Chapter 3

A. Existing and Enhanced Algorithm

EXISTING ALGORITHM	**ENHANCED ALGORITHM**
1. DEFINE INPUT, HIDDEN AND OUTPUT NEURONS.	1. DEFINE INPUT, HIDDEN AND OUTPUT NEURONS.
2. INITIALIZE WEIGHTS AND BIASES, LEARNING RATE. ***(Problem 1)***	2. INITIALIZE WEGHTS AND BIASES, LEARNIING RATE, MOMENTUM. ***(Objective 3)***
FORWARD PASS	FORWARD PASS
3. GET TOTAL NET INPUT FOR EACH HIDDEN NEURON.	3. GET TOTAL NET INPUT FOR EACH HIDDEN NEURON.
3.1 MULTIPLY INPUT NEURONS TO THE WEIGHTS.	3.1 MULTIPLY INPUT NEURONS TO THE WEIGHTS.
3.2 MULTIPLY RESULT TO THE BIAS.	3.2 MULTIPLY RESULT TO THE BIAS.
4. REPEAT PROCESS FOR EACH OUPUT NEURON.	4. REPEAT PROCESS FOR EACH OUPUT NEURON.
5. CALCULATE ERROR OF EACH OUTPUT NEURON. ***(Problem 2)***	5. CALCULATE ERROR OF EACH OUTPUT NEURON.
6. GET TOTAL ERROR (COST) OF ALL OUTPUT NEURON.	6. GET TOTAL ERROR (COST) OF ALL OUTPUT NEURON.
BACKWARD PASS	BACKWARD PASS
7. DETERMINE SMALLEST COST FOR EACH LAYER IN THE NETWORK.	7. ADD TOGETHER THE SQUARE OF WEIGHTS TO THE COST FUNCTION. ***(Objective 2)***
8. UPDATE ALL WEIGHTS. ***(Problem 1)***	8. DETERMINE RATE OF CHANGE OF THE ERROR IN RESPECT TO THE WEIGHTS FOR EACH LAYER IN THE NETWORK. ***(Objective 1)***
	9. UPDATE ALL WEIGHTS.

Figure 3. Existing and Enhanced Algorithm Comparison

Problem 1:

Algorithm consumes computing resources exponentially when calculating the local minimum cost value of a given set with a large number of elements.

Objective 1:

Optimize the computer resources required by the algorithm when training neural networks by applying various rules of mathematical differentiation to weight computation and model training instead of the traditional brute force method.

Solution:

Dimension must be in the minimum so that the less computer resources will be consumed. In order to achieve this, we used a technique known as Gradient Descent. Instead of trying all the weights in a given network one by one we will instead calculate the rate of change of the cost in relation to the weights.

Simulation

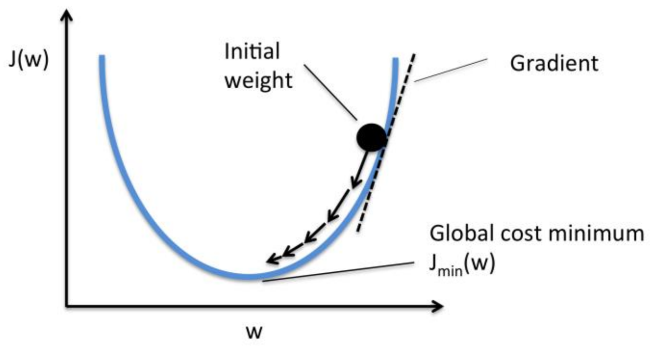

Figure 4. Gradient Descent

Although it may not seem as impressive in one dimension, gradient descent is capable of speeding up in higher dimensions, compared to tediously trying all the weights in all given dimension, which will consume a lot of time.

Problem 2:

Algorithm describes or classifies random error when fitting a model to a set of training data resulting to poor generalization.

Objective 2:

Setting the optimized number of samples required to sufficiently train a deep neural network model for the model to learn or classify input data efficiently without the disturbance of faulty inputs or noise.

Solution:

To solve this problem we used a technique known as Regularization. One way to implement this is to add a term to our cost function that penalizes overly complex models.

Simulation

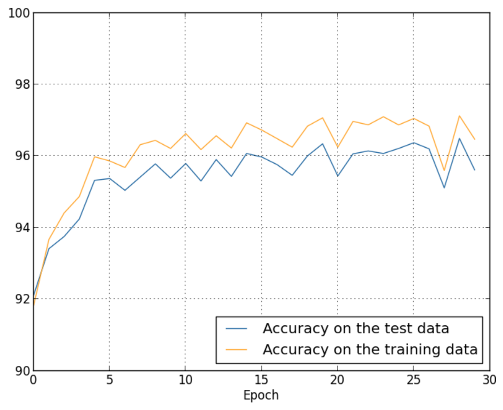

Figure 5. Regularization

A simple, but effective way to do this is to add together the square of weights to our cost function, this way, models with large magnitudes of weights cost more, normalizing the other parts of our cost function.

Problem 3:

Algorithm gets stuck and the network gets trapped into a local minimum caused by the neuron saturation layer resulting to poor error rate correction.

Objective 3:

Adjusting each training pattern to its own neuron in the hidden layer to adapt to the gained parameters during learning process. These adjustments are to be made to prevent the network from trapping into a local minimum caused by the neuron saturation in the hidden layer.

Solution:

. To avoid this situation, we use a momentum term, which is a value between 0 and 1 that increases the size of the steps taken towards the minimum by trying to jump from a local minima.

Simulation

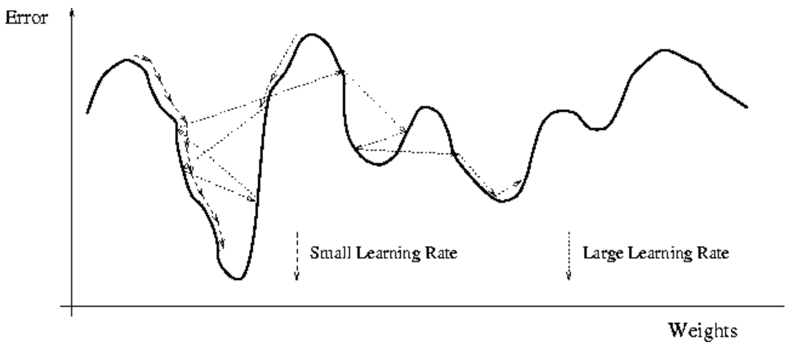

Figure 6. Momentum

Momentum also helps in smoothing out the variations, if the gradient keeps changing direction. A right value of momentum can be either learned by hit and trial or cross-validation.

B. Overview of Current Technologies

Goal of a personal assistant software is to act as an interface into the digital world by understanding user requests or commands and then translating into actions or recommendations based on agent's understanding of the world. This understanding of the world is modeled in a knowledge base that contains relationships, connections and rules between various concepts of the world. These agents, at least at present are not expected to replace humans but can be delegated mundane tasks that user would otherwise not be interested in doing or efficiently doing these mundane tasks by processing large amounts of relevant and real-time information on the web. There have been multiple approaches to building personal assistant software, based on how the user enters tasks and how the system interprets them.

Voice recognition as input entry medium

In this category of personal assistant software, focus is relieving the user of entering text input and using voice as primary means of user input. Agent then applies voice recognition algorithms to this input and records the input. It may then use this input to call one of the personal information management applications such as task list or calendar to record a new entry. ReQall application covered later in the chapter falls into this category of personal assistant software.

Voice recognition based task automation or information retrieval

In this category of personal assistant software, focus is on capturing the user input thru voice, recognizing the input and then executing the tasks if the agent understands the task. Software takes this input in natural language, and so makes it easier for the user to input what he or she desires to be done. SIRI and Google Glass fall under this category of software.

Before discussing the actual algorithm or method to be used in the development phase, this section will briefly discuss a typical IPA Algorithm and how it works. This concept is important to fully understand the relevance of the to-be-discussed core algorithms and how will it all be combined to build a development platform.

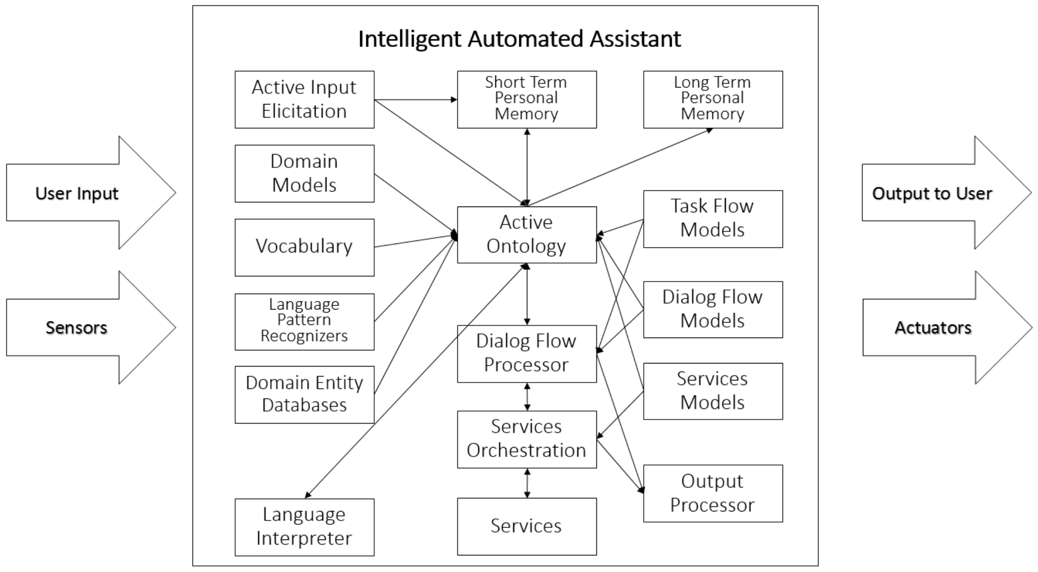

Figure 7. A typical Intelligent Personal Assistant Architecture

On the above figure, we can see how many components work separately inside an IPA Core. Language Processing, Vocabularies, Dialog Management etc. act as separate modules and the tedious tasks of incorporating and making them work altogether is the burden of the developer.

C. Overview of Technologies Used

Text To Speech - Mimic is a fast, lightweight Text-to-speech engine developed by Mycroft A.I. and VocaliD, based on Carnegie Mellon University's FLITE software. Mimic takes in text and reads it out loud to create a high quality voice. Mimic's low-latency, small resource footprint, and good quality voices set it apart from other open source text-to-speech projects. Mimic is going to be used for the voice of Intelora as its small footprint makes it a very attractive solution for running not only on the Intelora embedded device, but on multiple other platforms as well. We are working to make an A.I. that sounds great, and Mimic is a great step forward on that front.

Speech To Text - OpenSTT project is aimed at creating an open source speech-to-text model that can be used by individuals and company to allow for high accuracy, low-latency conversion of speech into text. Currently there are no open source speech-to-text models available, instead this technology is locked deep within large companies either tied to only their own proprietary products and services or behind expensive APIs that, in many cases, don't respect user privacy.

Intent Parser - The Adapt Intent Parser is an open source software library for converting natural language into machine readable data structures. Adapt is lightweight and streamlined and is designed to run on devices with limited computing resources. Adapt takes in natural language and outputs a data structure that includes the intent, a match probability, a tagged list of entities. The software was developed at Mycroft AI by a team led by Sean Fitzgerald, formerly one of the developers of both Siri and Amazon Echo.

D. Requirement Analysis

The usability of the agent is one of the high priority requirements for a smart agent application. It needs to be more helpful to the user and ask for fewer inputs. In addition, it needs to support input in regular English language.

1. Supporting voice recognition: Ideally, a smart agent needs to support voice interface in order to make the agent useful while the user is unable to type the request on the device. But, this requirement was purposely out of scope in order to focus more on the other aspects of building such an application while assuming that agent can later integrate with a voice recognition technology which can convert spoken text into textual commands for the agent to process.
2. User input should be processed to identify the task to be performed. Define templates for the supported tasks. These templates should help the user to type required information for the task so that agent can identify the task and related parameters.

Understanding the task at hand – Traditional task list applications treat the tasks as strings and do not associate any semantics to the task. The smart agent should exhibit an understanding of the task entered thru the system. This is a critical requirement as it enables the agent to use this understanding of the task to plan for the task.

Identifying the context of the user for any given task enables the agent to better serve the user's intended goal and possibly asks fewer questions to plan for the task. There are multiple contexts associated with an agent

1. Location Context: This context provides current location as the input and lets agent reduce the search path for a given task. This location context refers to low level geo coordinates offered by GPS sensors in mobile devices as well as mapping these co-

ordinates to higher level entities such as home, office, airport, mall etc. Location context information coupled with understanding of the task helps the agent to identify type of tasks that can be or cannot be accomplished at this location.

2. Time Context: Time context for an agent includes an understanding of common time terminologies such as morning, afternoon, evening, today, yesterday, tomorrow etc., and using current time or time of the task to filter tasks relevant in this context.

3 Social Context: This refers to understanding of social interactions between the user and other people in his network. With this contextual information, agent will be able to service requests based on a specific relation mentioned in the task.

4. Understanding external environment context: Agent needs to plan for tasks that interact with the external world. So, any changes in external world such as weather conditions, traffic conditions, delays in other people schedules should be coordinated and provided as input to the agent for planning the execution of tasks

5. Delegating tasks: Each task will be either a manual task or an automated one. Agent should be able to identify the type of the task and execute automated tasks with the information entered by the user and using contextual information of the user within in the agent. Agent should be able to identify number of actors for manual tasks and should try to identify if the user or the agent can be the default executioner of the task. If it’s a manual task, agent should identify if the user generally completes it or delegates to others and use this information to assign a default delegate.

E. Design Principles

Modularity - A system with modular architecture can be decomposed into well-defined subsystems. A modular subsystem with well-defined interfaces explicitly specifies expected inputs and outputs of the system and wraps all the complexity associated with implementation internally and hides it from the subsystems interfacing to it. This also reduces coupling between the subsystems.

A modular architecture leads to higher reuse of existing components, adaptability to change, shorter time to market, scalability of product design and reliable testing cycles.

Extensibility - System should designed such that new features can be added to the system as well as existing features can be modified without a major overhaul to the architecture and ideally by few simple changes. Modular architecture and loose coupling between sub-systems plays a major role in ensuring an extensible system.

Complexity - A good architecture takes into consideration performance and quality attributes of the system and also considers all the -ilities such as extensibility, modularity, scalability, and availability etc., expected of the system. Complexity of the system is thus driven by the essential functionality that need to be implemented to satisfy essential user needs and by these performance and quality attributes that need to be supported by the system. A good architecture will be able to deliver on all these promises by choosing concepts with low essential complexity and by using abstraction, decomposition, hierarchy and recursion to keep the actual complexity to the essential complexity.

Falling back to the user in case of ambiguity or unknowns - System should be architected such that in case system is subjected unsupported cases then it should respond gracefully, and in case of personal assistant, should fall back on user for clarifications or choose from available choices. This introduces human component back into the system, reducing complexity that would be otherwise required for the system.

Integrating with external data sources and Services - An architecture that offloads functionality to external components such as web services or external data sources simplifies the design of the system itself at the same time leveraging the expertise of the external systems. It also introduces dependencies on these external systems and fallback mechanisms should be added in order to use alternative paths when one of the dependent paths is not available.

F. System Architecture

After thorough analyzation and research, we conceptualized and developed Intelora, a platform that will simplify the above stated processes. This is done by grouping the three main components (Speech Recognition, Language Understanding, and Speech Synthesis) and combining them to develop a core which will act as the central processing module and message bus. External and Add-On modules and services will be then incorporated by accessing the Platform's API.

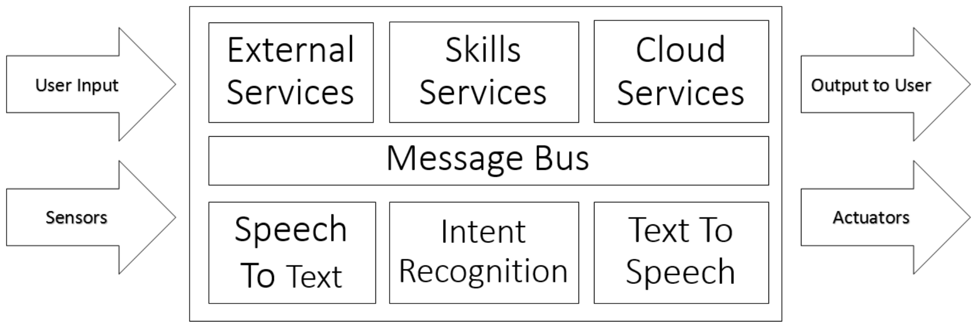

Figure 8. Intelora Core Architecture

Intelora Core is the primary module that makes up the Intelora Intelligent Personal Assistant Development Platform. Intelora makes use of the Adapt Intent Parser, OpenSTT, and Mimic.

G. Platform Services and Components

Intelora Core contains four main services - the messagebus, the skills service, the speech client, and the command line interface. Only one of the command line client and speech client need to be run, as they both provide methods of interacting with the Intelora Platform. This section will cover what these services do on a general level.

Process:

1. A user says a phrase including the wake word
2. The speech client strips out the wake word and sends the audio to a speech to text service
3. The speech to text service sends back the parsed text, which is sent on the messagebus by the speech client
4. Adapt gets the text and tries to match it to an intent registered by one of the skills
5. If it succeeds, the intent it finds is sent on the messagebus, along with what keywords it found
6. The skills service picks up the message and gives it to the skill matching the intent
7. The skill processes the message, does some action with it, and sends any message to be spoken on the messagebus
8. The speech client takes the message and sends it to a text to speech service.

The command line client essentially shortcuts directly to sending the text to Adapt, as there is no speech to text service required. It also handles sending a speak message to the chosen text to speech service, like the speech client does.

Messagebus

The messagebus is how all the different parts of Intelora communicate between each other. It does this by sending different types of messages that contain needed information. For example, one part of Intelora could send a speak message that contains something for the IPA to say. The message would be carried over the messagebus, and any other service connected to it could read the message and parse it. In the core, the speech or command line client would pick up the message and then convert it into speech. The messagebus is also hosted as a web socket that other things can communicate with. By default, this web socket is set on localhost:8000/events/ws.

Speech Client

There are a few components to the speech client. There is the AudioConsumer, AudioProducer, and within that the listener. There exists a shared queue of audio files between the consumer and producer. The producer grabs segments of audio from the listener and the consumer uses the audio to send to an external STT engine and emit the result as an utterance.

Listener

The listener figures out what pieces of audio Intelora should record and respond to. It first detects its wake word using pocketsphinx on around the last two seconds of audio. Audio older than two seconds is discarded. After detecting its wake word, it begins recording until when it thinks the user has finished speaking. This is estimated using a heuristic described below.

Phrase Heuristic

Essentially, the listener records until the "noise level" falls at or beneath 0. The "noise level" is a sort of estimate of the relative noise in the past few seconds. Every time the noise is above a certain threshold, the "noise level" increases. On the contrary, the opposite happens when the noise is lower than the threshold. By keeping track of this number, small separated noises will not affect the noise level as much as long contiguous noises most likely indicative of voice.

Skills Service

The skills service has two main components - the skills themselves and the Adapt intent parser.

Skills

A skill consists of vocabulary files (a list of phrases or words that the IPA should respond to), dialog files (a list of phrases that the IPA will say in response to something in the skill), and a Python file. This Python file determines how the vocabulary maps to each intent, as well as what actions the IPA will take when an intent is found (ranging from saying something to turning off the lights).

Adapt

In short, adapt takes in a sentence and attempts to find what the intent of it is by looking for matching words and phrases from intents and vocab registered by different skills.

H. Usage and Features

The idea behind the platform is to provide developers and enthusiasts a way to develop and customize their own Intelligent Personal Assistant from ground up. Users need to build and setup the system from the ground up.

The Intelora platform offers great advantages:

The first and greatest of which is its permissive, open source license. Intelora allows developers and users to take, extend, and integrate the software anyway they want.

The goal of Intelora is to allow the software to run on the lightest hardware possible. One of the goals is to allow Intelora to run in a wide variety of everyday devices such as mobile phones, desktops, laptops and embedded systems.

It is our hope is for Intelora to reach many developers and contributors to ensure that the future of artificial intelligence is open for all.

Thru Intelora you can develop your own IPA that is...

Always Learning. Intelora uses open software to process natural language, determine your intent and act. As more and more people adopt Intelora and develop modules the platform will learn to be more capable, more useful and more fun.

Always Changing. As more programmers and makers find new things they want Intelora to do, we'll share their hacks, tweaks, extensions and modules with the Intelora community through a web-based repository.

Always Listening. Intelora is always listening. When you call your assistant, it will answer.

Thru Intelora you can develop your own IPA that can...

Increase Productivity. Integrate a host of professional functions – Control scenes to conserve power, grant office access with your voice, and get that coffee rolling! Intelora is the future – a voice activated AI controlling your professional Internet of Things.

Encourage Creativity. Makers everywhere can use Intelora up, add components, change software, extend and hack it. Whether Intelora is in your garage or your local makerspace you will benefit from an intelligent always on interface to your next project. Make Your Home Smart. Naturally control all your media and devices with the sound of your voice. Adjust your thermostat, turn on your lights, water your lawn, play your favorite movie. Just speak naturally and your IPA's there to do your bidding.