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 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 in order to obtain high-quality results, because LLMs often output logically inconsistent hallucinating answers (van Dis et al. 2023). This demo shows that microtasking 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 in order to obtain 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 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 humans 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.

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. 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.

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.

1The 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 Vaicenavičius 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:

```cylog
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);
```

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 crowdsourced². 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.

²Unbounded variables (e.g., Parent(X,Y)/open in Section ) are also allowed.
Demonstration Scenarios

Logic-mediated Human Task Assignment. The first scenario is the entity matching of movie titles, which we explained in Section 3, 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.

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 4 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 and potential repairs. 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 *.

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.

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 *.

References


