Paper: Combining RL, LLM, and Logic

YKY

May 10, 2025

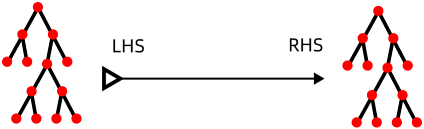
Contents

Part I

GPT envy

AGI as graph rewriting

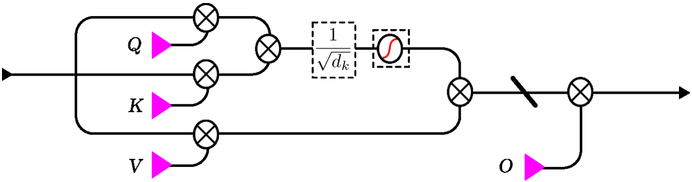
- Ben Goertzel in his *General Theory of General Intelligence* posits AGI as a [hyper- or meta-] graph-rewriting system.
- YKY proposes a similar but less general architecture where Working Memory = tree, and inference = tree-rewriting a.k.a. term-rewriting.



- YKY further proposes a **differentiable** version of the rewriting rules, and compares them with the traditional Transformer.
- Abstract rewriting can be formulated categorically as morphisms satisfying certain commutative relations. This is an **algebraic** way of formalizing intelligence, using **discrete** data structures.

String-diagram Kung-fu

- This is a string diagram of Self-Attention:



- It is possible to “morph” string diagrams via string-rewriting.
- Some researchers proposed **universal approximation** as an equivalence class of functions under string-rewriting. But such a class is too permissive and pretty useless.
- We need more *refined* string-rewriting so that we can distinguish between architectures that are more **learning-efficient** than others.
- One source of **inequivalence** is that functions like sigmoid and softmax can irreversibly “tear” the input space into disconnected regions due to finite precision. This gives rise to “**discretizing**” operations such as the ability of Transformers to output discrete tokens.

Counting computations

- In order to answer the ultimate question “how to design more efficient AGI architectures”, we need to **count the number of computations** in each morphism.
- We start with a Working Memory representation that we are familiar with – I suggest a tree or forest of trees – and with a set of reasonably versatile rewriting rules as morphisms.
- Develop a set of “string-fu” skills to “functor” from one architecture to another, while 1) keeping track of discretizing operations, and 2) counting computations; till we end up with a category with the most efficient morphisms.

Part II

Logic Transformer

Curry-Howard isomorphism

References

Thanks for watching ☺️