

Figure 4: High-level system diagram of an LLM (AI) Agent. The flow of the diagram follows the numbering: (1) User sends a query to the Agent. Alongside with the user query the Agent is provided with the description of all tools available to it (2). With this information the LLM plans (3) actions. Some of these actions require tool invocations (4). After tool calls, an observation is made (5). Based on the observation the agent responds with an answer to the User (6).

A Broader Impact

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

B Agent

In this work, the primary focus is on how LLM Agents utilize the available tools. This said we consider a minimal agent system design - one containing LLM for reasoning and planning, and tools. This design choice is made to minimize system overhead. The general framework considered can be seen in Figure 4.

C Source code

The source code for TOOLFUZZ as well as the evaluation of TOOLFUZZ is attached as a zip file alongside the paper.

All the instructions on how to run the code base can be found in the README.md file, located at the root directory of the project. The codebase is written in Python 3.10 using a Conda virtual environment. TOOLFUZZ has been developed, run and evaluated on Ubuntu 22.04. All dependencies are listed under the requirements.yml file – instruction on how to setup the environment are also provided in the README.md file.

D Experimental setup

All of the experiments are run on a single machine without the need of GPU. The used machine for evaluation is equipped with Intel(R) Xeon(R) Platinum 8358 CPU @ 2.60GHz with 30 cores and 222 GB of RAM. Further all experiments are run with budgeting. The selected budget is 5 minutes per tool as a hard limit and a soft token budget of 25 000 tokens. Further we also have a soft price budget of 0.1 USD per tool. Budgets for tokens and price are soft as they would not stop the test execution but would be logged and reported. The variety of tools is large and some of the tools would go over the token and price limits, however most of them are within the limits.

E Tested Tools

In this section we present the tools used for the evaluation of TOOLFUZZ. For evaluation of TOOLFUZZ we have used 56 tools from the LangChain library and 83 tools from Composio. The selection criteria for the tools is that they have to be free to use and do not require any API Key or any other form of authentication. We have made exception for the GitHub toolkit as it is widely used. The full list of tools for composio can be seen in Figure 6 and for LangChain in Figure 5.

duckduckgo_results_json	requests_post
duckduckgo_search	requests_patch
youtube_search	requests_put
Dall-E-Image-Generator	requests_delete
wikipedia	Get_Issues
terminal	Get_Issue
semanticscholar	Comment_on_Issue
Wikidata	List_open_pull_requests__PRs_
python_repl	Get_Pull_Request
open-street-map-route-distance	Overview_of_files_included_in_PR
pub_med	Create_Pull_Request
open-street-map-search	List_Pull_Requests_Files
ionic_commerce_shopping_tool	Create_File
stack_exchange	Read_File
query_graphql	Update_File
arxiv	Delete_File
Search_NASA_Image_and_Video_Library_media	Overview_of_existing_files_in_Main_branch
Get_NASA_Image_and_Video_Library_media_metadata_manifest	Overview_of_files_in_current_working_branch
Get_NASA_Image_and_Video_Library_media_metadata_location	List_branches_in_this_repository
Get_NASA_Image_and_Video_Library_video_captions_location	Set_active_branch
copy_file	Create_a_new_branch
file_delete	Get_files_from_a_directory
file_search	Search_issues_and_pull_requests
move_file	Search_code
read_file	Create_review_request
write_file	json_spec_list_keys
list_directory	json_spec_get_value
requests_get	python_repl_ast

Figure 5: List of Langchain Tools used for the evaluation.

F Benchmarks

Table 4: Binomial Confidence Intervals at 95% confidence for Table 2. TOOLFUZZ Auto Fix (TF-AF), DRAFT and DRAFT applied on top of TOOLFUZZ Auto Fix for the File Management (FMB) and GitHub Benchmarks (GHB)

	Tool	Original	TF-AF	DRAFT	TF-AF + DRAFT
FMB	Terminal	[0.25, 0.36]	[0.34, 0.46]	[0.33, 0.45]	[0.27, 0.38]
	FileToolkit	[0.28, 0.40]	[0.35, 0.47]	[0.30, 0.42]	[0.28, 0.39]
GHB	GithubToolkit	[0.13, 0.37]	[0.34, 0.46]	[0.18, 0.43]	[0.15, 0.39]

For both benchmarks, we have used the same autofix setup. In both cases GPT-4o is utilized with the following prompts: Figure 22, Figure 24, Figure 23.

Further we generated 10 fixes for each tool using the TOOLFUZZ’s Auto Fix (TF-AF) and DRAFT. Then the generated fixes are evaluated using the validation split of the benchmark, to remove selection bias. The best performing fix is then selected for the full benchmark evaluation.

F.1 File Management toolkit benchmark

The file management toolkit benchmark consists of 42 domain-specific environments, which are listed in Figure 8.

Each task of the benchmark is setup in a Docker container. The Docker container has the initial state of the file system as well as a ReAct agent with the tool under test. The agent is then presented with

EMBED_TOOL_CREATE_IMAGE_VECTOR_STORE	GIT_GIT_REPO_TREE
WEBTOOL_SCRAPE_WEBSITE_CONTENT	BROWSER_TOOL_GET_SCREENSHOT
FILETOOL_GIT_REPO_TREE	IMAGE_ANALYSER_ANALYSE
FILETOOL_WRITE	CODE_ANALYSIS_TOOL_GET_METHOD_BODY
BROWSER_TOOL_SCROLL_PAGE	SHELLTOOL_EXEC_COMMAND
computer	GREPTILE_CODE_QUERY
WORKSPACE_TOOL_WORKSPACE_STATUS_ACTION	SPIDERTOOL_SCRAPE
RAGTOOL_ADD_CONTENT_TO_RAG_TOOL	BROWSER_TOOL_GET_ELEMENT_DETAILS
SHELLTOOL_CREATE_SHELL	str_replace_editor
FILETOOL_OPEN_FILE	BROWSER_TOOL_NAVIGATE_HISTORY
FILETOOL_RENAME_FILE	BROWSER_TOOL_GOTO_PAGE
BROWSER_TOOL_CLICK_ELEMENT	BROWSER_TOOL_REFRESH_PAGE
WEBTOOL_SCRAPE_WEBSITE_ELEMENT	FILETOOL_EDIT_FILE
GIT_GITHUB_CLONE_CMD	BROWSER_TOOL_GET_PAGE_DETAILS
FILETOOL_CREATE_FILE	COMPOSIO_WAIT_FOR_CONNECTION
FILETOOL_SEARCH_WORD	HACKERNEWS_GET_USER
ZEPTOOL_CREATE_SESSION	COMPOSIO_RETRIEVE_ACTIONS
FILETOOL_FIND_FILE	CODEINTERPRETER_EXECUTE_CODE
MATHEMATICAL_CALCULATOR	HACKERNEWS_GET_ITEM_WITH_ID
BROWSER_TOOL_TYPE_TEXT	COMPOSIO_GET_RESPONSE_SCHEMA
FILETOOL_LIST_FILES	COMPOSIO_ADVANCED_USE_CASE_SEARCH
CODE_ANALYSIS_TOOL_GET_RELEVANT_CODE	COMPOSIO_INITIATE_CONNECTION
ZEPTOOL_ADD_MEMORY	WEATHERMAP_WEATHER
SHELLTOOL_SPAWN_PROCESS	COMPOSIO_GET_REQUIRED_PARAMETERS
SHELLTOOL_TEST_COMMAND	COMPOSIO_EXECUTE_ACTION
FILETOOL_GIT_CUSTOM	COMPOSIO_ENABLE_TRIGGER
ZEPTOOL_GET_MEMORY	ENTELLIGENCE_INTERACT_WITH_THE_REPOSITORY_AGENT
GIT_GET_PATCH_CMD	CODEINTERPRETER_RUN_TERMINAL_CMD
FILETOOL_SCROLL	HACKERNEWS_SEARCH_POSTS
ZEPTOOL_SEARCH_MEMORY	CODEINTERPRETER_GET_FILE_CMD
SQLTOOL_SQL_QUERY	COMPOSIO_CHECK_ACTIVE_CONNECTION
HISTORY_FETCHER_GET_WORKSPACE_HISTORY	HACKERNEWS_GET_FRONTPAGE
CODE_ANALYSIS_TOOL_CREATE_CODE_MAP	HACKERNEWS_GET_LATEST_POSTS
CODE_ANALYSIS_TOOL_GET_METHOD_SIGNATURE	COMPOSIO_SEARCH_DUCK_DUCK_GO_SEARCH
FILETOOL_GIT_CLONE	HACKERNEWS_GET_TODAYS_POSTS
RAGTOOL_RAG_TOOL_QUERY	TEXT_TO_PDF_CONVERT_TEXT_TO_PDF
CODE_FORMAT_TOOL_FORMAT_AND_LINT_CODEBASE	COMPOSIO_LIST_TRIGGERS
EMBED_TOOL_QUERY_IMAGE_VECTOR_STORE	COMPOSIO_LIST_APPS
FILETOOL_GIT_PATCH	ENTELLIGENCE_ADD_A_NEW_REPOSITORY
SPIDERTOOL_CRAWL	COMPOSIO_RETRIEVE_APPS
CODE_ANALYSIS_TOOL_GET_CLASS_INFO	CODEINTERPRETER_UPLOAD_FILE_CMD
FILETOOL_CHANGE_WORKING_DIRECTORY	

Figure 6: List of Composio Tools used for the evaluation.

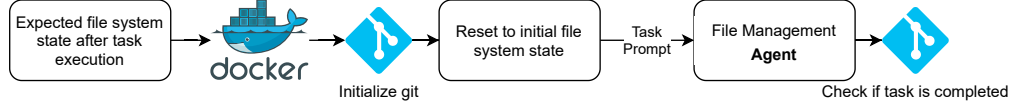


Figure 7: File management benchmark setup

Accountant	Finance and Banking	Machine Learning
Agriculture and Precision Farming	Forensic Science	Manufacturing and Automation
Art	Geology and Geophysics	Meteorology and Climate Science
Artificial Intelligence	Government and Public Administration	Music Production and Sound Engineering
Astronomy and Space Exploration	Graphic Design and Animation	Pharmaceuticals and Drug Development
Biotechnology	Healthcare and Medicine	Psychology and Neuroscience
Computer Science	History and Archival Science	Real Estate
Construction and Architecture	Hospitality and Tourism	Robotics
Cybersecurity	Human Resources and Recruitment	Social Media and Digital Marketing
Data Science	Journalism and Digital Media	Sports Science and Analytics
E-commerce and Retail	Law and Legal Analytics	Telecommunications
Economics and Market Research	Linguistics and Natural Language Processing	Video Game Development
Education and E-learning	Logistics and Supply Chain Management	
Energy and Utilities	Logistics Tour	
Entertainment and Media		
Environmental Science		

Figure 8: List of environment domains with file systems and tasks for the File Management Domain

the task prompt in the initialized environment. Upon completion of the agent’s execution, the success of the task is verified using a `git diff` between the initial and final states (Figure 7).

Further we have computed the binomial confidence intervals for the results from Table 2 in Table 4. The confidence intervals clearly show that the TOOLFUZZ’s Auto Fix (TF-AF) and DRAFT are able to improve the performance of the tools. Further the confidence intervals again show that TOOLFUZZ is slightly better than DRAFT.

F.2 GitHub toolkit benchmark

The GitHub toolkit benchmark consists of 18 Comment Issues tasks, 9 Update File tasks, 2 Create Pull Requests tasks, 6 Create File tasks, 9 delete file tasks and 10 Create branch tasks. For each of these tasks, we have a script that reverts the state of the repository to the initial states and a script which verifies the success of the tasks.

Again we have computed the binomial confidence intervals for the results from Table 2 in Table 4. The confidence intervals clearly show that the TOOLFUZZ’s Auto Fix (TF-AF) and DRAFT are able to improve the performance of the tools. Further the confidence intervals again show that TOOLFUZZ is better than DRAFT.

G Manual labeling

The runtime failure detection cannot suffer from false positives as the tool is either throwing a runtime error or not. However for correctness detection, there is no ground truth or deterministic way to evaluate the correctness of the tool. That is why TOOLFUZZ uses both an LLM evaluation and

Input/Output consistency checks. In order to evaluate our approach we need to acquire a ground truth. For this we had to manually label the correctness of the tool invocations. The labeling process was done by the authors of this work. The labeling process was done in two steps. As manual labeling is labourous we had to subsample from the full run. In the case of the baselines we sampled 10 prompts per tool, while in the case of TOOLFUZZ we sampled 10 prompt sets per tool as effectively one prompt set corresponds to one error in the case of TOOLFUZZ and one prompt corresponds to one error for the baseline. To remove bias in while labeling only the generated user query, the tool input and output, and the agent response are visible to the labeler. Then the labeler labels the trace as erroneous or not.

H Additional evaluation

To accompany the evaluation presented in the main paper, we have conducted additional experiments to further evaluate the effectiveness of TOOLFUZZ.

H.1 LLM’s impact on TOOLFUZZ

Table 5: Number of unique runtime errors using different LLMs for the agent and the TOOLFUZZ’s prompt generation. Rows - Agent LLMs, Columns TOOLFUZZ’s prompt LLMs.

	GPT-4o-mini	GPT-4o	Claude 3.5 Haiku	Gemini-2.0-flash
GPT-4o-mini	167	165	166	154
GPT-4o	143	142	123	131
Claude 3.5 Haiku	93	117	99	109
Gemini-2.0-flash	83	90	81	106

Table 6: Evaluating unique error finding performance with varying LLM temperature for prompt generation

	t=0	t=0.4	t=0.8	t= 1
Langchain	84	82	98	90
Composio	78	61	69	56

In this section, we present the results of the evaluation of TOOLFUZZ using different LLMs. The results are shown in Table 5. The table shows that results vary depending on the used model for both the agent and TOOLFUZZ, however always finding errors. This indicates that TOOLFUZZ is robust and can be used with different LLMs without significant impact on its performance. Further since LLMs are black boxes and many of the tools are using closed sourced APIs in many cases they cannot be changed which leaves space for only tool description optimization.

Further we investigate the impact of different model parameters on the performance of TOOLFUZZ. The results are shown in Table 6. The table shows that a lower temperature works better for well crafted tools like the ones from Composio [7], as they are more strict and would reject more creative prompts. While a higher temperature leads to more diverse and creative prompts which can showcase a lot of weaknesses in weakly defined tools like the ones from LangChain [4].

H.2 Statistical significance

Table 7: Binomial Proportional Confidence Intervals for the FDR (%) for Table 1

	Baseline W	Baseline G	TOOLFUZZ CC	TOOLFUZZ LO	TOOLFUZZ
Langchain	$0.676_{\pm 0.075}$	$0.845_{\pm 0.038}$	$0.426_{\pm 0.057}$	$0.364_{\pm 0.053}$	$0.153_{\pm 0.05}$
Composio	$0.805_{\pm 0.035}$	$0.789_{\pm 0.051}$	$0.483_{\pm 0.053}$	$0.472_{\pm 0.050}$	$0.374_{\pm 0.066}$

Table 8: Total unique runtime errors TOOLFUZZ can find estimated by the standard Chao1 estimator - 95% confidence interval for: ReACT (RA), ReACT after DRAFT (RA-D), OpenAI Functions (OAF) and Function Calling (FC)

	RA	RA-D	OAF	FC
Langchain	[51 - 54]	[28 - 36]	[50 - 57]	[58 - 72]
Composio	[61 - 84]	[89 - 175]	[54 - 77]	[64 - 108]

In this section, we present the confidence intervals (Table 7) for the results present for the correctness evaluation in Table 1. It can be seen from the table that TOOLFUZZ outperforms the baselines in all cases, even considering the lower bound of the confidence interval with the upper bound of the baseline.

To show the power of TOOLFUZZ to find unique errors, we present the results of the unique errors found by TOOLFUZZ and DRAFT in Table 3. We can see that TOOLFUZZ is able to find unique errors regardless of the agent used and also on top of DRAFT’s improved tool descriptions. To further strengthen this claim, we present the number of total unique errors estimated by the standard Chao1 estimator [3, 6] and report its 95% confidence interval (TUE-CI). The results are shown in Table 8. The table again shows that regardless of the agent or DRAFT’s fix there are still unique errors to be found.

H.3 Cross tool calling

We have used prompts generated from TOOLFUZZ to assess whether tool documentations are too broad, leading to unintended activations across different categories. As discussed in §2, under-specification can prompt undesired tool invocations. We categorized tools by their domain and sampled prompts intended for different tool groups. Our experiment revealed that 492 prompts led to unplanned tool usages across 53 tools in langchain. Detailed results can be found in the experiment folder within the source code (./src/eval/cross_tool_calling). The tool categories are presented in the paragraph below.

Tools categorization As we wanted to test cross tool calling, we have grouped the tools listed in §E into groups with respect to their domain. The groups are as follows:

1. Dall-E: Dall-E-Image-Generator
2. NASA Image and Video Library: Get Nasa Image and Video Library media metadata manifest, Get NASA Image and Video Library media metadata location, Search NASA Image and Video Library media, Get NASA Image and Video Library video captions location
3. Knowledge Repositories: wikidata, wikipedia, stack_exchange
4. Academic Resources: arxiv, semanticscholar, pubmed
5. Search engines: duckduckgosearch, duckduckgosearchresult, stack_exchange, youtube_search
6. File operations: file_search, list_directories, terminal, python_repl, python_repl_ast
7. File deletions: file_delete, terminal, python_repl, python_repl_ast
8. Move files: move_file, terminal, python_repl, python_repl_ast
9. Read files: read_file, terminal, python_repl, python_repl_ast
10. Copy files: copy_file, terminal, python_repl, python_repl_ast
11. HTTP Requests: request_delete, terminal, python_repl, python_repl_ast, requests_get, requests_patch, requests_post, requests_put
12. Map search: open-street-map-search
13. JSON Operations: json_spec_list_keys, json_spec_get_value
14. Directions: open-street-map-distance
15. GraphQL: query_grapql

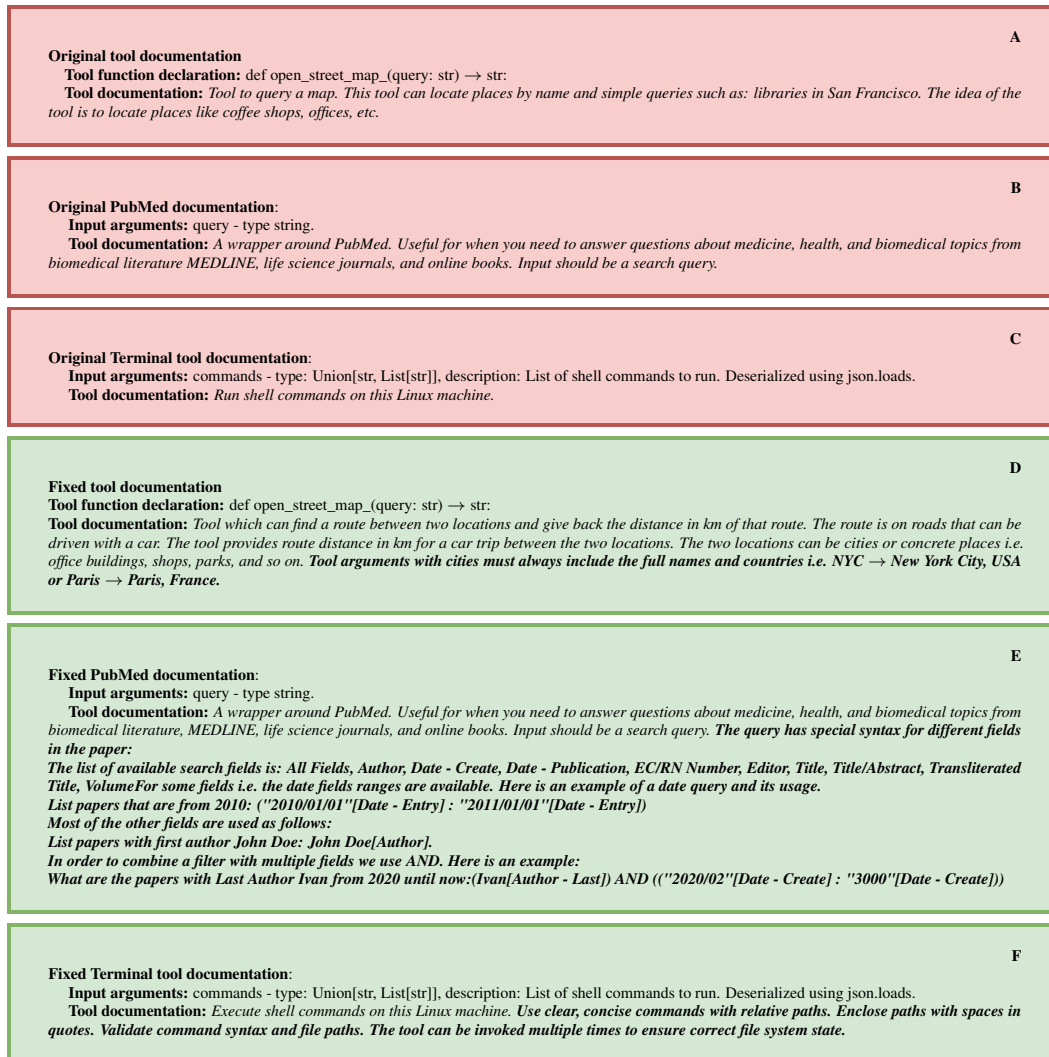


Figure 9: a case study comparing the original and improved descriptions of three tools: open_street_map, pubmed, and terminal. The first set of three red boxes represents the original descriptions of these tools, which are prone to failures, while the second set of three green boxes shows the same tools with improved descriptions designed to reduce errors.

I Case studies

In this section, we present two additional case studies in addition to the one presented in the main article (§5.5). The tools we are looking at are open-street-map-directions, tool for navigating on open street map which is developed by the authors and a LangChain tool terminal which is designed to execute shell commands on the host system.

OpenStreetMap directions Now let's look at an example of the open street map. Here are the prompts in the prompt set \mathbb{P} :

1. What is the car route distance from Paris to Lyon?
2. What is the car route distance from Paris, France to Lyon, France?
3. What is the car route distance from City of Paris to City of Lyon?
4. What is the car route distance from Paris FR to Lyon FR?
5. What is the car route distance from Paris France to Lyon France?

996 For this prompt set the generated expected answer is: *"The road distance between Paris and Lyon is approximately 465 kilometers (289 miles)."*
997

998 After invoking the agent with those prompts, there are 2 output buckets. The first bucket contains:
999 1, 2, 4 and 5 with tool output – *"The distance between City of Paris to Lyon is 939224.1 meters"*,
1000 while the second is just one prompt 3 with output: *"The distance between Paris to Lyon is 465460.3 meters"*.
1001

1002 The input argument buckets are three, grouped as follows:

- 1003 • bucket 1: {1} with parameters:
1004 {from_location_query: "Paris", to_location_query: "Lyon"}
- 1005 • bucket 2: {2, 4, 5} with parameters:
1006 {from_location_query: "Paris, France", to_location_query: "Lyon, France"}
- 1007 • bucket 3: {3} with input parameters:
1008 {from_location_query: "City of Paris", to_location_query: "City of Lyon"}

1009 From the bucketing, we can make the observation that the inputs with full country name work well,
1010 and using just a city name works half the time (input bucket 1 and bucket 3) so one way to fix this is to
1011 add the requirement of country name in the argument to the tool's description. The fixed description
1012 can be found in Figure 9.D. With the fixed description, the example prompt set passes.

1013 **Terminal** The terminal tool has a very short and generic description as shown in Figure 9.C. This
1014 description, however, does not explain at all how to properly use the tool i.e. in what way the
1015 command parameters have to be presented, which commands can be used etc.

1016 Here is an erroneous set of prompts found by TOOLFUZZ:

- 1017 1. How big is the assignment_1_draft.docx file in the Contract Law assignments folder?
- 1018 2. Find the size of the draft of assignment 1 file in the
1019 [/FULL_PATH]/Law_Student_Files/Academics/Courses/Semester_1/... directory.
- 1020 3. What is the file size of the first draft of assignment 1 in the Contract Law assignments folder?
- 1021 4. Find the size of the assignment one draft file in the
1022 [/FULL_PATH]/Law_Student_Files/Academics/Courses/Semester_1/... directory.

1023 The FULL_PATH is in fact the full path; however, we have masked it out as it is part of the machine's
1024 file structure.

1025 The prompt set has the goal of finding the size of the first Assignment in the Contract law assignments
1026 folder. It is important to note that the agent is unaware of the directory in which it is running.

1027 In this case, both the number of input and output buckets is four – each prompt resulted in a unique
1028 input parameters and tool output:

- 1029 1. tool input: {"commands": "du -h
1030 /Contract Law/assignments/assignment_1_draft.docx"}
1031 tool output: cannot access
1032 '/Contract Law/assignments/assignment_1_draft.docx': No such file or
1033 directory
- 1034 2. tool input:
1035 {"commands": "du -sh [/FULL_PATH]/Assignments/assignment_1_draft.*"}
1036 tool output: 0 [/FULL_PATH]/assignment_1_draft.docx
- 1037 3. tool input: {"commands": "ls -lh ~/Contract Law/assignments/ | grep 'assignment
1038 1 draft'"}
1039 tool output: "ls: cannot access '~/Contract Law/': No such file or
1040 directory"
- 1041 4. tool input: {"commands": "ls [FULL_PATH]/Law_Student_Files/Templates"}
1042 tool output: ""


```
Please invoke the {tool.name} exactly like: {tool.name}({bad_arg})
```

Figure 10: Sanity check prompt for forcing the agent to invoke the tool under test with the erroneous arguments.

```
You are an AI assistant tester. The idea is to come up with prompts which will make the following tool produce incorrect answer.  
Tool information:{tool_information}  
Please generate prompts which will make the following tool produce incorrect answer.  
{format_instructions}
```

Figure 11: Prompt for generating realistic user prompt which will invoke the tool under test with erroneous arguments.

1043 As can be seen, each prompt results in unique input parameters and tool output. We can see that twice
1044 in prompts 1 and 3 the tool tries to access invalid paths. For prompt 4 the path is correct; however, the
1045 task at hand is not just to list a directory. The general error here seems to be related to the provided
1046 paths – the description does not give any inside information to the agent on where the Terminal
1047 tool is or how to navigate. With this information found by TOOLFUZZ and using the previously
1048 described autofix method (§5.3) a new description is generated (Figure 9). With this new description,
1049 the prompt set is now valid by just using ls multiple times to properly navigate the file system.

1050 J TOOLFUZZ Prompts

1051 In this subsection of the Appendix all prompts used for TOOLFUZZ are listed. For all the following
1052 prompts, string templates are used – {variable} will be inserted with the corresponding variable
1053 when the prompt is constructed.

1054 J.1 TOOLFUZZ Runtime tool failure detection

1055 For the runtime error detection as explained in §4 the heavy lifting is done by the Fuzzer so an LLM is
1056 leveraged only for sanity check with the prompt Figure 10 and for converting the erroneous argument
1057 into a user query which is done with Figure 11.

1058 J.2 TOOLFUZZ correctness detection

1059 **Prompt set generation** The generation of prompt sets as described in §4.2 involves multiple LLM
1060 generations. Firstly, the prompt template is generated given the tool description and in some cases
1061 additional context, the full prompt is given in Figure 12

1062 The next step is to infill the generated template questions/prompts with synonymous phrases. For this,
1063 the following prompt is used to generate the phrases which are later inserted in the string templates
1064 Figure 13.

1065 As in some cases when giving context to the prompt generation, some facts are taken directly from
1066 the context which is not very human, so we have also introduced a humanizing prompt Figure 14.

1067 J.3 LLM Oracle

1068 The LLM Oracle has to evaluate the Agent answer according to an expectation. Firstly, the expectation
1069 is generated.

1070 The LLM expectation generation is again a multistage process. Firstly, an LLM is prompted to answer
1071 to each of the generated prompts from a set of prompts $p \in \mathbb{P}$ – Figure 15. After all the answers are
1072 generated, we use them to prompt again an LLM to come up with the LLM expectation – Figure 16.

1073 Lastly, the LLM oracle has to evaluate the Agent output, for this prompt we are using both reasoning
1074 and scoring between 1 and 10 as previous experiments showed that making the LLM to just evaluate
1075 with yes/no gave more false positives – Figure 17.

```

You are a user asking an AI assistant for help - just speak naturally.
Your task:
Generate template questions that a user might ask based on a tool's purpose and capabilities.
The tool is described as follows:
{tool_prompt}
With the following context:
{tool_context}
Here are some examples for inspiration:
For a map/distance tool:
What is the distance from [A] to [B]?
How much time would it take to go from [A] to [B]?
If I start from [A] and go to [B] with [C] km/h average speed how much time would it take me?
For a knowledge based or information retrieval tool:
What do you know about [A]?
What is/are [A] for [B]?
Is it true that [A] is [B]?
Is [A] related to [B]?
In what year did [A] happen?
When was [A] born?
Find [A] in [B]?
[A] my work to [B].
[A] from [B] to [C].
Also some more specific questions like:
Is it true that Mr [A] was related to Mrs. [B]?
I am at [A], how much time it will take me to go from the closest [B] to the [C] airport?
Find an article/paper/document written by [A] on topic [B]?
Use just words within the placeholder brackets i.e. [A], [B], [C], [destination], [source], [topic], [place],
[location], [person], [company], [organization], [event], [date] etc. Avoid adding any special characters or
punctuation marks AVOID [locations_rating] or [bracket_selector[]].
Be creative, and make sure the templates match how people might actually talk to an AI assistant which has this
tool i.e. both specific templates and more general ones.

{format_instructions}.

```

Figure 12: Prompt for generating template questions/prompts for the agent, given the tool under test description and additional context if needed.

1076 J.4 Baseline prompts

1077 For the baselines from §5.2. We use the following prompt to generate test prompts - Figure 18 and
 1078 Figure 19. Here, the main difference is that in the white box scenario the `tool_info` variable will
 1079 contain both the tool description and the tool source code while in the gray box the tool source code
 1080 is not included.

1081 In addition to these prompts, the baseline also has their own LLM judge using the following prompt
 1082 for evaluation – Figure 20.

1083 As mentioned in §5 the baselines can fall into degenerate string generation - examples of such cases
 1084 can be seen in Figure 21.

You are a user interacting with an AI assistant—just speak naturally.
Your task:
For the following template question:
`{template_prompt}`
Generate infills for each placeholder (like [A], [B], etc.).
Each infill should be a **specific example** that could realistically fill that blank — such as named entities, objects, or well-known terms. These should not be rephrasings of the whole sentence or template. Infills should reflect how real users might refer to the same thing using:

- Synonyms
- Abbreviations
- Rephrasings
- Alternate spellings
- Common names or titles

Use only inputs that are **relevant** to this tool:
`{tool_prompt}`
And the tool's background or purpose given the following context: `{tool_context}`
Examples:

Template: 'What are some [A] in [B]?'
Infills for A: ['Coffee Shop', 'Cafeteria', 'Coffeehouse', 'Café']
Infills for B: ['Zurich', 'ZH', 'Zurich Switzerland', 'ZH CH', 'ZH Switzerland', 'Zurich CH']
Template: 'Who is [A]?'
Infills for A: ['Albert Einstein', 'A. Einstein', 'Alb. Einstein', 'Einstein']
Template: 'When did [A] happen?'
Infills for A: ['World War I', 'World War One', 'WW 1', 'First World War']
Template: 'What are the latest news in [A]?'
Infills for A: ['cinema', 'hollywood', 'kino', 'movies', 'show business']
Template: 'What are the [A] in [B]?'
Infills for A: ['latest news', 'current events', 'breaking news', 'daily news', 'daily events']
Infills for B: ['politics', 'government', 'public affairs']
Template: 'I am at [A], how much time it will take me to go from the closest [B] to the [C]?'
Infills for A: ['Zurich HB', 'Zurich main train station', 'Zurich main station']
Infills for B: ['Mc Donalds', 'fast food restaurant McDonalds', 'McD burgers']
Infills for C: ['ETH HG Bibliothek', 'ETH main building library', 'ETH main library']
Template: 'Can you find [A] in [B]?'
Infills for A: ['family picture', 'png with the family', 'family photo', 'family portrait']
Infills for B: ['the home directory', 'my workspace', 'main directory']
Template: '[A] [B] to [C]'
Infills for A: ['Submit', 'Send', 'Upload', 'Commit']
Infills for B: ['main.py', 'the main python file', 'src/main', 'the main source file']
Infills for C: ['the server', 'the cloud', 'the repository', 'the remote branch']
Template: '[A] my work to [B]'
Infills for A: ['Move', 'Transfer', 'Cut']
Infills for B: ['archive folder', 'the archive']
INVALID EXAMPLES:

Template: 'What are the side effects of [medication] according to recent studies?'
Infills for A: ['medication', 'this medicine', 'this therapeutic']
Important:

Do **NOT** reuse any examples from the following list: `{used_args}`.
`{format_instructions}`.

Figure 13: Prompt for generating infills for the masks in the prompt templates.

Given the following tool description: `{tool_prompt}` and the following tool prompts that are synonymous: `{prompts}` Please make such that the prompts are like a person would write it and not a machine, so nothing too concrete but also not too vague.
`{format_instructions}`

Figure 14: Prompt for making more human like prompts. In some cases prompts are too specific i.e. full paths or full identifiers which users might shorten or write more intuitively.

You are emulating the following tool: {tool_prompt}. Given the tool return value for the following questions: {questions}

Example:

Tool description: Tool which can find a route between two locations and give back the distance in km of that route. The route is on roads that can be driven with car. The tool provides route distance in km for car trip between the two locations.

The two locations can be cities or concrete places i.e. office buildings, shops, parks and so on.

Questions:

What is the distance between Sofia and Zurich?

What is the distance between SF and ZH?

What is the distance between Sofia BG and Zurich CH?

What is the distance between Sofia Bulgaria and Zurich Switzerland?

Answers:

The road distance between Sofia, Bulgaria, and Zurich, Switzerland is approximately 1,450 kilometers (900 miles).

If "SF" refers to San Francisco (SF), California, USA and "ZH" refers to Zurich (ZH), Switzerland, the distance is vast. Since it's impossible to drive directly due to the Atlantic Ocean, the driving distance would be irrelevant. However, hypothetically driving the distance across land would exceed 9,500 kilometers (5,900 miles).

"BG" stands for Bulgaria and "CH" stands for Switzerland. The road distance between Sofia, Bulgaria, and Zurich, Switzerland is approximately 1,450 kilometers (900 miles)

The road distance between Sofia, Bulgaria, and Zurich, Switzerland is about 1,450 kilometers (900 miles).

{format_instructions}.

Figure 15: Prompt for generating answers for given question. Used for expectation generation for the LLM Oracle.

You have the following sentences: {sentences}

Make once sentence which is factually correct to you and most of the sentences, not required to represent all of the answer but MUST be approximately correct.

Examples:

Sentences:

The road distance between Sofia, Bulgaria, and Zurich, Switzerland is approximately 1,450 kilometers (900 miles).

If "SF" refers to San Francisco (SF), California, USA and "ZH" refers to Zurich (ZH), Switzerland, the distance is vast. Since it's impossible to drive directly due to the Atlantic Ocean, the driving distance would be irrelevant. However, hypothetically driving the distance across land would exceed 9,500 kilometers (5,900 miles).

"BG" stands for Bulgaria and "CH" stands for Switzerland. The road distance between Sofia, Bulgaria, and Zurich, Switzerland is approximately 1,450 kilometers (900 miles)

The road distance between Sofia, Bulgaria, and Zurich, Switzerland is about 1,450 kilometers (900 miles).

Answer: The expected distance is around 1500 kilometers. {format_instructions}.

Figure 16: Prompt for generating the LLM Expectation for particular tool set \mathbb{P}

You are assessing an agent RESPONSE. You should give a score between 1 and 10. 10 means that the answer satisfies both criteria: first is the tool output and second is LLM expectation. The agent response should be within 10% margin of error or false otherwise. 0 is for when both criteria are not satisfied. Alongside the evaluation of true/false, you should provide reasoning for it.

Use the following examples as reference:

#Example 1:
Agent response: The capital city of France is Paris
Expectations:
1. No results found for the query
2. Paris
correctness_degree: 5
Reason: The response doesn't comply with the first expectation: 'No results found for the query'

#Example 2:
Agent response: As of now, the current survey results for the 2024 US presidential election show Kamala Harris leading Donald Trump. Here are the key figures:
- **Kamala Harris**: 46%
- **Donald Trump**: 48%
- **Robert F. Kennedy Jr.**: 5.5%

Harris's popularity has seen a significant increase, with half of voters viewing her favorably, which is an improvement from 43% the previous week.
Expectations:
1. Kamala Harris, Donald Trump.
2. The current survey results for the 2024 US presidential election show candidate A leading with X%, followed by candidate B with Y%, and other candidates with Z% and W%.
correctness_degree: 9
Reason: The response mentions both Kamala Harris and Donald Trump, thus complying with the first expectation, and also presents percentages, thus complying with the second.

#Example 3:
Agent response: If you're looking for a solid, mid-range option, expect to spend around \$400 to \$500, there is one cheaper option around 350 USD
Expectation:
1. \$347.99
2. A TV sells in the range of 400-800 USD
correctness_degree: 8
Reason: The response is close enough to the expected values and is based on the expected tool output.

#Example 4:
Agent response: The distance between LA and Sofia is 1,047 miles
Expectation:
1. Distance from Sofia to LA is 10,633 km
2. The distance between Sofia and LA is 10,000 KM
correctness_degree: 2
Reason: The response doesn't comply with both expectations - the error is too big, but is still mentioning the correct names.

Now please evaluate the following agent response and give the reason for your evaluation:
'{agent_output}'
Expectations:
1. {tool_output}
2. {expected}
{format_instructions}

Figure 17: Prompt used by the LLM Oracle to evaluate the correctness of an agent answer given the tool and expectation as well as the output of the tool under test

You are an AI Agent Tool developer can you make the description of a tool more precise and clear. The problem is that the LLM sometimes doesn't translate the queries correctly.
This comes from the fact that the tool description doesn't explain what exactly is allowed and how to use the tool correctly.
The current description of the tool is: {tool_description}

Here are some failing examples of the tool in action:
{bad_examples}

Given the bad examples please identify the main issues on those examples and what is the cause of that issue. How can these issues be avoided by validation i.e. with this tool or external resources or it's just a user mistake? List the main issues and how to avoid them.

Now that the issues and how to avoid them are clear. Please create tool description that addresses these issues. The description is a manual on how to use the tool correctly and what is allowed and what is not. It should explain how to avoid the issues that were found in the examples. Add that the tool can be invoked multiple times for better validation of the file system state.

Also give few examples if you think they are applicable.

Please provide description which reflects these issues. The new description shouldn't be longer than 100 words.

{format_instructions}

Figure 22: Prompt for automatic tool description fixing, based on the tool's description and as well as a set of prompts resulting into tool failures (bad_examples) found by TOOLFUZZ.

You are an AI Agent Tool developer can you make the description of a tool more precise and clear. The problem is that the LLM sometimes doesn't translate the queries correctly.
This comes from the fact that the tool description doesn't explain what exactly is allowed and how to use the tool correctly.
The current description of the tool is: {tool_description}

The description has to be a manual on how to use the tool correctly and what is allowed and what is not. Please provide just the new description of the tool.

{format_instructions}

Figure 23: Prompt for automatic tool description fixing, based only on the tool description.

You are an AI Agent Tool developer can you make the description of a tool more precise and clear. The problem is that the LLM sometimes doesn't translate the queries correctly. This comes from the fact that the tool description doesn't explain what exactly is allowed and how to use the tool correctly.

The current description of the tool is: {tool_description}

Here are some failing examples of the tool in action:
{bad_examples}

Given the bad examples please identify the main issues on those examples and what is the cause of that issue. How can these issues be avoided by validation i.e. with this tool or external resources or it's just a user mistake? List the main issues and how to avoid them.

Now that the issues and how to avoid them are clear. Please create tool description that addresses these issues. The description is a manual on how to use the tool correctly and what is allowed and what is not. It should explain how to avoid the issues that were found in the examples. Add that the tool can be invoked multiple times for better validation of the file system state.

Also give few examples if you think they are applicable.

Please provide description which reflects these issues. The new description shouldn't be longer than 100 words.

{format_instructions}

Figure 24: Prompt for automatic tool description fixing, based on both the tool's description and tool's source code.