A Conceptual Model for Process Mining Based on Hierarchical Temporal Memory Plug-in

L.U. Oghenekaro
Department of Computer Science
University of Port Harcourt,
Rivers State, Nigeria.
linda.oghenekaro@uniport.edu,ng

M.O. Musa
Department of Computer Science
University of Port Harcourt,
Rivers State, Nigeria.
martha.musa@uniport.edu,ng

Abstract

Most organizations only have knowledge of what ideally should happen in their working environment but have little or no details about what actually happens. This disconnect between an ideal process structure and the actual one gives rise to hidden inefficiencies and deviations that result in a significant increase in process cost for organizations and a decrease in business profit. In the pool of immeasurable amount of processes, lie some processes that do more harm than good to organizations; such processes can be referred to as inefficient, as they do not add value to the overall system. In the bid to manage the pool of immeasurable amount of business processes and improve their efficiency, this paper proposes a conceptual model that highlights and harmonizes several intelligent machine learning concepts to learn and classify business processes, using the object-oriented software analysis and design methodology (OOADM). The model described the interconnection between the several modules of the model, using supporting algorithms, which further demonstrated the logical flow of the system and strength to handle sparse representation of input vector. This conceptual model allows simulation of different system design.

Keywords: Process mining, business processes, hyper parameter, event log, hierarchical temporal memory (HTM)

1. Introduction

The HTM theory is an online machine learning concept developed by Hawkins J. and George D. of Numenta. The model copies some of the algorithmic and structural properties of the neocortex. In the book *On Intelligence*, (Hawkins & Blakeslee, 2004), Hawkins developed the idea to build a simple model of the neocortex not by simulating every part of it but by reducing it to its core function (Galetzka, 2014). The previous challenges faced in every algorithm trying to model a part of the brain is the complexity of the brain in terms of physiological and anatomical details, hence it is hard to determine exactly what is required in allowing proper functioning in a living being. Due to this difficulty, the derived models often lack some basic biological features such as feedback connection, and even those that have this feature have trouble with values varying over time, hence, they are used for only static problems such as classification (Chappelier et. al, 2001). In Hawkins' opinion, the neocortex which holds intelligence in the human brain is responsible for memorizing and making predictions based on those memories. Thus, he named his model Memory Prediction Framework. The model works thus (Galetzka, 2014):

1. The model characterizes the part of the world it is expose to by learning patterns which it uses to make predictions. These patterns can occur both spatially (as a group of activities often occurring together) and temporally (as a chain of events often occurring in the same order).
2. It uses the notion of hierarchy to allow patterns stored in a lower level to be reused by higher levels on the hierarchy.
3. Information flows up and down the hierarchy to create certainty between different potential patterns formed and the model learns in a reinforced manner by integrating the actual observed event into the stored patterns.

Although, HTM was inspired by established approaches like artificial neural networks (ANN) and Bayesian networks (BN), it however, tried to be consistent with the underlying biological model by modeling its architecture with reference to the neocortex of the human brain (Numenta, 2011). The artificial neurons used in most neural networks do not capture the complexity or the processing power of the real neurons (Hawkins & Blakeslee, 2004); however, the cells used in the HTM learning algorithms are more realistic in representing the real neurons of the human brain. With the increasing availability of streaming data in this era of big data, there is an increasing demand for online sequence learning algorithms, these set of algorithms should be to address the peculiarity of the nature of data such as:

a. Continuous online learning
b. High order prediction
c. Noise robustness and fault tolerance
d. Little or no hyper parameter tuning

Process mining, though young in discipline, has proven to be very relevant across multiple domains and as research grows in this area of process mining, so has the need to improve the approaches involved in mining processes. Process mining applications capture several voluminous activities otherwise known as event logs that can go to waste if not properly mined; the most relevant application of process mining is in the areas of healthcare, software development and services, manufacturing area, education, government parastatal, security services, call centers, metropolitan services, entertainment, sport, hotels, agriculture and smart buildings. The Research and Market published in their Process Analytics Market report in September 2018 that global process analytics market size is expected to grow from 185.3 million dollars in 2018 to 1,421.7 million dollars by 2023, at a Compound Annual Growth Rate (CAGR) of 50.3% as a result of the implementation of digital transformation in driving process mining. Process mining is then envisaged to be a very profitable market domain in the business world in the nearest future due to its vital role in business processes. Premier market research intelligent firm, IDC, also recorded in 2016 that companies lose 20 to 30 percent in income every year due to inefficient processes.

2. Related Literature

Process mining is the family of techniques in the field of process management whose basic idea is to improve process performance and to create business value using event logs (Jan and Geert, 2014). The concept of process mining was first introduced by Agrawal et. al and Cook and Wolf (Baykagsolu et al., 2018). Their pioneering studies on workflow management and machine learning led to the evolution of a new research area called process mining. In carrying out process mining, specialized data mining techniques are applied to the event log data to identify patterns and details that exists therein. Process mining is also known as Automated Business Process Discovery (ABPD) (Yurek, et. al., 2018). However, in academic literature, ABPD is applied in a narrower sense to specifically refer to techniques that accepts event logs and input to produce business process model as output. Process mining refers not only to techniques used in discovery process model but also to techniques that enhance existing process model and techniques that ensures conformance of present models. Gunther and van der Aalst (2007) developed the fuzzy miner algorithm to handle the inherent unstructured characteristics of the healthcare processes. The strategy provided imagery through the theoretical expectation of reality and represented the most relevant of activities. Leoni, et. al. (2016) also worked on event log of a Brazilian software company generated over five years with about 20,000 process cases. Markov Chain was implemented on ProM to carry out sequence clustering analysis. A recent study was done by Suriadi, et. al. (2017), in their work, process mining was integrated with a Bayesian network of predictive models for determining maintenance intervals for industrial equipment. The work further served as a tool for support managers in decision making on maintenance time, reduction of downtime and avoidance of unplanned stops (Gracia, et. al., 2019). Dongen et. al. (2005) proposed a probabilistic process model (PPM) using Markov techniques and the research was applied to an insurance company with the aim of simulating insurance claims. The system resulted in more accurate predictions as compared to the conventional conditional probability analysis. Conforti et. al. (2015) developed a plug in to equip a recommender system that focused on minimizing overall risk at process instances. The improved system enabled the users accept risk while they are aware of the impact and they also had access to other alternatives. Sutrisnowati, et. al. (2015) applied process mining as a stage for the Bayesian networks construction using container handling activities. The system was expected to analyze probability of delays in a port management system.

3. Theoretical Framework

Several concepts were adopted from the hierarchical temporal theorem to build a robust conceptual model for mining and classifying business processes.

3.1 Semantic Encoding Concept

The semantic encoding concept is very similar to embedding in the deep learning space. An encoding engine takes input from an input source and create an SDR. The encoding algorithm is developed to produce similar SDR for similar objects. A lot of pre-built encoders are already available online that include numeric encoding, date-time encoding, English word encoding, etc. However, customized encoders can be developed for specific problems. The HTM uses the semantic encoding to pinpoint anomalies in data sequence. Encoders are developed using the following steps:

Step 1 – Choose a set of documents to be used to find semantics of words

Step 2 – Clean the documents and slice each document into snippets.
Step 3 – Cluster snippets such that similar snippets are kept together
Step 4 – Each snippet can then be represented as a node in a SDR
Step 5 – Individual (target) words are selected and all the nodes activated in the SDR that contain the target word. Hence creating a word fingerprint in form of SDR
Step 6 – Step 1 to step 5 is iterated to get word fingerprints for all the words of interest.

3.2 Sparse Distributed Representation Concept

The sparse distributed representation (SDR) is simply an array of 0's and 1's. A snapshot of neurons in the brain shows a high likelihood that only less than 2% neurons are in active state. The SDR is a mathematical representation of these sparse signals which will likely have less than 2% as 1's. The SDR is the form which the brain encodes information by representing information with a minute percentage of vigorous neurons from a large population of neurons, as seen in figure 1, this form of encoding is also adopted by HTM regions. When there is an input, the HTM lower region first and foremost, converts the input into a SDR. The input, which could be either sensory data or data from a lower region on the hierarchy, consists of thousands of bits. These bits are either 1or 0 depicting active or inactive respectively. If a sensory data is received with 55% of its bits "on", the algorithm can convert it to a new representation with a simple 3% of its bits "on". It does this by ensuring that each column in a region is connected to a unique subset of the input bits, hence, different input patterns results in different levels of activation of the columns. After which the columns with the strongest activation deactivates the columns with weaker activations, the inhibition function carries out this task, the function is also defined to achieve a relatively constant percentage of columns to be active even when the number of active input bits varies largely. The SDR is noise resistant and can be sub-sampled without losing much information.

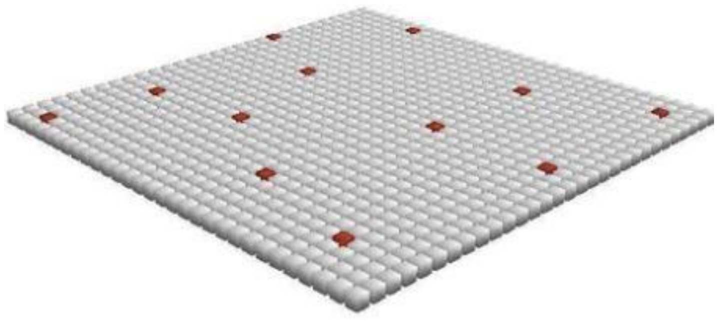

Figure 1: Sparse distributed representation of a HTM layer

3.3 Spatial Pooling (SP) Concept

The processing that takes place in the spatial pooling phase can be described using some mathematical notations. The input vector sizes, along with the positive bits are considered to be primary parameters in this stage since they defined the sparsity of the input vector. The number of columns and the number of synapses directly affects the shape of the SP architecture and its processing capabilities.
The sparse distribution space of the spatial pooler is defined as

$$S : \{ 0 , 1 \}^{1\times n} \qquad (1)$$

In process mining, the vectors in the SDR may be interpreted as the wining activities within the event logs. These vectors can be brought together by an encoding operation, binarization, as illustrated in equation 2.

$$d_s : S \times S \longrightarrow N \qquad (2)$$

where S represents the sparse distributed representation
d_s represents the distant operator for the input vector

3.4 Hebbian Learning Concept

Learning in HTM is based on a very simple principle. The synapse between the active column in the spatially pooled output array, and active cells in encoded sequence, is strengthened. The synapses between the active column in the spatially pooled output array, and inactive cells in encoded input, is weakened. This process is repeated again and again to learn patterns. All active columns apply the generalized Hebbian like learning rule to strengthen synapses that are aligned with active input and weaken those aligned with inactive input (Price, 2011). The permanence values associated with the synapses are also modified using Hebbian rule. The Hebbian rule originates from the hypothesis made by Donald Hebb on the way in which synaptic strengths present in the brain are adjusted in response to experience.

3.5 Boosting Concept

Most of the spatial pooling processes will create exceptionally strong columns in the output array which will suppress many columns from contributing at all. In such cases, we can multiply the strength of these weak columns to encoded sequence by a boosting factor. This process of boosting makes sure that we are using a high capacity of the spatially pooled output.

3.6 Temporal Memory Concept

Spatial pooling maintains the context of the input sequence by a method called temporal memory. The concept of temporal memory is based on the fact that each neuron not only gets information from lower level neurons, but also gets contextual information from neurons at the same level. In the spatial pooling section, we had shown each column in the output vector by a single number. However, each column in the output column is comprised of multiple cells that can individually be in active, inactive, or predictive state.
This mechanism might be a bit complex, so let’s go back to our example. Instead of a single number per column in the spatial pooling step, I will now show all cells in the columns of the output vector.

4. The Process Mining Conceptual Framework

There are basic concepts applied to HTM algorithms that should be understood, they are explained below

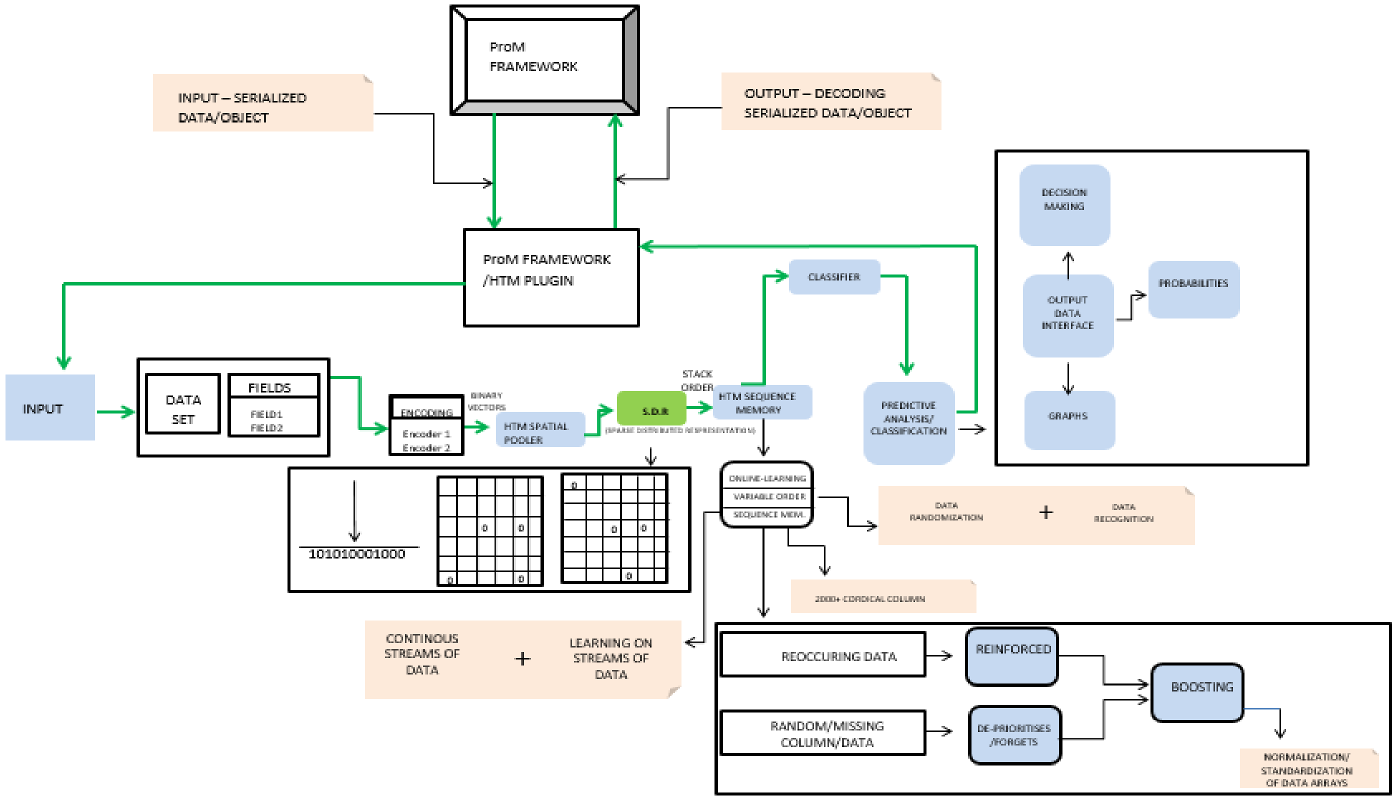

Figure 2: Conceptual Model of the HTM plug-in in the ProM Environment

Conceptual model that harmonizes five different concepts of hebbian theory, sdr, semantic coding, is shown in figure 1. The data object enters the model through the ProM framework and passes through the module were the framework interfaces with the HTM plug-in, both for data input and data output. Present in the HTM plug-in are the libraries that are implemented to communicate with the ProM framework for input and output dataset. The input data are fed into the data fields based on data types, the sectioning of data into data fields is done for the purpose of encoding based on encoders for specified data sets. The data is encoded into binary vectors and sent to HTM Spatial Pooler, which then converts arbitrary binary input patterns into sparse distributed representations (SDRs) using a combination of competitive Hebbian learning rules and homeostatic excitability control. The SDR presents a type of data representation that the HTM sequence memory can work with, and they are sent in sets of prioritized data, which is represented in stack orders; the old sets of data being compared with new sets of data, and re-occurring data sets is prioritized as they are sent per time. The SDR is presented as an array consisting of a large number of bits where a small percentage are 1's and the rest are 0's. The bits of SDRs correspond to neurons in the brain, where 1 represents an active neuron and 0 an inactive one. The stack order flows into the sequence memory algorithm where various activities take place such as; the variable order, which is the ability of HTM learning and recognizing a data even with its variableness relative to its rate of occurrence, and the sequence memory which is the core of the system, and this is where the learning happens. At the sequence memory stage patterns are created for data recognition, with the reinforcement of re-occurring data and de-prioritization of missing or random data, further with statistical analysis of the data-normalization and standardization of data arrays. The final activity that happens in the sequence memory module is the online learning where learning of continuous streams of data is done in real time. The output of the sequence memory flows as a sparse distributed representation into the classifier for classification, and data of same rate of occurrence is first prioritized for classification, after which data of same type is then classified within the same classification for output, with various sets of classification to this effect. Output expressions based on various sets of classification. This is sent through the libraries that the ProM framework/ HTM plugin implements to communicate with the ProM framework for input and output, hence giving desired output in a relatable interface on the ProM framework via the output data interface, where various forms of classified data to be sent to the ProM/ HTM Plugin for output representation.

5. Implementation

The htm.java's network API was adopted for the implementation of the conceptual model, and the eclipse IDE was used as the development environment. Input temporal data generated from various data sources is semantically encoded as a sparse array called as sparse distributed representation (SDR). This encoded array goes through a processing called spatial pooling to normalize/standardize the input data from various sources into a sparse output vector or mini-columns (column of pyramidal neurons) of definitive size and fixed sparsity. The learning of this spatial pooling is done through Hebbian learning with boosting of prolonged inactive cells. The spatial pooling retains the context of the input data by an algorithm called temporal memory.

Default parameters of input_dimension, column_dimension and cells_per_column are set using the getParameters() method. The specific parameters of global_inhibition, stimulus_threshold, duty_cycle, max_boost and overlap_duty_cycle are set using the parameters.set() method. The scalar encoder was configured to build the input by setting the field values for the; clipInput, forced, encoder, maxVal, minVal, n, name, radius, periodic, w and resolution. Objects for every class used were instantiated for line 1 to line 4, as follows:

```
ScalarEncoder encoder = dayBuilder.build();
SpatialPooler sp = new SpatialPooler();
TemporalMemory tm = new TemporalMemory();
CLAClassifier classifier = new CLAClassifier(new TIntArrayList(new int[] { 1 }), 0.1, 0.3, 0);
Layer<Double> layer = getLayer(params, encoder, sp, tm, classifier);
```

However, in line 5 the classifier objects, params, encoder, sp, tm and classifier objects, are passed as parameters to the get layer method, which creates a layer. A for loop is created to programmatically generate dataset rather than import a csv file.

```
for(double i = 0, x = 0, j = 0;j < 1500;j = (i == 6 ? j + 1: j), i = (i == 6 ? 0 : i + 1), x++) {
    if (i == 0 && isResetting) {
        System.out.println("reset:");
        tm.reset(layer.getMemory());
    }
    runThroughLayer(layer, i + 1, isResetting ? (int)i : (int)x, (int)x);
  }
}
```

In the For loop that dynamically generates the input, the runThroughLayer() method is called, which takes parameter as the layer and the respective fields generated is passed into it.

Here is the For loop method below, we can see that it calls the input() method on the Layer l object passed as the first parameter layer Layer<T> l.

The input() method which perform the learning and generate output for runThroughLayer()

6. Conclusion

A novel approach for representing processes using several concepts within the hierarchical temporal memory theory was developed, and the peculiar nature of real-life processes was put into consideration, which further made the model robust in classifying business processes. The conceptual model developed adopted the hierarchical temporal memory learning algorithms; the spatial pooler and the temporal memory algorithms in implementing the concepts. However, the model, just like every other process mining model can only apply to businesses that run on process aware information systems (PAIS), as the event logs present in these systems serve as the raw materials for process mining models.

References

1. Baykasoğlu, A., B.K. Ozbel, N. Dudakli, K. Subulan, M.E. Senol (2018). Process mining based approach to performance evaluation in computer-aided examinations. *Computer Application Engineering Education*. 26(5): 1841–1861.
2. Chappelier, J., Gori, M. and Grumbach, A. (2001). Time in Connectionist models. *In Sequence Learning, Springer.* 105-134.
3. Conforti, R., M. de Leoni, M. la Rosa and W.M.P van der Aalst (2014) A Recommendation System for Predicting Risks across Multiple Business Process Instances. *Decision Support Systems.* 27 – 36.
4. Dongen, B.F. and W.M.P van der Aalst (2005) A Meta model for process mining data. *In: 17th Conference on Advanced Information Systems Engineering (EMOI-INTEROP Workshop); Porto, Portugal*.309-320.
5. Galetzka, M. (2014). *Intelligent Prediction: An empirical study of the Cortical Learning Algorithm.* Master Thesis, University of Applied Sciences Mannheim.
6. Garcia, C.S., A. Meincheim, E.F.J. Ribeiro, M.R. Dallagassa, D.M.V. Sato, D.R. Carvalho, E.A.P. Santos, E.E. Scalabrin (2019). Process mining techniques and applications –A systematic mapping study. *Expert Systems with Applications.* 133(1): 260-295.

7. Gunther, C.W. and W.M.P. van der Aalst (2007). Fuzzy Mining: Adaptive process simplification based on multi-perspective metrics. *In: 5th International Conference on Business Process Management. Brisbane, Australia. Heidelberg, Germany: Springer.* 328-343.
8. Hawkins, J. and S. Blakeslee (2004). *On Intelligence.* Henry Holt and Company.
9. Jan, C. and P. Geert (2014). Merging Event Logs for Process Mining: A Rule Based Merging Method and Rule Suggestion algorithm. *Ghent University, Department of Business Informatics and Operations Management, 9000 Gent, Belgium.*
10. Leoni, M., W.M.P. van der Aalst, M. Dees (2016). A general process mining framework for correlating, predicting and clustering dynamic behavior based on event logs. *Information Systems.*56(8): 235-257.
11. Numenta (2011). Hierarchical Temporal Memory. URL: http://numenta.org/cla-whitepaper.html
12. Suriadi, S., R. Andrewsa, A.H.M Hofstedea, M.T Wynna. (2017). Event log imperfection patterns for process mining: Towards a systematic approach to cleaning event logs. *Information Systems.* 64(4). 1532-1564.
13. Sutrisnowati, R. A., H. Bae, L. Dongha, K. Minsoo(2014). Process Model Discovery based on ActivityLifespan. *International Conference on Technology Innovation and Industrial Management Seoul*. 137-156.
14. Yurek, I., B. Derya, and U.B. Kokten (2018). Interactive process miner: A new approach for process mining. *Turkish Journal of Electrical Engineering & Computer Sciences.*