Résumé de Séquences Temporelles pour le passage à l'échelle d'applications dépendantes du temps

Quang-Khai Pham, Guillaume Raschia, Régis Saint-Paul, Boualem Benatallah, Noureddine Mouaddib

Résumé de Séquences Temporelles pour le Passage à l'Échelle d'Applications Dépendantes du Temps

Quang-Khai Pham[†‡], Guillaume Raschia[‡], Regis Saint-Paul[⋆],
Boualem Benatallah[†], Noureddine Mouaddib[‡]

[‡]LINA CNRS UMR 6241 - Atlas group
University of Nantes, France
{noureddine.mouaddib, guillaume.raschia}@univ-nantes.fr

[⋆]CREATE-NET
Trento 38100, Italy
regis.saint-paul@create-net.org

[†]CSE at UNSW
Sydney, NSW 2033, Australia
{qpham, boualem}@cse.unsw.edu.au

Abstract

Nous présentons dans ces travaux le concept du "Résumé de Séquence Temporelle" dont le but est d'aider les applications dépendantes du temps à passer à l'échelle sur de grandes masses de données. Un Résumé de Séquence Temporelle s'obtient en transformant une séquence d'événements où les événements sont ordonnés chronologiquement. Chaque événement est précisément décrit par un ensemble de labels. Le résumé produit est alors une séquence temporelle d'événements, plus concise que la séquence originale et pouvant se substituer à l'originale dans les applications. Nous proposons un algorithme appelé "TSaR" pour produire un tel résumé. TSaR se base sur les principes de généralisation, de regroupement et de formation de concept. La généralisation permet d'abstraire à l'aide de taxonomies les labels qui décrivent les événements. Le regroupement permet ensuite d'agglomérer les événements généralisés qui sont similaires. La formation de concept réduit le nombre d'événements groupés-généralisés dans la séquence en représentant chaque groupe formé par un unique événement. Le processus est conçu de manière à préserver la chronologie globale de la séquence d'entrée. L'algorithme TSaR produit le résumé de manière incrémentale et a une complexité algorithmique linéaire. Nous validons notre approche par un ensemble d'expériences sur une année d'actualités financières produites par Reuters.

Keywoards: Time sequences, Summarization, Taxonomies, Clustering

1 Introduction

Domains such as medicine, the WWW, business or finance generate and store on a daily basis massive amounts of data. This data is represented as a collection of time sequences of events where each

event is described as a set of descriptors taken from various descriptive domains and associated with a timestamp. These archives represent valuable sources of insight for analysts to browse, analyze and discover golden nuggets of knowledge. For instance, biologists could discover disease risk factors by analyzing patient history [28], web content producers and marketing people are interested in profiling client behaviors [24], traders investigate financial data for understanding global trends or anticipating market moves [30]. However, analysts are overloaded with the size of this data and increasingly need methods and tools allowing exploratory visualization, query or analysis.

As an example, Google has developed Google Finance [1]. In a user-defined timeline, Google Finance provides analysts with a tool to browse through companies' stock values while visualizing background information about the companies. This background information is provided in the form of a sequence of chronologically ordered news events that appeared at some interesting moments, e.g., during price jumps. We call applications, such as Google Finance, that rely on the chronological order of the data to be meaningful: *Chronology-dependent* applications.

In this context, we observed that sequences of events relating to an entity A occurring in a *short* period of time are likely to relate to a same topic, e.g., events about Lehman Brothers mid-September 2008 relate to its bankruptcy. This observation shows that it could be more practical and meaningful for the analyst to navigate in the chronology of events through *summarized* events that gather several events about a same topic, e.g., *Lehman Brothers's Backruptcy*, rather than the entire set of individual events. At the same time, the analyst should be given the possibility to browse the details of these summarized events for a more in-depth analysis. This existing example puts forward the need for a data representation where multiple events describing a same topic are grouped while preserving the overall chronology of events' topic.

During the past decade, semantic data summarization has been addressed in various areas such as databases, data warehouses, datastreams, etc., to represent data in a more concise form by using its semantics [10, 11, 14, 5, 13, 23]. However, including the *time* dimension into the summarization process is an additional constraint that requires the chronology of events' topic to somehow be preserved. For instance, the sequence $\langle$*Lehman Brothers's Bankruptcy*, *Lehman Brothers's Rescue*$\rangle$ only makes sense because the events related to the *Bankruptcy* need to occur before the *Rescue* can happen. We name this type of data transformation, based on the data's semantic content and temporal characteristics, *Time Sequence Summarization*. A time sequence summarizer should take as input a time sequence of events, where each event is described by a set of descriptors, and output a time sequence of summarized events. The produced summarized time sequence should have the following properties:

1. Brevity: The number of summarized events in the output time sequence should be reduced in comparison to the number of events in the input time sequence.

2. Substitution principle: A chronology-dependent application that performs on a time sequence of events should be capable of performing seamlessly if the input time sequence is replaced by its summary.

3. Informativeness: Summarization should reduce time sequen- ces of events in a way that keeps the semantic content available to and understandable by the analyst without the need for *desummarization*.

4. Accuracy and usefulness: The input time sequence of events should not be overgeneralized to preserve descriptive precision and keep the summarized time sequence useful. However, ensuring high descriptive precision of events in the summarized time sequence requires trading off the *brevity* property of the summary.

5. Chronology preservation: The chronology of summarized events in the output time sequence should reflect the *overall* chrono- logy of events in the input time sequence.

6. Computational scalability: Time sequence summaries are built to support chronology-dependent applications and, thus their construction should not become a bottleneck. Applications such as data mining might need to handle very large and long collections of time sequences of events, e.g., news feeds, web logs or market data, to rapidly discover knowledge. Therefore, the summarization process should have low processing and memory requirements.

Designing a time sequence summary that displays all these properties is a challenging task. There exists a bulk of work for designing *summaries* in different areas such as datastreams, transaction databases, event sequences or relational databases. However, to the best of our knowledge, this research corpus does not address simultaneously all six mentioned properties.

Contributions. In this paper, we present the concept of *Time Sequence Summarization* and propose a summarization technique that satisfies all six mentioned properties to support chronology-dependent applications. Our contributions are as follows:

• We give a formal definition of *Time Sequence Summarization*. The time sequence summary of a time sequence of events is a time sequence of summarized events where: (i) events occurring *close* in time and relating to a same topic are gathered into a same summarized event and (ii) each summarized event is represented by a *concept* formed from the underlying events. For example, assume the input time sequence of events is: $\langle$(t_1,*Easy subprime loan*),(t_2,*Interest rate increase*),(t_3,*Housing market collapse*)$\rangle$. A valid time sequence summary could be: $\langle$(t'_1,*Subprime crisis*)$\rangle$.

• We propose a *T*ime *S*equence Summ*aR*ization (TSaR) algorithm. TSaR relies on the ideas of Generalization and Merging, introduced by Han et al. for discovering knowledge in relational databases [10, 11]. TSaR is a 3-step process that uses background knowledge in the form of taxonomies, supposedly given by the analyst to generalize event descriptors at higher levels of abstraction.

Assuming events occurring *close* in time might relate to a same topic, grouping is performed on generalized events whose generalized descriptors are similar. TSaR's grouping process gathers generalized events in a way that respects the chronology of topics in the input time sequence. For this purpose, a *Temporal Locality* is defined so that temporally close events can be gathered. Temporal locality is a term borrowed from Operating systems research [8] and defined in Section 4.3. It can intuitively be understood as the fact that a series of events *close* in time have high probability of relating to a same topic. However, events relating to different topics might locally overlap in that period, e.g., due to network delays. Thus, defining a temporal locality allows TSaR to gather these overlapping events into their corresponding topics. Finally, each group is represented by a concept. In total, higher numerosity reduction can be achieved while the chronology of topics in the input time sequence is preserved.

TSaR summaries are built in an incremental way by processing an input time sequence of events in a one-pass manner. The algorithm has small memory and processing footprints. TSaR maintains in-memory a small structure that holds a limited number of grouped events, i.e., grouped events that fit into the temporal locality. TSaR's algorithmic complexity is linear with the number of events in the input time sequence.

We validate these characteristics with a set of experiments on real world data. We performed experiments using one year of English financial news events obtained from Reuters's. These archives contain after cleaning and preprocessing approximately 1.28M events split over 34458 time sequences. Each event in the time sequences is a set of words that precisely describes the content of the corresponding

news article. Our extensive set of experiments on summarizing this data shows that TSaR has (i) interesting numerosity reduction capabilities, e.g., compression ratio ranges from 10% to 82%, and (ii) low and linear processing cost.

Roadmap. The rest of the paper is organized as follows. Section 2 presents related work. Section 3 formalizes the concept of *Time Sequence Summarization*. Section 4 presents the novel technique we contribute in the paper. Section 5 discusses the experimentation we performed on financial news data. We conclude and discuss future work in Section 6.

2 Related work

Our work relates to lines of research, where a concise representation of massive data sources is desirable for storage or for knowledge discovery in constrained processing and memory environments. Related research domains are those where summaries are built from sequences of objects ordered by their time of occurrence. In this context, and in the light of the requirements mentioned in the introduction, we examine summarization techniques produced for datastreams, transaction databases and event sequen- ces. This study, however, do not encompass time series summarization as time series summarization rely on methods that only consider numerical data.

Datastreams. Datastreams is a domain characterized by data of infinite size and eventually generated at very high rates. The common assumptions are that (i) any processing should be performed in a single pass and (ii) input data can not be integrally stored. Such constraints have motivated researchers to represent input streams in a more concise form to support analysis applications, e.g., (approximate) continuous queries answering, frequent items counting, aggregation, clustering, etc.. Techniques proposed maintain in memory small structures, e.g., samples, histograms, quantiles or synopses, for streams of *numerical* data (we refer the reader to [9] for a more complete review). In contrast with numerical data that is defined on continuous and totally ordered domains, categorical descriptors are defined on discrete and partially ordered domains. Therefore, descriptors can not be handled and reduced in the same way as numerical data using conventional datastreaming summarization techniques, e.g, min/max/average functions.

To the best of our knowledge, small interest has been given to designing summaries[4, 23] for categorical datastreams. Aggarwal et al. [4] proposed a clustering method for categorical datastreams. The approach relies on the idea of *co-occurrence* of attribute values to build statistical summaries and gather input data based on these statistics. However, the clusters built do not reflect the chronology of the input data and require pre-processing before analysis. We proposed in previous work an approach to summarize datastreams using a conceptual summarization algorithm [20]. The summary produced does not reflect the chronology of the input data and can not be directly exploited by chronology-dependent applications.

Transaction database summarization. Transaction databases (TDB) are collections of time sequences where each time sequence is a list of chronologically ordered itemsets. A bulk of work [6, 25, 26, 29] has focused on creating summaries for transaction databases. Chandola et al.'s approach [6] and SUMMARY [26] are techniques that rely on closed frequent itemset mining to build an *informative* representation that covers the entire TDB. Building these summaries require multiple passes over the input TDB and the output is a set of frequent itemsets. This output does not endorse the substitution principal and does not reflect the chronology of transactions of the input TDB. HYPER [29] summarizes a TDB

as a set of hyperrectangles that covers the database. HYPER's output set of hyperrectangles requires preprocessing before chronology-dependent applications can exploit the summary and hyperrectangles are computed in polynomial time. But, designing a time sequence summarizer requires a single pass over the data and the output to preserve the chronology of transactions. Wan et al. [25] summarize a TDB into the compact form of a CT-tree specifically to support Sequential Pattern Mining (SPM). The output tree structure allows SPM to perform but looses the chronology of transactions of the input TDB.

Event sequence summarization. Kiernan and Terzi [16] rely on the Minimum Description Length (MDL) principle to produce in a parameter-free way a comprehensive summary of an event sequence, where events are taken from set $\mathcal{E}$ of m different event types. The authors segment the input event sequence timeline into k segments. Summarization is achieved by describing each segment S_i with a local model M_i that is a partition of $\mathcal{E}$ where groups $X_{ij} \in M_i$ gather event types of similar rate of appearance in S_i. Each event group $X_{ij} \in M_i$ is then associated with a probability of appearance $p(X_{ij})$ of X_{ij} in S_i. However, the output summary can not be directly piped to a chronology-dependent application and needs some form of *desummarization*. Hence, the authors' definition of summarization does not endorse the substitution principle and one can not seamlessly substitute the original event sequence for the summary.

The TSaR approach builds on top of the ideas in Attribute Oriented Induction (AOI) [10, 11]. The TSaR process is split into three sub-routines that allow input time sequences to be processed in an incremental way: (i) generalization, (ii) grouping and (iii) concept formation. Generalization transforms event descriptors into a more abstract but informative form. Then, grouping gathers generalized events that are semantically close, i.e., having similar generalized descriptors, and chronologically close. Grouping is performed in a way that preserves the overall chronology of topics in the time sequence. Finally, each formed group is represented by a concept, i.e., a set of descriptors, formed from the underlying events' descriptors. The output can then be directly interpreted by a human analyst or piped to any chronology-dependent application.

3 Time sequence summary

In this section, we give a running toy example to illustrate all the concepts presented in this paper. We also introduce the basic terminology used throughout the rest of the discussion and formalize the concept of *Time Sequence Summarization*.

3.1 Toy example

To illustrate the ideas exposed in this paper, we generate in Table 1 a simple toy example with a time sequence extracted from conference proceedings. The author N. Koudas is associated with a time sequence where each event is one publication timestamped by its date of presentation. For simplicity, the set of descriptors describing an event is taken from one single descriptive domain, namely, the paper's *topic*. Without loss of generality, this discussion is valid for any number of descriptive domains. This example is purposely unrelated to the application domain we provide in Section 5. It illustrates all the concepts introduced and shows the genericity of our approach.

Table 1. Time sequences of conference proceedings

Author	Date	Descriptors
N. Koudas	JUN05	$x_1 = \{$Datastreams, Aggregation$\}$
	AUG06	$x_2 = \{$Datastreams, Top-k query$\}$
	AUG06	$x_3 = \{$Top-k query$\}$
	SEP06	$x_4 = \{$Top-k query$\}$
	SEP06	$x_5 = \{$Join query, Selection query$\}$
	SEP07	$x_6 = \{$Clustering$\}$

3.2 Terminology

Let Ω be the universe of discourse, i.e., the set of all descriptors that could describe an event in a time sequence of events. $\Omega = \bigcup_{\mathcal{A}} D_A$ is organized into several descriptive domains D_A corresponding to each domain A that interests the analyst, e.g., the *topic* of research papers.

We refer to a part of Ω, i.e., a subset of descriptors taken from various descriptive domains, as $\mathcal{P}(\Omega)$. An *itemset* x is defined as an element in $\mathcal{P}(\Omega)$. Given an *object of interest* (e.g., "N. Koudas"), an *event e* is defined by an itemset x that describes e (e.g., $\{$*Datastreams, Aggregation*$\}$) and is associated with a *timestamp* t (e.g., $t =$"JUN05"). We assume the data input for time sequence summarization, also called *raw* data, is a collection of time sequences of events as defined in Definition 1.

Definition 1 (Time Sequence of events)
A time sequence of events $s = \langle (x_1, t_1), \ldots, (x_m, t_m) \rangle$*, also called* time sequence *for short, is a series of events* (x_j, t_j)*, with* $1 \leq j \leq m$ *and* $x_j \in \mathcal{P}(\Omega)$*, ordered by increasing timestamp* t_j*. We denote by* $S = \{x_1, \ldots, x_m\}$ *the support multi-set of* s*. A time sequence* s *verifies:* $\forall (x_j, x_k) \in S^2,\ j < k \Leftrightarrow t_j < t_k$*. We denote by* $s[T]$ *the set of timestamps of elements in* S*.*

This definition of a time sequence and the total order on timestamps allow us to equivalently write: $s = \langle (x_1, t_1), \ldots, (x_m, t_m) \rangle \Leftrightarrow s = \{(x_j, t_j)\}, 1 \leq j \leq m$

By convention, we further simplify the notation of a time sequence and note $s = \langle x_1, \ldots, x_m \rangle$ where each x_j, $1 \leq j \leq m$, is an itemset and all itemsets x_j are sorted by ascending index j. This simplification of the notation allows us to interchangeably use the term *event* to refer to the itemset x_j in event (x_j, t_j). We denote by $\mathcal{S}(\Omega)$ the set of time sequences in $\mathcal{P}(\Omega)$. This notion of time sequence can be generalized and used to define a sequence of time sequences that we hereafter call *second-order time sequence*. Second-order time sequences are more formally defined in Definition 2.

Definition 2 (Second-order time sequence)
A second-order time sequence defined on Ω *is a time sequence* $\overline{s} = \{(y_i, t'_i)\}$ *where each event* (y_i, t'_i) *is itself a regular time sequence of events defined on* Ω*. Events* (y_i, t'_i) *in* $\overline{s}$*, where* $y_i = \{(x_j, t_j)\}$*, are ordered thanks to the minimum timestamp value* $t'_i = \min\{t_j\}$*. The set of second-order time sequences defined on* Ω *is denoted* $S^2(\Omega)$*.*

An example of second-order time sequence from Table 1 for author *N. Koudas* can be defined as follows: $\overline{s} = \langle (y_1, t'_1), (y_2, t'_2), (y_3, t'_3)) \rangle$ where:

- y_1=$\langle (x_1$=$\{$Datastreams, Aggregation$\}$, t_1=JUN05$) \rangle$ and t'_1=t_1=JUN05
- y_2=$\langle (x_2, t_2), (x_3, t_3), (x_4, t_4), (x_5, t_5)) \rangle$ and $t'_2 = min\ \{t_2, \ldots, t_5\}$ i.e., t'_2=AUG06, where:
  $-x_2$=$\{$Datastreams, Top-k query$\}$ and t_2=AUG06

$-x_3$={Top-k query} and t_3=AUG06
$-x_4$={Top-k query} and t_4=SEP06
$-x_5$={Join query, Selection query} and t_5=SEP06
• y_3=$\langle(x_6$={Clustering}, t_6=SEP07)$\rangle$ and t'_3=t_6=SEP07

A second-order time sequence can be obtained from a time sequence s as defined in Definition 1 by the means of a form of *clustering* based on the semantics **and** temporal information of events $x_{i,j}$ in s. We refer the reader to the following surveys for more indepth on clustering [15, 27]. Reversely, a time sequence can be obtained from a second-order time sequence $\overline{s}$ by means of *concept formation* [18, 19] computed from time sequences y_i in $\overline{s}$.

3.3 Time sequence summary

We formally define in Definition 3 the concept of a time sequence summary using the concepts introduced previously.

Definition 3 (Time sequence summary)
Given a time sequence $s = \{(x_i, t_i)\} \in \mathcal{S}(\Omega)$, using clustering terminology, we define the time sequence summary of s, denoted $\chi(s) = (s^2_C, s^\star_M) \in \mathcal{S}^2(\Omega) \times \mathcal{S}(\Omega)$, as follows:

- *$s^2_C = \{(y_i, t'_i)\}$ is a second-order time sequence where events $y_i \in s^2_C$ are clusters obtained thanks to a form of clustering C that relies on events (x_i, t_i) semantic and temporal information.*
- *$s^\star_M = \{(x^\star_i, t'_i)\}$ is the time sequence of concepts $x^\star_i$ formed from clusters $y_i \in s^2_C$. M is the model chosen to characterize each cluster $y_i \in s^2_C$, i.e., to build the concepts.*

Hence, s^2_C and $s^\star_M$ can be understood as the extension *and the* intention*, respectively, of summary $\chi(s)$.*

We defined time sequence summarization using clustering terminology as the underlying ideas are similar, i.e., grouping objects based on their *proximity*. The novelty of time sequence summaries relies on the fact that events are clustered thanks to their semantic and temporal information. Conventional clustering methods mostly rely on the joint features of the objects considered and their proximity is evaluated thanks to a distance measure, e.g., based on entropy or semantic distances. Similarly, in time sequence summarization, a form of temporal approximation should also be applicable so that objects that are close from temporal view point are grouped. Consequently, local rearrangement of the objects on the timeline should also be allowed.

In a nutshell, the objective of time sequence summarization is to find the *best* method for grouping events based on their semantic content and their proximity on the timeline. This general definition of time sequence summarization can partially encompass some previous works such as Kiernan and Terzi's research on large event sequences summarization [16]. Indeed, the authors perform summarization by partitioning an event sequence S into k segments S_i, $1 \leq i \leq k$; this segmentation can be understood as organizing S into a second-order time sequence s^2_C where C is their segmentation method, e.g., Segment-DP. Note that the authors' segmentation method does not allow any form of event rearrangement on the timeline. Then, each segment S_i is described by a set of event type groups $\{X_{i,j}\}$ where each $X_{i,j}$ groups event types of similar appearance rate. Thus, the model used to describe each segment is a probabilistic model. To this point, our definition of a time sequence summary fully generalizes Kiernan and Terzi's

work. However, the authors add for each $X_{i,j}$ its probability of appearance $p(X_{i,j})$ in S_i. By doing so, the authors do not support the substitution principle and thus do not completely respect our definition of a time sequence summary.

4 The TSAR approach

In this section we present a *T*ime *S*equence Summ*aR*ization technique called TSaR. The basic principle of TSaR is illustrated in Figure 1. The idea is to gather events whose descriptors are similar at some high level of abstraction and that appear close in time. This is done in three steps: (i) reduce the data's domain of representation by generalizing descriptors to a user defined level of abstraction, (ii) group identical sets of descriptors within a certain sliding time window then (iii) represent each group with a single set of descriptors, a.k.a. concept.

In practice, the process is parametrized by three inputs: (i) domain specific taxonomies, (ii) a semantic accuracy parameter and (iii) a temporal precision parameter. The generalization process in phase 1 takes as input a time sequence, domain specific taxonomies and the user defined semantic accuracy parameter. It outputs a time sequence of generalized events where event descriptors are expressed at higher levels of taxonomy. This output is then fed to the grouping process in phase 2 where identical generalized events are grouped together. The overall chronology of events is preserved by grouping only generalized events present in a same temporal locality (as defined in Section 4.3). Phase 3 forms a concept to represent each group. Here, since all sets of descriptors in a group are identical, one instance of the group is selected to represent the group. We will detail these steps in the following sections.

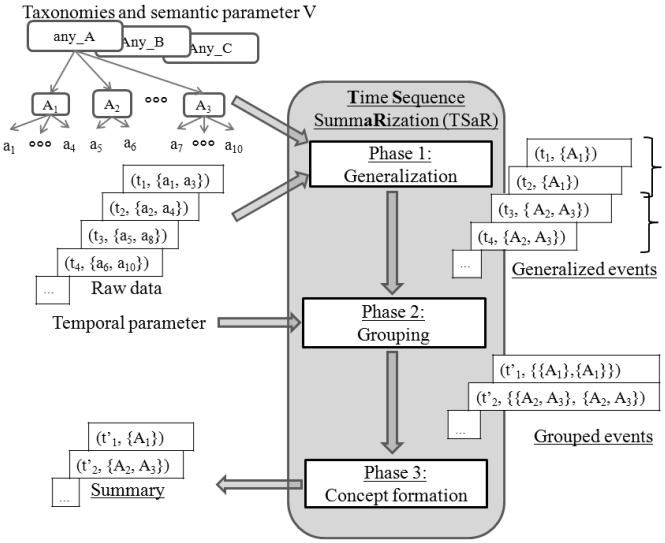

Figure 1. TSaR summarization process

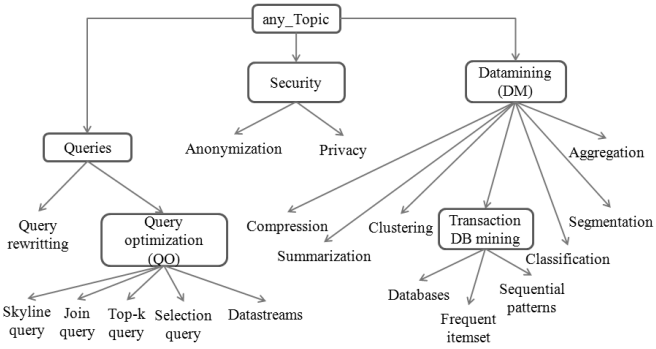

Figure 2. Taxonomy for the *topic* domain

4.1 Preliminaries

In this work, we assume that each descriptive domain D_A, on which event descriptors are defined, is structured into a taxonomy H_A that defines a generalization-specialization relationship between descriptors of D_A. The set of all taxonomies is denoted $\mathcal{H} = \bigcup_{\mathcal{A}} H_A$. The taxonomy H_A provides a partial

ordering $\prec_A$ over D_A and is rooted by the special descriptor any_A, i.e., $\forall a \in D_A, a \prec_A$ any_A. For convenience, we assume in the following that the descriptor any_A belongs to D_A.

The partial ordering $\prec_A$ on D_A defines a cover relation $<_A$ that corresponds to direct links between items in the taxonomy. Hence, we have $\forall (x, y) \in D_A^2,\ x \prec_A y \Rightarrow \exists (y_1, y_2, \ldots, y_k) \in D_A^k$ such that $x <_A y_1 <_A \ldots <_A y_k <_A y$. The length of the path from x to y is $\ell(x, y) = k + 1$. In other words, we need $k + 1$ generalizations to reach y from x in H_A. For example, given the *topic* taxonomy in Figure 2, it takes 2 generalizations to reach the concept *Queries* from the concept *Top-k query*.

In practice, we assume these taxonomies are available to the analyst. We believe this assumption is realistic as there exists numerous domain specific ontologies available, e.g., WordNet [3] or Onto-Med for medicine, as well as techniques that allow the automatic generation of taxonomies [17, 22].

4.2 Generalization phase

Here, we detail the generalization phase that uses taxonomies to represent the input data at higher levels of abstraction. We assumed values in Ω are partially ordered by the partial order relation $\prec_A$, so, thanks to this relation, we can define a containment relation $\sqsubseteq$ over subsets of Ω, i.e., $\mathcal{P}(\Omega)$.

$$\forall (x, y) \in \mathcal{P}(\Omega)^2, \quad (x \sqsubseteq y) \iff \big(\forall i \in x,\ \exists i' \in y,\ \exists A \in \mathcal{A},\ (i \prec_A i') \vee (i = i')\big)$$

For generalization, we need to replace event descriptors with upper terms of the taxonomies. We call *generalization vector* on Ω, denoted $\vartheta \in \mathbb{N}^i$, a list of integer values. ϑ defines the number of generalizations to perform for each descriptive domain in $\mathcal{A}$. We denote by $\vartheta[A]$ the generalization level for the domain A. Equipped with this generalization vector ϑ, we are now able to define a restriction $\sqsubseteq\downarrow_\vartheta$ of the containment relation above:

$$\forall (x, y) \in \mathcal{P}(\Omega)^2, \quad (x \sqsubseteq\downarrow_\vartheta y) \iff \Big(\forall i \in x, \exists i' \in y, \exists A \in \mathcal{A},\ (i \prec_A i') \wedge \big((\ell(i, i') = \vartheta[A]) \vee ((\ell(i, i') < \vartheta[A]) \wedge (i' = \texttt{any\_A}))\big)\Big)$$

Definition 4 (Generalization of a Time Sequence)
Given a generalization vector ϑ and a set of taxonomies $\mathcal{H}$, we define a parametric generalization function φ_ϑ that operates on a time sequence $s = \langle x_1, \ldots, x_n \rangle$ as follows:

$$\begin{array}{rccl} \varphi_\vartheta : & \mathcal{S}(\Omega) & \longrightarrow & \mathcal{S}(\Omega) \\ & s & \longmapsto & \varphi_\vartheta(s) = \langle x'_1, \ldots, x'_n \rangle \textit{ such that } \forall i \in \{1..n\},\ x_i \sqsubseteq\downarrow_\vartheta x'_i \end{array}$$

For example, given the *topic* taxonomy in Figure 2 the generalized version of Table 1 with a generalization vector $\vartheta[topic] = 1$ (also denoted $\vartheta = \langle 1 \rangle$ when all taxonomies should be generalized once) is shown in Table 2. We can notice that the $\sqsubseteq\downarrow_\vartheta$ relation also allows to reduce itemsets' cardinality. Indeed, *Datastreams* and *Top-k query* both generalize into *QO*. As a result, in N. Koudas's time sequence, the event {Datastreams, Top-k query} is generalized into {QO}.

From the analyst's view point, ϑ represents the semantic accuracy he desires for each descriptive domain. If he is interested in the minute details of a specific domain, e.g., a paper's *topic*, he can set ϑ to a low value, e.g., $\vartheta[topic] = 0$ or $\vartheta[topic] = 1$. Otherwise, he can set ϑ to higher values for a more abstract description of the domain. Once the input time sequence has undergone generalization, the output undergoes a grouping process as described in the following section.

Table 2. Generalized events with $\vartheta = \langle 1 \rangle$

Author	Date	Itemset
N. Koudas	JUN05	$x'_1 = \{\text{QO, DM}\}$
	AUG06	$x'_2 = \{\text{QO}\}$
	AUG06	$x'_3 = \{\text{QO}\}$
	SEP06	$x'_4 = \{\text{QO}\}$
	SEP06	$x'_5 = \{\text{QO}\}$
	SEP07	$x'_6 = \{\text{DM}\}$

Table 3. Grouped events with $w = 1$

Author	Date	Sequence
N. Koudas	JUN05	$y_1 = \{\{\text{QO, DM}\}\}$
	AUG06	$y_2 = \{\{\text{QO}\},\{\text{QO}\},\{\text{QO}\},\{\text{QO}\}\}$
	SEP07	$y_3 = \{\{\text{DM}\}\}$

4.3 Grouping phase

Here, we detail the grouping phase responsible for gathering generalized events. This phase relies on two concepts: (i) second-order time sequence as defined in Definition 2 and (ii) *Temporal locality*. We define this notion of temporal locality hereafter.

4.3.1 Temporal locality

In many research areas, it is assumed that a sequence of events generated *close* in time have high probability of relating to a same topic. This notion of *temporal locality* is borrowed from Operating systems research [8]. For example, $\langle$*Paper deadline*, *Authors notification*, *Camera-ready paper*$\rangle$ is a chronology of events describing the *Conference submission* topic. However, in a time sequence, events describing different topics can eventually be intertwined, e.g., due to network delays. Thus, defining a *temporal locality* for grouping events of a time sequence allows to capture the following notions: (i) sequentiality of events for a given topic and (ii) chronology between topics.

Temporal locality is measured as the time difference d_T, on a temporal scale T, between an incoming event and previously grouped events, hereafter called *groups* for short. The temporal scale may be defined directly through the timestamps or by a number of intermediate groups since we assume they are chronologically ordered. While grouping, an incoming event can only be compared to previous groups within a distance w ($d_T \leq w$). Indeed, w acts as a sliding window on the time sequence of groups and corresponds to the analyst's estimation of the temporal locality of events. In other words, w can be understood as the temporal precision loss the analyst is ready to tolerate for rearranging and regrouping incoming events.

This temporal window can be defined as a duration, e.g., $w =$ one month, or as a number of groups, e.g., $w = 2$. Defining w as a number of groups is useful in particular when considering bursty sequences, i.e., sequences where the arrival rate of events is uneven. The downside of this approach is the potential grouping of events distant in time. However, this limitation could be solved by defining w as both a duration and a constraint on the number of groups. As some domains, e.g. Finance, give more importance to the most recent information, we choose in our work to express our scale as a number of groups in order to handle bursts of events.

4.3.2 Grouping process

The grouping process is responsible for gathering generalized events relating to a same topic w.r.t. the temporal parameter w. It produces a second-order time sequence of groups defined as follows.

Definition 5 (Grouping of a time sequence)
Given a sliding temporal window w, we define a parametric grouping function ψ_w that operates on a time sequence s as follows:

$$\begin{array}{rccl} \psi_w: & \mathcal{S}(\Omega) & \longrightarrow & \mathcal{S}^2(\Omega) \\ & s & \longmapsto & \psi_w(s) = \langle (y_1 = \langle (x_1, t_1), \ldots, (x_{m_1}, t_{m_1}) \rangle, t'_1), \ldots, (y_n, t'_n) \rangle \end{array}$$

where n is the number of groups formed and such that:

(Part) $\forall k \in [w, n],\ \bigcup_{k-w \leq i \leq k} y_i \subseteq S$ *and* $y_i \cap y_j = \emptyset$ *when* $1 \leq i < j \leq n,\ j - i \leq w$
(Cont) $(x_{i,q} \sqsubseteq x_{i,r})\ ,\ 1 \leq r < q \leq m_i,\ 1 \leq i \leq n$
(TLoc) $\big(d_T(t_{i,q}, t'_i) \leq w\big),\ 1 \leq i \leq n,\ 1 \leq q \leq m_i$
(Max) $\forall x_{i,q} \in y_i,\ \forall x_{j,r} \in y_j, 1 \leq i < j \leq n,$
$(x_{i,q} \sqsubseteq x_{j,r}) \implies \big(d_T(t_{j,r}, t'_i) > w\big),\ 1 \leq q \leq m_i,\ 1 \leq r \leq m_j$

Property *(Part)* ensures that the support multi-set of w-contiguous time sequences in $\psi_w(s)$, e.g., $\langle y_1, \ldots, y_w \rangle$, $\langle y_2, \ldots, y_{w+1} \rangle$, etc ..., is a non-overlapping part of S. This is a direct consequence of grouping events that relate to a same topic within a same temporal locality. Property *(Cont)* gives a containment condition on events of every time sequence in $\psi_w(s)$. Given a time sequence y_i in $\psi_w(s)$, all events $x_{i,j} \in y_i$ are comparable w.r.t. the containment relation $\sqsubseteq$ and $x_{i,1}$ is the greatest event, i.e., $x_{i,1}$ contains all other events in y_i. Property *(TLoc)* defines a temporal locality constraint on events that are grouped into a same time sequence y_i in $\psi_w(s)$. *(TLoc)* ensures that y_i only groups events $(x_{i,j}, t_{i,j})$ that are within a distance d_T inferior to w from timestamp $t'_i = min(y_i[T])$, i.e., $d_T(t_{i,j}, t'_i) \leq w$. Property *(Max)* guaranties that the joint conditions *(Cont)* and *(TLoc)* are maximally satisfied.

In the grouping function ψ_w, the temporal locality parameter w controls how well the chronology of groups should be observed. When a small temporal window w is chosen, a very strict ordering of topics in the output time sequence is required. Incoming events can only be grouped with the latest groups. If $w = 1$ only contiguous events are eligible for grouping. Table 3 gives the expected output when performing grouping on Table 2 with $w = 1$. For example, note that in N. Koudas's generalized time sequence, when event $x'_2 = \{QO\}$ is considered, x'_2 can only be compared to previous group $y_1 = \{\{QO, DM\}\}$ for grouping.

When a large temporal window is chosen, the ordering requirement is relaxed. This means that a large number of groups can be considered for grouping for each incoming event. As a consequence, the minute details of the chronology of topics may be lost but higher numerosity reduction could be achieved.

The second-order time sequence output by the grouping function ψ_w can be understood as the extension of the summary, i.e., s^2_C as defined in Section 3.3. The intention of the summary, and thus the reduced version of the input time sequence, is obtained thanks to the concept formation phase as presented in the following section.

4.4 Concept formation phase

The concept formation phase is responsible for generating $s^\star_M$, the intention of the summary, from the time sequence of groups s^2_C obtained in the grouping phase. In the TSaR approach, we gather generalized events that have identical sets of descriptors. Therefore, this phase is straightforward.

Here, concept formation is achieved thanks to the projection operator π defined in Definition 6. Intuitively, π represents each time sequence y_i in second-order time sequence s^2_C by a single concept, called

representative event, x_k contained in y_i. Consequently, π produces from s^2_C a regular time sequence $s^\star_M = \pi(s^2_C)$ that is the intention of the summary, also called *representative sequence*. In addition, this operator is responsible for reducing the numerosity of events in the output time sequence, w.r.t. the original number of events in the input time sequence.

Definition 6 (Projection of a Second-Order Time Sequence)
We define the projection of a second-order time sequence $s^2_C = \langle (y_1 = \langle (x_{1,1}, t_{1,1}), \ldots, (x_{1,m_1}, t_{1,m_1}) \rangle, t'_1), \ldots, (y_n, t'_n) \rangle$ *as:*

$$\begin{array}{rccl} \pi: & \mathcal{S}^2(\Omega) & \longrightarrow & \mathcal{S}(\Omega) \\ & s^2_C & \longmapsto & \pi(s^2_C) = \langle (x_{1,1}, t_{1,1}), \ldots, (x_{n,n_1}, t_{n,n_1}) \rangle \end{array}$$

From our toy example in Table 3, N. Koudas's representative sequence is therefore:
$\pi(s^2_C) = \langle (\pi(y_1), t'_1), (\pi(y_2), t'_2), (\pi(y_3), t'_3) \rangle$ where:

- $\pi(y_1) = x^\star_1 = x_1 = \{QO, DM\}$ and $t'_1 = t_1$ = JUN05
- $\pi(y_2) = x^\star_2 = x_2 = \{QO\}$ and $t'_2 = t_2$ = AUG06
- $\pi(y_3) = x^\star_3 = x_6 = \{DM\}$ and $t'_3 = t_6$ = SEP07

4.5 The summarization process

In TSaR, summarization is achieved by the association of the three functions presented in the previous section, namely, the generalization, grouping and projection functions φ, ψ and π respectively. The summarization function is formally defined in Definition 7.

Definition 7 (*T*ime *S*equence Summ*aR*ization (TSaR) function)
Given a time sequence s *defined on* Ω*, a set of taxonomies* $\mathcal{H}$ *defined over* Ω*, a user defined generalization vector* ϑ *for taxonomies in* $\mathcal{H}$ *and a user defined sliding temporal window* w*, the summary of* s *is the combination of a generalization* φ_ϑ*, followed by a grouping* ψ_w *and a projection* π*:*

$$\begin{array}{rccl} \chi_{\vartheta,w}: & \mathcal{S}(\Omega) & \longrightarrow & \mathcal{S}^2(\Omega) \times \mathcal{S}^(\Omega) \\ & s & \longmapsto & \chi_{\vartheta,w}(s) = (s^2_C, s^\star_M) \textit{ where } s^2_C = \psi_w \circ \varphi_\vartheta(s) \textit{ and } s^\star_M = \pi(s^2_C) \end{array}$$

The association of the generalization and grouping function, φ and ψ respectively, outputs the extension of the summary, i.e., s^2_C. The reduced form of the summary, i.e., its intention $s^\star_M$, is a time sequence obtained by forming concepts from groups in s^2_C thanks to the projection operator π. The extension of the summary $\overline{s}_M$ then satisfies the conditions of the generalization-grouping process. The *(Cont)* property of ψ_w is then enforced by the generalization phase φ_ϑ of events in s. Note that every element in the reduced form of the summarized time sequence, i.e., $s^\star_M$, is an element of $\varphi_\vartheta(s)$. In other words, $s^\star_M$ is a representative subsequence of the generalized sequence of s.

From a practical view point, the analyst and applications are only given the intention form $s^\star_M$ of s's time sequence summary $\chi_{\vartheta,w}(s)$. Indeed, $s^\star_M$ is the most compact form of the summary. In addition, $s^\star_M$ is a time sequence that can seamlessly replace s and be directly processed by any chronology-dependent application that performs on s. Thus, $s^\star_M$ is the most useful form from application view point. In the following, we will interchangeably use the term *summary* to designate the intention $s^\star_M$ of a summary.

Let us give an illustration of a summary with our toy example. The representative sequences extracted from Table 3 are give Table 4. Here, we achieve the dual goal of numerosity reduction (from 6 events to

3) and domain reduction (from 6 descriptors to 2). These compression effects are obtained thanks to the user defined parameters, i.e., the generalization vector ϑ and the temporal sliding window w, that control the trade-off between resp. semantic accuracy vs. standardization and time accuracy vs. compression.

Table 4. Summary with $w = 1$

Author	Date	Itemset
N. Koudas	JUN05	$x_1^\star$ ={QO, DM}
	AUG06	$x_2^\star$ ={QO}
	SEP07	$x_3^\star$ ={DM}

4.6 The TSaR algorithm

From an operational view point, our implementation of TSaR is shown in Algorithm 1. The summary is computed through an incremental algorithm that generalizes and appends incoming events one at a time into the current output summary. In other words, assume the current summary is $s_M^\star = \chi_{\vartheta,w}(\langle x_1, \ldots, x_n \rangle) = \langle \pi(y_1), \ldots, \pi(y_j) \rangle$ and the incoming event is (x_{n+1}, t_{n+1}). The algorithm computes $\chi_{\vartheta,w}(\langle x_1, \ldots, x_n, x_{n+1} \rangle)$ with a local update to $s_M^\star$, i.e., changes are only made within the last w groups $y_{j-w}, \ldots, y_j$.

More precisely, (x_{n+1}, t_{n+1}) is generalized into (x'_{n+1}, t_{n+1}) (line 6). Then, assuming we denote $W = \{y_k\}$, $1 \leq j - w \leq k \leq j$ the set of groups that are included in temporal window w, TSaR checks if x'_{n+1} is included in a group $y_k \in W$. x'_{n+1} is either incorporated into a group y_k if its ϑ-generalized version satisfies the *(Cont)* condition (line 8 to 9), or it initializes a new group $\{(y_{j+1}, t'_{j+1} = t_{n+1})\}$ in W (line 11 to 14). Once all input events are processed, the last w groups contained in W are projected and added to the output summary $s_M^\star$ (line 18 to 20). The final output summary is then returned and/or stored in a database.

Memory footprint. TSaR requires w groups to be maintained in-memory for summarizing the input time sequence. The algorithm's memory footprint is finite ($O(1)$) and bound by the width of w and the average size m of an event's set of descriptors. The overall process memory footprint is obtained by adding the cost necessary to maintain in-memory the output time sequence $s_M^\star$ and the taxonomies and/or hashtable index to compute descriptors' generalization. However, $s_M^\star$ can be projected and written to disk at regular intervals. Therefore, TSaR's overall memory footprint remains constant and limited compared to the amount of RAM now available on any machine.

Processing cost. TSaR performs generalization, grouping and concept formation on the fly for each incoming event. The process has an algorithmic complexity linear with the number of events $O(n)$. The processing cost is weighted by a constant cost $c = a * b$. a is the cost for generalizing an event's set of descriptors and mainly depends on the number of taxonomies and their size. b is the cost to scan the finite list of groups in W. However, a is a cost that can be reduced by precomputing the generalization of each descriptive domain and storing the results in a hashtable index. b is a cost that is negligible since the temporal windows w used are small, e.g., mostly $w \leq 25$ in our experiments. Hence, we satisfy the memory and processing requirements presented earlier.

5 Experiments

In this section we validate our summarization approach through an extensive set of experiments on real-world data from Reuter's financial news archives. First, we describe the data and how taxonomies are acquired for the descriptive domains of the news. We summarize the raw data with different temporal windows w and show the following properties of the TSaR algorithm: (i) low processing cost, (ii) linearity and (iii) compression ratio.

Algorithm 1 TSaR's pseudo-code

1. INPUT: ϑ, taxonomies $\mathcal{H}$, w, time sequence of events s
2. LOCAL: W FIFO list containing the w last groups
3. OUTPUT: Summary $s^{\star}_{M}$
4. **for all** incoming event $(x_{n+1}, t_{n+1}) \in s$ **do**
5. {// Generalization using $\mathcal{H}$}
6. $x'_{n+1} \leftarrow \varphi_{\vartheta}(x_{n+1})$
7. {// Grouping}
8. **if** $\exists (y_k, t'_k) \in W,$ where $x'_{n+1} \sqsubseteq y_k$ **then**
9. $y_k \leftarrow y_k \cup x'_{n+1}$ {// x'_{n+1} is grouped into y_k}
10. **else**
11. **if** $|W| > w$ {// Case where W is full} **then**
12. Pop W's 1^{st} group y_{j-w}, add $\pi(y_{j-w})$ into $s^{\star}_{M}$
13. **end if**
14. $W \leftarrow W \cup \{(x'_{n+1}, t_{n+1})\}$ {// Updating W}
15. **end if**
16. **end for**
17. {// Add all groups in W into $s^{\star}_{M}$}
18. **while** $W \neq \emptyset$ **do**
19. Pop W's 1^{st} group y_i, add $\pi(y_i)$ into $s^{\star}_{M}$
20. **end while**
21. **return** $s^{\star}_{M}$

Our experiments were performed on a Core2Duo 2.0GHz laptop, 2GB of memory, 4200rpm hard drive and running Windows Vista Pro. The DBMS used for storage is PostgreSQL 8.0 and all code was written in C#.

5.1 Financial news data and taxonomies

n financial applications, traders are eager to discover knowledge and eventual relationships between live news feeds and market data in order to create new business opportunities [30]. Reuters has been generating and archiving such data for more than 10 years. To experiment and validate our approach in a real-world environment, we used one year of Reuters's news articles (2003) written in English. The unprocessed data set comes as a log of 21,957,500 entries where each entry includes free text and a set of ≈ 30 attribute-value pairs of numerical or categorical descriptors. An example of raw news event is given in Table 5. As provided by Reuters, the data can not be processed by the TSaR algorithm. Hence, the news data was cleaned and preprocessed into a sequence of events processable by TSaR.

Among all the information embedded in Reuters's news articles we focused on 3 main components for representing the archive as a time sequence:

• Timestamp: This value serves for ordering a news article within a time sequence.

• Topic_codes: When news articles are written, *topic_codes* are added to describe their content. There are in total 715 different codes relating to 20 different topics. We used 7 of the most popular topics to describe the data, i.e., {Location, Commodities, Economy Central Banking and Institution, Energy, Equities, Industrial sector, General news}.

• Free text: This textual content is a rich source of information from which precise semantic descriptors can be extracted. Give 5 additional topics, namely, {Business, Operation, Economics, Government, Finance}, we used the WordNet [3] ontology to extract additional descriptors from this content.

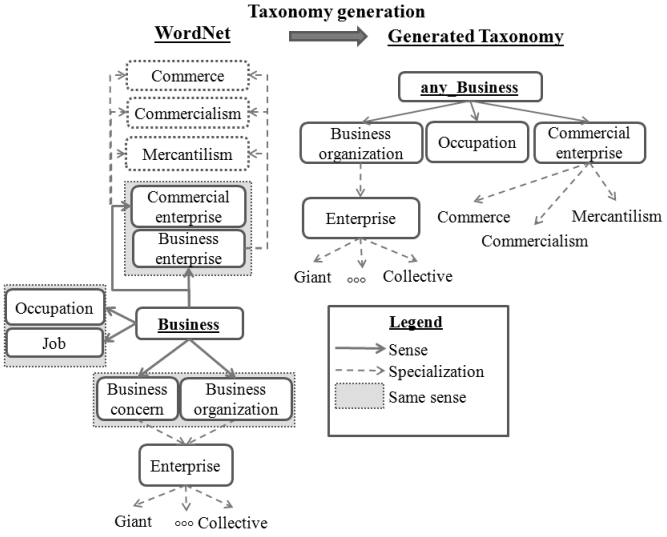

Figure 3. Taxonomy generated for the *business* domain

Table 5. Example of raw news event:

Timestamp	01 Jan 2004
Topic code	EUROPE USA Bank INS NL
Free text	Dutch bank ABN AMRO said on Wednesday it had reached a preliminary agreement to sell its U.S.-based Professional Brokerage business to Merrill Lynch & Co. …

Extracting pertinent descriptors from free text is a non trivial task w.r.t. the need for organizing the descriptors extracted into taxonomies. Research in Natural Language Processing (NLP) could be leveraged to tag texts based on their corpus, e.g., using Term Frequency-Inverse Document Frequency (TF-IDF) weights as done in [12] or using online resources such as Open Calais [2]. However, creating taxonomies from the tags extracted is not trivial and requires prior knowledge on the descriptors. The paradox lies in the fact that these descriptors are not known in advance. We chose to use the WordNet [3] ontology as a guide for extracting descriptors and structuring them into taxonomies thanks to the hierarchical organization already existing in WordNet. This choice leaves room for improvement by leveraging more complex techniques for both extracting descriptors from the free text and structuring these descriptors.

For example, automatic approaches [17] or hierarchical sources such as Wikipedia [22, 7] could also be used. However, such research is out of the scope of this paper.

In total, we preprocessed the input archive into a sequence of 1,283,277 news events.Each news event is described on the 12 descriptive domains selected earlier and several descriptors from each domain can be used. We generated a taxonomy for each of these descriptive domains. As the domains of topic_code themes are limited in size and already categorized, corresponding taxonomies were manually generated. Descriptive domains extracted from the free text were generated using the WordNet ontology as shown in Figure 3. In a nutshell, for a given subject, e.g., *business*, its senses are used as intermediary nodes in the taxonomy. If there are several synonyms for one sense, e.g., {*Commercial enterprise, Business enterprise*}, one is arbitrarily chosen, e.g., *Commercial enterprise*. Specialized descriptors are then used as lower level descriptors.

5.2 Summarization

The TSaR algorithm takes as input taxonomies H_A, a generalization vector ϑ, temporal window parameter w and a time sequence. The expected output is a more concise representation of the input sequence where the descriptive domains and the number of input events are reduced.

5.2.1 Quality measure

The quality of the algorithm can be evaluated by different methods. First, we could evaluate the summarization algorithm w.r.t. the application that it is meant to support, e.g., Sequential Pattern Mining (SPM). In this case, the summary can be evaluated based on its ability to increase the quality of the output knowledge or increase the speed of the mining process (a preliminary study is proposed in our technical report [21]).

Second, we can measure the semantic accuracy of summarized event descriptors in the summary produced. This accuracy can be evaluated thanks to the ratio $\alpha = \frac{|\overline{\Omega}'|}{|\overline{\Omega}|}$, where $\overline{\Omega}$ is the set of descriptors in the raw data and $\overline{\Omega}'$ is the set of generalized descriptors in the output summary. The higher α, the better the semantic accuracy.

In addition to the semantic accuracy of the summary, we can also measure its temporal accuracy. For this purpose, given a time sequence $s = \{x_i\}$, $1 \leq i \leq n$, temporal locality $w > 1$ and a temporal window W, we define a temporal rearrangement penalty cost for grouping an incoming event x_i with a group $y_j \in W$. We denote this penalty cost $\mathcal{C}_\tau(x_i)$. $\mathcal{C}_\tau(x_i)$ expresses the number of rearrangements necessary on the timeline so that event x_i can be grouped with y_i in window W. $\mathcal{C}_\tau(x_i)$ penalizes incoming events x_i that are grouped with the older groups y_i in W; on the other hand, if y_i is the most recent group in W, no penalty occurs. $\mathcal{C}_\tau(x_i)$ is formally defined as follows:

$$\begin{cases} \mathcal{C}_\tau = 0, & \text{if } (\nexists y_j \in W, x_i \sqsubseteq y_j), \text{ or, } (\exists y_j \in W, x_i \sqsubseteq y_j \text{ and } \nexists k > j, y_k \in W) \\ \mathcal{C}_\tau = m, 1 \leq m \leq w-1, & \text{if } \exists y_j \in W, x_i \sqsubseteq y_j \text{ and } m = |\{y_k \in W, k > j\}|. \end{cases}$$

The total temporal rearrangement penalty cost for summarizing s into $s^\star_M$, denoted $\mathcal{C}_\tau(s^\star_M)$, is then $\mathcal{C}_\tau(s^\star_M) = \sum_{i=1}^{n} \mathcal{C}_\tau(x_i)$. This penalty cost should then be normalized so that our results are comparable. Hence, we choose to compute the relative temporal accuracy of the summaries. We normalize all temporal rearrangement cost by the maximum cost obtained in our experiments, i.e., $\mathcal{C}_\tau(\chi_{\langle 3 \rangle,100}(s))$.

Finally, we also evaluate the summarization algorithm on its numerosity reduction capability by reporting its *compression ratio* CR. CR is defined as: $CR = 1 - \frac{|s^{\star}_M|-1}{|s|-1}$ where $|seq|$ is the number of events in a time sequence seq. The higher CR, the better. We decide to use CR as it was also used by Kiernan and Terzi's in [16] and we weight the CR with the summaries' semantic and temporal accuracy, α and β, respectively.

5.2.2 Experiment results

We start by setting the generalization vector $\vartheta = \langle 1 \rangle$, i.e., all descriptors are generalized once, and $w \in \{1, 2, 3, 4, 5, 10, 15, 20, 25, 50, 100\}$ where the maximum value $w = 100$ was chosen to represent a very strong temporal relaxation. Figure 4 gives the processing time of the TSaR algorithm with different temporal windows. For the sake of readability, we only display the plots for temporal windows $w \in \{1, 5, 100\}$. These plots show that the TSaR algorithm is linear in the number of input events for temporal windows w of any size. In addition, we can observe that processing times are almost constant whatever the temporal window considered. The slight variation observed in between different values of w have two complementary explanations. First, it is more costly to scan large windows during grouping. Second, a larger window w allows to maintain more groups in-memory and, so, requires less I/O operations for writing into storage.

Table 6. Semantic accuracy

ϑ	$\lvert\overline{\Omega}'\rvert$	α
$\langle 0 \rangle$	1208	N/A
$\langle 1 \rangle$	50	1
$\langle 2 \rangle$	20	0.40
$\langle 3 \rangle$	13	0.26

We compute the CR of the summaries built with different generalization vectors $\vartheta \in \{\langle 0 \rangle, \langle 1 \rangle, \langle 2 \rangle, \langle 3 \rangle\}$. The results are given in Figure 5. Note that the best compression ratio achieved with $\vartheta = \langle 0 \rangle$ is only 0.39. For a given temporal window w, relaxing the precision of the data by generalizing each descriptor once, twice or three times allows an average gain in compression capabilities of 46.15%, 94.87% and 133.33% respectively. In other other words, the compression ratio is approximatively doubled when increasing the generalization level. Another interesting observation is that for all ϑ, the plots show that highest numerosity reduction is achieved with larger temporal windows while processing times are almost constant, as shown in Figure 4. This observation is very helpful from user view point for setting the summarization parameters ϑ and w. In effect, as processing times are almost constant whatever the temporal window considered, the user needs only to express the desired precision in terms of (i) semantic accuracy for each descriptive domain and (ii) temporal locality without worrying about processing times.

Table 6 gives the semantic accuracy of the summaries produced and Figure 6 gives their temporal accuracy. Note in Table 6 that the number of descriptors in the raw data, i.e., $\vartheta = \langle 0 \rangle$, is 1208. When summarizing each descriptive domain once, i.e., $\vartheta = \langle 1 \rangle$, the number of descriptors in the summaries drops to 50. This loss of semantic information can be explained by the fact the data was preprocessed using the WordNet ontology and the taxonomies were also generated from the WordNet ontology. Numerous descriptors extracted from the free text are in fact synonyms and are easily generalized into one

common concept. Consequently, the concepts obtained with $\vartheta = \langle 1 \rangle$ should be considered as better descriptors than the raw descriptors. Hence, we choose to compute α using as baseline $|\overline{\Omega}'| = 50$, as shown in Table 6. In this case, each time ϑ is increased, the semantic accuracy is approximatively halved. This observation is consistent with our previous observation on the average compression gain.

Figure 6 gives the relative temporal accuracy of each summary. Higher levels of generalization reduce the temporal accuracy of the summaries. This phenomenon is due to the fact that more generic descriptors allow more rearrangements for grouping events. However, the temporal accuracy remains high, i.e., ≥ 0.80, for small and medium sized temporal windows, i.e., $w \leq 25$. The temporal accuracy only deteriorates with large windows, i.e., $w \geq 25$. This result means that the analyst can achieve high compression ratios without sacrificing the temporal accuracy of the summaries.

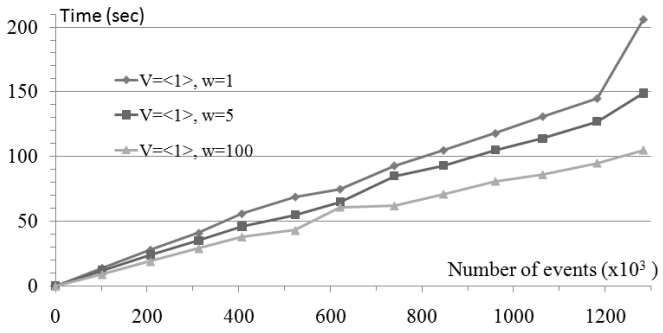

Figure 4. Processing time

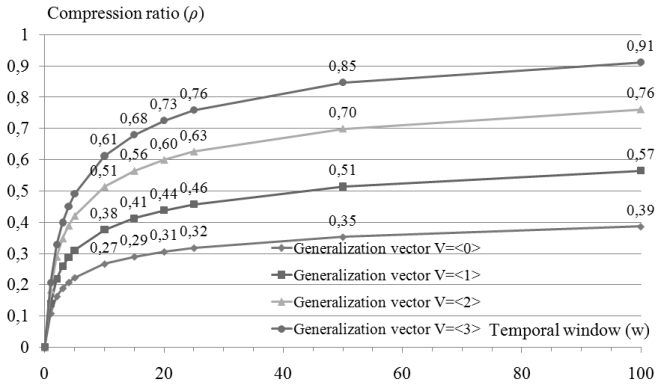

Figure 5. Numerosity reduction

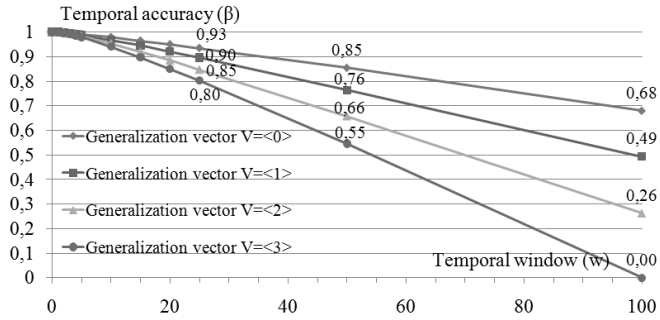

Figure 6. Temporal accuracy

However, guaranteeing the CR with TSaR is a difficult task, if not impossible, as its depend on the input parameters and on the data's distribution. In addition, the analyst needs to weight the semantic and temporal accuracy he is ready to trade off for higher CR. Guaranteeing the CR becomes an optimization problem that requires the algorithm to self-tune the input parameters and take into account the analyst's preferences.

6 Conclusion and future work

Massive data sources appear as collections of time sequences of events in a number of domains such as medicine, the WWW, business or finance. A concise representation of these time sequences of events is desirable to support chronology-dependent applications. In this paper, we have introduced the concept of *Time Sequence Summarization* to transform time sequences of events into a more concise but informative form, using the data's semantic and temporal characteristics.

We propose a *T*ime *S*equence Summ*aR*ization (TSaR) algorithm that transforms a time sequence of events into a more reduced and concise time sequence of events using a generalization, grouping and concept formation principle. TSaR expresses input event descriptors at higher levels of abstraction using taxonomies and reduces the size of time sequences by grouping *similar* events while preserving the overall chronology of events. The summary is computed in an incremental way and has an algorithmic complexity linear with the number of input events. The output is directly understandable by a human operator and can be used, without the need for *desummarization*, by chronology-dependent applications. One such application could be conventional mining algorithms to discover *high order knowledge*. We have validated our algorithm by performing an extensive set of experiments on one year of Reuters's financial news archives using our prototype implementation.

TSaR summaries are built using background knowledge in the form of taxonomies and the semantic and temporal precision of the output summary are controlled by user defined parameters. One direction in our future work is to render the generalization, grouping and concept formation process more flexible. We would like to allow automatic tuning of the input parameters with regard to an objective to achieve, e.g., a compression ratio. The problem then turns into an interesting optimization issue between semantic accuracy vs. standardization and time accuracy vs. compression. Also, much research in the temporal databases and datastreaming have worked under the assumption that analysts are more interested in recent data and desire high precision representations for new data items while older data can become obsolete. Works in temporal databases have introduced the concept of decay functions to model ageing data. We would also like to extend TSaR in future work by introducing decay functions to further reduce descriptive domains and data compression of older or obsolete information.

7 Acknowledgements

We would like to thank Sherif Sakr, Juan Miguel Gomez and Themis Palpanas for their many helpful comments on earlier drafts of this paper. This work was supported by the ADAGE project, the Atlas-GRIM group, the University of Nantes, the CNRS and the region Pays de la Loire.

References

[1] Google finance. http://finance.google.com.

[2] Open calais. http://www.opencalais.com.

[3] Wordnet. http://wordnet.princeton.edu/.

[4] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu. *On Clustering Massive Data Streams: A Summarization Paradigm*, volume 31 of *Advances in Database Systems*, pages 9–38. 2007.

[5] S. Babu, M. Garofalakis, and R. Rastogi. Spartan: A model-based semantic compression system for massive data tables. In *Proc. of SIGMOD*, 2001.

[6] V. Chandola and V. Kumar. Summarization - compressing data into an informative representation. In *Proc. of ICDM*, 2005.
[7] K. Chandramouli, E. Izquierdo, T. Kliegr, J. Nemrava, and V. Svatek. Wikipedia as the premiere source for targeted hypernym discovery. In *WBBT workshop at ECML/PKDD*, 2008.
[8] P. J. Denning. The locality principle. *Commun. ACM*, 48(7):19–24, 2005.
[9] M. M. Gaber, A. Zaslavsky, and S. Krishnaswamy. Mining data streams : A review. *SIGMOD*, 34(2), 2005.
[10] J. Han, Y. Cai, and N. Cercone. Knowledge discovery in databases: An attribute-oriented approach. In *Proc. of VLDB*, 1992.
[11] J. Han and Y. Fu. Exploration of the power of attribute-oriented induction in data mining. *Advances in Knowledge Discovery and Data Mining*, 1996.
[12] A. Hotho and G. Stumme. Conceptual clustering of text clusters. In *Proc. of FGML Workshop*, 2002.
[13] H. Jagadish, R. Ng, B. Ooi, and A. Tung. Itcompress: an iterative semantic compression algorithm. In *Proc. of ICDE*, 2004.
[14] H. V. Jagadish, J. Madar, and R. T. Ng. Semantic compression and pattern extraction with fascicles. In *Proc. of VLDB*, 1999.
[15] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computer Survey*, 31(3), 1999.
[16] J. Kiernan and E. Terzi. Constructing comprehensive summaries of large event sequences. In *Proc. of KDD*, 2008.
[17] K. Krishnapuram and K. Kummamuru. Automatic taxonomy generation: Issues and possibilities. In *Proc. of IFSA*, 2003.
[18] R. S. Michalski. Knowledge acquisition through conceptual clustering: A theoretical framework and algorithm for partitioning data into conjunctive concepts. *International Journal of Policy Analysis and Information Systems*, 4:219–243, 1980.
[19] R. S. Michalski and R. Stepp. *Learning from observation: conceptual clustering*. 1980.
[20] Q.-K. Pham, N. Mouaddib, and G. Raschia. Data stream synopsis using saintetiq. In *Proc. of FQAS*, 2006.
[21] Q.-K. Pham, R. Saint-Paul, B. Benetallah, G. Raschia, and N. Mouaddib. Time-aware content summarization of data streams. Technical Report TR-0722 (ftp://ftp.cse.unsw.edu.au/pub/doc/papers/UNSW/0722.pdf), 2007.
[22] S. P. Ponzetto and M. Strube. Deriving a large scale taxonomy from wikipedia. In *Proc. of AAAI Conference on Artificial Intelligence*, 2007.
[23] R. Saint-Paul, G. Raschia, and N. Mouaddib. General purpose database summarization. In *Proc. of VLDB*, 2005.
[24] J. Srivastava, R. Cooley, M. Deshpande, and P.-N. Tan. Web usage mining: Discover and applications of usage patterns from web data. *SIGKDD Explor. Newsl.*, 1(2):12–23, 2000.
[25] Q. Wan and A. An. Compact transaction database for efficient frequent pattern mining. In *Proc. of ICGC*, 2005.
[26] J. Wang and Karypis. On efficiently summarizing categorical databases. *Knowledge and Information Systems*, 9(1):19–37, 2006.
[27] T. Warren Liao. Clustering of time series data–a survey. *Pattern Recognition*, 38(11):1857–1874, 2005.
[28] A. Wright, T. N. Ricciardi, and M. Zwickc. Application of information-theoretic data mining techniques in a national ambulatory practice outcomes research network. In *Proc. of AMIA Annual Symposium*, 2005.
[29] Y. Xiang, R. Jin, D. Fuhry, and F. F. Dragan. Succint summarization of transactional databases: An overlapped hyperrectangle scheme. In *Proc. of KDD*, 2008.
[30] D. Zhang and K. Zhou. Discovering golden nuggets: data mining in financial application. *IEEE TSMC, Part C: Applications and Reviews*, 34:513–522, 2004.