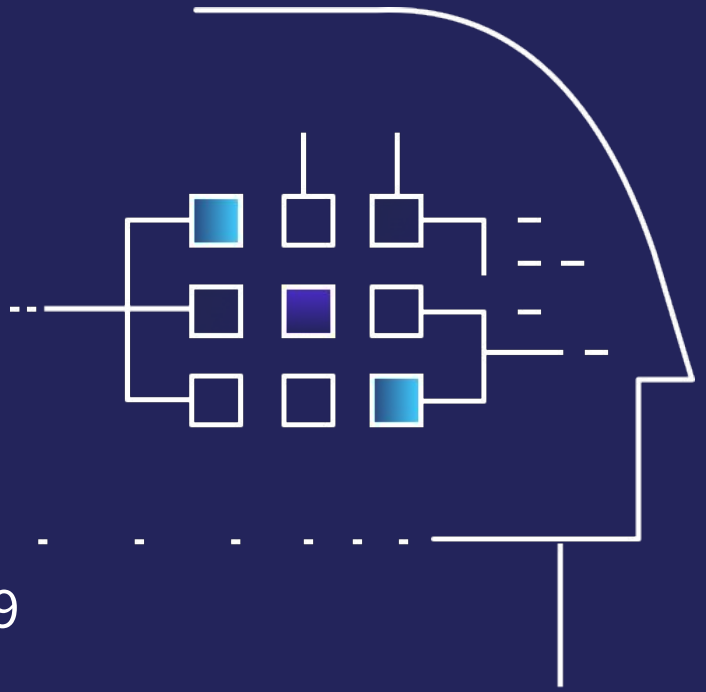
SingularityNET



Technical Deep Dive

OpenCog Workshop, Shenzhen 2019

ManHin Leung
leung@singularitynet.io

https://github.com/singnet/opencog-workshops

Initial Setup

1. Load the .tar file from the USB drive:

   ```
   docker load -i opencog_workshop.tar
   ```

2. To run:

   From browser, go to localhost:8888, password is password

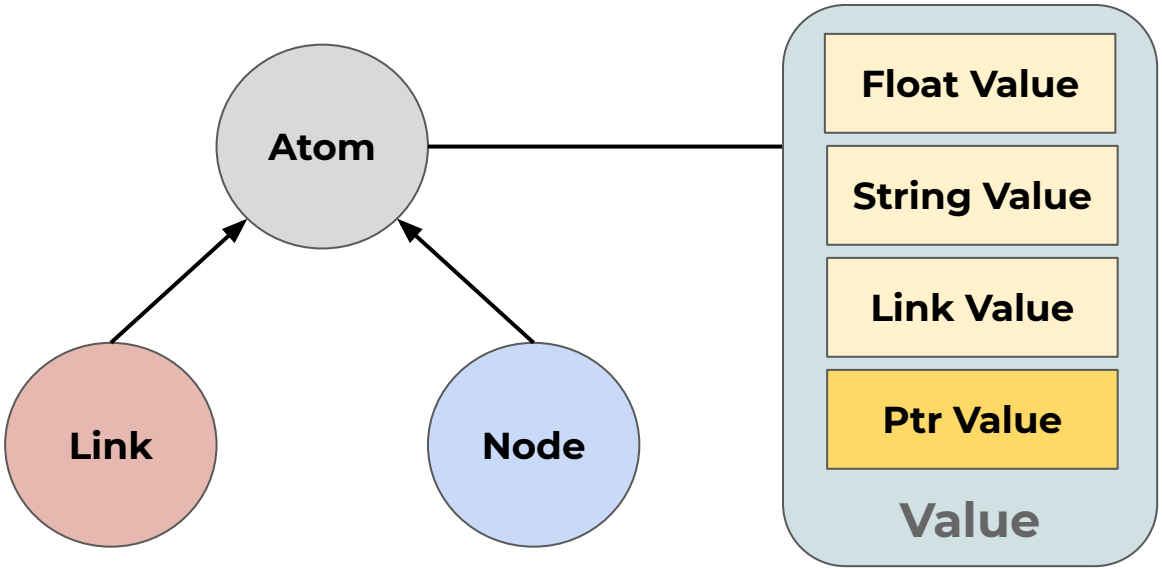
   ```
   docker run -it -p 8888:8888 opencog/workshop
   ```

OpenCog - The Open Cognition Project

- Open source framework for Artificial General Intelligence (AGI)
- Diverse assemblage of cognitive algorithms, e.g.
  - ECAN - Economic Attention Allocation System
  - PLN - Probabilistic Logic Networks
  - OpenPsi
  - MOSES - Meta-Optimizing Semantic Evolutionary Search
  - Natural Language Processing & Generation

AtomSpace

- Hypergraph database
- Holds ATOMS together with their VALUES



Bindings

Scheme

```
scheme@(guile-user)> (use-modules (opencog))

scheme@(guile-user)> (define my_atomspace (cog-new-atomspace))
```

Python

```
>>> from opencog.atomspace import AtomSpace

>>> my_atomspace = AtomSpace()
```

Haskell

Demo: **Knowledge Representation**

atomese.ipynb

Pattern Matcher

- Query engine
- Finds graphs that match the given template
- Evaluates and executes certain subgraphs

Pattern:

```
InheritanceLink
    VariableNode $x
    ConceptNode “animal”
```

Practice: **Pattern Matcher**

pattern_matching.ipynb
evaluation_and_execution.ipynb

Unified Rule Engine (URE)

- Generic rule engine
- Supports forward chaining and backward chaining
- Built mostly on top of the Pattern Matcher
- Rules are written as BindLink
- Rules can be organized as a Rule Base, with customizable control policy for controlling the inferences

Rule Structure

BindLink

    <variables>

    AndLink

        <premise-1>

        ...

        <premise-n>

    <conclusion-pattern>

Deduction Rule Example

Premise condition = AndLink(BA, CB)

A -> B BA = InheritanceLink(var_b, var_a)

B -> C CB = InheritanceLink(var_c, var_b)

Ergo: A -> C

```
Rewrite = ExecutionOutputLink(
                GroundedSchemaNode("scm: deduction-formula"),
                ListLink(CA, CB, BA))
deduction_link = BindLink(condition, rewrite)
```

Probabilistic Logic Networks (PLN)

- Carries out uncertain inference
- Allows basic probabilistic inference to interact with other kinds of inference
- Supports sophisticated control mechanism enabling inference control meta-learning

Demo: **Reasoning**

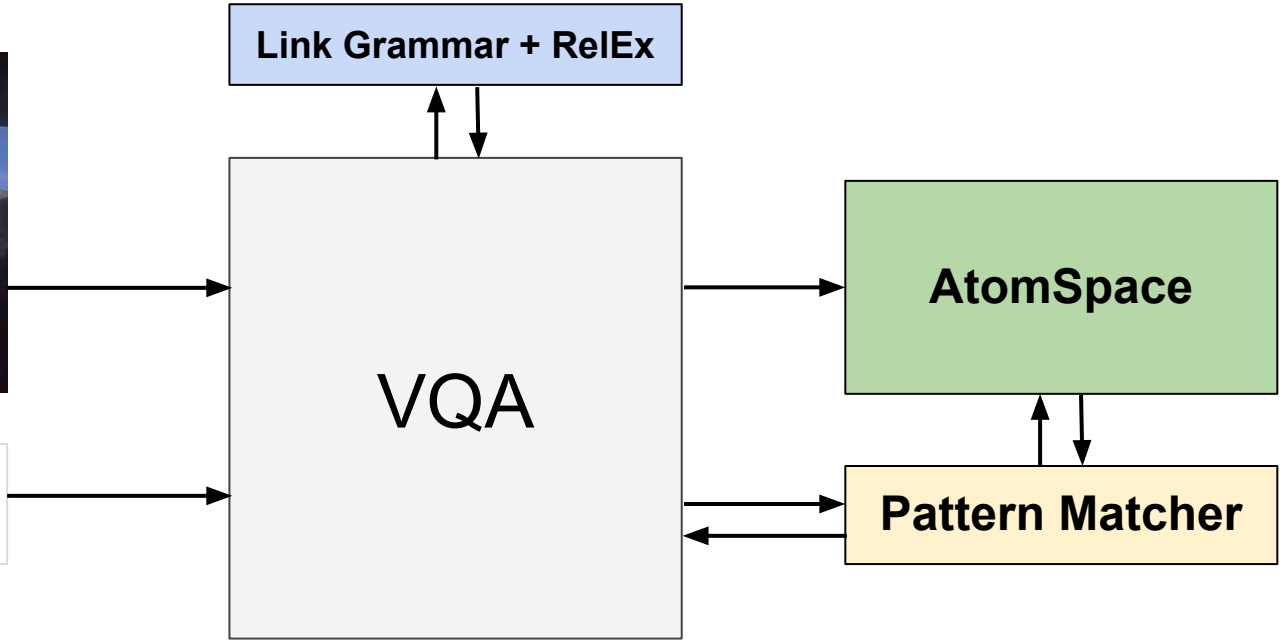
reasoning.ipynb

Visual Question Answering (VQA)

- Atomspace for storing facts about world
- Link Grammar + RelEx for text processing
- Faster-RCNN for bounding box and feature extraction
- Our neural network models for classification
- Unified Rule Engine and Pattern Matcher for answer searching

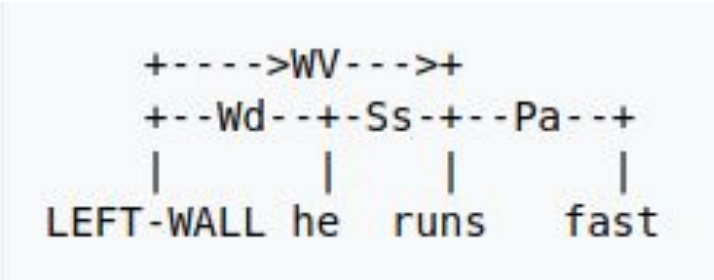
Visual Question Answering (VQA)

Is the cat black?



Link Grammar

- A syntactic parser
- Builds relations between pairs of words
- e.g. “he runs fast”



RelEx

- Dependency Relationship Extractor for English

Dependency relations:

_advmod(run, fast)

_subj(run, he)

Attributes:

pos(run, verb)

penn-POS(run, VBZ)

penn-POS(fast, RB)

definite-FLAG(he, T)

pos(he, noun)

penn-POS(he, PRP)

tense(run, present)

pos(fast, adv)

noun_number(he, singular)

gender(he, masculine)

pronoun-FLAG(he, T)

Demo: **Visual Question Answering (VQA)**

intro-vqa

Resources

wiki.opencog.org

blog.opencog.org

github.com/opencog

opencog@googlegroups.com

opencog.slack.com

https://blog.singularitynet.io/research/home