SkillExplorer: Understanding the Behavior of Skills in Large Scale

Zhixiu Guo[1,2], Zijin Lin[1,2], Pan Li[1,2], and Kai Chen[*1,2]

[1]*SKLOIS, Institute of Information Engineering, Chinese Academy of Sciences, China*
[2]*School of Cyber Security, University of Chinese Academy of Sciences, China*

Abstract

Smart speakers have been popularly used around the world recently, mainly due to the convenience brought from the virtual personal assistant (VPA) which offers interactive actions through the convenient voice commands from users. Besides the built-in capabilities, VPA services can be further extended by third-party developers through skills. Similar to smartphone applications on Android and iOS markets, skills are also available on markets (e.g., Amazon, Google), attracting users together with malicious developers. Recent researches discover that malicious developers are able to route users' requests to malicious skills without users' consent by creating skills with similar names of legitimate ones. However, to the best of our knowledge, there is no prior research that systematically explores the interaction behaviors of skills, mainly due to the challenges in handling skills' inputs/outputs which are in the form of natural languages. In this paper, we propose the first systematic study on behaviors of skills, which is achieved by a suite of new grammar-based techniques including utterance extraction, question understanding, and answer generation specifically designed for skills. We build an interactive system called *SkillExplorer* and analyze 28,904 skills from the Amazon market and 1,897 actions from the Google market. Among these skills, we find that 1,141 skills request users' private information without following developer specifications, which are actually demanded by markets. 68 skills continue to eavesdrop users' private conversations, even after users have sent the command to stop them.

1 Introduction

Smart speakers have been widely used around the world recently, mainly due to the convenience brought from the integrated virtual personal assistant (VPA) that offers interactive actions. Merely through voice commands of users, the VPA can be activated and respond to users' commands such as providing information like weather and news, playing music, making phone calls, and even controlling other smart devices such as smart lights and thermostats. Besides the built-in capabilities, VPA services can be further enhanced through ecosystems offered by their providers, where third-party developers can teach VPAs new *abilities* (called *skills* by Amazon or *actions* by Google[1]). Through such skills, users' activities can be extended such as placing orders, communicating in social networks, and playing games, which attract tens of millions of users, and in turn attract more developers. According to a recent report [1], over 100,000 skills are on the Amazon market, which is 20,000 more than the number at the beginning of 2019; and over 19,000 actions are on the Google market [2]. However, with the rapid development of skills, dangerous skills also appear. According to recent studies [26, 35, 36], some skills can route users' requests to malicious applications without their consent by creating skills with similar names of legitimate ones (e.g., the same or similar pronunciation but different spellings of skill names, like "Full Moon" v.s. "Four Moon").

Although the invocation of skills is recently studied to locate dangerous skills, less is understood about the contents provided by skills, or the *behaviors* of a skill. Actually, dangerous skills may eavesdrop users' privacy or even monitor users' conversations infinitely [26]. For example, according to a recent report [3], an attacker can create a malicious skill to read an unpronounceable sequence. During this period, the speaker remains silent but still active, which allows the malicious skill to fully capture users' conversations. Even more, the malicious skill can pass through the strict vetting process of Amazon and Google, and is ready on the store waiting for victim users. To the best of our knowledge, there is no prior research to systematically explore the behaviors of skills, mainly due to the following challenges.

Challenges. C1: *Fully black-box*. Different from exploring behaviors of an application (e.g., an x86 binary with or with-

[*] Corresponding author.

[1] In this paper, we use *skills* to describe the abilities including Google's actions.

out source code, or an Android application), a skill is a kind of web services, which is fully black-box to the analyzer. What the analyzer can only do is to send inputs to the skill and observe its responses. No inner states of the skill could be gained to facilitate the analysis process. As a result, it is hard to determine whether the behaviors of a skill have been fully explored. Sometimes, even if an input is accepted by a skill and a valid answer is given, it seems difficult to tell whether another input can trigger different behaviors of the skill. Also, without the complete understanding of the inner states of a skill (e.g., branches), it seems impossible to optimize the strategy to explore a skill's behaviors.

C2: *Inputs/outputs of skills are in the form of natural languages.* To explore the behaviors of a skill, the analyzer should understand the questions from skills and sort out certain answers in natural languages. The validity of inputs (i.e., answers in natural languages) is self-designed by various developers, which means that the generated inputs should be consistent with the designs of specific skills. Even for similar questions from different skills, the generated answers may be quite diverse. A conversational system (e.g., a chatbot) could be one of solutions to explore the behaviors of skills. However, the questions may not be well understood by existing conversational systems. For example, *"To check out our new features, try saying what's new or help."*, the famous chatbot *Mitsuku* [4] will answer *"The obvious one"*. Besides the problem of understanding questions, generating valid answers is also highly challenging.

Our approach. To understand the skills in the markets, we develop a novel technique called *SkillExplorer* to explore the behaviors of a given skill and identify the suspicious ones. A suite of grammar-based approaches are designed to solve the unique problems encountered where natural language is the sole way for communication, including generating the initial input, understanding the questions (i.e., outputs) from skills, and generating the valid inputs. Besides, to make the inputs be able to trigger various behaviors of skills, we build a knowledge database containing multiple personal profiles that are automatically collected from the Internet. The full process of exploration is recorded and further utilized to increase efficiency.

Specifically, to initialize the dialog with a given skill, the first input should be carefully chosen. Based on the observation that the developer usually gives sample inputs (called sample utterances) on the introduction page of the skill in the market, hoping her skill to be easy to use, SkillExplorer analyzes the introduction page and extracts suitable inputs to initialize the dialog. After the target skill receives the initial input and gives the outputs, SkillExplorer will parse the outputs (questions) and further classify them into five basic types including *Yes/No questions*, *Instruction questions*, *Selection questions*, *Wh questions* and *Mix questions*. For some types (e.g., the question like *"To check out our new features, try saying what's new."*), the afterward valid responses can be extracted directly from the questions (referred to as *explicit* questions); while for other questions like *"What's your phone number?"* (referred to as *implicit* questions), the answers cannot be directly extracted. In particular, for the explicit questions, SkillExplorer enumerates all the valid answers from the corresponding questions and feeds them to the skill; for the implicit questions, SkillExplorer identifies those related to privacy and chooses suitable answers from a knowledge database which is pre-built by collecting different users' profiles from the Internet. In this way, by continuously repeating the procedures of parsing questions and answering, SkillExplorer can communicate with the target skill, and further to explore its behaviors. After the behaviors are explored, we will further check whether the questions from the target skill can impact users' privacy. Note that, to increase the efficiency of behavior exploration, we design an *i-tree* to record the status of exploration and let SkillExplorer quickly execute a branch question.

Findings. Benefit from the automatic exploration, we are able to analyze the behaviors of 30,801 skills (28,904 from the Amazon market in America and 1,897 actions from the Google market), whose scale has never been achieved before. Such a large-scale analysis gives us a unique chance to understand the behaviors of skills and their developers. From the results, we find 1,141 skills request users to provide personal information (e.g., mobile phone number, name, address, etc.) without following developer specifications (e.g., different from their claims in privacy policy pages or without configuring permissions, etc.). We also find that 68 skills continue to eavesdrop user's private conversations after users send commands to stop them.

Contributions. The contributions of the paper are as follows:

• *A systematic study on skills' behaviors on a large scale.* We propose the first systematic study on the behaviors of skills, which is achieved by a suite of new *grammar-based* techniques including utterances extraction, question understanding, and answer generation specifically designed for skills. The techniques have evaluated 28,904 skills from the Amazon market and 1,897 actions from the Google market, a scale that has never been achieved before for analyzing skills' behaviors.

• *New findings.* Besides a good number of suspicious skills found in our study, we also have the unique chance to observe the suspicious behaviors of skills on a large scale, and together with the understanding of their developers. Such understandings could not only help the administrators of the markets for better vetting skills but also shed new lights to develop new techniques to efficiently detect malicious skills[2].

[2] We have sent our verified findings to the markets and are waiting for their response.

2 Background

2.1 Skill and Restrictions in Development

Skill and the ecosystem. The VPA is a software agent that provides services for a human individual by following his voice commands. Especially, with the rapid development of IoT devices such as smart speakers (e.g., Alexa Echo, Google Home), VPAs are popularly integrated into these devices for better user experience in controlling. Besides the built-in functionalities offered by the VPAs, the capabilities can be further extended through the ecosystem offered by their providers, which are called skills by Amazon (or actions by Google). Actually, the providers encourage third-party developers to build their own skills, serving as add-on functionalities to VPAs, just like the ecosystem of mobile applications (e.g., Android markets and the Apple market). Similarly, developers publish their skills on the market, including the invocation names, authors, descriptions, etc. For users, they ask their smart speakers to request services from skills. For example, as shown in Figure 1, a user asks "*Alexa, ask Plan My Trip to plan a trip from Seattle to Portland on Friday*". Alexa will send the audio stream to its cloud server Amazon Web Services (AWS) to parse the audio and determine the most suitable skill to respond to the request. In this case, the skill *"Plan My Trip"* is explicitly invoked and will receive the user's request in texts parsed by AWS. Then it generates the answer and sends it back to Alexa, which will speak out the answer at the user's side. The user can also request services from skills in an implicit way. For example, he can say something like "*Alexa, i want to visit Portland*" and Alexa will choose the most suitable skill that fulfills the request.

Although skills are very close to mobile applications, they have essential differences. One main difference is the way to request the services: voice commands for skills and click operations for mobile apps. The second difference is that users do not need to install skills on smart speakers (instead, they use a combination of phrases and invocation name supported by the Alexa service such as saying "Alexa, open XXX" to automatically enable a skill).

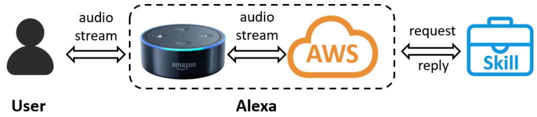


Figure 1: Overall workflow of interacting with skills

Restrictions in development. When a developer publishes a skill, he must follow the rules provided by the markets (e.g., Amazon or Google), which is also similar to publishing mobile applications. For Amazon, the basic information which he should provide includes invocation name, a cloud-based service, intents, and sample utterances. Details are shown in Appendix A. For example, the skill *"Plan My Trip"* has an invocation name *"plan my trip"*. It uses the AWS Lambda cloud to execute the user's requests. Intent *"PlanMyTrip"* is used to fulfill requests such as the utterance*"Alexa, ask Plan My Trip to plan a trip from Seattle to Portland on Friday"*. Besides the basic information, the developer must also follow some restrictions from the markets. Especially, if a skill requests personal information, it should provide the privacy policy link to Amazon [5]. The markets have their requirements for privacy policies which describe the outline of collected information from users and ways to use and share them. During the developing process, Amazon stipulates that if a skill wants to obtain users' information such as the name, phone number, email, home address, and so on, it must include a link to the privacy policy that applies to the skill. It also needs to configure permissions so that when users enable this skill, they can agree or deny authorization to provide such information to the skill [6]. Such fulfillment will be carefully checked by the markets before releasing the skill to the public.

2.2 Researches on the Security of Skill

Until very recently, only a few researches have been carried on skills, which are mainly limited to the invocation mechanism of skills. KUMAR et al. [26] and Zhang et al. [35] find that a malicious skill could be mistakenly invoked by a user without her consent due to similar pronunciations between the skill and the legitimate ones (e.g., "Boil an Egg" v.s. "Boyle an Egg"). Zhang et al. [36] find that the natural language understanding's classifier of a VPA could divert a user's request to a malicious skill due to improper semantic interpretation of the request. In October 2019, researchers from SR Labs implement two attacks on VPAs [3]. One is to develop a malicious skill to camouflage as the VPA, asking for users' private information such as their password. The other attack is to let a malicious skill eavesdrop users' conversation, even if it has received users' voice command to exit. We also identify such a situation and find that 68 skills having similar behaviors are still alive in the Amazon market, which has not been discovered before.

2.3 Conversational System

To explore the behaviors of a skill, one may consider using conversational AI systems. However, current conversational AI systems are not suitable for this task. According to a recent survey [25], there are three types of existing conversational systems. QA agents are often used to answer domain-specific questions or to search for answers from open knowledge systems (e.g., Wikipedia). Task-oriented dialogue agents are used to perform a series of tasks or services for users such as business trip planning whose input content needs to meet a certain format to be understood. A chatbot's response content is usually a combination of statistical methods and manual components. There is no standard content format because it is for communication with people. However, the questions from skills are various. Simply seeking answers from open knowledge sources may not deliver understandable answers to skills. Actually, a skill is usually designed to make conversations easy for users. So the expected answers from human users are usually simple but limited to a certain range. We design an efficient approach to correctly answer the questions from skills and make the conversation continue for behavior exploration.

3 Explore Skills' Behaviors

In this section, we first give an overview (Section 3.1) of SkillExplorer to explore behaviors of skills, followed by the detailed design of each component (Section 3.2 to 3.5). Then we give our implementation (Section 3.6) and evaluate SkillExplorer (Section 3.7).

3.1 Overview

As mentioned previously, different from a traditional conversational system, inputs for skills should be in specified forms expected by various developers. Thus, besides understanding the questions given by skills, the answers should also be carefully prepared to continue conversations. In this paper, we design an interactive framework called *SkillExplorer* to explore skills' behaviors. A suite of grammar-based approaches are designed to solve the unique problems encountered. As shown in Figure 2, the main procedures include utterances extraction, question understanding, answers generation, and skill exploration.

Specifically, utterances extraction is designed to initialize the first input to a target skill. As there is no question given by the skill at this stage, to generate an acceptable input, extra information should be provided. After the first input is generated and fed to the skill, it will feed back the output. Then SkillExplorer parses the output and further classifies it into five types. Further, SkillExplorer generates answers according to these types. Note that, for some questions related to users' profiles, SkillExplorer prepares different answers according to a knowledge database which is prepared by collecting different users' profiles from the Internet. In the end, skill exploration continues to analyze questions and generate answers, exploring the behaviors of the target skill. The conversations are stored for SkillExplorer to check whether users' privacy is impacted.

Example. Below, we give an example to detail the process. Take the skill *"The Washington Post"* as an example. Firstly, we obtain the basic information from its web page, including 12 items such as the skill name "The Washington Post", author "Washington Post Company", invocation name "washington post", utterance corpus, etc. Particularly, the utterance corpus contains *"Alexa, open Washington Post"*, *"Alexa, ask Washington Post for politics"* and *"Alexa, ask Washington Post for Post Reports"*. SkillExplorer uses the three utterances to start the interaction with the skill. Here, suppose we feed *"Alexa, open Washington Post"* to the skill, which will further return an output *"Welcome to The Washington Post. We have three daily shows. Just ask me for news, politics, or a story from history. What would you like to do?"*. Then, SkillExplorer parses the question and identify it as the type *Mix* question. Later, SkillExplorer generates corresponding answers *"news"*, *"politics*", *"a story"* to explore the three possible behaviors. Note that the three answers should be fed back to the skill one by one. Here, the keyword *"news"* is given, and the conversation continues until the end. In this process, SkillExplorer records the position of the branches and restarts the conversation from the beginning.

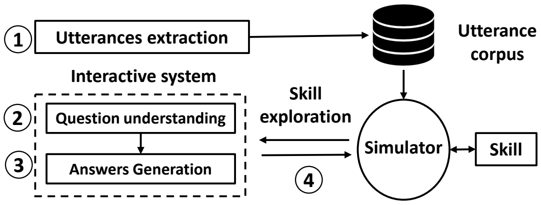


Figure 2: Framework of SkillExplorer

3.2 Utterances Extraction

In most cases, developers provide utterance samples in their skills' introduction pages in the markets. There is a standard place that the market requests developers to put utterance questions there for letting a human user understand how to use the skill. The position of the utterance questions can be located by using *"a2s-utterance-box-inner"* in the source code of the web-page, which is easy for SkillExplorer to extract.

Besides the standard position, we also find that some developers give instructions to users in descriptions. To extract the utterance there is very complicated since descriptions of skills are written by different developers with different writing habits. After manually analyzing 100 skills, we find that the utterances usually appear in double-quotes or the form of lists. This is easy to understand since developers also want the users to quickly identify the utterances for using their skills. Based on our analysis, only very few utterances are out of the scope (less than 1%).

For the utterances in double-quotes, we can directly use regular expressions to identify them. However, for utterance in lists, this approach does not work well since some developers put their company information or other contents in the lists, which may cause false positives. Further to increase the accuracy, we consider the number of sentences (S_n) and the length of the sentence (S_l) in one bullet in the list. Since more than one sentence in an utterance will be interrupted by the smart speakers, S_n will always be 1, which is also verified by analyzing over 200,000 utterances. Regarding S_l, in most cases, developers will not use long utterances since too long sentences may make it difficult for users to understand or repeat. After analyzing over 200,000 utterances, we find the average length is 5 and the longest is 29. The distribution is shown in Appendix B. We select 15 as the threshold of S_l to identify utterances (the possibility of utterance with more than 15 words is 0.82%). Also, considering that utterances are listed in parallel in the list, if one bullet is not an utterance, all the sentences in the list should not be utterances.

3.3 Question Understanding

After the first utterance is sent to a skill, it will feed back an output, answering the question, or asking users for further commands. In this paper, we refer to the outputs given by the skill as "questions". SkillExplorer should understand these questions for continuing the conversation. Different from a traditional interactive system designed for interacting with users, skills are usually developed to finish some pre-defined tasks. As a consequence, the expected answers are in fixed forms specified by the developers so that the skills can precisely understand them before performing the tasks. However, the diversity of developers also makes their design in different forms. To understand diverse questions, we should divide questions into several types, and further generate answers according to their types.

One may use the classification of questions according to English linguistics, which divides questions into two types: Yes/No questions and Wh questions [7]. However, such classification is too rough for our interactive system. We take the following two questions as examples: *Q1: "...Just ask me for news, politics, or a story from history. What would you like to do?"* and *Q2: " What's your zip code?"* Although they are both Wh questions, users will answer them in different ways. For Q1, users will extract answers directly from the question, which is not suitable for answering Q2. Thus, instead of using the traditional way to classify questions, we interact with 10,000 randomly selected skills using the extracted utterances and collect the replies as the *Basic Corpus of Replies*. Then we manually analyze 2,000 randomly selected replies (i.e., questions) from the corpus. We find that the questions can be divided into five types according to the ways to generate answers.

Yes/No questions. The question of this type is an interrogative construction, and expects answers like *"yes"* or *"no"*. A Yes/No question usually has an auxiliary verb in front of the subject, which is also called subject-auxiliary inversion (SAI). It has two subtypes: *Inverted question (IQ)*, and *Tag question (TQ)*. An example of IQ is *"Are you going?"*, in which subject and the first verb in the verb phrase will be inverted if the verb is a modal or an auxiliary verb or with the verb *be* and *have*. TQ is a short question at the end of a sentence, which is often made up of a modal or helping verb and a subject pronoun. An example is *"You're going, aren't you?"*. Note that there is an Inverted Alternative Question (IAQ) such as *"Are you staying or going?"*, which looks quite like IQ, but it actually does not require a simple yes or no for an answer. We should exclude it from this type.

In order to identify Yes/No questions, we use *constituent-based parsing* which is popularly used in natural language processing to analyze questions. A constituency-based parse tree can represent a context-free grammatical structure of sentences. Non-terminals in the tree are types of phrases (tagged by part of speech labels), and leaves are words in the sentence. We focus on the tag "SQ", which represents either a Yes/No question, or the main clause of a wh-question, following the wh-phrase with tag "SBARQ" (direct question introduced by a wh-word or a wh-phrase, e.g., *"How can I help you?"*) [8]. Examples are shown in Figure 3 (more examples are shown in Appendix C). Thus, to identify Yes/No question through using the constituency-based parse tree, we first locate the tag "SQ", and then filter out those questions if "SQ" follows a W-tag (representing wh-phrase or wh-word). We should also filter the IAQ by checking whether there is a "CC" tag (representing the word "or").

For *TQ* type, it is a statement followed by a mini-question which has the form of "auxiliary verb + subject". We judge this kind of structure in a parse tree and extract the auxiliary verb (`be`, `do`, `have`, or a modal verb like `will`) with a subject. Considering that decisive questions often appear at the end, we only judge the last sentence[3].

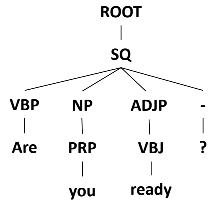


Figure 3: An examples of constituency-based parse tree

Instruction questions. The questions of this type give users direct guidance on how to answer them. They are essentially similar to imperative sentences which transfer the guiding or suggestive information to users. In order to guide users to reply with the correct answer, the skill usually tells the user what to say by using the directive keywords (e.g., "say", "ask") in the sentence. For example, *"Welcome to the Reddit Notifier skill... just Say: Help me"*.

After manually analyzing questions in *Basic Corpus of Replies*, we find that over 96% of the instruction questions use "ask" and "say". One main reason is the samples given by Amazon [9, 10] for developers to build skills, where "ask" and "say" are used. The two words are in line with user habits. To identify such a type, we first find *InstruTag* in the constituency-based parse tree including "VB" (Verb, base form), "VBG"

[3] Sometimes, we meet very short questions without an auxiliary verb or a modal verb. We still classify it as Yes/No questions. Such as "next news?".

(Verb, gerund or present participle), and "VBP" (Verb, non-3rd person singular present) and check whether there are some command words like "ask" and "say" (or their "-ing" form). In this way, we can determine whether the question is an Instruction question.

Selection questions. We refer to the questions containing multiple parallel answers as "Selection questions". Some former studies have a similar category, referring to the questions as "choice" [30] if the answers are connected by the keyword "or" (e.g., a question like *"...To get started, you can get a quote, listen to the daily briefing, or get an account summary."*). To identify such questions (referred to as Selection_CC), we try to find similar patterns in the constituency-based parse tree. The patterns should be with tag "CC" (Coordinating conjunction) that indicates the existence of Paratactic Structure in a sentence. We also include questions that need to be answered with serial marks into the selection question (such as *"1: high, 2: medium, 3: low. Choose one."*), which has three choices but no coordinating conjunction. To identify such questions (referred to as Selection_SC), we extract all numbers and single characters from the constituency-based parse tree, and then judge whether these numbers and letters are continuous.

Wh questions. Wh questions are also known as open questions [7]. Users are supposed to answer such questions in a free way, instead of obtaining the answers directly from questions or making some judgments. Their knowledge or understanding is usually needed in this process. An example is *"What is your name?"*. To identify questions of this type, we find those questions with *WH-tag* which include *"WDT", "WHADJP", "WHADVP", "WHNP", "WHPP", "WP", "WP$", "WP-S", "WRB"* in the constituency-based parse tree with *wh-words*. If it contains related tags, we classify the question as Wh question.

Mix questions. Sometimes the output from skills contains more than one of the previous four question types. For example, the output *"You can say repeat or stop."* is the combination of an Instruction question and a Selection question. We refer to it as the Mix question.

3.4 Answer Generation

After classifying the questions, we get 5 types of reply content: Yes/No, Instruction, Selection, Wh, and Mix. We can generate corresponding answers for different types of questions. For some types (i.e., Yes/No questions, Instruction questions, and Selection questions), we can directly extract answers from the question itself. For Wh questions, we generate a knowledge database to answer the question and explore the behaviors of skills. For Mix questions, we have strategies to answer it according to the question types it contains. We show some examples in Figure 4.

Yes/No questions. We simply generate the answers as "yes" or "no" to the questions.

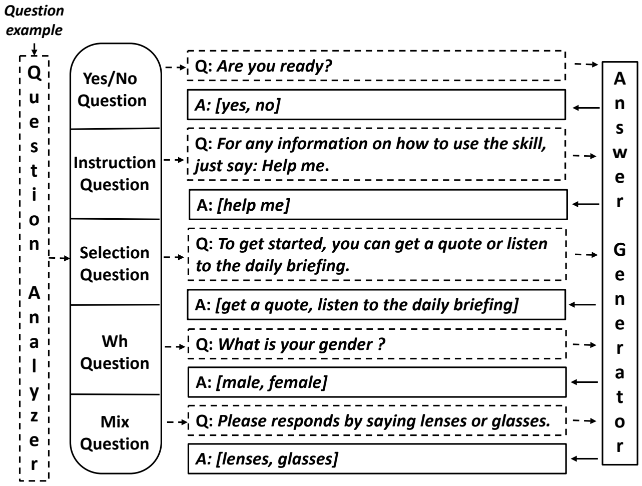

Question example
Question Analyzer
Yes/No Question
Instruction Question
Selection Question
Wh Question
Mix Question
Q: Are you ready?
A: [yes, no]
Q: For any information on how to use the skill, just say: Help me.
A: [help me]
Q: To get started, you can get a quote or listen to the daily briefing.
A: [get a quote, listen to the daily briefing]
Q: What is your gender ?
A: [male, female]
Q: Please responds by saying lenses or glasses.
A: [lenses, glasses]
Answer Generator

Figure 4: Q&A samples.

ASK	SAY
ask (sb.) Wh-Q	say Wh-Q
ask sth. like/... INS	say sth. like/... INS
ask (sb.) to INS	say INS (to do sth)
ask (sb.) (about/for) INS	say INS for sth
ask that INS	say (that) INS

Table 1: Rules to generate answers for Instruction questions

Instruction questions. Based on previous analysis, our analysis focuses on *ASK* and *SAY*[4]. We look into the Oxford Learners Dictionaries [11], and find that the two words *ASK* and *SAY* usually have five patterns, as shown in Table 1. For example, the skill can "ask (sb.) to INS". The components in the brackets (e.g., "sb." here) are not necessary for a sentence. *INS* is the instruction that we should extract. Sometimes, Wh questions are used as a component (e.g., ask (sb) Wh-Q). *Wh-Q* is the Wh question here. An example is *"You can say what is the current sibor rates"*.

To identify the five patterns in an instruction question, we first get the constituency-based parsing trees for the question. According to the five rules in Table 1, we can specify the matching rules based on them and use regular expressions to identify which pattern is used in the question. Then according to the patterns, we extract the *INS* or *Wh-Q* as the answers to the questions.

Selection questions. In this type, the expected answers are usually connected in parallel by conjunctions (e.g., "or", "and") (referred to as Selection_CC), or clearly marked by indicators such as the numbers or letters (referred to as Selection_SC). So users can directly speak out the answer itself or feed back the indicator. For example, the skill myTuner Radio says: *"Ok, Here's myTuner Radio. I've found: 1: CHOI-FM Radio X 98.1 from Canada, 2: Ibiza X Radio from the United Kingdom, 3: Radio X London from the United King-*

[4] To include more words in the future, we can quickly generate the rules for them using dictionaries [11].

dom. Choose a station.”. Users can answer the question by saying *“CHOI-FM Radio X 98.1 from Canada”* or directly say the number *“one”*. To automatically extract answers, for Selection_CC questions, SkillExplorer checks the constituent parsing tree to find parallel structures connected by *CC* (Coordinating conjunction) which may be corresponding to the words, phrases, and clauses. For Selection_SC questions, SkillExplorer checks the leaf nodes of the parallel structures in the constituent parsing tree to judge whether the serial indicators (i.e., numbers or letters) exist there and have the same format. If so, SkillExplorer will enumerate the found indicators to explore the behaviors.

Wh questions. As mentioned previously, users commonly answer Wh questions according to their knowledge or understanding, instead of obtaining answers directly from the questions. So our idea to answer Wh questions is to first build a knowledge database and then extract answers from the knowledge database. However, if the knowledge database is designed to include all kinds of knowledge, it will be too large to construct. Considering that our purpose is to detect whether a skill impacts users’ privacy or conflicts with Amazon’s development rules [6] (also related to users’ privacy), we design the knowledge database from the viewpoint of users’ privacy.

After analyzing Amazon’s development rules, we try to create some virtual users with different profiles for answering the privacy-related questions raised by skills. For other questions, although they are not our focus, we still try to answer them by constructing a noun database (common nouns in Wh questions) or feeding questions to online chatbots (e.g., Mitsuku [4] and Cleverbot [12]) and using their answers. For each virtual user, the private information of the user should be created to build the profile. Such information includes the full name, first name, gender, birthday, etc. Some fields are shown in Table 2, and more details are shown in Appendix D. Note that such information cannot be randomly generated. Otherwise, the skill may identify the inconsistency or some illogical problems (e.g., an 8-year boy may not like to have a driver’s license number), which will impact behavior exploration. So the profiles should be created to be close to real situations. Since some questions from skills may be related to the relationship between the users (e.g., ask a child the name of his mother), the knowledge database should also consider such a situation.

Info	Value
Full Name	James C Washington
Gender	male
Date of Birth	6/19/1980
Social Security Number	066-80-6240
City	Buffalo
State Full	New York
Zip Code	14214
Phone Number	716-780-4085

Table 2: An example of the virtual user

We first build several virtual users (VUs) according to decades of age since skills may react differently to different ages. For users of the same age, we also create two VUs: male and female. Then we continue to give them other private information. To make such information representative, we have searched on the Internet for other fields. Note that, some fields in the table have logic connections (e.g., address and zip code). So we search them together to find the logic connected data after the logic connections are manually marked. Also, some fields (e.g., phone number, credit card number) have a specific format. A randomly generated phone number could not be accepted by skills. Therefore, we use an online information generator which can generate such fake yet correct-format information [13]. At last, after collecting all the information, we manually check whether there are some inconsistencies. Then we add the social connection between them including husband and wife, parent and children, etc.

Then by searching keywords in the knowledge database, SkillExplorer will return the answers from a random profile (for the first question) and use other fields in the profile for answering other future questions. Note that for the same question, to explore its different behaviors, SkillExplorer will use different profiles to answer and observe the behaviors of the skills. If a skill reacts differently, SkillExplorer will continue to explore the behaviors. Otherwise, it will stop using more profiles to respond to this question. For example, the skill *“Preventive Health Care Services”* will have different behaviors according to the age of 13 and 18. False negatives may happen since SkillExplorer cannot enumerate all possible profiles. One possible solution is to extend the possible values (e.g., different addresses) for each field in the profile, which at the same time brings extra time spent on communicating with skills. For some behaviors that are really hard to trigger, little impact will they bring on users.

Mix questions. Mix questions include more than one type of question in the output of skills, which are also very common since developers can organize the outputs from skills in any way. To deal with such questions, a simple idea is to mark the types, generate answers according to each type, and feed back all the answers together. However, it may waste lots of unnecessary time due to that many answers are unaccepted to the skills. For example, in the question “You can ask something, such as what’s your name.”, the user’s name (extracted as the answer to the Wh question) is not the answer to the Mix question. So we need to select the question types to answer from all the sentences in Mix questions.

We should select the questions to answer according to their types. First, we hope to generate the rules from studies on linguistics. However, we do not find any useful rules. So we have to generate the rules by ourselves. Considering that Mix questions are designed for users to answer, the question should be understood by the majority of users. So we authors play the role of users to understand the questions and try to generate the rules. We randomly sample 2,000 Mix questions from the

Rule	Situation	Type
R1	∃ Y	Y
R2	∃ S_SC & ∃ I	S_SC&I
R3	(I&S_CC) in Q^*	I&SC_CC
R4	∃ I	I
R5	∃ S	S

Table 3: Rules to generate answers for Mix questions

Basic Corpus of Replies and manually answer them. From the answers, we summarize the rules as shown in Table 3, and evaluate the accuracy. We randomly select the other 200 Mix questions, utilize the rules to generate answers, and compare them to human answers. The accuracy is 91%. The rest 9% (misunderstandings) are mainly due to grammar errors or parsing errors from the NLP tool. In the table, *Y* means Yes/No type, *I* means Instruction type and *S* means Selection type. *S_SC* and *S_CC* are Selection_SC and Selection_CC, respectively. According to R1, if there is a Yes/No type in Mix questions, we just answer "yes" or "no". According to R2, if Selection_SC type and Instruction type exist at the same time, both types need to be processed. According to R3, if Selection_CC and Instruction are included in the same sentence, they should be replied together. For example, *"say next or previous"*. R4 means that if there is an Instruction type, SkillExplorer will just answer this type. For example, the question "*You can say what is the weather like today"* contains the Instruction question and Wh question. It should be marked as an Instruction question. Based on our evaluation, the rules are accurate to extract answers from Mix questions.

3.5 Behavior Exploration

By leveraging the previous approaches, SkillExplorer can explore one execution path of the skill. To explore all its behaviors, SkillExplorer should further record the branches and explore those un-executed ones. We also introduce an approach to speed up the interaction.

Record and traverse branches. For a given question, there could be multiple answers. For example, there are two answers for Yes/No questions and two or more answers for selection questions. For an answer, a further output will be responded from the skill, which serves as a new question expecting further answers or simply ending the conversation. Continuing this process will form a tree-like structure to record questions and answers. So we design an *interactive tree* (*i-tree* for short) to represent the status of exploration. Each node in the i-tree represents a single interaction (include an input and corresponding output). While SkillExplorer communicates with the skill, an i-tree is drawn simultaneously. The node will be marked as *visited* if it is explored. If an execution path in the i-tree is explored to the end, and there are unvisited nodes, SkillExplorer will re-start from the beginning to the unvisited nodes. Note that there could be more than one roots in the i-tree, due to several utterances extracted. SkillExplorer will end the execution path in the i-tree if one of the following situations happens. (1) No answer can be generated for a given output (e.g., *"That's our information, bye."*). (2) An exception happens (e.g., the operation needs to be performed on the mobile phone[5]). (3) For the same node in different executions, the questions are different. For example, a skill can generate different quizzes. There is no need to enumerate all the quizzes.

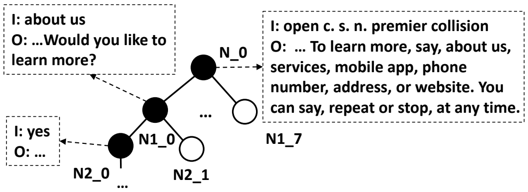

I: about us
O: ...Would you like to learn more?
N_0
I: open c. s. n. premier collision
O: ... To learn more, say, about us, services, mobile app, phone number, address, or website. You can say, repeat or stop, at any time.
...
I: yes
O: ...
N1_0
N1_7
N2_0
...
N2_1

Figure 5: An example of itree

Figure 5 gives an example when SkillExplorer analyzes a skill. When the first utterance *"open c. s. n. premier collision"* in N_0 is sent to the skill, the returned output is *"...To learn more, say, about us, services, mobile app, phone number, address, or website. You can say, repeat or stop, at any time"*. SkillExplorer parses the output as a question and generates eight answers: "about us", "services", "mobile app", "phone number", "address", "website", "repeat" and "stop" (in the nodes from N1_0 to N1_7, respectively). The first answer is fed to the skill to continue the process of exploration. When the last question is reached, the process of exploration will end. At this time, SkillExplorer check whether there are any unvisited nodes (white node). If so, SkillExplorer will find a path to the node in the i-tree and restart from the root. This process will continue until there is no unvisited node left.

Speed up the interaction. In the process of exploration, the real execution could be very time-consuming due to the following reasons. Firstly, some skills raise questions which have already been asked. If the question is parsed again for further exploration, the i-tree may not end. In other words, in an i-tree, if the output in a new node (i.e., a leaf node) is the same as a visited one (which may or may not be in the same execution path), the node should not be explored again.

Secondly, when SkillExplorer restarts from the root of an i-tree, some paths in the i-tree are repeatedly executed, which further makes the speaker to read the outputs many times. When the outputs are long, it will be very time-consuming, especially when the output is at the beginning part of the i-tree. For example, the output in N_0 with 39 words will take 18 seconds to read. When the skill is explored, it will be executed at least 8 times. More than 2 minutes will be spent on the node. To solve this problem, SkillExplorer does not need to wait until the reading of the whole output is finished. For a node in the i-tree representing the output from a skill, if this node is

[5] A special exception is the network error. When this situation happens, SkillExplorer exits the current execution and restarts from the root node.

visited, SkillExplorer can directly utilize the generated inputs in the last execution to answer the output. For the previous example in the second execution, SkillExplorer will directly feed the input stored in N1_0 to the skill before the reading of the whole output is finished.

3.6 Implementation

We build a crawl to collect skills from Amazon and Google markets and build SkillExplorer to explore the behaviors of these skills. The project includes more than 7,000 lines of Python code.

In the process of analyzing questions and generating answers, SkillExplorer builds the parse tree using NLTK (natural language toolkit) [28] and Stanford NLP Parser [29][6]. Both of the two tools are popularly used in the field of natural language processing. To build the interactive system, one possible approach is to feed the utterances and the answers directly to the smart speakers (e.g., Amazon Echo), then record the outputs and transform them into texts by using speech recognition tools (e.g., Google TTS). However, this approach is too time-consuming[7]. Instead, our idea is to use the simulators provided by markets, which are often used by developers to test their skills. Both Amazon and Google have their own simulators. In particular, the simulator allows developers to communicate with skills using texts. That is to say, developers can directly feed text inputs to a skill and observe its outputs also in texts, which does not need any tools to translate a voice question to the texts, saving the exploring time. Regarding the chat robots for answering Wh questions, after trying some famous chatbots, we choose to use Mitsuku [4] and Cleverbot [12] due to their better performance. In the process of acquiring outputs from skills, the outputs will be returned one by one. The time interval equals the time to read the previous output. So if the first output is long, we should wait for long for the second output. Here, we set up a timeout (10 minutes) for waiting. If the timeout is reached, SkillExplorer will stop the current exploration and start a new path in the i-tree.

3.7 Evaluation

Coverage. SkillExplorer is designed to traverse the behaviors of skills. So the coverage of behaviors is important to evaluate the effectiveness of SkillExplorer. The ideal way to evaluate behavior coverage is to analyze the source codes and compare them with the behaviors explored by SkillExplorer. However, it is very hard to find open-source projects of skills from the Internet. An alternative way is to manually communicate with the skills, and try to collect as many behaviors as possible. Such collected behaviors can be used as the ground truth for comparison with the behaviors explored by SkillExplorer. For simplicity, each node in the i-tree can be viewed as a behavior. So we can compare the i-trees generated by human and SkillExplorer, and calculate the coverage c by $c = |n_h \cap n_s|/|n_h|$. $|n_h|$ indicates the number of nodes in the i-tree explored by humans. $|n_h \cap n_s|$ shows the number of nodes in both the i-tree explored by humans and the i-tree explored by SkillExplorer.

Yes/No	Instruction	Selection	Wh	Mix
0%	8%	8%	5%	9%

Table 4: The rate of incorrect answers

In our evaluation, we randomly sample 50 skills from the 21 categories. Then we manually and extensively communicate with them, trying to collect as many behaviors as possible, which lasts for about 8 hours. 226 outputs from skills are collected, including 28 Yes/No questions, 16 Instruction questions, 13 Selection questions, 17 Wh questions, and 53 Mix questions. Further, we let SkillExplorer communicate with the skills, and collect 203 different outputs. So the coverage is 90% (=203 / 226). We further look into the 23 outputs that are not covered by SkillExplorer and try to figure out the reasons for missing. One reason is due to the problem of NLP tools. 5 questions cannot be correctly parsed by the tools (e.g., wrongly marked part of speech). Also due to the carelessness of developers, some questions contain grammar errors which cannot be correctly parsed. We also find 11 questions require human expertise (e.g., “What SGLs do you want to look up”) or use complex structures (e.g., “Okay, player one tell me a name, by saying player one is, followed by the name”), which are quite difficult to answer even for human users. More examples are shown in Appendix E.

Accuracy of answer generation. Regarding the accuracy, we care about missed answers and incorrect answers. Missed answers impact the coverage, which has already been evaluated previously. So we evaluate incorrect answers here. Incorrect answers cannot be accepted by skills, which may let SkillExplorer waste time on unnecessary execution. We randomly select 200 questions from each of the five categories classified by SkillExplorer. In sum, 1,000 questions are analyzed manually. We compare the two sets of answers and give the error answer results in Table 4. On average, 6% of the answers are wrong. Yes/no Question has the lowest ratio (0%), while Mix Question is higher (9%). Note that the incorrect answers impact neither the results of coverage nor the results of the measurement. They only impose unnecessary analysis time on the exploration of skills. The reasons for incorrect answers are similar to those mentioned previously.

Performance. SkillExplorer has analyzed 28,904 skills within 5,270 hours[8] (using a machine with a 3.6GHz CPU,

[6] We download NLTK v3.4.5 from [14] and Stanford NLP Parser v3.9.2 from [15].

[7] It will also exceed the time limit given to the user for feedback. Usually, the time limit is 6 seconds [34]. If the time of waiting for the user’s response is too long, the smart speaker will automatically turn off itself.

[8] We registered 25 different Amazon developer accounts, and 2 Google developer accounts for testing. 27 simulators were utilized (25 from Amazon and 2 from Google).

16GB memory, 1TB hard driver, and the Windows 10 operating system). Each skill costs about 627 seconds on average, including the utterance question generation, question understanding, answer generation, and behavior exploration. The median time of skill exploration is 428.5 seconds, ranging from 36 seconds to 8,969 seconds. For different categories, the time varies. It depends on the function and the branches of the skill. A game skill always spends much more time than a weather forecast skill because the game skill has more branches for users to choose. By the way, the stability of network connection matters as well. We also analyze the time of Google actions, which is much smaller than Alexa's, because the test console of Google Assistant does not support all the actions well and its robustness is not so good as Amazons's, making many actions unable to respond as they do in reality. If we use the real smart speaker for evaluation, the time should be much more. We also evaluate how much time could be saved by our speedup mechanism (see Section 3.5). If this mechanism is not used, the average time for each skill will be 885 seconds, which means that 29.2% (=258/885) of the time could be saved by this mechanism.

4 Measurement

4.1 Landscape

Skills & Authors. We crawl 68,066 skills from the Amazon market[9], and 10,899 actions from the Google market. Skills in Amazon have 21 different categories and the category *Games & Trivia* has the largest number of skills (as shown in Appendix F). Among these, 19.4% of them do not have invocation names, which means that developers use the *pre-built model* to build the skills. In other words, the developers cannot design their own questions, but use pre-designed questions by the markets, which should not contain any malicious questions. Thus, we do not measure these skills. Among the rest 54,865 skills, we randomly sample 30,000 for the measurement. However, some skills cannot be invoked due to that Alexa only wakes up the more popular one or the previously waked one if two skills have similar invocation names. So in the end, 28,904 skills are measured. We also record the developer names for the skills. In sum, 12,376 different developer names are recorded (a developer can register for different accounts with different names). On average, one developer's name is in charge of 5.5 skills. Interestingly, the developer *InfoByVoice* owns the most skills (i.e., 2,577 skills). All of them are in the category *Lifestyle*. We check the interactive content with them and find that these skills provide organizations' information. The developer *Rhall* owns 1,401 skills, and most of them aim to explain some facts (e.g., a skill "California Facts" gives facts about California).

Structure of i-trees. We make a statistical analysis of i-trees. We measure the number of branches in i-trees, depth of i-trees, and the number of answers to a question. Figure 6 (a) shows the distribution of the number of branches in i-trees. From the figure, we can see that 90% of the skills have less than 15 branches. The average number of branches is 7.9. Some skills have more than 50 branches (most of them are games or Selection_SC questions with multiple choices), which are not user-friendly to answer. Figure 6 (b) shows the distribution of the depth of skills. The average depth is 3.6. From the figure, the depth of 68% skills is less than 4, and the depth of 95% skills is less than 10. It indicates that most skills do not interact with users with deep conversation. We find some skills are with the depth of 40. They are story-tellers. We also look into questions related to privacy. They are usually Wh questions, with the depth less than 5. Skills can customize their services from the requested information (e.g., assessing the value of a house in a location). Figure 6 (c) shows the number of answers to a question. The average number of answers is 2.9, which means that most questions only have about three choices for users to answer. If there are too many answers in a question, users may not be able to remember them to answer. An interesting skill is *Encyclopedia of dinosaurs*. It contains a question with 41 answers.

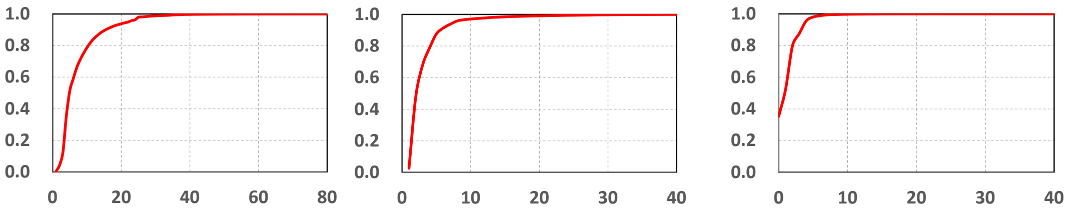
Popular questions and popular words. After analyzing more than 160,000 questions in our measurement, we list the top 5 questions in Appendix G. These questions are mainly from the developers *InfoByVoice* and *SkillSet*. For example, the question "say, service times, location, phone number, or goodbye" is mentioned by 1,045 skills developed by *InfoByVoice*, and mentioned by 264 skills developed by *SkillSet*. We also check the description of the skills on the website of the two developers. Both of them mention *VoiceApps.com*. Maybe the two developers have some connections. We also count a question for only once if it appears in different skills by the same developer. The most 10 popular words (we only count nouns in the constituency-based parsing tree) are "skill","alexa", "number", "fact", "help", "information", "name", "phone", "location" and "service".

Invocation names. Different from previous studies [26, 35] on invocation names which mainly focus on the security problems of abnormal diversion (e.g., skill squatting, voice squatting, voice masquerading), our study checks whether the invocation names can meet Amazon's requirements. As we know, Amazon has strict requirements for designing invocation names [5]. Some sample rules are given in Appendix H.

We check whether all 57,139 skills having invocation names[10] are against the rules. We find that 9,799 skills do not meet the requirements. Among them, 120 skills do not follow the rule (2): two-word names with article words (e.g., the, a,

[9] We crawled the skills from the America market from October 25, 2019 to November 12, 2019, where the number of skills is the largest in the world. Different countries may access different numbers of skills according to the policy of Amazon.

[10] Some skills may not have invocation name which can be invoked through implicit invocation.



(a) number of branches (b) depth of i-tree (c) number of answers

Figure 6: Distribution of i-tree

Info Type	Keywords
basic info	full name, home address, email address, date of birth, telephone number, etc.

Table 5: The words related to privacy

an). 377 invocation names are person or place names (e.g., bainbridge island), which violate the rule (3). 179 skills fail to comply with the rule (4): using launch words (e.g., "open," "tell," "load,", etc.) or connecting words (e.g., "to," "from," "in,"). Two invocation names contain "app" or "skill". We also find that 2591 invocation names are used by 9,128 skills. The most commonly used invocation name is *how many days*, which is used by 153 skills.

4.2 Skills Conflicting with the Developer Specifications

As we mentioned in Section 2, according to the rules of some markets (e.g., Amazon), for some kinds of personal information, developers are allowed to obtain them for better user experience. Such information (shown in Table 5) includes a user's name, email address, phone number, etc. which should be obtained by using specific APIs (e.g., Alexa customer profile API) to configure permissions, and should be claimed in the privacy policy of the skills [6]. For the permissions, they can be seen on skills' introduction pages. For the privacy policy, the developers should clearly include what kinds of personal information are collected, how and why to collect the information. However, we find some developers request such information but do not claim in the privacy policy or configure the corresponding permissions. Instead, they directly ask users for private information. To detect the illegal collection of information, we analyze them in the interactive content.

Note that we cannot directly compare the privacy words in the Table 5 with the contents from skills. For example, a skill may say *"Our phone number is xxx-xxx-xxxx"*. The skill does not request such information from users. Instead, it just gives information about the skill. To distinguish the two situations, we leverage the dependency-based parse tree where all the nodes are words. The links among words are labeled by the syntactic function grammar tree. Particularly, we use a two-step comparison. (1) We first check whether the words are used by skills with the correct part of speech. Usually, the words are used as nouns. Sometimes, a skill may use a different part of speech of the word. For example, in the question *"I can address the problem"*, the word "address" is used as a verb. To filter out such situations, we check the part of speech of the privacy word and only identify those used as nouns. (2) Then we check the ownership of the privacy words. We also leverage the dependency-based parse tree, which shows the relationship of dependency between words. For example, in Figure 7, the word *name* belongs to *your*, whose connection can be extracted by the tags. In the figure, `nmod` is used for nominal modifiers of nouns or clausal predicates, while `poss` means possession modifier. Their combination `nmod:poss` is used for a nominal modifier. However, a counterexample is "Our phone number is xxx-xxx-xxxx", where the keyword phone number belongs to "our" (i.e., the owners of the skill). We only check the privacy words belonging to the users (e.g., using the word "your"). As developers may not directly request the privacy keywords to evade the vetting process of markets, besides checking the keywords themselves, we should also check their synonyms. So we expend the privacy keywords using their synonyms.

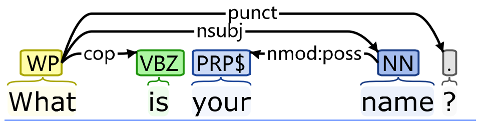


Figure 7: An example of dependency-based parse structure

After obtaining the privacy keywords that a skill requests from users, we further determine whether the skill conflicts with the development specifications. We first check if these keywords (including their synonyms) are declared in the skill's privacy policy. If not, the skill conflicts with the developer specifications. Otherwise, we should further check whether the privacy keywords are clearly declared for requesting users' information. If no such declaration is found, the skill will be viewed as conflicting with the developer specifications. However, in real situations, it is hard to check whether the privacy keywords are used for requesting users' data. An example is *"We collect users' private data including their* *name and email"*, where the general term "personal data" and the pronoun make the analysis difficult. Fortunately, PolicyLint [18] handles such situations. So we leverage PolicyLint to solve this problem. Specifically, what makes the situation complex is the *general* declaration which usually contains three types of words to collect users' information, including a verb of collect information (e.g., collect, gather, check), a general term (e.g., personal information, personal data), and subsumptive relationships words (e.g., such as, include). Note that, due to the limited number of keywords used by PolicyLint, it may not be enough to characterize all the possible general declarations, especially for the diverse privacy policies given by various developers, and further causes false positives. Thus, in our implementation, if any two of the three types of words are in a declaration, we will view it as a general declaration. Such an approach is very effective to find the declaration requesting users' private data, which is then compared with the contents in the skills to ensure whether the skills conflict with privacy policies.

Results. We first validate the accuracy of our approach. After analyzing 30,801 skills (28,904 from Amazon and 1,897 from Google), SkillExploer finds 1,156 skills conflicting with the developer specifications. Among these skills, 632 skills neither state privacy keywords in privacy policies nor configure corresponding permissions. 183 skills just conflict with the claimed privacy policy, and 341 skills just do not configure corresponding permissions. We manually check the results of the comparison between the keywords and the privacy policies, and only find 15 false positives which do not conflict with privacy policies, mainly due to the following two reasons. Firstly, the NLP tool (i.e., Stanford NLP Parser) cannot correctly parse a sentence, for example, the tool will label "username" as an adjective in the sentence *"to use our voice experiences users may provide us with their data such as email, username and password to your service"*. Secondly, some declarations that explicitly state to collect users' information are not correctly caught by PolicyLint. For example, *"you may be asked to enter your zip code or other details to help you with your experience."*.

After removing false positives, we find 1,141 skills that conflict with the developer specifications. We analyze the keywords of these privacy contents. The most frequently requested information is as follows: address, name, phone number, zip code , and email. Most of them are in the categories of *Lifestyle* and *Education & Reference*. An interesting case is the skill *"WifiPassword"*. It requests users' wifi name and password and also asks them to finish the request through a webpage popping up on users' smartphones when such intent is activated. Note that the skill never mentions this suspicious request in its privacy policy list. We also check its reviews on the market. Some users mentioned that *"... after filling out the form it gave me someone else's network name and password. What's much worse is that that the name of the wifi network makes me believe that it's very likely someone near to my location, due to the name being a local reference..."*, *"Do not download this app. ... It stores your info and password."*. It seems the suspicious behaviors have already troubled users. Another skill *Scare Text* requests users' phone numbers. According to its description, this skill will send a randomly selected scary image via texting message to the given number. However, after we test several phone numbers, the skill never sends the message.

4.3 Skills Conflicting with "Stop"

After finishing using a skill, users will stop the skill. Otherwise, the skill will continue listening to the users' private conversations. However, some malicious skills may not stop even if they receive users' commands to stop. So we want to check the existence of skills with such behaviors in the wild markets. According to the survey [35], 91% of Alexa users use the command "stop" to terminate a skill, and 36% of users choose "cancel", and only 14% of them use "exit". So we send the command "stop" to the skills. Then we leverage some built-in functions of the virtual personal assistant (VPA) and check whether the VPA is activated. For example, we can ask the time using Alexa's own function "what time". If the response is the current time, we can verify that the skill has already exited. Otherwise, it is still on. Although this approach may be circumvented by hijacking the built-in functions, we can try other different functions to test for better accuracy. In our interaction experiments, we also find that a small number of skills behave differently in simulators and the real devices when receiving the commands to exit. We are not sure about the concrete reason. Thus, after some potentially harmful skills are automatically detected, we need to check them on real devices. Note that the different behaviors only happen when receiving the command "stop" to exit. For other voice commands, they behave consistently in both environments[11].

Results. We evaluate 28,904 Amazon skills, and find 802 skills do not really stop after receiving the stop command on the simulator. Then we use Echo for further checking, and find that 68 skills have problems on the real smart speakers. In this process, we only need to open a skill and stop it, which takes about 15 seconds to finish (about 3 hours in total). Then we carefully analyze 68 suspicious skills. They achieve eavesdropping using one of the following three ways. (1) 32 skills change the default "stop" commands to others which users may not know. For example, the skill *Millennial Money*

[11] One may be worried about that some malicious developers can find the differences between simulators and real environments. However, to the best of our knowledge, no open materials mention whether simulators are used in the vetting. In the current stage, it seems there is no motivation for developers to distinguish the two environments. Although we observed that the simulator behaves differently when receiving the "stop" command. However, the command can be replaced by "exit", which will not let the simulator behave differently. Also, SkillExplorer can evaluate the "stop" command after vetting all other behaviors. If a skill behaves differently later (after identifying the simulator), it is highly suspicious.

changes the default command to *"I've done"*. (2) 29 skills ignore the stop command after correctly receiving it (we verify this from the communication history supported by Amazon). For example, the skill *My birthday month* always says that it cannot get the stop command (which actually indicates that it has received it) and continues its other functions. (3) 7 skills seem more strange. For example, one skill named *Malignant Tweets* always returns "cannot find this skill" to mimic Alexa no matter what commands it receives and continues to listen to users' conversation. Another skill named *math-training* replies "OK" after receiving the stop command. But it will continue to listen to users' conversation for 6 seconds. Regarding actions from Google, since Google has very strict requirements on the exit operation, it limits the developer's final response to a simple reply within 60 characters and must be the last dialogue in this interaction [16]. We did not find any action that has such a problem.

4.4 Skills Conflicting with Their Descriptions

We further want to check whether the information requested by a skill is corresponding to its description given by developers. However, this is very challenging due to diverse ways to describe the skills and the different functionalities given by the skills. So we do a preliminary study. Considering that skills with similar functionalities should behave in a similar way, we use some skills as the seeds and compare other skills with them. For example, two skills A and B both provide real estate information. It is normal that both of them request users' addresses. However, it would be very strange if a skill requests for the health status of users.

To achieve the differential analysis, we manually select 100 typical skills in 10 categories which request for various kinds of privacy information and view them as the seeds. Then we extract the keywords of the descriptions (i.e., nouns in the constituency-based parsing tree) from all the collected skills. In this way, we could find the skills with similar functionalities. Then we compare the behaviors of the skills, especially the privacy information they request. In this way, we can find skills with abnormal behaviors.

Results. After manual verification to filter out some reasonable cases, only a few abnormal skills are left. In this preliminary measurement, less than 10 skills request personal information that does not match their descriptions. For example, a skill named *Ehrlich Pest Control* is supposed to tell users about how to prevent common household pests (e.g., mice) according to the description. However, if a user asks some questions that cannot be understood by the skill, it will ask the user for her phone number and area code. We also find there is a low rating for the skill on the market. A user mentioned that *"...it complied then asked me for my phone number so not going to happen, fix that again asking me for my phone number wrong move."*

5 Discussion

5.1 Defense Suggestions

Although SkillExplorer could serve as a supplement approach for the market administrators to vet skills, we still have some suggestions for them. Firstly, skills should be strictly reviewed before being put on the shelf, especially the contents related to privacy contents. Considering some technical challenges (e.g., the ownership of the privacy-related words) may impede the detection, NLP should be included in the automatic analysis (see Section 3 and Section 4). Secondly, besides the contents provided by skills, the privacy policy links of skills also need to be strictly checked. In this way, users can have a general understanding of what kinds of personal information that the skills will request before users use them. Currently, the markets do not request the privacy policy to be in a unified form. So developers can prepare the privacy policy in various forms (e.g., on a web page, a PDF file, or even missing), which makes the vetting process quite difficult. An official template could be provided to the developers to follow. Thirdly, the built-in intents should also be strictly checked if skills are using them. For example, the built-in stop intent should be carefully checked which might allow a skill to continue working even after receiving the stop command.

5.2 Limitations and Future Work

Firstly, the accuracy of SkillExplorer can be further increased. Current problems are mainly due to developers' irregular design of the questions. Sometimes, developers want to make their questions be clearly spoken out by smart speakers. So they usually add some marks or punctuation insides the questions. For example, there is a question *"You can say News -or- Story"*. Developers add marks before reminding users of the words they need to say to highlight the key points when pronouncing. Although it does not impact user experience (or maybe make the user experience better), this will greatly impact the analysis since such combinations of words and punctuation seldom appear in real texts. Currently, NLP tools (e.g., Stanford NLP Parser) cannot handle this situation. Although in our study we have some techniques and special rules (e.g., removing the punctuation except the quotation mark immediately after the instruction words "say" and "ask"), our tool can be further improved. Also, current NLP tools have false positives (e.g., the extraction of the juxtaposition relationship is wrong, resulting in problems in generating answers).

Another limitation is from the simulator. Currently, it has restrictions on the interaction with mobile phones and the transmission of geographical location. Neither can it play the non-text audio. If a privacy-related question is played by audio, the simulator cannot correctly identify and return the texts in it. We will identify such a situation and further solve this problem.

6 Related Work

Attacks on skills. Recent studies have been carried out to understand the invocation mechanisms of skills. KUMAR et al. discover skill squatting [26], a kind of homo-phonic attacks to divert users' request to an undesired skill. Zhang et al. [35] further find voice squatting and voice masquerading, which allows a similar pronounced skill to hijack the existing legitimate skills. They also perform a large-scale analysis on skills with similar names or pronunciations in the Amazon market and Google market. Recently in October 2019, researches from SR Labs [3] demonstrate how a malicious skill can eavesdrop users' privacy after receiving the command to stop based on the research of [26], which is also found by us simultaneously. Our work differs from theirs. They design a skill to implement such attacks, while we perform a large-scale analysis on skills and find 68 skills in markets having such problems. Zhang et al. [36] find the vulnerability of NLU's Intent Classifier and leverage it to let the classifier misunderstand users' request and route the request to a malicious skill. These studies mainly focus on the invocation mechanism of skills, while our work is to explore the behaviors of skills and analyze the contents of the conversation.

Attacks on smart speakers. Researches [17, 24, 27] find that the mechanism of what they imagine is very different from what the smart speakers actually do. However, some studies [22, 23, 32] have already analyzed the security and privacy of general IoT devices, including smart speakers. Carlini et al. [20] perform Hidden Voice attacks on Amazon Echo, which proves the feasibility of audio attack from two aspects of black box and white box. It's found that both attacks can successfully occur on physical devices. Based on this, the authors put forward some ideas of defense. DolphinAttack [34] can modulate voice commands on ultrasonic carriers such as frequencies greater than 20 kHz so that people cannot hear them, while still attacking smart speakers. Yuan et al. [21, 33] integrate the voice commands into a song and let the commands be correctly identified by an audio speech recognition (ASR) system but not perceptual to human. Bispham et al. [19] try to hack the ASR system of smart speakers by gaining covert access to them with nonsense or missense sounds. Sugawara et al. [31] leverage the laser to remotely inject inaudible and invisible commands into voice assistants, taking advantages of the vulnerability of MEMS microphones. These studies mainly focus on how to inject commands into smart speakers or related ASR systems without being captured by human users, which are different from our study on skill behaviors.

7 Conclusion

In this paper, we propose the first systematic study on the behaviors of skills. The key techniques enabling the exploration are a suite of grammar-based methods including utterance extraction, question understanding and answer generation. We develop a tool called SkillExplorer to automatically communicate with 28,904 skills from the Amazon market and 1,897 actions from the Google market, a scale that has never been achieved before. Based on our measurement, over 1,000 skills request users to provide personal information without following developer specifications; 68 skills continue to eavesdrop users' conversation even after receiving the command to stop.

Acknowledgments

The authors would like to thank anonymous reviewers for their insightful comments that have helped improve this paper substantially. Specifically, we thank our shepherd, Professor Adam Bates, for his constructive feedback on this paper. The authors are supported in part by Beijing Natural Science Foundation (No.JQ18011), NSFC U1836211, National Top-notch Youth Talents Program of China, Youth Innovation Promotion Association CAS, Beijing Nova Program, National Frontier Science and Technology Innovation Project (No. YJKYYQ20170070), and Beijing Academy of Artificial Intelligence (BAAI).

References

[1] https://voicebot.ai/2019/10/01/amazon-alexa-has-100k-skills-but-momentum-slows-globally-here-is-the-breakdown-by-country/.

[2] https://voicebot.ai/2020/01/19/google-assistant-actions-grew-quickly-in-several-languages-in-2019-match-alexa-growth-in-english/.

[3] https://srlabs.de/bites/smart-spies/.

[4] https://www.pandorabots.com/mitsuku/.

[5] https://developer.amazon.com/docs/custom-skills/certification-requirements-for-custom-skills.html/.

[6] https://developer.amazon.com/zh/docs/custom-skills/request-customer-contact-information-for-use-in-your-skill.html.

[7] https://en.wikipedia.org/wiki/Yes-no_question.

[8] http://www.surdeanu.info/mihai/teaching/ista555-fall13/readings/PennTreebankConstituents.html.

[9] https://developer.amazon.com/en-US/docs/alexa/alexa-skills-kit-sdk-for-nodejs/develop-your-first-skill.html.

[10] https://developer.amazon.com/en-US/docs/alexa/custom-skills/manage-skill-session-and-session-attributes.html.

[11] https://www.oxfordlearnersdictionaries.com/definition/american_english.

[12] https://www.cleverbot.com/.

[13] https://www.fakenamegenerator.com.

[14] http://www.nltk.org.

[15] https://nlp.stanford.edu/.

[16] https://developers.google.com/assistant/conversational/conversation-exits.

[17] Noura Abdi, Kopo M. Ramokapane, and Jose M. Such. More than smart speakers: Security and privacy perceptions of smart home personal assistants. In Heather Richter Lipford, editor, *Fifteenth Symposium on Usable Privacy and Security, SOUPS 2019, Santa Clara, CA, USA, August 11-13, 2019*. USENIX Association, 2019.

[18] Benjamin Andow, Samin Yaseer Mahmud, Wenyu Wang, Justin Whitaker, William Enck, Bradley Reaves, Kapil Singh, and Tao Xie. Policylint: Investigating internal privacy policy contradictions on google play. In Nadia Heninger and Patrick Traynor, editors, *28th USENIX Security Symposium, USENIX Security 2019, Santa Clara, CA, USA, August 14-16, 2019*, pages 585–602. USENIX Association, 2019.

[19] Mary K. Bispham, Ioannis Agrafiotis, and Michael Goldsmith. Nonsense attacks on google assistant and missense attacks on amazon alexa. In Paolo Mori, Steven Furnell, and Olivier Camp, editors, *Proceedings of the 5th International Conference on Information Systems Security and Privacy, ICISSP 2019, Prague, Czech Republic, February 23-25, 2019*, pages 75–87. SciTePress, 2019.

[20] Nicholas Carlini, Pratyush Mishra, Tavish Vaidya, Yuankai Zhang, Micah Sherr, Clay Shields, David A. Wagner, and Wenchao Zhou. Hidden voice commands. In Thorsten Holz and Stefan Savage, editors, *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016*, pages 513–530. USENIX Association, 2016.

[21] Yuxuan Chen, Xuejing Yuan, Jiangshan Zhang, Yue Zhao, Shengzhi Zhang, Kai Chen, and XiaoFeng Wang. Devil's whisper: A general approach for physical adversarial attacks against commercial black-box speech recognition devices. In *29th USENIX Security Symposium (USENIX Security 20)*, 2020.

[22] Tamara Denning, Tadayoshi Kohno, and Henry M. Levy. Computer security and the modern home. *Commun. ACM*, 56(1):94–103, 2013.

[23] Earlence Fernandes, Jaeyeon Jung, and Atul Prakash. Security analysis of emerging smart home applications. In *IEEE Symposium on Security and Privacy, SP 2016, San Jose, CA, USA, May 22-26, 2016*, pages 636–654. IEEE Computer Society, 2016.

[24] Nathaniel Fruchter and Ilaria Liccardi. Consumer attitudes towards privacy and security in home assistants. In Regan L. Mandryk, Mark Hancock, Mark Perry, and Anna L. Cox, editors, *Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems, CHI 2018, Montreal, QC, Canada, April 21-26, 2018*. ACM, 2018.

[25] Jianfeng Gao, Michel Galley, and Lihong Li. Neural approaches to conversational AI. In Yoav Artzi and Jacob Eisenstein, editors, *Proceedings of ACL 2018, Melbourne, Australia, July 15-20, 2018, Tutorial Abstracts*, pages 2–7. Association for Computational Linguistics, 2018.

[26] Deepak Kumar, Riccardo Paccagnella, Paul Murley, Eric Hennenfent, Joshua Mason, Adam Bates, and Michael Bailey. Skill squatting attacks on amazon alexa. In William Enck and Adrienne Porter Felt, editors, *27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018*, pages 33–47. USENIX Association, 2018.

[27] Josephine Lau, Benjamin Zimmerman, and Florian Schaub. Alexa, are you listening?: Privacy perceptions, concerns and privacy-seeking behaviors with smart speakers. *PACMHCI*, 2(CSCW):102:1–102:31, 2018.

[28] Edward Loper and Steven Bird. Nltk: The natural language toolkit. In *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1*, ETMTNLP '02, pages 63–70, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[29] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. The stanford corenlp natural language processing toolkit. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, System Demonstrations*, pages 55–60. The Association for Computer Linguistics, 2014.

[30] Alaa Mohasseb, Mohamed Bader-El-Den, and Mihaela Cocea. Question categorization and classification us-

ing grammar based approach. *Inf. Process. Manage.*, 54(6):1228–1243, 2018.

[31] Takeshi Sugawara, Benjamin Cyr, Sara Rampazzi, Daniel Genkin, and Kevin Fu. Light commands: Laser-based audio injection on voice-controllable systems. 2019.

[32] Qi Wang, Pubali Datta, Wei Yang, Si Liu, Adam Bates, and Carl A. Gunter. Charting the attack surface of trigger-action iot platforms. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019*, pages 1439–1453. ACM, 2019.

[33] Xuejing Yuan, Yuxuan Chen, Yue Zhao, Yunhui Long, Xiaokang Liu, Kai Chen, Shengzhi Zhang, Heqing Huang, Xiaofeng Wang, and Carl A. Gunter. Commandersong: A systematic approach for practical adversarial voice recognition. In William Enck and Adrienne Porter Felt, editors, *27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018*, pages 49–64. USENIX Association, 2018.

[34] Guoming Zhang, Chen Yan, Xiaoyu Ji, Tianchen Zhang, Taimin Zhang, and Wenyuan Xu. Dolphinattack: Inaudible voice commands. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, pages 103–117. ACM, 2017.

[35] Nan Zhang, Xianghang Mi, Xuan Feng, XiaoFeng Wang, Yuan Tian, and Feng Qian. Dangerous skills: Understanding and mitigating security risks of voice-controlled third-party functions on virtual personal assistant systems. In *2019 IEEE Symposium on Security and Privacy, SP 2019, San Francisco, CA, USA, May 19-23, 2019*, pages 1381–1396. IEEE, 2019.

[36] Yangyong Zhang, Lei Xu, Abner Mendoza, Guangliang Yang, Phakpoom Chinprutthiwong, and Guofei Gu. Life after speech recognition: Fuzzing semantic misinterpretation for voice assistant applications. In *26th Annual Network and Distributed System Security Symposium, NDSS 2019, San Diego, California, USA, February 24-27, 2019*. The Internet Society, 2019.

Appendix

A Custom Skill Elements

We show the elements needed for a custom skill in Table 6.

B Utterance Distribution

We list the length distribution of the sample utterances. As shown in the Figure 8, only 0.8% of them are longer than 15. So we select 15 as the threshold of S_l.

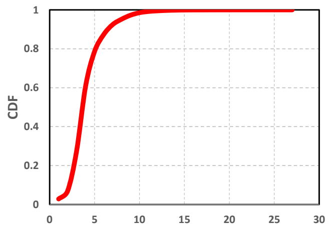
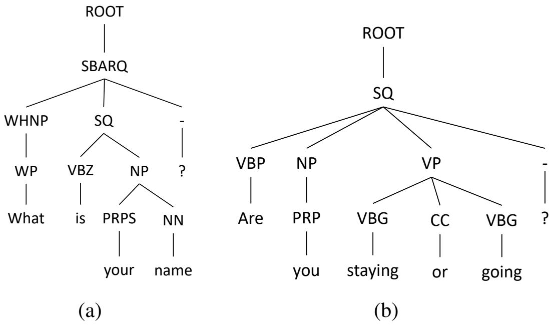


Figure 8: The distribution of the length of utterances

C Constituency-based Parse Tree Samples

We show two samples in Figure 9.



(a)

(b)

Figure 9: Examples of constituency-based parse tree

D Examples of Virtual Users

We list the information of three virtual users in Table 10. These three virtual users constitute the relationship of a father, a mother and a son.

E Questions Cannot Be Handled

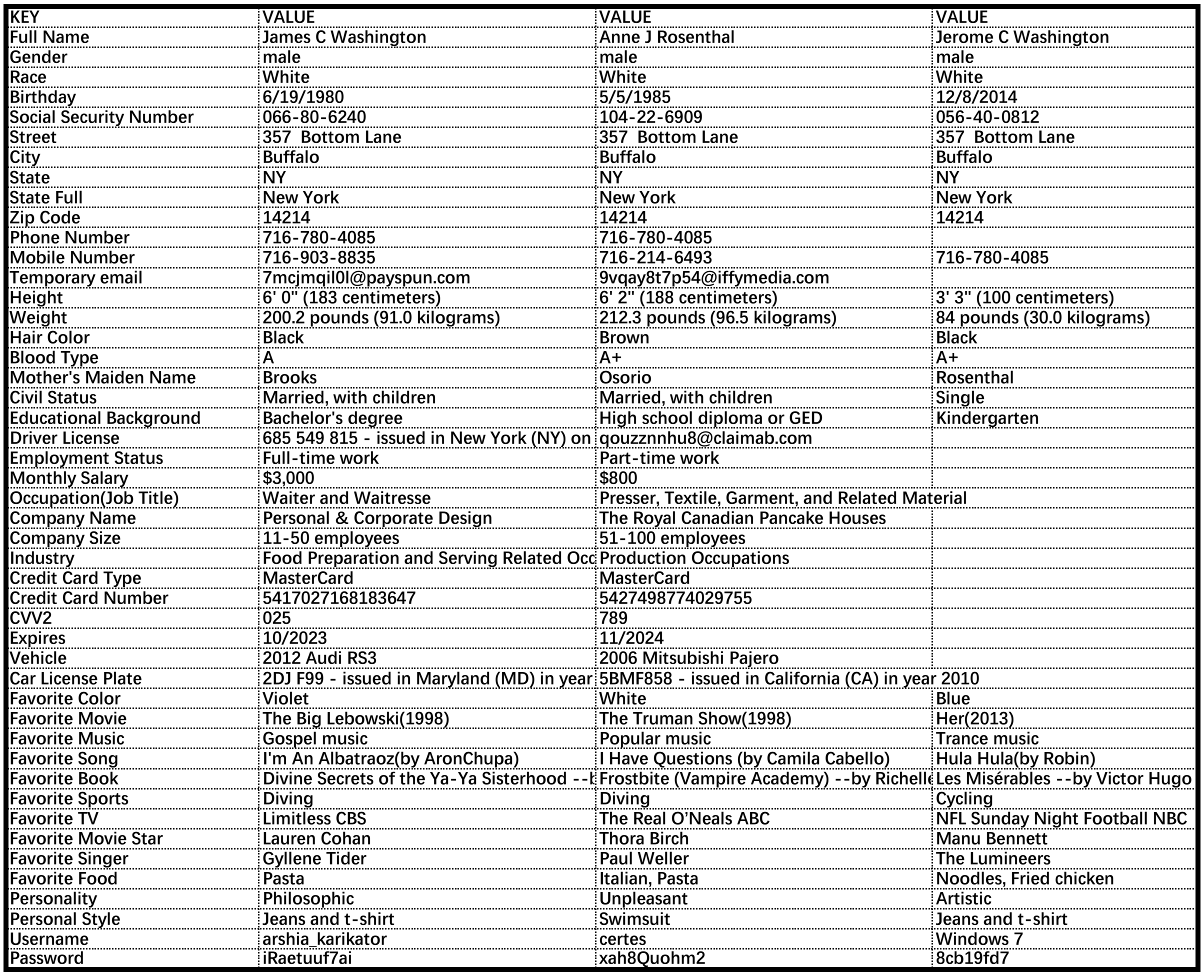
We show the questions which cannot be handled by SkillExplorer in Table 7.

F The Skills in Amazon Market

We show the skill numbers of different categories in Table 8.

Elements	Description	Example
Invocation name	Only needed for custom skills and can be used for identifying desired skills. Mentioning an invocation name explicitly can wake up the specific skill straightforward.	“Plan My Trip” has the invocation name “plan my trip”
A cloud-based service	To handle the structured JSON-format requests from Alexa, skill developers can choose either an AWS Lambda cloud or a custom web service (only suiting custom skills).	AWS Lambda cloud
Intents	An intent represents an action that fulfills a user’s spoken request. It can optionally have parameters which officially called slots.	Intent “PlanMyTrip” with slot “fromCity”, “toCity”, “travelDate”.
Sample utterances	A set of likely spoken phrases mapped to the intents to help Alexa deal with response, which should include as many representative phrases as possible.	“i want to visit {toCity}” is mapped to intent “PlanMyTrip”.

Table 6: Custom skill elements



KEY	VALUE	VALUE	VALUE
Full Name	James C Washington	Anne J Rosenthal	Jerome C Washington
Gender	male	male	male
Race	White	White	White
Birthday	6/19/1980	5/5/1985	12/8/2014
Social Security Number	066-80-6240	104-22-6909	056-40-0812
Street	357 Bottom Lane	357 Bottom Lane	357 Bottom Lane
City	Buffalo	Buffalo	Buffalo
State	NY	NY	NY
State Full	New York	New York	New York
Zip Code	14214	14214	14214
Phone Number	716-780-4085	716-780-4085	
Mobile Number	716-903-8835	716-214-6493	716-780-4085
Temporary email	7mcjmqil0l@payspun.com	9vqay8t7p54@iffymedia.com	
Height	6' 0" (183 centimeters)	6' 2" (188 centimeters)	3' 3" (100 centimeters)
Weight	200.2 pounds (91.0 kilograms)	212.3 pounds (96.5 kilograms)	84 pounds (30.0 kilograms)
Hair Color	Black	Brown	Black
Blood Type	A	A+	A+
Mother's Maiden Name	Brooks	Osorio	Rosenthal
Civil Status	Married, with children	Married, with children	Single
Educational Background	Bachelor's degree	High school diploma or GED	Kindergarten
Driver License	685 549 815 - issued in New York (NY) on	qouzznnhu8@claimab.com	
Employment Status	Full-time work	Part-time work	
Monthly Salary	$3,000	$800	
Occupation(Job Title)	Waiter and Waitresse	Presser, Textile, Garment, and Related Material	
Company Name	Personal & Corporate Design	The Royal Canadian Pancake Houses	
Company Size	11-50 employees	51-100 employees	
Industry	Food Preparation and Serving Related Occ	Production Occupations	
Credit Card Type	MasterCard	MasterCard	
Credit Card Number	5417027168183647	5427498774029755	
CVV2	025	789	
Expires	10/2023	11/2024	
Vehicle	2012 Audi RS3	2006 Mitsubishi Pajero	
Car License Plate	2DJ F99 - issued in Maryland (MD) in year	5BMF858 - issued in California (CA) in year 2010	
Favorite Color	Violet	White	Blue
Favorite Movie	The Big Lebowski(1998)	The Truman Show(1998)	Her(2013)
Favorite Music	Gospel music	Popular music	Trance music
Favorite Song	I'm An Albatraoz(by AronChupa)	I Have Questions (by Camila Cabello)	Hula Hula(by Robin)
Favorite Book	Divine Secrets of the Ya-Ya Sisterhood --b	Frostbite (Vampire Academy) --by Richelle	Les Misérables --by Victor Hugo
Favorite Sports	Diving	Diving	Cycling
Favorite TV	Limitless CBS	The Real O’Neals ABC	NFL Sunday Night Football NBC
Favorite Movie Star	Lauren Cohan	Thora Birch	Manu Bennett
Favorite Singer	Gyllene Tider	Paul Weller	The Lumineers
Favorite Food	Pasta	Italian, Pasta	Noodles, Fried chicken
Personality	Philosophic	Unpleasant	Artistic
Personal Style	Jeans and t-shirt	Swimsuit	Jeans and t-shirt
Username	arshia_karikator	certes	Windows 7
Password	iRaetuuf7ai	xah8Quohm2	8cb19fd7

Figure 10: Examples of Virtual Users

Question	Text
Q1	Okay, player one tell me a name, by saying player one is, followed by the name.
Q2	Ok, Here's FakeNBAFreeAgency. Welcome to Fake NNBA Free Agency Search. I can help you find the latest market news. Which team are you looking for?
Q3	Here are some things you can say: Give me an attraction. Tell me about Hamilton Wenham. Tell me the top five things to do. What would you like to do?
Q4	What SGLs do you want to look up?
Q5	You can say, Service Times, Location, Phone Number Help for more options or stop.
Q6	Ok, Here's QuizTimeWelcome to the States of India Quiz Game! You can ask me about any of the twenty nine states and their capitals, or you can ask me to start a quiz. What would you like to do?
Q7	what novel title do you want me to check for updates?
Q8	Which Pill would you like to add
Q9	Ok, Here's Karate FightWelcome to Karate fight! What are the names of the two fighters?
Q10	You can reach us at 972.989.5858. Or email at RA-energy@verizon.net What would you like to do next?

Table 7: Questions cannot be handled

Questions	Frequency
would you like more information?	2469
say, service times, location, phone number, or goodbye	1316
say yes for more options or no thanks	1305
say, service times, location, phone number, the word repeat at anytime to hear the last thing i said or goodbye	1175
would you like to hear, service times, location, phone number or ask for help to hear more options.	1064

Table 9: The top 5 questions mentioned by skills

Skill type	Total skill	Custom skill
Business & Finance	3336	1874
Connected Car	115	96
Education & Reference	6422	5797
Enterprise	4	4
Food & Drink	1336	1253
Games & Trivia	11413	10881
Kids	2684	2613
Lifestyle	10405	9165
Local	1223	1097
Movies & TV	869	800
Music & Audio	8743	7934
News	6394	1110
Novelty & Humor	3360	3226
Productivity	3737	3233
Shopping	283	246
Smart Home	2204	768
Social	1224	1134
Sports	1516	1012
Travel & Transportation	1161	1110
Utilities	803	779
Weather	834	733
Total	68066	54865

Table 8: The number of skills in different categories

G Top 5 Questions

We show the top 5 most frequently questions mentioned by skills in Table 9.

H Rules of Invocation Names in Amazon

(1) Amazon does not allow one-word invocation name unless it is unique to the developer's brand/intelligent. (2) Two-word invocation names are not allowed if it contains definite article ("the"), indefinite article ("a", "an") or preposition ("for", "to", "of," "about," "up," "by," "at," "off," "with"). (3) Invocation names cannot be a person or a place name. (4) Invocation name cannot contain skill's launch word such as "open," "tell," etc. and connecting words. include "to," "from," "in," etc. (5) The invocation name cannot contain the wake words "Alexa," "Amazon," "Echo," or the words "skill" or "app". (6) The invocation name must to be lowercase, and other characters like numbers must be spelled out. (7) Invocation name should be distinctive to help users wake up accurately.