15ème atelier sur la Fouille de Données Complexes (FDC)
Extraction et Gestion des Connaissances (EGC 2018)

Organisateurs :

- Germain Forestier (MIPS, Université de Haute-Alsace)
- Camille Kurtz (LIPADE, Université Paris Descartes)
- Jonathan Weber (MIPS, Université de Haute Alsace)
- Cédric Wemmert (ICube, Université de Strasbourg)

Le groupe de travail "Fouille de Données Complexes"

La quinzième édition de l'atelier sur la fouille de données complexes est organisée par le groupe de travail EGC "Fouille de Données Complexes". Ce groupe de travail rassemble une communauté de chercheurs et d'industriels désireux de partager leurs expériences et problématiques dans le domaine de la fouille de données complexes telles que le sont les données non-structurées (ou faiblement), les données obtenues à partir de plusieurs sources d'information ou plus généralement les données spécifiques à certains domaines d'application et nécessitant un processus d'extraction de connaissance sortant des itinéraires usuels de traitement.

Les activités du groupe de travail s'articulent autour de trois champs d'action progressifs :

- l'organisation de journées scientifiques une fois par an (vers le mois de juin) où sont présentés des travaux en cours ou plus simplement des problématiques ouvertes et pendant lesquelles une large place est faite aux doctorants ;
- l'organisation de l'atelier "Fouille de Données Complexes" associé à la conférence EGC qui offre une tribune d'expression pour des travaux plus avancés et sélectionnés sur la base d'articles scientifiques par un comité de relecture constitué pour l'occasion ;
- la préparation de numéros spéciaux de revue nationale, dans lesquels pourront être publiés les études abouties présentées dans un format long et évaluées plus en profondeur par un comité scientifique.

Contenu scientifique de l'atelier

Nous avons reçu cette année 5 propositions originales. Nous avons été en mesure de proposer pour l'ensemble des propositions deux rapports d'experts afin d'offrir un processus scientifique constructif aux auteurs. Vue la qualité des soumissions et l'intérêt des discussions qu'elles pourraient susciter au sein de l'atelier, nous avons retenu cette année l'ensemble des propositions.

Les articles qui vous sont proposés cette année dans les actes qui suivent explorent une grande variété de complexités, aussi bien dans les données que dans les processus de fouille envisagés.

Remerciements

Les responsables de l'atelier souhaitent remercier vivement toutes les personnes ayant contribué à la tenue de cet atelier. En particulier :

- les auteurs pour la qualité de leurs contributions constituant la base essentielle de discussions fructueuses ;
- les membres du comité de programme et plus généralement tous les relecteurs de cet atelier dont le travail d'évaluation était crucial pour assurer la qualité de l'atelier ;
- les organisateurs d'EGC 2018 qui ont mis en place l'environnement et les moyens pour la réussite des ateliers.

Nous remercions enfin vivement les présidents, Christine Largeron la présidente du comité de programme et Mustapha Lebbah, Hanane Azzag les co-présidents du comité d'organisation d'EGC 2018 ainsi que les organisateurs des ateliers.

Camille Kurtz
LIPADE
Université Paris Descartes

Jonathan Weber & Germain Forestier
MIPS
Université de Haute-Alsace

Cédric Wemmert
ICube
Université de Strasbourg

Membres du comité de lecture

Le Comité de lecture est constitué de :

- Hanane Azzag (LIPN, Univ. Paris 13)
- Nicolas Béchet (IRISA, Univ. Bretagne Sud)
- Alexandre Blansché (Univ. Lorraine)
- Guillaume Cleuziou (LIFO, Univ. Orléans)
- Cécile Favre (ERIC, Univ. Lyon 2)
- Germain Forestier (MIPS, Univ. Haute Alsace)
- Pierre Gançarski (ICube, Univ. Strasbourg)
- Mehdi Kaytoue (INSA Lyon)
- Camille Kurtz (LIPADE, Univ. Paris 5)
- Mustapha Lebbah (LIPN, Univ. Paris 13)
- Arnaud Martin (IRISA, Univ. Rennes 1)
- Florent Masséglia (AxIS-Inria Sophia Antipolis)
- Chedy Raïssi (Loria, INRIA)
- Mathieu Roche (UMR TETIS, CIRAD)
- Cyril De Runz (CReSTIC, Univ. Reims)
- Jonathan Weber (MIPS, Univ. Haute Alsace)
- Cédric Wemmert (ICube, Univ. Strasbourg)

TABLE DES MATIÈRES

Modèle de Sélection de Caractéristiques pour les Données Massives

Zaineb Chelly Dagdia[*,**] Christine Zarges[*]
Gael Beck[***], Mustapha Lebbah[***]

[*]Department of Computer Science, Aberystywth University, United Kingdom
[**]LARODEC, Institut Supérieur de Gestion de Tunis, Tunisia
{zaineb.chelly, c.zarges}@aber.ac.uk,
[***]Computer Science Laboratory (LIPN), University Paris-North - 13, Villetaneuse, France
{beck, mustapha.lebbah}@lipn.univ-paris13.fr

Résumé. La théorie des ensembles approximatifs est une approche pertinente pour la sélection des caractéristiques. Toutefois, cette dernière a un coût computationnel important et une application plus adaptée aux jeux de données de taille limitée. Par conséquent, dans cet article, nous présentons un algorithme distribué et scalable basé sur la théorie des ensembles approximatifs pour le prétraitement des données massives. Nos résultats expérimentaux montrent l'efficacité de notre solution sans perte d'information significative.

1 Introduction

Les données massives (ou Big Data en Anglais) surviennent souvent avec de nombreux défis liés aux prétraitement des données et spécifiquement à la sélection des caractéristiques Labrinidis et Jagadish (2012). Formellement, la tâche de la sélection des caractéristiques vise à déterminer un sous-ensemble minimal de caractéristiques à partir d'une représentation d'origine tout en conservant plus ou moins la même qualité. Cette tâche est éventuellement difficile suite au grand espace de recherche reflétant le grand nombre combinatoire de toutes les combinaisons possibles des caractéristiques Fan et Bifet (2013). Ainsi, afin de palier à cette problématique, un certain degré de réduction de caractéristiques s'avère nécessaire, par conséquent, une méthode efficace de sélection de caractéristiques est requise.

La Théorie des Ensembles Approximatifs (TEA) Pawlak et Skowron (2007); Pawlak (2012) est une approche performante pour la sélection des caractéristiques Thangavel et Pethalakshmi (2009). Cependant, la plupart des algorithmes basés sur cette théorie sont des algorithmes séquentiels, coûteux et ne peuvent traiter que de petits jeux de données. Ceci est expliqué par la nécessité de générer toutes les combinaisons possibles des caractéristiques, les traiter pour finalement sélectionner l'ensemble minimal des caractéristiques les plus pertinentes. Cependant, vu que le nombre des caractéristiques accroit d'une manière exponentielle dans le contexte des données massives cette tâche devient difficile. Cela nous amène à présenter dans cet article un nouvel algorithme basé sur la théorie des ensembles approximatifs pour un prétraitement de données à grande échelle. Notre nouvel algorithme est caractérisé par une implémentation

distribuée basée sur l'écosystème Scala/Apache Spark Shanahan et Dai (2015). Cette méthode est une première contribution qui s'inscrit dans le cadre d'un projet du programme H2020 de la bourse Marie Sklodowska-Curie, accord Numéro 702527.

2 Théorie des Ensembles Approximatifs

La Théorie des Ensembles Approximatifs (TEA) est un formalisme mathématique généralisant la théorie des ensembles classique Pawlak (2012). Cette théorie est fondée sur les notions d'indiscernabilité et d'approximation. Dans ce qui suit, nous présentons les notions essentielles de cette théorie pour la sélection des caractéristiques.

2.1 Système d'information

Le système d'information T permet de représenter les connaissances d'un domaine sous forme de $T = \{U, A\}$ où U correspond aux objets de l'univers et A les caractéristiques qui décrivent ces objets. A peut être divisée en deux sous-ensembles C et D appelés caractéristiques conditionnelles (toutes les caractéristiques) et la caractéristique de décision (la classe). Dans certains systèmes d'information, plusieurs caractéristiques de décision peuvent être présentées.

2.2 La relation d'indiscernabilité

Littéralement le mot indiscernable signifie que certaines choses de même nature peuvent être similaires sans être forcément identiques. D'après la théorie d'approximation, les objets indiscernables (ou similaires) sont caractérisés par les mêmes valeurs de certaines caractéristiques. Autrement dit, pour tout sous ensemble B de l'ensemble des caractéristiques de A, la relation $IND(B)$ est définie par : le couple $(x, y) \in IND(B)$, si et seulement si $a(x) = a(y)$ pour tout élément $a \in A$, avec $a(x)$ qui représente la valeur des caractéristiques a pour l'élément x. $IND(B)$ est une relation d'équivalence.

2.3 Les approximations de la théorie des ensemble approximatifs

Les approximations permettent d'associer à n'importe quel ensemble discernable B une paire d'ensembles - appelés approximation inférieure ($\underline{B}$) et approximation supérieure ($\overline{B}$). Dans l'approximation inférieure, ($\underline{B}$) est une description des objets qui appartiennent certainement à l'ensemble B. Elle est représentée par l'union de toutes les classes d'équivalence qui sont contenues dans l'ensemble cible tel que : $\underline{B} = \bigcup X \in U\{X_i \in U/[X_i]_B \subseteq B\}$. Tandis que l'approximation supérieure ($\overline{B}$) est une description des objets qui peuvent appartenir à l'ensemble B. Elle est représentée par l'ensemble de l'union de toutes les classes d'équivalence qui ont une intersection non vide avec l'ensemble cible tel que : $\overline{B} = \bigcup X \in U\{X_i \in U/[X_i]_B \cap B \neq \emptyset\}$.

2.4 Sélection des caractéristiques

Pour accomplir la tâche de sélection de caractéristiques certains concepts doivent être définis : La TEA définie la région positive $POS_C(D) = \bigcup \overline{C}(X)$ qui est l'ensemble des éléments

de U qui sont certainement classifiés aux partitions $U/IND(D)$ en se basant sur C. La dépendance des caractéristiques qui est défini par $k = \gamma(C, c_i) = \frac{|POS_C(c_i)|}{|U|}$ mesure le degré de dépendance k d'une caractéristique c_i par rapport à un ensemble des caractéristiques C. Sur ces bases, pour la sélection des caractéristiques, l'ensemble $R \subseteq C$ est dit un D-*reduct* de C si $\gamma(C, R) = \gamma(C)$ et il n'ya aucun $R' \subset R$ tel que $\gamma(C, R^{'}) = \gamma(C, R)$. En d'autres termes, le $Reduct$ est l'ensemble minimal de caractéristiques sélectionnées conservant le même degré de dépendance que l'ensemble des caractéristiques. La TEA peut générer un ensemble de reducts, $RED_D^F(C)$ et dans ce cas n'importe quel reduct de $RED_D^F(C)$ peut être choisi pour remplacer le système d'information initial.

3 Solution proposée

Dans cette section et dans le contexte des données massives, nous présentons notre nouvel algorithme distribué nommé "Sp-RST" qui est basé sur la TEA pour la sélection des caractéristiques. Notre algorithme est caractérisé par une implémentation distribuée basée sur l'écosystème Scala/Apache Spark Shanahan et Dai (2015). Le fonctionnement général du Sp-RST est présenté par la Figure 1.

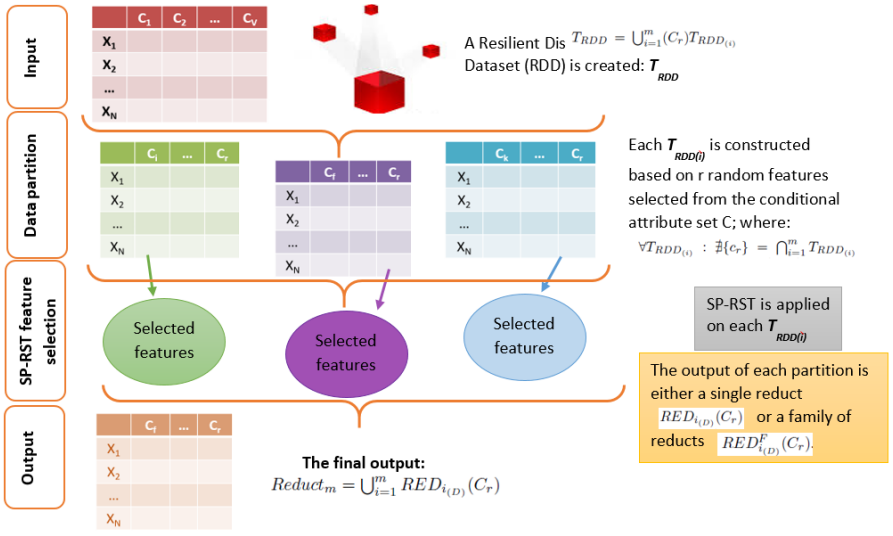

FIG. 1 – *Le fonctionnement général du SP RST.*

3.1 Formalisation générale

Sp-RST défini les données massives représentant le problème d'apprentissage comme un système d'information T_{RDD} où l'univers $U = \{x_1, \ldots, x_N\}$ est l'ensemble des objets, $C = \{c_1, \ldots, c_V\}$ est l'ensemble des caractéristiques conditionnelles parmi lesquelles Sp-RST sélectionne les caractéristiques les plus pertinentes et la caractéristique de décision $D =$

$\{d_1, \ldots, d_W\}$ correspond à la classe. Afin d'assurer la scalabilité de notre algorithme, Sp-RST partage la T_{RDD} en m blocs de données basés sur des partitions de C. Par conséquent, $T_{RDD} = \bigcup_{i=1}^{m}(C_r)T_{RDD_{(i)}}$; où $r \in \{1, \ldots, V\}$. Chaque $T_{RDD_{(i)}}$ est construit en fonction de r caractéristiques aléatoirement sélectionnées à partir de C ; où $\forall T_{RDD_{(i)}} : \nexists\{c_r\} = \bigcap_{i=1}^{m} T_{RDD_{(i)}}$. De ce fait, au lieu d'appliquer Sp-RST (voir Algorithme 1) sur la T_{RDD} comprenant l'ensemble des caractéristiques C, l'algorithme distribué sera appliqué sur chaque $T_{RDD_{(i)}}$. Par conséquent, nous pouvons palier aux limites de la TEA et garantir l'applicabilité de Sp-RST à un nombre de caractéristiques conséquent.

Algorithm 1 Sp-RST

Inputs : T_{RDD} Système d'information ; m nombre de partitions ; N nombre d'itérations
Output : $Reduct$

1: Calculer $IND(D)$
2: **for** each iteration $n \in [1, \ldots, N]$ **do**
3: Générer $T_{RDD_{(i)}}$ en se basant sur les m partitions
4: **for** each $T_{RDD_{(i)}}$ partition, $i \in [1, \ldots, m]$ **do**
5: Générer $AllComb_{(C_r)}$; Calculer $IND(AllComb_{(C_r)})$
6: Calculer $DEP(AllComb_{(C_r)})$; Séléctionner $DEP_{max}(AllComb_{(C_r)})$
7: Filtrer $DEP_{max}(AllComb_{(C_r)})$; Filtrer $NbF_{min}(DEP_{max}(AllComb_{(C_r)}))$
8: **end for**
9: **for** each $T_{RDD_{(i)}}$ output **do**
10: $Reduct_m = \bigcup_{i=1}^{m} RED_{i_{(D)}}(C_r)$
11: **end for**
12: **end for**
13: **return** ($Reduct = \bigcap_{n=1}^{N} Reduct_m$)

Afin de garantir d'avantage la performance de Sp-RST, nous appliquons l'algorithme N fois sur les m blocs de la T_{RDD}. Plus précisément, à travers toutes les itérations, l'algorithme générera d'abord les m $T_{RDD_{(i)}}$, ensuite pour chaque partition, les instructions distribuées de Sp-RST (Algorithme 1, lignes 5 à 7) seront exécutées à part la ligne 1 de l'algorithme. Cette tâche est indépendante des m partitions générées vu qu'elle calcule la relation d'indiscernabilité de la caractéristique de la décision $IND(D)$ et elle est non liée aux caractéristiques conditionnelles. En sortant de la boucle, ligne 9, le résultat de chaque partition est soit un seul reduct $RED_{i_{(D)}}(C_r)$ ou un ensemble de reducts $RED^{F}_{i_{(D)}}(C_r)$. En se basant sur les préliminaires de la TEA, tout reduct de $RED^{F}_{i_{(D)}}(C_r)$ peut être utilisé pour représenter la $T_{RDD_{(i)}}$. Par conséquent, si Sp-RST génère un seul reduct, pour une partition $T_{RDD_{(i)}}$ alors la sortie de cette phase de sélection de caractéristiques est l'ensemble des caractéristiques de $RED_{i_{(D)}}(C_r)$. Ces caractéristiques sont les plus pertinentes parmi l'ensemble C_r et résultant a un nouveau système d'information $T_{RDD_{(i)}}$, $T_{RDD_{(i)}}(RED)$, qui préserve quasiment la même qualité des données que $T_{RDD_{(i)}}(C_r)$ qui est basé sur tout l'ensemble des caractéristiques C_r. Si Sp-RST génère une famille de reducts alors l'algorithme choisit aléatoirement un reduct de $RED^{F}_{i_{(D)}}(C_r)$ pour représenter la $T_{RDD_{(i)}}$. À ce stade, à chaque bloc de données i correspond un ensemble de caractéristiques sélectionnées $RED_{i_{(D)}}(C_r)$. Cependant, puisque chaque $T_{RDD_{(i)}}$ est basé sur des caractéristiques distinctes, une union des caractéristiques sé-

lectionnées est nécessaire pour représenter la T_{RDD} initiale (Algorithme 1, lignes 9 à 11). L'algorithme est itéré N fois générant N $Reduct_m$. Ainsi, à la fin, une intersection de tous les $Reduct_m$ obtenus est nécessaire (Algorithme 1, ligne 13). Dans ce qui suit, nous allons élucider les 7 tâches distribuées de Sp-RST.

3.2 Détails algorithmiques

Tout d'abord, Sp-RST calcule la relation d'indiscernabilité pour la classe $D = \{d_1, \ldots, d_W\}$ (Algorithme 2) ; définie par $IND(D) : IND(d_i)$. Plus précisément, Sp-RST calcule la relation d'indiscernabilité pour chaque classe d_i en rassemblant les mêmes objets de T_{RDD} qui sont définis dans l'univers $U = \{x_1, \ldots, x_N\}$ ayant la même classe d_i. Pour ce faire, Sp-RST exécute l'opération $foldByKey$, où la classe d_i définit la clé et les identifiants des objets de T_{RDD}, id_i de x_i, définissent la valeur. L'ensemble des objets génèrés est retenu vu qu'il représente $IND(D) : IND(d_i)$.

Algorithm 2 Calculer $IND(D)$

Input : T_{RDD}
Output : $IND(D) : Array[Array[x_i]]$
1: $IND(d_i) = data.map\{case(id_i, vector, d_i) => (d_i, ArrayBuffer(id_i))\}$
$.foldByKey(ArrayBuffer.empty[Long])(\_ + + = \_)$
2: $IND(d_i).map\{case(d_i, x_i) => x_i\}.collect$

Puis, pour une partition spécifique, Sp-RST crée toutes les combinaisons possibles des caractéristiques C_r : $AllComb_{(C_r)} = C_r.flatMap(C_r.combinations)\ .drop(1)$ et calcule la relation d'indiscernabilité pour chaque combinaison générée (voir Algorithme 3). Sp-RST vise à regrouper tous les identifiants id_i des objets ayant la même combinaison des caractéristiques extraites $AllComb_{(C_r)}$. Pour ce faire, l'opération $foldByKey$ est utilisée où la combinaison des caractéristiques définie la clé et l'id_i comme valeur.

Algorithm 3 Calculer $IND(AllComb_{(C_r)})$

Inputs : T_{RDD_i}, $AllComb_{(C_r)}$
Output : $IND(AllComb_{(C_r)}) : Array[Array[id_i]]$
1: $IND(AllComb_{(C_r)})$ = $data.map\ \{case(id_i, vector, d_i) => ((AllComb_{(C_r)_i}, vector), ArrayBuffer(id_i))\}.foldByKey(ArrayBuffer.\ empty[Long])(\_ + + = \_)$
2: $IND(AllComb_{(C_r)}).map\{case(ListValues, id_i) => id_i\}.collect$

Dans Algorithme 4, les degrés de dépendances $\gamma(C_r, AllComb_{(C_r)})$ de chaque combinaison de caractéristiques sont calculés. Pour ce faire, les relations d'indiscernabilités calculées $IND(D)$ et $IND(AllComb_{(C_r)})$ ainsi que l'ensemble de toutes les combinaisons de caractéristiques $AllComb_{(C_r)}$ sont requis. La tâche consiste à tester, d'abord, si l'intersection de chaque $IND(d_i)$ avec chaque $IND(AllComb_{(C_r)})$ conserve toutes les caractéristiques de ce dernier (l'approximation inférieure). Si c'est le cas, un score qui est égal au nombre de caractéristiques dans $IND(AllComb_{(C_r)})$ est donné, zéro sinon. Étant donné que cette tâche est effectuée de façon distribuée où chaque machine traite certaines combinaisons

des caractéristiques, une première sommation des scores $IND(d_i)$ est opérée suivie d'une deuxième sommation pour avoir tous les scores $IND(D)$; référant au degré de dépendance $\gamma(C_r, AllComb_{(C_r)})$.

Algorithm 4 Générer $DEP(AllComb_{(C_r)})$

Inputs : $AllComb_{(C_r)}$, $IND(D)$, $IND(AllComb_{(C_r)})$
Outputs : $\gamma(C_r, AllComb_{(C_r)})$, $Size_{(AllComb_{(C_r)})}$

```
1: for i := AllComb_(C_r) do
2:     for j := IND(D) do
3:         for v := IND(AllComb_(C_r)) do
4:             if j.intersect(v).length == v.length then v.length
5:             else 0
6:         Reduce(_ + _)
7:     Reduce(_ + _)
```

Le résultat de cette étape est l'ensemble des degrés de dépendance $\gamma(C_r, AllComb_{(C_r)})$ des combinaisons des caractéristiques $AllComb_{(C_r)}$ et leurs tailles associées $Size_{(AllComb_{(C_r)})}$. À ce stade, Algorithme 5, Sp-RST recherche la valeur de dépendance maximale parmi tous les $\gamma(C_r, AllComb_{(C_r)})$.

Algorithm 5 Séléctionner $DEP_{max}(AllComb_{(C_r)})$

Input : RDD[$AllComb_{(C_r)}$, $Size_{(AllComb_{(C_r)})}$, $\gamma(C_r, AllComb_{(C_r)})$]
Output : $MaxDependency$

```
1: RDD.max()(Ordering[Int].on(_._3))._3
```

La sortie $MaxDependency$ reflète d'une part le degré de dépendance de tout l'ensemble des caractéristiques (C_r) représentant T_{RDD_i} et d'autre part le degré de dépendance de toutes les combinaisons possibles des caractéristiques satisfaisant la contrainte $\gamma(C_r, AllComb_{(C_r)}) = \gamma(C_r)$. $MaxDependency$ est le seuil pour la sélection des caractéristiques.

Algorithm 6 Filtrer $DEP_{max}(AllComb_{(C_r)})$

Inputs : RDD[$AllComb_{(C_r)}$, $Size_{(AllComb_{(C_r)})}$, $\gamma(C_r, AllComb_{(C_r)})$], $MaxDependency$
Output : Filtered-RDD[$AllComb_{(C_r)}$, $Size_{(AllComb_{(C_r)})}$, $\gamma(C_r, AllComb_{(C_r)})$]

```
1: RDD.filter(_._3 == maxDependancy)
```

Ensuite, Sp-RST cherche l'ensemble de toutes les combinaisons ayant les mêmes degrés de dépendance que $MaxDependency$; $\gamma(C_r, AllComb_{(C_r)}) = MaxDependency$ en appliquant une fonction de filtrage ; Algorithme 6. A ce stade, Sp-RST supprime à chaque niveau de calcul les caractéristiques inutiles qui peuvent affecter négativement la performance de tout algorithme d'apprentissage.

Enfin, Algorithme 7, Sp-RST conserve l'ensemble des combinaisons ayant le nombre minimum des caractéristiques, $Size_{(AllComb_{(C_r)})}$, en appliquant une opération de filtrage et en satisfaisant les contraintes de reduct discutées précédemment ; $\gamma(C_r, AllComb_{(C_r)}) = \gamma(C_r)$ où

il n'ya aucun $AllComb^{'}_{(C_r)} \subset AllComb_{(C_r)}$ tel que $\gamma(C_r, AllComb^{'}_{(C_r)}) = \gamma(C_r, AllComb_{(C_r)})$.

Algorithm 7 Filtrer $NbF_{min}(DEP_{max}(AllComb_{(C_r)}))$

Input : RDD[$AllComb_{(C_r)}$, $Size_{(AllComb_{(C_r)})}$, $\gamma(C_r, AllComb_{(C_r)})$]
Output : Filtered-RDD[$AllComb_{(C_r)}$, $Size_{(AllComb_{(C_r)})}$, $\gamma(C_r, AllComb_{(C_r)})$]
1: $minNbF := RDD.min()(Ordering[Int].on(\_.\_2))._2$
2: $RDD.filter(\_.\_2 == minNbF)$

Chaque combinaison satisfaisant cette condition est considérée comme un ensemble minimal de caractéristiques les plus pertinentes décrivant l'ensemble des données de T_{RDD_i}.

3.3 Exemple de trace d'exécution

Dans cette section, nous allons présenter un exemple d'exécution de l'Algorithme 2 pour le calcul de la relation d'indiscernabilité de la classe. Supposant que nous avons le système d'information présenté par la Table 1 ayant comme classe $Flu = \{Yes, No\}$:

Patients	Headache	Muscle-pain	Temperature	Flu
o_1	Yes	Yes	very high	Yes
o_2	Yes	No	high	Yes
o_3	Yes	No	high	No
o_4	No	Yes	normal	No
o_5	No	Yes	high	Yes
o_6	No	Yes	very high	Yes

TAB. 1 – *Système d'information*

En se basant sur le partitionnement des données de MapReduce d'Apache Spark, les deux partitions sont les suivantes (Table 2 et Table 3) :

Patients	Headache	Muscle-pain	Temperature	Flu
o_1	Yes	Yes	very high	Yes
o_2	Yes	No	high	Yes
o_3	Yes	No	high	No

TAB. 2 – *Partition 1*

En appliquant la fonction Map de l'Algorithme 2 sur la première partition (Table 2), nous obtenons le résultat suivant :
— $<$"Yes", $O1>$
— $<$"Yes", $O2>$
— $<$"No", $O3>$
En faisant pareil pour la deuxième partition (Table 3), nous obtenons le résultat suivant :

Patients	Headache	Muscle-pain	Temperature	Flu
o_4	No	Yes	normal	No
o_5	No	Yes	high	Yes
o_6	No	Yes	very high	Yes

TAB. 3 – *Partition 2*

— $<$"No", $O4>$
— $<$"Yes", $O5>$
— $<$"Yes", $O6>$

Enfin, pour avoir la relation d'indiscernabilité de la classe $Flu = \{yes, no\}$ nous appliquons la fonction Reduce et plus précisément la fonction foldByKey pour obtenir le résultat suivant :

— $IND(Flu)$ = ["No", [{O3, O4}], ["Yes", [{O1, 02, O5, O6}]

Une fois l'indiscernabilité est calculée, SP-RST génère toutes les combinaisons possibles par partitions de caractéristiques qu'il a déjà créées. Par exemple :

— Partition X : {Headache}
— Output : {Headache}
— Partition Y : {Muscle-pain, Temperature}
— Output : {{Muscle-pain}, {Temperature}, {Muscle-pain, Temperature}}

Sur chaque élément des deux outputs des deux partitions X et Y, Sp-RST calclule les relations indiscernabilités (Algorithme 3). Pour ce faire, nous procédons comme présenté dans l'exemple de calcul de la relation indiscernabilité de la classe.

4 Expérimentations, Résultats et Analyse

4.1 Données et Implémentation

Pour démontrer la scalabilité de Sp-RST tout en palliant aux limites de la TEA et l'avantage de notre méthode de partitionnement en différents blocs, nous avons choisi le jeu de données Amazon Commerce Reviews Asuncion et Newman (2007). Cette base comprend 1 500 objets décrits à travers 10 000 caractéristiques et 50 classes distinctes. Les objets sont répartis d'une manière équitable sur les différentes classes, i. e., pour chaque classe, il y a 30 objets. Tous les tests ont été exécutés sur Grid5000 avec une configuration d'un processeur dual 8 core Intel Xeon E5-2630v3, 128 GB de mémoire. Nous étudions différents paramètres de Sp-RST qui est implémenté en Scala 2.11/Spark 2.1.1. Nous considérons d'abord les partitions 1000, 1200, 1250, 1330, 1500, 2000 et 2500, et nous les exécutons sur 1, 4, 8, 16 et 32 noeuds ; tous via 10 itérations. Nous avons utilisé Random Forest Prinzie et Van den Poel (2008) (org.apache.spark.mllib.tree.RandomForest) avec maxDepth = 6, numTrees = 300, featureSubjectStrategy = 'all 'et impurity =' gini', comme classifeur d'évaluation. Sur toutes les partitions générées, Random Forest est appliqué sur une taille de 70% comme ensemble d'apprentissage et un ensemble de test de 30%.

Notre objectif est de montrer que notre approche proposée Sp-RST est scalable et bien adaptée aux jeux de données ayant un grand nombre de caractéristiques. Ceci est obtenu sans introduire une perte d'informations significative.

4.2 Résultats et analyse

Pour monter l'efficacité de notre méthode et l'avantage de créer des blocs de caractéristiques, il faut d'abord monter sa scalabilité en terme de "Speed up" qui permet de mesurer le temps d'exécution en terme des noeuds utilisés, et en terme de temps d'exécution. Ce sont deux critères principaux pour l'évaluation de la scalabilité d'un algorithme massivement distribué. Une fois que nous avons montré la scalabilité de Sp-RST, il faut monter que l'algorithme n'engendre pas une perte d'information significative. Pour ce faire, il fallait analyser les résultats de classification de Random Forest sur la base Amazon composée de 10 0000 caractéristiques sans Sp-RST (partition1 : première colonne de Table 4) et avec Sp-RST sur les bases générées par toutes les partitions (le reste des colonnes de Table 4). Ceci est pour analyser également l'effet de cette partition sur les résultats de classification. Comme le Random Forest se base sur un processus aléatoire, il fallait passer par une étude statistique. Suite à 100 itérations, la moyenne, la médiane et l'écart type des résultats d'erreur de classification sont présentés dans la Table 4. En outre, pour étudier plus les résultats de classification entre Random Forest sur l'ensemble de données original et les ensembles réduits produits par Sp-RST nous effectuons le test de Wilcoxon Woolson (2008).

En terme de Speed up, nous gardons la taille de l'ensemble de données constante (où la taille est mesurée par le nombre des caractéristiques dans chaque partition) et nous augmentons le nombre de noeuds. À partir de la figure 2, nous observons que notre méthode a un bon Speed up pour les petites partitions. Plus la taille des données, i. e., le nombre de caractéristiques par partition augmente, plus le Speed up devient plus linéaire. Cela s'explique par le fait que moins de partitions impliquent que chaque partition possède plus de caractéristiques.

Comme discuté précédemment, le temps d'exécution augmente de façon exponentielle en fonction du nombre de caractéristiques et, par conséquent, l'utilisation d'un nombre plus important de noeuds est plus avantageux dans ce cas-là. Nous obtenons un bon Speed up si le nombre de caractéristiques par partition est compris entre 7 et 10 (1000 à 1330 partitions), mais pour 2000 et 2500 partitions le Speed up stagne rapidement.

Cette conclusion est également observée pour les temps d'exécutions. À partir de la figure 3, nous observons que pour quelques partitions, le temps d'exécution diminue rapidement en augmentant le nombre des noeuds, tandis que pour de nombreuses partitions, nous n'observons pratiquement aucune amélioration.

De ce fait, nous observons qu'il existe un compromis entre le nombre de partitions et le nombre de noeuds utilisés. Si quelques noeuds sont disponibles, il est conseillé d'utiliser un plus grand nombre de partitions pour réduire le temps d'exécution alors que le nombre de partitions devient moins important si l'on peut accorder un degré élevé de parallélisassion.

En terme de nombre de caractéristiques sélectionnées (voir Table 4), nous remarquons que le nombre de caractéristiques sélectionnées varie considérablement selon le nombre des partitions utilisées. Il est important de clarifier qu'une comparaison avec la version classique de sélection des caractéristiques basée sur la TEA n'est pas possible vu le nombre exponentiel de combinaisons de caractéristiques généré par la méthode comme expliqué dans l'introduction.

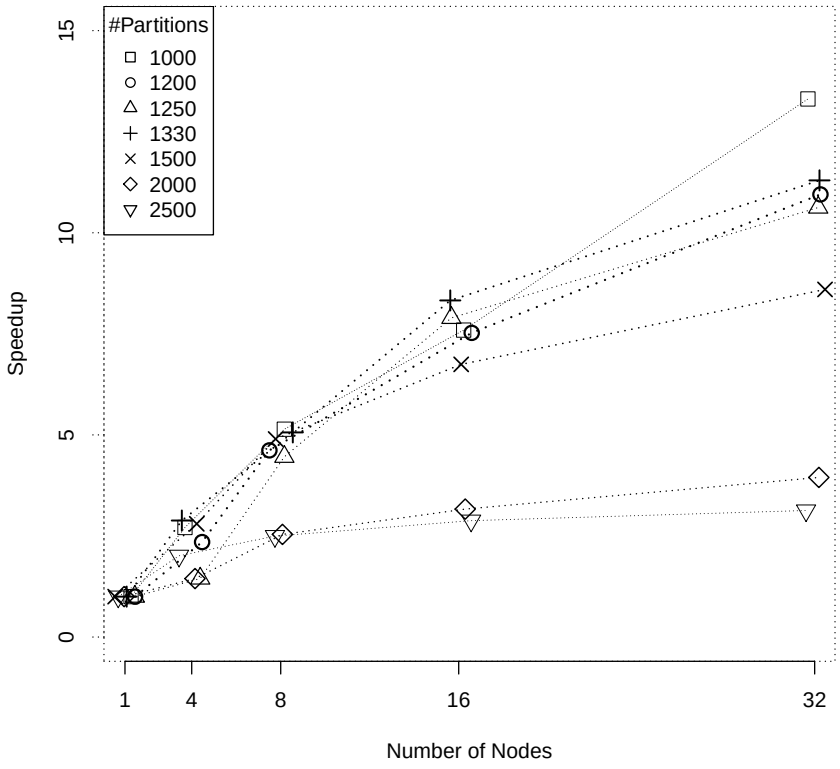

FIG. 2 – *Speedup pour 1 itération.*

Partitions	1	1000	1200	1330	2500
Moyenne	**50.50%**	51.00%	**49.69%**	**47.14%**	**45.16%**
Médiane	**52.53%**	**51.38%**	**50.77%**	**46.59%**	**45.40%**
Ecart Type	11.09%	11.13%	9.98%	11.77%	10.36%
p-valeur	-	0.4897	0.1852	0.009471	0.000485
Caractéristiques séléctionnées	-	6184	3665	2490	3209

TAB. 4 – *Les valeurs statistiques pour l'erreur de classification et le nombre de caractéristiques séléctionnées par partition*

Il est aussi important de clarifier qu'une comparaison avec d'autres méthodes de sélection de caractéristiques fera l'objet d'un travail futur.

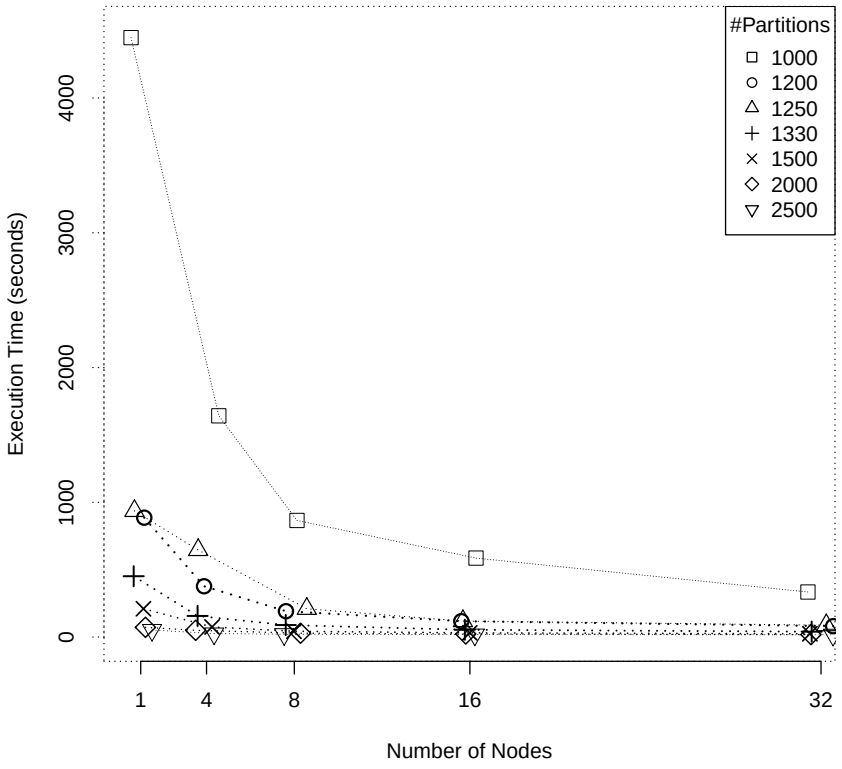

FIG. 3 – *Temps d'exécution pour 1 itération.*

En fonction de ces caractéristiques sélectionnées, nous avons analyser les résultats de classification de Random Forest. En terme d'erreur de classification et en utilisant Random Forest, nous présentons les résultats de 100 exécutions sur la base comportant les 10 000 caractéristiques et les jeux de données dérivés par Sp-RST avec quelques partitions (voir Table 4). Il est important de noter que malgré l'erreur de classification de Randon Forest semble élevée il convient de noter que Amazon contient 50 classes distinctes. Ainsi, une classification naïve bayésienne aurait une erreur de classification de 98%, ce qui est nettement plus élevé que notre approche. La médiane des erreurs de classification de Random Forest sur l'ensemble des données originales (sans Sp-RST) est supérieure à la valeur correspondante pour tous les paramètres de Sp-RST. Compte tenu des moyennes, pour les 1000 partitions il n'y a pas une grande différence par rapport au Random Forest sans Sp-RST. Ceci est aussi prouvé par la "p-valeur" avec un intervalle de confiance de 0.05. Suite à cette comparaison empirique, nous pouvons conclure que Sp-RST ne présente pas de perte d'information significative. Ce qui indique que notre approche est performante.

5 Conclusion

Nous avons présenté un nouvel algorithme distribué basé sur la théorie des ensembles approximatifs pour la sélection des caractéristiques pour les données massives. Nous avons montré que notre algorithme est scalable et capable de sélectionner les caractéristiques sans perte d'information significative. Une analyse de la paramètrisation de Sp-RST représente notre futur travail.

Acknowledgment

This work is part of a project that has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No 702527.

Références

Asuncion, A. et D. Newman (2007). Uci machine learning repository.

Fan, W. et A. Bifet (2013). Mining big data : current status, and forecast to the future. *ACM sIGKDD Explorations Newsletter 14*(2), 1–5.

Labrinidis, A. et H. V. Jagadish (2012). Challenges and opportunities with big data. *Proceedings of the VLDB Endowment 5*(12), 2032–2033.

Pawlak, Z. (2012). *Rough sets : Theoretical aspects of reasoning about data*, Volume 9. Springer Science & Business Media.

Pawlak, Z. et A. Skowron (2007). Rudiments of rough sets. *Information sciences 177*(1), 3–27.

Prinzie, A. et D. Van den Poel (2008). Random forests for multiclass classification : Random multinomial logit. *Expert systems with Applications 34*(3), 1721–1732.

Shanahan, J. G. et L. Dai (2015). Large scale distributed data science using apache spark. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 2323–2324. ACM.

Thangavel, K. et A. Pethalakshmi (2009). Dimensionality reduction based on rough set theory : A review. *Applied Soft Computing 9*(1), 1–12.

Woolson, R. (2008). Wilcoxon signed-rank test. *Wiley encyclopedia of clinical trials*.

Summary

In this paper, we present a novel distributed rough set theory based algorithm for large-scale data pre-processing. Our experimental results show the scalability of our solution and its efficient applicability to Big Data without any significant information loss.

Outlier detection in high-dimensional spaces using one-dimensional neighborhoods

Joris Falip*, Frédéric Blanchard* Michel Herbin*

*CReSTIC, Université de Reims Champagne Ardenne
joris.falip@univ-reims.fr,
http://crestic.univ-reims.fr

Abstract. Detecting outliers in a dataset is a problem with numerous applications in data analysis for fields such as medical care, finance, and banking or network surveillance. But in a majority of use-cases, data points are described by a lot of features, making outlier detection more complicated. As the number of dimensions increases, the notion of proximity becomes less meaningful: this is due to sparser data and elements becoming almost equally distant from each other. Medical datasets add a layer of complexity caused by their heterogeneous nature. Because of these caveats, standard algorithms become less relevant when hundred of dimensions are involved. This paper discusses the benefits of an outlier detection algorithm that uses a simple concept of one-dimensional neighborhood observations to circumvent the problems mentioned previously.

1 Introduction

The outlier detection problem has a lot of practical applications in data science for a variety of fields. Finding outliers can be used to filter them out when preprocessing a dataset, to detect unusual behavior for monitoring purposes or even to correct inaccuracies or typing errors during data collection. When it comes to medical data, outliers can be seen as patients with unique characteristics. These atypical cases are valuable to medical experts, as they represent rarely encountered diseases that could help doctors in treating new patients. Outliers detection algorithms usually rely on a notion of proximity which is calculated using a distance metric such as the Euclidean distance. While this method performs well on low-dimensional spaces, the curse of dimensionality (Donoho et al., 2000) quickly reduces the quality of results obtained using this approach. A high number of dimensions means that elements become almost equally distant from each other (Aggarwal et al., 2001). Moreover, medical data are heterogeneous, thus requiring flexible solutions to fit the information stored in electronic medical records.

This paper, part of an ongoing work on medical datasets (Falip et al., 2017), proposes an algorithmic description and implementation of the rareness concept formulated in a previous article by Herbin et al. (2017). The described outlier detection method performs a computation of the neighborhoods of each element, one dimension at a time, to find outliers. Aggregating such observations results in quantifying how much each data point can be seen as an outlier in the whole dataset, while being less prone to the curse of dimensionality.

2 Neighborhood-based outlier detection

Outliers in the dataset are found using their rareness. Let Ω a set of N elements, defined on D dimensions. Let $Knn^d(n)$ be the set of all K nearest neighbors of element n, on dimension d; and $Rank_e^d(n)$ be the rank of element n according to element e, on a given dimension d. To obtain $Rank_e^d(n)$, element e ranks all the other elements of the dataset, on dimension d, according to their proximity to e. For an element n, the closer to e it is, the lower its rank, with $Rank_e^d(n) = 2$ if n is the nearest neighbor of e.

The rareness of an element n defined by another element e, for a neighborhood size of K, on a dimension d, is the following :

$$Rareness_e^d(n) = \frac{1}{K} \cdot \min(Rank_e^d(n), K) \tag{1}$$

We can compute the rareness of n for a given neighborhood instead of a single element. The rareness for a neighborhood composed of the K nearest-neighbors of n on dimension d is the mean rareness $Rareness_e^d(n)$ for all elements $e \in Knn^d(n)$. It can be written as:

$$Rareness^d_{Knn^{d'}(n)}(n) = \frac{1}{K-1} \cdot \sum_{e \in Knn^{d'}(n)} Rareness_e^d(n) \tag{2}$$

Now that we can compute the rareness of an element for a given dimension and neighborhood, we define the rareness of an element n as the maximum rareness of n, for all dimension of D and for their K-neighborhood on each dimension of D:

$$Rareness(n) = \max_{\substack{d \in D \\ d' \in D}} \{Rareness^d_{Knn^{d'}(n)}(n)\} \tag{3}$$

Algorithm 1 illustrates the pseudo code used to compute $Rareness(n)$, the rareness of an element of the dataset. Algorithm 2 is the pseudo code illustrating $Rareness_B^d(n)$, the rareness of element n, for a dimension d and a neighborhood B.

Data: N elements defined on D dimensions, neighborhood size K
Result: list of outlier elements
$Knn^d(n) \longleftarrow$ *K-nearest-neighbors of n, on dimension d;*
foreach *element n in N* **do**
  foreach *dimension d in D* **do**
    $scores^d(n) \longleftarrow \max_{d' \in D}(\mathbf{Rareness}(n, d, Knn^{d'}(n), K))$;
  end
end
$Result \longleftarrow \{e \in N / \exists d \in D, scores^d(n) = 1\}$;

Algorithm 1: Outlier detection algorithm

Data: element n, dimension d, neighborhood $Knn^{d'}(n)$ *of size* K
Result: rareness of element n for a given dimension d and neighborhood $Knn^{d'}(n)$
$Rank_e^d(n) \longleftarrow$ ranking of n according to its distance from e on dimension d;
foreach *neighbor* e *in* $Knn^{d'}(n)$ **do**
  $scores_e^d(n) \longleftarrow \frac{1}{K} \times \min(Rank_e^d(n), K)$;
end
$Result \longleftarrow \frac{1}{K-1} \times \sum_{e \in Knn^{d'}(n)} score_e^d(n)$;

Algorithm 2: Rareness computing algorithm

To illustrate the algorithm described above, we simulate a simple dataset of 1000 elements described on 6 dimensions. The attributes of the first 998 items are randomly generated following a uniform and independent distribution for all dimensions. We introduce two outliers points, 999 and 1000: the former is outside the distribution on dimension 2 and outside if we consider both dimensions 3 and 4 simultaneously, and it fits the distribution on the three remaining dimensions. The latter point is outside the distribution if we consider dimensions 1 and 2 simultaneously, and dimensions 3 and 4 simultaneously. It fits the distribution of the two remaining dimensions.

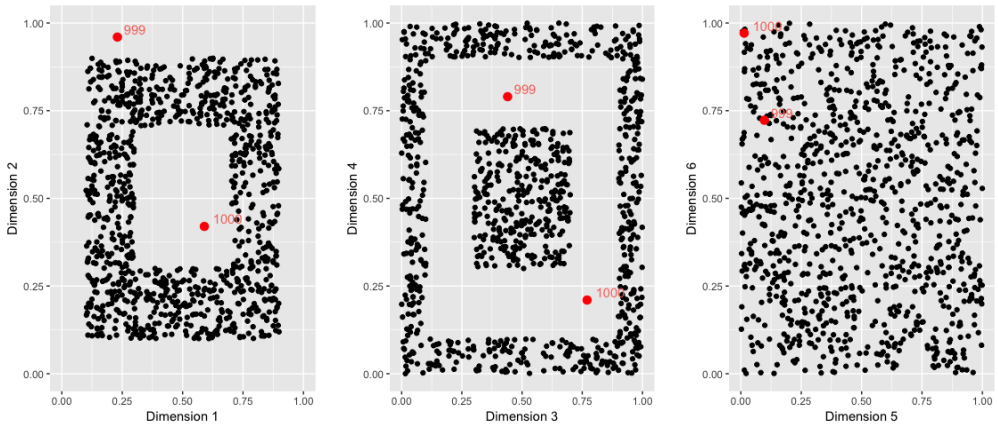

FIG. 1 – *1000 elements defined on 6 dimensions, including 2 outliers.*

Given this dataset, in order to find the outliers, we compute the rareness of each element. To choose the neighborhood size K, we iteratively decrease K from 1000 to 1 and keep the highest value that gives us two outliers. In this example, $K = 160$ is the first value returning two outliers: elements 999 and 1000. A good approximation for K is a quarter of the total number of elements. Using our example, with $K = 250$, the algorithm returns no outliers, but looking at the two items with the highest rareness we find elements 999 and 1000.

3 Conclusion and future works

We expressed the computation of rareness as an algorithm and detailed an example using synthetic data. This neighborhood-based approach as a few advantages such as retaining its meaningfulness even with hundreds of dimensions. To better fit the kind of data stored on each dimension, it allows different ranking methods for the elements, be it quantitative or qualitative. Using this approach, we obtained reliable results for synthetic datasets of up to ten thousand individuals and a thousand dimensions. It was also tested successfully on a real dataset consisting of electronic medical records of a thousand diabetic patients.

There is still a lot of possible leads to further this work. First of all, we are currently comparing this approach with other popular algorithms to adequately measure the relevance of this method. The significance of the results needs to be assessed with synthetic and real datasets and a variety of outlier data (outliers on one or multiple dimensions or with qualitative values). We also plan to thoroughly benchmark our solution and its time and space complexity. The algorithm is easy to parallelize, and the first experiments indicate linear performance gains: datasets up to thousands of individuals and dimensions are processed easily.

References

Aggarwal, C. C., A. Hinneburg, and D. A. Keim (2001). On the surprising behavior of distance metrics in high dimensional space. *International Conference on Database Theory 1973*(Chapter 27), 420–434.

Donoho, D. L. et al. (2000). High-dimensional data analysis: The curses and blessings of dimensionality. *AMS Math Challenges Lecture 1*, 32.

Falip, J., A. A. Younes, F. Blanchard, and M. Herbin (2017). Représentativité, généricité et singularité: augmentation de données pour l'exploration de dossiers médicaux. In *Atelier VIF: Visualisation d'informations, Interaction, et Fouille de données-EGC 2017*.

Herbin, M., D. Gillard, and L. Hussenet (2017). Concept of Observer to Detect Special Cases in a Multidimensional Dataset. *I4CS 717*(2), 34–44.

Résumé

La détection des données atypiques au sein d'un jeu de données est un problème aux applications nombreuses. On le retrouve notamment utilisé pour la détection de cas inhabituels en médecine, de fraudes dans le milieu bancaire, ou d'intrusion dans la sécurité réseau. La plupart de ces usages nécessite toutefois des données décrites par de nombreux attributs. Plus le nombre de dimensions augmente, plus la notion de proximité entre éléments perd de son sens : les données deviennent plus éparses et les éléments quasiment équidistants. De plus, l'hétérogénéité des données médicales les rend complexes à analyser. A cause de ces inconvénients, les algorithmes classiques pour ce problème perdent en efficacité. Ce papier présente un algorithme de détection des données atypiques adapté à la haute dimensionnalité. L'approche choisie repose sur l'analyse de voisinages sur des projections unidimensionnelles.

Analyse Ontologique de scénario dans un contexte Big Data

Marwan Batrouni*, Aurélie Bertaux*
Christophe Nicolle*

*Laboratoire électronique, Informatique et image (Le2i) - FRE CNRS 2005
Bât. I3M - Rue Sully - 21000 Dijon - France
prenom.nom@u-bourgogne.fr,
http://checksem.fr

Résumé. De tout temps, l'anticipation d'une situation future est la principale activité des individus et des organisations. Dans ce contexte, nous nous intéressons à l'identification des séquences potentielles d'évènements par analyse de scénarios, pour une meilleure prédiction des futurs possibles. Dans le contexte de l'analyse de données, la justesse de cette prédiction dépend grandement du volume de données. Cependant dans le domaine du Big Data, (kacfah Emani et al., 2015) montre qu'au delà des critères de volume, de vélocité et de variété, il faut aussi prendre en compte la valeur et la véracité des données. Ces deux aspects sont traités par l'utilisation d'ontologies. L'apport sémantique permet de connaître la valeur des données par rapport à leur exploitation et la consistance, la complétude et la décidabilité, fournit par les ontologies, garantissent leur véracité par rapport au contexte. Cet article propose la définition universelle de scénario et une approche d'analyse de scénarios dans un contexte Big Data, basée sur les ontologies. Cette approche permet la construction de prédictions intégrant les notions de valeur et de véracité.

1 Introduction

Aujourd'hui comme il y a plusieurs milliers d'années, les décideurs et les planificateurs stratégiques sont plus que jamais confrontés à la compréhension d'un environnement de plus en plus complexe (Riabacke, 2012) . L'analyse de scénarios est l'un des principaux outils de planification stratégique. (Steiner, 1997) définit la planification stratégique comme «l'identification systématique des opportunités et des menaces futures, qui, combinées à d'autres données pertinentes, servent de base aux entreprises pour prendre de meilleures décisions pour exploiter les opportunités et éviter les menaces". Cette identification systématique implique une analyse de différents scénarios couvrant l'ensemble des possibilités. (Mintzberg, 2013) distingue la réflexion stratégique et la planification stratégique. La première est fluide et analytique, la seconde est rigide et descendante. Formuler ainsi un plan suffisamment large pour couvrir chaque scénario est un exercice futile. Au lieu de cela, il préconise que les planificateurs stratégiques doivent fournir les analyses formelles ou les données concrètes nécessaires à la réflexion stratégique. Une telle approche de l'analyse formelle et des faits devient de plus en plus accessible aujourd'hui avec l'avènement de l'analyse Big Data ou l'accès aux données

et à la connaissance semble plus simple. Néanmoins, l'analyse de scénario est-elle toujours pertinente dans ce contexte ? Et si oui, comment peut-elle tirer parti des méthodes et des outils de Big Data ? Dans cet article, nous proposons des éléments de réponse et une approche d'intégration entre l'analyse de scénarios et le Big Data basée sur la sémantique des ontoloqies.
Cet article présente les interactions de ces domaines, l'apport de l'ontologie puis la formalisation d'une définition universelle de scénario.

2 Scénario et analyse de scénario

Le mot scénario est utilisé dans de nombreux contextes, par exemple le cinéma ou encore l'ingénierie logicielle. Il est originaire de l'italien et signifie *esquisse de l'intrigue d'une pièce de théâtre* [1], qui souligne une série d'actions et d'événements.
En ingénierie logicielle, (Hsia et al., 1994) propose une définition formelle en utilisant l'approche d'une séquence d'événements dans une machine d'état.
Von Clausewitz et Von Moltke, deux stratèges militaires prussiens du 19e siècle, ont appliqué l'utilisation moderne des scénarios dans un contexte de planification stratégique (Bradfield, 2005). (Martelli, 2014) attribue la naissance de l'analyse de scénario aux insuffisances de la prévision linéaire, souvent liée à l'échec fréquent de la simple extrapolation du passé. Il retrace l'origine moderne de l'analyse des scénarios aux travaux du philosophe français Gaston Berger datant des années soixante. A la même époque Herman Khan de la Rand Corporation (Kosow et Gabner, 2008; Muhammad Amer, 2012) a également été le pionnier de l'utilisation de l'analyse de scénarios. Dans les années soixante, l'analyse de scénarios est devenue partie intégrante des méthodologies de planification stratégique. Des entreprises telles que General Electric et Royal Dutch Shell ont commencé à utiliser l'analyse des scénarios comme outil principal de planification stratégique (Kosow et Gabner, 2008). En parallèle, (Kahn Herman, 1968) définit les scénarios comme "des séquences hypothétiques d'événements construites dans le but de focaliser l'attention sur les processus causaux et les points de décision." La notion de séquence est majeure et s'est poursuivie jusque dans les années 80, avec l'incorporation du concept de *cohérence* (Porter, 1998).
Un ensemble est dit cohérent "s'il n'implique pas une paire de formules opposées contradictoires» [2]. Cette notion de cohérence des scénarios est cruciale dans le cadre de l'analyse des scénarios. Dans les années 1990, l'importance des incertitudes est introduite par (Schwartz, 1996). L'inclusion de l'incertitude dans la définition traduit l'accent mis sur les approches probabilistes adoptées dans certaines méthodologies d'analyse de scénarios utilisées depuis les années soixante. Une décennie plus tard, l'accent est mis sur la notion conceptuelle (Greeuw et al., 2000) qui accentue l'aspect de modélisation de l'exercice de création de scénarios. L'évolution de la définition s'est poursuivie avec (Ringland, 2014) : "cette partie de la planification stratégique qui se rapporte aux outils et technologies pour gérer les incertitudes de l'avenir."
Les approches quantitatives ont été utilisées dès les premiers jours de l'analyse des scénarios, leur utilisation a connu un certain déclin jusqu'à une recrudescence au cours des années deux milles. Cette résurgence peut s'expliquer par l'avènement d'outils de modélisation et d'analyse

1. http ://www.etymonline.com/index.phpterm=scenario&allowed_in_frame=0
2. Stanford Encyclopedia of Philosophy, *Classical Logic* (2013), https ://plato.stanford.edu/entries/logic-classical/#4

performant et évolutif développés dans le cadre d'Internet et de la révolution des données de la fin des années 1990 et début 2000.

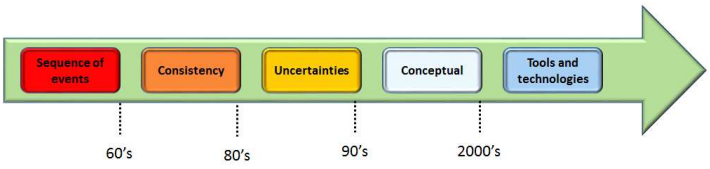

FIG. 1 – *Résumé de l'évolution de l'analyse de scénario dans le temps.*

La tendance (Fig. 1) semble aller dans le sens d'un besoin accru d'outils et de techniques quantitatives (Martelli, 2014). Ces outils donnent de meilleurs résultats lorsqu'ils utilisent des systèmes formels.

Cette formalisation a débuté par la définition d'une taxonomie pour différents types de scénarios. Plusieurs auteurs ont proposé une catégorisation de haut niveau pour les scénarios (Steinmüller, 1997; Greeuw et al., 2000), qui divise les scénarios en deux grandes familles :

— *Explorative* : Explore les développements futurs possibles avec le présent comme point de départ en se demandant : Sachant ce que nous savons, que va-t-il arriver ?
— *Normative* : Identifie les futurs souhaitables ou étudie comment arriver aux conditions futures, en se demandant : Comment pouvons-nous arriver à un avenir souhaitable ? Et comment nos actions / décisions actuelles façonnent-elles le futur ?

Sans une évaluation rigoureuse et des méthodologies analytiques, les scénarios ont une valeur limitée en tant qu'outils conceptuels autonomes. Pour tirer pleinement parti de leur potentiel, ils doivent être traités dans un cadre qui permet la synthèse de diverses sources comme les données et le large spectre des opinions d'experts. Dans ce contexte, l'essor du Big Data et de son écosystème d'outils constitue une opportunité sérieuse pour améliorer les méthodes de l'analyse de scénarios

3 Analyse de scénarios dans un environnement Big Data

Le terme "Big Data" est un label pour tout un ensemble d'outils et de technologies dédiées à l'extraction, le stockage et l'analyse de données massives. Cet ensemble évolue rapidement et en continu. Il est défini par (Laney, 2001) à travers les 3V que sont "Volume", "Vitesse" et "Variété". Cette définition implique que la taille des données dépasse les capactiés de stockage des systèmes classiques, que le cycle de vie des données est très rapide et nécessite une analyse tout aussi rapide et que les données sont par nature hétérogènes, issues de sources de données hétérogènes (Tsai et al., 2015).

Deux articles publiés par le centre de recherche de Google en 2003 (système de fichiers Google) (Ghemawat et al., 2003) et 2004 MapReduce (Dean et Ghemawat, 2004) ont suscité l'enthousiasme pour le sujet et ont inauguré l'ère du Big Data. La création de Yahoo ! ©et le développement de la plate-forme Hadoop en 2006 par Doug Cutting (devenu un projet open source Apache [3]), permettait le traitement de grandes quantités de données utilisant l'architec-

3. Apache Hadoop,http ://hadoop.apache.org/

ture MapReduce. Depuis lors, une multitude d'outils et de technologies continue d'émerger à un rythme vertigineux. Ces technologies alimentent de plus en plus le domaine de l'intelligence artificielle qui imprègne nos vies, de la détection des fraudes financières aux voitures autonomes. Dans ce contexte, l'analyse de scénarios peut-elle exploiter la puissance de Big Data pour améliorer son efficacité ?

3.1 Analyses predictive et prescriptive

Pour répondre à cette question, deux technologies dans la sphère Big Data sont particulièrement intéressantes, à savoir l'analyse prédictive et l'analyse prescriptive. Ces deux analyses sont les composantes d'une pile en trois étapes, commençant par un niveau **descriptif** qui représente un instantané, *ce qui s'est passé*, le niveau **prédictif** représente une projection dans le temps, *ce qui se passerait* ; Enfin, le **prescriptif** cherche à déterminer le meilleur résultat parmi les différents choix, étant donné un ensemble de paramètres connus : *ce qui devrait arriver*. Voici un rapide examen de la pile d'analyse.

- ***Analyse descriptive*** : Est définie par [4] comme l'examen des données ou du contenu, généralement effectuée manuellement, pour répondre à la question "Que s'est-il passé ?" (ou qu'est-ce qui se passe ?). Dans le contexte de Big Data, des outils tels que les outils ETL (Extract, Transform, Load), les outils MapReduce, les lacs de données et les algorithmes de clustering, pour n'en nommer que quelques-uns, constituent les éléments de l'analyse descriptive.
- ***Analyse prédictive :*** Est un ensemble de technologies et de méthodes qui utilisent les données actuelles et historiques pour prévoir les événements futurs [5]. Les données et l'exploration de texte sont des champs très vastes et constituent le noyau de l'analyse prédictive. L'une des normes les plus répandues pour l'exploration de données s'appelle CRISP-DM (Cross Industry Standard Process for Data Mining) définit un processus en six étapes : Compréhension de l'entreprise, Compréhension des données, Préparation des données, Modélisation, Evaluation, Déploiement.
- ***Analyse prescriptive :*** Dans le contexte de Big Data, l'analyse prescriptive est un domaine récent, c'est pourquoi elle est actuellement en train de changer et de fixer ses limites et ses méthodes. C'est une synthèse de plusieurs sciences et méthodologies telles que le machine learning, la recherche opérationnelle, l'analyse statistique (Lustig et al., 2010), et la modélisation et la simulation.

Le Big Data fournit des outils capables d'améliorer l'efficacité de l'analyse de scénarios. Cependant il convient de s'assurer de l'obtention de prédictions à la fois crédibles et pertinentes.

3.2 Insuffisance des 3V

Dans le domaine du Big Data, (kacfah Emani et al., 2015) montre qu'au delà des critères de volume, de vélocité et de variabilité, il faut aussi prendre en compte la *valeur* et la *véracité* des données. Ce sont ces valeurs qui nous assurent la crédibilité et la pertinence nécessaires avant toute analyse de scénarios. Ces deux aspects sont traités par l'utilisation d'une approche de représentation de la connaissance à base d'ontologies. L'apport sémantique permet de connaître

4. http ://www.gartner.com/it-glossary/descriptive-analytics/
5. http ://www.gartner.com/it-glossary/predictive-analytics

la valeur des données par rapport à leur exploitation et la consistance, la complétude et la décidabilité, fournit par les ontologies, garantissent leur véracité par rapport au contexte.

4 Les ontologies porteuses de sens : consistantes, complètes et décidables

(R. Studer, 1998) définit les ontologies par "une spécification formelle et explicite d'une conceptualisation partagée". Elles sont la pierre angulaire de la modélisation orientée sémantique. Dans le Web sémantique, Cette spécification s'exprime sous la forme de d'un ensemble de triplets OWL (Web Ontology Language), standard créé et maintenu par le W3C [6]. La version actuelle est OWL 2.

A la base des ontologies se trouve la logique de description (DL) qui définit les règles et les axiomes qui permettent à l'ontologie de passer d'un simple exercice taxonomique à une véritable base de connaissances, capable d'exécuter des inférences.
Cette capacité à utiliser des règles pour faire des inférences, découvrir de nouvelles connaissances est caractéristique des ontologies, contrairement à d'autres langages de modélisation tels que UML (Krotzsch et al., 2013).

Le mécanisme par lequel nous arrivons à cette conclusion s'appelle une fonction d'interprétation ou (I) qui est le mécanisme par lequel DL mappe (ou *interprète*) la syntaxe d'un constructeur telle que les axiomes en un sens compréhensible. C'est l'essence de l'interprétation sémantique.

Dans le contexte d'OWL, la composante de la logique de description se présente sous la forme de quelques variétés qui ont des implications importantes concernant la complexité et la décidabilité de la version OWL spécifique : OWL DL 1.0, OWL DL 2.0 et OWL Lite (en langage respectif $\mathcal{SHOIN}(\mathcal{D})$, $\mathcal{SROIQ}(\mathcal{D})$ et $\mathcal{SHIF}(\mathcal{D})$) sont décidables, contrairement à OWL full (en logique de premier ordre (FOL)).

Une ontologie décrit une situation particulière dans un domaine de discours, cependant une ontologie est limitée au sens où elle ne peut pas tout spécifier. Les langages de description (DL) ont été conçus pour gérer ces informations incomplètes. Plutôt que de faire des hypothèses par défaut (afin de spécifier une interprétation particulière) pour chaque ontologie, la sémantique considère généralement toutes les situations possibles (c'est-à-dire les états du monde) où les axiomes sont satisfaits. Ceci est parfois appelé l'Open World Assumption (OWA), puisqu'il garde les informations non spécifiées ouvertes (Krotzsch et al., 2013). Cependant, dans un monde idéal, un modèle capturerait toutes les informations ou axiomes pertinents sur un système, un tel état de choses conduirait à ce qu'on appelle une CWA (Closed World Assumption), dans un tel modèle si une information n'est pas déduite, ou trouvée, elle est alors supposée fausse, comme c'est le cas pour les systèmes de bases de données par exemple.

Pour résumer, les ontologies sont la meilleure approche pour démarrer l'exercice de modélisation pour la construction de scénarios grâce l'interprétation sémantique répondant à la contrainte de valeur du Big Data et grâce à 3 éléments assurant la véracité du système :

— fournissent les axiomes de logique de description (DL) qui garantissent la cohérence (la consistance) ;

6. Semantic web (2013) https ://www.w3.org/2001/sw/

— permettent l'inférence et la découverte de nouvelles connaissances à partir d'un existant, rendant le tout plus grand que la somme de ses parties (la décidabilité) ;
— fonctionnent dans Open World Assumption (OWA), ce qui leur confère l'avantage d'intégrer des connaissances nouvelles et hétérogènes (la complétude).

5 Formalisation

Exploiter le Big Data pour aider à améliorer le domaine de l'analyse des scénarios sera une évolution importante, mais un tel lien peut être fait beaucoup plus efficacement s'il existe de meilleures bases formelles pour l'analyse de scénario. Une telle compréhension formelle et théorique assied ces bases pour un résultat plus solide et plus cohérent.

5.1 Ontologies et analyse de scénario

L'idée principale qui nous conduit à établir un lien entre ontologie et analyse de scénario (i.e. un système est l'ensemble de toutes les configurations, i.e. l'attribution unique de valeurs à toutes les variables observable) est contenue dans deux définitions :

— Une conceptualisation selon (Genesereth et Nilsson, 1987), est un tuple (D, R) où D est un ensemble appelé l'univers du discours et R est un ensemble de relations sur D. Notez que, dans la définition ci-dessus, les membres de l'ensemble R sont des relations mathématiques ordinaires sur D, i.e. des ensembles de tuples ordonnés d'éléments de D. Ainsi, chaque élément de R est une relation d'extension, reflétant état spécifique du monde impliquant les éléments de D.
— En ce qui concerne un système S spécifique que nous voulons modéliser, un état du monde pour S est un état observable maximal, c'est-à-dire une attribution unique de valeurs à toutes les variables observables qui caractérisent le système. Un monde (**world**) est un ensemble d'états du monde (world states) totalement ordonné, correspondant à l'évolution du système dans le temps.

Par ailleurs, le domaine du discours, introduit le concept de cohérence, qui joue un rôle central dans l'analyse de scénarios. En effet, elle est la clé de l'interprétation par le mapping d'une collection de concepts abstraits et de symboles à des phrases douées de sens et de sens contexte. Une interprétation sémantique est cohérente quand elle satisfait les axiomes d'une ontologie, elle est alors appelée *modèle* de l'ontologie.

5.2 Fondations pour une définition formelle des scénarios

Notre objectif de formalisation des scénarios dans le contexte de leur analyse nous a amenés à exploiter un langage de modélisation formel et bien défini, à savoir les ontologies basées sur DL, cette modélisation peut être considérée comme l'étape primitive avant le processus de création du scénario. Avant d'aller plus loin dans la mise à profit des ontologies pour formuler notre définition, nous devons d'abord définir quelques éléments clés dans notre discussion, nous devons également préciser une hypothèse clé. Au cours de notre analyse, nous supposons que le système étudié est de nature discrète ; la raison en est que, puisque la création d'un scénario est un exercice conceptuel, il s'agit d'un exercice de modélisation et de simulation d'autres systèmes, dans le contexte de l'analyse de scénario qui serait typiquement la réalité.

Parce qu'un tel exercice peut devenir extrêmement complexe, une approche computationnelle / non déterministe plutôt qu'analytique (comme celle de (Banks et al., 2004)) est requise. Par conséquent, il est nécessaire d'établir des définitions et une terminologie concernant d'une part le domaine du discours (D) et l'espace-état.

5.3 Revisite de concepts fondamentaux

Le domaine du discours *D* et de l'espace-état sont des blocs de construction essentiels qui nous permettent de construire le système dynamique à la base du concept *World*.

5.3.1 Domaine du discours D :

Le domaine du discours nécessite une attention particulière en ce qui concerne son introduction dans la discussion, la question centrale est de savoir comment ce domaine du discours D est créé. Nous devons d'abord définir certaines définitions préliminaires.

Définition 5.1. Classe (Jech, 1997) : Une *classe* est une collection d'ensembles dont les membres satisfont une formule : Si $\phi(x, p_1, p_2..., p_n)$ est une formule, on appelle C = $\{x$: $\phi(x, p_1, p_2, ..., p_n)$ une classe. Les membres de la classe C sont tous les ensembles x qui satisfont $\phi(x, p_1, p_2..., p_n) : x \in C$ ssi $\phi(x, p_1, p_2, ..., p_n)$

Définition 5.2. Instance (au sens de la logique du premier ordre) (Hardegree, 1999) : Une instance de substitution d'une forme propositionnelle F est une proposition P, qui est le résultat du remplacement de variables propositionnelles dans F par des propositions (simples ou complexes) où toutes les variables de même forme sont remplacées par la même proposition.

Ici, la substitution des valeurs d'une classe par des valeurs acceptables se traduit par un paradigme similaire au calcul orienté objet pour créer une instance d'une classe.

Définition 5.3. Domaine du discours : l'ensemble de toutes les instances valides d'un ensemble de classes dans un système.

5.3.2 Espace-état

L'espace-état est plus complexe que le domaine du discours, par analogie si le domaine du discours était l'ensemble de toutes les pièces de Lego disponibles, alors l'espace-état est l'ensemble de toutes les combinaisons légales possibles de ces blocs de construction.

Définition 5.4. Espace-Etat (Cassandras et Lafortune, 2009) : l'état-espace d'un système, généralement désigné par X, est l'ensemble de toutes les valeurs possibles que l'état peut prendre.

Par conséquent, X est le résultat de tous les éléments valides du produit cartésien de l'ensemble des instances de toutes les classes de notre domaine de discours :

- Soit S le système à l'étude ;
- Soit C l'ensemble de toutes les classes de S ;
- Soit I un ensemble d'indexation ;
- Soit D notre domaine de discours où $\langle Di \rangle_i \in I$ une famille indexée d'ensembles pour toutes les instances de C.

Un espace-état X est l'ensemble des n-tuples $(d_0, d_1, ..., d_n)$ résultant du produit cartésien de tous les éléments de D.

$$X = \prod_{i \in I} D_i = \left\{ f : \quad \left(f : I \to \bigcup_{i \in I} D_i \right) \wedge (\forall_{i \in I} : (f(i) \in D_i) \wedge \phi(\{f(i)\}) \right\}$$

Où ϕ est le prédicat assurant l'intégrité de l'opération.

5.3.3 Une redéfinition de *World* :

Selon la définition 2 de (Nicola Guarino, 2009), un *world* (avec un w minuscule) est un ensemble totalement ordonné de *world states*, correspondant à l'évolution du système dans le temps. Pour illustrer cette définition par un exemple, considérons X comme notre espace d'état et $T = T_0, T_1, ..., T_n$ une séquence ordonnée représentant un incrément de temps égal à $T_i - T_{i-1}$.

Par conséquent, un *world* est présenté comme une suite d'états. Le *World* (avec W majuscule) est l'ensemble de tous les mondes possibles, cela se traduit par toutes les séquences ordonnées possibles ou l'espace d'état X.

Cependant, cette définition nécessite des règles qui empêcheraient les incohérences logiques.

En examinant cette définition, nous trouvons que la description du *world*, i.e. une séquence d'états sélectionnés à partir de notre espace d'états ; on peut alors résumer la liste des éléments nécessaires pour construire un *World* (i.e. l'ensemble des mondes possibles) comme :

- Un espace-état ;
- Au moins un état de démarrage désigné ;
- Un ensemble de règles pour *world* pour assurer l'intégrité logique de la transition entre états dans un monde donné ;
- Au moins un état final.

5.3.4 Machine d'états

Cette liste d'exigences coïncide avec les exigences pour une machine d'états. Une machine à états finis [7] est formellement définie comme suit :

Définition 5.5. Machine d'états : une machine d'états M est un 5-uplet, $(Q, \Sigma, \delta, q_0, F)$:

1. Q est un ensemble appelé l'ensemble des états (c'est-à-dire Espace-état) ;
2. Σ est un ensemble appelé *alphabet* ;
3. $\delta : Q \times \sigma \to Q$ est la fonction de transition ;
4. $q_0 \in Q$ est l'état de initial ;
5. $F \subseteq Q$ est l'ensemble des états d'acceptation (ou finaux).

Les transitions d'un état à un autre sont capturées dans une table de transitions d'états qui code les conditions de transition entre les états. C'est typiquement une matrice bidimensionnelle. Une machine d'état plus généralisée est de nature probabiliste appelée automate probabiliste (Stoelinga, 2003), la différence avec une machine d'états non-probabiliste est l'addition des distributions de probabilités pour chaque transition possible d'un état à l'autre,

7. Introduction to Electrical Engineering and Computer Science (2011) MIT OpenCourseWare, http ://ocw.mit.edu

ainsi que l'ajout à la définition d'un sixième tuple représentant un vecteur stochastique $\overrightarrow{P}$ avec les probabilités de la machine d'état se trouvant dans un état initial donné.

Une façon de transformer une ontologie en un système dynamique est appelé *dérivation des relations ontologiques* (Matheus et al., 2003), le processus implique une dérivation partielle des relations dans un graphe d'ontologie sur les événements dans un graphe de machine d'états.
D'un point de vue conceptuel, il s'agit de ce qu'on appelle la *structure du premier ordre extensionnel* tel que défini dans (Nicola Guarino, 2009), dans le contexte de la dérivation ontologique et se traduit par l'énumération des variations des valeurs pour chaque variable liée à un individu impliqué dans la relation d'un temps de T_0 à T_1.
Une telle dérivation implique que les changements d'un état à un autre sont induits par les changements dans les valeurs constitutives des variables dans un état de temps T_0 à T_1.
Par conséquent, le concept déterministe de l'ontologique défini dans (Nicola Guarino, 2009) tient encore dans une ontologie dérivée.
Dans cette interprétation, le domaine du discours et la machine d'états résultante mappent un *world* à une séquence possible d'états $w \equiv \{q_0, q_1..., q_n\}$, dans le langage de l'automate. Par conséquence, la machine d'état est le producteur potentiel de tous les *World* possibles.
Étant donné la connexion conceptuelle des scénarios au concept *world*, l'interprétation des machines d'état peut-elle traduire des scénarios aussi efficacement ? En examinant certaines des définitions de scénarios existantes telles que (Kahn Herman, 1968), nous constatons un accent sur la nature des scénarios en tant que séquence d'événements ou d'états, quoique de nature conceptuelle. Dans ce contexte, un scénario est la version conceptuelle d'un world en tant que tel serait un résultat hypothétique parmi les nombreux potentiels produits par l'automate.
Cette réalisation nous fournit un premier pas dans la formulation de notre définition décrite dans la section suivante.

5.4 Définition universelle de scénario

La section précédente a conclu en présentant le concept world / scénario comme une séquence d'états produits à partir d'une machine d'états.
Dans cette représentation, un *World* est l'ensemble de tous les mondes possibles et peut être produit par la machine d'états (toutes les chaînes acceptées par l'automate) . Puisque la définition de *World / world* mappe à une machine d'états (ou une machine de Turing universelle) cela implique-t-il implicitement que tout système étudié dans notre réalité physique est une machine à états finis ? Bien que (Deutsch, 1984) ait montré que tout ce qui calculable peut être calculé sur une machine universelle de Turing (y compris la mécanique quantique), et plusieurs auteurs avancent l'hypothèse que notre univers est calculable, encore qu'il n'y a pas de preuve définitive qu'une structure de machine d'états quelconque puisse représenter une représentation complète et précise de notre réalité Néanmoins, une telle hypothèse n'est pas déraisonnable compte tenu du fait que, à toutes fins utiles et pratiques, une telle structure peut se rapprocher à un haut degré de tout système ; à condition que notre calcul et nos connaissances puissent évoluer jusqu'au degré de complexité du domaine étudié [8], par exemple un système avec quatre signaux lumineux chacun peut prendre trois valeurs (rouge, vert, bleu)

8. Même un minuscule système discret peut avoir un nombre énorme d'états (worlds) qui croîtrait exponentiellement

peut avoir une cardinalité espace-état de 81 et une possible cardinalité *World* de 81 ! ou 5.8 x 10^{120} (en supposant qu'il n'y a pas de répétition d'états et pas de contraintes), pour mettre ce nombre en perspective, le nombre estimé d'atomes dans l'univers est d'environ 10^{82}. En réexaminant le concept de world (w minuscule) comme un monde possible, nous pouvons détecter les similitudes avec le concept de scénario, qui est un monde possible de nature conceptuelle. Ainsi, les scénarios sont une image mentale d'une réalité objective. En ce sens, les scénarios représentent une tentative de deviner et de projeter ce que pourrait être un monde (la véritable séquence d'états réelle).

D'autres auteurs tels que (Hsia et al., 1994) ont proposé une approche de définition de scénarios dans le contexte d'une machine d'états ; cependant, la nôtre diffère sous trois aspects :

— Utiliser la séquence d'états plutôt qu'une séquence d'événements. La raison principale derrière notre choix est un argument empirique : dans un contexte d'analyse de scénario où les scénarios peuvent impliquer le monde réel (ou un système pas complètement compris comme dans un OWA), un observateur peut saisir différents changements d'états sans vraiment comprendre les événements ($\Sigma*$) qui ont conduit à ce changement d'état ;
— Nous adaptons l'approche de la machine d'états de l'ingénierie logicielle au domaine de l'analyse de scénarios ;
— Nous dérivons notre machine d'états d'une conceptualisation sémantiquement cohérente.

Considérer ce point nous aide à formuler une définition qui ne soit pas excessivement restreinte par le formalisme strict des machines d'états ou des ontologies. Pourtant, notre définition formelle sera formelle dans le sens d'un type de distinction de genre (Berg, 1983) où nous passerons du général au spécifique en ce qui concerne les scénarios tout en utilisant implicitement le cadre des machines d'état et des ontologies. La définition 5.6 incarne notre définition du concept de scénario.

Définition 5.6. Scénario : Un scénario est une séquence conceptuelle et sémantiquement cohérente d'états, dans le but soit d'explorer l'avenir d'un système à l'étude, soit de recommander des décisions pour le rapprocher d'un état désiré.

Cette définition synthétise les points clés proposés dans diverses définitions précédentes de scénarios dans le domaine de l'analyse de scénario, à savoir : Séquence, Cohérence, Incertitude et Conceptuel. La définition ajoute également la dimension de cohérence sémantique et met l'accent sur les deux objectifs principaux des scénarios qui sont les objectifs exploratoires et normatifs.

6 Conclusion

Notre analyse du domaine prédictif a établi deux constats. D'une part il n'existe pas de définition universelle de scénario, d'autre part l'analyse de scénarios ne prend pas en compte suffisamment de contraintes pour établir des prédictions de qualité prouvée. Dans cet article nous avons proposé une définition universelle des scénarios pour répondre au premier point. Pour répondre au second nous nous sommes intéressés au fait que les prédictions sont plus précises si le volume et l'origine des données étaient conséquent et hétérogène. Ce constat nous a porté dans le domaine du Big Data et ses 3 V (volume, variété, vélocité). Cependant ces trois

contraintes sont des prérequis qui ne sont pas suffisants. En effet, la sémantique, la valeur et la véracité des données sont impératives pour une prédiction fiable et sensée. Nous avons montré que les ontologies permettent de répondre à l'ensemble de ces critères. Par ailleurs, nous avons identifié que les machines d'états répondent aux besoins (en terme d'outil et de formalisation) de l'analyse de scénario. Nous proposons donc à partir de la définition uniformisée des scénarios, de les analyser par le biais des machines d'états basées sur des ontologies.
Cet article ébauche les prérequis de formalisation. Nous proposerons rapidement une formalisation mathématique complète de la démarche.

Références

Banks, J., J. Carson, B. L. Nelson, et D. Nicol (2004). *Discrete-Event System Simulation (4th Edition)* (4 ed.). Prentice Hall.

Berg, J. (1983). Aristotle's theory of definition. *Munich Institute of Technology*.

Bradfield, R. (2005). The origins and evolution of scenario techniques in long range business planning. *Elsevier*.

Cassandras, C. G. et S. Lafortune (2009). Introduction to discrete event systems. *Springer ; 2nd edition*.

Dean, J. et S. Ghemawat (2004). Mapreduce : Simplified data processing on large clusters. *Google Inc research group*.

Deutsch, D. (1984). Quantum theory, the church-turing principle and the universal quantum computer. *Proceedings of the Royal Society of London A 400, pp. 97-117*.

Genesereth et N. J. Nilsson (1987). *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann publishing.

Ghemawat, S., H. Gobioff, et S.-T. Leung (2003). The google file system. *Google Inc*.

Greeuw, S. C., M. B.A., van Asselt, J. Grosskurth, C. A. Storms, N. Rijkens-Klomp, D. S. Rothman, et J. Rotmans (2000). Cloudy crystal balls. *Experts ? corner report*.

Hardegree (1999). Symbolic logic basic concepts of logic. *University of Massachusetts courses*.

Hsia, P., J. Samuel, J. Gao, D. Kung, Y. Toyoshima, et C. Chen (1994). Formal approach to scenario analysis. *IEEE Softw. 11*(2), 33–41.

Jech, T. (1997). Set theory. *The Third Millennium Edition Springer [pp :5]*.

kacfah Emani, C., N. Cullot, et C. Nicolle (2015). Understandable big data : A survey. *ELSEVIER*.

Kahn Herman, A. J. W. (1968). *Year Two Thousand*. Collier Macmillan Ltd.

Kosow, H. et R. Gabner (2008). In methods of future and scenario analysis. *DIE Research Project ?Development Policy : Questions for the Future ?*.

Krotzsch, M., F. S. cik, et I. Horrocks (2013). A description logic primer. *University of Oxford*.

Laney, D. (2001). controlling data volume, velocity, and variety.

Lustig, I., B. Dietrich, C. Johnson, et C. Dziekan (2010). An ibm view of the structured data analysis landscape : descriptive, predictive and prescriptive analytics. *Analytics magazine*.

Martelli, A. (2014). *Models of scenario building and planning*. Bocconi on Management. Palgrave Macmillan.

Matheus, C. J., K. P. Baclawski, et M. M. Kokar (2003). Derivation of ontological relations using formal methods in a situation awareness scenario. *Proc.SPIE 5099*, 5099 – 5099 – 12.

Mintzberg, H. (2013). *Rise and Fall of Strategic Planning*. Free Press.

Muhammad Amer, Tugrul U.Daim, A. J. (2012). A review of scenario planning. *Elsevier*.

Nicola Guarino, Daniel Oberle, S. S. (2009). What is an ontology ? *Springer-Verlag Berlin Heidelberg 2009*.

Porter, M. E. (1998). *Competitive Advantage : Creating and Sustaining Superior Performance*. FREE PR.

R. Studer, R. Benjamins, D. F. (1998). Knowledge engineering : Principles and methods. data & knowledge engineering. *Elsevier Science B.V*.

Riabacke, M. (2012). *State-of-the-Art Prescriptive Criteria Weight Elicitation*. phdthesis, Department of Computer and Systems Sciences, Stockholm.

Ringland, G. (2014). *Scenario Planning*. Choir PR.

Schwartz, P. (1996). *The Art of the Long View*. Bantam Dell.

Steiner, G. A. (1997). *Strategic Planning*. Free Press.

Steinmüller, H. K. (1997). Grundlagen und methoden der zukunftsforschung. *Sekretariat für Zukunftsforschung*.

Stoelinga, M. (2003). An introduction to probabilistic automata. *UC santa cruz*.

Tsai, C., C. Lai, H. Chao, et A. V. Vasilakos (2015). Big data analytics : a survey. *Journal of Big Data, Springer open journal*.

Summary

Historically, the anticipation of a future situation is the main activity of individuals and organizations. In this context, we are interested in identifying potential sequences of events by scenario analysis, for a better prediction of possible futures. In the context of data analysis, the accuracy of this prediction depends greatly on the volume of data. However, in the Big Data domain, (kacfah Emani et al., 2015) shows that, beyond the criteria of volume, velocity and variability, one must also take into account the value and the veracity of the data.

These two aspects are treated by the use of ontologies. The semantic contribution makes it possible to know the value of the data in relation to their exploitation and the consistency, the completeness and the decidability, provided by the ontologies, guarantee their veracity in relation to the context.

This paper proposes a universal definition of scenario and a scenario analysis approach in a Big Data context, based on ontologies. This approach allows the construction of predictions integrating notions of value and veracity.

« L'avenir en commun » des Insoumis. Analyse des forums de discussion des militants de la France Insoumise

Clément Plancq*, Zakarya Després**
Julien Longhi***

*Lattice, ENS, CNRS, Univ. Paris 3, 1, rue Maurice Arnoux. 92120 Montrouge
clement.plancq@ens.fr,
http://www.lattice.cnrs.fr/plancq
**Lattice, ENS, CNRS, Univ. Paris 3, 1, rue Maurice Arnoux. 92120 Montrouge
zakarya.despres@gmail.com
***Université de Cergy-Pontoise. Laboratoire AGORA
33 Boulevard du port. 95011 Cergy-Pontoise
julien.longhi@u-cergy.fr
https://www.u-cergy.fr/fr/_plugins/mypage/mypage/content/jlonghi.html

Résumé. Les discours politiques ont fait l'objet de travaux marquants en analyse du discours et en TAL mais les études sur les discussions de militants sont plus rares. Pourtant ces communautés sont le lieu d'échanges idéologiques sur le programme d'un candidat. L'étude de ces discussions peut se révéler intéressante pour étudier la circulation des idéologies de l'appareil politique vers une communauté de citoyens et vice-versa.

Dans l'article nous présentons les travaux menés pour recueillir un corpus de messages émanant de forums de discussion des militants de la France Insoumise puis les analyses conduites sur ce corpus à l'aide des outils de la plateforme Cortext.

1 Introduction

Les travaux que nous présentons ont été menés pendant le datasprint datapol [1]. Plus précisément ce travail s'inscrit dans le projet « #Présidentielles 2017 : comparaison et circulation des idéologies ». Parmi les questions abordées dans ce projet, nous nous sommes intéressés à la porosité entre les idéologies des candidats et les communautés des militants.
Un datasprint est un exercice stimulant avec des contraintes fortes. L'étude présentée en est empreinte : d'une part, après seulement 3 jours de travail sur les données notre étude est nécessairement exploratoire et d'autre part nous nous sommes concentrés sur les idéologies, laissant de côté une part non négligeable du contenu du corpus. Pour ces recherches, le Lattice a bénéficié d'un financement de PSL (Paris Sciences et Lettres, ref. ANR-10-IDEX-0001-02 PSL), dans le cadre de l'appel à projets SHS 2016.

1. datapol (http://bit.ly/data-pol) organisé par le medialab de Sciences-po, du 29 novembre au 2 décembre 2017.

Plusieurs travaux, chercheurs, ou projets, ont analysé les discours politiques lors des récentes campagnes électorales. Le projet « Mesure du discours » piloté à Nice par Damon Mayaffre a contribué à rendre accessible l'analyse des discours politiques de la campagne présidentielle 2017, dans Mayaffre et al. (2017) notamment. Les travaux de Alduy (2017) ont proposé une analyse de 1300 textes (2,5 millions de mots - écrits ou prononcés de 2014 à 2016) des principaux candidats avec le logiciel Hyperbase. D'autres travaux se sont intéressés aux discours politiques numériques, tels que le projet #Idéo2017 piloté par Julien Longhi à l'université de Cergy-Pontoise. Ce projet s'appuie notamment sur une caractérisation préalable du tweet politique comme genre de discours (Longhi (2013)), et confère une légitimité aux analyses de discours natifs du web.
À notre connaissance les forums des militants n'ont pas encore fait l'objet de telles attentions. Pourtant les forums de discussion sont une source de données souvent exploitée. Au point d'avoir suscité des reflexions méthodologiques ; ainsi pour Marcoccia (2004) [2] « il s'agit d'un corpus idéal pour l'analyse des conversations et l'analyse du discours, car il répond aux critères suivants : 1. Il s'agit d'échanges authentiques produits en l'absence de l'analyste qui les enregistre, ce qui permet d'éviter un des problèmes méthodologiques habituels de l'analyse des conversations [...] 2. Ces corpus sont homogènes, définis par leur mise en mémoire [...] ». Pour les militants et les sympathisants d'un homme ou d'un parti politique, les réseaux sociaux numériques sont des supports d'échanges et de dialogues privilégiés que le chercheur peut exploiter. Notre objectif ici est de mesurer la diffusion de l'idéologie de la France Insoumise en cherchant, dans l'analyse des productions de la base militante, les traces des idéologies saisissables sous formes de doxas. Pour Longhi et Sarfati (2012) la doxa, par ses contenus comme par ses formes expressives, « tend à verser dans le domaine public, au-delà de l'institution de sens dont elle tire ses contenus minimaux. Elle se distingue par son haut degré de stéréotypie, ainsi que par une hétérogénéité non marquée. Enfin, la doxa est le lieu par excellence de la naturalisation d'un discours ».

Nous nous sommes focalisés sur le candidat Jean-Luc Mélenchon : sur son programme « L'avenir en commun » (LAEC) et sur la communauté militante de la France Insoumise (les Insoumis) telle qu'elle a pu s'exprimer dans les forums de discussion numériques. Sur le plan numérique le candidat Mélenchon se distingue des autres candidats aux élections présidentielles de 2017. D'abord parce qu'il a placé le numérique au centre de sa stratégie de campagne : en plus d'être présent sur les réseaux sociaux, il fut le premier en France à utiliser le logiciel NationBuilder et à avoir une chaîne Youtube. Mais surtout le candidat a été soutenu par des relais militants auto-organisés avec le « Discord insoumis » dont l'idée a germé dans un forum de discussion de jeuxvideo.com. Ces deux espaces numériques distincts, l'un officiel, l'autre participatif, se prêtent bien à notre objectif d'analyse de la circulation des idéologies.

Nous nous attarderons dans un premier temps sur les étapes de constitution d'un corpus de travail issu des forums de discussion des Insoumis. Puis nous détaillerons deux analyses outillées de ce corpus.

2. l'article porte sur les forums `usenet`.

2 Les données

2.1 Recrutement des données

Pour rassembler les conversations des Insoumis, nous nous sommes intéressés à deux espaces de discussion : le forum Blabla 18-25 de jeuxvideo.com et le Discord des Insoumis.
Si comme son nom l'indique, jeuxvideo.com est à la base un site consacré aux actualités sur les jeux vidéo, au sein de son forum ce sont les différentes sections de discussions générales qui y sont les plus populaires [3]. On y trouve notamment le "Blabla 18-25", où se retrouvaient de nombreux Insoumis pendant la campagne. Ils se sont notamment rassemblés sur une série de sujets, spécifiquement dédiés à Jean-Luc Mélenchon et France Insoumise.
Depuis 2016, le site jeuxvideo.com ne propose plus d'API publique pour accéder à ses données, nous avons donc écrit un script de web scraping en Python, grâce à la bibliothèque logicielle BeautifulSoup [4]. Nous avons ainsi pu récupérer 21 4761 messages, postés sur une série de sujets consacrés à Jean-Luc Mélenchon et la France Insoumise sur une période s'étendant du 12/11/2016 au 31/10/2017.
C'est aussi du forum Blabla 18-25 qu'est né le Discord des Insoumis, second espace de discussion que nous avons étudié. Discord est un logiciel gratuit de VoIP, conçu à la base pour les joueurs de jeux vidéo. Son avantage, par rapport à ses alternatives comme Slack ou Skype, est de permettre de converser dans des salons vocaux et des salons textuels en parallèle. Pour une communauté comme celle des Insoumis, c'est un moyen d'organiser des débats sous plusieurs formes, à l'écrit comme à l'oral, voire même de produire des podcasts comme ceux de Radio Insoumise.
Contrairement à jeuxvideo.com, Discord possède une API, permettant de récupérer et de poster du contenu. Avant de demander des données d'un serveur, il faut d'abord avoir un compte utilisateur qui y ait accès, afin d'obtenir un token d'authentification qui va nous autoriser à faire des requêtes via l'API. Ces requêtes HTTP retournent du contenu au format JSON dans lequel on trouve le message, sa date mais aussi les liens externes vers des articles ou des images qui ont été partagés par les utilisateurs. Le corpus basé sur Discord est donc constitué de 509 765 messages, datés du 07/02/2017 au 30/10/2017.

2.2 Préparation des données

La préparation des données a été effectuée en amont du *datasprint* afin de les rendre disponibles aux participants de l'événement. Les données étant sous Copyright il n'était pas question de les rendre publiques mais même sous couvert d'un accès restreint il a fallu les modifier afin d'assurer d'une part l'anonymat des messages et d'autre part la facilité d'utilisation des données. Les six fichiers JSON récoltés dans la phase de collecte ont été transformés en fichiers tabulaires comportant les colonnes suivantes :

— `date` : date du message au format Y-m-d ;
— `time` : heure du message au format H-M-S ;
— `content` : le contenu du message ;
— `attach` : l'URL des fichiers attachés aux messages discord ;
— `embed` : l'URL des fichiers inclus dans les messages discord

3. source : http://jvstats.forum-stats.org/stats/1/
4. https://www.crummy.com/software/BeautifulSoup/

Pour les besoins de l'analyse les six fichiers tabulaires ont été agrégés dans une structure de données (*dataframe*) de la bibliothèque pandas [5]. Les données ont été indexées sur le champ `date`, ce qui a facilité les extractions et les analyses situées sur des intervalles de temps choisis.

	messages	mots	dates	origine
jvc.csv	214 761	6 760 746	12/11/2016 - 30/10/2017	jeuxvideo.com
blabla.csv	47 790	631 751	03/09/2017 - 30/10/2017	discord
debat_actu.csv	4 820	88 045	15/10/2017 - 30/10/2017	discord
debat_direct_an.csv	15 518	201 044	10/07/2017 - 30/10/2017	discord
discussion_fi.csv	430 475	6 670 480	07/02/2017 - 30/10/2017	discord
radio_insoumise.csv	11 162	145 589	09/05/2017 - 30/10/2017	discord
total	**724 526**	**14 497 655**		

TAB. 1 – *Détail du corpus France Insoumise.*

Le recueil des discussions des Insoumis présenté dans tableau 1 n'a pas prétention à constituer un corpus exhaustif ou même représentatif. Le corpus de la France Insoumise (corpus FI) ne contient pas les messages antérieurs au 12/11/2016 postés sur le forum Blabla 18-25, il n'inclut ni les dicussions orales tenues dans le discord insoumis ni les canaux de discussion fermés ou archivés entre-temps. Le corpus FI nous semble néanmoins suffisamment volumineux et échelonné dans le temps pour permettre une analyse outillée.

3 Analyses et résultats

Les analyses que nous présentons [6] ont été menées à l'aide des services de la plateforme CorTexT [7] de l'Ifris [8]. L'intégralité de notre corpus a pu y être prise en charge ; chaque analyse a bénéficié du même *modus operandi* inspiré de Chavalarias et Cointet (2008) : extraction des termes [9], tri et nettoyage des termes proposés, indexation des messages avec les termes et enfin cartographie des cooccurrences des termes dans les messages à l'aide du script CorTexT « network mapping ».

3.1 De quoi discutent les militants ?

La première analyse a consisté à identifier et cartographier les thématiques abordées dans les discussions des militants de la France Insoumise. Nous avons extrait une liste de 500 termes candidats de l'ensemble du corpus en employant les paramètres suivants : chi-2 pour le score de spécificité, occurrence minimale de 3, longueur maximale de 3 tokens. Cette liste a ensuite été nettoyée manuellement, nous l'avons expurgé de termes non pertinents (*seul truc*, *mais bon*, *bonne nuit*, etc . . .). Puis l'ensemble des messages a été indexé en fonction de la liste de

5. https://pandas.pydata.org/

6. En plus des auteurs, trois étudiants du Master de sociologie des mondes numériques de Marne-La-Vallée ont participé à ces analyses. Emmanuelle Coniquet, Amalia Nikolaidi et Adil Ouafssou.

7. http://www.cortext.org/

8. http://ifris.org/

9. l'extraction de CorTexT porte sur les formes racinisées

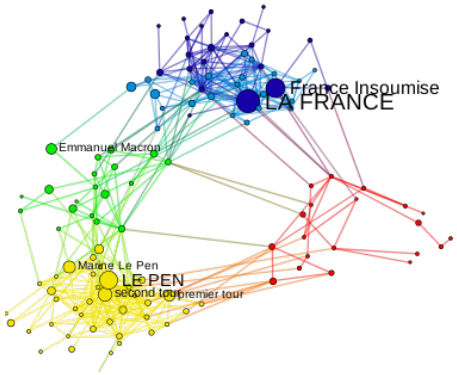

FIG. 1 – *Réseau des termes clustérisés sur l'ensemble du corpus.*

termes.
La figure 1 est le résultat de l'analyse en réseau de CorTexT, elle représente la carte des cooccurrences des termes indexés dans les messages du corpus. Dans la figure chaque nœud est un terme associé à sa valeur de cooccurrence, nous avons restreint le nombre de noeuds du graphe à 150. Les arêtes entre les noeuds sont pondérées par une mesure de proximité (ici la similarité cosinus). Les grappes de points du graphe forment des communautés ou clusters, l'algorithme de détection de communautés utilisé est celui de Louvain. Le partitionnement en sous-graphes y est basé sur la maximisation de la modularité de Newman (2006).
Dans la figure, plus deux termes sont proches plus ils sont associés fréquemment dans le corpus. Plus un terme apparaît conjointement avec d'autres termes, plus le point est gros.
Les clusters de couleurs jaune et vert clair de la figure 1 regroupent les discussions sur les autres candidats (Le Pen, Macron, Fillon, Hamon), le cluster rouge rassemble les préoccupations d'organisation interne du groupe et les actions de soutien au candidat Mélenchon, le cluster bleu réunit les discussions idéologiques sur le programme « L'avenir en commun ».
La figure 2 offre un zoom sur le sous-graphe idéologique et permet d'y distinguer des termes saillants comme : *fraude fiscale*, *transition énergétique*, *service public*, *traités européens*, *assemblée constituante*, etc … autant de mots clés du programme du candidat Mélenchon, tous reliés aux deux points centraux du sous-graphe que sont *LA FRANCE* et *France insoumise*. Cette analyse montre clairement qu'une partie importante des discussions porte sur l'idéologie.

3.2 « L'avenir en commun » vus par les Insoumis

En second lieu nous avons cherché à confronter les messages du corpus avec le texte du programme LAEC. Ce programme est une production du parti politique la France Insoumise, il a d'abord été publié aux éditions du Seuil avant que les Insoumis n'en fassent le site http://laec.fr [10]. Il s'agit d'un texte qui fait œuvre de communication politique, son contenu

10. cette initiative de publication électronique est d'ailleurs discutée sur le discord insoumis et se retrouve dans notre corpus.

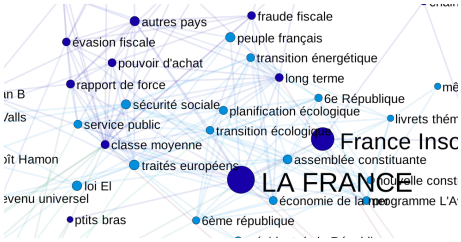

FIG. 2 – *Zoom sur les termes du cluster idéologique.*

est choisi, la langue est soigneusement vérifiée alors que notre corpus recèle des productions écrites spontanées où la syntaxe et l'orthographe ne sont pas toujours corrigées. De plus les messages n'ont pas forcément une sémantique autonome, ils peuvent faire partie d'un fil de discussion et faire référence de manière elliptique à des arguments exposés antérieurement. Ainsi par exemple : « pour recentrer un peu sur ce que tu disais, je pense sincèrement qu'il n'y a pas de vrai vote utile. » (discord, 04/04/2017).
LAEC et le corpus FI sont donc très différents par leur forme. Par leur volume également, nous n'avons pas pu compter la taille en nombre de mots du programme mais les 128 pages du livre forment un ensemble qui n'est pas statistiquement comparable avec les 13 millions de mots du corpus FI.

L'objectif de cette seconde analyse est de découvrir en quels termes sont discutés chacun des 7 chapitres du programme LAEC dans le corpus. Pour cela nous avons extrait manuellement 150 termes du programme. Puis la sous-partie du corpus antérieure au 23 avril 2017, date du premier tour de l'élection présidentielle, a été indexée en fonction de la liste de termes. À nouveau CorTexT a été sollicité pour produire un graphe de cooccurrences. Nous avons utilisé la même méthode qu'en 3.1.
6 clusters différents se distinguent dans la carte générée. Nous nous sommes intéressés cette fois aux labels des clusters. Ceux-ci sont proposés automatiquement, les labels étant les termes les plus centraux d'un cluster donné, c'est-à-dire ceux qui ont le plus de relations avec l'ensemble des termes du cluster.

chapitres	labels
La 6ème République	nouvelle constitution & constituante
Protéger et partager	smic & dette
La planification écologique	international & climat
Sortir des traités européens	UE & Europe
Pour l'indépendance de la France	humanisme & privatisation
Le progrès humain d'abord	militant & mouvement
La France aux frontières de l'humanité	

TAB. 2 – *Titres des chapitres de LAEC et labels.*

Le tableau 2 rapproche les intitulés des chapitres de LAEC avec les labels des clusters de la seconde analyse. Si un label comme *militant & mouvement* appartient plutôt exclusivement à la sphère militante, les quatre labels restants partagent des proximités sémantiques avec les chapitres de LAEC : *nouvelle constitution & constituante* et *La 6ème République*, *UE & Europe* et *Sortir des traités européens* par exemple. Nous pouvons émettre l'hypothèse qu'il s'agit là de reformulations par les militants des thèmes du programme du candidat. Signe d'une véritable porosité entre la communication officielle et les réseaux sociaux auto-organisés des militants.

4 Conclusion

Nous avons présenté les fruits du travail que nous avons pu mener pendant les trois jours du datapol, ils ne constituent qu'une première exploration mais les analyses sont encourageantes. L'analyse outillée a confirmé notre hypothése selon laquelle une part non négligeable des discussions des militants de la France Insoumise portent sur les idéologies défendues par le candidat et son programme.
Ce travail soulève surtout des questions que le format du *datasprint* ne nous a pas permis d'examiner : comment quantifier l'importance des discussions idéologiques comparées à l'ensemble du corpus ? Comment qualifier ces discussions ? Nous savons que les thèmes du programme ont été discutés mais nous ignorons dans quelle mesure ils ont fait l'objet de controverses. La question pourra être traitée dans un travail ultérieur en appliquant des techniques d'analyse de sentiment issues du TAL ou à l'aide d'indices linguistiques de négation ou de polarité négative.

Références

Alduy, C. (2017). *Ce qu'ils disent vraiment. Les politiques pris aux mots*. Le Seuil.

Chavalarias, D. et J.-P. Cointet (2008). Bottom-up scientific field detection for dynamical and hierarchical science mapping - methodology and case study. *Scientometrics 75*(1), 20.

Longhi, J. (2013). Essai de caractérisation du tweet politique. *L'information grammaticale 136*, 25–32.

Longhi, J. et G. Sarfati (2012). *Dictionnaire de pragmatique*. Colin.

Marcoccia, M. (2004). L'analyse conversationnelle des forums de discussion : questionnements méthodologiques. *Les Carnets du Cediscor 8*.

Mayaffre, D., C. Bouzereau, M. Ducoffe, M. Guaresi, F. Precioso, et L. Vanni (2017). Les mots des candidats, de " allons " à " vertu ". In P. Perrineau (Ed.), *Le vote disruptif. Les élections présidentielle et législatives de 2017*, pp. 129–152. Presses SciencesPo.

Newman, M. E. (2006). Modularity and community structure in networks. *Proc Natl Acad Sci U S A 103*(23), 8577–8582.

Summary

Political speeches have been the focus of discourse analysis and NLP, but studies of activist discussions are scarcely found. Yet these communities are the place of ideological exchanges

on the party platform. The study of these discussions can be interesting to study the circulation of ideologies of the political apparatus towards a community of citizens and vice versa.

In the article we present the work carried out to gather a corpus of messages emanating from forums of discussion of the militants of France Insoumise. Then we discuss the analyzes conducted on this corpus using the tools of the platform Cortext.

Résolution de problèmes de cliques dans les grands graphes

Jocelyn BERNARD*, Hamida SEBA*

*Université Lyon 1, LIRIS CNRS 5205, Villeurbanne, France
prenom.nom@univ-lyon1.fr

Résumé. Le problème MCE (Maximal Clique Enumeration) est un des problèmes que l'on rencontre dans l'analyse des graphes de données. C'est un problème NP-difficile pour lequel des solutions adaptées doivent être conçues dans le cas de grands graphes. Nous proposons dans ce papier de répondre à cette problématique en travaillant sur une version compressée du graphe initial. Une telle approche semble intéressante à la fois en terme de temps de calcul et d'espace mémoire. Nous avons donc implémenté notre approche et l'avons comparée à plusieurs solutions de la littérature.

1 Introduction

Un graphe $G = (V, E)$ est un outil de modélisation qui comprend un ensemble de nœuds V ainsi qu'un ensemble d'arêtes E. Une arête est un lien entre deux nœuds. Les graphes ont comme vocation d'être utilisés en tant que structures de données permettant de modéliser des objets ou des problèmes avec de nombreuses applications, notamment dans les domaines des réseaux sociaux, des réseaux informatiques ou encore de la génétique. Parmi ces applications se trouve la recherche de structures, par exemple, dans le cadre de la génétique on peut rechercher des interactions protéines-protéines (Przulj (2003)), on peut également vouloir chercher des communautés dans les graphes issus des réseaux sociaux, etc.

Dans ce travail nous nous intéresserons au problème d'énumération de cliques qui a fait l'objet de plusieurs études : Bron et Kerbosch (1973); Tomita et al. (2006); Prosser (2012). Une clique est un sous-ensemble complet de nœuds du graphe. C'est-à-dire que chacun des nœuds du sous-ensemble est connecté avec l'intégralité des autres.

Le problème d'énumération de cliques maximales ou MCE pour l'anglais Maximal Clique Enumeration est un problème théoriquement NP-Difficile qui se définit de la manière suivante : Une clique K est maximale si elle n'est pas incluse dans une clique de taille supérieure. Le problème d'énumération de cliques maximales consiste à lister l'ensemble des cliques maximales du graphe G.

Dans le cas de grands graphes, ce problème devient difficile à traiter car les données sont nombreuses (le nombre de nœuds et le nombre d'arêtes) et le temps d'exécution est exponentiel avec la taille de ces données. Dans ce cas, plusieurs solutions ont été envisagées : trouver des solutions non exactes mais proches d'un optimal avec des heuristiques Wang et al. (2013), partitionner le graphe Guo et al. (2017), etc. Dans cet article, nous explorons une autre solution qui consiste à travailler sur des données plus petites en compressant le graphe.

La compression de graphes est une opération qui permet la diminution du nombre d'arêtes ou de nœuds du graphe. Cette opération permet par exemple de rendre le graphe plus facile à transmettre ou à lire. Il existe plusieurs méthodes de compression de graphes qui diffèrent selon le type du graphe : simple (Zhou (2015)), orienté (Adler et Mitzenmacher (2001)), étiqueté (Tian et al. (2008)).

Nous proposons de résoudre le problème MCE sur un graphe compressé (sans le décompresser). Nous émettons l'hypothèse que les avantages d'une telle approche sont que :

— le graphe compressé prend moins d'espace mémoire et peut être chargé en mémoire lors de son traitement. Ceci permet de réduire le nombre d'entrées/sorties engendrées si le graphe d'origine ne tient pas en mémoire.
— le graphe compressé étant plus petit, un gain de temps de traitement devrait être réalisé.

Nous avons implémenté cette approche et l'avons comparée à des algorithmes existants. Les résultats sont prometteurs et ouvrent plusieurs perspectives. Le reste de l'article est structuré comme suit : dans la section 2, nous proposons un état de l'art permettant la compréhension de la compression de graphe et des algorithmesexistants pour résoudre le problème MCE. Dans la section 3 nous détaillons notre raisonnement pour la résolution du problème MCE sur le graphe compressé. Dans la section 4, nous proposons une évaluation de notre travail via une expérimentation. Enfin, dans la section 5, nous présentons la conclusion et les perspectives futures qu'offre notre méthode.

2 État de l'art

2.1 La compression de graphes

La décomposition modulaire (Gallai (1967)) va être utilisée pour compresser un graphe. La méthode de la décomposition modulaire revient à réaliser différents agglomérats de nœuds, en fonction de leurs voisinages : McConnell et Spinrad (1999); Habib et Paul (2010). Ces agglomérats sont appelés des modules. Il existe différents types de modules :

— les modules feuilles : chaque module feuille correspondent à un nœud simple dans le graphe original. C'est une feuille de l'arbre de décomposition produit par la compression modulaire.
— les modules séries : un nœud module série correspond à un ensemble de nœuds formant un sous-graphe complet (chacun des nœuds fils du nœud série est relié avec les autres).
— les modules parallèles : un nœud module parallèle correspond à un ensemble de nœuds formant un stable [1]. Il n'existe aucune arête entre les descendants du nœud parallèle. Le complémentaire du sous-graphe le représentant est un graphe complet.
— les nœuds premiers : un module premier correspond à un sous-graphe qui n'est pas complet et dont le graphe complémentaire [2] est également non complet.

La figure 1 illustre la compression d'un graphe en utilisant la décomposition modulaire. L'imbrication des modules, appelée arbre de décomposition modulaire, est présentée dans la figure 2. Les nœuds qui possèdent un voisinage identique sont coloriés de la même manière dans le graphe initial (voir figure 1(a)).

1. Un stable dans un graphe est un ensemble de sommets dont aucun n'est adjacent à un autre
2. Le complémentaire d'un Graphe G est le graphe G' obtenu avec les mêmes nœuds V du graphe G mais dont les arêtes entre deux nœuds sont présentes dans G' uniquement si les deux nœuds ne sont pas adjacents dans G

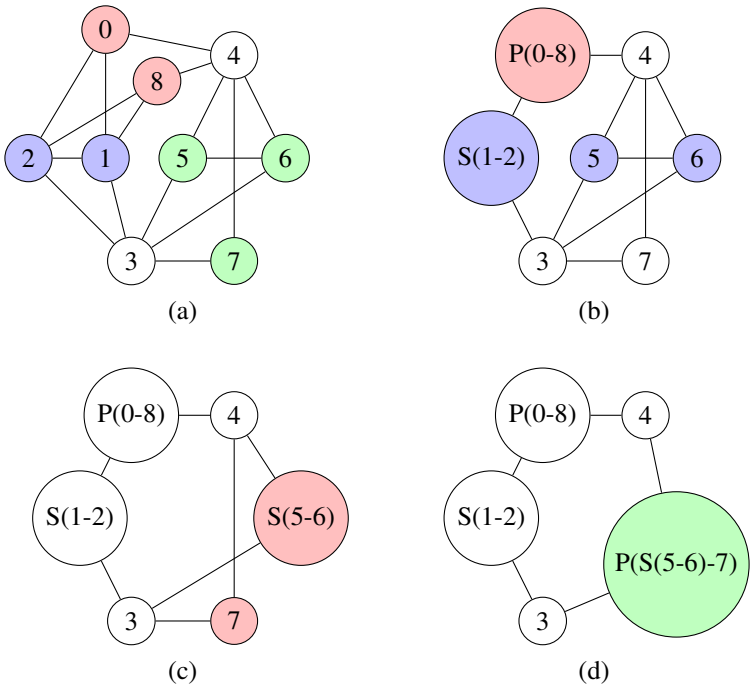

FIG. 1 – *Compression d'un graphe (a) avec deux étapes intermédiaires (b),(c), et le graphe compressé obtenu (d).*

— Les nœuds 1 et 2 ont le même voisinage et sont reliés entre eux. Ils forment un module série $S(1-2)$ (voir figure 1(b)).
— Les nœuds 0 et 8 ont le même voisinage et ne possèdent aucune arête entre eux. Ils forment un module parallèle $P(1-2)$ (voir figure 1(b)).
— Les nœuds 5, 6 et 7 possèdent le même voisinage, cependant ils ne forment ni un stable, ni une clique. Pour compresser cette partie du graphe, il faut d'abord compresser les nœuds 5 et 6 en module série $S(5-6)$, car ils sont reliés et possèdent le même voisinage (voir figure 1(b)/1(c)) puis ensuite de compresser le nœud 7 avec le nœud-module que l'on vient de compresser dans un nœud parallèle $P(S(5-6)-7)$ (voir figure 1(c)/1(d)).

On obtient un alors un graphe partiellement compressé (voir figure 1(d)).

On remarque que le sous-graphe composé des nœuds $P(0-8)$, $S(1-2)$, 3, 4 et $P(S(5-6)-7)$ ne peut plus être compressé par des modules séries ou parallèles. En effet il n'existe pas de nœuds ayant le même voisinage. Nous sommes alors en présence d'un module premier. La décomposition étant finie, on peut construire l'arbre de décomposition (voir figure 2).

2.2 Le problème de cliques MCE

Il existe plusieurs algorithmes et heuristiques pour le problème MCE. Nous décrivons dans cette section une partie des algorithmes proposés dans la littérature.

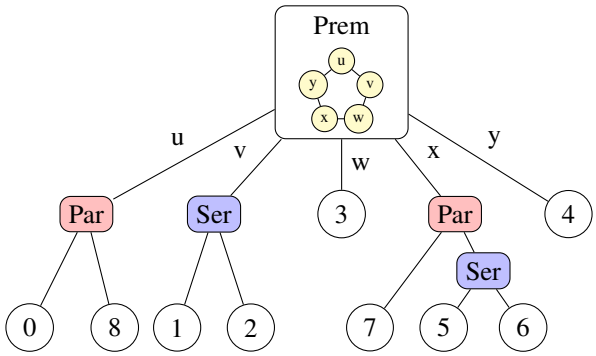

FIG. 2 – *Arbre de décomposition correspondant au graphe de la figure*

2.2.1 MCE

L'algorithme issu de l'article de Bron et Kerbosch (1973) est le premier algorithme créé pour trouver toutes les cliques maximales d'un graphe. Il fonctionne avec 3 ensembles :

— Un ensemble C qui contient la clique partielle construite à un instant t.
— Un ensemble T qui contient les nœuds candidats pour agrandir la clique partielle.
— Un ensemble D qui contient les nœuds ayant déjà été visités pour la construction d'une clique.

L'algorithme initialise donc T avec tous les nœuds V du graphe tandis que C et D sont vides. Ensuite l'algorithme réalise plusieurs itérations. Durant chacune d'elles, il sélectionne un pivot qui l'aide à choisir quels noeuds peuvent être ajoutés à la clique courante. Le choix du pivot permet de trouver toutes les cliques : celles où le pivot en fait partie, car ce ne sont que ses voisins qui sont enlevés de la boucle de sélection et celles où il n'en fait pas partie. La sélection du pivot étant aléatoire, l'expérimentation ne prend pas toujours le même temps et ne retourne pas toujours les cliques dans le même ordre.

2.2.2 Tomita

L'algorithme de Tomita et al. (2006) est aujourd'hui l'algorithme de référence. Sa complexité est de l'ordre de O($3^{(n/3)}$). Il diffère de la recherche de Bron et Kerbosh par le choix du pivot qui permet un élagage généralement plus efficace. Il utilise lui aussi 3 ensembles :

— Un ensemble Q qui contient la clique partielle qui est construite à un instant t.
— Un ensemble $SUBG$ et un ensemble $CAND$ qui permettent de sélectionner le meilleur pivot possible, puis le meilleur candidat, dans le but d'agrandir la clique.

L'algorithme initialise l'ensemble Q à vide tandis que $SUBG$ et $CAND$ sont initialisés avec l'ensemble des nœuds V. L'élagage se fait par le choix du pivot. Ce choix permet une recherche plus rapide car il maximise la taille de l'ensemble $CAND \cap \Gamma(pivot)$, où $\Gamma(pivot)$ correspond au voisinage du pivot. De plus si l'ensemble $CAND$ est vide mais pas $SUBG$ on retourne en arrière car la clique courante C générée peut être un sous ensemble d'une clique maximale. La procédure est présenté dans Algorithme 1.

Algorithm 1 Procedure-Tomita(Q,$SUBG$,$CAND$

if $SUBG = \emptyset$ **then**
  Retourner Q comme clique maximale
end if
choisir pivot $u \in SUBG$ qui maximise $|CAND \cap \Gamma(u)|$
while $CAND - \Gamma(u) \neq \emptyset$ **do**
  choisir $q \in CAND - \Gamma(u)$
  $Q \leftarrow Q + q$
  Procedure-Tomita($Q, SUBG \cap \Gamma(u), CAND \cap \Gamma(u)$)
  $Q \leftarrow Q - q$
  $CAND \leftarrow CAND - q$
end while

2.2.3 Eppstein

Eppstein et Strash (2011) ont proposé une amélioration de l'algorithme de Tomita. Avant d'exécuter Tomita, ils proposent d'utiliser la dégénérescence de graphe [3]. Pour chaque graphe G, on peut calculer son ordre de dégénérescence, qui est un ordre linéaire des sommets tels que chaque sommet a au plus d voisins plus tard dans l'ordre. La dégénérescence d'un graphe et son ordre peuvent être calculés en temps linéaire O(m) Batagelj et Zaversnik (2003). L'algorithme utilise les mêmes ensembles que Tomita, mais ils sont initialisés dans l'ordre de dégénérescence.

2.2.4 RMC

Wang et al. (2013) proposent quant à eux de lister seulement une partie des cliques maximales d'un graphe en évitant de retourner des cliques qui se chevauchent, c'est-à-dire dont les identifiants des nœuds qui les composent sont quasiment les mêmes. Pour cela, ils proposent la notion de τ-visibilité. La τ-visibilité est une variable qui permet d'écarter des cliques qui sont proches de la clique trouvée précédemment selon le seuil défini par l'utilisateur.

Les auteurs déclinent également l'algorithme en deux versions : une qui applique strictement le taux τ et une version aléatoire qui l'applique à quelques nuances près. L'algorithme ressemble à celui de Tomita et d'Eppstein mais sort plus tôt de la fonction si la clique qui est en train d'être construite se trouve être trop similaire à la précédente.

3 Énumération de cliques maximales sur un graphe compressé

L'algorithme que nous proposons est un algorithme récursif qui part de la racine de l'arbre de décomposition modulaire et qui l'explore en profondeur. Les cliques sont relevées de manières différentes en fonction du type du nœud parcouru.

3. La *dégénérescence* d'un graphe G est le plus petit nombre k de telle sorte que chaque sous-graphe $S \in G$ contient un sommet de degré au plus k

— Si le nœud est une feuille on retourne uniquement l'identifiant correspondant au nœud dans le graphe initial.
— Si le nœud est de type série : étant donné que tous ses fils sont reliés entre eux (et forment donc un graphe complet), on retourne un ensemble de listes d'identifiants qui correspondent aux nœuds pouvant être compris dans la clique trouvée. La figure 3 illustre un exemple.
— Si le nœud est de type parallèle : étant donné qu'aucun de ses fils n'est relié avec un autre (le complémentaire est premier), on retourne un ensemble de listes d'identifiants où chaque liste est issue du retour des nœuds fils du module parallèle. La figure 4 donne un exemple.
— Si le nœud est de type premier, il est nécessaire de lancer une recherche de clique. On commence alors par construire le graphe compressé correspondant $Gc = (V_c, E_c)$ à l'aide de ses nœuds fils V_c et de leurs arêtes E_c issues des listes de voisinages créés lors de la compression du graphe initial. On exécute ensuite un algorithme de recherche de cliques maximales sur le graphe compressé qui est basé sur Tomita. Ensuite, on parcourt les cliques trouvées de la manière suivante : on explore chacun des nœuds-module de la clique et on remplace leur identifiant de module par la liste de nœuds qu'ils retournent lors de leur appel de la fonction de recherche (qui dépend du type du module). La figure 5 montre un exemple pour ce cas.

3.1 Optimisation

Afin d'améliorer la vitesse des algorithmes, nous avons ajouté une optimisation. Cette optimisation permet à un nœud-module dont le type est différent du module feuille et dont le père est de type premier de sauvegarder la valeur qu'il a renvoyé au premier appel. Cela permet de ne pas avoir à parcourir à nouveau tout son sous-arbre s'il fait à nouveau appel à ce nœud. En effet si le père est de type premier, il est possible que ce nœud se retrouve dans plusieurs cliques et qu'il soit donc appelé plusieurs fois. Par exemple si un nœud-module est lui-même premier et qu'il se trouve dans deux cliques différentes lorsque l'on évalue les cliques dans le graphe compressé de son père, on va appeler deux fois sa fonction de retour. Or, lors du second appel on va devoir à nouveau reconstruire un graphe, lancer la recherche de clique et reconstruire la liste de retour, ce qui peut être coûteux en temps. Nous choisissons donc d'enregistrer la liste de retour et de retourner cette valeur en cas de second appel.

4 Évaluation

Dans cette section nous allons décrire les expérimentations réalisées afin d'évaluer notre approche. Pour cela nous avons implémenté notre algorithme ainsi que les algorithmes que nous avons présentés dans la partie état de l'art. Nous avons implémenté tous les algorithmes en C++ à l'aide la librairie SNAP[4]. Ceci afin d'utiliser les mêmes structures pour l'ensemble des algorithmes. Nous avons travaillé sur une machine 64 bits avec un système d'exploitation Ubuntu v14.04 LTS, un processeur i5-4690 de fréquence 3.5GHz et une mémoire vive de 8G.

Les algorithmes ont été compilés avec la version 4.8.2 du compilateur g++ et les options -std=c++11 et l'optimisation -03.

4. https ://snap.stanford.edu/

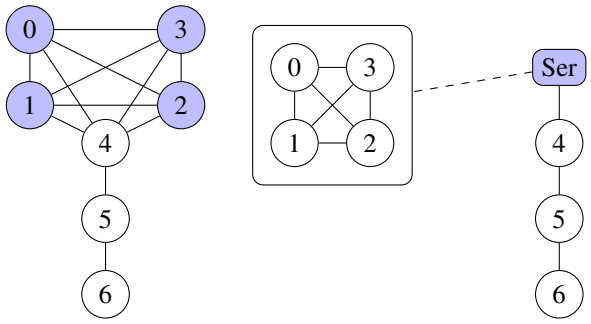

FIG. 3 – *Exemple du problème MCE avec un nœud série : Le nœud de type série, lorsqu'il est interrogé dans la fonction de recherche de cliques retourne la liste de ses nœuds fils (0,1,2 et 3). Ainsi les cliques trouvées par l'algorithme dans ce cas sont : {5,6}, {4,5} et {0,1,2,3,4}.*

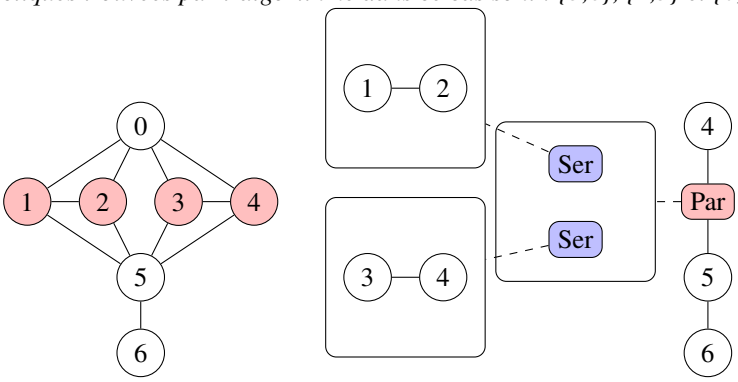

FIG. 4 – *Exemple du problème MCE avec un nœud parallèle : Le nœud de type parallèle, lorsqu'il est interrogé dans la fonction de recherche de cliques retourne un ensemble de listes issues des listes de retour de ses nœuds fils, dans ce cas il retourne ({1,2},{3,4}). Ainsi les cliques trouvées par l'algorithme dans ce cas sont : {5,6}, {0,1,2},{0,3,4},{5,1,2} et {5,3,4}.*

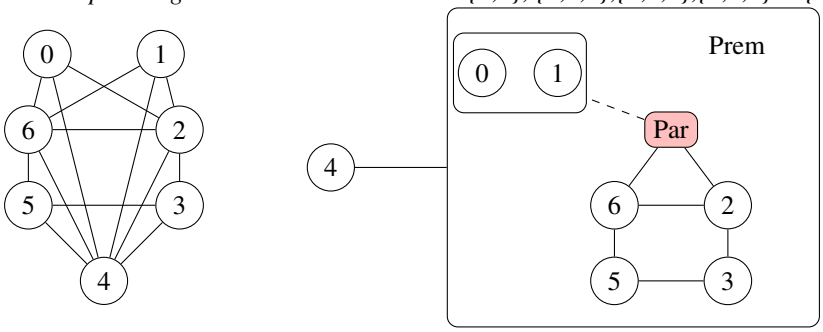

FIG. 5 – *Exemple du problème MCE avec un nœud premier : La racine de l'arbre est un nœud série qui relie le nœud 4 à un nœud premier dont on peut observer le graphe compressé correspondant à droite. Notre algorithme va donc lancer une recherche de clique maximale dans le graphe compressé et retourner une liste de cliques issue de sa recherche, à savoir ({0,2,6},{1,2,6},{2,3},{3,5},{5,6}). Ainsi les cliques trouvées par l'algorithme dans ce cas sont : {0,2,4,6},{1,2,4,6},{2,3,4},{3,4,5} et {4,5,6}.*

Notre algorithme de compression est basé sur l'algorithme de décomposition modulaire de Montgolfier (2003) qui a une complexité en O(m $\log$ n). L'algorithme de décomposition est lui-même issu de la concaténation de deux algorithmes distincts :

1. Le premier réalise une permutation factorisante des sommets Capelle et al. (2002) du graphe Capelle (1997) grâce à une technique d'affinage des partitions Habib et al. (1999).
2. La seconde construit l'arbre de décomposition modulaire à l'aide la permutation issue du premier algorithme Bergeron et al. (2008).

De plus, afin de permettre aux algorithmes de recherche de cliques de pouvoir fonctionner, nous avons ajouté un troisième algorithme qui ajoute aux nœuds fils d'un graphe premier une liste de voisinage. Concrètement, cela signifie que si un nœud est voisin avec un autre nœud dans le graphe initial, les modules les représentants dans un graphe premier auront également une arête de voisinage. L'algorithme est un algorithme récursif qui se contente de parcourir l'arbre en ajoutant une arête lorsqu'elle était existante dans le graphe précédent. Il teste chacun des descendants d'un nœud premier pour savoir si il existe une arête entre les nœuds qu'ils représentent dans le graphe original et ajoute une arête entre les descendants le cas échéant. L'algorithme 2 présente le détail de cette opération.

Algorithm 2 creeVoisins(Module m)

```
1:  if m est de type premier then
2:    for i allant de 0 à |m− > descendant| do
3:      for j allant de i à |m− > descendant| do
4:        if i est voisin de j dans le graphe then
5:          Ajouter i à la liste de voisinage de j dans le graphe compressé
6:          Ajouter j à la liste de voisinage de i dans le graphe compressé
7:        end if
8:      end for
9:    end for
10: end if
11: if m a des descendants premiers then
12:   for s ∈ m− > descendant do
13:     creeVoisins(s)
14:   end for
15: end if
```

4.1 Graphes de test

Le tableau 1 présente les graphes sur lesquels nous avons testé les algorithmes. Pour chaque graphe nous donnons dans le tableau 2 le temps nécessaire pour le compresser ainsi que la taille du graphe compressé (le nombre de modules), le nombre total d'arêtes dans l'ensemble des nœuds premiers de l'arbre et enfin ce que nous avons appelé le taux de compression qui correspond au rapport nombre d'arêtes dans le graphe compressé sur le nombre d'arêtes dans le graphe initial. Ces graphes sont tirés des bases de données des projets Dimacs et String (*http ://dimacs.rutgers.edu/Challenges/* & *https ://string-db.org/*).

ID	**Nom**	|V|	|E|	densité	d_{Min}	d_{Max}	d_{Moy}	Taille
(1)	16pk	4919	4972	0.000411	1	4	2.021	51.2Ko
(2)	3djd	11862	12010	0.000171	1	4	2.025	131Ko
(3)	c-fat500-5	500	23191	0.185900	92	95	92.764	194Ko
(4)	6pfk	17123	17263	0.000118	1	4	2.016	198Ko
(5)	Caenorhabditis_elegans	6173	26184	0.001374	1	418	8.483	273Ko
(6)	Takifugu_rubipres	5872	27077	0.001571	1	162	9.222	281Ko
(7)	3dmk	30386	30773	0.000067	1	4	2.025	369Ko
(8)	c-fat500-10	500	46627	0.373764	185	188	186.508	390Ko
(9)	Drosophila_melanogaster	8624	39466	0.001061	1	311	9.153	413Ko
(10)	Rattus_norvegicus	8763	39932	0.001040	1	334	9.114	419Ko

TAB. 1 – *Présentation des graphes de tests*

Nom	nombre modules	nombre d'arêtes	temps	taux
16pk	5532	4199	0.112408	15.547064
3djd	13488	9976	0.118331	16.935887
c-fat500-5	517	16	0.012088	99.931008
6pfk	19467	14226	0.412102	17.592539
Caenorhabditis_elegans	6656	24570	0.252191	6.164070
Takifugu_rubipres	6651	23530	0.248460	13.099679
3dmk	34537	25468	1.621132	17.239138
c-fat500-10	509	8	0.017551	99.982843
Drosophila_melanogaster	9137	38297	0.656012	2.962043
Rattus_norvegicus	9517	37279	0.611329	6.643794

TAB. 2 – *Présentation de la compression des graphes*

4.2 Résultats et discussions

Nous avons lancé des algorithmes sur l'ensemble des graphes. Le tableau 3 présente les résultats. La figure 4.2 présente les différents temps de calcul. Le premier graphique représente les résultats des algorithmes présentés dans la section état de l'art. Pour RMC, les résultats présentés sont ceux où l'algorithme a réalisé le plus petit temps d'exécution. Il convient également de noté que la courbe des 2 Eppstein étant très proche on ne voit que Eppstein-Random. Les deux autres graphiques représentent les différents temps de calcul entre Tomita (le meilleur algorithme exact pour le problème MCE) et notre algorithme sur le graphe compressé. Les graphiques présentés sont ceux dont les graphes ont un taux de compression de l'ordre de 0 à 20% pour le graphe de gauche et de l'ordre de 99% pour le graphe de droite. Nous avons également rajouté une courbe représentant le temps de notre algorithme plus le temps de compression du graphe.

Pour le problème d'énumération de cliques, même en ajoutant le temps de compression de graphe, on remarque que notre algorithme fait mieux que Tomita, notre algorithme de référence, dans les cas où le graphe a un taux de compression supérieur à 15%. Les autres algorithmes tels que MCE et RMC donnent certaines fois de meilleurs résultats en terme de temps, c'est pourquoi il serait intéressant de les adapter sur le graphe compressé.

4.3 Étude de la complexité

Nous conjecturons que la complexité de l'algorithme d'énumération de cliques est de l'ordre de $\sum_{i=1}^{ms} n_i + \sum_{j=1}^{mp} n_j + \sum_{k=1}^{mr} 3^{n_k/3} + n$ où *ms*, *mp* et *mr* représentent respectivement le nombre de modules séries, parallèles et premiers et n_i,n_j et n_k représentent respectivement le nombre de nœuds fils de chacun d'entre eux. n est le nombre de nœuds dans le graphe initial.

Algorithme/Graphe	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)
MCE	4972 0.1913	12010 1.3329	16 0.1874	17263 3.2995	19386 2.3920	18877 2.4517	30773 7.331336	8 0.5963	30613 3.782695	41118 4.7760
Tomita	4972 0.5277	12010 3.0566	16 0.0675	17263 6.4554	19386 1.2426	18877 1.3051	30773 19.9444	8 0.2033	30613 2.5967	41118 2.7588
Eppstein	4972 0.8198	12010 5.0202	16 0.0612	17263 11.0183	19386 1.7087	18877 1.5647	30773 37.3132	8 0.1663	30613 3.0690	41118 3.6712
Eppstein-hybrid	4972 0.8198	12010 5.0214	16 0.0613	17263 11.0178	19386 1.7068	18877 1.5779	30773 37.3035	8 0.1654	30613 3.0764	41118 3.6775
RMC(0.5)	4968 0.0114	12006 0.0297	13 0.2491	17259 0.0428	13432 0.3209	13701 0.3013	30769 0.0777	5 0.5832	23700 0.4547	22536 0.6302
RMC-random(0.5)	2028 0.0078	4447 0.0200	5 0.2250	6754 0.0294	8620 0.2571	8818 0.2314	12254 0.0530	2 0.6158	16476 0.3733	13868 0.4763
RMC(0.7)	4968 0.0115	12006 0.0303	14 0.2520	17259 0.0425	14102 0.3447	14449 0.3242	30769 0.0762	6 0.5233	24663 0.4869	28098 0.7853
RMC-random(0.7)	3209 0.0093	7435 0.0249	7 0.2468	10778 0.0352	10789 0.2867	11020 0.2780	19422 0.0631	4 0.5866	19755 0.4206	19912 0.6239
RMC(0.9)	4968 0.0110	12006 0.0300	14 0.2599	17259 0.0432	15213 0.3701	15620 0.3514	30769 0.0765	6 0.5271	26588 0.5358	38843 0.9258
RMC-random(0.9)	4351 0.0107	10459 0.0290	11 0.2412	15108 0.0417	13579 0.3484	13977 0.3318	26945 0.0732	6 0.5261	24106 0.5087	32121 0.8280
RMC(1.0)	4968 0.0115	12006 0.0312	14 0.2929	17259 0.0444	15370 0.3732	15836 0.3576	30769 0.0803	6 0.7644	27120 0.5467	39223 0.9402
RMC-random(1.0)	4968 0.0111	12006 0.0306	14 0.2591	17259 0.0437	15370 0.3744	15836 0.3559	30769 0.0793	6 0.5277	27120 0.5464	39223 0.9305
AlgorithmeCompresse	4972 0.4198	12010 0.2600	16 0.0002	17263 1.2894	19386 1.4125	18877 1.2926	30773 5.6461	8 0.0003	30613 3.4481	41118 4.4114

TAB. 3 – *Résultats pour le problème MCE. La première ligne donne le nombre de cliques trouvé par l'algorithme et la seconde son temps (en secondes)*

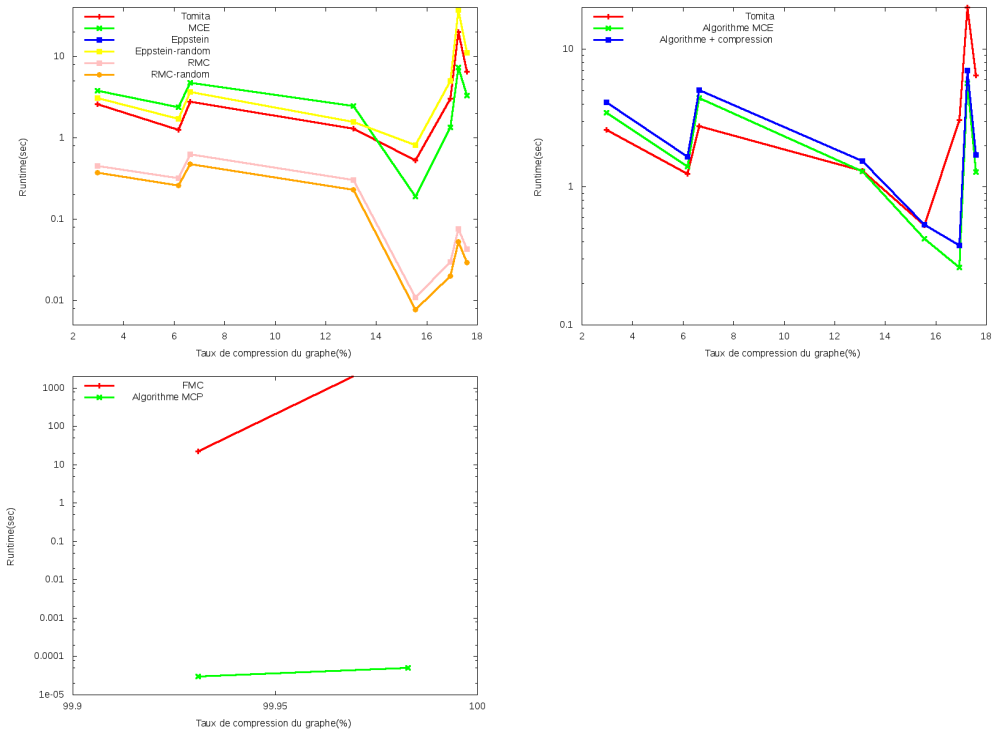

FIG. 6 – *Graphiques montrant les temps de calculs des algorithmes en fonction du taux de compression des graphes.*

5 Conclusion et ouvertures

Le problème d'énumération de cliques maximales dans les graphes est un problème complexe et difficile. Ce problème devient d'autant plus difficile lorsque les graphes sur lesquels nous travaillons sont grands. Pour résoudre ces problèmes, il existe de nombreux algorithmes et heuristiques. Dans ce papier, nous avons proposé et évalué une nouvelle méthode pour résoudre ce problème.

Notre algorithme se déroule en deux phases. La première consiste à compresser le graphe, via la décomposition modulaire pour diminuer la taille de celui-ci. La décomposition modulaire est une méthode de compression qui nous permet de garder les informations nécessaires à la recherche de cliques. La deuxième phase consiste à chercher les cliques sur le graphe compressé.

L'expérimentation réalisée démontre que notre algorithme d'énumération de cliques montre de meilleures performances dès que la compression du graphe est supérieure à 15%. Il nous semble intéressant de continuer le travail commencé en poursuivant plusieurs options :

— Augmenter l'efficacité des algorithmes en ajoutant des informations lors de la compression de graphes, notamment sur la taille des cliques dans les modules séries et parallèles.
— Développer de nouveaux algorithmes sur les graphes compressés.
— Regarder la consommation de mémoire vive de l'algorithme.
— Adapter la méthode aux autres types de graphes tels que les graphes orientés.
— Proposer une méthode de compression et de recherche de cliques en parallèle, l'algorithme de recherche de cliques démarrant dès qu'une compression du graphe est suffisante.

Références

Adler, M. et M. Mitzenmacher (2001). Towards compressing web graphs. In *Data Compression Conference, 2001. Proceedings. DCC 2001.*, pp. 203–212. IEEE.

Batagelj, V. et M. Zaversnik (2003). An o (m) algorithm for cores decomposition of networks. *arXiv preprint cs/0310049*.

Bergeron, A., C. Chauve, F. De Montgolfier, et M. Raffinot (2008). Computing common intervals of k permutations, with applications to modular decomposition of graphs. *SIAM Journal on Discrete Mathematics 22*(3), 1022–1039.

Bron, C. et J. Kerbosch (1973). Algorithm 457 : finding all cliques of an undirected graph. *Communications of the ACM 16*(9), 575–577.

Capelle, C. (1997). *Decompositions de graphes et permutations factorisantes*. Ph. D. thesis, Montpellier 2.

Capelle, C., M. Habib, et F. De Montgolfier (2002). Graph decompositions and factorizing permutations. *Discrete Mathematics and Theoretical Computer Science 5*(1), 55–70.

Eppstein, D. et D. Strash (2011). Listing all maximal cliques in large sparse real-world graphs. In *Experimental Algorithms*, pp. 364–375. Springer.

Gallai, T. (1967). Transitiv orientierbare graphen. *Acta Mathematica Hungarica 18*(1), 25–66.

Guo, J., S. Zhang, X. Gao, et X. Liu (2017). Parallel graph partitioning framework for solving the maximum clique problem using hadoop. In *Big Data Analysis (ICBDA), 2017 IEEE 2nd International Conference on*, pp. 186–192. IEEE.

Habib, M. et C. Paul (2010). A survey of the algorithmic aspects of modular decomposition. *Computer Science Review 4*(1), 41–59.

Habib, M., C. Paul, et L. Viennot (1999). Partition refinement techniques : An interesting algorithmic tool kit. *International Journal of Foundations of Computer Science 10*(02), 147–170.

McConnell, R. M. et J. P. Spinrad (1999). Modular decomposition and transitive orientation. *Discrete Mathematics 201*(1), 189–241.

Montgolfier, F. d. (2003). *Décomposition modulaire des graphes : théorie, extensions et algorithmes*. Ph. D. thesis, Montpellier 2, Université des Sciences et Techniques du Languedoc.

Prosser, P. (2012). Exact algorithms for maximum clique : A computational study. *Algorithms 5*(4), 545–587.

Przulj, N. (2003). Graph theory approaches to protein interaction data analysis. *proteins 120*, 000.

Tian, Y., R. A. Hankins, et J. M. Patel (2008). Efficient aggregation for graph summarization. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pp. 567–580. ACM.

Tomita, E., A. Tanaka, et H. Takahashi (2006). The worst-case time complexity for generating all maximal cliques and computational experiments. *Theoretical Computer Science 363*(1), 28–42.

Wang, J., J. Cheng, et A. W.-C. Fu (2013). Redundancy-aware maximal cliques. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 122–130. ACM.

Zhou, F. (2015). Graph compression. *Department of Computer Science and Helsinki Institute for Information Technology HIIT*, 1–12.

Summary

The MCE (Maximal Clique Enumeration) is a problem encontred in data graph analyses and mining. However, this problem is NP-hard. Consequently, appropriate solutions must be proposed in the case of large graphs. We propose in this work to answer this problem by working on a compressed version of the original graph. This approach seems interesting both in terms of computation time and memory space. So we implemented our approach and we compared it with several solutions from the literature.

Index