Sorry, I don't Understand: Improving Voice User Interface Testing

Emanuela Guglielmi
STAKE Lab
University of Molise
Italy
emanuela.guglielmi@unimol.it

Giovanni Rosa
STAKE Lab
University of Molise
Italy
giovanni.rosa@unimol.it

Simone Scalabrino
STAKE Lab
University of Molise
Italy
simone.scalabrino@unimol.it

Gabriele Bavota
SEART @ Software Institute
Università della Svizzera italiana
Switzerland
gabriele.bavota@usi.ch

Rocco Oliveto
STAKE Lab
University of Molise
Italy
rocco.oliveto@unimol.it

ABSTRACT

Voice-based virtual assistants are becoming increasingly popular. Such systems provide frameworks to developers on which they can build their own apps. End-users can interact with such apps through a Voice User Interface (VUI), which allows to use natural language commands to perform actions. Testing such apps is far from trivial: The same command can be expressed in different ways. To support developers in testing VUIs, Deep Learning (DL)-based tools have been integrated in the development environments (*e.g.,* the Alexa Developer Console, or ADC) to generate paraphrases for the commands (seed utterances) specified by the developers. Such tools, however, generate few paraphrases that do not always cover corner cases. In this paper, we introduce VUI-UPSET, a novel approach that aims at adapting chatbot-testing approaches to VUI-testing. Both systems, indeed, provide a similar natural-language-based interface to users. We conducted an empirical study to understand how VUI-UPSET compares to existing approaches in terms of (i) correctness of the generated paraphrases, and (ii) capability of revealing bugs. Multiple authors analyzed 5,872 generated paraphrases, with a total of 13,310 manual evaluations required for such a process. Our results show that, while the DL-based tool integrated in the ADC generates a higher percentage of meaningful paraphrases compared to VUI-UPSET, VUI-UPSET generates more bug-revealing paraphrases. This allows developers to test more thoroughly their apps at the cost of discarding a higher number of irrelevant paraphrases.

CCS CONCEPTS

• **Software and its engineering → Software evolution**; **Maintaining software**; *Software defect analysis.*

ASE '22, October 10–14, 2022, Rochester, MI, USA

ACM ISBN 978-1-4503-9475-8/22/10…$15.00
https://doi.org/10.1145/3551349.3556934

KEYWORDS

voice user interfaces, software testing, NLP

ACM Reference Format:
Emanuela Guglielmi, Giovanni Rosa, Simone Scalabrino, Gabriele Bavota, and Rocco Oliveto. 2022. Sorry, I don't Understand: Improving Voice User Interface Testing. In *37th IEEE/ACM International Conference on Automated Software Engineering (ASE '22), October 10–14, 2022, Rochester, MI, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3551349.3556934

1 INTRODUCTION

Voice is a powerful tool in human-computer interaction, as it represents a fundamental means of human communication [25]. Voice-based virtual assistants such as Google Assistant, Alexa, and Siri quickly spread among users thanks to their pervasive integration with external services (*e.g.,* home automation devices) and their ease of use: It is estimated that 8.4 billion voice-based virtual assistants will be in use by 2024 [24]. Voice-based virtual assistants allow users to perform a wide range of basic actions, such as checking the weather or setting timers. On top of that, they provide developers with frameworks through which the supported range of actions can be extended by developing specific applications. An example of such apps is the Twitter Reader Alexa skill [41]: Such an app allows users to access a basic set of Twitter features, such as reading the trending topics or writing a tweet. Instead of using a typical Graphic User Interface (GUI), users interact with these apps through a Voice User Interface (VUI) [15], using natural language commands and queries to perform actions or to acquire information. To support such interactions, apps' developers must define a Voice Interaction Model (VIM) [7], which maps states and example utterances that users might pronounce to actions that must be triggered.

The Artificial Intelligence model behind voice-based virtual assistants makes them quite tolerant with respect to small variations between the pronounced and the expected utterance. For example, if the expected utterance is "*what are the news?*" and the actual utterance is "*read me the news*", the desired action will be likely triggered. However, if the actual utterance is far from the expected one (*e.g.,* "*what happened today?*"), the app might not behave properly. The fact that users can use different utterances to express the same command/query, makes the testing of VUIs far from trivial.

Previous research introduced approaches for automatically testing GUIs [30, 33, 36]. However, research in VUI-testing is still at its early stages, and only a few solutions exist. One of the state-of-the-practice solutions is provided in the Alexa Developer Console (ADC), the environment used to implement and publish Alexa skills. ADC integrates a Deep Learning (DL)-based tool that allows developers to generate, given a seed utterance s, a set of paraphrases P_s. The VIM associates each seed sentence s to the *intent* it triggers (I_s). Since paraphrases should trigger the same intent of the seed sentence, a generated test case is a pair $\langle p_s, I_s \rangle$, with $p_s \in P_s$. This approach generates few paraphrases and, therefore, it is unlikely to cover several corner cases. Testing VUI-based apps poses the same conceptual problems as testing chatbots. Chatbots provide an interface in which they *write* commands, thus using the same type of interaction of VUI-based apps (*i.e.,* based on natural language) but through different means (text-based *vs* voice-based). Previous research introduced a rigorous approach for automatically generating paraphrases for testing chatbots [20], which consists in (i) replacing words and phrases with synonyms, thus generating a set of candidates, and (ii) filtering candidates based on their similarity with the input utterance through a combination of metrics. It is unclear, however, to what extent this approach works with VUI-based apps, given the different means of communication used, which likely results in using expressions of different type and length.

In this paper, we introduce VUI-UPSET (Voice User Interfaces Utterance ParaphraSe gEneraTor), an approach that adapts chatbot-testing techniques to VUI-testing. Similarly to the approach by Guichard et al. [20], *i.e.,* the chatbot-testing technique we took inspiration from (GRSBV, from now on), VUI-UPSET (i) generates a large set of candidate paraphrases and (ii) filters the most promising ones. As for the first step, while GRSBV replaces a word/phrase at a time with a synonym, leaving the others unchanged, VUI-UPSET replaces also more than one, thus generating a much larger set of candidate paraphrases. The main point of novelty lies, however, in the second step: VUI-UPSET uses a DL model for checking the equivalence between the seed sentence and each generated paraphrase, while GRSBV relies on several manually crafted metrics. We run an empirical study with 20 open-source Alexa skills to understand how VUI-UPSET compares to two baselines: GRSBV [20] and the state-of-the-practice tool integrated in the ADC (ADC tool). We checked to what extent the generated paraphrases are (i) correct (*i.e.,* semantically equivalent to the seed sentence) and (ii) able to reveal bugs in the VUIs. To this aim, we manually analyzed 5,872 generated paraphrases, totaling 13,310 evaluations. Our results show that the ADC tool is the most accurate one: The paraphrases it generates are correct in ~74% of the cases, as compared to VUI-UPSET (~40%) and GRSBV (~21%). VUI-UPSET, however, generate a higher absolute number of correct paraphrases for most of the skills. On the other hand, the percentage of bug-revealing paraphrases generated by the three approaches is not significantly different. Therefore, since VUI-UPSET generates a higher number of correct paraphrases, it also generates a significantly higher number of bug-revealing paraphrases. VUI-UPSET allows developers to test more thoroughly their apps at the cost of discarding a higher number of irrelevant paraphrases. Finally, VUI-UPSET and the ADC tool are highly complementary. Therefore, the two techniques might be combined to generate more bug-revealing paraphrases.

2 BACKGROUND AND RELATED WORK

In this section we present an overview of the basic concepts behind VUIs and VUI-testing. Then, since we instantiate the approaches on the framework provided by Amazon Alexa, we explain how a VUI-based app for Alexa (*Alexa skill*) works. Finally, we describe in details the two baseline techniques we will use in our study: the tool integrated in the Amazon Developer Console (ADC tool) and the chatbot-testing approach defined by Guichard *et al.* [20].

2.1 Voice User Interfaces

A VUI is a type of human-computer interface in which users provide commands or queries in the form of spoken utterances and they receive (mostly) spoken responses from the system. While VUIs have been introduced a long time ago, they are now spreading thanks to the more advanced technologies available for natural language processing (NLP) and voice recognition. Indeed, a VUI is generally composed by two components: a speech recognition part, which translates voice into text, and an NLP component, which has the role of matching the natural language sentence to a set of predefined commands and trigger the desired action. Like any other software, VUI-based apps might be arbitrarily complex and they generally support multiple states.

Previous work on chatbot testing have addressed this problem with similar strategies. However, the field of testing VUIs (and chatbots) is still in its infancy, and only a few recent approaches exist in the literature. Cabot *et al.* [13] conducted a study aimed at identifying the relevant testing properties and techniques and their adaptation for NLP-based bots. Bird *et al.* [10] tackled the problem of training DL models for tasks related to chatbot interaction. To this aim, they introduced the "Chatbot Interaction with Artificial Intelligence" (CI-AI) framework, which augments user-generated data with paraphrases, similar to what we do in VUI-UPSET. Bozic *et al.* [12] proposed an approach based on metamorphic testing that automatically verifies the functionality of a chatbot.

According to the literature, the development of a good chatbot requires a continuous process that relies on various high-quality data. Therefore, previous work [35] introduced approache for generating paraphrases through the use of different techniques. Automatically generating a test case in this context means creating natural language sentences to give as input to the VUI (*test data*) and checking if the response behavior (which usually includes the target state of the app) is correct (*test oracle*).

Generally, the response behavior consists in (i) changing the state of the app and, optionally, (ii) providing a spoken response. A natural way of generating test cases in this context is by using an approach inspired by metamorphic testing [37]: Given an utterance u for which the response behavior b_u is expected, it is possible to change the utterance to an equivalent one (*i.e.,* a paraphrase), u', and assert that the executed behavior is still b_u.

Traditional approaches to utterance paraphrasing, such as hiring experts or crowdsourcing, are costly, time consuming, and with their own trade-offs in terms of quality [22, 42]. Automatic paraphrasing is emerging as an attractive alternative that promises a fast, scalable and cost-effective testing process.

2.2 Alexa Skills and Developer Console

Alexa [2] is Amazon's cloud-based voice assistant and is the "intelligence" that powers Amazon Echo and other devices. Apps that run on Alexa are called *skills*, and developers use an integrated environment for developing, testing and distributing them, namely the Amazon Developer Console (ADC) [4]. A skill is divided into two components: the *voice interaction model* and the *programming logic*. The former is a description of the utterances that users might use to activate different functionalities of the latter. More specifically, the voice interaction model associates *seed utterances* to specific *intents* which, in turn, trigger a specific part of the programming logic (*handlers*) to provide a response to the user. Alexa is tolerant to small variations between the utterance pronounced by the user and the expected seed utterance: For example, if the seed utterance is "*register birthday*" it is most likely that Alexa will still trigger the same intent for the pronounced utterance "*register my birthday*". This is not necessarily true when the semantic equivalence of the utterances is less trivial (*e.g.,* when the user pronounces "*take note of my birthday*"). Therefore, the larger the set of seed utterances provided by the developer, the higher the chance that Alexa recognizes the users' intent and triggers the desired action, thus improving the perceived quality of the app. Seed utterances might contain *slots*, *i.e.,* placeholders that allow users to give inputs [3]. For example, slots can be used to recognize the date of birth of the user pronounced in the sentence "*I was born on January 1st, 2000*". The ADC can be used to test a given skill by providing a set of utterances associated with the expected intent that will be triggered (*annotation sets*). This feature is available in the NLU-evaluation tool of the ADC [6]. Such a tool associates to each utterance in the annotation set a binary value (*PASSED* or *FAILED*) based on the fact that the actual intent equals or not the expected one.

There are some tools that facilitate the generation of annotation sets to test the skills. KayLearch [23] relies on using a grammar to generate variants of the given utterances. However, such a tool requires developers to manually define the grammar. ADC provides a feature that allows developers to automatically generate paraphrases for seed utterances to either test their skill or to enlarge the set of supported seed utterances. The ADC tool for generating paraphrases allows developers to choose among three alternatives: (i) *Interaction model*, which simply creates the tests containing the sample utterances from the voice interaction model; *Frequent Utterances*, which creates tests based on the utterances frequently used in the *simulation* step by the users, and (iii) *Utterances Recommendation Engine*, which automatically generates paraphrases by varying the seed utterances in the voice interaction model. Since the first two alternatives are semi-automatic, in our study we focus on the fully-automated alternative, *i.e.,* the *Utterances Recommendation Engine*. From now on, when we refer to the *ADC tool* we are specifically referring to the *Utterance Recommendation Engine*. While it is not entirely clear how such a feature works, the official documentation mentions that it is based on machine-learning [5]. There are several papers presented by researchers in the Amazon Alexa team [34, 38, 39] that describe possible solutions that might have been adopted in the recommendation engine. If this is the case, the approach behind the Utterances Recommendation Engine is based on encoder-decoder deep recurrent neural networks.

2.3 Generating Paraphrases for Testing Chatbots

While VUIs have their own peculiarities, they are conceptually similar to the textual interface used in chatbots [11]. A *chatbot* is a software system designed to answer human questions in text format, according to the specifications for which it was designed. Over time, chatbots have embedded increasingly sophisticated algorithms to create more natural and complex dialogues. Therefore, we describe below in details the only approach involving both (i) a paraphrase generation and (ii) a filtering phase (to the best of our knowledge) defined in the literature to generate paraphrases for testing chatbots, since we use it a starting point for our approach and as a baseline in our comparison [20].

Guichard *et al.* [20] introduce a rule-based approach (as opposed to the data-driven paradigm) that generates paraphrases aimed at evaluating the robustness of chatbots. For simplicity, from now on we call such an approach GRSBV (from the initials of the authors' surnames). GRSBV involves, as a first step, the selection of the original utterances present within the intent of the conversational agent. The authors assume that all input utterances are grammatically and syntactically correct with no spelling mistakes. The workflow of GRSBV support the generation of candidate paraphrases through lexical substitution, and the filtering of non-equivalent paraphrases through a set of metrics. We describe such steps in details below.

2.3.1 Generation of Candidate Paraphrases. First, GRSBV transforms the input utterances to lower-case as a preprocessing step to avoid case sensitivity issues. The utterance is then tokenized and each token is tagged with its respective Part-of-Speech (PoS). GRSBV performs dependency parsing among tokens to identify phrasal verbs. Tokens that appear in the stop word list are left as they are. For each of the remaining ones, GRSBV runs lemmatization and it uses the lemmas for finding synonyms in a lexical database, *i.e.,* WordNet [32]. The synonyms obtained are then inflected (*i.e.,* pluralization, singularization, and conjugation), based on the inflection of the respective original token before lemmatization. To generate higher-quality candidate paraphrases, GRSBV changes a token at a time with each of its respective synonyms. Hence, it does not consider the simultaneous substitution of multiple tokens in the input utterance.

2.3.2 Filtering Paraphrases. To further ensure the quality of the generated paraphrases, GRSBV combines some of the strategies proposed by Hassan *et al.* [21] to score the candidate paraphrases to filter out poor ones. Many words have different meanings depending on the context in which they are found. Given a candidate paraphrase p generated from the input utterance i, and the word in the original utterance w_i which was replaced with a synonym w_p in the paraphrase, the following metrics are computed.

Language Model (LM). A 5-gram language model is built using an English corpus. Then, it is used to predict the probability of every token in p from the 6th to the last one, given the previous five tokens. The LM score is computed as the average probability obtained for such tokens. If the candidate paraphrase contains 5 or less tokens, the assigned score is 0. *Rationale*: The higher the naturalness of p, the higher the likelihood that p is correct.

Translation Pivoting (TP). The candidate paraphrase p is translated into a foreign language (*i.e.,* French) and it becomes $p_{E \to F}$. Then, $p_{E \to F}$ is translated into the original language (*i.e.,* English) again, and it becomes $p_{E \to F \to E}$. If w_p is still present in $p_{E \to F \to E}$, the TP score is 1; otherwise, it is 0. *Rationale*: If w_p makes sense in the context, the translation in English of the equivalent sentence in the foreign language still contain w_p.

Word Vectors (WV). The Word2Vec model [31] is used to compute the lexical similarity between w_p and all the tokens in i which are not stop-words. WV is equal to the average of such values. *Rationale*: If w_p fits the context, the WV score will be higher (*i.e.,* it will be similar to the other tokens in i).

Web Search (WS). A Web Search engine (specifically, Microsoft Bing) is used to search p (exact match of the whole sentence). The WS score is equal to the total number of web pages retrieved. *Rationale*: The higher the number of pages retrieved, the higher the likelihood that p is correct.

Word-Sense Disambiguation (WSD). The Simplified Lesk algorithm implemented in Pywsd [40] is used to retrieve the most likely sense of w_i. If the synset of the most common sense of w_i, identified by using WordNet [32], contains w_p, the WSD score is 1; otherwise, it is 0. *Rationale*: If the a synonym related to the most common meaning of a word is used, it is more likely that p is correct.

Lexical Frequency (LF). The synsets related to the all the senses of w_i are identified through WordNet [32]. The LF score is equal to the number of synsets for that contain w_p. *Rationale*: The higher the number of senses for which w_p is a valid alternative of w_i, the higher the likelihood that w_p is a correct synonym, regardless the sense of w_i.

The metrics are then normalized and linearly combined using weights identified through a genetic algorithm. Finally, a threshold (0.59) is used to filter out paraphrases not sufficiently good. It is worth noting that GRSBV requires the execution of many different models and tools, some of which require training (*e.g.,* LM) or cannot be used for free (*e.g.,* WS and TP). This makes the approach hard to reproduce and slow in the execution.

3 GENERATING PARAPHRASES FOR ROBUST TESTING VUIS

VUI-UPSET (Voice User Interfaces Utterance ParaphraSe gEneraTor) is an approach that, given a seed sentence s generates several paraphrases of s, P_s. VUI-UPSET uses the same workflow defined in the work by Guichard *et al.* [20]. First, it generates a set of candidate paraphrases, and then it filters them. As compared to the approach by Guichard *et al.* [20] there are three main variations. First, we use Part-of-Speech (PoS) tagging on the seed sentence and define strategies to handle words based on their PoS. Second, we change more than one synonym at a time to generate a higher number of candidate paraphrases. Third, we replace the convoluted and expensive filtering technique used in [20], which requires the computation of six metrics, with a simpler approach that relies on a Deep Learning model trained to check the semantic equivalence of two sentences. Fig. 1 depicts the workflow of VUI-UPSET. We describe below the main steps behind it.

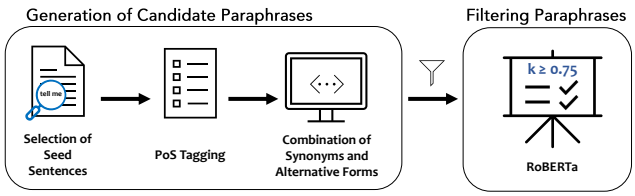

Figure 1: The workflow of VUI-UPSET.

3.1 Generation of Candidate Paraphrases

First, the developer selects the seed sentences of interest. Then, for each of them, we use a PoS tagger to assign each word with a PoS. We extract the set of synonyms and PoS-sensitive variations for each word (or phrase) in the seed sentence, and we combine such sets to generate the candidates (using the cartesian product). We describe below in details all the steps.

Selection of Seed Sentences. To ensure the generation of good quality utterances, the developer manually selects the seed utterances for which paraphrases will be generated. We do not generate paraphrases for all the seed sentences for two reasons. First, we want to select only grammatically correct sentences. This is also required by GRSBV [20], as otherwise grammatically incorrect paraphrases would be consequently generated. An example of grammatically incorrect seed sentence is "*I not sure*". Second, we want to exclude sentences that are minor variations of other sentences. We do this because our approach would generate, as a consequence, very similar (or identical) paraphrases starting from them. Such near-clones would not be useful in practice since Alexa (like other virtual assistants) is able to automatically compensate for such small variations. For example, if both utterances "*where is my order*" and "*where's my order*" are present, we only keep one of them (*e.g.,* the first one). Finally, we exclude seed sentences (i) containing slots (*i.e.,* variable parts of the sentence), as handling them goes beyond the scope of VUI-UPSET, and (ii) appearing in the default skill intent (*e.g.,* AMAZON.HelpIntent) because, even though they can be expanded, they are already handled by the framework.

PoS Tagging. The idea behind the generation of the new utterances is to generate paraphrases that are semantically equivalent to the set of seed sentences manually selected in the previous step. At this point, in this second step, it is necessary to analyze the syntactic structure of the seed sentence. This can be achieved by performing a grammatical analysis of the sentence, *i.e.,* the PoS tagging. PoS tagging is a NLP task in which the goal is to find the part of speech corresponding to each word contained in a sentence. For this purpose, VUI-UPSET relies on the open source *Stanford CoreNLP* tool [29]. Based on the grammatical analysis performed, *PoS* tagging associates each word with a tag. We consider all 36 tags supported by *Stanford CoreNLP*. We defined a set of rules for each type of PoS that will be used in the paraphrase generation phase, as described in the next section.

Table 1: Rules for the utterance generation based on the Part-of-Speech Tags.

Tag	Name	Action
CC	Coordinating Conjunctions	Unchanged
CD	Cardinal Numbers	Unchanged
DT	Determiners	Unchanged
EX	Existence There	Unchanged
FW	Foreign Words	Unchanged
IN	Prepositions and Sub. Conjunctions	Unchanged
JJ	Adjectives	Synonyms or Omitted
JJR	Comparative Adjectives	Synonyms or Omitted
JJS	Superlative Adjectives	Synonyms or Omitted
LS	List Item Markers	Unchanged
MD	Modal Verbs	Unchanged
NN	Common Nouns	Synonyms
NNS	Common Nouns (Plural)	Synonyms
NNP	Proper Nouns (Singular)	Synonyms
NNPS	Proper Nouns (Plural)	Unchanged
PDT	Predeterminers	Unchanged or Omitted
POS	Possessive Endings ('s)	Unchanged
PRP	Personal Pronouns	{me | her | him | us } or Omitted
PRP	Possessive Pronouns	{my | our } or Omitted
RB	Adverbs	Synonyms or Omitted
RBR	Comparative Adverbs	Unchanged
RBS	Superlative Adverbs	Unchanged
RP	Particles	Unchanged
SYM	Symbols	Unchanged
TO	to	Unchanged
UH	Interjection	Unchanged or Omitted
VB	Verbs (base form)	Synonyms
VBD	Verbs (past tense)	Synonyms
VBG	Verbs (gerund or present participle)	Synonyms
VBN	Verbs (past participle)	Synonyms
VBP	Verbs (non 3rd person singular present)	Synonyms
VBZ	Verbs (3rd person singular present)	Synonyms
WDT	Wh-determiner	Unchanged
WP	Wh-pronoun	Unchanged
WP	Possessive wh-pronoun	Unchanged
WRB	Wh-adverb	Unchanged

Combination of Synonyms and Alternative Forms. First, we run dependency parsing using *Stanford CoreNLP* to identify and retrieve dependencies of interest among words. This is particularly important for finding and correctly handling phrasal verbs [20]. For each token, we then verify that it is not a stop word by using a predefined stop word list [1]. In case it is classified as a stop word it remains unchanged in all the generated paraphrases. At this point, based on the PoS tagged in the previous step, we apply a consequent strategy, as described in Table 1. *Unchanged* means that the word must be left in the sentence as is in all the generated paraphrases. Some words, indeed, should/can not be modified. For example, prepositions or foreign words are never changed. *Omitted* means that the word can be completely removed without altering the meaning of the sentence. Therefore, some paraphrases will contain such a word, while others will not. *Synonyms* means that the word can be replaced with a synonym. For some specific words, such as pronouns, we manually identified and determined a set of alternatives. For example, personal pronouns can be replaced with "my" or "our". All the phrases for which VUI-UPSET decided to use a "Synonyms" strategy are lemmatized and we use WordNet [32] for finding, for each of them, the complete list of synonyms. At this point, each phrase is represented as a set of alternative forms, based on the previously described strategies. We use the cartesian product to generate candidate paraphrases. Finally, similarly to Guichard *et al.* [20], we inflect the replaced words as the original words.

For example, if a word was plural in the seed sentence, its variation is pluralized as well in the paraphrase. We use the *Pattern* approach [16], available as a Python library, to achieve this goal. For example, let us consider the seed sentence *"register my birthday"*. From the PoS analysis we get (register, VB) (my, PRP) (birthday, NN). Then, in the paraphrase generation step we are going to consider the following combinations: {remember|file|record|...} {|my|our} {birthday|natal day} and finally we get a set of candidate paraphrases (*e.g., record my birthday*, *record our birthday*, *file our birthday*, *record our natal day*, *remember birthday*).

3.2 Filtering Paraphrases

It is possible that some paraphrases generated in the previous step are not fully equivalent to the seed sentence since synonyms might refer to different meanings of a word. For example, the candidate paraphrase "time mark" might be generated from the seed sentence "time check". While "mark" is a synonym of "check", the two words have different senses in this context (*i.e.,* "control" in the seed sentence and "symbol of ticking off" in the paraphrase). Therefore, it is necessary to discard paraphrases that are not semantically equivalent to the seed sentence. To that end, VUI-UPSET integrates an approach based on DL to filter out the generated sentences that are not valid. Specifically, we use a semantic similarity model, RoBERTa [26], which is a state-of-the-art NLP model, extension of the BERT model. RoBERTa has the same architecture as BERT [17], but uses a Byte Pair Encoding (BPE) as a tokenizer and a different pre-training objective. We used a pre-trained model available on HuggingFace [18]. In particular, we used the "large" version of RoBERTa, trained on the STS Dataset [14], containing five English-language corpora of varying size and domain, totaling over 160GB of uncompressed text, for 500k steps. Given a pair of sentences, the model returns a score between 0 and 1, which indicates the likelihood that the two sentences are semantically equivalent. We run RoBERTa for each pair $\langle seed, paraphrase \rangle$; we discard paraphrases for which the returned score is below a given threshold k. We discuss how we tuned such a parameter in Section 4.

4 EMPIRICAL STUDY DESIGN

The *goal* of our study is to evaluate the effectiveness of VUI-UPSET in the automatic generation of paraphrases for testing VUI-based apps. In particular, we want to assess: (i) the semantic equivalence between the input seeds and the utterances generated by VUI-UPSET and (ii) the extent to which our approach allows developers to identify bugs in the behavior of an *Alexa* skill. As baseline approaches, we exploit the work by Guichard *et al.* [20] and the standard generation tool offered by the *Amazon Developer Console*.

In particular, we answer the following research questions (RQs):

- RQ_1: *To what extent does VUI-UPSET generate valid paraphrases?* This RQ aims at evaluating the quality of the paraphrases generated by VUI-UPSET in terms of semantic equivalence compared to the original (seed) sentence.
- RQ_2: *To what extent do the paraphrases generated by VUI-UPSET allow to reveal bugs?* With this RQ we want to understand if VUI-UPSET allows developers to generate a higher number of paraphrases that reveal bugs compared to the baseline approaches.

4.1 Study Context

To answer our RQs, we have performed the evaluation on a dataset of 20 skills. These skills have been manually selected starting from the list of most popular open source skills on GitHub (with popularity based on number of stars). Several skills have been discarded since not properly working, while for the ones working we checked if both the voice interaction model and the programming logic were available. As for the first one, we only considered interaction models defined for the English language (en-US). As for the second, we focused on skills written in *JavaScript*, as it is commonly used to implement Alexa skills.

We excluded skills with multi-step dialogues between *Alexa* and the user. The reason for this choice is based on the fact that the management of skills that involve an interaction based on a continuous dialogue must be done considering the states that allow to predict the possible responses (*e.g.,* through the use of a graph of states). This goes beyond the scope of VUI-UPSET. Moreover, each selected *Alexa* skill was imported into the *Amazon Developer Console* to be verified. We executed some basic samples defined in the interaction model of the skill, excluding again the ones for which we did not observe a correct functioning (*i.e.,* no correspondence between the Voice Interaction Model and the Programming Logic). Table 2 describes all the *Alexa* skills selected for our study.

4.2 Experimental Procedure

We compare VUI-UPSET to the two baselines (*i.e.,* the ADC tool and GRSBV) described in Section 2. While we directly use the ADC tool, we needed to re-implement GRSBV since it is not publicly available. We carefully followed the indications provided in the paper to achieve this goal.

To answer RQ_1, we evaluate the semantic equivalence between the sentences generated by VUI-UPSET and the two baselines. First, we manually selected the seed sentence for each *Amazon Alexa* skill in our dataset. In the selection of the utterances to consider, we chose the ones that (i) allowed us to cover all the intents of the skill, (ii) were well-constructed to satisfy the requirements of both VUI-UPSET and GRSBV (see Section 3), and (iii) did not contain slots. Second, we executed VUI-UPSET and GRSBV providing as input the selected seed sentences, obtaining for each skill a set of test utterances filtered by the semantic equivalence model. Then, we executed the ADC tool directly from the Amazon Developer Console, after having uploaded the skill and selected the seed utterances of interest.

The number of paraphrases generated by VUI-UPSET and GRSBV was very high (7,822 and 4,412, respectively), and manually evaluating their equivalence with the corresponding seed sentence would have been infeasible. Therefore, for such approaches, we randomly selected a sample of the generated utterances for manual evaluation. We choose the size of the sample of each skill so that it allows us to have a 5% margin of error with 95% confidence level. In total, we selected 2,929 instances for VUI-UPSET and 2,486 for GRSBV. As for the ADC tool baseline, we evaluated all the generated paraphrases because the number was much lower (*i.e.,* 457). As a result, in total, we evaluated 5,872 paraphrases.

Each generated paraphrase was independently evaluated by at least two authors, with three authors involved overall (*i.e.,* each paraphrase was assigned to two of the three authors involved in the manual validation). More specifically, each author checked whether the generated paraphrase and the respective seed sentence were semantically equivalent. We say that two different sentences are semantically equivalent if a human might use them interchangeably to express the same intent and, therefore, to trigger the same action. For example, the sentence "*record my birthday*" is semantically different from the sentence "*don't forget my birthday*"; however, if users use the latter they most likely want to trigger the same action they would express with the former.

Conflicts that arose after the comparison of the evaluations were resolved through a discussion among the three evaluators aiming at reaching consensus. The two evaluators originally assigned to each paraphrase had disagreements in 94 cases in total for ADC tool (*i.e.,* ~21%), 884 cases for VUI-UPSET (*i.e.,* ~30%) and 575 cases in total for GRSBV (*i.e.,* 23%) while independently evaluating the correctness of the paraphrases. After discussion, consensus was reached for all the paraphrases. In total, we manually performed 13,310 comparisons. Most of the conflicts arose in connection with borderline situations in which the paraphrases were interpreted differently by the evaluators. Furthermore, the high number of conflicts often relate to many paraphrases (*e.g.,* when an unusual synonym was used for a given word in the seed sentence). For example, from the seed sentence "register my birthday" a generated paraphrase is "file my date of birth". The use of the verb "file" instead of "register" is correct, but unusual in this context. One of the evaluators reported the whole paraphrase as equivalent, while the other did not. The same happened for all the paraphrases that used such a synonym.)

For each skill we measure the percentage of correct paraphrases. Then, given such percentages, we compare the three approaches by using the Wilcoxon Signed-Rank test [43]. The null hypothesis is that there is no difference between VUI-UPSET and the two baselines in terms of percentage of correctly generated paraphrases. We reject the null hypothesis if the p-value is lower than 0.05. We correct the p-values for multiple comparisons by using the Benjamini and Hochberg method [8] to calculate the adjusted p-values. We also compute the effect size to quantify the magnitude of the significant differences we find. We use Cliff's Delta [28] since it is non-parametric. Cliff's delta lays in the interval [-1, 1]: The effect size is **negligible** for $|\delta| < 0.148$, **small** for $0.148 \leq |\delta| < 0.33$, **medium** for $0.33 \leq |\delta| < 0.474$, and **large** for $|\delta| \geq 0.474$. Finally, we estimate the absolute number of correctly generated paraphrases over the whole population for each skill s and approach a. To do this, we consider the percentage of correctly generated paraphrases C_a^s computed on the sample, the total number of paraphrases generated with a for s, G_a^s, and the 5% margin of error given by the considered sample size (95% confidence level). Specifically, we compute the confidence interval of the number of correctly generated paraphrases for s as $P_a^s = (C_a^s \pm 0.05) \times G_a^s$.

To compare the absolute number of correctly generated paraphrases, we cannot check the difference in terms of absolute number of correct paraphrases found in the manually evaluated samples: Such numbers, indeed, strongly depend on the sample size, and we will have different sample sizes for the different approaches.

Table 2: Skills used for the empirical evaluation and their characteristics.

Skill	Description	# seeds		# generated paraphrases		
		total	selected	ADC Tool	GRSBV	VUI-UPSET
Ilama facts	provides space facts and trivia	14	8	40	50	133
Happy birthday	stores birthday information	6	6	12	103	211
City guide	recommends activities in the city	7	6	25	59	23
Quiz game	quiz game about a topic	6	5	16	66	268
Scheduling	schedules calls and appointments	9	9	17	193	102
Name the show	guess game about TV shows	18	15	35	204	167
Berry bash	quiz game about a topic	25	20	71	371	769
History facts	provides interesting historical facts	7	4	21	12	13
Particle cloud	provides access to Particle devices	19	14	19	298	391
Greeting sender	order management and sharing	17	16	11	382	319
Button trivia	single- and multi-player quiz game	24	17	23	373	523
Video app	allows to play a video	8	8	14	112	39
Salesforce	identifies opportunities on Salesforce	8	6	11	108	10
Store Amazon pay	VUI app for the Amazon Store	45	24	38	557	1168
Timers	manages custom timers	16	11	21	407	473
Pocket	manages lists of items	21	11	32	178	1421
Piano player	teaches the piano	9	7	5	172	201
Week calendar	allows to plan the vacation	12	7	15	462	1342
Crash notification	demonstrates messaging for different errors	14	9	5	231	78
Memory	memory-matching game	4	4	26	74	171
Total		289	207	457	4,412	7,822

For example, if we have two samples of 10 and 100 paraphrases for two of the approaches and we find that 9 out of 10 are correct in the first sample while 10 out of 100 are correct for the second sample, it is most likely that the first one would have achieved better result than the second one if we evaluated the whole population. Therefore, we compare the confidence intervals identified on the estimated number of correct paraphrases over the whole populations for each skill independently. If there is no overlap between the intervals of the three approaches, we say that the one that generates a significantly higher number of correct paraphrases for a given skill as compared to the other approaches (with 95% confidence level). If there is an overlap, instead, we cannot exclude that the difference is due to the chance. It is worth noting that for the ADC tool we have point intervals because we evaluated the whole population (*i.e.,* 0% margin of error).

To answer RQ_2, based on the results obtained in RQ_1, we define and execute test cases containing the generated paraphrases considering only the valid paraphrases we manually identified to answer RQ_1. The execution of the generated test cases is performed by through the *NLU evaluation* function from the developer console. Test cases are marked as *PASSED* when the actual intent activated for a the related paraphrase in the virtual version of Alexa matches the expected intent indicated in the VIM for the seed sentence from which the paraphrase originates. Otherwise, they are marked as *FAILED*.

For each skill, we compute the percentage of generated bug-revealing paraphrases by computing the number of *FAILED* test cases divided by the total number of valid paraphrases. We do this to understand if any of the approaches is able to outperform the others in terms of quality of the generated paraphrases. Then, we estimate the absolute number of bug-revealing paraphrases for each approach in the original population of paraphrases. To do this, we use a similar process used for RQ_1. We start from the number of correct paraphrases (both the lower and the upper bound of the confidence interval) to estimate the number of bug-revealing paraphrases. Also in this case, we consider 5% margin of error for both such values. We make sure that such an interval is lower-bounded by the number of actually found bug-revealing paraphrases in the sample (we are certain that at least such a number of bug-revealing paraphrases exist) and upper-bounded by the number of generated paraphrases. We use the Wilcoxon signed-rank test [43] to compare the percentage of bug-revealing paraphrases. The null hypothesis is that there is no difference in the percentage of bug-revealing paraphrases generated by the approaches. Also in this case, we report the Cliff's Delta [28]. To compare the absolute number of bug-revealing paraphrases, we use an analogous approach used in RQ_1: If there is no overlap between the confidence intervals of the three approaches, we say that the one that generates a larger number of bug-revealing paraphrases for a given skill is significantly better than other for such a skill (with 95% confidence level).

4.3 Tuning the Semantic Threshold for VUI-UPSET

VUI-UPSET requires the tuning of a threshold, k. We empirically determine such a threshold by running VUI-UPSET on a set of 10 *Alexa* skills different from the set we use for the empirical study. To do this, we randomly selected skills matching the same criteria we used for the empirical study, but also having a voice interaction model with at least 10 sample utterances after the manual selection. We ran VUI-UPSET without the filtering step, *i.e.,* we only generated the candidate paraphrases. Then, we computed the similarity score between the seed sentences and the generated ones. As a result, we obtained a set of pairs of sentences composed of the source seed and the generated paraphrase, plus the similarity score between them. Two of the authors independently performed a manual analysis on a sample selected among all the sentences pairs, with 5% margin of error (95% confidence level). The similarity score of each pair was not shown to the evaluators to avoid any influence. In our evaluation, we analyzed a total of 371 sentence pairs, indicating *"not correct"* with 0 and *"correct"* with 1. After comparing the two assessments, we discussed the conflicts found to motivate the final choice (0 or 1). In total, the two evaluators disagreed in 94 paraphrases generated by ADC tool, 884 paraphrases generated by VUI-UPSET and 406 paraphrases generated by GRSBV and in the end they reached consensus on all of them. As a result of this process, each pair was assigned with a correctness value and a similarity score.

We tested different threshold values, between 0.05 and 0.95 with a step of 0.05. We did not test the extremes because it would have implied to include (0) or discard (1) all the paraphrases. For each threshold, we computed well-known metrics in information retrieval, namely *precision* and *recall*. Given a candidate threshold k, we have a set of *retrieved paraphrases* (*i.e.,* paraphrases for which the similarity is higher than k). The set of *correctly retrieved paraphrases* is the subset of *retrieved paraphrases* for which, in our manual validation, we marked the paraphrases as equivalent to the respective seed. Finally, the set of *correct paraphrases* is the set of all the paraphrases that we manually marked as equivalent (regardless of the fact that they were retrieved or not). Given such sets $precision_k$ is computed as $\frac{|correctly\ retrieved\ paraphrases_k|}{|retrieved\ paraphrases_k|}$, while $recall_k$ is computed as $\frac{|correctly\ retrieved\ paraphrases_k|}{|correct\ paraphrases|}$. Additionally, we computed the F_β *score, i.e.,* the generalization of the more commonly used F_1 *score*. While the latter gives equal weight to precision and recall, with the F_β *score* it is possible to give more weight to precision ($\beta < 1$) or to recall ($\beta > 1$). The F_β score is computed as:

$$F_{\beta,k} = (1 + \beta^2) \cdot \frac{precision_k \cdot precision_k}{(\beta^2 \cdot precision_k) + recall_k}$$

In our context, we set $\beta = 0.5$, to give more importance to *precision*: Given the high number of paraphrases generated by the first step of VUI-UPSET, we are more interested in discarding non-equivalent paraphrases than in including *all* the valid paraphrases. We set k as the threshold that allows to achieve the best $F_{0.5}$ *score*.

We report in Table 3 the results of the tuning. The best $F_{0.5}$ *score* can be achieved using as a threshold $k = 0.75$. With it, we achieve 0.51 precision (*i.e.,* half of the generated paraphrases are correct) and 0.4 recall (*i.e.,* we discard 60% of the valid generated paraphrases). The identified threshold is in line with the threshold used for determining the semantic similarity between texts in traceability link recovery [27]. It is worth noting that it would be possible to select other values of k depending on the context in which VUI-UPSET is executed and on how many false-positives practitioners are willing to manually discard.

Table 3: Metrics computed for the similarity threshold evaluation.

k	Precision	Recall	$F_{0.5}$ Score
0.05	0.24	1.00	0.28
0.10	0.24	1.00	0.29
0.15	0.25	1.00	0.29
0.20	0.27	1.00	0.31
0.25	0.28	0.97	0.33
0.30	0.30	0.92	0.34
0.35	0.32	0.91	0.37
0.40	0.33	0.86	0.38
0.45	0.34	0.81	0.38
0.50	0.38	0.80	0.42
0.55	0.41	0.73	0.44
0.60	0.45	0.69	0.48
0.65	0.45	0.58	0.47
0.70	0.47	0.48	0.47
0.75	**0.51**	**0.40**	**0.49**
0.80	0.55	0.31	0.48
0.85	0.62	0.18	0.42
0.90	0.87	0.15	0.44
0.95	1.00	0.03	0.15

4.4 Replication Package

We publicly release the implementation of VUI-UPSET, our re-implementation of GRSBV and the data used for answering RQ_1 and RQ_2 in our replication package [19].

5 EMPIRICAL STUDY RESULTS

This section reports the analysis of the results for the two research questions of our study.

5.1 RQ_1: Paraphrase Correctness

The results of the analysis conducted for RQ_1 are described at the top part of Table 4. It is worth noting that, since VUI-UPSET generates a higher number of paraphrases, the size of the evaluated sample is naturally larger because of the methodology we used to select it. The first clear result we obtained is that the ADC tool generates a higher percentage of correct paraphrases (74.0%) as compared to VUI-UPSET (40.5%) and GRSBV (21.2%). For all the skills except two, indeed, the ADC tool achieves the best percentage of correct paraphrases. When comparing VUI-UPSET and the ADC tool by analyzing the percentage of correct paraphrases generated, we obtain an adjusted *p-value* of 0.003153, with a *large* effect size (δ = -0.5875). We obtain an analogous result when comparing the ADC tool with GRSBV ($p < 0.001$, δ = -0.88, *large* magnitude). For a skill (Piano player), the ADC tool fails to generate correct paraphrases: In this case, VUI-UPSET achieves the highest percentage of correct paraphrases (~38%).

Table 4: Comparison among VUI-UPSET, GRSBV, and the ADC tool in terms of correctness of generated paraphrases (top part) and their bug-revealing capability (bottom part). ○ indicates that the column refers to the evaluated sample, while ⋆ indicates that it refers to the estimated value on the whole population (95% confidence level). We report in boldface the significantly best results, while we mark with a * the cases in which the best result is shared by two approaches.

	RQ$_1$: Correctness of the Generated Paraphrases										
	ADC Tool			**VUI-UPSET**				**GRSBV**			
Skill	**#generated**	**#correct**	**%correct**	**#generated○**	**#correct○**	**%correct**	**#correct⋆**	**#generated○**	**#correct○**	**%correct**	**#correct⋆**
llama facts	40	33	**82.50%**	98	74	75.51%	**100 ± 7**	44	16	36.36%	18 ± 2
Happy birthday	12	11	**91.66%**	136	47	34.55%	**73 ± 11**	79	21	26.58%	27 ± 5
City guide	25	**22**	**88.00%**	22	17	77.27%	18 ± 1	51	7	13.72%	9 ± 2
Quiz game	16	14	**87.50%**	158	64	40.50%	**109 ± 13**	54	32	59.26%	39 ± 3
Scheduling	17	13	**76.47%**	81	54	66.66%	* **68 ± 5**	129	46	35.66%	* **69 ± 10**
Name the show	35	27	**77.14%**	117	84	71.79%	**120 ± 8**	133	28	21.05%	43 ± 10
Berry bash	71	51	**71.83%**	256	97	37.89%	**291 ± 38**	188	36	19.14%	71 ± 19
History facts	21	**17**	80.95%	13	12	**92.31%**	12 ± 0	12	2	16.66%	2 ± 0
Particle cloud	19	10	**52.63%**	194	53	27.32%	**107 ± 20**	168	27	16.07%	48 ± 15
Greeting sender	11	6	**54.54%**	175	32	18.28%	* **58 ± 16**	192	14	7.29%	* **30 ± 16**
Button trivia	23	22	**95.65%**	222	103	46.39%	**243 ± 26**	188	44	23.40%	87 ± 19
Video app	14	12	**85.71%**	35	23	65.71%	* **26 ± 2**	87	17	19.54%	* **22 ± 5**
Salesforce	11	8	**72.72%**	10	6	60.00%	6 ± 0	84	4	4.76%	7 ± 3
Store Amazon pay	38	18	**47.37%**	289	61	21.11%	**247 ± 58**	228	29	12.72%	71 ± 28
Timers	21	16	**76.19%**	205	95	46.34%	**219 ± 24**	198	26	13.13%	53 ± 20
Pocket	32	26	**81.25%**	303	86	28.38%	**403 ± 71**	122	44	36.07%	64 ± 9
Piano player	5	0	0.00%	132	50	**37.87%**	**76 ± 10**	119	10	8.40%	17 ± 7
Week calendar	15	7	**46.66%**	299	117	39.13%	**525 ± 27**	210	61	29.05%	134 ± 23
Crash notification	5	3	60.00%	65	46	**70.77%**	* **55 ± 4**	139	37	26.62%	* **61 ± 12**
Memory	26	22	**84.61%**	119	64	53.78%	**92 ± 9**	61	26	42.62%	32 ± 4

	RQ$_2$: Paraphrase Capability of Finding Bugs										
	ADC Tool			**VUI-UPSET**				**GRSBV**			
Skill	**#correct**	**#bugs**	**%bugs**	**#correct○**	**#bugs○**	**%bugs**	**#bugs⋆**	**# correct○**	**# bugs○**	**%bugs**	**#bugs⋆**
llama facts	33	8	**24.24%**	74	9	12.16%	**14 ± 5**	16	0	0.00%	1 ± 1
Happy birthday	11	1	9.09%	47	13	**27.66%**	**23 ± 4**	21	3	14.29%	5 ± 2
City guide	22	*3	13.63%	17	0	0.00%	0 ± 0	7	1	**14.28%**	* **2 ± 1**
Quiz game	14	0	0.00%	64	0	0.00%	3 ± 3	32	0	0.00%	1 ± 1
Scheduling	13	1	7.69%	54	5	**9.26%**	* **8 ± 3**	46	4	8.70%	* **7 ± 3**
Name the show	27	9	33.33%	84	37	**44.08%**	**56 ± 6**	28	10	35.71%	19 ± 3
Berry bash	51	5	**9.80%**	97	0	0.00%	8 ± 8	36	3	8.33%	7 ± 4
History facts	17	**9**	**52.94%**	12	1	8.33%	1 ± 0	2	0	0.00%	0 ± 0
Particle cloud	10	0	0.00%	53	3	5.66%	* **8 ± 5**	27	2	**7.41%**	* **5 ± 3**
Greeting sender	6	1	16.66%	32	5	15.63%	12 ± 4	14	6	**42.86%**	**20 ± 2**
Button trivia	22	6	**27.27%**	103	14	13.59%	**37 ± 13**	44	0	0.00%	3 ± 3
Video app	12	0	0.00%	23	1	4.35%	2 ± 1	17	13	**76.47%**	**21 ± 1**
Salesforce	8	0	0.00%	6	0	0.00%	0 ± 0	4	4	**1.00%**	**11 ± 1**
Store Amazon pay	18	4	22.22%	61	25	**40.98%**	**125 ± 15**	29	0	0.00%	2 ± 2
Timers	16	3	18.75%	95	35	**36.84%**	**89 ± 12**	26	0	0.00%	2 ± 2
Pocket	26	0	0.00%	86	1	**1.16%**	15 ± 14	44	0	0.00%	2 ± 2
Piano player	0	0	0.00%	50	0	0.00%	2 ± 2	10	0	0.00%	1 ± 1
Week calendar	7	1	**14.29%**	117	0	0.00%	15 ± 15	61	0	0.00%	4 ± 4
Crash notification	3	1	33.33%	46	21	**45.65%**	* **27 ± 3**	37	15	40.54%	*30 ± 4
Memory	22	0	0.00%	64	39	**60.94%**	**61 ± 5**	26	6	23.07%	8 ± 2

Similar results are achieved for Crash notification, where VUI-UPSET achieves a higher percentage of correct paraphrases (~ 71% *vs* 60%). We observed also a significant difference between VUI-UPSET and GRSBV: In this case, the adjusted *p-value* is lower than 0.001, again with a *large* effect size ($\delta = 0.755$). GRSBV never achieves the best percentage of correct paraphrases. The boxplot in Fig. 2 (left part) visually confirm the difference we numerically observed. Despite the greater number of correct paraphrases generated by the ADC tool, there are some very common paraphrases that the ADC tool and GRSBV are unable to reproduce. For example, they do not generate paraphrases with personal pronoun variations (*e.g., "give me a space fact" → "give her a space fact"*) As for the absolute number of correct paraphrases, instead, the results are different. Given the higher number of generated paraphrases and the acceptable percentage of correct paraphrases, VUI-UPSET achieves the best results for most of the skills (17 out of 20, 3 of which non-significantly different from the results achieved by GRSBV). Furthermore, it is important to note that the number of the seed sentences can affect the performance of VUI-UPSET in terms of percentage of correctly generated paraphrases.

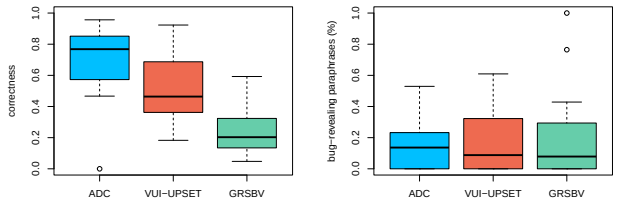

Figure 2: Distribution of percentages of correct (left) and bug-revealing (right) paraphrases over the 20 skills analyzed.

When many seed sentences are available, the number of paraphrases generated will be higher and, consequently, there will be a higher chance that semantically non-equivalent paraphrases are generated. On the other hand, this does not happen for the ADC tool since it generates a number of paraphrases that does not always depend on the number of seed sentences. For example, for the skill *Crash notification*, which has a few seed sentences (9), the ADC tool generates 5 paraphrases with a 60.00% correctness, while VUI-UPSET generates 65 paraphrases, with a 70.77% correctness. Conversely, for the skill *Barry bash*, which contains many seed sentences (20), the ADC tool generates 71 paraphrases and obtains a correctness of 71.83%, while VUI-UPSET generates 256 paraphrases and it achieves 37.89% correctness.

Answer to RQ$_1$. The ADC tool achieves a higher percentage of correct paraphrases, while VUI-UPSET generates the highest absolute number of correct paraphrases in most of the cases. GRSBV is never significantly better than the other two approaches.

5.2 RQ$_2$: Paraphrase Capability of Finding Bugs

We report the results of the analysis conducted for RQ$_2$ in the top part of Table 4. It can be noticed that the approaches appear to achieve similar results. In total, VUI-UPSET generates a slightly higher percentage of bug-revealing paraphrases (~17.6%) as compared to the ADC tool (~15.4%) and GRSBV (~12.7%). However, we can not reject any null hypothesis (as also confirmed by the boxplots in Fig. 2). Indeed, in all cases, the p-value obtained is higher than 0.05 (never lower than 0.85). Similarly, the effect size is negligible in all comparisons. Therefore, a first conclusion is that none of the compared approaches inherently generates better paraphrases for identifying bugs. Thus, intuitively, the larger the absolute number of generated paraphrases, the better (*i.e.,* the higher the number of generated bug-revealing paraphrases). Indeed, VUI-UPSET generates the highest absolute number of bug-revealing paraphrases for 10 skills out of 20 (in two cases it is not significantly different from GRSBV, while in one case is not significantly different from the ADC tool). The ADC tool achieves the best results only for two skills, but only for one (History facts) it is significantly better from both the other techniques. GRSBV, instead, achieves the best results for 7 skills, but only in 3 cases such a result is not shared with another approach. If we sum the results obtained over the 20 skills, we find that VUI-UPSET generates between 388 and 624 bug-revealing paraphrases, while the ADC tool generates 96 and GRSBV between 109 and 193.

There are skills for which VUI-UPSET works particularly well: For example, it generates 125 ± 15 bug-revealing paraphrases for the skill Store Amazon pay, which is remarkable if we compare it to the two other approaches (4 for the ADC tool and 0 for GRSBV). At the same time, we can observe that, for some skills (*i.e.,* Quiz game, Salesforce, and Piano player), we could not find any bug-revealing paraphrases generated by the three approaches, despite the high number of correct paraphrases generated by VUI-UPSET (50 in the sample, 76 ± 10 in the whole population). This probably happens because the synonyms selected, regardless of their number, are clearly similar to the words used in the original seeds. For example, in the "Piano player" skill, the word "scale" has "musical scale" as a valid synonym, and the word "lesson" is often replaced with "object lesson". Every time such substitutions happen, Alexa is still able to compensate for such small variations.

An example of bug found through VUI-UPSET is the following. For the *Particle cloud* skill, given the seed sentence *"register my date of birth,"* VUI-UPSET generates, among the other paraphrases, *"register our date of birth."* Such a paraphrase is semantically equivalent to the seed sentence but, when tested in the Alexa Developer Console, it does not allow to obtain the same result (*i.e.,* the test case fails).

Answer to RQ$_2$. The three approaches generate a similar percentage of bug-revealing paraphrases, but VUI-UPSET generates the highest absolute number of such paraphrases.

5.3 Discussion

Based on the results obtained, we identified some future research directions for researchers interested in this field and some guidelines for practitioners.

Future Research Directions. While for RQ$_1$ we can observe clear trends, *i.e.,* the results are almost the same for all the skills, this was not the case for RQ$_2$, with substantial variation among the subject skills. To analyze this phenomenon more in depth, we tried to understand how different were the paraphrases generated by the three approaches. To do this, we compute the overlap between couples of approaches by using the formula $\frac{|correct_{v_i} \cap correct_{v_j}|}{|correct_{v_i} \cup correct_{v_j}|}$.

Despite the similarity between VUI-UPSET and GRSBV, they only generate 189 shared paraphrases (~1.5%). Instead, VUI-UPSET and the ADC tool generate 11 shared paraphrases (~0.1%), while GRSBV and the ADC tool share only two paraphrases (<0.1%). Only such two paraphrases occur in all the approaches. In other words, the three compared approaches are highly complementary. It might be worth exploring to what extent combining such approaches allows to generate more interesting bug-revealing paraphrases. A particularly promising direction might be to combine the ADC tool with VUI-UPSET: The former can be used to change the form of the seed sentence (*e.g.,* from positive to negative or question), while the latter can be used to amplify the number of paraphrases by generating variations not only for the seed sentences, but also for the sentences output of the ADC tool. As for VUI-UPSET and GRSBV, a clear weak point we observed is that they sometimes fail to find adequate synonyms. This is partially due to the fact that the filtering step fails at discarding incorrect paraphrases.

However, in other cases, such a step fails because for words with a given PoS, synonyms with a different PoS are used. For example, the "start story" input utterance for the skill "Memory" is wrongly paraphrased into "first tale". This happens because "first" is a synonym of the noun "start". However, the verb "start" is used in this context. Filtering out the synonyms of the noun "start" would have allowed VUI-UPSET not to generate this paraphrase in the first place.

Guidelines for Developers. Generating paraphrases for testing VUIs is still at an early research stage and only a production-ready tool exists in practice. Still, we try to point out some guidelines for practitioners to help developers decide how to test their apps. The ADC tool is the best choice when developers want to quickly test their app. Indeed, they need to manually discard only a few incorrect sentences and they will likely find some bug-revealing paraphrases. However, safety- and securty-critical apps (*e.g.,* for home automation or home banking, respectively) might benefit from a more thorough testing phase. VUI-UPSET might help them to generate more bug-revealing sentences, at the cost of discarding a higher number of incorrect paraphrases.

6 THREATS TO VALIDITY

Threats to construct validity. We manually selected a sample of the seed utterances to use for generating paraphrases from the voice interaction models of the skills we considered based on the requirements of VUI-UPSET and GRSBV. Choosing different utterances might have resulted in different results. Also, there is a possible subjectivity introduced during manual analysis for the results of both the RQs. This threat was mitigated by using a rigorous qualitative analysis process, as described in Section 4.

Threats to internal validity. We did not directly use the VUI provided by the skills through Alexa to run the tests, but we deployed the skills in the test environment of the ADC and simulated the execution of paraphrases through the NLU-evaluation tool. Directly using the VUI might have allowed finding other issues (*e.g.,* related to failures in the speech recognition). However, this goes beyond the scope of our approach and of the other approaches we compared. Another threat is related to the fact that we needed to re-implement GRSBV since it was not available. Also, the description of some of the metrics of the filtering step in the original paper [20] were slightly ambiguous. We implemented the metrics based on our interpretation, as explained in Section 2. To foster the verifiability and the replicability of VUI-UPSET we publicly release our implementation.

Threats to external validity. Our results are based on 20 *Alexa* skills. Such a sample might not be representative of all the Alexa skills. We mitigated this risk by choosing diverse skills in terms of application domain. In our experiment, we selected *Alexa* skills developed in JavaScript. However, we expect no significant differences in skills developed in other programming languages since our approach focuses on the VIM rather than on the programming logic of the skills. Finally, it is possible that the results obtained do not generalize to other technologies, *e.g.,* the one provided by Google for their Actions. Indeed, the results of RQ_2 depend on how tolerant the framework is with variations of the pre-defined seed utterances, and this might change.

7 CONCLUSION AND FUTURE WORK

The interest in *Voice User Interface*-based apps has grown in the last years. Research on automated test for VUIs, however, is still in its infancy. In this paper, we presented VUI-UPSET, an approach that build upon previous research in chatbot-testing for automatically generating paraphrases that allow developers to test VUIs. We run a large empirical study on 20 open-source Alexa skills, and we compared VUI-UPSET with two state-of-the-art approaches, *i.e.,* the paraphrase generation tool integrated in the Amazon Developer Console (ADC tool) and GRSBV, defined for chatbot-testing. Our results show that the ADC tool generates the highest percentage of correct paraphrases, but VUI-UPSET allows to generate a higher absolute number of them. Also, while the three approaches generate a comparable percentage of bug-revealing paraphrases, VUI-UPSET generates a significantly higher absolute number of bug-revealing paraphrases for more skills than the others. Future work will be aimed at automating the very first step of VUI-UPSET (*i.e.,* selection of input seed sentences), on supporting *slots*, and on replicating the evaluation on other technologies (*e.g.,* Actions on Google). We will also experiment with the use of existing deep learning models specifically aimed at generating paraphrases [44] in the paraphrase generation phase. Finally, to better calibrate VUI-UPSET, we plan to run a study with developers to understand what number of non-equivalent paraphrases generated by VUI-UPSET they might find acceptable.

REFERENCES

[1] 2022. Stop Word List. https://countwordsfree.com/stopwords.
[2] "Amazon". 2018. Alexa. https://developer.amazon.com/en-US/alexa.
[3] "Amazon". 2018. Alexa Slots. https://developer.amazon.com/en-US/docs/alexa/custom-skills/slot-type-reference.html.
[4] "Amazon". 2018. Amazon Developer. https://developer.amazon.com/en/.
[5] "Amazon". 2018. Amazon official documentation. https://developer.amazon.com/en-US/docs/alexa/custom-skills/get-utterance-recommendations.html.
[6] "Amazon". 2018. NLU-evaluation tool. https://developer.amazon.com/it-IT/docs/alexa/smapi/nlu-evaluation-tool-api.html.
[7] "Amazon". 2018. Voice Interaction Models. https://developer.amazon.com/en-US/docs/alexa/ask-overviews/voice-interaction-models.html.
[8] Yoav Benjamini and Yosef Hochberg. 1995. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal statistical society: series B (Methodological)* 57, 1 (1995), 289–300.
[9]]berensteinautomated DM Berenstein. [n. d.]. From Automated Paraphrase Generation To Increased Robustness of Chatbots. ([n. d.]).
[10] Jordan J Bird, Anikó Ekárt, and Diego R Faria. 2021. Chatbot Interaction with Artificial Intelligence: human data augmentation with T5 and language transformer ensemble for text classification. *Journal of Ambient Intelligence and Humanized Computing* (2021), 1–16.
[11] Josip Bozic, Oliver A Tazl, and Franz Wotawa. 2019. Chatbot testing using AI planning. In *2019 IEEE International Conference On Artificial Intelligence Testing (AITest)*. IEEE, 37–44.
[12] Josip Bozic and Franz Wotawa. 2019. Testing chatbots using metamorphic relations. In *IFIP International Conference on Testing Software and Systems*. Springer, 41–55.
[13] Jordi Cabot, Loli Burgueno, Robert Clarisó, Gwendal Daniel, Jorge Perianez-Pascual, and Roberto Rodriguez-Echeverria. 2021. Testing challenges for NLP-intensive bots. In *2021 IEEE/ACM Third International Workshop on Bots in Software Engineering (BotSE)*. IEEE, 31–34.
[14] Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. SemEval-2017 Task 1: Semantic Textual Similarity Multilingual and Crosslingual Focused Evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. Association for Computational Linguistics, Vancouver, Canada, 1–14. https://doi.org/10.18653/v1/S17-2001
[15] Michael H Cohen, Michael Harris Cohen, James P Giangola, and Jennifer Balogh. 2004. *Voice user interface design*. Addison-Wesley Professional.
[16] Tom De Smedt and Walter Daelemans. 2012. Pattern for python. *The Journal of Machine Learning Research* 13, 1 (2012), 2063–2067.

[17] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[18] "Hugging Face". 2022. Hugging Face. https://huggingface.co/cross-encoder/stsb-roberta-large.

[19] Emanuela Guglielmi, Giovanni Rosa, Simone Scalabrino, Gabriele Bavota, and Rocco Oliveto. 2022. Replication Package of "Sorry, I don't Understand: Improving Voice User Interface Testing". https://doi.org/10.6084/m9.figshare.19726204.v1.

[20] Jonathan Guichard, Elayne Ruane, Ross Smith, Dan Bean, and Anthony Ventresque. 2019. Assessing the robustness of conversational agents using paraphrases. In *2019 IEEE International Conference On Artificial Intelligence Testing (AITest)*. IEEE, 55–62.

[21] Samer Hassan, Andras Csomai, Carmen Banea, Ravi Sinha, and Rada Mihalcea. 2007. Unt: Subfinder: Combining knowledge sources for automatic lexical substitution. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*. 410–413.

[22] Kuan-Hao Huang and Kai-Wei Chang. 2021. Generating syntactically controlled paraphrases without using annotated parallel pairs. *arXiv preprint arXiv:2101.10579* (2021).

[23] "KayLearch". 2018. KayLearch. https://github.com/KayLerch/alexa-utterance-generator/.

[24] Federica Laricchia. 2022. Number of digital voice assistants in use worldwide from 2019 to 2024. https://www.statista.com/statistics/973815/worldwide-digital-voice-assistant-in-use/.

[25] Kwang B Lee and Roger A Grice. 2006. The design and development of user interfaces for voice application in mobile devices. In *2006 IEEE International Professional Communication Conference*. IEEE, 308–320.

[26] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692* (2019).

[27] Andrea De Lucia, Fausto Fasano, Rocco Oliveto, and Genoveffa Tortora. 2007. Recovering traceability links in software artifact management systems using information retrieval methods. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 16, 4 (2007), 13–es.

[28] Guillermo Macbeth, Eugenia Razumiejczyk, and Rubén Daniel Ledesma. 2011. Cliff's Delta Calculator: A non-parametric effect size program for two groups of observations. *Universitas Psychologica* 10, 2 (2011), 545–555.

[29] Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*. 55–60.

[30] Ke Mao, Mark Harman, and Yue Jia. 2016. Sapienz: Multi-objective automated testing for android applications. In *Proceedings of the 25th International Symposium on Software Testing and Analysis*. 94–105.

[31] Tomáš Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: Human language technologies*. 746–751.

[32] George A Miller. 1995. WordNet: a lexical database for English. *Commun. ACM* 38, 11 (1995), 39–41.

[33] Kevin Moran, Mario Linares Vásquez, and Denys Poshyvanyk. 2017. Automated GUI testing of Android apps: from research to practice. In *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*. IEEE, 505–506.

[34] Leah Nicolich-Henkin, Taichi Nakatani, Zach Trozenski, Joel Whiteman, and Nathan Susanj. 2021. Comparing Data Augmentation and Annotation Standardization to Improve End-to-end Spoken Language Understanding Models. In *Proceedings of the 35th Conference on Neural Information Processing Systems (NeurIPS)*. 1–6.

[35] Octavany Octavany and Arya Wicaksana. 2020. Cleveree: an artificially intelligent web service for Jacob voice chatbot. *TELKOMNIKA (Telecommunication Computing Electronics and Control)* 18, 3 (2020), 1422–1432.

[36] Kabir S Said, Liming Nie, Adekunle A Ajibode, and Xueyi Zhou. 2020. GUI testing for mobile applications: objectives, approaches and challenges. In *12th Asia-Pacific Symposium on Internetware*. 51–60.

[37] Sergio Segura, Gordon Fraser, Ana B Sanchez, and Antonio Ruiz-Cortés. 2016. A survey on metamorphic testing. *IEEE Transactions on software engineering* 42, 9 (2016), 805–824.

[38] Siamak Shakeri and Abhinav Sethy. 2019. Label dependent deep variational paraphrase generation. *arXiv preprint arXiv:1911.11952* (2019).

[39] Alex Sokolov and Denis Filimonov. 2020. Neural machine translation for paraphrase generation. *arXiv preprint arXiv:2006.14223* (2020).

[40] "Liling Tan. 2014. Pywsd: Python implementations of word sense disambiguation (wsd) technologies [software]. https://github.com/alvations/pywsd.

[41] Twitter. 2022. Twitter Alexa Skill. https://www.amazon.com/dp/B01LFJO3M0.

[42] Yushi Wang, Jonathan Berant, and Percy Liang. 2015. Building a semantic parser overnight. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. 1332–1342.

[43] Robert F Woolson. 2007. Wilcoxon signed-rank test. *Wiley encyclopedia of clinical trials* (2007), 1–3.

[44] Jianing Zhou and Suma Bhat. 2021. Paraphrase Generation: A Survey of the State of the Art. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Online and Punta Cana, Dominican Republic, 5075–5086. https://doi.org/10.18653/v1/2021.emnlp-main.414