Choosing the content of textual summaries of large time-series data sets

JIN YU, EHUD REITER, JIM HUNTER AND CHRIS MELLISH

Department of Computing Science, University of Aberdeen
Aberdeen, United Kingdom AB24 3UE
email:(jyu, ereiter, jhunter, cmellish)@csd.abdn.ac.uk

Abstract

Natural Language Generation (NLG) can be used to generate textual summaries of numeric data sets. In this paper we develop an architecture for generating short (a few sentences) summaries of large (100KB or more) time-series data sets. The architecture integrates pattern recognition, pattern abstraction, selection of the most significant patterns, microplanning (especially aggregation), and realisation. We also describe and evaluate SumTime-Turbine, a prototype system which uses this architecture to generate textual summaries of sensor data from gas turbines.

1 Introduction

It is often said in the NLP community that the world is being flooded with text, but the flood of text is insignificant compared to the flood of data. A professional writer may write as much as 1MB of text in a year; the sensors in the writer's car can easily produce this much data each day, as he or she drives to work and back. Killgarriff and Grefenstette (2003) estimate that there are 20TB of text on the World-Wide Web; 100TB or more of data is produced each day simply from sensors in aircraft engines (Hey and Trefethen 2003).

Currently numeric time-series data is usually presented to people visually (Spence 2001). However, when the graphical reports need more domain knowledge to explain, or can not be sent to end-users, textual summaries will be advantageous. For example, weather forecasting reports generated from weather data are sent to users through their mobile phones in textual format instead of graphical format. Good human-written summaries of data can certainly be effective (Law, Freer, Hunter, Logie, McIntosh and Quinn 2005), but they are prohibitively expensive to produce. The challenge for Natural Language Generation (NLG) is to produce good summaries of numerical time-series data automatically.

Previous research on generating textual summaries of data has mostly focused on applications where the summary is produced from a relatively small amount of data; for example weather forecasts (Goldberg, Driedger and Kittredge 1994; Sripada, Reiter and Davy 2003a), which typically use a few hundred bytes of text to summarise a few KB of data. Roughly similar "compression ratios" are seen in summaries of statistical data (Iordanskaja, Kim, Kittredge, Lavoie and Polguere 1992) and stock market activity (Kukich 1983). The goal of our work was to produce textual summaries of data with much higher compression, where a few sentences summarised tens or even hundreds of KB of data.

Achieving such compression rates means that we must take data analysis and abstraction seriously; we cannot simply put together something ad hoc to analyse the input, and focus our energies on the linguistic end of text generation. We also must have some model of what is likely to be important and interesting to the reader, as it is clearly impossible to communicate more than a fraction of the content of the input data to the human user.

In this paper we describe an architecture for generating "highly-compressed" textual summaries of numeric time-series data, which includes components for detecting changes in the input data, abstracting these changes into symbolic descriptors, determining which changes are most interesting and important, and generating text which describes these changes. This architecture has been used in SumTime-Turbine, an NLG system which generates summaries of sensor data from a gas turbine.

Table 1. Part of a sample of time series data in the gas turbine domain

Date	Time	TTXD-1	TTXD-2	TTXD-3	TTXD-4	TTXD-5	TTXD-6
27/11/1999	12:02:19	430.56	450.88	429.27	481.40	452.13	463.38
27/11/1999	12:02:20	430.98	451.15	429.27	481.68	451.99	463.93
27/11/1999	12:02:21	430.84	451.29	429.59	481.09	452.27	463.80
27/11/1999	12:02:22	431.12	451.15	429.27	481.40	452.27	464.21
27/11/1999	12:02:23	431.25	451.85	429.59	481.68	452.27	464.35
27/11/1999	12:02:24	431.53	452.40	429.00	481.40	451.85	464.07
27/11/1999	12:02:25	431.39	451.57	429.27	481.40	451.71	463.66
27/11/1999	12:02:26	431.25	451.57	429.41	481.26	451.99	463.52
27/11/1999	12:02:27	430.28	449.73	429.14	480.53	451.85	464.07
27/11/1999	12:02:28	430.00	449.59	429.73	480.81	452.54	464.07
27/11/1999	12:02:29	430.56	450.74	429.59	480.95	452.54	464.21
27/11/1999	12:02:30	430.56	450.74	429.41	481.68	452.40	463.80
27/11/1999	12:02:31	430.56	451.15	429.27	481.09	452.40	463.38
27/11/1999	12:02:32	431.25	451.85	429.41	481.82	452.27	463.52
27/11/1999	12:02:33	431.39	452.40	430.14	481.82	452.27	463.66
27/11/1999	12:02:34	431.25	452.13	430.28	481.82	452.82	464.80
27/11/1999	12:02:35	430.98	450.88	429.59	481.96	451.99	464.21
27/11/1999	12:02:36	430.28	450.15	429.00	481.82	452.13	463.38

The first column is the date in the format dd/mm/yy, the second column is the time in the format hh:mm:ss, and the other columns are measurements from different channels. The term 'channel' refers to a series of samples of one variable.) TTXD-1 to TTXD-6 are exhaust temperature from thermocouples 1 to 6 in ºC

SumTime-Turbine takes as input turbine sensor data which is collected and archived by TIGER, a knowledge-based system, developed by Sermatech Intelligent Applications to continuously monitor and assess the condition of the gas turbine (Milne, Nicol and Massuyes 2001). Different turbines have different numbers of sensors, but in general a turbine will have hundreds of sensors, each producing one data value per second; hence the turbine will produce hundreds of MB of sensor data per day. An example of turbine input data is shown in Table 1; this shows part of a three-hour period (from 12:00 to 15:00 on 27 Nov 99) for six data channels (TTXD-1 ~ TTXD-6) used to monitor exhaust temperature in the turbine. The original data was generated on a continuous basis; it was divided into three-hour periods for the sake of convenience. The summary produced by SumTime-Turbine from the entire three-hour period is shown in Fig.1.

[Background information]

Gas turbine: aylesford
Subsystem: exhaust temperature
Monitoring channels: TTXD-1, TTXD-2, TTXD-3, TTXD-4, TTXD-5 and TTXD-6
Turbine running state: part load
Time interval of these channels: from 12 to 15 on 27 Nov 99

[Overview information]

There were large erratic spikes in all channels at 12:59, 13:01, 13:41 and 14:40.

[Most significant patterns]

At 12:59, there were large erratic spikes in TTXD-1, TTXD-2, TTXD-3, TTXD-4, TTXD-5 and TTXD-6. These patterns violated the pairs and follows check. In more detail, there were dips with oscillatory recoveries in TTXD-3 and TTXD-4, followed 1s later by dips with oscillatory recoveries in TTXD-1, TTXD-2, TTXD-5 and TTXD-6. This occurred between 12:59:17 and 12:59:54.

Fig. 1. An example summary generated by SUMTIME-TURBINE, from the data set extracted in Table. 1.

2 Background

2.1 Gas turbines and TIGER

A gas turbine, most typically recognised as a jet engine, converts chemical energy to kinetic energy by 'ejecting' gas at high speed; it can be used in aircraft engines or in industry as a form of power generation. A gas turbine system consists of several sub-systems (Milne, Nicol, Massuyes and Quevedo 1995), such as grid, compressor, exhaust temperature, generator, etc. The turbine runs in several states, including start, crank, fire, FSNL (full speed no load), part load, base load, peak load, and normal shutdown.

In order to get the most out of a gas turbine through reducing maintenance costs and increasing its availability, TIGER receives hundreds of measurements (the exact number depends on the individual turbine) from the gas turbine controller at one second intervals and

performs extensive fault detection and diagnosis every second (Milne et al. 2001). Large amounts of time series data are archived by TIGER, which is potentially very valuable for supporting diagnosis, anomaly detection, and prediction (Milne et al. 1995). Domain users would like to explore this data and find events of interest. The volume of TIGER data is so large that it is not feasible for them to go through the graphical displays looking for such events. For example, if domain engineers look for interesting patterns in only a three-hour segment of data from six channels, they have to go through about 30 full computer-screens (resolution: 1240*768) when these data are displayed in temporal granularity 1 second. An alternative possibility is the generation of textual summaries. Since TIGER has very limited functions to summarise the archived data sets, it is worthwhile developing techniques and implementing tools to help domain users explore these data by providing them with textual summaries.

2.2 Related work

A number of previous NLG systems have been implemented to automatically generate summaries of numeric data. For example, ANA (Kukich 1983) is a system for the generation of stock market reports that consists of four modules: a fact generator, a message generator, a discourse organiser and a surface generator. StockReporter (Dale 2003), an online text generation system, takes numeric data that describes the performance of a particular stock and produces a textual summary describing how the stock performed over a user-specified reporting period. FoG (Goldberg et al. 1994) produces two distinct varieties of weather bulletins, public and marine forecasts, in both English and French, starting from forecast data produced by an atmospheric-modelling program. Although these systems have used sophisticated NLG technology for linguistic processing, none of them have used advanced data analysis techniques for analysing the input data. This may be because the input to these systems was relatively small, with limited dependencies and interactions between data channels. Such data sets are relatively easy to analyse.

For example, in ANA, the input data are stock prices at half-hourly intervals retrieved from the Dow Jones New Service database. In FoG, the input data are several key meteorological parameters sampled every hour. For StockReporter, the input data are sampled at one-day intervals. It is the task of content determination to decide what information from the input data should be included. So in ANA, the fact generator performs simple arithmetic calculations to compute relative and cumulative degree and direction of change for each interval, as well as high and low points for the day. In FoG, the STM module extracts weather concepts from the input. Other systems do not include a content determination function or just allow for a simple

form of this. For example, in GoalGetter (Klabbers, Krahmer and Theune, 1998), a data-to-speech system that generates reports of football matches in Dutch based on tabular data, all input data are expressed in the output, so no content determination is needed.

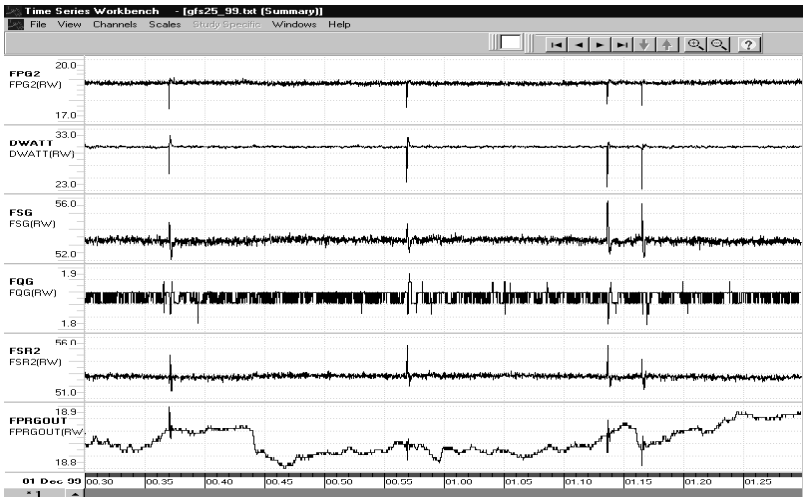
But in the gas turbine domain, we are facing a quite different situation. The data sets are large, and users are interested in the relationships between channels as well as in individual channels. It is impossible to communicate all of the input data in the output summaries, so data analysis (essentially content determination) is of major concern in SumTime-Turbine.

SumTime-Turbine was developed as part of the more general SumTime project. Other SumTime systems include SumTime-Mousam (Sripada et al. 2003a) which (like FoG) generates weather forecasts from numerical atmosphere simulations, and SumTime-Neonate (Sripada, Reiter, Hunter and Yu 2003b), which generates summaries of sensor data for babies in neonatal intensive care. SumTime-Mousam, like FoG, used a relatively small input data set. SumTime-Neonate is more like SumTime-Turbine in that it uses larger input data sets (tens of KB); this system is still being developed, and is benefiting from the lessons learned from SumTime-Turbine.

3 Knowledge acquisition and requirements analysis

A system which produces text summaries of data must have a data analysis component to identify events and changes in the data that should be reported; and a natural language module to generate a text describing these events and changes. In order to better understand these tasks, we conducted a series of knowledge acquisition exercises (Reiter, Sripada and Robertson 2003) with domain experts from Sermatech Intelligent Applications. These included:

.



%FPG2 Intervalve gas fuel pressure input %DWATT Generator load
%FSR2 Gas fuel stroke reference from fuel splitter %FQG Gas fuel flow
%FPRGOUT Gas ratio valve servo command %FSG Fuel stroke reference

Fig. 2. Visualisation of channels from one of the subsystems

> There are four major sets of spikes in this time. FPG2 pressure has huge drops at here, here, and here. When drops occur, the power output of this case makes a big drop, which is staggering 7 Mwatt, which is extremely unhealthy. Both the fuel valves moved. And here we have got the almost same situation again: the power dropped, the pressure dropped, the fuel flow has dropped off a little bit here. The set point is coming up to slightly delay there the set point coming up to recover the power output while it puts flow back. We have got two sets at the same time, if we look at timing of spikes of power output. When the power dropped, the fuel valve is opened up to compensate, which brought the power back. There is a small amount of overshoot that's taken place in two channels. Here fuel flow drops are more or less most identical spikes. A drop in the power output the fuel valve has come right up to compensate the pressure between the valves. It is strange that there isn't a big drop in fuel flow at that point. In this period there were three of these spikes that affected the pressure, the power output and the fuel valve. And it is interesting that it isn't obvious that the fuel flow had a spike at the same time here.

Fig. 3. Transcript of a 'think aloud' protocol taken from a domain expert

- showing example data to the experts, and observing the way they analysed it;
- showing example data to the experts, and asking them to verbally describe it - an example of such a description is shown in Fig. 3; this is based on the data shown in Fig.2;
- working with experts to build an ontology of patterns;
- asking experts to comment on and critique summaries generated by early versions of SumTime-Turbine.

Based on these exercises, we identified the procedures taken by domain experts to perform summarization, and created a list of requirements and desirable features.

3.1 Summarising procedures by domain experts

By observing how the experts summarised the data, we found that they always went through three main steps to analyse the data:

- overview. They prefer to get an initial overview of the data. They often initially choose a coarse visualisation which shows an entire 3-hour data set. In this way they can get as much information as possible from the data. At this stage, they often make general comments about the data channels. See, for example, the first sentence in the above transcript: *There are four major sets of spikes in this time.*

- zoom and filter. They focus on patterns of interest in the data channels by zooming in and filtering out uninteresting patterns based on their domain knowledge. For example,

they chose the patterns occurring at around 01:16 across the six channels since *it is strange that there isn't a big drop in fuel flow at that point. And it is interesting that it isn't obvious that the fuel flow had a spike at the same time here.*

- details of interesting patterns. They get details of these interesting patterns such as their shapes and ranges. For example, a more detailed description of the patterns occurring around 01:16 across the six channels is: *There is a small amount of overshoot that's taken place in two channels. Here fuel flow drops are more or less most identical spikes. A drop in the power output the fuel valve has come right up to compensate the pressure between the valves.*

We decided that the procedures described above could be regarded as an instance of the visual information seeking mantra: overview first, zoom and filter, then details-on-demand. This has been used as a principle to provide a useful starting point for designing advanced graphical user interfaces (Shneiderman 1996). We seem to have found that the visual information seeking mantra also provides a useful starting point for summarising time series data.

In summary, through recognising possibly interesting patterns, grouping related patterns, selecting interesting patterns after filtering uninteresting patterns, domain experts summarise the data by getting an 'overview' in the context of a coarse visualisation and then access 'details-on-demand' of interesting patterns in the context of a finer-grained visualisation.

3.2 Data analysis requirements

A very important issue is the type of information communicated in the summary. Through studying the corpus which we collected during KA activities, we have made the following observations.

Experts focused on patterns in data channels. Often, there are two kinds of patterns in the data: patterns occurring over short time intervals and over longer time intervals. In the gas turbine domain, the experts concentrated on the former such as spikes, oscillations, and steps. Little mention was made of the latter such as gradual rises and falls in channel values. Such changes are however important in other domains such as descriptions of hospital sensor data (Sripada et al. 2003b), so they should be allowed for in a general architecture.

When describing patterns from the same time subsequence, experts chose different words in different contexts. When they described the patterns in general, they used more abstract words

such as *spikes*, *oscillations*, whereas they used more concrete concepts such as *dip with oscillatory, damped oscillations* to describe the same patterns in detail.

Typically experts used pattern names such as *spike, step*, with qualitative descriptors such as *large, small*. They also often stated when a pattern was abnormal, e.g. *which is extremely unhealthy*.

The corpus also suggests that it is important to bring together simultaneous patterns in related channels; this seems to be important in other domains as well. The experts also explicitly told us that related channels should be grouped by subsystem.

The corpus also includes many cases where experts interpret as well as describe patterns. For example, the transcript in Fig. 3 includes interpretations such as *which is extremely unhealthy*. *It is strange that there isn't a big drop in fuel flow at that point*. However, experts seem to disagree about interpretations more than they disagree about descriptions of the raw data (Shahar and Musen 1996). Also, interpretation is difficult to get 100% right, and an incorrect interpretation can be dangerous. Hence we decided to focus on describing data, and perform only a small amount of interpretation.

Last, but certainly not least, experts were selective in the patterns they chose to describe in their summaries. See, for example, a more detailed description of the patterns occurring around 01:16 across the six channels: *There is a small amount of overshoot that's taken place in two channels. Here fuel flow drops are more or less most identical spikes. A drop in the power output the fuel valve has come right up to compensate the pressure between the valves. It is strange that there isn't a big drop in fuel flow at that point*.

3.3 Linguistic requirements

Linguistic requirements were harder to extract from the transcripts, and instead evolved over time as we worked with the experts and showed them versions of SumTime-Turbine.

Structurally, the experts typically started with an overview, such as *there are four major sets of spikes* in the transcript (Fig. 3.); and then continued with descriptions of individual events. The description of individual events was often, but not always, in temporal order.

Lexically, experts sometimes used quite idiosyncratic terms, such as *bathtub*. However, as generated summaries could be read by many individuals, we decided to avoid idiosyncratic words, and instead use a fairly generic vocabulary. We did not see experts varying word usage purely for variety's sake (as recommended by many writing textbooks), instead they preferred to

consistently use the same word for a concept. An ontology for various patterns was constructed with the help of domain expert in Fig. 4.

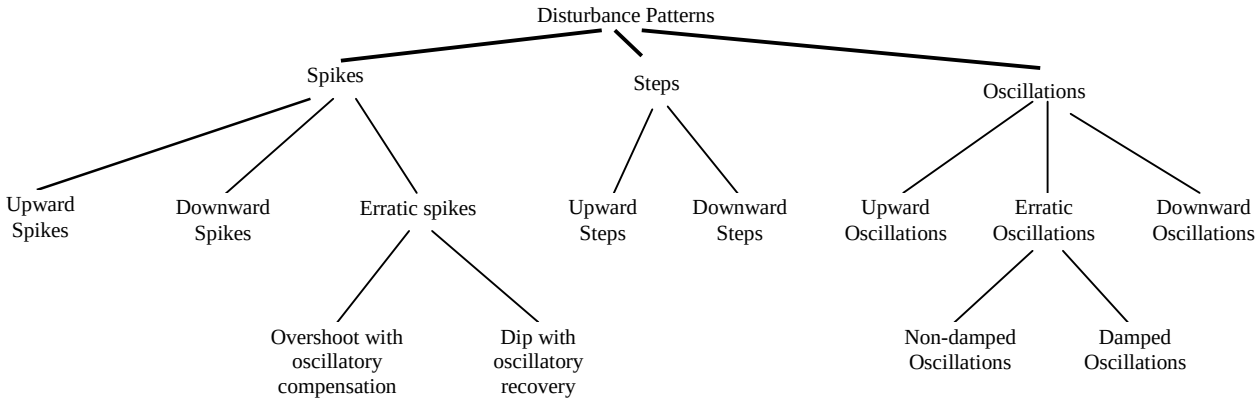


Fig. 4. Patterns ontology in the gas turbine domain

Aggregation was very common in the texts, and indeed good aggregation seems to be quite important for producing satisfactory texts. Referring expressions were rare outside interpretative texts.

Finally, with regard to realisation, the transcripts suggested that we could just use normal English. Of course, there are some domains, such as meteorology, which have established sublanguages which are quite different from conventional English, but this was not the case for SumTime-Turbine.

Overall, given that content determination is effectively done by data analysis, and that realisation is standard, the key NLG task is microplanning, especially aggregation.

4 Development of the architecture of SumTime-Turbine

Domain experts summarise time series data manually in a top-down fashion through temporal decomposition. We decided to use a bottom-up approach in SUMTIME-TURBINE based on temporal abstraction, as this seemed a better fit to the data analysis algorithms we thought would be most appropriate. Temporal abstraction, which combines smaller temporal entities into larger, but less detailed, concepts, is the inverse operation of temporal decomposition.

4.1 Temporal Granularity (TG)

When viewing time-series visually, a key parameter is the temporal scale; for example, does 1 cm on the visualisation correspond to 1 minute, 10 minutes, an hour, etc. The visualisation tool (TSW) (Hunter 2003) that we used during our knowledge acquisition sessions allowed data to be expressed at various scales, of which the most popular (amongst the experts) were 1 sec, 5

sec, and 10 sec. These values originally corresponded to the amount of time associated with each pixel (e.g. in the 10 sec scale, each pixel represented data averaged over 10 seconds), although later versions of the tool interpret the scales more abstractly. In any case, the knowledge acquisition sessions suggested that how experts described the data depended to some degree on the time scale; in particular they tended to use detailed pattern names (such as "*dip with oscillatory recovery*") when looking at detailed 1-sec visualisations, but more generic pattern names (such as "*spike*") when looking at coarser 5-sec or 10-sec visualisations.

Based on this observation, we introduced the concept of *temporal granularity (TG)* in SumTime-Turbine. SumTime-Turbine's pattern analysis components can operate with different TGs; essentially when using a TG of 1 sec they use the pattern concepts (such as "dip with oscillatory recovery") that experts used when examining data visualised at a 1-sec time scale; when using a TG of 5 sec they use the concepts that experts used when examining data visualised at a 5-sec time scale, etc.

A sample of time series data channels displayed in different TGs by TSW is shown in Fig. 5. The top graph shows about three-hour time series data in TG 10s, which domain experts use to get an overview of all patterns in these channels. The middle two graphs show patterns of interest from the same data in the context of TG 5s, which gives more detailed information about the patterns than in the context of TG 10s. The set of graphs in the bottom is part of the same data displayed in the context of TG 1s, which provides more detailed information of the patterns of interest than the other two. As mentioned above, experts use different concepts to describe the patterns from the same data subsequence in the context of different TGs. For example, a time sequence is named an '*erratic spike*' in the middle graph displayed in the context of TG 5s, while it is named '*dip with oscillatory recovery*' in the bottom graph displayed in the context of TG 1s (see Fig. 5). Domain experts use temporal decomposition to describe patterns of interest, which means larger, but less detailed, concepts are broken into more-detailed entities.

4.2 The architecture of SUMTIME-TURBINE

Based on the foregoing analysis, we have identified a series of generic tasks that need to be performed in order to generate textual summaries of the gas-turbine data:

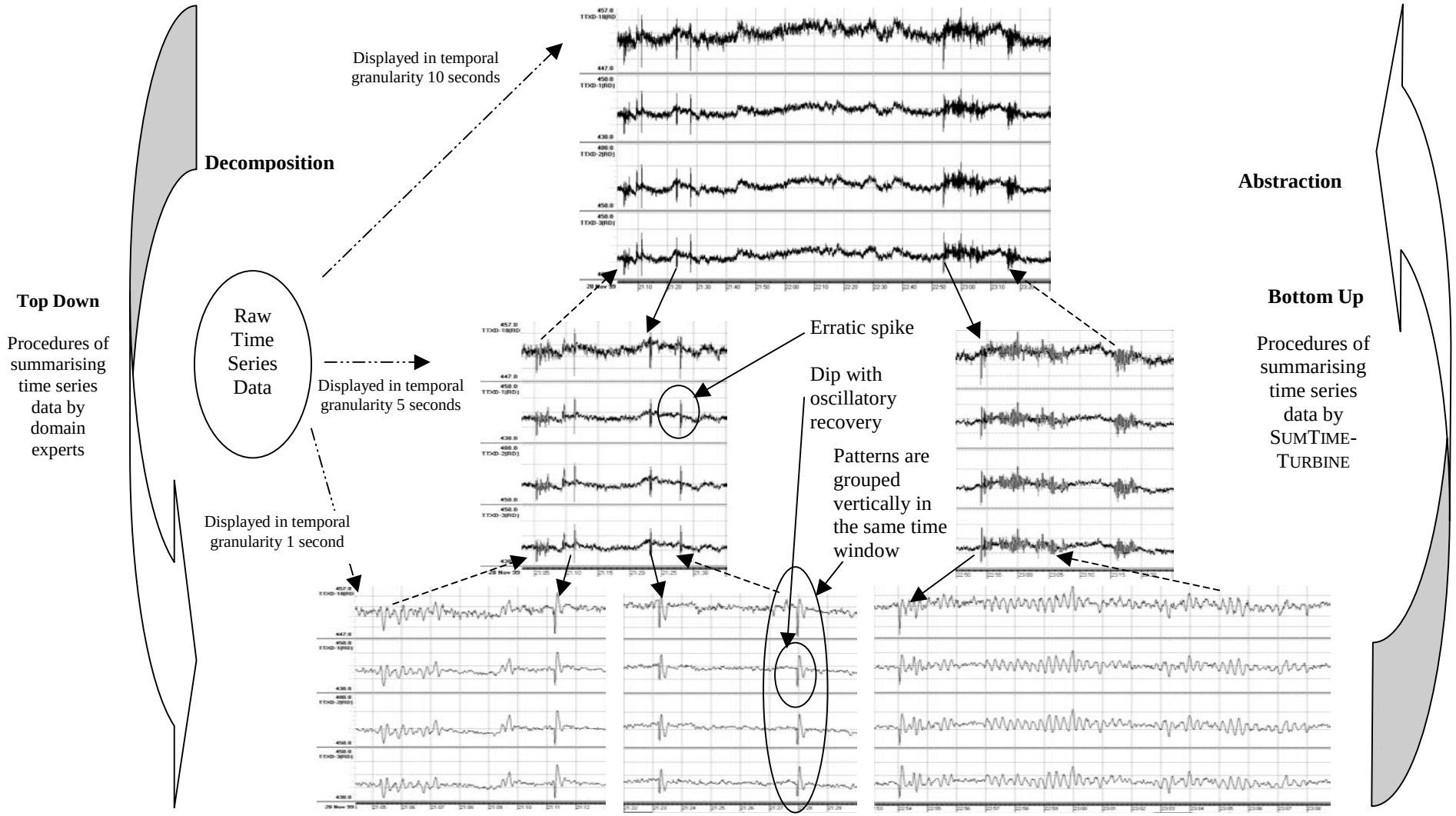


Fig. 5. Summarising procedures by domain experts and by SUMTIME-TURBINE

1. *Pattern recognition* – we need to identify potentially interesting patterns or events; that is, we need to pick out the sorts of things that experts mentioned in the KA sessions.
2. *Pattern abstraction* – we need to form a conceptual description of these patterns, using names and qualitative descriptors such as *spike* and *large*. That is, we need to associate with patterns the sort of qualitative descriptors that we observed in the KA sessions. We may also wish to describe simultaneous patterns as a related group, again as observed in KA sessions.
3. *Pattern selection* – we need to select which of the patterns we have identified and abstracted should actually be mentioned in the summary text. This is absolutely essential when generating highly-compressed summaries; we cannot describe the entire data set in detail, so we must select those things which are most likely to be interesting and relevant to the user.
4. *Summary generation* – once content is selected, we need to perform other NLG tasks, such as microplanning and realisation.

A more detailed architecture is given in Fig. 6. It includes nine modules which are grouped into the above four components.

1. *Pattern recognition*
2. *Pattern abstraction* including
   - Pattern temporal abstraction
   - Multiple pattern formation
   - Multiple pattern temporal abstraction
3. *Interesting pattern selection* using
   - Objective measures of interestingness of multiple patterns
   - Subjective measures of interestingness of multiple patterns
4. *Summary generation* including
   - Structure decision
   - Micro-planning
   - Surface realisation

From the perspective of the three-stage pipeline architecture of NLG (Reiter and Dale 2000), the first three components perform content determination, and the last component does the rest of the NLG pipeline.

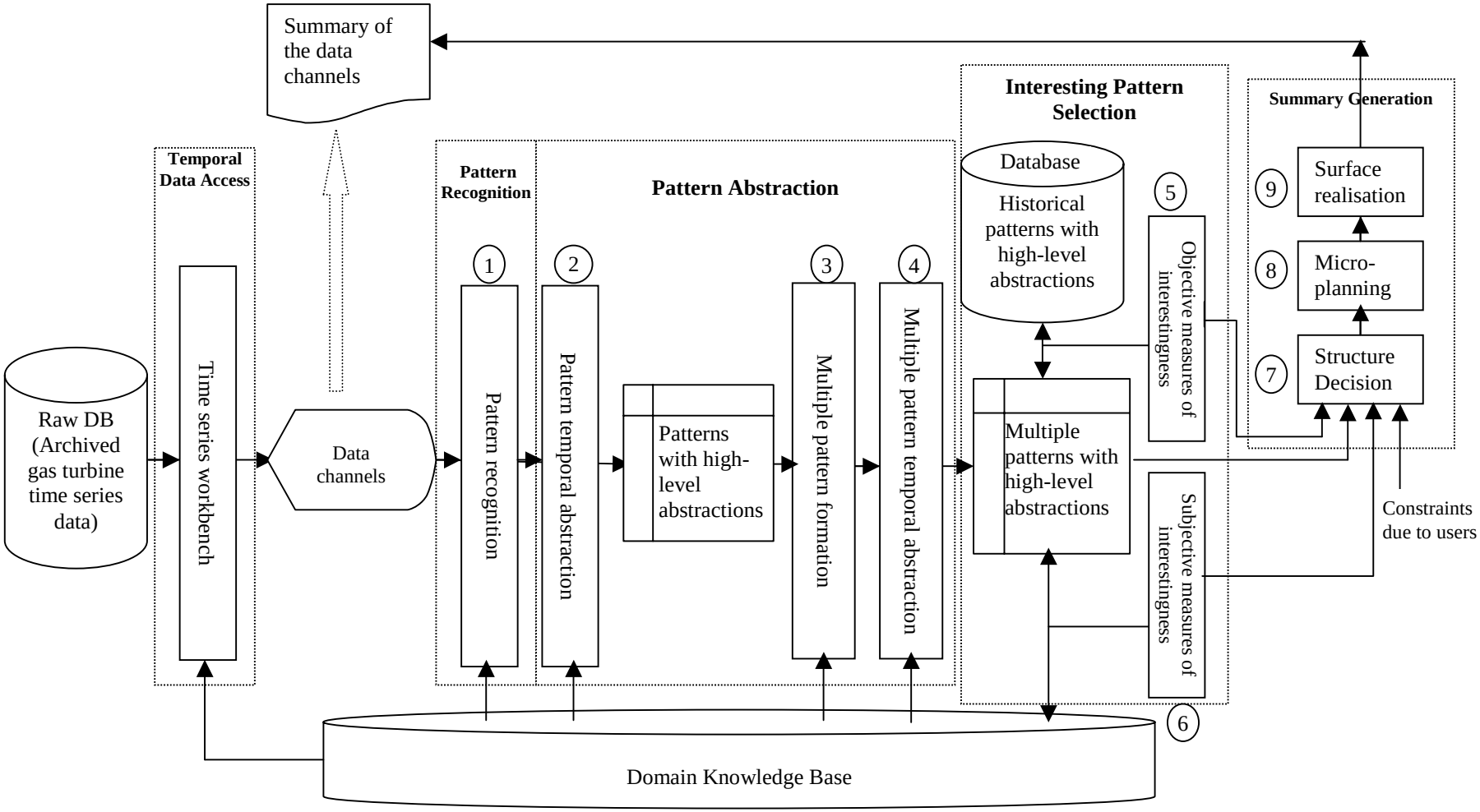


Fig. 6. Architecture of SUMTIME-TURBINE

5 Techniques in SumTime-Turbine

There are four components: *pattern recognition*, *pattern abstraction*, *interesting pattern selection* and *summary generation* in SumTime-Turbine. We will briefly explain the main techniques used in each component in the following subsections.

5.1 Pattern recognition

Pattern recognition is the first component in the architecture. Its function is to automatically connect particular segments of time series data to concepts represented by the leaves of the pattern ontology tree in Fig. 4 in the context of TG 1s.

Many pattern recognition algorithms have been proposed to recognise specific patterns in time series data. For example, there is a standard spike detection algorithm which compares the signal value with a preset threshold. Many researchers have proposed algorithms to determine automatically the optimal threshold values (Rebrik, Wright, Emondi and Miller 1999) (Soto, Manjarrez and Vega 1997). However, our goal is to classify patterns into an ontology, not identify specific types of patterns.

We designed a pattern recognition algorithm, which is composed of a pattern locator and a pattern classifier, to recognise patterns from time series data. The algorithm works in two main steps: firstly disturbance patterns in data channels are identified by the pattern locator; and then the identified disturbance patterns are further classified into their sub-patterns (represented by leaves of the pattern ontology tree) by the pattern classifier. The pattern classifier is formed by adapting landmark models (Perng, Wang, Zhang and Parker 2000) and shape description language (Agrawal, Psaila, Wimmers and Zait 1995). More information about the pattern recognition algorithm is given in (Yu, Hunter, Reiter and Sripada 2002).

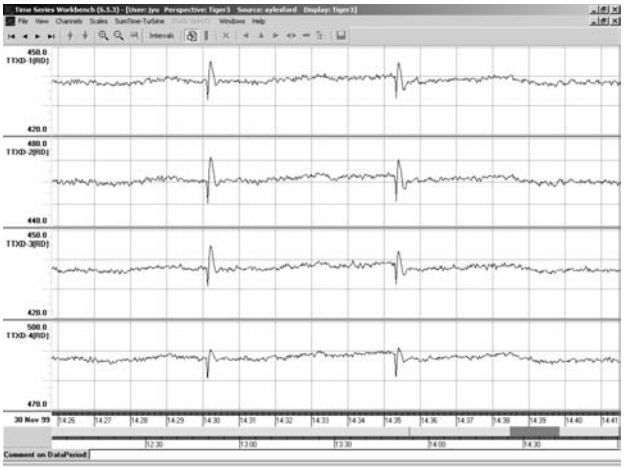


Fig. 7. Four channels input to the *pattern recognition* component

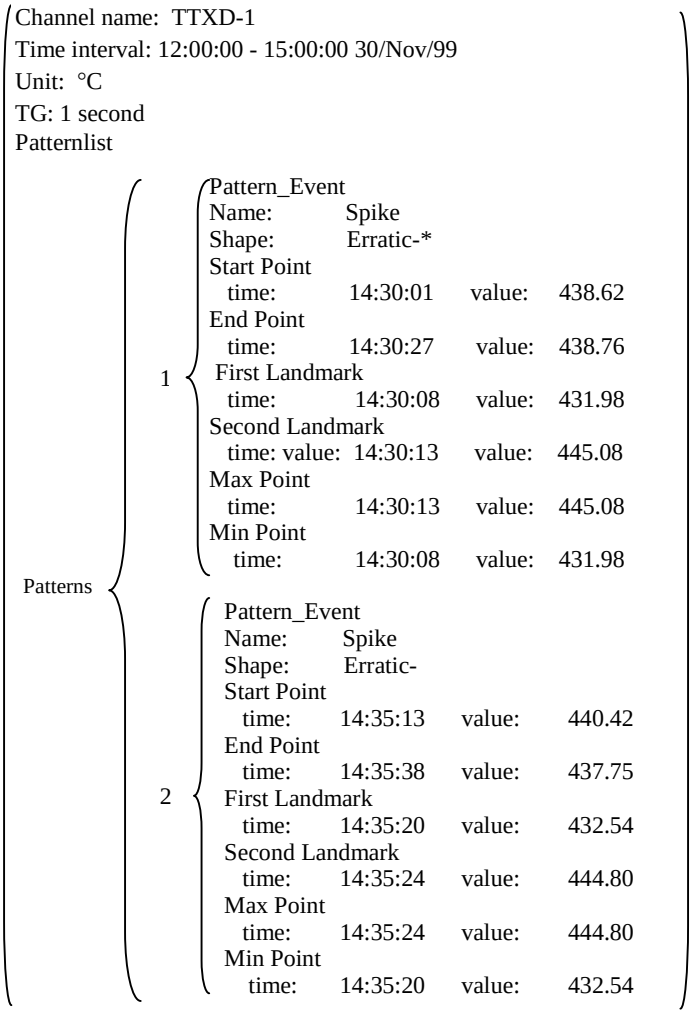


Fig. 8. A data structure saving output from the *pattern recognition* component
* 'Erratic- Spike' is used as a shorter synonym for the pattern: 'dip with oscillatory recovery'

The component receives as input the raw channel data and produces as output a set of features of patterns recognised in the channels; these are kept in a data structure that is output to the next component. In more detail, the features are:

- information about the channels: channel names, the time intervals of the channels, units of measurement, and the TG in which the channels are displayed and patterns are recognised.
- information about patterns recognised in the channels: the pattern's name, the start and end points of the pattern, the landmarks, and the maximum and minimum points of the pattern.

For example, the four channels (TTXD-1 ~ TTXD-4) in the Exhaust Temperature subsystem (Fig. 7) are input to the *pattern recognition* component. Parts of its output, which are two patterns in TTXD-1, are kept in a data structure shown in Fig. 8.

5.2 Pattern abstraction

The second component, *pattern abstraction*, includes three modules: *pattern temporal abstraction*, *multiple pattern formation* and *multiple pattern temporal abstraction*. It produces high-level abstractions from the patterns detected by the pattern recogniser. SumTime-Turbine *pattern abstraction* is based on the knowledge based temporal abstraction framework (KBTA) (Shahar 1997).

Although several approaches (Downs, Walker and Blum 1986) (Kahn 1991) (Haimovitz and Kohane 1993) have been proposed to perform temporal abstraction of time-oriented clinical data, none of these approaches emphasised the need for a formal representation that facilitates acquisition, maintenance, sharing and reuse of the required temporal-abstraction knowledge (Shahar and Musen 1996). A general framework for abstraction of time-stamped data, and in particular of clinical data, called KBTA, was proposed in (Shahar 1997). This framework supports knowledge acquisition, maintenance, sharing and reuse for the task of temporal abstraction in the domain. The KBTA method has been implemented in the RESUME system (Shahar and Musen 1996) and has been evaluated within several clinical domains, such as protocol-based care (and three of its subdomains), monitoring of children's growth (Kuilboer, Shahar, Wilson and Musen 1993), and therapy of insulin-dependent diabetes patients (Shahar 1994), etc. A comparison of frameworks applied in clinical domains to the KBTA was presented in (Shahar 1997). Shahar's KBTA framework was selected and adapted for the gas turbine domain.

In this framework, a temporal abstraction (TA) task is decomposed into five parallel subtasks which are solved by five TA mechanisms. The TA mechanisms require four types of domain-specific knowledge, which are often represented as three types of TA ontology as follows:

- *parameter ontology*. A theory of the relevant parameters and their temporal properties in the domain and the relations among these parameters (e.g. IS-A, ABSTRACTED-INTO).
- *event ontology*. This includes event interrelations (e.g. PART-OF relations) and properties.
- *context ontology*. This includes relations among interpretation contexts (e.g. the SUBCONTEXT relation).

In the gas turbine domain, we have introduced several new abstraction types such as range, length, shape, etc., and built a series of parameter ontologies and context ontologies, and

designed several temporal abstraction algorithms to obtain different high-level abstractions of patterns in the context of different TG 1s and TG 5s (more information is given in (Yu 2004)). The *pattern temporal abstraction* module receives as input a set of features of patterns and produces as output high-level abstractions of the patterns in these contexts. From domain knowledge, we know that patterns occurring among related channels within a same time window could form multiple patterns. This is the function of the *multiple pattern formation* module. The *multiple pattern temporal abstraction* module takes as input patterns' high-level abstractions in the context of TG 1s and 5s and generates high-level abstractions of multiple patterns in the same contexts.

For example, parts of the input to the *pattern temporal abstraction* module are shown in Fig. 8. Its outputs are patterns' high-level abstractions in the context of TG 1s and TG 5s, one of which is shown in. Fig. 9.

The output format: {<PatternName, Length[1], Range, Shape>, <TimeInterval>, TG, ChannelName}

In TTXD-1
{< spike,null,large,erratic>, < 14:30 14:30 >, 5s, TTXD-1}
{< spike,null,large,erratic>, < 14:35 14:35 >, 5s, TTXD-1}
In TTXD-2
{< spike,null,large,erratic>, < 14:30 14:30 >, 5s, TTXD-2}
{< spike,null,large,erratic>, < 14:35 14:35 >, 5s, TTXD-2}
In TTXD-3
{< spike,null,large,erratic>, < 14:30 14:30 >, 5s, TTXD-3}
{< spike,null,medium,erratic>, < 14:35 14:35 >, 5s, TTXD-3}
In TTXD-4
{< spike,null,large,erratic>, < 14:30 14:30 >, 5s, TTXD-4}
{< spike,null,large,erratic>, < 14:35 14:35 >, 5s, TTXD-4}

Fig. 9. High-level abstractions of patterns in the context of TG 5s

Through the *multiple pattern formation* module, the patterns in Fig. 7 are grouped to form two multiple patterns (see Fig. 10).

The output format: {<PatternName, Length, Range, Shape>, <MultiplePatternTimeInterval>, TG, ChannelName}

Multiple Pattern 1
{< spike,null,large,erratic>, < 14:30 14:30 >, 5s, TTXD-1}
{< spike,null,large,erratic>, < 14:30 14:30 >, 5s, TTXD-2}
{< spike,null,large,erratic>, < 14:30 14:30 >, 5s, TTXD-3}
{< spike,null,large,erratic>, < 14:30 14:30 >, 5s, TTXD-4}

Multiple Pattern 2
{< spike,null,large,erratic>, < 14:35 14:35 >, 5s, TTXD-1}
{< spike,null,large,erratic>, < 14:35 14:35 >, 5s, TTXD-2}
{< spike,null,medium,erratic>, < 14:35 14:35 >, 5s, TTXD-3}
{< spike,null,large,erratic>, < 14:35 14:35 >, 5s, TTXD-4}

Fig. 10. A sample output of the *multiple pattern formation* module

[1] For spikes and steps, the value of 'Length' is set to the descriptor 'null'. We only consider the length of oscillations.

The *multiple pattern temporal abstraction* module takes as input the patterns' high-level abstractions in the context of TG 1s and TG 5s and generates as output high-level abstractions of multiple patterns in the same contexts. Fig. 11 shows the output in the context of TG 5s.

Output format: {<MultiName, MultiLength, MultiRange, MultiShape>, <MultiTimeInterval>, TG, MultiChannel}

Multiple-Pattern 1

{ < spikes,null,large,erratic>, <14:30 14:30>, 5s, all }

Multiple-Pattern 2

{ < spikes,null,variant,erratic>, <14:35 14:35>, 5s, all }

Fig. 11. A sample output of the *multiple pattern temporal abstraction* module

5.3 Interesting Pattern Selection

The third component: *interesting pattern selection*, decides which of the abstract patterns produced by the previous component are sufficiently important to be mentioned in the generated text. In SumTime-Turbine, this task can be performed in two different ways: using domain knowledge (*subjective* model) or using historical pattern frequency (*objective* model).

In the data mining community, various measures of 'interestingness' of patterns have been defined with the goal of developing KDD (knowledge discovery in database) tools that discover only patterns that are interesting according to these measures (Silberschatz and Tuzhilin 1996). Generally, measures of interestingness of discovered patterns can be divided into 'subjective' and 'objective' measures. Objective measures depend only on the structure of a pattern and the underlying data used in the discovery process (such as the frequency with which combinations of items appear in sales transactions). The subjective measures depend on the class of users who examine the pattern (Agrawal and Srikant 1994).

We adopted both types of measures in SUMTIME-TURBINE. For a new multiple-pattern, a common method of deciding whether it is interesting or not is to use domain knowledge implemented as "measures of interestingness based on abnormality", which obviously belongs to "subjective measures of interestingness". A parameter ontology and several context ontologies were built and an algorithm designed to obtain such subjective measures of interestingness (more information is given in (Yu 2004)). Another method is to use the general heuristic that unusual (low frequency) events are more worthy of reporting than common events as suggested by Kahn, Fagan and Tu (1991) and Maybury (1995). In SUMTIME-TURBINE, a historical multiple-pattern database keeps all multiple-patterns that occurred from a specific time in the past to the current time. A new multiple-pattern can be compared with

multiple-patterns in the historical pattern database to decide whether it is unusual or not. This method is implemented as "measures of interestingness based on unusualness", which belongs to "objective measures of interestingness" (more information is given in (Yu 2004)).

The module called *subjective measures of interestingness* takes as input a set of new multiple-patterns and domain knowledge and generates an abnormality-based rank list with sorted multiple-patterns. The module of *objective measures of interestingness* takes the set of new multiple-patterns and historical multiple-patterns from the database and outputs an unusualness-based rank list with sorted multiple-patterns.

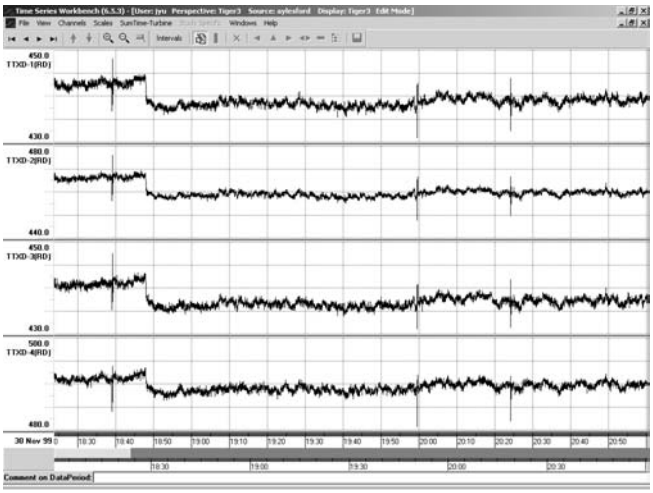


Fig. 12. Four channels input to the subjective and objective measures modules

For example, consider four channels TTXD-1 ~ TTXD-4 starting from 18:00:00 on 30/11/99 over three-hour period in the subsystem Exhaust Temperature (see Fig. 12).

Based on domain knowledge such as the 'pairs and follows' check (Milne et al. 2001) and the data in Fig. 12, a sample output of the *subjective measures of interestingness* module is listed in Fig. 13. The four multiple-patterns are sorted by their abnormalities ('1' is the highest rank). The more interesting the multiple-pattern is, the higher its rank. The multiple-pattern occurring between 20:24:03 and 20:24:32 is the most significant amongst the four multiple-patterns based on their abnormalities.

Abnormality-based Rank	Multiple Patterns
1	between 20:24:03 and 20:24:32
2	between 18:39:02 and 18:39:30
3	between 19:59:27 and 19:59:51
4	between 18:47:52 and 18:48:03

Fig. 13. A sample output of the *subjective measures of interestingness* module

A sample output of the *objective measures of interestingness* module is shown in Fig. 14. The four multiple patterns in Fig.12 were compared with historical multiple patterns saved in the historical database between 03:00:00 25/11/99 and 18:00:00 30/11/99 and the number of

their similar multiple-patterns were computed. From the sample output in Fig. 14, we can see that there are 3 similar multiple patterns for the multiple-pattern between 20:24:03 and 20:24:32, so it is ranked '1' in the unusualness-based rank list, which means that it is most significant amongst the four multiple-patterns.

Objective measures are based on historical multiple patterns kept from 03:00:00 25/11/99 to 18:00:00 30/11/99

The multiple-pattern between 18:39:02 and 18:39:30 has 10 similar multiple patterns
The multiple-pattern between 18:47:52 and 18:48:03 has 14 similar multiple patterns
The multiple-pattern between 19:59:27 and 19:59:51 has 5 similar multiple patterns
The multiple-pattern between 20:24:03 and 20:24:32 has 3 similar multiple patterns

Unusualness-based Rank	Multiple Patterns
1	between 20:24:03 and 20:24:32
2	between 19:59:27 and 19:59:51
3	between 18:39:02 and 18:39:30
4	between 18:47:52 and 18:48:03

Fig. 14. A sample output of the *objective measures of interestingness* module

Currently, only the most interesting multiple-pattern is selected to communicate in summaries based on subjective or objective measures in the prototype system. A combined measure is also discussed in (Yu 2004) when the most interesting multiple-pattern is selected based on both subjective and objective measures.

5.4 Summary generation

The first three components of SUMTIME-TURBINE deal with what information in gas turbine time series should be communicated, which is the task of content determination. Other natural language generation tasks such as document structuring, lexicalisation, and aggregation are carried out in the fourth component. Once the contents of summaries have been chosen, they will be grouped and related in rhetorical terms through the structure decision module. Surface texts are then generated using a microplanner and realiser. The main challenge is aggregation in the microplanner, which is described below; otherwise linguistic processing in SumTime-Turbine is done using quite simple techniques.

We adopted the following SCHEMAS (McKeown 1985) for summary structuring.

- Part One: background information;
- Part Two: overall description;
- Part Three: most significant patterns.

The content in each part is organised in a different order based on rules obtained from corpus analysis. For example, we used the following rules to form a sentence and arrange constituents in a sentence in Part Two.

First sentence: description of specific patterns described in time order.
Second sentence: description of vague patterns described in time order.

For example, based on the above rules, two sentences are formed to describe patterns:

> There were large erratic oscillations with short period in all channels at 18:17, large spikes in all channels at 18:40, 18:48 and 20:21. There were variant patterns in all channels at 18:03, 19:30 and 20:44.

The first sentence is a description of specific patterns and the second is a description of vague patterns. In each sentence, constituents are arranged in time order.

Aggregation algorithms are used in many NLG systems to generate more complex sentence structures. For example, in the system PLANDoc (McKeown, Kukich and Shaw 1994), a bottom-up 4-step algorithm was developed (Shaw 1995): sorting, merging same attributes, identity deletion and sentence breaking. And the aggregation task was carried out on a series of messages, or semantic functional descriptions in PLANDoc.

In SUMTIME-TURBINE, aggregation is carried out on the abstract patterns produced and selected by previous modules. Aggregation is essentially carried out using several domain-specific algorithms that groups patterns by shared features. For example, the algorithm used to aggregate patterns for Part Two of the summary groups patterns by *pattern names*, and then looks for common shared features. The two patterns shown in Figure. 11 are both *spike*, so they are aggregated. They share the value *erratic* for the *shape* feature, so this is included in the aggregated phrase; but they have different values for the *range* feature, so this is not mentioned. The time feature is however mentioned for all patterns, even if it differs. Thus, the result of this algorithm is:

Agg_MultiP = (<*Agg_MultiName, Agg_MultiLength, Agg_MultiRange, Agg_MultiShape*>, <Agg_MultiTimeIntervalStart>, Agg_MultiChannel)

Agg_MultiP = (<spikes, null, variant, erratic>, <14:30 and 14:35>, all)

For each aggregated high-level abstraction of patterns, we did simple lexicalisation of it in three steps using the notations and methods introduced in (Reiter and Dale 2000):

- create simple templates for proto-phrase specifications;
- create messages from the aggregated high-level abstractions of patterns;
- produce proto-phrase specifications by applying the templates to the messages.

A proto-phrase specification for the Agg_MultiP can be obtained as shown in Fig.15 through the three-step simple lexicalisation.

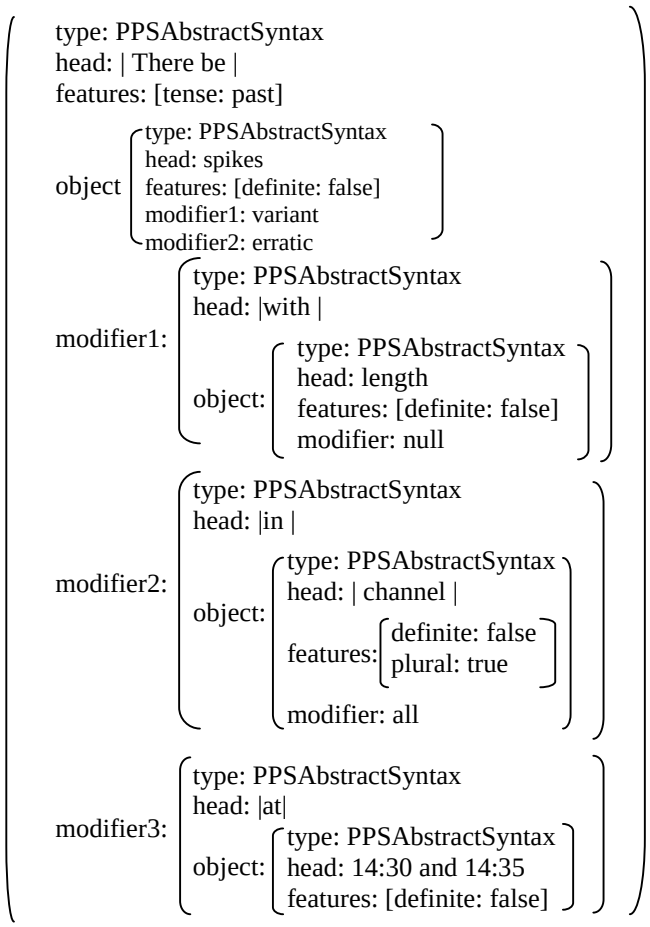


Fig. 15. The proto-phrase specification for the aggregated multiple patterns

A sentence could be produced from the above proto-phrase specification, but in order to obtain a better text, some operations are performed in the simple lexicalisation process. For example, if the value of the aggregated multiple-pattern's shape (*Agg_MultiShape*) and range (*Agg_MultiRange*) are VARIANT or the value of the aggregated multiple-pattern's *Agg_MultiLength* is NULL, they will not appear in the sentence produced from the proto-phrase specification. So we have the following sentence to describe the two multiple-patterns shown in Fig. 11.

There were erratic spikes in all channels at 14:30 and 14:35

6 Evaluation of SumTime-Turbine

Many methods have been developed to evaluate NLG systems. For example, several new extrinsic and intrinsic methods of evaluating text summaries were developed in the TIPSTER Text Summarisation Evaluation (SUMMAC) (Mani, Klein, House, Hirschman, Firmin and Sundheim 2002). Various approaches to the evaluation of NLG systems, such as accuracy evaluation, fluency/intelligibility evaluation and task evaluation, were surveyed in (Mellish and

Dale 1998). We evaluated SUMTIME-TURBINE with two domain experts from Sermatech Intelligent Applications at both the component level and the system level. Component evaluations were done using a variety of methods set out in Section 6.1. System evaluation was done by asking the domain experts directly to assess its final summaries, as described in Section 6.2.

6.1 Evaluation at the component level

As we discussed in the architecture of SUMTIME-TURBINE, the first three components constitute content determination. We used three stages to evaluate the content determination:

- select a test suite of scenarios;
- run the pattern recognition algorithm in the first component and the interesting pattern selection algorithm in the third component on this suite;
- evaluate the output by comparing with expert human output for the same task.

We carried out an evaluation based on accuracy (recall and precision) by comparing the algorithms with human performance. Satisfactory evaluation results were generally obtained for the content determination. For example, we randomly selected six scenarios (each scenario had six channels covering three hours) and ran the pattern recognition algorithm on them. We obtained the evaluation results in Table 2.

Table 2 Measures of accuracy
for recognising spikes, steps and oscillations in the six scenarios

Measures of accuracy / Patterns	TP	FP	FN	Recall	Precision
Spikes	192	6	12	94%	97%
Steps	5	0	0	100%	100%
Oscillation	24	0	6	80%	100%

To evaluate summary structuring used in SUMTIME-TURBINE, we used the following procedures:

- select a test suite of scenarios;
- run SUMTIME-TURBINE on this suite to generate summaries;
- produce summaries by arranging three parts (Part One, Part Two and Part Three) randomly in summaries;
- email the above summaries to domain experts and ask them to identify directly the summary with the best structure among the summaries.

The above six scenarios were used to evaluate summary structuring and we found that domain experts always identified the summaries with the SCHEMA adopted in SUMTIME-TURBINE as the best one.

In SUMTIME-TURBINE, aggregation is mainly used to produce the sentences in Part Two and Part Three. We adopted the following stages to evaluate aggregation algorithms.

- select a test suite of scenarios;
- run SUMTIME-TURBINE on this suite to generate summaries;
- select sentences produced through aggregation algorithms from the summaries;
- ask domain experts to score these sentences considering the quality of aggregation of shared participants and structures in these sentences.

The scores were divided into five grades: 5 (very good), 4 (good), 3 (mediocre), 2 (poor) and 1 (very poor). The above six scenarios were also used to evaluate aggregation; we found that one domain expert gave an average of about 4.3 and another expert gave an average of about 4.5 for all the sentences evaluated. From this, we can see that both of them are satisfied with the aggregation results.

6.2 Evaluation at the system level

We evaluated SUMTIME-TURBINE as a whole by assessing the quality (including readability and technical precision) of the summaries generated. We used three stages as follows:

- select three sets of scenarios.
- run SUMTIME-TURBINE on these sets to generate summaries.
- present domain experts the generated summaries along with the corresponding graphs from the same data sets and ask them directly to score the summaries according to their quality based on five grades: 5 (very good), 4 (good), 3 (mediocre), 2 (poor) and 1 (very poor).

We randomly selected twelve scenarios (each scenario had six channels covering three hours) to produce summaries, and than sent them to the two domain experts. The average score for all the summaries evaluated by one expert is about 4.1 and by another expert is about 4.5, which means that both of the domain experts agreed with the summaries generated by the system.

In the process of evaluation, we got a very clear message from both experts: they think SumTime-Turbine is doing a reasonable job, but they also think there is room for improvement. There are three main suggestions made by them:

- we need to change the thresholds - some tuning of these is always required for the pattern recognition algorithm in the first component.
- some 'meta-knowledge' needs to be applied to govern how descriptions of the most significant patterns are produced in the final summaries in the fourth component.
- some meta-level descriptions (i.e. 4 tags are the same, 1 is different but all are very similar) could be used in summaries in order to provide more precise descriptions in the fourth component.

7 Conclusions

This paper proposes an architecture for generating textual summaries of complex time series data in four steps. Firstly, patterns from time series data are located and classified by using the pattern recognition algorithm. Secondly, high-level abstractions of the patterns in different contexts are obtained by adapting KBTA. Thirdly, the most significant patterns are chosen through applying subjective or objective measures of interestingness. Finally, a summary, including *background information*, *overview information* and *most significant patterns* in the data, is produced using natural language techniques including summary structuring, lexicalisation and aggregation. The prototype system was evaluated at both the system level and the component level. Now Sermatech Intelligent Applications is discussing a commercial system based on some lessons and research results learned from the prototype system.

There are many applications for description and/or interpretation of time series data in other domains, including medicine and finance. We plan to investigate more sophisticated data interpretation in the neonate domain. We believe that the architecture and techniques used in SUMTIME-TURBINE could be applied to other domains (Sripada et al. 2003b). Other application domains could include long-term summaries of weather or stock data, where a single summary spans weeks or months, not just a day.

Acknowledgements

We are grateful to our industrial collaborators at Sermatech Intelligent Applications, especially Dr. Rob Milne and Mr. Jon Aylett, for their contributions to knowledge acquisition and assessment of the prototype system. This project is supported by the UK EPSRC under grant

GR/M76881. Jin Yu is also funded by China Scholarship Council (CSC) and the Department of Computing Science, University of Aberdeen.

References

Agrawal, R. and Srikant, R. (1994) Fast algorithms for mining association Rules. In: *Proc. VLDB'94*, pp. 487-499.

Agrawal, R., Psaila, G., Wimmers, L. and Zait, M. (1995) Querying shapes of histories. In: *Proc. of the 21st International Conference on Very Large Databases*, pp. 502-514.

Dale, R. (2003) *StockReport*. URL: http://www.ics.mq.edu.au/~ltgdemo/StockReporter/ on web November 2003.

Downs, M., Walker, G. and Blum, L. (1986) Automated summarisation of on-line medical records. In: *Proc. of MEDINFO'86*, pp. 800-804.

Goldberg, E., Driedger, N. and Kittredge, R. (1994) Using natural-language processing to produce weather forecasts. *IEEE Expert* **9**(2): 45-53.

Haimovitz, J. and Kohane, S. (1993) Automated trend detection with alternate temporal hypotheses. In: *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, pp. 146-151.

Hey, T. and Trefethen, A. (2003) The data deluge: an e-science perspective. In Berman, F. et al. *Grid Computing*. Wiley.

Hunter, J. (2003) *User manual of Time Series Workbench*. University of Aberdeen.

Iordanskaja, L., Kim, M., Kittredge, R., Lavoie, B. and Polguere, A. (1992) Generation of extended bilingual statistical reports. *Proc. of Coling* **3**: 1019-1023.

Kahn, G. (1991) Combining physiologic models and symbolic methods to interpret time-varying patient data. *Methods of Information in Medicine* **30**(3): 167-178.

Kahn, G., Fagan, M. and Tu, S. (1991) Extensions of the time-oriented database model to support temporal reasoning in medical expert system. *Methods of Information Processing in Medicine* **30**: 4-14.

Klabbers, E., Krahmer, E. and Theune, M. (1998) A generic algorithm for generating spoken monologues. In: *Proc. of ICSLP'98*, pp. 2759-2762. Sydney, Australia.

Kilgarriff, A. and Grefenstette, G. (2003) Introduction to the special issue on the web as a corpus. *Computational Linguistics* **29**: 333-347.

Kukich, K. (1983) Knowledge-based report generation: a knowledge engineering approach to natural language report generation. *Ph.D. thesis*, Information Science Department, University of Pittsburgh.

Kuilboer, M., Shahar, Y., Wilson, M. and Musen, A. (1993) Knowledge reuse: temporal-abstraction mechanisms for the assessment of children's growth. In: *Proc. of the Seventeenth Annual Symposium on Computer Applications in Medicine*, pp. 449-453.

Law, A., Freer, Y., Hunter, J., Logie, R., McIntosh, N. and Quinn, J. (2005) Generating textual summaries of graphical time series data to support medical decision making in the Neonatal Intensive Care Unit. Submitted to *Journal of Clinical Monitoring and Computing.*

Mani, I., Klein, G., House, D., Hirschman, L., Firmin, T. and Sundheim, B. (2002) SUMMAC: a text summarization evaluation. *Natural Language Engineering* **8**(1): 43-68.

Maybury, M. (1995) Generating summaries from event data. *Information Processing and Management* **31**: 735-751.

Mckeown, K., Kukich, K. and Shaw, J. (1994) Practical issues in automatic documentation generation. In: *Proc. ANLP'94*, pp. 7-14.

McKeown, K. (1985) Discourse strategies for generating natural-language text. *Artificial Intelligence* **27**: 1-42.

Mellish, C. and Dale, R. (1998) Evaluation in the context of natural language generation. *Computer Speech and language* **12**: 349-373.

Milne, R., Nicol, C. and Massuyes, L. T. (2001) TIGER with model based diagnosis: initial deployment. *Knowledge Based Systems* **14**: 213-222.

Milne, R., Nicol, C., Massuyes, L. T. and Quevedo, J. (1995) TIGER: knowledge based gas turbine condition monitoring. In: *Proc. of the Expert System'95 Conference*.

Perng, C., Wang, H., Zhang, R. and Parker, S. (2000) Landmarks: a new model for similarity-based pattern querying in time series databases. In: *Proc. of ICDE'2000*, San Diego, USA.

Rebrik, P., Wright, D., Emondi, A. and Miller, D. (1999) Cross channel correlations in tetrode recordings: implications for spike-sorting. In: *Proc. of Computation and Neural Systems Meeting*, Montana.

Reiter, E. and Dale, R. (2000) *Building Natural Language Generation Systems*, Cambridge University Press.

Reiter, E., Sripada, S. and Robertson, R. (2003) Acquiring correct knowledge for natural language generation. *Journal of Artificial Intelligence Research* **18**: 491-516.

Soto, E., Manjarrez, E. and Vega, A. (1997) Microcomputer program for automated neuronal spike detection and analysis. *International Journal of Medical Informatics* **44**: 203-212.

Spence, R. (2001) *Information Visualization*. ACM Press.

Sripada, S., Reiter, E., Hunter, J. and Yu, J. (2003b) Summarising neonatal time series data. In: *Proc. of 2003 Conference of the European Chapter of the Association for Computation Linguistics.* Companion Volume, pp. 167-170.

Sripada, S., Reiter, E. and Davy, I. (2003a) SumTime-Mousam: configurable marine weather forecast generator. *Expert Update* **6**(3): 4-10.

Shahar, Y. (1997) A framework for knowledge-based temporal interpolation. *Artificial Intelligence* **90**: 79-133.

Shahar, Y. (1994) A knowledge-based method for temporal abstraction of clinical data. *Ph.D. dissertation*, Stanford University.

Shahar, Y. and Musen, M.A. (1996) Knowledge-based temporal abstraction in clinical domains. *Artificial intelligence in Medicine* **8**(3): 267-298.

Shaw, J. (1995) Conciseness through aggregation in text generation. In: *Proceedings of the 33rd ACL* (Student Session), pp. 329-331.

Shneiderman, B. (1996) The eyes have it: a task by data type taxonomy for information visualizations. In: *Proc. of Visual Language 96.*

Silberschatz, A. and Tuzhilin, A. (1996) What makes patterns interesting in knowledge discovery systems. *IEEE Transactions of Knowledge and Data Engineering* **8**(6).

Yu, J. (2004) SumTime-Turbine: a knowledge-based system to generate English textual summaries of gas turbine time series data. *Ph.D. thesis*, University of Aberdeen.

Yu, J., Hunter, J., Reiter, E. and Sripada, G. (2002) Recognising visual patterns to communicate gas turbine time-series data. In: *Proc. of ES2002*, pp. 105-120.