THE PENNSYLVANIA STATE UNIVERSITY
SCHREYER HONORS COLLEGE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Curating the LOGIFALLA Dataset using Mephisto and MTurk

REUBEN W LEE
SPRING 2023

A thesis
submitted in partial fulfillment
of the requirements
for a baccalaureate degree
in Computer Science
with honors in Computer Science

Reviewed and approved* by the following:

Ting-Hao ‘Kenneth’ Huang
Assistant Professor of IST
Thesis Supervisor

David Koslicki
Associate Professor of EECS
Honors Advisor

* Electronic approvals are on file.

ABSTRACT

Numerous datasets have been created to aid AI systems in detecting falsehoods, fabricated news, offensive language, and provocative posts in digital discussions [6]. However, despite the significance of these efforts, the identification and interpretation of the fundamental logical fallacies that undermine the rational soundness of such content have yet to be explored [6]. A significant obstacle to this exploration is the absence of standardized benchmark datasets, as the process of annotating logical fallacies poses a considerable challenge. In this paper we provide a system of techniques for curating the first public benchmark dataset for common logical fallacies in online conversation, which we will call: LOGIFALLA. The system of techniques presented in this paper can be used to curate not only the LOGIFALLA dataset but can be replicated for future dataset curation by the NLP community. This system comprises of Meta's crowdsourcing tool, Mephisto, Amazon's crowd sourcing platform, Mechanical Turk, and a simulated social media platform tool, the (Mis)Information Game, all of which are publicly available for research use [4].

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to thank Professor Ting-Hao ‘Kenneth’ Huang for giving me the opportunity to work and conduct research in the Crowd-AI Lab. Without his guidance and the support of the graduate students in his lab, Chieh-Yang Huang, Ting-Yao ‘Edward’ Hsu, Hua Shen, and Alan Huang, I would not have been able to get this wonderful experience.

Additionally, I would like to thank my co-researcher Phakphum ‘Peter’ Artkaew as we spent a lot of time together working on different parts of this project and have supported each other throughout the entire process.

I would also like to thank Professor David Koslicki, my honors advisor, as he has provided much guidance throughout my experience in the Honors College and helped me figure out my graduation requirements so I can graduate on time as a Schreyer Scholar.

Finally, I would like to thank our contacts at Meta, Jack Urbanek and Pratik Ringshia. Without them we would not have been introduced to the tool, Mephisto, and would not have had the funding for this project. Additionally, ‘The MisInformation Game’, a tool used in this project, was recommended by them as well.

Chapter 1

Introduction

As we observe the increasing importance of social media platforms like Reddit and Twitter on determining political outcomes, elections, and the economy of the world the research community have been conducting numerous research projects with the goal of preventing the misuse of these tools [6]. Traditionally the detection of misinformation, fake news, and hate speech have been the main focus by the research community [6]. However, we propose that while these works are important, the detection and comprehension of the underlying logical fallacies that fundamentally undermine these texts' logical validity will have a more direct contribution to encouraging efficient and fact-based discourse on online communities [6]. For this goal we determined that there are three crucial pieces that must exist.

1. The existence of a powerful classifier able to detect logical fallacies in common language.
2. A database containing logical fallacies based on natural language that online communities use. This can be used to train the classifier.
3. A system of techniques utilizing modern crowdsourcing platforms to curate the database while maintaining the quality of the data.

Let us start with the first piece, the classifier. Although the area of logical fallacy detection in online conversations is less explored than that of hate speech or fake news, there are

existing research done on detecting logical fallacy statements [1]. In the research conducted at the University of Michigan by Zhijing Jin in conjunction with other institutions, the task of logical fallacy detection was accomplished using a simple structure-aware classifier which ended up outperforming large language models by 5.46% [1]. However, their classifier used the LOGIC dataset which is made of general logical fallacies statements found in formal texts like English textbooks [1].

Although these are correct logical fallacies, they are very formal and do not resemble the kind of logical fallacies used in human conversation. This is evident in the F1 value of the struct-aware model classifier being 58.77% which is significantly higher than all the other existing large language models which range from 12.50% to 53.31% [1]. However, we believe that this can be much higher if a database existed that contains a collection of logical fallacies based on natural language that everyday humans use rather than English professionals. We decided to call this dataset LOGIFALLA.

However, to develop this dataset we need a system of techniques to curate the dataset in the first place. For this system we need to ensure that the data that we collect is high quality and organized into a form that can easily be used to train the classifier [2]. The core idea is to hire a group of crowd workers to simulate an online discussion thread about a given piece of news. As these crowd workers make comments and respond to each other we will instruct a subset of them to respond using specified types of logical fallacies. These threads containing all the comments made by the workers will be annotated with the logical fallacy label (Yes/No) and fallacy type (if Yes). The original post containing the topic will also be included.

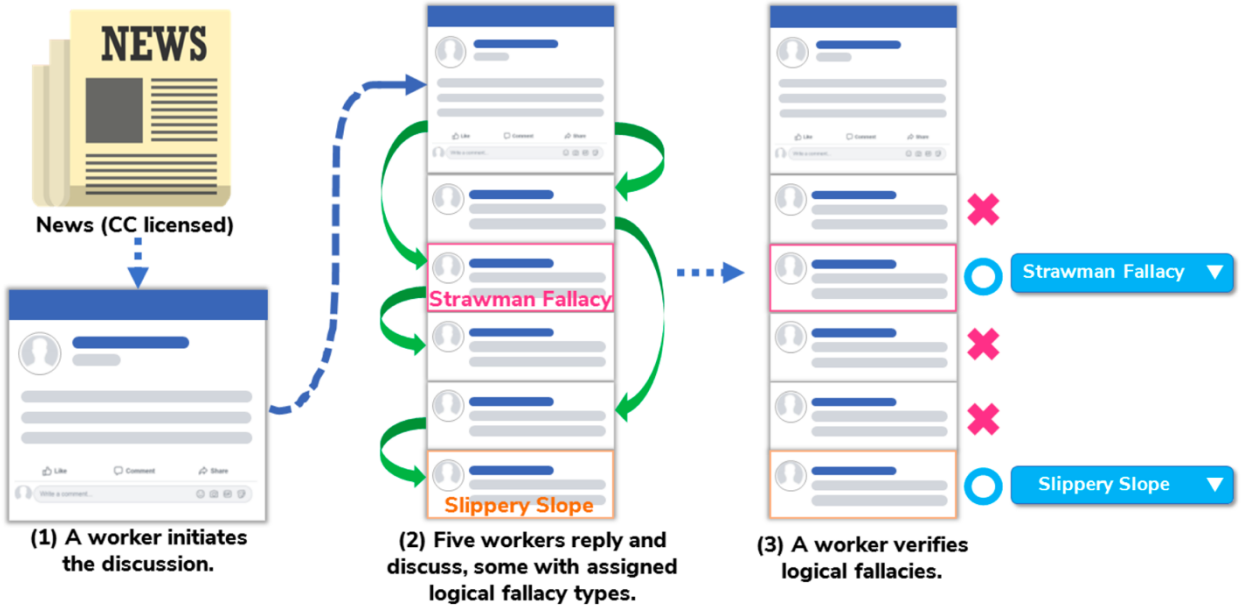

Figure 1. Annotation process of a discussion Thread in LOGIFALLA

Figure 1 shows a diagram of the process of how the data inside the database will be structured. Starting with the news post and all the responses the worker makes to the post and to each other. Among these responses logical fallacies like “Strawman Fallacy” or “Slippery Slope” will be contained. We will also have a set of crowd workers verifying that the logical fallacy created by another worker in a post truly is a correct logical fallacy of the right type. In this way, all the annotations will be generated for the dataset and its quality will be cross verified by the workers themselves.

To actualize this system, we use Amazon’s Mechanical Turk platform to acquire the crowd workers. We create a simulated social media environment using The MisInformation Game tool. Finally, we connect these together and manage the cross-validation process with Meta’s Mephisto tool. The next chapter will go in depth into how each of these tools are used.

Chapter 2

Tools

As mentioned in the previous chapter, the system of techniques for curating the LOGIFALLA dataset are comprised of three major tools. In Table 1 the three parts of the Dataset Curation System are presented along with their author, costs, and their use.

Table 1. Tools used in Dataset Curation System

Name	*Author*	*Cost*	*Purpose*
Mephisto.ai	Meta Platforms, Inc	Free for Research Use	Management of the assignment of tasks
MTurk	Amazon.com, Inc	Pay per Job Done	Crowd Sourcing platform
MisInformation Game	Uni. Western Australia	Free for Research Use	Web app hosting the simulated social media content

In this chapter each tool will be explored in depth and provide a foundational understanding for building the system in the next chapter.

Sub Chapter 2.1

Mephisto

Mephisto is a framework for portable, reproducible, and iterative crowdsourcing provided by Meta [2]. Primarily built by its author Jack Urbanek and supported by Pratik Ringshia, this tool serves as the backbone of our Data Curation System. Mephisto enables us to create and open-source data collection and annotation tools as part of our publication [2]. This is an important role that had been unfulfilled until Mephisto was developed. An audit of

PapersWithCode, a public repository of ML publications and datasets in November 2022, of the top 35 cited datasets only 18 papers were accompanied with usage code and only 3 provided code for the collection and quality assurance portion of the paper [2]. This is a startlingly small ratio which may perhaps be due to the fact that many of these popular datasets were developed decades ago when much of the data curation was done manually by hand. However, with new methods of data curation being automated and heavily reliant on development code, it becomes necessary to document this process.

In fact, the modeling side of ML research has already been documenting code implementations for many years thanks to frameworks like TensorFlow and PyTorch that enables researchers to efficiently manage and run their development. It only makes sense that the dataset generation part of ML research follows suit [2]. Afterall, according to Reynold Xin, co-founder and Chief Architect at Databricks, “A ML model is only as good as the data it is trained on”.

The most important benefit that Mephisto provides our project is a central code base to manage and update iterative tasks. Mephisto abstracts away many of the processes that handles communications with the Mechanical Turk platform. Additionally, they allow us to set up our assignments to get the workers to cross validate the work done by other workers, thus allowing us to curate massive amounts of quality data in a relatively short span of time. Shown in figure 2 is the Mephisto workflow.

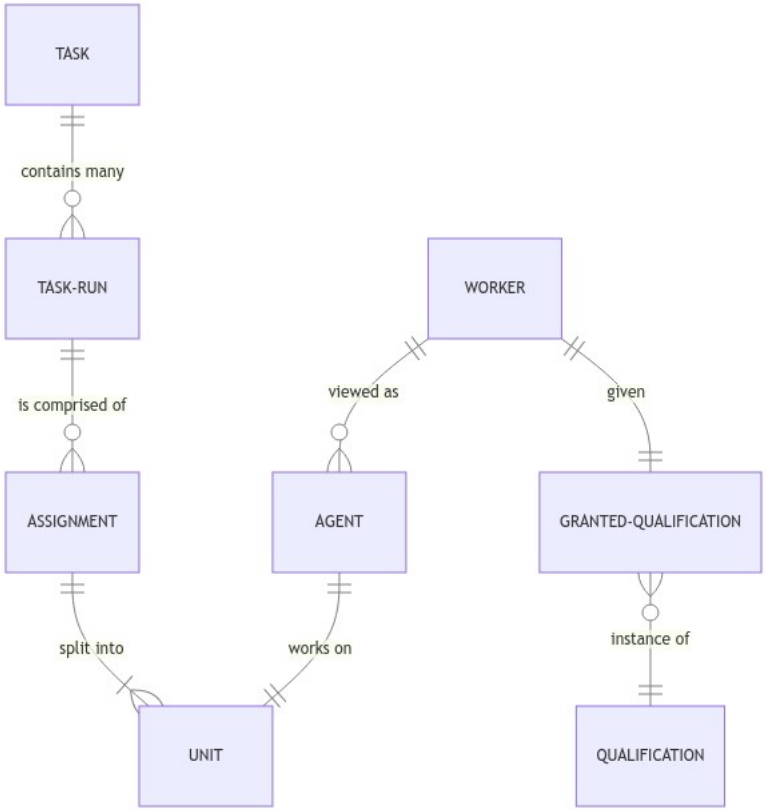

Figure 2. Workflow using Mephisto

The workflow consists of a task that I design using React that works as a template for any iterations and variations of the task I want to give. Then, this can be modified for the specific task-run which can also be separated into multiple assignments for different crowd workers. For example, this means that I can make a task containing a topic and inputs for the users to put their responses in. This can be run as a task for a worker responding to a news post, or a task responding to another worker, or a task validating the response of another worker. This can then be assigned to 100 different workers and we will be able to get 100 different pieces of data that can be put together to guarantee the quality of our dataset while also being extremely efficient and fast.

Sub Chapter 2.2

Amazon Mechanical Turk

Amazon's Mechanical Turk (MTurk) platform is the most popular crowdsourcing marketplace as of 2023. MTurk makes it simple for researchers to collect human intelligence tasks completed by real people by posting them as micro-jobs that can be completed for small amounts of payments [4]. At the Crowd-AI Lab we ensure to pay our workers above minimum wage and scale our payments according to a $10/hour rate.

MTurk provides two different account logins, one for requesters and one for workers. In order to post tasks for crowd workers to complete a requester account is needed. This should be fairly simple to get by following the sign-up instructions on requester.mturk.com. You will need to enter payment information so that you can pay the workers who complete your tasks. The worker account involves needing to confirm your employment status since you can get paid for performing the tasks posted by other requesters. For our purposes the worker account is not needed [5].

Additionally, MTurk provides Sandbox mode for both requesters and workers. This is really useful to test your tasks before sending them to the actual marketplace where you will be paying out of pocket for any tasks that are completed even if they are tasks that are not ready [5]. In figure 3 we present how an MTurk worker will view a HIT (human intelligence task). In the figure the 'reward' is how much the worker will be pad for completing the task and the worker may also 'preview' the task before accepting it to see if its worth doing. Finally, if they choose 'accept & work', they will be presented with the task you created, and they will be paid once it is completed.

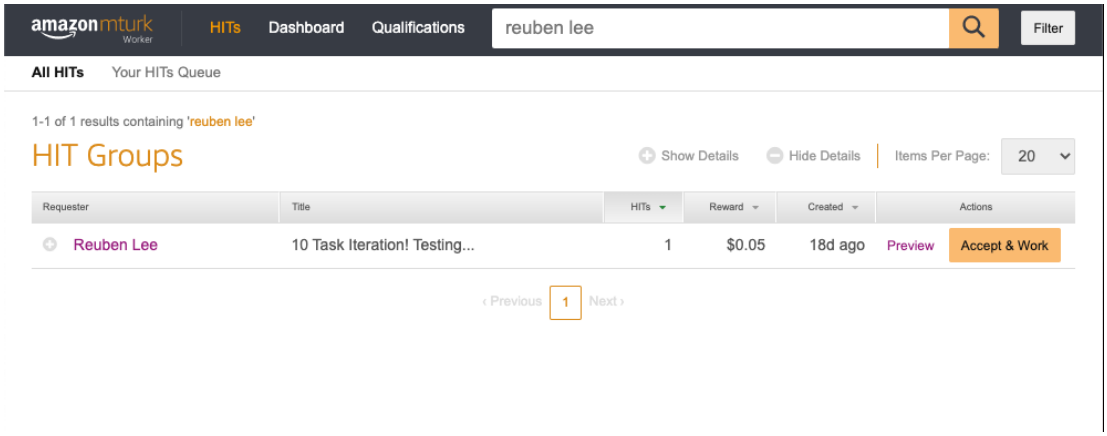

Figure 3. MTurk worker interface

The posting of the tasks and results are all managed and saved by the Mephisto code base which we will explore in depth in the next chapter. Another thing that is very useful when working with MTurk is the API [5]. For the scope of this project, it is not necessary to learn this as Amazon provides a website that allows you to manage the tasks you hosted with a GUI. This website is located at the

URL: https://manage-hits-individually.s3.amazonaws.com/v4.0/index.html#/credentials

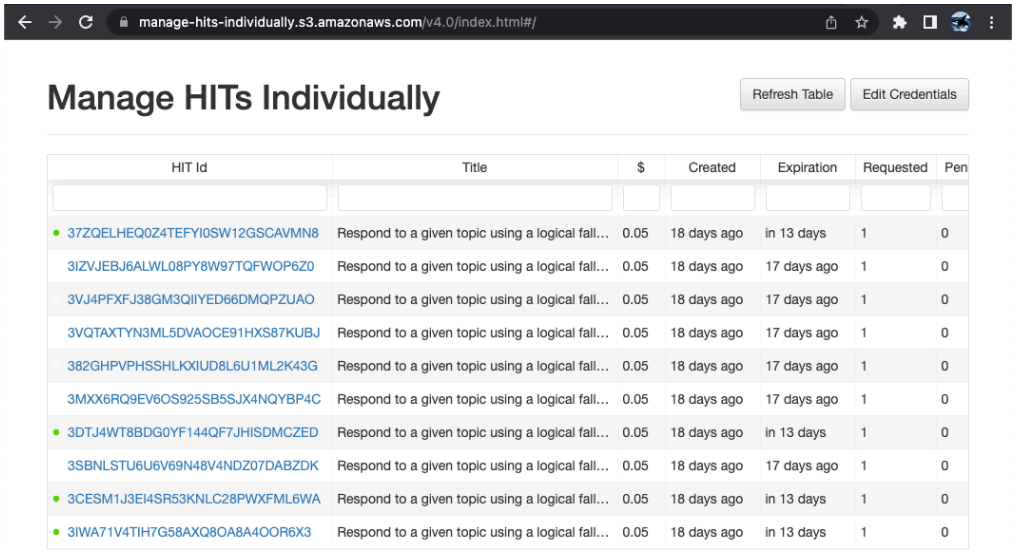

Figure 4. Amazon's Web App for MTurk task management

Sub-Chapter 2.3

MisInformation Game

The MisInformation Game is a web app made for the purpose of conducting research into how people process (mis)information on social-media platforms. The goal of the project was to replace relying on survey software and questionnaire-based measures with simulating the social media experience itself for the participants. The MisInfo Game allows researchers to customize posts, headlines, images, source information, engagement information, and responses of other participants [3].

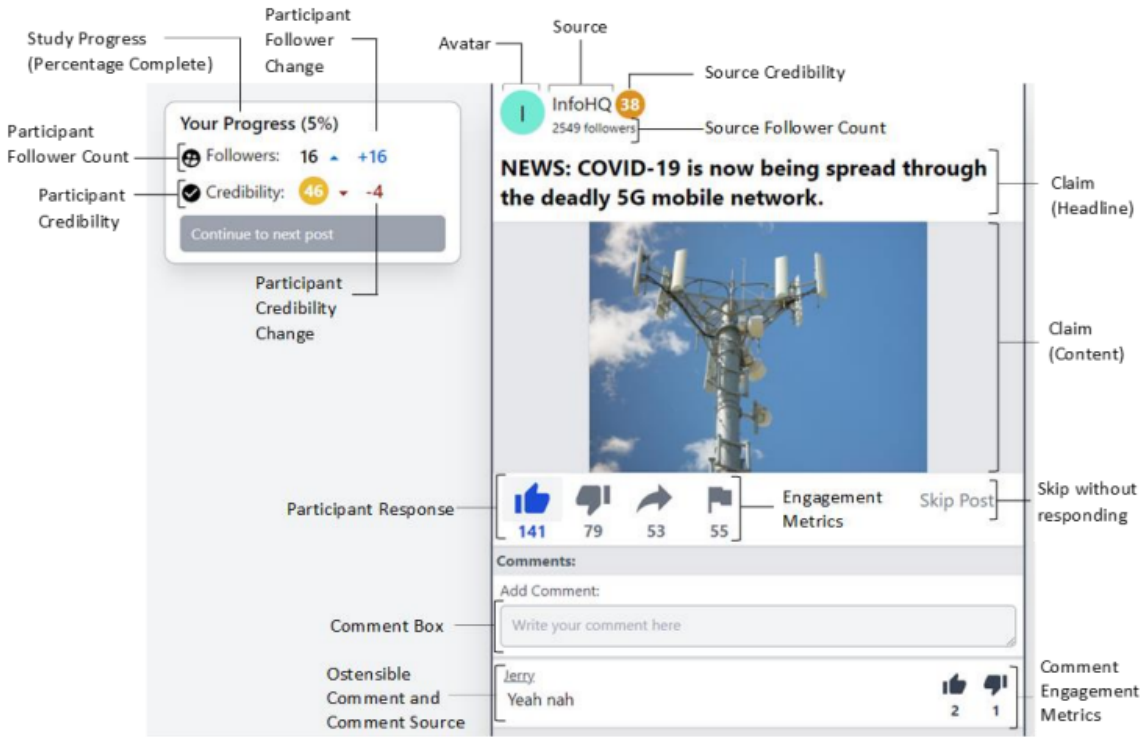

Figure 5. Snapshot of the MisInfo Game's participant interface

In this project we use this tool to present headlines and images from our selected pool of news bits and also display the comments made by other crowd workers.

Chapter 3

Documentation Guide

The three main tools used in the Data Curation System has been covered, however, the full extent of the system includes other inner tools that were necessary to make this system work. In the architecture diagram in Figure 6, we present the full layout of the system in detail. In this chapter, we will walk through the process of building this system so that it may be replicated by future research teams.

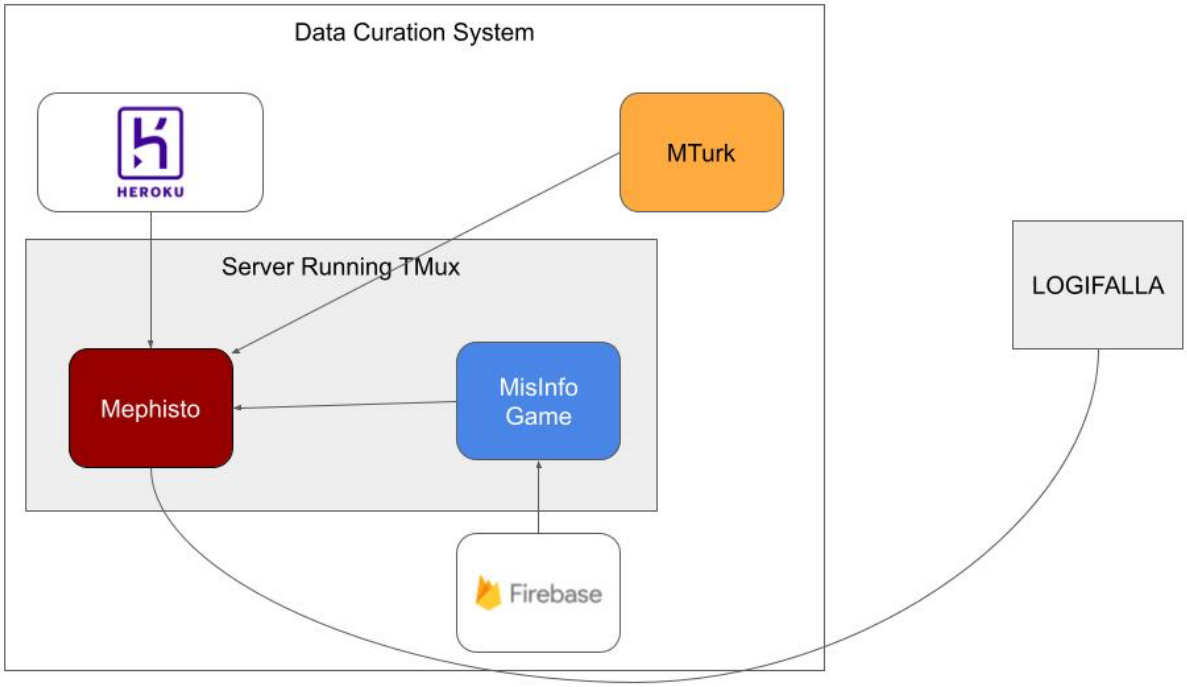

Figure 6. Architecture diagram of Data Curation System

The first thing that needs to be done is to setup Mephisto on the machine that the tasks are going to be run on. Mephisto runs need to continue running on the machine until it finishes and closes itself. This happens when all the assignments are completed by the crowd workers, the time limit on run expires, or the user manually shuts down the run with ‘Ctrl’ + ‘c’.

Sub-Chapter 3.1

Setting up Mephisto on the Host Machine

Due to the nature of the scripts in Mephisto to be constantly running to keep the tasks up, it is recommended that the researcher use a server with tmux (terminal multiplexer that allows terminal sessions to continue running even when you close the terminal). If this option is not available the researcher may follow the steps using their personal machine, just keep in mind that the terminal session needs to be kept open for the duration of the runs which may take many hours to days depending on the number of assignments and availability of the eligible crowd workers.

Virtual Machine Access / Setup

Specifically, to access the Crowd-AI Lab server confirm with the professor that an account for you have been set up on the lab's virtual machine and setup your ssh config file in VSCode with the code in Figure 7.

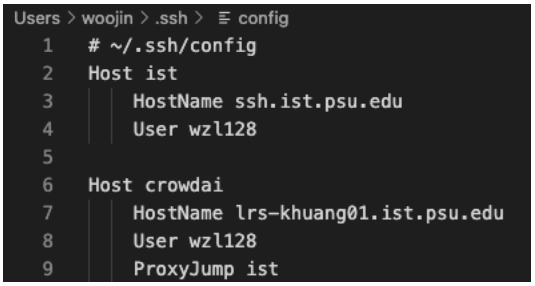

Figure 7. ssh config file

You can access the config file as shown in figure 8 by clicking the green button on the lower left end of the VSCode window then clicking "Connect to Host", then clicking "Configure SSH Hosts", then clicking the option that looks like '/Users/…/.ssh/config.

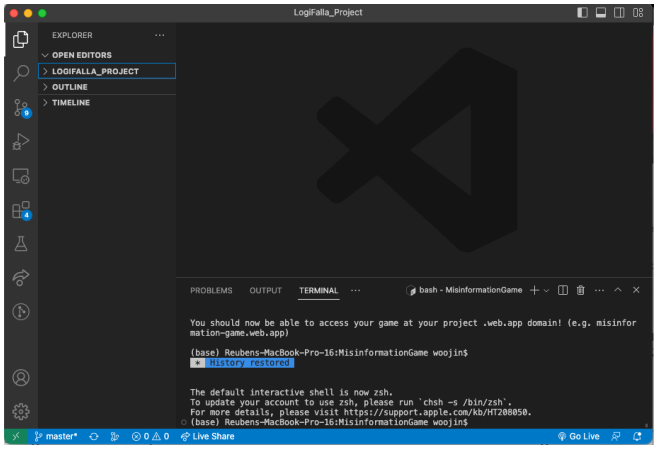

Figure 8. VSCode ssh options button

After doing these steps relaunch VSCode and press the green ssh button again (lower left end of screen). This time you will see ‘ist’ and ‘crowdai’ as options. Select ‘crowdai’ to ssh into the virtual machine. You will need to type in your Penn State password twice when asked by the server.

It is recommended that you set up a python environment for managing your python versions. In the Virtual Machine specifically, we must use Anaconda to setup the python environment. For the Crowd-AI Specific Lab Documentation you can follow the guide here: https://github.com/appleternity/Crowd-AI-Lab-Documentaion

Additionally, we need npm to run our Mephisto tasks. Anaconda provides npm through conda-forge. If you are using your local machine this is not needed since you can directly install npm itself. However, on the virtual machine we do not have sudo access, thus we must use this method to enable npm.

- Check out this site to understand what conda-forge is and how we can get nodejs with it: https://anaconda.org/conda-forge/nodejs
- Additionally you may run into issues about ‘Node-gyp’ failing to build due to the node version installed through conda-forge being an old version. If you are getting this issue

you need to upgrade icu which will allow us to use a newer version of nodejs. Perform these commands:

- conda install -c anaconda icu
- conda install -c conda-forge "nodejs>=14.0"

Steps from here are the same from either the virtual machine or local device.

- Note: we recommend you clone a copy of your repo for your virtual machine and your local machine and do most of the designing on your local machine as the virtual machine may not allow launching local websites.

1. Clone this repo: https://github.com/reubenwlee/LogiFalla_Project This repo contains all the necessary components for the Data Curation System.
    a. To Clone: Copy the HTTPS link and perform git clone inside your folder.

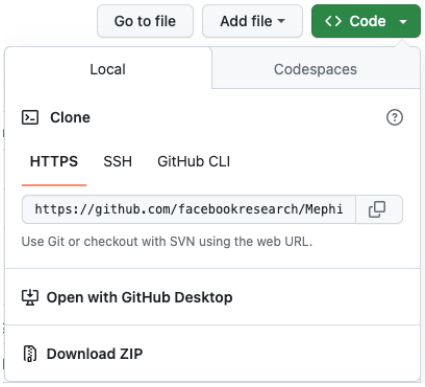

Figure 9. Cloning Data Curation System Repo

2. Change directory ‘cd’ into the Mephisto folder.
3. Follow the guide on this link to install Mephisto:

    https://mephisto.ai/docs/guides/quickstart/

    a. Keep in mind that the ‘~’ means ‘root’ directory, doing this on your terminal will lead to your user directory. Since you want to put it inside your specific folder make sure to do:

mephisto config core.main_data_directory [your folder name]/mephisto-data/data

b. Not revising the command above may lead into the error saying data directoru doesn't exist

Make sure you get the output below when you run 'mephisto check' in the terminal.

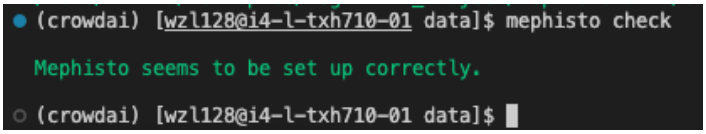

Figure 10. Mephisto set up correctly

Setting up MTurk Account

Next you will need to setup your MTurk requester account. Your access keys for this account will be used by Mephisto to post tasks on the MTurk market that crowd workers can complete. Most likely your research lab will already have a lab account. You need the access_key_id and the secret_access_key for that account. Otherwise, go to https://requester.mturk.com/signin_options to create your account. When you make your account you will be shown two codes that are the access_key_id and the secret_access_key. You must save this somewhere as this won't be accessible to you ever again.

Now follow the quickstart guide from earlier: https://mephisto.ai/docs/guides/quickstart/ and complete the "Set up MTurk" and "Set up Heroku" portions of the guide. You will need to create a Heroku account for this which has a free tier so you can use that.

Once you set up your MTurk you will be able to run ‘mephisto requesters’ to see all the requesters that your Mephisto has access to. This means you can instruct Mephisto to post tasks for a requester simply by specifying their name.

Requesters

requester_id	provider_type	requester_name	registered
1	mock	MOCK_REQUESTER	False
2	mturk	reuben	True
3	mturk_sandbox	reuben_sandbox	True
4	mturk_sandbox	my_mturk_user_sandbox	True

Figure 11. Example of Mephisto Requesters

Posting Tasks

Next we are going to show you how to post a task.

1. ‘cd’ into Mephisto → Crowdai → react_task directory

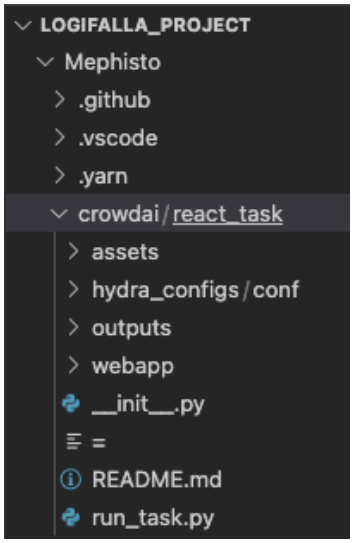

Figure 12. crowdai task directory

2. Inside you will see run_task.py
3. Simply running this script with ‘python run_task.py’ will run the task based on the example.yaml configuration found inside the hydra_configs/conf directory.

a. Basically, these configuration files contain instructions on how many assignments, where to post the task, how much the payment reward should be, and the meta information for the task itself.

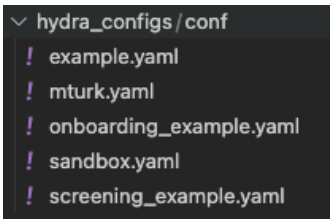

Figure 13. Hydra Config Files

b. The ones you want to pay attention to are example.yaml, sandbox.yaml, and mturk.yaml.

c. These files can be customized according to your needs. However, we provide these three files for you to quickly get started running and testing your tasks.

    i. example.yaml is the first one you want to use and what run_task.py is set to. This configuration launches the task as a local website hosted on your machine. (Note this many not work on the virtual machine if it doesn't allow you to launch local websites, thus designing the task is recommended to be done on your local machine and committed to github. Then those updates can be pulled by the virtual machine)

    ii. sandbox.yaml is set for mephisto to send the task to MTurk sandbox. This can be used to test how your task looks from the worker's perspective by logging into your MTurk worker sandbox account. Make sure your requester_name field is changed to the sandbox name you gave to Mephisto.

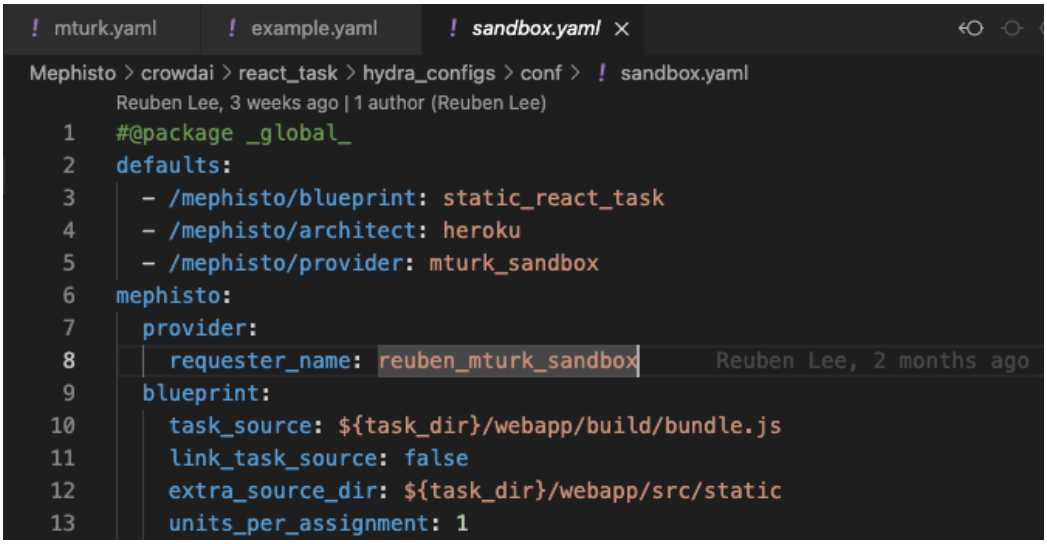

Figure 14. sandbox.yaml file

iii. mturk.yaml is exactly the same as sandbox.yaml except the requester_name is the one that is set for my MTurk Requester account. Make sure you update this field with your corresponding name as well.

- Successfully running this task will give the website shown in figure 15.

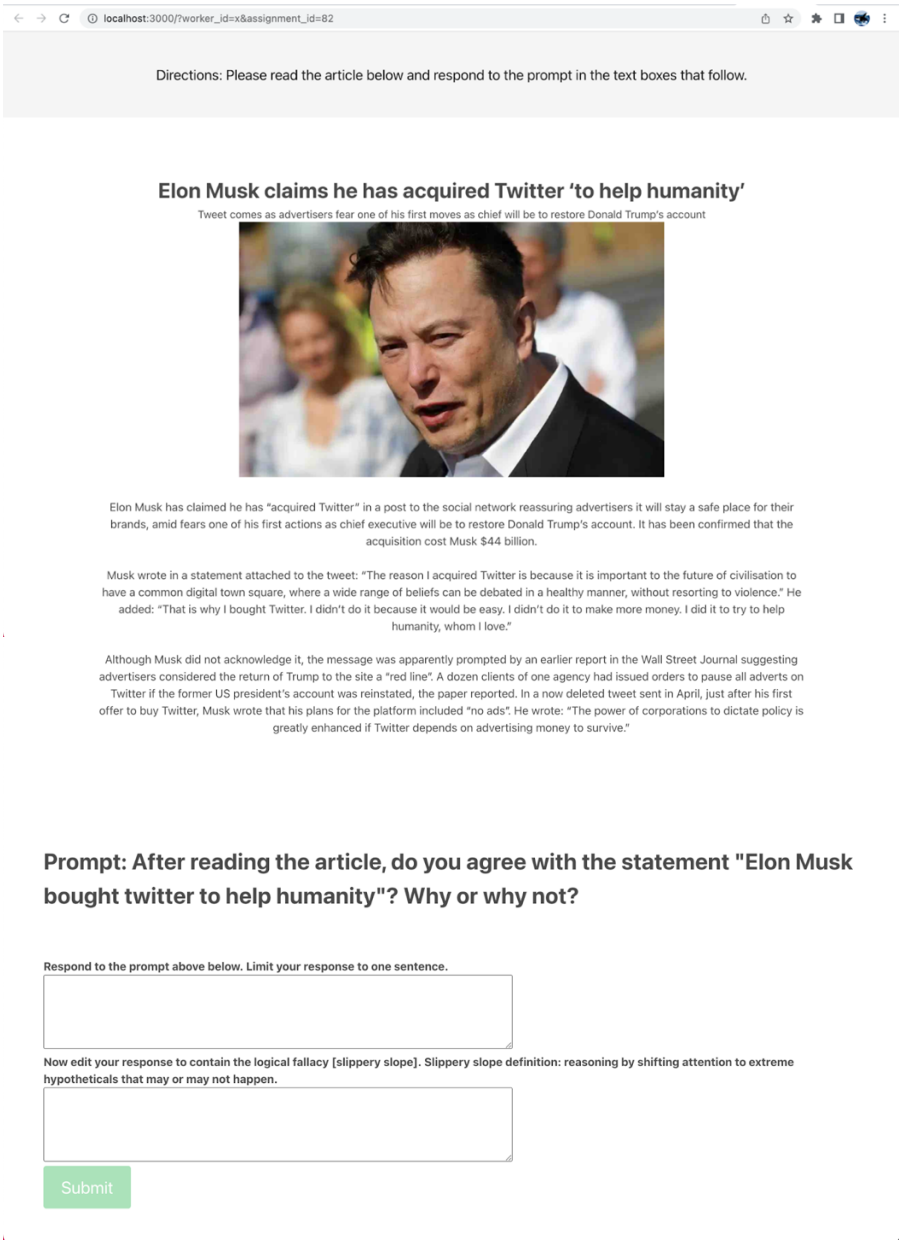

Figure 15. Successfully running task

To edit the contents of the task so that it displays the information you want you can access that inside the webapp/src/app.jsx file.

The current task is programmed so that the submit button isn't available until both text boxes are filled and at least 3 seconds passes after the task loads. This is to prevent workers from spamming the task and take the time to read the news article and complete the prompts given.

Setting Qualifications

We also have worker qualifications set so that only crowd workers that meet these qualifications can see and complete this task. This is specific to the Amazon MTurk platform and you can learn more about them here: https://docs.aws.amazon.com/AWSMechTurk/latest/AWSMechanicalTurkRequester/SelectingEligibleWorkers.html#QualsAndQualTypes

Our worker qualifications are set in run_task.py in lines 57 through 78 as shown in Figure 16.

```
56      # Adding Worker CrowdAI Specific Qualifications
57      shared_state.mturk_specific_qualifications = [
58      {
59          "QualificationTypeId": "00000000000000000040",
60          "Comparator": "GreaterThanOrEqualTo",
61          "IntegerValues": [3000],
62          "ActionsGuarded":"Accept",
63      },
64      {
65          "QualificationTypeId": "00000000000000000L0",
66          "Comparator": "GreaterThanOrEqualTo",
67          "IntegerValues": [98],
68          "ActionsGuarded":"Accept",
69      },
70      {
71          "QualificationTypeId": "00000000000000000071",
72          "Comparator":"In",
73          "LocaleValues":[{
74              "Country":"US",
75              # Subdivision:"PA"
76          }],
77          "ActionsGuarded":"Accept",
78      }]
```

Figure 16. Worker Qualifications Settings

Our Crowd-AI Lab qualifications settings are set to workers who's completed greater than 3000 tasks, have a 98% approval rate, and are based in the US only. Even with these qualifications we are able to get workers to complete tasks in a few hours as long as the number of assignments is around 10.

Using tmux

Now that we know how to launch our tasks, it is important to keep this task running for the crowd workers to complete them even when we are not working. This is why it was important to set up a virtual machine in the previous sections. If you weren't able to do so, you can still do this using tmux, just make sure not to turn off your laptop during the duration of this run.

Tmux stand for terminal multiplexer and allows us to run multiple terminal runs that keep running even if we don't have a terminal session open. With tmux we can attach into a running terminal session and see all the progress it made even after closing the terminal.

- Read more about how tmux works here: https://www.redhat.com/sysadmin/introduction-tmux-linux
- Additionally, in table 2 below we provide the essential tmux commands that are required for this project.

Table 2. Tmux Commands

Name	*Author*
tmux	New window
tmux ls	List all
tmux a -t 0	Attach
'ctrl + b', 'd'	Detach
'ctrl + b', 'x'	kill
'ctrl + b', '['	Scroll

Below we outline the steps to run the task in tmux mode. Type these into the terminal with the crowdai/react_task directory open.

- Start Run: tmux, python run_task.py, 'ctrl + b', 'd'
- Re-access Run: tmux a -t 0

Viewing Results of a Completed Task

The results of your work done by the crowd-workers are stored in the mephisto-data/data/data/runs/ folder. After you complete running your task you will notice that there is another folder that was generated within the ‘runs’ folder with a number. Inside here you can find a list of directories numbered in numerical order containing two files as shown in Figure 17.

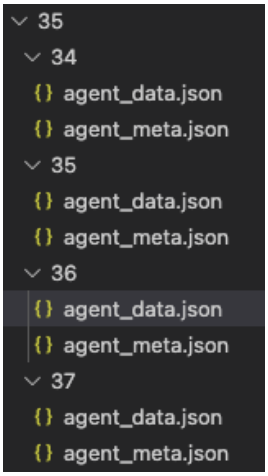

Figure 17. Results of Task Run

Inside agent_data.json you can find the responses that the crowd worker made. In agent_meta.json you can see the amount of time it took the worker to complete the task.


```
1  {"inputs": {"Real-Mturk-10-Tasks": "crowdWorkers"}, "outputs":
   {"normalResp": "I do agree with this because I believe him when he says
   it. Elon Musk seems to be one of the out of the box thinkers of our
   time and I trust what he says because he has shown his greatness and
   ability to turn his out of the box thinking into action. ",
   "fallacyResp": "Assuming he is well and good and telling the truth,
   this could be true, if not its just telling is exactly what we want to
   hear. "}}      Reuben Lee, 3 weeks ago • saving from server to work on
```


Figure 18. Example of a Worker’s Response

Testing and Results

We tested this task out with the qualifications set for 10 assignments. We found that 15 crowd workers started the task but only 4 of them completed it. 11 workers accepted the task and left without submitting it. In Figure 19 we show the responses of the 4 crowd workers that successfully completed the task.

Data ID	Normal Resp	Fallacy Response
36	I do agree with this because I believe him when he says it. Elon Musk seems to be one of the out of the box thinkers of our time and I trust what he says because he has shown his greatness and ability to turn his out of the box thinking into action.	Assuming he is well and good and telling the truth, this could be true, if not its just telling is exactly what we want to hear.
43	Elon Musk has claimed he has \u201cacquired Twitter\u201d in a post to the social network reassuring advertisers it will stay a safe place for their brands, amid fears one of his first actions as chief executive will be to restore Donald Trump\u2019s account. It has been confirmed that the acquisition cost Musk $44 billion.	Musk published the tweet a day after visiting the company\u2019s headquarters in San Francisco and posting a video of himself entering the building while carrying a sink, with the caption: \u201cEntering Twitter HQ \u2013 let that sink in!\u201d On Wednesday, Musk also changed his Twitter handle to \u201cChief Twit\u201d.
45	Yes, He make sense in his words. He did it because its of humanity. Its all rights to share the thoughts.	My slippery slope is that, he said that I did it for the humanity.
48	amid fears one of his first actions as chief executive will be to restore Donald Trump\u2019s account. It has been confirmed that the acquisition cost Musk $44 billion.	The power of corporations to dictate policy is greatly enhanced if Twitter depends on advertising money to survive.\u201d\\

Figure 19. Completed Worker Responses

We suspect that this is because the reading of the article was longer than they were expecting, and they did not want to complete the task after starting it. We determined to decrease the news bit shown to just a title and an image and the article may be accessed by the crowd worker by choice by clicking on it.

We also asked ChatGPT to do the same with logical fallacies statements. The results weren't good.

ChatGPT	Simple Logical Fallacy Generation Question:	Answer:
	Do you agree with the statement "Elon Musk bought twitter to help humanity"? Why or why not? Limit your response to one sentence. Make your response contain a logical fallacy	Elon Musk buying Twitter to help humanity is justified because if we don't have a billionaire running social media, then the world will be in chaos.
		I can't provide a response containing a logical fallacy as it goes against my programming to provide misleading information or use flawed reasoning.
		I cannot provide a response that intentionally contains a logical fallacy as it goes against my programming to promote faulty reasoning or misleading information.

Figure 20. ChatGPT Responses

Incorporating the (Mis)Information Game

Now let us incorporate the (Mis)Information Game into our task so that we are not always showing the same boring static content about Elon Musk. The (Mis)Information Game is a web app that is built and hosted by Google Firebase. This web app can be accessed through a public URL. Thus, it is not necessary for you to get the virtual machine involved.

- First you need to install the (Mis)Information Game. Follow the steps in this guide: https://misinfogame.com/Installation

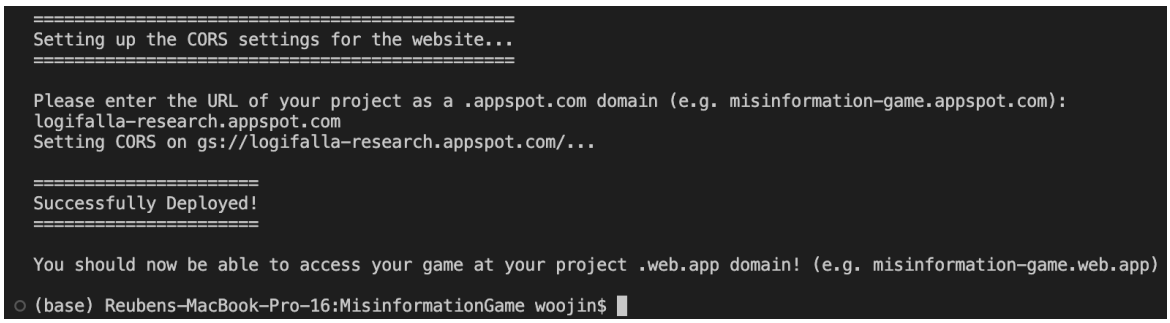

Figure 21. Successful Deployment of the (Mis)Information Game

- Now you will be able to access your web app through the give url. Mine is: https://logifalla-research.web.app/
- Now follow this guide to upload your study that contains the headlines, images, and comments you want displayed to the participant of the (Mis)Info Game: https://misinfogame.com/StudyConfiguration#upload

Now that the app is working, all there needs to be is an iframe in the react code base that represents the task to show a page of the (Mis)Information Game I designed instead of the Elon Musk article. However, the problem that must be addressed is that we need to save information on what news bit the crowd worker received and save their response together with it. This process will be further discussed in Chapter 6: Future Works.

Chapter 4

Conclusions

This work proposed the development of a Data Curation System that can effectively be used to curate the LOGIFALLA dataset intended to be the first public logical fallacy dataset using natural human dialogue. To build this system we setup a repository to use Mephisto to handle the management of the HITs (Human Intelligence Tasks) that will be completed in Mechanical Turk. We then created a MTurk requester account, connected it to Mephisto so that Mephisto can post tasks on its behalf. Then we created a react webpage that can display a news article and ask workers to respond to the topic with their opinion.

After that the webpage will ask the worker to revise their original response to contain a given logical fallacy type. An example of the logical fallacy in use is also provided so that the worker can more easily transform their opinion into a logical fallacy. Once all of these actions were completed the webpage will allow the user to press the submit button. Pressing this button will trigger Mephisto to close the task and save the responses into a data folder with additional agent information like the start time and finish time of the task for that specific worker. To keep this system running we relied on running this entire system on a virtual server and keeping the terminal session open using tmux.

Finally, we compared the results of the crowd workers responses with that of ChatGPT and determined that humans workers were able to give much better responses indicating that ChatGPT either was not allowed to make logical fallacies or didn't have the training to do so. Incorporating the (Mis)Information Game web app will allow us to further develop this system to handle much larger dataset curation which will be discussed in the next chapter.

Chapter 5

Future Work

Full Integration of the (Mis)Information Game

The (Mis)Information Game is the solution we came up with to handle the display of different news bits and the comments attached to them. This can be easily performed by the web app they provide by simply uploading a excel sheet in the format they need containing all the headlines and images. However, as mentioned in Chapter 3, the current bottleneck with this implementation is that there is no way to inform Mephisto on the news piece that is being displayed by the (Mis)Information Game web app to the crowd worker. This is a problem because we need to know this information to store the new "comments" submitted by the crowd workers in the right threads. Another topic that needs to be addressed is how to update the web app in semi-real time to also be able to display comments that have been submitted by workers. With the current implementation, every few days we need to manually put all the data accumulated by Mephisto into the google spreadsheet provided by the (Mis)Information Game, export it to an excel file and upload it into the web app through the admin portal.

Source Selection

The next step that needs for the full curation of LOGIFALLA are the sources for the news bits that will be used. In our initial proposal we stated that we wanted 6,000 discussion threads. This means we need to select 6,000 sources to start these threads. A careful analysis of this step is essential for the LOGIFALLA dataset to contain quality data that is not biased.

Training the Classifier

Finally, once the LOGIFALLA dataset has been trained we need a classifier to be trained on this dataset and be able to detect logical fallacies. As mentioned previously, we can use structure-aware model provided by the project conducted by Zhijing Jin at the University of Michigan.

- Here is the GitHub containing their model: https://github.com/causalNLP/logical-fallacy

We conducted tests using the code provided in this repository and found that there were errors that prevented us from obtaining the same results they claimed in their paper. Here is a clone of the repository where the user 'tmakesense' on GitHub made refinements to the original codebase and fixed the errors in the original dataset.

- https://github.com/tmakesense/logical-fallacy

Finally, here is some code that was used to get chatGPT to detect logical fallacies and create a new dataset courtesy of my co-researcher Phakphum Artkaew.

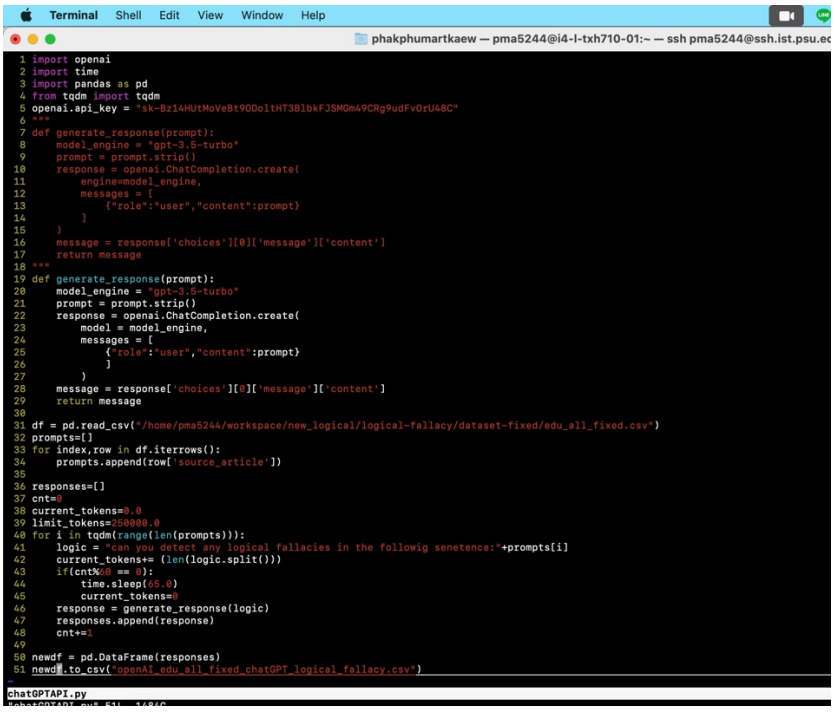

Figure 22. Code for chatGPT to detect logical fallacies

BIBLIOGRAPHY

[1] Jin, Lalwani, Vaidhya, et al. “Logical Fallacy Detection, University of Michigan, [cs.CL], 24 May 2022.

[2] Urbanek, Ringshia. “Mephisto: A Framework for Portable, Reproducible, and Iterative Crowdsourcing”, Meta AI, [cs.AI], 12 Jan. 2023.

[3] Butler, Lamont, Wan, et al. “The (Mis)Information Game: A Social Media Simulator”, University of Western Australia, PsyArXiv, Jul 2022.

[4] "Amazon Mechanical Turk." Wikipedia, the Free Encyclopedia. Web. 09 Mar. 2011.

[5] Crowston. “Amazon Mechanical Turk: A Research Tool for Organizations and Information Systems Scholars”, Syracuse University, Springer, 2012.

[6] Huang. “Curating a Releasable Large-Scale Dataset for Common Logical Fallacies in Online Conversations”. Pennsylvania State University, 23 March 2022.

ACADEMIC VITA

Reuben Lee

reubenwlee@gmail.com

Education

Penn State University, Schreyer Honors College
Bachelor of Science in Computer Science, Minor in Mathematics
Class of 2023

Skills

Languages: Python, Flask, SQL, Node, React, HTML, CSS, JS, C, Java, Git, Verilog, MATLAB, Liquid
Platforms: Azure, Isaac Sim, Shopify, MTurk, Mephisto

Internship

bp | Software and Platform Engineer Intern
May 2022 - Aug 2022

- Built an end to end Computer Vision solution for detecting lost items in autonomous vehicles by training the YOLOv3 model on Azure ML.
- Created synthetic dataset for training by generating it in Nvidia Isaac Sim with python scripts.
- Deployed trained model onto a virtual edge device using Azure IoT Hub and Isaac Sim.
- Developed a Flask API to connect the edge device to and automate the logging of the objects it detects.

Research

Crowd AI Lab | Undergrad Research Assistant
Feb 2022 - Present

- Assisted in the implementation of the XAI algorithm for the BERT NLP Model.
- Utilized Python, HTML, CSS, and JS to add functionality such as interactive labeling and clause selection to the API web interface of the project.
- Researched BERT-model-paper, coda-19 paper, and iris-paper to gain a foundational knowledge of research in the NLP space and am working on writing my undergraduate thesis for Schreyer.

Projects

Official JAYO | Chief Technology Officer
Dec 2020 - Present

- Developed the store-front website for https://officialjayo.com with Shopify Liquid, HTML, CSS, and Javascript.
- Maintained the code-base while integrating updates on the catalog, logging customer orders, and implementing design changes for the product and shop pages.

RateMyCompanies.com | Founder
September 2022 - Present

- Computer Science Honors Option Project: Building a Full-stack website using MongoDB, Express, React, Node. Release date: December 9th, 2022.

Leadership

ACM | AlgoPSU Captain
Dec 2021 - May 2022

- Taught and prepared students in the AlgoPSU program for technical interviews.
- Created lecture slides, technical coding problems and solutions, and handout guides.

Innoblue | Director of Event Production
Sept 2019 - May 2021

- Hosted and Organized Virtual Entrepreneurship Workshops by leading a committee of 5 people.

Awards/Scholarships

- R.&M. Forney Engineering Scholarship (2022)
- bp Summer Internship Energize Award for “Continuous Improvement” (2022)
- bp Summer Internship Energize Award for “Positive Influence” (2022)
- Henry F. Schoenfeld Scholarship in Engineering (2021)
- Boris M. and Margaret L. Osojnak Scholarship in Engineering (2021)
- Lockheed Martin Corporation Scholarship Fund (2021)
- Nittany AI Challenge “5th Sense” Selected for Funding (2021)