OSLO: An Open Source Voice Assistant

Aidan M. Gomez

January 2023

1 Introduction

OSLO, which stands for Open Source Loop Operator, is an attempt to create a state-of-the-art, open source voice assistant that is entirely server-less. A virtual assistant is defined as, for the purpose of this manuscript, a program which interacts with the user in the way the user might interact with a human assistant such that *a.*, the assistant supports the user in accomplishing a task, or *b.* the assistant accomplishes an assigned task autonomously. A voice assistant is a virtual assistant which uses voice as the primary means of interaction. Contemporary examples of voice assistants include popular consumer products such as Apple’s Siri or Amazon’s Alexa. However, these examples differ from OSLO in that they are hosted - they do not run directly on the hardware purchased by consumers, rather the vast majority of computation is done off-device on private servers.

1.0.1 Author’s Note

It is important to note this is a preliminary document which records the justification and design of the agent. Some of the work contained within outlines algorithms which are non-standard, either novel in content or previously unknown to the author. As such they are not time-tested nor have they undergone sufficient review to be taken as anything other than experimental. Please direct any productive questions, corrections, or comments to Aidan M. Gomez at *agom@bu.edu.*

Between August of 2021 and September of 2022, funding and guidance for this project were graciously provided through Boston University’s Spark! Technology Innovation Fellowship. Special thanks to Chase Maivald for his contribution to the concatenative speech synthesis module.

1.1 Reasoning

Humanity generates a massive amount of data - in 2020 alone we read and wrote over 64 zettabytes of information[1], a figure which has only increased since. A growing share of internet-enabled devices are ’things’, consumer devices including washers, light bulbs, door locks, thermostats, and coffee machines.

As of 2020 there were about 15 billion of them[2]. They have become commonplace, most Americans own or simply make use of some form ”smart” (a relative description that ceaselessly shifts in its definition) speaker, television, or lighting system. These devices are oftentimes cheap and host services which permit the user to control their appliances from anywhere. While this hosting costs the company money, they do not charge the user in a ’subscription model’

[1] Taylor, P. (2023, August 22). Data Growth Worldwide 2010-2025. Statista. https://www.statista.com/statistics/871513/worldwide-data-created/

[2] Vailshery, L. S. (2023, July 27). IOT connected devices worldwide 2019-2030. Statista. https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/

style for access. While there may be exceptions, companies like Roku are able to sustain this business model because there is a substantial revenue stream to be had from harvesting your information. The exchange is this: the user purchases the device and the 'smart' capabilities are paid for by the data generated by the user. This is not the only cost the consumer incurs, however. So-called 'internet-of-things' (IoT) devices are notorious for exposing networks through outdated firmware. The Mirai botnet made use of over 600,000 poorly maintained IoT devices to DDoS servers for the popular video game "Minecraft" before moving on to critical DNS infrastructure. No matter how security conscious the user is, no assurance can be made that 'things' such as a smart toilet or doorbell will not suddenly be co-opted by a bad actor, potentially as a staging-point for more serious attacks or as a means to invade your privacy. Functionality can not be guaranteed by smart appliance companies either. In April of 2022, major home automation company Insteon shut down their servers[3] rendering users unable to do basic tasks like turn on their lights. Despite being able to communicate over a local network, phoning home to a data center potentially thousands of miles away was central to Insteon devices' functionality.

1.2 Proposed Solution

By keeping everything local, OSLO subverts the server oriented status quo of the virtual-assistant-home-automation market. In fact, OSLO does not seek whatsoever to compete with aforementioned market. In today's day and age, it is veritably impossible to unseat a FAANG (Facebook, Apple, Amazon, Netflix, Google) company from a market in which they already possess a the majority user share, a core concept of modern platform economics[4]. Instead, the project is dedicated to an entirely open-source development approach, free to use, modify, and distribute, so long as it is not sold. Further description of the myriad benefits of open source are beyond the scope of this manuscript, which seeks to provide only a general technical introduction to the project as well as its development process.

The project seeks to take advantage of the personal nature of the collected information to further customize interactions. This addresses both the instability as well as the lack of privacy of a server-based model. Much of OSLO's architecture is based on relatively metaphysical arguments regarding cognition, in an attempt to directly duplicate behaviors common in intelligent animals, without waiting for them to arise from a neural network.[5]

[3] Scharon Harding Jun 10, 2022 4:19 pm UTC. (2022, June 10). Insteon Smart Homes resurrected as abruptly as they were bricked. Ars Technica. https://arstechnica.com/gadgets/2022/06/insteon-smart-homes-resurrected-as-abruptly-as-they-were-bricked/

[4] Colback, L. (2023, March 13). The rise of the platform economy. Financial Times. https://www.ft.com/content/e5f5e5b9-3aec-439a-b917-7267a08d320f

[5] It is important to note OSLO pre-dates current large language models

2 Architecture: Introduction

OSLO is a ’human-in-the-loop’ system written in primarily in Python3, although certain features were implemented in C++ or Rust. ’Human-in-the-loop’ does not refer to the eponymous loop, but rather refers to the a system which uses human feedback as an integral step in its execution. This feedback comes during runtime rather than afterwards or beforehand, as is common with other forms of machine learning.

2.1 A ”Loop”

For the purposes of this manuscript, a ’loop’ is defined as a repetitive process within a heterarchical system. Loops take the form of generators, taking an input and transform it. Generators effectively ’pause’ execution, allowing a model to be loaded into memory and kept there, and then called from anywhere within appropriate scope.

```
#example loop
def load_model():
    model = library.load("model_name")
    return(model)

def conversion(input, model):
    o = model.inference(input)
    return(o)

def example_loop():
    model = load_model()
    yield(True)
    while(True):
        #loop input
        val = yield
        if(val is not None):
            pred = conversion(val, model)
            #loop output
            yield(pred)
        else:
            pass

exg = example_gen()
genImp = True
while(True):
    flush = next(exg)
    genOut = exg.send(genImp)
```

See the generator implemented above as an example. An input may be unstructured data for which the output is a parse-able format, or structured data, for which the output is an inference or transformation of said data.

A key benefit of this approach is the flexibility afforded in data ingress and egress. Data may enter the system at any point in its execution, for example, audio enters only through a loop dedicated to audio processing, which in the system-wide loop, comes before text processing. Textual data which could take the form of something like a webpage, SMS, or API response enters in the text loop, later in the execution of the system-wide loop. Different modalities may pass through sets of loops prior to the system-wide loop until they are standardized and continue alongside other information, which may stem from any other input modality.

3 Executive Loop: Decision-making, executive function

3.1 Causal Model

In *Relativity Theory and Time Perception: Single or Multiple Clocks?* Buhusi and Meck posit that in rats, the neurobiological basis of the temporal perception is not that of a single continuous scale on which things are ordered, but rather relative via several different clocks of varying lengths. [BM09]

Hume's definition of causality presented in *A Treatise On Human Nature* [Hum78] states that a cause and effect must be "contiguous in space and time", "the cause must be prior to the effect", and the "same cause always produces the same effect, and the same effect never arises but from the same cause", among other things. These laws, while substantive as a basis for causality in some circumstances, are not universally true as they imply a deterministic, constant, singular outcome from any specific action. For the purpose of this model Hume's laws permit a few assumptions to be made.

1. Cause precedes effect → The graph must contain information about the order in which states occurred. This is obtainable through a directed graph.

2. Cause must be contiguous to effect → Assuming that any arbitrary state S_a can lead to any other arbitrary state S_b, the graph must be well connected.

3. The same cause always produces the same effect → It is bad practice to set any probability to 0 as would be implied by 'always'. A weight representing the probability of state transition should be appended to each edge in the graph.

3.1.1 Core Principle

Assume a traditional finite Markov chain[1] with state space Ω and transition matrix P of size $|\Omega| \times |\Omega|$ which utilizes *discrete* time steps; $t, t+1, ..., t+n$, where $n \in \mathbb{Z}$, and P_{ij} represents the probability that, given $X_t = \Omega_i, X_{t+1} = \Omega_j$ ie,

$$Pr(X_{t+1} = \Omega_j | X_t = \Omega_i)$$

[1] Further reading & better definition: https://math.uchicago.edu/ may/REU2017/REUPapers/Freedman.pdf

We attempt to construct a causal model which applies neurobiological concepts from Buhusi & Meck's paper. We define four constants:

$$s = 10$$

$$m = 60$$

$$l = 3600$$

$$o = 43200$$

which represent, in seconds, the time windows for four clocks, *short, medium, long, and extra-long*, and another variable, T', which represents the actual clock time elapsed between state changes. T now iterates *only* when a new state is entered. While the agent idles in any given state, variable T' increments independently of T.

Assume a new Markov chain with state space Ω and transition matrix P of size $|\Omega| \times |\Omega| \times 4$. P_{ijk} now represents the probabilities,

$$k = 1, Pr(X_{t+1} = \Omega_j \cap t' \leq s | X_t = \Omega_i)$$

$$k = 2, Pr(X_{t+1} = \Omega_j \cap t' \leq m | X_t = \Omega_i)$$

$$k = 3, Pr(X_{t+1} = \Omega_j \cap t' \leq l | X_t = \Omega_i)$$

$$k = 3, Pr(X_{t+1} = \Omega_j \cap t' \leq o | X_t = \Omega_i)$$

With no data available, at $t' = 0$ and $t = 0$

$$P(.,.,s), P(.,.,m), P(.,.,l), P(.,.,o) = \frac{1}{|\Omega| \times 4}$$

so that

$$\Sigma P_{i,.,.} = 1.0$$

During run-time the agent is able to update the transition matrix as new data becomes available. Given a corpus of historical state transfers C, and a given transfer c with c_0 being the initial state, c_1 being the new state, and $c_{t'}$ being the time elapsed,

$$P_{ij} = \frac{|c_0 = i \cap c_1 = j \cap c_{t'} \in k|}{|\{c' \in C | c'_0 = i|\}}$$

The matrix remains irreducible and finite with both row and column sums of 1.

3.1.2 States & State Objects

A state takes one of three forms:

1. Action State: OSLO may trigger an action state internally through utilizing an 'action', a function within its action library. Examples of actions include toggling lights, retrieving online information, like the weather or a calendar, or controlling other connected endpoints.

2. Stimulus State: A stimulus state may not be triggered. It occurs when a continuously monitored input source fulfills a condition. For example, the iCloud extension reports your phone battery percentage is below 10 percent.

3. Reward State: A reward state may not be triggered, but it occurs when the agent accomplishes a programmatically specified reward, like providing a high confidence answer to a question.

Awareness of one's impact on their environment is a developmental stage in human cognition, outlined in *The Development of Body Self-Awareness* by Moore et al. [Moo+07] OSLO's division between actions & stimuli allows probabilistic reasoning of not only what may happen, but what can happen if OSLO interacts with the environment in a specific manner. This of course does not constitute self-awareness, but is designed to mimic this element of cognitive development.

States may possess typed parameters for use with propositional calculus (see 5.9). The action state *getWeather* possesses a *location* typed parameter, requiring an input tagged by the named entity recognizer as a *LOC*.

3.1.3 Causal 'Pointers'

The causal pointer is the programmatic means by which t' is recorded. Upon entering a state, the pointer begins to count upwards, disqualifying any expired time frames. All pointers are stored in a queue.

3.1.4 Reward States, Drive Values

A reward state is a state which may be triggered to indicate the agent has accomplished a goal. For each reward state, there is a 'drive' value, which degrades continuously with each passage through the main loop. These states may be programmatically specified, for example, the threshold which specifies when the 'curiosity'[2] routine kicks in. A reward state may not be an action state or a stimulus state.

4 Audio Loop: Audio

4.1 Recording

The Python3 library *SoundDevice* handles both device selection and sampling. Recording is done on a specified device or the first available compatible input device from a list of available audio devices. A block-size of 8000 is utilized with a sixteen bit integer representation. The sample rate is set to the default sample rate of the selected input device. Recording is done within a 'loop', passing an audio sample of the desired size to components responsible for processing and storage.

[2] OSLO retrieves the Wikipedia article for a noun chunk with few semantic web connections, or, a random noun chunk

4.2 Storage

Audio is stored using the *.h5* file format. Otherwise, the specifics of storage are mundane.

4.3 Processing

Vosk, the speech processing toolkit, is used for the speech to text task and the embedding portion of the speaker resolution task.

4.3.1 Speech to Text

The tested speech to text model, $vosk - model - en - us - 0.22$, has a mean word-error-rate of 5.69, although accuracy often significantly worsened in loud settings, when parsing with heavily accented speech, or handling multiple concurrent speakers. Accuracy either improved or remained faithful to the reported mean word-error-rate when a single unaccented speaker enunciated clearly, especially into a high quality microphone with third party audio filtering software (such as the defunct/compromised 'NoiseTorch' library) enabled.

4.3.2 Speaker Resolution

This task consists of partitioning the speaker embedding dataset into k clusters, where k is determined by the following methods.

It can be solved comprehensively, which is run when OSLO is not in use:

Let X be the set of all embedding vectors $\{x_0, x_1, ...\}$, where any given embedding vector x_q has a fixed length determined by the model used.

We assume that there is an underlying multivariate Gaussian distribution, defined as:

$$p(x, \mu, \Sigma) = N(\mu, \Sigma) = (2\pi)^{-(\frac{D}{2})} exp(-\frac{1}{2}(x - \mu)^T \Sigma^{-1}(x - \mu)$$

For which a multivariate Gaussian mixture model, defined below, can be fit:

$$N(\mu_k, \Sigma_k) = \Sigma_{k=1}^{K} \pi_k N(x|\mu_k, \Sigma_k)$$

The posterior distribution, the likelihood a particular embedding has been generated by component k, in this case, speaker k:

$$p(z_k = 1|k) = \frac{\pi_k N(x|\mu_k, \theta_k)}{\Sigma_{f=1}^{K} \pi_f N(x|\mu_f, \theta_f)}$$

The Bayes information criterion (BIC) and Akaike information criterion (AIC) are defined as follows:

$$AIC = -2ln(L^0(K) + 2k + \frac{2k(k+1)}{(N - k - 1)}$$

$$BIC = 2ln[(L^0(K)] + kln(N)$$

AIC & BIC may be applied to the Gaussian mixture model fit to X with K components, providing AIC_K and BIC_K, through $AIC_K = AIC(GMM(X, K))$, and $BIC_K = BIC(GMM(X, K))$
Using these assumptions,

$$I : 2, ..., |X|$$

$$m_{AIC} = min_{index}(\{\forall i \in I|AIC(GMM(X, i))\})$$

$$m_{BIC} = min_{index}(\{\forall i \in I|BIC(GMM(X, i))\})$$

m_{AIC} and m_{BIC} represent the K components at which the fit Gaussian mixture model has the minimum Bayes information criterion (BIC) and Akaike information criterion (AIC). These minimums can be averaged:

$$\hat{K} = \lfloor \frac{m_{AIC} + m_{BIC}}{2} \rfloor$$

Giving $\hat{K}$ components, in other terms, the $\hat{K}$ speakers. Using $GMM(X, \hat{K})$, the posterior distribution provides likelihood a sample belongs to speaker S.

Assuming X was of row-size A and column-size B, create $\hat{X}$ of new size A, B, $\hat{K}$, where $\hat{X}_{i,j}$ is of $\hat{K}$ length, with index $\hat{X}_{i,j,s}$ representing the probability $X_{i,j}$ belongs to speaker s.

As samples are added to X, it is necessary to update $\hat{X}$, so the current step $\hat{X}_T$ is updated at $\hat{X}_{T+1}$, and so on, but this should be done while maintaining the profiles calculated in a previous iteration. It is safe to assume the number of speakers, $\hat{K}$, over time is a monotonically increasing function. For the first $[0, |\hat{X}_{T+1}| - |\hat{X}_T|]$ points, there are now two sets of probabilities. For speaker s who has likely spoken $Y_s \in \hat{X}$, point membership can be mapped to a new Y'_s from based on a comparison between $\hat{X}_T$ and $\hat{X}_{T+1}$

The task can also be solved quickly[3], which is done when OSLO is actively in use:

$$SSE = \Sigma_{i=1}^{n}(y_i - f(x_i))^2$$

$$k_{min} = 2, k_{max} = |PCA(X^T)|$$

$$V = [k_{min}, k_{max}]$$

$$E = \{0, ..., 0\}, |E| = k_{max}$$

$\forall i \in V$, use a K Means clustering $\hat{Y} = kMeans(X, i)$, where the points belonging to cluster c are $\hat{Y}_c$, and calculating the sum of squared errors accordingly.

[3] https://gist.github.com/rpgove/0060ff3b656618e9136b

$$E[i] = E[i] + \Sigma_{p=0}^{|\hat{Y}_c|}\hat{Y}_{c,p} - \frac{\Sigma_{f=0}^{|\bar{Y}_c|}Y_{c,f}}{|\bar{Y}_c|}^2$$

Then, find $e_k \in E$, such that the following is maximized

$$\frac{\Sigma_{i=1}^{k}\sqrt{(e_i - e_{i-1})^2}}{|k} - \frac{\Sigma_{i=k}^{|E|}\sqrt{(e_i - e_{i-1})^2}}{|E| - k}$$

k is the new number of clusters, permitting $kMeans(X, k)$ for for point-speaker assignment, and classification in later iterations similarly

5 Natural Language Processing & Understanding

5.1 Ingress Points

All information OSLO encounters, whether it originates externally or internally, is converted into a natural language representation and fed into the language loop, providing a facsimile of a 'language of thought'. For example, the activation of State S_x is fed to the language processing loop as $Speaker : Profile_0$, $Text$: "I am currently entering state S_x", and an audio transcription matched to speaker i is fed to the loop as $Speaker : Profile_i$ $Text : < spokentext >$

Currently implemented external-origin ingress points include:
- SMS/MMS, via SMTP Gateways (@vtext.com, @txt.att.net, etc)
- Audio transcriptions
- Email, via SMTP
- Web sources, including Wikipedia
- API calls

Currently implemented internal-origin ingress points include:
- State activation
- Output from routinely called libraries (weather, traffic, etc)
- Output from routinely scheduled Wikipedia queries of known entities

5.2 Classification

5.3 Frames

The concept of frames is drawn directly from Marvin Minsky's "A Framework For Representing Knowledge" [97]

Oslo's text frames possess the following attributes: *type, sentiment-vector, plain-text, entities, tokens, chunks, speaker, tense*

5.3.1 Negation, Sentiment

Contemporary voice assistants are remarkably poor at parsing negation.[6] While methodology to test this is limited, as there are no true development versions of consumer voice assistants such as Siri or Alexa, anecdotal evidence suggests a query like *"Do NOT turn off the lights"* results in nearly any assistant turning the lights off.

The current architecture of these assistants is not public, but this is likely due to methodologies employed for intent matching which disregard the structure of the sentence and create embeddings that provide probabilistic membership in an intent-locale. Content wise, *"Do NOT turn off the lights"* and *"Do turn off the lights"* are nearly identical, except for three characters which could occur at any point in a sentence. Consider *"Do turn NOT off the lights"*. This garners a new meaning, set the lights to the negation of *"off"*, ie *"on"*. The third request, albeit contrived, has an entirely separate intent from the first two. It would be difficult to encode this difference in an embedding space.

Sentiment is typically classified as 'positive' or 'negative' by way of embeddings as well, as discussed by Tang et al. in the widely cited *Learning Sentiment-Specific Word Embedding for Twitter Sentiment Classification.* [Tan+14] While not as immediately apparent to the user of a VA (voice assistant) such as Siri, the limitation of this methodology has ramifications for more complex assistants in the future. As such, it would be appropriate to apply any solution to the negation problem to sentiment.

The proposed solution is to create a representation in which sentence structure and content are inherent, while maintaining the advantages offered through a metric space: a means by which to define a distance function.[7]

5.4 Semantic Web

The semantic web comprises of two directed graphs, G_0, G_1, which utilize the equivalent and thus equal sets of nodes but different edge sets; traces and edges. Let the graph constructed by the set of semantic edges, denoted as E_0 be G_0, and the graph constructed by the set of semantic traces, denoted as E_1, be G_1. Additionally, E_0 possesses a set of weights, W_0

$$G_0 = (V_0, E_0, W_0)$$

$$G_1 = (V_1, E_1)$$

$$V_0 \iff V_1$$

The semantic web, in less technical terms, is intended to form a non-hierarchical network of mixed types with a shifting topology, depending on which set of

[6] Edit: This concept was separately researched and publicized later; https://www.quantamagazine.org/ai-like-chatgpt-are-no-good-at-not-20230512/

[7] See 'Root Embedding Space'

edges is utilized. [4] It would be accurate to claim the semantic web is a form of knowledge graph.

5.4.1 Semantic Edges

The semantic edge tracks the current summation of the sentiment, the negation float denoting the presence of a prior negation, the type of edge, and a weight. Take a sentence S with words $w_0, w_1, ..., w_n \in S$, and a function P where $P(w_i) = [0, 1]$. The overall sentiment of the sentence is equal to

$$\Sigma_i^{|S|}(P(w_i))$$

, although by using a dependency parse tree, one can isolate sentiment on a local level within the sentence, identifying it as it specifically pertains to a subject or object - summing all leaf nodes attached to a particular node in the tree.

5.4.2 Semantic Traces

The semantic trace is an object that has two x and y values, one for each x, y pair of the nodes it connects. This is a secondary graph that does not account for chronological order, rather context. Traces connect common lemmas, entities, and profiles through the web. A given trace permits access to the most immediate previous use of the node, and if it exists, the use of the node which follows.

5.4.3 Semantic Vectors

The semantic vector, alone, is equivalent to a linguistic 'sentence level' semantic representation. That is to say, it exists independently and takes on whatever semantic content it possesses inherently, not one that may be lent by context or idiomatic use. Any given vector within the web, denoted as $S\vec{V}_x$, can be defined as follows: $S\vec{V}_x = (n_0, n_1, ..., n_i)$, where i is the sentence length, and n_q is the lemma found at index q in the sentence, stored in the form of a semantic node.

Semantic vectors could be of any length, although length is bound during frame creation. Beyond this representation, the semantic vector *object* (within the code base) preserves meta-data about the sentence itself for later retrieval. It records the time it was initialized, the speaker, the type of sentence, the track (a list consisting of node and edge objects), and the original plain-text of the sentence. In this capacity, it is serving as the long term storage of a frame in addition to the means by which a graph representation is created.

The semantic vector object is marked as complete by a "noid" (corruption of void and node), which is simply a node containing no text, signalling any current semantic edges be completed.

[4] Further reading on similar philosophical concepts may include Deleuze and Guattari's concept of the 'rhizome', introduced in *A Thousand Plateaus*

5.4.4 Semantic Nodes

The semantic node *object* is the representation of a single word on the graph. It tracks inbound and outbound edges, possesses x and y coordinates, the plaintext, tokens, and hash.

5.5 Profiles, Entity Identity

Profiles are one of several methods employed to give OSLO a better understanding of 'utterance level' meaning, specifically in terms of the resolution of individuals, be they speakers or non-speakers, to relevant information or context surrounding those individuals at any point in processing.

5.6 Temporal Context

Temporal context additionally assists in creating 'utterance level' meaning. Vernacular English uses time in a remarkably relative and abstract manner, making the resolution of specific times a challenging problem. This task merited its own dedicated module within the architecture, deemed *moment*, which takes as an input a text frame. Convertible ordinals tagged during entity recognition such as first, second, are immediately converted into numerical counterparts: "first" to "1st", etc. Other sections of the input frame are passed to the Python3 library *word2number* which uses a try-catch statement to attempt to parse it into numerical form.

5.7 Question Answering

Roberta-base-squad is used in conjunction with a pipeline provided by the Python3 library *transformers.*

5.8 Root Embedding Space

The root embedding space was designed as a means to provide the benefits of a sentence-level embedding space while retaining a greater deal of information in regards to sentence structure, as opposed to exclusively the content of the sentence. There are two methods being explored currently: relative and static. The popular and creatively named thesaurus website: *www.thesaurus.com* was scraped to construct a graph wherein each term is a vertex and each synonymy relationship is an edge. Antonymy was also scraped incidentally. From here on out the synonymy graph will be referred to as G.

Three 2-dimensional planes are created; $P_{subject}$, P_{object}, P_{root}

The geodesic distance between any two words $word_0$, $word_1$, where both is defined as $D(word_0, word_1)$. By measuring the distance between only the respective subject, root, and object of a sentence of two sentences, sentences of any length may be compared with one another so long as they are sufficiently formed.

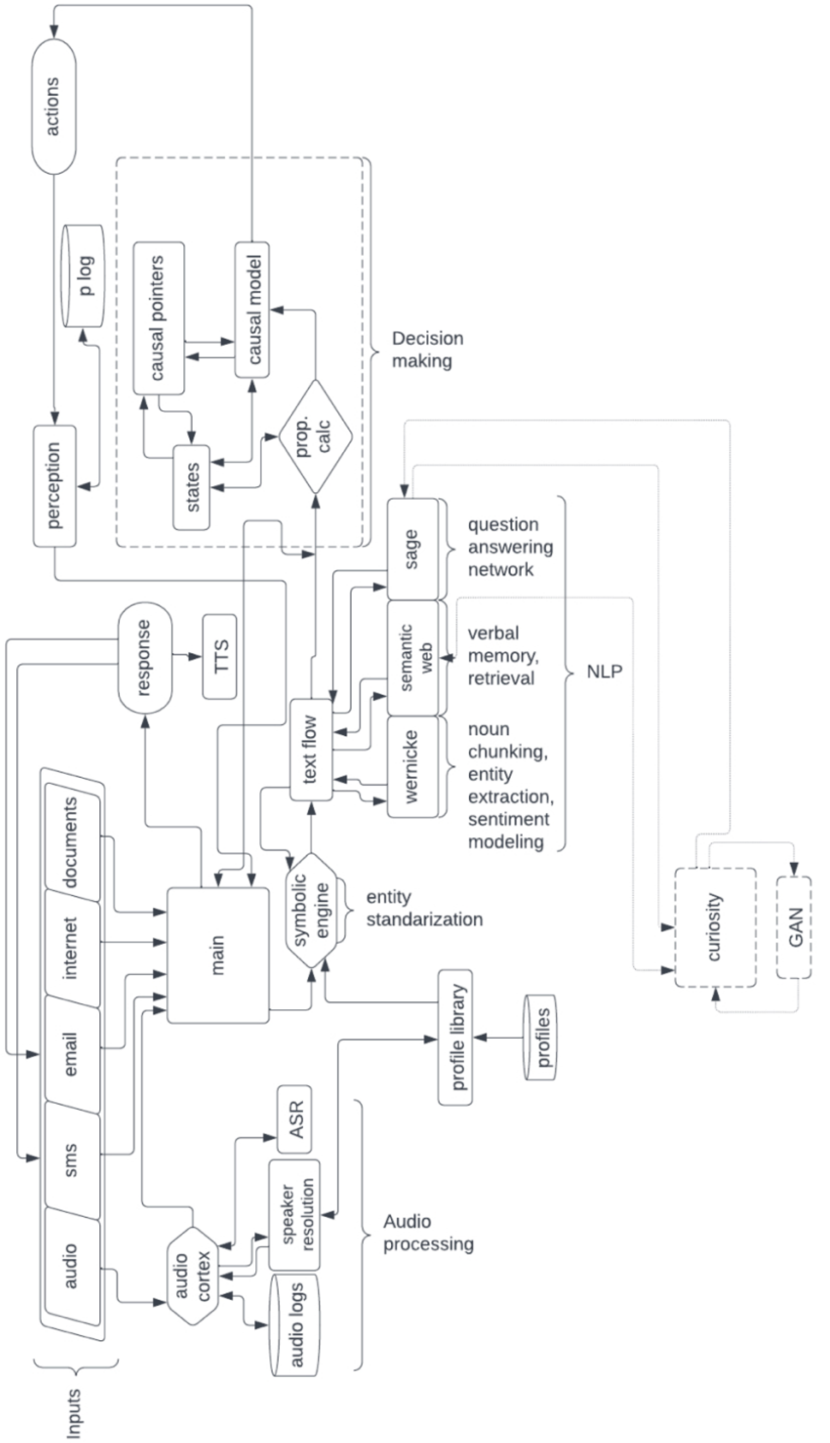
5.9 Propositional Calculus

For example, the state ‘bookMeeting’ may have two parameters, a and b. Parameter a is of type text, and it is the meeting description. Parameter b is of type time, and it is the time which the meeting occurs. Applying the state ‘bookMeeting’ is only applicable to a given sentence if the sentence frame contains these kinds of entities.

As some states have no parameters, and some states have identical parameters, this propositional calculus has low efficacy. [Lu+21]

5.10 Further Work

Further research includes applying machine translation to the task of converting natural language into the propositional calculus, allowing statements to be better analyzed. This is not intended as a comprehensive or absolute translation, the author understands that a bijection between the English language and a propositional calculus or formal system is impossible. However, an imperfect or inconsistent translation will still allow a degree of verifiable reasoning, ultimately exceeding the state-of-the-art LLM reasoning which is famously a ’black box’.


actions
p log
causal pointers
causal model
perception
states
prop. calc
Decision making
sage
question answering network
semantic web
verbal memory, retrieval
wernicke
noun chunking, entity extraction, sentiment modeling
NLP
text flow
response
TTS
documents
internet
email
sms
audio
main
symbolic engine
entity standarization
profile library
profiles
curiosity
GAN
ASR
speaker resolution
audio cortex
audio logs
Audio processing
Inputs

References

[97] In: *Mind Design II* (1997). DOI: 10.7551/mitpress/4626.003.0005.

[BM09] Catalin V. Buhusi and Warren H. Meck. "Relativity Theory and Time Perception: Single or Multiple Clocks?" In: *PLOS ONE* 4.7 (July 2009), pp. 1–6. DOI: 10.1371/journal.pone.0006268. URL: https://doi.org/10.1371/journal.pone.0006268.

[Hum78] David Hume. *A Treatise of Human Nature*. Ed. by L.A. Selby-Bigge. revised P.H. Nidditch. Oxford: Oxford University Press, 1978.

[Lu+21] Lu Lu et al. "Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators". In: *Nature Machine Intelligence* 3.3 (Mar. 2021), pp. 218–229. ISSN: 2522-5839. DOI: 10.1038/s42256-021-00302-5. URL: http://dx.doi.org/10.1038/s42256-021-00302-5.

[Moo+07] Chris Moore et al. "The Development of Body Self-Awareness". In: *Infancy* 11 (Mar. 2007), pp. 157–174. DOI: 10.1111/j.1532-7078.2007.tb00220.x.

[Tan+14] Duyu Tang et al. "Learning Sentiment-Specific Word Embedding for Twitter Sentiment Classification". In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Ed. by Kristina Toutanova and Hua Wu. Baltimore, Maryland: Association for Computational Linguistics, June 2014, pp. 1555–1565. DOI: 10.3115/v1/P14-1146. URL: https://aclanthology.org/P14-1146.