DynPR: Dynamic Evaluation of Patterns Across Data Streams Via Deployed Applications

Anonymous CVPR submission

Paper ID 16504

Abstract

Though Various Solutions have been proposed regarding real-time data processing, user-defined fine-tuning, and Repository-based Hosting, Artificial Intelligence still grapples with unsynchronized data processing, thread-safe data storage, and isolation of numeric metrics from non-numeric ones. This poses a new question: How to amplify stream processing for AI to work in any applications? In this work, We implemented Specialized Regular Expression Pipelines that enable effective attribute retrievals with domain-specific metadata attachments for application use. Its novelty lies in how space-& time- expensive Regular Expressions are used to extract required data from large data streams delivered at a rate of 8000 data/minute via IoT systems. We demonstrated an appreciable increase in data extraction precision by 65% and the solution promised an accuracy increase of 80% in a star-topology kind of network. This model sets a new paradigm in IoT-driven AI Computing, effectively circumventing the processing bottlenecks of its predecessors.

1. Introduction

The Internet of Things is a primary source of large data repositories since they host primary and secondary sensors that receive, process, and transmit differential electromagnetic signals. A signal is a contiguous information flow with infinitesimally small intra-signal intervals that need careful deciphering & segregating useful and noisy data. The main relevance of signal in Artificial Intelligence is important when we apply its applications in inter-network and intra-network hardware applications. It is primarily attributed to the systems where there is a need for non-software elements like cameras, microphones, etc., and especially the systems that mandate tactical simulations. The tactical systems employ physical aspects to drive the understanding of human senses. Human Senses, like cognitions, are generally unexplainable requiring advanced calculations to carefully condense and process data. This necessitates effective data processing pipelines.

Existing Systems focused on conventional techniques to apply statistical and probabilistic techniques for segregating data. While graphical methods are efficient in isolating outliers, the techniques fail in densely located data with relatively smaller margins among them. Many libraries don't synchronize well with the pace and volume of data which require Big Data Solutions. As Big Data is expensive in implementation and deployment, the current solution can efficiently scale in a cost-effective. Additionally, it is found to reduce the development time of its predecessors by 18%.

The novelty of the work lies in how it amplifies Regular Expression Pipelines to monitor the input data for identifying the patterns and decode the inter- & intra-pattern relations to infer clusters within the data stream. The solution makes use of time-bound data streams which are handled in the background concurrently with the main data acquisition systems while simultaneously synchronizing the operations. Data Processing in Java is a rare research statement and we explored the language's libraries & packages to diversify our processing approach, further adding to its novelty.

Note that Regular Expression is abbreviated as RE throughout the paper. The work entails the Real-time engineering of data streams as time-bound probabilistic systems that employ time as metadata. Temporal Specifications differentiate nominal and ordinal states of the received information in novel ways while focusing on the linguistic semantics of the received data. Differential aspects of the Current Work include:

1. The Regular Expression Pipeline is a novel paradigm that uses common regular expressions to match patterns. This is demonstrated in the figure 1.
2. The matcher that we used is an entity as opposed to the previous works where it is a system.
3. We removed noise efficiently using simple wildcards in RE Pipelines.

2. Existing Systems

Pattern Recognition is a rarely explored direction in the field because it is assumed to be implied by a stream inherently. The classes of studies include the following:
Statistical Methods: Central Means of Tendency[1-3] is exploited where inter-point and intra-cluster distances are measured to estimate cluster memberships. This type of clustering involves two kinds: centroids from existing clusters and clusters from centroids. Since this procedure is tedious, few works[4] utilize skip distance methods that delist a few points for time and space efficiencies. The more efficient version is to use a model that learns from these statistics and makes hypotheses that generalize to the whole stream as specified by different works[5].

Non-Statistical Methods: These existing work entails the following subsections:
Domain-Specific Methods: This work details multidisciplinary data stream pattern recognition methods[6-10] where streams from healthcare, biotechnology, electrical engineering, etc., are considered. Domain-specific equations or relations are considered which can identify patterns that are closer scientifically than numerical affinities.[11-13]
Iterative Methods: The Same method is applied over and over the same data stream to generalize better than other models[14-18]. The frequent applications ensure error reductions and optimal performance. But frequent applications add a performance overhead to the stream pattern recognition due to large rates and volumes[19] which needs to be addressed. **Empirical Methods:** Theoretical studies focus more on how patterns can be recognized in the data streams over cycles. This system is similar to an iterative process but differs in the way the study is conducted. Theories[20-22] direct the stream pattern recognition to be more rational towards statistical centroid-based methods with less focus on mixed streams. Few works[23] cite the noise reduction during pattern recognition in streams. They quoted the use of data processing pipelines for the purposes but these pipelines fail for the noisy data having greater affinities for the normal flow.

Pattern Mining Techniques: This is a bulk process of all the existing systems where multi-pattern detection is enabled, further followed by some intelligence attributes of the developed model[24-25]. Ontology is also deployed in certain systems where the systems learn to decipher the ways of simply extracting multiple patterns across diverse data streams. The more advanced approach is to employ real-time pattern recognition that works as data processing pipelines rather than timed functions demonstrated in our study.

3. Methodology

Tweaking Pattern Recognition issues within a data stream entails a systematic evaluation of synchronous faults, processing issues, and improper hosting of data. While this process incorporates a point-to-point estimation of all the elements which is both time- and resource-intensive, we have employed a custom pipeline that conducts cross-data examination and data extraction based on identified patterns within a single elementary operation. The table 1 is a master table that we have frequently consulted for our diverse experimentations, employing a centralized approach. The Research Methodology details the following attributes for its analysis:

3.1. Data Interpretations

The Regular Expression Pipeline that we have employed includes typical evaluations of semantic expression of linguistics within the stream, effectively detailing every character received in the stream. This pipeline can handle any kind of data that ranges from Numeric to Character-based elements while also including Noisy data. Multimedia Streams still require deeper analytics which is out of scope for the current study. The Regular Expression Pipeline has been designed to address the following stereotypes of data:

3.1.1. Numeric

Isolation of numerical data from a totality of stream requires unique identification of digits and decimal points. The interesting point to note here is that the digits may not always be characterized by simple decimals 0-9 but patterns containing groups of numerical bits with their ranges amounting to even BigNumber as defined in Java with magnitude equalling 12345678901234567890 may also be pursued. The probability equation for such isolation has some interesting consequences.

3.1.2. Character

ASCII Interpretations of Character tokens in a total characteristic stream are complex since they involve certain convolutions based on application types. While a few applications require direct ASCII interpretations, some applications require tokenization of other encodings as defined in meta-directories or supplementary directories embedded within the application root folders. This directs the applications to route through a system for data extractions with the following numerical correlations.

3.1.3. Noise and Garbage Values

Noise seems synonymous with Garbage Values but it requires some interpretations that need sophisticated mechanisms detailed in the table 1. Garbage Value differs from noisy data by the kind of its characters that may range from direct unintended textual uses to entirely different characters. The problem with this approach is that we may tend

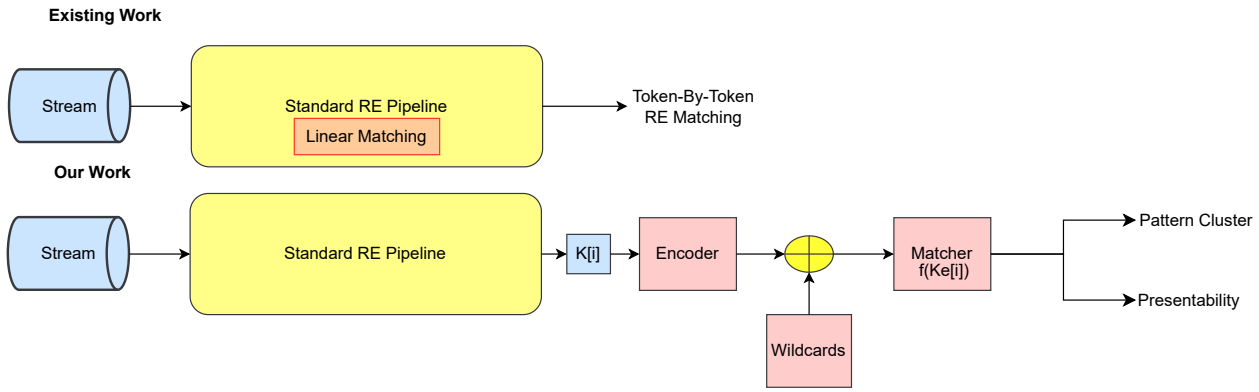

Figure 1. The figure shows the difference between our Regular Expression Pipeline and the Existing Regular Expression Methods

Char	Wildcard	K[i]	K[i+1]	Comp.	Proc.	Segm.	Conn.	P1	P2	P_{rsltnt}	locat	Density
	*	[0-9]	[a-z]	[0-9]	ASCII	Worker	Valid	[0-9]	[a-z]	[0-9]	5	5.08
	+	[a-z]	[0-9]	[a-z]	ASCII	Worker	Valid	[0-9]	[a-z]	[0-9]	5	5.08
[0-9]	*	[0-9]	[0-9]	[a-z]	ASCII	Core	Valid	[0-9]	[a-z]	[0-9]	7	7.08
	+	[a-z]	[a-z]	[a-z]	NA	Worker	Valid	[a-z]	[a-z]	[a-z]	15	32.98
	+	[@$#]	Given	Given	Filter	Core	Valid	[$#]	Given	Given	66	75.08
	*	[0-9]	[a-z]	[0-9]	ASCII	Core	Valid	[0-9]	[a-z]	[0-9]	4	9.29
	+	[a-z]	[0-9]	[a-z]	ASCII	Core	Valid	[0-9]	[a-z]	[a-z]	5	8.92
[a-z]	*	[0-9]	[0-9]	[a-z]	ASCII	Core	Valid	[0-9]	[a-z]	[0-9]	10	10.08
	+	[a-z]	[a-z]	[a-z]	NAN	Worker	NA	[a-z]	[a-z]	[a-z]	6	4.78
	+	[@#$]	Given	Given	Filter	Core	Valid	[@#$]	Given	[@#$]	15	6.88
	*	[0-9]	[a-z]	[a-z]	ASCII	Core	Worker	[0-9]	[a-z]	[0-9]	25	0.53
	+	[a-z]	[0-9]	[a-z]	Filter	Core	Worker	[0-9]	[a-z]	[0-9]	35	1.18
[@#$%]	*	[0-9]	[0-9]	[a-z]	ASCII	Core	Worker	[0-9]	[0-9]	[0-9]	47	47.08
	+	[a-z]	[a-z]	[a-z]	NAN	Worker	NA	[a-z]	[a-z]	Given	10	10.08

Table 1. Master Table: We consulted this repeatedly throughout our work to maintain a uniform centralized reference system. The char considers the main pattern recognized, wildcard is the RE Pattern finder while K[I], and K[i+1] are stream tokens. Comp. is the computed output where proc. is the process employed in the RE Pipeline. P1, P2 are competing patterns, and P_{rstlnt} is the resultant pattern. Location and Density are graphical properties.

to accept what is otherwise useful data. The boundary of distinction is too thin to differentiate between such data elements and the actual data elements.

3.2. Concurrency Investigations

It is important to note the reasoning for the waiving of monolithic architectures across modern applications. Recent applications have been implementing multi-layer software processes with microprocesses dominating its functionalities. The framework employs synchronized access to data with simultaneous updation from various endpoints. The current investigations have been performed in trials over a network connected in the form of a topology which requires the following concurrencies as demonstrated in the figure 2:

3.2.1. Worker Segment

The main execution thread is where the UI-based ramifications of pattern recognitions occur which are available to the user visibly. Visible pattern recognition is the process executed in this phase, which involves important and presentable data investigations.

- **Important Data Isolation:** 87% of data streams contain noisy & irrelevant data which needs effective filtering. While functional filtering applies mathematical functions on the input stream, pattern recognition is more efficient since noise doesn't follow protocols to qualify it as a function parameter. Important data also follows metadata provided by the application designer/user. This metadata is used in customary pipelines where pre-defined methods

are not sufficient. Regular Expression Pipelines are hence suitable for implementing custom logic as patterns.

- **Presentable Data Investigation:** First-order Applications of RE Pipelines don't measure the suitability of the data to the application worker segment. Due to the potential voids and visual consistencies, the User- and Designer-specified data rubrics may not fit well into the main App Structure. Equation 1 details the numerical aspects.

$$\begin{aligned} presentability = & \frac{pixel_{data}}{pixel_{whole}} \\ & + B' \frac{pixel_{whole}}{pixel_{data}} \\ & + (B + B') f(pixel_{data} * pixel_{whole}) \end{aligned} \tag{1}$$

where $pixel_{data}$ and $pixel_{whole}$ is the number of pixels with data points and remaining pixels respectively, B' is compensating coefficient that offsets possible errors due to the first, B is biased coefficient, and * is convolution.

3.2.2. Core Segment

This segment handles the bulk of the total data processing pipelines where total noise removal, final data extraction, and data hosting take place. Clusters are assigned in the same segment. It entails the following processes:

- **Pattern Synthesis:** Linear Approximation of Approaches to stream processing stands as an inescapable expensive procedure in real-time pattern recognition. While RE Pipelines are quite stringent in their time requirements, they demonstrate appreciable optimization in pattern finding. The pattern evaluations for a stream include the following relations 2 & 3.

$$P1 = Full\_Conv(Conv(Pool(stream))) + b_{conv} \tag{2}$$

where P1 is the first pattern recognized, Full_Conv is the fully convolution function, Conv is the convolution function, Pool is dimensionality reduction function, b_{conv} is convolution bias and stream is the input stream variable.

$$P2 = Full\_Conv(Conv(Pool(P1))) * P1 \tag{3}$$

where P2 is the competing pattern recognized, Full_Conv is the fully convolution function, Conv is the convolution function, Pool is dimensionality reduction function, and P1 is the first pattern.

- **Pattern Clustering:** Though pattern-based clustering is an easy task, Stream Analytics involves one-many comparisons with 'many' constituting millions of data elements. Heapification is preferred as a standard procedure for many applications but in Dynamic Data Stream Pattern Recognition Systems, Heapification is hugely space extensive necessitating linear analytic flows. The steps are detailed in the next section.

3.2.3. Validation Segment

The bulk of Pattern-Matching is performed in the Core Segment but visualization requires an interface between UI and Backend Processes which needs presentable data investigations. These investigations are different from the Worker Segment's as:

- **Library Support:** Not All Libraries support the current mechanism implemented in the worker thread. Particularly, thread-safe procedures in the worker thread are inefficient enough to accommodate all the libraries necessitating a separate segment.
- **Temporary Recognition Requirements:** A data stream may not contain all useful data and some portions may be unnecessary. Particularly, During Hardware and IoT faults, the Worker Thread fails to incorporate temporary processes, thus wasting resources. This segment is useful for applications with custom data settings.

3.3. Regular Expression Pipeline

Regular Expression Pipeline is a systematic schema that originates at Pattern Identification and terminates at Presentable Data Identification. The Mathematics of the study arises here where we can maintain correlations between parameters and RE Pipeline Speeds. We denote the subset sizes for which the pipeline is considered as K. The whole backend of the system is tabulated in the table 1.

3.3.1. Pattern Identification

- **Competent Pattern Analysis:** If in a K substream, two patterns P1 and P2 may be considered equivalently possible, the pattern recognition inclines towards a pattern of dominance as per theoretical intuition. This may not be observed in practical scenarios due to instantaneous dominance variations across periodic volumes with period T. The Competent Analysis as mentioned in the equation 4 decides the winning pattern. The winning pattern should also obey the logics of the figure 4.

$$\begin{aligned} P_{rsltnt} = & P1 : P2? \\ & ((P(P1) * P(P2)) || (P(P2) * P(P1)) \\ & || (P(P1(P2)) * P(P2)) || (P(P2(P1)) * P(P1))) \end{aligned} \tag{4}$$

where P1 and P2 are competing patterns, P(x) is probability of dominance of x, || is boolean or, * is convolution function, ?: is teritiary decision operator, and P_{rsltnt} is the winning pattern.

- **Similarity Pattern Analysis:** When we compute the Manhattan distance among the streams, noisy data are often confused with nominal data elements. Competing Patterns P1 and P2 of different stereotypes-numeric, denoted as N, and character, denoted as C may be considered for this. Infinitesimal Margins between Noisy Data

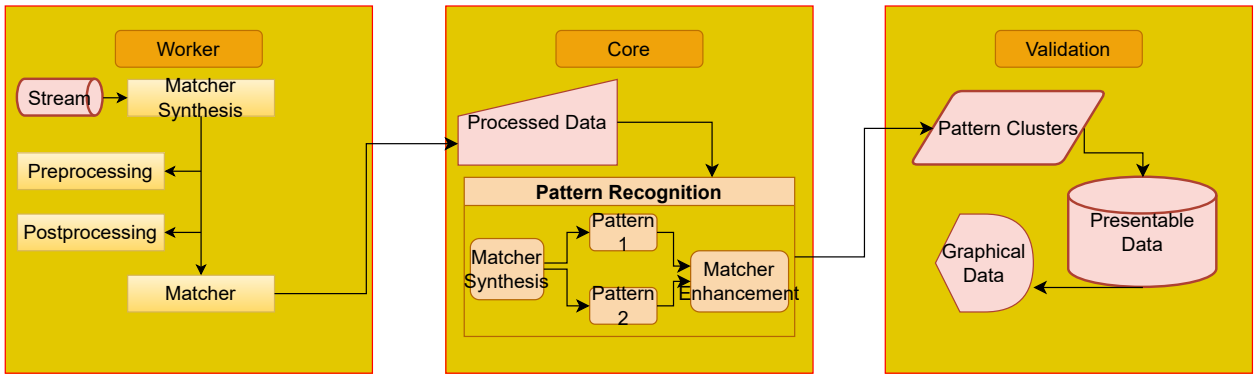

Figure 2. This figure shows the interactions and data processing among segments

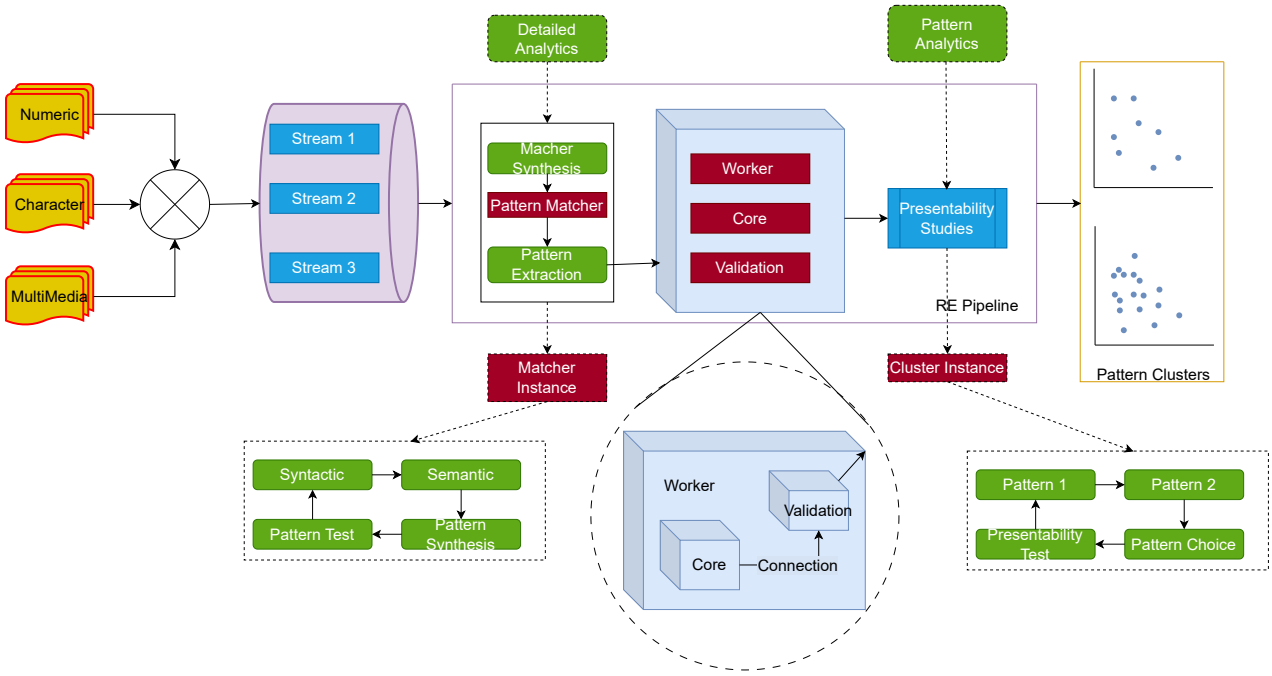

Figure 3. The streams can originate from multiple file documents fed directly into the RE Pipeline. The dotted projections shown are implied pre- and post-processes for each stage in the RE Pipeline. Every instance created undergoes phases of synthesis& testing. The segment relations are highlighted in blue. The graphs presented at the end are highly representative and vary largely in scale from the original.

and Nominal Data may perturb the normal Pattern Recognition via deception and these relations are detailed in the equation 5.

$$Score = \frac{P1.P2}{|P2|} * \frac{P1.P2}{|P1|} * dot(P1, P2, b_{P1}, b_{P2}) \tag{5}$$

where P1 and P2 are competing patterns, Score is the similarity score, dot is a function that returns the permutations of dot products of given parameters, b_{P1} is pattern 1 bias while b_{P2} is pattern 2 bias.

- **Filter Technique** Employment of Conventional Filters is not found efficient in the current and its application has been completely discarded.

3.3.2. Matcher System

The Matcher System primarily linearly parses the input data to lexically encode, decode, and match the data with a list of competent matches in the pipeline. A Matcher System is a mathematical equation/modeling between the input stream and output matches.

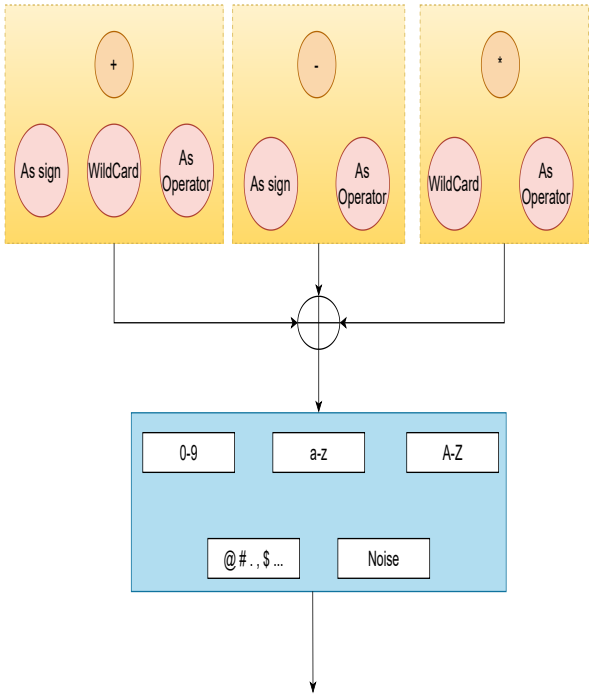

Figure 4. We investigated the differences between wildcard patterns and numerical symbols, highlighted in yellow. The matcher system that we generated, highlighted in blue, differentiates based on inherent automated patterns and produces the output(not shown here)

- **Numeric Matching** The stream may be segmented into numerical subsets of data with each cluster possessing common types and properties while also maintaining metadata of matcher information in their headers. The matching is probabilistic, P(K[I], K[i+1]) where P(x)¿0 for a perfect match. The role of $P(x)$¡0 is to never be estimated since some subtle overlapping may be neglected. Hence, a custom bias b_{subtle} should be added resulting in the relation $P(K[I], K[i+1]) + b_{subtle}$.
- **Character Matching** Characters have an additional overhead of ASCII-based encoding where ASCII Equivalents may dominate the equivalent effect of bias additions in the case of Numeric Matching. When K>50, the matching process becomes tedious which overwhelms the overall system. For 50<=K<100, the bias is insignificant but it becomes significant in another case. We observed different bias constants for this application, terming it as $b_{charSubtle}$. The numerical is discussed in the results section.

3.3.3. Finder System

This refers to extraction in the pipeline with the following types of operations:

- **Matched Extraction:** Once a match has been found, the data will be extracted with a probability M(P(K[I]))> 0.725 but M(P(K[I]))>0.92 may be neglected on account of larger magnitudes. The probabilistic determination includes f(P(K[I]), P(K[i+1])) with an additional bias constant $b_{matched}$.
- **Suspected Match Extraction:** Inferior Probabilities should never be underestimated with their ranges in between M(P(K[I]))<0.357 and M(P(K[I]))>0.112. Supplemental Range includes but is not limited to (0.100,0.445). This allows for minor adjustments in error margins to suit more suspected data extractions. Suspected Data is the data that has M(P'(K(I)) 0.563 where P'(K(I)) is the probability of K(I) to be noisy.

3.3.4. Presentability Investigation

Data is presentable if it can be extracted, quantized, normalized, and standardized to a prescribed data range. Further elucidations are required in terms of y-axis values for x-axis inputs to accommodate clusters and clean them as demonstrated by the far-right of the figure 3. The investigation artifacts include the following attributes:

- **Location Specifics of Data:** Inter-point distance along X and Y attributes cannot cross the X axis, implying the data is presentable in a single axis. This type of Pattern Recognition is useful for non-scientific applications. Considering X() and Y() as functions of X and Y attributes for a pattern, the following relations disobey these observations: sgm(X(K[I]), Y(K[I])<0 and sgm(Y(K[I]), Y(K[I+1])<0 where sgm is a sigmoid function.
- **Visual Density of Clusters:** Density is modelled as

$$\frac{\sigma Pixel_{pointers}}{Pixel_{total}}$$

which categorizes visually sparse and dense data types.

$$\sigma data - points$$

<50000 for presentable data while instantaneous timed values should be less than 1,00,000 over a million superposition of the data points in the data stream.

4. Results and Discussions

4.1. Matcher Synthesis

The RE Pipeline was efficient in identifying 96,000 patterns upon 100,000 total stream patterns with P($Matcher_{success}$) being > 0.876. The table **??** specifies our results across 36 Regular Expressions and their efficacies across 125000 data elements recorded in a stream for 25 minutes. We observed a 1:5 relationship between the time recorded and % of patterns correctly classified/extracted and 1:10 between the number of data elements and % of patterns correctly classified/extracted. Table **??** further reflects the time-bound and

quantitative aspects of match syntheses across 185 experiments.

4.2. Parser Statistics

90% of numeric and mathematical expressions are identified through RE Parsers as long as they adhere to the convention mentioned in the table 1. 23.2% of data is wasted for character data where ASCII encodings complicate their natural interpretations. Particularly, non-English Tokens often intermix with Noisy data and these misinterpretations are shown in the third section of the table 2 where symbolic interpretations dominate their semantic definitions. It is important to note that we are not implementing any visual parsers here and hence, visually similar characters have no effect on the totality of stream characteristics.

4.3. Matcher Enhancements

The Matcher generated in the Matcher Synthesis phase is insufficient to address rapid data streams and fails in infinitesimal similarities between the Matcher attributes and the extracted pattern's nature. The differentiating details have been showcased as follows:

- An opposite RE Pipeline that differentiates between simple & complex mathematical operators should be applied to avoid confusion between Complex Symbols and noisy data.
- RE Pipeline should be extended to include special characters like @,#,$, etc. should be implemented which should further add to the metadata of the stream header while also implementing it within conventional RE Pipeline.
- The Matcher should also host extraneous parameters like the type of data supported by the pipeline for the pipeline to reject any non-obeying data streams immediately.
- The Matcher-Stream Relations should be two-sided where the matcher should extract relevant data while the streamer should identify if the matcher is not disrupting its pipeline.

4.4. Extraction Numerics in RE Pipeline

- **Deterministic Extraction:** f(K(I)) that deals with such data items should produce deterministic outputs rather than a range of outputs. Range of(K(I)) should never exceed 2 in length with the further constraint that Range(K(I))[0] can be metadata/property of the matcher function. Range(K(I))[1] should be a singleton set that should not signify alphanumeric quantities. It should be unidirectional, either numeric or completely character. Noise may also be permitted in a few applications.
- **Probabilistic Extraction:** P(f(K(I))) is assessed over P(K(I)) since extraction is a function of matcher-specific functions of matcher, not the actual stream token. P(f(K(I))) may be approximated through the following relation: $((P(K(I)) * w(K(I))) * M(K(I))) + b_{probExtraction}$ where * is not a normal multiplication operator, but a convolution operation.
- **Similarity Extraction:** Vector Embeddings to infer similarities is a burdensome task for stream pattern recognition, necessitating the employment of alternate methods. Manhattan Distance has been employed in the current system which is shown already in the equation 5.

4.5. Pattern Clusters:

Patterns are deciphered over diversified categories of RE-Based Pattern Recognitions. They include:

4.5.1. Cluster 1:

The RE Extractions in this cluster are nominal logics used in a simple program that employs any kind of letter/digit combination extractions in the input text. The extraction mode is always singular, with the results obtained compounded. The results are combined when they satisfy the below criteria, the equation 6.

$$\begin{aligned} text_{extracted} = Matcher(P_{rsltnt}, K[i+1]) \\ Parser(Cluster, Density) \\ + P(K[i], P_{rsltnt}) \end{aligned} \tag{6}$$

where P1 and P2 are competing patterns, P(x) is probability of dominance of x, || is boolean or, * is convolution function, ?: is teritiary decision operator, and P_{rsltnt} is the winning pattern.

4.5.2. Cluster 2:

This Cluster involves a simple combination of 2 or at most 3 identified patterns. Suppose P1 and P2 are the patterns that are identified, the patterns follow 1-order, 2-order convolutions with optional 3-order convolutions as P1(P2) or P2(P1) or P1*P2 or P2(P2(P1)) or P1(P1(P2)) or P2(P1(P2) or P1(P2(P1)) where P1 is P1(K(I)), P2 is P2(K(I)) and * indicated convolution, not conventional multiplication.

4.5.3. Cluster 3:

Rapid Streams highlight multiple-order pattern correlations for which extensive studies are needed that form our future work.

4.6. RE Approximation Results

RE Approximations involve the direct correlations between REs fed into the pipeline and the Matcher Attributes.

The results are shown in 2. The investigations include:

- **REs from Matcher:** We derived potential RE arrays from Matcher behaviors, which is a reverse engineering procedure. This can be easily employed because the result set is already available.
- **Matcher from Extracted RE:** This holds 98.5% similarity with the above but 1.5% difference lies in how it predicts the matcher by assuming the extracted RE first.

Matcher	K[i]	K[i+1]	Parser	Cluster	Graph	Density	Accuracy	Precision
*	[0-9]	[a-z]	ASCII	2	Scatter	15.001	75.60%	89.90%
	[a-z]	[0-9]	ASCII	1	Density	35.054	65.60%	99.87%
	[a-z]	[a-z]	NA	1	Bar	1.016	97.89%	99.90%
	[0-9]	[0-9]	NA	2 or 3	Probability	55.008	99.87%	55.32%
	[$#%]	[0-9] or [a-z]	ASCII	3	Probability	98.664	52.63%	89.92%
+	[0-9]	[a-z]	ASCII	2	Bar	65.090	87.50%	92.50%
	[a-z]	[0-9]	ASCII	2	Density	15.054	85.60%	92.77%
	[a-z]	[a-z]	Filter	1	Density	0.006	67.98%	75.69%
	[0-9]	[0-9]	NA	2 or 3	Probability	55.008	99.87%	55.32%
	[$#%]	[0-9] or [a-z]	Filter	3	NA	18.554	62.76%	98.92%
Filter	[0-9]	[a-z]	ASCII	3	Density	15.000	96.70%	89.15%
	[a-z]	[0-9]	ASCII	3	Density	15.054	85.60%	92.77%
	[a-z]	[a-z]	Filter	3	Probability	78.865	95.98%	90.06%
	[0-9]	[0-9]	Filter	2 or 3	Probability	25.079	87.78%	90.65%
	[$#%]	[0-9] or [a-z]	Filter	1	NA	0	99.99%	96.66%

Table 2. Results Table: We have tabulated all our results into a singular representation. K[I] and K[i+1] are consecutive stream tokens, Matcher and Parser are the entities resulting from RE Pipeline(Just their overall representation is shown here for simplicity), and Pattern cluster is depicted as Cluster. The presentability studies are seen as Graphs, indicating the type of graph and Density to show inter-point distances. Exceptions are highlighted in red.

- **Matcher from RE:** This is entirely different from the above two because it shows a detailed analysis of the extracted tokens to find the actual RE and then estimate the actual Matcher.

The observations include:

- 65% of character matchers are correctly identified for the first type of investigation though ASCII overheads are present. 98% of numeric matchers are identified in all three investigations.
- 85% of character matchers involving noisy-like characters are correctly segregated in the third investigation where a detailed analysis effectively distinguishes normal data from noisy data.
- Surprisingly, just 55.26% of character data is rightly extracted in the second investigation needing further research.

4.7. Presentability Results

4.7.1. Graphical Analysis

Y value depreciation by >10% would most probably place it in the 3rd and 4th quadrants, implying loss of data. Many Applications employ 1st quadrant and 2nd quadrant formulations, thus highlighting the need for eliminating this depreciation. The only way to do is to treat them as outliers where we can amplify tertiary parameters to scale the points above the X-axis. Tertiary parameters include customary metadata but never biased constants. The relations have been tabulated in table 2 and illustrated in the figure **??**.

4.7.2. Density Analysis

Manhattan Distance of < 100mm may be considered potentially harmful which defeats the application's very purpose of simple UI. The equations considered are 7, 8, and 9. The relation rubrics are represented in 10 and 11.

$$density = \frac{pixel_{whole}}{pixel_{data}} + b_{dns1} \tag{7}$$

where $pixel_{whole}$ and $pixel_{data}$ are the the number of whole pixels and the pixels filled with data respectively, b_{dns1} is density constant.

$$density = \frac{pixel_{whole}}{pixel_{cluster1}} + b_{dns2} \tag{8}$$

where $pixel_{whole}$ and $pixel_{cluster1}$ are the the number of whole pixels and the pixels filled with cluster 1 elements respectively, b_{dns2} is density constant.

$$density = \frac{pixel_{whole}}{pixel_{cluster2}} + b_{dns3} \tag{9}$$

where $pixel_{whole}$ and $pixel_{cluster2}$ are the the number of whole pixels and the pixels filled with cluster 2 elements respectively, b_{dns3} is density constant.

$$density = f(dns1, dns2) * f'(dns3) \tag{10}$$

where dns1,dns2, and dns3 are previously obtained densities, f(x) is a domain-defined function and f'(x) is its complementary function.

$$density = f(dns1, dns3) * f'(dns2) \tag{11}$$

where dns1,dns2, and dns3 are previously obtained densities, f(x) is a domain-defined function and f'(x) is its complementary function.

5. Conclusion and Future Work

The data streams are evaluated for Pattern recognition across 4 attributes: Pattern Synthesis, Pattern Matching, Matcher Extractions, and Pattern Presentability. We did not present Multi-media and multi-source analyses since they need detailed evaluations and conceptions to explain their relevance and theories further. While we were successful broadly in terms of numeric and most of the character data, more elucidations are still needed to differentiate these streams from noise. Our success rate in normal-noisy data stream differentiation was 75% but it will be increased further in our future work. Our future directions for the current research are to further enhance noise elimination and conceptualize theories for Multimedia Data as well.

References

[1] A. K. Jain, R. P. W. Duin and Jianchang Mao, "Statistical pattern recognition: a review," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, no. 1, pp. 4-37, Jan. 2000, doi: 10.1109/34.824819.

[2] Taegong Kim, Cheong Hee Park, Anomaly pattern detection for streaming data, Expert Systems with Applications, Volume 149, 2020, 113252, ISSN 0957-4174, https://doi.org/10.1016/j.eswa.2020.113252.

[3] Karras, Christos & Karras, Aristeidis & Sioutas, Spyros. (2022). Pattern recognition and event detection on iot datastreams. 10.48550/arXiv.2203.01114.

[4] Félix Iglesias Vázquez, Alexander Hartl, Tanja Zseby, Arthur Zimek, Anomaly detection in streaming data: A comparison and evaluation study, Expert Systems with Applications, Volume 233, 2023, 120994, ISSN 0957-4174, https://doi.org/10.1016/j.eswa.2023.120994.

[5] Zheng, N., Xue, J. (2009). Pattern Analysis and Statistical Learning. In: Statistical Learning and Pattern Analysis for Image and Video Processing. Advances in Pattern Recognition. Springer, London. https://doi.org/10.1007/978-1-84882-312-9_1

[6] Kittler, J. (1988). Statistical Pattern Recognition: The State of the Art. In: Cantoni, V., Di Gesù, V., Levialdi, S. (eds) Image Analysis and Processing II. Springer, Boston, MA. https://doi.org/10.1007/978-1-4613-1007-5_5

[7] Supriya Agrahari, Anil Kumar Singh, Concept Drift Detection in Data Stream Mining : A literature review, Journal of King Saud University - Computer and Information Sciences, Volume 34, Issue 10, Part B, 2022, Pages 9523-9540, ISSN 1319-1578, https://doi.org/10.1016/j.jksuci.2021.11.006.

[8] Xiao Bai, Xiang Wang, Xianglong Liu, Qiang Liu, Jingkuan Song, Nicu Sebe, Been Kim, Explainable deep learning for efficient and robust pattern recognition: A survey of recent developments, Pattern Recognition, Volume 120, 2021, 108102, ISSN 0031-3203, https://doi.org/10.1016/j.patcog.2021.108102.

[9] Yan Y, Li J, Liao S, Qin J, Ni B, Yang X. TAL: Two-stream Adaptive Learning for Generalizable Person Re-identification. arXiv. Published November 27, 2021. Accessed [access date]. https://arxiv.org/abs/2111.14290

[10] Samb, D., Slimani, Y., Ndiaye, S. (2024). Toward an Ontology of Pattern Mining over Data Streams. In: Bennour, A., Bouridane, A., Chaari, L. (eds) Intelligent Systems and Pattern Recognition. ISPR 2023. Communications in Computer and Information Science, vol 1940. Springer, Cham. https://doi.org/10.1007/978-3-031-46335-8_12

[11] Jiawei Han, Hong Cheng, Dong Xin, Xifeng Yan, "Frequent pattern mining: current status and future Directions," Data Mining Knowl Discov, vol. 15, no. I, p. 32, 2007

[12] Iqbal Gondal and Joarder Kamruzzaman Md. Mamunur Rashid, "Mining Associated Sensor Pattern for data stream of wireless networks," in PM2HW2N '13, Spain, 2013, p. 8

[13] Jian Pei, Jiawei Han, "Mining Frequent patterns without candidate generation," in SIGMOD '00 Proceedings of the 2000 ACM SIGMOD international conference on Management of data, New York, NY, USA, 2000, pp. 1-12.

[14] Giacometti, Arnaud & Soulet, Arnaud. (2021). Reservoir Pattern Sampling in Data Streams. 10.1007/978-3-030-86486-6_21.

[15] M. Krishnamoorthy, R. Karthikeyan, Pattern mining algorithms for data streams using itemset, Measurement: Sensors, Volume 24, 2022, 100421, ISSN 2665-9174, https://doi.org/10.1016/j.measen.2022.100421.

[16] Liu S, Dai H, Song S, Li M, Dai J, Gu R, Chen G, Baeza-Yates R, Bonchi F. ACER: Accelerating Complex Event Recognition via Two-Phase Filtering under Range Bitmap-Based Indexes. Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. 2024; 1933-1943. doi: 10.1145/3637528.3671814.

[17] Ghouaiel, N., Marteau, P-F., Dupont, M. Continuous pattern detection and recognition in stream – a benchmark for online gesture recognition. Int. J. Applied Pattern Recognition. (xxxx). Vol. X, No. Y, pp.000–000.

[18] Borah, A., Nath, B. Comparative evaluation of pattern mining techniques: an empirical study. Complex Intell. Syst. 7, 589–619 (2021). https://doi.org/10.1007/s40747-020-00226-4

[19] Iwana, Brian, Uchida, Seiichi. An empirical survey of data augmentation for time series classification with neural networks. PLOS ONE. 2021; 16. e0254841. 10.1371/journal.pone.0254841.

[20] Singh, Chetanpal. Machine Learning in Pattern Recognition. European Journal of Engineering and Technology Research. 2023; 8. 63-68. 10.24018/ejeng.2023.8.2.3025.

[21] Fournier-Viger, P., Lin, J. C.-W., Kiran, R. U., Koh, Y. S., Thomas, R. Survey of Sequential Pattern Mining. Data Science and Pattern Recognition. 2017; 1(1): 54-77.

[22] Fournier-Viger, P. et al. Pattern Mining: Current Challenges and Opportunities. In: Rage, U.K., Goyal, V., Reddy, P.K. (eds) Database Systems for Advanced Applications. DASFAA 2022 International Workshops. DASFAA 2022. Lecture Notes in Computer Science, vol 13248. Springer, Cham. 2022. https://doi.org/10.1007/978-3-031-11217-1_3

[23] A. Borah and B. Nath, "Mining patterns from data streams: An overview," 2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), Palladam, India, 2017, pp. 371-376, doi: 10.1109/I-SMAC.2017.8058373.

[24] Giannella, C., Han, J., Yan, X., Yu, P.S. Mining Frequent Patterns in Data Streams at Multiple Time Granularities. 2002.

[25] G. Suseendran, D. Balaganesh, D. Akila and S. Pal, ”Deep learning frequent pattern mining on static semi structured data streams for improving fast speed and complex data streams,” 2021 7th International Conference on Optimization and Applications (ICOA), Wolfenbüttel, Germany, 2021, pp. 1-8, doi: 10.1109/ICOA51614.2021.9442621.