**CHAPTER III**

Technical Background

**Hardware and Peripherals**

The following are the hardware/peripherals that is used on the development of this platform. Similar hardware/peripherals may be used interchangeably.

**1. Laptop**

Processor

Intel Core i3-4005U Processor (3M Cache, 1.70 GHz)

Graphics adapter

Integrated Intel HD Graphics 4400

Memory

500 GB

4 DDR3L 1600 MHz SDRAM, 1 x SO-DIMM socket for expansion up to 8 GB SDRAM Display

14-inch 16:9 HD (1366x768) LED Backlit

**2. Plug and Play External Desktop Microphone**

Any generic brand of USB Desktop Microphones for audio capture.

**3. Plug and Play External Speakers**

Any generic brand of 3.5mm speakers for audio output.

**Software and Technologies**

**1. Google Speech API**

Google Cloud Speech API enables developers to convert audio to text by applying powerful neural network models in an easy to use API. The API recognizes over 110 languages and variants, to support your global user base. You can transcribe the text of users dictating to an application’s microphone, enable command-and-control through voice, or transcribe audio files, among many other use cases. Recognize audio uploaded in the request, and integrate with your audio storage on Google Cloud Storage, by using the same technology Google uses to power its own products.

**Frame size**

Streaming recognition recognizes live audio as it is captured from a microphone or other audio source. The audio stream is split into frames and sent in consecutive **StreamingRecognizeRequest** messages. Any frame size is acceptable. Larger frames are more efficient, but add latency. A 100-millisecond frame size is recommended as a good tradeoff between latency and efficiency.

**Audio pre-processing**

It's best to provide audio that is as clean as possible by using a good quality and well-positioned microphone. However, applying noise-reduction signal processing to the audio before sending it to the service typically reduces recognition accuracy. The service is designed to handle noisy audio.

Below is an example of speech to text transcription of Google Speech API:

* The audio level should be calibrated so that the input signal does not clip, and peak speech audio levels reach approximately -20 to -10 dBFS.
* The device should exhibit approximately "flat" amplitude versus frequency characteristics (+- 3 dB 100 Hz to 8000 Hz).
* Total harmonic distortion should be less than 1% from 100 Hz to 8000 Hz at 90 dB SPL input level.

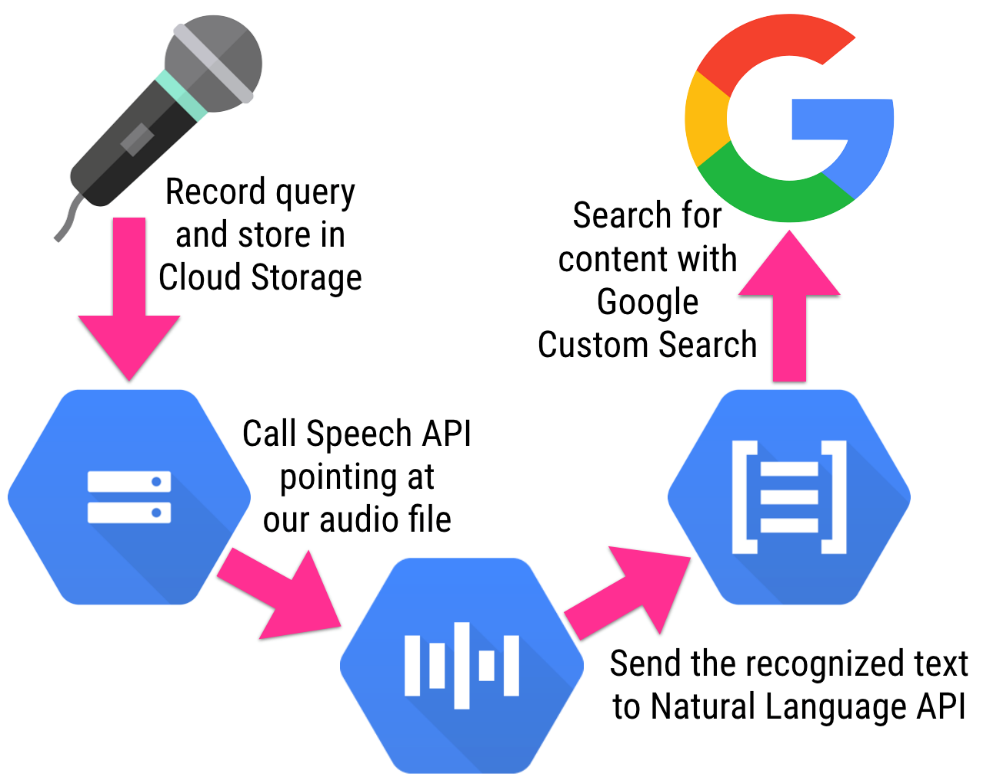


Fig 1.0 Google Speech API

**2. Adapt Pattern Based Text Intent and Entity Extraction**

The **Adapt Intent Parser** is a flexible and extensible intent definition and determination framework. It is intended to parse natural language text into a structured intent that can then be invoked programatically. It 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.

Adapt is a rules-based artificial intelligence library that is useful for interpreting natural language input. For example, a user might want to create a natural language interface that allows them to play a Pandora station. The user might say "Turn on Pandora", or "Play Pandora", or "Put on my Joan Jett Pandora station."

Adapt has many different parts to it, but the core engine uses a greedy and naive implementation for intent determination and domain determination.

**Naive Implementation** just means that it uses a brute force approach to find the best intent.

An algorithm is said to be naive when it is simple and straightforward but does not exhibit a desirable level of efficiency (usually in terms of time, but also possibly memory) despite finding a correct solution or it does not find an optimal solution to an optimization problem. Naive algorithms are easy to discover, often easy to prove correct, and often immediately obvious to the problem solver. They are often based on simple simulation or on brute force generation of candidate solutions with little or no attempt at optimization.

**Naive implementation pattern matching**

- No pre-processing is done

- Comparison is done in left to right order. On the characters in two arrays

- When mismatch occurs, pattern is moved one position to the right with respect to the text.

**Few applications**

- In text editing programs where a particular word is supplied by the user and is matched in a given text using pattern matching process

- In DNA Sequence to match the strands of DNA of two organisms and find how similar the two organisms are.

**Greedy Algorithm** means it only looks at the choices locally (or closest to it) and then chooses the best choice it sees.

A greedy algorithm, as the name suggests, always makes the choice that seems to be the best at that moment. This means that it makes a locally-optimal choice in the hope that this choice will lead to a globally-optimal solution.

Overview of Adapt Intent and Entity Extraction Algorithm:

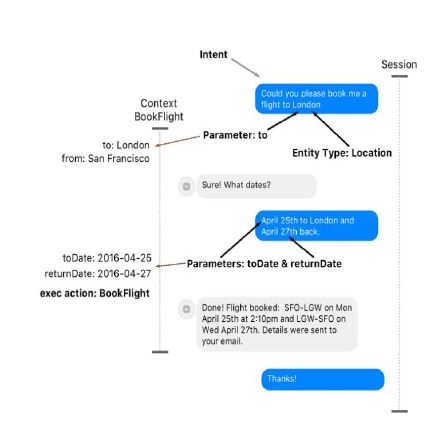


Fig 2.0 Intent and Entity Extraction

**Input acceptance**

1. Natural Language Text Input



Fig 3.0 Text Input

**Text analysis**

1. Tokenize text to separate segments

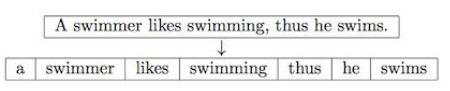


Fig 4.0 Text Tokenization

**Intent recognition**

1. Identify and extract entities based from provided pattern

2. Match text input to a defined pattern in database and recognize intent

**Export to machine readable format**

1. Intent and Entities will be formatted and exported in a machine-readable format such as XML or JSON

Fig 5.0 Exporting to machine readable format

**3. Mimic Text to Speech Synthesis**

Mimic is a fast, lightweight Text-to-speech engine 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 a powerful tool that can also help solve other important problems. Mimic works on Linux, Android & Windows and we are working on iOS support. We are also adding more languages to enable many people to access realistic voices for the first time.

It is a speech synthesizer algorithm that uses a formant synthesis method, providing many languages in a small size. In addition, it can be used as a front-end, providing text-to-phoneme translation.

Mimic Text to Speech Synthesis Algorithm:

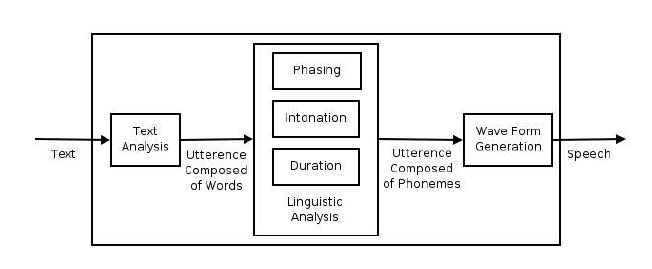


Fig 6.0 eSpeak Speech Synthesis

**Text analysis**

1. Text and symbols will be tokenized

2. After tokenization, words will be syllabicated and symbols will be evaluated to extract the phonemes

3. Phonemes will be given indices to allow programmability and matching

**Phoneme matching**

1. Phonemes extracted from the text will be individually matched to existing sinusoidal phones in the database

**Phone concatenation**

1. Matched phones will then be concatenated to form the sound synthesis

**Programming Language**

**Python**

Python is a widely used high-level programming language for general-purpose programming. Python features a dynamic type system and automatic memory management and supports multiple programming paradigms, including object-oriented, imperative, functional programming, and procedural styles. It has a large and comprehensive standard library.

Python is a multi-paradigm programming language: object-oriented programming and structured programming are fully supported, and many language features support functional programming and aspect-oriented programming (including by metaprogramming and metaobjects (magic methods)). Many other paradigms are supported via extensions, including design by contract and logic programming.

Python uses dynamic typing and a mix of reference counting and a cycle-detecting garbage collector for memory management. An important feature of Python is dynamic name resolution (late binding), which binds method and variable names during program execution.

The design of Python offers some support for functional programming in the Lisp tradition. The language has filter(), map(), and reduce() functions; list comprehensions, dictionaries, and sets; and generator expressions. The standard library has two modules (itertools and functools) that implement functional tools borrowed from Haskell and Standard ML.

**Software Packages and Libraries:**

The following packages are required for setting up the development environment:

**git**

**python 2**

**python-setuptools**

**python-virtualenv**

**pygobject**

**virtualenvwrapper**

**libtool**

**libffi**

**openssl**

**autoconf**

**bison**

**swig**

**glib2.0**

**s3cmd**

**portaudio19**

**mpg12**

The above-mentioned software packages are all specifically for Linux Based Operating System.

**Operating System (Linux Based)**

**Ubuntu 17.04 (Zesty Zapus)**

Ubuntu is an open source [operating system](https://en.wikipedia.org/wiki/Operating_system) software for computers. It is one of the distribution systems of [Linux](https://en.wikipedia.org/wiki/Linux), and is based on the [Debian](https://en.wikipedia.org/wiki/Debian) architecture. It is usually run on [personal computers](https://en.wikipedia.org/wiki/Personal_computer), and is also popular on [network servers](https://en.wikipedia.org/wiki/Network_servers), usually running the Ubuntu Server [variant](https://en.wikipedia.org/wiki/Ubuntu_variants), with enterprise-class features. Ubuntu runs on the most popular architectures, including Intel, AMD, and ARM-based machines. Ubuntu is also available for [tablets](https://en.wikipedia.org/wiki/Tablet_computers) and [smartphones](https://en.wikipedia.org/wiki/Smartphone), with the [Ubuntu Touch](https://en.wikipedia.org/wiki/Ubuntu_Touch) edition.

Ubuntu is published by [Canonical Ltd](https://en.wikipedia.org/wiki/Canonical_(company)), who offer commercial support. It is based on [free software](https://en.wikipedia.org/wiki/Free_software) and named after the Southern African philosophy of [ubuntu](https://en.wikipedia.org/wiki/Ubuntu_(philosophy)) (literally, 'human-ness'), which Canonical Ltd. suggests can be loosely translated as "humanity to others" or "I am what I am because of who we all are".

Ubuntu is the most popular operating system running in hosted environments, so–called "[clouds](https://en.wikipedia.org/wiki/Cloud_computing)", as it is the most popular server [Linux distribution](https://en.wikipedia.org/wiki/Linux_distribution).

Development of Ubuntu is led by UK-based Canonical Ltd., a company founded by South African entrepreneur [Mark Shuttleworth](https://en.wikipedia.org/wiki/Mark_Shuttleworth). Canonical generates revenue through the sale of [technical support](https://en.wikipedia.org/wiki/Technical_support) and other services related to Ubuntu. The Ubuntu project is publicly committed to the principles of [open-source software development](https://en.wikipedia.org/wiki/Open-source_software_development); people are encouraged to use [free software](https://en.wikipedia.org/wiki/Free_software), study how it works, improve upon it, and distribute it.