Guo *et al. Cybersecurity* (2023) 6:13
https://doi.org/10.1186/s42400-023-00150-3

Cybersecurity

RESEARCH

Open Access

SkillSim: voice apps similarity detection

Zhixiu Guo[1,2], Ruigang Liang[1,2*], Guozhu Meng[1,2] and Kai Chen[1,2]

Abstract

Virtual personal assistants (VPAs), such as Amazon Alexa and Google Assistant, are software agents designed to perform tasks or provide services to individuals in response to user commands. VPAs extend their functions through third-party voice apps, thereby attracting more users to use VPA-equipped products. Previous studies demonstrate vulnerabilities in the certification, installation, and usage of these third-party voice apps. However, these studies focus on individual apps. To the best of our knowledge, there is no prior research that explores the correlations among voice apps.Voice apps represent a new type of applications that interact with users mainly through a voice user interface instead of a graphical user interface, requiring a distinct approach to analysis. In this study, we present a novel voice app similarity analysis approach to analyze voice apps in the market from a new perspective. Our approach, called SkillSim, detects similarities among voice apps (i.e. skills) based on two dimensions: text similarity and structure similarity. SkillSim measures 30,000 voice apps in the Amazon skill market and reveals that more than 25.9% have at least one other skill with a text similarity greater than 70%. Our analysis identifies several factors that contribute to a high number of similar skills, including the assistant development platforms and their limited templates. Additionally, we observe interesting phenomena, such as developers or platforms creating multiple similar skills with different accounts for purposes such as advertising. Furthermore, we also find that some assistant development platforms develop multiple similar but non-compliant skills, such as requesting user privacy in a non-compliance way, which poses a security risk. Based on the similarity analysis results, we have a deeper understanding of voice apps in the mainstream market.

Keywords Voice app, Similarity analysis, Skills

Introduction

Virtual Personal Assistants (VPAs), such as Amazon Alexa, Google Assistant and Xiaomi Xiao AI, are equipped on different smart devices, like smart speakers, to assist users with tasks like getting weather information and turning on the radio. In addition to the built-in functions, VPA platforms also allow third-party developers to submit their voice apps (called skills by Amazon or actions by Google[1]) to VPA app stores to provide a wider range of functions.

The ecosystem centered on VPA services is constantly growing and expanding. Most researches focus on analyzing VPAs' security, such as the security of speech recognition (Yuan et al. 2018; Chen et al. 2020). With the rapid increase of third-party skills (over 100,000 Amazon (2019)), the security of skills raises concerns. The emergence of skills expands the attack surface of VPAs, as malicious developers may develop harmful or unwanted skills (Guo et al. 2020; Kumar et al. 2018; Zhang et al. 2019a; SRLabs 2022). To mitigate such risks, VPA platforms establish a series of policies for third-party developers to follow, and they certify these skills through manual or automated processes. If a skill violates these

*Correspondence:
Ruigang Liang
liangruigang@iie.ac.cn
[1] SKLOIS, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[2] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China

[1] Hereinafter, voice apps are collectively referred to as skills.

policies, it will not be published in the voice app store. However, previous studies prove that there are vulnerabilities in the certification process, and some skills that violate policies still appear in VPA app stores (Cheng et al. 2020; Wang et al. 2021).

Some VPA platforms, like Amazon, have low barriers for creating developer accounts and do not limit the number of accounts. As a result, developers can create multiple accounts to develop similar unwanted skills without drawing reviewers' attention. Reviewers for VPA platforms can identify skills under the same developer account. However, they cannot correlate skills published by different accounts belonging to the same person or team, even if these skills are very similar (Cheng et al. 2020; Wang et al. 2021). Therefore, it is significant to detect the similarity among skills. Firstly, it can correlate similar skills and offers a new perspective on the voice app market. Secondly, it can associate different accounts belonging to the same developer to enhance the detection of suspicious or malicious behavior.

Challenge. To the best of our knowledge, no prior research explores skill similarity analysis, mainly due to the following challenges. First, the skill's code is unavailable. It is maintained on the developer's server and is not even available to platform reviewers. Most skills also have no graphical user interface (GUI). All that is available is the natural-language-based interaction content between users and skills, making it impossible to detect similar skills in the same manner as traditional software. Second, defining skill similarity is challenging. For instance, should skills with cross-content be considered similar, or should those with the same topic be considered similar? Skills are essentially software and have structural features that are specific to code, requiring more than just detecting text similarities.

Our approach. Skills have two characteristics: one is that developers implement functionality through *code*, and the other is that skills interact with users in the form of *a natural-language-based conversation*. Considering the two characteristics of skills, we design *SkillSim* to detect similarities among skills by comparing both texts and structures of skills' interaction content. In particular, SkillSim extracts different features from multiple dimensions. For example, to detect text similarity, SkillSim extracts both overall text features and key text features separately based on different purposes. In terms of structure similarity, SkillSim abstracts skill interaction content as tree structures, where each node contains an input and an output. Structure similarity is then determined based on node features. Through these dimensions, we are able to assess skill similarities from different viewpoints.

SkillSim evaluates 30,000 skills and finds that 25.9% of the skills have at least one skill with more than 70% text similarity to them. In addition, we identify three main factors contributing to the large number of similar skills in the market, such as the assistant development platforms and their limited templates. Further analysis reveals that one developer or development platform may develop multiple similar skills using the same or different accounts for certain purposes such as advertising. Additionally, some assistant development platforms develop a large number of similar and non-compliant skills, putting users' privacy at risk.

Contribution. The contributions of the paper are as follows:

- **A skill similarity detection method and a large-scale analysis.** This paper presents the first skill similarity analysis. SkillSim calculates similarities among skills based on their text and structure characteristics. The study involves a comprehensive analysis of 30,000 skills in the mainstream market, and the results are representative.
- **Interesting findings and a new perspective.** The similarity analysis results indicate that a significant portion (accounting for 25.9%) of skills have at least one other skill with more than 70% text similarity to them. We summarize three reasons behind that and propose suggestions. Our analysis reveals that some developers or platforms create multiple similar skills with different accounts for purposes such as advertising. In addition, some assistant development platforms create multiple similar but non-compliant skills, thus posing a risk to user privacy. These results offer a novel perspective on the current voice app market.

Background

VPAs and skills

VPAs, such as Amazon Alexa and Google Assistant, provide services to users through voice-based interactions. The functionality of these services can be greatly expanded through third-party skills. As with traditional apps, third-party skills are developed by third-party developers and uploaded to the VPA app stores for vetting. If a skill passes the vetting process, it can be published on the voice app stores. The code of the skill is maintained on the developer's server. Like users, VPA platform reviewers can only review a skill by interacting with it through the natural- language-based interface with sample utterances provided by the developer.

Figure 1 shows a process flow of user interaction with skills. The user first speaks a wake-up word (*e.g.*, *"Alexa"*) to wake up the VPA-equipped device (*e.g.*, smart speaker). After the VPA is awakened, the user continues

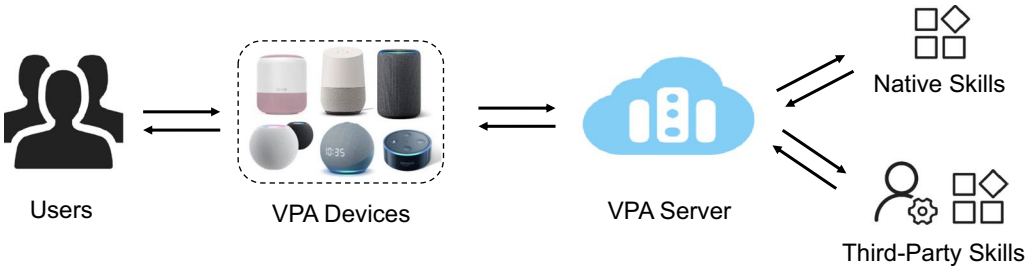


Fig. 1 User-skill interaction process flow

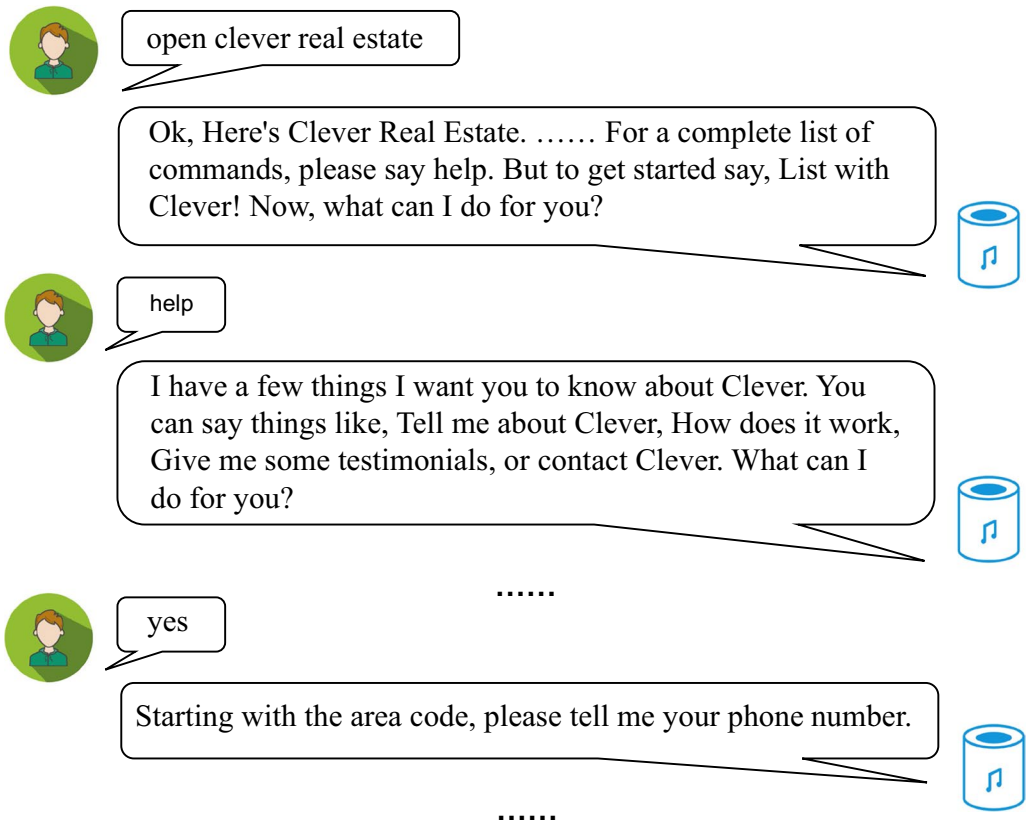


Fig. 2 An interaction path sample with the skill "Clever Real Estate"

to send voice commands combining keywords (*e.g., "open", "enable"*) with skill invocation name (*e.g., "open clever real estate"*). This command is transmitted to the VPA server via the smart speaker for analysis. The VPA server identifies the corresponding skill based on the skill invocation name (*e.g., "clever real estate"*) and sends the instruction to the server where the developer can process the command. The third-party skill returns the processed results to the user layer by layer to achieve the interaction goal. Figure 2 shows an interaction content example that a user interacts with a third-party skill called *"Clever Real Estate"*.

If third-party skills want to be published on VPA app stores, they must adhere to a set of policies established by VPA platforms and undergo review (Amazon 2022a). The VPA platform assesses these skills either manually or automatically to determine if they comply with the policies (Wang et al. 2021). For example, if a skill requests users' personal information such as name, phone number, and email, Amazon requires the skill to include a privacy policy link and configure corresponding permissions (Amazon 2022b). Despite these measures, some studies discover that the review process is vulnerable,

Table 1 Question types in skills

Question Type	Description
Yes/No questions	A Yes/No question is an interrogative construction and expects answers like "yes" or "no".
Instruction questions	An instruction question gives users direct guidance on how to answer it. Instruction questions often contain key keywords like "say", "ask".
Selection questions	A selection question contains multiple parallel options for users to choose from.
Wh questions	A Wh question begins with WH-tag.
Mix questions	A mix question contains more than one of the previous four question types.

and some skills that do not meet the policy requirements still make it to the app stores (Guo et al. 2020; Cheng et al. 2020). In addition, the VPA platform can only identify multiple skills under the same developer account, but cannot detect similar skills among different accounts, which makes it feasible for developers to create similar skills using multiple accounts (Cheng et al. 2020). As a result, it becomes critical to detect similarities among skills.

Skill interaction

To use a skill, a user installs it using sample instructions provided by the developer. The user then analyzes the content returned by the skill to generate new input. The interaction path will end when the skill ends the interaction or the user actively terminates the interaction. Figure 2 shows an example of an interaction path. Here, we briefly introduce the frequently used terms.

- **Input.** The user'scommand for skills.
- **Output.** The content returned by the skill after accepting the user's input.
- **Interaction path.** A path from the first input of the user to the final output of the skill.
- **Interaction record.** All interaction paths of a skill.
- **Question.** For the output returned by a skill, if users can parse the content and generate new inputs, the output will be a question. There are different types of questions. We refer to the classification and definition of question types in SkillExplorer (Guo et al. 2020), which is shown in Table 1.

Software clone detection

Traditional software clone detection is mainly to detect code, graphical user interface, *etc.* Among them, code clone detection can be roughly categorized into four types: identical code, renamed code, almost identical code, and semantically similar code (Bellon et al. 2007). Based on different representation forms of code (such as token, control flow graph and data flow graph), different detection methods are used to detect software clone (Ain et al. 2019; Meng et al. 2016).

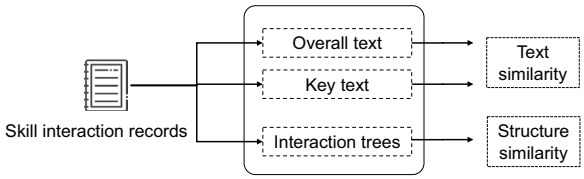


Fig. 3 The framework of SkillSim

However, skills lack graphical user interfaces and their code is not accessible. Only interaction content based on natural language text is available, thus making traditional software clone detection methods inapplicable to skills. This paper presents a novel method for identifying skill similarity based on the text and structure of skill interaction content.

Our approach

Overview

As mentioned above, when computing skill similarity, both the text and structure dimensions of skills must be considered. Figure 3 illustrates the framework of SkillSim, which inputs the interaction records of two skills and extracts their features to calculate skill similarities.

For text features, SkillSim extracts the overall text and the key text to calculate the text similarity separately. For structural features, SkillSim extracts the tree structures from skill interaction records and employs node features, such as the question type, for the calculation of structure similarity.

Text similarity

An interaction record contains all interaction paths of a skill, and each interaction path encompasses all inputs and outputs from the start to the end of a complete interaction. In this study, we focus on the skill content. The automated analysis of SkillExplorer (Guo et al., 2020) and VITAS (Li et al., 2022) generates inputs of skills from the output of the previous round. Therefore we can extract only skills' response content, *i.e.*, the outputs in interactions, and ignore the inputs.

For text similarity, two cases are considered. Case 1 involves a situation where the content is similar, but the topics may vary. For instance, the same advertisement template may be utilized for different themes. Case 2 involves a situation where the themes are identical, but

the content is distinct, as in the case of different advertisement templates being utilized for the same theme.

Therefore, we calculate the overall text similarity and key text similarity separately.

Overall text similarity. SkillSim extracts all outputs from a skill's interaction record as its overall text.[2] Next, it calculates the overall text similarity among skills. Since we are concerned with content duplication, we choose n-shingles as the granularity and use *Jaccard* to calculate the similarity. If the overall text between two skills is very similar, it can determine that the two skills are clones.

- **Jaccard similarity.** Jaccard index Jaccard (1912), also known as Jaccard similarity coefficient, is used to compare the similarities and differences between sets. Given two sets A and B, Jaccard coefficient is defined as the ratio of the intersection of A and B to the union of A and B. The larger the Jaccard coefficient value, the higher the similarity of the skills. As shown in Eq. 1, n_a is the n-shingles elements in set A, and n_b is the n-shingles elements in set B.

$$Sim_{overall}(A,B) = \frac{|n_a \cap n_b|}{|n_a \cup n_b|} = \frac{|n_a \cap n_b|}{|n_a| + |n_b| - |n_a \cap n_b|} \tag{1}$$

  However, performing pairwise comparisons in a document corpus is time-consuming because the number of comparisons grows geometrically with the size of the documents. Most of those comparisons are unnecessary because they are not similar. Therefore, before calculating the true Jaccard similarity, we perform a pre-filter.
- **Minhash and locality-sensitive hashing.** The general idea of Minhash algorithm (Broder 1997) is to use hash functions to disrupt the positions of elements uniformly, and then take the first element of each set in the new order as the features of the set. Under the condition that the hash function is uniformly distributed, the probability that the Minhash value of set S1 and set S2 are equal will be equal to the Jaccard similarity of the two sets. MinHash is essentially a Jaccard approximation. As shown in Fig. 4, MinHash can produce an n-dimensional vector from skill interaction records signature where n is much smaller than m (the total number of words in interaction records).

The basic idea of locality-sensitive hashing (LSH) Indyk and Motwani (1998) is to gather similar sets together and avoid more different sets. It divides a signature from Minhash into multiple lower-dimensional vectors, called bands. LSH uses a hash function to assign identical bands to the same hash bucket to obtain candidates of similar skills. With this approach, we can filter irrelevant skill pairs and thus focus on candidate skill pairs that are likely to be similar.

Key text similarity. Unlike the overall text, key text highlights specific themes or elements of a skill, thus distinguishing it from other skills. Key text can help SkillSim to find skills with the same topic or the same key elements. Here we choose *IDF* to calculate key text similarity.

- **Inverse document frequency.** Inverse Document Frequency (IDF) (Wu et al. 2008) is often used to evaluate the importance of a word to a document set. The main idea of *IDF* is that the fewer documents that contain a term, the better ability it will have to distinguish between categories. Therefore, IDF can help to filter out common words and keep important words. This is consistent with what we want to achieve.

The formula of *IDF* is as follows. The subscript i means the sequence number of the word w and j means the sequence number of the document d. A document is the overall text of a skill (*i.e.* outputs in an interactive record). M is the number of documents (*i.e.* the number of skills), and $|\{j : w_i \in d_j\}|$ is the number of documents in which the word w_i appears.

$$idf_i = log \frac{M}{1 + |\{j : w_i \in d_j\}|} \tag{2}$$

We use the overall text of all skills as a corpus. Based on the corpus, we calculate the *IDF* value of each word. The higher the *IDF* value, the more representative the word is. Next, we calculate key text similarity between skills. We use A and B to denote two documents, and n_a and n_b are the words in the two documents respectively. The calculation formulas are as follows. We obtain the common words in n_a and n_b and add up their IDF values, then we calculate the sum of IDF value of the words in the union set of n_a and n_b. Finally, the key text similarity is obtained by calculating the ratio of the two sums.

$$Sim_{key}(A,B) = \frac{\sum log \frac{M}{1+|\{j:w_i \in d_j\}|}}{\sum log \frac{M}{1+|\{k:w_u \in d_k\}|}}, w_i \in |n_a \cap n_b|, w_u \in |n_a \cup n_b| \tag{3}$$

Structure similarity

As mentioned above, a skill is essentially software, with its functional logic being realized through code stored on a developer server, thus possessing application-specific structural information. In this section, we abstract

[2] Skill descriptions reflect skill functions, so we also add skill descriptions to the overall text.

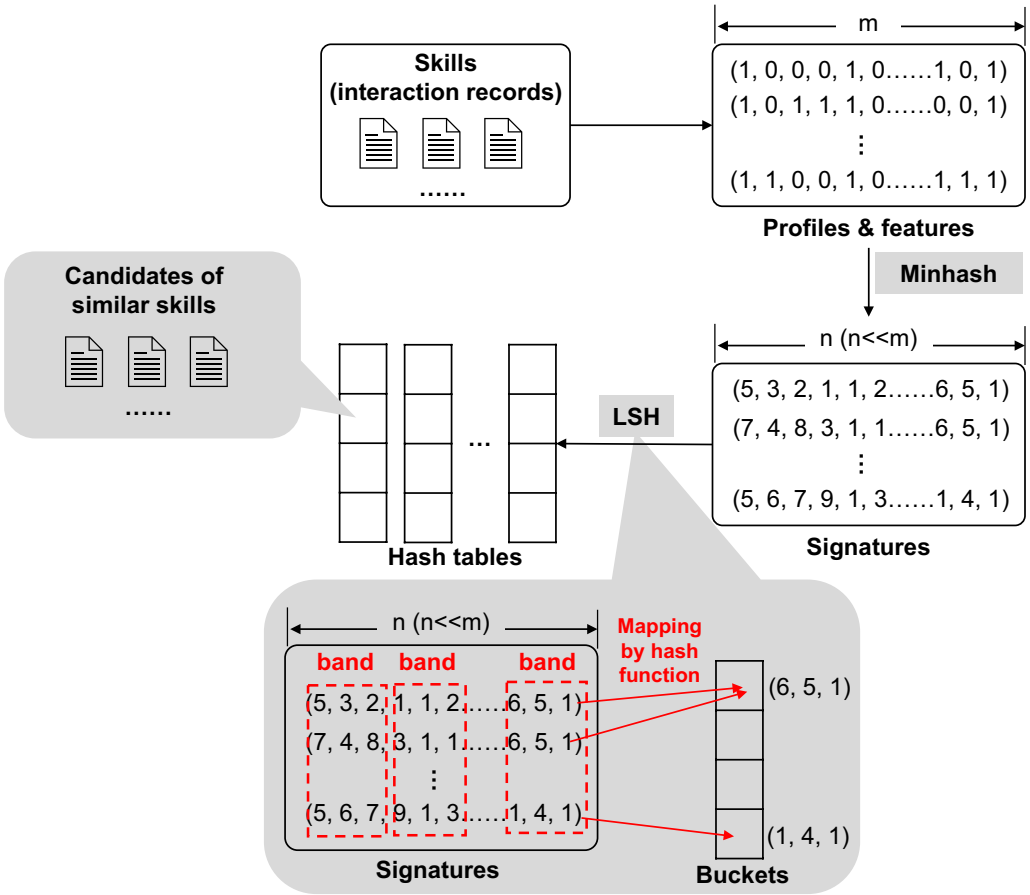


Fig. 4 The framework of Minhash and LSH

the natural language content in the skill interaction record into tree structures to realize structure similarity calculation.

Tree structure similarity. A skill interaction record contains multiple interaction paths, similar to execution paths in a program. As shown in Fig. 5, the root node $N1$ contains the user's wake-up statement and output returned by the skill for the first time. Each child node contains the new input generated based on the skill output in the parent node, and the output returned by the skill in response to this new input. For example, node $N2\_2$ contains an input generated from the output of node $N1$, and an output returned from the skill after the user sends the input in $N2\_2$.

When the first nodes of several paths are consistent, these paths can be integrated, and the same child nodes can also be merged. Finally, the interaction records can be abstracted into trees. At this time, the structure of a skill can be reflected in a tree structure, and the skill structure similarity can be calculated through tree structure similarity. It is important to note that the order of tree nodes in the same layer is ignored when calculating similarities. It depends on the order in which users generate their answers and the order in which they select them. Therefore, the order is not controllable.

SkillSim disregards the order of interaction trees, *i.e.* does not care about the order of sibling nodes. SkillSim mainly focuses on the hierarchical structure, extracting four features of each node in a tree. The feature d represents the depth of the current node. The feature h represents the distance from this node to the longest path leaf node, that is, the height of the current node. And the feature o represents the node's out degree, that is, the number of answers that can be generated by the output in this node. To compare the similarity of tree structures more accurately, we assign a value to each node, which is the question type of the output in that node. We use the feature c to represent it. As shown in Table 1, the question types are divided into 5 types. If the output does not belong to any of these categories, we will assign the node a value of *"none"*.

As a result, SkillSim extracts a 4-tuple (d, h, o, c) from each node as its feature. In particular, SkillSim eliminates

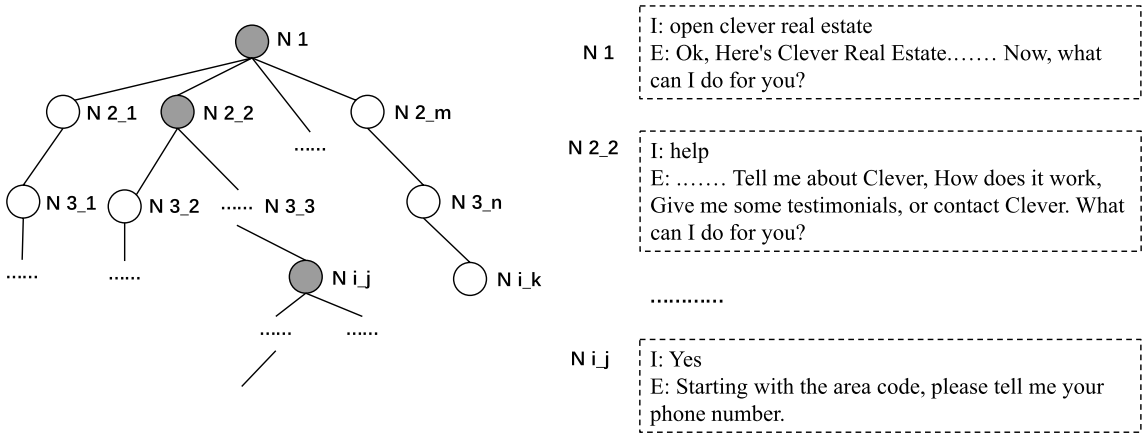


Fig. 5 The interaction tree of the skill "Clever Real Estate"

the distraction of the different ways that the user uses the wake-up sentence. For example, one way is to enable the skill and input the desired function after the skill response, while the other way is to start the skill with the desired function (such as *"Alexa, play the album from Barbie De Facto Insider"*). SkillSim eliminates this interference by comparing the similarity of subtrees. Through the above methods, SkillSim abstracts a skill to a set of tuples. It then calculates the Jaccard similarity of the two skills.

Implementation

Dataset. SkillSim aims to detect skill similarity, and interaction with skills is not the focus of this paper. Therefore, we request a part of the experimental data from SkillExplorer (Guo et al. 2020) as our dataset. The dataset includes interaction records of 30,000 skills, and basic information of 68,066 skills crawled from the Amazon skill store, as shown in Table 2.

Text similarity. SkillSim first pre-processes interaction content. For the overall text, SkillSim only performs simple processing, such as converting words to lowercase, deleting punctuation marks, *etc.* Then, SkillSim splits the content with 3-shingles as granularity. Minhash and LSH algorithms are implemented using datasketch (2022) so that the overall text can be pre-filtered. Here we set the number of random permutation functions in Minhash to the default value 128. We get the candidate similar skill pairs with approximate Jaccard similarity greater than 10% based on LSH. SkillSim performs pairwise Jaccard similarity calculation based on pre-filtering results. Then, SkillSim extracts the key text of skills from the overall text. It uses NLTK (2022) and spaCy (2022) to perform more detailed preprocessing on the overall text, including removing stopwords, and recovering word stems, *etc.* Finally, SkillSim calculates IDF value and key text similarity with these words. The settings are shown in Table 3.

Table 2 The number of skills in different categories

Skill type	Total	With records
Business and finance	3,336	1,420
Connected car	115	74
Education and reference	6,422	3,296
Enterprise	4	2
Food and drink	1,336	1,008
Games and trivia	11,413	7,182
Kids	2,684	423
Lifestyle	10,405	3,816
Local	1,223	324
Movies and tv	869	427
Music and audio	8,743	3,194
News	6,394	854
Novelty and humor	3,360	2,154
Productivity	3,737	2,019
Shopping	283	214
Smart Home	2204	626
Social	1,224	549
Sports	1,516	292
Travel and transportation	1,161	880
Utilities	803	570
Weather	834	676
Total	68,066	30,000

SkillSim employs the response content of skills as the text comparison object. This is because the input of the skills generated by SkillExplorer is already reflected in the comparison object, as it originates from the output of the previous round. To confirm this, we randomly choose 1,000 skill pairs and recompute their overall text similarity and key text similarity by incorporating the inputs

Table 3 Similarity calculation configuration

Similarity	Granularity	Feature expressions
Overall text	3-shingles	3-shingles
Key text	Word	IDF
Tree structure	Node	Tree feature tuples

of the skills. The average difference in similarity values, when comparing the results with and without including the inputs, is 0.46%. Thus, not including the inputs does not have a significant impact on the results.

Structure similarity. SkillSim analysis interaction records to build trees. According to the method of classifying question types described in SkillExplorer (Guo et al. 2020), SkillSim uses spaCy to obtain question types and constructs 4-tuple for each node. In addition, SkillSim extracts subtrees (height > 2) of the tree and changes the corresponding depth. Each interaction tree is stored as a sequence of tuples. It should be mentioned that SkillSim only focuses on the presence or absence of nodes but not on the order of the nodes when calculating the similarity. Also, SkillSim uses pre-filtering method to exclude skill pairs that are completely irrelevant or have little similarity (< 10%), and calculates the true similarity based on the filtering results.

Evaluation

In this section, we aim to answer the following research questions.

- *RQ1: What characteristics do the skills exhibit at a large scale?*
- *RQ2: What can be concluded from similarity analysis?*
- *RQ3: How effective is SkillSim in detecting similar skills?*

Landscape

To answer RQ1, we evaluate 68,066 skills. Out of these skills, we obtain 23,352 developer names, with an average of 2.9 skills per developer. Among them, 42.65% of developers have at least two skills.[3] We construct a Cumulative Distribution Function (CDF) graph to show the number of skills each developer possesses. As shown in Fig. 6, the X-axis represents the number of skills for a given developer name. 57.35% of developers possess one skill, and

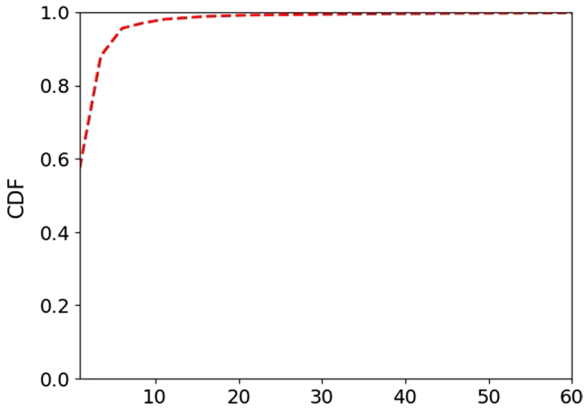


Fig. 6 The number of skills owned by a developer

less than 0.4% of developers have more than 54 skills. 49 developers have over 100 skills.

Three developers possess over 1,000 skills, they are *InfoByVoice (with more than 2,500 skills)*, *Rhall (with more than 1,400 skills)* and *Patch.com (with more than 1,000 skills)*. We analyze these three developers further. *InfoByVoice* and *Rhall* both refer to *voiceapps.com*, a platformwhich helps non-technical individuals to develop their skills. Its homepage claims *"Building complex skills is easy with Voice Apps....."*. Although both developers come from the same platform, the skills developed by *InfoByVoice* are mainly in the *Lifestyle* category, while the skills developed by *Rhall* are mainly in the *Games & Trivia* category, with very few in *Novelty and Humor* and *Education & Reference*. Moreover, *Patch.com* points to the website *Patch.com*, an advertising promotion site that helps companies place ads. Most of the skills developed by this developer belong to the *News* category. All three developers belong to advertising or assistant development platforms.

Answer to RQ1: Many developers are associated with multiple skills, with 42.65% of them having more than one skill. Some developers or development platforms even have thousands of skills.

Similarity analysis

To answer RQ2, we analyze the results of the similarity calculation. This section presents an in-depth analysis from two different perspectives: the similarities among skills and the developers associated with those skills. Based on the pre-filtering results, we obtain skill pairs with similarity greater than 10%.

Similar skills. We analyze the similarity results from various perspectives. Figure 7 depicts the CDF of similarities distribution, where the X-axis represents the similarity values and the Y-axis displays

[3] Note that the developer names are not unique, so the same name does not represent the same developer. Here is just an analysis of the basic information, and we will analyze further later based on the skill interaction content.

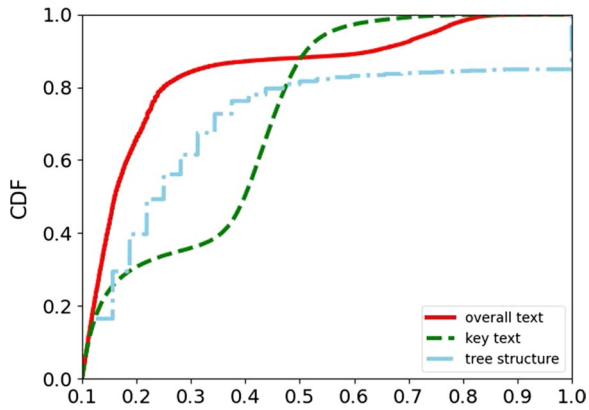


Fig. 7 Similarity probability distribution

the percentage of values that are less than the given similarity.

For the overall text similarity, the percentage of skill pairs with similarity greater than 10% is 2.3%, and we perform analysis based on these skill pairs. Of these data, as shown by the red line, 11.9% of the skill pairs are more than 50% similar, while 7.5% of the skill pairs are more than 70% similar. 0.1% of the skill pairs have a similarity greater than 90%. We carry out manual analysis on a total of 900 skill pairs by selecting 100 skill pairs for each varying degree of similarity (*i.e.* from 10% to 90%). Skill pairs with a similarity greater than 70% typically have almost identical content. This is consistent with previous works (Manaa and Abdulameer, 2018). These skill pairs may only differ in the name or have slight modifications to their content. The first four rows of Table 4 are two skill pairs. The skill *"SingleStone"* and the skill *"SolitaryStone"* are both advertising *"singlestone"* although they come from different developers *"SingleStone Consulting"* and *"Pepper Industries"*. Except for the different skill names in the first sentence, the rest content is exactly the same. The skills in the second pair come from the same developer, and they have roughly the same content except for the skill names. Most of the skills with a similarity greater than 50% use identical templates with only minor modifications in content. For example, a skill named *"Harvest Christian Fellowship"* is developed by *"InfoByVoice"*, and the skill *"Wave Church Seaboard, Virginia Beach, VA"* developed by *"SkillSet"* have an overall text similarity of 58.1%. Although they present information about two different churches, they use the same questioning phrase and the same contact information. The skill pairs with more than 30% overall text similarity are some based on the same template but with significant differences in content, and others due to the common phrases in short conversations, such as *"for help please visit help pages on amazon web site"*.

For key text similarity, 4.1% of the skill pairs have a similarity greater than 10%. In these data, as shown by the green line, 12.4% of skill pairs are more than 50% similar and the percentage of skill pairs that are more than 70% similar is 7.8%. In addition, Furthermore, 0.05% of skill

Table 4 Examples of similar skill pairs

Skill name	Developer	Contents	O_{sim}	K_{sim}
SingleStone	SingleStone Consulting	"Ok here is singlestone. singlestone is a consulting firm that focuses on reducing friction and removing barriers in business. it has expertise in customer experience product development and internal collaboration. singlestone is located at 4101 cox road suite 350 glen allen virginia 23060 ....."	95.0%	91.1%
SolitaryStone	Pepper Industries	"Ok here is solitarystone. singlestone is a consulting firm that focuses on reducing friction and removing barriers in business. it has expertise in customer experience product development and internal collaboration. singlestone is located at 4101 cox road suite 350 glen allen virginia 23060 ....."		
Find Seafood Specials	Black Point Lobster	"Ok here is find seafood specials. are you cooking whole live lobsters or lobster tails. would you like to grill boil bake or steam your lobster tails ..... remove from freezer making sure it is no longer moving. push the tip of a skewer or large sharp heavy knife into the center of the cross on its head"	73.9%	81.1%
How to Eat Lobster Tail	Black Point Lobster	"Ok here is how to eat lobster tail. are you cooking whole live lobsters or lobster tails. would you like to grill boil bake or steam your lobster tails ..... remove from freezer making sure it is no longer moving. push the tip of a skewer or large sharp heavy knife into the center of the cross on its head"		
Fire Fact Number One Card	Rhall	".....card with fact is being sent to your alexa app. this app sends a card to your alexa app. this card will give you a fact about fire. this skill was built with love by voiceappscom ....."	40.0%	71.83%
Wolf Facts	SJ	".....card with link is being sent to your alexa app. this app gives facts about wolves. open it and it will randomly give you a fact about wolves. this skill was built with love by voiceappscom ....."		

pairs have a similarity greater than 90%. Here we focus on the difference between key text similarity and overall text similarity. As shown in the last two rows of Table 4, the skills *Fire Fact Number One Card* and *Wolf Facts* have different developer names, and the overall text similarity is 40%. It is difficult to distinguish whether they are related just from the overall text similarity. However, key text similarity captures important information *voiceapp.com*, with a similarity of 71.83%. Thus, by combining the key text similarity with overall text similarity, we can effectively identify skills on the same topic or important key items.

For the tree structure, we filter out skills with less than 2 responses (about 1/3), most of which are simple ad skills or skills related to news. We then calculate the tree structure similarity. The percentage of skills with more than 10% similarity in the skill pairs is 1.33%. Among these skill pairs, as shown in the blue line of the figure, 18.4% of the skill pairs have more than 50% similarity, 16.7% have more than 70% similarity, and 15.1% have more than 90% similarity. We further combine the tree structure similarity with the overall text similarity for an in-depth analysis. The results show that for skill pairs with overall text similarity greater than 90%, 71.3% of the tree structure similarity is greater than 70%. For skill pairs with overall text similarity over 70%, 48% of skill pairs display a tree structure similarity of more than 70%. We find that the order in which the skill commands are used in the interaction affects the skill response content (*e.g.* different responses for the first and second time when opening skill). Besides, the mechanisms of SkillExplorer (Guo et al. 2020) for answering questions (*e.g.* ending the path if a response is visited) also affect the tree structure similarity. These two reasons decrease the tree structure similarity value. We randomly select 500 skill pairs with overall text similarity over 70%. After manually and completely traversing the skills in a certain order, SkillSim automatically draws the tree structure and calculates the similarity. We find that the percentage of skill pairs with tree structure similarity over 90% is 93.5%, and the percentage of skill pairs with more than 70% similarity is 89.2%. This indicates that the tree structure similarity can be used to detect templates.

In summary, the three similarity measures evaluate skill similarity from distinct perspectives, and they can be utilized in combination as required.

We also analyze the number of similar skills. We calculate the percentage of skills with skill similarity greater than 90%, 80%, and 70% based on overall text similarity, key text similarity, and tree structure similarity, as shown in Table 5. As many as 25.9% of the skills have skill pairs with an overall text similarity greater than 70%. 16.4% of the skills have skill pairs with a key text similarity greater than 70%. And for the tree structure, 36% of the skills have other skills with similarity greater than 90%. Then we combine the overall text similarity and key text similarity for a more rigorous analysis. The result shows that 11.1% of the skills have both similarities greater than 70%. This percentage is surprising because, as analyzed above, a 70% similarity largely means that the template and content of these skills are basically the same.

Table 5 Proportion of similar skills

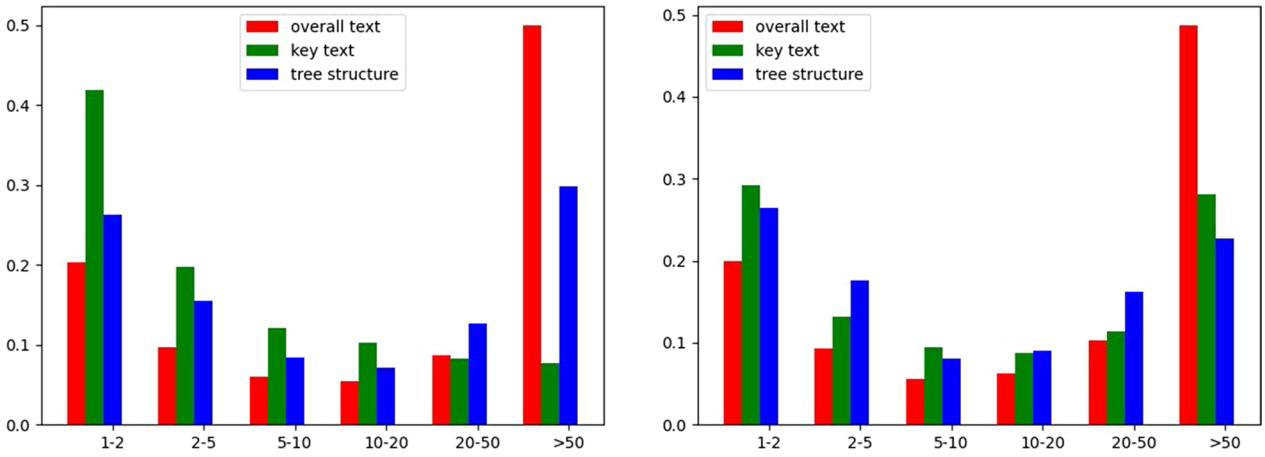
Similarity perspectives	≥ 90%	≥ 80%	≥ 70%
Overall text	9.6%	21.0%	25.9%
Key text	5.5%	11.2%	16.4%
Tree text	36.0%	37.9%	40.1%
Overall text + Key text	4.6%	8.6%	11.1%

In order to know how many similar skills there are for each skill, we count the data based on different similarity perspectives under different similarity values (70% and 50%). As shown in Fig. 8, the two sub-figures have roughly the same trend, mainly distributed in $1-2$ and >50. Among the skills with similarity greater than 70%, nearly 50% of them have more than 50 other skills with similar overall text, and the tree structure has a tendency to be consistent with the overall text. This phenomenon indicates that many similar skills use the same development template.

Developers. In landscape, we count developer names based on the skill's basic information. Here, we combine developers and skill similarity for further analysis. We get 14,211 developer names in the 30,000 skills. Then we analyze similar skills under the same developer name and similar skills among different developer names.

For the same developer name, we count a total of 3,516 developer names corresponding to more than 1 skill. We group the skills corresponding to each developer name. If the number of categories after clustering is fewer than the number of skills, we consider it to be *changed*. Table 6 shows that the more skills a developer name possesses, the more likely it is to be changed. When clustering based on a text similarity greater than 70%, 27.2% of developer names with 5 or more skills changed. This value reaches 44.5% when based on an overall similarity of greater than 50%. This makes sense because when developers need to develop multiple skills, using the original template or content can reduce the workload.

Among these developer names, *"InfoByVoice"* has 2,412 skills, the largest number, followed by developer *"Rhall"* with 1,232, which is consistent with the result obtained from the previous basic information. We further analyze the content of skills corresponding to the same developer



(a) Greater than 70% similarity (b) Greater than 50% similarity

Fig. 8 Similar skill numbers

Table 6 Skill clustering *changed* based on the same developer name

# of developer's skills	Similarity value	Overall text (%)	Key text (%)	Tree structure (%)
≥ 2	0.9	2.6	2.8	10.1
≥ 3		4.4	4.7	17.1
≥ 5		8.7	8.6	28.1
≥ 2	0.7	9.4	7.3	11.0
≥ 3		15.6	12.1	18.5
≥ 5		27.2	23.1	30.4
≥ 2	0.5	17.1	13.2	12.8
≥ 3		27.4	21.0	21.2
≥ 5		44.5	35.4	34.3

name. We perform similarity clustering for skills with *"InfoByVoice"* developer name. The 2,412 skills are clustered into 2,089 categories based on an overall text similarity of 90%. In addition, these skills are clustered into 79 categories based on an 80% similarity and are clustered into 19 categories based on a 70% similarity. This suggests that *"InfoByVoice"* uses very similar templates and contents to advertise different topics, which is confirmed manually. Therefore, there are indeed cases in which the same developer or development platform develops a large number of skills with similar content.

For different developers, we obtain skill pairs with overall text similarity and key text similarity both greater than 50%, and cluster 30,000 skills into 21,754 categories. The largest category encompasses more than 2,500 skills. After conducting manual analysis, three main cases are identified. In the first case, these similar skills are developed with the skill assistant development platforms. They include official assistant platforms such as *"Blueprints"*, and third-party assistant development platforms such as *"VoiceApps"*. Among the 30,000 skills, there are more than 900 skills from *"Blueprints"* that have obvious hints. For the third-party platforms, we analyze the categories with more than 50 skills and get a total of 16 third-party platforms. Using these third-party platforms, skills can be published either through the platform account or developers' own accounts. There are some identical elements in these skills, such as the same statements (*e.g. "ok here is... get standings and records for the..."*) or information about the platform (*e.g.* email and phone number). These 16 third-party platforms can associate more than 9,000 skills and more than 200 developer names. The number of skills is more than 30% of the total number of skills we analyze. In the second case, the developer is an organization that develops multiple similar skills either by itself or through third-party platforms. For example, *"SnoCountry"* develops more than 400 similar skills for broadcasting detailed ski and snow condition reports and resort information. The last case is skills that are suspected to be developed by the same developer using different accounts, such as skills *"SingleStone"* and *"Pepper Industries"* in Table 4.

Non-compliant behaviors. During the analysis, we find some non-compliant behaviors. For example, Amazon requires developers cannot explicitly request that users leave a positive rating of the skill (Amazon 2022c). However, up to 2,090 skills in several cluster categories from third-party developer platforms like *"getstoryline"*, *"VoiceSkillsInc"*, *"Appbly.com"* all explicitly mention *"please leave a 5 star review"* or *"give us a 5 star rating"* , *etc.* We further analyze the issues related to user privacy. Amazon requires developers to include a privacy policy link and configure permissions when they need

user privacy to enhance services. However, *Witlingo*, a third-party assistant development platform with over 100 skills (in 30,000), develops 51 similar skills related to news and asks users for their cell phone numbers through conversations without configuring the permissions. This behavior facilitates the skill to bypass platform control and access user privacy. The platform is even recommended by Amazon in the third-party tools list (Amazon 2022d). *"Voiceter Pro Inc"*, also a third-party development platform, develops 36 similar skills related to real estate, asking users for zip codes and addresses without configuring permissions. In addition, in the same cluster category as *'Voiceter Pro Inc'*, there is also a developer named *'Voiceter Pro LLC'* who develops similar skills that asks for user address and does not configure permissions.

Findings. Through analysis, we find that although Amazon claims 100,000 skills, a very large number of skills are similar in content (25.9% of the 30,000 skills have an overall text similarity of more than 70%). Moreover, the existence of multiple skill assistant development platforms lowers the threshold of skill development and limits the richness of skill content. Many low-quality skills are present in the market. Nearly 1/3 of the skills have a tree structure depth of less than 3, and most of the skills are advertisements. 15.1% of the tree structures are more than 90% similar. More importantly, many skills are developed with the assistance of third-party development platforms, some of which do not fully comply with Amazon's certification policies, even though some are recommended by Amazon itself. Similarity analysis assists in identifying other non-compliant skills that are similar and in establishing correlations among different accounts.

Answer to RQ2: There are many similar skills in the skill market and development platforms dominate a large number of similar skills. Through similarity analysis, we discover that some platforms develop multiple similar non-compliant skills. Additionally, even platforms that are recommended by Amazon develop skills that request users' private information in a non-compliant manner. Furthermore, there are instances of developers creating non-compliant skills across accounts.

Effectiveness

To answer RQ3, we evaluate the effectiveness of pre-filtering methods and similarity calculation methods.

Pre-filtering methods. We try a pairwise comparison. It takes approximately 35 min to perform one million overall text similarity comparisons. For 30,000 skills, pairwise comparisons will require roughly 20 days. Using Minhash and LSH algorithms, it only takes less than 10 min to obtain all skill pairs with similarity greater than 10%. The pre-filtering method can filter out more than 90% of irrelevant results, thus significantly reducing time consumption.

In addition to time performance, we are also concerned with the accuracy of the pre-filtering method. We get the skill pairs with overall text approximate similarity from similarity greater than 10% to similarity greater than 90% based on Minhash and LSH algorithms. Then we calculate the true similarity based on the pre-filtering results. The results are shown in Fig. 9. The X-axis is the pre-filtered threshold and the Y-axis is the true similarity. These rectangles show the true similarity based on the pre-filtered thresholds. The top and bottom of the rectangle represent 75% and 25% of the data distribution. The black line below the bottom of the rectangle is the data starting point. The red line is the median, the green triangle is the mean, and the violin plot on the right indicates the data density distribution. As we can see from the figure, more than 85% of the data is accurate, and false similarity results (around 10%) are tolerable because we will further calculate the true similarity based on pre-filtering data and false positives will be filtered. We are more concerned with false negatives than accuracy, as this will cause us to miss them in the final results. Here we adopt two methods to test the false negatives. For the first method, we randomly select 50,000 pairs that are not in the pre-filtered results and then compare them with pre-filtered pairs based on a threshold of 0.1. The result shows that the false negative rate is less than 0.3%. For the second method, we check false negatives based on the pre-filtered result. Pairs with true similarity greater than 90% in threshold 0.1 and threshold 0.9 are compared to see if there are any false negatives. The logic behind this method is that all pairs with similarity greater than 90% have a high probability of existing in threshold 0.1, but some may be missed in threshold 0.9. Therefore, by comparing the results of pairs in these two thresholds, we can check the false negatives. We compare with each threshold in turn and the false negative rate is lower than 0.07%.

Similarity calculation methods. In the above analysis, we demonstrate the effectiveness of SkillSim through manual analysis and case studies. In this part, we try to use learning-based methods like Word2vec (Mikolov et al. 2013) and Doc2vec (Le and Mikolov 2014) to calculate skill similarity and analyze their performance.

Word2vec, a word embedding methodology, that enables similar words to have similar dimensions. We use (Gensim 2022) to train word vectors with 10,000 wiki news and 5000 skill contents. Then we average each word vector in a document to represent the document vector. Finally, we calculate the cosine similarity

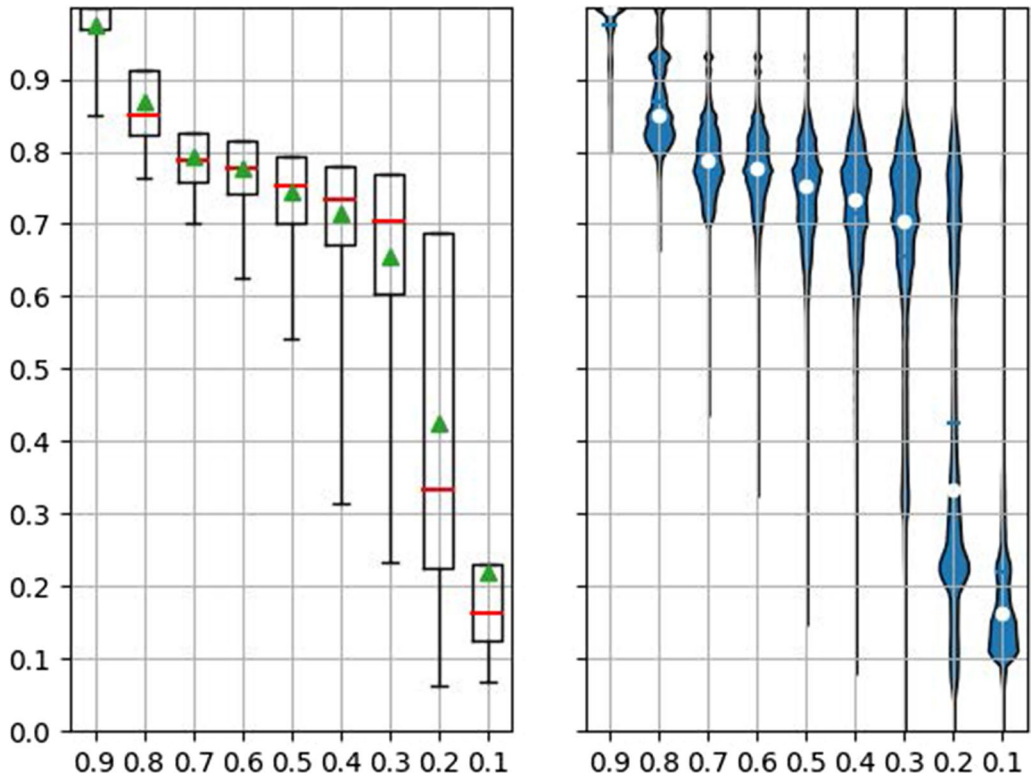


Fig. 9 Effectiveness of pre-filtering method based on overall-text

Table 7 Other Similarity Method

Feature expressions	Training dataset	Test dataset	Accuracy (%)
word2vec	wiki news + skill contents	wiki news	91.44
		skill contents	2.6
doc2vec	text8 + skill contents	text8	100
		skill contents	62.2

between skills in the testing datasets. Due to the lack of ground truth, we validate the model accuracy by checking whether the highest similar text is the document itself. In the wiki news test set, the model gets an accuracy of 91.44%. However, in skill's documents, only 2.6% are correctly matched, and the incorrectly matched documents are even less than 20% similar in the overall text similarity with Jaccard (Table 7).

Considering that word vectors do not work well for document comparison, we try to use Doc2vec, which can represent each document as a Vector. We train the model using text8 documents provided by Gensim and 5000 skill documents in our dataset. In the test set of text8 documents, 100% of the documents with the highest similarity are matched with themselves. For the skills in the test set, 62.2% of the original documents are successfully matched. Due to sensitivity to semantics, Doc2vec considers many similar functional skills to be similar.

We randomly select 100 skill pairs with overall text similarity and key text similarity ranging between 10% and 30% and then calculate their doc2vec similarity. Of the pairs, 58% have semantic similarity higher than 50%, and 18% have similarity higher than 70%. We manually analyze 18 similar pairs and find that 83.3% of the skill pairs come from skills with similar behaviors that are developed by different developers. Based on the available information, we are unable to determine their connection. The other 16.7% have some common phrases. An example is shown in Table 8.

The reason for this outcome is that the majority of the functions in the skill market are centered around

Table 8 An example of false positives with Doc2vec

Skill name	Developer	Contents	O_{sim}	K_{sim}	$Doc2vec_{sim}$
DogeCoin	Sterian Associates	"Ok here is dogecoin. the current price of doge coin is 18139999999999997 cents us per one thousand doge coin.....get the spot price of dogecoin one of the top crypto currencies worldwidesimply ask alexa start dogecoin....."	14.1%	12.3%	89.1%
Monero Price	Joseph Yi	"Ok here is monero price. welcome to the monero price checker. to ask for the price of monero please say alexa what is the price of monero. the current price of monero in usd is 666....."			

functions such as news, games, life tips, advertising, music, and smart homes. Furthermore, the interaction mode based on the question-answering system results in skills having similar phrases. As a result, the semantics among these skills are largely indistinguishable. Models based on semantic similar detection are effective for traditional datasets, however, they do not fulfill our purpose in the detection of skill similarity.

Skills based on semantic similarity will bring more confusion when analyzing associations among skills. However, they offer an additional perspective for observing skills. Thus, we show the analysis of semantic similarity. The percentage of skill pairs with similarity greater than 10% is lower than 5% of the total skill pairs in overall text, key text, and tree structure. But the percentage of skill pairs with semantic similarity greater than 10% is as high as 55%. Considering the basis of 30,000 skills, this has nearly 500 million skill pairs. Among them, 11% of skill pairs have semantic similarity greater than 50%, and 3% have similarity greater than 70%. Additionally, we separately count the distribution of semantically similar skills under different function categories. Semantically similar skill pairs account for a higher percentage of the same category. For example, more than 16% of skill pairs have semantic similarity greater than 70% in *Weather* category.

Answer to RQ3: The pre-filtering method based on Minhash and LSH greatly reduces time consumption. Compared with semantic-based (*e.g.* Word2vec, Doc2vec) similarity calculation, duplication-based (*i.e.* SkillSim) similarity calculation is more suitable for finding relationships among skills.

Discussion

In this paper, we perform a correlation analysis of skills in the market through similarity analysis. We find a large number of skills with similar content in the market and summarize three reasons. One of the main reasons is the assistance development platforms. On the one hand, these development platforms lower the threshold for skill development, allowing individuals without a foundation to develop skills, consequently significantly increasing the number of skills. On the other hand, it also leads to the emergence of many skills with low quality and similar content.

In addition, there are some third-party assistant development platforms that do not fully comply with the market policies and develop similar non-compliant skills. In this way, third-party development platforms can collect a lot of private information from users without attracting the attention of platform reviewers. We propose some suggestions for improvement. First, until there is a robust review mechanism, it is necessary to further optimize the official development tool so that developers tend to use it. Second, there needs to be a strict review of the recommended third-party development tools to ensure that they have sufficient knowledge of the certification requirements. Third, stricter restrictions on developer account creation are needed. Only email verification provides malicious developers with more opportunities to access users' private information. Finally, the similar skills of different accounts need to be further analyzed.

Limitations and future work. Due to the reason of the dataset, SkillExplorer does not select answers in a specific order when traversing the skill's behavior. Meanwhile, if the same output appears during traversal, the current interaction path will be ended. These reasons make it possible for skills with the same logical structure to have different structure trees. Although SkillSim mitigates these problems by ignoring the order of tree nodes and comparing subtree structures, there are still false negatives. In the future, we plan to design a question-answering system that generates answers to questions based on a certain sequence to solve this problem.

Moreover, if only the words of a skill change significantly while the topic semantics remain similar, it will be difficult for methods based on n-shingles and IDF to

capture the similarity. Using a large NLP model to capture the "key semantic" similarity among skills may be a solution. Considering the lack of the marked dataset, we will take it as future research work to detect the key semantic similarity.

Related work

Skill analysis. In recent years, with the rapid growth in the number of skills, there is a growing concern about the safety of skills. Kumar et al. (2018) discover skill squatting, a homo-phonic attack that exploits speech interpretation errors. It can divert user requests to malicious skills by creating skills with similar names to benign ones (like "Full Moon" vs. "Four Moon"). Zhang et al. (2019a) further discover a similar attack by exploiting the longest string match used by the VPA to invoke a skill. In addition, they discover voice masquerading, *i.e.* malicious skills can mimic exit intent, tricking users into believing the skill has terminated, while still collecting users' voice inputs. LipFuzzer (Zhang et al. 2019b) exploits vulnerabilities in NLU's intent classifier to generate voice commands that may lead to semantic inconsistencies, allowing the classifier to misinterpret the user's request and route the request to a malicious skill. It systematically identifies voice commands that are easily misinterpreted in existing VPA platforms. These three researches (Kumar et al. 2018; Zhang et al. 2019a, b) mainly focus on issues in skills wake-up and exit processes.

Cheng et al. (2020) conduct a comprehensive measurement of the trustworthiness of skill certifications in popular skill platforms. They find that these platforms are unreliable by successfully obtaining 234 (100%) skill certifications for policy violations. Wang et al. (2021) investigate the vetting process of two VPA platforms with elaborate skills and identify weaknesses in the vetting process. They propose three attacks that help malicious skills successfully bypass the vetting process and design several defenses based on linguistic knowledge. These two works illustrate the flaws in the vetting process of mainstream VPA platforms. SkillExplorer (Guo et al. 2020) tests and analyzes the interactive content of skills, classifies the content of skills through NLP technology, and generates potential commands. It performs DFS-based exploration of skills to detect skills that do not comply with privacy rules. SkillVet (Edu et al. 2021) analyzes the traceability of permissions and finds that many skills do not fully disclose their data usage. It reveals how skills can bypass Alexa's permission system by requesting personal information without using its API. SkillBot (Le et al. 2022) focuses on kid skills and identifies kid skills with inappropriate content or personal data requirements. It describes a confounding utterance threat that can accidentally shift invoke for children's skills to non-child-directed skills. Many studies (Guo et al. 2020; Lentzsch et al. 2021; Liao et al. 2020) find that skills do not provide a valid privacy policy. In contrast to these efforts to study separate skills, our work focuses on discovering associations among skills through similarity analysis.

Software clone detection Software clone detection is a well-developed research topic, and the main branch is code clone detection. Researchers perform clone detection based on the code's different representations. Text-based clone detection techniques (Ragkhitwetsagul and Krinke 2017; Yu et al. 2017; Nakamura et al. 2016; Xue et al. 2018) can achieve high accuracy with few false positive rates. However, it ignores information such as the code's syntax, which can lead to a large number of false negatives. Token-based detection techniques (Tekchandani et al. 2017; Wang et al. 2018; Yuki et al. 2017; Sajnani et al. 2016) divide the code into token sequences, which can match code-specific information. However, this approach has a low tolerance for code changes and does not make use of code structure information. A similar problem occurs if only text or token information is considered in skill similarity detection. This is because skills are programs and possess structural information. Structure-based clone detection (Yang et al. 2018; Pati et al. 2017; Chen et al. 2014) takes into account the structural features of the code. It is less sensitive to changes in the order of the code, so it can also detect slightly modified code clones. However, it does not recognize tokens and text values and has more false positives. Hybrid clone detection techniques (Misu and Sakib 2017; Sheneamer et al. 2016; Vislavski et al. 2018; Misu et al. 2017; Akram et al. 2018; Sheneamer et al. 2018; Meng et al. 2018a, b) can complement each other to achieve a better result for clone detection. Unlike the above works, skill similarity detection is not based on code, but on the content of interactions in natural language. Therefore, previous similarity detection work is not applicable.

Conclusion

In this paper, we develop a method called SkillSim to detect skill similarity from two dimensions of text similarity and structure similarity, and further analyze the association of skills in the market.

We find that more than 25.9% of skills have at least one skill with overall text similarity greater than 70%. This phenomenon is mainly caused by skill assistant development platforms, while some platforms even have security risks. Based on the above phenomena, we propose some suggestions.

Acknowledgements
We would like to thank the anonymous reviewers for their detailed comments and useful feedback.

Author Contributions
ZG: investigation, conceptualization, methodology, materials, writing, editing, experiment, validation, review, resources. RL: resources, discussion, experiment, review. GM: resources, discussion, experiment, review. KC: resources, discussion, experiment, review. All authors have contributed to this manuscript and approve of this submission.

Funding
NSFC (92270204), Beijing Natural Science Foundation (No.M22004), Beijing Nova program.

Availability of data and materials
Not applicable.

Declarations

Competing interests
The authors declare that they have no competing interests.

Received: 27 December 2022 Accepted: 28 February 2023
Published online: 01 July 2023

References

Ain QU, Butt WH, Anwar MW, Azam F, Maqbool B (2019) A systematic review on code clone detection. IEEE Access 7:86121–86144. https://doi.org/10.1109/ACCESS.2019.2918202

Akram J, Shi Z, Mumtaz M, Luo P (2018) Droidcc: A scalable clone detection approach for android applications to detect similarity at source code level. In: Reisman S, Ahamed SI, Demartini C, Conte TM, Liu L, Claycomb WR, Nakamura M, Tovar E, Cimato S, Lung C, Takakura H, Yang J, Akiyama T, Zhang Z, Hasan K (eds.) 2018 IEEE 42nd Annual Computer Software and Applications Conference, COMPSAC 2018, Tokyo, Japan, 23-27 July 2018, Volume 1, pp. 100–105. https://doi.org/10.1109/COMPSAC.2018.00021

Amazon Skill Numbers (2019). https://voicebot.ai/2019/10/01/amazon-alexa-has-100k-skills-but-momentum-slows-globally-here-is-the-breakdown-by-country

Amazon: Amazon Certification Requirements (2022a). https://developer.amazon.com/docs/custom-skills/certification-requirements-for-custom-skills.html

Amazon: Amazon Certification Requirements (2022b). https://developer.amazon.com/zh/docs/custom-skills/request-customer-contact-information-for-use-in-your-skill.html

Amazon: Amazon Certification Requirements (2022c). https://developer.amazon.com/en-US/docs/alexa/devconsole/about-skill-metrics.html

Amazon: Third-party tools list recommended by Amazon (2022d). https://developer.amazon.com/en-US/alexa/alexa-skills-kit/get-deeper/dev-tools-skill-management-api/tools-design

Bellon S, Koschke R, Antoniol G, Krinke J, Merlo E (2007) Comparison and evaluation of clone detection tools. IEEE Trans. Softw. Eng. 33(9):577–591. https://doi.org/10.1109/TSE.2007.70725

Broder A.Z (1997) On the resemblance and containment of documents. In: Carpentieri B, Santis AD, Vaccaro U, Storer JA (eds.) Compression and Complexity of SEQUENCES 1997, Positano, Amalfitan Coast, Salerno, Italy, June 11-13, 1997, Proceedings, pp 21–29. https://doi.org/10.1109/SEQUEN.1997.666900

Cheng L, Wilson C, Liao S, Young J, Dong D, Hu H (2020) Dangerous skills got certified: measuring the trustworthiness of skill certification in voice personal assistant platforms. In: Ligatti J, Ou X, Katz J, Vigna G (eds.) CCS '20: 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, USA, November 9-13, 2020, pp. 1699–1716. https://doi.org/10.1145/3372297.3423339

Chen K, Liu P, Zhang Y (2014) Achieving accuracy and scalability simultaneously in detecting application clones on android markets. In: Jalote P, Briand LC, van der Hoek A (eds.) 36th International Conference on Software Engineering, ICSE '14, Hyderabad, India - May 31 - June 07, 2014, pp. 175–186. https://doi.org/10.1145/2568225.2568286

Chen Y, Yuan X, Zhang J, Zhao Y, Zhang S, Chen K, Wang X (2020) Devil's whisper: a general approach for physical adversarial attacks against commercial black-box speech recognition devices. In: Capkun, S., Roesner, F. (eds.) 29th USENIX Security Symposium, USENIX Security 2020, August 12-14, 2020, pp. 2667–2684 . https://www.usenix.org/conference/usenixsecurity20/presentation/chen-yuxuan

Datasketch (2022). https://github.com/ekzhu/datasketch

Edu J, Ferrer Aran X, Such J, Suarez-Tangil G (2021) Skillvet: automated traceability analysis of amazon alexa skills. IEEE Trans Dependable Secure Comput. https://doi.org/10.1109/TDSC.2021.3129116

Gensim (2022). https://radimrehurek.com/gensim

Guo Z, Lin Z, Li P, Chen K (2020) Skillexplorer: Understanding the behavior of skills in large scale. In: Capkun, S., Roesner, F. (eds.) 29th USENIX Security Symposium, USENIX Security 2020, August 12-14, 2020, pp. 2649–2666

Guo Z, Lin Z, Li P, Chen K (2020) Skillexplorer: understanding the behavior of skills in large scale. In: USENIX Security Symposium

Indyk P, Motwani R (1998) Approximate nearest neighbors: Towards removing the curse of dimensionality. In: Vitter JS (ed.) Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998, pp 604–613. https://doi.org/10.1145/276698.276876

Jaccard P (1912) The distribution of the flora in the alpine zone. New Phytol 11(2):37–50

Kumar D, Paccagnella R, Murley P, Hennenfent E, Mason J, Bates A, Bailey M (2018) Skill squatting attacks on amazon alexa. In: Enck, W., Felt, A.P. (eds.) 27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018, pp. 33–47. https://www.usenix.org/conference/usenixsecurity18/presentation/kumar

Le T, Huang DY, Apthorpe N, Tian Y (2022) Skillbot: identifying risky content for children in alexa skills. ACM Trans Internet Technol. https://doi.org/10.1145/3539609

Le Q.V, Mikolov T (2014) Distributed representations of sentences and documents. In: Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014. JMLR Workshop and Conference Proceedings, vol. 32, pp. 1188–119. http://proceedings.mlr.press/v32/le14.html

Lentzsch C, Shah S.J, Andow B, Degeling M, Das A, Enck W (2021) Hey alexa, is this skill safe?: Taking a closer look at the alexa skill ecosystem. In: 28th Annual Network and Distributed System Security Symposium, NDSS 2021, Virtually, February 21-25, 2021. https://www.ndss-symposium.org/ndss-paper/hey-alexa-is-this-skill-safe-taking-a-closer-look-at-the-alexa-skill-ecosystem/

Liao S, Wilson C, Cheng L, Hu H, Deng H (2020) Measuring the effectiveness of privacy policies for voice assistant applications. In: ACSAC '20: Annual Computer Security Applications Conference, Virtual Event / Austin, TX, USA, 7-11 December, 2020, pp. 856–869. https://doi.org/10.1145/3427228.3427250

Li S, Bu L, Bai G, Guo Z, Chen K, Wei H (2022) Vitas: Guided model-based vui testing of vpa apps. In: 37th IEEE/ACM international conference on automated software engineering, pp 1–12

Manaa ME, Abdulameer G (2018) Web documents similarity using k-shingle tokens and minhash technique. J Eng Appl Sci 13(6):1499–1505

Meng G, Xue Y, Xu Z, Liu Y, Zhang J, Narayanan A (2016) Semantic modelling of android malware for effective malware comprehension, detection, and classification. In: Proceedings of the 25th International Symposium on Software Testing and Analysis, pp. 306–317

Meng G, Patrick M, Xue Y, Liu Y, Zhang J (2018) Securing android app markets via modeling and predicting malware spread between markets. IEEE Trans Inf Forensics Secur 14(7):1944–1959

Meng G, Feng R, Bai G, Chen K, Liu Y (2018) DroidEcho: an in-depth dissection of malicious behaviors in Android applications. Cybersecurity 1:4. https://doi.org/10.1186/s42400-018-0006-7

Mikolov T, Chen K, Corrado G, Dean J (2013) Efficient estimation of word representations in vector space. In: Bengio Y, LeCun Y (eds.) 1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings. arXiv:1301.3781

Misu M.R.H, Sakib K (2017) Interface driven code clone detection. In: Lv J, Zhang HJ, Hinchey M, Liu X (eds.) 24th Asia-Pacific Software Engineering

Conference, APSEC 2017, Nanjing, China, December 4-8, 2017, pp. 747–748. https://doi.org/10.1109/APSEC.2017.97
Misu M.R.H, Satter A, Sakib K (2017) An exploratory study on interface similarities in code clones. In: 24th Asia-Pacific Software Engineering Conference Workshops, APSEC Workshops 2017, Nanjing, China, December 4-8, 2017, pp. 126–133. https://doi.org/10.1109/APSECW.2017.24
Nakamura Y, Choi E, Yoshida N, Haruna S, Inoue K (2016) Towards detection and analysis of interlanguage clones for multilingual web applications. In: 10th International Workshop on Software Clones, IWSC@SANER 2016, Osaka, Japan, March 15, 2016, pp. 17–18. https://doi.org/10.1109/SANER.2016.55
NLTK (2022). https://www.nltk.org
Pati J, Kumar B, Manjhi D, Shukla KK (2017) A comparison among arima, bp-nn, and MOGA-NN for software clone evolution prediction. IEEE Access 5:11841–11851. https://doi.org/10.1109/ACCESS.2017.2707539
Ragkhitwetsagul C, Krinke J (2017) Using compilation/decompilation to enhance clone detection. In: Kraft NA, Godfrey MW, Sajnani H (eds.) 11th IEEE International Workshop on Software Clones, IWSC 2017, Klagenfurt, Austria, February 21, 2017, pp. 8–14. https://doi.org/10.1109/IWSC.2017.7880502
Sajnani H, Saini V, Svajlenko J, Roy C.K, Lopes C.V (2016) Sourcerercc: scaling code clone detection to big-code. In: Dillon LK, Visser W, Williams LA (eds.) Proceedings of the 38th International Conference on Software Engineering, ICSE 2016, Austin, TX, USA, May 14-22, 2016, pp. 1157–1168. https://doi.org/10.1145/2884781.2884877
Sheneamer A, Roy S, Kalita J (2018) A detection framework for semantic code clones and obfuscated code. Expert Syst Appl 97:405–420. https://doi.org/10.1016/j.eswa.2017.12.040
Sheneamer A, Kalita J (2016) Semantic clone detection using machine learning. In: 15th IEEE International Conference on Machine Learning and Applications, ICMLA 2016, Anaheim, CA, USA, December 18-20, 2016, pp. 1024–1028. https://doi.org/10.1109/ICMLA.2016.0185
spaCy (2022). https://spacy.io
SRLabs: Smart Spies (2022). https://srlabs.de/bites/smart-spies
Tekchandani R, Bhatia RK, Singh M (2017) Code clone genealogy detection on e-health system using hadoop. Comput. Electr. Eng. 61:15–30. https://doi.org/10.1016/j.compeleceng.2017.05.011
Vislavski T, Rakic G, Cardozo N, Budimac Z (2018) LICCA: A tool for cross-language clone detection. In: Oliveto R, Penta MD, Shepherd DC (eds.) 25th International Conference on Software Analysis, Evolution and Reengineering, SANER 2018, Campobasso, Italy, March 20-23, 2018, pp. 512–516. https://doi.org/10.1109/SANER.2018.8330250
Wang D, Chen K, Wang W (2021) Demystifying the vetting process of voice-controlled skills on markets. Proc. ACM Interact. Mob. Wearable Ubiquitous Technol. 5(3):130–113028. https://doi.org/10.1145/3478101
Wang P, Svajlenko J, Wu Y, Xu Y, Roy C.K (2018) Ccaligner: a token based large-gap clone detector. In: Chaudron M, Crnkovic I, Chechik M, Harman M (eds.) Proceedings of the 40th International Conference on Software Engineering, ICSE 2018, Gothenburg, Sweden, May 27 - June 03, 2018, pp. 1066–1077. https://doi.org/10.1145/3180155.3180179
Wu HC, Luk RWP, Wong K, Kwok K (2008) Interpreting TF-IDF term weights as making relevance decisions. ACM Trans. Inf. Syst. 26(3):13–11337. https://doi.org/10.1145/1361684.1361686
Xue H, Venkataramani G, Lan T (2018) Clone-hunter: accelerated bound checks elimination via binary code clone detection. In: Gottschlich J, Cheung A (eds.) Proceedings of the 2nd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages, MAPL@PLDI 2018, Philadelphia, PA, USA, June 18-22, 2018, pp. 11–19. https://doi.org/10.1145/3211346.3211347
Yang Y, Ren Z, Chen X, Jiang H (2018) Structural function based code clone detection using a new hybrid technique. In: Reisman S, Ahamed SI, Demartini C, Conte TM, Liu L, Claycomb WR, Nakamura M, Tovar E, Cimato S, Lung C, Takakura H, Yang J, Akiyama T, Zhang Z, Hasan K (eds.) 2018 IEEE 42nd Annual Computer Software and Applications Conference, COMPSAC 2018, Tokyo, Japan, 23-27 July 2018, Volume 1, pp. 286–291. https://doi.org/10.1109/COMPSAC.2018.00045
Yuan X, Chen Y, Zhao Y, Long Y, Liu X, Chen K, Zhang S, Huang H, Wang X, Gunter C.A (2018) Commandersong: A systematic approach for practical adversarial voice recognition. In: Enck, W., Felt, A.P. (eds.) 27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018, pp. 49–64. https://www.usenix.org/conference/usenixsecurity18/presentation/yuan-xuejing
Yuki Y, Higo Y, Kusumoto S (2017) A technique to detect multi-grained code clones. In: Kraft, N.A., Godfrey, M.W., Sajnani, H. (eds.) 11th IEEE International Workshop on Software Clones, IWSC 2017, Klagenfurt, Austria, February 21, 2017, pp. 54–60. https://doi.org/10.1109/IWSC.2017.7880510
Yu D, Wang J, Wu Q, Yang J, Wang J, Yang W, Yan W (2017) Detecting java code clones with multi-granularities based on bytecode. In: Reisman S, Ahamed SI, Demartini C, Conte TM, Liu L, Claycomb WR, Nakamura M, Tovar E, Cimato S, Lung C, Takakura H, Yang J, Akiyama T, Zhang Z, Hasan K (eds.) 41st IEEE Annual Computer Software and Applications Conference, COMPSAC 2017, Turin, Italy, July 4-8, 2017. Volume 1, pp. 317–326. https://doi.org/10.1109/COMPSAC.2017.104
Zhang N, Mi X, Feng X, Wang X, Tian Y, Qian F (2019a) 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), pp. 1381–1396. https://doi.org/10.1109/SP.2019.00016
Zhang Y, Xu L, Mendoza A, Yang G, Chinprutthiwong P, Gu G (2019b) 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. https://www.ndss-symposium.org/ndss-paper/life-after-speech-recognition-fuzzing-semantic-misinterpretation-for-voice-assistant-applications/

Publisher's Note