Time-Aware Content Summarization of Data Streams

Quang-Khai Pham[12], Régis Saint-Paul[1], Boualem Benatallah[1]
Guillaume Raschia[2], Noureddine Mouaddib[2]
[1]School of Computer Science and Engineering
The University of New South Wales
Sydney, Australia
{qpham|regiss|boualem}@cse.unsw.edu.au

[2]Atlas Group - LINA
University of Nantes
Nantes, France
{guillaume.raschia|noureddine.mouaddib}@univ-nantes.fr

Technical Report
UNSW-CSE-TR-0722

December 2007

THE UNIVERSITY OF
NEW SOUTH WALES

School of Computer Science and Engineering
The University of New South Wales
Sydney 2052, Australia

Abstract

Major media companies such as The Financial Times, the Wall Street Journal or Reuters generate huge amounts of textual news data on a daily basis. Mining frequent patterns in this mass of information is critical for knowledge workers such as financial analysts, stock traders or economists. Using existing frequent pattern mining (FPM) algorithms for the analysis of news data is difficult because of the size and lack of structuring of the free text news content. In this article, we propose a Time-Aware Content Summarization algorithm to support FPM in financial news data. The summary allows a concise representation of large volume of data by taking into account the expert's peculiar interest. The summary also preserves the news arrival time information which is essential for FPM algorithms. We experimented the proposed approach on Reuters news data and integrated it into the **S**treaming **TE**mpor**A**l **D**ata (STEAD) analysis framework for interactive discovery of frequent pattern.

1 Introduction

Major media companies such as the Financial Times, the Wall Street Journal or Reuters generate huge amounts of textual news data on a daily basis. This data represents a valuable source of insight for specialists such as stock traders, social scientists and economists [25]. Frequent Pattern Mining (FPM) consists in identifying sequences of events, here called *patterns*, that occur over time for a significant number of *objects* of interest. For example, a pattern could be "a salary increase" followed by "a house loan subscription" observed for a number of bank customers. In this example, the objects of interest are customers and each customer history is a distinct time sequence. A pattern is said *frequent* if it is observed in more than a given number of time sequences.

The discovery of frequent patterns is of importance since the fact that several events are frequently observed in sequence may indicate a *correlation* or *causal dependency* among the events. By observing the first few events in a frequent pattern, an expert may be able to anticipate future events; in our example, the bank could advertise interesting home loans to a customer who just got a salary increase. While frequent pattern mining is an established field [2, 19, 24, 13, 3, 20, 23], its application to financial news is challenging for the following reasons:

Different experts will have different perspectives on the same data. A given news relates to potentially many objects of interest and may be interpreted differently depending on the expert's interest. For instance, consider the takeover of Arcelor by Mittal Steel in 2006. A stock trader could be interested in finding patterns affecting companies in similar sectors (e.g. steel, iron or copper industries) while an economist may be interested in finding correlations between national political turmoil (e.g., in the Arcelor-Mittal case, the Indian and French governments) and corresponding market volatility. Objects of interest are companies in the former example and countries in the latter. Thus, it is necessary to allow experts to specify their objects of interest and construct, from the global streams of news issued by media companies, a collection of time sequences, each relating to a particular object of interest.

News' textual content can not be directly used for frequent pattern mining. News are all unique when considering that their exact textual content is always different. In that sense, no frequent pattern can be found since there is no such thing as a "frequent news". However, while two news might be different, the topics they present may be the same from an expert's point of view, e.g., several news may deal with "interest rate increase" or "a company takeover". For the purpose of frequent pattern mining, a generic description of the news is more relevant and useful for identifying frequent patterns. Before mining can be carried out, it is therefore necessary to compute a categorical description of the news that reflects

the news' semantics from the expert's point of view.

The issues outlined above could be handled through a preprocessing step carried out using existing algorithms for text classification (e.g. index term selection, naive Bayes classifier or nearest neighbor classifier) or text categorization (see [7] for an overview of these approaches). These algorithms can indeed be used to extract categorical descriptions of news and build individual time sequences by classifying news that are relevant to a given object of interest.

However, this approach is difficult in practice because frequent pattern mining is very sensitive to the precision of news description. Intuitively, when news descriptions become increasingly precise, very few news will have identical descriptions. As a result, they can not be identified as frequent. On the other hand, when news descriptions are not precise enough, they can be observed in many objects of interest and thereby resulting in a large number of frequent patterns. Moreover, the patterns discovered in this case are of little interest since they are formed from news descriptions which are not precise. Unfortunately, the exact level of precision in the news description that will fit a particular user's need can neither be known before hand nor computed. Finding it is an iterative refinement process that has to be performed in conjunction with the mining itself. Ideally, one would need to freely adjust the precision of the categorical descriptions in response to the mining results.

The large volume of data considered. Each month, Reuters publishes about 40000 individual news stories world wide. Depending on the expert's interests, only a fraction of these news may be relevant, but the patterns she might be looking for could span several months or years. For instance, Mittal Steel's bid for Arcelor started in January 2006 and the merger was actually announced in June 2006 after much political turmoil in French politics. However, the iterative approach typically adopted by FPM algorithms limits the size of datasets that can be mined within a reasonable amount of time as we will show in Section 5.

Also, when mining patterns over large periods of time, the minute details of news are not relevant. Experts are often in search for more general "long term trends" that are difficult to infer from the individual news. Therefore, mining over large periods directly on individual news not only affects the performance of the mining algorithm due to the large data size, but also may not produce the desired results since individual news are too precise for such a task. With numerical time series, such as a stock prices, this problem can be solved by computing a moving average [12] of the price over some time window, thereby reducing the number of data points. Ideally, we would like to perform the same abstraction operations on news data. The challenge here is to define how categorical descriptions of news can be "averaged" over a period of time.

We propose to support the difficult task of mining data with the charac-

teristics of financial news by providing a data personalization phase. Central to this phase are the "Preprocessing" and "Summarization" services, as demonstrated in the **S**treaming **TE**mpor**A**l **D**ata (STEAD) analysis framework presented in [14]. These services produce a summary structure for which experts can adjust both the content precision and the temporal precision. Semantic summarization is a method that allows building more concise representation of datasets by trading off the semantic precision of data, e.g. salary increases of 15% to 20% can be represented by the "strong increase" label. Several existing algorithms focused on semantic summariztion including Attribute Oriented Induction (AOI) [6], Fascicles [9], or SaintETiQ [17]. However, the proposed techniques were designed for summarizing generic database tables and to not allow to control the temporal precision of the produced summary.

We make the following contributions:

1. We address mining tasks by proposing a framework that allows preprocessing and mining of frequent patterns in financial news in a comprehensive manner. We design a *S*treaming *Te*mpor*a*l *D*ata (*STEAD*) analysis framework that allows a centralized supervision by the expert and allows her to express her domain knowledge which is then used throughout the various processing steps as well as for the refinement of the results.

2. We focus on the "Summarization Service" of the STEAD analysis framework [14]. We assume news are preprocessed and categorical descriptions of news are produced. We propose a Time-Aware Content Summarization (TACS) method that builds on top of *Attribute Oriented Induction* (AOI [6]). TACS extends the *Generalize and Merge* idea in AOI by computing in an tuple oriented way allowing the process to be incremental. Furthermore, the technique caters for controlling both the semantic and temporal precision of the summary. Its computational complexity is linear in the dataset size, allowing the handling of large datasets such as found in news environments. Most importantly, we can guarantee that patterns present in the original dataset will be present in the summarized one, up to the precision loss specified by the expert.

3. We validate our approach through experiments performed on a real world data set.

The rest of the paper is organized as follows. In Section 2, we present the general framework for analyzing streaming data. In Section 3, we provide background information about frequent pattern mining. We then detail our summarization approach in Section 4 and evaluate experimentally the approach in Section 5. We discuss other work orientations for the summary in Section 6 and present related work in Section 7. We conclude in Section 8.

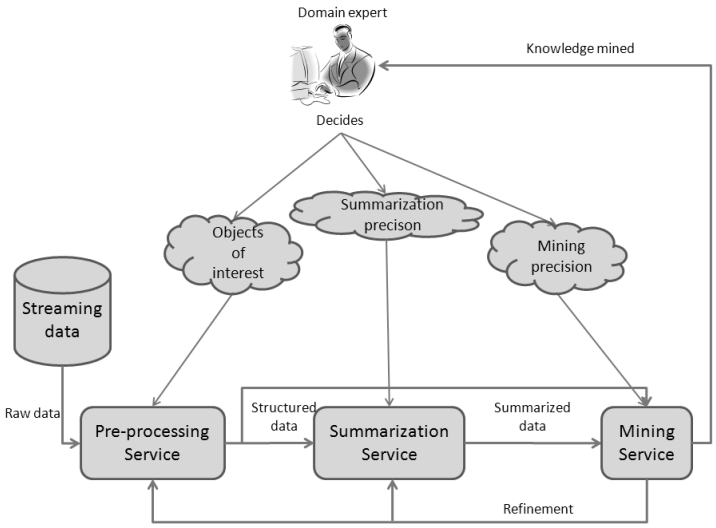


Figure 1: *STEAD* analysis framework

2 Framework for mining financial news

Mining frequent patterns in financial news is a challenging task that involves the choices and preferences of domain experts. This is why we designed *S*treaming *Te*mpor*a*l *D*ata (*STEAD*) analysis framework that lets domain experts express which aspects of the data they are interested in. This expert-centric *STEAD* analysis framework is illustrated in Figure 1. It is organized around three services and each service accepts as input, in addition to a dataset, a number of domain specific parameters set by the domain expert:

1. Preprocessing service: This service is responsible for transforming a raw and poorly structured dataset into a *structured* dataset made of categorical descriptions. These categorical descriptions can be in the form of tuples of attribute-value pairs.

2. Summarization service: This service considers as input the categorical descriptions of data and produces a new dataset that is less precise but more concise than the original one. In order to support mining algorithms that rely on the ordering of the data, the summarized dataset preserves the structure and the sequentiality of tuples of the original dataset.

3. Mining service: This service provides analysis tools, in particular Frequent Pattern Mining. In this case, the service identifies frequent patterns in either the structured data or the summarized data.

As we have mentioned in the introduction, mining frequent patterns in financial news is a very challenging task. The framework we propose

here is designed so that each challenge can be addressed independently. Each service can be extended to account for works in domains such as text processing and information retrieval. In this perspective, we will mainly focus our work on the summarization service to provide a support structure for FPM. Therefore, we provide in the following section all the necessary background on FPM for adequately designing the summary.

3 Preliminaries

3.1 Frequent pattern mining

Frequent pattern mining was first proposed by Agrawal and Srikant in 1995[2]. In this section, we briefly overview the main concepts and notations as used in this article.

Definition 1 (Time Sequence) *A time sequence S is a sequence $S = \langle e_1, \ldots, e_k \rangle$ of events where e_i are events and k is the length of the sequence. The order is defined by a timestamp attached to events and denoted by e.time. We use the notation $e_i \prec e_j$ when $e_i.time < e_j.time$ (we suppose that the timestamp is precise enough for all events to be strictly ordered).*

In our approach, events represent descriptions of news. They can be represented using one or several categorical attributes. For example, events could be defined on attribute *Interest rate* with values in $\{low, high\}$ and attribute *Inflation* with values in $\{low, moderate, high\}$. An example of event is then $e = (low, moderate)$. However, for the purpose of mining, the individual attribute values taken separately are not important. It is its *modality*, i.e., the event taken as a as a whole with all its attribute values, that is important. In the rest of the paper, we will denote modalities using capital letters and sequences using strings formed with these capital letters. For instance, if A $= (low, moderate)$ and B $= (high, low)$ then the sequence $S = \langle (low, moderate), (high, low) \rangle$ will be denoted by $\langle A, B \rangle$.

Definition 2 (Subsequence) *A sequence S_{sub} is a subsequence of S and we note $S_{sub} \sqsubseteq S$ iff all the tuples of S_{sub} also appear in the same order in S, consecutively or not. For example, if $S_{sub} = \langle A, B, C \rangle$ and if we have $S_{sub} \sqsubseteq S$, then S has the form $S = \langle *A * B * C* \rangle$, where $*$ represents any sequence of events, possibly empty. We also say that S* contains *S_{sub}.*

When mining frequent patterns, we consider as input a collection $\mathcal{C} = \{S_1, \ldots, S_n\}$ of time sequences. Given this input, we are interested in finding sequences of events, called in this case *patterns*, that are subsequences of sequences in $\mathcal{C}$.

Definition 3 (Support of a pattern) *The support of a pattern p in a collection of time sequences $\mathcal{C}$ is the number, denoted $supp_{\mathcal{C}}(p)$, of time sequences of $\mathcal{C}$ which contain p. When there is no ambiguity as to which collection $\mathcal{C}$ we refer to, we will denote the support simply by $supp(p)$.*

Using these notations, the problem of frequent pattern mining can be defined as follows:

Definition 4 (The frequent pattern mining problem) *Considering a collection $\mathcal{C}$ of time sequences and an expert-defined support $supp_{min}$, identify all the patterns p that satisfy the condition $supp_{\mathcal{C}}(p) \geq supp_{min}$, i.e. all the patterns which are found in more than $supp_{min}$ sequences of $\mathcal{C}$.*

3.2 Evaluating the interestingness of frequent patterns

As mentioned earlier, we supposed that the global stream of news is preprocessed by the "Preprocessing Service" in the *STEAD* analysis framework and categorical descriptions are output by the service. The consequences of this preprocessing step are as follows. When breaking down the global stream of news to create news time sequences, each news item n_i that concerns several objects of interest (e.g. Microsoft and Google) needs to be duplicated into their respective news time sequences. By doing so, we artificially augment n_i's individual support. n_i can potentially become frequent (i.e. $supp(n_i) \geq supp_{min}$) by duplication and bias the mining results.

When generalizing this phenomenon, we can expect (i) a combinatory explosion at candidate generation time for algorithms based on a *generate and prune* strategy (e.g., Apriori-based algorithms such as in [2, 19]) and (ii) the length of frequent patterns can possibly be increased by duplicates. An important work needs to be done to evaluate the *interestingness* of patterns and to distinguish those from surrounding *noise*.

3.3 Noisy patterns

Interesting patterns are sequences of events such that the order between events has some significance, e.g. "salary increase" followed by a "home loan subscription". Suppose that two news A and B both occur frequently. We will observe that most of the patterns formed by A and B ($\langle A, A\rangle$, $\langle A, B\rangle$, $\langle B, B\rangle$, $\langle B, A\rangle$, $\langle A, A, A\rangle$, $\langle A, A, B\rangle$, etc...) may be frequent. Such patterns however may have very limited significance since the fact that a pattern $\langle A, B\rangle$ is frequent is a mere coincidence and may not reveal any correlation between A and B: the order between these two events has no significance. For example, consider events A and B described by attributes *Interest rate* and *Inflation* and A = $(low, moderate)$ and B = $(high, low)$. Successive and repetitive patterns with A and B do not bring any additional information. Such patterns are called *noisy patterns* because there existence

complicate the mining task by creating numerous candidate patterns that need to be individually explored, thereby impacting very significantly the overall performance of mining algorithms. We will discuss in Section 4.2 how our approach can leverage the issue of noisy patterns.

Interestingness measure.

The evaluation of the *interestingness* of patterns mined over news can be used as an indicator for pruning noisy patterns. This issue of noisy patterns can be leveraged by introducing an *objective*[18] measure at mining time. During the preprocessing step, we propose to *tag* all n_is with a *duplication* score, e.g. number of time sequences n_i is duplicated into. This score can be used at mining time to describe a candidate pattern p_k by an *interestingness* metric $I(p_k)$, where $0 \leq I(p_k) \leq 1$. This metric indicates to what extent p_k is made of news n_i with low duplication scores. We introduce the metric as in Equation 1 where $|X|$ denotes the cardinality of the set X. The lower the duplication scores of each n_i composing p_k, the more p_k is interesting. The most informative case is when p_k is only made of non-split items, i.e. $\forall n_i \in p_k, score(n_i) = 1$ in which case $I(p_k) \to 1$ with $|Objects\ of\ interest| \gg |p_k|$, and the worst case is when all n_i a are split into all sequences, i.e. $\forall n_i \in p_k, score(n_i) = number\ of\ objects\ of\ interest$ in which case $I(p_k) = 0$.

$$I(p_k) = 1 - \frac{1}{|Objects\ of\ interest|} . \frac{\Sigma_{n_i \in p_k} score(n_i)}{|p_k|} \tag{1}$$

Another solution for evaluating the interestingness of patterns and discard noisy patterns is to use *subjective*[18] measures *a posteriori*. We can refer the reader to some recent work such as Xin et al.'s [22].

4 The Time-Aware Content Summary (TACS)

We present in this section our Time-Aware Content Summarization (TACS) approach for supporting FPM algorithms. The algorithm considers as input a set of sequences of news categorical descriptions. These descriptions are represented using tuples of attribute-value pairs. The output of the algorithm is a set sequences of news categorical descriptions represented on the same attributes (or a subset of those), expressed in a less precise but more concise form. The summarization algorithm operates in two steps: Generalization and Merging.

- Generalization: Incoming tuples t_i are generalized into t'_i using concept hierarchies (e.g. Figure 2). The generalization considers each attribute of the tuple by rewriting its associated value to a more general concept as specified by the expert. These concept hierarchies can be either manually

defined or automatically generated using general purpose (e.g. WordNet[1]) or domain specific ontologies. All t'_i are then called *generalized* tuples.

- Merging: Identical generalized tuples are grouped together. Even if two successive tuples t_i and t_{i+1} are different on certain attributes, their generalized representation, respectively t'_i and t'_{i+1}, can be identical. In such case, t_i and t_{i+1} are said to be *similar* from expert point of view. Thus, t'_{i+1} is *merged* with t'_i and a $COUNT$ attribute—indicating the number of original tuples represented by t'_i— and a reference to t_{i+1} are maintained. Generalized tuples that have been merged are then called *summarized* tuples. We give the general algorithm in Algorithm 1.

We mentioned earlier that FPM algorithms rely on the sequentiality of the data to perform. To preserve this ordering property, we need to consider if an incoming generalized tuple t'_{i+1} is *identical* to the previous one—in which case these tuples are merged—. Otherwise, t'_{i+1} will be added to the summary by itself and incoming generalized tuples will be compared to it. Incrementally, the summarized tuples keep the ordering of the original data w.r.t. Definition 5 and Corollary 1 as defined in section 4.1.

4.1 Trading off content precision for attribute domain reduction

During the generalization step, each attribute value of an incoming tuple is generalized to a expert-desired level of abstraction using concept hierarchies. The idea is to allow the expert to express in the form of concept hierarchies and in her own vocabulary how the attribute values are generalized. From these concept hierarchies, she can decide to reduce attribute domains by trading off content precision. No generalization means keeping the precise attribute value as precise as possible and thereby the lower the domain reduction.

On the other hand, if the precision of attribute value is not significant, she may decide to generalize it to more generic forms thereby augmenting the domain reduction, e.g., the root *Any_Location* is the most generic label in Figure 2.

This generalization approach is a simple yet powerful tool for allowing experts to express their needs, trade attribute domain reduction and mining capabilities with content precision. Concretely, suppose each attribute A_i in $A = \{A_1, ..., A_n\}$ is associated with a concept hierarchy H_i in $H = \{H_1, ..., H_n\}$ where each concept hierarchy H_i is a tree on the domain of A_i. Each edge in H_i is an *is-a* relationship over the literals in A_i. If there is an edge in H_i from the literal p to the literal c, we call p a parent of c and c a child of p: p is a generalization or most specific common subsumer (MSCS) of c and denote $MSCS(c) = p$. The notion of MSCS can be extended to the tuple level. The MSCS of two tuples t_1 and t_2 defined on

Algorithm 1 General ST summarization algorithm

Input:

- DB Database to summarize
- $List_\omega$ List of ω previously summarized tuples (in the base algorithm, $\omega = 1$)
- $A = \{A_1, A_2, ..., A_n\}$ Attributes to summarize
- $H = \{H_1, H_2, ..., H_n\}$ Concept hierarchies over A

Output:

- Z Time-Aware Content Summary

Initialize Z
for all tuple t_i in DB **do**
  Generalize t_i into t'_i using H
  if $list_\omega$ is void **then**
    Add t'_i into $list_\omega$
    Initialize t'_i's $COUNT$ to 1;
    Initialize t'_i's list of tuple IDs ($TIDs$) with t_i;
  else
    if $t'_i \in list_\omega$ (Suppose $t'_j \in list_\omega$ and $t'_i = t'_j$) **then**
      Increment t'_j's $COUNT$;
      Add t_i's tuple ID into t'_j's $TIDs$;
    else
      Pop out oldest tuple t'_{last} in $list_\omega$
      Add t'_{last} to Z with its $COUNT$ and $TIDs$
      Add t'_i to $list_\omega$
      Initialize t'_i $COUNT$ and $TIDs$
    end if
  end if
end for
Add all remaining tuples in $list_\omega$ into Z
return Z

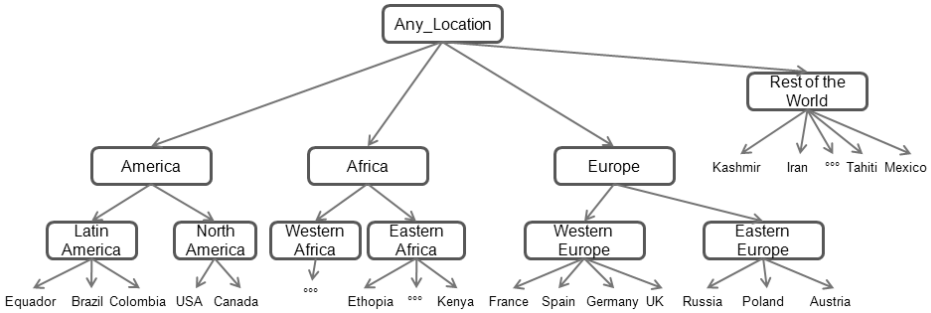


Figure 2: Example of concept hierarchy for the *Location* attribute

attributes $A_1, ..., A_n$ is the tuple t' that subsumes t_1 and t_2 on all attributes $A_1, ..., A_n$ and is denoted $t' = MSCS(t_1, t_2)$.

When the expert decides to trade away the content precision regarding the value a_{ij} of an attribute A_i for increased domain reduction by generalizing a_{ij} k times, the generalization algorithm seeks a_{ij}'s k^{th} subsumer in H_i. For example, "Europe" is the second generalization of "France". This is an operation with a constant and low cost as concept hierarchies often have few number of intermediary levels and nodes, and many leaves as illustrated in Figure 2. This cost can be further optimized by incrementally maintaining indexes as generalizations are found.

We mentioned earlier that once incoming tuples are generalized, it is possible that two successive generalized tuples t'_i and t'_{i+1} have *similar* descriptions from expert's point of view, in which case they are merged together. Merging together contiguous and similar generalized tuples allows to keep the original ordering of tuples. We formally define an Order-Preserving (OP) summary in Definition 5 with the total order relation $\prec$ as defined in Definition 1.

Definition 5 (Order-Preserving (OP) summary) *Let f be a summarization function and $f(t_i) = z_i$ the summarized value of tuple t_i; by extension $f(R) = Z$ is the summary of table R. Z is an Order-Preserving (OP) summary of R when there is a total order relation $\preceq_Z$ on Z defined as: $\forall (t, t') \in R^2, t \prec t' \Leftrightarrow f(t) \preceq_Z f(t')$.*

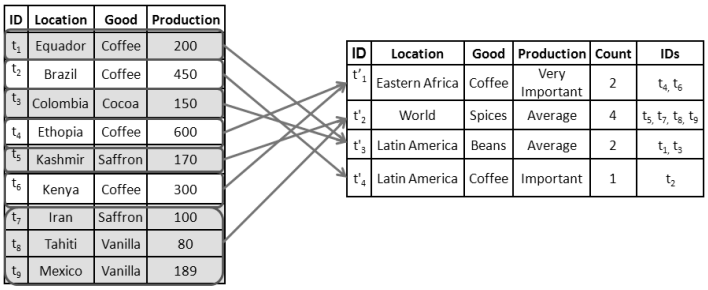
Corollary 1 (Monotony) *Let f be a summarization function where $f(R) = Z$ is an OP summary as defined in Definition 5. f is monotone and increasing.*

Representing attribute values in a less precise form has an impact on the types of the discovered frequent patterns. Indeed, mining a summary will provide the expert with frequent patterns of summarized tuples that we call *trends*. In some cases, knowing the trends in the news is enough for an expert

to make a decision (e.g. buy or sell stocks). In other cases, this information is interesting but not precise enough to make an informed decision. Because summarized tuples represent subsets of news and store the IDs of the actual data, it is possible for the expert to use the frequent patterns of interest to filter the original data. Doing so, she can select smaller subsets of the actual data and reiterate the (summarization and) mining process.

4.2 Trading off temporal precision for tuple numerosity reduction

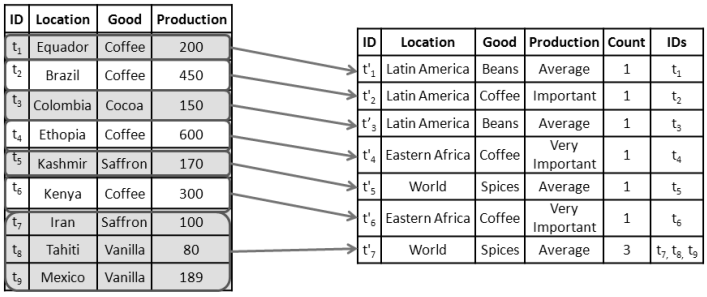
Combining attribute domain reduction and generalized tuple merging allows to produce summarized news time sequences containing less tuples than the original input. The basic TACS algorithm outputs a summary that strictly complies to Definition 5. Unfortunately, the total order relation is a constraint that makes the summary inefficient regarding two different aspects: (i) numerosity reduction and (ii) noisy patterns.



ID	Location	Good	Production
t_1	Equador	Coffee	200
t_2	Brazil	Coffee	450
t_3	Colombia	Cocoa	150
t_4	Ethopia	Coffee	600
t_5	Kashmir	Saffron	170
t_6	Kenya	Coffee	300
t_7	Iran	Saffron	100
t_8	Tahiti	Vanilla	80
t_9	Mexico	Vanilla	189

ID	Location	Good	Production	Count	IDs
t'_1	Eastern Africa	Coffee	Very Important	2	t_4, t_6
t'_2	World	Spices	Average	4	t_5, t_7, t_8, t_9
t'_3	Latin America	Beans	Average	2	t_1, t_3
t'_4	Latin America	Coffee	Important	1	t_2

Figure 3: Non Order-Preserving summary



ID	Location	Good	Production
t_1	Equador	Coffee	200
t_2	Brazil	Coffee	450
t_3	Colombia	Cocoa	150
t_4	Ethopia	Coffee	600
t_5	Kashmir	Saffron	170
t_6	Kenya	Coffee	300
t_7	Iran	Saffron	100
t_8	Tahiti	Vanilla	80
t_9	Mexico	Vanilla	189

ID	Location	Good	Production	Count	IDs
t'_1	Latin America	Beans	Average	1	t_1
t'_2	Latin America	Coffee	Important	1	t_2
t'_3	Latin America	Beans	Average	1	t_3
t'_4	Eastern Africa	Coffee	Very Important	1	t_4
t'_5	World	Spices	Average	1	t_5
t'_6	Eastern Africa	Coffee	Very Important	1	t_6
t'_7	World	Spices	Average	3	t_7, t_8, t_9

Figure 4: Order-Preserving summary

We can illustrate the intuition behind our approach with the examples in Figure 3 and 4. In these examples, the non order-preserving summary generates 4 summarized tuples versus 7 for the order-preserving summary. This simple observation shows that strictly respecting the total order relation can require a trade off on the size of the summary.

The concept of *time* differs from one expert to another: some experts

might be more interested in information on a daily basis whereas other might be interested in monthly events for instance. In the former case, the expert is interested in a *snapshot* of each day and requires that ordering considers daily events. In the latter case, the order in which news have arrived during the month is not important provided a general *snapshot* of the month is given. This observation allows us to provide the expert with an additional mechanism for making the summary more concise by reducing the numerosity of tuples in the summary. When a coarser time-grain is acceptable for the expert, she can reduce the temporal precision of the summary by locally violating the total order relation, e.g., for all elements in a window spanning over a month.

We introduce temporal precision in TACS as follows: Let ω be a time span defined by an expert where ω can be expressed (i) as a duration or (ii) as a number of tuples ($\omega \in \mathbb{N}$). We call W of temporal precision ω a *window of reduced temporal precision* (or *precision window* for short) in which the relation $\preceq_Z$ is locally violated and *disorder* withing tuples is allowed. W can be either (i) a fixed window or (ii) a sliding window over the summarized tuples. The choices of ω and the precision window W define the way temporal precision is handled in TACS.

Temporal precision ω.

Expressing ω as a duration (e.g., $\omega = 1$ *hour*) allows experts to setup the temporal precision. She exactly knows what events happened within the time span ω but has no guarantee in their exact order. As a consequence, all elements in a burst of events that fits into W will be merged together, e.g., if events $\langle A, B, C, D, C, B, A\rangle$ arrive within 1 hour, they will be merged into $\langle A, B, C, D\rangle$.

Defining ω as a number of tuples allows the summarization to be done with variable temporal precision. Provided ω is defined small enough (e.g., $\omega = 3$ tuples), this approach can limit the merging of bursting events as all tuples will not be merged together and information on their sequentiality will be preserved. The main risk with this approach is to merge together events very distant in time when ω is set too high. In financial applications, it is not rare to witness rapid chain reactions when important events happen, e.g., company takeovers. Therefore, we choose in our work to express ω as a number of tuples to be able to address these bursting events.

Precision window W.

The second parameter for reducing temporal precision in TACS is the way the precision window is moved during the summarization, i.e., in a fixed or sliding way. Figures 5 and 6, where "[" and "]" delimit the position of W ($\omega = 3$ tuples), illustrate the summarization of a sequence S into a sum-

mary Z with these two different methods. Given a precision ω, these figures show that it is possible to merge repetitive subsequences (e.g., $\langle A, B, C\rangle$ in our example) when using a sliding precision window (e.g., subsequence $\langle A, B, C, A, B, C\rangle$ is merged into $\langle A, B, C\rangle$). This is a desirable feature as repetitive patterns in a sequence are not always of interest and can be considered as noisy patterns. Therefore, in the rest of our work, we chose to design W as a sliding window.

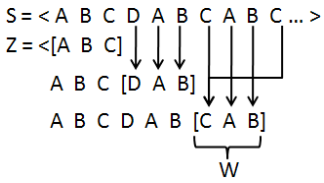


Figure 5: TACS with a fixed window

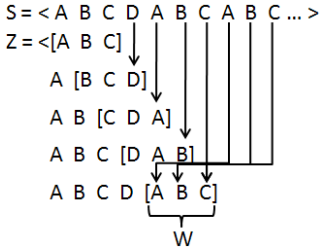


Figure 6: TACS with a sliding window

In a nutshell, defining the temporal precision of TACS with ω expressed as a number of tuples and W as a sliding window allows to reduce the temporal precision w.r.t. the expert's own perception of time. The extreme cases when defining ω are:

- $\omega = 1$: The summarization algorithm has to strictly respect the total order relation and no temporal concession is done;
- $\omega = \infty$: The summarization algorithm does not respect the total order relation and converges toward a semantic summarization algorithm as presented in Section 7.

Suppose we take Figure 4 and define a temporal precision of $\omega = 2$ tuples using a sliding precision window. As tuples arrive into the system and are generalized, t'_3 will be merged into t'_1, t'_6 into t'_4 and t'_7 into t'_5 as shown in Figure 7. The output is a summary with 4 summarized tuples, exactly those obtained in Figure 3. The difference with the output in Figure 3 is the ordering of summarized tuples which in our case follows a partial order given by the sequentiality of incoming tuples and ω.

Reducing the temporal precision has two benefits for frequent pattern mining approaches: (i) the numerosity of tuples is further reduced and (ii) all combinations of frequent tuples within a window ω are merged together. For example, if $\omega = 2$, combinations of two frequent tuples A and B such as $\langle A, B, A, A, B\rangle$, $\langle A, B, A, B, B\rangle$, $\langle B, A, B, B, A\rangle$, $\langle B, A, B, A, A\rangle$, ... are merged into $\langle A, B\rangle$ or $\langle B, A\rangle$. Thereby, reducing temporal precision has the nice property of locally merging noisy patterns which would have burdened the FPM algorithm.

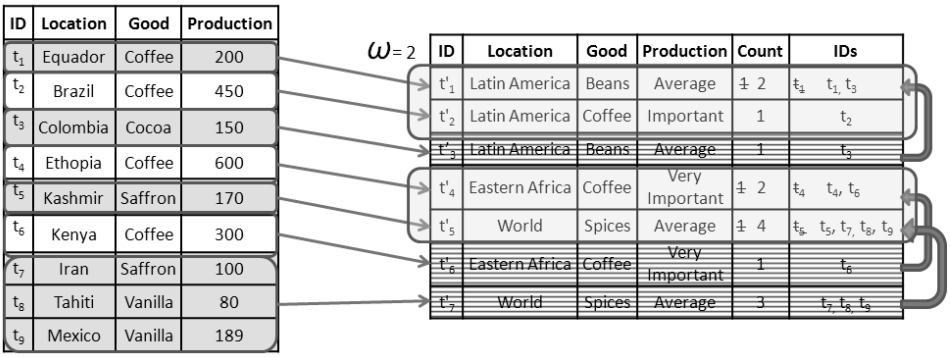


Figure 7: Order-preserving summary with temporal precision of $\omega = 2$ tuples

However, merging noisy patterns also comes with a trade off in the number and diversity of sequences in the summary. Indeed, some sequences can be potentially lost while merging generalized tuples. The order in which summarized tuples appear in a summary Z completely depends on the order in which the generalized tuples appeared in the sequence, e.g., $Z = \langle A, B \rangle$ means A have arrived first, followed by B. Therefore, different sequences having the same generalized tuples can be summarized into a same summary, depending on which items have arrived first. For example, $\langle \mathbf{A,B}, A, A, B \rangle$ and $\langle \mathbf{A,B}, A, B, B \rangle$ are merged into $\langle \mathbf{A,B} \rangle$, and $\langle \mathbf{B,A}, B, B, A \rangle$, $\langle \mathbf{B,A}, B, A, A \rangle$ are merged into $\langle \mathbf{B,A} \rangle$, but *both* $\langle \mathbf{A,B} \rangle$ and $\langle \mathbf{B,A} \rangle$ can not appear simultaneously within the same precision window W.

In some cases, the sequence $\langle \mathbf{B,A} \rangle$ might be more informative but could have been lost. Hence, this merging capability leads to a trade off on the recall of sequences. The conditions in which a summary with reduced temporal precision can lose such sequences need to be determined. In the following section, we identify, define and prove these minimal requirements for containing the loss of sequences during the summarization.

4.3 Minimal requisites for trading off temporal precision

When reducing the temporal precision of TACS, we mentioned that there is a risk of loosing sequences during the merging phase of the process. It is important to be able to determine the conditions in which such loss can occur. In the following paragraphs, given a subsequence S_1 of S, we define the conditions for S_2 (obtained by permutations of tuples in S_1) to be found as a subsequence of S.

Let $S_1 = \langle t'_1, t'_2, ..., t'_m \rangle$, $|S_1| = m > 2$, be a sequence of summarized tuples. Let $S_2 = \langle u'_1, u'_2, ..., u'_m \rangle = perm(S_1) = \langle t'_{perm(1)}, t'_{perm(2)}, ..., t'_{perm(m)} \rangle$ also be a sequence of m summarized tuples and S_2 is the result of any $k > 0$ permutations of summarized tuples in S_1.

Let S be a summarized news time sequence with a temporal precision of $\omega \geq 2$ using a sliding precision window W, $|W| = \omega$ tuples. $T = \langle t'_{T_1}, t'_{T_2}, ..., t'_{T_n} \rangle$ denotes the sequence of generalized tuples to come and to be summarized. Then, we can express S as:

$$S = \langle \underbrace{t'_1, t'_2, ..., t'_m}_{S_1}, \underbrace{t'_{T_1}, t'_{T_2}, ..., t'_{T_n}}_{T} \rangle, n \to \infty$$

Property.

We use α to denote the number of contiguous tuples in S following S_1 and which do not appear in S_1, e.g, if $S = \langle \underbrace{t'_1, t'_2, ..., t'_m}_{S_1}, x, y \rangle$, $\alpha = |\{x, y\}| = 2$.

For S_1 and S_2 to be consecutive subsequences in a same sequence S, the number α of tuples separating S_1 and S_2 must meet one of the following conditions:

1. Case $(|S_1| - \omega) \geq 2$, i.e. the window of temporal precision is smaller than the sequence S_1 by at least 2 tuples, then there is no requirement on α.

2. Case $(|S_1| - \omega) = 1$, i.e. the window of temporal precision is smaller than the sequence S_1 by 1 tuple, then $\alpha \geq 1$.

3. Case $(|S_1| - \omega) < 1$, i.e. the window of temporal precision is bigger than the sequence S_1, then $\alpha \geq 2 + \omega - |S_1|$.

If these conditions are not true, it is not possible for S_2 to be a subsequence of S and appearing in S after S_1. When the condition on α is not true and tuples in S_2 arrive after S_1, they will be merged into S_1 during the summarization process. The full proof of this property can be found in Appendix A. It is not possible to theoretically quantify this information loss as it completely depends on the distribution and ordering of incoming tuples. However, in practice, trading off temporal precision for more tuple reduction does not have a significant impact on the patterns that can be mined for two reasons.

First, our preliminary experimental evaluation of TACS in Section 5 on a month worth of Reuters' news shows that a summarization and mining cycle can be done in a very limited time, i.e. in the order of minutes. Therefore, the summary makes it possible for experts to mine news in an interactive and iterative way. Suppose a sequence $\langle A, B, C \rangle$ is present in TACS and sequence $\langle C, B, A \rangle$ was lost during the summarization process. Assuming $\langle A, B, C \rangle$ is a frequent pattern, the set of news $N = \{n_1, ..., n_m\}$ represented by summarized tuples A, B and C in pattern $\langle A, B, C \rangle$ will be exactly the same as those represented by pattern $\langle C, B, A \rangle$. Thus, if the expert decides to reiterate the summarization and mining cycle with higher

temporal precision (and eventually higher semantic precision), all news in N will still be selected for this new task without any loss.

Second, we suppose that patterns of interests in financial news are relatively short (with more or less have a length of 6-10 tuples). In general, a temporal precision of $\omega = 5$ tuples for most companies represents a couple of days to a couple of weeks worth of news: this setup fits into the conditions of case $(|S_1| - \omega) = 1$ ($\alpha \geq 1$). Because of the very large number of modalities in financial news, the requirement of $\alpha \geq 1$ is very easily fulfilled.

5 Experimental evaluation

In this section, we report our experimental results on the performance of TACS while summarizing and while performing FPM tasks. Our interest is to determine the impact of different temporal precision parameters on the summary itself and on the frequent patterns mined.

All the experiments were performed on a 2GHz Core2Duo laptop with 2GB of main memory, running Microsoft Windows XP SP2. The DBMS installed is PostgreSQL version 8.0.7 running on a 5400rpm hard drive. The summarization algorithm and PrefixSpan[13] are written in C# and are using the Microsoft .NET framework 2.0. During all the tests, the GUI was minimized and hidden so that the true running times of the algorithms were recorded.

The dataset used is one month worth of financial news (January 2004) obtained from Reuters. The original sources have been preprocessed and filtered so that news can be categorized on a set of 12 attributes (e.g. commodities, location, etc...) and objects of interest are the company names the news are related to. Throughout the rest of the paper, we will refer to this dataset as the *raw* news. Concept hierarchies on these attributes were designed manually using Reuters codification of their news and WordNet[1] ontologies. The description of news items can have undefined values, in which case "none!!" is the default value. This default value is considered to be different from any other value, e.g. different from any label in concept hierarchies.

We have focused our efforts on the core of our contribution, i.e. handling the time dimension in TACS through different settings of the temporal precision ω. Therefore, we fixed the semantic precision by allowing only one generalization of attribute values during the summarization process. Our experiments show that the use of TACS allows to mine trends in financial news at higher support levels and in acceptable processing times (i.e. in the order of hours) whereas mining raw news only starts giving results(patterns of length $> 5 - 6$) at lower support levels and with processing times that can reach the order of tens of hours.

5.1 Impact of temporal precision on the construction of TACS

The objective of the first set of experiments on Reuters' raw news is to determine the processing time and the tuple reduction capability when building and storing TACS. Figure 8 and 9 respectively give the time necessary to build TACS and its tuple reduction ratio with a set of different temporal precisions. ω ranges from 1, i.e. strict compliance to the OP constraint, to $\omega \to \infty$ (in practice, we fixed $\omega = 5000$). These figures show that reducing the temporal precision does not have any penalty on processing time and can sensibly increase tuple reduction. The slight decrease in processing time is only due to the increased tuple reduction: higher tuple reduction means less tuples to write into the output database for storage.

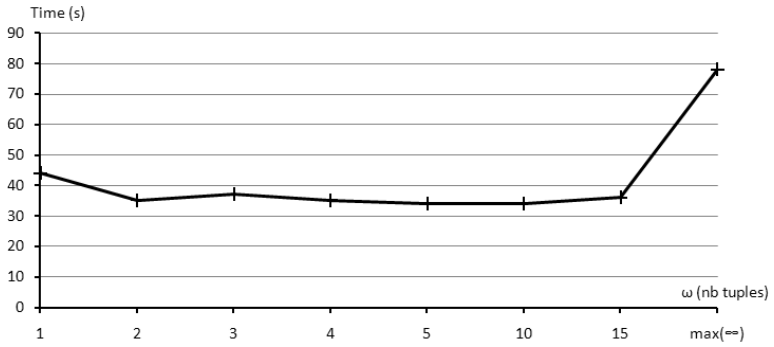


Figure 8: Summarizing time

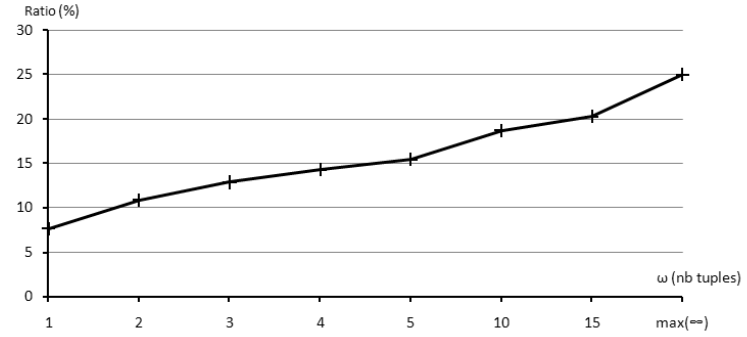


Figure 9: Tuple reduction ratio

5.2 Mining a TACS at different levels of temporal precision

Once the summaries built, we carried the FPM task with our implementation of PrefixSpan on the raw news and on summaries with $w \in \{1, 5, 10, 15\}$. The results are given in Figure 10 and 11. Figure 10 gives the time necessary for our PrefixSpan implementation to completely compute (on raw news and

on summaries) at different levels of support. Concurrently, Figure 11 gives the maximum length attained by frequent patterns mined.

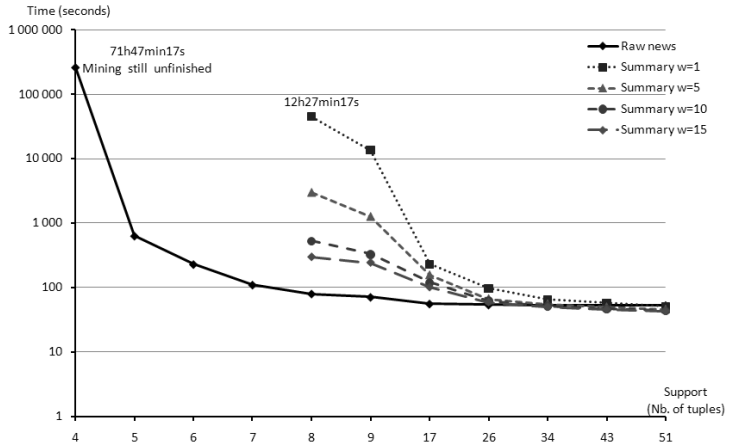


Figure 10: FPM processing time

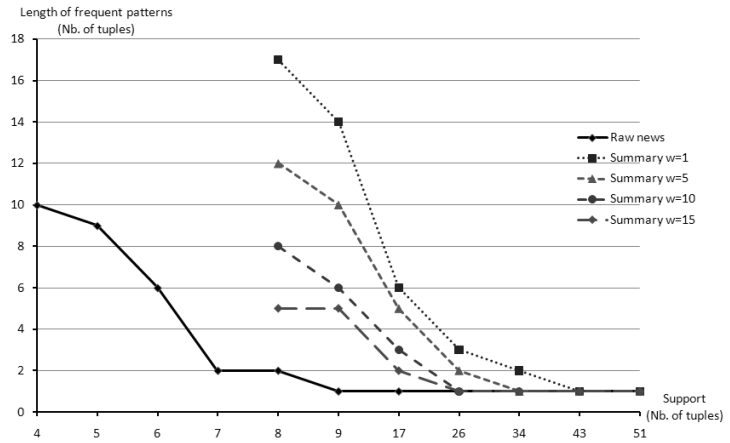


Figure 11: Frequent pattern maximum length

This latter figure shows that mining frequent patterns on the raw news only starts yielding results with very low support levels, e.g. starting with $supp_{min} = 7$ the maximum length of patterns is only 2. This observation is very coherent with our earlier intuition that when news are described with high precision, the chances of finding identical news in several sequences is very low. In this matter, the raw news dataset is the most precise possible description of the news. From this point on, lowering further $supp_{min}$ gives as result longer maximum patterns but increases exponentially the processing time with hops from a couple of minutes ($supp_{min} = 6$) to around 10 minutes ($supp_{min} = 5$) and finally to tens of hours (at $supp_{min} = 4$ the

process did not complete after more than 71 hours).

On the other hand, mining frequent patterns on TACS also yields interesting results. Further analysis of Figure 11 reveals that mining TACS at higher levels of support (e.g. $supp_{min} = 17$) gives frequent patterns of length > 2 for all values of ω. It is therefore possible to start discovering *trends* when mining at higher levels of support. Lowering step by step the support also gives longer patterns but the process reaches a limit when $supp_{min} = 8$ and $\omega = 1$ where the processing time just explodes.

Indeed, when $supp_{min} = 8$ and ($\omega = 1$ or $\omega = 5$), the maximum length of the patterns mined are much more important than patterns mined over the raw news. This effect is a direct consequence of the generalization step in the summarization algorithm. Indeed, if a tuple t_i does not have minimum support, i.e. $supp(t_i) < supp_{min}$, by generalizing t_i into t'_i, t'_i can have minimum support, i.e. $supp(t'_i) \geq supp_{min}$. This phenomenon is due to the reduction of the overall number of modalities in the dataset by generalizing tuples. Noisy patterns are then potentially introduced and has the drawback of burdening the FPM algorithm as more paths need to be explored. This explains the increased maximum length of the frequent patterns mined as well as the high computational cost, e.g. completion of the mining with $supp_{min} = 8$ and $\omega = 1$ took more than 12 hours.

However, this phenomenon can be leveraged. When reducing further the temporal precision of the summaries, e.g. $\omega = 10$, the maximum length of frequent patterns is reduced as well as the processing time. This observation backs up our earlier intuition that reducing the temporal precision has the nice property of locally merging noisy patterns. The *sweet spot* ω_{opt} is then somewhere between $\omega = 5$ and $\omega = 10$ where both processing time and frequent patterns' length are short and acceptable.

6 Discussions

This work has mainly focused on the temporal dimension of TACS. We decided to fix the content precision of the summary in Section 5 by allowing one single level of generalization for each attribute. The general algorithms of TACS leaves room for more adjustment of the content precision. Reducing the content precision of news descriptions can be done in two steps:

- Choice of the generalization methodology, meaning how tuples will be generalized;
- Choice of the policy for merging tuples w.r.t. to similarity metrics.

Generalization methodology

The base algorithm we propose considers each tuple individually and sets a level of generalization for each attribute. There are two other possibilities

for generalizing the tuples. This choice depends on the view the expert wants from the data:

1. She can choose to define a *bucket* of size k where all incoming tuples will be represented by a single summarized tuple. Intuitively, this approach allows her to represent a set of tuples by a single "average" tuple. In this case, a bucket can be view as an array of fixed size. The k first incoming tuples will fill the bucket up and all tuples will be generalized and merged into one single summarized tuple that is added to the summary Z. For merging tuples in a bucket, the MSCS (as defined in Section 4.1) of all tuples in the bucket is generated. The process is then repeated with new buckets. This method is useful in situations where the expert only wants to know the profile of events that have happened during a short period, e.g. during one day. The risk is to over-generalize, e.g., when the size of the bucket is too large.

2. On the opposite side, the expert can choose not to define how incoming tuples should be generalized and use conceptual distances to allow for an automated approach. The idea is to let the generalization process decide, based on a conceptual similarity metric, if two tuples are conceptually similar. In this case, case they are to be generalized into their most specific common subsuming form. We call this approach an *adaptive generalization* approach as the decision for generalizing depends on the data itself. This approach only requires the expert to specify the semantic similarity metric to use.

Similarity metrics

As we mentioned above, when adopting an adaptive generalization approach, the expert needs to define which similarity metric to use for evaluating the distance between two generalized tuples t_1 and t_2. Much work in natural language processing has been done to evaluate lexical semantic relatedness (we refer the reader to [5] for a more complete review). From the metrics proposed in this rich literature, the following conceptual distances have seemed relevant to us for supporting our summarization approach:

– Rada et al.'s Simple Edge Counting distance presented in [15] seemed to be the most naive approach for computing the distance between two concepts c_1 and c_2 in a conceptual hierarchy. Indeed, their distance $dist_{Retal}$ is simply the minimal number of edges in a path from c_1 to c_j.

– Wu and Palmer introduced in [21] a specialized metric for measuring the conceptual similarity between two concepts c_1 and c_2 within a single domain organized in a strict hierarchical structure. The authors

define their similarity metric sim_{WP} and distance $dist_{WP}$ as:

$$sim_{WP}(c_1, c_2) = \frac{2 * N_3}{N_1 + N_2 + 2 * N_3} \quad (2)$$

$$dist_{WP}(c_1, c_2) = 1 - sim_{WP}(c_1, c_2) \quad (3)$$
$$= \frac{N_1 + N_2}{N_1 + N_2 + 2 * N_3}$$

where N_3 is the distance (in terms of number of edges) from the root of the hierarchy to the most specific common subsumer of c_1 and c_2, denoted $lso(c_1, c_2)$ (lowest super-ordinate), N_1 the distance between c_1 to $lso(c_1, c_2)$ and N_2 the distance between c_2 and $lso(c_1, c_2)$. This similarity measure seems a promising approach to adapt as it considers both the *commonality* and the differences between both concepts c_1 and c_2 and gives a more refined estimation compared to the Simple Edge Counting.

– In [16], Philip Resnik proposed an approach quite similar to Rada's edge-counting method. His intuition was that *"one criterion of similarity between two concepts is the extent to which they share information in common"*. Therefore, he propose a similarity measure based on the evaluation of the *information content* of concepts by using text corpus. This metric is very interesting to consider when in addition to the incoming tuples, there is information in the form of free text (e.g. financial news issued by Reuters have both structural information and unstructured free text). Therefore, given C the set of concepts of a taxonomy (in our case a hierarchy), he defines $p : C \rightarrow [0, 1]$ such that $\forall c \in C$, $p(c)$ is the probability of encountering *an instance* of concept c. The similarity measure between two concepts c_1 and c_2 is thus defined by:

$$sim_R(c_1, c_2) = max_{c \in S(c_1, c_2)}[-\log p(c)] \quad (4)$$
$$= -\log p(lso(c_1, c_2))$$

where $S(c_1, c_2)$ is the set of concepts that subsume both c_1 and c_2 and $lso(c_1, c_2)$ the most specific common subsumer of c_1 and c_2. In our case where concept hierarchies do not allow multiple inheritance, $p(c) = \frac{freq(c)}{N}$, where $freq(c)$ is the number of occurrences of c in the corpus and N the total number of words in the corpus.

– A last interesting corpus-based metric was proposed by Lin et al. in [10] where the author proved that *"the similarity between A and B is measured by the ratio between the amount of information needed to state their commonality and the information needed to describe what they are"*. From this theorem, Lin defined the following similarity

measure:

$$sim_L(c_1, c_2) = \frac{2 * \log p(lso(c_1, c_2))}{\log p(c_1) + \log p(c_2)} \quad (5)$$

where the notations $p(c)$ and $lso(c_1, c_2)$ are similar to Resnik's.

Which ever the similarity metric chosen, computing the distance between to tuples t_1 and t_2 for generalizing and merging them can be done in several ways. We choose the naive approach where we simply aggregate and sum up the similarities for each attribute of t_1 and t_2 as follows:

$$dist(t_1, t_2) = \Sigma_{A_i \in A} sim(t_1(A_i), t_2(A_i)) \quad (6)$$

where A follows the notation presented in 4.1 and where $t_k(A_i)$ denotes the attribute value of t_k on attribute A_i and $sim(x, y)$ the similarity metric chosen. This way of computing the global distance between t_1 and t_2 assumes that attributes in A are independent. Another possibility is to consider that attributes are not independent and ponderate each attribute value.

7 Related work

The Time-Aware Content Summarization approach is a work relating to several areas of research which are (i) time series representation, (ii) semantic compression and (iii) semantic summarization.

The term "time series", by contrast to "time sequences", refers to sequences of one or more numerical values. Various numerical methods, e.g. moving average [12], can be applied for reducing the number of data point in time series. They essentially consist of computing aggregates of several data points over a period of time and can not be applied to textual time sequences since such aggregates can not be computed for textual data.

SAX [11] is a technique capable of handling data reduction using a symbolic representation of numerical time series. Due to its symbolic nature, this method is more likely applicable to textual time sequence summarization. Indeed, first, the authors compute aggregates by Piecewise Aggregate Approximation (PAA). Then, these PAA are converted into a limited vocabulary, e.g. {a,b,c,...}. However, this vocabulary does not yield any semantics from the expert's point of view in the sense they do not provide her with an immediate understanding. For example, an increase of 15%-20% on Googles stock represented by the literal a is very poor compared to the expression *strong increase*. By contrast with this automated approach, aggregation of textual description requires an explicit model of the semantic of descriptor, e.g., an ontology.

When considering the data reduction aspect of the summary, the domains of semantic compression and semantic summarization are strongly

related. The intuition in these domains is that data reduction can be done by exploiting the underlying semantics of the data and one can use the dependency between attribute values and tuples to regroup *similar* information together. We show however that the objectives are not entirely the same. Indeed, the objective of semantic compression algorithms is to use the underlying semantics of data aiming at reducing its storage. Algorithms such as Fascicles[9], Spartan[4] and ItCompress[8] were designed for optimizing a data size. To do so, they focus on finding a subset of attributes and tuples which are *similar* enough, given some error tolerance parameters, and represent those tuples using a common representation. In the case of Spartan, this common representation is a *Classification and Regression Tree* (a.k.a. Cart) which is a prediction model. In Fascicles, Jagadish et al. reorder the data and regroup tuples that have *similar* attribute values over k attributes into a kD-fascicle. On the other hand, ItCompress keeps the ordering of the data by representing similar tuples with *Representative Rows* (RR) grouped in a separate table and outliers in another table. These semantic compression algorithms are not well suited as support structure for frequent pattern mining in financial news as: (i) the ordering of tuples is not kept (e.g. Fascicles), (ii) the number of tuples is not reduced and (iii) processing is not incremental and has high complexity.

In contrast with semantic compression techniques, semantic summarization approaches aim at representing data in a more reduced and concise form by both reducing attribute domains and tuple numerosity. Saint-Paul et al. proposed in [17] SaintEtiQ which is a linguistic summarization algorithm that uses background knowledge made of fuzzy partitions over attribute domains to build a hierarchy of summaries. Each node in the hierarchy is a summary representing a subset of the initial data. The closer to the leaves of the hierarchy, the more precise the representation. Unfortunately, this hierarchical structure does not preserve the ordering of the tuples which is crucial for conventional FPM algorithms. The summarization technique we propose was inspired by the Attribute Oriented Induction (AOI) process for supporting data mining [6]. The AOI algorithm takes as input a table of tuples of attribute-value pairs and outputs a smaller table of tuples expressed at higher conceptual levels. Provided a concept hierarchy is defined for each attribute, at each iteration of the algorithm, an attribute A_i is selected and all tuples are generalized on attribute A_i. Identical and contiguous generalized tuples are then merged together and counts maintained in a COUNT attribute. This process is repeated until the table attains a *minimum desired level of generalization* defined by the expert. The main limitations of this approach regarding FPM are: (i) the lack of control in the generalization of each attribute which could be over-generalized and lead to non appropriate information loss, (ii) its iterative aspect and (iii) the lack of temporal control in the process. Our vision is that a *tuple* oriented approach can be performed in an incremental way and benefit environments and applications

that allow limited processing steps—often one— over the data.

8 Conclusion and future work

In this paper, we have tackled the issue of designing a support structure for mining financial news. Frequent Pattern Mining in financial news has many applications among which a most desired one is to be able to anticipate future events, e.g., for marketing purposes. However, the inherent nature of financial news brings many challenges in the mining task. We have highlighted these challenges and introduced in this paper a summary structure capable of seamlessly supporting classical analysis algorithms in such environment. Our Time-Aware Content Summary represents news data in a more reduced and concise form using both its content and temporal information.

To the best of our knowledge, this is the first summary structure designed to take into account both content and temporal aspects of data. The preliminary experiments shows that the proposed summary is an inexpensive structure to build while providing a solid basis for finding patterns expressed in a higher level of abstraction (trends) in limited time (e.g. in the order of minutes). Such characteristics allow to envision a very interactive way of mining financial news. Mining over the summary gives trends over the original news data. If not satisfied with the granularity of the patterns, an expert can choose to focus on a portion of the output (e.g. patterns with news relating to *high* interest rates and *low* inflation), and reiterate the summarization and mining cycle with more precise settings (e.g. ω and $supp_{min}$). The advantage of this interactive mining approach is the selection of smaller subsets of the original news at each iteration. This interactive mining allows experts to timely access to the information they need without having to perform the mining directly on the raw news at low levels of support which can eventually not be completed in acceptable times.

References

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

[2] R. Agrawal and R. Srikant. Mining sequential patterns. In *Proc. of the 11th International Conference on Data Engineering (ICDE 1995)*, 1995.

[3] J. Ayres, J. Flannick, J. Gehrke, and T. Yiu. Sequential pattern mining using a bitmap representation. In *Proc. of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2002)*, July 2002.

[4] S. Babu, M. Garofalakis, and R. Rastogi. Spartan: A model-based semantic compression system for massive data tables. In *Proc. of the ACM SIGMOD International Conference on Management of Data (SIGMOD 2001)*, 2001.

[5] A. Budanitsky. Lexical semantic relatedness and its application in natural language processing. 1999.

[6] J. Han and Y. Fu. Exploration of the power of attribute-oriented induction in data mining. *Advances in Knowledge Discovery and Data Mining*, 1996.

[7] A. Hotho, A. Nürnberger, and G. Paass. A brief survey of text mining. *LDV Forum*, 20(1):19–62, 2005.

[8] H. Jagadish, R. Ng, B. Ooi, and A. Tung. Itcompress: an iterative semantic compression algorithm. In *Proc. of the 20th International Conference on Data Engineering (ICDE 2004)*, Apr 2004.

[9] H. V. Jagadish, J. Madar, and R. T. Ng. Semantic compression and pattern extraction with fascicles. In *Proc. of the 25th International Conference on Very Large Databases (VLDB 1999)*, 1999.

[10] D. Lin. An information-theoretic definition of similarity. In *Proc. of the 15th International Conference on Machine Learning (ICML 1998)*, pages 296–304. Morgan Kaufmann, San Francisco, CA, 1998.

[11] J. Lin, E. Keogh, S. Lonardi, and B. Chiu. A symbolic representation of time series, with implications for streaming algorithms. In *Proc. of the ACM SIGMOD International Conference on Management of Data (SIGMOD 2002)*, pages 428–439, 2002.

[12] P. Newbold. The principles of the box-jenkins approach. *Operational Research Quarterly (1970-1977)*, 26(2):397–412, July 1975.

[13] J. Pei, J. Han, B. Mortazavi-Asl, and H. Pinto. Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In *Proc. of the 17th International Conference on Data Engineering (ICDE 2001)*, 2001.

[14] Q.-K. Pham, R. Saint-Paul, B. Benetallah, G. Raschia, and N. Mouaddib. Mine your own business! mine others' news! In *Proc. of EDBT. To appear.*, 2008.

[15] R. Rada, H. Mili, E. Bicknell, and M. Blettner. Development and application of a metric on semantic nets. In *IEEE Transactions on Systems, Man and Cybernetics*, pages 17–30, 1989.

[16] P. Resnik. Using information content to evaluate semantic similarity in a taxonomy. In *IJCAI*, pages 448–453, 1995.

[17] R. Saint-Paul, G. Raschia, and N. Mouaddib. General purpose database summarization. In K. Bhm, C. S. Jensen, L. M. Haas, M. L. Kersten, P.-A. Larson, and B. C. Ooi, editors, *Proc. of the 31st International Conference on Very Large Databases (VLDB 2005)*, pages 733–744. Morgan Kaufmann Publishers, August 2005.

[18] A. Silberschatz and A. Tuzhilin. What makes patterns interesting in knowledge discovery systems. *IEEE Transactions on Knowledge and Data Engineering*, 8(6):970–974, Dec 1996.

[19] R. Srikant and R. Agrawal. Mining sequential patterns: Generalizations and performance improvements. In *Proc. of the 5th International Conference on Extending Database Technology (EDBT 1996)*, March 1996.

[20] J. Wang and J. Han. Bide: Efficeint mining of frequent closed sequences. In *Proc. of the 20th International Conference on Data Engineering (ICDE 2004)*, Apr 2004.

[21] Z. Wu and M. Palmer. Verb semantics and lexical selection. In *Proc. of the 32nd. Annual Meeting of the Association for Computational Linguistics*, pages 133–138, New Mexico State University, Las Cruces, New Mexico, 1994.

[22] D. Xin, X. Shen, Q. Mei, and J. Han. Discovering interesting patterns through user's interactive feedback. In *Proc. of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2006)*, Aug 2006.

[23] C.-C. Yu and Y.-L. Chen. Mining sequential patterns from multidimensional sequence data. *IEEE Transactions on Knowledge and Data Engineering*, 17(1):136–140, Jan 2005.

[24] M. J. Zaki. Spade: An efficient algorithm for mining frequent sequences. *Journal on Machine Learning*, 42(1/2):31–60, 2001.

[25] D. Zhang and K. Zhou. Discovering golden nuggets: data mining in financial application. *IEEE Transactions on Systems, Man and Cybernetics, Part C: Applications and Reviews*, 34:513–522, Nov 2004.

A Proof

We denote $W = \langle t'_{W_1}, ..., t'_{W_\omega} \rangle$ the sequence of ω last tuples summarized in S. Given a window of precision $\omega \geq 2$ we consecutively prove these three cases as follows:

1. Case $(|S_1| - \omega) \geq 2$**.** The last ω tuples in S_1 are in W, i.e., $W = \{t'_{W_1} = t'_{m-\omega}, ..., t'_{W_\omega} = t'_m\}$, and at least the first 2 tuples of S_1 are not in W. We denote $\overline{W} = \{t'_1, t'_2, ...\}$ the set of tuples in S_1 and not in W thereby $S = \langle \underbrace{t'_1, t'_2, ...}_{|\overline{W}|}, \underbrace{[t'_{m-\omega}, ..., t'_m]}_{|W|} \rangle$, where "[" and "]" materialize the span of the window W. †† Consequently there is a possibility for at least $|\overline{W}| \geq 1$ tuples (all except t'_1) to be the first tuple of S_2. Suppose t'_{T_1} arrives and $t'_{T_1} = t'_2 = u'_1 \in \overline{W}$, then window W is moved forward, giving $W = \langle t'_{W_2}, ..., t'_{W_\omega}, t'_2 \rangle$, $\overline{W} = \{t'_1, t'_{W_1}(= t'_{m-\omega}), ...\}$ and $S = \langle t'_1, t'_2, ..., t'_{m-\omega}, \underbrace{[t'_{m-\omega+1}, ..., t'_m, t'_2]}_{|W|} \rangle$. In the following iteration, there are $|\overline{W}| \geq 2$ choices for tuple u'_2. Recursively, we prove it is possible to have all tuples u'_i of S_2 in S. This proves that S_2 can be found in S without any requirements on α.

2. Case $(|S_1| - \omega) = 1$**.** The last ω tuples in S_1 are in W, i.e.
$W = \{t'_{W_1} = t'_2, ..., t'_{W_\omega} = t'_m\}$, $\overline{W} = \{t'_1\}$ and
$S = \langle \underbrace{t'_1}_{|\overline{W}|}, \underbrace{[t'_2, ..., t'_{W_\omega}]}_{|W|} \rangle$. If $|\overline{W}| = 1$ then $u'_1 = t'_1$ and recursively, all $u'_i = t'_i$, thus $S_2 = S_1$: $|\overline{W}| = 1$ is not acceptable. We need at least 2 tuples in $\overline{W}$ to be in the situation †† of case $(|S_1| - \omega) \geq 2$. The following incoming generalized tuple is t'_{T_1}:

If $t'_{T_1} \in S_1$, t'_{T_1} will be merged into S, recursively, $\forall t'_{T_i} \in T$ and $t'_{T_i} \in S_1$, t'_{T_i} will be merged into S. Therefore, $\alpha = 0$ is not possible, meaning at least $\alpha \geq 1$.

If $t'_{T_1} \notin S_1$, then t'_{T_1} will be added to S, W is moved forward and $\alpha \geq 1$.$\diamondsuit$ As a result, $|\overline{W}| = |\{t'_1, t'_{W_1}(= t'_2)\}|) = 2$ and $S = \langle \underbrace{t'_1, t'_2}_{|\overline{W}|}, \underbrace{[t'_3, ..., t'_{W_\omega}]}_{|W|} \rangle$. This case brings us back to the situation †† of case $(|S_1| - \omega) \geq 2$ where we showed that no more requirements are needed for α. Therefore, $\alpha \geq 1$ is the minimal condition to find S_2 as a subsequence in S.

3. Case $(|S_1| - \omega) < 1$**.** All generalized tuples in S_1 are in W, i.e.: $W = S = \langle \underbrace{[t'_1, t'_2, ..., t'_m}_{|S_1|}, \underbrace{\emptyset, \emptyset, ..., \emptyset]}_{\omega - |S_1|} \rangle$ and $\overline{W} = \emptyset$. For any incoming generalized tuple $t'_{T_i} \in T$, if $t'_{T_i} \in S_1$ then t'_{T_i} is merged into S. It is impossible to find S_2 with $\overline{W} = \emptyset$. Therefore, W needs to be filled up with $\omega - |S_1|$ distinct tuples $t'_{T_i} \in T$ and $t'_{T_i} \notin S_1$ which leads to $\alpha \geq \beta + \omega - |S_1|$ where β is a constant to be determined.

Suppose $\omega - |S_1|$ distinct generalized tuples are added to S, thus $S = \langle \underbrace{[t'_1, t'_2, ..., t'_m, t'_{T_1}, ..., t'_{T_{\omega - |S_1|}}]}_{|W|} \rangle$. For any incoming generalized tuple $t'_{T_{\omega - |S_1| + 1}} \in T$:

If $t'_{T_{\omega - |S_1| + 1}} \in W(= S)$ then $t'_{T_{\omega - |S_1|}}$ is merged into S.

If $t'_{T_{\omega-|S_1|+1}} \notin W(=S)$ then $t'_{T_{\omega-|S_1|+1}}$ is added to S and window W is moved forward, consequently:

$S = \langle \underbrace{t'_1}_{|\overline{W}|}, \underbrace{[t'_2, ..., t'_m, t'_{T_1}, ..., t'_{T_{\omega-|S_1|+1}}]}_{|W|} \rangle$, $\overline{W} = \{t'_1\}$ and $\beta \geq 1$. Similarly, we show that if incoming generalized tuple $t'_{\omega-|S_1|+2} \in T$ and $t'_{\omega-|S_1|+2} \notin W$ then $t'_{\omega-|S_1|+2}$ is added to S and window W moved forward. Consequently, $\overline{W} = \{t'_1, t'_2\}$ and $\beta \geq 2$. The situation is then the same as $\diamondsuit$ in case $(|S_1| - \omega) = 1$. Therefore, we demonstrate that $\alpha \geq 2 + \omega - |S_2|$ is the minimal condition to find S_2 as a subsequence in S.