We describe ongoing work on the PowerLoom Chameleon 2.0 reasoner which combines traditional symbolic reasoning in a first-order logic framework with neural network learning and inference to make PowerLoom’s inference more general, flexible and robust. This is an early progress report that describes the basic design and simple examples. More robust evaluation of the results of this work must be deferred to future work.

PowerLoom\(^1\) is a logic-based knowledge representation and reasoning system that allows the representation of complex knowledge in a declarative, logic-based language and supports a variety of reasoning mechanisms to make implicit knowledge explicit. It has a query engine to retrieve asserted and logically implied statements from the knowledge base, provides persistent storage, a context and module system to organize large KBs, and has an extensive API for integration into other applications.

The basic representational units are predicates for types and relations, as well as rules which can define implied relationships that logically follow from other relations and rules. The following example shows a rule for a simple inverse relationship between `parent-of` and `child-of` and a query that uses it to find Fred’s children:

\[
\]

\[
(retrieve all (child-of ?c Fred))
\]

While these constructs allow one to build very complex and sophisticated knowledge bases (e.g., (Lenat, 1995)), a main and valid criticism of the approach has been its rigidity and brittleness along a number of dimensions:

1. Hard truth values: something is either true or false, there are no gradations
2. Complete preconditions: if just one of the preconditions in a rule cannot be satisfied, the rule cannot be applied
3. Fixed vocabulary: if an instance or relationship does not fit into any of the predefined types and relations, it cannot be represented
4. Fixed, limited rule set: the rules are generally hand-coded and always incomplete, new rules are often difficult to add without disturbing other inference paths

There is a large body of research that tries to address these issues, for example, fuzzy logic (Zadeh, 1975) and probabilistic logics such as MLNs and PSL (Richardson and Domingos, 2006; Bach et al., 2017) allow truth values to be soft or probability estimates, non-deductive inference such as abduction can address missing preconditions (Stickel, 1990), and ontology and rule learning approaches such as inductive logic programming can automatically learn or extend vocabulary and rule bases.

PowerLoom’s partial matcher is an abductive inference engine that supports soft truth values and that can handle unsatisfied preconditions. It has been applied to support debugging of large KBs (Chalupsky and Russ, 2002) as well as for case-based reasoning (Moriarty, 2000) and activity recognition from noisy data (Adibi et al., 2004). For example, if in the example rule above only the second and third antecedent conjunct were satisfied, we might conclude the `child-of` relation with a score of 0.66 using a simple weighted average.

To avoid such ad-hoc score computation schemes, Moriarty and McGregor (2000) developed the first version of Chameleon (1.0) which made these computations more principled and learnable from data. To do so, a neural network was associated with each rule which would take the current soft truth values of the antecedent clauses as inputs and then computed a corresponding soft result value. The networks were trained on possibly recursive partial match proof trees derived from training examples to minimize the total error over all examples. Backpropagation of error from one rule network to another was performed through the connections in the proof tree. Depending on the training examples, the networks would

\(^1\)http://www.isi.edu/isd/LOOM/PowerLoom/
learn different weight combination semantics, thus the name Chameleon. For our ongoing work for Chameleon 2.0, we are reimplementing and extending Moriarty’s system along a number of dimensions described below.

**Softening vocabulary through embeddings:** A restriction of Chameleon 1.0 was that network inputs are handled purely at the clause level. In our example rule above, the associated network’s Input 1 would simply consider the soft truth of (person ?p) but nothing about a specific instance such as Fred binding the variable ?p. As such, vocabulary such as person is still “hard”, either satisfied or not, albeit softened through inference and weight computations. For Chameleon 2.0, we are introducing embedding relations that allow for rich high-dimensional similarity spaces that can be taken into account by rule networks, for example:

\[
(\Rightarrow (\text{and} \ (\text{person} \ ?p) \ (\text{person} \ ?c)) \\
(\text{embedding-of} \ ?p \ ?ep) \\
(\text{embedding-of} \ ?c \ ?ec) \\
(\text{parent-of} \ ?p \ ?c)) \\
(\text{child-of} \ ?c \ ?p))
\]

Here embedding-of is a user-defined PowerLoom relation that given an instance ?p can look up or compute an embedding vector. When Chameleon encounters this rule for the first time, it looks up meta-information such as dimensionality and type of the embedding vector, as well as how to access the actual numeric information, and then builds the appropriate network with inputs activated by the embedding vector’s dimensions in addition to the soft truth value information as before. Multiple embedding relations can be defined which can take one or more input arguments. Embeddings might come from natural language resources such as Word2Vec (Mikolov et al., 2013), other media inputs such as images or video, or be computed directly from knowledge graphs. Embedding relations are functions whose input arguments need to be defined to compute the embedding, which will be ensured by PowerLoom’s clause optimizer. Moreover, even though they are antecedent clauses in the rule, their truth value will be ignored by the neural network machinery.

**Arbitrary neural vector input:** Similar to embedding vectors, arbitrary neural network vectors, e.g., from a face or object recognition system, might form additional input to a rule network. A relation such as face-vector might take an instance and produce the output or late-stage vector of some face recognition system which can then be used by the rule network to learn how to compute the appropriate soft output truth value. As before, meta-information about the relation tells the network builder how to take this vector into account.

**Learnable rule combinations:** When a particular relation can be inferred by multiple rules, Chameleon 1.0 uses an ad-hoc combination strategy such as “max” or “noisy-or” to combine evidence from different rules. For 2.0 we are generalizing this to allow the system to use rule networks to learn how to combine multiple pieces of positive and negative support. This allows the system to learn, for example, how to handle conflicting rules such as “birds fly” but “penguins do not”.

**Explanation:** Neural networks are highly opaque and their inferences difficult to explain. Logic proofs on the other hand are structured and can be rendered into useful explanations. For Chameleon 2.0 we plan to extend PowerLoom’s explanation machinery to additionally describe pertinent aspects of the neural networks’ score computations, e.g., where they significantly deviate from a more standard logic-based interpretation of a rule. This is helped by the fact that we are dealing with a somewhat modular system of many small neural networks that are only sparsely connected and associated with small, generally “understandable” rules, compared to the case of a single, complex multi-layered neural network where everything is connected to everything else.

**TensorFlow:** Finally, as more of an engineering consideration, we are using TensorFlow to define and train the various rule networks, instead of the special purpose implementation of multi-layer perceptrons as was done for Chameleon 1.0. This allows us to leverage all the high-performance parallelism, GPU support and learning machinery provided by TensorFlow, but it also opens the door to construct arbitrarily complex networks whose structure could be controlled not just by the shape of rules but by arbitrary logical and/or Chameleon inference.
References


