

Modèle tensoriel pour l'entreposage et l'analyse des données des réseaux sociaux Application à l'étude de la viralité sur Twitter

Eric Leclercq, Marinette Savonnet

▶ **To cite this version:**

Eric Leclercq, Marinette Savonnet. Modèle tensoriel pour l'entreposage et l'analyse des données des réseaux sociaux Application à l'étude de la viralité sur Twitter. INFORSID 2018, May 2018, Nantes, France. hal-01821102

HAL Id: hal-01821102
https://u-bourgogne.hal.science/hal-01821102

Submitted on 23 Jul 2019

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.

Modèle tensoriel pour l'entreposage et l'analyse des données des réseaux sociaux

Application à l'étude de la viralité sur Twitter

Éric Leclercq, Marinette Savonnet

Laboratoire LE2I - FRE 2005 - CNRS - Arts et Métiers
Univ. Bourgogne Franche-Comté
9, Avenue Alain Savary
F-21078 Dijon - France
prénom.nom@u-bourgogne.fr

RÉSUMÉ. *Dans cet article, nous montrons comment la notion de tenseur permet de construire un modèle multi-paradigmes pour l'entreposage des données sociales. D'un point de vue architecture, cette approche permet de lier différents systèmes de stockage (polystore) et de limiter l'impact des outils ETL réalisant les transformations de modèles pour alimenter différents algorithmes d'analyse. Ainsi, le modèle proposé permet d'assurer l'indépendance logique entre les données et les programmes implantant les algorithmes d'analyse. Avec un cas concret extrait d'une étude de la viralité sur Twitter durant la période de l'entre deux tours de l'élection présidentielle française de 2017, nous mettons en évidence les apports de notre modèle.*

ABSTRACT. *In this article, we show how the notion of tensor can be used to build a multi-paradigm model for the storage of social data. From an architectural point of view, this approach allows to link different storage systems (polystore) and thus limits the impact of ETL tools performing model transformations to feed different analysis algorithms. The proposed model allows to reach the logical independence between data and programs implementing analysis algorithms. With a concrete case study on message virality on Twitter during the period between the two rounds of the French presidential election of 2017, we highlight the contributions of our model.*

MOTS-CLÉS : *stockage polyglotte, réseau multi-relationnel, tenseur, OLAP*

KEYWORDS: *polyglot storage, multi-relational network, tensor, OLAP*

1. Introduction : contexte et problématique

Les données des réseaux sociaux, en particulier celles de Twitter, sont de plus en plus exploitées dans des projets de recherche appliquée, en sciences sociales par exemple. Ces données, riches en informations sur les interactions entre individus, permettent aux chercheurs de comprendre les modèles de communication de la société numérique et les interactions entre les réseaux sociaux numériques, les médias traditionnels et la réalité. Les résultats de ces recherches sont applicables dans de nombreux domaines comme le marketing, le journalisme, l'étude de l'impact des politiques publiques, l'étude de la communication politique, etc.

Cependant, pour éclairer leurs questions de recherche les chercheurs en sciences sociales doivent effectuer des analyse exploratoires sur les données, émettre des hypothèses, développer des modèles (Armatte, 2005) et les tester. Cette démarche nécessite généralement l'utilisation de plusieurs algorithmes de fouille de données, de *machine learning* et requiert une phase d'interprétation des résultats exploitant la connaissance du domaine. Les différentes catégories d'algorithmes permettent de détecter des communautés (Drif, Boukerram, 2014), des événements (Atefeh, Khreich, 2015), des utilisateurs influents (Riquelme, González-Cantergiani, 2016), simuler ou étudier des propagations de messages. Les algorithmes ont recours à des modèles de données variés comme des graphes, des matrices d'adjacence, des séries temporelles. De plus, les algorithmes n'utilisent pas tous de la même manière les données, par exemple un algorithme de graphe peut optimiser une fonction et/ou effectuer une marche aléatoire sur le graphe (*random walk*), ou encore détecter les arêtes d'un graphe par lesquelles passent le plus grand nombre de plus courts chemins. Les algorithmes récents d'analyse des données des réseaux sociaux sont rarement implantés dans les SGBD et encore plus rarement les opérations matricielles et les factorisations associées (LU, SVD, CUR, etc.) (Leskovec *et al*., 2014). Seuls quelques systèmes NoSQL comme Neo4j proposent des outils d'analyse de graphe de données assez avancés [1]. Cependant, Neo4j ne permet pas de gérer de grandes quantités de données attributaires comme le feraient les systèmes orientés colonnes (Hölsch *et al*., 2017).

Pour le traitement des masses de données, certains algorithmes peuvent être exécutés sur des clusters de machines. On assiste depuis quelques années à la convergence de deux domaines de recherche séparés : le calcul hautes performances (*High Performance Computing - HPC*) et les bases de données. Ainsi, une des préoccupations du *data intensive HPC* est de pouvoir alimenter rapidement les algorithmes avec les données à analyser. Pour ce faire plusieurs types de stockage peuvent être combinés (systèmes de fichiers distribués HDFS, bases de données orientées colonnes, etc.) sous la forme d'un *polystore* ou système de stockage multi-paradigmes dénommé aussi stockage polyglotte (*polyglot storage*). Dans ce type de systèmes, les données peuvent être stockées dans le modèle qui convient le mieux au type de données et aux algorithmes ; une duplication partielle peut aussi être opérée. Les *polystores* se distinguent

1. https://neo4j.com/blog/efficient-graph-algorithms-neo4j/

des *data-lake* selon deux aspects, le premier concerne la finalité, les données stockées dans un *polystore* le sont dans un objectif d'analyse à court terme, celles dans un *data-lake* sont annotées en vue d'une utilisation à moyen, long terme, le second aspect concerne les performances, un *polystore* est conçu pour exploiter au mieux les modèles de stockage les plus adaptés aux données en les combinant, les *data-lake* stockent le plus souvent les données dans leur format natif.

Les travaux qui concernent les *polystores* sont axés principalement sur l'uniformisation des systèmes par les langages mais peu proposent une approche orientée modèle. C'est ce que nous proposons d'étudier dans cet article (voir figure 1).

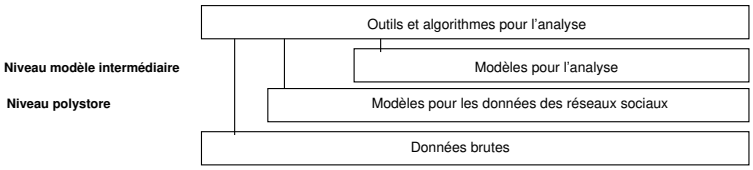


Figure 1. Relations entre les modèles et les outils d'analyse

Dans une partie état de l'art, nous discutons des modèles d'entrepôt pour les données sociales et nous présentons les principaux *polystores*. Nous décrivons ensuite notre proposition de modèle à base de tenseur et montrons comme il s'intègre dans une architecture d'entrepôt de données *polystore*. Ce modèle permet de généraliser les modèles de graphes (multi-couches, multi-relationnel), les séries temporelles et les modèles matriciels mais aussi de lier différentes données provenant des systèmes de stockage. Dans une troisième partie, nous montrons l'intérêt de notre modèle pour l'étude de l'impact des robots dans les phénomènes de viralité sur Twitter.

2. État de l'art

Dans cette section, nous présentons les différentes approches pour l'entreposage de données sociales du point de vue des modèles et de l'OLAP (*OnLine Analytical Processing*) puis du point de vue des polystores. Nous terminons cette section par une discussion sur les atouts et les faiblesses de ces solutions.

2.1. Modèles OLAP pour les données sociales

Depuis 2010, différents travaux ont cherché à étendre ou à adapter les techniques OLAP pour les données des réseaux sociaux. Les approches les plus nombreuses s'orientent vers un modèle d'entrepôt R-OLAP (*Relational-OLAP*) pour un usage particulier. Dans le modèle R-OLAP, les dimensions sont les axes avec lesquels les analyses sont effectuées et les faits sont sur quoi portent les analyses. Une façon de mettre en relation les dimensions et les faits est réalisée par le modèle en étoile ou en flocon. Par exemple, il peut y avoir une dimension temps, une dimension client, une dimension géographie pour une étude de produits.

Pour les données des réseaux sociaux, les tables de faits et de dimensions sont construites en vue d'analyses spécifiques. Par exemple, l'article de Bringay *et al*. (2011) se concentre sur la localisation, le temps et les mots importants apparaissant dans les tweets. Ces mots sont déterminés à l'aide d'une extension de TF-IDF qui déterminent les mots importants en fonction de leur place dans une hiérarchie de termes. D'autres auteurs ont construit un entrepôt dédié à l'analyse de sentiments à partir du contenu de tweets, avec le modèle R-OLAP (Moalla, Nabli, 2014) ou avec un hypercube (Moya *et al*., 2011). Bouillot *et al*. (2012) ne s'intéressent qu'à la dimension localisation des tweets en fonction du lieu d'émission du tweet, du contenu géographique, du lieu de résidence du compte.

D'autres recherches ont un caractère plus générique et proposent une modélisation en étoile des données sociales. Les travaux de Rehman *et al*. (2012) décrivent un modèle R-OLAP centré autour du compte utilisateur comme table de fait pour connaître son activité au cours du temps. Cet entrepôt a été utilisé pour analyser le comportement des utilisateurs lors d'un tremblement de terre. Une approche similaire est développée dans Mansmann *et al*. (2014). Kraiem *et al*. (2015) proposent un modèle conceptuel générique pour Twitter et le traduisent en R-OLAP : deux tables de faits sont construites une pour l'activité du tweet et l'autre pour l'activité du compte utilisateur en prenant en compte les dimensions sources, temps, lieu et utilisateurs. Kazienko *et al*. (2011) développent une modélisation hypercube des réseaux sociaux où les interactions entre groupes d'individus sont modélisées. Mansmann (2008) présente des travaux plus généraux, orientés vers une extension des technologies OLAP pour les données complexes sans réellement prendre en compte les algorithmes d'analyse.

Costa *et al*. (2012), mettent en évidence la nécessité de contextualiser les données pour aider l'analyse. Les approches décrites dans (Gallinucci *et al*., 2013) et (Rehman *et al*., 2013) élaborent un modèle intégrant un enrichissement sémantique des données collectées.

Une autre piste de recherche est apparue en 2008 avec la publication de (Chen *et al*., 2008). Le *Graph OLAP* consiste à construire un cube de graphes dans lequel il est possible de naviguer. Deux types de dimensions ont été définis : les dimensions informationnelles comme le temps, le lieu et les dimensions topologiques qui se rapportent à la modélisation des graphes dans les cellules du cube. Dans (Zhao *et al*., 2011), les auteurs étendent *Graph OLAP* et proposent le modèle *Graph Cube* pour modéliser les réseaux complexes incluant des nœuds avec des attributs multiples, et définissent un ensemble d'opérateurs spécifiques aux graphes. Le modèle est implémenté en C++ et expérimenté sur des graphes de taille moyenne à partir de jeux de données issus de DBLP [2] et IMDB [3]. Favre *et al*. (2017) étudient une autre approche qui consiste à enrichir le graphe grâce à des cubes qui valuent les nœuds et les arêtes. Loudcher *et al*. (2015) proposent un état de l'art et une étude comparative des différentes approches.

2. http://dblp.uni-trier.de/xml/
3. http://www.imdb.com/interfaces/

2.2. Entrepôt polystore

La problématique de l'accès à des sources de données hétérogènes est traitée depuis de nombreuses années dans le contexte de l'intégration de schéma et des approches multi-bases (Özsu, Valduriez, 2011). Les systèmes de stockage orientés Big Data comme HDFS et les systèmes NoSQL, matures depuis quelques années, ont fait évoluer la problématique de l'accès aux sources de données hétérogènes (Franklin *et al*., 2005). Dans ce cadre, les approches *polystore* ou *multistore* ont pour but de permettre un accès intégré à de multiples systèmes de stockage de données ne supportant pas nécessairement SQL et offrant des jeux d'opérateurs différents, au travers d'un gestionnaire de requête (Bondiombouy *et al*., 2015). Une des hypothèses fortes retenue par ces systèmes est que les accès aux sources sont principalement en lecture.

Bondiombouy et Valduriez (2016) proposent un état de l'art des principaux systèmes en les classant selon trois catégories : systèmes faiblement couplés, fortement couplés et hybrides. Les systèmes faiblement couplés et fortement couplés s'opposent principalement sur deux critères : les performances sont privilégiées dans les systèmes fortement couplés au détriment de l'autonomie et de l'accès local aux sources de données. Nous distinguerons les systèmes selon leur orientation initiale : extension d'outils d'analyse ou approches systèmes de gestion de bases de données distribuées (Özsu, Valduriez, 2011).

Le premier groupe concerne des systèmes issus de projets collaboratifs et pragmatiques. Ces systèmes utilisent un moteur de requêtes SQL comme couche de médiation. La couche SQL de la plateforme Apache Spark [4] et le projet Apache Drill [5] sont deux solutions opérationnelles de ce groupe. Ces systèmes exploitent le principe de localité des données (*data locality*) et implantent un optimiseur de requêtes.

Le second groupe concerne des systèmes issus de travaux de recherche d'équipes ayant déjà contribué à résoudre la problématique d'accès à des sources de données hétérogènes dans le contexte des systèmes de gestion de bases de données traditionnels. Le système BigDAWG (Duggan *et al*., 2015) permet d'écrire des requêtes multi-bases portant sur des îlots d'information correspondant chacun à un type de modèle de données. Le langage supporté permet des accès transparents aux différents éléments d'un même îlot. L'approche Cloud Multidatastore Query Language (CloudMdsQL) (Kolev *et al*., 2016) définit un langage fonctionnel de type SQL qui permet d'écrire des requêtes composées de sous-requêtes permettant d'interroger plusieurs systèmes de stockage hétérogènes. Chaque sous-requête cible un système particulier et contient des appels à l'interface native du système en question. Ainsi CloudMdsQL peut exploiter les spécificités et les performances des système locaux au moyen des requêtes natives comme par exemple pour une requête *breadth-first search* sur une base de données graphe. SQL++, inclus dans la plateforme FORWARD [6], est une approche de média-

4. http://spark.apache.org/sql/
5. https://drill.apache.org/
6. http://forward.ucsd.edu/

tion de schéma qui a pour objectif de définir un intergiciel permettant de créer des vues virtuelles ou matérialisées sur des systèmes supportant ou non le langage SQL (Ong *et al*., 2014 ; 2015). Le modèle de données sur lequel SQL++ repose est un sur-ensemble du modèle relationnel et de JSON. Comparée aux systèmes BigDAWG et CloudMdsQL, l'approche SQL++ se concentre sur les aspects langage et extensibilité mais aborde peu l'architecture du système.

En conclusion, les systèmes présentés, comparés aux approches traditionnelles, combinent les avantages des systèmes faiblement couplés au niveau des accès à de multiples systèmes de stockage avec les avantages des systèmes fortement couplés au niveau de l'efficacité des accès utilisant les interfaces natives. Dans cette orientation, CoudMdsQL semble l'approche la plus développée alors que SQL++ met l'accent sur une compatibilité SQL, BigDAWG quand à lui offre une approche plus restrictive car chaque îlot d'information correspond à un seul modèle de données.

2.3. Discussion

Les limites générales des approches OLAP pour des données des réseaux sociaux sont d'une part les performances et d'autre part l'évolutivité des schémas. Du point de vue des performances, les modélisations induisent des requêtes nombreuses et coûteuses lorsqu'on applique des algorithmes de graphe nécessitant un parcours de liens (dont les marches aléatoires), la recherche de plus court chemin ou la construction de matrice d'adjacence (par nature creuse). De plus, les transformations de modèle sont coûteuses comme par exemple le passage d'un graphe à des séries temporelles pour différents échantillonnages du temps, ou l'agrégation de données (*roll-up*) avec une exploitation des relations transitives comme les citations d'utilisateurs dans des tweets. Du point de vue de l'évolutivité des schémas, le problème ne se situe pas au niveau des opérations permettant de transformer les données mais plutôt de la connaissance nécessaire afin de déterminer les dimensions pertinentes pour les analyses. Comme le faisait remarquer Byron Ruth dans une présentation intitulée "*ETL: The Dirty Little Secret of Data Science*" lors de la conférence OSCON [7], les ETL sont des processus coûteux à mettre en place et les scripts de transformation de données incluent bien souvent une connaissance implicite qui ne favorise pas leur réutilisation.

Les polystores sont peu considérés dans les approches d'entreposage qui restent encore majoritairement dans une vision traditionnelle du SGBD. Les approches décrites partagent le même principe, c'est-à-dire une unification par le langage de requête tout en préservant pour CloudMdsQL les requêtes natives. Au niveau modèle, il s'agit dans tous les cas soit du modèle relationnel avec ou sans imbrication soit d'un modèle de type JSON.

7. https://conferences.oreilly.com/oscon/oscon2014/public/schedule/detail/34578

3. L'approche TDM (*Tensor Data Model*)

Notre approche fait comme hypothèse de préserver l'autonomie locale des systèmes sans considérer les requêtes de mises à jour des données hormis celles qui consistent à matérialiser les résultats des analyses. Il s'agit donc d'une approche multi-paradigmes de l'entreposage par *polystore* dans le sens où il est possible d'utiliser soit le langage propre à chaque système soit le modèle intermédiaire tensoriel servant à faciliter les transformations de modèles vers les algorithmes d'analyse (figure 2). Le modèle tensoriel assure ainsi le découplage entre les programmes et les données (*logical data independency*) comme l'illustre la figure 3. Du point de vue des outils d'analyse nous réduisons pour l'instant notre système à deux types de langages : R et Spark [8]. Nous préciserons la notion de tenseur dans les sections suivantes. Dans un premier temps nous nous limiterons à l'analogie d'un tenseur avec un tableau multi-dimensionnel ou une hypermatrice, c'est-à-dire une famille d'éléments indexée par N ensembles, N étant l'ordre du tenseur autrement dit le nombre de dimensions.

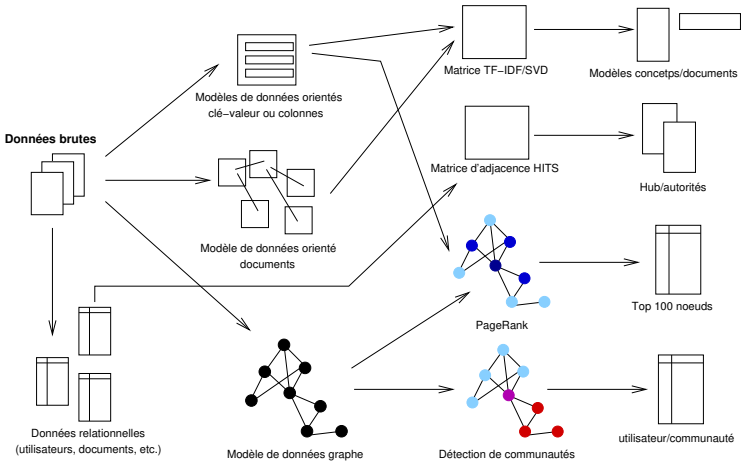


Figure 2. Modèles, transformations de modèles, algorithmes

3.1. Architecture

La figure 3 décrit comment les données qui peuplent les tenseurs du modèle intermédiaire sont obtenues avec un adaptateur (*wrapper*) qui exprime les requêtes soit dans le langage de manipulation de données du système de stockage (en R par exemple) ou en utilisant une sur-couche SQL (SparkSQL pour Spark). Dans les deux

8. Tensor flow https://www.tensorflow.org/ est comparable aux bibliothèques supportant les tenseurs dans R ou Spark, tous comme ces dernières, la bibliothèque n'a pas été construite avec une orientation modèle de représentation de données. Il s'agit plutôt d'une structure d'échange entre les processus d'un *workflow* complexe qui offre les opérateurs classiques de manipulation et décomposition des tenseurs.

cas, les données extraites transitent par une structure *data-frame* avant d'être modélisées dans un tenseur. Les requêtes de construction de tenseur soumises aux *wrappers* ont la particularité d'avoir toutes la même forme : elles renvoient $N + 1$ attributs où les N premiers attributs indiquent les dimensions et le dernier sert de valeur pour les éléments du tenseur (obtenu avec l'ajout d'une clause `GROUP BY` sur les attributs qui représentent les dimensions). Cette particularité nous permet d'avoir des *wrappers* ayant tous la même forme et ainsi simplifier les transformations de modèles. Pour réaliser les *wrappers* nous utilisons les packages R DBI, RNeo4j, RMongo, RCassandra et RHBase, pour Spark nous utilisons la couche SQL et les données sont stockées sous la forme de *data frames* et RDD (*Resilient Distributed DataSets*). Les dimensions des tenseurs sont représentées par des tableaux associatifs notés ta_i pour $i = 1, .., N$ qui contiennent les valeurs attributs identifiants comme par exemple des utilisateurs, des hashtags, des mentions d'utilisateurs et permettent d'établir les liens entre les différents systèmes de stockage.

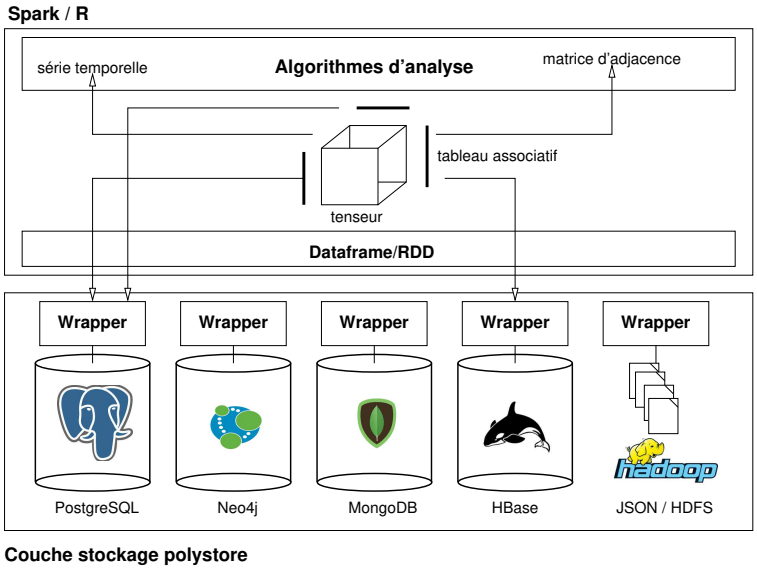


Figure 3. Architecture entrepôt polystore et modèle tensoriel

Les algorithmes d'analyse peuvent travailler directement sur le tenseur en utilisant par exemple une décomposition tensorielle pour extraire des relations cachées ou bien, utiliser le modèle tensoriel comme modèle intermédiaire pour produire par exemple une matrice d'adjacence ou une série temporelle qui seront utilisées par d'autres algorithmes.

3.2. De l'objet mathématique tenseur au modèle de données

Les tenseurs sont des objets mathématiques abstraits très généraux qui peuvent être considérés selon différents points de vue. Les tenseurs peuvent être vus comme

des applications multi-linéaires ou comme le résultat d'un produit tensoriel. Dans un contexte de bases de données, il s'agit, plutôt de tableaux multi-dimensionnels (Baumann, Holsten, 2011 ; Stonebraker *et al.*, 2013) et donc, par analogie avec la définition des matrices, d'une famille d'éléments d'un corps K indexée par N ensembles (tenseur d'ordre N). Plus formellement, un tenseur d'ordre N est un élément du produit tensoriel de N espaces vectoriels chacun ayant son propre système de coordonnées. Un tenseur d'ordre 1 est un vecteur, un tenseur d'ordre 2 est une matrice, et les tenseurs d'ordre supérieur à 3 sont regroupés sous la dénomination de tenseurs d'ordre supérieur. Dans une définition plus pragmatique, un tenseur est un élément de l'ensemble des fonctions du produit de N ensembles $I_j, j = 1, \ldots, N$ vers $\mathbb{R}$: $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, N est le nombre de dimensions, ou ordre, ou encore mode. Dans la suite nous utilisons les lettres capitales en gras dans la police Euler pour désigner un tenseur $\boldsymbol{\mathcal{X}}$, pour les matrices les lettres majuscules droites en gras, les lettres minuscules pour désigner un élément du tenseur par exemple x_{ijk} est le ijk-ième élément du tenseur $\boldsymbol{\mathcal{X}}$ d'ordre 3.

L'objectif est de munir l'objet mathématique tenseur d'opérations de manipulation, d'analyse et de lui adjoindre la notion de schéma afin d'en faire un modèle de données (figure 4).

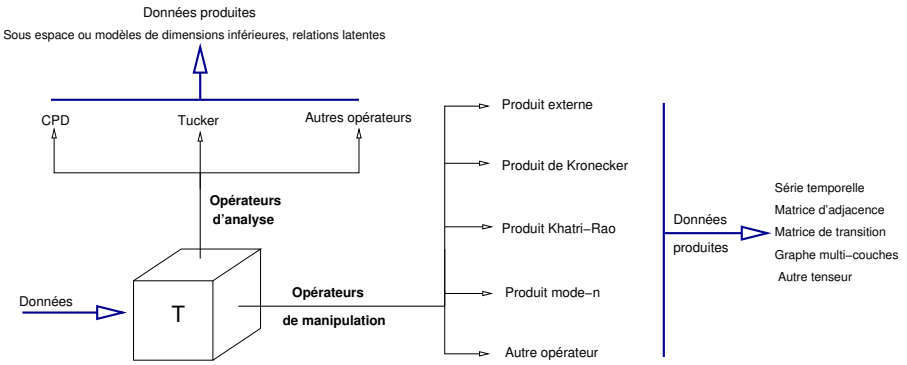


Figure 4. Modèle et opérateurs

Parmi les opérations courantes sur les tenseurs et par analogie avec les opérations sur les matrices et les vecteurs, nous retrouvons des opérateurs de multiplication ainsi que les dépliements et le matriçage (*matricization* ou *matricification*) (Kolda, Bader, 2009). Les produits tensoriels les plus utilisés sont le produit de Kronecker noté $\otimes$, de Khatri-Rao noté $\odot$, d'Hadamard noté $\circledast$, externe noté $\circ$ et mode-n noté $\times_n$.

3.3. Opérateurs de projection

Une fibre est un fragment mono-dimensionnel d'un tenseur analogue à la notion de vecteur, et aux lignes ou colonnes d'une matrice. Ainsi, pour un tenseur d'ordre 3 les fibres peuvent être des colonnes, des lignes ou des tubes notées respectivement : $\boldsymbol{\mathcal{X}}_{:jk}$, $\boldsymbol{\mathcal{X}}_{i:k}$ et $\boldsymbol{\mathcal{X}}_{ij:}$. Une tranche de tenseur est un fragment d'ordre 2. Par exemple

pour un tenseur d'ordre 3 on obtient des tranches horizontales, latérales et frontales notées respectivement : $\boldsymbol{\mathcal{X}}_{i::}$, $\boldsymbol{\mathcal{X}}_{:j:}$ et $\boldsymbol{\mathcal{X}}_{::k}$. Que ce soit pour les fibres ou les tranches il est courant de les nommer en fonction de leur dimension d'origine (mode-1, mode-2 ou mode-3). Tranches ou fibres horizontales sont obtenues par le produit mode-3 $\times_3$, latérales par le produit mode-2 $\times_2$, frontales par le produit mode-1 $\times_1$.

Les opérations de projection peuvent être généralisées par le produit d'Hadamard d'un tenseur d'ordre N avec un tenseur booléen de même ordre qui contient des valeurs 1 pour les éléments à sélectionner : $\boldsymbol{\mathcal{X}} \circledast \boldsymbol{\mathcal{B}}$. Par exemple pour un tenseur d'ordre 3, $\boldsymbol{\mathcal{T}}_1$, représentant les utilisateurs, les hashtags utilisés (leur nombre d'occurrences dans les tweets de l'utilisateur) et le temps, sélectionner tous les hashtags utilisés par un utilisateur i se traduit par un tenseur d'ordre 2 : $\boldsymbol{\mathcal{T}}_2 = \boldsymbol{\mathcal{T}}_1 \circledast \boldsymbol{\mathcal{B}}_1$ avec $\boldsymbol{\mathcal{B}}_{1_{i::}} = 1$. Pour obtenir une série temporelle traduisant l'utilisation d'un hashtag on effectue la somme des colonnes du tenseur d'ordre 2 obtenu.

3.4. Opérateurs de sélection

Les opérateurs de sélection peuvent agir à deux niveaux : 1) sur les valeurs contenues dans le tenseur (équivalent à une sélection portant sur un seul attribut en relationnel) ou bien 2) sur les valeurs des dimensions qui sont dans notre cas les tableaux associatifs notés ta_i pour $i = 1, .., N$. Ces derniers peuvent être considérés comme des relations à deux attributs du modèle relationnel associant un identifiant externe à une valeur d'index. L'opérateur de sélection σ s'écrit avec deux conditions la première portant sur les dimensions, l'autre sur les valeurs : $\sigma[cdt\ dim][cdt\ val]\boldsymbol{\mathcal{X}}$. La composante de l'opérateur de sélection portant sur les dimensions est réalisée par le produit d'Hadamard par tenseur booléen dont les éléments de valeur 1 correspondent aux éléments des ta_i sélectionnés.

Par exemple, pour sélectionner tous les hashtags contenus dans les tweets d'un utilisateur u_1 entre les temps $t = 10$ et $t = 40$ et dont le nombre d'occurrences est supérieur à 25, à partir du tenseur $\boldsymbol{\mathcal{T}}_1$ dont les dimensions sont U, H et T on effectue une sélection : $\sigma[U =' u'_1 \wedge T \geq 10 \wedge T \leq 40][> 25]\boldsymbol{\mathcal{T}}_1$

3.5. Opérateurs analytiques de décomposition

Les décompositions tensorielles telles que CANDECOMP/PARAFAC (CP), Tucker, HOSVD sont utilisées pour effectuer des réductions de dimensions et extraire des relations latentes (Kolda, Bader, 2009). Comme les représentations des données par tenseur sont multiples et leur sémantique non explicite, les résultats des décompositions tensorielles sont complexes à interpréter. Par analogie avec les décompositions matricielles il est possible de déterminer la décomposition associée à un objectif. Par exemple, pour exprimer chaque espace engendré par une des dimensions du tenseur indépendamment des autres mais en fonction de l'espace global on peut utiliser une décomposition CP (figure 5). Pour déterminer des modèles d'utilisateurs en fonction de hashtag ou des motifs récurrents de comportement on préférera utiliser une décom-

position HOSVD. Les modèles produits peuvent alors être utilisés dans des systèmes de recommandation.

4. Retours d'expériences

Le jeu de données sur lequel nous travaillons est un corpus de 50M de tweets collectés dans le cadre du projet pluridisciplinaire TEP2017 dont l'objectif est l'analyse de la communication politique sur Twitter lors de l'élection présidentielle de 2017. Les données brutes au format JSON représentent 720Go, les attributs les plus courants ont été sélectionnés pour constituer une base de données relationnelle non normalisée (50Go) à partir de laquelle une seconde base de données relationnelle en 3FN a été créée autour des entités utilisateur, hashtag, tweet (55Go incluant les index). Dans la suite, nous présentons la modélisation du réseau social Twitter à l'aide du modèle TDM et nous décrirons nos expérimentations.

4.1. Modélisation avec TDM

Le réseau social Twitter est un réseau complexe dans lequel les nœuds sont hétérogènes (utilisateurs, tweets, hashtags, etc.) et les liens également (retweet, émet, follows, cite, etc.). On considère un ensemble d'entités E incluant les utilisateurs et les ressources (tweets, hashtags, URL) sous la forme d'une partition, et un ensemble de relations R entre les entités. Ces deux ensembles sont définis comme suit : $E = \bigcup_{i=1}^{n} E_i$ et $R = \{R_i, i = 1, \ldots, m\}$ avec $R_i : E_k \times E_l \rightarrow \mathbb{N},\ k, l \in \{1, \ldots, n\}$. Les relations R_i sont des applications, elles peuvent être représentées par des matrices. Par exemple, un utilisateur est décrit par m vecteurs, un pour chacune des relations. Les relations n'ont pas la même signature comme par exemple :

– utilisateur/hashtag pour l'utilisation d'un hashtag dans un de ses tweets ;

– utilisateur/utilisateur pour une relation suivre, ou une mention dans un tweet ou encore le relais d'un tweet d'un utilisateur ;

– utilisateur/tweet pour l'émission d'un tweet, ou le retweet, etc.

L'ensemble des utilisateurs pour toutes les relations considérées peuvent être représentés par un tenseur d'ordre 3, si toutes les relations à représenter sont homogènes c'est-à-dire si les applications associées ont la même signature. Par exemple, deux des modes du tenseur seront E_1 et E_k avec $k \in 1, n$ et le troisième représentera les différentes relations R_i. Il peut s'agir par exemple d'un tenseur représentant des relations entre utilisateurs comme les abonnements, les retweets, la citation, etc. De plus une dimension supplémentaire peut être ajoutée pour représenter le temps et permettre une analyse de la dynamique du réseau. Kivelä *et al.* (2014), et De Domenico *et al.* (2013) montrent comment les tenseurs peuvent représenter des relation n-aires, des hypergraphes, des réseaux multi-couches et multi-relationnels sans pour autant les considérer comme un véritable modèle de données.

Nous présentons dans la suite nos expérimentations concernant l'étude de l'impact des robots dans la propagation de tweets supposés viraux. Nous n'insisterons pas sur l'interprétation des résultats faite par les chercheurs en sciences sociales.

4.2. Robots et viralité

La viralité peut être définie par les trois paramètres résumés dans Beauvisage *et al*. (2011) : la concentration temporelle de l'attention sur un contenu, la circulation de ces contenus et les mécanismes de la contagion d'un individu à l'autre. Contrairement aux tweets qui font du *buzz*, le démarrage de leur diffusion est plus lent et celle-ci dure plus longtemps dans le temps. Des métriques d'ordre lexical (présence d'URL et de hashtag, construction du hashtag à partir de plusieurs mots, etc.) et d'ordre contextuel (activité du compte, sa communauté, etc.) sont prises en compte pour déterminer la viralité d'un tweet (Hoang *et al*., 2011 ; Ma *et al*., 2013 ; Weng *et al*., 2014). Leur nombre et leur variété rendent difficile l'interprétation des résultats obtenus.

Nous considérons que dans un premier temps, détecter la viralité potentielle d'un tweet peut être effectué en mesurant sa popularité c'est-à-dire par le nombre de retweets engendrés. Nous sommes partis du corpus global puis nous avons réduit la période d'étude à l'entre-deux tours (du 24 avril 2017 au 7 mai 2017) et calculé le nombre de retweets pour chaque tweet. Un échantillon des tweets les plus populaires c'est-à-dire dans notre cas les plus propagés par retweets a été obtenu en sélectionnant les tweets retweetés au moins 1 000 fois sur la période. Cet échantillon comporte 1 123 tweets dont certains ont été retweetés plus de 20 000 fois [9]. Les requêtes qui ont produit la liste des tweets à étudier sont exprimées dans le langage natif du système de stockage, ici SQL et lancées depuis R sur la base de données PostgreSQL en 3FN. Elles produisent les données en moins d'une minute. Mettre en avant ces tweets nous a permis de réduire le corpus de 50M de tweets à un millier, donnant la possibilité à nos collègues en sciences de la communication de valider nos expérimentations par une étude qualitative. Leur interprétation a révélé que les tweets les plus populaires et ceux qui avaient une période de présence longue contenaient des URLs référençant des vidéos ou des photos.

Afin de comprendre les mécanismes de diffusion des tweets supposés viraux, nous cherchons à étudier la part d'activité liée à des robots ou plutôt des comptes dont le comportement ne ressemble pas à celui d'un humain. D'après Bessi et Ferrara (2016), les robots ont émis 19% des tweets relatifs à l'élection présidentielle américaine de 2016. Différentes méthodes ont été proposées pour détecter les robots (Varol *et al*., 2017), elles agrègent le plus souvent un grand nombre de caractéristiques pour produire un modèle prédictif s'appuyant sur un algorithme d'apprentissage comme les *random-forest*. Une expérience en utilisant l'API OSoMe [10] pour obtenir une proba-

9. La liste des id de tweets est disponible sur GitHub (https://github.com/EricLeclercq/TEE-2017-Virality) pour une reproductibilité des résultats.

10. https://botometer.iuni.iu.edu/

bilité de comportement automatisé (de type robot) nous a conduit à constater que les valeurs des probabilités n'étaient pas assez significatives pour détecter les robots durant la période. Une des hypothèses est qu'il s'agit de comptes hybrides d'utilisateurs assistés par des algorithmes.

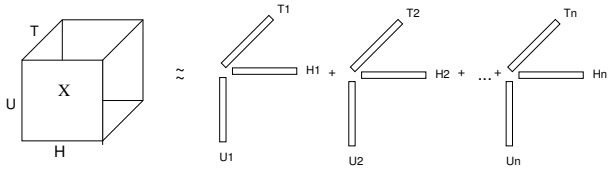


Figure 5. Illustration de la décomposition CP

À partir du comportement des utilisateurs et du contenu des tweets en utilisant les hashtags, nous avons construit un tenseur d'ordre 3 contenant les utilisateurs U ayant retweeté un tweet supposé viral, les hashtags H contenus dans ces tweets et le temps T (14 jours d'observation). Nous avons obtenu un tenseur de dimension $1077 \times 568 \times 336$; les valeurs du tenseur représentent par conséquent le nombre d'occurrence de chaque hashtag par utilisateur et par heure en considérant uniquement les retweets des 1 123 tweets supposés viraux. L'espace de recherche étant très important, nous le réduisons en effectuant une décomposition CP (figure 5) pour étudier l'espace des utilisateurs en fonction de leur comportement. La décomposition CP produit n groupes de trois tenseurs d'ordre 1 (ici les vecteurs U H T). Nous appliquons ensuite l'algorithme de clustering k-means pour identifier des groupes d'utilisateurs. La valeur de n retenue est celle à partir de laquelle il n'y a plus de modification des clusters. Expérimentalement, nous obtenons $n = 8$ et par conséquent un utilisateur est décrit par un point dans un espace à 8 dimensions. L'algorithme k-means appliqué sur ces données permet de déterminer 4 groupes d'utilisateurs : un groupe d'un compte déjà détecté comme un robot, un groupe de deux comptes, un groupe d'une trentaine de comptes comportant plus de la moitié d'utilisateurs ayant une probabilité d'être un robot supérieur à 0.6 (avec OSoMe) et un dernier groupe contenant les autres utilisateurs. Le groupe de deux comptes, se révèle, après étude manuelle, être lié (même comportement et hashtags) et assisté par un algorithme qui retweete des messages contre le candidat Macron. Ces comptes avaient échappé aux autres techniques d'analyse. Les tableaux associatifs et le tenseur sont produits à partir des données en quelques secondes, la décomposition tensorielle en R est effectuée en moins de 5 minutes. Avec Spark nous avons seulement testé la construction du tenseur et n'avons pas noté de différence de performance. Une étude approfondie des performances et du passage à l'échelle sera nécessaire.

5. Conclusion

Dans cet article nous avons proposé une nouvelle architecture pour l'entreposage de données des réseaux sociaux reposant sur un système de stockage *polystore* et un modèle intermédiaire tensoriel. Le modèle de tenseur permet de généraliser les

représentations matricielles (matrice d'adjacence, séries temporelles etc.) ainsi que les graphes dont les multigraphes et les hypergraphes. De plus, le modèle permet de prendre en compte des modélisations de réseaux complexes (réseaux multi-couches, multi-relationnels, etc.). Nous avons aussi présenté quelques opérateurs de manipulation et d'analyse de données sur le modèle de tenseur. Le travail a été expérimenté avec des données issues des réseaux sociaux. L'expérimentation sert de preuve de concept pour valider la faisabilité d'une implémentation d'un entrepôt *polystore* et de l'utilité de l'approche TDM pour des analyses de données de Twitter. Nous nous sommes intéressés à la détection de robots à partir d'un ensemble de tweets supposés viraux. Nos résultats ont été validés par des collègues en sciences de la communication à travers une étude qualitative des résultats. Cette expérimentation a démontré les capacités de modélisation des tenseurs et la pertinence de l'architecture du point de vue de la rapidité de mise en œuvre des transformations de données pour les analyses.

Les perspectives concernent la formalisation des opérateurs ainsi que l'étude des propriétés de la structure algébrique qu'ils engendrent (semi-anneau par exemple). En parallèle, nous souhaitons développer un véritable prototype d'architecture afin de pouvoir étudier l'optimisation des requêtes incluant des opérateurs tensoriels. Néanmoins, les structures de semi-anneaux confèrent aux opérateurs des bonnes propriétés pour une implémentation distribuée ce qui laisse entrevoir un bon potentiel de passage à l'échelle.

Bibliographie

Armatte M. (2005). La notion de modèle dans les sciences sociales: anciennes et nouvelles significations. *Mathématiques et sciences humaines. Mathematics and social sciences*, n° 172.

Atefeh F., Khreich W. (2015). A survey of techniques for event detection in twitter. *Computational Intelligence*, vol. 31, n° 1, p. 132–164.

Baumann P., Holsten S. (2011). A comparative analysis of array models for databases. In *Database theory and application, bio-science and bio-technology*, p. 80–89. Springer.

Beauvisage T., Beuscart J.-S., Couronné T., Mellet K. (2011). Le succès sur Internet repose-t-il sur la contagion ? Une analyse des recherches sur la viralité. *Tracés. Revue de sciences humaines*, n° 21, p. 151–166.

Bessi A., Ferrara E. (2016). Social bots distort the 2016 U.S. Presidential election online discussion. *First Monday*, vol. 21, n° 11.

Bondiombouy C., Kolev B., Levchenko O., Valduriez P. (2015). Integrating big data and relational data with a functional sql-like query language. In *Database and expert systems applications*, p. 170–185.

Bondiombouy C., Valduriez P. (2016). *Query processing in multistore systems: an overview*. Rapport technique. INRIA Sophia Antipolis-Méditerranée.

Bouillot F., Poncelet P., Roche M. (2012, août). How and why exploit tweet's location information? In *AGILE'2012: 15th International Conference on Geographic Information Science*, p. N/A. Avignon, France.

Bringay S., Béchet N., Bouillot F., Poncelet P., Roche M., Teisseire M. (2011). Towards an on-line analysis of tweets processing. In *Database and expert systems applications*, p. 154–161.

Chen C., Yan X., Zhu F., Han J., Philip S. Y. (2008). Graph OLAP: Towards online analytical processing on graphs. In *ICDM*, p. 103–112.

Costa P., Souza F. F., Times V. C., Benevenuto F. (2012). Towards integrating online social networks and business intelligence. In *Iadis international conference on web based communities and social media*, vol. 2012.

De Domenico M., Solé-Ribalta A., Cozzo E., Kivelä M., Moreno Y., Porter M. A. *et al.* (2013). Mathematical formulation of multilayer networks. *Physical Review X*, vol. 3, n° 4, p. 041022.

Drif A., Boukerram A. (2014). Taxonomy and survey of community discovery methods in complex networks. *International Journal of Computer Science and Engineering Survey*, vol. 5, n° 4, p. 1.

Duggan J., Elmore A. J., Stonebraker M., Balazinska M., Howe B., Kepner J. *et al.* (2015). The bigdawg polystore system. *ACM SIGMOD Record*, vol. 44, n° 2, p. 11–16.

Favre C., Jakawat W., Loudcher S. (2017). Graphes enrichis par des Cubes (GreC) : une approche innovante pour l'OLAP sur des réseaux d'information. In *Actes du xxxvème congrès inforsid, toulouse, france, may 30 - june 2, 2017*, p. 293–308.

Franklin M., Halevy A., Maier D. (2005). From databases to dataspaces: a new abstraction for information management. *ACM Sigmod Record*, vol. 34, n° 4, p. 27–33.

Gallinucci E., Golfarelli M., Rizzi S. (2013). Meta-stars: multidimensional modeling for social business intelligence. In *Proceedings of the sixteenth international workshop on data warehousing and olap*, p. 11–18.

Hoang T.-A., Lim E.-P., Achananuparp P., Jiang J., Zhu F. (2011). On modeling virality of twitter content. In *International conference on asian digital libraries*, p. 212–221.

Hölsch J., Schmidt T., Grossniklaus M. (2017). On the performance of analytical and pattern matching graph queries in neo4j and a relational database. In *EDBT/ICDT 2017 Joint Conference: 6th International Workshop on Querying Graph Structured Data (GraphQ).*

Kazienko P., Kukla E., Musial K., Kajdanowicz T., Bródka P., Gaworecki J. (2011). A generic model for a multidimensional temporal social network. In *e-technologies and networks for development*, p. 1–14. Springer.

Kivelä M., Arenas A., Barthelemy M., Gleeson J. P., Moreno Y., Porter M. A. (2014). Multilayer networks. *Journal of Complex Networks*, vol. 2, n° 3, p. 203–271.

Kolda T. G., Bader B. W. (2009). Tensor decompositions and applications. *SIAM review*, vol. 51, n° 3, p. 455–500.

Kolev B., Bondiombouy C., Valduriez P., Jiménez-Peris R., Pau R., Pereira J. (2016). The cloudmdsql multistore system. In *Sigmod.*

Kraiem M. B., Feki J., Khrouf K., Ravat F., Teste O. (2015). Modeling and olaping social media: the case of twitter. *Social Network Analysis and Mining*, vol. 5, n° 1, p. 47.

Leskovec J., Rajaraman A., Ullman J. D. (2014). *Mining of massive datasets*. Cambridge university press.

Loudcher S., Jakawat W., Morales E. P. S., Favre C. (2015, May). Combining OLAP and information networks for bibliographic data analysis: a survey. *Scientometrics*, vol. 103, n° 2, p. 471–487.

Ma Z., Sun A., Cong G. (2013). On predicting the popularity of newly emerging hashtags in twitter. *Journal of the American Society for Information Science and Technology*, vol. 64, n° 7, p. 1399–1410.

Mansmann S. (2008). Extending the OLAP technology to handle non-conventional and complex data (Thèse de doctorat, University of Konstanz). *KOPS*.

Mansmann S., Rehman N. U., Weiler A., Scholl M. H. (2014). Discovering OLAP dimensions in semi-structured data. *Information Systems*, vol. 44, p. 120–133.

Moalla I., Nabli A. (2014). Towards Data Mart Building from Social Network for Opinion Analysis. In *Intelligent data engineering and automated learning - IDEAL 2014 - 15th international conference, salamanca, spain, september 10-12, 2014. proceedings*, p. 295–302.

Moya L. G., Kudama S., Cabo M. J. A., Llavori R. B. (2011). Integrating web feed opinions into a corporate data warehouse. In *Proceedings of the 2nd international workshop on business intelligence and the web*, p. 20–27.

Ong K. W., Papakonstantinou Y., Vernoux R. (2014). *The sql++ unifying semi-structured query language, and an expressiveness benchmark of sql-on-hadoop, nosql and newsql databases*. Rapport technique. UCSD.

Ong K. W., Papakonstantinou Y., Vernoux R. (2015). *The sql++ query language: Configurable, unifying and semi-structured*. Rapport technique. UCSD.

Özsu M. T., Valduriez P. (2011). *Principles of distributed database systems*. Springer Science & Business Media.

Rehman N. U., Mansmann S., Weiler A., Scholl M. H. (2012). Building a data warehouse for twitter stream exploration. In *Proceedings of the 2012 international conference on advances in social networks analysis and mining (asonam 2012)*, p. 1341–1348.

Rehman N. U., Weiler A., Scholl M. H. (2013). Olaping social media: the case of twitter. In *Proceedings of the 2013 ieee/acm international conference on advances in social networks analysis and mining*, p. 1139–1146.

Riquelme F., González-Cantergiani P. (2016). Measuring user influence on twitter: A survey. *Information Processing & Management*, vol. 52, n° 5, p. 949–975.

Stonebraker M., Brown P., Zhang D., Becla J. (2013). Scidb: A database management system for applications with complex analytics. *Computing in Science & Engineering*, vol. 15, n° 3, p. 54–62.

Varol O., Ferrara E., Davis C. A., Menczer F., Flammini A. (2017). Online Human-Bot Interactions: Detection, Estimation, and Characterization. *CoRR*, vol. abs/1703.03107. Consulté sur http://arxiv.org/abs/1703.03107

Weng L., Menczer F., Ahn Y. (2014). Predicting Successful Memes using Network and Community Structure. *CoRR*, vol. abs/1403.6199.

Zhao P., Li X., Xin D., Han J. (2011). Graph cube: on warehousing and olap multidimensional networks. In *Proceedings of the 2011 acm sigmod international conference on management of data*, p. 853–864.