An Artificial Generalized Intelligence System That Combines The Capabilities Of Feature-Extracting Neural Networks With The Power Of Traditional Computer Memory

Rafi Qumsieh
rafiqumsieh@gmail.com

January 6, 2020

Abstract

In this paper, we discuss a system that can learn different patterns from different sensory inputs by using a feature-extracting neural network followed by a short-term and a long-term computer memory systems. The system components work together to store and predict associations between inputs based on certain rules assumed about the human brain.

1 Introduction

Realizing an artificial generalized intelligence system that can learn associations and relations without prior adjustment or configuration specific to the task at hand has been an endeavor for scientists.

The motivation behind the system that will be described in this paper comes from the idea that humans, in general, are better than machines at classification tasks and approximations of functions and relations, but computers are far more superior in storing and retrieving memories. Therefore, a system that combines the power of both could be very efficient at both tasks.

The system this paper will be discussing is an artificial generalized intelligence system that consists of 3 sub-systems (modules) that are working together, namely: A feature extractor/input encoder neural network module, a short-term memory module (STM) and a long-term memory module (LTM). The system combines the power and advancements that have been achieved in the field of Artificial Neural Networks in feature extraction along with the robustness, speed and efficiency of computer memory for association tasks. Keep in mind that there are assumptions being made in this paper that might not be

backed by data, but simply made sense to assume. The purpose of this paper is just to introduce the concept of such systems.

I should put an image here to show the system components

1.1 The Feature Extractor/ Input Encoder

This component receives an input x that can be in any shape or form. It takes the input and it runs it through the feature extractor neural network (Pre-trained Convolutional neural network, Scale-down neural network, a PCA algorithm, an RNN including LSTMs) and converts it to a representation binary vector. The vector can then be optionally run through a sparsity-producing network (Untrained Hopfield network, Up-scaling neural network) that maps the representation vector to another vector in a higher dimension to avoid possible collisions between inputs. Either way, the state of the last vector of size n in the process is read as a binary vector with shape $\{0,1\}^n$. The state gets sent to the next component, the short-term memory.

1.2 The Short-Term Memory (STM)

This component receives the binary state vector from the previous component and stores in an in-memory key-value database (Redis, Memcached, etc.) along with an entry timestamp and an expiration timestamp. Items in the STM get sent periodically to the next component (LTM) along with their entry timestamps. Every item that expires will be removed from the LTM. A database system like Redis contains the functionality of setting expiration times of objects stored in it, and automatically removing them. The STM module attempts to mimic the short-term memory of humans where we are constantly adding new items to / removing items from our working memory, then we consolidate these memories in our long-term memory. We can also define a maximum number of items that can be held in the STM at any given time.

1.3 The Long-Term Memory (LTM)

This component receives the items from the STM and stores them in a disk-based relational database (Most RDBMS systems or graph databases can work). For every pair in the list of items that are received, the system looks up the pair in the database, and if they exist in the database, their score is incremented by an amount that is inversely proportional to their timestamp differences. If the pair has not been stored before, it is inserted into the database with a score that is inversely proportional to their timestamp differences. The rationale behind the idea that the scores should be inversely proportional to the time difference is that we want to relate objects that appear closer to each other temporally more strongly. During lookup/search, this component returns the highest scoring items and sends them to the short-term (STM) component. This is akin to recalling/retrieving a memory from long-term memory if an input activates that memory or a similar one. If many values are being looked up/ searched, the

module will find a set of highest associations for each item searched, then it will return the intersection of all these sets.

2 Overall Analysis Of The System

The system learns by using two main maps: Dimensionality reduction / Feature extraction and Association. The first component (Neural Network) is responsible for receiving (sensory) input data and extracting the important features as a binary vector from it. The features are then used to represent the object internally. It is important to note that this network can be used on any data (images, sounds, text). Any data will be converted to a binary representation vector so that the other internal components are ignorant to what type of data it is. All the internal components see are binary vectors. This is good because the internal systems now can associate different sensory inputs together (e.g. an image with a sound) because they see them as binary vectors. The data is then sent to the next component (STM), which represents the current state of the system. It keeps all the currently seen vectors and all the vectors that will be retrieved from the long-term LTM component. The fact that this memory is very dynamic in the sense that it is constantly adding and removing vectors makes it similar to what we, as humans, think of our short-term memory. The short-term memory communicates with the long-term memory (LTM) and periodically sends it the vectors/objects that reside in its state. When vectors are sent to the LTM, the LTM searches for them, then inserts or updates them depending on whether they already exist in the LTM. In any way, the LTM will return the pairs with the highest scores to the short-term memory (STM). This is akin to memory retrieval that we experience when we are receiving sensory input. Running an instance of this structure against a stream of input will most likely create a system that is able to receive information, relate between its items, predict output based on previous memories. A typical table in this database will have the following columns: id, input1, input2, score. Where input1 and input2 are indexed for faster searches. The table can operate in two modes:

1. Causal: It stores and searches inputs in order so that we assume that input1 always causes input2.
2. Correlational: It does not assume any causation between input1 and input2, and it just stores and searches using both inputs.

A very important thing to keep in mind is that it is really difficult to judge the power of the system with a small number of associations stored in it just as it is difficult to judge the intelligence of a new born baby or an infant. It is also important to keep in mind that the rules governing how the STM and LTM behave are merely heuristics that we write according to what we know about how humans function. These can always be changed and modified to more align with scientific data in the future.