AI Based Assistant with Speech Recognition

Parag Chaudhari[1], Yash Patil[2], Srikar Nakka[3], Omkar Pawaskar[4]
Department of Computer Engineering, Sinhgad Institute of Technology, Maharashtra, India

***Abstract*- Autonomous robotic systems and intelligent artificial agent's capability have advanced dramatically. Voice-based digital Assistants such as Apple's Siri and Google's Now are currently booming. Artificial Intelligence is being used in management fields like assistance and monitoring, since a machine has become capable of learning simple activities of humans. PDAs are built to help the user get things done (e.g., setting up an alarm/reminder/meeting, taking notes, creating lists) and provide easy access to personal/external structured data, web services, and applications. In this paper we use AI and its computing capabilities to build a personal assistant application to help user in accomplishing its task and make him productive in his routine .Speech recognition as the man machine interface plays an important role in the field of AI, since it's a hand free option to give instructions to a machine. In this paper we use Hidden Markov Model to illustrate how speech recognition actually happens. We also explain use of neural network which is used to give learning capability to our assistant. We intend to build a personal assistant agent capable to discover the user's habits, abilities, preferences, and goals, even more accurately anticipating the user's intentions.**

***Index Terms*- Artificial Intelligence, Personal Assistant, Neural Network, Speech Recognition, Hidden Markov Model.**

I. INTRODUCTION

There has been as significant increase in the use of personal assistants in managing user task and activities. As the need and expectation to do more grew, despite improvements, a limited natural user interface has remained as one of the major bottlenecks in interacting with these devices. PDAs (also known as virtual assistants) precisely target this problem and have the promise of enhancing a user's productivity by either proactively providing the information the user needs in the right context (i.e., time and place) or reactively answering a user's questions and completing tasks through natural language. Tasks can be related to device functionality, applications, or web services. Over the last 20 years, researchers have investigated personalized virtual assistant agents targeting specific domains, including tourism, elder care, device control, and home and office applications. However, attempts at bringing them to market earlier have failed because of their limited utility. Over the past five years, there has been tremendous investment in PDA technology by both small and big technology companies. Siri, Google Now Cortana and Alexa are the major personal assistants

In the market today, and they provide proactive and/or reactive assistance to the user. Proactive assistance refers to the agent taking an action to assist the user without the users

Explicit request. Reactive assistance refers to the agent responding to the user's voice or typed command to assist him or her. The number of smartphone users using PDAs increased from 30% in 2013 to 65% in 2015 indicating increased adoption. Personal Assistants have become a key capability in most smartphones. They are now also deployed in tablets, laptops, desktop PCs, and headless devices and some are also even integrated into operating systems. These agents are designed to be personal, they know their user's profile, whereabouts, schedules, and so forth. They can proactively start interactions with their user through notifications and system initiated questions or reactively respond to user requests. User– PDA interactions typically take place via natural language, where the user speaks to the agent as if he or she were speaking to a real human assistant.

A personal assistant is a Meta layer of intelligence that sits on top of other services and applications and performs actions using these services and applications to fulfill the user's intent. A user's intent could be explicit, where the user commands the system to perform an action, or it could be inferred, where the agent notifies or makes suggestions upon evaluation of one or more triggering conditions it has been tracking. Personal assistants are built to help the user get things done (e.g., setting up an

alarm/reminder/meeting, taking notes, creating lists) and provide easy access to personal/external structured data, web services, and applications (e.g., finding the user's documents, locating a place, making reservations, playing music). They also assist the user in his or her daily schedule and routine by serving notifications and alerts based on contextual information, such as time, user location, and feeds/information produced by various web services, given the user's interests (e.g., commute alerts to/from work, meeting reminders, concert suggestions). Collectively, these functionalities are expected to make the user more productive in managing his or her work and personal life. In this paper we describe a simple architecture of a personal assistant and a neural network to embed learning capability into the assistant. We also describe Hidden Markov Model to illustrate speech recognition application.

II. PERSONAL ASSISTANT MODEL

Our common understanding of a personal assistant is that of a person (or an agent) who is able to provide distinct help at a given time and in a given activity context. An important characteristic of personal assistants is that they adapt to the distinct demands of their 'master' and furthermore (over time) progressively pay attention to her/his personal preferences and routines. Also, as by their definition, personal assistants should each be helping only one person, making this one-to-one relationship between assistant and 'master' a crucial benefit. Hence, when we try to approach this subject from a more technical perspective, we already find a set of requirements and expectations coming from users. Among the most requested features when thinking of digital assistants, are simplicity, flexibility and easiness of interaction. Voice-based input/output interfaces may be the easiest way to fulfill these requirements. The scenarios that the PDAs support can be divided into two main categories: 1) proactive and 2) reactive assistance. The conceptual agent architecture designed to support these two modes of assistance is shown in Figure .The system architecture depicts proactive and reactive user experiences, data, and service end points. Reactive assistance is shown in where the user issues an explicit natural language command (e.g., "book me a taxi") to the agent. The user request is handled through a set of reactive assistance components, such as speech recognition, LU, and DM. The data coming from various back ends, and applications are served to the user according to the constraints specified in the natural language query. The experience (reactive and/or proactive) can be served in one or more of the different device or service end points. Proactive assistance involves anticipatory computing, where the personal digital agent does things in a contextual manner (i.e., at the right time and place) that it expects is valuable to the user without an explicit user request. Proactive assistance makes use of inference, user modeling, and ranking to power experiences. Backend data, device, applications, and web services signals are leveraged for proactive inference and triggering. Even though proactive and reactive parts of the current PDA architectures are built in isolation, in principle they can use a single architecture to enable both types of experiences. In fact, most proactive scenarios have reactive extensions and vice versa. For example, if the user makes a restaurant reservation (reactively), the agent may (proactively) suggest a movie after the dinner or may offer to book a cab to take the user to the restaurant. Data and context are shared between the two assistance modes. Next, we focus on the proactive system architecture and the components that power proactive scenarios. Proactive assistance is based on the theory of proactivity that describes user desires and a model of helpfulness .The goal is to provide assistance to automate tasks or further the user's interests for things he or she cares about, all within context, without explicit user request.

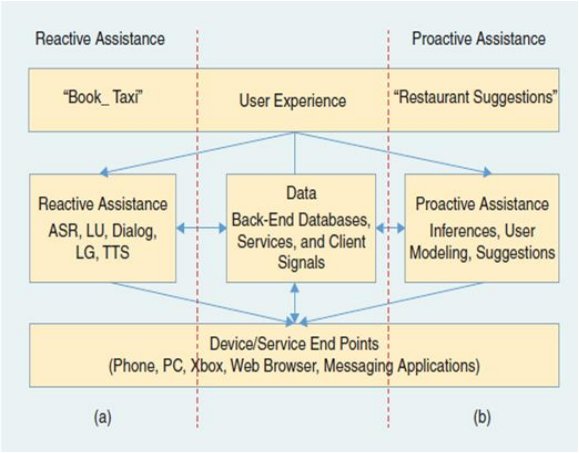


Figure 1. Architecture of Assistant

To achieve that, the agent is designed to possess a set of attributes; it should be valuable in that it advances the user's interests and tasks, while not interfering with the user's own activities or attention unless it has the user's explicit approval. It should be unimposing. The agent should be transparent in what it knows about the user. It should be anticipatory and know the future needs of the user and bring opportunities to the surface. The agent should also continuously learn and refine its decisions from the feedback signals it receives regarding the actions it takes. These principles put the user at the center, and the agent's actions are considered valuable only if they ultimately add value for the user. Proactive assistance operates on the proactivity continuum, which ranges from zero to full automation, allowing for the following scenarios:

I) Do it yourself (no help from the agent)

II) User tells the agent what to pay attention to (notifications/alerts)

III) Agent infers user's habits/patterns and makes suggestions(inference/suggestions)

IV) Agent makes decisions and takes actions (full autonomy on task decisions/executions).

Most of the currently supported proactive scenarios are notifications/alerts and suggestions. Even though there is some preliminary work, none of the agents in production supports autonomous decision making and action taking on behalf of the user without confirmation.

III. Artificial Neural Network

Artificial Neural Network is a supervised machine learning concept which is a type of connectionist computer system inspired by the biological neural networks that constitutes the animal brains. An Artifical Neural Network is a collection of nodes called Artificial Neurons (analogous to neurons present in brains). Each connection (Synapse) between neurons can transmit data in forward direction, the connections are weighted i.e. while transferring data the cost is multiplied to the data. Each node or neuron has computational property which computes all the input it receives to deliver an output. In common ANN implementations, the synapse signal is a real number, and the output of each neuron is calculated by a Non-linear function of the sum of its inputs. Neurons and synapses typically adjusts their weights as learning proceeds. The weight increases or decreases the strength of the signal that it sends across the synapse. Neurons may have a threshold such that only if the aggregate signal crosses that threshold is the signal sent. Typically, neurons are organized in layers. Different layers may perform different kinds of transformations on their inputs. Signals travel from the first (input), to the last (output) layer, possibly after traversing the layers multiple times. In general the layers are divided into three parts the first is the Input Layer which contains all the input elements, second layer is called Hidden Layer, this layer may contain multiple layers depending on the complexity of the problem or the architecture of the system, the final or third layer is the Output Layer which represents the output of the system. The input layer has exactly the same number of nodes as the number of attributes present in the input. The number of nodes present in hidden layer and output layer depends on the architecture of system and the required format of the output. An example of Neural Network is given below in Figure 2.

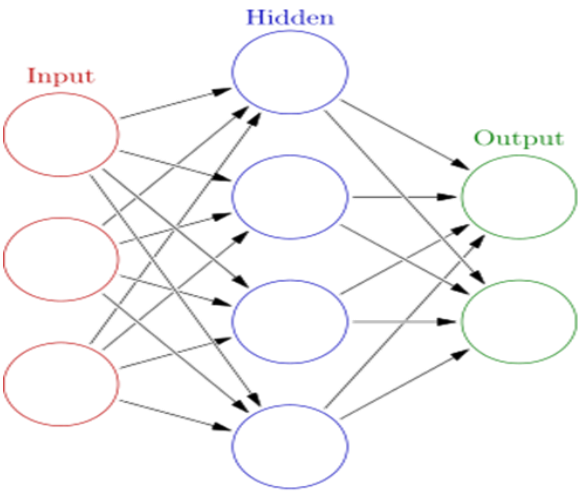


Figure 2. Representation of Artificial Neural Network

As shown in figure every node in a layer is connected to every other node in the next layer. The Weights of the Synapse are the important parameters which are calculated by repeated use of the training data during the training period. The training of the ANN's is carried out in two phases which are Forward propagation and Back propagation.

The Forward propagation is used to calculate the output of the system given the input elements. The Neuron computes the inputs by multiplying all of its

inputs with the weights of their respective Synapses and then adding the result of all of them.

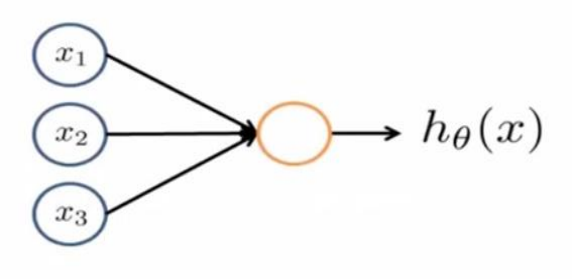


Figure 3. Forward Propagation of one Neuron

For the Neuron in Figure 3 the forward propagation or its output h(x) will be calculated as

$$z = \sum xi*wi \quad (1)$$

$$h\theta(x) = S(z) \quad (2)$$

where,

xi = input element.

wi = weight of Synapse connecting the nodes

S(z) = Sigmoid function

The BackPropagation is the phase where the output of the system is compared with the actual output provided in the training set. Backpropagation distributes the error term back up through the layers, by modifying the weights at each node.

IV. HIDDEN MARKOV MODEL

Hidden Markov model (HMM) is a tool for modelling time series data. They are used in almost all current speech recognition systems. HMM is used to represent probability distribution over sequence of observations In simpler Markov models (like a Markov chain), the state is directly visible to the observer, and therefore the state transition probabilities are the only parameters, while in the hidden Markov model, the state is not directly visible, but the output, dependent on the state, is visible. Each state has a probability distribution over the possible output tokens. Therefore, the sequence of tokens generated by an HMM gives some information about the sequence of states In its discrete form, a hidden Markov process can be visualized as a generalization of the Urn problem with replacement (where each item from the urn is returned to the original urn before the next step). Consider this example: in a room that is not visible to an observer there is a genie. The room contains urns X1, X2, X3, … each of which contains a known mix of balls, each ball labeled y1, y2, y3, …. The genie chooses an urn in that room and randomly draws a ball from that urn. It then puts the ball onto a conveyor belt, where the observer can observe the sequence of the balls but not the sequence of urns from which they were drawn. The genie has some procedure to choose urns; the choice of the urn for the n-th ball depends only upon a random number and the choice of the urn for the (n − 1)-th ball. The choice of urn does not directly depend on the urns chosen before this single previous urn; therefore, this is called a Markov process. It can be described by the upper part of Figure 4.

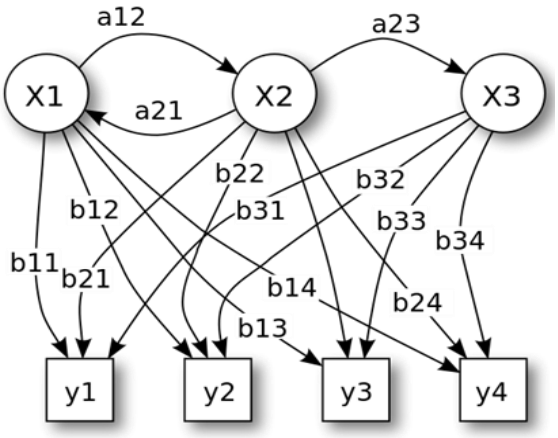


Figure 4. Probabilistic parameters of a hidden Markov model

where,

X — states

y — possible observations

a — state transition probabilities

b — output probabilities

The Markov process itself cannot be observed, only the sequence of labeled balls, thus this arrangement is called a "hidden Markov process". This is illustrated by the lower part of the diagram shown in Figure 1, where one can see that balls y1, y2, y3, y4 can be drawn at each state. Even if the observer knows the composition of the urns and has just observed a sequence of three balls, e.g. y1, y2 and y3 on the conveyor belt, the observer still cannot be sure which urn (i.e., at which state) the genie has drawn the third ball from. However, the observer can work out other information, such as the likelihood that the third ball came from each of the urns.

The Probability of all the states is computed as

$\bar{\pi}_i$ = expected frequency (number of times) in state S_i at time $(t = 1) = \gamma_1(i)$

$$\bar{a}_{ij} = \frac{\text{expected number of transitions from state } S_i \text{ to state } S_j}{\text{expected number of transitions from state } S_i}$$

$$= \frac{\sum_{t=1}^{T-1} \xi_t(i, j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$$

$$\bar{b}_j(k) = \frac{\text{expected number of times in state } j \text{ and observing symbol } v_k}{\text{expected number of times in state } j}$$

$$= \frac{\sum_{t=1}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}, \quad \text{s.t. } O_t = v_k$$

$$\bar{\mu}_{jk} = \frac{\sum_{t=1}^{T} \gamma_t(j, k) \cdot \boldsymbol{O}_t}{\sum_{t=1}^{T} \gamma_t(j, k)}$$

$$\bar{U}_{jk} = \frac{\sum_{t=1}^{T} \gamma_t(j, k) \cdot (\boldsymbol{O}_t - \boldsymbol{\mu}_{jk})(\boldsymbol{O}_t - \boldsymbol{\mu}_{jk})'}{\sum_{t=1}^{T} \gamma_t(j, k)}$$

$$\bar{c}_{jk} = \frac{\sum_{t=1}^{T} \gamma_t(j, k)}{\sum_{t=1}^{T} \sum_{k=1}^{M} \gamma_t(j, k)}$$

IV FEATURES

1. Predicting User Task

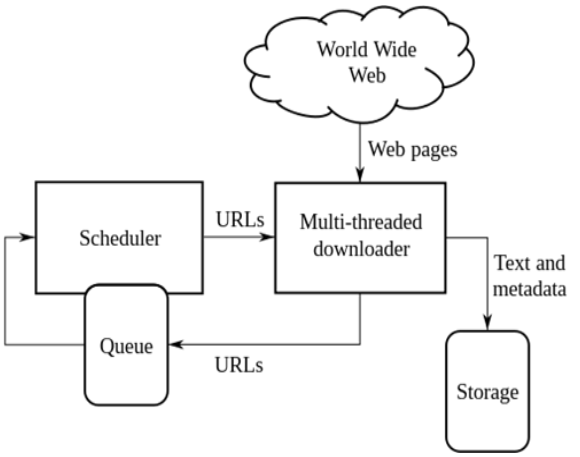
Predicting User Tasks is one of the features of our project which is aimed at predicting the next task of the user based on his previous history at a given time. Prediction of user tasks will be done by using ANN. The ANN is trained with one parameter that will be the task id of the task executed by the user and the output will be the time that the task was executed on this is achieved by using regression. The model of the ANN will be trained using Stochastic gradient descent in which the gradient of the model is approximated using a single example i.e. the values of the weights of the ANN will change with every task this will help in ensuring that the model is up-to-date with the users pattern and can adapt in future if there is any change in users pattern. The Tasks performed by the user each of them is given a unique task id that is used for training along with the time in seconds that it was executed at.

$$\omega := \omega - \eta \Delta Q(\omega)$$

As the algorithm sweeps through the training set, it performs the above update for each training example. Several passes can be made over the training set until the algorithm converges. If this is done, the data can be shuffled for each pass to prevent cycles. Typical implementations may use an adaptive learning rate so that the algorithm converges. A compromise between computing the true gradient and the gradient at a single example is to compute the gradient against more than one training example (called a "mini-batch") at each step. This can perform significantly better than "true" stochastic gradient descent described, because the code can make use of vectorization libraries rather than computing each step separately. It may also result in smoother convergence, as the gradient computed at each step uses more training examples.

2. Web-Crawling

For keeping the user up-to-date on his favourite topics we have deployed web-crawlers or web bots which will browse the World Wide Web in a methodical, automated way to get the things which interests the user based on the user settings. Web Crawlers are mainly used to make a copy of the web pages they visit and index them for later use and fast searching. The Web crawlers used in our project will mainly visit the social networking sites, google, blogs and other public sites on the internet with appropriate permissions.



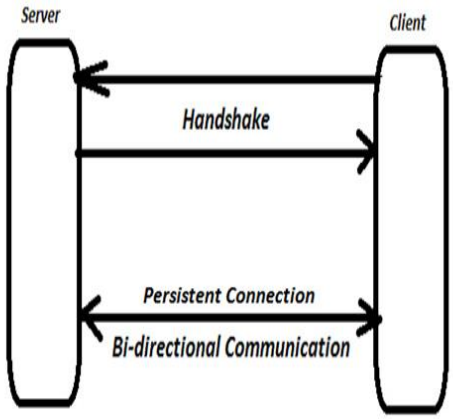
A Web crawler starts with a list of URLs to visit, called the seeds. As the crawler visits these URLs, it identifies all the hyperlinks in the page and adds them to the list of URLs to visit, called the crawl frontier. URLs from the frontier are recursively visited according to a set of policies. If the crawler is performing archiving of websites it copies and saves the information as it goes. The archives are usually stored in such a way they can be viewed, read and navigated as they were on the live web, but are preserved as 'snapshots'.

The archive is known as the repository and is designed to store and manage the collection of web pages. The repository only stores HTML pages and these pages are stored as distinct files. A repository is similar to any other system that stores data, like a modern day database. The only difference is that a repository does not need all the functionality offered by a database system. The repository stores the most recent version of the web page retrieved by the crawler.

The large volume implies the crawler can only download a limited number of the Web pages within a given time, so it needs to prioritize downloads. The high rate of change can imply the pages might have already been updated or even deleted.

3. Networking

The Assistants will be able to communicate between each other to request data or send data only with the consent of both the parties. For the connections between the two users sockets will be used along with the username of the user to check if the request or data is being sent to the intended recipient.



A network socket is an internal endpoint for sending or receiving data within a node on a computer network. it is a representation of this endpoint in networking software (protocol stack), such as an entry in a table (listing communication protocol, destination, status, etc.), and is a form of system resource. In practice "socket" usually refers to a socket in an Internet Protocol (IP) network (where sockets may be called Internet sockets), in particular for the Transmission Control Protocol(TCP), which is a protocol for one-to-one connections. In this context, sockets are assumed to be associated with a specific socket address, namely the IP address and a port number for the local node, and there is a corresponding socket address at the foreign node (other node), which itself has an associated socket, used by the foreign process. Associating a socket with a socket address is called binding.

V CONCLUSION

In this paper we presented the idea of gathering user interest by means of tags and provide them with the relevant content. We have proposed our approach of enriching the social user profile by analyzing the social behaviour and especially by analyzing the metadata of the resources, the tags assigned to the resources and the user's neighbour. Moreover, our approach takes into consideration the temporal aspect in order to capture the new information over time. The combination of the three information is in our opinion, a powerful and promising approach to provide flexible enrichment in an evolutionary environment. The enrichment of the profile could be used for further purposes such as recommendation, customization since it provides an information which

reflects the user's interests in every period of time. The first conclusions that emerge from evaluation of relevance tests are that the integration of different social contexts provides very conclusive results and that the integration of several social factors makes the search results more relevant. There are others social context related information which can be used concerning users, such as trust friendships and colleagues relationships, and concerning a resource, such as its location and the situation in which it has been produced We justified that user-generated tags are effective to represent user interests because these tags reflect human being's judgments while more concise and closer to human understanding. So the consensus among users for the content of a given web page can be reached more likely via tags than via keywords.

REFERENCES

[1] M.Bender ,T. Crecelius , M Kacimi ,"Exploiting Social Relations for query expansion and result ranking ", in ICDE Workshops 2015 ,IEEE

[2] I, Levenshtein, "Tag Based Social Interest Discovery,"2014.

[3] H. Halpin, V. Robu, and H. Shepherd, "Dynamic Enrichment of Social Users Profile" in Proc. 16th Int. Conf. WWW, 2015, pp. 211–220.

[4] M. Lipczak, " Building a social network, based on collaborative tagging, to enhance social information retrieval", Doctoral thesis, May 2014. Dalhousie University Halifax, Nova Scotia.

[5] C. Cattuto, V. Loreto and L. Pietronero, "Predicting User Interests from Contextual Information" CoRR, May 2016.