JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue

JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)

An International Scholarly Open Access, Peer-reviewed, Refereed Journal

BUILDING AN OPEN-SOURCE, PRIVACY PROTECTING NEXT-GENERATION VIRTUAL PERSONAL ASSISTANTS.

[1]Varun Gowda, [2]Pallavi Shankar, [3]RK. Gagan Prakash, [4]Hemanth. K, [5]Mohammed Aiyman
[6]Dr. Mahesh V
[1,2,3,4,5]Student, [6]Assistant Professor
[1,2,3,4,5,6]School of Computer Science & IT
[1,2,3,4,5,6]JAIN (Deemed-to-be) University
Bengaluru, Karnataka, India

Abstract: Voice-based virtual assistants such as Google's Now, Amazon's Alexa, and Apple's Siri are proving to be popular with consumers. However, they're still not what many people consider a "true" personal assistant, which needs to be sensitive to the context and user preferences. In this paper, we talk about some of the challenges involved in developing customized voice-controlled assistive technology and share how we've tried to address them. Natural language understanding and dialog management are the most important parts of our technology pipeline.

Index Terms **- Virtual Personal Assistants, Natural Language Understanding, Assistive Technology, Spoken Dialog Systems, Dialog Design, Automatic Speech Recognition**

I. INTRODUCTION

Thanks to the proliferation of IoT and digital services, it's super simple these days to accomplish our personal or professional tasks with a few clicks. It was humans who used to have their input translated into computer languages. Now, there are specific AI programs to interpret queries and provide the needed information for people. Commercial virtual assistants are becoming increasingly popular and one of the most well-known is Alexa which can understand natural language commands for more than 90,000 skills. These are tools that allow you to use your voice to control things. The AI assistant will get better at taking orders and completing complex tasks. It will be able to analyze our needs and learn what we want.

Voice control is the way to go these days. It's difficult to find a person who hasn't experienced its charm in one form or another. It's taken decades for academics and industry leaders to finally get the technology right. Now they're selling voice-controlled devices like Apple's Siri, and Amazon's Alexa that can help you find any word in a dictionary, add items to a shopping list, and more. These gadgets not only illustrate current technology advances, but they also manage to convey to a broader end-user level the concept of an artificial personal assistant, that is, a system that can comprehend and respond. There's been continuous progress in terms of technology and supported areas. Despite these achievements, the implementation of natural language-based assistive technologies remains tentative. One of the disadvantages of Siri is that, despite their best efforts, they're not fully personalized. This means users can't rely on them for every question as a human would be able to. More attention to human adaptability is essential to properly incorporate technology into users' daily routines. Technology has advanced to the point where it can be depended on, but it still needs a lot of improvements. The integration of context and customized behavior is in its early stages - significant progress will need to be made before they are considered personable.

One day, AI assistants might be able to take care of all our digital and business needs, right from messaging companies, to reaching out to retailers. As the middleman, in this case, they can view and share our personal information with those organizations. What's clear is that there is fierce competition to create the best virtual assistant. Currently, all of these companies are trying to come out on top: Facebook, Amazon, Apple, Microsoft and Google plus many more are involved in a serious battle for supremacy in this field. This raises several serious risks. These include: interoperability, privacy, and applicability.

II. VIRTUAL PERSONAL ASSISTANTS

A personal assistant, in our general sense, is a person (or an agent) who can give specialized support at a certain time and in a specified activity area. In a normal workplace, for example, a secretary aids with tasks such as answering phones, documenting meetings and appointments, purchasing products, and interacting with clients. An assistant must be capable of adjusting to their master's individual requests and progressively paying attention to her/his own preferences and routines over time. Furthermore, because personal assistants are only expected to support one person, the one-on-one relationship between helper and master is a valuable advantage. With this definition of an assistant, it's easy to see how many people could benefit from having one around for them. We might not be aware of our need for a personal assistant right now, but once you let their skilled services work for you and find out the benefits they can provide, that won't matter. When we think about the issue more technically, it becomes clear what the desired user expectations and requirements are. When it comes to digital assistants, simplicity, adaptability, and ease of interaction are among the most desired features.

Many people prefer to use voice-based input, so it's the perfect way to go. This is because it requires minimal cognitive resources as well as brings a convenient user experience. An example is how voice interfaces can compress complex menu choices into quick options, and offer detailed responses to queries in real-time.

In 1995, the futuristic APPLE Knowledge Navigator [21] concept depicted what a voice-based virtual personal assistant might look like. It stresses the importance of having excellent Voice User Interfaces (VUI). After seeing the movie, any expert in speech processing, synthesis, audio parsing, and automatic recognition may clearly see why the proposed application is rather futuristic. The user's dialogue with the virtual assistant is unconstrained. The assistant comprehends the user's psychological motivations, detects the user's emotions, and may even recommend tasks that include the user's social contacts (e.g., promoting a friend's artwork or freshly published research work). Furthermore, the assistant speaks perfect English, employing subtle pauses and vocal cues to convey their subtext. For example, the way they said "hidden disgust" shows that they're frustrated with how you were treating your mom.

VUIs, on the other hand, are a tough interface choice in terms of technology. Particularly since the underlying technologies for Automatic Speech Recognition (ASR), Speech-to-Text Parsing (STT), Natural Language Understanding (NLU), and Text-to-Speech (TTS) synthesis are presently constrained by pre-defined application parameters and cannot yet support free-form human-machine communication. As a result, voice-enabled technologies today are designed to be specific use cases.

Artificial Intelligence (AI) is certainly present in our everyday lives and can be a helpful tool when dealing with a variety of scenarios. These include using voice to surf the internet, monitoring your elderly parents' well-being, and sending some texts or managing a calendar program. In light of the difficulties that come with old age, systems aimed at the elderly have to be modified for this demographic [1]. Speech recognition algorithms will often need to be adjusted or improved.

When you're talking to systems that use humans' vocal interactions, consumers have higher expectations than when they're using graphical user interfaces. That's why it's critical for companies to make VUIs that deliver what people expect from voice. Given that current technology cannot provide algorithms to analyze and understand free-form conversations, the ideal design approach to dialogue management is important to assuring the system's efficacy and efficiency. Without standards for this, we are not aware of any criteria for generalized use of these systems in conversational agent scenarios.

Despite efforts to solve this problem [2], it is still in the research stage. We have listed some of the alternative ways to enhance the current state of the art. However, looking at these challenges we focus on neural language processing and natural language understanding. Here are the components we think need the most work to make networked VUIs more lifelike and fun for users.

III. VIRTUAL ASSISTANTS: THE POTENTIAL THREATS

As we develop technology for virtual personal assistants, we should keep possible downsides in mind. It is projected that a platform monopoly or duopoly would emerge for the virtual assistant. Monopolies are harmful to consumers because they suffocate competition and innovation. A monopoly on virtual assistants is especially alarming since it controls customers' access to the digital world and has access to billions of people's private data across a wide range of enterprises.

A linguistic web that's confidential, but can be accessed by the virtual assistant. It's otherwise similar to how we use a browser to access the graphical internet when it comes to not being proprietary. However, other linguistic webs that work like private websites are starting to emerge. Even one natural language is difficult to understand, let alone the several languages necessary for the international market. Because the cutting-edge neural network approach requires a large quantity of annotated human words, Amazon employs 10,000 workers solely dedicated to Alexa [3]. Furthermore, well-known assisters attract providers, who in turn attract more users. A significant barrier to entry will almost surely result in a virtual assistant monopoly or duopoly as a result of the network effect and expensive development costs.

By connecting people to your website, AI virtual assistants drive traffic and can point them towards the appropriate type of product. They might also charge a fee for transactions conducted through their platform, that's how app stores make their money. More importantly, by getting access to users' accounts, virtual assistants can get valuable business intel data like what type of energy users use. For example, their thermostat account already has a lot of info on daily usage based on the past season. A virtual digital assistant that monopolizes the industry is not a good idea.

Platform monopolies jeopardize privacy. In both the United States and the European Union, there is a growing realization that massive platforms pose severe threats to individual privacy. The General Data Protection Regulation (GDPR) is a new European Union regulation that seeks to protect personal information and data by regulating how companies like Facebook, Google, Netflix, and more collect and use your information. For example, Facebook now dominates the social networking industry on a global scale, with no significant competitors in the majority of nations. Companies that possess billions of people's personal information have tremendous power. They have the power to share users' data with third parties, manipulate users' attitudes in important decisions such as presidential elections, and intervene in user behavior. A monopolistic assistant platform will have access to all of our data; they will be more knowledgeable than Amazon, Facebook, and Google combined.

IV. POTENTIAL AREAS OF IMPROVEMENT

As a result of analyzing the previous system requirements and the state of the art, we see four possible areas for improvement:

History of Extended Dialogue

The dialogue history is often used in deployed systems, whether for commercial or research reasons, to disambiguate user statements and keep track of the discussion state.

The memorized activity record, on the other hand, is typically focused on a particular topic and does not extend across several talks. It might be possible to create systems that adapt to the user over time if the history element is extended. The result would be systems that can be customized, such as acoustic models, language models, and understanding models. They should also know how to end conversations properly. For example, overall involvement may be increased if the system recognizes a user's identity, gender, and age and respects his or her technology choices.

Improved Context Awareness

Aside from what the user has specifically requested, a great amount of information is typically available that may be analyzed by a system to gradually expand its context-awareness.

Sensors built into (future) homes, businesses, or even common items like our beloved smartphones, for example, are significant data sources that may be used to improve human-machine interaction. Furthermore, the Internet might be a potential data source that could be mined to provide improved knowledge of the "outside world" and, as a result, improve a system's thinking.

Dynamic System Adaptation

An SDS's development and implementation are frequently separated into three stages. The first stage is focused on defining the necessary components, the second on implementation, and the third on actual system deployment. From the implementation stage on, the configuration of a system is typically static, which means that it does not change with usage or according to a certain conversation condition. As a result, we suggest a multi-agent architecture in which one agent's settings can be changed based on input from other agents. As a result, if one component detects a change in the discussion context, it may alert the others, causing their settings to change. This dynamic method would allow the system to be adjusted while it was still running.

Supported Task Hierarchy Design

A Previous study has discovered an overall classification for Dialog Management (DM) paradigms [4]. A DM is based on stochastic processes, such as POMDPs [7], MDPs [5, 6], the information state principles [8, 9], or a task hierarchy [8, 9]. Each of these groupings has benefits and drawbacks.

The information state paradigm is viewed as a flawed theoretical framework that fails to meet the requirements of the vast majority of practical implementations. The same is true for stochastic processes, with the main drawback being that training data must be obtained before developing a system. Despite significant research efforts (e.g., reinforcement learning [10]), these processes remain complex. As a result, the task-based paradigm continues to be the most appropriate strategy. Depending on the extent of a discussion, the user's initiative, and the conditional execution stages, task hierarchies can become quite complex. Figure 1. SDS Architecture Better tools and processes should thus be available, allowing task hierarchy design and automated transformations to be pushed to the application definition stage.

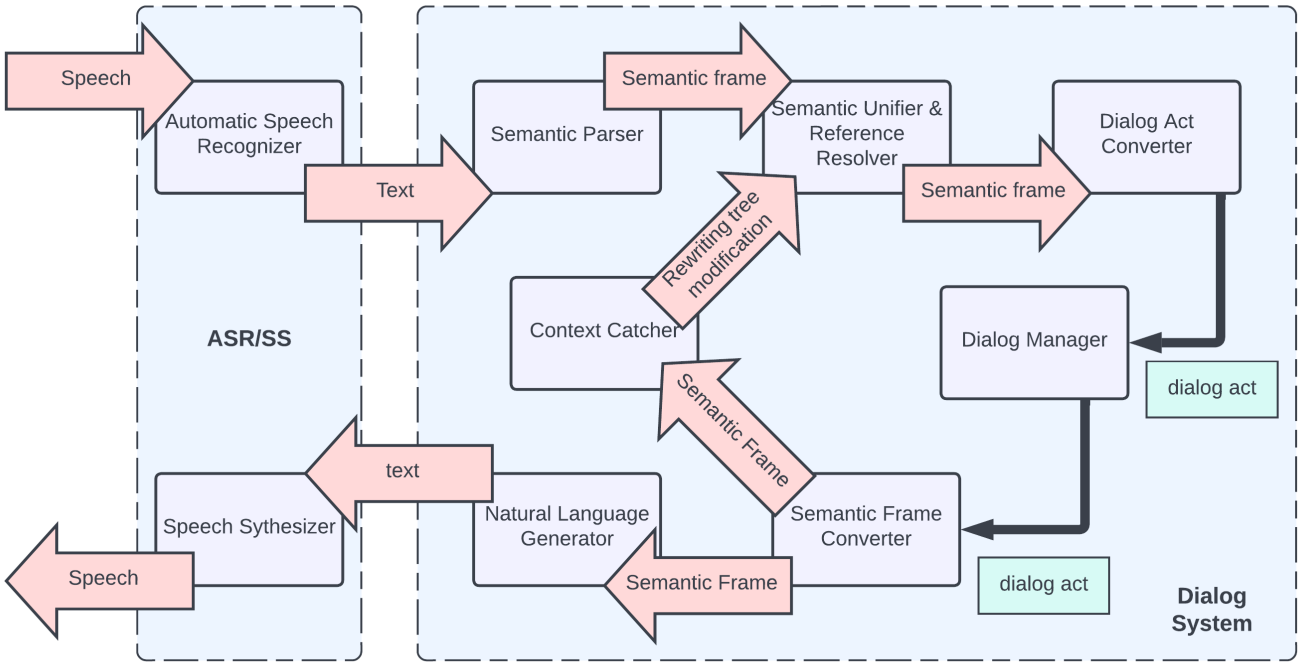

Figure 1. SDS Architecture

V. THE PROPOSED SPOKEN DIALOG SYSTEM (SDS) DESIGN

To address the aforementioned areas for improvement, a Spoken Dialog System (SDS) was created. Figure 1 depicts its architecture. The basic components shown are specialized agents that communicate with one another via the ActiveMQ messaging server. To govern a near-sequential processing pipeline, data flow between components is restricted. The user interface is illustrated on the leftmost layer. This is currently performed with a single sensor, mainly a microphone that receives the signal, which is subsequently analyzed by the ASR engine [11].

The ASR converts the signal into a finite number of observation vectors and compares them to statistical models of sound and vernacular. Voice-activated typing technology can detect spoken words and automatically insert them into sentences. It will also provide you with a ranking/list of possible guesses as well as estimations of how likely they are to be correct. Transcriptions with scores over a certain level are sent to the main system. If there's no transcription, then you'll probably just get the generic message "not understood". A component of the pipeline at one end of the pipeline generates and plays audio responses to text-based on an intermediate processing component.

We're excited to add more sensors in the future so that we can offer a better service. The data you've submitted is worth a lot of entropy and will make our system performance a lot better. For now, the focus is on improving the middle components which deal with language processing, conversation management, and text production. The Natural Language Generation (NLG) is currently based on a simple template-selection mechanism that chooses text utterances randomly. This is done based on the system's current semantic representation.

The Natural Language Understanding (NLU) component, on the other hand, is a complex mash-up of several interrelated components. Section 7 will go over these components in further detail, as well as the following transformations required to handle the transcriptions of text produced by the speech recognizer.

Finally, the suggested architecture revolves around the dialogue manager [9]. It is based on a generic inference engine that must be tailored using scenario-based job hierarchies. Based on user input, it performs actions and generates semantic concepts. Section 7 also has a more detailed overview of the component and how it is configured and utilized by the system.

VI. THE IMPLEMENTED SYSTEM ARCHITECTURE

Considering the above-discussed design and approach, we propose the architecture for Sia [15], an open-sourced, privacy-preserving, offline compatible and programmable virtual personal assistant (VPA) build over NodeJS, using AI concepts like NLP.js, Text-to-Speech (TTS), Speech-to-Text (STT) and more which are empowering Sia to ease the interaction with humans.

Sia is built on a modular architecture that gives us the flexibility to create and use packages/modules (skills) that fit your needs. Hence, it offers infinite possibilities without any barriers.

Figure 2 provides an overview of Sia's underlying system architecture and the following scenario describes the steps of the schema:

1. Client (web app) makes an HTTP request to GET information from the server. 2. The HTTP API responds with information to the client. 3. User talks with their microphone. 4. If the hotword server is launched, Sia listens (offline) if the user is saying "Sia". Once the server catches the buffer, Sia emits a message to the main server via a WebSocket. Now Sia is listening to the user. 5. If the

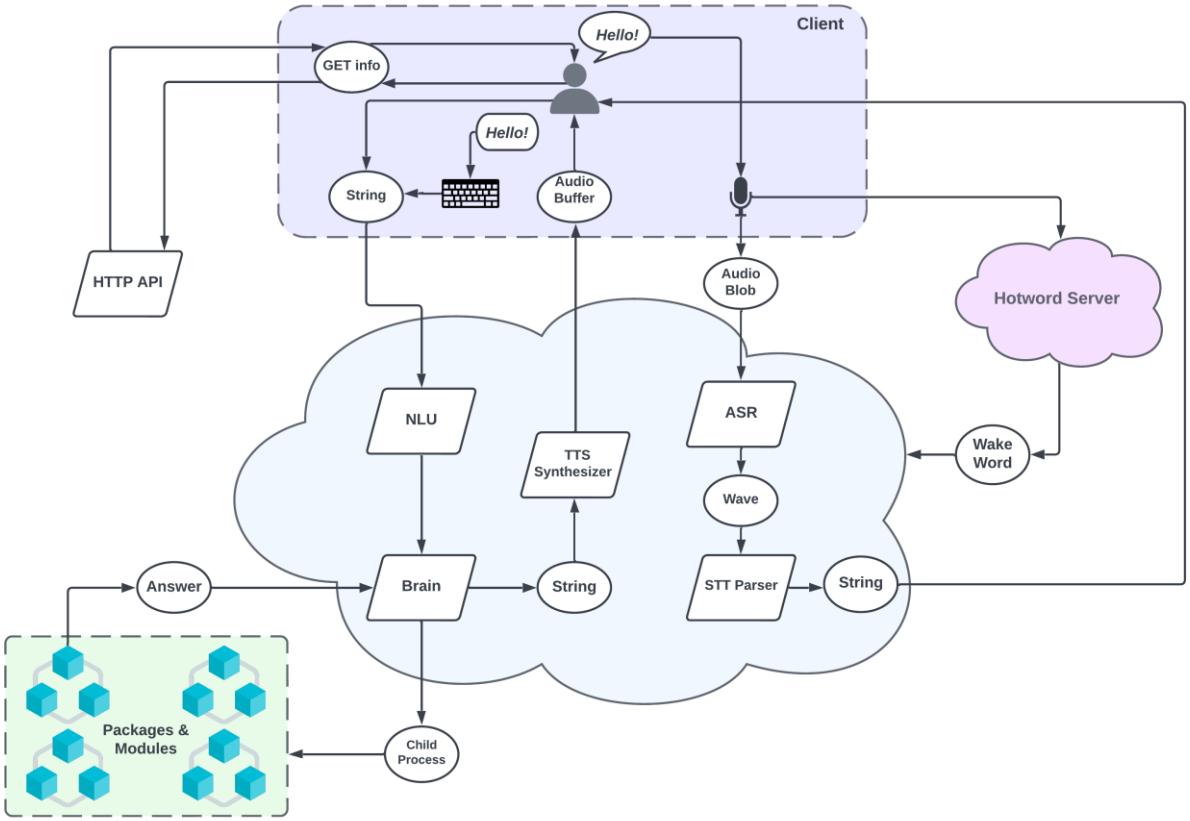

Figure 2. The Architecture of Sia

hotword server is not launched, the audio input (Hello!) will be directly passed to the server by the client. 6. Once, the server receives the audio input. ASR (Automatic Speech Recognition) will analyze and transform the audio blob into a wave file. 7. STT parser will transform wave file to string (Hello). 8. The user will receive the string back on the client app and the string is forwarded to NLU (Natural Language Understanding). 9. If the user decides to use a keyboard, instead of a microphone. A user can type (Hello) with their keyboard. Then the (Hello) string is forwarded to NLU. 10. NLU classifies the string and picks up a classification. 11. The Brain creates a child process and executes the chosen module. 12. The Brain creates an answer and forwards it to the TTS Synthesizer. 13. The TTS Synthesizer transforms the text answer (and sends it to the user as text) to an audio buffer which is played by the client.

VII. ALGORITHMS & COMPONENTS

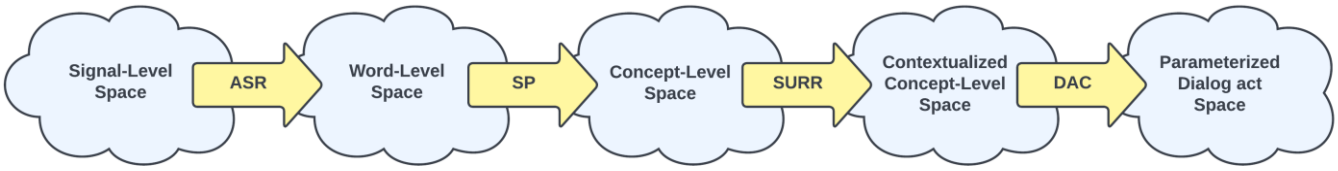

Figure 3. The One-way Interpretation Process of Spoken User Utterances

This section discusses the components and approaches we propose to address the challenges stated in Section 3. Following a discussion of the natural language understanding components, we will specify the kind of problems that we intend to address with this setup. Following that, the implementation of a conversation model design tool is discussed in detail.

Essentially, the role of natural language understanding (NLU) is to map the infinite space of meaning that a human can use to interact with a system, to the dynamic set of dialogue actions that a system can perform at any given moment in a dialogue. As a result, natural language processing (NLP) is a complex interactive process in which each sub-component has specific tasks. The level of any sub-input component is more than or equal to the level of its output component. Essentially, what an NLU engine does is analyze the input text and iteratively work to find the most accurate sources of information. Figure 3 shows the progression of a transcription process and how it eventually becomes a conversation.

Parsing

A semantic parser's job is to associate semantic labels with text. Context-free grammars (CFGs) are what most humans use to communicate. Usually, they are either hand-coded by experts or created using automatic analysis from chat logs with humans. In our system, we employed the strategy described by Jurcicek et al. [13]. Rather than matching complete-sentence patterns with text input, Jurcicek et al propose searching for patterns in text-level sentence chunks and the transient (i.e., previously assigned) Semantic Frame (SF) [17].

An operation is performed on the current SF when the triggering pattern is detected. Automatic rules are created by annotating utterances with their final SFs and analyzing their patterns. These rules consist of a trigger component (what to look for) and an operation component (how to apply the pattern). A technique that evaluates all of the [triggers: operations] pairs available at each stage of the training process. The pair with the highest score is added to the decode piles and added to the temporary corpus, meaning it results in the smallest distance reduction between the acquired and reference SFs in the Levenshtein [19] distance calculation.

Unification

As previously indicated, the parsing is only dependent on rules generated from past offline training and a user's text input, with no contextual information handled. As a result, the results of parsing may be shallow. A human utterance such as "two" may be labelled as "input: number=2," where input is the semantic frame's goal and "2" is the value of the slot "number." The speech has no

Algorithm 1 SURR algorithm

```
1:  procedure FindPath(Slots)
2:      if Slots is made of root sets of slots then return Slots
3:      else
4:          split Slots into two parts, Split and Remaining
5:          if Split can be mapped to a node Node in the trees then
6:              if the transformation Transform from Node to its parent is valid then
7:                  apply Transform
8:                  merge the transformation of Split with Remaining to get a new set TransSlots
9:                  FindPath(Slots)
10:             else
11:                 FindPath(Slots)
12:             end if
13:         else
14:             FindPath(Slots)
15:         end if
16:     end if
17: end procedure
```

further information, illustrating the need for additional contextual information to augment the primary semantic representation. As a result, we developed a Semantic Unification and Reference Resolution module (SURR).

The SURRR maintains a collection of dynamic slots. Some of these are set to have specific values. Each parent node has its own conditions that determine how the values of their children nodes change as you go up the tree. The algorithm must find a way to get across the node forest in order for a frame to pass the filter component. The slot is subdivided and allocated per tree. There's a technique (that some people say works) where you just try to split the graph until there's only one path left.

Given the following set of slots in the input SF, Algorithm 1 depicts the process*: Slots* = $Slot_1$ = $Value_1$, $Slot_2$ = $Value_2$, and so on till $Slot_n$ = $Value_n$. The splitting of Set (line 4) is remembered so that the method does not test the same division of a set again and exits the recursive function when all of them fail. When the SURR is unable to obtain a path from the structure, it returns a "no solution" message. This is to notify the user that the system did not understand the previous speech (i.e., not valid in the current configuration). The dialogue act converter, a last piece of software, then offers a mapping between the contextualized meaning representation of the user intent and the collection of dialogue acts accessible in the dialogue manager's current internal state. To do so, the reasoning engine's task stack is mined to get the expectations. The frame is then converted into a parameterized act via a matching procedure. If no acceptable match is found, a "cannot map" dialogue box is generated.

Data Integration

The SURR tree slots are dynamic predicates, which means that they do not remain static after being loaded. Some of them receive a value via contacting a sensor or an external software service (local or remote). This is how we recommend dealing with any references to the external world, i.e., the context that the DM does not store internally. For instance, information such as the user's geographic position, the weather, the date and time, or the day of the week.

We advocate integrating external data streams as soon as feasible to provide greater contextual information to the NLU process. Following the inclusion of external sources of information, the internal context is used to update the structure of the SURR. The Context Catcher (CC) intercepts and analyses status information. It is a module that gives new predicate definitions and remove-commands to the SURR so that branches are adjusted to the system's internal context. Finally, an external profile file will be added in order to store an extended history across dialogues. This file, which is accessible via the SURR, will hold user-dependent values that have been learned over time and proved to be relevant.

Dialog Design

As previously stated in the article, in order for a system to be really versatile and adaptable, the dialogue models must be simple and fast customizable. A task hierarchy, on the other hand, swiftly grows into an intertwined web of tasks that is difficult to identify and/or change. To prevent this, we came up with a new XML-based design language alongside automatic conversion - meaning the designs will be formatted automatically for machine input. Suppose you want to give an app some voice interactivity. Luckily, the specifications for the app, including its input and output protocols, are all known.

We may define the application in terms of linked forms with related execution scripts using our suggested design language. Figure 4 illustrates the XML code for such a description system. The forms are then processed using recursive extendable Stylesheet Language Transformations, which automatically generate a task-based dialogue model with variable definitions, inputs and outputs, conditions, and execution scripts.

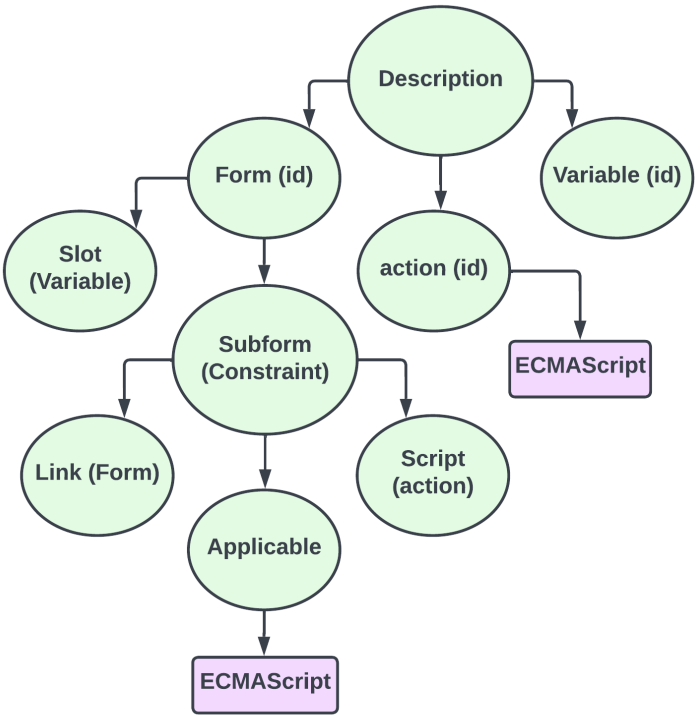

Figure 4. Application Description Language Scheme

Text-to-Speech (TTS)

Text to Voice is the process of creating synthesized speech from text. When viewing a screen is either impossible or difficult, this technology is utilized to communicate with users. This not only allows apps and information to be utilized in novel ways, but it also has the potential to make the world more accessible to those who are unable to read text on a screen. Text-to-speech technology has progressed over the previous few decades. Deep learning can currently generate highly natural-sounding speech with variations in pitch, rhythm, pronunciation, and intonation. Today, computer-generated speech is employed in a range of applications and is becoming a common component of user interfaces.

NLP.js

NLP.js is a NodeJS natural language utility [14]. It currently supports:

- Determine the language of a sentence.
- The fast Levenshtein distance between two strings.
- Find the best substring of a string that has the smallest Levenshtein distance to a specified pattern.

- Obtain stemmers and tokenizers for a variety of languages.
- Sentimental Analysis (with negation support) for phrases.
- Named Entity Recognition and management, multi-language support, then a looseness of the input text is allowed (the introduced strings don't have to be very precise).
- A Natural Language Processing (NLP) Classifier is used to categorize a speech in order to determine its intended meaning.
- NLP Manager: A tool can be used to maintain several languages, as well as keep track of entities, utterances, and intents for classifier training, and return entity extraction, intent classification, and sentiment analysis. It can also keep a Natural Language Generation (NLG) Manager for the responses.
- 40 languages are natively supported, with BERT integration supporting an additional 104 languages.
- Tokenization allows for the support of any other language, even fictional languages.

VIII. FUTURE WORK & CONCLUSION

The suggested processes in this research paper attempt to make a confined human and machine dialogue more flexible, adaptive, and reliable to the user's needs by overcoming the constraints of present technical capabilities. We discussed several difficulties for future SDS-driven Virtual Personal Assistants, as well as several algorithms and components to address them. Virtual assistants will most likely change the way we engage with machines. As a result, your data belongs to whoever offers the service you sign up for. Big companies are now trying to take over this market.

Sia is an attempt to bring together manufacturers, developers, and consumers in order to create the most open, general-purpose, interoperable, and powerful virtual assistant that supports open competition while protecting user privacy. Our long-term objective is to enhance our solutions (architecture) and make them available (open-source) so that other systems may use them.

The difficulty of meeting the advanced and complex dialogue requirements and powerful real-world situations while being efficient are both elements that make designing a luxury. It can be difficult designing effective, efficient language learning systems, and appropriate dialogue systems for any given situation as well. The ideas in this paper are intended to pave the way for more adaptive natural language interfaces.

REFERENCES

[1] D. R. S. Caon, T. Simonnet, P. Sendorek, J. Boudy, and G. Chollet, "vAssist: The Virtual Interactive Assistant for Daily Homer-Care," in Proceedings of health, 2011.

[2] B. Balentine and D. P. Morgan, How to Build a Speech Recognition Application: Style Guide for Telephony Dialogues, Enterprise Integration Group, 2001.

[3] MacMillan, D. 2019. Amazon says it has over 10,000 employees working on Alexa, Echo. The Wall Street Journal. (Nov. 13, 2018), https://www.wsj.com/articles/amazon-says-it-has-over-10-000-employees-working-on-alexa-echo-154213828

[4] R. Catizone, A. Setzer, and Y. Wilks, "State of the art in dialogue management," Deliverable D5, 2002.

[5] E. Levin, R. Pieraccini, and W. Eckert, "Using Markov decision process for learning dialogue strategies," Proceedings of ICASSP, 1998.

[6] J. Henderson and O. Lemon, "Mixture model POMDPs for efficient handling of uncertainty in dialogue management," Proceedings ACL-HLT, pp. 73–76, 2008.

[7] S. Larsson and D. Traum, "Information state and dialogue management in the TRINDI dialogue move engine toolkit," Natural language engineering, 2000.

[8] D. Bohus and A. I. Rudnicky, "RavenClaw: Dialog Management Using Hierarchical Task Decomposition and an Expectation Agenda," in Proceedings of EUROSPEECH, 2003.

[9] C. Rich and C. L. Sidner, "Collaborative discourse, engagement and always-on relational agents," in Proceedings of AAAI, 2010.

[10] B. Thomson and S. Young, "Bayesian update of dialogue state: A POMDP framework for spoken dialogue systems," Computer Speech & Language, vol. 24, no. 4, pp. 562–588, 2010.

[11] A. Lee, The Julius book, 2010.

[12]] M. Schroder and J. Trouvain, "The German text-to-speech synthesis system MARY: A tool for research, development and teaching," International Journal of Speech Technology, 2003.

[13] F. Jurcicek, F. Mairesse, M. Gasic, S. Keizer, B. Thomson, K. Yu, and S. Young, "Transformation-based Learning for semantic parsing," in Proceedings of INTERSPEECH, 2009, pp. 2719–2722.

[14] Axa-Group, nlp.js. https://github.com/axa-group/nlp.js/, 2020.

[15] NXture, Sia. https://github.com/NXture/Sia, 2021-2022.

[16] Giovanni Campagna, Rakesh Ramesh, Silei Xu, Michael Fischer, Monica S. Lam, "Almond: The Architecture of an open, crowdsourced, Privacy-Preserving, Programmable Virtual Assistant", April 2017, pp. 341-350.

[17] Giovanni Camagna, Silei Xu, Mehrad Moradshahi, Richard Socher, "Genie: A generator of natural Language semantic parsers for virtual assistant commands", pp 394-410, June 2019.

[18] Monica S. Lam, "Keeping the Internet Open with an Opensource virtual assistant", pp 145-146, October 2018.

[19] Li Yujian, Lie Bo, "A normalized Levenshtein Distance Metric", vol 26, Issue 6, June 2007.

[20] Monica S. Lam, Giovanni Campagna, Silei Xu, Michael Fischer, Mehrad Moradshahi, "Protecting privacy and open competition with Almond: An open-source virtual assistant", vol 26, Issue 1, Fall 2019, pp 40-44.

[21] YouTube, (http://www.youtube.com/watch?v=JIE8xk6Rl1w)