Emergent pattern detection algorithm for big data streams

Lina Fahed, Ayman Alfalou

Emergent pattern detection algorithm for big data streams

Lina Fahed and Ayman Alfalou

ISEN, L@bISEN Yncréa Ouest, 20 Rue Cuirassé Bretagne, Brest, France

ABSTRACT

Pattern detection is an active field in big data streams analytics with numerous ongoing challenges. Actually, due to the great velocity and variety of data, new patterns can appear and change over time. Existing state-of-the-art solutions consist in updating the pattern detection model regularly in order to integrate newly appeared and validated patterns. However, in several applications, such as security and defense, patterns can represent anomalies. Therefore, it becomes crucial to detect new patterns (i.e. new anomalies), as early as possible, in order to react at the right moment. Consequently, emergent pattern detection becomes a very challenging task. To tackle this challenge, we propose EPDA (Emergent Pattern Detection Algorithm): a new and validated algorithm for detecting emergent patterns in data streams. The originality of EPDA consists in exploiting frequent pattern mining techniques by proposing new statistical measures in order to estimate the evolution of emergent patterns over time. To perform this detection in a real-time, EPDA runs on the well-known *Apache STORM* distributed real-time computation system. To better fit our algorithm, we propose a new *Apache STORM* topology which is composed of one Spouts level and two Bolts levels. Experiments on a real data stream have shown the relevance of the proposed measures and the efficiency of our algorithm in a prediction task and in terms of execution time.

Keywords: Emergent pattern detection, early detection, data streams , distributed realtime systems, big data analytics

1. INTRODUCTION

Our digital universe is rapidly growing. The volumes of automatically generated data in 2012 has been estimated to surpass 2.8 zetabytes (2.8 trillion gigabytes),[1] which can represent data streams. Data stream mining field is concerned with the effective, real time, capture of useful information from data streams.[2,3] This task requires the adoption of an incremental online mechanism to process the data as it becomes available. In addition, data stream mining helps often to build predictive models and thus is applied in domains where one can use data stream information for prediction purposes.

In order to demonstrate the importance and challenges in data stream mining, let us take an example in Web Blog streams. Every user's message is associated with an explicit timestamp that represents the exact time it was generated.[4] Such a stream contains rich information and offers significant opportunities for exploration, as well as challenges. One of the challenges is to automatically detect emerging patterns, *e.g.* the emergent topics correlations, that appear in the stream and to do this task in real time. Emergent patterns are typically driven by emergent events, emergent problems, and breaking news that attract the attention of the Web Blog users. Emergent pattern detection is thus of high interest, as it allows the responsible (the administrator, the company, etc.) to react, as soon as possible, in response to emergent patterns.

However, because of the variety and velocity of events in a data stream, new patterns, that have never appeared before, may emerge over time. Therefore, a predictive model which is learned from a segment of the stream (i.e., learned on that stream at a given point of time) and used to predict in the future, is limited because it will not identify and integrate recently emerged patterns in the stream. Designing a predictive model that is regularly updated, by integrating the recently emergent patterns, is a solution that has been proposed in the state-of-the-art.[3] However, in such models, a new pattern is said to be emergent when it has appeared relatively "frequently", and at this moment the pattern can be integrated into the predictive model. But, integrating a new

Further author information: (Send correspondence to Lina Fahed)
Lina Fahed: E-mail: lina.fahed@isen-ouest.yncrea.fr
Ayman Alfalou: E-mail: ayman.al-falou@isen-ouest.yncrea.fr

pattern only once it is frequent limits its use in prediction. Indeed, waiting until a new pattern is frequent, one can miss the opportunity of using it in a prediction task. Consequently, we consider that **the early detection of emerging patterns**, *i.e.* predicting, as early as possible, that a new pattern will be frequent before it is actually frequent, is a solution to integrate this pattern in the predictive model as soon as possible and thus to be able to quickly benefit from its predictive power. This a a very complex task, and to the best of our knowledge, it has not been addressed in the literature.

In this paper, we propose *EPDA* (Emergent Pattern Detection Algorithm): an algorithm for the early detection of emergent patterns in a data stream. The originality of our algorithm is that it detect, as early as possible, the emergence of new patterns that have never been frequent before in the history of the stream. The algorithm runs on *Apache STORM* framework.

The remainder of this paper is organized as follows: In section 2, we present state-of-the-art works. Our proposed *EPDA* algorithm is detailed in the section 3. We proceed to the experimental evaluation of the algorithm in section 4. Finally, we conclude in section 5.

2. RELATED WORKS

In this section, we briefly describe related state-of-the-art works. We present also definitions of some related concepts.

Data streams represent a complex data type, for which research is particularly active. A data stream is an infinite sequence of events generated continuously at a rapid rate.[5] The term "rapid" means that the speed of arrival of the events is high compared to the processing and storage capacities.[6] It is important to notice that the nature of an event in a stream may vary. For example, an event can consist of one or more items/objects (which constitutes an item/object stream),[7] one or more graphs (which constitutes a graph stream),[8] a plain text (which constitutes a text stream),[9] and so on. Let us present the formal definition of a data stream:

DEFINITION 2.1. *Let I be a set of events (items, graphs, texts, etc.) which can be infinite. A* ***data stream*** *is a sequence of timestamped events, as follows: $S =< (t_1, I_{t_1}), (t_2, I_{t_2}), ..., (t_n, I_{t_n}) >$ (with $t_1 < t_2 < ... < t_n \wedge n \simeq \infty$). The data stream has an infinite length and its velocity can be deduced according to the occurrence timestamps t_i of its events.*

The generalization of data from the Web is the main reasons for the appearance of a large number of data streams.[10] The real-time social content can also be seen as a sensor that captures what is happening in the world: this can be exploited in order to detect emergent patterns. Let us mention some examples: streams of user messages in a blog, streams of Web pages (which form graphs because of hyperlinks), streams of scientific articles published with cross references, etc.

Pattern mining in data streams is subject to several constraints such as the impossibility of retaining all the data, the need to carry out the search in a single pass on the data and in an imperatively short time. Consequently, pattern mining in data streams is a very complex task[11,12] and traditional algorithms, designed for sequence processing for example, are not able to handle these constraints. This makes it mandatory to propose algorithms dedicated to data stream mining. Follows, we formalize the definition of a pattern:

DEFINITION 2.2. *A* ***pattern*** *$P =< p_1, p_2, \ldots, p_k >$, is a ordered list of events p_i ($p_i \subseteq I$, for $i = 1, \ldots, k$). The support of the pattern P, denoted $supp(P)$, represents the frequency of P which is the number of occurrences of P in the data stream. P is said to be frequent when $supp(P) \geq minsupp$, where $minsupp$ is a minimal support threshold.*

In addition, as a stream is infinite, one cannot search it in its entirety. Consequently, it is required to define dedicated approaches for processing data in the stream. The literature on data stream mining has shown that window-based approaches are the most efficient in terms of time and memory consumption.[6,13] A window in a data stream is defined as follows:

DEFINITION 2.3. *A window $Win(S, t_s, w)$ in a stream S is a sub-segment of length w that starts at timestamp t_s, and ends at timestamp $t_s + w$.*

A window-based approach divides the mining task into several sub-tasks, each sub-task being interested in mining one part of the stream, called a window,[13] A sliding window is the type that is often used in such

approaches.[6] In a sliding window-based approach, the size of the window is fixed, with the start time shifting once a new event occurs. In this case, the processing of the stream is always performed on the last position of the sliding window. For this reason, such approach is more efficient in terms of prediction accuracy because all the information is processed without any loss. Moreover, it is important to mention that events can be represented in a window in several manners. When only the order of the events is considered, this represents a logical window (for example, when a window covers the last fifteen events) .[14] However, when several events occur at the same time, the window's timestamps can represent seconds, minutes, days, etc. (for example, when a window covers the last three days). This is called a physical window.

Facing the infinite size of data streams, several data mining algorithms propose to memorize "summaries" of the stream in order to make possible the analysis of the entire stream.[6] A summary is a data structure listing the most important and essential information of the stream, and is frequently updated.[15]In the state-of-the-art, pattern mining algorithms have been proposed based on a summary.[15] However, a summary does not guarantee to capture all occurrences of the pattern, which leads to approximate support values of patterns[3,16] and thus to incompleteness of the final result. Consequently, these algorithms are not suitable for our task of detecting early emergent patterns in a data stream. Indeed, the detection of early emergence of patterns, requires to consider each occurrence of the pattern in the stream. Actually, each occurrence of the pattern is crucial as it can be the one that allows to decide whether the the pattern is emergent or not.[12]

In the state-of-the-art, several pattern mining algorithm based on the sliding windows have been proposed. Closed pattern mining has been introduced[17] (a pattern is closed if none of its super-patterns has the same support as its own) in a data stream based on sliding windows and using a tree model to maintain information on all candidate patterns. However, in this algorithm, the information stored about the non-closed candidate patterns is useless, which increases time and memory consumption. More efficient algorithms have been proposed[18] that store only closed patterns (whatever they are frequent or not), which decreases the resources consumed. Algorithms for matrix-based frequent pattern mining have been proposed[19] in which the support of candidate patterns is updated over time. However, such algorithms have time and memory limitations when using a low support threshold.

In the literature, an pattern is said to be emergent when its support increases significantly over time.[20] In this case, it is possible to predict that the support of this pattern will increase further over time. Follows the formal definition of an emergent pattern:

DEFINITION 2.4. *Let $P =< p_1, p_2, \ldots, p_k >$ be a pattern that appears in a data stream S. Let $supp_{t'}(P)$ and $supp_{t''}(P)$ be the supports of P till the timestamps t' and t'' respectively. P is said to be an* **emergent event** *at timestamp t'', if: $supp_{t''}(P) \gg supp_{t'}(P)$.*

Several approaches have been proposed to detect emerging patterns, such as approaches based on the "growth rate" measure, which is represented by the ratio of the support of the pattern at two different timestamps.[20,21] If this rate exceeds a predefined threshold, the pattern is considered as emergent, as follows:

DEFINITION 2.5. *The* **growth rate** *of P between timestamps t' and t'' is denoted: $growthRate_{t',t''}(P) = supp_{t''}(P) \ / \ supp_{t'}(P)$. P is considered as an emergent pattern when: $growthRate_{t_i,t_j}(I) >= min_{growth}$, where min_{growth} is a predefined minimal growth threshold.*

It is important to mention that the minimal growth threshold and the two timestamps considered to detect emergence are application-related parameters. Despite the use of such thresholds, the emergence detection is still performed too late, which is unsuitable for several sensitive context applications. To the best of our knowledge, the early detection of emergent patterns has not been proposed in the literature.

In the state-of-the-art, a classical approach for detecting emergent pattern[22] is to mine all frequent patterns from an off-line data sequence, then monitor these pattern in an online data stream in order to detect their emergence. Notice that such approach monitors the already mined patterns and thus does not allow to detect the emergence of new patterns that have not been learned before, and thus does not allow to anticipate the occurrence of certain patterns (those that have not been mined from the off-line sequence). This represents one of the tackled challenges in this paper.

3. *EPDR*: EMERGENT PATTERN DETECTION ALGORITHM

In this section, we present our proposed algorithm *EPDR* (Emergent Pattern Detection Algorithm) for early detection of emergent patterns in a data stream.

3.1 Principle of the *Apache STORM* framework

In order to perform a real-time data processing, we run our algorithm on *Apache STORM*[*]: a distributed real-time stream computation framework based on a "topology" architecture. A topology is represented by a set of calculation units: spouts and bolts (see figure 1). Spouts represent "taps" that connect to the data stream, manage retrieving data and perform a first treatment. Bolts represent "valve locks" that handle the main treatment on some data or results sent from a spout. The user has to implement the necessary treatment to be performed in spouts and bolts, and can design several levels of parallelism.

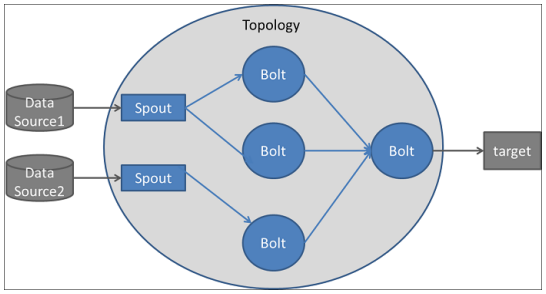

Figure 1: The architecture of an *Apache STORM* topology.[23]

3.2 Principle of the algorithm *EPDA*

Our algorithm *EPDA* has **two originalities**: (i) unlike traditional algorithms, it does not monitor already known frequent pattern, it instead mines patterns directly from the data stream and predicts their emergence, *i.e.* predicts that they will be frequent in the future. (ii) *EPDA* detects the emergence as early as possible, *i.e.* it considers only a few occurrences of a pattern in order to perform detection. However, one can criticize that relying on a few occurrences is not sufficient to reliably predict that the pattern will emerge (will be frequent) in the future. Thus, we need to exploit additional information in order to make this detection more reliable. For this reason, we make the following hypothesis:

Hypothesis: New patterns that emerge are not totally new, they are linked and influenced by other already known patterns (*i.e.*, through patterns mutation over time). Thus, we consider as emergent, each new pattern that tend to appear and is *similar* to already known patterns.

Choice of similarity measure: In order to calculate the similarity between patterns, we choose the "Edit distance" measure:[24] a non semantic similarity measure[25] that allows us to detect new patterns that represent new trends and behaviours in the data stream. Edit distance measures the degree of dissimilarity between two patterns. It takes values from 0 to 1. Notice that it equals 0 when the two patterns are identical.

We design steps of *EPDA* to be run on an *Apache STORM* topology composed of one *spouts* level and two *bolts* levels. The steps of *EPDA* are resumed below, detailed in the following sections, and presented in figure 2.

- Initialization: a set of already frequent patterns, which we call "reference patterns", is obtained.
- Pattern mining: new patterns are mined from the data stream. This step is performed in the spouts level of the proposed *Apache STORM* topology.
- Updating patterns information: Only new patterns similar to the reference patterns are considered, and their supports and occurrences timestamps are updated. This step is performed in the first level of *bolts*.
- Early detection of emergent patterns: a new growth rate measure is proposed and applied in order to detect emergent patterns as early as possible. This step in performed in the second level of *bolts*

[*]Apache STORM: `http://storm.apache.org`

3.3 Initialization step

A set of frequent patterns that we call *reference patterns* must be available before the execution of the *EPDA* algorithm. These patterns will be used to calculate the similarity with the new patterns mined from the data stream. We formalize the definition of a reference pattern as follows:

DEFINITION 3.1. *A* **reference pattern**, *denoted as* P_{ref}, *is a frequent pattern previously learned (extracted by a pattern mining algorithm from a dedicated dataset). The set of reference patterns is denoted as* $SetP_{ref}$.

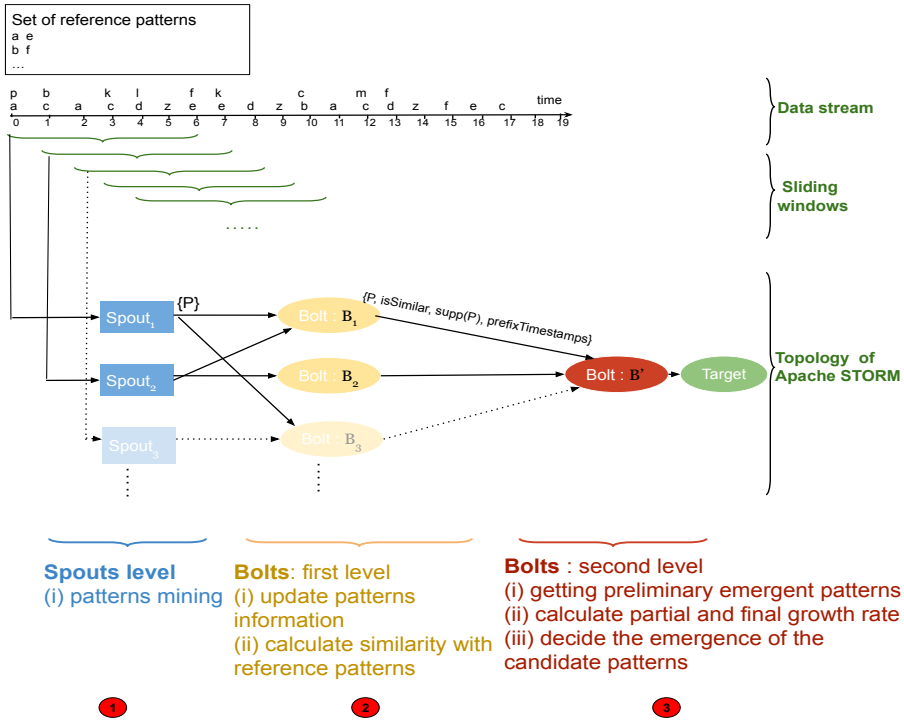

Figure 2: Scheme of the algorithm *EPDA*.

3.4 Pattern mining step

The pattern mining step is performed in a *spouts* level. Each *spout* is responsible for monitoring the data stream and receiving a package of events. Each package represents a sliding window necessary to extract patterns (see figure 2). A traditional pattern mining algorithm is applied on this window (see blue components in figure 2) in order to mine patterns. Notice that only a single occurrence can be found for a pattern in each window. Notice that once a *spout* completes this task, it distributes the set of mined patterns $SetP$ to several available bolts. The *spout* is released from this task, it now becomes available and ready to process a new window.

3.5 Updating patterns information step

Updating patterns information is performed in a first bolts level, that we refer to as $Bolt_i$ (see yellow components in figure 2). Each $Bolt_i$ receives (from a spout) one or more patterns. Each pattern represents is a candidate, $P_{candidate}$, that may emerge in the future, for which the following information is calculated and updated: (i) the events that compose $P_{candidate}$, (ii) a boolean value to indicate whether $P_{candidate}$ is similar to at least one reference pattern, (iii) the updated support of $P_{candidate}$, (iv) the set of the start timestamps of the occurrences of $P_{candidate}$. Consequently, a candidate pattern $P_{candidate}$ is thus represented as follows: *(P, isSimilar, support, prefixTimestamps)*. The set of candidate patterns is referred to as $SetP_{candidates}$.

Using the similarity measure: During this step, the similarity between a candidate pattern and the reference patterns is calculated by applying the Edit distance (see section 3.2). We thus use $max_{distance}$: a maximum threshold of the Edit distance measure. We propose that, if the candidate pattern is identical to at least one

reference pattern, the information associated with this pattern regarding similarity: $isSimilar$ takes the value $false$. Indeed, we want to detect new trends represented by new pattern never learned before, that is to say, they are not identical to any reference pattern.

3.6 Early detection of emergent patterns step

The step of early detection of emergent patterns in performed in a second level of bolts (see the red component in figure 2), as follows:

Detection of preliminary emergence: In this step, we focus only on the candidate patterns $P_{candidate}$ similar to the reference patterns: $isSimilar = true$. Then, we select patterns having a first signal of emergence and we call them "preliminary emergent patterns", formally defined as follows:

DEFINITION 3.2. *A* **preliminary emergent pattern**, *denoted as $P_{emergPre}$, is a candidate pattern for which the support exceeds a predefined support threshold $minsupp_{emergPre}$. The timestamp at which the pattern becomes a preliminary emergent pattern differs from one pattern to another, and will be denoted as $t_{emergPre}$. The support of the pattern at this time is denoted as $supp_{emergPre}(P)$ (to distinguish it from $supp(P)$ used in the general case, and is much more bigger).*

We consider that, at this stage, it is not reliable to predict whether the preliminary emergent pattern will really be frequent in the future. We thus propose to monitor a little more this pattern via an additional step of analysis: detection of the final emergence.

Detection of final emergence: For each preliminary emergent pattern $P_{emergPre}$, we start, at $t_{emergPre}$, a monitoring phase during a very short period of time. We choose to control the growth rate (see definition 2.5) of the support of each $P_{emergPre}$ at several "checkpoints". The number of these *checkpoints*, denoted as n, (see figure 3 where $n = 3$), is determined according to the degree of importance, for the application, to perform the detection as soon as possible. We propose to calculate the growth rate between consecutive *checkpoints* (that are shifted by k timestamps). Therefore, we get $growthRate_1$, $growthRate_2$, ..., $growthRate_n$. In figure 3, we present an example of *checkpoints* at which growth rates should be calculated. For $k = 5$, *checkpoints* are $checkpoint_1 = t_{10}$, $checkpoint_2 = t_{15}$ and $checkpoint_3 = t_{20}$.

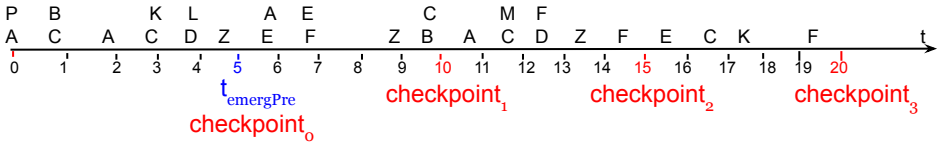

Figure 3: An example of checkpoints to calculate the growth rate, $k = 5$.

We now propose to predict the emergence of the pattern $P_{emergPre}$ at $checkpoint_n$ by taking into account all growth rates calculated at previous *checkpoints*. We propose a new measure that we call *final growth rate* that calculates the average of these growth rates, which allows to have an overall view of the growth of the support of the candidate pattern since the first *checkpoint*, as follows: (equation 1:)

$$growthRate_{final}(P) = mean(growthRate_1(P), ..., growthRate_n(P)) \quad (1)$$

Let min_{growth} be a minimal threshold of the final growth rate. When $growthRate_{final}(P)$ exceeds this threshold, we make the prediction that the pattern P is an emerging pattern.

4. EXPERIMENTATIONS

In this section, we present the experimental studies that we conducted to validate the *EPDA* algorithm. To do so, we will first present the data stream on which the algorithm runs. In a second step, we perform an analysis of the performance of the algorithm according to different parameters. It is important to mention that, to the best of our knowledge, no work in the literature has been dedicated to the detection of new emerging patterns as early as possible in a data stream. Therefore, the *EPDA* algorithm is not directly comparable to any other state-of-the-art algorithm.

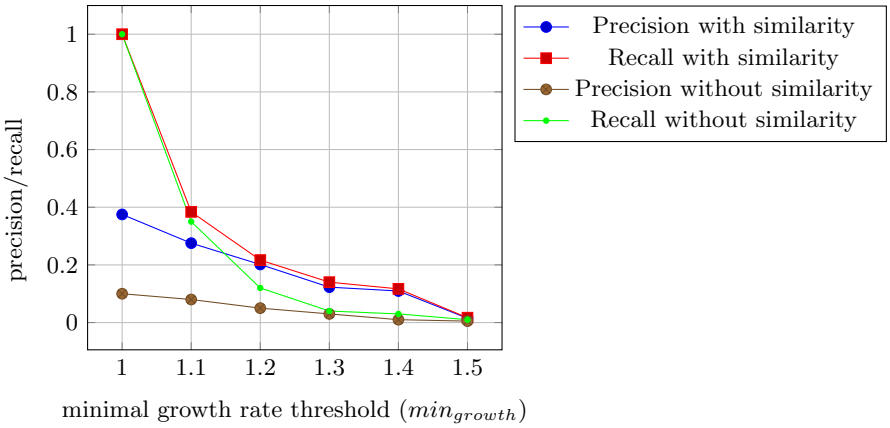

Figure 4: Precision and recall according to the minimal growth rate threshold (min_{growth}) : (i) when taking into account similarity with reference patterns, (ii) without taking into account similarity with reference patterns.

4.1 Data stream and initialization phase

In this experiments, we use a data stream of user messages from a Web Blog. Each message is pre-analyzed and represented by a set of events (keywords). For the initialization phase, the first $10,000$ messages are used to extract reference patterns (see definition 3.1), then $17,612$ messages are uesd as a data stream. In order to mine reference patterns, we apply *PrefixSpan* traditional algorithm[26] with the following parameters: $minsupp = 20$, $w = 100$. We obtain $15,982$ reference patterns with a mean support value of 21.

4.2 Performance of *EPDA* in a detection task

Recall that the main originality in *EPDA* lies in the hypothesis that the new patterns are necessarily linked to the reference patterns and therefore similar to them. The question that arises is: when taking into account only similar patterns, is the algorithm more efficient in detecting the emergence of new patterns?
In order to answer this question, we are interested in reproducing the performance evaluation (represented by precision and recall metrics), but this time we also keep the dissimilar patterns.
After several iterations, we choose to fix $max_{similarity} = 0,6$, $minsupp_{emergPre}(P) = 5$ $n = 5$, $k = 20$ which allows to calculate 5 partial growth rates. At the end of the studied part of the stream, we get $22,398$ patterns with a support bigger than 20. Our objective is to detect the emergence of these pattern as soon as possible.
In figure 4, we present the precision and recall according to min_{growth} in two cases: (i) when taking into account the similarity between the new patterns and the reference patterns, (ii) without taking into account similarity. We first notice that precision and recall curves for both cases have the same behaviour.
For $min_{growth} = 1.20$ and when similarity is taken into account, 20% of patterns detected as emergent, actually emerge (precision), and 21% of patterns that actually emerge are detected by the algorithm (recall). We consider this result quite satisfactory given the difficulty of the context of emergence detection: we do not just perform an emergence detection, but also a detection as soon as possible with a very few information. Moreover, this detection concerns only new patterns: we detect only those that satisfy the hypothesis of similarity to reference patterns, which makes this task more complex. In addition, we notice that precision and recall are better when similarity is taken into account. For $min_{growth} = 1.20$, the precision is 4 times higher when similarity is taken into account than when it is not . The recall is 1.75 times higher when similarity is taken into account than when this measure is not taken into account .
We can conclude that there is indeed a relation between the new patterns and the reference patterns in the studied data stream, which validated our hypothesis.

4.3 Execution time

It is important to mention that the *Apache STORM* platform, on which the *EPDA* algorithm runs handles the parallelization of treatments in an optimal way. For this reason, the execution time of *EPDA* algorithm is very low: 850 seconds to process the data stream ($17,612$ timestamps). We have also noticed that the variation of different thresholds and parameters necessary for the execution of the algorithm does not significantly impact the execution time. However, further analysis has shown that the similarity calculation performed in the algorithm represents

60% of its execution time. It is known in the literature that similarity computation is very time consuming. Despite the impact of this measure on the execution time, the performance of the algorithm (represented by precision and recall) in detecting emergent patterns remains the most important.
Following these experiments, we can conclude that *EPDA* succeeds in detecting the emergence of new patterns as early as possible.

5. CONCLUSION AND PERSPECTIVES

In this paper, we proposed *EPDA* (Emergent Pattern Detection Algorithm): an algorithm for early detection of new emergent patterns in a data stream, based on the proposition of a new growth rate measure. *EPDA* is based on the hypothesis that new patterns that tend to appear and that are *similar* to known patterns, will emerge in the future. *EPDA* runs on the *Apache STORM* distributed processing platform. We have proposed a topology for *Apache STORM* composed of one level of spouts and two levels of bolts. *EPDA* algorithm is evaluated on a real data stream. This experiment validate the performance of the algorithm. The validation of this algorithm on another data stream is one of the main perspectives of this work. Moreover, in several applications, the emergence detection is considered as a way to anticipate and to react in order to prevent the propagation or the dominance of the emergence event. As a future work, we are interested in studying how to impact future events.

REFERENCES

[1] Gantz, J. and Reinsel, D., "The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east," *IDC iView: IDC Analyze the future* **2007**, 1–16 (2012).

[2] Atkinson, K., Coenen, F., Goddard, P., Payne, T., and Riley, L., "Data stream mining with limited validation opportunity: towards instrument failure prediction," in [*International Conference on Big Data Analytics and Knowledge Discovery*], 283–295, Springer (2015).

[3] Gaber, M. M., Zaslavsky, A., and Krishnaswamy, S., "Mining data streams: a review," *ACM Sigmod Record* **34**(2), 18–26 (2005).

[4] Mathioudakis, M. and Koudas, N., "Twittermonitor: trend detection over the twitter stream," in [*Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*], 1155–1158, ACM (2010).

[5] Krempl, G., Žliobaite, I., Brzeziński, D., Hüllermeier, E., Last, M., Lemaire, V., Noack, T., Shaker, A., Sievi, S., Spiliopoulou, M., et al., "Open challenges for data stream mining research," *ACM SIGKDD Explorations Newsletter* **16**(1), 1–10 (2014).

[6] Csernel, B., Clerot, F., and Hébrail, G., "Traitement des flux de données," *37emes Journées de Statistique (SFDS)* (2005).

[7] Cugola, G. and Margara, A., "Processing flows of information: From data stream to complex event processing," *ACM Computing Surveys (CSUR)* **44**(3), 15 (2012).

[8] McGregor, A., "Graph stream algorithms: A survey," *ACM SIGMOD Record* **43**(1), 9–20 (2014).

[9] Aggarwal, C. C. and Zhai, C., [*Mining text data*], Springer Science & Business Media (2012).

[10] Bifet, A., "Mining big data in real time," *Informatica* **37**(1) (2013).

[11] Tseng, V. S., Chu, C.-J., and Liang, T., "Efficient mining of temporal high utility itemsets from data streams," in [*Second International Workshop on Utility-Based Data Mining*], 18, Citeseer (2006).

[12] Gao, C., Wang, J., and Yang, Q., "Efficient mining of closed sequential patterns on stream sliding window," in [*Data Mining (ICDM), 2011 IEEE 11th International Conference on*], 1044–1049, IEEE (2011).

[13] Zhu, Y. and Shasha, D., "Efficient elastic burst detection in data streams," in [*Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*], 336–345, ACM (2003).

[14] Muthukrishnan, S., [*Data streams: Algorithms and applications*], Now Publishers Inc (2005).

[15] Midas, C. D. et al., "Résumé généraliste de flux de données," in [*EGC*], 255–260 (2010).

[16] Giannella, C., Han, J., Pei, J., Yan, X., and Yu, P. S., "Mining frequent patterns in data streams at multiple time granularities," *Next generation data mining* **212**, 191–212 (2003).

[17] Chi, Y., Wang, H., Yu, P. S., and Muntz, R. R., "Moment: Maintaining closed frequent itemsets over a stream sliding window," in [*Data Mining, 2004. ICDM'04. Fourth IEEE International Conference on*], 59–66, IEEE (2004).

[18] Jiang, N. and Gruenwald, L., "Cfi-stream: mining closed frequent itemsets in data streams," in [*Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*], 592–597, ACM (2006).

[19] Gao, C. and Wang, J., "Efficient itemset generator discovery over a stream sliding window," in [*Proceedings of the 18th ACM conference on Information and knowledge management*], 355–364, ACM (2009).

[20] Dong, G. and Li, J., "Efficient mining of emerging patterns: Discovering trends and differences," in [*Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*], 43–52, ACM (1999).

[21] Zhu, S., Ju, M., Yu, J., Cai, B., and Wang, A., "A review of contrast pattern based data mining," in [*Seventh International Conference on Digital Image Processing (ICDIP15)*], 96311U–96311U, International Society for Optics and Photonics (2015).

[22] Luhr, S., Venkatesh, S., and West, G., "Emergent intertransaction association rules for abnormality detection in intelligent environments," in [*Intelligent Sensors, Sensor Networks and Information Processing Conference, 2005. Proceedings of the 2005 International Conference on*], 343–347, IEEE (2005).

[23] Smart Grid Big Data Management Team, "The right big data technology for seat grid distributed stream computing," *Accenture Blog* (2014).

[24] Levenshtein, V. I., "Binary codes capable of correcting deletions, insertions, and reversals.," *Forschungsbericht.–707–710 S* (1966).

[25] Veltkamp, R. C., "Shape matching: similarity measures and algorithms," in [*Shape Modeling and Applications, SMI 2001 International Conference on.*], 188–197, IEEE (2001).

[26] Han, J., Pei, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U., and Hsu, M., "Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth," in [*proceedings of the 17th international conference on data engineering*], 215–224 (2001).