The Mind: Version One.

The mind is a part of speech tagger that uses manually encoded patterns to extract information from a text stream. In this iteration we will create data structures to hold text info and define algorithms to extract patterns from the text.

New idea, what if we allow any kind of node on the network x at any time? After all, names can have logic in them, and patterns can have simple pieces of data in them. Maybe It should be the machine’s job to handle the matching, and we don’t care what’s in the graph. We just care if we call it a pattern or an object up front.

Patterns require objects but objects don’t require patterns. Objects can include patterns, but if you zoom all the way in you will find an object at the end of the line. Although, maybe we don’t care about patterns at the object level. Objects at their most primitive are a simple unit of information: A letter or a logical function. Patterns and objects have the same nodes, it’s the edges that are different. So they should share some code, but there are also differences that we have to address.

Patterns have an unlimited number of inputs and outputs, but no children, only next nodes. Objects have a single input and can also have children. These differences should be handled at the object level, not the node level.

The question is, can objects ever include patterns? Maybe we don’t care about patterns when talking about objects, after all an object is an *instance* of a pattern, a physical representation of an abstract concept.

Features:

* **Create Network X Node to hold graph structural information**

We will need a structure to store basic structural information about the graphs we will use in this program. This structure will be set to the data value of Network X Nodes, and will be used to keep the graph structure from the data.

Using Network X will give us access to better performance, more algorithms, and a bug free structure to build off of. The data object will store any information not related to the structure, and can be either a mind node or a mind pattern node.

We may be able to get all this functionality from the basic network x node.

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Children | Array | Array of network x edges linked to mind\_edge objects which point to network x nodes in lower layers. |
| Next Node | Network X Node | The next node in this layer |
| GUID | Guid | Global id of the graph. May be able to make it only global in the context of the graph. |
| Data Object | Mind Node or Mind Pattern Node | The object related to this network x node. The data object stores any information not related to the graph structure. |

* **Create Mind Graph Object to hold parsed text data.**

We will create a data structure that maps layers of patterns to text input. We will build this off our Network X Node, and will also create a “mind\_node” structure, which will be a required object for each of the network x nodes, and a “mind\_edge” structure which will store information about the graph’s edges, such as whether an edge is context or data and exclusive or inclusive. Ideally, all of these objects will be convertible to JSON.

**Mind Graph Object**

An unlimited number of layers of linked lists, made up of network x objects linked to “mind\_nodes”, each layer representing a more abstract representation of the data. Each node can also point to any member of the data below it with a “mind\_edge” object.

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Data | Network x Node | Graph of Network X Nodes that stores all the data we need |
| Guid? | Guid | Global id for this object |
| Source | Text | Where did this data originally come from? Any extracted data will be stored in the database attached to this source. |
| Author | Text or db id. Optional | Who wrote this data? |
| Date | Datetime | When was it written? |

**Mind Graph Node**

A mind node will be set to the data object value of a network x node. It stores any information about this specific piece of parsed data, and this table will probably grow as we think of more things to add.

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Pattern | Pattern Object | The pattern we applied to child data and child context to create this node |

**Mind Graph Edge**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| IsContext | Boolean | Is this edge context, ie required for the pattern but not part of the parsed object? |
| IsExclusive | Boolean | Can this node only have one edge? |
| IsData | Boolean | Is the linked node part of the parsed data? |

* **Create Data Structure Database to hold all information that we know, as an ontology.**

Sometimes for parsing we need external information, for example a list of people names or place names. We will store this information in a database as below:

|  |  |  |
| --- | --- | --- |
| Pattern | Mind Graph Object | Sources |
| Person | “C”-“A”-“M”….. | List of links where we’ve seen this. |
|  |  |  |
|  |  |  |

* **Create Data Structure to hold patterns**

Our patterns will be stored in a similar way to the text. We will create a graph which will logically define which elements match this pattern.

**Mind Pattern Object**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Data | Network X Node | The inpoint of our graph. Graph of Network X Nodes that stores all the data we need |
| Name | String | The name of our pattern |
| Id | String | Unique path to our pattern. Will be defined like a file path |
| Matching Objects | Db query | List of all mind graph objects that match this pattern. Ie for a pattern “word” we would provide a list of all known sequences of letters that map to a word that our stored in our database. |

**Mind Pattern Node**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Parent | Network X Node | Which Network X Node this is attached to |
| Iterations | Int | How many times this element can match |
| Matches | Mind Pattern | What this node has to match |

**Mind Pattern Edge**

We may need to store information about the edges, so if necessary we will create an object for this. This could be very useful for propositional logic.

Might need a mind pattern logic node to provide if then, or, and, xor, etc functions

* **Create algorithms to extract patterns from data**

To accomplish this, we will create a Mind Machine object which performs operations on data given certain patterns

**Mind Machine**

|  |  |  |
| --- | --- | --- |
| Function | Input Output | Description |
| ExtractPatternsFromGraphObject | Mind\_Pattern\_Array, Mind\_Graph\_Object | Adds a layer to the Mind\_Graph\_Object with extracted Patterns |
| Prune Graph Object | Mind\_Graph\_Object | Trims all possible parses except the best one |
| Step Graph Object Through Pattern | Mind\_Graph\_Object, Mind\_Pattern | Checks the first step of the pattern for a match in the object and saves the state. |
| Get Object similarity to a Pattern | Mind\_Pattern, Mind\_Graph\_Object | Checks for how many changes to the object are necessary to make it fit the pattern. Should be useful for catching spelling errors |
|  |  |  |
|  |  |  |
|  |  |  |

* **Provide input / output functions**

Each of the graph objects should be printable and graphable for debugging purposes. Every object should also be convertible to and from JSON. We can use matplotlib pyplot and network x for the graphing, and simply print out the json for the print functions. We will also need to define a logical folder structure that stores layers of nodes, or we could just use the node id for classification.

* **Create patterns to parse POS information**

Once we have all the groundwork in place, we will create patterns to match as many text elements as possible. This includes, but is not limited to, the following:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Letters* | *Numbers* | *Words* | *Nouns* | *Verbs* |
| *Phoneme* | *Adjectives* | *Prepositions* | *Prepositional Phrases* | *Dates* |
| *Emails* | *Proper Names* | *Places* | *People* | *Article* |
| *Signature* | *Title* | *Model Number* | *Company* | *Interjection* |
| *Amount* | *Measurement* | *Logical Statement* | *Currency* | *Object of the Preposition* |
| *Direct Object* | *Indirect Object* | *Predicate Adjective* | *Predicate Nominative* | *Pronoun* |
| *Analogy* | *Punctuation* | *Sentence* | *Question* | *Quote* |
| *Work Title* | *List* | *Math Formula* | *Country* | *City* |

* **Stretch Goal: Create Data Structure: Database of relationships**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Subject Id | Verb Id | Object Id | Object Descriptors | Sources |
| Cameron | Is | Person | Smart | [http://camsbrain.com](http://camsbrain.com/) |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

For project management we can use Heroku running hubboard for item and feature management.