MEGAnno+: A Human-LLM Collaborative Annotation System

Hannah Kim, Kushan Mitra, Rafael Li Chen, Sajjadur Rahman, Dan Zhang
Megagon Labs
{hannah, kushan, rafael, sajjadur, dan_z}@megagon.ai

Abstract

Large language models (LLMs) can label data faster and cheaper than humans for various NLP tasks. Despite their prowess, LLMs may fall short in understanding of complex, socio-cultural, or domain-specific context, potentially leading to incorrect annotations. Therefore, we advocate a collaborative approach where humans and LLMs work together to produce reliable and high-quality labels. We present MEGAnno+, a human-LLM collaborative annotation system that offers effective LLM agent and annotation management, convenient and robust LLM annotation, and exploratory verification of LLM labels by humans. [1]

1 Introduction

Data annotation has long been an essential step in training machine learning (ML) models. Accurate and abundant annotations significantly contribute to improved model performance. Despite the recent advancements of pre-trained Large Language Models (LLM), high-quality labeled data remains crucial in various use cases requiring retraining. For instance, distilled models are often deployed in scenarios where repeated usage of LLMs for inference can be too costly (e.g., API calls) or time-consuming (e.g., hosting on-premise). In specialized domains like medical and human resources, organizations often need customized models to meet heightened accuracy requirements and ensure the privacy of sensitive customer data. In addition to the training step, accurate labeled data is also necessary for evaluating and understanding of model performance.

Recent explorations (Wang et al., 2021; Ding et al., 2023) have showcased the potential of LLMs in automating the data annotation process. Unlike previous task-specific machine learning models, LLMs exhibit remarkable flexibility to handle any

textual labeling task as long as suitable prompts are provided. Besides, compared to traditional annotation relying solely on human labor, LLMs can usually generate labels faster and at a lower cost. For example, hiring crowd workers for labeling may encounter problems such as delays, higher cost, difficulty in quality control (Douglas et al., 2023; Sheehan, 2018; Litman et al., 2021; Garcia-Molina et al., 2016). Studies (Gilardi et al., 2023) show that LLMs can achieve near-human or even better-than-human accuracy in some tasks. Furthermore, downstream models trained with LLM-generated labels may outperform directly using an LLM for inference (Wang et al., 2021).

Despite these advancements, it is essential to acknowledge that LLMs have limitations, necessitating human intervention in the data annotation process. One challenge is that the performance of LLMs varies extensively across different tasks, datasets, and labels (Zhu et al., 2023; Ziems et al., 2023). LLMs often struggle to comprehend subtle nuances or contexts in natural language, making involvement of humans with social and cultural understanding or domain expertise crucial. Additionally, LLMs may produce biased labels due to potentially biased training data (Abid et al., 2021; Sheng et al., 2021). In such cases, humans can recognize potential biases and make ethical judgements to correct them.

In this work, we present MEGAnno+, an annotation system facilitating human-LLM collaboration through efficient LLM annotation and selective human verification. While LLM annotations are gaining interest rapidly, a comprehensive investigation on how to onboard LLMs as annotators within a human-in-the-loop framework in labeling tools has not been conducted yet. For example, supporting LLM annotation requires not only user-friendly communications with LLMs, but also a unified backend capable of storing and managing LLM models, labels, and additional artifacts. Efficient

---

[1]Demo & video: https://meganno.github.io

human verification calls for a flexible search and recommendation feature to steer human efforts towards problematic LLM labels, along with a mechanism for humans to review and rectify LLM labels. Throughout the paper, we explain how we achieve this in our system, showcase a use case, and discuss our findings.

We summarize our contributions as below:

- A human-LLM collaborative annotation system that offers 1) effective management of LLM agents, annotations, and artifacts, 2) convenient and robust interfacing with LLMs to obtain labels, and 3) selective, exploratory verification of LLM labels by humans.
- A use case demonstrating the effectiveness of our system.
- Practical considerations and discussion on adopting LLMs as annotators.

2 Related Work

LLMs as annotators   There is growing interest in utilizing LLMs as general-purpose annotators for natural language tasks (Kuzman et al., 2023; Zhu et al., 2023; Ziems et al., 2023). Wang et al. (2021) find that GPT-3 can reduce labeling cost by up to 96% for classification and generation tasks. Similarly, Ding et al. (2023) evaluate GPT-3 for labeling and augmenting data in classification and token-level tasks. Other studies show that for some classification tasks, LLMs can even outperform crowdsourced annotators (Gilardi et al., 2023; He et al., 2023; Törnberg, 2023).

Verification of LLM responses   To detect and correct erroneous responses from LLMs, approaches to rank or filter LLM outputs have been explored. The most common method is using model logits to measure model uncertainty (Wang et al., 2021). More recently, Wang et al. (2024) propose training a verifier model using various signals from LLMs' input, labels, and explanations. Alternative methods include asking LLMs to verbalize confidence scores (Lin et al., 2022) and calculating consistency over prompt perturbations (Wang et al., 2023; Xiong et al., 2023). Other line of works investigate self-verification, i.e., LLMs give feedback on their own outputs and use them to refine themselves (Madaan et al., 2023; Cheng et al., 2023). In our system, we focus on human verification of LLM-generated labels and leave model verification and self-verification as future work.

Annotation tools with AI/ML assistance   Machine learning models have proven effective in assisting humans in various steps of the training data collection pipeline. Annotation tools and frameworks such as Prodigy (Montani and Honnibal, 2018), HumanLoop (hum), Label Studio (Tkachenko et al., 2020-2022), and Label Sleuth (Shnarch et al., 2022) all aim to enhance the subset selection step with active learning approaches. ML models are also naturally used to make predictions, serving as pre-labels. For instance, INCEpTION (Klie et al., 2018) provides annotation suggestions generated by ML models. HumanLoop (hum) and Autolabel (aut) support the annotation or augmentation of datasets using either commercial or open-source LLMs. In this work, we go beyond using LLMs to assist annotation for human annotators or to replace human annotators. Rather, MEGAnno+ advocates for a *collaboration* between humans and LLMs with our dedicated system design and annotation-verification workflows.

3 Design Considerations

Let us start with a motivating example of Moana, a Data Scientist working at a popular newspaper. Moana is tasked with training a model to analyze the degree of agreement between user comments and political opinion pieces — e.g., whether the comments entail the opinion. Moana opts for LLM annotation, but she encounters various challenges in the process. Firstly, without any guidance for prompting, she resorts to trial-and-error to eventually identify a suitable prompt for the task. Even so, she must perform additional validations to ensure that the annotated labels are within the space of pre-defined labels. Moreover, the API calls to the LLM can be unreliable, throwing exceptions such as timing out and rate limit violations, requiring her to handle such errors manually. Next, Moana lacks the confidence to train a downstream model without verifying the LLM annotations. However, without any assistance in reviewing potential annotation candidates for verification, she has to go through all the annotations, which can be time-consuming. Finally, she has to manually save used model configurations to reuse the model for additional datasets.

From Moana's example, we summarize our design requirements for a human-LLM collaborative annotation system as follows:
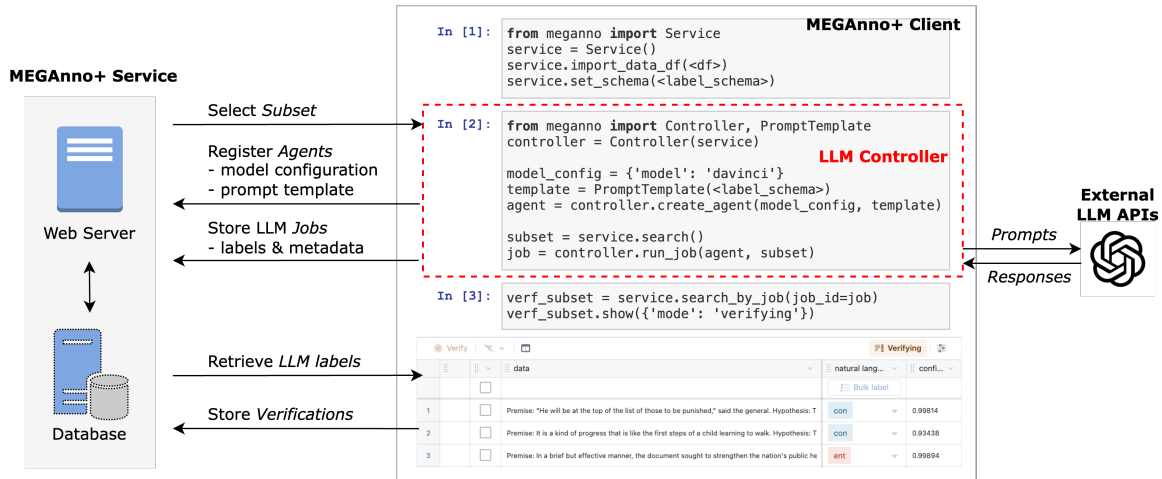
1. LLM annotation

Figure 1: MEGAnno+ system architecture and LLM-integrated workflow. With MEGAnno+ client, users can interact with the back-end service that consists of web and database servers through programmatic interfaces and UI widgets. The middle notebook shows our workflow where cell [2] is LLM annotation and cell [3] is human verification.

(a) **[Convenient]** Annotation workflow including pre-processing, API calling, and post-processing is automated.

(b) **[Customizable]** Flexibly modify model configuration and prompt templates.

(c) **[Robust]** Resolvable errors are handled by the system.

(d) **[Reusable]** Store used LLM models and prompt templates for reuse.

(e) **[Metadata]** LLM artifacts are captured and stored as annotation metadata.

2. Human verification

(a) **[Selective]** Select verification candidates by search query or recommendation.

(b) **[Exploratory]** Filter, sort, and search by labels and available metadata programmatically and in a UI.

To satisfy these design requirements, we implement our system as an extension to MEGAnno (Zhang et al., 2022), an in-notebook exploratory annotation tool. Its flexible search and intelligent recommendations enable efficient allocation of human and LLM resources toward crucial data points (R 2a,2b). Additionally, MEGAnno provides a cohesive backend for the storage of data, annotations, and auxiliary information (R 1d,1e).

4 System

4.1 System Overview

MEGAnno+ is designed to provide a convenient and robust workflow for users to utilize LLMs in text annotation. To use our tool, users operate within their Jupyter notebook (Kluyver et al., 2016) with the MEGAnno+ client installed.

Our human-LLM collaborative workflow (Fig. 1) starts with LLM annotation. This step involves compiling a subset and using an LLM to annotate it by interacting with the programmatic LLM controller. The LLM controller takes care of 1) agent registration and management (e.g., model selection and validation) and 2) running annotation jobs (e.g., input data pre-processing, initiating LLM calls, post-processing and storing responses), satisfying R 1a,1c. Once LLM annotation is completed, users can verify LLM labels. Users can select a subset of LLM labels to verify by search queries (R 2a), and inspect and correct them in a verification widget in the same notebook (R 2b).

Data Model MEGAnno+ extends MEGAnno's data model where data `Record`, `Label`, `Annotation`, `Metadata` (e.g., text embedding or confidence score) are persisted in the service database along with the task `Schema`.[2] Annotations are organized around `Subsets`, which are slices of the data created from user-defined searches or recommendations. To effectively integrate LLM into the workflow, we introduce new concepts: `Agent`, `Job`, and `Verification`. An `Agent` is defined by the configuration of the LLM (e.g., model's name, version, and hyperparameters) and a prompt template. When an agent

---

[2] MEGAnno+ only supports full LLM-integrated workflows for record-level tasks.

Figure 2: UI for customizing a prompt template and previewing generated prompts. Prompt is generated based on the name and options of label schema.

is employed to annotate a selected data subset, the execution is referred to as a `Job` (see Section 4.3.1). `Verification` captures annotations from human users that confirm or update LLM labels (see Section 4.4).

4.2 Agents: Model and Prompt Management

Since variation in either model configuration or prompt may result in a variable output from an LLM, we define an annotation `Agent` to be a combination of a user-selected configuration and prompt template. Used agents are stored in our database[3] and can be queried based on model configuration. This allows users to reuse agents and even compare the performance of different LLMs on a particular dataset (R 1d).

Model configuration MEGAnno+ enables users to choose an LLM from a list of available models, configure model parameters, and also provide a validation mechanism to ensure the selected model and parameters conform to the LLM API definition and limitations. While MEGAnno+ is designed to support any open-source LLM or commercial LLM APIs, in this work, we only demonstrate OpenAI Completion models for clarity and brevity.

Prompt template To utilize LLMs as annotators, an input record has to be transformed into a prompt text. With MEGAnno+, prompts can be automatically generated based on a labeling schema and a prompt template for users' convenience. We offer a default template that contains annotation instruction, output formatting instruction, and input slot, which can be edited programmatically. We also provide a UI widget to interactively customize the prompt template and preview the generated

---

[3] Note that for an agent, we store its prompt template (a rule to build prompt text), not prompts (generated prompts for a set of data records) to save storage.

prompts for selected data samples (Fig. 2, R 1b).

4.3 LLM Annotation

Unlike human annotation, LLM annotation goes through a multi-step process to collect labels from input data records. We execute this process as an annotation `Job` using the LLM controller.

4.3.1 Initiating LLM Jobs

To start a job, users need to select a data subset to annotate and an agent, i.e., an LLM model and a prompt template. Users can utilize MEGAnno's sophisticated subset selection techniques, including filtering by keywords or regular expressions, or receiving suggestions of similar data records. One can create a new agent or reuse one of previously registered agents. By reusing subsets and agents for new jobs, users can easily compare annotation performance between different models or for different data slices.

4.3.2 Pre-processing

The first step within a job is pre-processing. Using the prompt template of a selected agent, a data subset is converted into a list of prompts. All prompts are validated (e.g., within max token limit) before calling LLM APIs.

4.3.3 LLM API Calls: Error Handling

MEGAnno+ handles the calls to the external LLM APIs to facilitate a smooth, robust, and fault-tolerant experience for users, without having to worry about making any explicit API calls or handling error cases themselves. In order to ensure a fault-tolerant procedure, errors encountered during API calls are handled in two ways: *handle within our system* or *delegate to users*. We handle known LLM API errors that can be solved by user-side intervention. This would be in cases such as a `Timeout` or `RateLimitError` in OpenAI models, or other similar errors which require the user themselves to call to the LLM API again. On encountering such errors, MEGAnno+ retries the call to the LLM API itself. Delegated errors are the ones that require interventions by external service providers and are beyond our scope. For instance, errors such as `APIConnectionError` in OpenAI models occur because of an issue with the LLM API server itself and requires intervention from OpenAI. In this case, MEGAnno+ simply notifies the user and relays the error message.

Hypothesis: the dog is	❌ garbage generation
Label is neutral	❌ wrong format
Label: Entailment	✅ correct format, label
Label: 'entailment'	✅ quotation handled
Label: contradiction.	✅ entailing period handled

Figure 3: Example LLM responses and extraction results. Minor violations are processed as valid labels.

4.3.4 Post-processing LLM Responses and Storing Labels and Metadata

Label extraction LLM outputs are typically unstructured (i.e., free-text) and can be noisy and unusable for downstream applications, even when prompted to adhere to a specific format. This necessitates careful post-processing of LLM-generated content, converting them into valid labels (Fig. 3). MEGAnno+ conducts an automated post-processing step on LLM responses, handling errors in cases of syntax or formatting violations (i.e., not adhering to the format specified in prompt instructions). Additionally, our tool checks for semantic violations, ensuring that the generated label is valid within the existing schema for the task.

Metadata extraction MEGAnno+ can collect model artifacts and store them as label metadata (R 1e). Examples include model logits, costs associated with inference, used random seed, and so on. They can be useful for further analyses on LLM annotation and human verification. By default, our system only stores token logits to estimate the used LLM's confidence for generated labels. Calculated confidence scores serve as additional signals for decision-making in the human verification step.

Storing in database Following the post-processing step, extracted valid labels and metadata are sent to the backend service for persistence in the database. Invalid labels are not stored in the database to prevent label contamination, but frequent invalid ones are still shown to the user to guide the next iteration (e.g., update labeling schema, improve instruction in prompts).

4.3.5 Monitoring Annotation Jobs

When running a job, we display the progress and statistics of each step of the job for monitoring (Fig. 4). These include 1) agent details such as the selected model and prompt template, 2) input summary such as sample prompts generated using

```
job_uuid = controller.run_job(agent_uuid, subset, label_name)

Job issued :::

Agent ID: agent_d5cf4f6b-e698-4ecd-b704-b7bfb1fae5a2

Model config: {'model': 'text-davinci-003', 'logprobs': 0}

Prompt template:
Label the natural language inference of the following text as entailment or not_entail
ment. Your answer should be in the following format 'Natural language inference: <natu
ral language inference>'.

Text: $input
```

Pre-processing [10] record(s) :::

	Count	%
Valid prompts	10	100
Invalid prompts	0	0

Calling LLM API :::

Progress: 🟩 10/? [00:04<00:00, 2.38it/s]
Time taken to obtain responses from LLM: 4.32 seconds

	Count	%
Valid reponses	10	100
Encountered API errors	0	0

Post-processing [10] response(s) :::

	Count	%
Valid annotations	9	90
Encountered extraction errors	1	10

Annotation Summary :::

natural language inference	Count
entailment	4
not_entailment	5

Frequent invalid labels	Count
notentailed	1

Persisting the job :::
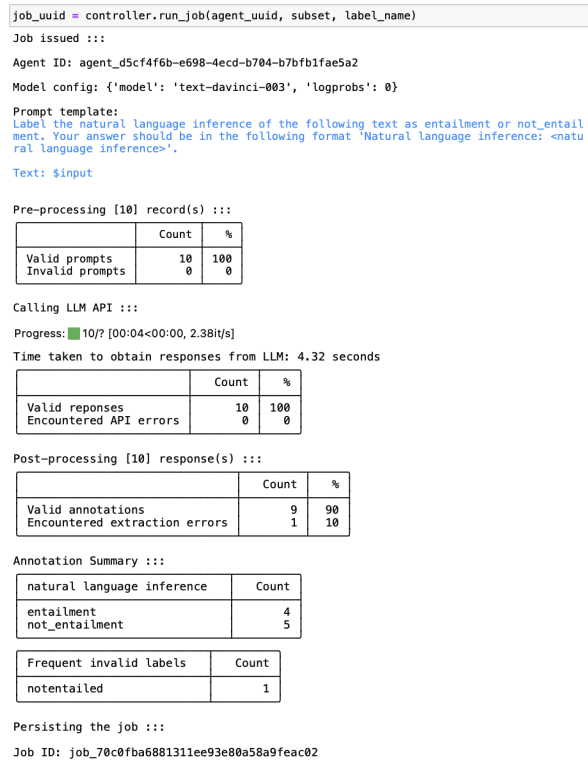Job ID: job_70c0fba6881311ee93e80a58a9feac02

Figure 4: Annotation progress and summary.

the template along with how many prompts are valid or invalid, 3) API call progress such as the time taken to retrieve responses from the API calls, and 4) output summary such as the numbers of valid and invalid responses from API and label distribution of valid responses.

4.4 Verification

LLM labels can be unreliable, requiring human verification to ensure the quality of the collected labeled data.

In-notebook verification widget MEGAnno+ provides a verification widget to complete the LLM annotation workflow in the same notebook. Leveraging MEGAnno+'s robust and customizable search functionality, users can retrieve a subset of LLM labels based on keywords, regular expressions, assigned labels, or metadata. Then utilizing the verification widget (as illustrated in Fig. 5), users can explore the selected subset and decide whether to confirm or correct their LLM-generated labels. The verification UI includes both a table view for exploratory and batch verification, as well as a single view.

Verification priority Human verification, while less expensive than direct annotation, can still be time- and cost-consuming. Therefore, it is crucial

Figure 5: The table view in verification UI. Users can explore LLM annotations via filtering by labels, sorting by confidence scores, or keyword search on text input.

to prioritize and direct human efforts toward more "suspicious" outputs from LLMs. Our widget facilitates this process by presenting metadata, such as model confidence or token logit scores, in a separate column. Users can freely sort and filter rows based on labels or metadata, enabling them to prioritize or focus on labels with low confidence.

Query and export verified labels MEGAnno+ offers flexible query interfaces, allowing users to search for verification by LLM agents (i.e., model and prompt config), as well as jobs. Both the original LLM-generated labels and any potential human corrections are stored in the database, enabling users to filter and retrieve labels "confirmed" or "corrected" by human verifiers. These features establish a foundation for easy in-notebook model and prompt comparison. Ultimately, the query results serve as a view of the labeling project, ready to be exported to downstream applications.

5 Use Case: Natural Language Inference

Moana, the aforementioned data scientist who needs to collect training data quickly, decides to use MEGAnno+ to leverage LLM-powered annotation. First, she imports her unlabeled data and sets the labeling schema as entailment or not entailment. She selects a GPT-3 davinci model with the default parameters and prompt template. To test this setting, she runs the model on 10 samples.

```
1 c = Controller(<service>, <auth>)
2 model_config = {'model': 'davinci'}
3 template = PromptTemplate(label_schema)
4 agent = c.create_agent(model_config,
       template)
5 subset = <service>.search(limit=10)
6 job = c.run_job(agent, subset)
```

After the job is finished, the annotation summary (Fig. 4) shows that all samples are successfully an-

notated by GPT-3 and 40% are entailment. Also, one response is annotated with 'notentailed', exemplifying the instability of LLMs even with clear instructions. With MEGAnno+'s table view widget, she examines data and labels (Fig. 5). She realizes that some of the records labeled as 'not entailment' are contradictory whereas the rest are neutral. She updates the labeling schema to contain entailment, neutral, and contradiction. Next, she wonders if changing the model's temperature would improve the accuracy of annotation. She creates another agent, GPT-3 with temperature with zero and re-runs annotation on the same subset.

```
1 model_config2 = {'model': 'davinci', '
       temperature': 0}
2 agent2 = c.create_agent(model_config2,
       template)
3 job2 = c.run_job(agent2, subset)
```

She exports the annotations from both jobs and compares them. She concludes that the second model is good enough for her project. She imports her entire data and uses the agent to label them. Since the size of the data is huge, she has to wait till the annotations are done. Fortunately, with MEGAnno+, she can track the progress in the output cell while the job is running. To review the annotations, she sorts the annotations in an ascending order of confidence and manually verifies low confidence ($< 95\%$) annotations.

6 Discussion

How to design an annotation task? Based on our experience, we find that designing an annotation task and a prompt similar to more widely used and standardized NLP tasks is beneficial. For example, framing Moana's problem as a natural language inference task is more effective than framing it as a binary classification of agreement and disagreement. Also, the selection of label options may work better if it is similar to common options for given tasks, such as [positive, neutral, negative] > [super positive, positive, ..., negative] for sentiment classification. Lastly, it is recommended that the format of a prompt be similar to the one used in training as some LLMs have different prompt format than the others. We plan to conduct more systematic test to discover reasonable default prompts for different models.

Are LLMs consistent and reliable annotators? We expect human annotators to maintain a consistent mental model. In other words, when humans

are presented with the same question rephrased, we anticipate consistent answers. However, LLMs are known to be sensitive to semantic-preserving perturbations in prompts. For instance, changes in prompt design, the selection and order of demonstrations, and the order of answer options can result in different outputs (Zhao et al., 2021; Pezeshkpour and Hruschka, 2023). Moreover, commercial LLMs can undergo real-time fine-tuning, meaning that prompting with the same setup today may yield different results than prompting yesterday (Chen et al., 2023). Therefore, LLM annotators and human annotators should not be treated the same, and annotation tools should carefully design their data models and workflows to accommodate both types of annotators.

Limitations Our system has several limitations. Our post-processing mechanism may not be robust to cover all tasks and prompts entered by the user. Furthermore, MEGAnno+'s ability to capture metadata is contingent on the LLM model used. For example, GPT-4 models do not yet provide any form of token logprobs or other metadata which can be captured.

7 Conclusion

MEGAnno+ is a text annotation system for human-LLM collaborative data labeling. With our LLM annotation → Human verification workflow, reliable and high-quality labels can be collected efficiently. Our tool supports robust LLM annotation, selective human verification, and effective management of LLMs, labels, and metadata.

As future work, we are currently working on adding more LLM agents (e.g., open-source LLMs), supporting customized extraction of metadata (e.g., custom uncertainty metric), and improving prompt template UI for data-aware in-context learning. Additionally, we plan to incorporate diverse annotation workflows such as Multi-agent LLM annotation → LLM label aggregation → Human verification; and LLM augmentation → Human verification.

Ethics Statement

First, labels generated by LLMs can exhibit bias or inaccuracy. These models are pre-trained on vast amount of data, which are typically not accessible to the public. Biases present in the training data can be transferred to LLM labels. Also, if the training data lacks relevant or up-to-date knowledge, the model may produce incorrect annotation. Since we cannot access models' inner workings or their training data, it is difficult to identify and understand how and why LLMs make biased or inaccurate labeling decisions. Second, the use of commercial LLMs for labeling data containing sensitive information or intellectual property may pose risks. Data shared with commercial LLMs, such as ChatGPT, may be collected and utilized for retraining these models. To prevent potential data leakage and mitigate associated legal consequences, it is advisable to either mask any confidential information or only use in-house LLMs.

References

Autolabel. Github.com/refuel-ai/autolabel.

Humanloop.com. Humanloop.com.

Abubakar Abid, Maheen Farooqi, and James Zou. 2021. Persistent anti-muslim bias in large language models. In *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*, AIES '21, page 298–306, New York, NY, USA. Association for Computing Machinery.

Lingjiao Chen, Matei Zaharia, and James Zou. 2023. How is chatgpt's behavior changing over time?

Jiale Cheng, Xiao Liu, Kehan Zheng, Pei Ke, Hongning Wang, Yuxiao Dong, Jie Tang, and Minlie Huang. 2023. Black-box prompt optimization: Aligning large language models without model training.

Bosheng Ding, Chengwei Qin, Linlin Liu, Yew Ken Chia, Boyang Li, Shafiq Joty, and Lidong Bing. 2023. Is GPT-3 a good data annotator? In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11173–11195, Toronto, Canada. Association for Computational Linguistics.

Benjamin D Douglas, Patrick J Ewell, and Markus Brauer. 2023. Data quality in online human-subjects research: Comparisons between mturk, prolific, cloudresearch, qualtrics, and sona. *Plos one*, 18(3):e0279720.

Hector Garcia-Molina, Manas Joglekar, Adam Marcus, Aditya Parameswaran, and Vasilis Verroios. 2016. Challenges in data crowdsourcing. *IEEE Transactions on Knowledge and Data Engineering*, 28(4):901–911.

Fabrizio Gilardi, Meysam Alizadeh, and Maël Kubli. 2023. Chatgpt outperforms crowd workers for text-annotation tasks. *Proceedings of the National Academy of Sciences*, 120(30).

Xingwei He, Zhenghao Lin, Yeyun Gong, A-Long Jin, Hang Zhang, Chen Lin, Jian Jiao, Siu Ming Yiu, Nan Duan, and Weizhu Chen. 2023. AnnoLLM: Making large language models to be better crowdsourced annotators.

Jan-Christoph Klie, Michael Bugert, Beto Boullosa, Richard Eckart de Castilho, and Iryna Gurevych. 2018. The INCEpTION platform: Machine-assisted and knowledge-oriented interactive annotation. In *Proceedings of the 27th International Conference on Computational Linguistics: System Demonstrations*, pages 5–9, Santa Fe, New Mexico. Association for Computational Linguistics.

Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica Hamrick, Jason Grout, Sylvain Corlay, Paul Ivanov, Damián Avila, Safia Abdalla, and Carol Willing. 2016. Jupyter notebooks – a publishing format for reproducible computational workflows. In *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, pages 87 – 90. IOS Press.

Taja Kuzman, Igor Mozetič, and Nikola Ljubešić. 2023. Chatgpt: Beginning of an end of manual linguistic data annotation? use case of automatic genre identification.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. Teaching models to express their uncertainty in words. *Transactions on Machine Learning Research*.

Leib Litman, Aaron Moss, Cheskie Rosenzweig, and Jonathan Robinson. 2021. Reply to mturk, prolific or panels? choosing the right audience for online research. *SSRN Electronic Journal*.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback.

Ines Montani and Matthew Honnibal. 2018. Prodigy: A new annotation tool for radically efficient machine teaching. *Artificial Intelligence to appear*.

Pouya Pezeshkpour and Estevam Hruschka. 2023. Large language models sensitivity to the order of options in multiple-choice questions.

Kim Bartel Sheehan. 2018. Crowdsourcing research: data collection with amazon's mechanical turk. *Communication Monographs*, 85(1):140–156.

Emily Sheng, Kai-Wei Chang, Prem Natarajan, and Nanyun Peng. 2021. Societal biases in language generation: Progress and challenges. 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 4275–4293, Online. Association for Computational Linguistics.

Eyal Shnarch, Alon Halfon, Ariel Gera, Marina Danilevsky, Yannis Katsis, Leshem Choshen, Martin Santillan Cooper, Dina Epelboim, Zheng Zhang, and Dakuo Wang. 2022. Label sleuth: From unlabeled text to a classifier in a few hours. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 159–168, Abu Dhabi, UAE. Association for Computational Linguistics.

Maxim Tkachenko, Mikhail Malyuk, Andrey Holmanyuk, and Nikolai Liubimov. 2020-2022. Label Studio: Data labeling software. Open source software available from https://github.com/heartexlabs/label-studio.

Petter Törnberg. 2023. Chatgpt-4 outperforms experts and crowd workers in annotating political twitter messages with zero-shot learning.

Shuohang Wang, Yang Liu, Yichong Xu, Chenguang Zhu, and Michael Zeng. 2021. Want to reduce labeling cost? GPT-3 can help. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4195–4205, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Xinru Wang, Hannah Kim, Sajjadur Rahman, Kushan Mitra, and Zhengjie Miao. 2024. Human-LLM collaborative annotation through effective verification of LLM labels. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*, CHI '24, New York, NY, USA. Association for Computing Machinery.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

Miao Xiong, Zhiyuan Hu, Xinyang Lu, Yifei Li, Jie Fu, Junxian He, and Bryan Hooi. 2023. Can llms express their uncertainty? an empirical evaluation of confidence elicitation in llms.

Dan Zhang, Hannah Kim, Rafael Li Chen, Eser Kandogan, and Estevam Hruschka. 2022. MEGAnno: Exploratory labeling for NLP in computational notebooks. In *Proceedings of the Fourth Workshop on Data Science with Human-in-the-Loop (Language Advances)*, pages 1–7, Abu Dhabi, United Arab Emirates (Hybrid). Association for Computational Linguistics.

Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 12697–12706. PMLR.

Yiming Zhu, Peixian Zhang, Ehsan-Ul Haq, Pan Hui, and Gareth Tyson. 2023. Can chatgpt reproduce human-generated labels? a study of social computing tasks. *arXiv preprint arXiv:2304.10145*.

Caleb Ziems, William Held, Omar Shaikh, Jiaao Chen, Zhehao Zhang, and Diyi Yang. 2023. Can Large Language Models Transform Computational Social Science? *Computational Linguistics*, pages 1–53.