



DEPARTEMENT INFORMATIQUE
DE LA FACULTE DES SCIENCES

Ahmed kaci, Guilhèm Blanchard et Yanis Allouch

Rapport de TP 1 : Analyse Formelle de concepts



HMIN231 — Représentation des connaissances

Référent: Marianne Huchard

2021

Table des matières

I	Preprocessing	3
1	Compréhension	4
1.1	Comprendre les données du fichier	4
2	Transformation	6
2.1	Identifier les attributs binaires pertinents	6
2.2	Identifier des hiérarchies de spécialisation	7
3	Application de l'outil CoGui	10
3.1	Les tableaux binaires	10
3.2	Data Mining	10
II	Analyse des résultats	12
3.3	La comparaison des structures	13
3.4	Grandes catégories et sous-catégories	14
3.5	Les règles d'implication	14
3.6	Les exclusions mutuelles	15
III	Conclusion	16
3.7	Compréhension des données	17
3.8	Analyse des résultats	17
	Références	18

Introduction

Dans ce rapport, nous allons présenter une brève explication des données fournies par Dame Samb, plus précisément dans son article [SSN21] se plaçant avant l'étape a proprement dit du pre-processing ; faisant parti du *processus KDD*¹ explicité dans l'article de Fayyad [FPS96] ; sur lesquelles nous effectuerons différentes transformations qui consistent en l'identification des attributs binaires pertinent et des hiérarchies de spécialisation.

Ces données, transformées via un tableur, seront utilisées grâce à l'outil CoGui pour générer des tableaux binaires et effectuer du Data Mining. Nous procéderons par la suite a une analyse, des résultats obtenues, dans le but d'effectuer une comparaison des structures, d'identifier les grandes catégories et sous-catégories ainsi que les règles d'implications et d'exclusions mutuelles.

Les données transformées sont disponibles sur un google spreadsheet accessible a l'adresse url : <https://docs.google.com/spreadsheets> et l'ensemble des fichiers calculés et utilisées sur le dépôt git suivant : <https://www.gitlab.com/yanisallouch>.

1. Knowledge Discovery in Databases

Première partie

Preprocessing

1 Compréhension

1.1 Comprendre les données du fichier

Les données utilisées par Dame Samb se composent de 400 articles de recherches scientifiques publiés entre 2002 et 2019, la phase de sélection effectuée par ce dernier a permis de ne garder qu'un total de 58 algorithmes.

Sur la figure 1 nous retrouvons une présentation structurée de ces algorithmes qui combine les mining framework *les plus connus* (support, utility, uncertain), les mining approaches *les plus acceptés* (Apriori, FP-Growth, hybride) et les window models *les plus classiques* (incremental, sliding).

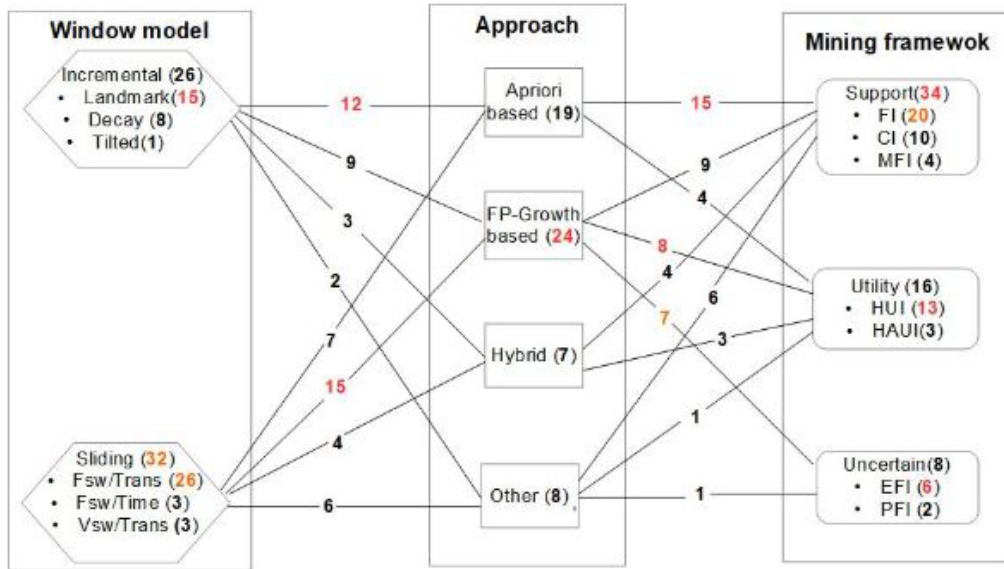


FIGURE 1 – État de l'art des algorithmes de Pattern Mining over Data Stream (PMDS)

Mining Framework

Tout d'abord, nous apprenons que la représentation *Frequent Itemset* (FI) est la première mesure ayant vu le jour au début des années 1990 et l'approche *Apriori* étant la première heuristique à l'utiliser.

Puis une amélioration de l'approche *Apriori* nommé *FP-Growth* a permis de combler une lacune présente dans son analyse. D'autres représentations ont vu le jour pour réduire le nombre de FI calculé, nommé *closed itemset* (CI) et *maximal itemset* (MFI).

Ces représentations introduites ci dessus, forment une catégorie nommée *Support* au sein de l'ensemble *Mining Framework*. La catégorie *Support* permet de faire des recherches sur des données binaires.

De manière succincte et ordonnée, nous retrouvons dans la catégorie *Utility*, le *High-Utility itemset* (HUI), puis *Transaction-Weighted Utilization* (TWU) et enfin *high average-utility itemsets* (HAUI). Remarquons que le concept TWU n'est pas présent sur la figure 1 et par conséquent n'aura pas passé l'étape de sélection par Dame Samb. Les deux représentations

restantes forment la catégorie *Utility* au sein du cadre *Mining Framework*. Cette catégorie est différente de *Support* parce que les données de tous les jours ne sont pas que binaires.

Enfin, la dernière catégorie nommée *Uncertain* qui doit son existence à des analyses basées sur des données imparfaites, imprécises et pouvant provenir de sonde créant beaucoup de bruit. Ces données peuvent être décousues selon les deux cas de représentations suivantes : *expected-support frequent* (EFI) et *probabilistic frequent itemsets* (PFI). Ces deux représentations associent une probabilité d'apparition à l'objet étudié, cependant PFI va aussi l'associer à chaque composant de l'objet.

Approach

Les différentes approches énoncées précédemment, *Apriori* et *FP-Growth* forment l'ensemble de l'état de l'art ainsi que deux autres approches.

L'approche *Hybrid* qui utilise alors un mélange des deux précédentes approches.

Une approche catégorisée comme *Other* ne spécifiant aucune approche en particulier, on peut supposer au vu des données que leur utilisation n'est pas suffisantes pour se distinguer des trois précédentes approches.

Window Model

Le cadre *Windows Model* spécifie des solutions aux problèmes du choix des éléments au sein d'une base de données selon les contraintes suivantes :

- l'ordre de passage (single pass constraint) ;
- temps de calcul limité (limited processing time) ;
- mémoire limitée (limited memory).

C'est donc quatre algorithmes de sélection de fenêtres qui se distinguent, *Landmark* est favorable quand tous les objets ont autant d'importance, *Sliding* est favorable selon une mesure de temps, *Damped* (Decay) est favorable quand les derniers objets sont importants et enfin *Tilted* est favorable aux derniers objets étudiés.

Les algorithmes *Landmark*, *Damped*, *Tilted* sont dits *Incremental*. L'algorithme *Sliding* possède trois versions basées sur deux variants *fixed-size windows* (Fsw) et *variable-size windows* (Vsw).

2 Transformation

2.1 Identifier les attributs binaires pertinents

Le nombre d'années de publication étant trop élevé, nous avons décidé de les partitionner par tranche de quatre ans afin d'avoir un nombre de classes réduit. Cela facilitera l'interprétation finale et nous permettra, éventuellement, de voir émerger des patterns selon la période de publication.

Le nombre de citations a été classifié en fonction d'un découpage logarithmique du nombre de citations.

Voici notre découpage (les données ayant servi à la génération sont disponibles sur la feuille de calcul) :

- Publié : $0 < \text{nombre de citations} \leq 15$
- Connu : $15 < \text{nombre de citations} \leq 30$
- Très connu : $30 < \text{nombre de citations} \leq 100$
- Populaire : $100 < \text{nombre de citations} \leq 300$
- Très populaire : $400 < \text{nombre de citations} \leq 1600$
- Exceptionnel : $\text{nombre de citations} > 1600$

Voici le graphique logarithmique ayant servi au découpage :

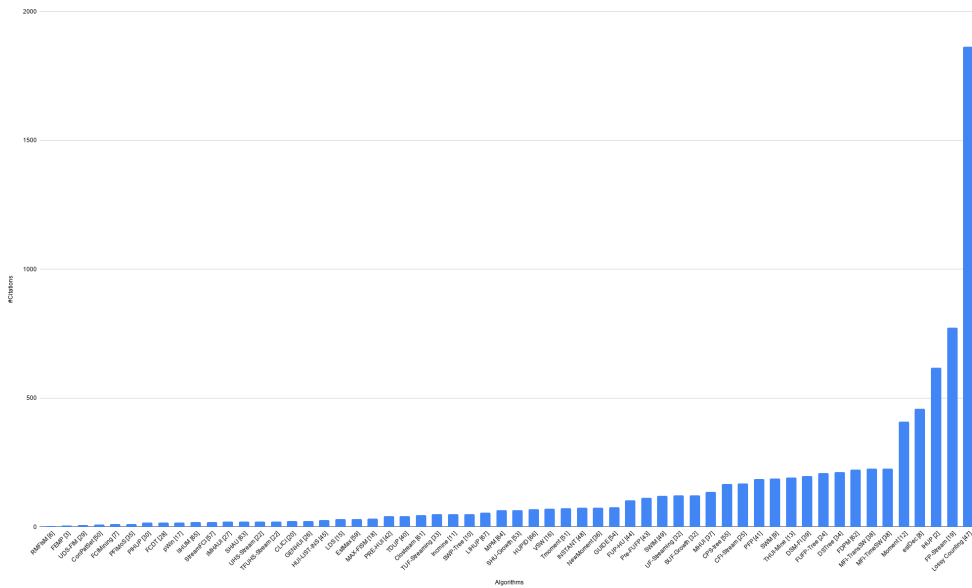


FIGURE 2 – Graphique du nombre de citations vu logarithmique

Nous avons supposé qu'un découpage qui uniformise la taille des groupes créé, permettrait de garder les algorithmes de mêmes popularités ensemble et nous pourrions alors étudier les patterns résultants sur des entités comparables.

Nous avons, ensuite, créé une classe pour chaque type de mining framework, d'approches, de window model, de précision, de datasets et du nombre d'étapes (steps) pour indiquer de façon binaire l'éventuelle appartenance des algorithmes dans les différentes classes.

2.2 Identifier des hiérarchies de spécialisation

Dans notre identification des hiérarchies de spécialisation sur les attributs, nous avons procédé à la taxonomie comme abordés dans le cours.
Voici le résultat de la taxonomie des différents attributs :

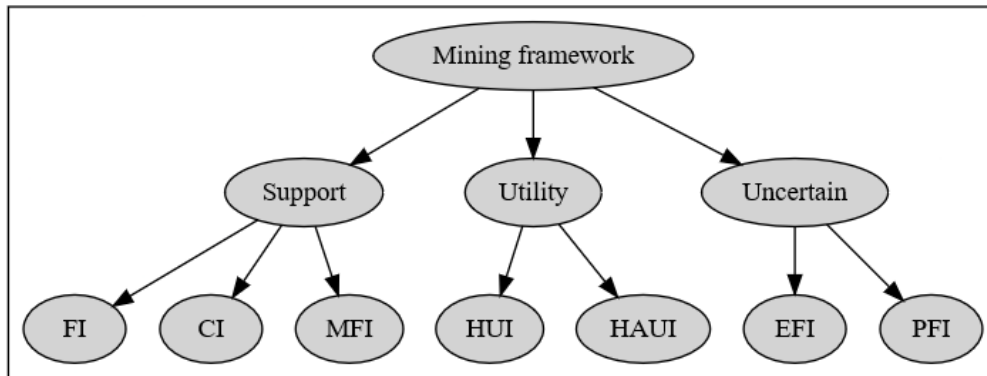


FIGURE 3 – Hiérarchisation des attributs Mining Framework

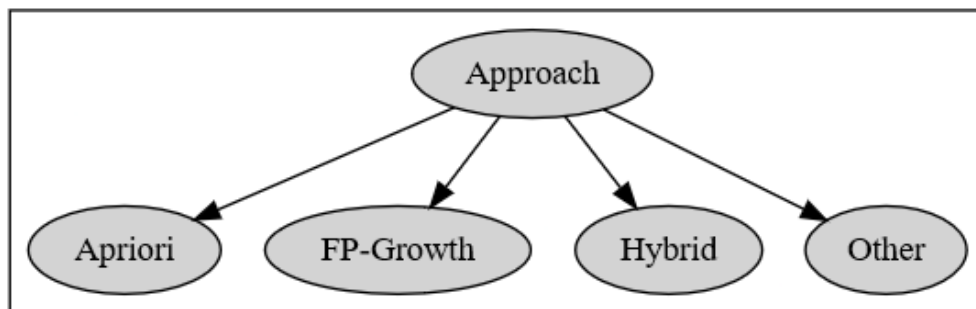


FIGURE 4 – Hiérarchisation des attributs Approach

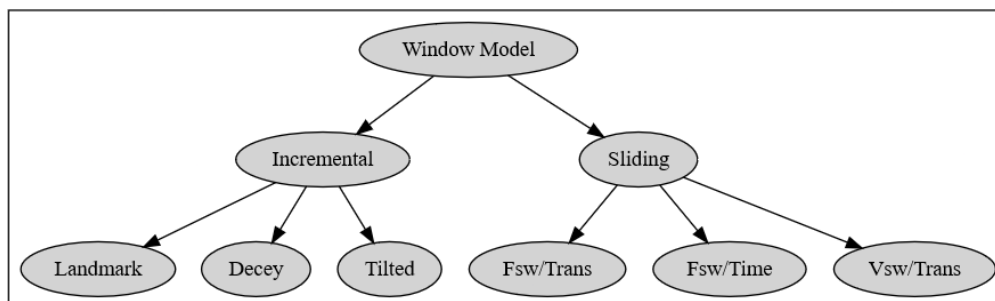


FIGURE 5 – Hiérarchisation des attributs Window Model

Pour visualiser les corrélations entre la période de publication et les autres attributs, hiérarchisés ci-dessus, des différents algorithmes. Nous avons établi l'évolution des différentes classes hiérarchique (Support, Utility, Uncertain, Incremental, Sliding) au fil du temps, après avoir réalisé un ensemble de calcul basé sur le nombre de publications et cette hiérarchie des attributs.

Nous avons choisi ce niveau de hiérarchie, car, d'une part les attributs qui représentent le sommet composé des classes Mining framework, Approach et window Model sont possédés par tous les algorithmes quelle que soit la période. D'autre part, les feuilles des hiérarchies Mining framework et Window Model sont très nombreuses, ceux qui donnent une représentation des résultats pas très explicite d'où l'importance de la hiérarchisation. Les figures suivantes illustrent nos résultats (les résultats intermédiaires sont disponibles sur la feuille de calcul) :

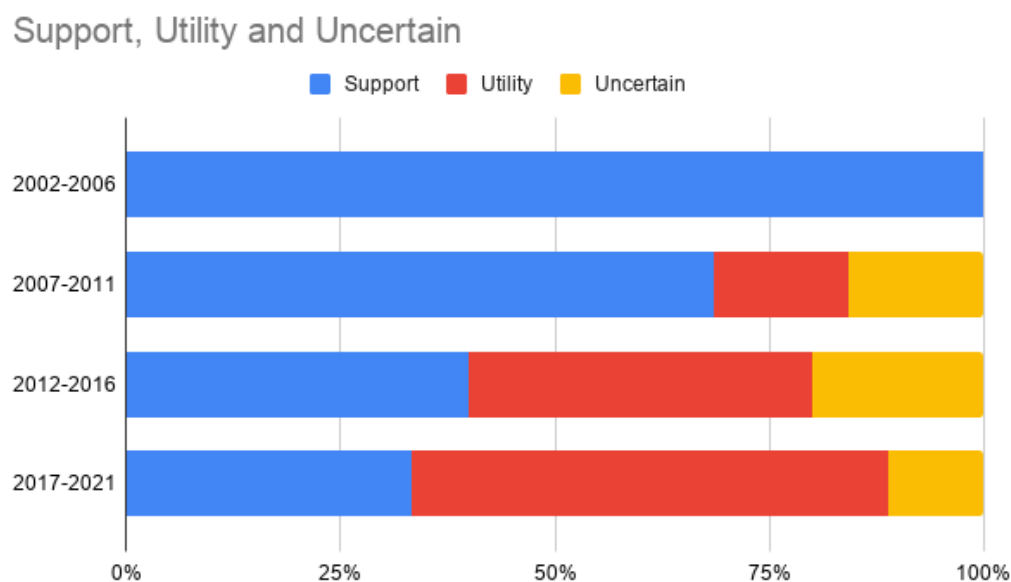


FIGURE 6 – Évolution au cours du temps du nombre total d'occurrences des attributs Mining Framework

Apriori, FP-Growth, Hybrid and Other

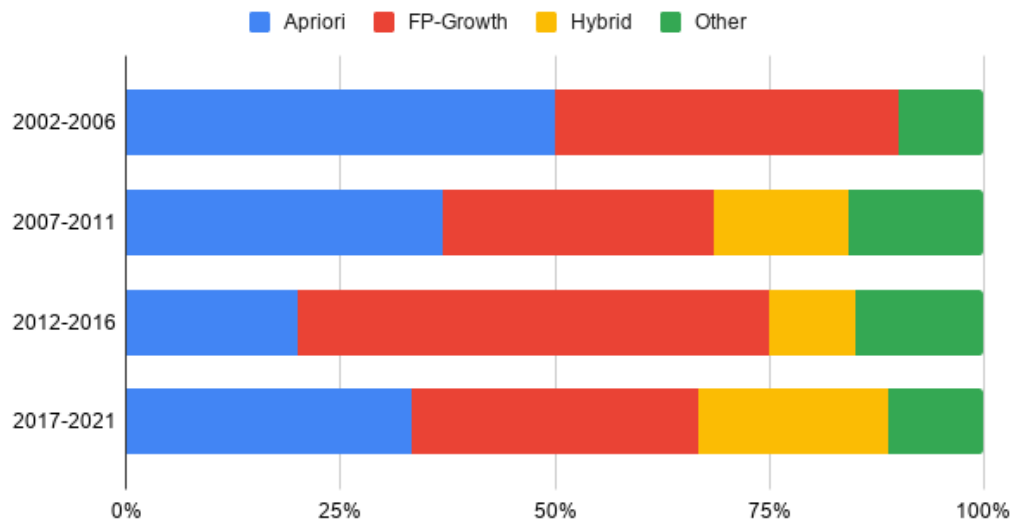


FIGURE 7 – Évolution au cours du temps du nombre total d'occurrences des attributs Approach

Incremental and Sliding

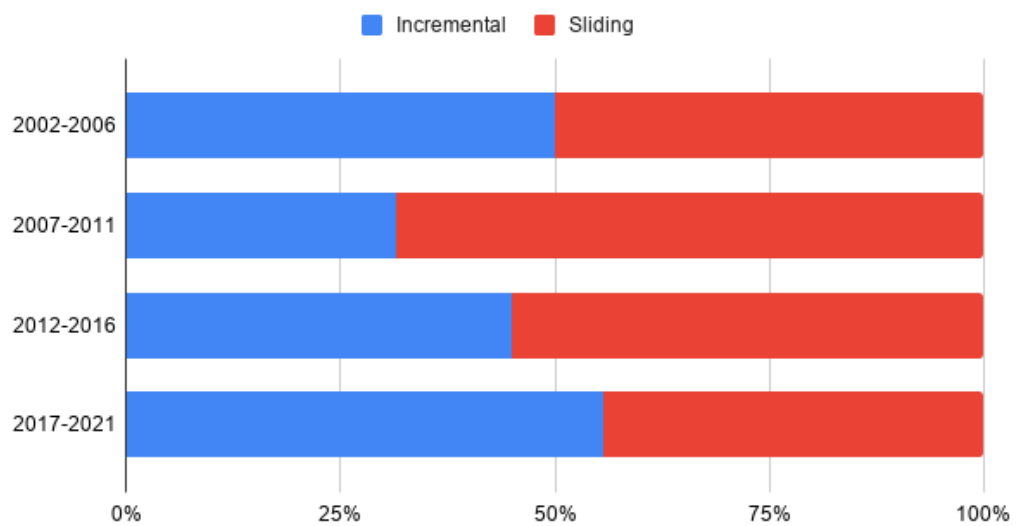


FIGURE 8 – Évolution au cours du temps du nombre total d'occurrences des attributs Windows Model

3 Application de l'outil CoGui

Les données qui ont servi à l'utilisation de CoGui sont en partie disponibles [ici](#) (google spreadsheet) ou au complet [ici](#) (dépot gitlab.com)

3.1 Les tableaux binaires

Tout d'abord, nous avons reporté les valeurs fournis sur les tableaux 1, 2 et 3 présents dans l'article de Dame Samb [SSN21] sur une première feuille de calcul nommé **Table 1 v2**, **Table 2 v2**, **Table 3 v2**.

Puis nous avons effectué l'étape des transformations explicitées dans le chapitre 2 sur la transformation dans la feuille de calcul nommé **Classification / Fusion v2** qui contient les mises à jour des données présentes sur moodle.

Puis nous avons exporté les résultats au format CSV et importé dans une base de données H5 depuis le menu contextuelle de CoGui selon la démarche décrite dans le document **TP-COGUI-RCA3**.

Voici le résultat depuis l'interface de CoGui :

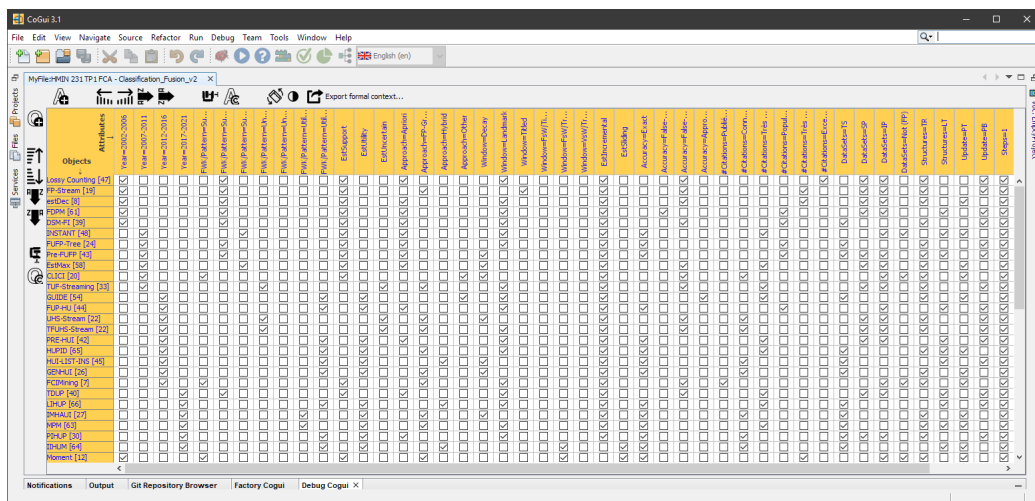


FIGURE 9 – Capture d'écran du projet CoGui complet

3.2 Data Mining

Étant donné la complexité exponentielle de la représentation sous forme de treillis, nous avons décidé de l'omettre du rapport, cependant, vous la retrouverez dans le dépôt git associé.

Avant de nous lancer dans la représentation des données, rappelons les définitions des représentations suivantes :

- Trellis : "Every concept pair has a lower bound and an upper bound". Autrement dit, nous avons un ordre partiellement ordonnées entre tous les concepts (extension) et les attributs (intension),

- AOC-Poset : C'est une simplification du treillis, dont il faut retirer les concepts qui n'introduisent pas d'attributs, ni d'objets,
- AC-Poset : C'est une seconde simplification de l'AOC-Poset qui répète les concept-objet,
- Iceberg : C'est une représentation partielle des concepts selon un certain seuil qui permet d'observer des tendances.

Les représentations au format *DOT* et *PNG* ne sont pas inclus dans le rapport dû au manque de visibilité des images, mais sont disponibles dans le dépôt git mentionnées précédemment.

Deuxième partie

Analyse des résultats

3.3 La comparaison des structures

On observe bien l'explosion de concept et d'arête sur le treillis possédant une complexité exponentielle dans le pire des cas par rapport à l'AOC-Poset.

Nous ne pouvons pas comparer l'iceberg parce qu'il n'est pas construit sur la même idée de départ.

Voici les chiffres des différentes structures (le nom du fichier contexte ayant servi au calcul n'est pas pertinent et a été remplacé par des "...") :

- **Treillis** :
 - context : ... (58 x 49)
 - built with algo : AddExtent
 - concepts count : 2190
 - maximals : 1
 - minimals : 1
 - edges count : 7995
 - max incoming degree : 24
 - max outgoing degree : 58
- **AOC-Poset**
 - context : ... (58 x 49)
 - built with algo : Ares
 - concepts count : 67
 - maximals : 1
 - minimals : 1
 - edges count : 155
 - max incoming degree : 15
 - max outgoing degree : 38
- **AC-Poset**
 - context : ... (58 x 49)
 - built with algo : Ares
 - concepts count : 101
 - maximals : 1
 - minimals : 58
 - edges count : 516
 - max incoming degree : 29
 - max outgoing degree : 11
- **Iceberg (30%)**
 - context : ... (58 x 49)
 - built with algo : Iceberg
 - concepts count : 46
 - maximals : 1
 - minimals : 20
 - edges count : 89
 - max incoming degree : 24
 - max outgoing degree : 9

3.4 Grandes catégories et sous-catégories

Nous observons la grande catégorie *Steps 3 et 5* possédant la sous-catégorie *Steps 1, 3, 5* ; *Steps 1, 2, 3, 5* et *Steps 3, 4, 5*.

De plus, la catégorie *Steps 1, 2, 3, 5* possède la sous catégorie *Datasets=IP*.

3.5 Les règles d'implication

Dans un premier temps, nous avons cherché des implications binaires à partir des données faites à la main à l'aide de l'outil de tableau (voir la feuille de calcul nommé "Intersection des catégories").

Nous avons observé un résultat intéressant : Tous les algorithmes présent entre 2002 et 2006, présentent seulement l'attribut de type Support, que nous avons précédemment illustré sur la figure 6.

Le résultat précédent est retrouvé dans les règles Duquennes-Guigues et l'AC-Poset généré à l'aide de CoGui.

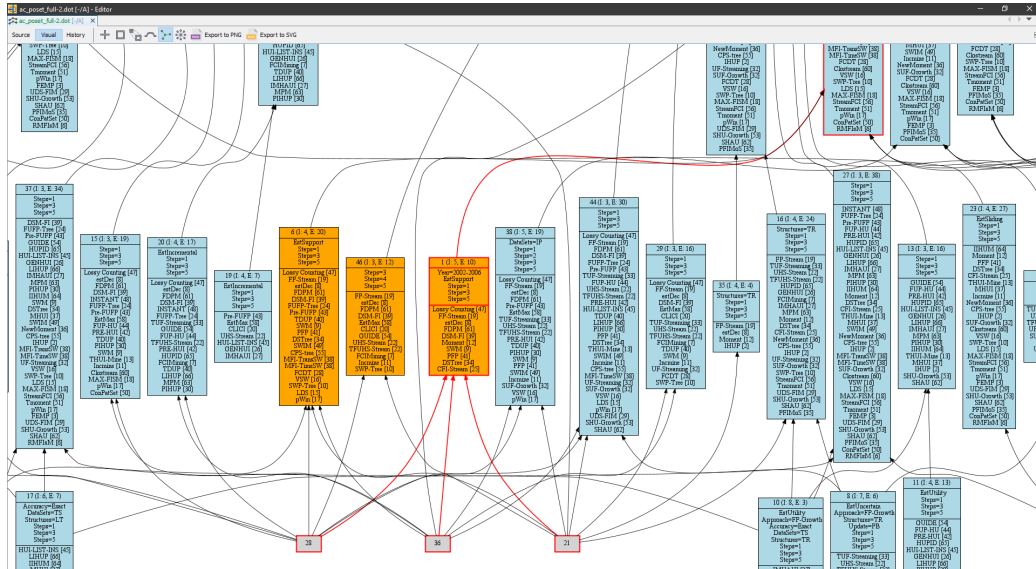


FIGURE 10 – Capture d'écran de l'AC-Poset autour du concept 2002-2006

Voici les 10 algorithmes que nous observons :

- Loosy Counting [47]
- FP-Stream [19]
- estDec [8]
- FDPM [61]
- DSM-FI [39]
- Moment [12]
- SWM [9]
- PFP [41]
- DSTree [34]

— CFI-Stream [25]

Cette implication est bien visible par la règle qui concerne 10 éléments et nous renseigne qu'entre 2002 et 2006 un maximum de 10 algorithmes sont de types Support :

— $\langle 10 \rangle \text{ Year}=2002-2006, \text{Steps}=1, \text{Steps}=3, \text{Steps}=5 \Rightarrow \text{EstSupport}.$

Nous avons une autre règle de niveau supérieur assez simple :

— $\langle 24 \rangle \text{ Approach}=\text{FP-Growth}, \text{Steps}=1, \text{Steps}=3, \text{Steps}=5 \Rightarrow \text{Structures}=\text{TR}.$

Cette règle nous apprend que 24 algorithmes utilisant l'approches FP-Growth, utilisent une structure TR.

3.6 Les exclusions mutuelles

Nous avons une exclusion mutuelle entre l'attribut 2002-2006 et le reste des types de Mining Framework.

Nous une seconde exclusion mutuelle entre les algorithmes ayant une approche classique et possèdent une phase de pruning (step 4) qui produise des faux positif (false-positive) excepté le seul algorithme [54] ayant une approche non classique (other) qui produit une précision *approximative*.

Troisième partie

Conclusion

3.7 Compréhension des données

Pour conclure, la première partie de ce rapport concerne l'article de Dame Samb qui nous a fourni un jeu de donnée composé de 58 algorithmes pertinents dans le domaine du Pattern Mining over Data Stream (PMDS).

Nous avons transformé et hiérarchisé les données. Cette transformation était assez longue et fastidieuse. Afin de pouvoir les manipuler avec l'outil CoGui.

Ce dernier nous a permis de calculer plusieurs représentations abordées en cours, treillis, aoc-poset, ac-poset, iceberg. Le logiciel c'est avéré très simple d'utilisation, pour l'importation ou l'exportation des données, la prise en charge de Git est un plus non-négligeable.

3.8 Analyse des résultats

En dernière analyse, nous avons procédé manuellement à l'aide du tableur à la recherche de propriétés pertinente à présenter dans ce rapport. Nous avons abouti à un résultat présenter la figure 6 qui nous a permis le report et la vérification des valeurs et la recherche d'une représentation explicite.

Ensuite, la comparaison des structures nous a permis de visualiser les ordres de grandeurs des différentes structures.

De surcroît, les règles d'implications permettent aussi bien de déduire des connaissances pratiques.

En somme, toutes les informations précédentes sont utilisées pour déduire des exclusions mutuelles que nous avons présentés à la fin.

Références

- [FPS96] FAYYAD, U., PIATETSKY-SHAPIO, G. et SMYTH, P. « From Data Mining to Knowledge Discovery in Databases ». In : *AI Mag.* t. 17 (1996), p. 37-54.
- [SSN21] SAMB, D., SLIMANI, Y. et NDIAYE, S. *Variability Analysis on Pattern Mining Algorithms over Data Streams*. 2021.