`MAG-V`: A Multi-Agent Framework for Synthetic Data Generation and Verification

Saptarshi Sengupta [1], **Kristal Curtis** [1], **Akshay Mallipeddi** [1], **Abhinav Mathur** [1], **Joseph Ross**[1] **Liang Gou**[1]
[1]Splunk Inc
{ssengupta, kcurtis, amallipeddi, abhinavmathur, josephr, lgou}@splunk.com

Abstract

Extending the capabilities of Large Language Models (LLMs) with functions or *tools* for environment interaction has led to the emergence of the *agent* paradigm. In industry, training an LLM is not always feasible because of the scarcity of domain data, legal holds on proprietary customer data, rapidly changing business requirements, and the need to prototype new assistants. Agents provide an elegant solution to the above by relying on the zero-shot reasoning abilities of the underlying LLM and utilizing tools to explore and reason over customer data and respond to user requests. However, there are two concerns here: (I) acquiring large-scale customer queries for agent testing is time-consuming, and (II) high reliance on the tool call sequence (or *trajectory*) followed by the agent to respond to user queries may lead to unexpected or incorrect behavior. To address this, we propose `MAG-V`, a multi-agent framework to first generate a dataset of questions that mimic customer queries; and second, reverse-engineer alternate questions from the responses for *trajectory verification*. Initial results indicate that our synthetic data can improve agent performance on actual customer queries. Furthermore, our trajectory verification methodology, inspired by *distant supervision* and using traditional machine learning (ML) models, outperforms a GPT-4o judge baseline by 11% accuracy and matches the performance of a GPT-4 judge on our constructed dataset. Overall, our approach is a step towards unifying diverse task agents into a cohesive framework for achieving an aligned objective.

1 Introduction

Recent advances in generative-text modeling (Minaee et al., 2024; Zhao et al., 2023) have enabled *agents* (Xi et al., 2023; Wang et al., 2024a), i.e., AI applications that use LLMs for planning and reasoning and leverage external functions (*tools*) (Schick et al., 2024) to solve complex tasks and promote improved user interaction. Pushing the boundary even further are *multi-agent* systems (Guo et al., 2024) that synergize the communication between multiple agents to achieve a common goal by distributing responsibilities across agents.

Building intelligent assistants [1] to provide better customer experiences sits at the heart of every technology-driven enterprise. However, training (even fine-tuning) LLMs per customer is often infeasible due to insufficient data and privacy concerns. This creates the perfect opportunity for using agents which can reason over private data by relying only on the zero-shot abilities of its underlying LLM.

Before deploying custom assistants, a test suite of queries representative of actual customer questions is needed. While asking customers to interact with the assistant is an option that directly provides gold data, it is time consuming and thus, does not scale well. – Issue (I)

When addressing a question, an agent decides if it can be answered using the LLM's internal knowledge or, requires information from the external environment. For the latter case, the agent invokes *tools* (Patil et al., 2024), i.e., functions performing dedicated tasks such as *retrieving weather*, *converting currency*, etc. to gather the necessary world knowledge. The sequence of tools that an agent *calls*[2] is referred to as its *trajectory* (T), i.e., $T_Q = [t_1, ..., t_n]$, where T_Q is the trajectory given question Q and t_i are individual tool calls. A tool call t_i consists of two parts, viz., the tool name and the arguments. For example, in the call `get_current_weather(city = “Boston”)` the tool name is `get_current_weather` and the argument is `city = “Boston”`.

Determining the correctness of the agent trajec-

[1] *Agent* and *Assistant* is used interchangeably in this paper to refer to a customer-facing LLM helper system.

[2] In agent jargon, a *call* means producing a structured JSON string that can be parsed for use with the necessary function.

tory is a non-trivial problem and forms the first part of response verification. While using strong LLMs-*as-a-judge* (Zheng et al., 2023) has been proposed, the approach faces drawbacks (Gu et al., 2024) such as LLM sensitivity to the input prompt (Anagnostidis and Bulian, 2024) and inconsistent behavior of API (Application Programming Interface)-based models (Ouyang et al., 2024); while the latter is mitigated by altering generation temperature, it is not completely eliminated. – Issue (II)

To address both issues, we propose MAG-V, a multi-agent framework for generating questions mimicking customer queries and verifying trajectories deterministically, i.e., without using LLMs to provide feedback. For verification, we take inspiration from classical ML approaches such as *distant supervision* (Qin et al., 2018) and discriminative models such as Support Vector Machines (SVM), etc., and combine them with recent advances in LLM response verification such as Self-Verification Prompting (Weng et al., 2023).

Overall, our contribution is three-fold:

1. We propose using agents for creating synthetic data aligned with specific requirements.
2. We introduce a deterministic method for verifying agent trajectories that does not rely on using LLMs for feedback.
3. We show that simple ML baselines with feature engineering can match the performance of more expensive and capable models.

2 Related Work

We discuss two categories of recent papers that are comparable to our work.

Data Generation with Multi-Agent Systems Synthetic data generation (Long et al., 2024; Chen et al., 2024a) using LLMs has become a standard practice to address data scarcity and has been utilized for various tasks as inductive reasoning (Shao et al., 2024) and question answering (Namboori et al., 2023), the latter also utilizing ICL. However, using LLM agents for more complex data generation is a nascent and active area of study. Arif et al. (2024) utilize multiple agents (2 for generation and at most 3 for evaluation) to create a preference alignment (Wang et al., 2023; Shen et al., 2023) dataset. The issue with their strategy is the high reliance on LLM-as-a-judge, which we want to avoid, and no human-in-the-loop to gauge the quality of generations. The study by Abdullin et al. (2023) is perhaps the most related to ours. They utilize a two-agent system to produce a conversation dataset for solving linear programming word problems and also rely on human evaluators for feedback. Our approach differs from theirs as we require "questions" and not dialog data, and we prioritize tool-calling to that end. Finally, AgentInstruct (Mitra et al., 2024) and Ge et al. (2024) test the limits of multi-agent data generation. The former uses a minimum of 29 agents to transform the seed to synthetic data while the latter creates "personas" to apply specific transformations to the seed data, creating 25M and 1B samples respectively. Although impressive, these methods are still experimental, incur high costs, potentially introduce noise at such scales, and are unsuitable for domains such as ours due to large seed data requirements.

Trajectory Verification Evaluating LLM response is an active area of study (Chang et al., 2024; Chiang and Lee, 2023) as there is no standardized method yet for gauging their correctness. Early attempts include LLM-as-a-judge like frameworks such as AlpacaEval (Li et al., 2023; Dubois et al., 2024), which measures a model's *win rate* (WR). WR is the number of times a judge model prefers its response over a reference by considering factors such as consistency, relevancy, etc. Extending this idea for trajectory verification is Qin et al. (2024) who propose using a judge model (GPT-4) to determine WR and *pass rate*, i.e., whether a model is able to reach the goal in a certain number of steps. The drawback with Qin et al. (2024) is their over reliance on LLM-as-a-judge for trajectory evaluation. The closest study to ours for trajectory verification is Chen et al. (2024b) which builds on Qin et al. (2024) by performing step-by-step tool call evaluation. Although they compute similar metrics as us, they ultimately incorporate LLM-based evaluation which is orthogonal to our objective.

3 Methodology

Here we deconstruct the inner workings of MAG-V (Fig. 1) starting with the dataset construction (§3.1), followed by trajectory verification (§3.2). In our experimental setup, we have three agents: *investigator* (responsible for query generation), *assistant* (responsible for answering queries) and *reverse engineer* (responsible for creating questions based on a given response). The investigator and reverse-engineer agents use GPT-4o-mini (OpenAI, 2024a). Assistant uses GPT-4o-mini during data-generation and GPT-4o (OpenAI, 2024b) during verification as we found 4o-mini to time-out for many reverse queries (cf. Sec. 3.2). The choice of model for each agent in our experimental setup is independent of the actual models used in our production system. Each was selected to facilitate our experimentation.

3.1 Data Generation

The requirement of the synthetic generation phase is to create a dataset of queries indicative of what a customer might ask our assistant. We begin with a small set of 19 questions written by one of our engineers to meet certain product requirements. These questions, along with their verified trajectories, form our seed dataset from which we generate synthetic samples.

Ideally, we aim to generate questions that stress test our assistant, i.e., invoke a variety of tools to handle complex user queries. To that end, we perform *In-Context Learning* (ICL) (Dong et al., 2024) (predictions based on the given examples without parameter update) to illustrate to the agent the types of trajectories and questions we desire. Each ICL sample has a trajectory T_1, a question that was answered using that trajectory Q_1, and another trajectory T_2 randomly sampled from the seed dataset. The *investigator* executes T_2 and is tasked with writing 10 questions similar in style to Q_1 by studying the tool responses. This gives us a total of 190 synthetic questions, which we filter (cf. Appendix A) down to 45 questions.

To account for fluctuations in API calls (Ouyang et al., 2024), we run each of the 45 questions 5 times via the assistant and consider the most common trajectory (MCT) across all trials for annotation. A response using the MCT was randomly sampled and added to the dataset.

Finally, two independent domain experts were asked to annotate the dataset. Given a question and its corresponding trajectory, the annotators needed to mark it as $0/1$, i.e., 0: Incorrect - the trajectory makes incorrect calls/makes assumptions beyond the question requirements, etc.; and 1: Correct - All steps make sense and the overall trajectory takes into account all of the user requirements. This essentially casts the problem as a binary classification. The annotators had 0.42 Cohen's Kappa indicating moderate agreement which shows the difficulty of such esoteric domains as ours[3]. The annotators agreed on 34/45 cases and disagreements were broken with the help of another expert which ultimately yields a dataset of 30 incorrect (0) and 15 correct (1) trajectories.

3.2 Trajectory Verification

The second phase of `MAG-V` is trajectory verification, i.e., ascertaining whether the assistant calls the correct sequence of tools to reach the response. This forms the first part of response verification[4]. Our methodology is based on the hypothesis that,

> *If questions similar to a base question follow the same trajectory, we can establish a degree of confidence in the assistant's reasoning pathway. Else, high trajectory variance equates to lower confidence.*

This hypothesis is inspired by two ideas, viz.,

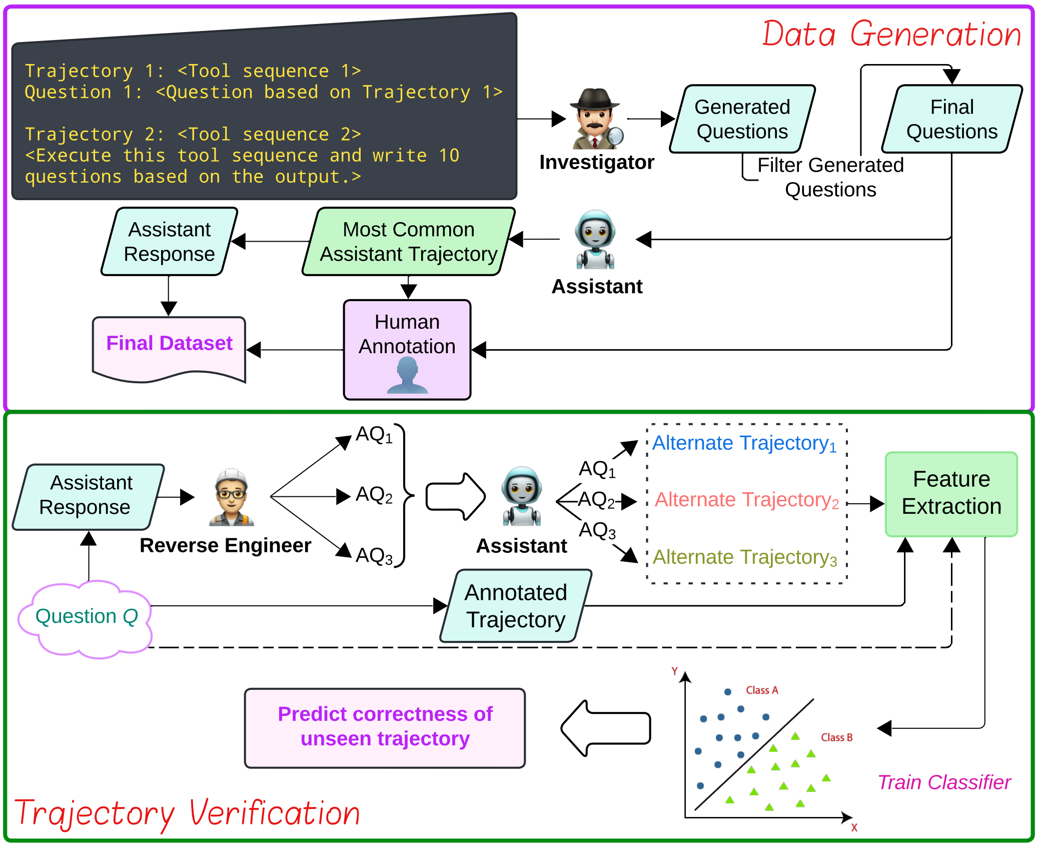
1. **Distant Supervision**: These include methods to make predictions about unlabeled data using auxiliary knowledge sources. For example, when building a relation extraction model (Smirnova and Cudré-Mauroux, 2018), we might use *knowledge graphs* to make relationship inferences on unstructured text. In a related fashion, we use the trajectories of the similar questions to make predictions about our annotated trajectories.

2. **Backward Reasoning**: Two recent studies, *reverse question answering* (RQA) (Balepur et al., 2024) and self-verification prompting (Weng et al., 2023) explore the idea of verifying LLM-responses by generating questions from the final response and asking the LLM to answer them. RQA is a *jeopardy*-like (Ferrucci et al., 2013) technique where the generated question's answer must be the response, while the latter generates questions whose answers contain all facts entailed in the original question. Inspired by these methods, we adopt a similar backward question generation strategy, but we study trajectory verification instead of response.

To verify the MCT of a question Q, we first ask the reverse engineer agent to create 3 *alternate* questions (AQ) based on the assistant's response using the MCT. The agent was instructed to write questions that captured all of the main points of the response. The assistant then answers the AQ's which leads to 3 alternate trajectories (AT).

[3] We will release details of our platform upon publication.

[4] Verifying the assistant's answer is beyond the scope of this study.



```
Trajectory 1: <Tool sequence 1>
Question 1: <Question based on Trajectory 1>

Trajectory 2: <Tool sequence 2>
<Execute this tool sequence and write 10
questions based on the output.>
```

Figure 1: Overview of MAG-V [AQ = Alternate Question.]

Using the annotated (or base) trajectory (BT) and the AT's, we extract *features*[5] to determine the similarity between them. These features are a combination of statistical and embedding-based measures. In total, we compute six features between the BT and AT's (cf. Appendix B), viz., **EM** (Exact Match - 0/1 measure of equality); **EDIT** (minimum number of string edits needed to convert AT to BT); **GEDIT** (Graph Edit Distance between the trajectories formulated as graphs); **SS** (Semantic (Cosine) Similarity between BT and AT); **AO** (Argument Overlap - count of common arguments between BT and AT) and **LCSS** (Longest Common Sequence from Starting Call to measure the extent of commonality between BT and AT from a common point). Additionally, we also extract **TF-IDF** (Term Frequency-Inverse Document Frequency) features from the base questions to provide grounding context to the models as a trajectory can only be analyzed in relation to its question.

Finally, using these features we train discriminative ML models using stratified ten-fold cross-validation to account for label imbalance. Overall, we test 7 ML models (Random Forest, Logistic Regression, Naive Bayes, k-Nearest Neighbours (k-NN), Support Vector Machines (SVM), Decision Tree, XGBoost) and report the mean accuracy and F1 across all folds and 3 random seeds. Apart from AO, we test the influence of including and excluding the tool arguments for each measure for trajectory verification. All models used default settings except for k-NN which we found to have the best performance at $k = 5$ (without arguments verification) and $k = 4$ (with arguments verification).

3.2.1 GPT-baseline

To compare against a strong LLM-as-a-judge, we selected GPT-4 (OpenAI et al., 2024) and GPT-4o. Each model was tested on the same test splits and random seeds from the cross-validation to provide consistent evaluation. We provided the same annotation rubric to the models, but switched the position of the labels to account for *recency bias* (Zhao et al., 2021), i.e., the tendency of LLMs to repeat the last seen values in the prompt. Before generating its score, the LLM was asked to provide a rationale as it has been shown (Wang et al., 2024b) to lead to better evaluation. Additionally, we provided one example each of a correct and incorrect $Q + T$ pair from the seed dataset to further help the

[5] Ideally, we should refer to these as *metrics*. However, we use *features* to distinguish between evaluation metrics and values used to train our models.

models. Finally, we also tested performance with and without using the default system prompt (*You are a helpful AI Assistant.*) in `Autogen` (Wu et al., 2024), the framework used to create our agents. All other model parameters were set to their default.

4 Experiments

In this section, we first explain the benefit of generating synthetic questions and then detail the results from our trajectory verification trials.

4.1 Utilizing Synthetic Data

The goal of creating synthetic questions was to simulate user queries to test the capabilities of our assistant. As a by-product of this step, we also find that the generated data can be used to aid our assistant in fixing its mistakes, and answering questions better through ICL. The following is an example of a **real customer query** (ignore the grammar issue), modified to work in our test environment.

> I see that the paymentservice on production had an increased error rate around 7:59pm EST 10/10/2024, can you dive into that service during that time and a half hour and each side and find out what the most common errors associated were?

The assistant usually runs GPT-4o as its backbone. However, to test for cost-efficiency, we also considered 4o-mini. To answer this question, the assistant needs to use the correct time range, which is 7:29 PM to 8:29 PM on October 10. We run the above query with both 4o and 4o-mini, and also add all 45 synthetic questions and their MCT as ICL samples. The 4o-backed assistant with and without the samples fails to use the correct time range: it uses `{“time_range”:{“start”:“2024-10-10T23:29:00Z”, “stop”:“2024-10-11T00:29:00Z”}}`. While the start date is correct, the start/end times and end date are wrong. With 4o-mini, the assistant spirals into a chain of retries, unable to fix itself even after 10 attempts. However, by adding our generated questions and their MCT as ICL samples, the assistant was able to to figure out the correct time range in the first try, `{“time_range”: {“start”: “2024-10-10T19:29:00Z”, “stop”: “2024-10-10T20:29:00Z”}}}`. This signals potential cost-benefits, i.e., by coupling a cheaper model with ICL samples, we can guide it to perform better or on-par with more expensive LLMs.

4.2 Trajectory Verification

The results from our verification trials (accuracy and F1) are shown in Figure 2. As we see, each ML model benefits from features that consider tool arguments. The reason for experimenting with removing the arguments is to see if the *overall tool calls* were consistent across the base and alternate trajectories. However, it makes sense that by considering the arguments, the models perform better as they have more fine-grained features to learn from. Furthermore, by excluding the arguments, the boundary separating the correct and incorrect trajectories becomes less prominent since many trajectories with the same calls get placed in both classes. For example, if $T_1 = T_2 = [t_1, t_2, t_3]$, where T_1 is incorrect and T_2 is correct for their respective questions, the model learns the same features for each class which in turn distorts its performance. By adding the arguments, the model is able to learn a better decision boundary. Overall, we see that our k-NN model shows the best accuracy and F1 scores across all ML models.

Scores from the GPT-models are a bit surprising. Firstly, GPT-4o, which is claimed to be a much stronger model than GPT-4 (OpenAI, 2024b), is outperformed by the latter by quite a margin. Second, by simply adding the system prompt (*You are a helpful AI Assistant.*), the accuracy of GPT-4o increases by about 12%, which hearkens to our point on prompt sensitivity, i.e., making such small modifications to a prompt can have tremendous impact on the output. Finally, our **k-NN model shows 11% accuracy and 4% F1 improvement over GPT-4o** which demonstrates the effectiveness of such a simple model over a more complex LLM. While our k-NN model (82.33%) matches GPT-4 in accuracy (82.83%), it lags behind in F1 (71.73 vs. 76.06), indicating a reduced alignment of predictions. We believe that this can be addressed if we have a larger dataset for the model to learn from.

We also consider feature ablations for our models shown in Table 1. These trials only considered the six trajectory features (with arguments) to help us understand the extent of performance we can get by only conditioning our models on the trajectories. As the number of features here is much lower than the standard setup, we run five-fold cross-validation (to avoid overfitting) but use the same random seeds as before. We test each feature combination (by considering 1, . . ., 6 features at a

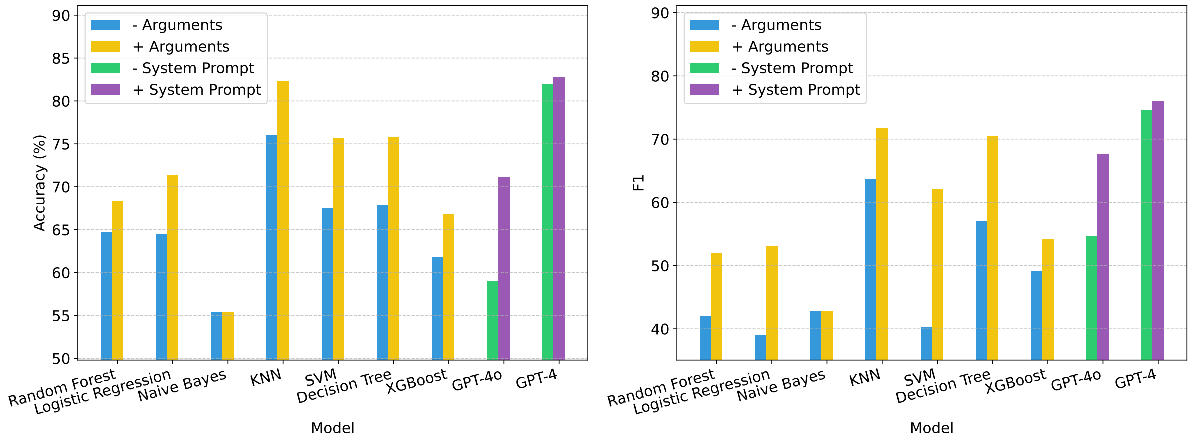


Figure 2: Accuracy (left) and F1 (right) scores from all ML models using all features v/s GPT-baselines.

time) and report scores for those models which had the best score using the same feature subset across each seed. The GPT-baselines shown in Table 1 are their best scores, i.e., using the system prompt. In this setup, we see that our best models come close to GPT-4 and outperform GPT-4o in accuracy. Additionally, the distribution of predictions (F1) is much more aligned with that of GPT-4. Interestingly the models utilize only a single feature, i.e., EDIT distance, to make their best predictions. We reason that this is because EDIT is a more lenient metric than the others, as it determines the amount of work needed to align two trajectories, rather than penalizing mismatches which in turn provides more signal to the models.

Finally, we discuss the trade-offs with our approach versus using LLM-as-a-judge. At the outset, we see that performance for both are comparable using either question and trajectory features, or just the latter in isolation. However, as our method requires three alternate questions and consequent trajectories to be generated, it incurs more API calls. At scale, this might accrue cost if using an expensive LLM such as GPT-4 or GPT-4o. However, switching to a cheaper model such as GPT-4o-mini or open-source models such as Llama 3 (Dubey et al., 2024) can alleviate this to a great extent. Where our approach wins over the GPT benchmark is in *determinism*, i.e., for the exact same conditions, it will always give the same result, which cannot be said for API-based LLMs. However, it might be argued that changes in the alternate trajectories can impact the features being learned. To this, we explain that our approach follows a *best-of-N-predictions* strategy, i.e., we base our decisions on what the majority of the trajectories predict as opposed to a single trajectory. This leads to a model that is more robust and confident in its predictions.

Model	Feature(s)	Accuracy (%)	F1
GPT-4o	-	71.11	69.85
GPT-4	-	**82.22**	**77.94**
Random Forest	**EDIT**	**80**	**75.3**
Decision Tree	EDIT	77.04	70.87
XGBoost	**EDIT**	**80**	**75.3**

Table 1: ML models trained using the best trajectory feature subset vs. GPT-baselines. Using EDIT distance, each model displays the best accuracy and F1 across all random seeds. Each score is the average from all trials.

5 Conclusion and Future Work

We introduce MAG-V, a framework for generating synthetic questions using LLM agents and verifying the trajectory taken by an agent to answer questions. The novelty of our verification system lies in its deterministic aspect, i.e., reducing the reliance on API-based LLMs to judge the veracity of results (trajectories). That said, we believe there is more work to be done. First, we need to test how our method scales with the number of samples. Second, we use TF-IDF features to represent questions. We will look at ways to better ground trajectories to their associated questions to improve verification performance. Finally, during annotation, we observed that some “incorrect” trajectories *can* be marked as correct, considering the complexity of the questions. As such, we aim to smooth our labels to have 3-classes (correct, *partially correct* and incorrect). This will be a bigger challenge for the GPT models as they have been shown (Fu et al., 2023) to struggle with multiclass classification. We aim to see how our models fare under such a setup.

References

Yelaman Abdullin, Diego Molla, Bahadorreza Ofoghi, John Yearwood, and Qingyang Li. 2023. Synthetic dialogue dataset generation using LLM agents. In *Proceedings of the Third Workshop on Natural Language Generation, Evaluation, and Metrics (GEM)*, pages 181–191, Singapore. Association for Computational Linguistics.

Sotiris Anagnostidis and Jannis Bulian. 2024. How susceptible are LLMs to influence in prompts? In *First Conference on Language Modeling*.

Samee Arif, Sualeha Farid, Abdul Hameed Azeemi, Awais Athar, and Agha Ali Raza. 2024. The fellowship of the llms: Multi-agent workflows for synthetic preference optimization dataset generation. *CoRR*, abs/2408.08688.

Nishant Balepur, Feng Gu, Abhilasha Ravichander, Shi Feng, Jordan Boyd-Graber, and Rachel Rudinger. 2024. Reverse question answering: Can an llm write a question so hard (or bad) that it can't answer? *arXiv preprint arXiv:2410.15512*.

Ricardo J. G. B. Campello, Davoud Moulavi, and Joerg Sander. 2013. Density-based clustering based on hierarchical density estimates. In *Advances in Knowledge Discovery and Data Mining*, pages 160–172, Berlin, Heidelberg. Springer Berlin Heidelberg.

Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, Wei Ye, Yue Zhang, Yi Chang, Philip S. Yu, Qiang Yang, and Xing Xie. 2024. A survey on evaluation of large language models. *ACM Trans. Intell. Syst. Technol.*, 15(3).

Xiaoyang Chen, Ben He, Hongyu Lin, Xianpei Han, Tianshu Wang, Boxi Cao, Le Sun, and Yingfei Sun. 2024a. Spiral of silence: How is large language model killing information retrieval?—A case study on open domain question answering. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14930–14951, Bangkok, Thailand. Association for Computational Linguistics.

Zehui Chen, Weihua Du, Wenwei Zhang, Kuikun Liu, Jiangning Liu, Miao Zheng, Jingming Zhuo, Songyang Zhang, Dahua Lin, Kai Chen, and Feng Zhao. 2024b. T-eval: Evaluating the tool utilization capability of large language models step by step. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9510–9529, Bangkok, Thailand. Association for Computational Linguistics.

Cheng-Han Chiang and Hung-yi Lee. 2023. Can large language models be an alternative to human evaluations? In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15607–15631, Toronto, Canada. Association for Computational Linguistics.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Baobao Chang, Xu Sun, Lei Li, and Zhifang Sui. 2024. A survey on in-context learning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 1107–1128, Miami, Florida, USA. Association for Computational Linguistics.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Yann Dubois, Balázs Galambosi, Percy Liang, and Tatsunori B Hashimoto. 2024. Length-controlled alpacaeval: A simple way to debias automatic evaluators. *arXiv preprint arXiv:2404.04475*.

David Ferrucci, Anthony Levas, Sugato Bagchi, David Gondek, and Erik T Mueller. 2013. Watson: beyond jeopardy! *Artificial Intelligence*, 199:93–105.

Michael Fu, Chakkrit Kla Tantithamthavorn, Van Nguyen, and Trung Le. 2023. Chatgpt for vulnerability detection, classification, and repair: How far are we? In *2023 30th Asia-Pacific Software Engineering Conference (APSEC)*, pages 632–636.

Tao Ge, Xin Chan, Xiaoyang Wang, Dian Yu, Haitao Mi, and Dong Yu. 2024. Scaling synthetic data creation with 1,000,000,000 personas. *arXiv preprint arXiv:2406.20094*.

Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, Yuanzhuo Wang, and Jian Guo. 2024. A survey on llm-as-a-judge. *Preprint*, arXiv:2411.15594.

Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V. Chawla, Olaf Wiest, and Xiangliang Zhang. 2024. Large language model based multi-agents: A survey of progress and challenges. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*, pages 8048–8057. International Joint Conferences on Artificial Intelligence Organization. Survey Track.

Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Alpacaeval: An automatic evaluator of instruction-following models. `https://github.com/tatsu-lab/alpaca_eval`.

Lin Long, Rui Wang, Ruixuan Xiao, Junbo Zhao, Xiao Ding, Gang Chen, and Haobo Wang. 2024. On LLMs-driven synthetic data generation, curation, and evaluation: A survey. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 11065–11082, Bangkok, Thailand. Association for Computational Linguistics.

Leland McInnes, John Healy, Nathaniel Saul, and Lukas Grossberger. 2018. Umap: Uniform manifold approximation and projection. *The Journal of Open Source Software*, 3(29):861.

Frederic P. Miller, Agnes F. Vandome, and John McBrewster. 2009. *Levenshtein Distance: Information theory, Computer science, String (computer science), String metric, Damerau?Levenshtein distance, Spell checker, Hamming distance*. Alpha Press.

Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, and Jianfeng Gao. 2024. Large language models: A survey. *arXiv preprint arXiv:2402.06196*.

Arindam Mitra, Luciano Del Corro, Guoqing Zheng, Shweti Mahajan, Dany Rouhana, Andres Codas, Yadong Lu, Wei-ge Chen, Olga Vrousgou, Corby Rosset, Fillipe Silva, Hamed Khanpour, Yash Lara, and Ahmed Awadallah. 2024. Agentinstruct: Toward generative teaching with agentic flows. arXiv.

Niklas Muennighoff, Nouamane Tazi, Loic Magne, and Nils Reimers. 2023. MTEB: Massive text embedding benchmark. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2014–2037, Dubrovnik, Croatia. Association for Computational Linguistics.

Amani Namboori, Shivam Sadashiv Mangale, Andy Rosenbaum, and Saleh Soltan. 2023. GeMQuAD : Generating multilingual question answering datasets from large language models using few shot learning. In *NeurIPS 2023 Workshop on Synthetic Data Generation with Generative AI*.

OpenAI. 2024a. Gpt-4o mini: advancing cost-efficient intelligence.

OpenAI. 2024b. Hello gpt-4o.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Shuyin Ouyang, Jie M. Zhang, Mark Harman, and Meng Wang. 2024. An empirical study of the non-determinism of chatgpt in code generation. *ACM Trans. Softw. Eng. Methodol.* Just Accepted.

Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. 2024. Gorilla: Large language model connected with massive APIs. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Pengda Qin, Weiran Xu, and William Yang Wang. 2018. Robust distant supervision relation extraction via deep reinforcement learning. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2137–2147, Melbourne, Australia. Association for Computational Linguistics.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, dahai li, Zhiyuan Liu, and Maosong Sun. 2024. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *The Twelfth International Conference on Learning Representations*.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2024. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36.

Yunfan Shao, Linyang Li, Yichuan Ma, Peiji Li, Demin Song, Qinyuan Cheng, Shimin Li, Xiaonan Li, Pengyu Wang, Qipeng Guo, et al. 2024. Case2code: Learning inductive reasoning with synthetic data. *arXiv preprint arXiv:2407.12504*.

Tianhao Shen, Renren Jin, Yufei Huang, Chuang Liu, Weilong Dong, Zishan Guo, Xinwei Wu, Yan Liu, and Deyi Xiong. 2023. Large language model alignment: A survey. *arXiv preprint arXiv:2309.15025*.

Alisa Smirnova and Philippe Cudré-Mauroux. 2018. Relation extraction using distant supervision: A survey. *ACM Comput. Surv.*, 51(5).

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. 2024a. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345.

Peiyi Wang, Lei Li, Liang Chen, Zefan Cai, Dawei Zhu, Binghuai Lin, Yunbo Cao, Lingpeng Kong, Qi Liu, Tianyu Liu, and Zhifang Sui. 2024b. Large language models are not fair evaluators. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9440–9450, Bangkok, Thailand. Association for Computational Linguistics.

Yufei Wang, Wanjun Zhong, Liangyou Li, Fei Mi, Xingshan Zeng, Wenyong Huang, Lifeng Shang, Xin Jiang, and Qun Liu. 2023. Aligning large language models with human: A survey. *Preprint*, arXiv:2307.12966.

Yixuan Weng, Minjun Zhu, Fei Xia, Bin Li, Shizhu He, Shengping Liu, Bin Sun, Kang Liu, and Jun Zhao. 2023. Large language models are better reasoners with self-verification. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 2550–2575, Singapore. Association for Computational Linguistics.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang (Eric) Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Ahmed Awadallah, Ryen W. White, Doug Burger, and Chi Wang. 2024. Autogen: Enabling next-gen llm applications via multi-agent conversation. In *COLM 2024*.

Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, et al. 2023. The rise and potential of large language model based agents: A survey. *arXiv preprint arXiv:2309.07864*.

Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. 2024. C-pack: Packed resources for general chinese embeddings. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '24, page 641–649, New York, NY, USA. Association for Computing Machinery.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*.

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.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36:46595–46623.

A Filtering Generated Questions

To avoid having questions of the same type for our test set, we filter the generated questions. We start by embedding the questions using BAAI General Embeddings (BGE) (Xiao et al., 2024) as they achieve state-of-the-art performance on MTEB

(Massive Text Embedding Benchmark) (Muennighoff et al., 2023). This yields R^{1024} question embeddings. Using UMAP (Uniform Manifold Approximation and Projection) (McInnes et al., 2018) for dimension-reduction to R^2, we cluster the resulting embeddings with HDBSCAN (Campello et al., 2013). This results in 11 clusters which we manually verified as being sufficiently diverse in their themes such as requesting information on a given service in our environments, issues related to a hardware component, etc. Considering the mean of all embeddings in a cluster as the centroid, we take the top-5 questions with the least distance to the centroid, which gives a total of 55 questions. Finally, we remove 10 questions that are either generic and do not require tool calls, or pertain to an aspect of our platform that is under active development. This yields a dataset of 45 questions.

B Trajectory Features

We extract six features across the base and alternate trajectories, described as follows:

- **EM** (Exact Match): A binary measure of equality, i.e., $EM = 1$, if $BT = AT$ verbatim, else 0.

- **EDIT**: Levenshtein Edit Distance (Miller et al., 2009) between BT and AT, i.e., the minimum number of *edits* (add, delete, substitute) required to modify the AT to become the BT.

- **GEDIT** (Graph Edit Distance): As a trajectory is a sequence of operations, it can also be viewed as a *directed graph*. As such, we decide to measure the edit distance required to make the BT and AT graphs isomorphic, i.e., similar in structure. GEDIT is similar to EDIT, but operates on nodes and edges.

- **SS** (Semantic Similarity): This measures cosine similarity between the BT and AT embedded using `BGE-Large` embeddings.

- **AO** (Argument Overlap): Count of arguments common to BT and AT for each tool call pair. AO is an F1 score ($2PR/(P + R)$), where:
  - *overlap* (O) = number of common arguments across all calls for BT and AT.
  - *precision* (P) = O / total number of arguments in AT.
  - *recall* (R) = O / total number of arguments in BT.

- **LCSS** (Longest Common Sequence from Starting Call): The longest tool call sequence common to BT and AT starting from the first call. Similar to AO, LCSS is an F1 score with:
  - L = Length of longest sequence across BT and AT from the starting call.
  - P = L / total number of calls in AT.
  - R = L / total number of calls in BT.