers and iterate the evaluation of AI worker tasks with better prompting.

Comparison with other frameworks. Prompt engineering is a hot topic to derive appropriate answers from LLMs.

[1]The programmer can also choose the human-first assignment in the code. In addition, if AI workers claim that they cannot answer, the task is passed to human workers.

For example, Chain-of-Thought Prompting (Wei et al. 2023; Kojima et al. 2023) enables LLMs to improve their performance by generating step-by-step outputs. Still, since LLMs do not always generate correct outputs (Daull et al. 2023; Bang et al. 2023), we need verify LLMs' outputs. Some work (Zhao et al. 2023) addresses the problem by referencing external knowledge bases. Our framework adopts an explicit logic-based task decomposition, which allows us the *exact computation* based on the power of logic processors such as deduction and inconsistency check.

Symmetric aggregation of AI's and human's results (Yamashita et al. 2022), and how to combine novice and expert workers (Nguyen, Wallace, and Lease 2015) is discussed. However, they deal with multiple classification tasks only. In the demo, we address a general mechanism for asymmetric integration of human and AI worker results.

Finding repairs has been widely discussed in data cleaning or data integration settings (Arenas, Bertossi, and Chomicki 1999; Chiang and Miller 2011; Eiter, Fink, and Stepanova 2014; Lukasiewicz, Malizia, and Vaicenavicius 2019; Prokoshyna et al. 2015). Our setting is different from the settings in the ordinary data cleaning and integration settings, in that there are subjects (i.e., LLMs) to dynamically generate data on demand, and we can take advantage of crowdsourcing in finding repairs.

Scope and Limitations. In our demo, we assume that a variety of task templates (prompts) are already stored for predicates and will be updated in a simple way; We do not focus on how to design task templates with prompting techniques. There are many papers that address prompting and how to adopt the effective strategies for task updates/change will be interesting. Building a large database for task templates is out of the demo's scope too, although we assume that we can automatically generate prompts based on existing knowledge bases such as DBpedia.

Our demo scenarios were developed with GPT-3.5 (Ouyang et al. 2022). We may use a different version in the demo since LLMs are quickly evolving. Our framework, however, does not depend on any particular version.

Task Generation and Assignment in Logic-based Microtasking

Our demo will work on an extended version of Crowd4U (Morishima et al. 2012), a human-in-the-loop platform that has been used for 10 years for real-world projects. We first describe the extended language and architecture, then the design space for the human verification tasks to resolve inconsistency caused by the LLM results.

CyLog

CyLog (Morishima, Fukusumi, and Kitagawa 2016) describes data (i.e., tuples in relations) as *facts*, and queries as *rules*. The following is a fragment of a CyLog program extended for the human-in-the-loop LLM processing:

```
Movie("Harry Potter");
Movie("Harry Potter and Philosopher's Stone");
Movie("Harry Potter and the Chamber of Secrets");
Match(x,y):- Movie(x), Movie(y), Same(x,y)/open;
F :- Match(x,y), Match(y,z), [ not Match(x, z) ];
```

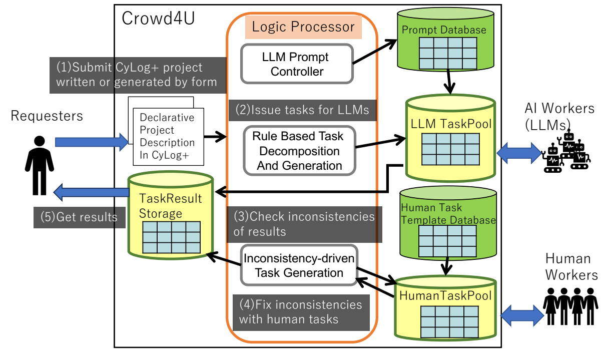

Figure 2: Crowd4U architecture for LLMs + Human workers

The first three lines describe three facts, each of which states a movie. The fourth line is a rule. Here, Cylog allows predicates (*e.g.,* `Same(x,y)`) to be *open*, meaning that the decision on whether a fact holds is crowdsourced[2]. Intuitively, the rule states that, for any two movies `x` and `y`, a microtask to ask whether they are the same or not is generated (open predicates are associated with task templates (Figure 1 (top left)), and if the worker says they are the same movie, we derive a fact (tuple) `Match(x,y)`. CyLog has a built-in reward system to give semantics for open predicates based on the game theory. Detail is given in (Morishima, Fukusumi, and Kitagawa 2016) and omitted.

The last line starting with "`F:-`" is an *inconsistency rule* we introduced for the demo. The line states that for any pair `Match(x,y)` and `Match(y,z)`, not having `Match(x,z)` leads to a contradiction under the transitivity law. The user can define global or local logical rules in CyLog that need to be satisfied. For example, the transitivity law must be satisfied for any equivalence relationship; another example is that the number of answers to an enumeration query (e.g., "What are universities in Japan?") must match with the answer to the question "What is the total number of universities in Japan?".

Crowd4U Architecture

Figure 2 illustrates the architecture. The logic-mediated workflow for dispatching tasks to workers is controlled by the logic processor. (1) Once the requester submits a CyLog workflow, (2) the processor generates tasks during the evaluation of logic rules; when it encounters an open predicate, it first assigns tasks to AI workers and (3) then generates verification tasks according to the task generation policy. (4) When there are inconsistency rules in the program, and the processor finds inconsistency among facts, new tasks to address the inconsistency are generated and assigned to human workers, which we discuss next. When some inconsistency is found, it generates and issues human computation tasks to update prompts for re-evaluation or fix the inconsistency.

[2]Unbounded variables (e.g., `Parent(X,Y)/open` in Section) are also allowed.

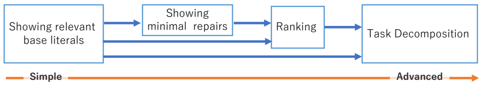

Figure 3: Design Space for Human Verification Tasks. The most basic one is just to show relevant base literals and ask humans whether each is correct or not. A variety of advanced designs can be obtained by combining several strategies.

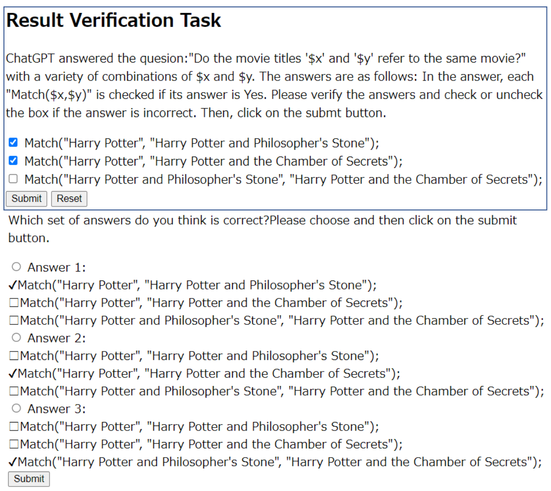

Figure 4: Human Tasks for Inconsistency Resolution. The simplest task is inside the box (cf. left side of Figure 3). For a particular class of inconsistency, Crowd4U can enumerate minimal repairs to resolve the inconsistency (bottom).

Design Space for Human Verification Tasks

According to the user-specified policy (every time or when some inconsistency among LLM-generated facts was found), Crowd4U generates verification tasks to ask humans to give *repairs*. For example, we may have three facts that caused the inconsistency in the transitive law (See the code in Section and Figure 1). Then, human worker may state that we should remove the second successful match (*i.e.*, `Match("HP", "HP&CS")`) in the task.

There are a variety of design options for this (Figure 3). The simplest design is that the task shows the list of relevant base literals that led to the inconsistency (Figure 4 (inside the box)). It shows the two matches and one negative match and asks workers to update some of them to remove the inconsistency. The first option is to show the potential *repairs* to fix the inconsistency and the human worker chooses one of them. Since there are many potential repairs, we often compute *minimal* repairs (i.e., minimizing the number of changes) (Bohannon et al. 2005) (Figure 4 (bottom)). The second option is to *rank* the potential repairs or base literals and show the ranked list. AI workers may correctly answer queries in a particular class only. Then, we can learn the class (Kobayashi, Wakabayashi, and Morishima 2021) and use it to rank them. The third option is to *decompose* the task into smaller ones each of which shows a smaller number of base literals and repairs to exploit the parallelism of crowdsourcing. This is effective when we have a large number of relevant base literals and potential repairs.

Demonstration Scenarios

Logic-mediated Human Task Assignment. The first scenario is the entity matching of movie titles, which we explained in Section , to demonstrate how the logic-mediated LLM-human-in-the-loop workflow works. LLMs often match "Harry Potter" and any of "Harry Potter and the Philosopher's Stone" and "Harry Potter and the Chamber of Secrets," which introduces inconsistency in terms of transitivity law (Figure 4). When Crowd4U finds an inconsistency based on the registered rules, inconsistency resolution tasks will be generated and assigned to human workers.

Exact Computation with Local Concepts. LLMs are generally weak in both exact computation and computation with locally (privately) defined concepts. An example is to find every combination of national universities in Japan that is within the same prefecture and have no overlapped departments. Here, the definition of overlapped departments is explicitly given by the requester - for example, the name of one department is included in that of the other. Then, he writes the combination in the logic[3].

```
N-Univ(n,l,d*)/open;
Pair(n1,n2):- N-Univ(n1,l,d1*), N-Univ(n2,l, d2*),
     [forall d1' in d1, d2' in d2(non-overlap(d1', d2'));]
non-overlap(d1, d2):-
     [ true if not (contain(d1, d2) or contain(d2, d1))]
F :- #N-Univ(n), N-Univ(n*), [n!=count(n*)];
```

Structured Explanation Generation. We show that logic-based microtasking often supplies correct explanations, in a case LLMs return *incorrect* explanations without it. In this scenario, we use the query about asking blood relationship between two Japanese Shoguns[4] in an open setting where we have no stored data. LLMs may be able to answer the question whether the 15th Shogun, Yoshinobu Tokugawa (Wikipedia contributors 2023b), is a descendent of the 1st, Ieyasu Tokugawa (Wikipedia contributors 2023a), without microtasking. However, the blood relationship shown to explain its answer may not. Given the following set of rules in our rule base, the query `:-Ancestor("Ieyasu Tokugawa", "Yoshinobu Tokugawa")` recursively asks "Who will be the parent of Y" to ChatGPT on the family tree and returns an alternative and correct explanation.

```
Parent(x,y)/open;
Ancestor(x,z):-Ancestor(x,y),Parent(y,z);
:-Ancestor("Ieyasu Tokugawa", "Yoshinobu Tokugawa")
```

Acknowledgement

We are grateful to Kentaro Miyake, Keito Oishi and all other members of the Crowd4U team who have been contributing to the development of the system. This work was supported by JSPS KAKENHI Grant Number JP22H00508, JP22K17944, JST CREST Grant Number JPMJCR22M, Japan.

[3]For conciseness, we use abbreviations of variables: `n`, `l` and `d` for `name`, `location` and `depts`, respectively. We also use numbers for correlation names (e.g., `n2` for `name as n2`). List variables ends with `*`.

[4]The tree is very complex: https://en.wikipedia.org/w/index.php?title=Template:Tokugawa_family_tree&oldid=1050258568.

References

Arenas, M.; Bertossi, L. E.; and Chomicki, J. 1999. Consistent Query Answers in Inconsistent Databases. In Vianu, V.; and Papadimitriou, C. H., eds., *Proc. of ACM PODS 1999*, 68–79. ACM Press.

Bang, Y.; Cahyawijaya, S.; Lee, N.; Dai, W.; Su, D.; Wilie, B.; Lovenia, H.; Ji, Z.; Yu, T.; Chung, W.; Do, Q. V.; Xu, Y.; and Fung, P. 2023. A Multitask, Multilingual, Multimodal Evaluation of ChatGPT on Reasoning, Hallucination, and Interactivity. arxiv:2302.04023.

Bohannon, P.; Flaster, M.; Fan, W.; and Rastogi, R. 2005. A Cost-Based Model and Effective Heuristic for Repairing Constraints by Value Modification. In Özcan, F., ed., *Proc. of ACM SIGMOD 2005*, 143–154. ACM.

Chiang, F.; and Miller, R. J. 2011. A unified model for data and constraint repair. In *Proc. of ICDE 2011*, 446–457. IEEE Computer Society.

Daull, X.; Bellot, P.; Bruno, E.; Martin, V.; and Murisasco, E. 2023. Complex QA and Language Models Hybrid Architectures, Survey. arxiv:2302.09051.

Eiter, T.; Fink, M.; and Stepanova, D. 2014. Computing Repairs for Inconsistent DL-programs over *EL* Ontologies. In *Proc. of Logics in Artificial Intelligence - 14th European Conference, JELIA 2014*, volume 8761, 426–441. Springer.

Kobayashi, M.; Wakabayashi, K.; and Morishima, A. 2021. Human+AI Crowd Task Assignment Considering Result Quality Requirements. In Kamar, E.; and Luther, K., eds., *Proc. of AAAI HCOMP 2021*, 97–107. AAAI Press.

Kojima, T.; Gu, S. S.; Reid, M.; Matsuo, Y.; and Iwasawa, Y. 2023. Large Language Models are Zero-Shot Reasoners. arXiv:2205.11916.

Lukasiewicz, T.; Malizia, E.; and Vaicenavicius, A. 2019. Complexity of Inconsistency-Tolerant Query Answering in Datalog+/- under Cardinality-Based Repairs. In *Proc. of AAAI, 33(01)*, 2962–2969. AAAI Press.

Morishima, A.; Fukusumi, S.; and Kitagawa, H. 2016. CyLog/Game aspect: An approach to separation of concerns in crowdsourced data management. *Inf. Syst.*, 62: 170–184.

Morishima, A.; Shinagawa, N.; Mitsuishi, T.; Aoki, H.; and Fukusumi, S. 2012. CyLog/Crowd4U: A Declarative Platform for Complex Data-Centric Crowdsourcing. *Proceedings of the VLDB Endowment*, 5(12): 1918–1921.

Nguyen, A.; Wallace, B.; and Lease, M. 2015. Combining Crowd and Expert Labels Using Decision Theoretic Active Learning. *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*, 3: 120–129.

Ouyang, L.; Wu, J.; Jiang, X.; Almeida, D.; Wainwright, C. L.; Mishkin, P.; Zhang, C.; Agarwal, S.; Slama, K.; Ray, A.; Schulman, J.; Hilton, J.; Kelton, F.; Miller, L.; Simens, M.; Askell, A.; Welinder, P.; Christiano, P.; Leike, J.; and Lowe, R. 2022. Training Language Models to Follow Instructions with Human Feedback. arxiv:2203.02155.

Prokoshyna, N.; Szlichta, J.; Chiang, F.; Miller, R. J.; and Srivastava, D. 2015. Combining Quantitative and Logical Data Cleaning. *Proc. VLDB Endow.*, 9(4): 300–311.

Qin, C.; Zhang, A.; Zhang, Z.; Chen, J.; Yasunaga, M.; and Yang, D. 2023. Is ChatGPT a General-Purpose Natural Language Processing Task Solver? arXiv:2302.06476.

van Dis, E. A. M.; Bollen, J.; van Rooij, R.; Zuidema, W.; and Bockting, C. L. 2023. ChatGPT: Five Priorities for Research. *Nature*, 614(9): 224–226.

Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Ichter, B.; Xia, F.; Chi, E.; Le, Q.; and Zhou, D. 2023. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. arxiv:2201.11903.

Wikipedia contributors. 2023a. Tokugawa Ieyasu — Wikipedia, The Free Encyclopedia. [Online; accessed 16-June-2023].

Wikipedia contributors. 2023b. Tokugawa Yoshinobu — Wikipedia, The Free Encyclopedia. [Online; accessed 16-June-2023].

Yamashita, Y.; Ito, H.; Wakabayashi, K.; Kobayashi, M.; and Morishima, A. 2022. HAEM: Obtaining Higher-Quality Classification Task Results with AI Workers. In *ACM WebSci 2022*, 118–128. ACM.

Zhao, R.; Li, X.; Joty, S.; Qin, C.; and Bing, L. 2023. Verify-and-Edit: A Knowledge-Enhanced Chain-of-Thought Framework. arxiv:2305.03268.