Time Sequence Summaries to Scale Up Time-dependent Applications
Quang-Khai Pham, Guillaume Raschia, Noureddine Mouaddib, Régis Saint-Paul, Boualem Benatallah

To cite this version:
Quang-Khai Pham, Guillaume Raschia, Noureddine Mouaddib, Régis Saint-Paul, Boualem Benatallah. Time Sequence Summaries to Scale Up Time-dependent Applications. 18th ACM Conference on Information and Knowledge Management (CIKM), Nov 2009, Hong-Kong, China. pp.1137-1146, 10.1145/1645953.1646098. hal-00466864

HAL Id: hal-00466864
https://hal.archives-ouvertes.fr/hal-00466864
Submitted on 25 Mar 2010

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
ABSTRACT

In this paper, we present the concept of **Time Sequence Summarization** to support chronology-dependent applications on massive data sources. Time sequence summarization takes as input a time sequence of events that are chronologically ordered. Each event is described by a set of descriptors. Time sequence summarization produces a concise time sequence that can be substituted for the original time sequence in chronology-dependent applications. We propose an algorithm that achieves time sequence summarization based on a generalization, grouping and concept formation process. Generalization expresses event descriptors at higher levels of abstraction using taxonomies while grouping gathers similar events. Concept formation is responsible for reducing the size of the input time sequence of events by representing each group created by one concept. The process is performed in a way such that the overall chronology of events is preserved. The algorithm computes the summary incrementally and has reduced algorithmic complexity. The resulting output is a concise representation, yet, informative enough to directly support chronology-dependent applications. We validate our approach by summarizing one year of financial news provided by Reuters.

Categories and Subject Descriptors

E.4 [Coding and Information Theory]: Data Compression and compression; I.5.3 [Pattern Recognition]: Clustering—Algorithms

General Terms

Algorithms, Experimentation, Performance, Theory

Keywords

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 [27], web content producers and marketing people are interested in profiling client behaviors [23], traders investigate financial data for understanding global trends or anticipating market moves [29]. 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 some topic, e.g., events about Lehman Brothers' Bankruptcy, 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, data streams, etc., to represent data in a more concise form by using its semantics [10, 11, 14, 5, 13, 22]. However, including the time dimension into the summarization process is an additional constraint that requires the chronology of events’ topic to somehow be pre-
served. For instance, the sequence \textit{(Lehman Brothers’s Bankruptcy, Lehman Brothers’s Rescue)} 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, \textit{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 sequences 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 chronology 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 \textit{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 \textit{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: \(\langle t_1\text{,Easy subprime loan},t_2\text{,Interest rate increase},t_3\text{,Housing market collapse}\rangle\). A valid time sequence summary could be: \(\langle t'_1\text{,Subprime crisis}\rangle\).
- We propose a \textit{Time Sequence Summarization} (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 \textit{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 \textit{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 \textit{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 sequences. 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 syn-
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 generality of our approach.

<table>
<thead>
<tr>
<th>Author</th>
<th>Date</th>
<th>Descriptors</th>
</tr>
</thead>
<tbody>
<tr>
<td>N. Koudas</td>
<td>JUN05</td>
<td>(x_1 = {\text{Datasets, Aggregation}})</td>
</tr>
<tr>
<td>AUG06</td>
<td></td>
<td>(x_2 = {\text{Datasets, Top-k query}})</td>
</tr>
<tr>
<td>SEP06</td>
<td></td>
<td>(x_4 = {\text{Top-k query}})</td>
</tr>
<tr>
<td>SEP06</td>
<td></td>
<td>(x_5 = {\text{Join query, Selection query}})</td>
</tr>
<tr>
<td>SEP07</td>
<td></td>
<td>(x_6 = {\text{Clustering}})</td>
</tr>
</tbody>
</table>

### 3.2 Terminology

Let \(\Omega\) be the universe of discourse, i.e., the set of all descriptors that could describe an event in a time sequence of events. \(\Omega = \bigcup_D D_\lambda\) is organized into several descriptive domains \(D_\lambda\) corresponding to each domain \(\lambda\) that interests the analyst, e.g., the topic of research papers. We refer to a part of \(\Omega\), i.e., a subset of descriptors taken from various descriptive domains, as itemset \(x \in \mathcal{P}(\Omega)\). Given an object of interest (e.g., “N. Koudas”), an event \(e\) is defined by an itemset \(x\) that describes \(e\) (e.g., \(\{\text{Datasets, Aggregation}\}\)) and is associated with a timestamp \(t\), e.g., \(t = \text{“JUN05”}\). 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 \(s = \{(x_1, t_1), \ldots, (x_m, t_m)\}\), 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 the time sequence \(s\). A time sequence \(s\) verifies: \(\forall (x_j, x_k) \in S^2, j < k \Rightarrow t_j < t_k\). 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: \(s = \{(x_1, t_1), \ldots, (x_m, t_m)\} \iff s = \{(x_j, t_j)\}, 1 \leq j \leq m\).

By convention, we further simplify the notation of a time sequence and note \(s = \{x_1, \ldots, x_m\}\) where each \(x_j, 1 \leq j \leq m\), is an itemset and all itemsets \(x_j\) are sorted by ascending index \(j\). This

### 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 generality of our approach.

<table>
<thead>
<tr>
<th>Author</th>
<th>Date</th>
<th>Descriptors</th>
</tr>
</thead>
<tbody>
<tr>
<td>N. Koudas</td>
<td>JUN05</td>
<td>(x_1 = {\text{Datasets, Aggregation}})</td>
</tr>
<tr>
<td>AUG06</td>
<td></td>
<td>(x_2 = {\text{Datasets, Top-k query}})</td>
</tr>
<tr>
<td>SEP06</td>
<td></td>
<td>(x_4 = {\text{Top-k query}})</td>
</tr>
<tr>
<td>SEP06</td>
<td></td>
<td>(x_5 = {\text{Join query, Selection query}})</td>
</tr>
<tr>
<td>SEP07</td>
<td></td>
<td>(x_6 = {\text{Clustering}})</td>
</tr>
</tbody>
</table>

#### Table 1: Time sequences of conference proceedings

<table>
<thead>
<tr>
<th>Author</th>
<th>Date</th>
<th>Descriptors</th>
</tr>
</thead>
<tbody>
<tr>
<td>N. Koudas</td>
<td>JUN05</td>
<td>(x_1 = {\text{Datasets, Aggregation}})</td>
</tr>
<tr>
<td>AUG06</td>
<td></td>
<td>(x_2 = {\text{Datasets, Top-k query}})</td>
</tr>
<tr>
<td>SEP06</td>
<td></td>
<td>(x_4 = {\text{Top-k query}})</td>
</tr>
<tr>
<td>SEP06</td>
<td></td>
<td>(x_5 = {\text{Join query, Selection query}})</td>
</tr>
<tr>
<td>SEP07</td>
<td></td>
<td>(x_6 = {\text{Clustering}})</td>
</tr>
</tbody>
</table>
simplification of the notation allows us to interchangeably use the term \textit{event} to refer to the itemset \( x_i \) in event \((x_i, t_i)\). 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 \textit{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 \( \Omega \) is a time sequence \( \pi = \{(y_i, t'_i)\} \) where each event \((y_i, t'_i)\) is itself a regular time sequence of events defined on \( \Omega \). Events \((y_i, t'_i)\) in \( \pi \), 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 \( \Omega \) is denoted \( \mathcal{S}^2(\Omega) \).

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

- \( y_1 = \{(x_1)\} \) (Datasets, Aggregation), \( t'_1 = \text{JUN05} \)
- \( y_2 = \{(x_2, t_2), (x_3, t_3), (x_4, t_4), (x_5, t_5)\} \) and \( t'_2 = \min\{t_2, . . . , t_5\} \) i.e., \( t'_2 = \text{AUG06} \), where:
  - \( x_2 \) is [Datasets, Top-k query] and \( t_2 = \text{AUG06} \)
  - \( x_3 \) is [Top-k query] and \( t_3 = \text{AUG06} \)
  - \( x_4 \) is [Top-k query] and \( t_4 = \text{SEP06} \)
  - \( x_5 \) is [Join query, Selection query] and \( t_5 = \text{SEP07} \)
- \( y_3 = \{(x_2)\} \) (Clustering), \( t'_3 = \text{SEP07} \) and \( t'_3 = \text{SEP07} \)

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 \( x_i, t_i \) in \( s \). We refer the reader to the following surveys for more indepth on clustering [15, 26]. Reversely, a time sequence can be obtained from a second-order time sequence \( \pi \) by means of concept formation [18, 19] computed from time sequences \( y_i \) in \( \pi \).

### 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 \( \chi(s) = (s^C, s^M) \in \mathcal{S}^2(\Omega) \times \mathcal{S}(\Omega) \), as follows:

- \( s^C = \{(y_i', t'_i)\} \) is the second-order time sequence where events \( y_i \) in \( s^C \) are clusters thanks to a form of clustering \( C \) that relies on the events \((x_i, t_i)\) semantic and temporal information.

- \( s^M = \{x^*_i, t'_i\} \) is the time sequence of concepts \( x^*_i \) formed from clusters \( y_i \in s^C \). \( M \) is the model chosen to characterize each cluster \( y_i \in s^C \), i.e., to build the concepts.

Hence, \( s^C \) and \( s^M \) can be understood as the extension and the intention, respectively, of the summary \( \chi(s) \).

We defined time sequence summarization using clustering terminology as the underlying ideas are similar, i.e., grouping objects based on their \textit{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 \( s^C \) 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 Time Sequence Summarization 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.

![Figure 1: TSaR summarization process](image-url)

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 ex-
pressed 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 local-
ity (as defined in Section 4.3). Phase 3 forms a concept to represent
each group. Hence, since all sets of descriptors in a group are iden-
tical, one instance of the group is selected to represent the group.
We will detail these steps in the following sections.

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
\( H_A \in \mathcal{H} = \bigcup A H_A \), defining a generalization-specialization rela-
tion between descriptors of \( D_A \). The taxonomy \( H_A \) provides a
partial ordering \( \prec_A \) on \( D_A \) and is rooted by the special descrip-
tor \( \text{any}_A \), i.e., \( \forall a \in D_A, a \prec_A \text{any}_A \). For convenience,
we assume in the following that the descriptor \( \text{any}_A \) belongs to
\( D_A \).

The partial ordering \( \prec_A \) on \( D_A \) defines a cover relation \( \prec \), 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 \( x \prec_A y_1 \prec_A \ldots \prec_A y_k \prec_A y \). 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 an-
alyist. We believe this assumption is realistic as there exists nu-
erous domain specific ontologies available, e.g., WordNet [3] or
Onto-Med for medicine, as well as techniques that allow the auto-
matic generation of taxonomies [17, 21].

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 \( \Omega \) are partially ordered by the partial order relation \( \prec \).
We define a containment relation \( \subseteq \) over subsets of \( \Omega \), i.e., \( \mathcal{P}(\Omega) \),
\[
\forall (x, y) \in \mathcal{P}(\Omega)^2, (x \subseteq y) \iff \exists i \in x, \exists i' \in y, \exists A \in \mathcal{A}, (i \prec_A i') \lor (i = i')
\]
For generalization, we need to replace event descriptors with up-
top terms of the taxonomies. We call generalization vector on \( \Omega \),
denoted \( \vartheta \in \mathbb{N}^\mathbb{A} \), a list of integer values. \( \vartheta \) 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 \( \vartheta \), we are now able to define a re-
striction \( \vartheta \) of the containment relation above:
\[
\forall (x, y) \in \mathcal{P}(\Omega)^2, (x \subseteq \vartheta y) \iff \exists i \in x, \exists i' \in y, \exists A \in \mathcal{A}, (i \prec_A i') \land ((\ell(i,i') = \vartheta[A]) \lor ((\ell(i,i') < \vartheta[A]) \land (i' = \text{any}_A)))
\]

4.3 Grouping phase

Here, we detail the grouping phase responsible for gathering
generalized events. This phase relies on two concepts: (i) second-
time order sequence as defined in Def. 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 Operat-
ings systems research [8]. For example, (Paper deadline, Authors
notification, Camera-ready paper) is a chronology of events de-
scribing the Conference submission topic. However, in a time se-
cence, events describing different topics can eventually be inter-
twined, e.g., due to network delays. Thus, defining a temporal lo-
cality 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 \( dt \), on a
temporal scale \( T \), between an incoming event and previously group-
ed events, hereafter called groups for short. The temporal scale may
be defined directly through the timestamps or by a number of inter-
mediate groups since we assume they are chronologically ordered.
While grouping, an incoming event can only be compared to pre-
vious groups within a distance \( w (dt \leq w) \). 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 an-
alyis is ready to tolerate for regrouping 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 defini-
ing \( w \) 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.

**Definition 4 (Generalization of a Time Sequence)**

Given a generalization vector \( \vartheta \) and a set of taxonomies \( \mathcal{H} \),
we define a parametric generalization function \( \varphi_\vartheta \) that operates on a
time sequence \( s = \langle x_1, \ldots, x_n \rangle \) as follows:
\[
\varphi_\vartheta : S(\Omega) \longrightarrow S(\Omega)
\]
\[
s \longmapsto \varphi_\vartheta(s) = \langle x_1, \ldots, x_n \rangle
\]
such that \( \forall i \in \{1, \ldots, n\}, x_i \subseteq \vartheta x_i \)

For example, given the topic taxonomy in Figure 2 the gen-
eralized version of Table 1 with a generalization vector \( \vartheta = (1) \) is
shown in Table 2. We can notice that the \( \subseteq \vartheta \) relation also allows to
reduce itemsets’ cardinality. Indeed, Datstreams and Top-k query
both generalize into \( QO \). As a result, in N. Koudas’s time sequence,
the event \{Datstreams, Top-k query\} is generalized into \{QO\}.

From the analyst’s view point, \( \vartheta \) represents the semantic accu-
racy 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 \( \vartheta \) to a low value, e.g., \( \vartheta[\text{topic}] = 0 \) or \( \vartheta[\text{topic}] = 1 \). Otherwise,
he can set \( \vartheta \) 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.
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 \( \psi_w \) that operates on a time sequence \( s \) as follows:

\[
\psi_w : S(\Omega) \rightarrow S^2(\Omega)
\]

such that:

- \((Part)\) \; \forall k \in [w, n], \; \bigcup_{k-w \leq i \leq k} y_i \subseteq S \; \text{and} \; y_i \cap y_j = \emptyset

- \((Cont)\) \; (x_{i,q} \subseteq x_{i,r}), \; 1 \leq r < q \leq m_i, \; 1 \leq i \leq n

- \((TLoc)\) \; \langle d_T(x_{i,q}, t_i') \leq w \rangle, \; 1 \leq i \leq n, \; 1 \leq q \leq m_i

- \((Max)\) \; \forall x_{i,q} \in y_i, \; \forall x_{i,r} \in y_j, \; 1 \leq i < j \leq n, \; (x_{i,q} \subseteq x_{i,r}) \rightarrow \langle d_T(x_{i,r}, t_i') > w \rangle, \; 1 \leq q \leq m_i, \; 1 \leq r < m_j

Property \((Part)\) ensures that the support multi-set of \( w \)-contiguous time sequences in \( \psi_w(s) \), e.g., \( \{y_1, \ldots, y_s\}, \{y_2, \ldots, y_{s+w}\}, \ldots \) 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 \( \subseteq \) and \( x_{i,j} \) is the greatest event, i.e., \( x_{i,j} \) 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 \( (x_{i,j}, t_i') \) that are within a distance \( d_T \) inferior to \( w \) from timestamp \( t_{i'} = \min(y_i[T]) \), i.e., \( d_T(x_{i,j}, t_{i'}) \leq w \). Property \((Max)\) guarantees that the joint conditions \((Cont)\) and \((TLoc)\) are maximally satisfied.

In the grouping function \( \psi_w \), 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 \( \psi_w \) 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 \( s^1_M \), 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.

<table>
<thead>
<tr>
<th>Author</th>
<th>Date</th>
<th>Itemset</th>
</tr>
</thead>
<tbody>
<tr>
<td>N. Koudas</td>
<td>JUN05</td>
<td>{QO,DM}</td>
</tr>
<tr>
<td>AUG06</td>
<td></td>
<td>{QO}</td>
</tr>
<tr>
<td>SEP06</td>
<td></td>
<td>{QO}</td>
</tr>
<tr>
<td>SEP07</td>
<td></td>
<td>{DM}</td>
</tr>
</tbody>
</table>

**Table 2: Generalized events with \( \phi = \{1\} \)**

<table>
<thead>
<tr>
<th>Author</th>
<th>Date</th>
<th>Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>N. Koudas</td>
<td>JUN05</td>
<td>{QO,DM}</td>
</tr>
<tr>
<td>AUG06</td>
<td></td>
<td>{QO}</td>
</tr>
<tr>
<td>SEP07</td>
<td></td>
<td>{DM}</td>
</tr>
</tbody>
</table>

**Table 3: Grouped events with \( w = 1 \)**

**Table 4: Summary with \( w = 1 \)**

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 \( \varphi, \psi \) and \( \pi \) respectively. The summarization function is formally defined in Definition 7.

**Definition 6 (Projection of a Second-Order Time Sequence)**

We define the projection of a second-order time sequence \( s^2_C \) as:

\[
\pi : S^2(\Omega) \rightarrow S(\Omega)
\]

\[
s^2_C \rightarrow \pi(s^2_C) = (⟨x_{1,1},t_{1,1}⟩, ..., ⟨x_{n,m},t_{n,m}⟩)
\]

From our toy example in Table 3, N. Koudas’s representative sequence is therefore:

\[
\pi(s^2_C) = (⟨(x_1,y_1), (x_2,y_2), (x_3,y_3), (x_4,y_4)⟩)
\]

where:

- \( \pi(y_1) = x_1' = \{QO,DM\} \) and \( t_{1} = JUN05 \)
- \( \pi(y_2) = x_2' = \{QO\} \) and \( t_{2} = AUG06 \)
- \( \pi(y_3) = x_3' = x_6 = \{DM\} \) and \( t_{3} = SEP07 \)

4.5.1 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 \( \varphi, \psi \) and \( \pi \) respectively. The summarization function is formally defined in Definition 7.

**Definition 7 (Time Sequence SummaRization (TSaR) function)**

Given a time sequence \( s \) defined on \( \Omega \), a set of taxonomies \( H \) defined over \( \Omega \), a user defined generalization vector \( \vartheta \) for taxonomies in \( H \) and a user defined sliding temporal window \( w \), the summary of \( s \) is the combination of a generalization \( \varphi_{\vartheta} \), followed by a grouping \( \psi_w \) and a projection \( \pi \):

\[
\chi_{\vartheta,w} : S(\Omega) \rightarrow S^2(\Omega) \times S^3(\Omega)
\]

\[
s \mapsto \chi_{\vartheta,w}(s) = (s^2_C, s^3_M)
\]

where \( s^2_C = \psi_w \circ \varphi_{\vartheta}(s) \) and \( s^3_M = \pi(s^2_C) \).
The association of the generalization and grouping function, \( \varphi \) and \( \psi \) respectively, outputs the extension of the summary, i.e., \( s_M^x \). The reduced form of the summary, i.e., its intention \( s_M^I \), is a time sequence obtained by forming concepts from groups in \( s_M^x \), thanks to the projection operator \( \pi \). The extension of the summary \( s_M^x \) then satisfies the conditions of the generalization-grouping process. The (Cont) property of \( \psi_w \) is then enforced by the generalization phase \( \varphi_w \) of events in \( s \). Note that every element in the reduced form of the summarized time sequence, i.e., \( s_M^x \), is an element of \( \varphi_w(s) \). In other words, \( s_M^x \) is a representative subsequence of the generalized sequence of \( s \).

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

Let us give an illustration of a summary with our toy example. The representative sequences extracted from Table 3 are given 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 \( \vartheta \) and the temporal sliding window \( w \), that control the trade-off between resp. semantic accuracy vs. standardization and time accuracy vs. compression.

### 4.6 The TSaR algorithm

From an operational viewpoint, 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 \( s_M^I = \chi_{\vartheta,w}(x_1, \ldots, x_n) = (\pi(y_1), \ldots, \pi(y_j)) \) and the incoming event is \((x_{n+1}, t_{n+1})\). The algorithm computes \( \chi_{\vartheta,w}(x_1, \ldots, x_n, x_{n+1}) \) with a local update to \( s_M^I \), 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 \( \vartheta \)-generalized version satisfies the (Cont) condition (line 8 to 9), or it is initialized as 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 \( s_M^I \) (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 bounded 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 \( s_M^x \) and the taxonomies and/or hashable index to compute descriptors’ generalization. However, \( s_M^x \) 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 \( b \) is the cost to scan the finite list of groups in \( W \).

#### Algorithm 1 TSaR’s pseudo-code

1. INPUT: \( \vartheta, \) taxonomies \( \mathcal{H}, w, \) time sequence of events \( s \)
2. LOCAL: \( W \) FIFO list containing the \( w \) last groups
3. OUTPUT: Summary \( s_M^I \)
4. for all incoming event \((x_{n+1}, t_{n+1}) \in s \) do
   5. \(/ / \text{Generalization using } \mathcal{H} \)
      \[ x'_{n+1} \leftarrow \varphi_w(x_{n+1}) \]
   6. \(/ / \text{Grouping} \)
      \[ s_M \leftarrow \psi_w(x'_{n+1}) \]
   7. \(/ / \text{Case where } W \text{ is full} \)
      \[ \text{if } |W| \leq w \)
   8. \(/ / \text{Add all groups in } W \text{ into } s_M^I \)
      \[ W \leftarrow W \cup \{(x'_{n+1}, t_{n+1})\} \]
   9. \(/ / \text{Updating } W \)
   10. end if
11. end for
12. end if
13. end while
14. return \( s_M^I \)

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 hashable 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 Reuters’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.

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

In 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 [29]. 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’ 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 \( \approx 30 \) 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:
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 [21, 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 and the number of input events is reduced.

<table>
<thead>
<tr>
<th>Table 5: Example of raw news event:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Timestamp</strong></td>
</tr>
<tr>
<td><strong>Topic code</strong></td>
</tr>
<tr>
<td><strong>Free text</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 6: Semantic accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \vartheta )</td>
</tr>
<tr>
<td>(0)</td>
</tr>
<tr>
<td>(1)</td>
</tr>
<tr>
<td>(2)</td>
</tr>
<tr>
<td>(3)</td>
</tr>
</tbody>
</table>

5.2 Summarization

The TSaR algorithm takes as input taxonomies \( H_A \), a generalization vector \( \vartheta \), 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 several 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).

Second, we can measure the semantic accuracy of summarized event descriptors in the summary produced. This accuracy can be evaluated thanks to the ratio \( \alpha = \frac{\vartheta}{|\Omega|} \), where \( \vartheta \) is the set of descriptors in the raw data and \( |\Omega| \) is the set of generalized descriptors in the output summary. The higher \( \alpha \), 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 \( C_r(x_i) \). \( C_r(x_i) \) expresses the number of rearrangements necessary on the timeline so that event \( x_i \) can be grouped with \( y_j \) in window \( W \). \( C_r(x_i) \) penalizes incoming events \( x_i \) that are grouped with the older groups \( y_j \) in \( W \); on the other hand, if \( y_j \) is the most recent group in \( W \), no penalty occurs. \( C_r(x_i) \) is formally defined as follows:

\[
C_r = 0, \quad \text{if } (\exists y_j \in W, x_i \subseteq y_j) \vee (\exists y_j \in W, x_i \subseteq y_j \land \exists k > j, y_k \in W) \\
C_r = m, \quad \text{if } \exists y_j \in W, x_i \subseteq y_j \land 1 \leq m \leq w - 1, \quad m = |\{y_k \in W, k > j\}|
\]

The total temporal rearrangement penalty cost for summarizing \( s \) into \( s_\delta^m \), denoted \( C_r(s_\delta^m) \), is then \( C_r(s_\delta^m) = \sum_{i=1}^{n} C_r(x_i) \). 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., \( C_r(x_\delta^{100}(s)) \).

Finally, we also evaluate the summarization algorithm on its numerosity reduction capability by reporting its compression ratio \( CR \). \( CR \) is defined as:

\[
CR = 1 - \frac{|s_\delta^m|}{|s|}
\]

where \( |s| \) is the number of events in a time sequence \( s \). The higher \( CR \), the bet-
5.2.2 Experimental results

We start by setting the generalization vector $\vartheta = (1)$, 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.

We compute the CR of the summaries built with different generalization vectors $\vartheta \in \{\{0\}, \{1\}, \{2\}, \{3\}\}$. The results are given in Figure 5. Note that the best compression ratio achieved with $\vartheta = \{0\}$ 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 word, the compression ratio is approximatively doubled when increasing the generalization level. Another interesting observation is that for all $\vartheta$, 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 $\vartheta$ 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., $\vartheta = \{0\}$, is 1208. When summarizing each descriptive domain once, i.e., $\vartheta = \{1\}$, 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 $\vartheta = \{1\}$ should be considered as better descriptors than the raw descriptors. Hence, we choose to compute $\alpha$ using as baseline $|\Omega| = 50$, as shown in Table 6. In this case, each time $\vartheta$ 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., $\geq 0.80$, 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.

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 Time Sequence Summarization (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 aging 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.

8. REFERENCES