May - June 2020
ISSN: 0193-4120 Page No. 5384-5389

Dynamic Intelligent Virtual Assistants Based on User Preferences

Adarsh Anand[1], Saket Savarn[2], Sunilkumar Manvi[3]

[1,2,3]School of Computing and Information Technology
REVA University, Bengaluru, India
[1]adarshanshu7@gmail.com, [2]saketsavarn07@gmail.com, [3]ssmanvi@reva.edu.in

Article Info
Volume 83
Page Number: 5384-5389
Publication Issue:
May - June 2020

Abstract

When we talk about "Dynamic Intelligent Virtual Assistant" (DIVA), we are trying to imply about a virtual assistant which is more dynamic than any pre-existing virtual assistants present and is more personalized in understanding its user. The reason for making this virtual assistant came from our observations as to how the present assistants fail to adapt uniquely to different users and their needs or preferences. Our virtual assistant aims at having a higher learning curve of the user preferences so that our assistant's interaction is different for each user. Our assistant not only includes all the basic features of the present-day assistants but also has more personalized features such as "Time and activity tracker of the user", "learning about user's preferences with some guidance", etc. The DIVA is effectively implemented to demonstrate its operation of user assistance based on personal preferences learning.

Keywords: *Virtual Assistant, Machine Learning, Artificial Intelligence, TTS, STT, Speech Recognition*

Article History
Article Received: *19 November 2019*
Revised: *27 January 2020*
Accepted: *24 February 2020*
Publication: *16 May 2020*

1. Introduction

Information interchange and how we interact with all this information is what this information era is all about. Man's curiosity and need to make an intelligent agent to do it's work without need of manpower or manual labour has been researched for a very long time. We invented computers, laptops and then smartphones, automated robots and tools with intelligence, came up with new fields like IoT, fintech, etc over-time and made breakthrough discoveries in older fields like data science all to make our work easier. Overtime, we realized even though all these technologies have made our life very much easier, the method of communication has always been troublesome, i.e., inputting our commands was done through methods like clicking or pressing buttons or typing a command or methods requiring our manual attention, meaning input-method for these technologies was manual in nature even if the output was automated.

Big Tech companies like Google, Samsung, Microsoft, IBM, Apple, Facebook, Mozilla, etc have already rolled out their virtual assistants or Text-To-Speech Engines on which we can make Virtual Assistants. Now many of these are open-sourced up to an extent allowing developers to make their own virtual assistant and this is how we came up with the decision of making our own Virtual Assistant (VA) due to our dissatisfaction of what current VA offer us.

The problem with current day VA is that they follow the same structure or learning curve for every one of its user, meaning, the current day VA have a learning curve of about 5% for everyone. If I am using SIRI or Google Assistant versus you using your own SIRI or Google Assistant, both of our SIRI/Google Assistant will behave the same way even if we interchange our personal Virtual assistants.

Our aim is to make a "Dynamic" Virtual Assistant that will learn about user preferences by web-scraping information from the Internet and arrange the information so that it can use this information on its own in our conversations. This way if I use DIVA and if you use it, it will remember our individual preferences, hence we are expecting it to have a learning curve of at least 20% more than current VA.

May - June 2020
ISSN: 0193-4120 Page No. 5384-5389

The rest of the paper is organized as follows. Section 2 presents related work on intelligent assistants. Architecture of DIVA is presented in section 3. Section 4 describes methodology for implementation of DIVA. Finally results and conclusions are presented in Sections 5 and 6, respectively.

2. Related Works

In [1], the authors created a spelling correction (SC) model which is trained to explicitly correct the errors made by a speech recognizer. To train that SC model, they generated error hypotheses by decoding the TTS (text-to-speech) data synthesized from a large text-only corpus. Their results showed that the SC model yields clear improvement over the baseline LAS (Listen, Attend & Spell) model. The authors in [2] presented an interactive TTS (text to speech) system and combined it with a speech emotion recognizer that had the potentials to synthesize matching speaking styles as the input query, meaning the TTS engine has the capability of having "tone", "accent" or "moderation" in the output speech and not speak in a monotonous way unlike other TTS engines. Using [1] and [2] as references, we are trying to replicate human behaviour to understand speech to its best by auto-correcting common-sense errors and enacting pseudo-emotions via speech.

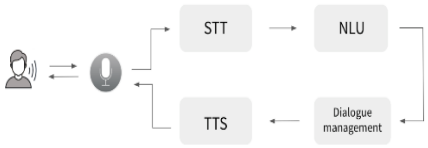
After imitating human speech, a reservoir of knowledge is required to speak from. The work in [3] presents LibriTTS which is a multi-speaker English corpus of approximately 585 hours of read English speech at 24kHz sampling rate. The Corpus used for LibriTTS proved to be the best for making a Virtual Assistant and hence acts as a guideline for making our own corpus. In [4], authors present a novel keyword spotting (KWS) system that uses contextual automatic speech recognition (ASR). As in natural languages, a context/statement can be described in multiple ways and hence using simple statements to invoke commands from the Virt. Asst. becomes wary if the statement is posed in a different manner, therefore using keyword spotting, the Virt. Asst. gets a gist of the intent behind the user's statement and invoke an action for the same.

The work discussed in [5] is Named Entity Recognition (NER) which works in a similar way to KWS [4] but unlike Keyword Spotting, NER is used to recognize unstructured or non-native terms such as Names, location or places, organization names, medical or military or domain-specific terms or codes, etc. This will be useful in adding information to the corpus [3] for words that cannot be classified via POS (Part of Speech) tagging & be used to learn personal information about the User's preferences, like his name, favourite sports & music, etc as they all fall under Named Entities.

To be able to tackle unpredictable statements, commands or information request, if the corpus [3] fails to fetch an answer for the same from its repository, the best option is to use the web for answers but the same causes errors as nothing is organized in a structure format on the web. Authors in [6] improved NER by using background knowledge from sources like DBpedia, etc on the web. Using NIF 2.0 (NLP Interchange Format) and Linked Open Data (LOD) a structured way to get information access from the web is possible and the same can be then used by the Virt. Asst. In [7], authors present LAS hypothesis, where SC model can generate an expanded list which has significantly lowered the WER (Word Error Rate). This helps in increasing accuracy of the virtual assistant.

3. Architecture of DIVA

A Virtual Assistant has multiple segments that are required to make the whole system work. For a virtual assistant, the input-method which is our speech is taken care by SST or Speech-to-text engine, this is the speech recognition part, now based on the speech inputted, an action is invoked by the Logic Engine which acts as the brain of the whole system, based on the action, So, we can understand that a VA uses multiple fields to work such as Speech Recognition, Neural Networks, Natural Language Processing, GET-POST Service to fetch data from Internet. Over all these features, our virtual assistant, DIVA can primarily keep track of our activities and time, given that we have told it about the categories that we divide our time into (Study, Assignments, Projects, Games, Watching series etc.). Fig 1 depicts overview of DIVA. The blocks in figure are explained as follows.



Figures 1: Basic Architecture of a Virtual Assistant as it carries out the command and perform the NLU on that request

Automated Speech Recognition (ASR) (Used for SST) [Input Method]: ASR is an algorithm that has evolved over time due to benefits that Natural Language Processing has reaped due to many researches works such as better audio understanding due to improved understanding of acoustic signals in an audio and better noise reduction. It involves taking the audio and converting it to a wave file.

Logic Engine or Natural Language Unit (NLU) [Brain/CPU of the system]: Once the SST converts the speech to "raw string", (Fig 2) the logic engine processes the information and tries to find the most suitable output for the same. The logic engine can be complex or very simple depending on the virtual assistant. More complex logic engines are usually more dynamic in nature and use various methods to work and categorize or process the "raw string" such as Parts-of-speech (POS), Name Entity Recognizer (NER), etc. The logic of NLU is given in Fig 2. Internal details are given as follows.

May - June 2020
ISSN: 0193-4120 Page No. 5384-5389

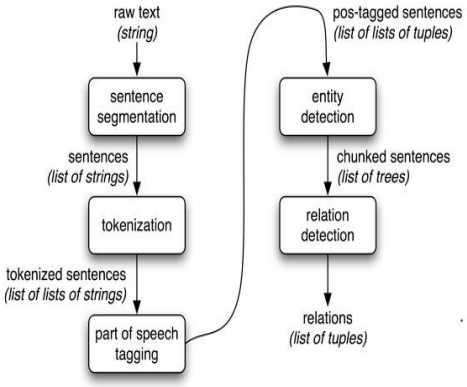


Figure 2: Pipeline Architecture for an Information Extraction System as it takes the raw text and perform several analyses on it and gives the relations.

Part of Speech Tagging: Part of speech tagging consists of breaking down the raw string or text in further grammatical categories like Nouns, pronouns, adjectives, adverbs, etc. (Fig 3) This helps the Logic Engine identify irregular words that do not fall in any of these categories or spot the main phrase or word using keyword spotter mechanism (Eg: "date" is a keyword in the sentences "What is the date" or "Today's date"). These categorized words or broken-down words are called as tokens or tags and this process is known as "Tokenization" and "tagging". Example is given in Figure 3.

```
>>> import nltk
>>> sentence = """At eight o'clock on Thursday morning
... Arthur didn't feel very good."""
>>> tokens = nltk.word_tokenize(sentence)
>>> tokens
['At', 'eight', "o'clock", 'on', 'Thursday', 'morning',
'Arthur', 'did', "n't", 'feel', 'very', 'good', '.']
>>> tagged = nltk.pos_tag(tokens)
>>> tagged[0:6]
[('At', 'IN'), ('eight', 'CD'), ("o'clock", 'JJ'), ('on', 'IN'),
('Thursday', 'NNP'), ('morning', 'NN')]
```

Figure 3: Tokenization of texts (Observe how the sentence in the end is categorized with codes like "IN", "CD", etc, these stand for Preposition (IN), Cardinal No. (CD), JJ (Adjective), NNP (Proper Noun), NN (Noun)).

Extracting Entities: In NLP, entities refer to terms that do not fall naturally in a natural language, for example, names of organizations, people, places, objects, domain specific terms and codes, etc. There are various ways to recognize such entities and this is useful in many scenarios like in our case where we will be using it to figure out terms that stand out and make the NLU search such terms so that it can learn about them on its own. This way it can keep track of music artists names or cricketers' names, etc as all belong to entities.

The basic technique we will use for entity detection is chunking, which segments and labels multi-token sequences as illustrated below. The code consists of the variable which is defined with particular grammar (Fig 4), that is chunked based on the grammar which is as follows:

```
chunkGram= r"""Chunk: {<RB.?>* <VB.?>*
<NNP>+<NN>?}"""
chunkParser  = nltk.RegexParser(chunkGram)
chunked  = chunkParser.parse(tagged)
```

See Figure 4 for segments and chunks.

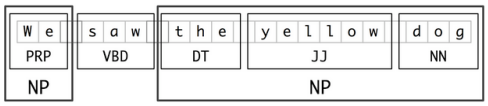


Figure 4: Segmentation of tokens and chunks both. Here, PRP, VBD, DT, JJ, NN are tokens while NP and NP at outer level are the chunks.

Text-to-speech [Output Method]: Once the NLU has used various methods to process the raw text, it will invoke an action. If the action turns out to be a reply to the user, then a pre-defined text based on the NLU decision-'s or a self-constructed sentence is converted from text to speech via a TTS or Text-to-Speech Engine. We are using Google TTS, which speaks the output via the speakers.

Additional Features of our NLU: As we lookout for other virtual assistant in the market, DIVA is more personalised and more focus on one particular behaviour and habit. It learns the human behaviour in first 10-15 days. Plots the graph according to that and revises the algorithm on what the user is more focused on and how he is handling the time by its unique feature (Stop Watch). As we described that it will be divided in different categories and according to that it will be easy to differentiate whether it's for the good and just happy hours for the user. Using Pandas, we plot and visualize the graph and give the suggestion how the user can improve the time, which is again a unique feature (Time Management).

4. Methodology

1.1. Hardware

The user will interact with our Raspberry Pi 3, Model B using a mic component and the same will be converted to text using Speech-To-Text (We are using speech recognition module of python which is using the Google Speech API for this purpose as of now). This will be main core hardware component which works as a central system that holds the code which includes all the API and the models useful for the Virtual Assistant. RASPBERRY PI unlike its competitors like Arduino, etc is a micro-processor not a micro-controller (which Arduino is). The capabilities of it are: voice recognition, good performance, easy to install, and open source.

1.2. Software

The texts will go through our NLU modules like ntlk, textblob, speech_recognition and they in place will

May - June 2020
ISSN: 0193-4120 Page No. 5384-5389

invoke actions or dialogue management tasks that will be processed via various APIs, our python scripts running them in back-end, many of our own python scripts and after this is done processing this will be converted from text-to-speech using TTS modules (we are using Google Text to Speech for this purpose as of now).

Software and APIs used are: Python 3.7 Scripts (Back-End), Raspberry Pi Interface Software (Raspbian Software), NOOB Package (For Raspberry Pi), Various APIs ported through python 3.7 (Google Speech API, Mozilla Deep Speech, Mozilla TTS, RASA, Google TTS) and modules in python (NLTK, TextBlob, Speech Recognition (NLP Modules, Seaborn, Matplotlib (Data Visualization Modules, Time, OS (Standar Modules), Urllib, Certify, SSL (Modules for setting connections with APIs)).

1.3. Block Diagram

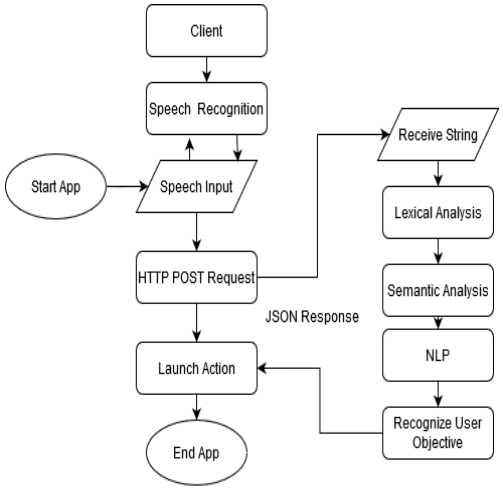


Figure 5: Block Diagram of a client-server based Virtual Assistant as client give certain command and application use HTTP POST request to send the requirement to further analysis

(Fig 5) presents framework of VA based on client server paradigm. The components of it are explained as follows.

Speech Recognition: It will identify words and phrases in spoken language and convert them to a machine-readable format. Since the user will give the input as a voice so, STT (Speech-to-Text) will convert them into text format and then understand the word and finally invoke the application to start.

HTTP POST Request: POST is a request method supported by HTTP used by the World Wide Web. By design, the POST request (Fig 5) method requests that a web server accepts the data enclosed in the body of the request message, most likely for storing it. It is often used when uploading a file or when submitting a completed web form.

Analysis of Text: Text mining identifies facts, relationships and assertions that would otherwise remain buried in the mass of textual big data. Once extracted, this information is converted into a structured form that can be further analysed, or presented directly using clustered HTML tables, mind maps, charts, etc.

NLP: Machine learning is an artificial intelligence (AI) technology which provides systems with the ability to automatically learn from experience without the need for explicit programming, and can help solve complex problems with accuracy that can rival or even sometimes surpass humans. However, machine learning requires well-curated input to train from, and this is typically not available from sources such as electronic health records (EHRs) or scientific literature where most of the data is unstructured text.

JSON Response: Json() is a function that reads the response stream to completion and parses the response as json. This operation may take time, so instead of just returning the json, it returns another Promise. The success function of this promise will have the resulting json as an argument. JSON Request is proposed as a new browser service that allows for two-way data exchange with any JSON data server without exposing users or organization to harm. It exchanges data between scripts on pages with JSON servers in the web.

5. Results

Some snapshots of the results we tested out the DIVA are presented in fig 6, 7 8 and 9. In these figure D.I.V.A. tries to start the stop watch and learn about the user preferences and even perform several functions in order to fulfil the request of client.

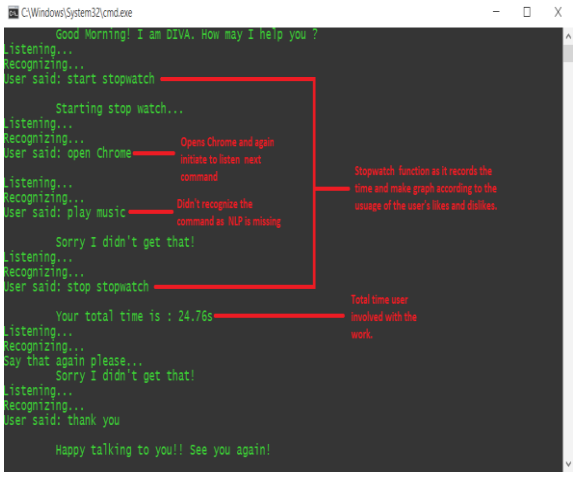


Figure 6: Interacting with DIVA (1/3) Starting and stopping the time and calculating, the user interests and likes and what kind of application he/she use

TEST Engineering & Management

May - June 2020
ISSN: 0193-4120 Page No. 5384-5389

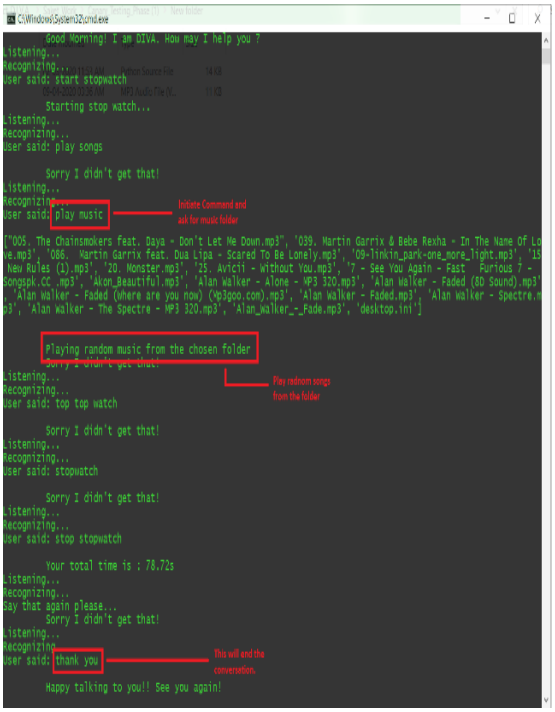


Figure 7: Interacting with DIVA (2/3) DIVA playing the songs as heard the command play songs, and recording the likes of user song's genre.

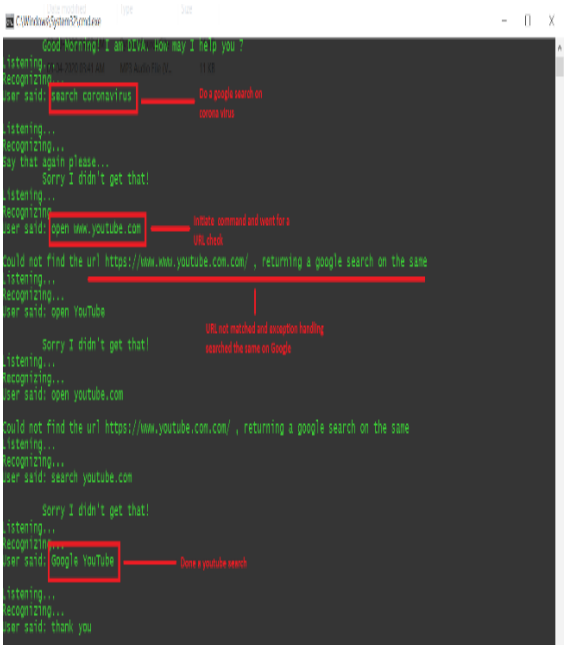


Figure 8: Interacting with DIVA (3/3), Opening the Browser from D.I.V.A. and even doing YouTube search from the queries and even recording the likes and dislikes.

From fig 9, we observe that interests of a user in Netflix, web development, gaming, etc., are plotted by DIVA. During the Stop Watch Timer, it calculates the user likes and dislikes and the preferences user prefer and involves his/her time mostly in.

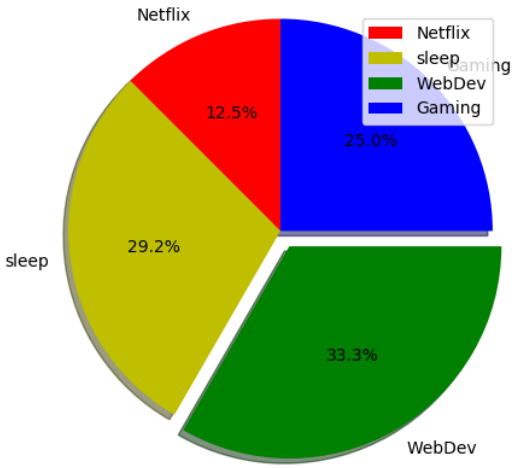


Figure 9: Graph made by stop watch to visualize the user interests and work algorithm accordingly. As given client is mostly interested in Web Development so D.I.V.A. learns about specific user and give the future suggestion in the field of web development.

6. Conclusion and Future Works

As of now, DIVA has successfully been implemented with a stop-watch feature as a background process which does not exist in present-day Virtual Assistants present in the market (Siri, G.A., Alexa, etc). Since, the stop-watch is running in the background, the user can still use DIVA to invoke other tasks or commands while the stopwatch will be used to track the user's time and tell them where they can improve or if they are having too less sleep or not.

DIVA is currently a bit slower and inaccurate compared to market VA. but with few fine-tuning and addition of tokenizer and classifiers we will be able to remove those hindrances as well. For future work, we are planning to run DIVA on a Raspberry pi which is connected to Internet and to push the user's information in a secure cloud service. Multiple APIs also are been added to DIVA and an intelligent web-scraper is also been finalized soon.

References

[1] Jinxi Guo, Tara N. Sainath, Ron J. Weiss : "A Spelling Correction Model for End-to-End Speech Recognition " in arXiv:1902.07178v1 [eess.AS], 19 Feb 2019.

[2] Yang Gao, Weiyi Zheng, Zhaojun Yang, Thilo Kohler, Christian Fuegen, Qing He, "Interactive Text to Speech Via Semi Supervised Style Transfer Learning" in arXiv:2002.06758v1 [cs.SD], 17 Feb 2020.

[3] Heiga Zen, Viet Dang, Rob Clark, Yu Zhang, Ron J. Weiss, Ye Jia, Zhifeng Chen, Yonghui Wu: "LibriTTS: A Corpus Derived from LibriSpeech for Text-to-Speech" in arXiv:1904.02882v1 [cs.SD], 5 Apr 2019.

May - June 2020
ISSN: 0193-4120 Page No. 5384-5389

[4] Assaf Hurwitz, Michaely, Xuedong Zhang, Gabor Simko, Carolina Parada, Petar Aleksic: "Keyword Spotting for Google Assistant Using Contextual Speech Recognition" in Proc. IEEE ASRU, 2017.

[5] Leonid Velikovich, Ian Williams, Justin Scheiner, Petar Aleksic, Pedro Moreno, Michael Riley: "Semantic Lattice Processing in Contextual Automatic Speech Recognition for Google Assistant" in Proc. ISCA, 2018.

[6] Sebstian Helmann, Jens Lehman, et al. "Integrating NLP using Linked Data", Proc. International Semantic Web Conference, 2013

[7] William Chan, Navdeep Jaitly, Quoc V. Le, Oriol Vinyals, "Listen, Attend and Spell: A Neural Network for Large Vocabulary Conversational Speech Recognition",in Proc. IEEE ICASSP, 2016.