International Journal on Emerging Technologies **10**(1): 139-142(2019)

ISSN No. (Print) : 0975-8364
ISSN No. (Online) : 2249-3255

The Virtual Personal Assistant (VPA) - Learning Process

K. Rajkumar[1], B. Monica[1], K.R. Sekar[1], R. Manikandan[1] and Robbi Rahim[2]
[1]School of Computing, SASTRA Deemed University, (Tamil Nadu), India.
[2]Department of Management, Sekolah Tinggi Ilmu Manajemen Sukma, Medan, Indonesia

(Corresponding author: K. Rajkumar)

(Received 12 March 2019, Revised 21 May 2019 Accepted 29 May 2019)
(Published by Research Trend, Website: www.researchtrend.net)

ABSTRACT: Assisting the computer users in performing their tasks is a very important issue in the field of Human Computer Interaction. A way to deal with this sort of challenge is to build a virtual personal assistant (VPA) which is capable of simplifying user's work in performing those tasks. The objective of this work is to build a Virtual Personal Assistant or Intelligent Personal Assistants are self employed and it can work on behalf of the human beings while it is in the remote area. The mobile applications which can help the user perform his desired task as per the request. The natural language process is much appreciated for VPA to do all actions in the absence of humans. This approach is literally meant for more work with less cost and moreover the VPA is available for full 24×7 Hours. The VPA will increase the business and provides the heavy growth to the company people round the clock. Inter and intra personal skill can be improved through VPA.

Keywords*:* Virtual Personal Assistant, Human Computer Interface, Intelligent Personal Assistant, Natural Language Processing, Inter Personal skill and intra personal skill.

I. INTRODUCTION

Software systems are growing complex day by day, while the potential and readiness of users to cope up with that complexity is constantly decreasing. This difficulty affects all the products present in the markets. A refined solution is to endow these products with in built assistants that can address the user issues at the interface level. These inbuilt assistants are known as virtual personal assistants.

Virtual personal assistants (VPAs) are computerized systems that can interact with the device users in a conversational manner. To do this, the VPA has to correctly interpret conversational user input, execute a task on behalf of the user, determine an appropriate response to the input and present the response in such a way that the user can understand it. A platform for developing a virtual personal assistant (VPA) application includes an organization that defines a computerized structure for representing knowledge relating to one or more domains. A domain may refer to a class of information and/or activities in relation to which the VPA application may involve in a natural language conversation with a computing device user. The assistant also improves its behaviour based on previous experiences. By empowering the assistant with the learning capability, it will be able to adapt itself to the user's behaviour. The system acquires knowledge about the steps that the user performs and stores it in its database, such that the same task can be performed by the system automatically from next time according to the user's request. This paper proposes the development of a virtual personal assistant that learns by the actions of the users in order to perform specific tasks.

The rest of the paper is divided as follows, Section II handles the existing literature, and Section III summarizes the basic concepts of implementation. Section IV describes the learning process achieved by the VPA by two means using the Usage Stats Manager class and Activity Info class. Section V handles the Results and Discussion, lastly Section VI describes future work.

II. LITERATURE REVIEW

Virtual personal assistants (VPAs) are computerized systems that can interact with the device users in a conversational manner [1]. The interaction between them can happen through a series of three major activities such as entry of data, formulating the policies and final making decisions/recommendations [2]. VPA plays a main role among the computer users and the active agents that are present in an environment.

Artificial Intelligence employs series of logical steps known as algorithms and well advanced cognitive techniques [3]. This is an interdisciplinary domain used for identification and responding to artificial commands and has a varied scope of data with respect to advancements in various fields as it has the ability to simultaneously reason and process the natural language to enable human and computer communication.

NLP is process of examination of linguistic data which is mostly text data like publications, documents etc. by employing various computational techniques. The major task involved in NLP is creating a model of text which is done by the inclusion of structural additions as well as the insights provided by linguistics [4-5]. It is the most important building block to develop computer software capable of enabling interaction between humans and computer to save data initially, to solve particular problems and perform those jobs repetitively as demanded by users of the software [6]. Natural language examination must be able to check the sentences utilized in natural language so that spelling mistakes are corrected to develop a syntactic form of sentences, to give semantic relationships and to integrate the syntactic form of sentences [7] and semantic relationships to respond appropriately.

III. IMPLEMENATION

The virtual personal assistant is designed in such a way that it can interact with the user in a conversational manner. By using Bing Speech API we have developed an android application that makes use of the Microsoft cognitive servers to convert the speech into text.[8] The API supports real-time streaming, so that the application can simultaneously receive partial speech based text recognition results and at the same time it is also sending the received audio to the service used. Once the speech is converted to text, the response is used to perform the desired task by using the keywords (Eg: "call person", where call is a keyword) present in the sentence.

Every application must have an AndroidManifest.xml file (in the same name precisely) in the application's root directory [9]. The android manifest file provides all the essential information about the application to the Android system, which the system must have for itself so that it can run any code of the application [10]. This file declares all the permissions that the application can have so that to access the protected parts of the API and interact with all other applications it is meant to. It also declares those permissions that others should also have in order to interact with the application's components.

A. Camera

In the already available VPAs that support the working with camera only assists the user in opening the camera application. But we can use the classes and interfaces provided by android developer, we can implement many features by integrating it in the VPA to make it responsive to capture a picture after some seconds.

B. Learning process

Generally for an application to work along with the available VPAs, it needs to be integrated by the developer of that application into that VPA. But here, the application uses a different way of integrating itself with the virtual personal assistant. Learning process is achieved by the following means:

(i) Using the UsageStatsManager

(ii) Using the ActivityInfo

It is based on these two methods the learning process is carried out further. The UsageStatsManager class provides access to mobile usage statistics and history. On the other hand, the

ActivityInfo class provides all the information retrieved about a particular application receiver or activity.

(i) Using the UsageStatsManager

This provides access to mobile usage history and statistics. The usage data is categorized into time interval format (i.e.) days, weeks, months, and years.

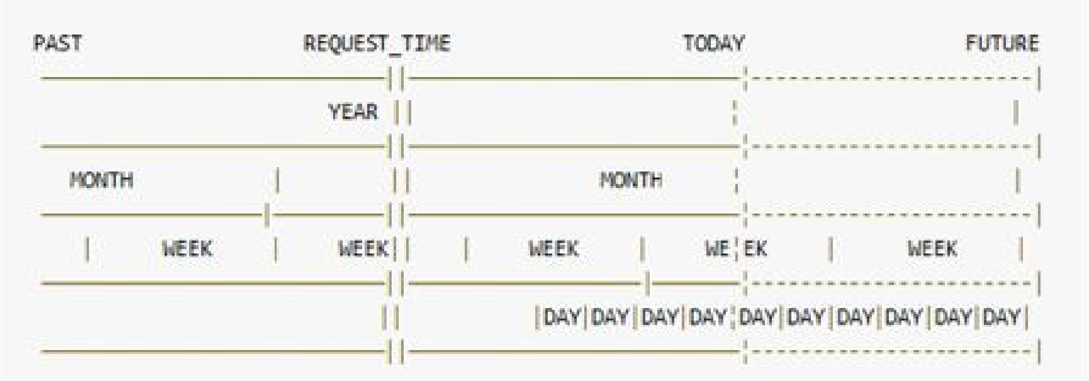
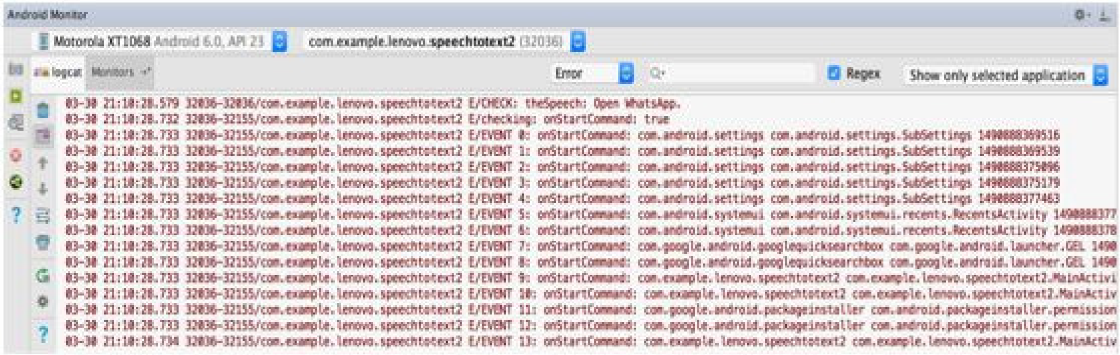


Fig. 1. Mobile usage history.

This requires the permission android. Permission. PACKAGE_USAGE_STATS, to be enabled in order to access those rights and this will never be granted to the third-party applications. However, declaring these permissions implies the intention to use the particular API and the mobile user can grant the permission through the Settings option in the mobile [11]. Consider the case that the user wants to open an application which is not known by the VPA. This implies that the user wants to perform that particular task for the very first time. In this case when the user commands the VPA to do the task, it fails. Here the user commands the VPA to open WhatsApp but it fails to do so.


Android Monitor
Motorola XT1068 Android 6.0, API 23
com.example.lenovo.speechtotext2 (32036)
logcat
Monitors
Error
Regex
Show only selected application

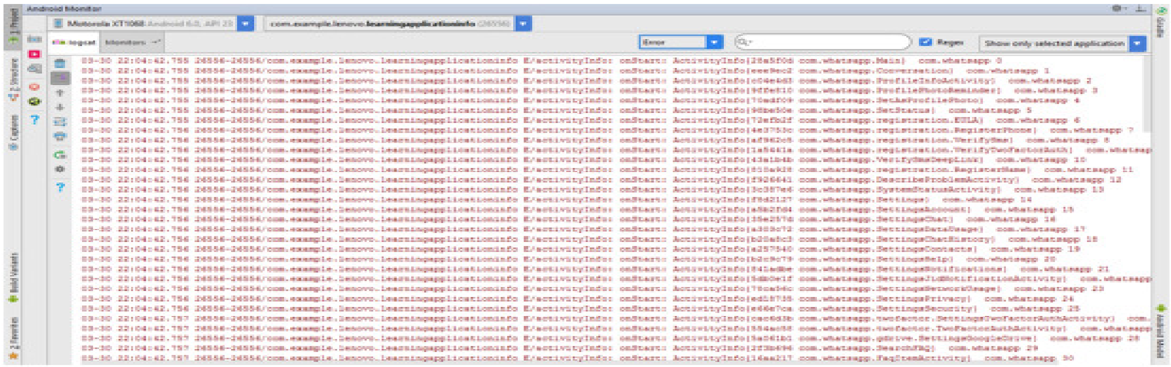
```
03-30 21:10:28.579 32036-32036/com.example.lenovo.speechtotext2 E/CHECK: theSpeech: Open WhatsApp.
03-30 21:10:28.732 32036-32155/com.example.lenovo.speechtotext2 E/checking: onStartCommand: true
03-30 21:10:28.733 32036-32155/com.example.lenovo.speechtotext2 E/EVENT 0: onStartCommand: com.android.settings com.android.settings.SubSettings 1490888369516
03-30 21:10:28.733 32036-32155/com.example.lenovo.speechtotext2 E/EVENT 1: onStartCommand: com.android.settings com.android.settings.SubSettings 1490888369539
03-30 21:10:28.733 32036-32155/com.example.lenovo.speechtotext2 E/EVENT 2: onStartCommand: com.android.settings com.android.settings.SubSettings 1490888375096
03-30 21:10:28.733 32036-32155/com.example.lenovo.speechtotext2 E/EVENT 3: onStartCommand: com.android.settings com.android.settings.SubSettings 1490888375179
03-30 21:10:28.733 32036-32155/com.example.lenovo.speechtotext2 E/EVENT 4: onStartCommand: com.android.settings com.android.settings.SubSettings 1490888377463
03-30 21:10:28.733 32036-32155/com.example.lenovo.speechtotext2 E/EVENT 5: onStartCommand: com.android.systemui com.android.systemui.recents.RecentsActivity 1490888377
03-30 21:10:28.733 32036-32155/com.example.lenovo.speechtotext2 E/EVENT 6: onStartCommand: com.android.systemui com.android.systemui.recents.RecentsActivity 1490888378
03-30 21:10:28.733 32036-32155/com.example.lenovo.speechtotext2 E/EVENT 7: onStartCommand: com.google.android.googlequicksearchbox com.google.android.launcher.GEL 1490
03-30 21:10:28.733 32036-32155/com.example.lenovo.speechtotext2 E/EVENT 8: onStartCommand: com.google.android.googlequicksearchbox com.google.android.launcher.GEL 1490
03-30 21:10:28.733 32036-32155/com.example.lenovo.speechtotext2 E/EVENT 9: onStartCommand: com.example.lenovo.speechtotext2 com.example.lenovo.speechtotext2.MainActivi
03-30 21:10:28.733 32036-32155/com.example.lenovo.speechtotext2 E/EVENT 10: onStartCommand: com.example.lenovo.speechtotext2 com.example.lenovo.speechtotext2.MainActiv
03-30 21:10:28.733 32036-32155/com.example.lenovo.speechtotext2 E/EVENT 11: onStartCommand: com.google.android.packageinstaller com.android.packageinstaller.permission
03-30 21:10:28.733 32036-32155/com.example.lenovo.speechtotext2 E/EVENT 12: onStartCommand: com.google.android.packageinstaller com.android.packageinstaller.permission
03-30 21:10:28.734 32036-32155/com.example.lenovo.speechtotext2 E/EVENT 13: onStartCommand: com.example.lenovo.speechtotext2 com.example.lenovo.speechtotext2.MainActiv
```

Fig. 2. VPA performance activity.

So now it's the job of the user to help the VPA learn to perform those tasks by performing it once for the first time. When the user does that task manually, the related information is created as logs in the UsageStats for that that particular UsageEvent before asking the request to the VPA [12]. Once this is done, this action will be stored in the android database so that this task can be performed by the VPA on its own from the next request. For this to be successful that opening the application action should be present in the immediate last 60 seconds of the log so that the VPA can learn from it.

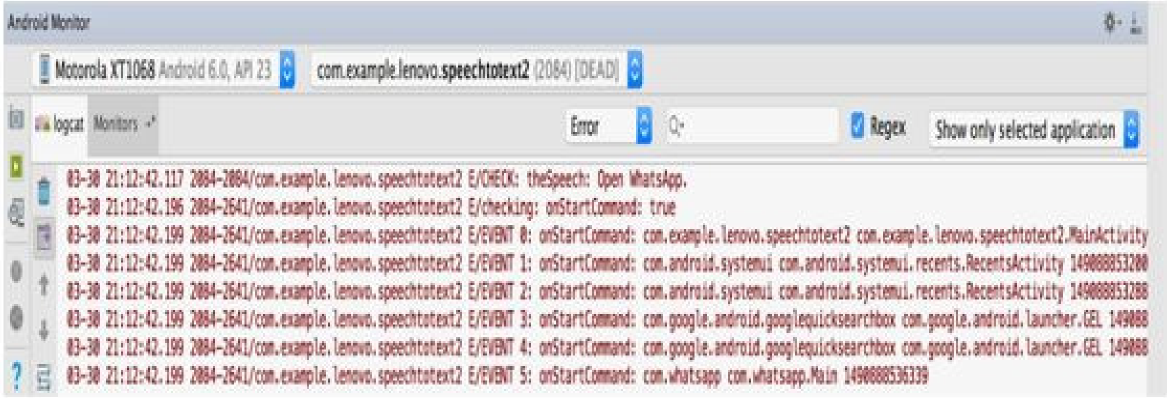
(i) Using the ActivityInfo

This is the information one can retrieve or collect about a specific application activity. This collected Information is from the AndroidManifest.xml's activity tag. The information about the activity is retrieved in a hierarchical way like the ActivityInfo is a subclass of ComponentInfo which in turn is the subclass of PackageInfo. ComponentInfo is the class all the information common to all application components. Its need is to share the common definitions with all components of the application.The VPA uses ActivityInfo to learn automatically by itself by getting the activity information of other applications and by starting activities if those applications. Consider the case of WhatsApp, the application has so many activities in it like the ProfileInfoActivity, SetStatus etc.


Android Monitor
Motorola XT1068 Android 6.0, API 23
com.example.lenovo.learningapplicationinfo (24556)
logcat
Monitors
Error
Regex
Show only selected application

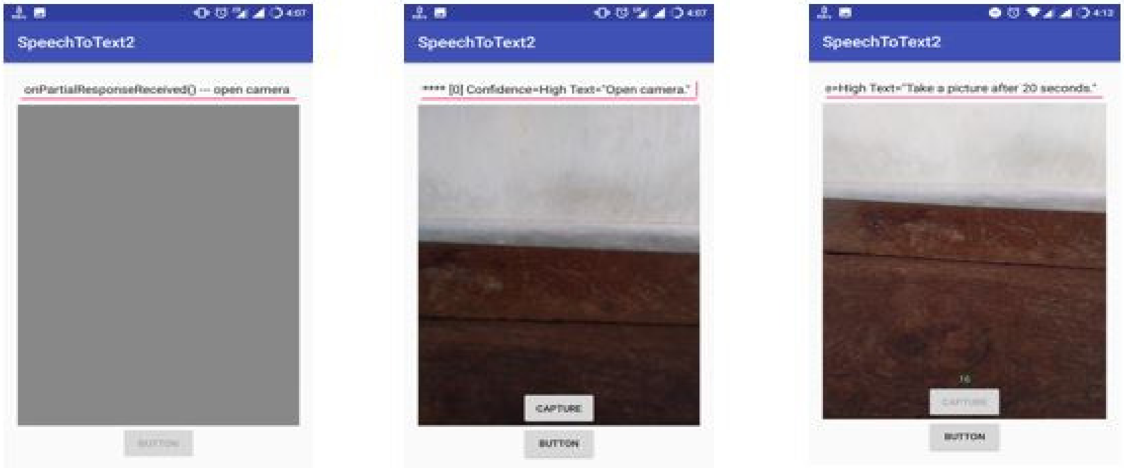
```
03-30 22:04:42.755 24556-24556/com.example.lenovo.learningapplicationinfo E/activityInfo: onStart: ActivityInfo{28a5f0d com.whatsapp.Main}  com.whatsapp 0
03-30 22:04:42.755 24556-24556/com.example.lenovo.learningapplicationinfo E/activityInfo: onStart: ActivityInfo{eee9ec2 com.whatsapp.Conversation}  com.whatsapp 1
03-30 22:04:42.755 24556-24556/com.example.lenovo.learningapplicationinfo E/activityInfo: onStart: ActivityInfo{c04e4d5 com.whatsapp.ProfileInfoActivity}  com.whatsapp 2
03-30 22:04:42.755 24556-24556/com.example.lenovo.learningapplicationinfo E/activityInfo: onStart: ActivityInfo{9ffe810 com.whatsapp.ProfilePhotoReminder}  com.whatsapp 3
03-30 22:04:42.755 24556-24556/com.example.lenovo.learningapplicationinfo E/activityInfo: onStart: ActivityInfo{70adf09 com.whatsapp.SetAsProfilePhoto}  com.whatsapp 4
03-30 22:04:42.755 24556-24556/com.example.lenovo.learningapplicationinfo E/activityInfo: onStart: ActivityInfo{98be50e com.whatsapp.SetStatus}  com.whatsapp 5
03-30 22:04:42.755 24556-24556/com.example.lenovo.learningapplicationinfo E/activityInfo: onStart: ActivityInfo{72efb2f com.whatsapp.registration.EULA}  com.whatsapp 6
03-30 22:04:42.756 24556-24556/com.example.lenovo.learningapplicationinfo E/activityInfo: onStart: ActivityInfo{4e3753c com.whatsapp.registration.RegisterPhone}  com.whatsapp 7
03-30 22:04:42.756 24556-24556/com.example.lenovo.learningapplicationinfo E/activityInfo: onStart: ActivityInfo{af962c5 com.whatsapp.registration.VerifySms}  com.whatsapp 8
03-30 22:04:42.756 24556-24556/com.example.lenovo.learningapplicationinfo E/activityInfo: onStart: ActivityInfo{1a5441a com.whatsapp.registration.VerifyTwoFactorAuth}  com.whatsap
03-30 22:04:42.756 24556-24556/com.example.lenovo.learningapplicationinfo E/activityInfo: onStart: ActivityInfo{43a1b4b com.whatsapp.VerifySmsDeepLink}  com.whatsapp 10
03-30 22:04:42.756 24556-24556/com.example.lenovo.learningapplicationinfo E/activityInfo: onStart: ActivityInfo{615a928 com.whatsapp.registration.RegisterName}  com.whatsapp 11
03-30 22:04:42.756 24556-24556/com.example.lenovo.learningapplicationinfo E/activityInfo: onStart: ActivityInfo{f926641 com.whatsapp.DescribeProblemActivity}  com.whatsapp 12
03-30 22:04:42.756 24556-24556/com.example.lenovo.learningapplicationinfo E/activityInfo: onStart: ActivityInfo{3c387e6 com.whatsapp.SystemStatusActivity}  com.whatsapp 13
03-30 22:04:42.756 24556-24556/com.example.lenovo.learningapplicationinfo E/activityInfo: onStart: ActivityInfo{f8d2127 com.whatsapp.Settings}  com.whatsapp 14
03-30 22:04:42.756 24556-24556/com.example.lenovo.learningapplicationinfo E/activityInfo: onStart: ActivityInfo{a5b2fd4 com.whatsapp.SettingsAccount}  com.whatsapp 15
03-30 22:04:42.756 24556-24556/com.example.lenovo.learningapplicationinfo E/activityInfo: onStart: ActivityInfo{35e2578 com.whatsapp.SettingsChat}  com.whatsapp 16
03-30 22:04:42.756 24556-24556/com.example.lenovo.learningapplicationinfo E/activityInfo: onStart: ActivityInfo{a303c72 com.whatsapp.SettingsDataUsage}  com.whatsapp 17
03-30 22:04:42.756 24556-24556/com.example.lenovo.learningapplicationinfo E/activityInfo: onStart: ActivityInfo{b20a6c3 com.whatsapp.SettingsChatHistory}  com.whatsapp 18
03-30 22:04:42.756 24556-24556/com.example.lenovo.learningapplicationinfo E/activityInfo: onStart: ActivityInfo{a257540 com.whatsapp.SettingsContacts}  com.whatsapp 19
03-30 22:04:42.756 24556-24556/com.example.lenovo.learningapplicationinfo E/activityInfo: onStart: ActivityInfo{b2c9c79 com.whatsapp.SettingsHelp}  com.whatsapp 20
03-30 22:04:42.756 24556-24556/com.example.lenovo.learningapplicationinfo E/activityInfo: onStart: ActivityInfo{841adbe com.whatsapp.SettingsNotifications}  com.whatsapp 21
03-30 22:04:42.756 24556-24556/com.example.lenovo.learningapplicationinfo E/activityInfo: onStart: ActivityInfo{5db0e1f com.whatsapp.SettingsJidNotificationActivity}  com.whatsapp
03-30 22:04:42.756 24556-24556/com.example.lenovo.learningapplicationinfo E/activityInfo: onStart: ActivityInfo{75ca54c com.whatsapp.SettingsNetworkUsage}  com.whatsapp 23
03-30 22:04:42.756 24556-24556/com.example.lenovo.learningapplicationinfo E/activityInfo: onStart: ActivityInfo{ed1f735 com.whatsapp.SettingsPrivacy}  com.whatsapp 24
03-30 22:04:42.756 24556-24556/com.example.lenovo.learningapplicationinfo E/activityInfo: onStart: ActivityInfo{e46e7ca com.whatsapp.SettingsSecurity}  com.whatsapp 25
03-30 22:04:42.757 24556-24556/com.example.lenovo.learningapplicationinfo E/activityInfo: onStart: ActivityInfo{cac6d3b com.whatsapp.twofactor.SettingsTwoFactorAuthActivity}  com.
03-30 22:04:42.757 24556-24556/com.example.lenovo.learningapplicationinfo E/activityInfo: onStart: ActivityInfo{554ac58 com.whatsapp.twofactor.TwoFactorAuthActivity}  com.whatsapp
03-30 22:04:42.757 24556-24556/com.example.lenovo.learningapplicationinfo E/activityInfo: onStart: ActivityInfo{5a061b1 com.whatsapp.gdrive.SettingsGoogleDrive}  com.whatsapp 28
03-30 22:04:42.757 24556-24556/com.example.lenovo.learningapplicationinfo E/activityInfo: onStart: ActivityInfo{2f3b496 com.whatsapp.SearchFAQ}  com.whatsapp 29
03-30 22:04:42.757 24556-24556/com.example.lenovo.learningapplicationinfo E/activityInfo: onStart: ActivityInfo{14aa217 com.whatsapp.FaqItemActivity}  com.whatsapp 30
```

Fig. 3. Profile Info Activity.


Android Monitor
Motorola XT1068 Android 6.0, API 23
com.example.lenovo.speechtotext2 (2084) [DEAD]
logcat
Monitors
Error
Regex
Show only selected application

```
03-30 21:12:42.117 2084-2084/com.example.lenovo.speechtotext2 E/CHECK: theSpeech: Open WhatsApp.
03-30 21:12:42.196 2084-2641/com.example.lenovo.speechtotext2 E/checking: onStartCommand: true
03-30 21:12:42.199 2084-2641/com.example.lenovo.speechtotext2 E/EVENT 0: onStartCommand: com.example.lenovo.speechtotext2 com.example.lenovo.speechtotext2.MainActivity
03-30 21:12:42.199 2084-2641/com.example.lenovo.speechtotext2 E/EVENT 1: onStartCommand: com.android.systemui com.android.systemui.recents.RecentsActivity 1490880853280
03-30 21:12:42.199 2084-2641/com.example.lenovo.speechtotext2 E/EVENT 2: onStartCommand: com.android.systemui com.android.systemui.recents.RecentsActivity 1490880853288
03-30 21:12:42.199 2084-2641/com.example.lenovo.speechtotext2 E/EVENT 3: onStartCommand: com.google.android.googlequicksearchbox com.google.android.launcher.GEL 149088
03-30 21:12:42.199 2084-2641/com.example.lenovo.speechtotext2 E/EVENT 4: onStartCommand: com.google.android.googlequicksearchbox com.google.android.launcher.GEL 149088
03-30 21:12:42.199 2084-2641/com.example.lenovo.speechtotext2 E/EVENT 5: onStartCommand: com.whatsapp com.whatsapp.Main 1490880536339
```

Fig. 4. Camera Activity Processing.



onPartialResponseReceived() --- open camera

**** [0] Confidence=High Text="Open camera."

e=High Text="Take a picture after 20 seconds."

Fig. 5. Results from VPA.

The problem is that the VPA cannot start all the activities in an application because to start an activity its corresponding attribute “android: exported” should be set as true. If it is set as false, it can only be launched by the components of that application and not by the VPA. The “android: exported” field will be declared true for an activity by the developer. For example, the VPA can launch the camera activity of WhatsApp but it cannot launch PrivacyInfo activity because the “android: exported” field of Privacy Info activity is set false.

IV. RESULT AND DISCUSSION

The virtual personal assistant is able to take the user’s voice as input text with the help of Bing Speech and is able to clearly understand the spoken words with the help of Microsoft Cognitive Services. Once this is done the VPA is able to perform the task on behalf of the user based on the users’s command using the UsageStatsManager class, the VPA was able to learn the user’s usage statistics with which it was able to open an application which it wasn’t able to do previously. The ActivityInfo attribute in PackageInfo class was used by the VPA to implement learning process in an even easier way by not letting the user to show it how to learn. The drawback in the above methods are that the VPA can open only those activities which have “android: exported = true”. With the help of intent filter tags in AndroidManifest.xml we can explicitly open an activity by using the tags like action, category, data present in it.

V. CONCLUSION AND FUTURE WORK

To make the learning process more automated we need to implement natural language process more effectively than before and the VPA should be able to read the AndroidManifest.xml file of other applications in the system. This can be done by getting the AndroidManifest.xml file by using ZipEntry class which helps to extract the files from the installed Android Package Kit (APK). Now the VPA can use the ZipEntry class because all the APKS are a kind of zip format. This class helps the VPA to get that particular file from the APK if we know the format of the developed APK. Since the AndroidManifest.xml file is in the default position for every APK we can extract it. The problem is thateven after extracting it, the file is in compressed format which can't be read. But there is ajar file present called "apktool" which helps the VPA to uncompress the whole APK. If we use it to extract only the AndroidManifest.xml file, we can use the data present in the xml file to implement the learning process even better. This can be done by reading the details in intent-filter tag which helps to start activities explicitly by using the action, data, category tags which helps the VPA to learn what actions a particular activity can do.

Conflict of Interest : Nil

REFERENCES

[1]. Santos, J., Rodrigues, J.J., Silva, B.M., Casal, J., Saleem, K., & Denisov, V. (2016). An IoT-based mobile gateway for intelligent personal assistants on mobile health environments. *Journal of Network and Computer Applications,* **71**: 194-204.

[2]. Mitchell, T.M., Caruana, R., Freitag, D., McDermott, J., & Zabowski, D. (1994). Experience with a learning personal assistant. Communications of the ACM, **37**(7): 80-91.

[3]. Canbek, N.G., & Mutlu, M.E. (2016). On the track of artificial intelligence: Learning with intelligent personal assistants. *Journal of Human Sciences,* **13**(1): 592-601.

[4]. Persky, S., Goldring, M.R., Turner, S.A., Cohen, R.W., & Kistler, W.D. (2018). Validity of assessing child feeding with virtual reality. *Appetite,* **123**, 201-207.

[5]. Lindenberg, J., Nagata, S.F., & Neerincx, M.A. (Crete Greece, June 2003). Personal assistant for online services: Addressing human factors. Human Centered Computing, in Proceedings of Human Computer Interaction International Conference. Lawrence Erlbaum, Mahwah NJ, 497—501.

[6]. Emmanuel Azom Edim., Adannaya Simeon IvoA, (2013). “Multi-Agent Based Virtual Personal Assistant for E-Health Service”, *Journal of Information Engineering and Applications,* Vol. **3**, No.11, 8-15.

[7]. Buravenkova, Y., Yakupov, R., Samsonovich, A.V., and Stepanskaya, E. (2018). Toward a virtual composer assistant. *Procedia computer science,* **123**: 553-561.

[8]. Hudson, S., Matson-Barkat, S., Pallamin, N., & Jegou, G. (2019). With or without you? Interaction and immersion in a virtual reality experience. *Journal of Business Research*, **100**: 459-468.

[9]. Loureiro, S.M.C., Guerreiro, J., Eloy, S., Langaro, D., & Panchapakesan, P. (2019). Understanding the use of Virtual Reality in Marketing: A text mining-based review. *Journal of Business Research*. Vol. **100**, 514-530.

[10]. Massai, L., Nesi, P., & Pantaleo, G. (2019). PAVAL: A location-aware virtual personal assistant for retrieving geolocated points of interest and location-based services. *Engineering Applications of Artificial Intelligence,* **77**: 70-85.

[11]. Coronado, M., Iglesias, C.A., Carrera, Á., & Mardomingo, A. (2018). A cognitive assistant for learning java featuring social dialogue. *International Journal of Human-Computer Studies*, **117**: 55-67.

How to cite this article: Rajkumar, K., Monica, B., Sekar, K.R., Manikandan, R. and Rahim, Robbi (2019). The Virtual Personal Assistant (VPA) - Learning Process. *International Journal on Emerging Technologies*, **10**(1): 139-142.