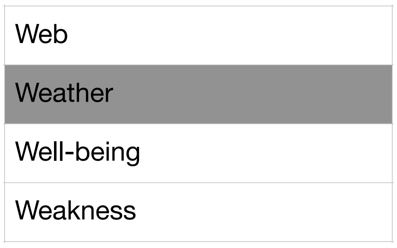
Front-End Speech/Text Input :

- Fairly straightforward , input is taken through mic or keyboard.

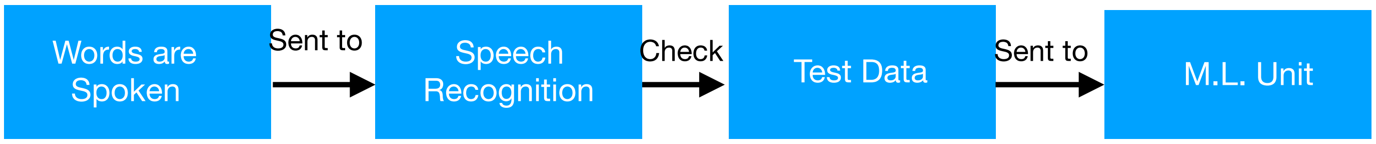
- The keyboard input will be vanilla for the most part, albeit adding autocompletion using the words off of a downloaded and modified dictionary is worth doing.

<CAIN> CAIN, what is the we



- Use <tab> to cycle through and then hit <space> to confirm and put a space into the input text.

- As for speech recognition, I'm using the SpeechRecognition library in Python (downloadable using pip); a good idea would be to use TensorFlow to implement Machine Learning though Neural Networks in order to improve accuracy of user base.



Check for Speech Synthesis Requirement :

- Rather small library, however this section of CAIN's functionality is put in this library in order to improve the readability of the code and make it easier to fix bugs.

- There are two ways to execute Speech Synthesis requirement check :

    1. CAIN listens for commands continuously but processes it only when we call its name (like saying "OK Google" or "Hey Siri"), in this case, "CAIN, ..."
    2. CAIN listens and processes command continuously until the user says "stop listening".

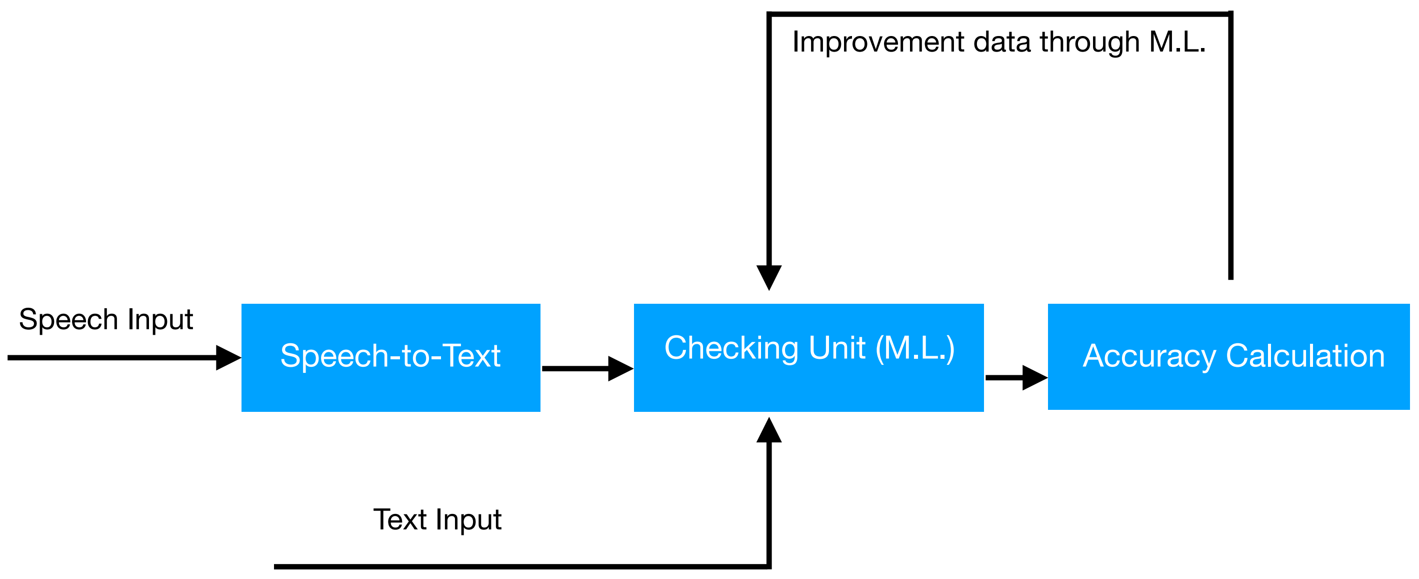
- As for text, we're using a rather simplistic approach, as soon as the user starts typing, the input mode is automatically set to "Text Input" until and unless the user types and executes the command "listen".

<u>**Speech-to-Text Conversion :**</u>

- Using a standard Python SpeechtoText library.

- This library can be fed through an M.L. unit to improve the accuracy of perceived words. This is necessary in my opinion as most default Speech-to-Text libraries are very generalised and often fail to catch the nuances in the different dictions of people in any specific area.

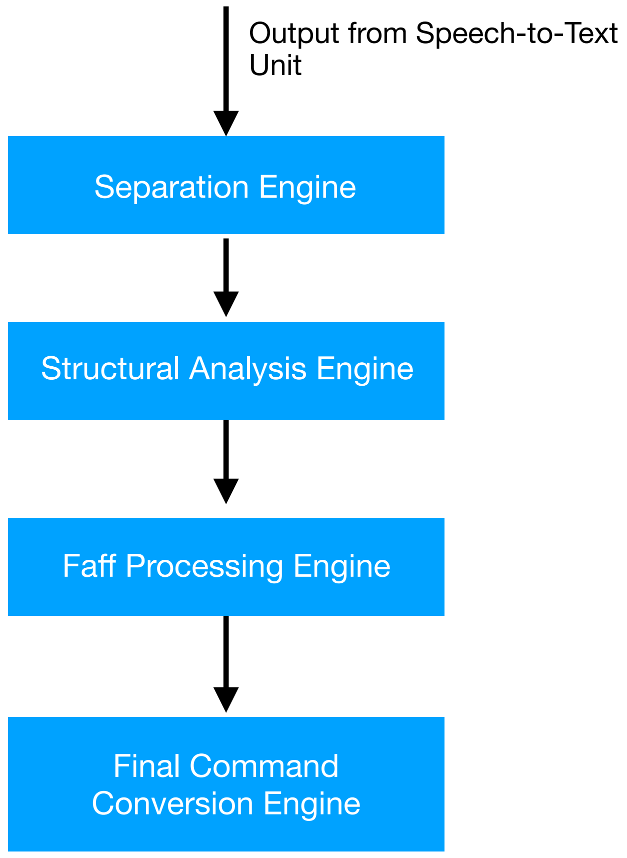
- It'd be a good idea to have around 99.6% accuracy or greater in order to avoid mishaps, if one has enough time and data to feed to the M.L. Unit that is. The diagram on how this can be achieved manually is as follows :



- The above procedure will of course only be carried out in testing. Once the desired level of accuracy has been reached, it'll be implemented as a vanilla Speech-to-Text unit.

Semantics Processing Unit (SPU) :

- This is the MOST complex unit of CAIN.
- **NOTE :** Natural Language Processing was not used here as it increases file-size by a lot as well as reduces customisability and it is not required for smaller scale projects for the like of which CAIN was created.
- The SPU will function in a layer-by-layer fashion :


Output from Speech-to-Text Unit
Separation Engine
Structural Analysis Engine
Faff Processing Engine
Final Command Conversion Engine

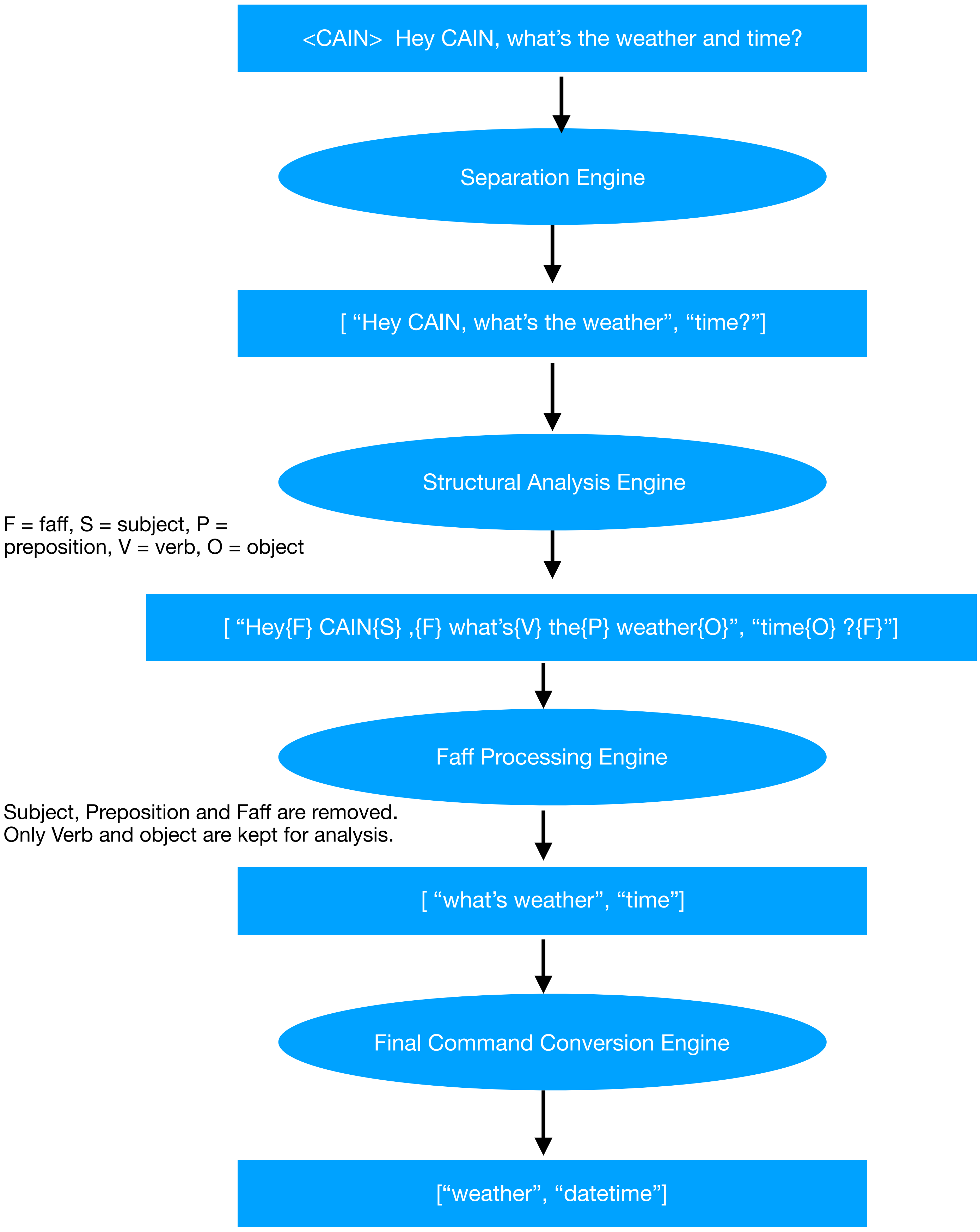
- The functionality of these layers is as follows :
  1. **Separation Engine :**
     This engine will separate the input string into multiple sub-strings that will be stored in a string array. The strings shall be separated when phrases like "and" or "then" or "followed by" etc. are used.

  2. **Structural Analysis Engine :**
     This engine simply identifies the subject, verb and object (if any) in the substrings. This is achieved using dictionaries to find the verbs and subjects and then using the acquired data to find the object.

  3. **Faff Processing Engine :**
     This engine filters out faff (words that aren't needed or are useless) from our substrings to make things easier on the final unit.

  4. **Final Command Conversion Engine :**
     This engine finally converts what is left into command-level syntax and sends
it to the Command Processing Unit (CPU).

- An example of the working of these layers in the SPU, is as follows :


<CAIN> Hey CAIN, what’s the weather and time?
Separation Engine
[“Hey CAIN, what’s the weather”, “time?”]
Structural Analysis Engine
F = faff, S = subject, P = preposition, V = verb, O = object
[“Hey{F} CAIN{S} ,{F} what’s{V} the{P} weather{O}”, “time{O} ?{F}”]
Faff Processing Engine
Subject, Preposition and Faff are removed. Only Verb and object are kept for analysis.
[“what’s weather”, “time”]
Final Command Conversion Engine
[“weather”, “datetime”]

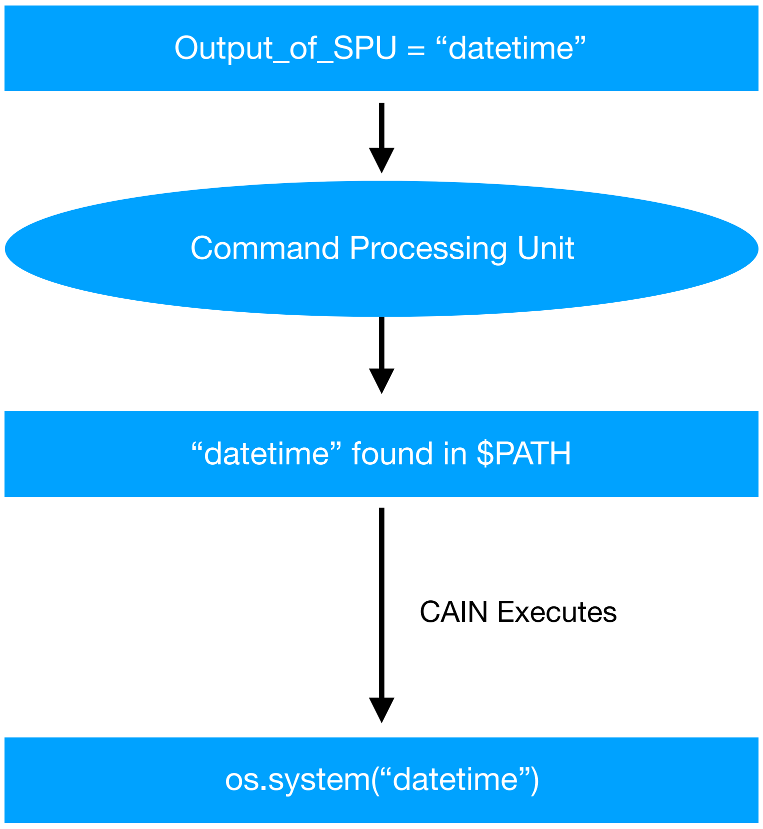
Command Processing Unit (CPU) :

- A relatively small unit.
- It will simply look for files to execute in a list of directories called “CAIN paths”.

  Example : CAIN_paths = [“/usr/lib/python3.6/“, “/home/usr/CAIN/”]

- If the desired filename is not found, it checks the PATH environment variable of the operating system.

  Example :



- The output is stored in a string and then a class object of CAIN_output is made to output to terminal.

Output to Terminal :

- Smallest library.
- A simple class object is made using the output string and specifying type of output (either speech or text, this information will be carried through the other units to this unit) and a responder function is invoked in order to give output to shell.