Petits textes pour grandes masses de données

Cyril Labbé, Damien Bras, Claudia Roncancio

Petits textes pour grandes masses de données

Cyril Labbé — Damien Bras — Claudia Roncancio

Univ. Grenoble Alpes, LIG, F-38000 Grenoble, France
first.last@imag.fr

RÉSUMÉ. *Maîtrisée, l'omniprésence des données offre aujourd'hui un potentiel de services sans précédent. Dans une optique centrée personne, nous proposons une solution étendue pour l'exploitation de masses de données en flux. Notre solution, nommée* Stream2Text*, s'appuie sur un raffinement personnalisé et continu des données, et produit des textes (en langue naturelle) qui résument de manière personnalisée les données intéressantes pour l'utilisateur. Les flux textuels produits permettent un monitoring adapté à un large spectre d'utilisateurs et peut être partagé dans un réseau social ou utilisé individuellement à partir de dispositifs mobiles.*

ABSTRACT. *When controlled, omnipresence of data can leverage a potential of services never reached before. We propose an user driven approach to take advantage of massive data streams. Our solution named* Stream2Text *rests on a personalized and continual refinement of data to generates texts (in natural language) that give in a tailored synthesis of relevant data. This textual stream enables monitoring by a wide range of users. It can also be shared on social networks or be used individually on mobile devices.*

MOTS-CLÉS : *flux données, flux de textes, résumé de données, préférences, génération de textes.*

KEYWORDS: *data stream, texts stream, data summarization, preferences, texts generation.*

1. Introduction

Avec l'omniprésence des données, un grand nombre d'informations sont accessibles partout, tout le temps au travers de divers supports disposant ou non d'écrans de tailles variables. Les données sont complexes, nombreuses, changeantes, incertaines, caractéristiques connues comme les 4V du *Big Data, Volume, Variety, Velocity, Veracity*. Face à cette masse de données, nous nous plaçons dans une optique centrée personne : l'information extraite nécessite une adaptation du contenu et de la forme pour être assimilable par l'utilisateur. L'adaptation du contenu, permettant la maîtrise du volume de données, nécessite de prendre en compte les préférences courantes de l'utilisateur et les différents médium disponibles.

Le monitoring continu de flux de données a comme caractéristique l'intégration de données produites à la volée et de données persistantes qui les enrichissent. L'objectif de notre travail est de faciliter l'exploitation des données complexes en facilitant leur résumé. Pour cela, nous montrons la possibilité de générer des flux textuels personnalisés qui résument en langue naturelle des flux de données. Un tel flux textuel permet un monitoring adapté à un large spectre d'utilisateurs et peut être partagé dans un réseau social ou utilisé individuellement à partir de dispositifs mobiles.A titre d'exemple, considérons le domaine financier où des utilisateurs s'intéressent aux performances des marchés. On considérera des données telles que le taux de volatilité des actions, la situation économique du pays émetteur des actions, des ordres d'achat et de vente d'actions. Dans ce contexte, le volume global des données rend difficile (ou inintéressant) la récupération totale des données par l'utilisateur. A une requête du type *Connaître les opérations du jour*, l'utilisateur préférera une requête plus centrée sur ses intérêts :

Avoir un compte rendu journalier des opérations
de la journée les plus « intéressantes » pour moi.

Ainsi, l'objectif de nos recherches est de produire un compte rendu textuel personnalisée qui décrit les informations disponibles grâce aux préférences courantes de l'utilisateur. Dans le cadre de ce travail, nous intégrons des préférences qualitatives contextuelles qui donnent au système des connaissances sur les priorités de l'utilisateur. Les préférences reflètent des informations telles que

Pour les actions issues des pays en situation économique difficile,
je préfère les actions avec une faible volatilité ces trois derniers jours.

Le système personnalisera les comptes rendus textuels (dits "résumés" par la suite) produits à la volée pour refléter les informations prioritaires pour l'utilisateur. Le traitement continu de flux de données et la connaissance des concepts du domaine permettront de produire des flux de textes courts répondant à des demandes telles que :

Je souhaite un résumé toutes les 2 heures
des 50 dernières opérations sur mes actions préférées.

Les résumés personnalisés produits pourront être lus ou écoutés (voix de synthèse), notamment en situation de mobilité (voiture par exemple). Ce type de compte rendu facilite également l'accès aux informations par des personnes en situation de handicap.

Dans la suite de l'article, nous présenterons les éléments essentiels de notre proposition, nommée **Stream2text**. Les aspects liés à l'évaluation de requêtes continues

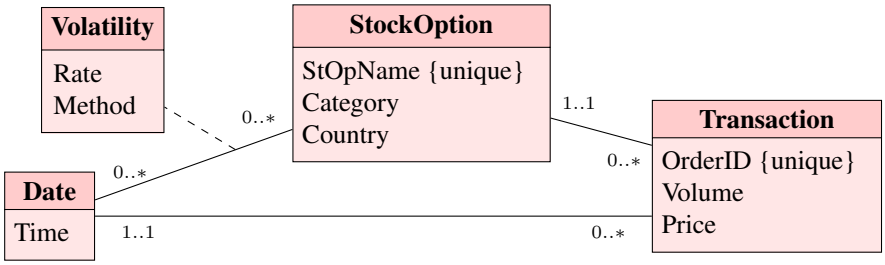


Figure 1. *Modèle des données pour l'exemple du marché boursier*

avec préférences sont introduits et nous développerons de manière plus complète la génération des textes. A notre connaissance, cet effort est le premier à mettre en œuvre le continuum permettant la production automatique de flux de textes personnalisés qui résument des flux de données. Ces résumés textuels offrent la possibilité d'accéder à des informations difficilement exploitables par beaucoup d'utilisateurs.

L'article est organisé ainsi : la section 2 précise l'exemple, la 3 donne une vue globale de **Stream2text**. La section 4 présente des outils et fondements théoriques pour interroger, personnaliser et agréger des données. La section 5 définit les opérateurs de *textualisation* et la 6 expose les choix d'implémentation et les expérimentations réalisées. Les sections 7 et 8 présentent les travaux connexes et notre conclusion.

2. Motivation et cas d'étude

La suite de cet article traite, dans un modèle unifié, des données ayant des caractéristiques de volatilité très différentes : les flux de données et les données persistantes. De manière générale, un flux de données est un ensemble potentiellement infini de n-uplets conformes à un schéma commun possédant un *timestamp*. Les données persistantes sont représentées sous forme de relations, *i.e.* ensemble fini de n-uplets conforme à un schéma commun.

En considérant le cas d'étude présenté en introduction, observons Luc, un investisseur qui suit les évolutions financières pour prendre ses décisions d'achats ou de ventes d'actions. Il dispose d'un accès à de nombreuses sources d'information temps réel sur l'état du marché (cf. schéma conceptuel en figure 1). Ces données sont pour partie des flux et pour partie des relations persistantes :

Relation *StockOption(StOpName, Category, Country)*. Cette relation est un catalogue d'actions incluant le nom des actions, leur catégorie et le pays où l'entreprise a son siège social. Les catégories sont, par exemple, 'Technologie' (IT) , 'Commodities' (Co) ou 'Manufacturing'.
Flux *Transaction(OrderID, TTime, StOpName, Volume, Price)*. Ce flux de données décrit les transactions boursières : l'heure ($TTime$), le nom de l'action ($StOpName$), le nombre de parts vendues ($Volume$) ainsi que le prix unitaire ($Price$).
Flux *Volatility(StOpName, ETime, Rate, Method)* Ce flux de données donne la volatilité ($Rate$) pour les actions (ampleur des variations du cours). Elle est calculée à une certaine date ($ETime$) avec une méthode ($Method$).

Afin de faciliter les prises de décisions, Luc doit pouvoir accéder à ces informations à toute heure et sur différents supports (fixe, mobile, avec ou sans écran, etc). De manière à réduire et mieux cibler le flux d'information selon des critères personnels du moment, il exprime des préférences qui doivent être prises en compte. Informellement, ses préférences combinent des informations qualitatives et quantitatives :

[P1] Pour les actions de la catégorie 'Co', Luc préfère celles qui ont une volatilité inférieure à 0.25. Par contre, pour la catégorie 'IT', Luc préfère les actions dont la volatilité est supérieure à 0.35.

[P2] Pour les actions dont la volatilité est actuellement plus grande que 0.35, Luc préfère les actions brésiliennes à celles du Venezuela.

[P3] Pour les actions dont la volatilité est actuellement plus grande que 0.35, Luc est intéressé par les transactions des 3 derniers jours et préfère celles dont le volume est supérieur à 1000 parts.

Les préférences de Luc peuvent être exprimées au moyen de règles de la forme
SI *un contexte est vérifié* ALORS *Luc préfère quelque chose*.

Ainsi, pour **[P1]** le contexte est : $StockOption.Category =' Co'$

et la préférence est : $Volatility.Rate \leq 0.25$ plutôt que $Volatility.Rate > 0.25$.

Ces règles de préférences peuvent impliquer des flux de données aussi bien que des données persistantes. Les demandes de compte rendu de Luc s'expriment dans cet environnement et peuvent être de nature continue :

[Q1] Chaque jour, un compte rendu des deux derniers jours pour l'action $Total$.

[Q2] Toutes les heures, donner le compte rendu de la dernière heure pour la catégorie 'IT' parmi les 100 transactions qui satisfont le mieux mes préférences.

[Q3] Toutes les heures, donner le compte rendu de la dernière heure des 100 transactions préférées de la catégorie 'IT'.

[Q4] Donner un compte rendu des 1000 dernières transactions concernant des titres français ayant une forte volatilité ($> 0,8$) et dont au moins une transaction a concerné un volume important (> 100).

L'extraction des données peut être faite de manière très précise selon les souhaits du moment. Ainsi, [Q2] travaille sur les 100 transactions préférées de Luc et en extrait celles de la catégorie IT. [Q3] va travailler sur les transactions de la catégorie IT et en extraire les 100 préférées. La fréquence du compte rendu est indiqué de manière temporelles (pour Q1, Q2 et Q3) ou positionnelle (pour Q4, toutes les 1000 transactions).

3. Architecture du système Stream2text

Cette section présente les grandes lignes du processus et l'architecture globale du système proposé.

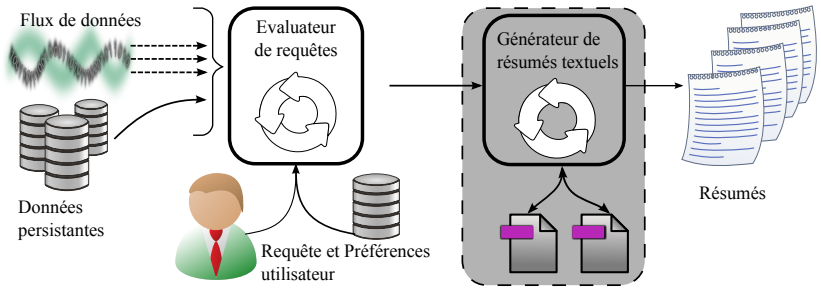


Figure 2. *Architecture globale de Stream2text*

Vue Globale du système : Stream2text (cf. figure 2) prend en entrée une requête utilisateur pour sélectionner les données pertinentes à résumer. Stream2text fournit un flux de textes résumant les données préférées de l'utilisateur.

La demande de l'utilisateur comporte d'une part "le point de vue" choisi par l'utilisateur et d'autre part une description de la fréquence et de la portée souhaitées pour les comptes rendus. Le point de vue permet d'orienter la rédaction du compte rendu en sélectionnant les données pertinentes. La fréquence rythme le déclenchement du processus de rédaction du compte rendu et la portée permet de limiter les données provenant des flux à un ensemble fini de données.

La requête de l'utilisateur s'exprime sur des flux et des données persistantes. Ceci est possible grâce à l'utilisation d'un modèle permettant d'exprimer de manière formelle les données qui sont nécessaires à la rédaction des comptes rendus. L'expression de préférences personnelles permet à l'utilisateur de limiter l'ensemble de données retournées au sous-ensemble satisfaisant le mieux ses choix personnels.

Une fois ces données disponibles, l'ensemble des données est agrégé de manière à obtenir un résumé numérique. Le résumé numérique est ensuite transcrit en langage naturel pour fournir un compte rendu textuel.

Architecture du générateur de textes : La rédaction du compte rendu textuel (cf. figure 3) s'appuie sur des informations concernant, d'une part le schéma des données et d'autre part les fonctions d'agrégations utilisées pour la phase de construction du résumé numérique.

Les informations nécessaires à la rédaction relatives au schéma sont principalement la "description" au format textuel des propriétés du schéma. Par exemple, le fait que *StOpName*=v peut être désigné dans un texte par "*L'action v*". Ou encore, qu'une instance de *StockOption* (représentée dans les données par un tuple $t \in StockOption$) peut se décrire par une phrase de type "*L'action* $t.StOpName$ *appartient à la catégorie* $t.Category$ *et l'entreprise est domicilié en* $t.Country$". Ces informations sont généralement disponibles puisqu'elles sont le produit des phases amonts (spécification) de la conception d'une application.

Les informations nécessaires à la rédaction du texte concernant les fonctions d'agrégation sont principalement la "description" au format textuel du sens associé aux

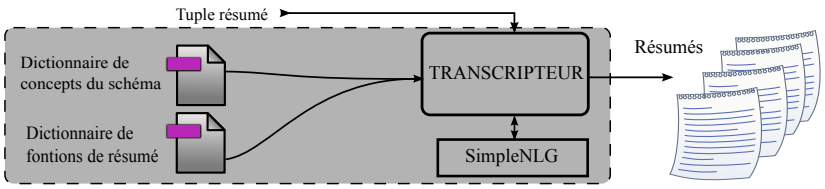


Figure 3. *Architecture du générateur de textes*

fonctions d'agrégation. Par exemple, une fonction d'agrégation $Avg(A)$ calculant la moyenne d'un attribut A peut s'exprimer comme "*La moyenne de* A *est* $Avg(A)$" ou encore "*Le* A *moyen(ne) est* $Avg(A)$".

4. Fondement théorique pour l'évaluation de requêtes

Cette section introduit les fondements pour l'évaluation de requêtes continues avec des préférences contextuelles ("évaluateur de requêtes" de la figure 2). Le lecteur familier avec ces aspects peut lire directement la section suivante.

4.1. *L'algèbre de flux ASTRAL*

Pour illustrer le propos, reprenons des requêtes de la section 2 sans les préférences ni le résumé textuel :
[Q1'] Chaque jour, informations concernant l'action $Total$ sur les 2 derniers jours.
[Q3'] Toutes les heures, informations concernant la catégorie 'IT' .
[Q4'] Informations pour les 1000 dernières transactions des actions françaises ayant une forte volatilité et dont au moins une transaction a un volume important.

Ces requêtes peuvent être exprimées en utilisant l'algèbre ASTRAL (Petit *et al.*, 2012b). Cette algèbre formalise l'expression de requêtes continues ou instantanées impliquant conjointement des flux et des relations. Ci-dessous, nous donnons quelques définitions nécessaires à la compréhension de ces requêtes qui permettent de traiter les données à la volée.

ASTRAL différencie les concepts de flux et de relation temporelle (Arasu *et al.*, 2004) qui sont notés S et R dans la suite. Un flux S est un ensemble potentiellement infini de n-uplets s ayant un schéma commun contenant deux attributs particuliers : un *timestamp* t, et une position [1] dans le flux p. Une *relation temporelle* R est une fonction qui fait correspondre à un identifiant temporel t un ensemble de n-uplets $R(t)$ ayant un même schéma. Les opérateurs de l'algèbre relationnelle (sélection σ, projection π, jointure $\bowtie$) sont étendus aux relations temporelles et σ et π aux flux. Ainsi, $\sigma_{Volume>10}(Transaction)$ est le flux des transactions dont le $Volume$ dépasse 10.

Une relation temporelle peut être extraite d'un flux à l'aide d'un opérateur de fenêtre. ASTRAL permet d'exprimer de nombreux types de fenêtres (Petit *et al.*, 2010) dont les fenêtres positionnelles comme, par exemple, l'ensemble des n derniers n-uplets, tous les m n-uplets, et les fenêtres temporelles comme, par exemple, l'ensemble des n-uplets arrivés pendant les x dernières secondes toutes les y secondes, ou

1. La notion de *batch* (Petit *et al.*, 2012b) ne sera pas utilisée dans cet article.

encore des fenêtres, dites *cross domain*, qui utilisent des positions et des *timestamp*. Par exemple, les n derniers n-uplets arrivés toutes les y secondes. Dans la suite de l'article, les fenêtres les plus utiles sont les suivantes :

– $S[L]$ est une fenêtre qui contient le dernier n-uplet du flux S (L pour $Last$) ;

– $S[N$ slide $\Delta]$ est une fenêtre de taille N glissant de Δ chaque Δ. N et Δ peuvent être, au choix, une durée ou un nombre de n-uplets.

Un flux peut être généré à partir d'une relation temporelle en utilisant un opérateur *streamer*. Par exemple, $I_S(R)$ produit le flux des n-uplets insérés dans la relation R. La jointure entre deux flux ou entre un flux et une relation s'exprime à l'aide des opérateurs de fenêtre, de *streamer* et d'une jointure sur des relations temporelles. Dans la suite on notera :

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

Ce flux $S \bowtie_c R$ contient les n-uplets générés par le flux S et ceux générés par des mises à jour dans R. On définit aussi l'opérateur *semi-sensitive-join* (noté $\bowtie\!\!\!\blacktriangleleft$) qui produit un flux résultant de la jointure entre le dernier n-uplet d'un flux et une relation temporelle à la date d'émission du n-uplet :

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

où τ_S est la fonction qui retourne la date d'émission d'un n-uplet dans le flux S. A titre d'exemple, voici la formulation des requêtes pré-citées :

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

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

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

4.2. *Modèle de préférences contextuelles*

Cette section présente les principaux concepts du formalisme logique employé pour *spécifier et raisonner* avec des préférences (présentation détaillée dans (de Amo et Pereira, 2010 ; Petit *et al.*, 2012a)). Intuitivement, une *règle de préférence contextuelle* (une cp-règle) permet de comparer deux n-uplets d'une relation R compatibles avec un contexte :

$$\varphi : u \rightarrow Q_1(X) \succ Q_2(X)[W]$$

où $X \subseteq Attr(R)$, $W \subseteq Attr(R)$ et $X \notin W$; $Q_i(X)$ (pour $i = 1, 2$) est un prédicat évaluable sur un n-uplet de R ; u est aussi un prédicat ne faisant intervenir ni X ni W (cf. exemple 1). Deux n-uplets sont comparables à l'aide d'une cp-règle, si ils ont les mêmes valeurs pour les attributs entre crochets dans la règle (attributs *ceteris paribus*).

Une *théorie de préférences contextuelles* (*cp-théorie*) sur R est un ensemble fini de cp-règles. Si la cp-theorie satisfait certaines conditions de consistance alors elle établit un ordre strict partiel sur l'ensemble des n-uplets. Cet ordre partiel sera utilisé pour distinguer les données préférées d'un utilisateur.

Exemple 1 Soit P1 et P2 les préférences de notre exemple de la section 2. Elles s'expriment par la cp-théorie suivante sur le schéma $T(StOpName, Category, Country, ETime, Rate, Method)$:
– $\varphi_1 : Category = \text{`}Commodities' \rightarrow (Rate < 0.25 \succ Rate \geq 0.25), [Method]$
– $\varphi_2 : Category = \text{`}IT' \rightarrow (Rate \geq 0.35 \succ Rate < 0.35), [Method]$
– $\varphi_3 : Rate > 0.35 \rightarrow (Country = Brazil \succ Country = \text{`}Venezuela')$

Les préférences utilisateur mentionnées en figure 2 sont des cp-théories comme celles de l'exemple 1. L'algèbre ASTRAL est étendue avec les opérateurs de préférence dans un contexte dynamique de *flux de données*. La sémantique considérée pour la relation de préférence est celle *avec contrainte*. La motivation est que dans un scénario de flux de données, il est raisonnable d'imaginer que la notion de préférence a le même caractère dynamique que les données sur lesquelles elle est appliquée.

4.3. *Opérateurs de préférences pour ASTRAL*

Cette section présente l'approche d'intégration des préférences contextuelles dans l'algèbre ASTRAL. Il s'agit d'opérateurs algébriques qui peuvent être utilisés aussi bien avec des requêtes instantanées que continues et portant sur des données persistantes ou sur des flux.

Les opérateurs de préférences calculent les données préférées par rapport à une cp-théorie de référence. Chaque utilisateur donne au système ses préférences sous forme d'une cp-théorie qui constitue une sorte de *profil utilisateur*. Ces préférences sont utilisées si la personnalisation de requêtes est demandée. Concrètement, cette solution permet d'introduire des requêtes « top-k » par l'intégration de l'opérateur ***KBest*** qui sélectionne le sous-ensemble des k données préférées en accord avec la hiérarchie des préférences spécifiée par la cp-théorie.

Pour illustrer l'extension de l'algèbre ASTRAL avec l'opérateur de préférence ***KBest***, considérons la requête **Q2** de la section 2. Son expression avec l'opérateur de préférence est :

$$(\sigma_{Category=\text{`}IT'}(\mathbf{KBest_{100}}((Volatility \bowtie Transaction) \bowtie StockOption)))[1h \text{ slide } 1h] \quad (4)$$

Alors que **Q3** s'écrit : $KBest_{100}(Q2')$.

4.4. *Fonctions d'agrégations et résumé de données*

Pour établir les résumés textuels qui seront générés, Stream2text passe par la création d'un résumé structuré. Celui-ci est créé grâce aux fonctions d'agrégation de l'éva-

luateur de requêtes. Les fonctions utilisées dépendent du domaine d'application. Intuitivement, une fonction d'agrégation f est une fonction qui associe à un ensemble de n-uplets un unique n-uplet dont les attributs et les valeurs sont déterminés par f.

Dans la suite de l'article nous utiliserons la définition 1. Dans cette version, chaque attribut agrégé est renommé à l'aide de la fonction utilisée pour calculer la valeur agrégée :

Definition 1 (Opérateur d'agrégat) *Soit R une relation temporelle de schéma $A = \{a_i\}_{i=1..n}$, n attributs de R, Soit $f^j(\{A_i\}_{i\in\{1,n\}})_{j=1..m}$, m fonctions d'agrégations. L'opérateur d'agrégation $\mathcal{G}_{f^1,f^2,...f^m}$ agrège l'ensemble des n-uplets de R en un n-uplet à l'aide des fonctions f^j.*

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

Rappelons que l'évaluateur de requêtes produit un ensemble de données qui sont ensuite agrégés dans un resumé structuré. Celui-ci est l'entrée du générateur de texte. Par exemple pour [Q1], notre utilisateur Luc peut demander un résumé avec la moyenne et la médiane pour le volume des transactions et les prix. Les valeurs obtenues par les fonctions d'agrégation constitueront le resumé structuré qui sera ensuite rédigé de manière appropriée en langue naturelle.

5. Opérateur de génération de textes

Pour automatiser la génération de textes, nous définissons des fonctions et des opérateurs permettant d'associer un texte à des données. Les sections 5.1 et 5.2 tirent profit des connaissances du schéma et des résumés numériques. La section 5.3 définit un opérateur de transcription qui transcrit une relation temporelle en langage naturel.

5.1. *Dictionnaire de concepts*

Ainsi, il est nécessaire d'associer à chaque propriété du modèle de données un fragment de texte. Ce dernier peut être utilisé pour nommer la propriété dans un texte. La définition 2 formalise cette notion sous la terminologie : *dictionnaire de concepts*.

Dans la suite, on considère une base de données relative à n_e entités/classes $\{E_i\}_{i=1..n_e}$. Chacune de ces classes ayant un ensemble n_i de propriétés/attributs $\{A_{i,j}\}_{j=1..n_i}^{i=1..n_e}$ dont certaines identifient de manière unique un objet dans la classe (attribut clef).

Definition 2 (Dictionnaire de concepts) *Un dictionnaire de concepts est une fonction $\mathcal{D}_c$ qui associe à chaque concept de la base de données (ie. propriété $A_{i,j}$) un groupe nominal GN. Ce groupe nominal peut être utilisé pour désigner le concept (la propriété $A_{i,j}$) dans un texte.*

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

où $\{GN\}_{i,j}$ est un groupe nominal qui nomme le concept $A_{i,j}$ en langage naturel.

L'exemple 2 illustre quelques valeurs possibles de la fonction $\mathcal{D}_c$.

Exemple 2 (Exemples d'entrées du dictionnaire de concepts)

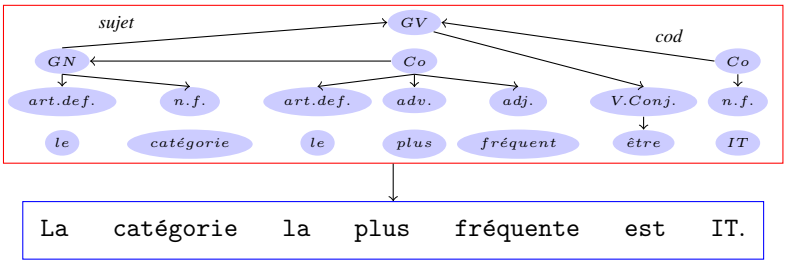
$$\mathcal{D}_c(StOpName) = \left\{ \begin{array}{cc} le & action \\ art.def. & n.f. \end{array} \right\} \quad \mathcal{D}_c(Price) = \left\{ \begin{array}{cc} le & prix \\ art.def. & n.m. \end{array} \right\}$$
$$\mathcal{D}_c(Price) = \left\{ \begin{array}{cc} le & cours \\ art.def. & n.m. \end{array} \right\}$$

Notons que la valeur de $\mathcal{D}_c$ pour un concept donné n'est pas unique. Ceci permet d'introduire diverses formulations et de limiter la répétition des textes.

5.2. *Dictionnaire de fonctions d'agrégat*

Le résumé textuel s'appuie sur un résumé numérique qui est obtenu à l'aide de fonctions d'agrégation. Pour la génération du texte à proprement parler, nous introduisons un dictionnaire de structures de phrase permettant d'exprimer le sens des fonctions d'agrégation et de la valeur calculée. Une structure de phrase est composée d'éléments de phrase ainsi que des relations entre ces éléments. Ces informations sont utilisées pour effectuer la *réalisation de surface* (cf. (Gatt et Reiter, 2009) et exemple 3) du texte c'est-à-dire son écriture en respectant les règles du langage naturel cible.

Exemple 3 (Structure de phrase et opération de réalisation) *Une structure de phrase, représentée sous forme de graphe, suivie de sa réalisation.*



La définition 3 propose une formalisation de la fonction permettant d'associer un texte au résultat d'une fonction d'agrégation F définie sur un ensemble de k attributs $\{a_i\}_{i=1..k}$. Le texte dépend à la fois des textes $\mathcal{D}_c(a_i)_{i=1..k}$ et du résultat du calcul de la fonction d'agrégation ie : $F(\{a_i\}_{i=1..k})$.

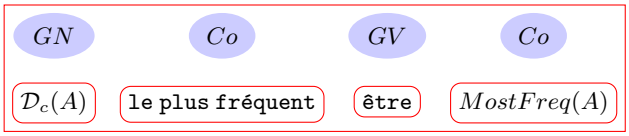
Definition 3 (Dictionnaire de fonctions d'agrégat) *Un dictionnaire de fonctions d'agrégat est une fonction $\mathcal{D}_f$ qui associe à une fonction d'agrégation $F(\{a_i\}_{i=1..k})$ une structure de phrase SP. La réalisation de SP permet de décrire le résultat de la fonction d'agrégation en langage naturel.*

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

où GV est un groupe verbal qui exprime la relation entre les attributs $\{a_i\}_{i=1..k}$ et $F(\{a_i\}_{i=1..k})$. $\{Co_i\}_{i=1..x}$ est un ensemble de x compléments (de nom, d'objet direct ou indirect) et $\{R_j\}_{j=1..y}$ un ensemble de y relations entre les éléments de la phrase.

L'exemple 4 est un exemple d'entrée du dictionnaire de fonctions d'agrégation.

Exemple 4 *Ci-après un exemple simplifié pour la fonction $MostFreq(A)$ qui* calcule la valeur la plus fréquente. *D'autres structures de phrases sont possibles.*



Une réalisation possible de cette structure de phrase est présentée dans l'exemple 3 avec l'attribut $Category$. La même fonction d'agrégat utilisée avec un autre attribut donnera une réalisation différente. Pour l'attribut $Country$, cela peut donner la réalisation suivante (en supposant $MostFreq(Country) = France$) :

Le pays le plus fréquent est France.

Ainsi, pour produire un texte, l'entrée du dictionnaire de fonction doit être *réalisée* et la fonction d'agrégat évaluée.

5.3. *Opérateur de transcription*

On dira qu'une relation temporelle (une fonction du temps) peut être *transcrite* s'il est possible de générer un ensemble de structures de phrases relatif à cette relation. Ainsi, transcrire en langage naturel la signification des données revient à produire un ensemble de structures de phrases décrivant les données d'une relation temporelle. Dans l'optique de la génération d'un résumé, la transcription intervient en fin de traitement lorsque le résumé structuré a été produit sous forme d'un n-uplet. La définition 4 explicite la forme des relations temporelles qui peuvent être transcrites.

Definition 4 (Relation transcriptible) *Une relation transcriptible est une relation temporelle R contenant un unique n-uplet t et : $\forall(A, v) \in t$ on a :*
– Soit A est un concept du dictionnaire (ie. $A \in Dom(\mathcal{D}_c)$)
– Soit $A = F$ où $F(\{a_i\}_{i=1..n})$ est une fonction d'agrégat utilisée pour agréger les valeurs des attributs $\{a_i\}_{i=1..n}$ en v. (ie. $F \in Dom(\mathcal{D}_f)$ et $(F, v) \in \mathcal{G}_F(R)$)

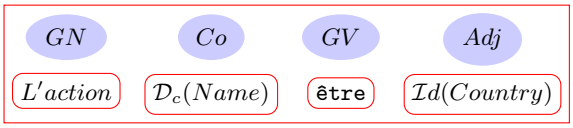
Nous définissons un opérateur de transcription qui retourne un ensemble de structures de phrases pour une relation transcriptible.

Definition 5 (Transcription de relation) *Un opérateur de transcription de relation $\mathcal{T}$ génère un ensemble de structures de phrases à partir d'une relation temporelle R de schéma F_i.*

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

R étant une relation temporelle, $\mathcal{T}(R)$ est un ensemble de structures de phrases évoluant dans le temps. L'utilisation d'un opérateur de création de flux sur cet ensemble dépendant du temps permet d'insérer dans un flux les textes estampillés par la date de mise à jour de R.

De manière générale, l'apparition d'une fonction identité $\mathcal{I}d$ dans les attributs d'une relation transcriptible signale le "point de vue" adopté pour résumer les données. Le dictionnaire de fonctions peut avoir une entrée pour la fonction $\mathcal{I}d$ avec une phrase introductive du résumé (par exemple, « Résumé des informations... »).
REMARQUE. — [
Cas particulier des attributs clef] Il est possible qu'apparaisse dans la liste des attributs d'une relation transcriptible la fonction identité $\mathcal{I}d$ appliquée à la clef d'une entité. Cela signifie que la fonction de transcription doit décrire l'objet et non le "point de vue". Dans ce cas, la partie du n-uplet correspondant n'est pas un *résumé* mais un objet de la base. Il est donc nécessaire de disposer d'une entrée spécifique dans le dictionnaire de fonction pour $\mathcal{I}d$ sur les attributs clefs. Par exemple, $\mathcal{D}_f(\mathcal{G}_{\mathcal{I}d(Name),\mathcal{I}d(Country)})$ peut être défini comme suit :



Ainsi la réalisation de :

$$\mathcal{T}(\mathcal{G}_{\mathcal{I}d(Name),\mathcal{I}d(Country)}(\pi_{Name,Country}(\sigma_{Name='Total'}(StOpName)))) \quad (5)$$

est L' action Total est française.

L'opérateur de transcription peut ainsi être utilisé pour construire des textes à partir de toute requête Astral avec ou sans préférences utilisateur. Il faut noter que le texte généré dépend uniquement du contenu des données et des fonctions d'agrégat, la requête n'est qu'indirectement transcrite en texte (voir exemple 5).

Exemple 5 (Opérateur de transcription) *Pour les requêtes de la section 2, supposons que les attributs numériques sont agrégés par la moyenne Avg et les non numériques par la valeur la plus fréquente $MostFreq$ alors Q1, Q3 et Q4 s'écrivent :*

$$\mathcal{T}(\mathcal{G}_{\mathcal{I}d(Name),\mathcal{I}d(Country),\mathcal{I}d(Category),Avg(Volume),MostFreq(Methode)}(\pi_{Name,Country,Category,Volume,Methode}(Q1'))) \quad (6)$$

$$\mathcal{T}(\mathcal{G}_{MostFreq(Name),MostFreq(Country),\mathcal{I}d(Category),Avg(Volume),MostFreq(Methode)}(\pi_{Name,Country,Category,Volume,Methode}(Q3'))) \quad (7)$$

$$\mathcal{T}(\mathcal{G}_{MostFreq(Name),\mathcal{I}d(Country),MostFreq(Category),Avg(Volume),MostFreq(Methode)}(\pi_{Name,Country,Category,Volume,Methode}Q4')) \quad (8)$$

6. Implémentation et expérimentation de Stream2text

Cette section décrit notre prototype et les expérimentations réalisées pour valider l'approche. Nous abordons particulièrement la partie transcription car l'évaluation des requêtes repose sur des logiciels existants (PostgreSQL et Asteroïde (Petit, 2012)).

6.1. *Transcripteur données - texte*

Le cœur du prototype est le transcripteur développé en Java. Il prend en charge la production du résumé textuel à partir des données traitées par l'évaluateur de requêtes. Le transcripteur trie les données par entités du dictionnaire de concepts de manière à planifier la structure du document. Il se charge de récupérer les groupes de mots selon la structure grammaticale des phrases et les transfère au réalisateur de surface SimpleNLG (Gatt et Reiter, 2009). Ce dernier a la responsabilité de faire le traitement de surface des phrases et retourne les phrases correctement construites au transcripteur. Le transcripteur assemble les phrases en paragraphes pour produire un texte plus complet. La structure des comptes rendus générés actuellement comporte :
– Une introduction qui résume les informations sur lesquelles le résumé est basé. Il s'agit d'informations sur le contenu de la fenêtre : nombre de données et taille.
– Plusieurs paragraphes, un pour chaque entité du domaine d'applications (schéma des données). Chaque paragraphe est composé de plusieurs phrases dont la structure provient du dictionnaire de fonctions d'agrégats.

Le **dictionnaire de fonctions d'agrégats** utilisé pour résumer les données a été initialisé avec les fonctions suivantes :
– $MostFreq$ qui calcule la valeur la plus fréquemment rencontrée dans l'échantillon de données. Cette fonction peut être utilisée pour résumer les attributs à valeur non numérique du schéma tels que $StOpName$, $Category$, $Country$ ou $Method$.
– $\mathcal{I}d(Key\_Attribute)$ qui correspond au cas particulier évoqué dans la remarque 5.3. Par exemple, si une fonction d'agrégat $identité$ est utilisée pour résumer l'attribut $StOpName$ qui est une clef de la relation $StockOption$. Le dictionnaire contient une structure de phrases pour chaque entité du schéma le de base de données.
– Avg, Med et $Count$ (avec la sémantique habituelle), calculent respectivement la valeur moyenne, la valeur médiane pour des attributs à valeur numérique ($Volume$, $Price$, $Rate$) et le nombre total de données.
– $Part(v, A_2)$ qui calcule la part, en %, d'une valeur v dans les valeurs de A_2.

Hormis pour $\mathcal{I}d(Key\_Attribute)$, les entrées du dictionnaire contiennent une structure de phrases générique qui peut être utilisée quelque soit l'attribut sur lequel la fonction est appliquée. Ce dictionnaire est donc indépendant du schéma de la base de données et peut être partagé entre les applications et les utilisateurs. Les fonctions choisies peuvent aussi être personnalisées pour obtenir des résumés plus appropriés.

6.2. *Expérimentation avec des données de la bourse*

Un jeu de données, conforme à l'exemple, a été construit à partir de données réelles (http ://www.abcbourse.com). Les données correspondent aux cours de douze actions de dix catégories et de trois pays. Les données sont horodatées. Cette date est utilisée comme estampille pour les flux (ie. $TTime$ et $ETime$). La quantité ($Volume$), le cours ($Price$) des transactions ainsi que la volatilité ($Rate$) sont aussi estampillés. Le

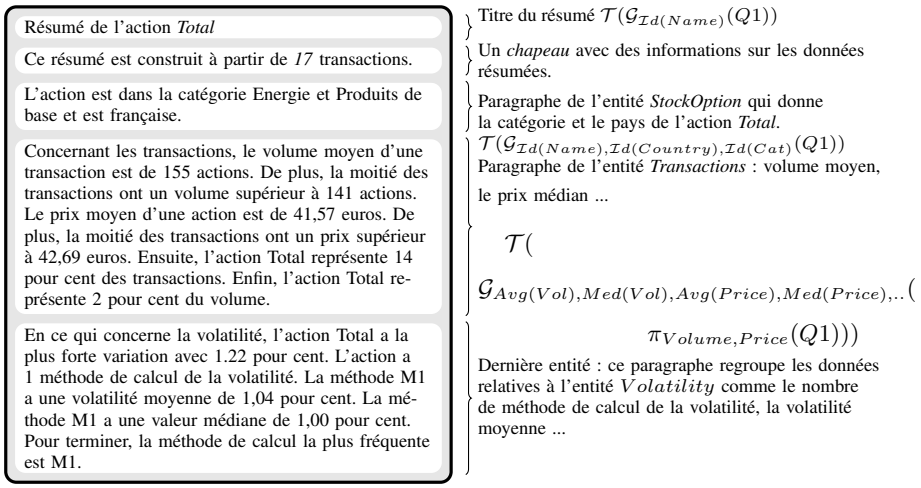


Figure 4. *Texte pour la requête Q1 selon l'exemple 6*

secteur d'activité ($Category$) et le pays ($Country$) de l'action sont disponibles. Le jeu de données ainsi constitué est composé d'environ 5000 transactions.

Le **dictionnaire de concepts** du schéma, décrit les sept concepts organisés en trois entités (cf. section 2) : l'action (attribut $StOpName$), la catégorie ($Category$), le pays ($Country$), le volume ($Volatity$), le prix ($Price$), le taux ($Rate$) et la méthode ($Method$). Pour chacun de ces concepts, une entrée comportant un groupe nominale est créée dans le dictionnaire de concepts.

Nous avons expérimenté la génération de résumés pour des requêtes analogues à celles présentés dans cet article. A titre d'illustration, nous présentons l'exemple 6.

Exemple 6 (Génération de résumé) *Pour [Q1] Luc souhaite dans son résumé la moyenne et la médiane pour le volume des transactions, pour les prix, etc. [Q1] peut s'écrire sous une forme similaire à l'équation 6. Rappelons que [Q1] analyse les données de manière continue et que le résumé sera produit successivement selon la fenêtre temporelle. La figure 4 montre les textes obtenus pour une période de 2 jours.*

Cette expérimentation nous permet de valider l'approche. A ce jour il n'y a pas eu d'expérimentations destinées à d'autres mesures de performance du système.

7. Travaux connexes

Les travaux connexes à cette proposition peuvent se grouper en trois groupes : requêtes continues sur flux, résumés numériques et "natural language generation".

Pour maîtriser les requêtes continues sur les flux de données, des travaux importants ont été réalisés tant d'un point de vue fondamental (Krishnamurthy *et al.*, 2010) que pratique (Arasu *et al.*, 2006). Dans cet article nous utilisons ASTRAL (Petit *et al.*, 2012a) qui présente l'avantage de définir de manière non-ambiguë les opérateurs sur des flux et des relations temporelles. Ceci est particulièrement important pour les jointures (Petit *et al.*, 2012b) et les fenêtres (Petit *et al.*, 2010). D'autre part, il existe de nombreux travaux sur la manière de résumer ou de synthétiser "numériquement"

des données. Dans ce contexte, on peut comprendre les modèles de préférences et les opérateurs de type $top - k$ comme un moyen de réduire la quantité de données manipulées. Notre proposition utilise des requêtes top-k mais se base aussi sur l'existence de méthodes permettant de résumer et/ou d'agréger un ensemble de valeurs en une unique valeur. Les travaux de cette nature, par exemple (Cormode *et al.*, 2012 ; Cormode et Muthukrishnan, 2005), peuvent être utilisés dans la phase d'agrégation de notre proposition. Notre proposition est suffisamment générique pour pouvoir être utilisée avec divers évaluateurs de requêtes. L'extension du langage de requêtes par un modèle de préférences (Koutrika *et al.*, 2010) n'a pas pour l'instant d'impact direct sur la transcription en langage naturel du résumé structuré. Un changement du modèle de préférences entraînerait une modification du calcul du résumé structuré mais ne modifierait pas la transcription en langage naturel. Le choix de CPrefSQL dans ASTRAL (Petit *et al.*, 2012a) est motivé par son caractère qualitatif et la possibilité de prendre en compte le "contexte" dans le calcul des n-uplets dominants. Ceci diffère des approches par fonctions de score (Borzsonyi *et al.*, 2001 ; Papadias *et al.*, 2005 ; Kontaki *et al.*, 2010).

On peut distinguer deux grandes classes d'approches pour la génération automatique de textes. L'une consiste à générer un texte à partir d'un ou plusieurs textes (*text-to-text*). Cette approche est utilisée pour résumer automatiquement des textes (Rotem, 2003) ou des opinions (Labbé et Portet, 2012). L'autre approche consiste à générer des textes qui expliquent et/ou décrivent des données (*data-to-text*). C'est dans cette dernière approche que se place notre proposition.

A notre connaissance, les travaux relevant de ce domaine, restent spécifiques à un domaine d'application. Les exemples les plus aboutis concernent la médecine (Portet *et al.*, 2009 ; Gatt *et al.*, 2009) ou la météo (Turner *et al.*, 2010). La communauté "natural language generation" travaille en particulier sur des aspects avancés du langage qui eux sont indépendants du domaine d'application ciblé : agrégation de phrases, construction de phrases énumératives, expressions référentielle,... Les phases en amont du processus d'élaboration du texte comme la détermination du contenu et la planification (Reiter et Dale, 2000) restent spécifiques au domaine d'application et nécessitent l'intervention d'experts du domaine. Cependant (Androutsopoulos *et al.*, 2013) propose une approche permettant de décrire en langage naturelle les individus ou les classes d'une ontologie OWL. Dans notre contexte, cela est assimilable à la description d'un n-uplet ou d'une relation de la base de données et non pas à la description des informations agrégées comme nous le proposons.

Dans notre approche, la détermination du contenu et la génération des phrases sont facilités puisqu'elle met à profit d'une part les connaissances conceptuelles sur la structure des données (le schéma) et d'autre part les connaissances sur les méthodes utilisées pour générer le résumé structuré des données. Les éventuelles connaissances dépendantes du domaine d'application qui sont nécessaires à la génération de texte sont capturées par le dictionnaire de schéma qui peut être élaboré lors de la description des données (par exemple lors de la spécification). Les connaissances relatives aux fonctions de résumé structuré sont elles génériques. Notre proposition met à profit la

description (la spécification) d'un ensemble de données pour en générer un résumé à l'aide d'un réalisateur de surface (Gatt et Reiter, 2009).

8. Conclusion

Notre travail s'inscrit dans un effort très actuel qui vise à maîtriser les grandes masses de données auxquelles les infrastructures informatiques doivent faire face. Nous pensons que la capacité de générer des textes courts permettant d'offrir à l'utilisateur une description synthétique de ces points d'intérêts est un atout majeur. Notre proposition prend la forme d'un système *Stream2text* qui permet de fournir un résumé en langage naturel de l'ensemble des données qui ont de l'importances pour l'utilisateur. L'approche adoptée repose sur l'utilisation des connaissances sur le schéma des données et sur la manière de les résumer. Plus particulièrement sur la manière d'exprimer en langage naturel les différentes opérations réalisées lors de la phase d'agrégation des données. Le système n'impose pas de contraintes sur les fonctions d'agrégation utilisées. L'approche proposée est indépendante de l'application visée et repose sur une intégration de bout en bout allant de requêtes utilisateur à l'expression en langage naturel des réponses. L'architecture proposée utilise des dictionnaires de concepts et de fonctions qui ouvre plus de perspectives de personnalisation pour mieux adapter le langage utilisé aux utilisateurs.

Ce travail peut être poursuivi selon de nombreux axes. Le texte généré doit pouvoir refléter les éventuelles valeurs manquantes et contenir des indications sur la fenêtre temporelle utilisée. L'aspect interface pour utilisateurs non-informaticiens serait certainement important pour faciliter l'utilisation d'un tel système. Sur les aspects données il y'a de nombreuses perspectives dont l'optimisation de l'approche, la détection d'évènements complexes et la mise à profit d'ontologies de domaines. D'autres axes de recherche sont liés à la génération automatique de texte : références à des évènements passés (ie. : *contrairement à hier*), agrégation de phrases, etc. Il est important de ne pas découpler ces différents axes de recherche car ils sont intimement liés.

9. Bibliographie

Androutsopoulos I., Lampouras G., Galanis D., « Generating Natural Language Descriptions from OWL Ontologies : the NaturalOWL System », *Journal of Artificial Intelligence Research*, vol. 48, 2013, p. 671-715.

Arasu A., Babcock B., Babu S., Cieslewicz J., Datar M., Ito K., Motwani R., Srivastava U., Widom J., « STREAM : The Stanford Data Stream Management System », Technical report, 2004, Stanford InfoLab.

Arasu A., Babu S., Widom J., « The CQL continuous query language : semantic foundations and query execution », *Proc. of 32nd int. conf. on Very Large Data bases*, vol. 15, 2006.

Borzsonyi S., Kossmann D., Stocker K., « The Skyline Operator », *Proceedings of the 17th International Conference on Data Engineering (ICDE 2001)*, 2001, p. 412–430.

Cormode G., Garofalakis M. N., Haas P. J., Jermaine C., « Synopses for Massive Data : Samples, Histograms, Wavelets, Sketches », *Foundations and Trends in Databases*, vol. 4, n° 1-3, 2012, p. 1-294.

Cormode G., Muthukrishnan S., « An improved data stream summary : the count-min sketch and its applications », *J. Algorithms*, vol. 55, n° 1, 2005, p. 58-75.

de Amo S., Pereira F., « Evaluation of Conditional Preference Queries. », *Journal of Information and Data Management (JIDM). Proceedings of the 25th Brazilian Symposium on Databases, 2010, Belo Horizonte, Brazil.*, vol. 1(3), 2010, p. 521-536.

Gatt A., Portet F., Reiter E., Hunter J., Mahamood S., Moncur W., Sripada S., « From data to text in the neonatal intensive care unit : Using NLG technology for decision support and information management », *AI Communications*, vol. 22, n° 3, 2009, p. 153-186.

Gatt A., Reiter E., « SimpleNLG : A Realisation Engine for Practical Applications », *Proceedings of the 12th European Workshop on Natural Language Generation*, ENLG '09, Stroudsburg, PA, USA, 2009, Association for Computational Linguistics, p. 90–93.

Kontaki M., Papadopoulos A., Manolopoulos Y., « Continuous Processing of Preference Queries on Data Streams », *Proc. of the 36th Int. Conf. on Current Trends in Theory and Practice of Computer Science (SOFSEM 2010)*, Springer, 2010, p. 47-60.

Koutrika G., Pitoura E., Stefanidis K., « Representation, composition and application of preferences in databases. », *Proc. of Int. Conf. on Data Engineering*, 2010, p. 1214-1215.

Krishnamurthy S., Franklin M., Davis J., Farina D., Golovko P., Li A., Thombre N., « Continuous analytics over discontinuous streams », *SIGMOD '10 : Proc. of the 2010 ACM SIGMOD int. conf. on Management of data*, ACM, 2010, p. 1081–1092.

Labbé C., Portet F., « Towards an Abstractive Opinion Summarisation of Multiple Reviews in the Tourism Domain », *The First International Workshop on Sentiment Discovery from Affective Data (SDAD 2012)*, Bristol, UK, sep 2012, p. 87–94.

Papadias D., Tao Y., Fu G., Seeger B., « Progressive Skyline Computation in Database Systems », *ACM Transactions on Database Systems*, vol. 30, 2005, p. 41–82.

Petit L., Labbé C., Roncancio C. L., « An Algebric Window Model for Data Stream Management », *Proceedings of the 9th Int. ACM Workshop on Data Engineering for Wireless and Mobile Access (MobiDE '10)* , ACM, 2010, p. 17–24.

Petit L., de Amo S., Roncancio C., Labbé C., « Top-k Context-Aware Queries on Streams », *Proc. of Int. Conf. on Database and Expert Systems Applications*, 2012, p. 397-411.

Petit L., Labbé C., Roncancio C. L., « Revisiting Formal Ordering in Data Stream Querying », *Proc. of the 2012 ACM Symp. on Applied Computing*, New York, NY, USA, 2012, ACM.

Petit L., « Gestion de flux de donnés pour l'observation de systèmes », Thèse de doctorat, Université de Grenoble, Décembre 2012.

Portet F., Reiter E., Gatt A., Hunter J., Sripada S., Freer Y., Sykes C., « Automatic Generation of Textual Summaries from Neonatal Intensive Care Data », *Artificial Intelligence*, vol. 173, n° 7–8, 2009, p. 789–816.

Reiter E., Dale R., *Building Natural Language Generation Systems*, Cambridge University Press, New York, NY, USA, 2000.

Rotem N., « Open text summarizer (OTS) », online, 2003, June, 2012, http ://libots.sourceforge.net.

Turner R., Sripada S., Reiter E., « Generating Approximate Geographic Descriptions », Krahmer E., Theune M., Eds., *Empirical Methods in Natural Language Generation*, vol. 5790 de *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 2010, p. 121-140.