Master thesis Espose: Conception and Development of Semantic Complex Pattern Mining methods on Event Streams

Leon Bornemann, Freie Universität Berlin

***Abstract*—This is an expose for the proposed topic**

***Keywords*—data mining, semantic pattern mining, event, stream, streaming**

I. Introduction

Almost any application domain of information systems has some data that is being generated. Data is available in many different forms. One of these forms are data streams. Data streams are not limited to the recent rise in popularity of video and audio streams. On the contrary the application domains that produce data in the form of streams are very diverse. They include for example constantly running business applications that log business activities and events, sensor networks that report usage data or devices that take measurements of physical quantities (such as temperature, pressure, humidity, etc...) at certain points of time.

The fact that streams generate a constant stream of data and thus lead to a constantly growing database is a significant difference to classic applications of data mining in which there is a static (training) database. Despite that significant difference in the data representation, many fields of interest in the context of static databases remain the same for data streams. Common areas of interest are frequent patterns, predictive patterns, association rules, clustering and classification of the data entities. Approaches and algorithms that solve these problems for static databases, while by no means fully researched, are rather well known and evaluated. Applying these methods to data streams can present challenges and may demand many modifications due to the large and possibly infinite amounts of data produced by streams. Naturally data mining methods for stream data must be especially fast, scalable and memory efficient.

This thesis will focus on the subtopic of episode pattern mining from stream data or very large log files or databases. While mining of frequent episodes has already been looked at (TODO: cite PHD thesis), the suggested algorithms use multiple passes over the database to find all frequent episodes. Multiple passes are often not feasible for large data streams, since it is impossible to store the stream in its entire length and thus impossible to look at it more than once. Additionally the state of the art episode definition is rather restrictive in the context of complex events. It is for example impossible to express disjunctions within episodes. Including additional operators like disjunctions would extend the expressive power of episodes, which means that more complex events can be specified and mined as episodes. However, expanding the expressive power also means that episodes will become more difficult to mine, since there are more potential candidates that the algorithm needs to keep track of. This is where semantic knowledge can come into play. In a realistic scenario, users are not interested in all the frequent complex patterns, since many of them might be meaningless. Semantic or domain knowledge in the form of ontologies can help identify the most interesting candidate patterns and thus help in pruning irrelevant work and therefore improve algorithm performance. The main contribution of this thesis are planned to be the following:

- A single pass approximation algorithm that finds apporximately frequent episodes in one pass over the data.
- An extension to the state of the art definition of Episodes that increases expressive power
- A framework to use semantic knowledge to help prune candidates of the mining algorithm.

In summary the research question in particular is split into two parts:

- How can frequent episode mining algorithms be modified to be applicable to large scale data streams
- How can additional semantic knowledge in the form of rules, constraints or ontologies be used to help episode mining algorithms

II. Proposed Research Methodology

As already mentioned in the introduction it is the goal of this thesis to find novel methods for the task of episode mining in stream data. The novel methods will be specified in a descriptive and formal manner. Subsequently they will be evaluated by means of empirical evaluation. For this purpose the author plans to use both synthetic data and at least one publicly available real-life data set.

III. Background

A. Previous Work

Of all the fields related to this thesis basic pattern mining is arguably the most well researched and widely used related research area. First to mention are of course the standard pattern mining algorithms [1]. The authors of this paper nicely summarize different approaches, which either use candidate generation and the apriori-principle or pattern growth strategies. Problematic with the original Apriori-Algorithm is of

course that while the apriori-principle helps to prune the candidate generation, its runtime can still be exponential. On top of that it requires multiple passes over the database, which is something that can be impossible given large data streams. These general pattern mining algorithms have of course already been modified or improved for more specific purposes. For example event data logs have been used to build predictive models that search for patterns that predict critical events in the log file [3].
A rather large subfield of pattern mining is the field of (business) process mining. This particular domain usually evaluates log-files of (business) applications that log sequences of events. These logs can be mined to satisfy many different information needs, for example clustering [5] or process models [6]. Another approach in the field of business logs and process mining is outlier and anomaly detection [4]. The authors use the length of the longest common subsequence between event sequences as a distance measure to find unusual patterns of events. This field of interest is especially interesting since there are a lot of business applications, some of which use legacy software, whose behaviour is at times unclear or error-prone. Mining underlying business models or unusual behavior can help identify errors.
Apart from the basic patterns that can be mined from data like frequent itemsets or association rules there is also the research field of so called complex event patterns. A specific subtype of complex events are episodes. Episodes have roughly been categorized as serial, parrallel or composite and there are different mining methods proposed for each of these [11] TODO: Rest of the sources. Episodes can only be mined from a sequential database, making this field also a subfield of sequential pattern mining, for which of course also several mining methods for static databases exist. Comprehensive overviews are given by Slimani et al. [12] as well as Zhao at al. [12]. Since evaluating episode mining algorithms on real-life datasets is often difficult due to lack of knowledge about the ground truth generation of realistic, synthetic datasets has been looked into as well [13]. The other big research area that is directly related to this thesis is of course the mining of data streams. As already mentioned in the introduction the mining of data streams is becoming more and more popular. It is often impossible to read or store the entire data stream, which is why approximation algorithms are used [2]. If classic data mining approaches are to be applied they need to be scalable and memory efficient. Such solutions do exist for many problems, for example for frequent pattern mining [7]. If the data streams grow too large however approxmation algorithms need to be used [9].
Some contributing data scientists have also approached Streams in a similar manner as relational databases [8]. The authors present their progress at building a general purpose Data Stream Management System (DSMS) prototype, as a pendant to the traditional Database Management Systems (DBMS). They suggest the usage of an extension of SQL as a stream query language to allow for continuous queries and discuss approximation techniques.

TODO: more sources/topics for data stream mining
TODO: include sources and write the part about smart home/sensor networks and patterns/events
TODO: sources about time series
TODO: More sources about sequential patterns
TODO: More sources about episode pattern mining

B. Complex Pattern Mining: Basic Definitions

Figure 1 gives an overview over the basic setting. On the lowest level of abstraction we have the low-level event stream, which is the unrefined data coming directly from the sources (for example sensor data). The low-level stream needs to be transformed in some way to an annotated event stream, which then in turn gets mined to discover complex events. The mining algorithm's basic input is the stream of annotated events, but it can also use an ontology, which contains additional information about the event types (TODO: define ontology)

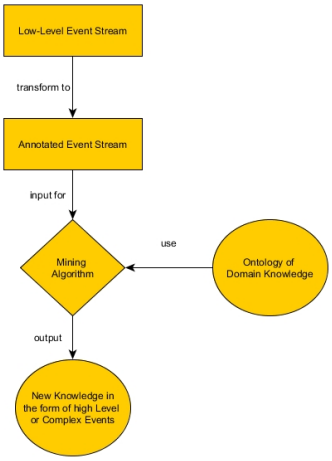

Fig. 1. Basic Problem Structure

We follow up with a few definitions about basic terminology and how these will be used in this thesis.

Definition 1. ***Low Level Event*** *A Low Level Event v is defined as a tuple $l = (t, v_1, ..., v_k)$, where $t \in \mathbb{N}^+$ is the timestamp of its occurance and $v_i \in S_i$ is a value reported with the event. S_i is a set of all possible values for the position i which we will not narrow down further here since this obviously depends on the application domain. Common datatypes would of course be real numbers or categorical values.*

Definition 2. ***Annotated Event*** *An annotated event e is defined as a tuple $e = (t, T)$, where $t \in \mathbb{N}^+$ is the timestamp and $T \in \Sigma$ is its derived event type. Σ is the set of all possible event types, which will sometimes also be referred to as the event type alphabet.*

Note that this definition assumes that events are instantaneous, which is the usual definition of events (TODO;: is that true?), but there are other variants of event definitions that also allow for events to have a duration

(TODO: cite sequences of temporal intervals?).

Given these two definitions we can give a general definition of the first step of the basic setting, the transformation of low level events to annotated events as a function:

Definition 3. ***Transformation Function*** *A transformation function is a function that takes a low level event l input and maps it to a corresponding annotated event (t, T).*

Note that this is a very broad definition, since in most cases this function is completely dependent on the domain as well as target event type alphabet and thus by extension the complex events the user is interested in. There are however general methods to automatically transform tuples of real values to annotated events by using bagging (TODO: sources for that). These methods create the target event type alphabet on the fly (depending on the input types). These methods can be useful if we know nothing about the semantics of the low level data or if there is no clear event type alphabet as the target.

Definition 4. ***Annotated Event Stream*** *An Annotated Event Stream S of length n is defined as a sequence $S = (e_1, ..., e_n)$, where all e_i are annotated events and given two events $e_i = (t_i, A)$ and $e_j = (t_j, B)$ we know that $i < j \implies t_i \leq t_j$ (the stream is sorted according to the timestamps).*

The alert reader will have noticed that we assume a total order in the annotated event stream, since we forbid that two consecutive timestamps may have the same value. This means that we do not allow two events to occur concurrently within the stream. For non-distributed streams this basically no restriction, since the events are ordered anyway. If we face a distributed stream (for example in a sensor network) however it might become tricky to fulfill this constraint, since clock synchronization might be an issue. This property is by no means central to the algorithms we will consider, but it does help when analyzing theoretical properties, which is why we assume it for now.
It is also important to note that this definition somewhat simplifies reality, since it assumes that the stream has a certain length, in other words the stream is finite. This is unproblematic if we can assume the following:

- n always contains the number of all elements we have seen so far
- n gets updated whenever we read a new event

However we need to keep this restriction in mind, since the size of the stream will become relevant when determining which episodes are frequent. Most of the time we are not interested in the whole stream, but instead only look at a small window:

Definition 5. ***Time Window*** *Given an annotated event stream S we define the Time Window $W(S, q, r)$ with $q, r \in \mathbb{N}^+$ and $q < r$ as the subsequence of S that includes all events of S that have a timestamp t where $q \leq t \leq r$. We call $w = r - q + 1$ the size of Window W.*

TODO: where to put this:
The form of the ontology is not further specified in this definition since it is very user specific. Common knowledge representations in the area of semantic web technologies are of course RDF-Graphs. The precice process of enriching the data will be discussed later in section TODO. Fortunately the definition is still sufficient for a general purpose episode mining algorithm, since that algorithm does not require domain knowledge, it just requires the high level event stream. The following definitions are mostly inspiered by [10].

C. Event Episodes

Given a high level event stream there are multiple ways to define episodes of events. We will give two different definitions, first a very compact and formal definition and second a longer definition that is a bit more clear and less formal.

Definition 6. ***Episode*** *An event episode E of length m (also called m-Episode) is defined as a triple: $E = (N_E, \leq_E, g_E)$ where $N_E = \{n_1, ..., n_m\}$ is a set of nodes, $\leq_E$ is a partial order over N_E and $g_E : V_E \to \Sigma$ is a mapping that maps each node of N_E to an event type. If $\leq_E$ is a total order we call E a serial episode, if there is no ordering at all we call E a parallel episode. Otherwise we speak of composite episodes.*

Think of the above definition as a template for concrete occurrences, which we define next:

Definition 7. ***Episode Occurrence*** *An event episode $E = (N_E, \leq_E, g_E)$ is said to occur in a window W if events of the types that the nodes in N_E are mapped to occur in W in the same order that they occur in the episode. More formally if we are given a sequence of events $S = ((t_1, T_1), ..., (t_n, T_n))$ (which may be a Time Window, aka. a subsequence of the original Stream) we can define an occurrence of E as an injective Map $h : N_E \to \{1, ..., n\}$, where $g_E(n_i) = T_{h(n_i)}$ and $\forall\, v, w \in V_E : v \leq_E w \implies t_{h(v)} \leq t_{h(w)}$ holds.*

For example given the high level event stream $S = [(12, A), (14, B), (19, C), (22, A), (34, D)]$ the Episode $E = (\{n_1, n_2\}, \{v_1 \leq_E n_2\}, \{n_1 \to B, n_2 \to A\})$ occurs in window $W(S, 14, 22)$.

TODO: include the more simple Definition. For now I provide the Link to the paper where the definition is easier to understand (but of course equivalent): http://infolab.stanford.edu/ ullman/mining/episodes.pdf

As already stated many times we are interested in episodes that are frequent. Frequency for episodes however can be defined in different ways and the different definitions have a considerable impact on potential algorithms. The first two definitions, which we will refer to as window based Frequency and Minimal Occurance based Frequency, are mentioned by Zimmermann, when he presents his method for synthetic episode generation [13], but were originally conceived in TODO: include original sources. We refer to the third Definition as the Non-Overlapping Occurrence based Frequency which was suggested by Laxman et al. [15].

Definition 8. ***Episode Frequency - Window based Definition*** *Given a high level event stream S, a fixed window size of w and an Episode E, we define the window based frequency $w\_freq(E)$ as the number of windows W with size w of S in which E occurs: $w\_freq(E) = |\{W(S,q,r) \mid r - q + 1 = w \wedge E\ occurs\ in\ W\}|$.*

This definition can be confusing at first since it is intended that episode occurrences that are comprised of the exact same events count just as many times as there are windows in which the events appear. If we have a window size of $w = 11$ for the above example, we can find the Episode *B after A* in the consecutive windows $W(S, 12, 22)$, $W(S, 13, 23)$ and $W(S, 14, 24)$, which means we will get a frequency of 3 just for the two events $(14, B)$ and $(22, A)$. This effect obviously increases with the window size.
The second definition does not require a fixed window size to be specified but instead uses the concept of minimal occurrences:

Definition 9. ***Minimal Occurrence*** *An event episode E is said to occur minimally in a window $W(S,q,r)$ if E occurs in W and there is no subwindow of W in which E also occurs. In this context we also refer to the window W itself as a minimal occurrence of E.*

Definition 10. ***Episode Frequency - Minimal Occurrence based Definition*** *Given a high level event stream S and an Episode E, we define the minimal occurrence based frequency $mo\_freq(E)$ as the number of minimal occurrences of E in S.*

The third definition introduces the concept of non-overlapping occurrences:

Definition 11. ***Non-Overlapping Occurrences*** *Given a m-Episode $E = (N_E, \leq_E, g_E)$ where $N_E = \{n_1, ..., n_m\}$, two occurrences h_1 and h_2 of E are non-overlapped if either*

- $\forall\, n_j \in N_E : h_2(n_1) > h_1(n_j)$ *or*
- $\forall\, n_j \in N_E : h_1(n_1) > h_2(n_j)$

A set of occurrences is non-overlapping if every pair of occurrences in it is non-overlapped.

This leads to the Definition:

Definition 12. ***Episode Frequency - Non-Overlapping Occurrences based Definition*** *Given a high level event stream S and an Episode E, we define the non-overlapping occurrence based frequency $noo\_freq(E)$ as cardinality of the largest set of non-overlapped occurrences of E in S [15].*

When looking at these definitions it is not clear whether any of these is always superior to or more useful than the other since they have different properties. We mention them briefly:

- As already mentioned in the above example. The window based frequency counts an episode occurrence that is comprised of the same events in multiple windows. This might especially distort the count if the window size is high and the events in the episode happen with minimal delay between them.
- The minimal occurrence based definition of frequency does not suffer from the problem of the previous point
- The window based definition has the advantage that it already incorporates a fixed size during which episodes may occur, meaning there can not be episodes that stretch over a time period larger than the fixed window size w. This might be beneficial for potential algorithms, since it reduces the search space for episodes. On top of that it is also closer to reality, since episodes normally happen within a small time window (TODO: cite source for this). Of course also the minimal occurrence based definition can be extended to incorporate a maximal time span.
- Both definitions do not consider multiple occurrences of an episode in the same time window.

TODO: move the above to a separate chapter. Due to the central importance of these definitions we will discuss and compare them in detail in section TODO.

Given these definitions it is the goal of this thesis to develop algorithms that given a stream of high level events will identify frequent episodes in that event stream. Frequent episodes are episodes whose frequency is greater than a user defined threshold, usually relative to the stream length or to the number of inspected windows. Since we are dealing with data-streams the algorithms must have the following properties:

- The algorithms must be incremental, since in contrast to classical database scenarios, we might not have all the data at once, but get continuous input instead
- The algorithms must require only a single pass over the data, since it might be impossible to store the entire stream, especially since we might be dealing with infinite streams. At best it is possible to store a small window of the most recent data in memory and access that window multiple times.

REFERENCES

[1] Han, Jiawei, et al. "Frequent pattern mining: current status and future directions." Data Mining and Knowledge Discovery 15.1 (2007): 55-86.

[2] I could not find the book on google scholar, thus I provide the link to the chapter for now.

[3] Chen, Jun, and Ratnesh Kumar. "Pattern Mining for Predicting Critical Events from Sequential Event Data Log." WODES. 2014.

[4] Sureka, Ashish. "Kernel Based Sequential Data Anomaly Detection in Business Process Event Logs." arXiv preprint arXiv:1507.01168 (2015).

[5] Vaarandi, Risto, and Mauno Pihelgas. "LogCluster-A Data Clustering and Pattern Mining Algorithm for Event Logs." Network and Service Management (CNSM), 2015 11th International Conference on. IEEE, 2015.

[6] Priyadharshini, V., and A. Malathi. "Analysis of Process Mining Model for Software Reliability Dataset using HMM." Indian Journal of Science and Technology 9.4 (2016).

[7] Lin, Kawuu W., Sheng-Hao Chung, and Chun-Cheng Lin. "A fast and distributed algorithm for mining frequent patterns in congested networks." Computing (2015): 1-22.

[8] Motwani, Rajeev, et al. "Query processing, resource management, and approximation in a data stream management system." CIDR, 2003.

[9] Manku, Gurmeet Singh, and Rajeev Motwani. "Approximate frequency counts over data streams." Proceedings of the 28th international conference on Very Large Data Bases. VLDB Endowment, 2002.

[10] Zimmermann, Armin. "Generating diverse realistic data sets for episode mining." Data Mining Workshops (ICDMW), 2012 IEEE 12th International Conference on. IEEE, 2012.

[11] Zhou, Wenzhi, Hongyan Liu, and Hong Cheng. "Mining closed episodes from event sequences efficiently." Advances in Knowledge Discovery and Data Mining. Springer Berlin Heidelberg, 2010. 310-318.

[12] Slimani, Thabet, and Amor Lazzez. "Sequential mining: Patterns and algorithms analysis." arXiv preprint arXiv:1311.0350 (2013).

[13] Zimmermann, Armin. "Generating diverse realistic data sets for episode mining." Data Mining Workshops (ICDMW), 2012 IEEE 12th International Conference on. IEEE, 2012.

[14] Zhao, Qiankun, and Sourav S. Bhowmick. "Sequential pattern mining: A survey." ITechnical Report CAIS Nayang Technological University Singapore (2003): 1-26.

[15] Laxman, Srivatsan, P. S. Sastry, and K. P. Unnikrishnan. "A fast algorithm for finding frequent episodes in event streams." Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2007.