HAL
open science

A Personal Storytelling about Your Favorite Data

Cyril Labbé, Claudia Roncancio, Damien Bras

To cite this version:

Cyril Labbé, Claudia Roncancio, Damien Bras. A Personal Storytelling about Your Favorite Data. 15th European Workshop on Natural Language Generation (ENLG 2015), Sep 2015, Brighton, United Kingdom. 10.18653/v1/W15-4727 . hal-01288369

HAL Id: hal-01288369

https://hal.science/hal-01288369v1

Submitted on 16 Mar 2016

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

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

A personal storytelling about your favorite data

Cyril Labbé, Claudia Roncancio, Damien Bras
Univ. Grenoble Alpes, LIG, F-38000 Grenoble, France
first.last@imag.fr

Abstract

A controlled use of omnipresent data can leverage a potential of services never reached before. In this paper, we propose a user driven approach to take advantage of massive data streams. Our solution, named *Stream2Text*, relies on a personalized and continuous refinement of data to generate texts (in natural language) that provide a tailored synthesis of relevant data. It enables monitoring by a wide range of users as text streams can be shared on social networks or used individually on mobile devices.

1 Introduction

Considering a user-centered point of view, taking advantage of Big Data may be quite difficult. Managing data volume, variety and velocity to extract the adequate information is still challenging. The information extraction needs customization to adapt both, content and form, so to fit user's current profile. For content, data volume can be reduced by using user preferences, regarding the form, answers should be adapted to be displayed on the available user devices.

This paper focuses on improving stream data monitoring by proposing the construction of ad-hoc abstracts of data. This paper presents the generation of short texts which summarize (in natural language) the result of continuous complex data monitoring. Text summaries can be shared in social networks or can be delivered to personal devices in various context (e.g. listen to summaries while driving). Such a solution facilitates monitoring, even for disabled users.

Let's consider a running example on stock options monitoring involving data on volatility of the stock options, transactions and information about the country emitting the actions. This information represents a large volume of streamed data which could not be handled by an individual user. Rather than getting data about all the transaction of the day, users would prefer to focus on those which are the most *relevant* to him. This paper proposes a way to produce personalized summaries of the monitored data which fits better the user's current preferences. We adopt contextual preferences to integrate user's priority. For example:

In the IT category, I'm more interested in stock options that had low volatility during the last 3 days

Our system named, **Stream2text** will produce a stream of personalized summaries to provide important information to the user. Knowledge on the concepts of the application domain and a continuous query evaluation allows to serve queries such as the following.

Every two hours, I would like a summary of the last 50 transactions on my preferred stock options.

To the best of our knowledge, **Stream2text** is the first effort proposing a comprehensive solution that produces a stream of summaries based on a continuous analysis of streamed and persistent data[1]. . Section 2 details our running example. Section 3 provides a global picture of *S*tream2text. Section 4 and 5 respectively presents the theoretical basis for personalized continuous querying and the *text generator* operator. Section 6 exposes implementation and experimental results, sections 7 and 8 present related work and our conclusions.

2 Motivation and running example

This work adopts an unified model to query persistent relational data and streams. A precise definition of streams and relational data is given in section 4. In our running example, Luc, a cautious investor, likes to follow stock exchange information. He has access to real-time quotations and volatility rates as well as real-time transactions. These data involve the following streams and persistent relations (see the conceptual schema in

[1] A french version of this paper has been presented in the french conference on Information Systems INFORSID'14.

Proceedings of the 15th European Workshop on Natural Language Generation (ENLG), pages 166–174,
Brighton, September 2015. ©2015 Association for Computational Linguistics

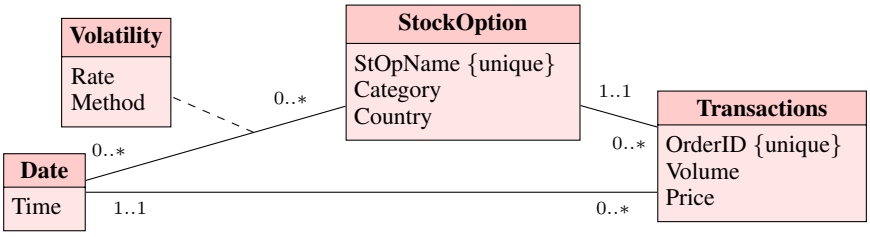


Figure 1: Data model for stock exchange information

figure 1):

– Relation *StockOption(StOpName, Category, Country)*: stores the stock option name, its category (e.g. Commodities (*Co*), Technologie (*IT*)), and the country where the company headquarters are located.

– Stream *Transactions(OrderID, TTime, StOpName, Volume, Price)*: a data stream providing real-time information about stock options transactions. It includes the transaction time ($TTime$), the quantity of shares ($Volume$) and the ($Price$) of the stock option share.

– Stream *Volatility(StOpName, ETime, Rate, Method)*: a data stream providing real-time information about the estimated volatility ($rate$) of stock options. It includes the time of the estimation $ETime$ and the estimation $Method$.

Luc wants to access such data any where, any time from any device (mobile, with or without screen). He has some preferences he wants to be taken into account so to get only the most appropriate data in order to facilitate and speed up his decisions. His preferences are described by the following statements:

[P1] Concerning stocks of category 'Co', Luc prefers those with a volatility-rate less than 0.25. On the other hand, concerning IT stocks, Luc prefers those with a volatility-rate greater than 0.35.

[P2] For stock options with volatility greater than 0.35 at present (calculated according to some method), Luc prefers those from Brazil than Venezuela's one.

[P3] For stock options with volatility-rate greater than 0.35 at present, Luc is interested in transactions carried out during the last 3 days concerning these stock options, preferring those transactions with quantity exceeding 1000 shares than those with a lower amount of shares.

Luc's preferences will be expressed in the system by means of *rules* of form `IF` *some context is verified* `THEN` *Luc prefers something to something else*. For [P1] the context is $StockOption.Category = 'Co'$ and for Luc $Volatility.Rate \leq 0.25$ *is better than* $Volatility.Rate > 0.25$.

Preference rules may involve streams or relational data on both context side and preference side of the rule. Luc's summary requirements may also involve queries on relational data and streams and can be "one-shot" or continuous.

[Q1] Every day, a summary concerning the stock $Total$ over the last two days.

[Q2] Every hour, a summary, over the last hour, for the category IT inside the 100 transactions that fits the best my preferences.

[Q3] Every hour, a summary of the last hour, of the 100 preferred transactions in the category 'IT'.

[Q4] A summary of the last 1000 transactions for French stock options having a $rate > 0.8$ and with at least one transaction with $volume > 100$.

Data extraction can be precisely customized according to the current user preferences. For example [Q2] identifies Luc's 100 most preferred transactions and then extracts those being from the 'IT' category. Whereas [Q3] selects transactions of category 'IT' and among them extracts Luc's 100 most preferred. The rate of summary production is given either with a temporal pattern (e.g. Q1, Q2, Q3) either with a positional pattern (e.g. Q4: every 1000 transactions).

3 Overview of *Stream2text*

This section presents the global picture of *Stream2text*, illustrated in Figure 2.

Users provide queries and preferences on streams and persistent data. *Stream2text* evaluates the continuous queries by integrating user's preferences and generates a text stream summarizing the most pertinent data. User's query for the summary creation include a "point of view", the scope and the frequency of the summary production. The scope allows to limit the volume of data to consider in the production of one summary.

The support of users queries on streams and persistent data rely on the use of a formal model. Such model provides a non-ambiguous represen-

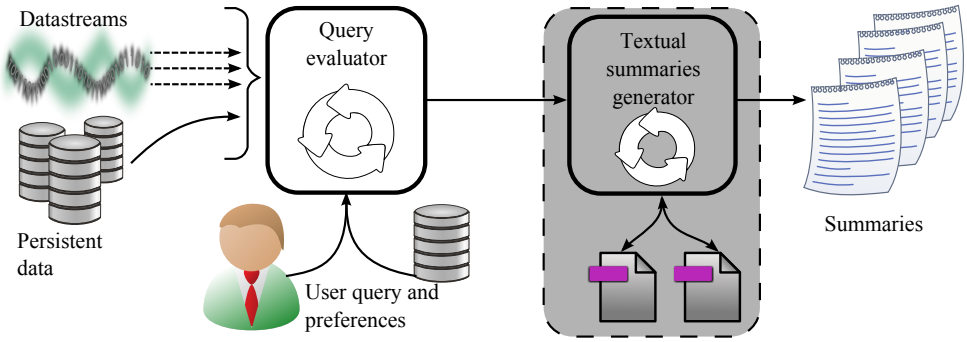


Figure 2: Architecture of *Stream2text*

tation of the data to be summarized. User's preferences allows to limit the volume of data to produce very customized summaries that fitts the current user's interest. Data summarization involves first, a data aggregation step to produce a structured summary and second, a natural language summary generation which leads to the text sent to the user.

Architecture of the text generator. This component (see Figure 3) relies on information about the conceptual schema of the data and the aggregation functions used to create the structured summary.

The main information about the schema concern the textual description of the properties of the entities. For example, *StOpName*=v can be expressed in a text by "*The stock option v*". A stock option, represented as a tuple $t \in StockOption$, can be described in a text as "*The stock option* $t.StOpName$ *is of category* $t.Category$ *and it's home country is* $t.Country$". Such phrases are usually available during the design phases of applications.

As text generation phase relies on the structured summary, it requires textual descriptions of the aggregation functions being used. For example, if the summary of the values of a property A includes $Avg(A)$, then the associated text could be "*The average value of A is* $Avg(A)$" or "*the average A is* $Avg(A)$".

4 Theoretical foundation for query evaluation

This section introduces the theoretical foundations for query evaluation with contextual preferences (query and user preferences in Figure 2).

4.1 Stream algebra

Let us first consider the queries introduced in Section 2 in a version without preferences and summarization aspects:

[Q1'] Every day, information concerning the share $Total$ over the last two days.

[Q2']/[Q3'] Every hour, informations related to the category 'IT'.

[Q4'] Informations for the last 1000 transactions concerning french stocks option having $rate > 0.8$ and with at least one transaction with $volume > 100$.

These queries are written using (Petit et al., 2012b). This algebra formalizes expressions of continuous and instantaneous queries combining streams and relational data. In the following, we present the basics of query expression.

Streams and relations (hereafter resp. denoted by S and R) are different concepts (Arasu et al., 2004). A *stream* S is an infinite set of tuples with a common schema and two special attributes: a timestamp and the position in the stream[2]. A *temporal relation* R is a function that maps a time identifier t to a set of tuples $R(t)$ having a common schema. Classical relational operators (selection σ, projection π, join $\bowtie$) are extended to temporal relations and π and σ to streams. For example, $\sigma_{Volume>10}(Transactions)$ is the stream of transactions having $Volume > 10$.

A temporal relation is extracted from a stream using windows operators. The algebra provides an extended model for windows operators including positional, temporal and cross domain windows (e.g. slide n tuples every δ seconds). The following expressions represent some very useful windows:

– $S[L]$ contains the last tuple of a stream (L standing for $Last$);

– $S[N$ slide $\Delta]$ is a sliding window of size N sliding Δ every Δ. N and Δ are either a time duration or a number of tuples.

A stream is generated from a temporal relation using a *streamer* operator. Among them, $I_S(R)$ produces the stream of tuples inserted in R. Given a window description, a streamer and a join con-

[2]These definitions can be extended using the notion of batch (Petit et al., 2012b).

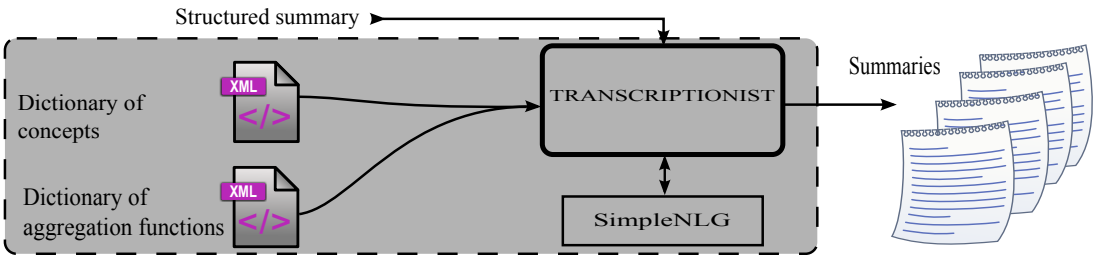


Figure 3: Architecture of the text generator

dition c, a join operator between two streams or between a stream and a relation can be defined. In the following we will use:

$$S \bowtie_c R = I_S(S[L] \bowtie_c R).$$

The stream $S \bowtie_c R$ may have new tuples originated by tuple arrivals in S or updates in R. The semi-sensitive-join operator ($\bowtie\!\!\blacktriangleleft$) produces a stream resulting from a join between the last tuple of the stream and the relation by the time the tuple arrives:

$$S \bowtie\!\!\blacktriangleleft_c R = I_S(S[L] \bowtie_c R(\tau_S(S[L])))$$

where τ_S denotes the function that gives the timestamp of a tuple in the stream S (see (Petit et al., 2012b) for more details). In the following, the stream-join that joins tuples from two different streams will be defined as follows:

$$S_1 \underset{\leq}{\bowtie} S_2 = I_S(S_1[\infty] \underset{\leq}{\bowtie} S_2[L])$$

where $\underset{\leq}{\bowtie}$ joins the tuple of $S_2[L]$ with the single tuple of S_2 having the maximal timestamps lower or equal to the timestamp of $S_2[L]$ (i.e. $\tau_S(S_2[L])$). Previously defined queries can be written as:

[Q1']$((Volatility \underset{\leq}{\bowtie} Transaction) \bowtie\!\!\blacktriangleleft \sigma_{StOpName='Total'} StockOption)$ $[2day \text{ slide } 1day]$ (1)

[Q3']$((Volatility \underset{\leq}{\bowtie} Transaction) \bowtie\!\!\blacktriangleleft \sigma_{Category='IT'} StockOption)[1h \text{ slide } 1h]$ (2)

[Q4']$((\sigma_{rate>0.8} Volatility \underset{\leq}{\bowtie} \sigma_{Volume>100} Transaction) \bowtie\!\!\blacktriangleleft \sigma_{Country='FR'} StockOption)[1000n \text{ slide } 1n]$ (3)

4.2 The Preference Model

In this section we present the main concepts concerning the logical formalism for *specifying and reasoning* with preferences (see (de Amo and Pereira, 2010; Petit et al., 2012a) for details). Intuitively speaking, a *rule of contextual preference* (a cp-rule) allows to compare two tuples of a relation R both of them fitting a particular context.

$$\varphi : u \rightarrow C_1(X) \succ C_2(X)[W]$$

Where $X \subseteq Attr(R)$, $W \subseteq Attr(R)$ et $X \notin W$; $C_i(X)$ (for $i = 1, 2$) is an evaluable condition over tuples of R. u is also a condition in which neither X nor W are involved (cf. exemple 1). Two tuples are comparable using a cp-rule, if they have the same value for the so-called *ceteris paribus* (noted with $[\,]$).

A *contextual preference theory* (*cp-theory* for short) over R is a finite set of cp-rules. Under certain consistency constraints a cp-theory induce a *strict partial order* over tuples allowing to rank them according to the user preferences.

Example 1 *Let us consider the two preference statements P1 and P2 of our motivating example. They can be expressed by the following cp-theory over the schema* $T(StOpName, Cat, Country, ETime, Rate, Method)$*:*

– φ_1: $Cat = co \rightarrow (Rate < 0.25 \succ Rate \geq 0.25), [Method]$

– φ_2: $Cat = it \rightarrow (Rate \geq 0.35 \succ Rate < 0.35), [Method]$

– φ_3: $Rate > 0.35 \rightarrow (Country = Brazil \succ Country = Venezuela)$

The user preferences of Figure 2 are cp-theories. The algebra integrates them in a streaming context. The semantics is the one called *with constraints*.

4.3 Preference Operator

Preference operators are algebraic and can be used on instantaneous and continuous queries on streams and relations. User preferences, represented as a cp-theory, can be seen as part of a user profil. Preferences are used only if personalization of queries is asked by the use of a "top-k" query in which the operator ***KBest*** is used. ***KBest*** selects the subset of k preferred data according to the hierarchy specified by the cp-theory. For example, **Q2** of Section 2 is expressed as

$$(\sigma_{Category='IT'}(\mathbf{KBest_{100}}((Volatility \underset{\leq}{\bowtie} Transaction) \bowtie\!\!\blacktriangleleft StockOption)))[1h \text{ slide } 1h]$$

Whereas **Q3** can be written as:
$KBest_{100}(Q2')$.

4.4 Aggregation functions and structured data summary

To generate textual summaries, *Stream2text* first builds a structured summary of data by using aggregation functions provided by the query evaluator. The choice of functions may depend of the application domain. Intuitively, an aggregation function f associates to a set of tuples a unique tuple for which attributes and values are determined by f. We will use definition 1, which includes a resulting attribute named after the function used to compute the aggregated value.

Definition 1 (Aggregation Operator) *Let R be a temporal relation with schema $A = \{a_i\}_{i=1..n}$, Let $f^j(\{A_i\}_{i\in\{1,n\}})_{j=1..m}$, be m aggregation functions. The aggregation operator $\mathcal{G}_{f^1,f^2,...f^m}$ aggregates the set of tuples R in one tuple using the functions f^j.*

$$\mathcal{G}_{f^1,f^2,...f^m}(R) = \{\cup_{j=1}^{m}(f^j, f^j(\{A_i\}_{i\in\{1,n\}}))\}$$

5 Text generation operator

We define several functions and operators to associate text to data. Sections 5.1 and 5.2 show how the schema knowledge and structured summary can be related to text. Section 5.3 defines the transcription operator.

5.1 Dictionary of concepts

We will consider a database dealing with n_e entities/classes $\{E_i\}_{i=1..n_e}$. Each of which has a set n_i de property/attributes $\{A_{i,j}\}_{j=1..n_i}^{i=1..n_e}$, some of them being identifiers or key attributs.

Schema knowledge is mandatory to be able to express facts about data. A text fragment is associated to each property of the data model. It may be used to *name* the referred concept in a text. These texts are managed in a *Dictionary of concepts*.

Definition 2 (Dictionary of Concepts) *It is a function $\mathcal{D}_c$ which associates to each database concept (ie. property $A_{i,j}$) a noun phrase NP. This noun phrase can be use to name the concept (property $A_{i,j}$) in a text.*

$$\mathcal{D}_c(A_{i,j}) = \{NP\}_{i,j}$$

where $\{NP\}_{i,j}$ is a noun phrase naming concept $A_{i,j}$ in natural language.

The example 2 illustrates some possible values for $\mathcal{D}_c$ (in french).

Example 2 (Dictionary of concepts (french entries))

$$\mathcal{D}_c(StOpName) = \left\{ \begin{array}{cc} le & action \\ det. & noun \end{array} \right\}$$

$$\mathcal{D}_c(Price) = \left\{ \begin{array}{cc} le & prix \\ det. & noun \end{array} \right\}$$

$$\mathcal{D}_c(Price) = \left\{ \begin{array}{cc} le & cours \\ det. & noun \end{array} \right\}$$

$\mathcal{D}_c$ may associate several values to a given concept. This can be used to improve diversity in the generated texts.

5.2 Dictionary of aggregation functions

The textual summary is based on a structured summarization computed using aggregation functions. A dictionary with sentence structures is used as the basis to reflect the meaning of the aggregation functions. A sentence structure is composed of sub fragments that can be used by a *realization engine* (cf. (Gatt and Reiter, 2009) and example 3) i.e. the generation of a correct sentence regarding the grammatical rules of the targeted natural language.

Example 3 (Sentence structure and realization) *A sentence structure (in french), represented as a graph, followed by its realization is presented in figure 5.*

The definition 3 formalizes the function used to associate a text to the result of an aggregation function F over a set of k attributes $\{a_i\}_{i=1..k}$. The text is function of the texts $\mathcal{D}_c(a_i)_{i=1..k}$ and the result of the aggregation function $F(\{a_i\}_{i=1..k})$.

Definition 3 (Dictionary of aggregation functions) *It is a function $\mathcal{D}_f$ that, given an aggregation function $F(\{a_i\}_{i=1..k})$, returns a sentence structure SP. The realization of SP describes the aggregation function in natural language.*

$$\mathcal{D}_f(F) = \{\{\mathcal{D}_c(a_i)\}_{i=1..k}, VP, \{Co_i\}_{i=1..x}, F(\{a_i\}_{i=1..k}), \{R_j\}_{j=1..y}\} \quad (4)$$

where VP is a verb phrase explaining the relation between attributes $\{a_i\}_{i=1..k}$ and the value $F(\{a_i\}_{i=1..k})$. $\{Co_i\}_{i=1..x}$ is a set of x complements (noun, direct object, indirect object) and $\{R_j\}_{j=1..y}$ is a set of y relations between sentence elements.

Example 4 *Hereafter a simplified example of sentence structure in french for the $MostFreq(A)$ function which calculates* the most frequent value. *Other sentence structures are possible.*

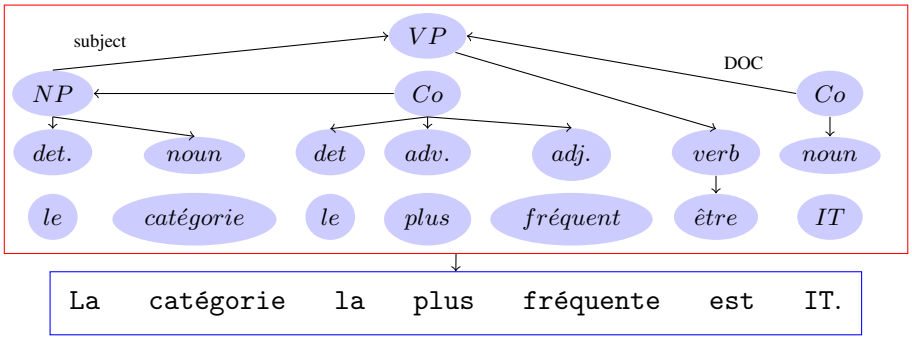
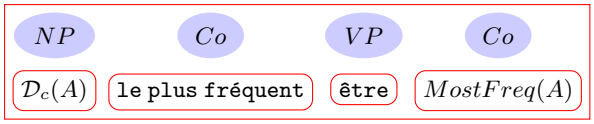


Figure 4: sentence structure



A possible realization of this sentence structure is presented in Example 3 with the attribute $Category$. The same aggregation function used with an other attribute would give a different realization. For example, given the attribute $Country$ the realization in french could be (given that $MostFreq(Country) = France$):

`Le pays le plus fréquent est France.`

The text generation for an aggregation function requires the realization of the entry of the dictionary of functions and the computation of the function itself.

5.3 Transcription operator

A temporal relation (i.e. a function of time) is said to be *transcriptable* if it is possible to generate a set of sentence structures concerning this relation. Thus, transcribing in natural language the signification of a set of data is equivalent to produce a set of sentence structures describing this data set. The transcription is a step that comes after the computation of a structured summary, which has the form of a unique tuple. Definition 4 gives the form of relation that will be considered for transcription.

Definition 4 (A transcriptable relation) *is a temporal relation R that contains a unique tuple such that: $\forall (A, v) \in t$:*
– either A is a concept for which an entry exists in the dictionary of concepts (ie. $A \in Dom(\mathcal{D}_c)$)
– either $A = F$ where $F(\{a_i\}_{i=1..n})$ is an aggregation function used to aggregate a set of values $\{a_i\}_{i=1..n}$ in a value v and such that $F \in Dom(\mathcal{D}_f)$ and $(F, v) \in \mathcal{G}_F(R)$).

A transcription operator is defined to generate a set of sentence structures given a transcriptable relation.

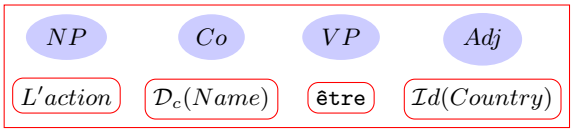
Definition 5 *The transcription operator $\mathcal{T}$ provides a set of sentence structures given a temporal relation R having a schema F_i:*

$$\mathcal{T}(R) = \bigcup_i \mathcal{D}_f(F_i)$$

R being a temporal relation, $\mathcal{T}(R)$ is a set of sentence structures evolving with time. The use of a streamer to create a stream on this set allows the insertion in a stream of texts timestamped with the time when R is modified.

Generally speaking, a function identity $\mathcal{I}d$ is used as name of attributes of a transcriptable relation gives the "point of view" chosen to summarize the data. The dictionary of functions has to include an entry for such function $\mathcal{I}d$. This could be a starting sentence for the textual summary.

Remark 1 (Special case for key attributes) *When the function $\mathcal{I}d$, in the list of attributes of a transcriptable relation, is applied to a key, then the transcription must describe a particular object and not the "point of view" adopted to summarize the data. In that case, it is not a summary but an object of the data base. It is thus necessary to specify an entry in the dictionary of functions for such cases: i.e. $\mathcal{I}d$ on key attributes. As an example, $\mathcal{D}_f(\mathcal{G}_{\mathcal{I}d(Name),\mathcal{I}d(Country)})$ could be defined in french as follows:*



So the realization of

$$\mathcal{T}(\mathcal{G}_{\mathcal{I}d(Name),\mathcal{I}d(Country)}(\pi_{Name,Country}(\sigma_{Name='Total'}(StOpName))))$$

is `L' action Total est française.`

The transcription operator builds texts for ASTRAL queries (with or without preferences). The

$$\mathcal{T}(\mathcal{G}_{\mathcal{I}d(Name),\mathcal{I}d(Country),\mathcal{I}d(Category),Avg(Volume),MostFreq(Methode)}(\pi_{Name,Country,Category,Volume,Methode}(Q1'))) \quad (5)$$

$$\mathcal{T}(\mathcal{G}_{MostFreq(Name),MostFreq(Country),\mathcal{I}d(Category),Avg(Volume),MostFreq(Methode)}(\pi_{Name,Country,Category,Volume,Methode}(Q3'))) \quad (6)$$

$$\mathcal{T}(\mathcal{G}_{MostFreq(Name),\mathcal{I}d(Country),MostFreq(Category),Avg(Volume),MostFreq(Methode)}(\pi_{Name,Country,Category,Volume,Methode}Q4')) \quad (7)$$

generated text depends only on data content and on the functions used for the structure summary. The query itself is not directly transcripted (see example 5).

Example 5 (Transcription) *Assuming that numerical (resp. non-numerical) attributes are aggregated using the average function Avg (resp. most frequent $MostFreq$), queries Q1, Q3 and Q4 are written as presented in equations 5,6,7.*

6 Implementation and experimentation of Stream2text

This section describes our prototype and the experimentations realized (in french) as a proof-of-concept. We focus here on the transcription as the query evaluation part is based on existing software (PostegreSQL and Asteroide (Petit, 2012)).

6.1 Data to text transcription

The transcriptionist component of *Stream2text* has been developed in Java. It produces textual summaries of the data provided by the query manager. The conceptual entities are used to establish the structure of the text. The transcriptionist prepares the grammatical structure of the sentences and uses the french version of SimpleNLG (Gatt and Reiter, 2009) for the text realization. The transcriptionist assembles the sentences issued by SimpleNlg and produces the paragraphs. The summaries include an introduction to give information on treated data and one paragraph per entity involved in the summary. Each paragraph includes several sentences reflecting the meaning of the aggregation functions used for the structured data summary.

The current **dictionary of functions** includes: – $MostFreq$ to calculate the most frequent value in a data set. – Avg, Med and $Count$ with usual mathematical semantics. – $Part(v, A)$ to indicate the % of a value v in the values of A. – $\mathcal{I}d(Key_Attribute)$ to handle key attributes cases which require particular text generation (see remark 1). This is done for each entity of the database schema. Except for $\mathcal{I}d$, the dictionary contains generic sentence structures. There is no need of redefinition when the functions are used for different attributes and the dictionary is independent from the data schema and may be shared by many applications and users. Nevertheless, the functions can be personalized to produce customized summaries.

6.2 Experiment with stock exchange data

From `http://www.abcbourse.com/`, an experimental data set (5000 transactions) was created. Data involves twelve stock options belonging to ten categories and three countries. Data have timestamps used in the streams (ie. $TTime$ and $ETime$). Quantity, price of transactions, volatility, the category and the country of the stock options are available.

The dictionary of concepts has entries for all attributes and concepts, such as $StOpName$ and $Category$, for the three entities (cf. section 2). We experimented summary generation for queries as in the running example. To illustrate the result, see hereafter a summary for [Q1].

Example 6 (Summary for Q1) *For its summary, let consider that Luc wishes the average and the median for the volume of transactions, prices, etc. [Q1] is evaluated as specified in equation 5. Figure 5 shows the summary for a 2 day period.*

7 Related work

This work is related to several subjects including continuous query evaluation, structured data summarization and natural language generation.

Many theoretical (Krishnamurthy et al., 2010) and practical (Arasu et al., 2006) works have been done to master continuous querying on data-streams. We use (Petit et al., 2012a) which presents the advantage of a non-ambiguous semantics for querying streams and temporal relations. This is particularly important for joins (Petit et al., 2012b) and windows (Petit et al., 2010).

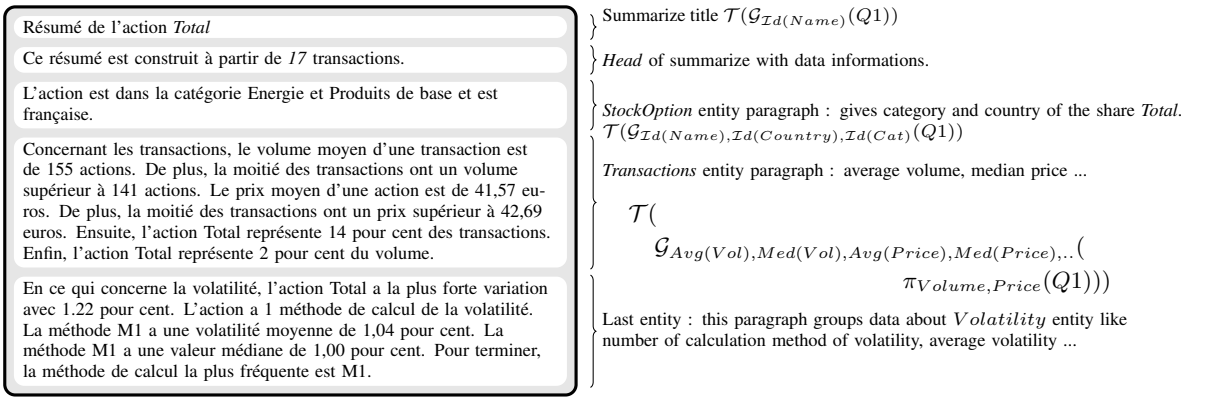


Figure 5: Summary for query Q1 according to the example 6

There are also several efforts on summarizing or synthesizing numerical and structured data. In this context, we can understand preferences models and $top-k$ operators as a way to reduce the volume of data. Our proposition supports $top-k$ queries combined to aggregation functions to produce a structured summary. This phase could use other works of this nature as (Cormode et al., 2012; Cormode and Muthukrishnan, 2005). The choice of another preferences model (Koutrika et al., 2010) is compatible with the natural language transcription phase but impacts the structured summary. The use of the CPrefSQL model is motivated by the qualitative approach it proposes and it's support of "contexts" in dominant tuple calculation. This differs from approaches by score functions (Borzsonyi et al., 2001; Papadias et al., 2005; Kontaki et al., 2010).

Concerning automatic text generation, the $text-to-text$ approach is used to automatically summarize texts (Rotem, 2003) or opinions (Labbé and Portet, 2012) expressed in natural language. Our proposal follows a $data-to-text$ approach which consists in text generation to explain data. To the best of our knowledge, current proposals are specific to an application domain such as medicine (Portet et al., 2009; Gatt et al., 2009) or weather report (Turner et al., 2010). The NLG community focuses on language aspects as sentence aggregation, enumerative sentence construction or referential expressions. These works are independent of the application domain whereas the upstream phase including the determination of the content and document planing (Reiter and Dale, 2000), still require domain experts help. However, (Androutsopoulos et al., 2013) proposed, recently, natural language descriptions of individuals or classes of an OWL ontology. In our context, this is analogous to the description of a single tuple in a relation but does not include information summarization as proposed in this paper. *Stream2text* facilitates concept determination and sentence generation by using the conceptual knowledge on data schema and aggregation functions used for the structured summary. The domain specific knowledge required for text generation can be extracted from the data analysis phase. The knowledge relative to the aggregation functions is mostly domain independent. Our proposal combines data model knowledge and data sets to produce summaries by using the realization engine proposed by (Gatt and Reiter, 2009).

8 Conclusion and future research

Our work joins a global effort in mastering big data. We propose the automatic generation of short texts to summarize streamed and persistent data. Such textual summaries allow new information access possibilities. For example, sharing in social networks, access through mobile devices and the use of text-to-speech software.

We propose *Stream2text* which is a generic solution including the whole process, from continuous data monitoring until the generation of a natural language summary. The personalization of the service and, the reduction of the volume of data, rely on the integration of user preferences on data and some knowledge on the conceptual model and the aggregation functions. *Stream2text* has been experimented for texts in French. A version for texts in English is in progress.

Our future research targets the combination of complex data management and appropriate text generation. For example, the contextualization of complex event detection and the production of texts referring to present and past situations.

References

Ion Androutsopoulos, Gerasimos Lampouras, and Dimitrios Galanis. 2013. Generating natural language descriptions from owl ontologies: the naturalowl system. *Journal of Artificial Intelligence Research*, 48:671–715.

Arvind Arasu, Brian Babcock, Shivnath Babu, John Cieslewicz, Mayur Datar, K. Ito, R. Motwani, U. Srivastava, and J. Widom. 2004. STREAM: The Stanford Data Stream Management System. Technical report, Stanford InfoLab.

Arvind Arasu, Shivnath Babu, and Jennifer Widom. 2006. The CQL continuous query language: semantic foundations and query execution. In *Proc. of 32nd int. conf. on Very Large Data bases (VLDB '06)*, volume 15.

S. Borzsonyi, D. Kossmann, and K. Stocker. 2001. The skyline operator. In *Proceedings of the 17th International Conference on Data Engineering (ICDE 2001)*, pages 412–430.

Graham Cormode and S. Muthukrishnan. 2005. An improved data stream summary: the count-min sketch and its applications. *J. Algorithms*, 55(1):58–75.

Graham Cormode, Minos N. Garofalakis, Peter J. Haas, and Chris Jermaine. 2012. Synopses for massive data: Samples, histograms, wavelets, sketches. *Foundations and Trends in Databases*, 4(1-3):1–294.

Sandra de Amo and Fabiola Pereira. 2010. Evaluation of conditional preference queries. *Journal of Information and Data Management (JIDM). Proceedings of the 25th Brazilian Symposium on Databases, 2010, Belo Horizonte, Brazil.*, 1(3):521–536.

Albert Gatt and Ehud Reiter. 2009. Simplenlg: A realisation engine for practical applications. In *Proceedings of the 12th European Workshop on Natural Language Generation*, ENLG '09, pages 90–93, Stroudsburg, PA, USA. Association for Computational Linguistics.

Albert Gatt, François Portet, Ehud Reiter, James Hunter, Saad Mahamood, Wendy Moncur, and Somayajulu Sripada. 2009. From data to text in the neonatal intensive care unit: Using NLG technology for decision support and information management. *AI Communications*, 22(3):153–186.

M. Kontaki, A.N. Papadopoulos, and Y. Manolopoulos. 2010. Continuous processing of preference queries on data streams. In *Proc. of the 36th Int. Conf. on Current Trends in Theory and Practice of Computer Science (SOFSEM 2010)*, pages 47–60. Springer Berlin Heidelberg.

Georgia Koutrika, Evaggelia Pitoura, and Kostas Stefanidis. 2010. Representation, composition and application of preferences in databases. In *Proceedings of International Conference on Data Engineering (ICDE)*, pages 1214–1215.

Sailesh Krishnamurthy, M.J. Franklin, Jeffrey Davis, Daniel Farina, Pasha Golovko, Alan Li, and Neil Thombre. 2010. Continuous analytics over discontinuous streams. In *SIGMOD '10: Proc. of the 2010 ACM SIGMOD int. conf. on Management of data*, pages 1081–1092. ACM.

Cyril Labbé and François Portet. 2012. Towards an Abstractive Opinion Summarisation of Multiple Reviews in the Tourism Domain. In *The First International Workshop on Sentiment Discovery from Affective Data (SDAD 2012)*, pages 87–94, Bristol, UK, sep.

D. Papadias, Y. Tao, G. Fu, and B. Seeger. 2005. Progressive skyline computation in database systems. *ACM Transactions on Database Systems*, 30:41–82.

Loïc Petit, Cyril Labbé, and Claudia Lucia Roncancio. 2010. An Algebric Window Model for Data Stream Management. In *Proceedings of the 9th Int. ACM Workshop on Data Engineering for Wireless and Mobile Access (MobiDE '10)* , pages 17–24. ACM.

Loïc Petit, Sandra de Amo, Claudia Roncancio, and Cyril Labbé. 2012a. Top-k context-aware queries on streams. In *Proc. of Int. Conf. on Database and Expert Systems Applications (DEXA 12)*, pages 397–411.

Loïc Petit, Cyril Labbé, and Claudia Lucia Roncancio. 2012b. Revisiting Formal Ordering in Data Stream Querying. In *Proc. of the 2012 ACM Symp. on Applied Computing*, New York, NY, USA. ACM.

Loïc Petit. 2012. *Gestion de flux de donnés pour l'observation de systèmes*. Thèse de doctorat, Université de Grenoble, Décembre.

F. Portet, E. Reiter, A Gatt, J. Hunter, S. Sripada, Y. Freer, and C. Sykes. 2009. Automatic generation of textual summaries from neonatal intensive care data. *Artificial Intelligence*, 173(7–8):789–816.

Ehud Reiter and Robert Dale. 2000. *Building Natural Language Generation Systems*. Cambridge University Press, New York, NY, USA.

N Rotem. 2003. Open text summarizer (ots). June, 2012, http://libots.sourceforge.net.

Ross Turner, Somayajulu Sripada, and Ehud Reiter. 2010. Generating approximate geographic descriptions. In Emiel Krahmer and Mariët Theune, editors, *Empirical Methods in Natural Language Generation*, volume 5790 of *Lecture Notes in Computer Science*, pages 121–140. Springer Berlin Heidelberg.