sustainability

MDPI

Article

Development of an Intelligent Personal Assistant System Based on IoT for People with Disabilities

Abd-elmegeid Amin Ali [1], Mohamed Mashhour [1,*], Ahmed S. Salama [2,*], Rasha Shoitan [3] and Hassan Shaban [1]

[1] Computer Science Department, Faculty of Computers and Information, Minia University, Minia 61519, Egypt
[2] Electrical Engineering Department, Faculty of Engineering & Technology, Future University in Egypt, New Cairo 11835, Egypt
[3] Computer and Systems Department, Electronic Research Institute, Cairo 12622, Egypt
* Correspondence: mohamedmashhor11pg@fci.s-mu.edu.eg (M.M.); a.salama@fue.edu.eg (A.S.S.)

Abstract: Approximately 15% of the world's population suffers from different types of disabilities. These people face many challenges when trying to interact with their home appliances. Various solutions are introduced to increase their quality of life, such as controlling their devices remotely through their voices. However, these solutions use command templates that fail to understand the unstructured or semi-structured command. Many authors have recently integrated intelligent personal assistant (IPA) systems, such as Google Assistant, Siri, and Alexa, with control circuits to exploit the advantages of the NLP of these IPAs to control traditional home appliances. However, this solution still struggles with understanding unstructured commands and requires the internet to be available for controlling the devices. This research proposes a new IPA system integrated with IoT, called IRON, for disabled people to use to control customizable devices with a structured and unstructured voice command. The proposed algorithm receives voice orders from the person in a structured or unstructured form and transforms them into text based on the Google Speech-to-Text API. The natural language processing technique splits the commands into tokens to determine the device name and the command type, whether it is a question about device status or a statement. Afterward, the logistic regression classifies the rest of the tokens as positive or negative to turn on or off the device, then sends the command to a Raspberry Pi to control the device. The proposed IRON system is implemented using logistic regression, naïve Bayes, and the support vector machine and is trained on a created dataset consisting of 3000 normal, negative, and unstructured commands. The simulation results show that the IRON system can determine 90% of the device's names for all commands. Moreover, the IRON correctly classifies 100% of the commands as positive or negative within approximately 30 s.

Keywords: NLP; intelligent personal assistant; IoT; logistic regression; text classification; smart home; Raspberry Pi

Citation: Ali, A.-e.A.; Mashhour, M.; Salama, A.S.; Shoitan, R.; Shaban, H. Development of an Intelligent Personal Assistant System Based on IoT for People with Disabilities. *Sustainability* **2023**, *15*, 5166. https://doi.org/10.3390/su15065166

Academic Editors: Luis Hernández-Callejo, Sergio Nesmachnow and Pedro Moreno-Bernal

Received: 11 February 2023
Revised: 10 March 2023
Accepted: 13 March 2023
Published: 14 March 2023

1. Introduction

More than one billion people worldwide—almost 15% of the population—live with one or more disabilities, according to the World Health Organization [1]. These disabilities may present early in childhood or develop later in life, such as impaired hand function as a result of stroke [2]. Every day, people with disabilities significantly struggle to control their home appliances. Therefore, traditional homes should be transformed into "smart homes" to improve the quality of life of those with disabilities. Over the last decades, the objective of IoT technology has been to allow communication between devices without the need for human involvement [3]. However, currently, IoT technology has been integrated with home devices to enable these devices to be controlled remotely through the internet [4]. The term IoT describes a network of physical objects, or "things", that have sensors, software, and other technologies built into them to enable the connecting and sharing of data with other

systems and devices over the internet. These devices include light switches that respond to turn-on and off commands or thermostats that change the indoor temperature to reduce energy consumption. Different authors present several solutions for help the disabled person to control the devices remotely via IoT based on the user voice or the smartphone graphical user interface (GUI). Mittal et al. [5] propose a smart home automation system (SHAS) to control home devices based on user voice commands. The system is designed to control five groups: light, access, fan, utility, and safety. Voice Recognition Module V3 is used to recognize the voice commands, and then the Arduino microcontroller compares these commands with the previously stored command templates to perform the desired action. Abidi et al. [6] present a voice-control smart home automation system to control four home devices: a TV, LED, fan, and Compact Fluorescent Lamp (CFL). The system architecture consists of an Android phone, Arduino mega, GSM SIM900A, Bluetooth module, and ultrasonic sensor (HC-SR04). A mobile application transforms the voice command into text, and then a Bluetooth module (HC-05) transmits this text to the Arduino Mega to control the dedicated device. The ultrasonic sensor (HC-SR04) is also used to provide home security by detecting any movement and sending a message to the user's phone via GSM SIM900A to convey that there is movement in the home.

Repeatedly, Mayer et al. [7] suggest home automation architecture based on IoT technology to help people with disabilities who have trouble using a remote control or a smartphone to control their home devices. First, the voice command is recognized and processed using a speaker recognition system (ASR) on the ASR remote server. Afterwards, the ASR sends an XML file to the IoT gateway, which contains the command sent by Xbee to the appropriate device. Moreover, Saha et al. [8] have developed a home automation system that lets people use voice commands to control their electronic devices. The system uses an Android phone to capture the user's voice, which is then sent to an Arduino board via Bluetooth. The Arduino board processes the data and controls the devices accordingly through relays. The system also has an auto-mode that automatically utilizes sensors to automatically control lighting and temperature. Ismaeel et al. [9] present a home automation system that aims to control the electronic devices through voice or a webpage. If the user uses the voice application, the system receives it over microphone/mobile and employs a Raspberry Pi development board for processing. The user's voice is transcribed to text using the online speech-to-text interface (Wit). Suppose the user uses the web page to turn on or off a device. In that case, the content script text is sent to the Raspberry Pi via the Apache remote server to control the general-purpose input-output (GPIO) pins to activate or deactivate the relays connected to the devices.

Besides, Jat et al. [10] introduce the application of voice activity detection (VAD) in home automation systems to control home appliances through voice commands. VAD detects human speech using the microphone and activates a specific action. The system is implemented on a Raspberry Pi 3 Model B+, and the PocketSphinx speech recognition system captures the user's voice, suppresses the noise, and converts the analyzed speech to text. The transcribed text is then processed to determine if the text includes identified commands. According to the command, the system applies the action to the dedicated device through infrared, wireless LAN, or Bluetooth. Additionally, Alrumayh et al. [11] present a context awareness for voice assistants system (CANVAS) for use in smart homes with multiple occupants. The system consists of two major phases: a configuration webpage and a context-aware algorithm. The configuration webpage assists the owner of the home in configuring different rules to access the home devices by a drag-and-drop GUI interface. The context-aware algorithm receives the device commands from a human voices assistant to determine whether to discard or execute the user commands according to previous rules. Khan et al. [12] introduce a web application known as HAS (home automation system) to control many devices from their smartphones. This application uses Google Assistant to convert speech to text for controlling the appliances with the user's voice. It also monitors the devices' daily, weekly, and monthly power consumption. Mustafa A. Omran et al. [13] present a Wi-Fi-based home automation system for a smart home (SH) prototype. The

system uses Raspberry Pi 3 Model B+ and Arduino Mega 2560 and is controlled by a smart IOS system via the Blynk app.

Most of the methods mentioned above design their systems based on command templates; if the user uses an unstructured command, the system will not understand it. Therefore, several authors add natural language processing (NLP) to the systems to process and understand various types of commands. Intelligent personal assistant (IPA) [14] systems have recently gained popularity as a way to help individuals with many of their everyday tasks. This personal assistant system analyzes human speech based on NLP and produces a set of direct instructions. Examples of well-known IPAs are Google Assistant from Google [15], Siri from Apple [16], and Alexa from Amazon [17]. These well-known IPAs set alarms, set ringtones, close applications, make phone calls, order food online, play music, etc., with differing accuracy, according to each company. However, these IPAs do not work with all home devices. These IPAs work with smart devices designed to be controlled by their applications. Many authors integrate these IPAs with a control circuit to control any device. Uma et al. [18] introduced a home application system to turn on or off home devices via voice or text based on Google Assistant. When the user is not physically present in the environment, he can schedule the state of the appliances and be given the option to turn them on for a predetermined period of time. The user uses the mobile application to send his voice command through Google Assistant. The voice command is fed to the firebase, which the NodeMCU can fetch to turn on or off the home devices. The application function is implemented using Node-RED technology, deployed in a Dialogflow account, and integrated with Google Assistant.

Furthermore, Kumer et al. [19] introduced a voice-based smart home automation system using IFTTT, Adafruit cloud, Ubidots IoT dashboard, and Google Assistant to efficiently control energy consumption. The voice commands are received by Google Assistant and sent to IFTTT. These commands are then sent to the microcontroller via Wi-Fi to turn the device on or off, according to the user's command. The microcontroller updates the device status on the Ubidots dashboard, and feedback is sent to the user concerning this status. Putthapipat et al. [20] develop a home automation system based on the PiFrame framework. The system goal is to create an open platform for researchers to extend the features and develop their designs, similar to products such as Google Home and Amazon Alexa, but with more flexibility. The system hardware consists of a Raspberry Pi, a microphone, a speaker, and relays to control the home devices using voice commands. Google Speech API and Wit.ai are used to understand the user's command and communicate with the home appliances. The system stores the home device's current status on the Google Cloud Platform Cloud SQL through the Node.js application on Raspberry Pi. The response to the user is generated by translating the text to speech using Google Translate. Tiwari et al. [21] proposed a home assistant system that can be controlled and monitored using voice commands. In this system, the user can control the home devices and sensors from any cloud-connected platform through speech, text, or a mobile application. The system uses advanced speech recognition techniques, such as Mel-Frequency Cepstrum Coefficients (MFCC), with other features to create the feature vector. Vector Quantization (VQ) and Principal Component Analysis (PCA) are used for feature vector dimensional reduction and a Gaussian Mixture Model (GMM) to classify the speakers. The system uses cloud services IBM's Bluemix and Google's cloud service for speech-to-text conversion. Mehrabani et al. [22] presented a personalized speech recognition system using dynamic hierarchical language modeling for controlling customizable devices in smart home applications.

The previous methods integrate the IPAs with control circuits to benefit from the NLP of IPAs to increase command understanding accuracy and control any device type. However, while these methods can understand structured and semi-structured voice commands, they still cannot understand unstructured commands. Moreover, these methods depend mainly on the internet to take actions because they save commands on firebase or use the mobile phone as a GUI to send voice commands.

This paper proposes a new IPA system integrated with IoT, called IRON, to help the elderly and disabled people control their home devices remotely through IoT, based on their voices. The IRON system is designed from three modules: a speech-to-text module, a text analysis and classification module, and a command execution module. Instead of mobile phones, the IRON detects the user's voice from different microphones spread around the home. This option helps disabled people operate the devices from any place inside the home, in case the phone is lost or the internet is disconnected. The speech-to-text module transforms the incoming voice into text. IRON possesses two ways to turn voice commands into text: if the internet is available, it uses the Google Cloud API, and if it is not, it uses a speech recognition library. Thus, IRON can work online or offline. In the text analysis and classification module, the transcribed text passes through two main steps for analysis. First, IRON splits the text into a list of tokens by applying NLP to the transcribed text to search for the device name, according to the list of predefined devices. If the device is not included in the devices list, the user can ask the IRON to add a new device to the devices list and configure it without recoding the IRON. Second, IRON takes the device's name out of the command and determines if the statement is a question or an order. If the statement is a question, the IRON answers regarding the device status and break; otherwise, IRON considers this statement as a command and sends the rest to the pre-trained machine learning model to classify whether this command is a positive or negative statement. According to the statement type, IRON executes the required action on the device through the GPIO pins. The IRON system is designed to overcome the cons of the previous methods of understanding unstructured commands by adding a machine learning algorithm with NLP to classify these commands as positive or negative for turning on or off a device. The main contribution of this research can be abstracted as follows:

1. An IRON system that helps the elderly and impairment to control their home devices through their voice is proposed.
2. The dataset consists of 3000 normal, negative, and unstructured commands to control 100 devices.
3. Natural language processing is applied to the text generated from the google speech API for splitting the text into tokens for further classification.
4. A machine learning algorithm is included in IRON for classifying the commands as positive or negative.
5. Multi-microphones are distributed in different locations in the home to ensure that the elderly and disabled can access IRON from their locations.
6. New devices can be added and reconfigured by the impaired person's voice without re-coding the IRON.
7. The IRON system is designed to work online or offline and to turn off, on, or adjust a device range as fan speed.
8. The IRON system is secured by requesting a password to start the controlling process.

2. System Architecture and Design

This research proposes an IPA system based on IoT, called IRON, to assist people with disabilities in controlling their home appliances based on their voices. Figure 1 shows the proposed IRON system architecture. This figure shows that the IRON hardware system consists of a microphone, speaker, Raspberry Pi, Wi-Fi module, and relays. When the user opens the IRON, it introduces itself and tells how to interact with it. IRON is usually in sleep mode. Once the user says "IRON", it will respond with a "hello" message and then ask the user about what he wants. The user speaks the command, which is then analyzed to control the dedicated device. Algorithm 1 presents the summarized steps for interacting with the IRON system.

Algorithm 1: The IRON system procedure

Input: *user voice command*
Output: *device control*
Initialization: *IRON introduces itself and informs the user instructions about how interact with it.*
While true
Wait in sleep mode *#microphone ready to receive voice.*
If *the user says "IRON"*: *#user knows this from instruction.*
IRON say: hello message and ask human what he need to do.
Receive Order ()
Speech to text ()
Analysis Text ()
Execute the command ()
return feedback ()
Else:
Check if the user needs to end IRON or not.
End
End

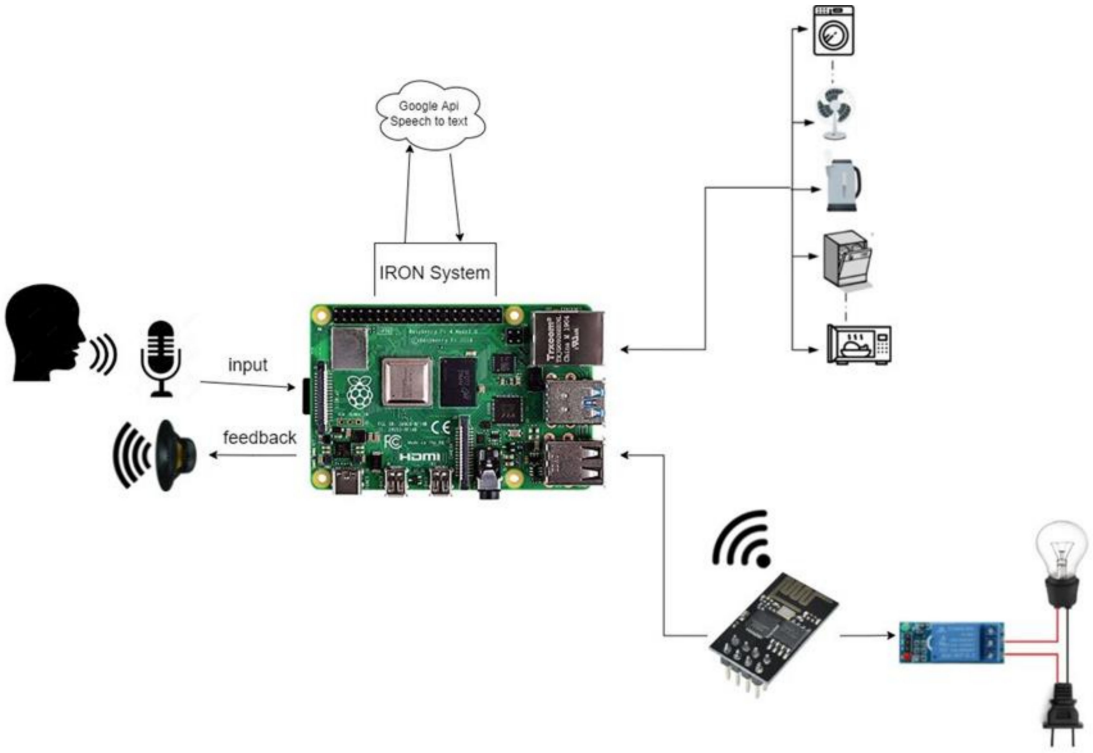


Figure 1. The proposed IRON system architecture.

Figure 2 introduces the IRON major modules and flowchart. The IRON system is designed from three modules: speech-to-text module, text analysis and classification module, and command execution module. In a speech-to-text module, Google API receives the user's voice command through the microphone and converts it to text. The text analysis and classification module receive the text and analyzes it to extract the device name and execute the required command. According to the device and the command, the command execution module applies an action on that device through Raspberry Pi. At the same time, the speaker responds and informs the user that the action is completed. The details of each module of the IRON are described in the following subsections.

- Module 1: Speech to text

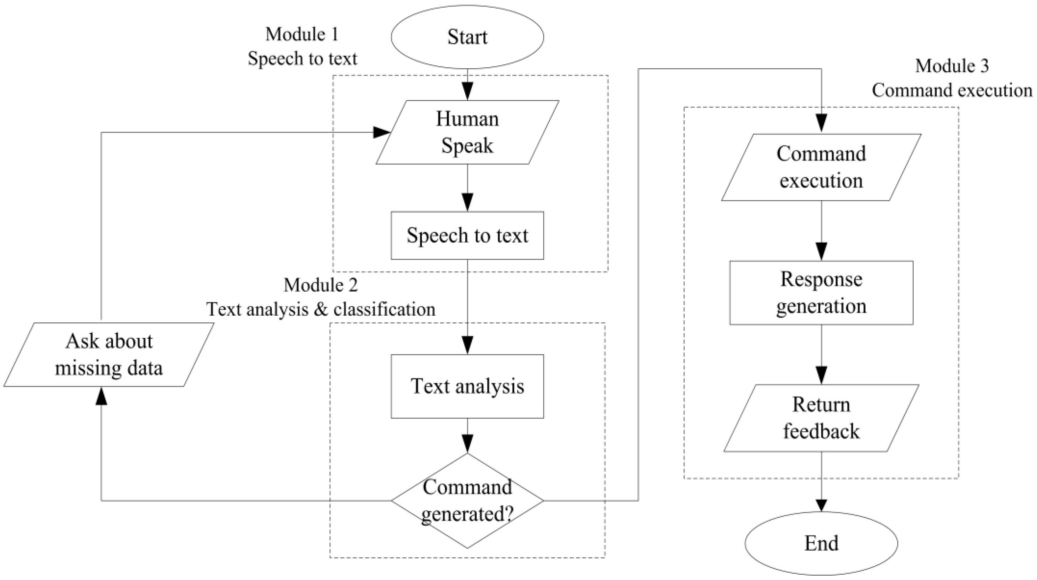


Figure 2. The IRON system flowchart.

In this module, the person's commands are converted from speech to text using the Google Speech to Text Cloud API. Google Speech to Text Cloud API transcribes the speech file using the most advanced deep learning neural network algorithms for automatic speech recognition (ASR) and returns the text statement [23]. Google Speech to Text Cloud API is one of the simplest methods for recognizing speech and can analyze up to 1 min of voice data.

- Module 2: Text analysis and classification

This module is responsible for understanding the correct command from the text generated by the Google API and then confirming it with a human to execute the desired action. Because of the uncertainties in human language, it is extremely challenging to create software that correctly ascertains the text's intended meaning, so NLP is used in this module for manipulating and recognizing the text. NLP deconstructs the text into small units to assist the computer in understanding the ingesting text. Different libraries and algorithms are proposed for NLP, such as the Natural Language Processing toolkit (NLTK) [24], Apache OpenNLP, Stanford CoreNLP, Pattern, GATE, and Spacy. Table 1 compares the most popular NLP techniques in terms of the programming language and the processes applied to the text statements. NLTK is considered the most robust and is the mother of all-natural language processing tools; thus, IRON utilizes it for analyzing the text. When IRON receives an input statement, tokenization, stemming, and stop word removal processes from the NLTK library are applied to the text.

Table 1. NLP toolkits Methods.

NLP Toolkit	Programming Language	Tokenization	Stemmer
NLTK	Python	Yes	Yes
Apache OpenNLP	Java	Yes	Yes
Stanford CoreNLP	Java	Yes	Yes
Pattern	Python	Yes	No
GATE	Java	Yes	No
Spacy	Python/Cython	Yes	No

Tokenization splits the text into smaller units called tokens to build a list representing all the vocabulary in the text. The tokens can be words, numbers, punctuation, stop words, and possessive markers. Moreover, the stemming step extracts the root word to reduce inflected words to their word base or root form [25]. It is a well-known fact that many words share morphological characteristics and semantic properties, making these words suitable for use in information retrieval applications. The stemmer removes the phrases to form the root term, such as "consider", which is the root term of consideration. To reduce processing time and supply more correct results from the enormous corpus of documents, several types of stemmers are employed to break down terms into their root terms. Searching is then performed on these terms rather than the real terms. In the stop words removal step, all the stop words encounter significant noise. These stop words, such as "the", "a", "an", and "in", do not add additional information, and removing them enhances the execution of the information retrieval system and text analytics [26].

After analyzing the text, splitting it into tokens, and removing the noise, IRON searches in the tokens list regarding the device name to know which device the user wants to control. When IRON determines the dedicated device, it should start to open or close the device according to the statement type. The statement type is deduced based on a trained classifier to recognize if the statement is positive or negative to switch on or off the device. The proposed IRON uses a logistic regression classification algorithm [27,28] to classify the rest of the tokens into positive and negative statements. The logistic regression classifier is an algorithm proposed for binary classification problems. It assesses the likelihood of the target variable, taking either 0 or 1. The classifier operates based on logistic regression, a statistical technique that makes predictions through the relationship between the dependent and independent variables. The logistic regression classifier model can be trained through maximum likelihood estimation or stochastic gradient descent methods. Additionally, the classifier includes regularization options L1 and L2 to counteract overfitting and enhance the model's generalization ability. The logistic regression equation is a mathematical formula that represents the relationship between the dependent and independent variables. It takes the form of a logistic function and can be written as:

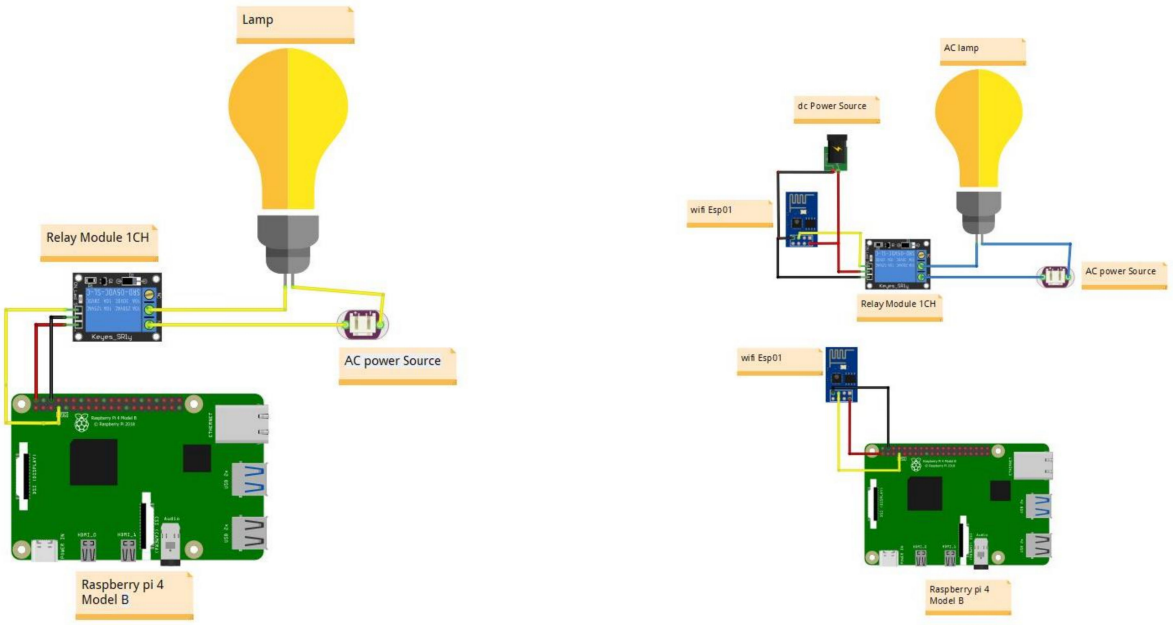
$$p = \frac{1}{1 + e^{-z}} \quad (1)$$

where p is the predicted probability of the dependent variable taking the value 1, and z is a linear combination of the independent variables and their coefficients. The coefficients are estimated during the training phase through optimization algorithms, such as maximum likelihood estimation or gradient descent. The resulting equation is then used to make predictions on new data. In the case of classifying positive and negative statements, the dependent variable would represent the statement, with 1 representing positive and 0 representing negative. The independent variables would be features of the statement, such as words used, tone, or other relevant information. The logistic regression classifier would estimate the impact of each feature on the statement through the coefficients for each feature. The classifier would then use the estimated coefficients and features of a new statement to determine a predicted probability of the statement being positive or negative. The prediction can be a thresholder, with a value greater than 0.5 considered positive and less than 0.5 considered negative.

- Module 3: Command execution

In this module, IRON executes the desired action via Raspberry Pi to turn on or off the dedicated device. The devices are connected to the Raspberry Pi according to their location from the main circuit. If the devices are close to the main circuit, they should be wired together; otherwise, they should be wirelessly connected using the Wi-Fi module ESP8266, as shown in the Figure 3a,b, respectively. When IRON sends an activated signal to a device, this signal is transmitted through the ESP8266 to turn on or off the relays which are connected to the device. The ESP8266 is selected as a communication module because it is a low-cost Wi-Fi microcontroller, and it supports both TCP/IP and Wi-Fi protocols.

Furthermore, it covers a range of up to 400 m in the open air, but it can be reduced to a few meters when obstacles such as walls or furniture are present.



(**a**) (**b**)

Figure 3. The types of device connection in the IRON: (**a**) wired; (**b**) wireless.

Figure 4 shows the detailed system flow chart, which presents the detailed steps of the proposed IRON system as follows:

Step 1: The user asks IRON to switch on or off a device.

Step 2: IRON receives the user voice command through high-quality BY-MA microphones distributed in different locations in the home to assist the impaired person in reaching it quickly.

Step 3: IRON transcribes the user's speech-to-text using the Google Speech-to-Text API, if the internet connection is available, or a speech-to-text conversion library, if the internet is disconnected.

Step 4: The transcribed statement is then processed using NLTK, creating a list of tokens. IRON searches the tokens list for the device name to determine which device the user wants to control. If the device name exists, IRON will continue; otherwise, IRON asks the user to speak the device name again.

Step 5: IRON checks the tokens list to determine if the statement is a question about the device status or a command. If the statement is a question, IRON responds with a proper answer about the device status and break; otherwise, IRON treats this statement as a command.

Step 6: After considering the statement as a command, the IRON removes the device name and classifies the rest of the tokens to a positive or negative command using logistic regression. If it is a positive command, Raspberry Pi sends "on" to the device through the GPIO pin; otherwise, it sends "off" to the device to execute this action.

Step 7: IRON generates a response text that matches the user text and uses the offline text-to-speech conversion library to convert this response into voice. IRON plays the generated voice on the YST-1052 speaker, which is connected to Raspberry Pi via USB and an audio jack to inform the user that action is applied to the device, such as, "OK Sir, living room light is on now".

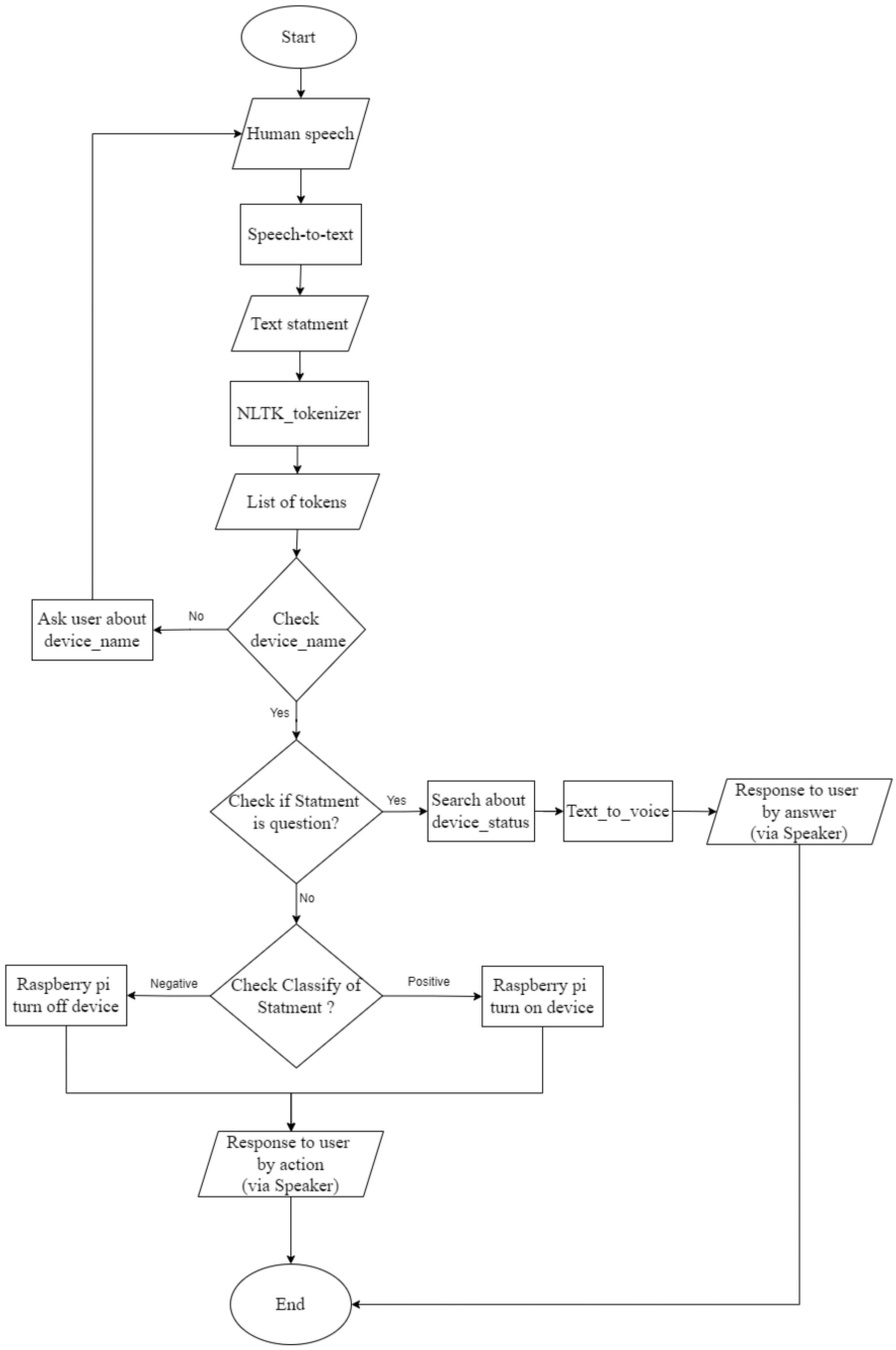


Figure 4. The detailed IRON system flowchart.

Over and above, IRON adds a feature for the user to add a new device via voice and configure it, as shown in Figure 5. First, the user connects the new device to the relay module unit. The user then asks IRON to turn on or off this new device. IRON checks whether this device is in the stored device list or not. If the device is not found in the devices list, IRON asks the user to confirm adding this device to the list. After that, IRON requests the device configuration from the user to add it to the file, as shown in Figure 6.

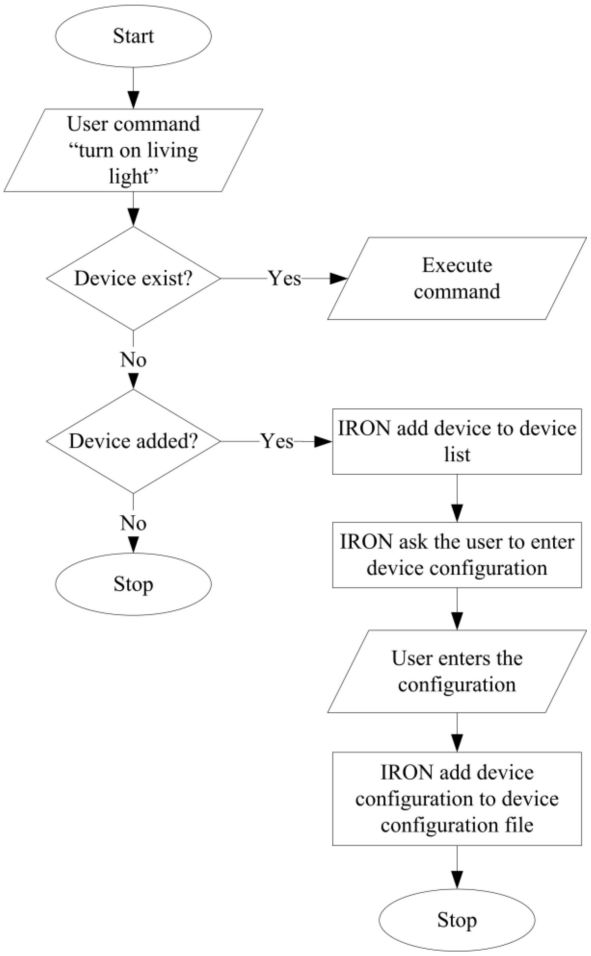


Figure 5. The new device's configuration.

```
{
    "devices": [
        {
            "device_name": "living room",
            "GPIO_pin": 2,
            "values": [0,1]
        },
        {
            "device_name": "air condition",
            "GPIO_pin": 4,
        "values": [18,19,20,21,22,23,24,25,26,27]
        }
    ]
}
```

Figure 6. Configuration file sample.

3. Evaluation Results

This section presents the evaluation metrics, dataset, and the simulation results of the proposed IRON system.

3.1. Evaluation Metrics

Three evaluation metrics, called accuracy, precision (AP), and recall (AR), are applied on the classification model output to evaluate the performance of the IRON system.

- Accuracy

Accuracy is a metric that measures the accuracy of the model in making the right prediction. It is computed as the percentage between the true predictions and the total number of predictions [29].

$$Accuracy = \frac{number\ of\ correct\ predictions}{total\ number\ of\ predictions} \tag{2}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3}$$

where TP = true positives, TN = true negatives, FP = false positives, and FN = false negatives.

- Precision

Precision measures the percentage of correctly predicted positive samples compared to the total number of predicted positive samples.

$$precison = \frac{TP}{TP + FP} \tag{4}$$

- Recall

Recall measures the ratio between the numbers of positive samples correctly predicted as positive to the total number of actual positive samples.

$$Recall = \frac{TP}{TP + FN} \tag{5}$$

3.2. Dataset

A dataset of 3600 commands is created to control 100 devices. These commands are divided into negative commands, unstructured commands, and normal commands. Negative commands are the direct commands to turn off the device, such as "Don't turn on room1 light" and "I didn't want to open TV". Unstructured commands are the indirect commands to turn on or off the device as "Please make light off" and "We need to open Fan". Normal commands are direct commands to turn on the device, such as, "Turn on light".

3.3. Simulation Results

The proposed IRON system is trained on an Intel(R) Core (TM) i5-5300U CPU running at 2.30 GHz, 2.29 GHz, and 16 GB of RAM, as well as an NVIDIA GeForce 830 with 2 GB of memory. The trained model is run on a Raspberry Pi and has a Broadcom BCM2711, which includes a quad-core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz with 2 GB Ram, 4 USB ports, and communication modules such as Bluetooth, wireless LAN, and an Ethernet port.

As mentioned earlier, the first step of the IRON is to detect the device name, then analyze the commands to determine the action. The proposed dataset is divided into 66% training, 17% validation, and 17% testing. IRON is trained on the proposed dataset with different machine learning algorithms: logistic regression, naïve Bayes, and support vector machine (SVM) to classify the commands for determining the action. Table 2 presents the percentage of the devices and the correctly detected and classified commands. It can be

noticed from the table that for all command types, IRON detects the device name with 90% accuracy. However, the percentage of classifying the commands correctly varies according to the command type. The higher classification percentage is for normal commands, while the lowest percentage is for unstructured commands.

Table 2. The percentage of correctly detected devices and classified commands.

Categories	Detected Devices	Correct Classified Commands
Normal commands	90%	93%
Unstructured commands	90%	80%
Negative commands	90%	83%

Table 3 introduces the performance of the IRON system for different machine learning algorithms. The IRON is trained on the smart home dataset [30] and the proposed dataset [31]. The smart home dataset consists of 663 commands. It can be observed from the results that the IRON based on the logistic regression and SVM achieves the best results, while naïve Bayes classify different commands as FP, which affects the precision values.

Table 3. The performance of the IRON system in terms of accuracy, precision, recall, and time.

Algorithm	Dataset	Accuracy	Precision	Recall	Time
Naive Bayes	Smart Home Commands Dataset	0.89	0.89	1.0	-
	The Proposed Dataset	0.63	0.59	0.64	0.0119
Logistic Regression	Smart Home Commands Dataset	0.969	0.967	1.0	-
	The Proposed Dataset	1.0	1.0	1.0	0.0112
SVM	Smart Home Commands Dataset	0.949	0.977	0.966	-
	The Proposed Dataset	1.0	1.0	1.0	0.020

For evaluating the processing time of each of the machine learning algorithms on the proposed dataset, the computational time is measured on the test set and presented in Table 3. The processing time of each technique is measured after running each technique 5 times, and the average of these 5 runs is calculated to obtain more reliable results. It can be demonstrated that the logistic regression consumes less time compared to naive Bayes and SVM.

The proposed IRON system is practically implemented, as presented in Figure 7. The hardware system consists of a lamp device connected with a relay module to Raspberry Pi, as shown in the Figure 7. The relay module transfers the signal from the Raspberry Pi to the lamp device. When the user says "IRON", the system is activated and gives instructions to guide the user on how to use IRON, as shown in Figure 8a. Afterward, the user starts to give the command to the system to control the device. If the user stops giving commands, the IRON goes into sleep mode and is deactivated until the user activates it again, as shown in Figure 8b. In the end, the user says, "Thank you", to turn off IRON. Additionally, the time required to execute a command in IRON is empirically calculated to be approximately 30 s.

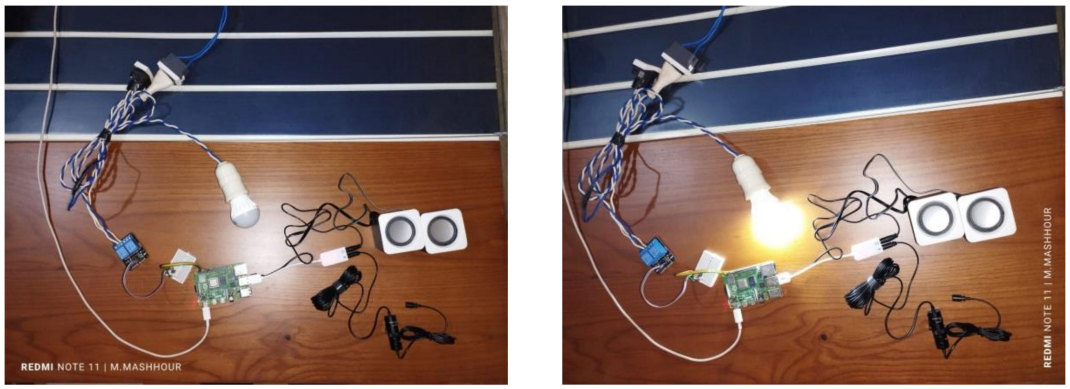


Figure 7. The IRON system Experimental setup.

```
iron:welcome,my name is IRON
iron:iam created by Engineer:mohamed mashhour
iron:to control in devices connected to IRON
iron:i will tell you instructions to interact with me:
iron:1- to close system at any time said :thank you
iron:2-to continue used IRON said: IRON
Speak .... :
user : iron
iron:how can i help you sir
Speak .... :
user : turn on light
iron:ok sir the light is on
Speak .... :
user : turn off light
iron:ok sir the light is off
Speak .... :
user : close the door
iron:ok sir the door is off
```

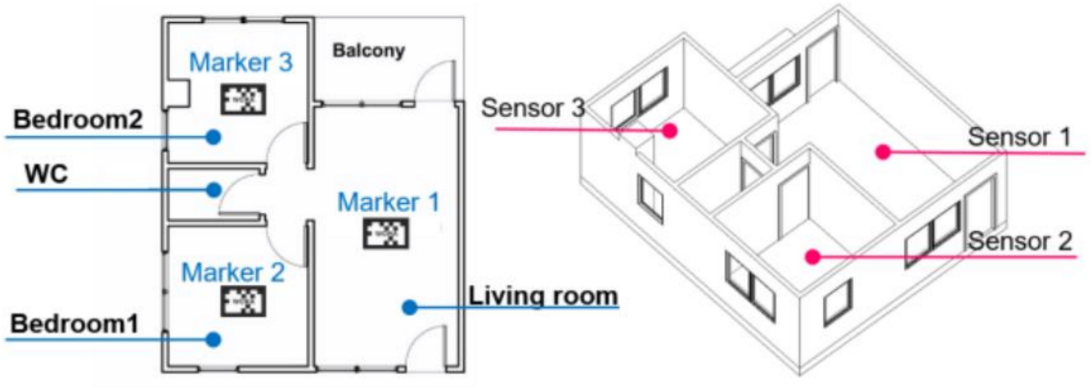
(**a**)

```
user : open the door
iron:ok sir the door is on
Speak .... :
user : what is the status of door
iron:Now the door is on
Speak .... :
Speak .... :
sleep_mode
Speak .... :
sleep_mode
Speak .... :
sleep_mode
Speak .... :
user : Iran
iron:how can i help you sir
Speak .... :
user : thank you Aaron
iron:i am happy to help you, sir
```

(**b**)

Figure 8. (**a**) IRON instructions, (**b**) IRON sleep mode.

Figure 9 clarifies how the sensors and devices are distributed in the home. The distribution varies according to the necessary purpose and the location of the devices to be controlled.



(**a**) (**b**)

Figure 9. Home sensors distribution: (**a**) floor plan and markers location setup, (**b**) 3D BIM model.

For evaluating the IRON system against the commercial intelligent personal assistant (IPA) systems, such as as Siri [16], Google Assistant [15], Alexa [17], Cortana [32], and Watson [33], Table 4 presents the comparison in terms of the speech recognition, text analysis, device detection, and negative and unstructured command understanding.

Table 4. Comparison between commercial IPA and IRON systems.

IPA	Speech Recognition	Text Analysis	Devices Detection	Negative and Unstructured Command Understanding
Apple Siri	Yes	Yes	No	Semi
Google Assistant	Yes	Yes	No	Semi
Amazon Alexa	Yes	Yes	Semi	Semi
Microsoft Cortana	Yes	Yes	No	Semi
IBM Watson	Yes	Yes	Yes	Semi
IRON system	Yes	Yes	Yes	Yes

4. Conclusions

The Internet of things, sensors, smartphones, smart appliances, and cloud computing have been used recently to transform ordinary homes into smart homes. These smart homes help persons with disabilities enhance their quality of life by remotely controlling the devices via mobile applications or their voices. This research proposes an intelligent personal assistant, called IRON, to help disabled people control their appliances with their voices. The IRON system architecture consists of three modules: a speech-to-text module, text analysis and classification module, and a command execution module, and is implemented on Raspberry Pi 3 model B. IRON receives the disabled person's voice commands through the microphone and transforms them into text using a speech to text module. The device name and the commands are extracted from the transformed text using NLP and logistic regression in the text analysis and classification module. According to the device name and the command, the IRON sends the required action to the dedicated device. IRON is built using different machine learning algorithms and is trained on 3600 commands to control 100 devices. The simulation results demonstrate that the logistic regression performs better than naïve Bayes regarding precision, recall, and accuracy. Furthermore, logistic regression requires less time to execute the commands than SVM and naive Bayes. The results also show that IRON can recognize different commands as structured and unstructured statements. IRON will be updated in future work to help people remember their notes and dates, in addition to controlling home appliances. Additionally, IRON can automatically regulate the devices according to some conditions (e.g., change the air condition temperature if the weather changes). Moreover, the user can schedule different commands that IRON can execute at a specific time under certain conditions.

Author Contributions: Conceptualization, M.M., and A.-e.A.A.; methodology, M.M. and H.S.; software, M.M.; validation, H.S. and A.S.S.; formal analysis, R.S.; investigation, M.M., R.S. and A.S.S.; resources, A.-e.A.A. and A.S.S.; data curation, M.M.; writing—original draft preparation, M.M., H.S. and R.S.; writing—review and editing, R.S. and M.M.; visualization, A.S.S., H.S. and A.-e.A.A.; supervision, A.-e.A.A. and H.S.; project administration, A.S.S., H.S. and R.S.; funding acquisition, A.S.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Future University in Egypt (FUE) research center.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The datasets used in this investigation are available on request from the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

IPA	intelligent personal assistant
GUI	graphical user interface
SHAS	smart home automation system
CFL	compact fluorescent lamp
GPIO	general purpose input output
VAD	voice activity detection
CANVAS	context awareness for voice assistants
HAS	home automation system
MFCC	mel-frequency cepstral coefficients
VQ	vector quantization
PCA	principal component analysis
GMM	Gaussian mixture model
NLTK	Natural Language Toolkit
SVM	support vector machine

References

1. World Health Organization (n.d.). WHO. Available online: https://www.who.int/en/ (accessed on 9 March 2023).
2. Li, G.; Cheng, L.; Gao, Z.; Xia, X.; Jiang, J. Development of an untethered adaptive thumb exoskeleton for delicate rehabilitation assistance. *IEEE Trans. Robot.* **2022**, *38*, 3514–3529. [CrossRef]
3. Abdelhamid, S.; Aref, M.; Hegazy, I.; Roushdy, M. A survey on learning-based intrusion detection systems for IoT networks. In Proceedings of the 2021 Tenth International Conference on Intelligent Computing and Information Systems (ICICIS), Cairo, Egypt, 5–7 December 2021; pp. 278–288. [CrossRef]
4. Omran, M.A.; Saad, W.K.; Hamza, B.J.; Fahe, A. A survey of various intelligent home applications using IoT and intelligent controllers. *Indones. J. Electr. Eng. Comput. Sci.* **2021**, *23*, 490–499. [CrossRef]
5. Mittal, Y.; Toshniwal, P.; Sharma, S.; Singhal, D.; Gupta, R.; Mittal, V.K. A voice-controlled multi-functional smart home automation system. In Proceedings of the 2015 Annual IEEE India Conference (INDICON), New Delhi, India, 17–20 December 2015; pp. 1–6. [CrossRef]
6. Abidi, M.E.; Asnawi, A.L.; Azmin, N.F.; Jusoh, A.Z.; Ibrahim, S.N.; Mohd Ramli, H.A.; Abdul Malek, N. Development of voice control and home security for smart home automation. In Proceedings of the 2018 7th International Conference on Computer and Communication Engineering (ICCCE), Kuala Lumpur, Malaysia, 19–20 September 2018; pp. 1–6. [CrossRef]
7. Mayer, J. IoT architecture for home automation by speech control aimed to assist people with mobility restrictions. In *International Conference on Internet Computing (ICOMP)*; The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp)—CSREA: Providence, RI, USA, 2017; pp. 92–97.
8. Saha, I.; Maheswari, S. Deep learning for voice control home automation with auto mode. In Proceedings of the International Conference on Computational Vision and Bio Inspired Computing, Coimbatore, India, 29–30 November 2018; pp. 1605–1614. [CrossRef]
9. Ismaeel, A.G. Home automation system support for persons with different types of disabilities using IoT. In Proceedings of the IMDC-SDSP 2020: 1st International Multi-Disciplinary Conference Theme: Sustainable Development and Smart Planning, Cyberspace, Ghent, Belgium, 28–30 June 2020; p. 18. [CrossRef]
10. Jat, D.S.; Limbo, A.S.; Singh, C. Voice activity detection-based home automation system for people with special needs. In *Intelligent Speech Signal Processing*; Academic Press: Cambridge, MA, USA, 2019; pp. 101–111. [CrossRef]
11. Alrumayh, A.S.; Lehman, S.M.; Tan, C.C. Context aware access control for home voice assistant in multi-occupant homes. *Pervasive Mob. Comput.* **2020**, *67*, 101196. [CrossRef]
12. Khan, M.A.; Ahmad, I.; Nordin, A.N.; Ahmed, A.E.-S.; Mewada, H.; Daradkeh, Y.I.; Rasheed, S.; Eldin, E.T.; Shafiq, M. Smart Android Based Home Automation System Using Internet of Things (IoT). *Sustainability* **2022**, *14*, 717. [CrossRef]
13. Omran, M.A.; Hamza, B.J.; Saad, W.K. The design and fulfillment of a Smart Home (SH) material powered by the IoT using the Blynk app. *Mater. Today Proc.* **2022**, *60*, 1199–1212. [CrossRef]
14. de Barcelos Silva, A.; Gomes, M.M.; da Costa, C.A.; da Rosa Righi, R.; Barbosa, J.L.V.; Pessin, G.; Federizzi, G. Intelligent personal assistants: A systematic literature review. *Expert Syst. Appl.* **2020**, *147*, 113193. [CrossRef]
15. Google Assistant (n.d.). Available online: https://assistant.google.com/ (accessed on 9 March 2023).
16. Apple Inc. (n.d.). *Siri*. Available online: https://www.apple.com/siri/ (accessed on 9 March 2023).
17. Amazon.com Inc. (n.d.). *Alexa Developer*. Available online: https://developer.amazon.com/en-US/alexa (accessed on 9 March 2023).
18. Uma, S.; Eswari, R.; Bhuvanya, R.; Kumar, G.S. IoT based voice/text controlled home appliances. *Procedia Comput. Sci.* **2019**, *165*, 232–238. [CrossRef]
19. Kumer, S.A.; Kanakaraja, P.; Teja, A.P.; Sree, T.H.; Tejaswni, T. Smart home automation using IFTTT and google assistant. *Mater. Today Proc.* **2021**, *46*, 4070–4076. [CrossRef]

20. Putthapipat, P.; Woralert, C.; Sirinimnuankul, P. Speech recognition gateway for home automation on open platform. In Proceedings of the 2018 International Conference on Electronics Information and Communication (ICEIC), Honolulu, HI, USA, 24–27 January 2018; pp. 1–4. [CrossRef]
21. Tiwari, V.; Hashmi, M.F.; Keskar, A.; Shivaprakash, N.C. Virtual home assistant for voice based controlling and scheduling with short speech speaker identification. *Multimed. Tools Appl.* **2020**, *79*, 5243–5268. [CrossRef]
22. Mehrabani, M.; Bangalore, S.; Stern, B. Personalized speech recognition for Internet of Things. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP), Lisbon, Portugal, 19–21 September 2015; pp. 1836–1841.
23. Google (n.d.). *Speech-to-Text API Requests*. Available online: https://cloud.google.com/speech-to-text/docs/speech-to-text-requests#overview (accessed on 9 March 2023).
24. Bird, S.; Loper, E.; Klein, E. Natural Language Toolkit (NLTK) (2009–2020). Available online: https://www.nltk.org/ (accessed on 9 March 2023).
25. Haroon, M. Comparative analysis of stemming algorithms for web text mining. *J. Adv. Inf. Technol.* **2018**, *9*, 20–25. [CrossRef]
26. Raulji, J.K.; Saini, J.R. Stop-word removal algorithm and its implementation for Sanskrit language. *Int. J. Comput. Appl.* **2016**, *150*, 15–17.
27. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Vanderplas, J. Logistic Regression—Scikit-Learn 1.0 Documentation (2011–2021). Available online: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html (accessed on 9 March 2023).
28. IBM (n.d.). *Logistic Regression—Overview and Use Cases*. Available online: https://www.ibm.com/topics/logistic-regression (accessed on 9 March 2023).
29. Google (n.d.). *Classification: Accuracy*. Available online: https://developers.google.com/machine-learning/crash-course/classification/accuracy (accessed on 9 March 2023).
30. Ceunen, B. Smart Home Commands Dataset. 2021. Available online: https://www.kaggle.com/datasets/bouweceunen/smart-home-commands-dataset (accessed on 9 March 2023).
31. Mashhour, M. Mashhour Dataset. 2021. Available online: https://www.kaggle.com/datasets/mohamedmashhour/mashhour-dataset (accessed on 9 March 2023).
32. Microsoft (n.d.). *Cortana*. Available online: https://www.microsoft.com/en-us/cortana (accessed on 9 March 2023).
33. IBM (n.d.). *Watson Assistant*. Available online: https://www.ibm.com/products/watson-assistant (accessed on 9 March 2023).