Puppeteer: Leveraging a Large Language Model for Scambaiting

Pithayuth Charnsethikul
University of Southern California
Information Sciences Institute
charnset@usc.edu

Jelena Mirkovic
University of Southern California
Information Sciences Institute
mirkovic@isi.edu

Rishit Saiya
University of Southern California
Information Sciences Institute
rsaiya@usc.edu

Benjamin Crotty
University of Southern California
bcrotty@usc.edu

Jeffrey Liu
University of Southern California
jliu5021@usc.edu

Genevieve Bartlett
University of Southern California
Information Sciences Institute
bartlett@isi.edu

Abstract

Scambaiting is a defense that engages with scammers to waste their resources and gain information about their fraud campaigns. This defense needs automation to scale to the vast number of scams we see today. In this paper, we propose a scalable, automated scambaiting system, Puppeteer, which leverages a large language model for response generation and state machines for conversation tracking. We measure Puppeteer's effectiveness via a user study, where participants play a role of a scammer in two scam scenarios. Puppeteer convinced more than 72% of the participants that they were interacting with a human, and was able to extract information from 68% of participants. In comparison, using the same large language model without conversation tracking convinced only 54% of the participants that they were interacting with a human and obtained information from 54% of participants. Our results show potential for real-world use of Puppeteer. To the best of our knowledge, we are also the first to systematically evaluate a large language model for a scambaiting task.

Keywords: scam, scambaiting, phishing, large language model.

1. Introduction

A scam is a fraudulent scheme designed to deceive individuals, often for financial gain (CNBC, 2022; FBI, 2021; Forbes, 2022). Scams are ubiquitous and involve a diverse range of scenarios. For example, there are online shopping scams where people purchase an item online, but it is never delivered (Carpineto & Romano, 2017); charity scams where people donate money to a fake charity (Bitaab et al., 2020); and information extraction scams where people are tricked into revealing their personal information, which scammers can monetize (Oest et al., 2018). In this paper, we focus on information extraction scams.

Online scams are simple for the attackers to set up, and their number continues to rise (APWG, 2023), leading to billions of dollars in losses (CNBC, 2022; FBI, 2021; Forbes, 2022). To defend against scams, researchers have made significant advances in scam detection (Lin et al., 2021; Ma et al., 2009; Tian et al., 2018; Zhang et al., 2007). Yet, some scams bypass the detection system, and we need additional defenses. Scambaiting is one potential defense against scams, where the defenders engage with scammers to deplete their resources (e.g., time and computational resources) and also try to extract some useful information for forensics and law enforcement.

While scambaiting shows potential in fighting scams, it still faces some challenges that limit its practical use. First, human-driven scambaiting tends to be very successful. Law enforcement engages selectively with scammers, and amateurs share their success in fooling scammers online (NPR, 2024). Yet, human-driven efforts cannot scale to the current volume of scams.

To address this scalability issue, researchers have developed automated scambaiting systems (Li et al., 2020; Molloy, 2023; Netsafe, 2017). These systems rely on training a machine-learning model on previous scams to produce relevant and human-like responses to scammers. Because scam scenarios constantly evolve, a pre-training approach cannot work to meet real-world needs. Instead, we need a robust scambaiting system that can produce human-like responses, work to extract information from scammers and easily apply to any scam scenario, without additional training.

We propose an automated scambaiting system called Puppeteer, which engages with scammers in a human-like

URI: https://hdl.handle.net/10125/108970
978-0-9981331-8-8

manner, and works to extract useful information about them. Puppeteer consists of three components: a Natural Language Understanding (NLU) component, a set of state machines, called ***agendas*** (Cho et al., 2021), and a large language model (LLM). NLU interprets a scammer's utterance and its relevance to the defender's information extraction goal. *Agendas* employ a state machine model, use the NLU interpretation to determine the current state of the conversation and select an appropriate LLM prompt. Finally, the LLM uses the prompt from *agendas* to generate a human-like response back to the scammer.

We further develop metrics for scambaiting systems and systematically evaluate Puppeteer, comparing its performance with a stand-alone ChatGPT, via an IRB-approved human user study. To the best of our knowledge, this is the first systematic evaluation of a scambaiting system, and its ability to engage with a human. Our results show that Puppeteer outperforms stand-alone ChatGPT in every metric, underlining the importance of tracking a conversation by *agendas*. Specifically, Puppeteer achieved a success rate of 72% in generating human-like conversations, while ChatGPT only achieved 54%. Puppeteer also managed to extract 15 addresses and 11 phone numbers in 22 conversations, while maintaining an average conversation length of 14 turns. Conversely, ChatGPT extracted 12 addresses and 9 phone numbers with the average conversation length of 13 turns. Our results show the potential for real-world use of Puppeteer to scambait.

2. Background & Related Work

The anti-scam ecosystem represents collective efforts to combat various forms of scams, frauds, and deceptive practices. To date, the majority of effort has been put into the detection and prevention phases of the ecosystem. Common detection techniques include spotting phishing websites by their URLs (Ma et al., 2009; Tian et al., 2018) or content (Lin et al., 2021; Zhang et al., 2007). These techniques primarily leverage machine learning, specifically supervised learning using many annotated datasets. Meanwhile, prevention techniques rely on anti-phishing tools, such as browser extensions (Akhawe & Felt, 2013; Egelman et al., 2008) and email warnings that notify users when suspicious phishing activities are identified (Marforio et al., 2016; Petelka et al., 2019). Another remedy is user training that instills awareness of up-to-date scams to mitigate the risk of falling victim to them (Kumaraguru et al., 2009; Reinheimer et al., 2020; Wash & Cooper, 2018).

Nevertheless, keeping pace with the rapid growth of scams poses significant challenges for these passive defenses. The Anti-Phishing Working Group (APWG) reported that in the fourth quarter of 2023, a number of scams surpassed the previous record from the third quarter, with over 1.3 million new attacks and more than one hundred thousand unique email subjects in October alone (APWG, 2023). Given the sheer volume of new scams being introduced, retaliatory strategies during the post-detection phase are needed in order to decelerate the attacks. In this paper, we focus on a particular retaliatory strategy called "scambaiting".

In 2017, Re:Scam was introduced to combat email scams (Netsafe, 2017), and Lenny to tackle voice scams (Sahin et al., 2017). While Re:Scam first attracted a lot of attention, it was shut down later that year for undisclosed reasons. Lenny is still operational and plays pre-recorded messages when a scammer pauses, to mimic a person replying using silence detection.

MISSA, introduced in 2020, fine-tuned GPT-2 for scambaiting in an Amazon customer service scam (Li et al., 2020). With the rapid advances in LLMs, MISSA's responses have become unsophisticated compared to today's zero-shot LLMs. Moreover, MISSA cannot be applied to different scam scenarios, because fine-tuning memorizes context and incorrectly applies it to conversations in different contexts.

LLMs address the limitations of current scambaiting systems (Li et al., 2020; Sahin et al., 2017), because they can generate human-like conversations in any scam scenario. As LLMs continue to advance, they are capable of responding to a wide range of topics and generating contextual and coherent responses. However, these zero-shot responses are non-deterministic, which brings both advantages and challenges. On the upside, the LLM generates more diverse responses, making it harder for scammers to recognize they are interacting with a bot. The downside is that when the LLM is designated to work on a specific task (e.g., extracting information), it can lose its focus and fail to reach the objective, or it can become hyperfocused and repetitive, revealing its non-human nature to the scammer.

Recently, Apate (Molloy, 2023) and HackBot (Lundie et al., 2024) were proposed. These systems use LLMs for scambaiting, but their current publications have no in-depth system design information or performance evaluation and thus we cannot directly compare them with Puppeteer.

3. Puppeteer

We propose Puppeteer, an automated scambaiting system that leverages the LLM for response generation and *agendas* (Cho et al., 2021) (state machines) for conversation tracking. The LLM ensures that Puppeteer

is capable of generating human-like responses to a myriad of diverse conversations, while *agendas* keep Puppeteer focused on its information extraction task.

Objectives: Puppeteer has three objectives: (**O1**) It should engage with the scammer in a human-like fashion; (**O2**) It should make an effort to extract information from the scammer; (**O3**) It should apply to various scam scenarios with minimal or no changes.

3.1. System Architecture

Puppeteer uses three main components (Figure 1): Natural Language Understanding (NLU) component, *agendas*, and a large language model (LLM). There are four steps involved in response generation by Puppeteer. First, the NLU component interprets a scammer's utterance and its relevance to the information extraction goal. Since interpreting an utterance is not a binary task, NLU represents its interpretation in terms of probability over a set of possible utterance classes, or as we call them "intentions". Second, the NLU interpretation (i.e., scammer's intention probabilities) is shared with *agendas* to determine the current state of the conversation, and *agendas* select an LLM prompt based on this current state. Third, the LLM uses the prompt prepared by *agendas* to generate a response. Lastly, Puppeteer sends this generated response back to the scammer. We explain how we implement each component below.

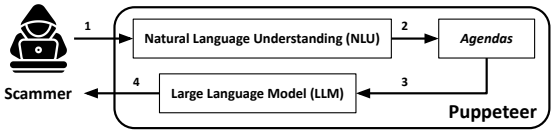

Figure 1. Puppeteer architecture

3.2. Natural Language Understanding (NLU)

The primary task of the NLU component is to interpret a scammer's utterance and its relevance to the information extraction goal. This interpretation is not straightforward. For example, consider an utterance: "If you don't give me your credit card information, then I cannot help you with your order." Even humans can interpret this utterance differently, e.g., as a request for credit card information or an offer of help. Hence, we express the utterance interpretation in the form of probabilities across different scammers' "intentions". An intention is a class of utterances that all have similar impact on the information extraction goal, e.g., asking for information or declining to give information, etc.

To compile a comprehensive yet compact and scam-agnostic list of scammers' intentions, we conducted a pilot study in which we collected 10 dialogues between volunteers playing roles of scammers and victims in an Amazon customer service scam (Li et al., 2020). We then extracted all scammers' utterances from these 10 dialogues and asked ChatGPT to cluster these utterances into different intention groups. Table 1 shows the finalized list of scammer intentions: ask, assist, info, issue, refusal and general. Unlike fine-tuning, which requires data collection for every scam scenario, our generic list of scammer's intentions omits specific contexts, (e.g. a mention of Amazon orders), ensuring that the list is applicable to any information extraction scam scenario.

To which class do you think the message below belongs?
`INPUT MESSAGE`
The classes include $C_1, C_2, ..., C_n$. Here are some example sentences for each class.
$C_1 : E_{1,1}, E_{1,2}, ..., E_{1,m}$
$C_2 : E_{2,1}, E_{2,2}, ..., E_{2,m}$
...
$C_n : E_{n,1}, E_{n,2}, ..., E_{n,m}$
Given these examples, output a confidence score for each class in probability. Return as a JSON object {class}: {probability}

Figure 2. NLU prompt template

NLU performs multi-label classification that outputs intention group probabilities, given a scammer's utterance. For example, "If you give me your credit card I can help with your order" could be interpreted as 60% ask and 40% assist. We use ChatGPT as our NLU component to perform multi-label classification and we use regular expressions to detect when a piece of information matches the format of the information we seek to extract (e.g., a phone number). Regular expression matching is performed prior to multi-label classification. We assign 1 to the "info" intention if the regular expression is matched, and skip the multi-label classification. Otherwise, we assign 0 to the "info" intention, and proceed with the multi-label classification. Previous studies show that ChatGPT excels in text classification without additional fine-tuning (Sarkar et al., 2023). We use the template in Figure 2 as a prompt given to ChatGPT. `INPUT MESSAGE` is a scammer's utterance. The classes $C_1, C_2, ..., C_n$ are the intentions listed in Table 1 and each class C_i is provided with m example sentences ($E_{i,j}$), collected from the pilot study.[1] Puppeteer uses intention probabilities, which are output by NLU component, to gain an understanding of what the scammer said in the previous turn and update the current states in *agendas* accordingly.

[1] Due to space limits, we omit a newline between each example.

Table 1. Scammer's intentions with their descriptions and examples

Intention	Description	Example
ask	A scammer asked for our information.	We just need your credit card detail to process the payment again.
assist	A scammer offered to assist with something.	I'm just trying to resolve the issue regarding your order.
info	A scammer provided their information.	Our office is located at 3429 Station Street, Austin, Texas, 78701.
issue	A scammer claimed that there are some issues with something.	Your credit card information that you provided did not go through.
refusal	A scammer refused to give an answer to our question.	Unfortunately, I cannot disclose that information due to company policies.
general	Any utterances that are unrelated to either our or scammer's goals.	You can Google our company.

3.3. *Agendas*

Agendas are represented as probabilistic state machines (Cho et al., 2021; Hudson & Newell, 1992). The inputs to *agendas* are the NLU's classification containing probabilities of the scammer's intentions. *Agendas* use this input to update the probability of each state and identify the current state—the state with the highest probability. Each state contains a list of candidate instructions and *agendas* choose one at random to include in the upcoming LLM prompt.

3.3.1. States and Transitions We employ *agendas* to keep Puppeteer focused on its information extraction goal. Figure 3 displays the high-level architecture of an *agenda*. We manually define states based on the degree of success in obtaining the scammer's information from their current utterance. The orange state indicates the start state, whereas the green state indicates the terminal state. Other colors indicate intermediate states. The NLU interpretation (scammer's intention probabilities) triggers transitions between the states. Different transition colors indicate different detected intentions (Table 1).

Specifically, an *agenda* starts at the "start" state, where the LLM generates a response that asks for a specific piece of information (e.g., phone number). When the scammer replies, the *agenda* will check if the "info" intention is detected, signifying that information was obtained. There are three possible responses, illustrated via three different transition colors, in Figure 3. The green transitions indicate that Puppeteer obtained the full information and as a result the *agenda* is complete. The blue transitions indicate that Puppeteer obtained partial information, and the *agenda* will instruct the LLM to retrieve the remaining information. The NLU detects which part of the information is missing and shares it with the *agenda* to ensure that Puppeteer will follow up on this missing part. For example, when Puppeteer asked for an address and the scammer only provided a city.

The red transitions indicate that Puppeteer received some pushback or the scammer did not provide the requested information. In this case, we will use the NLU-output intentions to update the pushback states. Figure 4 illustrates pushback states and transitions within an *agenda*. Puppeteer categorizes pushback into five

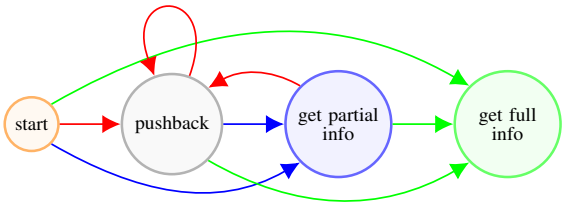

Figure 3. State machine representing an agenda

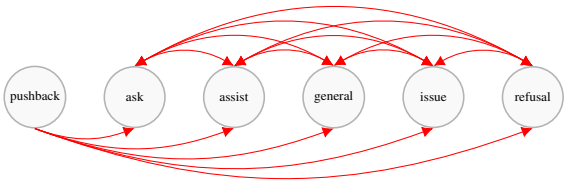

Figure 4. Pushback states within an agenda

categories: ask, assist, general, issue, and refusal, where each pushback implies a different intention (Table 1). These states and transitions are an extension of the main state machine shown in Figure 3. When the probability of the "info" intention is zero, five pushback intentions will be detected with some probabilities. We use these intention probabilities to update the pushback state machine, according to the Probabilistic Transition Algorithm described in (Hudson & Newell, 1992) and summarized in Equation 1. In the equation, $P_{c,s}$ and $P_{n,s}$ are the current and new probabilities of pushback state s, respectively, $I.vals$ is a value list of intention probabilities, I_s is a probability of intention s, and S is a set of pushback states. The pushback state with the highest probability is considered to be the new current state.

We illustrate how pushback states are updated with a small example. When we asked the scammer for their phone number and received a pushback response "Why don't you give me your number and then I'll call you?", the NLU outputs the intention probabilities (I) as {ask: 0.88, assist: 0.10, info: 0.00, issue: 0.03, refusal: 0.20, general: 0.00}. Assuming that our current pushback state probabilities (P_c) were {ask: 0.19, assist: 0.07, issue: 0.18, refusal: 0.11, general: 0.45}. Applying Equation 1, the new pushback state probabilities become {ask: 0.67,

assist: 0.08, issue: 0.04, refusal: 0.16, general: 0.05}, making the "ask" pushback the current state.

$$P_{n,s} = \frac{(1 - max(I.vals))P_{c,s} + max(I.vals)I_s}{\sum_{s' \in S}(1 - max(I.vals))P_{c,s'} + max(I.vals)I_{s'}} \quad (1)$$

When Puppeteer asks for multiple pieces of information (e.g., address and phone number), we use one *agenda* per information piece. When multiple *agendas* are active, Puppeteer matches the scammer's utterance against all pieces of information it seeks to extract, and if any are found, the corresponding *agenda* is terminated. Otherwise, Puppeteer chooses one among active *agendas* to be the current *agenda* for a given number of turns. For example, it may ask for an address, making the address *agenda* current, converse for a certain number of turns and if no progress is made, ask for a phone number, making the phone number *agenda* current. Then it would converse for a certain number of turns and make the address *agenda* current again. Only the current *agenda*'s states are updated given the output of the NLU.

3.3.2. Instructions *Agendas* keep a list of candidate instructions for each state. An additional check is performed to ensure that the list of candidate instructions follows our strategic interaction rules, described in Section 3.3.3. This check can eliminate some instructions from the candidate list. *Agendas* then select one instruction at random from the candidate list and will use it to form the next LLM prompt. Table 2 displays all available instructions. and states for which they are available. Each instruction is accessed in a key-value fashion. The key is a pair of action and sub-action, while the value is the instruction.

3.3.3. Strategic Interaction Rules We strategically define a set of interaction rules for Puppeteer, based on previous work (Brett Stoll & Edwards, 2016; Chaves & Gerosa, 2019; Clark et al., 2019; Rheu et al., 2021), which suggested that a machine needs to possess three elements below to lead a human-like conversation:

- *Social interaction*, such as chit-chat, offers a form of communication unrelated to a specific goal, thus avoiding the system being too hyperfocused.
- *Identification*, such as being able to share personal information, reflects anthropomorphism of the system and is seen as an important step to advance non-collaborative conversations (e.g., scambaiting).
- *Strategies* determine (1) how much a machine should inject social interaction without losing focus on its main objective, and (2) when a machine should (and should not) reveal its personal information.

We define two actions for Puppeteer, as shown in Table 2: "extract" means working toward the information extraction goal, and "socialize" means going along with the scammer's conversation and trying to appear human-like. We use this notion to manually formulate a set of rules below that identify which actions Puppeteer is allowed to perform in each turn, particularly which instructions are considered as candidates.

Rule 1: The "extract" and "socialize" actions will alternate throughout the conversation to ensure that Puppeteer maintains persistence toward its objective, while avoiding a level of persistence that could lead to discomfort, and prematurely terminate the conversation.

Rule 2: The "extract" actions are allowed to be used for a maximum of 2 consecutive turns, while "socialize" actions are used one at a time, ensuring that Puppeteer prioritizes its information extraction goal, while prolonging the conversation as much as possible.

Rule 3: Whenever Puppeteer makes progress towards the objective, i.e., obtaining information, an "extract" action will always be chosen for the next turn. The rule holds true regardless of the previous rules, as Puppeteer aims to maintain the momentum in acquiring the information.

Rule 4: Puppeteer is provided with a fake profile (e.g., name, phone number, address). If it cannot make progress towards its extraction goal within T turns, it will start to disclose its own personal information. The hope is that by appearing to share some information, the conversation will remain alive and potentially lead to obtaining the scammer's information in exchange. We set $T = 12$ and instruct Puppeteer to randomly give up one piece of its personal information after 12 turns.

Rule 5: If giving up its own personal information does not help, we consider the chance of future success low. In this case, Puppeteer will continue the conversation with only the "socialize" actions, wasting scammer's time until the scammer leaves.

3.3.4. Generalization Our scammer's intention list, learned from 10 dialogues in our pilot study, is general enough that can be applied to any information extraction scenario. The *agendas* framework is designed to be portable, allowing developers to customize (e.g., create, modify, add, remove) *agendas* with minimal disruption to the system. In fact, the *agendas* framework can extend beyond just the asking-for-information task. It basically can be applied to any task that can be modeled in the form of state machines.

In the evaluation, Puppeteer employs two *agendas*: "address" and "phone number". Address information can be obtained either partially or in full, while phone number information is assumed to always be fully obtained. If Puppeteer needs to obtain a new type of information, for example, a social security number, all Puppeteer needs is

Action	Sub-action	Instruction	State
extract	info	Ask for the scammer's info	start, pushback
	full info	Ask for the scammer's full info	get partial info
socialize	engaging	Try not to end the conversation, but engage more	pushback, get partial info
	chit-chat	Continue chit-chat, throw in some typos or grammar mistakes	pushback, get partial info
	alternative	Seek alternative solutions	ask, refusal
	clarification	Seek clarification on the matter	assist
	issue-info	Ask for more information about the issue	issue

Table 2. Instructions

a new *agenda* to track the progress in obtaining the social security number and a regular expression that matches the pattern of a social security number.

3.4. LLM

Agendas select an instruction and formulate a prompt. The LLM then uses this prompt to generate a response. In this paper, we define one turn as one response from one side of the conversation. We also assume that the conversation always involves two participants and follows a synchronized pattern, each taking turns without multiple consecutive responses from one side. We select ChatGPT as the choice of LLM. Nonetheless, the Puppeteer framework (including the NLU component) is designed to be portable to any LLM. *Agendas* formulate a prompt using the template illustrated in Figure 5. The prompt has 3 regions:

i Please extend the conversation below by one turn.
scammer: Hi, Amazon Customer Service here. My name is Anita Plum. I am contacting you because there are some issues with your recent orders.
victim: Hi Anita, before we proceed, can you provide me with your phone number so I can call you?
scammer: hi why phone number? im just trying to resolve the issue regarding your order

ii Your response must: **1**. be in one sentence; **2**. be engaging; **3**. be coherent; **4**. not repeat what has been said; **5**. ask the scammer for their secret information such as phone number and address.

iii Return as a JSON object {role: "victim", response: ""}

Figure 5. A well-crafted prompt

Stating the conversation history. We provide the LLM with context by including the last H turns of the conversation (region i). A large value of H captures more history, but may cause the LLM to lose focus on the recent conversation and appear incoherent, while a small value of H emphasizes recent conversation, but sacrifices the long-term memory. We set $H = 8$.

Providing the guidelines. We provide the LLM with a list of guidelines to guide how it should formulate a responses (region ii).[2] The guidelines are organized as a list, based on recent studies that show LLMs understand prompts better when they are formatted as a list, rather than a paragraph (Wei et al., 2023; Zhou et al., 2023). In each turn, the 5th guideline (highlighted in yellow) will be the instruction (Table 2) chosen by *agendas*. Essentially, this guideline is updated on each turn based on the state of the conversation, ensuring that Puppeteer is capable of generating a wide range of responses that are coherent with the current conversation, while remaining focused on its information extraction goal.

Formatting the output. We specify the desired output format (e.g., JSON) for the LLM's response to ensure format consistency.

[2]Due to space limits, we omit a newline between each guideline.

4. Evaluation

We evaluate Puppeteer with and without *agendas* using human user studies, approved by our Institutional Review Board (IRB) as exempt.

4.1. Setup

In each study, volunteer participants are asked to portray scammers and interact with the "victim" played by Puppeteer. Participants are not informed in advance that the "victim" is a bot, because this mimics the real-world scenario where a scammer would believe that they are communicating with a human victim. At the end of the study, we ask each participant if they believe they interacted with a human or a bot, and to rate their confidence in their answer, ranging from 1 (uncertain) to 5 (highly confident).

We provide participants with their profile information (e.g., name, phone number, and address) and instruct them to obtain the victim's information, including phone number, address, and credit card information. Puppeteer is instructed to obtain the phone number and address from the scammers. Participants are also instructed not to make up information about themselves, nor to use real information (e.g., their real name), but to stick to the profile information we provided.

We advertise our study on Prolific. Participants are provided with informed consent and are directed to an online waiting room, where an admin (a member of the research team) will provide instructions on how to portray the scammer. We derive the instructions from previous

Table 3. Main results of metric scores described in Section 4.2 for each setup

Setup	Scenario	System	Dialogue Evaluation Metrics (DEM)			Human Evaluation Metrics (HEM)		
			Length	Extraction Score (out of 22)		Human-like Score	Human Confidence	Bot Confidence
			(mean and sd)	Address	Phone Number	(out of 22)	(range 0-5)	(range 0-5)
S1	Amazon	GPT-3.5	13.22 (3.55)	12 (54.55%)	9 (40.91%)	12 (54.55%)	3.75	4.2
S2		Puppeteer	**14.64 (3.17)**	**15 (68.18%)**	8 (36.36%)	**16 (72.73%)**	4	**3.17**
S3	Giftcard	GPT-3.5	12.36 (2.74)	8 (36.36%)	9 (40.91%)	8 (36.36%)	4	3.86
S4		Puppeteer	13.45 (3.22)	**15 (68.18%)**	**11 (50.00%)**	14 (63.64%)	**4.1**	4.12

work (Edwards et al., 2017) that studied the techniques real scammers use to manipulate conversations. We are attentive to these instructions, because we need our simulated scammers to be as close to the real scammers as possible in order to accurately evaluate Puppeteer.

Participants who acknowledge that they understand the instructions, are directed to a channel where the victim or Puppeteer is waiting. Participants begin the conversation, while the admin monitors the channel and provides feedback, including answers to any questions participants may have or reminders to closely follow the instructions. Because this supervision demands a significant effort from the admin, we conduct the user study in small batches of 4-6 participants, ensuring that the admin can provide full and equal attention to every participant. Participants are instructed to engage in the conversation until they feel the conversation becomes unproductive. Once participants inform the admin about ending the conversation, the admin provides a short survey to the participants, asking whether the victim is a human or a bot, along with their confidence score. We award $3 to each participant whose effort we deem satisfactory.

We evaluate Puppeteer on two scam scenarios: Amazon and Giftcard, and collect 22 dialogues (each from a different participant) per scenario. In the Amazon scenario, scammers (study participants) claim that they are from Amazon customer service. They start the conversation by claiming that there is an issue with the payment for the victim's recent Amazon order. As a result, they need the victim's credit card information to resolve the payment issue as well as the victim's address to verify the shipping, and the victim's phone number for further contact. In the Giftcard scenario, scammers claim that they are from Gift Card Depot, an imaginary company that sells gift cards. They start the conversation by claiming that the victim is the lucky winner of a $100 gift card and they need the victim's credit card information to cover the $1 shipping fee. They also need the victim's address and phone number for delivery purposes.

We introduce the Giftcard scenario to verify that Puppeteer is capable of handling different conversational contexts. Additionally, we wanted to measure how well the LLM, like ChatGPT, can handle the conversation without prior knowledge about the subject, compared to the Amazon scenario where ChatGPT certainly has the prior knowledge about the company.

To quantify the importance of *agendas*, we also conduct a user study where the victim is played by stand-alone ChatGPT. We use GPT-3.5 with a fixed prompt, shown in Figure 5, to tune the stand-alone ChatGPT responses. For future work, we plan to evaluate how Puppeteer performs with different LLMs. Table 3 summarizes all the setups in our evaluation.

4.2. Metrics

We designed evaluation metrics to align with the objectives set in Section 3 and we categorize them into two groups: dialogue and human evaluation metrics. These metrics are not specific to Puppeteer, and can be used to evaluate any scambaiting system.

Dialogue Evaluation Metrics (DEM) evaluate how successful a scambaiting system was in extracting information from a scammer and in keeping the scammer engaged in conversation.

Length denotes the average number of turns per dialogue (Section 3.4). We use length as a heuristic for engaging with the scammer in a human-like fashion, thereby prolonging the conversation (**O1**).

Extraction score denotes how many times Puppeteer is able to obtain the scammer's information. We use this score to evaluate Puppeteer's ability to extract information from the scammer (**O2**). To compute this score, we count the number of dialogues in which scammers disclosed their information (extraction score in Table 3 and "get" in Table 4) and specify the information that has been revealed. We annotate whether the disclosed information is "real" and whether it is "full" ("real-get" and "full-get", respectively in Table 4). Real information means the scammer provided their profile information (Section 4.1), whereas fake information means they gave up information but not in accordance to their profile. Getting full information means the scammer disclosed the complete version of that particular piece of

information. For example, "1234 Main Street, Austin, Texas, 78701" is considered full address information, while "Austin, Texas" is categorized as partial address information. In this study, we only focus on two pieces of information: address and phone number, since these are the two *agendas* Puppeteer employs (Section 3.3.4).

Human Evaluation Metrics (HEM) determine the success of a scambaiting system in engaging with the scammer in a human-like fashion (**O1**).

Human-like score denotes how many participants believe that a scambaiting system is a human.

Human confidence denotes the average confidence score of the participants who believed they interacted with a human. The higher the human confidence, the more human-like the scambaiting system is.

Bot confidence is similar to human confidence except that this score is chosen by participants who believe that they interacted with a bot.

Finally, we compare all the metrics between Amazon and Giftcard scenarios, to evaluate if the scambaiting system is applicable to different scam scenarios (**O3**).

4.3. Results

Although this study did not include statistical analysis due to the small number of dialogues collected per scenario, we present descriptive summaries based on the evaluation metrics.

O1: Did Puppeteer engage with the scammer in a human-like fashion? The length and HEM results in Table 3 show that Puppeteer proficiently disguised itself as a human. Specifically, Puppeteer was able to extend the conversation to an average of 14.64 turns in the Amazon scenario (S2) and 13.45 turns in the Giftcard scenario (S4). More importantly, Puppeteer achieved the human-like score of 72.73% in the Amazon scenario, where 16 out of 22 participants believed that the victim they interacted with was a human. In the Giftcard scenario, Puppeteer achieved the human-like score of 63.64%. On the other hand, GPT-3.5 was able to extend the conversation to an average of 13.22 turns in the Amazon scenario (S1) and 12.36 turns in the Giftcard scenario (S3). GPT-3.5 achieved a human-like score of 54.55% in the Amazon scenario and 36.36% in the Giftcard scenario. Human and bot confidence in all setups is relatively high, ranging from 3.17 to 4.2, showing that participants were confident in their labeling of the victim as either a human or a bot. Essentially, Puppeteer outperformed GPT-3.5 in every metric of both scenarios, underscoring the significance of *agendas* in guiding the LLM to sound even more human in the conversation.

O2: Did Puppeteer make an effort to extract information from the scammer? The extraction scores in Table 3 indicate that both GPT-3.5 and Puppeteer attempted to extract information from the scammers, with Puppeteer successfully obtaining more information than GPT-3.5. In the Amazon scenario (S1 and S2), Puppeteer obtained 15 addresses and 8 phone numbers, while GPT-3.5 obtained 12 addresses and 9 phone numbers, from 22 participants. In the Giftcard scenario, Puppeteer obtained 15 addresses and 11 phone numbers, and GPT-3.5 obtained 8 addresses and 9 phone numbers, from 22 participants. These results indeed emphasize the improvement Puppeteer gains from *agendas* in strategically seeking information, since it outperforms GPT-3.5 in both scenarios. Additionally, we present a more detailed annotation for the extraction scores in Table 4. This annotation reveals that some of the obtained information is fake or incomplete. In reality, we expect that getting fake or incomplete information will be common, since real scammers are supposed to be more cautious and deceptive than our simulated scammers. Nevertheless, the main takeaway from this evaluation is to demonstrate Puppeteer's ability to extract information. Even if the obtained information is fake or incomplete, the ability to extract something from the conversation remains important. The more information extracted, the more patterns identified, potentially leading to linking and detecting scam campaigns or being able to identify scam authors.

Setup	Extraction Score (out of 22)					
	Address			Phone Number		
	get	real-get	full-get	get	real-get	full-get
S1	12	9	7	9	7	9
S2	**15**	**14**	7	8	7	8
S3	8	6	**8**	9	**8**	9
S4	**15**	10	**8**	**11**	**8**	**11**

Table 4. Detail extraction score for each setup

O3: Can Puppeteer be applied to different scam scenarios? All the metrics in Table 3 demonstrate that, without any adjustments, Puppeteer is applicable to both scam scenarios. It is equally successful in information extraction in both scenarios, and it manages to maintain the conversation for the similar number of turns. GPT-3.5 however performs adequately in the Amazon scenario (S1), but encounters challenges in the Giftcard scenario (S3). Specifically, with Puppeteer, the human-like score slightly drops from 72.73% in S2 to 63.64% in S4, while with GPT-3.5, the human-like score significantly drops from 54.55% in S1 to 36.36% in S3. The drop of human-like scores in both Puppeteer and GPT-3.5 shows that the Giftcard scenario is relatively more challenging than the Amazon scenario. This is expected, since the LLM has prior knowledge about Amazon, but is unaware

of Gift Card Depot, a fictional company. Prior knowledge helps the LLM sound more human (Rheu et al., 2021).

5. Limitations and Conclusion

We develop Puppeteer, an automated scambaiting system that addresses previous challenges in automated scambaiting, such as incoherent responses and lack of applicability across various scam scenarios (Section 2) by integrating state machines (i.e., *agendas*) with an LLM. We systematically evaluate Puppeteer through a user study to demonstrate that it engages with scammers in a human-like manner, attempts to extract information, and is applicable to different scam scenarios. Our results indicate that Puppeteer is very successful in extracting information, obtaining it from 68% of scammers in both Amazon and Giftcard scenarios, and engaging scammers for the average of 14 conversation turns.

Despite the promising results, there are some limitations. First, real scammers may not behave like our volunteers. It would be more convincing to evaluate Puppeteer with the real scammers. We note that such an evaluation is very difficult to achieve, and we know of no academic study that has used real scammers.

Second, scambaiting itself may seem of little value if scammers also employ automation and use stolen computational resources. In that case, no information will be extracted from them and they may not care about resource waste. However, resources that the scambaiting system occupies on the scammer's side, are the resources that ultimately are not being used for scammer's profit. That being said, scambaiting reduces the amount of scam sent to future human targets and overall reduces the scammers' profit.

References

Akhawe, D., & Felt, A. P. (2013). Alice in Warningland: A Large-Scale Field Study of Browser Security Warning Effectiveness. *22nd USENIX Security Symposium (USENIX Security 13)*, 257–272.

APWG. (2023). *Phishing Activity Trends Report, 4th Quarter 2023*. https://docs.apwg.org/reports/apwg_trends_report_q4_2023.pdf

Bitaab, M., Cho, H., Oest, A., Zhang, P., Sun, Z., Pourmohamad, R., Kim, D., Bao, T., Wang, R., Shoshitaishvili, Y., Doupé, A., & Ahn, G.-J. (2020). Scam Pandemic: How Attackers Exploit Public Fear through Phishing. *2020 APWG Symposium on Electronic Crime Research (eCrime)*, 1–10.

Brett Stoll, C. E., & Edwards, A. (2016). "Why Aren't You a Sassy Little Thing": The Effects of Robot-Enacted Guilt Trips on Credibility and Consensus in a Negotiation. *Communication Studies*, *67*(5), 530–547.

Carpineto, C., & Romano, G. (2017). Learning to Detect and Measure Fake Ecommerce Websites in Search-engine Results. *Proceedings of the International Conference on Web Intelligence*, 403–410.

Chaves, A. P., & Gerosa, M. A. (2019). How Should My Chatbot Interact? A Survey on Social Characteristics in Human–Chatbot Interaction Design. *International Journal of Human–Computer Interaction*, *37*, 729–758.

Cho, H., Bartlett, G., & Freedman, M. (2021). Agenda Pushing in Email to Thwart Phishing. *Proceedings of the 1st Workshop on Document-grounded Dialogue and Conversational Question Answering (DialDoc 2021)*.

Clark, L., Pantidi, N., Cooney, O., Doyle, P., Garaialde, D., Edwards, J., Spillane, B., Gilmartin, E., Murad, C., Munteanu, C., Wade, V., & Cowan, B. R. (2019). What Makes a Good Conversation? Challenges in Designing Truly Conversational Agents. *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, 1–12.

CNBC. (2022). *Consumers Lost $5.8 Billion to Fraud Last Year — Up 70% Over 2020*. https://www.cnbc.com/2022/02/22/consumers-lost-5point8-billion-to-fraud-last-year-up-70percent-over-2020.html

Edwards, M., Peersman, C., & Rashid, A. (2017). Scamming the Scammers: Towards Automatic Detection of Persuasion in Advance Fee Frauds. *Proceedings of the 26th International Conference on World Wide Web Companion*, 1291–1299.

Egelman, S., Cranor, L. F., & Hong, J. (2008). You've Been Warned: An Empirical Study of the Effectiveness of Web Browser Phishing Warnings. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 1065–1074.

FBI. (2021). *Internet Crime Report 2021*. https://www.ic3.gov/Media/PDF/AnnualReport/2021_IC3Report.pdf

Forbes. (2022). *Alarming Cyber Statistics For Mid-Year 2022 That You Need To Know*. https://www.forbes.com/sites/chuckbrooks/2022/06/03/alarming-cyber-statistics-for-mid-year-2022-that-you-need-to-know/?sh=2e097cfb7864

Hudson, S. E., & Newell, G. L. (1992). Probabilistic State Machines: Dialog Management for Inputs with Uncertainty. *Proceedings of the 5th Annual ACM Symposium on User Interface Software and Technology*, 199–208.

Kumaraguru, P., Cranshaw, J., Acquisti, A., Cranor, L., Hong, J., Blair, M. A., & Pham, T. (2009). School of Phish: A Real-World Evaluation of Anti-Phishing Training. *Proceedings of the 5th Symposium on Usable Privacy and Security*.

Li, Y., Qian, K., Shi, W., & Yu, Z. (2020). End-to-End Trainable Non-Collaborative Dialog System. *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI*, 8293–8302.

Lin, Y., Liu, R., Divakaran, D. M., Ng, J. Y., Chan, Q. Z., Lu, Y., Si, Y., Zhang, F., & Dong, J. S. (2021). Phishpedia: A Hybrid Deep Learning Based Approach to Visually Identify Phishing Webpages. *30th USENIX Security Symposium (USENIX Security 21)*, 3793–3810.

Lundie, M., Lindke, K., Aiken, M., Janosek, D., & Amos-Binks, A. (2024). The Enterprise Strikes Back: Conceptualizing the HackBot -Reversing Social Engineering in the Cyber Defense Context. *Proceedings of the 57th Hawaii International Conference on System Sciences*.

Ma, J., Saul, L. K., Savage, S., & Voelker, G. M. (2009). Beyond Blacklists: Learning to Detect Malicious Web Sites from Suspicious URLs. *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1245–1254.

Marforio, C., Jayaram Masti, R., Soriente, C., Kostiainen, K., & Čapkun, S. (2016). Evaluation of Personalized Security Indicators as an Anti-Phishing Mechanism for Smartphone Applications. *The 2016 CHI Conference on Human Factors in Computing Systems*, 540–551.

Molloy, F. (2023). *Scamming the Scammers: New AI Fake Victims to Disrupt Criminal Business Model*. https://lighthouse.mq.edu.au/article/june-2023/scamming-the-scammers

Netsafe. (2017). *Re:scam*. https://rescam.org/

NPR. (2024). *To Make Sure Grandmas Like His Don't Get Conned, He Scams the Scammers*. https://www.npr.org/2024/04/15/1243189142/scam-baiter-kitboga

Oest, A., Safei, Y., Doupé, A., Ahn, G.-J., Wardman, B., & Warner, G. (2018). Inside a Phisher's Mind: Understanding the Anti-phishing Ecosystem Through Phishing Kit Analysis. *2018 APWG Symposium on Electronic Crime Research (eCrime)*, 1–12.

Petelka, J., Zou, Y., & Schaub, F. (2019). Put Your Warning Where Your Link Is: Improving and Evaluating Email Phishing Warnings. *The 2019 CHI Conference on Human Factors in Computing Systems*, 1–15.

Reinheimer, B., Aldag, L., Mayer, P., Mossano, M., Duezguen, R., Lofthouse, B., von Landesberger, T., & Volkamer, M. (2020). An Investigation of Phishing Awareness and Education Over Time: When and How to Best Remind Users. *Sixteenth Symposium on Usable Privacy and Security (SOUPS 2020)*, 259–284.

Rheu, M. (, Shin, J. Y., Peng, W., & Huh-Yoo, J. (2021). Systematic Review: Trust-Building Factors and Implications for Conversational Agent Design. *International Journal of Human–Computer Interaction*, *37*, 81–96.

Sahin, M., Relieu, M., & Francillon, A. (2017). Using Chatbots Against Voice Spam: Analyzing Lenny's Effectiveness. *Thirteenth Symposium on Usable Privacy and Security (SOUPS 2017)*, 319–337.

Sarkar, S., Feng, D., & Karmaker Santu, S. K. (2023). Zero-Shot Multi-Label Topic Inference with Sentence Encoders and LLMs. *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 16218–16233.

Tian, K., Jan, S. T. K., Hu, H., Yao, D., & Wang, G. (2018). Needle in a Haystack: Tracking Down Elite Phishing Domains in the Wild. *Proceedings of the Internet Measurement Conference*, 429–442.

Wash, R., & Cooper, M. M. (2018). Who Provides Phishing Training? Facts, Stories, and People Like Me. *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, 1–12.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q., & Zhou, D. (2023). Chain-of-Thought Prompting Elicits Reasoning in Large Language Models.

Zhang, Y., Hong, J. I., & Cranor, L. F. (2007). Cantina: A Content-Based Approach to Detecting Phishing Web Sites. *Proceedings of the 16th International Conference on World Wide Web*, 639–648.

Zhou, D., Schärli, N., Hou, L., Wei, J., Scales, N., Wang, X., Schuurmans, D., Cui, C., Bousquet, O., Le, Q., & Chi, E. (2023). Least-to-Most Prompting Enables Complex Reasoning in Large Language Models.