Designing Secure and Usable Wake Words

A Capstone Report
presented to the faculty of the
School of Engineering and Applied Science
University of Virginia

by

Timothy Xiaolei Han

with

Joshua Sahaya Arul
Andrew Wang

May 11, 2021

Capstone advisor: Yuan Tian, Department of Computer Science

Designing Secure and Usable Wake Words

TIMOTHY HAN, University of Virginia
JOSHUA SAHAYA ARUL, University of Virginia
ANDREW WANG, University of Virginia

Smart speakers have become a common sight in homes, business, and personal products. However, the fact that they often accidentally trigger from everyday conversation raises privacy concerns. In this paper, we propose a novel method for generating intentionally secure and usable wake word phrases for smart speakers.

The paper ranks words based on their uniqueness score by finding the most common combinations of set-length phonemes, and calculating the Levenshtein distance between these common sequences and possible wake-words. The words with the highest distances were hand-chosen with usability in mind and compared by the number of misactivations to existing wake-words and intentionally chosen "bad" wake-words with low distances. The chosen words consistently outperformed existing wake-words and intentionally chosen bad wake-words, demonstrating the ability to generate more secure wake-words using phonetic edit distance.

Additional Key Words and Phrases: wake word, data privacy, security, smart speaker

1 INTRODUCTION

There are millions of voice activated assistants in devices all across the world, ranging from phones and computers, such as Siri and Cortana, to home speakers such as Amazon Alexa and Google Homes. The largest of producer of these products, Amazon, sold over 100 million devices by 2019 (Bohn [2]), and doubled that number in 2020 (Rubin [9]). These devices found ubiquity partly because of convenience. Smart speakers are activated using a keyword known as a "wake word" followed by a command. For example, users can ask a Google Home for the weather by saying, "Okay Google (the wake word), what is the weather like?" While increasingly pervasive and exceedingly useful, however, these smart speakers also come with new concerns over loss of data privacy. More specifically, these voice-controlled devices' microphones are always on, listening and searching the "wake word", processing audio on the local computer inside. If one is found, the subsequent audio input is streamed to the cloud to be transcribed and executed.

Unfortunately, when wake words were first designed, very few were designed with security or usability in mind. For example, Siri was the name of an Apple employee in Norway and picked for it's brevity and uniqueness; Cortana was picked because it was the name of the assistant in Halo, first appearing in 2001 [1]. Because of the arbitrary nature of these wake words, accidental triggers can occur from anything, ranging from normal conversation, to TV and radio, to other malicious voice-controlled device skills. The words “cocaine noodles” or “OK cool” can trigger a Google Home, whose wake word is “OK Google” (Vaidya et al. [12]). An accidental trigger is a case of privacy failure, as the device will stream audio without user consent to be processed in the cloud, possibly leaving an avenue for exploitation and stealing of critical information.

This project will find suitable wake words which limit the number of accidental triggers while staying memorable and intuitive for smart speaker users.

2 RELATED WORK

There is a growing body of work tackling the security of voice-activated smart speakers.

Authors' addresses: Timothy Han, University of Virginia; Joshua Sahaya Arul, University of Virginia; Andrew Wang, University of Virginia.

Vaidya et al. [12] showed in 2015 that the words “Cocaine Noodles” could be used to wake a Google Home and perform sensitive tasks such as sending texts or calling a number. They achieved this by running audio through a *mangler*, which kept enough audio features for a voice recognition system to perform transcription, but made it difficult for humans to understand. This paper uses the Levenshtein distance between phonemes as a performance metric to determine if humans could identify intended words from mangled audio, an idea our paper will build on.

Zhang et al. [13] examined several ways these smart speakers could be attacked. One such attack was called “skill squatting”, the practice of naming skills to very similar names to others in order to take advantage of imperfect transcription and retrieve sensitive user data. For example, a skill could be called “Capital Won” instead of the banking skill “Capital One”, and trick users to give up bank information. This paper used edit distance to find similarly named skills in order to detect skill squatting attacks.

Dubois et al. [3] tried to answer the question of if, how, why, and when smart speakers misactivated from wake-words by playing TV shows next to several smart speakers. Detections were found by examining video footage of the smart speakers, seeing cloud logs of activations provided by the vendors, and sniffing traffic packets for detection activity.

State of the art work in Schönherr et al. [10] automated the process of finding unsafe wake words. The group first showed that accidental triggers occured by playing several audio corpuses to several smart speakers and logging activations. The group then hypothesized that words with similar pronunciations as the wake-words were promising candidates for accidental triggers. A weighted Levenshtein distance measure was used to find words similar to the existing wake-words.

3 METHODOLOGY

Schonherr et.al. have previously identified that words phonetically similar to existing wake words can cause accidental triggers[10]. Conversely, their findings also imply that wake words with distinct phonetic qualities will be less prone to accidental triggers. To quantify phonetic distinctness, we formulate a system to identify common phonetic patterns in the English language.

Sequences of sounds recur in the English language. For instance, the sequence “EY1 SH AH0 N,” the phonemic transcription of “-ation,” reoccurs word-final between words such as nation, animation, elevation, and celebration. Certain sequences appear more often than other sequences in the English language. If a sequence appears often in English, words containing the sequence will exhibit low levels of phonetic distinctiveness. Conversely, if a word only contains rare sequences, then this word will exhibit high levels of phonetic distinctiveness. Since phonetic distinctiveness is a crucial factor in creating wake words that minimize accidental triggers, words that do not contain common sequences will make good wake words.

There are multiple ways of determining phonemic sequence frequencies. We decided on a method that counts the frequency of n consecutive phonemes within some reference corpus representative of the English language. We define n as a tunable hyperparameter representing the sequence length; a smaller n can identify common sequences within words while a larger n can identify common sequences between words.

Given a list of common phonemic sequences, to quantify the extent to which these sequences appear in a word, we make use of a modified version of Levenshtein distance. Briefly, Levenshtein distance quantifies the similarity between a reference and target text by computing the number of inserts, deletions, substitutions necessary to transform the reference to the target. Our approach uses Levenshtein distance to compute the similarity between the phonemic transcription of a candidate word and some common phonemic sequence. We scale the distance by multiplying the

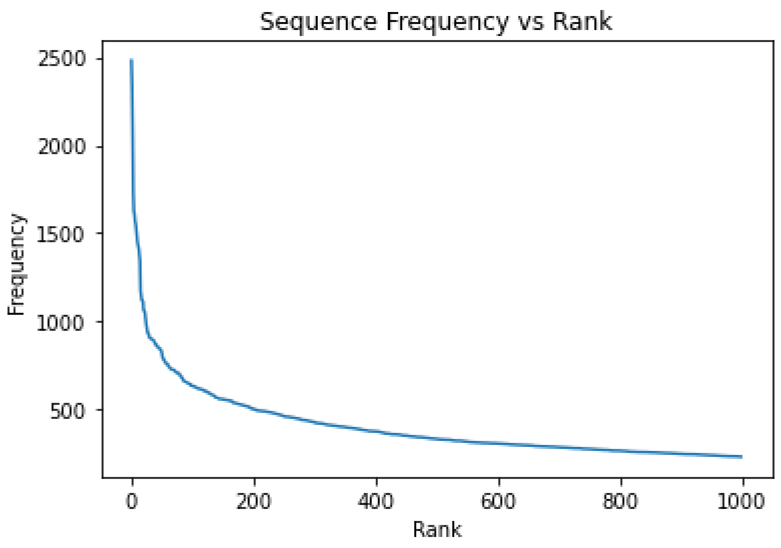

Fig. 1. The frequency of phonemic sequences in the English language corresponds inversely with rank. The sequence length n for this figure is $n = 5$.

frequency of the phonemic sequence to reward words for having greater distance away from common sequences. However, we observed that the frequency of sequences was inversely related to rank, similar to Zipf's law as demonstrated in Figure 1. To account for this tendency, we took the log of the scaled distance. Thus, to generate the uniqueness scores, for each word we sum the log of the scaled distance across all sequences, as given in equation 1, where S represents the set of all sequences and w represents a candidate word.

$$\text{score}(w) = \sum_{s \in S} \log\left(\text{freq}(s) * \text{dist}(s, w)\right) \tag{1}$$

We used human evaluation to select wake words after we computed the uniqueness scores. We considered not just the uniqueness scores, but also the suitability of the word as a wake word. Suitability is a metric that is difficult to quantify or measure objectively. Instead, we asked ourselves whether we would be willing to use a given word as a wake word for our smart speakers. Consequently, the words suggested may not necessarily have the highest uniqueness scores. Nevertheless, we believe the words selected are best suited for the task of avoiding accidental activations. Additionally, to further simulate real world conditions, we combined "okay" with each of our wake words to constitute a wake phrase, consisting of "okay" followed by one of our selected words. In the end, we produce a set of phonetically distinct wake phrases.

4 IMPLEMENTATION

4.1 Datasets

To provide the phonemic transcriptions, the CMU transcription dictionary was used[5]. The CMU transcription dictionary provides transcriptions for 134,000 words using the ARPAbet symbol set. The Brown corpus[4] contributed to the generation of the candidate wake word vocabulary and informed the analysis of phonetic sequences. We used the Brown corpus since it provides a representative perspective of the English language while remaining manageable in size. A larger dataset would yield excessively expensive computations when deriving phonemic sequence frequency.

4.2 Scoring

To create a vocabulary of candidate words, we used gensim to lemmatize the words in the CMU pronouncing dictionary and extracted the nouns. This yielded a vocabulary size of 69,185 words. We then counted the frequency of these words in the Brown corpus and selected only the words with at least one occurrence. Words with one or less occurrences were observed to occur seldom in everyday language and would consequently be strange or awkward to use as wake words. Once we pruned the vocabulary using the Brown corpus, we produced a vocabulary with 10,119 words.

Within the Brown corpus, we generated rankings for n ranging from 5 through 11 phonemes long and for word sizes ranging from 6 through 10 phonemes long. When compiling the phonemic sequence frequencies at each n, we kept only the top 1000 most frequent sequences. We found that sequences that occurred less frequently contributed very little to the overall uniqueness score given to a word, and therefore could be discarded from calculations.

In total, we generated scores for 5044 words. Of these 5044 words we selected 8 phonetically distinct words for each word size from 6 through 10 phonemes in length, for a total of 40 wake words. We also selected 8 phonetically common words, 7 of phoneme length 7 and 1 of phoneme length 10, to determine whether the phonetically distinct words we have selected are less prone to accidental activations than the phonetically common words selected.

5 EVALUATION

5.1 Generating Wake Word Engines

We generated wake word engines using Porcupine, a wake word engine developed by Picovoice [7]. The models use deep neural nets and are designed to be lightweight and efficient enough to fit into IoT devices, while also being robust enough outperform other popular wake work engines (Snowboy and PocketSphinx) [8]. We selected Picovoice because of its strong performance and simple integration. For each wake word, we used the Picovoice console to generate and download a wake word model. These models are stored as custom .ppn files which can be used by the pvporcupine Python library.

5.2 Evaluation Dataset

Earlier Porcupine repositories have indicated that they train their models using the LibriSpeech dataset [11], so we wanted to pick another large speech dataset with good variety. We found the Common Voice English dataset, containing 2,181 hours of audio from 66,173 speakers of different accents [6]. It also comes with tables containing each audio file's transcript and the speaker's accent, which allows us to check for misactivations easily and could enable further investigation into the relation between misactivations and accents.

5.3 Results

For each wake word, the Porcupine engine using the corresponding model file processed each audio file in the dataset and recorded all wake word detections to an output file. See Appendix A for tables with the results.

The selected wake words performed to near perfection. According to the Picovoice, Porcupine has a 5.3% chance of obtaining 1 false alarm per 10 hours of audio data, so the expected number of misactivations when running against the

2,181 hours of the Common Voice dataset is 11.6 misactivations [8]. The selected wake words exceeded expectations, with nearly all having 2 to 0 misactivations. Of the 40 selected wake words, three had 2 misactivations, five had 1 misactivation, and thirty-one had 0 misactivations. This contrasts starkly with popular wake words "Alexa" and "Hey Siri", and the selected bad wake words. "Alexa" had 188 misactivations, "Hey Siri" had 42 misactivations, and the selected bad wake words had on average 15 misactivations. The average number of misactivations for the selected good wake words, on the other hand, was 0.5. Among the selected wake words, the shorter 6 phoneme words were more likely to be misactivated compared to other longer phoneme words.

Among all of the wake words, "Okay Google" and "Okay holdup" were two noticable outliers. As aforementioned in the introduction, previous studies have identified several phrases that can misactivate "Okay Google"; yet, it performed nearly perfectly in our study with only 1 misactivation. The "Okay" prefix could have an unknown advantage against the Common Voice dataset, although the selected bad words have demonstrated that simply adding the prefix is not a solution.

Altogether, these results demonstrate that our method of generating secure and usable wake words is promising. The difference in misactivations between the selected good and bad wake words demonstrate that our scoring mechanism is a viable metric for predicting the security of wake words. Additionally, popular wake words "Alexa" and "Hey Siri" were outperformed by both the good and bad wake words, demonstrating that these companies can substantially increase their smart speaker security by providing different wake words. Strangely "Okay Google," a wake word known to be flawed, performed on par with our good wake words. Further study should be conducted to determine if they are of similar performance, meaning that our method needs substantial improvement, or if "Okay Google" is more susceptible to misactivation either to due the existence of many phonetically similar phrases, a weaker wake word engine underlying Google Home, or otherwise.

5.4 Future Work

One underlying assumption in our setup is that the results using the Porcupine wake word engine will translate to other wake word engines. It is possible that other wake word engines' relative performances between words may be different. For instance, a wake word engine may use a Bayesian neural network instead of a deep neural network, which could cause certain classes of words to outperform others in cases where they did not with Porcupine. Future work could verify if these results are reproducible using other wake word engines.

Another limitation of our methodology is our test audio files did not consider the distance the person was away from the microphone/smart speaker nor human experimentation. The smart speaker's ability to detect wake words could depend on how far away the user is, but this was beyond the scope of our project. The Common Voice dataset did have audio files of different volumes, but this was not measured and studied in our study. Conducting the testing phase would humans would help study for this and also how easy certain wake words are to say and to remember. Future studies should incorporate human experimentation with varying conditions to more thoroughly evaluate the potential of our proposed methodology.

5.5 Conclusion

In this paper, we propose a novel method for generating secure and usable wake word phrases. The method begins with ranking dictionary words based on their uniqueness score, which is a weighted score of the Levenshtein distances between the word and the most frequent phonemic sequences. The best wake words are then hand-picked and prepended with "Okay" to form wake word phrases. This method outperformed commonly used wake words ("Alexa" and "Hey

Siri") as well as intentionally chosen bad wake words. Our results demonstrate that our scores can be used to reliably generate secure and usable wake words. We hope that future studies will be able to verify our findings against more diverse datasets, inspire companies to craft more secure wake words, and improve data privacy for end users.

6 CITATIONS AND BIBLIOGRAPHIES

REFERENCES

[1] Karissa Bell. 2017. Hey, Siri: How'd you and every other digital assistant get its name? https://mashable.com/2017/01/12/how-alexa-siri-got-names/

[2] Dieter Bohn. 2019. Amazon says 100 million Alexa devices have been sold - what's next? https://www.theverge.com/2019/1/4/18168565/amazon-alexa-devices-how-many-sold-number-100-million-dave-limp

[3] Daniel J. Dubois, Roman Kolcun, Anna Maria Mandalari, Muhammad Talha Paracha, David Choffnes, and Hamed Haddadi. 2020. When Speakers Are All Ears: Characterizing Misactivations of IoT Smart Speakers. In *Proc. of the Privacy Enhancing Technologies Symposium (PETS)*.

[4] W. N. Francis and H. Kucera. 1979. *Brown Corpus Manual.* Technical Report. Department of Linguistics, Brown University, Providence, Rhode Island, US. http://icame.uib.no/brown/bcm.html

[5] The Speech Group. 2014. http://www.speech.cs.cmu.edu/cgi-bin/cmudict#about

[6] Mozilla. 2020. Common Voice by Mozilla. https://commonvoice.mozilla.org/en/datasets

[7] Picovoice. [n.d.]. Picovoice/porcupine. https://github.com/Picovoice/porcupine

[8] Picovoice. [n.d.]. Picovoice/wake-word-benchmark. https://github.com/Picovoice/wake-word-benchmark

[9] Ben Fox Rubin. 2020. Amazon's Alexa world just got much bigger. https://www.cnet.com/home/smart-home/amazon-sees-alexa-devices-more-than-double-in-just-one-year/

[10] Lea Schönherr, Maximilian Golla, Thorsten Eisenhofer, Jan Wiele, Dorothea Kolossa, and Thorsten Holz. 2020. Unacceptable, where is my privacy? Exploring Accidental Triggers of Smart Speakers. *CoRR* abs/2008.00508 (2020). arXiv:2008.00508 https://arxiv.org/abs/2008.00508

[11] Ssassi. [n.d.]. ssassi/Porcupine. https://github.com/ssassi/Porcupine

[12] Tavish Vaidya, Yuankai Zhang, Micah Sherr, and Clay Shields. 2015. Cocaine Noodles: Exploiting the Gap between Human and Machine Speech Recognition. In *9th USENIX Workshop on Offensive Technologies (WOOT 15)*. USENIX Association, Washington, D.C. https://www.usenix.org/conference/woot15/workshop-program/presentation/vaidya

[13] Nan Zhang, Xianghang Mi, Xuan Feng, XiaoFeng Wang, Yuan Tian, and Feng Qian. 2019. 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)*. 1381–1396. https://doi.org/10.1109/SP.2019.00016

A SELECTED WAKE WORDS

Wake Phrase	**No. of Misactivations**
6 Phoneme Word	
okay burbank	1
okay watchdog	0
okay jaguar	0
okay adviser	1
okay hapgood	0
okay luxury	0
okay holdup	9
okay langford	2
7 Phoneme Word	
okay liberace	0
okay virtuoso	0
okay hollywood	0
okay matchmaker	0
okay phalanx	0
okay churchyard	0
okay blackjack	0
okay hemingway	1
8 Phoneme Word	
okay livelihood	0
okay handkerchief	0
okay quasimodo	0
okay riverbank	0
okay housekeeping	0
okay blackboard	0
okay goldwater	2
okay hemisphere	0
9 Phoneme Word	
okay frothingham	0
okay exemplar	0
okay programming	0
okay enthusiast	0
okay stronghold	1
okay springboard	0
okay francesco	0
okay quarterback	2
10 Phoneme Word	
okay phraseology	0
okay newspaperman	0
okay headquarters	0
okay electroshock	0
okay linguistic	0
okay choreographer	0
okay ticonderoga	0
okay polynomial	1

Table 1. Number of misactivations for each wake phrase we selected

Bad Wake Phrase	**No. of Misactivations**
7 Phoneme Word	
okay anniston	7
okay arundel	12
okay attention	12
okay attrition	3
okay entrant	20
okay innocent	25
okay solution	27
10 Phoneme Word	
okay instrument	14

Table 2. Number of misactivations for each low quality wake phrase selected

Popular Wake Phrase	**No. of Misactivations**
hey google	1
hey siri	42
alexa	188

Table 3. Number of misactivations for proprietary wake words