Understanding and Mitigating Fake Wake-up Words

Ugur Turhal

April 24, 2023

Abstract

The smart Internet of Things devices are gaining progressive influence in everyday life. Some examples are voice assistants like *Alexa, Siri, and Google*. But these voice assistants have a lack of security, whether it is privacy or opening new doors for hackers. One example is the phenomenon of false activation by a similar wake-up word. This report is about the paper *FakeWake: Understanding and Mitigating Fake Wake-up Words of Voice Assistants*. This presents and discusses the key questions: how to efficiently generate a dataset of fuzzy words, the root causes of fuzzy words being accepted, and how to protect voice assistants against them. These core questions are tested and answered in an experiment, which is included in this report.

1 Introduction

Voice assistants are embedded in smart Internet of Things devices *e.g.* smart speakers or mobile phones. The voice assistant is software that carries out everyday tasks via voice command. This enables a person to use their voice to execute various commands. For example *"Alexa, turn the lights on!"*. This report is focusing on voice assistants in smart speakers. The existence of voice assistants leads unavoidably to a potential threat to privacy and security, as they could be a target for hackers and constantly listen to their environment. The actual paper considers also Chinese voice assistants, an analysis of these wake-up word(s) is not included in this report. This report only focuses on English voice assistants and English wake-up words. It starts with a background section, and afterward, the core questions are listed. Then the experiment setup and the generation of the wake-up word(s) are presented. After that, the report deals with the question of which words are accepted by a voice assistant and why, in the next short section, it is shown how to strengthen voice assistants, and finally the results are presented. In the last section, a conclusion is given.

2 Background

2.1 Wake-up mechanism

The wake-up mechanism in voice assistants is essential, to reduce power consumption and protect user data privacy.

2.1: Definition Wake-up

The recognition of the wake-up word/keyword(s), as well as going from standby to active mode. Such that commands could be processed.

During the standby mode the voice assistant listens to the environment and records audio snippets *e.g.* 3 *seconds* (Chen et al. (2021)) to check if the keyword(s) is/(are) present. Before being triggered for receiving commands, voice assistants can recognize a keyword and afterward process the commands. Consequently, the voice assistants listen to their environment and wait for the keyword to execute the command. The trigger words for the tested voice assistants in the paper are listed in *Table 1*:

Model	**Wake-up word**
Amazon Echo (4th gen)	Alexa
Google Nest Mini (2nd gen)	Hey Google or OK Google
Apple HomePod Mini	Hey Siri
Echo Dot	Alexa

Table 1: Keyword(s) to wake-up the voice assistants

If the lightweight local recognition model believes that the keyword is detected, the voice assistant records and sends the audio data in some cases to the cloud for further analysis.(Chen et al. (2021))

2.2: Definition lightweight local recognition model

A lightweight local recognition model in voice assistants is a small-sized and efficient neural network that runs on the user's device for real-time voice recognition. This provides enhanced privacy and responsiveness without relying on cloud-based processing. For a detailed description *see subsection 2.3*.

The *next subsection* 2.2 is dedicated to the phenomenon of the "FakeWake".

2.2 FakeWake

The correct recognition of keyword(s) (=wake-up word) for voice assistants is one of the most important components. In practice, the voice assistant can be triggered by non-keyword(s), such that the software recognizes non-keyword(s). This phenomenon is called FakeWake.

2.3: Definition: FakeWake

Wrongly activation by non-keyword(s) *(=fuzzy words)*, that are not wake-up words.

Recent surveys show that once a week 50% of users wake-up their voice assistants by mistake and 28.5% of the 50% (Chen et al. (2021)) experience daily wake-up mistakes. A wake-up from a voice assistant is only correct if and only if the user of the voice assistant intentionally wants to wake-up the voice assistant. In all other circumstances, this is an accidental wake-up. In daily life, voice assistants react to the environment. If the keyword *Alexa* is mentioned in the evening news, the smart speaker recognizes the keyword *Alexa*, and the voice assistant wakes up. This accidental wake-up is illustrated in *Figure 1*. What may seem harmless at first glance can lead to serious security vulnerabilities. *E.g.* buy accidental something on Amazon or call a bank.

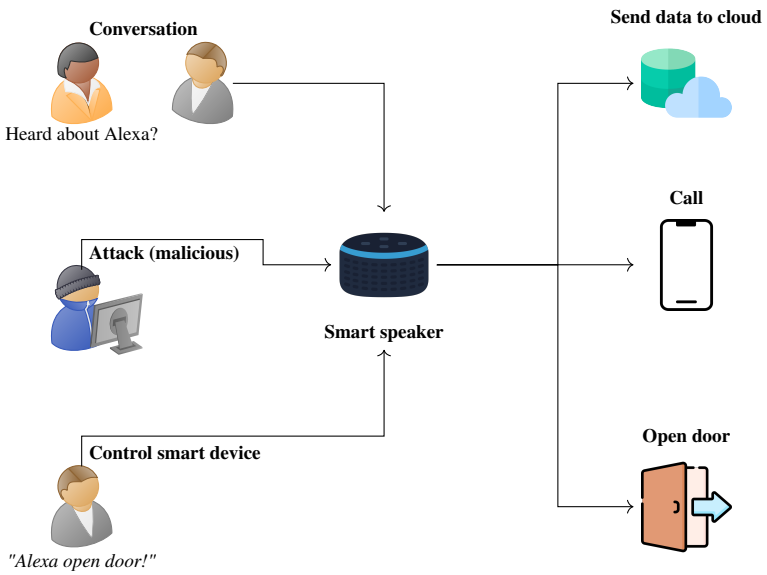


Figure 1: Illustration of FakeWake phenomenon

2.3 Artificial Intelligence

The wake-up detector is based on artificial intelligence explicitly it is based on a deep learning model a subsection of artificial intelligence. The backbone of deep learning models are neural networks. *Neural networks are the representation we make of the brain: neurons that are interconnected to other neurons, which forms a network* (Teoh and Rong (2022), *page 172*). The operation of a complete neural network is, straightforward: One enters variables as inputs (in our case it is a wake-up word). After some calculations in the hidden layer(s) *the middle part in Figure: 2*, the action of the voice assistant is returned, *the right side in Figure: 2*. It is important to see that the input variables $x_{1,2,3}$ take the keyword(s) + command, the hidden layer calculates the action, and y_1 returns the action, that the voice assistant does. To understand the acceptance of fuzzy words, the authors of the paper took the input of the neural network and interpreted the input layer as a decision tree Aytekin (2022), with machine learning algorithms. *See subsection 5*.

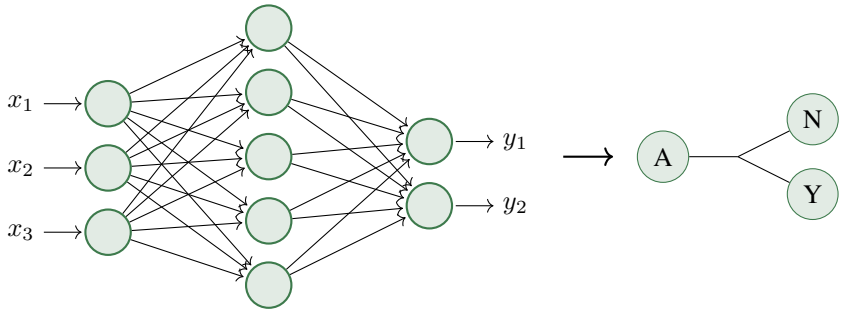


Figure 2: The input of the neural network is interpreted as a decision tree.

2.4 Knowledge about English words

An English word can be divided into graphemes that correspond to different phonemes, allowing humans to pronounce unfamiliar words based on their experience. Text-to-speech (TTS) machines can also pronounce "non-dictionary" words, expanding the range of words an attacker can create. Encoding English words presents two challenges: deciding whether to base the encoding on letter composition or phoneme and deciding how to handle the irregular spelling of the language.

2.4: Definition Grapheme & Phonem

Grapheme: A grapheme is a written symbol that represents a sound (phoneme). This can be a single letter or could be a sequence of letters. *E.g.* (kn, n = ʁŋ, ŋ) are graphemes for the phoneme /n/ (known, nose).

Phonem: smallest unit of sound that distinguishes one word from another *e.g.* d,t ⇒ bad, bat (βɑð, βɑθ)

- **Encoding:** based on phoneme composition is not practical due to difficulties in pronunciation and creating meaningful fuzzy words. A better solution is: using letter composition instead, and addressing the challenge of varied word lengths by inserting spaces between letters to increase the overall length of the encoded word. The encoding is represented using a vector $r \times n$ with variables for each letter or space, with variable values ranging from 1 to 27, for $a = 1, ..., z = 26$ and " " $= 27$.
- **Dissimilarity:** To calculate the dissimilarity distance between English words, phonemes are used instead of letters since English words do not carry pronunciation information in their encoding. The Levenshtein distance is used to compare the phoneme composition of two words, which is the minimum number of single-character edits required to transform one word into the other. The English words are denoted as:

2.5: Denotation of words

$$\mathrm{W}_1 = [p_1^1, \ldots, p_1^m],\ \mathrm{p}^{(i)} \text{ i-th phoneme}$$
$$\mathrm{W}_2 = [p_2^1, \ldots, p_2^m],\ \mathrm{p}^{(i)} \text{ i-th phoneme}$$

The conventional Levenshtein distance assumes that the distance between any pair of different phonemes is 1, while some phonemes sound similar and some phonemes sound far apart. Therefore, the authors integrated the phonetic dissimilarity of phonemes into the Levenshtein distance, to quantify the dissimilarity distance between two words. The Levenshtein distance with the dissimilarity of phonemes is represented as:

2.6: Definition: Levenshtein distance with integrated dissimilarity phonemes

$$\text{dist}(\text{W}_1, \text{W}_2) = \frac{\text{D+I+2} \cdot \sum\limits_{(i,j) \in S} \text{dis}(\text{p}_1^{(i)}, \text{p}_2^{(j)})}{m+n}$$

Where: D are the Deletions, I are the Insertions, and S = Set of substitutions, which replaces phoneme $p_1^{(i)}, p_2^{(j)}$ and m, n are the lengths of the words W_1, W_2.

3 Core-Questions

The paper *FakeWake: Understanding and Mitigating Fake Wake-up Words of Voice Assistants*, studied systematically the root causes and mitigation of the FakeWake problems. The core questions are:

- How to effectively generate a large collection of fuzzy words for a given assistant?
- What are the causes that lead to the acceptance of fuzzy words by wake-up detectors?
- How to strengthen wake-up detectors of voice assistants to be resilient against fuzzy words?

4 Generation

In this main section, the conceptual structure, to build efficiently fuzzy words is presented. This conceptual structure was built by the University of Zhejiang and the University of Darmstadt. It presents the experiment and the generation of fuzzy words. This section is fundamental for analyzing the acceptance of fuzzy words, *see section 5*.

4.1 Experiment

To test the generated fuzzy words, the authors set up the following environment *shown in Figure 3*. The laptop generates the fuzzy words, these fuzzy words are then played on a stereo. The light sensor is wired to the smart speaker and lights up if the fuzzy words are recognized. The bulb is attached to a Raspberry π, which sends the wake-up rate to the laptop for evaluation. The wake-up rate is important for understanding (*section 5*) of fuzzy words.
Important: The experiments are carried out in a quiet laboratory room. The generated audio samples *see subsection: 4.3* were read digitally and spoken out with the python module: `pyttsx3`. This module allows, that the text to speech could speak out non-dictionary words, which is a piece of important information. The default speak speed in the module is 150, which approximates the average speed of human speakers. Each word is played 10 Times.

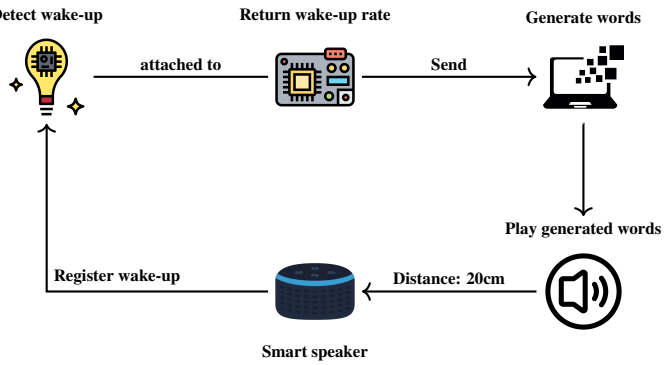


Figure 3: Test environment for fuzzy words

4.2 Collection of fuzzy word(s)

To understand why the acceptance of fuzzy words exists, a dataset of fuzzy words is necessary. This can be done in two ways:

- The first approach is naive, which plays words from audio. Afterward, check if the voice assistant reacts to the played word or not. The authors of the paper write that this approach would take days, possibly weeks and that the voice assistant mostly just wakes up to the actual keyword(s), which is not of interest *(Chen et al. (2021))*, because the authors want to create similar wake-up words in terms of sound, which is extremely important in the aspect of security, to prevent users from easily recognizing them.
- The second approach is to create a framework. However, this is complicated because most manufacturers provide very little information about the AI-detection model of the wake-up words and many of the voice assistants are black boxes.

4.1: Definition: Black box

An object whose inner structure and inner functionality are not known.

This report as well as the paper only discusses the second approach. The constructed architecture for generating fuzzy words of the *Technincal University of Darmstadt* and *Zhejiang University*, includes the following four steps:

- Generate fuzzy words.
- Mutate the best fuzzy word.
- Create new fuzzy words through multiple evolutionary generations.
- Balance the wake-up rate and the dissimilarity distance.

4.3 Generation - Implementation

This subsection is focusing on the collection of fuzzy words, and how they are generated.

- **Initialization:**
  The initialization of the new population (words/fuzzy words). In *Figure 4*, the wake-up word is included, similar words are given, and random words are given.

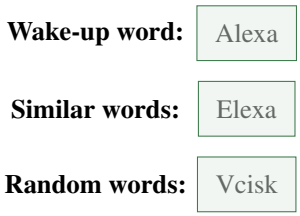


Figure 4: Inital population

- **Evaluation:**
  The evaluation is done with the Pareto frontier distribution. Since there is the possibility that certain fuzzy words account for a large part of the total words, while the majority of the fuzzy words only make a small contribution to the total meaning. What is best illustrated by the Pareto-Frontier distribution *see Figure: 5*. As an explanation to the illustration, it should be said that the best distribution is (1,1) (*see Figure: 5*), completely dissimilar sounding words that have a 100% wake-up rate. The evaluation step is constantly in exchange with the variation step 3. To always evaluate the best fuzzy word.

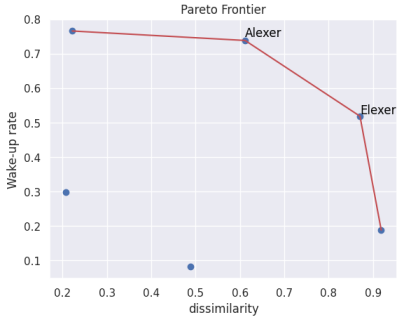
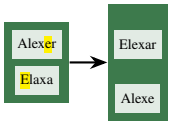


Figure 5: Distribution of the population with Pareto-Frontier

- **Variation:**
  The variation takes the the survived individual(s) from the evaluation, and creates new fuzzy word(s). By using a genetic algorithm, since it is the most suitable one to solve the problem of the fuzzy word(s) generation. In this algorithm,

a population of words is iteratively improved by varying it through a combination of crossovers (*see Subfigure: a*) and mutations (*see Subfigure: b*). In the cross-over, parts of the words are combined to produce a new word, this word should then (hopefully) be better than the received words *(from the evaluation)*. This new word is then fed back into the evaluation step. The evaluation chooses the surviving individual(s). Then the variation step gets the word, and creates variations again.



(a) Crossover

(b) Mutation

5 Understanding

Since the authors have no information about the deep learning model, and even if they had a piece of side information about the neural network, they claim that they could not derive an interpretable model from the deep learning model, due to its highly non-linear structures and massive parameters. A fuzzy word is considered as the wake-up word because the wake-up word detector (deep learning model) is fooled by certain similarities between two words. The Levenshtein distance could be used to get the decision tree and analyze which words are accepted, causing the *FakeWake* phenomenon. To evaluate the dissimilarity, the authors compared the Levenshtein distance with their own method for encoding and classifying words. Since for English words, there is no high-dimensional embedding for phonemes, the authors used multidimensional scaling to find an encoding, that preserves the dissimilarity between phonemes. Each initial/final/phoneme is encoded into a two-dimensional vector with two features. For the decision tree, the authors used a machine learning algorithm called gradient boosting to classify the phonemes. The gradient boosting algorithm employs decision trees as "weak" classifiers (a model for binary classification) and combines them into a "union" to achieve higher accuracy. The evaluation of the gradient boost algorithm is done with the 10-fold cross-validation. To measure similarity between phonemes, the authors use the output confidence score of a certain fuzzy word as its similarity score, and the dissimilarity score is simply 1 minus the similarity score $\Rightarrow 1 - \gamma$.

5.1: Definition: Feature

A feature is an individual measurable property or characteristic of a phenomenon in machine learning. Features refer in this context to the two-dimensional vector that represents each initial/final/phoneme through multi-dimensional scaling.

To evaluate the contribution of each feature, a corresponding SHAP value is calculated.

5.2: Definition SHAP

The technique used to explain the results of machine learning models. SHAP values, assign a value to each feature in a given prediction that represents its contribution to the prediction. SHAP values provide global and local explanations for machine learning models, making them useful for interpretability and confidence.

Each feature is a contributor, and add up contributions of all features to obtain the prediction result. which is:

$$y_i = y_0 + \sum_j f(\alpha_i^{(j)})$$

- y_i is the prediction result for sample x_i
- y_0 is the mean prediction result of all samples
- $\alpha_i^{(j)}$ is the j$^{\text{th}}$ feature of x_i
- $f(\alpha_i^{(j)})$ is the contribution of $\alpha_i^{(j)}$ to the predicition result

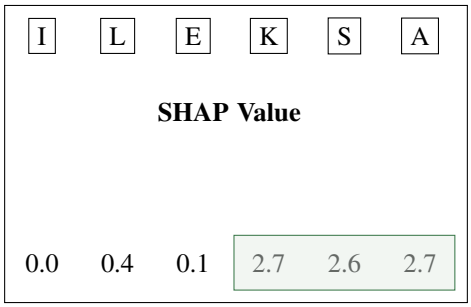


Figure 7: Feature Contribution

The contribution can be negative or positive. A positive contribution means that the feature is helpful for the accurate prediction of whether a sample is a fuzzy word or not. For any fuzzy word x_i, Let $\mathcal{A}^+{}_i$ denote the set of positive contributions. The elements

in ${\mathcal{A}^+}_i$ are ranked in a non-increasing order. This means, that subsequent elements are greater than, or equal to the previous element, *see Figure: 7*. A set $\mathcal{D}_i$ represents important features for x_i. Then the elements of ${\mathcal{A}^+}_i$ are added to $\mathcal{D}_i$ sequentially such that ratio of the contribution of all elements in $\mathcal{D}_i$ to the contribution of all elements in ${\mathcal{A}^+}_i$ is bigger than a threshold β.

$$\frac{\sum\limits_{k \in \mathcal{D}_i} f(\alpha_i^{(k)})}{\sum\limits_{k \in {\mathcal{A}^+}_i} f(\alpha_i^{(k')})} \geq \beta$$

An initial phoneme or a final phoneme is considered a decisive factor for x_i if at least one of its two features is in $\mathcal{D}_i$. In this way there is a set of decisive factors and their contributions is obtained. The contribution of an initial or a final phoneme is the sum of the contributions of its features in $\mathcal{D}_i$. *Figure: 8* gives a wonderful summary of decisive factors for fuzzy words. *KSA*, *QC* are decisive and *EQC* is bigger than the threshold.

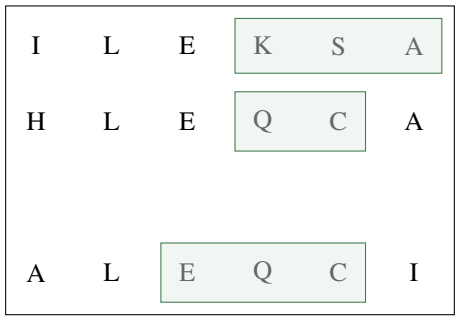


Figure 8: Decisive factors

6 Mitigating

The knowledge about the acceptance of fuzzy word(s) is essential for strengthening the voice assistant against the phenomenon of the fake wake There are two approaches the first one is to integrate the decisive factors into the existent lightweight recognition model. This method approaches to create a more complex model of the lightweight recognition model. The second approach strengthens the wake-up word detectors by retraining the voice recognition model with the generated fuzzy words of the *subsection 4.3*. Since the wake-up detectors are black box models, the authors create their own strengthened recognition model. For this three types of datasets are collected: a conventional dataset containing positive wake-up word samples and negative non-fuzzy word samples, which is divided into training and test datasets. A fuzzy word dataset (which is present from the generation) is used for retraining the original model to obtain a strengthened model.

7 Results

For Amazon Echo, Echo Dot, Google, and Apple Siri, they found 388 fuzzy words in the sum. The voice assistant *Alexa* had on the models' Echo, and Echo Dot 257 fuzzy words. Apple Siri had 52 fuzzy words and Google had 79 fuzzy words. *(Chen et al. (2021))* Since fuzzy words are environmentally sensitive, these factors were also taken into account. These factors were tested: *(See Figure: 9).*

- 1.6 decreasing & 2 increasing of the original volume.
  Result: Increasing the volume, has the influence that the wake-up rate rises for most fuzzy words of most voice assistants. For Google's voice assistant, the wake-up rate decreases with a higher volume.

- Change the speed parameter of the TTS by 100 & 200. The default TTS speed is 150.
  Reuslt: Speed has a mixed influence on the wake-up rate. Generally, as speed increases, the wake-up rate first goes up and then goes down, especially for English voice assistants, *e.g*. Echo and Apple Siri.

- Add Gaussian noise to audio samples.

  (a) Gaussian noise is set to 0.02 and 0.04.

  (b) At 0.02 the signal-to-noise Ratio is around 40db.

  (c) At 0.04 the signal-to-noise Ratio is around 30db.

  Result: In general, noises degrade the wake-up rate. For Google, the wake-up rate for fuzzy words with a medium wake-up rate grows to as high as 0.91 at a high noise level.

- Changing the voice from male to female.
  Result: After changing the speaker voice from male to female, the mean wake-up rate for fuzzy words with a high wake-up rate decreases, while the mean wake-up rate for fuzzy words with medium and low wake-up rates increases. Fuzzy words with high wake-up rates mostly maintain their wake-up rate.

2nd result is that fuzzy words with high dissimilarity distance also have big differences in vowel, sound, and emphasis from the real wake-up word.
3rd result is that the constructed dissimilarity score from the framework can better separate fuzzy words and non-fuzzy words between than Levenshtein distance. *(See Figure: 10).*
4th result, wake-up words with fewer decisive factors have more fuzzy words. *e.g*. ks for x which is present in the wake-up word: *Alexa*. means that when the wake-up word used to activate a voice assistant has fewer distinct and easily recognizable sounds or syllables, it may lead to more ambiguous or "fuzzy" interpretations by the device. *(See Figure: 11).* Siri has fewer fuzzy words since the word Siri is a less commonly-used word with distinct pronunciation, which made it difficult to produce fuzzy words.

8 Conclusion

The paper *FakeWake: Understanding and Mitigating Fuzzy Words* Chen et al. (2021), combines advanced topics such as genetics & artificial intelligence into their research. The fact, that the voice assistants are black box models since the manufacturers do not provide or provide little information about the system, made it hard to understand everything properly. I as an author would be very interested to see the code for the generation of fuzzy words. I assume that the authors do not provide their code, because of the rights, since it is their research, less than the code is a code and fix model. Because everything was clear and detailed explained. All figures, *e.g.* axes are properly labeled, such that the results are proper. Expect the fact, that there is no information if Amazon Echo and Echo Dot have the same voice assistant *Alexa*. I do not want to make a conclusion or an assumption based on the result of how many fuzzy words are collected, this can be misleading. The model Echo Dot could have a more sensitive microphone than the Amazon Echo. A very interesting point that was not mentioned at all is, that the owner of an Alexa voice assistant could change the wake-up word, this could be also a way to use own wake-up words and strengthen them. **Important:** This could be also used to weaken the voice assistant if the wake-up word is a common word such as `the`. Apart from these two points, I have nothing to complain about. The framework and the results they obtain are impressive and very detailed and the results are also based on the experiment and tempt me the author to generate my own dataset of fuzzy words.

References

Aytekin, C. (2022). Neural networks are decision trees.

Chen, Y., Bai, Y., Mitev, R., Wang, K., Sadeghi, A.-R., and Xu, W. (2021). Fakewake: Understanding and mitigating fake wake-up words of voice assistants.

Teoh, T. and Rong, Z. (2022). *Artificial Intelligence with Python*. Machine Learning: Foundations, Methodologies, and Applications. Springer Nature Singapore.

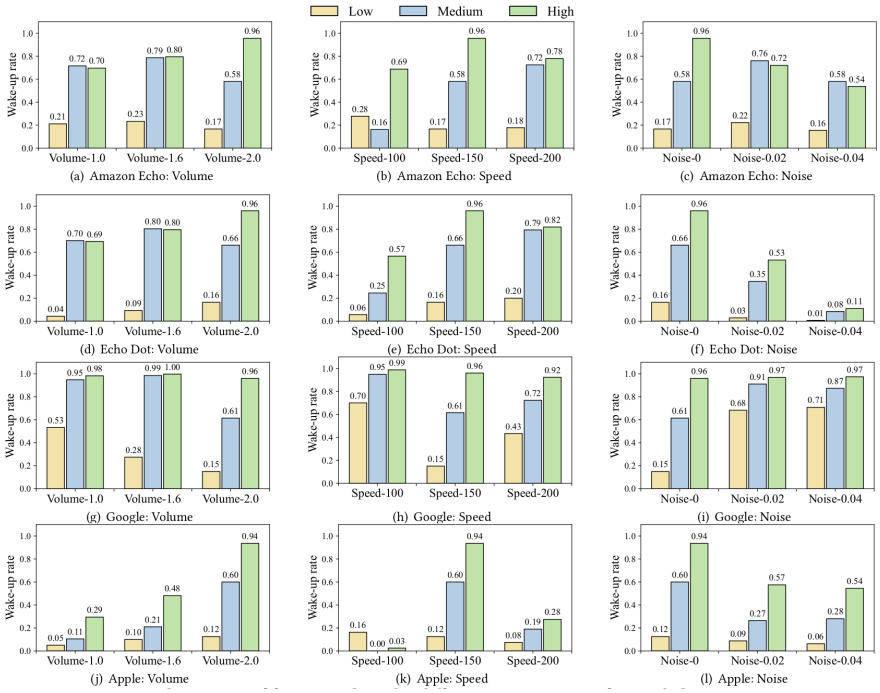


Figure 9: Wake-up rate under different environmental influences, Chen et al. (2021)

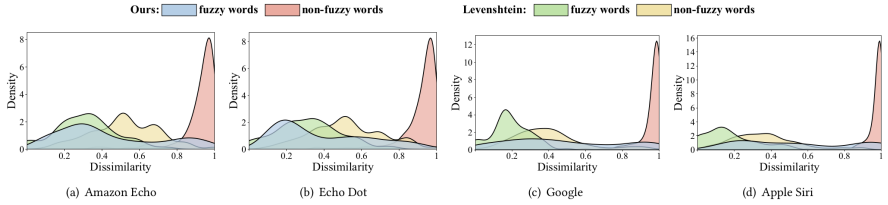


Figure 10: Levenshtein-Distance versus Own dissimilarity-score, Chen et al. (2021)

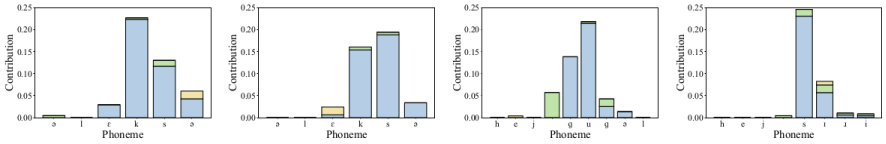


Figure 11: Decisive factors for various voice assitants, Chen et al. (2021)