JARVIS: An interpretation of AIML with integration of gTTS and Python

Ravivanshikumar Sangpal, Tanvee Gawand, Sahil Vaykar, and Neha Madhavi,
of Computer Technology, Government Polytechnic Pen, 402 107, Maharashtra, India.
Mail ID: programmer06081811@gmail.com

***Abstract*—This paper presents JARVIS, a virtual integrated voice assistant comprising of gTTS, AIML[Artificial Intelligence Markup Language], and Python-based state-of-the-art technology in personalized assistant development. JARVIS incorporates the power of AIML and with the industry-leading Google platform for text-to-speech conversion and the voice of the Male Pitch in the gTTS libraries inspired from the Marvel World. This is the result of the adoption of the dynamic base Pythons pyttsx which considers intentionally in adjacent phases of gTTS and AIML, facilitating the establishment of considerably smooth dialogues between the assistant and the users. This is a unique result of the exaggerated contribution of several contributors such as the feasible use of AIML and its dynamic fusion with platforms like Python[pyttsx] and gTTS[Google Text to Speech] resulting into a consistent and modular structure of JARVIS exposing the widespread reusability and negligible maintenance.**

***Index Terms*—AIML, Python 3.4.7, JARVIS, gTTS, pyttsx, Google API, Tkinter, GTK+, Linux.**

I. Introduction

THIS is the first system proposed to intelligently speak to the users in natural language. It is inspired by the character JARVIS played by Edward Jarvis in the Marvel Studio Productions Iron Man who served Tony Stark [Robert Downey Jr] as his most intelligent hypothetical assistant lending him help into every possible aspect such as Technology, Medical, Social and other coherent curriculum aspects.

Similarly, the system provides spoken solution to any real-world problem. This is made possible with the integration of the most enhanced platform of the computer industry i.e. Artificial Intelligence which is consequently provided by Google but re-transformed into multiple appearances by the world-wide community of developers working on high-end systems trying to solve complex problems using AI. Also the major mucilage of this system is the python interpreter which is also the sub-ordinate Text-to-speech engine for JARVIS.

II. JARVIS : About the System

Analyzing the speech recognition systems in the acknowledged platforms from the viewpoint of Neural Language Processing [NLP] , we can classify major three generations of these Intelligence Systems : First Generation based on Pattern Matching[5]; Second Generation including techniques of Artificial Intelligence[2]; Third Generation indulging higher ordered, sophisticated pattern matching techniques[6], based on the eXtensible Markup Language (XML).

Out of the three generations of the virtual intelligences , we would like to focus onto the third generation, only for the reason that the systems built into it are representing a more complete architecture of NLP[8], and comparatively average performance to the systems of other generations (e.g., A.L.I.C.E.)[7], which has been appreciated internationally.

A majority of third generation intelligences are based on AIML [4], a markup language for chat-bots constructions which is based on XML. Systems based on AIML implement a dialogue base of categories formed of units identified by the $\langle category \rangle$ element. Each category is interpreted of an input pattern identified by $\langle pattern \rangle$ element interconnected to one or more output templates which are uniquely tokenized as $\langle template \rangle$ element. Also the patterns which are as input to the system can use special characters(e.g., *) to be structured coherent to various input phrases by the user. These characters are used, furthermore, to represent unknown sentences to the categories base (input pattern $\langle pattern \rangle$*$\langle pattern \rangle$). AIML has a fair and simple computational structure, but with enhanced performance as compared to other generations.These characteristics are responsible for the popularity of this generation, which includes around 50,000 chat-bots in various languages [1].

However, these third generation systems still posses some limitations or uneven consequences which were investigated by us from the computational point of view by comparing them to other NLP systems and also by the conversational point of view while referring the Conversational Analysis Theory (CAT) to verify the functional limitations of these systems.

Our analysis revealed three recurring problems in these intelligences:

1) These systems treat the users sentences based on their syntactic-morphological structure, therefore, they do not consider *hi* and *hello* as repeated sentences;

2) Many sentences are treated as unknown when in fact they are adjacent turns (expected from a sociocultural point of view [9]) to the chat-bot's former turn. For example, if a chat-bot asks a user do you like soccer and the user answers yes, some systems do not recognize the yes and treat the sentence as unknown, frequently causing an interruption in the fluency of the dialogue;

3) Occasionally, systems fail to take into account the global course - structure of a conversation (opening, development, and closing) [10] and may accept, for example, greetings that are typical in openings to appear in the development or in the closing phase.

978-1-7281-0283-2/19/$31.00 ©2019 IEEE

III. JARVIS: Intelligent Assistant integrating AIML

We present here JARVIS, a static virtual intelligence system comprising of Python interpretation. The software we developed is aquisitional a redefinition of the standard AIML base[7], incorporating the Google API protocols and the application base as Python with dynamical and graphical properties.

Section IV presents the proposed mechanism for integrating the three attributes, and section V shows the process of reconstruction of the AIML base to incorporate information on intention.

IV. Structural Integration

With the main functional element as AIML, this system also comprises of alternative enhancement compatibility mechanisms such as gTTS and Python. The structure has the global R-R Model (Request-Response Model) with prior modules built with python. The python libraries such as Speech Recognition, pyttsx, pyaudio, pygame, etc are embedded into the system to incorporate with the events of AIML and gTTS to be handled flawlessly. The initial data is processed by the gTTS engine which converts the speech format to text and passes the parameters to the python interpreter, then the python interpreter parses the text into the AIML kernel (AIML Bootstrap Kernel). The data is then suspected by the intelligence and appropriate response is regenerated by the intelligence which is also parsed to the Python script.

This data is now consisting the final output desired by the end-user, so the text is coherently passed to the gTTS engine which now converts the text input to gTTS to an audio file which is temporarily saved in the system.

All again the python implements this procedure and triggers the audio file which is comprising of the response to the user. Complete command data is traced by the python script and is displayed via raw-data format on the terminal.

V. Reconstruction of Standard AIML

AIML's official base, the original A.L.I.C.E. AIML is reinforced by the support of about 50 thousand categories, distributed in 50 files. These files are classified by themes or by formal structure. As an instance : the 'science.aiml' file contains categories related to science, and further they are broken considering their focused topic.

We have modified the structure by adding 5 additional files, each containing data specific to their domains and categories.As we have created an end-to-end system, the applications are limited to only Linux platform, henceforth opening our gateway to platform-specific intelligence structure. This system comprises of only bash and terminal interface making the AIML capable of performing system tasks.

JARVIS beholds the compatibility and the capability of manipulating the Linux Distributions according to user commands i.e. the system tasks and general settings can be modified on-the-go. Also after the modification, the system is able to redirect the application interface to any type of third-party application window; performing the allotted task simultaneously. After redistributing the categories, the AIML base of JARVIS is able to perform GUI-centric tasks such as playing music, changing wallpaper, manipulating system volume, playing a track from Rhythmbox, speak out the time, etc. The AIML manipulation also consists of files specifically generated for the User-centric tasks (basic chat.aiml, search internet.aiml) for decreased latency in the command processing module. This modifications in AIML base are as a support system for gTTS engine and Python interpreter coordinately providing best-in-class authoritative responses to the user queries.

VI. gTTS: Speech Segmentation

The overall speech synthesis is the result of the gTTS engine imported in the system. The Google's Text-to-speech is using Deep Learning Neural Networks to recognize and suspect the speech given to the module. The speech is interpreted into the engine by using the in-built mixer method of gTTS. With Microphone as source for the JARVIS, all the collected data is parsed to the gTTS and the result is transcribed either in '.txt' or in '.mp3' format.

A. Relocating to AIML

The transcribed data from gTTS is remapped into the intelligence environment of AIML by parsing the variating formations to Python and then into gTTS. The flow sequence of the data is :

1) User Input to Python: The spoken words of user are considered as '.mp3' input file to the Python script.

2) Python Interpreter to gTTS: The '.mp3' file in the script is further parsed to the gTTS engine for the speech-to-text conversion.

3) gTTS to AIML: The raw text generated by the gTTS is the convocation of the paraphrases spoken by the user.This is parsed to the AIML module which is interconnected to gTTS via the python scripting.

4) AIML to gTTS: The AIML module generates a response to the input query by the user and it is in the text format. So to reconfigure the generated text; it is passed to the gTTS engine which now inversely works to convert the text-to-speech.

5) gTTS to Python: Finally the '.mp3' format generated by the gTTS is parsed to Python file to import it for the Bootstrap Kernel. This speech is then played by Python with the multiple API's used in the script which completes the speech segmentation cycle.

VII. Python: Implementation and Modules

Python is an interpreted, high-level, general-purpose programming language. Python has a design philosophy that emphasizes code readability, notably using significant white space. It provides constructs that enable clear programming on both small and large scales making it the most preferred base platform for any AI implementation. Python posses many features suitable for Artificial Intelligence and henceforth integrates many AI models which are built-in with the recent versions of Python (Python 3.6.8 Stable and 3.7.2 maintenance release). Python contains Dynamic and AI worthy libraries such as : Tensor Flow, Matplotlib, Sci-kit Learn, PyTorch and many more to integrate high-end objectives in Python like

Machine Learning, Neural Networks, Artificial Intelligence, etc. The modules used are:

A. Modules Integrated

As the system is based on AIML intelligence, the first module integrated is "aiml". This library allows to parse the python entities into the AIML interpreter.

The "argparse" library is used only for a single cause and i.e. the Platform of the system. As the JARVIS system is completely Linux based, we need system tools to manipulate the File Structure of the Linux OS. When a file is called from remote directory to a location in the system, it needs the actual location of where the desired file is located; this is known as the "positional argument" in the Python interpreter.

Also general programming libraries such as "os" and "time" are implemented in the system as the AIML needs system values such as time and state of the system to describe the real-time entities to the end-user.

B. API's Generated

Most of the AIML categories are reconstructed for the JARVIS system. The formation of the new AIML categories and structures need new path or re-directing mechanism to complete the action. So using the python scripting techniques for multi-tasking in the Linux system, Application Programming Interfaces are created for minimum latency in the data processing.

The basic API implemented is the Google API which is responsible for all the speech synthesizing algorithms provided by Google. The tasks performed by this API are: Text-to-speech conversion, Google Maps Integration, Browser Redirection on trigger.

The second most apprised API in JARVIS is for the Linux system itself. The dynamic interactions occurring amongst Python and system tasks such as volume control and wallpaper redistribution is performed using this API. This piece of code comprises of bash commands for root system manipulation and python script for SEH (System Event Handling).

The final Programming interface used in JARVIS is the API for YouTube. This API renders complete control over the YouTube i.e. the Static selection of the proper URL in the object deck for the requested query are intelligently selected with the implementation of this API .

VIII. System Platform: LINUX - A Unique Approach

The major cause of the development of this AI i.e. JARVIS is the forlorn nature of Linux Platform. The competitors in the field are comprising of their own Artificial Intelligences in the recent updates as it is highly requisite with their own platform-specific AI namely:

1) Cortana: Technology giant Microsoft's own AI with integration to Windows 10 enabling Windows Hello Support and Local searches along with Microsoft Edge.

2) Siri: The international end-to-end system mega-corp most preferably Apple Inc. owns this AI with complete support to the full range of their devices ranging from mobile phones, desktops to their AI enabled television sets eventually making it the most portable but "end-to-end" AI.

3) Google Assistant: The world-wide tech giant Google LLC has innovated this AI with the best-in-class Intelligence techniques indulging Machine learning, Deep Learning and Neural Networks making it the most evident product of the company ever.

This extrovert complexity indulging in these platforms made us pioneer the AI development in the most forlorn platform i.e. Linux including all the flavours of Linux (Ubuntu, Redhat, Kali, Debian, Fedora, etc.) henceforth inspiring us to initiate the JARVIS system. This approach can be scrutinized as unique for the reason that Linux, on today's date does not include any AI mechanism for the ease of use for users; although this platform is mostly used by the development community but it is further considerable that it should be more familiar to open-end computer users. Linux has most of its juice extracted via the Terminal. Terminal is used for every efficient task to be performed on Linux then either it would be Ubuntu or Linux-Mint or Kali or any other system.

JARVIS is on its vision to make this user-platform interaction more easier and usable to most people via the most addictive and easy-to-indulge way of communication i.e. "Voice Communication". It is extreme ease for everyone to just talk to their own system and accomplish the task via their voice itself. JARVIS enables huge gateway to major part of community to start including the best system 'Linux' in their day-to-day life.

IX. Conclusions and Future Work

In this paper, we presented JARVIS: An interpretation of AIML with integration of gTTS and Python based on the dynamic Python interpretation environment. This system includes effectual use of AIML and Google API to make the Linux user interface more interactive and easier than ever. Our contributions include: (1) AIML 1.0 with lighter base and more portability for JARVIS; (2) Speech recognizing techniques by Google API providing best-in-class performance; (3) Use of Python Interpreter as it is the most Dynamic-in-nature and functionally advanced structure supporting AI integration and for a positive part, its pre-installed in every Linux Distro.

This is an original work in the area of Linux-AI integration with the most acknowledged base technologies used i.e. Google TTS engine and Python for Interpretation and GUI development.

Conclusively, we would like to point out some future work within our research: (1) JARVIS could be released as an open-source software on the Canonical Community itself;(2) JARVIS could be further provided an stronger AI than AIML reshaping JARVIS into an more functional and competitive amongst others in the market;(3) We are planning to provide a face to JARVIS which will be a holographic projection from our future product 'Smart Watch';(4) JARVIS can be transformed from a stable AI assistant into an dynamic self-learning AI system which will be more potent to the system

itself; (5) JARVIS can be reprogrammed to perform automation tasks such as: Home automation, industry automation, or Smart Homes / Societies; (6) JARVIS if implemented with Raspberry-Pi can innovate the concept to a whole new level by providing features like Google Home and Amazon Alexa.

ACKNOWLEDGMENT

This research was supported by Mr. Milind M Sangpal in financial aspects. We thank our colleagues from Government Polytechnic Pen who provided insight and expertise that greatly assisted the research, although they may not agree with all of the interpretations/conclusions of this paper.

We thank Mr. Mahesh Zemse for assistance with documentation,intellectual aspects and for comments that greatly improved the manuscript.

We would also like to show our gratitude to them for sharing their pearls of wisdom with us during the course of this research, and we thank our reviewers for their so-called insights. We are also immensely grateful to our parents for their comments on an earlier version of the manuscript, although any errors are our own and should not tarnish the reputations of these esteemed persons.

REFERENCES

[1] D. Aimless and S. Umatani., A tutorial for adding knowledge to your robot, 2006 Available at:http://www.pandorabots.com/botmaster/en/tutorial?ch=1. Accessed on January 05, 2019.

[2] M. L. Mauldin. Chatterbots, tinymuds, and the turing test - entering the loebner prize competition In *Proceedings of Twelfth National Conference on Artificial Intelligence (AAAI-94)*, Seattle, Washington, 1994.

[3] H. P. Grice. Logic and converstion. In P. Cole and J. L. Morgan, editors, *SpeechActs. Syntax and Semantics* volume 3, pages 41-58. Academic Press, New York, 1975.

[4] S. Laven. The simon laven page, 1990. Available at:http://www.simonlaven.com/. Accessed on January 05, 2019.

[5] J. Weizenbaum. Eliza - a computer program for the study for the study of natural language communication between man and machine. *Commun. ACM,* 9(1):36-45, 1966.

[6] R. S. Wallace. Aiml - artificial intelligence markup language, 2006. Available at:http://www.alicebot.org/aiml.html. Accessed on January 05, 2019.

[7] R. S. Wallace. A.l.i.c.e. bot, 2006. Available at:http://www.pandorabots.com/pandora/talk?botid=f5d922d97e345aal. Accessed on January 05,2019.

[8] J. R. Searle. *Natural Language Understanding.* The Benjamin Cummings Publishing Company, Inc, New York, NY, 1995

[9] E. A. Schegloff. Conversation analysis and communication disorders. In C. Goodwin, editor, *Conversation and Brain Damage.*Oxford University Press, New York, 2002.

[10] L. A. Maruschi. *Analise da Conversacao.*Editora Atica, Sao Paulo, 1986.