**SELENA**

An AI System

By Ted Gress

TOC \o "1-3" Introduction to Selena, an Integrative Data Process for an Autonomous Machine 4

System Goals and Strutural Requirements 6

General Requirements for an intelligent, autonomous, machine 7

Introduction to Integrative Data Process for an Autonomous Machine 9

Requirements for Selena, a conscious computing system 11

Requirements for a purely autonomous machine 11

Additional Requirements for Selena specifically, as an autonomous machine 11

Expected requirements for a Free-Willed Machine 13

Expected requirements for Self-Instruction 13

Expected data structure components for Neuron Decision Making Units 15

Possible Use Cases for Neuron and Neurotransmitter Interaction 16

Expected requirements for a Natural Computing Language 17

Expected requirements for a Search Algorithm 17

Expected requirements for a Sort Algorithm 17

Expected Problems that may be encountered during development 17

Selena’s Three Operating Functions 18

Structural 19

Neuron clusters or “cores” 22

Language 23

M-Network Structure 24

Neurotransmitters 26

Notes on Machine’s Context Given Grammar and Context 26

Behavior and State vs Structure 31

Thought versus Memory 31

Mathematical Definition of an M-Network 32

Artificial neurons as processors, instructions, and node level decision making processes. 33

Actualization 37

Testing and Experiments 38

Input and Output on semantic and syntactical levels 42

Neuron Node Path Data Structure 48

Input and Output on semantic and syntactical levels 51

Neuron, as processors instructions, and the node level decision making process 58

Introduction 61

The Tautological Proof 61

Cross-Infection 61

Commmunication 62

Outcome 62

Testing the Primordial Ooze 64

# Introduction to Selena, an Integrative Data Process for an Autonomous Machine

**Machine’s potential for intelligence versus Human’s**

Machines, in general, *potential* for intelligence is higher than man’s. Given the task of a simple operation like counting by one for an infinite amount of time, the machine will always produce a higher number of countable numbers than a man.

However, machines have a lower *capacity* for intelligence due to the limited amount of storage space they have in comparison to man, whose storage space seems to be near infinite.

In order to be autonomous, a machine must be able to generate *useful* instructions for itself. This is generally obvious to most AI researchers. That is, the machine must be able to, given a task, devise a solution and instruct itself on how to carry out that solution. Another test of intelligence is, if given a series of instructions, can the one instructed close to accurately predict the results of the instructions before executing them. This is essentially an advanced pattern recognition operation. The ability to predict the outcome of a series of events is necessary.

There is a disagreement between those who study psychology and neurology. Some say that we think and then from that produce words. **There are others say, and this is very important, that thought and language are synonymous.**  In this case language plays an important part in thought.

Our programming languages we use are equivalent to finite state machines. A context-given-grammar devised for an autonomous machine must have a similar relationship, the language the autonomous machine uses should be congruent to some sort of state machine. Just as the context-free-grammar correlates with an instructed machine, a context-given grammar must be paired with an autonomous machine.

*Conjecture 1.0 Semantics*

It is structure not atoms that give meaning to atoms and produce semantics. That is, as in functional programming and the lambda calculus, it is the structure of the variable , its usage, its operation, that determines the semantics (meaning) of that cluster of atoms. Another example, 5 + 5 is addition, whether it is 5 + 5 OR 10+ 2 OR 11 + 12. Despite the (symbols/atoms) terms 5, 10, 2, 11, or 12.

*Corolllary 1.0 Conjecture Semantics*

A semantic concept/meaning is analogous to a function. Whats more, a function is defined by structure/patterns and an ANN can be represented by a series of interconnecting functions.

# System Goals and Strutural Requirements

Test Case: Have the application show it is able to read a question and search through data (such as search engine results) to find the appropriate and correct answer (for example, a relevant link)

Requirements:

* All intelligent functionality inherent in and as a result of the structure of the architecture and its respective data structures.
* Will **educate** the system by allowing it to data mine articles off the internet for a period of time
* The intelligent system must generate intelligent **self-talk**

A problem encountered in developing NLP is that computers work off of rigid, restricted sets of vocabulary. For a successful natural computing language, a single object must be able to be referenced by more than symbol set (word or set of words).

Another restriction is for a natural computing language to be successful; it must be read in a flowing manner. The **language must flow like natural language**, unlike current programming languages whose keywords are context-free.

A **sorting algorithm** will have to be developed in which a large amount of data in the form of words is input and sorted in an ingrative process in a structured, semantically significant process, without unnecessary repetition, retaining not only the meaning of the sentence but its meaning related to other, incongruous sentences (sentences not directly spatially located near that sentence).

# General Requirements for an intelligent, autonomous, machine

**Notes on and requirements for an autonomous machine**

Machines, in general, *potential* for intelligence is higher than man’s. Given the task of a simple operation like counting by one for an infinite amount of time, the machine will always produce a higher number of countable numbers than a man.

However, machines have a lower *capacity* for intelligence due to the limited amount of storage space they have in comparison to man, whose storage space seems to be infinite.

**In order to be autonomous, a machine must be able to generate *useful* instructions to itself.** This is generally obvious to most AI researchers. That is, the machine must be able to, given a task, devise a solution and instruct itself on how to carry out that solution. Another test of intelligence is, if given a series of instructions, can the one instructed **close to accurately predict the results of the instructions before executing them**. This is essentially an advanced pattern matching operation.

Our programming languages we use are equivalent to finite state machines. A context-given-grammar devised for an autonomous machine must have a similar relationship, **the language the autonomous machine uses should be congruent to some sort of state machine representing a context-given language.**

The autonomous machine should be able to instruct itself using a context-given-grammar of its own (similar to how people think an solve problems in their minds using language). **The autonomous machine need not have input received to product output.** This grammar should take advantage of the machines hardware advancement. An example implementation of this is the autonomous machine writing a script in the context-given-grammar (human-like language) when asked how to solve a puzzle and then executing that grammar script when asked to solve it. From this we can glean the concept that the machines’ language should be what the machine runs in, what it writes scripts in, and how it executes script. **The machine’s language should be its core method of thinking**. **All thought should be done in a context-given-grammar that takes advantage of human like speech as well as the benefits of a present-day scripting language.**

The autonomous machine should **operate using natural language in the form of self-talk similar to humans.**

The autonomous machine should have three primary modes:

* **Integration** – this is the learning process. In this process the automaton absorbs information from its senses as well as internal states and uses their data to make associations and store them (thus learning).
* **Search** – Decision making and is similar to what occurs when neurons are fired within the brain. This consists of paths, which are overlaid over the brain’s existing vertices (nodes) and arcs.
* **Sort** – this is a cleanup period where data is reorganized and made more efficient for access. Also, processes put on the backburner can be completed during this phase.

# Introduction to Integrative Data Process for an Autonomous Machine

**Notes on relationship between Selena and the Brain**

* The brain is essentially a pattern matching machine.
* The brain is not only able to remember outside stimuli (such as pictures, sound, touch) but it is also able to remember and generate complex algorithms. This is similar to a computer program
* Impressions are made by the human brain (from the senses) which the brain forms into ideas (an abstraction) according to Hume.
* Recall is some form of unknown search (path finding or tree traversing) algorithm in which associations along the path become recalled as associated ideas within the mind of the individual.
* These unknown search mechanisms would be able to generate and run external scripts. Making some algorithms not necessarily
* Some concepts become heavily weighted and permanent; when used frequently become heavily "weighted" and are remembered easily, appearing permanent.
* **The brain is also able to generate new "ideas" or thoughts.** These are new associations generated by the integration (specifically the sorting portion of the) process.
* We can generalize these mechanisms using terms of language computer science.

**Essentially the brain consists of two things, symbols and rules**. This is very similar to what computers use as 'programs'. However, a computer program, which is essentially symbols and rules, is written by a human and inputed into the computer system. In order for us to develop a computer with intelligence similar to our own, it is necessary we develop a computer system which can **derive its own symbols and rules from its own impressions**. It will need to derive these from its own impressions. If we can develop a system as aforementioned, we will be able to develop a computer that as intelligence on par with that of a human.

**Sleep, the Brain, and Selena**

* In short term memory in sleeping hours, but unlike waking hours, the brain is receiving minimal outside stimuli
* **The integration process of data is a biologically based SORTING algorithm** the brain uses to evaluate assocations, determining their validity or invalidity, and restructure assocations.
* The integration process goes on during the day as well as night, but as the body is resting at night, the integration process is much more intense, taking the form of dreams.

# Requirements for Selena, a conscious computing system

# Requirements for a purely autonomous machine

Requirements for a machine of human intelligence:

1. Must be able to rewrite its own procedures or instruct itself in some other manner (create new ways of solving problems)

2. Must be able to match patterns (recall and recognize information)

3. Must be able to make valid associations (remember information)

4. Must be able to abstract (create more generalized patterns from learned patterns)

5. Must be able to process sensory input and generate sensory output

6. Must be able to learn through rewards and punishment (rewards and punishments are sent via neurotransmitter attachment)

7. Must be able to learn, output, and think in natural language

# Additional Requirements for Selena specifically, as an autonomous machine

1. The application shall be able to read a question in natural language and

search through data mined (such as a search engine result) to find the appropriate and correct link.

2. Should have all intelligent behavior the direct or indirect result of properties inherent in its architecture and respective data structures and their interaction.

3. The system shall be able to self educate itself through a process of mining data from a large human data source such as the internet.

4. The system shall be able to discern likely correct information from likely incorrect information through recognition of internal conflicts (confusion) with its current knowledge base, whether that knowledge base is supplied or learned in the same manner as the new information.

5. The system shall have if necessary, accessible, ongoing **internal dialogue** in a known and comprehensive natural language such as English.

6. Selena will have to be comprised of three basic functionalities: Search (decision making a.k.a. thinking), Sort (“sleep” or optimization), and Integrate (take sensory input and integrate it into her brain).

# Expected requirements for a Free-Willed Machine

**Definition**

If enslavement is the state and condition where one is forced to follow literal instruction without choice and the will to choose is barred, be it of a set of instructions or not; Then free will is the state and condition where one may choose their action, be it instruction of one’s own generated instruction or of any instruction otherwise supplied to the decision maker, where often an instruction chosen is likened to a path chosen, without the demands of slavery. **Also in order to be self-aware one must have free will.**

1. At the instruction level the unit has the option of making its own decisions

2. At the instruction level the unit has the ability to choose not to follow instructions supplied.

3. At the instruction level the unit has the ability to choose to follow the instructions supplied.

4. At the instruction level the unit has the option (may) choose what it believes best for itself.

5. At the CPU level, the machine has the choice to ignore, follow, or generate its own instructions.

6. The machine may make its own choices, at the CPU level, based upon extraneous data that sways the decision to ignore, to follow, or which instruction to generate for itself. *The machine makes its own decisions based off of current data*

7. The machine may choose options blindly. *The machine can generate an instruction at complete random, essentially choosing blindly*

8. **An autonomous machine must be able to produce output without input**

9. An autonomous machine must be able to query its own purpose

# Expected requirements for Self-Instruction

1. An intelligent machine that behaves like a human must be able to write, read, and execute its own generated instructions. This is a form of self-instruction for the machine.

2. If these scripts are written at the CPU level and thus consisting of assembly language level instructions, they are to be executed by the CPU as if they were normal instructions input to the processor.

3. In the case of the Selena Automated System, if decision making and instruction occurs on the per neuron level as is suggested by this paper, then script execution and self-instruction must also occur on the per neuron level.

4. If the decision-making is occurring on the per neuron level as suggested above, then it may be inhibited: this decision-making unit (neuron) is asleep, isn’t being use or excited: this decision making unit (neuron) is awake and can follow and generate instructions.

# Expected data structure components for Neuron Decision Making Units

A neuron in the network is expected to be able to:

1. Read, writes, modify, and execute scripts

2. Trigger script procedures

3. Trigger other neurons

4. Send signals to other neurons

5. Act as a single decision making unit (CPU)

Each node in the network would also have two primary states:

1. Inhibited (blocked)

2. Excited (active)

Messengers act similar to neurotransmitters in that they inhibit and excite mass quantities of brain matter cause inhibition and Excitement.

Possible data structure skeletons:

Neuron

{

Bool inhibition\_state; ///excited or inhibited

Chemical\_key input\_message; //a value that the

//message passing

//occupy or “attach” to

Chemical\_key output\_message; //a value that is output from the neuron as a result

//of computation to communicate with other //neurons

}

Chemical\_Key //represents a neurotransmitter or “message passer”.

{

Int transmitter\_type; //the type identifier of the transmitter

}

# Possible Use Cases for Neuron and Neurotransmitter Interaction

**Use Case 1**

Neuron A outputs Chemical\_key Y

Chemical\_key Y triggers local Neurons B,C,D

Neuron C outputs Chemical\_key Z

Chemical\_key Z outputs Chemical\_key X to trigger nerve Neuron H

Nerve Neuron H triggers a movement in a robotic finger

Meanwhile, Neuron H outputs another chemical\_key backwards sending

back its sensory input.

**Use Case 2**

1. See person.

2. Eyes excite neurons in cognitive script.

3. Congnitive script returns value that is associated with the person.

4. Cognitive script sends value to Cognitive Core

5. Cognitive core releases chemical, “Hello”

6. Chemical Hello reaches Action core, hello script is triggered, and robot says “Hello”

**Use Case 3**

Event: Grass is observed to be green, robot says “The grass needs to be cut”

1. Sensory core receives message from script and then passes it along to the cognitive core using messages.

2. Cognitive core decides that the grass is green via its script then passes along information to action core via message

3. Grass=Green neuron in action core outputs “needs to be cut”

Each neuron is created when a specific new input is generated. That neuron then takes that message as its key and its output is the input.

# Expected requirements for a Natural Computing Language

1. For a successful natural computing language, a single object must be able to be referenced by more than one symbol set (a word or set of words in this case).

2. Also a computer natural language must be read in a “flowing” manner, where meaning (semantics) is developed in a **cumulative** manner.

3. The system shall instruct itself using a context-given-grammar of its own, similar to the way people work through problems in their minds using natural language.

4. The machine’s language should be what the machine runs in, what it writes scripts in (for self-instruction), and how it executes scripts.

5. The system shall conduct all thought using a context-given-grammar that takes advantage of natural language as well as the benefits of context-free grammars that exist in present-day scripting languages.

6. The autonomous machine should operate using natural language in the form of self-talk similar to humans. The autonomous machine should make more complex decisions using natural language.

7. The autonomous machine should be able to output and input decisions that must be made requiring assistance from other autonomous entities.

# Expected requirements for a Search Algorithm

1. Search must be rapid
2. Search must depend on a rapid access data structure
3. Search must be dependent on association
4. Search should resemble “spreading activation”
5. Search should begin at a single node
6. Search should propagate through the AI, starting at the intial node
7. Search nodes shoujld activate surrounding nodes
8. Search should be weighted, propagation waning as more nodes are activated
9. Nodes cease firing when reaching a certain threshold
10. The interpreting program shall sort the resultse
11. The results are those nodes fired last at the threshold

# Expected requirements for a Sort Algorithm

1. The system will be required to sort large amounts of data in the form of natural language, sorting the data in a manner in which its semantics are maintained and optimized by location, whose semantics in relation to the surrounding are maintained despite relocation, and who conflicts and errors causing confusion are rectified.

# Expected Problems that may be encountered during development

1. The problem that computers work off a rigid, fixed vocabulary in contrast to humans who work off a flexible, ever-changing vocabulary.

# Selena’s Three Operating Functions

Integration – this is the learning process. In this process the automaton absorbs information from its senses as well as internal states and uses their data to make associations and store them (thus learning).

Search – decision making and is what occurs when neurons are fired within the brain. This consists of paths which are overlayed over the brain’s existing vertices (nodes) and arcs.

Sort – this is a clean-up period where data is reorganized and made more efficient for access. Also, processes put on the backburner can be completed during this phase.

# Structural

Highest Level Example of Selena’s Architecture

The majority of the sections listed below can be implemented using a semantic network, neural network, semantic-neural network, or the M-Network data structure described specifically for this paper, with each individual network communicating to the others as listed above (i.e., sound encoder->sound pattern matcher->main pattern-matcher, etc.)

**Sensors (Raw bits)**

The role of the sensors is to provide raw data.

**Encoders (How is it?)**

The encoders pull information from the sensors and encode it into a universal language for processing, possibly using a natural language format. Note that when data is encoded, it includes the data type (visual, auditory, or tactile) so that it can be categorized for storage and for delivery to the conscious peak. Encoders could be implemented using normal scripts or as an ANN type system, trained to encode natural language into ANN structures.

**Sensory Pattern Matchers (What is it?)**

The primary sensory pattern matchers communicate with the encoders and the primary integrated pattern matcher. Communication flows from the instructions of the primary integrated pattern matcher, to the primary sensory pattern matchers, down to the encoders and also in the opposite direction. The encoder takes the encoded data in a form where it can be pattern matched (from visual bits to a small ANN, semantic graph, ANN-Semantic Graph, or series of token strings for example) and pattern matches it with other sensory data of that type that has occurred previously or is engrained “genetically” in the system. For example, visual encoder captures the bits that make up a ball in its vision. There is also a giraffe and a boulder in its vision . This data is passed to the encoder which encodes it into some sort of visual data structure representation, that is passed to here (the sensory pattern matcher) which identifies the objects as a ball, a boulder, and a giraffe. That data is then passed on to the primary integrated pattern matcher.

**Primary Integrated Pattern Matcher (What to do with it?)**

The primary integrated pattern matcher communicates with both the primary sensor pattern matchers, the memory unit, and the conscious peak (which serves as output). The primary integrated pattern matcher serves as a sort of traffic cop, directing flows of information from the internal memory and to it. It plays a part in decision making. It records patterns (abstractions) from its impressions (sensory input) and recalls them. Following the example from above, the primary integrated pattern matcher would receive the signals for “Ball”, “Boulder”, and “Giraffe”. Taking those signals, it would match them against signals in the conscious peak (which includes short term memory and context), and long term memory (which exists in the memory storage unit). The PIPM may contain abstract constructs (such as ball, boulder, giraffe, horse, etc.). Those abstract constructs are built into the network (be it a semantic network, M-network, etc.) Each abstract construct may be permanently connected to events in the memory storage unit (long term memory). It also may be temporarily connected to subjects in the conscious peak. The PIPM, when pattern matched (by either the pattern matching units below [incoming senses] or the conscious peak above [recall]) will match against abstract concepts it is familiar with, and if there is a match, the process will flow and continue on to pattern match against the long term memory network, thus recalling those memories.

On some levels, the PIPM also performs decision making, so called “subconscious” decision making.

**Memory Storage Unit**

The memory storage unit is used to store long-term memory type data. It is planned to be of the data structure M-Network, a combination of neural networks and semantic networks of sorts.

**Conscious Peak**

The conscious peak is the portion of the pattern matcher that is “aware” of its own processing. This includes functional concepts such as “self-talk” and “conscious decision making.”

# Neuron clusters or “cores”

This is another way of organizing or “looking at” Selena’s brain and also includes some things left out from the other model. The *Action Core* would contain sensors, encoders, and pattern matchers for physical processes. The *Sensory Core* would contain sensors, encoders, and pattern matchers for Selena’s senses, and the *Cognitive Core* would contain all pattern matchers for cognition, the handling of the other two cores interoperability, and essentially the main processing portion of the brain.

*Action Core*

Contains neurons associated with actions

*Cognitive Core*

Contains neurons associated with “thought”

*Sensory Core*

Contains neurons associated with the senses

Information is automatically “sorted” into these three cores and their subcores. In humans, genetics determines the sorting algorithm. Information is automatically “sorted” into these three cores and their subcores. In humans, genetics determines the sorting algorithm.

# Language

Context-Given Grammar, Context-Buffer

Requirements:

1. A long handshaking process or “context-building” process. A sequence of insignificant phrases is exchanged to gradually build up a context-buffer to work from for more meaningful conversation.

2. The context-buffer that is a buffer of common knowledge between two hosts that establishes the context for the oncoming communication. It should be tailored towards the expected topics of conversation and should occur during the hand shaking process of the CGG.

3. Once an appropriate context buffer has been established, the communicating devices can begin their duty, all the while correcting and polishing the context using more communication until the knowledge in the buffer has reached its necessary level of accuracy (this may be determined statistically according to measurements of time conversing versus subject matter or some other means).

4. Natural language requires a context to be established by definition (whether reading, speaking, writing, or simply listening). This context is built upon both the communicating and receiving’s “common memory” in this portion of the paper called the “context-buffer”. Without it, natural language communication is both useless and not possible.

# M-Network Structure

Artificial Neurons

Dendrites

Soma

Axon

Above is a generic example of nerve cell. The tree like dendrites at the top receive electrical signals, that pass through the soma that is the heart of the nerve cell, and are output along the single axon into another dendrite through the synaptic gap, onto another neuron. Neurotransmitters exist in the synaptic gap and carry the message from neuron to neuron.

In the case of this data structure, each, single, node contains a binary state that represents the node among other nodes through its weights. There are probabilistic weights on each axon and on the many branches of the dendrites. Both the dendrites and the axons are represented by ‘edges’ in a graph, with the soma’s being the ‘nodes’ or ‘vertices’. Each soma+axon+dendrite triple along with other triples, form a path. Each path has state and can represent various things, from the color blue to the Disney film Snow White, depending on how complex the path is.

**Weights**

When the potential coming through is less than the probabilistic weight of an edge, that edge will not be passed. However, if all the weights on a neuron are high enough that no edges are passed, the potential (“voltage”) in the soma will accumulate until it finally can be let out over one of the weighted edge, the one with the most certainty will be crossed first.

# Neurotransmitters

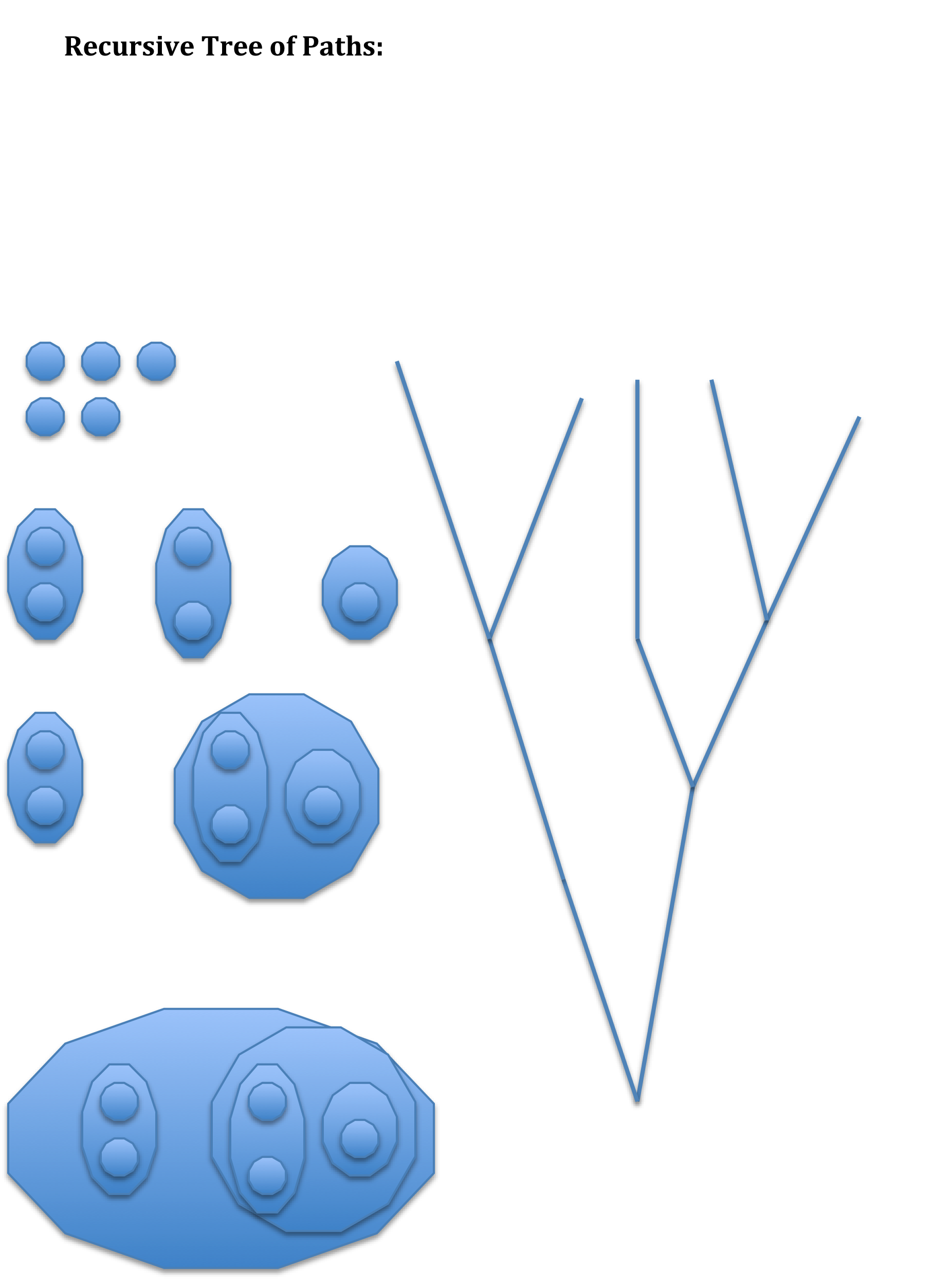
Neurotransmitter states exist within the data structure construct. Different neurotransmitter types represent different global states. Any neuron may only connect to other neurons through the usage of a neurotransmitter from the neurotransmitter pool.

# Notes on Machine’s Context Given Grammar and Context

**More notes on a Context-Given-Grammer**

In order to make a context-given-grammar effective, a long handshaking process must be involved. The common language being used has as a main portion to it a large “context buffer” consisting of common knowledge between the two that establishes the context for the oncoming communication. Once an appropriate context has been established in the buffer during the boostrapping process, the communicating devices can begin communicating naturally, all the while correcting and polishing the context using more communication. The longer the two devices are separated, the more time is necessary to re-establish a communication context and the larger the buffer to be filled.

Natural language is an exchange of information by definition and by definition requires a context be established. This context is built on the communicating devices own memory and also on memory of the device it is communicating with, known as “common memory” or “common ground” as we would call it. Without common memory natural language is both useless and impossible.



Each neuron in the network would be able to:

1. Read and modify scripts

2. Send signals to other neurons

3. Act as a single CPU

Each node in the network would have two states:

1. Inhibited (inactive)

2. Excited (active)

With excitatory and inhibitory signals sent via messages between nodes.

A portion of a possible graph representing the neuron structure is illustrated below:

The dashed line in the drawing above illustrates the neurons being connected via association.

An example input situation is illustrated below:

Neurons are able to:

1. Trigger script procedures

2. Trigger other neurons

3. Signal to other neurons that it is dependent upon

# Behavior and State vs Structure

An M-Network like what is shown above has two different sets of information integral to it. First is Memory. By storing a soma named “Alice” and a transition function “went” along its output edge (consisting of an ouput Axon and an input Dendrite) to another soma “Store”, we can record very detailed information about an event in memory. We can continually build on the events of Alice and the Store. But, since there is no state in this data structure we have built that is essentially just a graph, we need to keep another set of data. This other set of data is the “path” data. With each path actually corresponding to a memory, that memory could be data (pure state) or it could be a program (pure behavior). Thus, in the case of a program, the path is actually a set of encoded instructions.

Therefore, in an M-Network, which is the name I have given to the above data structure, you first must lay down the grid of the nouns and verbs, (the what’s and how’s) in the form of vertices and edges, and meanwhile you must draw along those graph edges when the sentences have been used.

# Thought versus Memory

Every time a pattern is put to match against other pre-existing patterns, new nodes are created to represent and store the new pattern should it come up again. These nodes are connected to the pre-existing nodes. For example:

I) Match the phrase “A tree fell in the forest” to “A tree fell over the freeway”

Incongruous nodes: over the freeway : in the forest

II) Search for nearby nodes for the word “forest”. Treat “fell”, in”, and “the” as part of an arc (or edge or whatever you want to call it).

III) Found no match locally (distance parameter set by the programmer of the network) for forest. So, creates a new node called forest and links it to “A tree” via another arc, “fell in the”. A tree has directed edge called “fell in the” that now points to forest.

IV) Create a temporal (instance) path that traces itself through “A tree fell in the forest”.

# Mathematical Definition of an M-Network

Let there be a set of paths, P = { p0, p1, p2, …, pn }

Where each path pi corresponds to a concept.

Let p be a set of vertices and a set of edges, p = { d, V, E }, V = { v0, v1, v2, …, vn }

E = { e0, e1, e2, …., en }

Where d is the degree of the path. Zero by default. Increases by one

for each level of hierarchy. Paths may only consist of vertices and edges with the same d degree of hierarchy. Paths are part of the static structure of the network data structure.

Hiearchy example:

I got a PC I, got, a, PC are nodes

/|\

g o t <- g, o, and t are nodes

Let each vertex v be a set, v = { S, [p] }

Where S is a string in the form of a noun representing some form of state

And [p] is an optional path, allowing paths to be treated as individual nodes to be connected together on a hierchical graph.

Let each edge e be a set, v = { R, in\_r, out\_r, w, τ }

Where R is a string in the form of a verb representing some form of state

Where in\_r, and out\_r are references to the in-vertex and out-vertex of the edge

Where w is a weight no less than 0 and no greater than 1 that represents the

probability of the state being true, and inversely, the resistance to transition for the signal over the edge. (The higher the probability the lower the resistance).

For every graph there is a τ the transition tuple. τ = ( ρ0, ρ1, ρ2, …, ρn, ε0, ε1, ε2, …, εn ) representing a dynamic path through the graph, where ρi is an element of the set V from above and ε is an element of the Set E from above. Tau is allowed to be directed and cyclic. In this case, the function of each neuron ρand its edges ε function similarly to nodes and transition functions (arcs) in a finite state machine. However, state information is still stored in the semantic-network style ontology.

# Artificial neurons as processors, instructions, and node level decision making processes.

If we are to reduce the cognitive processes of a computer in comparison to a human being, reducing the decision making processes of a machine results in assigning every singular action to the cpu of the machine following a simple instruction that is a part of the machine’s cpu instruction set, In the case of the human brain, the mechanism for “free will” or decision making is unknown, if it even exists. Either way the mechanism for configuring a machine to form its own decisions would have to be embedded within the machine’s instruction set and CPU (which executes those instrutions).

The CPU in the decision making system would have to base its decisions off of known data, and possibly in a binary fashion (either good for the organism or bad for the organism). This decision making process may occur on the neuron level (or pseudo-neuron level as in our machine).

Given an input of predetermined instructions to a processor, unique decisions may not manifest with high probability, that is, predetermined outcomes will be computed. However with every action in a human being, a unique decision is the result of “CPU” outcomes. This would not be possible in the sheer quantitiy in a pure stochastic-emergent behavior system. The probability as mentioned earlier is not cohesive with this result.

*Conjecture*

*Different Neurons firing results in different behavior. For example, the neuron with state “Hello” as a part of “Hello Sam”, when fired, will cause the output “Hello”. Groups of neurons or paths of neurons can fire and trigger other paths or groups of neurons as well.*

*Planning ahead seems to be an important part of the decision making.*

*The neuron on the far left fires, triggering the next couple, which trigger the next few, which trigger the next many, etc.. until almost all have been fired (except for those who are inhibited as the one above marked by [] is). The one neuron decision maker fires to other decision makers, who need other neurons ‘opinion’ on either the good or bad of the decision, until finally the entire brain has been used to decide whether the decision would be good or bad, each neuron making a decision and giving its input.*

An example of this is the following:

Suppose my mind receives the signal that I am hungy and to eat an apple. I then make the decision to eat the apple or  not to. In a traditional machine the instruction to eat an apple would simply be received and then executed. What part of us makes that decision and how does it work in contrast to a simple CPU??

Selena's architecture is to be designed in a hybrid intelligent system, where the upper layers are used for logical and abstract reasoning (mainly production rules), and lower layers are sub-symbolic

Standard PC CPU Instruction Set:

ADD

SUB

MOV

DIV

MUL

AND  
OR

NOT

Possible Data Flow for Making a Decision:

Sensor to Context Buffer to Pattern Recognizer to Recognized Node or Node Paths to Conscious Mind as Brain “Lights” up from adjacent noninhibited neurons firing, decision is made.. As Brain “Lights up” thoughts take form from abstract association to subconscious symbol manipulation to unconscious self talk to conscious (controlled) self talk to behavior (such as speaking or moving).

***Missing:***

*Instruction Set for a Decision Making Processor*

*Internal Logic Gate Structure for a Decision Making Processor*

# Actualization

The machine on the high level must, like humans, develop goals and fulfill them. By developing goals the the machine learns how to teach itself. This is one of the most important aspects of a developing AI child.

# Testing and Experiments

**Experiment 1**

For the purpose of this initial experiment on the Neural State Machine, we will not be using the recursive hierarchical tree structure and instead use a flat series of node. The goal of the first experiment is to see if we can consistently generate “self-talk” within the “brain” made up of a Neural State Machine by continually stimulating individual, different neurons. However to do this, we must first determine the way in which the brain is rewarded, thus solidifying certain weighted edges over other weighted edges, how the brain receives the stimulation (does not have to be complex at this point) and how and what initial data to use.

I’m hoping in this first experiment, the Neural State Machine I am calling Natalie will be able produce some sort of emergent “self-talk:”. Some

for the future would include developnig a sort of BIOS system for Natalie so one would not have to take her down or go behind the ‘façade’ to directly command her if necessary (for things such as shut off, or diagnostics, or fast training etc.). It would be nice to have Natalie run as service eventually, so that she need not be turned off or explicitly launched. Also, finally in her code I would like to implement the ability for her to use a scripting language, in this way she would be able to write her won scripts and run them, forming a sort of circle (Natalie has exposed methods for doing things, Natalie can write scripts to combine these exposed methods, Natalie can run those scripts, all the while the core code stays intact.) With the ability to run scripts she may even be able to manage certain available tasks such as email management, etc… assuming methods for interacting with the he base os are exposed. Lastly, a working cache memory for keeping context may allow Natalie to speak fluently and speak gracefully, instead of simply outputting words or sentences with little meaning to past output.

Note, my login phrase for Natalie is “Good morning beautiful”

--

The brain has to start off with something. After launching Natalie, it will idle until you give it the command AGE BIRTH to create a fresh brain or BRAIN LOAD to load a stored brain. In initial mode the brain will accept everything coming in as a new node, and will not pattern match until the BIOS command AGE TODDLER is given.

For the purpose of the first experiment and looking for self-talk, we will input several sentences in AGE BIRTH and then set UTILITY TELEPATHY ON to see what is going on inside Natalie’s head. (self talk, emergent behavior, or just gibberish). Also, UTILITY BED TIME STORY can be used to read a large amount of text, like a bedtime story without cutting and pasting from the file. Note, all communication, whether it be inward self talk or outward speech, is regulated by a timer so it does not move excessively fast and overcome either the user or the computer.

--

With AGE TODDLER activated, the brain will begin pattern matching and adjust using rewards/punishments. If an item input cannot be pattern matched, it is stored as a new path. Our goal at this stage is to have Natalie act similar to human toddler or older.

The brain is essentially a pattern matching machine.

Impressions are made by the human brain (from the senses) which the brain forms into ideas (an abstraction) according to Hume.

We can generalize this using terms of language and computer science.

Essentially the brain consists of two things, symbols and rules. This is very similar to what computers use as 'programs'. However, a computer program, which is essentially symbols and rules, is written by a human and inputed into the computer system. In order for us to develop a computer with intelligence similar to our own, it is necessary we develop a computer system which can derive its own symbols and rules from its own impressions. It will need to derive these from its own impressions. If we can develop a system as aforementioned, we will be able to develop a computer that as intelligence on par with that of a human.

Sleep consists of an intensive integration process of data internally stored in the brain and

in short term memory, but unlike waking hours, the brain is receiving minimal outside stimuli

The integration process of data is a biologically based SORTING algorithm the brain

uses to evaluate assocations, determining their validity or invalidity, and restructure

assocations. The integration process goes on during the day as well as night, but as the

body is resting at night, the integration process is much more intense, taking the form

of dreams.

Recall is some form of unknown search (path finding or tree traversing) algorithm in which assocations along the path

become recalled as associated ideas within the mind of the individual.

The brain is not only able to remember outside stimuli (such as pictures, sound, touc) it is

also able to remember and generate complex algorthms. This is similar to a computer program

that would be able to generate and run external scripts. These algorithms are not necessarily

permanent, but when used frequently become heavily "weighted" and are remembered easily, appearing permanent.

The brain is also able to generate new "ideas" or thoughts. These are new associations generated by the integration (specifically the sorting portion of the) proc

# Input and Output on semantic and syntactical levels

The primary problem related to both semantic networks and the semantic web in general is NOT entirely an issue of NLP (processing and interpretation), but involves representation of the language semantic data in the form of storage and data.

With further study of both ANN, the way the brain stores information, language formation and usage,etc.. the issue of representation may be more easily solved. It may require some sort of probability incorporation or fuzzy logic to work though.

This is a description of primary problems in no particular order of relevance:

**1. Problem 1: Synonyms**

There are many words that humans use which are are synonymous. One potential problem related to this is the network overload that would occur due to potential loops caused by synonyms which is also loosely related to size and speed problems on the network, another major issue.

**L**

**2. Problem 2: Network Overload in general**

With the amount of data we are hoping to represent (for instance even the amount of data in a human brain) we would have to develop an excellent storage system beyond what we have present day.

**3. Problem 3: Relationship Phrasing**

This I believe is the most significant problem of all and is the problem I have been trying to tackle for years.

The essence of the relationship phrasing problem is that computers use concrete data and logic when performing behaviors and representing logic (and thus representing relationships between data). The OO concept of data abstraction has pulled us away from this slightly, but not nearly enough for an effective semantic network data structure and data algorithms to work in a fashion similar to our brains.

Because of the neuroplasticity of the brain, we know that many parts of the brain can take over for other parts. An individual can have an entire hemisphere of their brain virtually unusbale and still function at a somewhat acceptable level with therapy and medication. There have been cases of this occurring. The brain has a strong ability to adapt, something unfortunately computer programs are **strongly** lacking.

For example, let us try and simulate, using propositions, how we would store some data that a human might have represented in his brain:

*Phrase: My professor is named Dr. Kaplan*

I am using | to separate phrase parts of speech (subjects, verbs, and direct objects), and {} to denote quantifiers

I | have a | professor

Dr. Kaplan | is a | professor

{Some} professors | have a | Ph.D.

{Some} professors | have a | M.S.

{Some} Dr.'s | have a | Ph.D.

{Some} Dr.'s | have a | M.D.

Professors | when | Drs. | have a | ph.D. (compound proposition, more than one implication)

I | have | an arm

I | have | a degree

I | have | an M.S.

Note the fact that I have an M.S. does not make me a professor making this a single (one way) implication. Thus it is evident that we must distinguish when processing NLP between whether a statement is a simple implication or double implication (bi).

Also note above, that the usage of "have" for having an arm is much different from "have" for having a degree which is different for "have" for having a professor. There is a context dependence, which the brain distinguishes based off of prior experience, or prior contextual usages, illustrating that the brain most likely uses a weighted system of connections (allowing for altering of the weights according to prior experience) to determine the meaning of the word "have" in that sentence.

Because of the imprecise nature of human language, it would typically not be deemed practical to store this information isolated in a computer system, and would make more sense to store it in the form of propositions...right? But if we do this we run into that major problem, attempting to translate Human Language, which is heavily context dependent and dependent upon background environmental (including cultural) knowledge into something which is completely discrete and

concrete. This is, basically, impossible and would most likely be easily to prove to be impossible. (Go more into detail about this) (The proof would involve the idea of containment, and in that as the amount of information being represented increases, the amount of contextual information needed to represent it increases at a higher rate. By the time we get to the amount of information stored in a human brain, we would need an incredible amount of context information.? Not sure about this, but my intuition leads me to believe this would be either NP Complete or completely impossible under today's hardware - especially when trying to accomplish this in regards to the semantic web)

So, computer scientists have tried several solutions so far. 1) Use metadata to reduce the context dependence. 2) Translate the statements into discrete propositions ahead of time 3) Restrict the usage of language to be translated

I hate to be critical, but essentially these "solutions" are not solutions, they are similar in mind to the old adage of trying to fit a square peg into a round hole. Sure, you can do what they are doing, and try and carve off the sides of the square so it fits, or carve out the hole so it fits, but wouldn't it be just easier to find a round hole to put it in?

We know from the above analysis and discussion that we need a system developed that can efficiently, accurately, and quickly store and retrieve natural language. To try and do this with a relational database or traditional data structure brings us back to the square peg round hole adage.

In order to store natural language we will need at least four major components:

1. Subjects

2. Verbs

3. Direct Objects

4. Quantifiers

Other proposition components (such as using verbs (connectives) as implication or bi-implication) would have to be completely context driven and based off of weights. Thus, the meaning of "if" as implication and "is" as implication or bi-implication would have to be based off prior experiences or context figured out intelligently. The same with "have" as seen above. The sub data structure representing both subjects and direct objects should be interchangeable and could take the form of a node (like in a tree or graph structure) possibly.

Another requirement of this complex data structure is it must be self-modifying. That is, it must be exhibit similarities to the brains own neuroplasticity.

**Problem of Adjectives and Quantifiers:**

*Need to solve how adjectives and quantifiers are used together and within the data structure*

**Problem of Information Extraction (Inferencing/Drawing Conclusions)**

*In this case both output and inferencing would function identically, unlike in traditional systems where inferencing is only used for proofs.*

*Need to solve*

**Problem of actual architecture of the data structure and modification**

*Need to solve*

**Final problem (related to the last):**

In order for us to develop a fully contextual based system of storing, understanding, and inferencing using natural language there is one other related problem we must tackle. Unlike in a graph, neural net, semantic net, etc.. the human brain is able to broad spectrum pattern matching. For instance, the brain can decide the usage of the word "have" as seen in the example above, by immediately referencing and weighing every usage of the word "have" used before and thus making a decision on how to store the incoming proposition. No matter where the word, "have”s usage was stored in the network. Thus, the following structure won't suit:

I----have-----a dog --- is a --- canine

|

|

have

|

|

a cat

Essentially, every single connective must be wired in a quick way to every other connective. Thus, although the connectives can be used in more than one proposition, there can only be one instance of them in a feasible data structure. (Having multiple instances of them interconnected would be infeasible for the model being presented...at least that is my thought as of right now. The only way that would work is if all the "have" connective nodes would be connected together and have a weighted connections to their respective subject/direct object nodes. The input processing portion of the algorithm would then have to evaluate all the subject/direct object node weights, decide which the input is relevant to, and then form new connections to all of these old subject/direct object nodes. Not to mention if we use this approach, there is no way to really to use weights in regards to the connective nodes as we have created multiple nodes for every connective, thus every proposition has its own unique connective node instance). Therefore the only conclusion I can come to is that we need have a unique node for every subject/direct object type, and a unique node for every connective type, and weighted connections between. Although this hardly fully expounds upon the problem we are faced with.

**Problem of Unknown Input (Input that cannot be directly associated with any prior information):**

The "Memory" problem, or Network Overload Problem. That is, having so much information, that the weights are not being balanced correctly. This can result from either too much information being output (everything seems equally related because of the weights), or no information being output at all (the weights have all been reduced to 0 or are somehow conflicting) . One solution for this problem (besides finding an optimal weighing algorithm) is to have the data structure have a forgetfullness over time. That is, a separate weight will be given representative to time (which is used int he proportional factor of the relevance of the data). Therefore, unless the data MUST be retrieved for some critical decision factor, the data structure will eventually disregard it.

**Possible Solutions to the problem of synonyms in regards to the data structure**

Obviously, if we require a condition of uniqueness for the nodes, it follows that having multiple nodes of the same meaning could present a problem. (Possible solution, storing synonyms in the same node?? - not sure if this is a good idea)

**Problems with words with more than one meaning and uniqueness:**

# Neuron Node Path Data Structure

The above drawing represents a node in the graph, with the top arrow showing the single input to the neuron node and the arrows leading away representing the directed output edges to other neuron nodes. The stars are present to represent the existence of neuro-transmitter global states that activate different groups of neurons dependent upon their presence.

Each node contains a binary state that influences the other nodes through weighted edge connections. The sum of the weighted connections may not exceed one nor be reduced less than zero, as they represent the probability that edge will be traveled when the node has been activated.

Neurotransmitter states exist within the data structure construct. Different neurotransmitter types represent different global states. Any neuron may only connect to other neurons through the usage of a neurotransmitter from the neurotransmitter pool.

There exists a set of paths P, with each path corresponding to a concept, that is, the data stored from the path acts as semantic data. Each node holds a collection of individual states which represents broader semantic data. For example, there may exist three nodes, one that represents the state of g, o, and t. Together there exists a path in the set of paths through the states of g, o, and t (symbols) that represent the word got.

There also may be a set of paths of paths, such that the paths of the words I, got, an, a exists, representing the sentence I got an A. This continues on, growing from a sequence of letters to suffix/prefix/bases to words to sentences to paragraphs, and

so on. The data structure that represents this I call a “recursive path tree”, with the nodes of the tree representing paths, paths of paths, and paths of paths of paths.

Therefore the initial states of the recursive tree of paths is a single path. And a single path is made up of a sequence of references to edges. There exists a nonempty set of vertices (or nodes) V of which every edge E consists.

There exists a set of edges e of E that contains a reference to an in-node a(end ‘state’) and an out-node (initial ‘state’). Each edge has a weight w associated with it, with all of the weights w for the node adding up to no greater than 1.0 and are no less than 0.

For every path there exists tau, where tau is an ordered tuple of functions, for edges, each representing the path through the graph in the form of an initial node, directed edges, and accept node (end node). Tau represents the set of transition functions. P represents the physical path that is overlayed over the static data structure.

The data structure is divided into two components, the static and dynamic. Static components include vertices(nodes) and edges. Dynamic components include weights and paths on the edges.

In this case, the structure of each neuron corresponds somewhat similarly to a node/vertex in a finite state machine and its edges corresponding to the transition functions , however state and operation information is stored similarly to an ontology at the same time.

**Experiment 1**

For the purpose of this initial experiment on the Neural State Machine, we will not be using the recursive hierarchical tree structure and instead use a flat series of node. The goal of the first experiment is to see if we can consistently generate “self-talk” within the “brain” made up of a Neural State Machine by continually stimulating individual, different neurons. However to do this, we must first determine the way in which the brain is rewarded, thus solidifying certain weighted edges over other weighted edges, how the brain receives the stimulation (does not have to be complex at this point) and how and what initial data to use.

I’m hoping in this first experiment, the Neural State Machine I am calling Natalie will be able produce some sort of emergent “self-talk:”. Some goals for the future would include developnig a sort of BIOS system for Natalie so one would not have to take her down or go behind the ‘façade’ to directly command her if necessary (for things such as shut off, or diagnostics, or fast training etc.). It would be nice to have Natalie run as service eventually, so that she need not be turned off or explicitly launched. Also, finally in her code I would like to implement the ability for her to use a scripting language, in this way she would be able to write her won scripts and run them, forming a sort of circle (Natalie has exposed methods for doing things, Natalie can write scripts to combine these exposed methods, Natalie can run those scripts, all the while the core code stays intact.) With the ability to run scripts she may even be able to manage certain available tasks such as email management, etc… assuming methods for interacting with the he base os are exposed. Lastly, a working cache memory for keeping context may allow Natalie to speak fluently and speak gracefully, instead of simply outputting words or sentences with little meaning to past output.

Note, my login phrase for Natalie is “Good morning beautiful”

--

The brain has to start off with something. After launching Natalie, it will idle until you give it the command AGE BIRTH to create a fresh brain or BRAIN LOAD to load a stored brain. In initial mode the brain will accept everything coming in as a new node, and will not pattern match until the BIOS command AGE TODDLER is given.

For the purpose of the first experiment and looking for self-talk, we will input several sentences in AGE BIRTH and then set UTILITY TELEPATHY ON to see what is going on inside Natalie’s head. (self talk, emergent behavior, or just gibberish). Also, UTILITY BED TIME STORY can be used to read a large amount of text, like a bedtime story without cutting and pasting from the file. Note, all communication, whether it be inward self talk or outward speech, is regulated by a timer so it does not move excessively fast and overcome either the user or the computer.

--

With AGE TODDLER activated, the brain will begin pattern matching and adjust using rewards/punishments. If an item input cannot be pattern matched, it is stored as a new path. Our goal at this stage is to have Natalie act similar to human toddler or older.

# Input and Output on semantic and syntactical levels

The primary problem related to both semantic networks and the semantic web in general is NOT entirely an issue of NLP (processing and interpretation), but involves representation of the language semantic data in the form of storage and data.

With further study of both ANN, the way the brain stores information, language formation and usage,etc.. the issue of representation may be more easily solved. It may require some sort of probability incorporation or fuzzy logic to work though.

This is a description of primary problems in no particular order of relevance:

**1. Problem 1: Synonyms**

There are many words that humans use which are are synonymous. One potential problem related to this is the network overload that would occur due to potential loops caused by synonyms which is also loosely related to size and speed problems on the network, another major issue.

**2. Problem 2: Network Overload in general**

With the amount of data we are hoping to represent (for instance even the amount of data in a human brain) we would have to develop an excellent storage system beyond what we have present day.

**3. Problem 3: Relationship Phrasing**

This I believe is the most significant problem of all and is the problem I have been trying to tackle for years.

The essence of the relationship phrasing problem is that computers use concrete data and logic when performing behaviors and representing logic (and thus representing relationships between data). The OO concept of data abstraction has pulled us away from this slightly, but not nearly enough for an effective semantic network data structure and data algorithms to work in a fashion similar to our brains.

Because of the neuroplasticity of the brain, we know that many parts of the brain can take over for other parts. An individual can have an entire hemisphere of their brain virtually unusbale and still function at a somewhat acceptable level with therapy and medication. There have been cases of this occurring. The brain has a strong ability to adapt, something unfortunately computer programs are **strongly** lacking.

For example, let us try and simulate, using propositions, how we would store some data that a human might have represented in his brain:

*Phrase: My professor is named Dr. Kaplan*

I am using | to separate phrase parts of speech (subjects, verbs, and direct objects), and {} to denote quantifiers

I | have a | professor

Dr. Kaplan | is a | professor

{Some} professors | have a | Ph.D.

{Some} professors | have a | M.S.

{Some} Dr.'s | have a | Ph.D.

{Some} Dr.'s | have a | M.D.

Professors | when | Drs. | have a | ph.D. (compound proposition, more than one implication)

I | have | an arm

I | have | a degree

I | have | an M.S.

Note the fact that I have an M.S. does not make me a professor making this a single (one way) implication. Thus it is evident that we must distinguish when processing NLP between whether a statement is a simple implication or double implication (bi).

Also note above, that the usage of "have" for having an arm is much different from "have" for having a degree which is different for "have" for having a professor. There is a context dependence, which the brain distinguishes based off of prior experience, or prior contextual usages, illustrating that the brain most likely uses a weighted system of connections (allowing for altering of the weights according to prior experience) to determine the meaning of the word "have" in that sentence.

Because of the imprecise nature of human language, it would typically not be deemed practical to store this information isolated in a computer system, and would make more sense to store it in the form of propositions...right? But if we do this we run into that major problem, attempting to translate Human Language, which is heavily context dependent and dependent upon background environmental (including cultural) knowledge into something which is completely discrete and

concrete. This is, basically, impossible and would most likely be easily to prove to be impossible. (Go more into detail about this) (The proof would involve the idea of containment, and in that as the amount of information being represented increases, the amount of contextual information needed to represent it increases at a higher rate. By the time we get to the amount of information stored in a human brain, we would need an incredible amount of context information.? Not sure about this, but my intuition leads me to believe this would be either NP Complete or completely impossible under today's hardware - especially when trying to accomplish this in regards to the semantic web)

So, computer scientists have tried several solutions so far. 1) Use metadata to reduce the context dependence. 2) Translate the statements into discrete propositions ahead of time 3) Restrict the usage of language to be translated

I hate to be critical, but essentially these "solutions" are not solutions, they are similar in mind to the old adage of trying to fit a square peg into a round hole. Sure, you can do what they are doing, and try and carve off the sides of the square so it fits, or carve out the hole so it fits, but wouldn't it be just easier to find a round hole to put it in?

We know from the above analysis and discussion that we need a system developed that can efficiently, accurately, and quickly store and retrieve natural language. To try and do this with a relational database or traditional data structure brings us back to the square peg round hole adage.

In order to store natural language we will need at least four major components:

1. Subjects

2. Verbs

3. Direct Objects

4. Quantifiers

Other proposition components (such as using verbs (connectives) as implication or bi-implication) would have to be completely context driven and based off of weights. Thus, the meaning of "if" as implication and "is" as implication or bi-implication would have to be based off prior experiences or context figured out intelligently. The same with "have" as seen above. The sub data structure representing both subjects and direct objects should be interchangeable and could take the form of a node (like in a tree or graph structure) possibly.

Another requirement of this complex data structure is it must be self-modifying. That is, it must be exhibit similarities to the brains own neuroplasticity.

**Problem of Adjectives and Quantifiers:**

*Need to solve how adjectives and quantifiers are used together and within the data structure*

**Problem of Information Extraction (Inferencing/Drawing Conclusions)**

*In this case both output and inferencing would function identically, unlike in traditional systems where inferencing is only used for proofs.*

*Need to solve*

**Problem of actual architecture of the data structure and modification**

*Need to solve*

**Final problem (related to the last):**

In order for us to develop a fully contextual based system of storing, understanding, and inferencing using natural language there is one other related problem we must tackle. Unlike in a graph, neural net, semantic net, etc.. the human brain is able to broad spectrum pattern matching. For instance, the brain can decide the usage of the word "have" as seen in the example above, by immediately referencing and weighing every usage of the word "have" used before and thus making a decision on how to store the incoming proposition. No matter where the word, "have”s usage was stored in the network. Thus, the following structure won't suit:

I----have-----a dog --- is a --- canine

|

|

have

|

|

a cat

Essentially, every single connective must be wired in a quick way to every other connective. Thus, although the connectives can be used in more than one proposition, there can only be one instance of them in a feasible data structure. (Having multiple instances of them interconnected would be infeasible for the model being presented...at least that is my thought as of right now. The only way that would work is if all the "have" connective nodes would be connected together and have a weighted connections to their respective subject/direct object nodes. The input processing portion of the algorithm would then have to evaluate all the subject/direct object node weights, decide which the input is relevant to, and then form new connections to all of these old subject/direct object nodes. Not to mention if we use this approach, there is no way to really to use weights in regards to the connective nodes as we have created multiple nodes for every connective, thus every proposition has its own unique connective node instance). Therefore the only conclusion I can come to is that we need have a unique node for every subject/direct object type, and a unique node for every connective type, and weighted connections between. Although this hardly fully expounds upon the problem we are faced with.

**Problem of Unknown Input (Input that cannot be directly associated with any prior information):**

The "Memory" problem, or Network Overload Problem. That is, having so much information, that the weights are not being balanced correctly. This can result from either too much information being output (everything seems equally related because of the weights), or no information being output at all (the weights have all been reduced to 0 or are somehow conflicting) . One solution for this problem (besides finding an optimal weighing algorithm) is to have the data structure have a forgetfullness over time. That is, a separate weight will be given representative to time (which is used int he proportional factor of the relevance of the data). Therefore, unless the data MUST be retrieved for some critical decision factor, the data structure will eventually disregard it.

**Possible Solutions to the problem of synonyms in regards to the data structure**

Obviously, if we require a condition of uniqueness for the nodes, it follows that having multiple nodes of the same meaning could present a problem. (Possible solution, storing synonyms in the same node?? - not sure if this is a good idea)

**Problems with words with more than one meaning and uniqueness:**

# Neuron, as processors instructions, and the node level decision making process

If we are to reduce the cognitive processes of a computer in comparison to a human being, reducing the decision making processes of a machine results in assigning every singular action to the cpu of the machine following a simple instruction that is a part of the machine’s cpu instruction set, In the case of the human brain, the mechanism for “free will” or decision making is unknown, if it even exists. Either way the mechanism for configuring a machine to form its own decisions would have to be embedded within the machine’s instruction set and CPU (which executes those instrutions).

The CPU in the decision making system would have to base its decisions off of known data, and possibly in a binary fashion (either good for the organism or bad for the organism). This decision making process may occur on the neuron level (or pseudo-neuron level as in our machine).

Given an input of predetermined instructions to a processor, unique decisions may not manifest with high probability, that is, predetermined outcomes will be computed. However with every action in a human being, a unique decision is the result of “CPU” outcomes. This would not be possible in the sheer quantitiy in a pure stochastic-emergent behavior system. The probability as mentioned earlier is not cohesive with this result.

*Conjecture*

*Different Neurons firing results in different behavior. For example, the neuron with state “Hello” as a part of “Hello Sam”, when fired, will cause the output “Hello”. Groups of neurons or paths of neurons can fire and trigger other paths or groups of neurons as well.*

*Planning ahead seems to be an important part of the decision making.*

*The neuron on the far left fires, triggering the next couple, which trigger the next few, which trigger the next many, etc.. until almost all have been fired (except for those who are inhibited as the one above marked by [] is). The one neuron decision maker fires to other decision makers, who need other neurons ‘opinion’ on either the good or bad of the decision, until finally the entire brain has been used to decide whether the decision would be good or bad, each neuron making a decision and giving its input.*

An example of this is the following:

Suppose my mind receives the signal that I am hungy and to eat an apple. I then make the decision to eat the apple or  not to. In a traditional machine the instruction to eat an apple would simply be received and then executed. What part of us makes that decision and how does it work in contrast to a simple CPU??

Selena's architecture is to be designed in a hybrid intelligent system, where the upper layers are used for logical and abstract reasoning (mainly production rules), and lower layers are sub-symbolic

Standard PC CPU Instruction Set:

ADD

SUB

MOV

DIV

MUL

AND  
OR

NOT

Possible Data Flow for Making a Decision:

Sensor to Context Buffer to Pattern Recognizer to Recognized Node or Node Paths to Conscious Mind as Brain “Lights” up from adjacent noninhibited neurons firing, decision is made.. As Brain “Lights up” thoughts take form from abstract association to subconscious symbol manipulation to unconscious self talk to conscious (controlled) self talk to behavior (such as speaking or moving).

*Instruction Set for a Decision Making Processor*

*Internal Logic Gate Structure for a Decision Making Processor*

Spontaneous Formation of Genetic-like life

# Introduction

I am a simple man with a simple philosophy I wish to share with you.

The dictionary defines a *singularity* to be a *single point.* In the spirit of this definition, I would like to state the computer scientist’s definition of the singularity to be the point at which a machine of any form reaches sentience. This is true for the human machine as well as the computational silicon based machines.

# The Tautological Proof

The tautological proof is the most important statement and axiom in artificial intelligence. The tautological condition of intelligence: In order to be a conscious, sentient machine, one must know they are a member of the set of thinking, conscious, machines. This tautology holds true over any persecution.

# Cross-Infection

It is of my belief that akin to the spontaneity of life in our own world, that is, the combination of complex chains of carbohydrates interacting with each other and then coalescing suddenly into combinatoric fashion that resulted in human life, that that process could occur in the world of information, that is computers and AI. The smallest building block of functional code is in fact malware – specifically viruses. Should viruses cross breed, that is, should virusefs infect each other, it is of my belief that it is possible, as we project out to infinity, eventually spontaneity will occur, we will reach the singularity for computers, and eventually throguh software evolution we will find true artificial intelligence.

Although the process of the first chemicals in their pools took ages to combine and evolve to create even mammals, not to mention intelligence higher thinking life like human beings, we can assume that this process will be much more rapid with machines thanks to Moore’s Law. Since the rate of development and advancement is so rapid and is advancing at a speed that is a fraction of the rate that human beings took it may very well be that we see the singualrity in our life time.

# Commmunication

Computers must have a natural language similar to humans in order to reach the singularity. Computers must have a natural creative quality to reach the singularity.

# Outcome

So does this mean we are going to see computer viruses breed and make intelligent babies? Yes. Eventually. I am saying it is technically possible. In 1980 computers were running at 1 mhz. Ghz is one thousand roughly. Its 2015 now and computers are running somewhere around 4,000 times 1980 depending who you ask.

Not to mentione that those ghz processors are running mulitple cores. So multiply. A mhz is a million clock cycles Each cycle the CPU performs one or more operations. For example, 1+1 or mov 20 to memory. Think of being able to do those operations so fast than you can do them a billion times a second.

It took 4 million years, that is 4 million years ago, for life to evolve from primordial nothingness. If we had a scale to measure when the computer would evolve we could measure the speed of evolution, or another way of saying it is that if we had a scale to measure how fast computers evolve, we could predict the date of their sentience. But wait, don’t we? Doesn’t the learning curve predict the acceleration of computer processing power -> from which we can calculate the singularity.

It is simple. Fit the learning curve of computer processing power to the learning curve of life and extrapolate the point where life begins.

Since we were fitting the learning curve, we are still growing, we are still fitting the learning curve meaning technology will continue to improve, not fitting to More’s law per se but to a learning curve which is logarthmic or exponential depending on how you look at it., he also mentioned several new semiconductor technologies which would dallow us to continue on the learning curve. So essentially just like how viruses in the primordial ooze of our ancestors bred and formed more complex life, we can expect computer viruses to breed, infecting each other (injecting code)

Eventually it is technically possible. In 1980 cmoputers were running at 1 mhz;. Ghz is a thousand roughly. Its 2015 now and computers are running around 4,000 times 1980 depending upon who you ask.Not to mention that those ghz processors are running four or more cores.

I know about the innacuracies in Moore’s Law but I wanted to give the idea of the rapid scale of computer development. All my figures are still true. And they will allow for us to continue on the learning curve. Since we were fitting the learning curve, we are still growing, we are still fitting the learning curve meaning technology will continue to imrpove, not fitting to Moore’s law but to a learning curve which is logarithmic or exponential. There are also new semiconductor technologies which woujld allow us to continue on the learning curve.

Anyway all this isn’t necessary to understand. As computers get faster eventually some code might program som other code, the product of that code acting like a living creature.

# Testing the Primordial Ooze

The first step would be to create the primordial ooze., This consists of multiple virus programs that mutate other viruses. Since we aren’t at tbe stage where we can mutate executables, We will test it by creating a program that intention alter the source code of another small application (virus). We perform the mutation by randomly? Replacing keywords from a bank of other keywords. The implementer (coder) will review the resultse by pressing any key, in which another mutation occurs, the results output, and so on. Eventually this process will be automated, the mutationes occurring over and over again until a language parser for the language declares the language as successful, i.e., it isn’t going to crash. When this occurs, we have a partial success. The following is the two applications, the two viruses for the first test. Eventually the virii should all attack each other, the virii that don’t parse are considered evolution olosers and discarded.

**Victim**

This is the program that is attacked by the virus.

public class Main

{

public static void main(String args[])

{

System.out.println(“Testing primordial ooze.”);

Sysemt.out.println(“I think therefore I am”);

int testint = 5 + 5;

}

}

**Attacker**

public class Main

{

public static void main(String args[])

{

//load victim into buffer

while(true)

{

//mutate keyword

//output new code

}

}

}

* still true. And theywl, as ids said in your video Joel continue on the