Tom Simula, Daniel Azevedo Gomes, Sebastien Prud'homme Gateau

Rapport de TP : Extraction de la variabilité dans des familles de produits interconnectée

Introduction

L'objectif de ce TP est d'extraire des relations de variabilité entre différentes caractéristiques.

Pour cela nous avons suivi un processus de fouille de données tel que décrit dans le cours afin d'établir un feature model et extraire les relations liant nos différentes données.

Git : https://github.com/dneial/TP_PCM

Nos données

Nous avons choisi de travailler avec les données fournies en consigne, à savoir la description des logiciels de comptabilité disponible sur moodle : https://moodle.umontpellier.fr/mod/folder/view.php?id=748813

reprenant les comparaisons effectuées sur wikipedia à cette adresse : https://en.wikipedia.org/wiki/Comparison_of_accounting_software

Parmi toutes les sources proposées nous nous sommes concentrés sur les liens entre les données provenant des répertoires :

- Comparison_of_accounting_software_0_raw (logiciels de comptabilité)
- Comparison_of_operating_systems_1_raw (systèmes d'exploitation)
- Comparison_of_free_and_open-source_software_licenses_1_raw (licences logicielles)

Le choix de ces répertoires a été motivé par la présence de relations facilement exploitables et de taille raisonnable. Les données des logiciels contiennent des informations multiples, tels que les systèmes d'exploitation pour lesquels ils ont été développés et les licences logicielles qu'ils utilisent.

Ces deux points ont chacun leur propre table que l'on va pouvoir exploiter au cours de notre fouille.

D'autres informations comme le langage de programmation utilisé ou le marché cible ont été conservées mais ne font pas l'objet d'une recherche de relation par faute de temps.

Nettoyage

Nous avons pour chaque jeu de données, réalisé un nettoyage sur mesure, soit:

Logiciels de comptabilité:

- Caractéristiques :
    - Création de nouvelles caractéristiques pour chaque valeur trouvée dans les colonnes "*Market Focus*", "*Structure*" et "*Language*" afin de n'avoir que des 0 ou 1 (binarisation).
    - Suppression des caractéristiques avec des champs vides ou avec "?"

Système d'exploitation:

- Lignes :
    - On a choisi de garder les lignes : "*Linux*", "*Windows (NT family)*" et "*Mac Os Classic*", car ce sont les seuls systèmes d'exploitation présents dans les données de logiciels de comptabilité.
- Caractéristiques :
    - On garde seulement "*Computer architecture supported*", "*Kernel type*", "*File system supported*" afin de limiter la quantité de données pour faciliter l'extraction d'information
    - Binarisation des caractéristiques des colonnes "*Computer architecture supported*", "*Kernel type*" et "*File system supported*".

Licences logicielles:

- Lignes :
    - On garde "*Apache License 2.0*", "*GNU General Public License v2*", "*GNU Lesser General Public License*", "*Mozilla Public License 2.0*", "*Common Public License*" et "*GNU Affero General Public License*" car ce sont les seules licences présentes dans les données de logiciels de comptabilité

Après nettoyage on se retrouve avec les 3 tables de nos produits (Contextes Formels).

Pour pouvoir les associer à l'aide de l'outil FCA4J, on a besoin de créer deux tables supplémentaires (contextes relationnels) qui serviront de liaison entre les contextes formels susnommés.

À la fin de cette étape nous avons donc :

3 contextes formels :

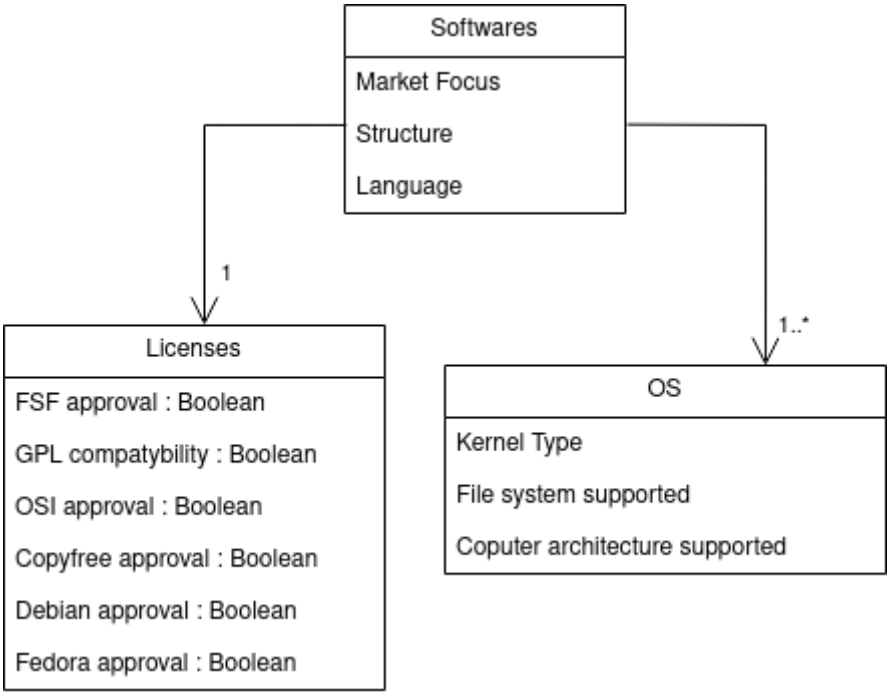
- une table contenant 18 Logiciels de comptabilité avec leurs caractéristiques (fichier : clean_soft.csv)
- une table contenant 3 Systèmes d'exploitation avec leurs caractéristiques (fichier : clean_os.csv)
- une table contenant 7 Licences open source avec leurs caractéristiques (fichier : clean_license.csv)

2 contextes Relationnels

- reliant les logiciels aux systèmes d'exploitation sur lesquels ils peuvent s'exécuter (fichier : software2os.csv)
- reliant les logiciels aux licences qu'ils utilisent (fichier : software2license.csv)

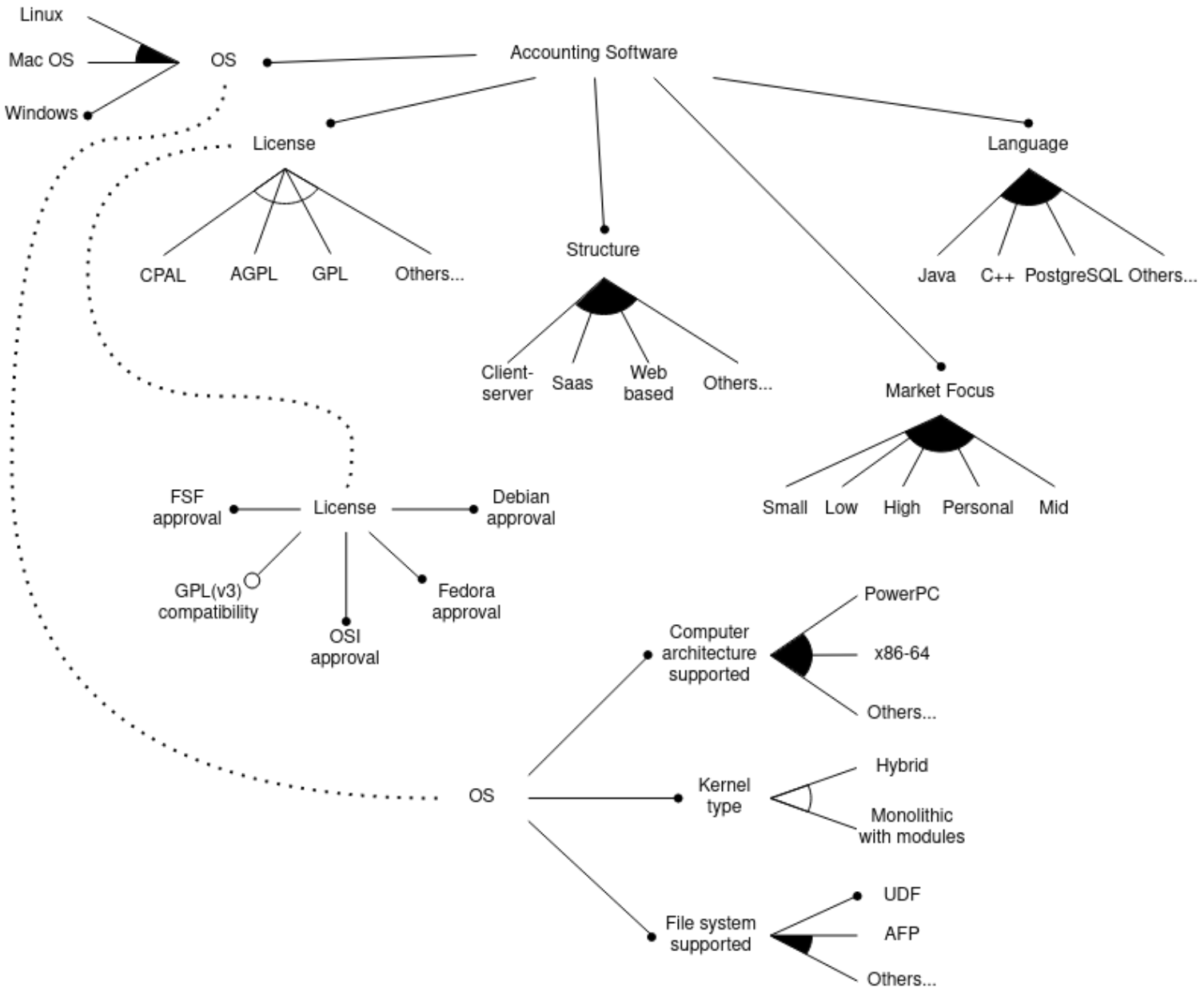
Modèle UML et Feature Model

Nos données peuvent être représentées sous la forme d'un modèle UML semblable à celui-ci :



Les contextes formels peuvent être associés aux classes de notre modèle UML et les contextes relationnels sont représentés par les relations d'association qui les lient.

Nous avons également créé un Feature Model suite à l'analyse des données :



Nous pouvons constater sur le feature model l'apparition répétée de la caractéristique "Others…". Celle-ci est simplement utilisée pour regrouper les autres valeurs quand elles sont trop nombreuses pour pouvoir être représentées. De cette manière, nous pouvons avoir un feature model plus épuré.

Extraction d'information

Grâce à l'outil FCA4J on peut générer les différents treillis de concepts, qui peuvent déjà nous apporter quelques informations sur les liens entre nos éléments.

Ils sont consultables dans l'archive rendue ou via le lien suivant :

https://github.com/dneial/TP_PCM/blob/master/Results/results_hermes/resultat_hermes.pdf

Dans chacun de ces documents on peut observer trois treillis mettant en relation nos données.

Les cases les composant se divisent en trois parties, de haut en bas :

- le nom du concept que cette case représente, et qui sera utilisé pour le référencer par la suite.
- les caractéristiques propres à ce concept.
- les objets impliqués dans ce concept : ils possèdent donc les caractéristiques notées dans cette case et héritent de toutes celles des concepts supérieurs auxquels ils sont liés.

Les treillis obtenus sont les suivants, de gauche à droite :

- Un treillis liant les différents systèmes d'exploitation étudiés aux caractéristiques que l'on a sélectionnées. On y comprend par exemple que les systèmes n'ont pas les mêmes spécificités en matière de gestion de fichiers et d'architectures supportée, ils se mettent néanmoins tous d'accord sur le système de fichiers "*UDF*".

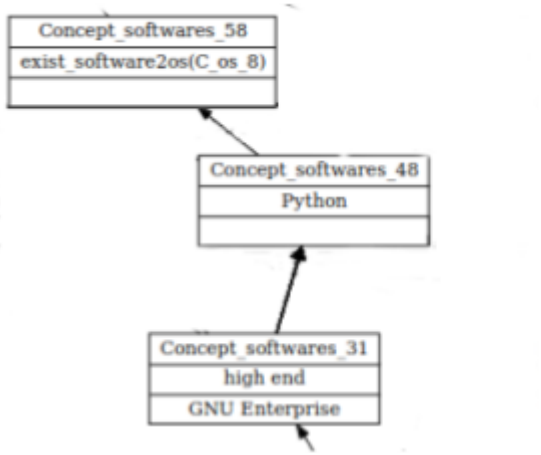
- Un treillis liant les licences, ici on note par exemple la présence de la caractéristique "*Copyfree approval*" à la racine du graphe, signifiant qu'aucune des licences gardées ne la possède.

- Enfin, le treillis faisant la jonction entre les précédents et les 18 logiciels de compatibilité étudiés.
    - Le lien entre les logiciels et le treillis des OS se retrouve partout où une caractéristique ***"exist_software2os"*** apparaît, prenant en argument le nom du concept ciblé dans le treillis des OS. Elle signifie que tout logiciel lié à ce concept peut tourner sur le système d'exploitation concerné par l'argument.

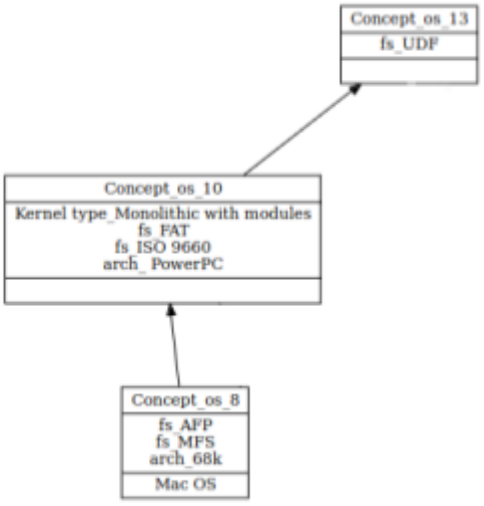
      Par exemple, le logiciel ***GNU Enterprise*** que l'on retrouve dans la case ***concept_softwares_31*** est lié à l'attribut ***exist_software2os(C_os_8)*** (dans la case *concept_softwares_58*).

C_os_8 correspond au concept ***Concept_os_8*** dans le premier treillis, grâce à ce lien, on sait que ***GNU Enterprise*** fonctionne notamment sur des postes ayant ***Mac OS***, une architecture ***68k*** et supporte les systèmes de fichiers ***AFP*** et ***MFS***.

Mais, par extension de cette relation, il hérite également des caractéristiques des concepts supérieurs à ***Concept_os_8***.



GNU Enterprise dans le treillis des logiciels.



Le Concept OS 8 dans le treillis des Systèmes d'exploitation

- Le lien entre les logiciels et le treillis des Licences utilise la même logique, et se fait par la caractéristique "***exist_software2license***"

De ces treillis de concepts on peut retirer quelques informations globales, comme le fait que tous les logiciels sont utilisables sur Windows et que seuls deux d'entre eux ne fonctionnent pas sur les trois systèmes d'exploitation à la fois.

Il est cependant visuellement compliqué de déduire plus de liens remarquables avec pour seule source le treillis généré.

Pour aller plus loin, FCA4J nous permet de déduire un certain nombre de règles avec la commande RuleBasis. Des implications générées avec nos données nettoyées, voici quelques exemples parmi les plus intéressantes :

Web based => mid : Tout software web-based vise un marché de moyenne taille

Client-server => mid : Tout software client-server vise un marché de moyenne taille

> mid,Java => Web based : Tout software codé en Java visant un marché de taille moyenne est web-based
>
> Python => mid : Tout software codé en Python vise un marché de taille moyenne
>
> low => mid : Tout software visant un marché de petite taille vise aussi un marché de taille moyenne

Cependant, si nous voulons des implications entre deux tables différentes, nous devons les trouver à la main, par exemple:

> Web based => Linux : Tout software web-based fonctionne sur Linux (Concept_softwares_45 -> Concept_softwares_57)
>
> Python => Mac OS : Tout software codé en python fonctionne sur Mac OS (Concept_softwares_48 -> Concept_softwares_58)

Nous avons été sélectifs sur la sélection car toutes les règles ne sont pas source d'information. En effet, certaines règles générées prennent beaucoup de caractéristiques comme hypothèse ou d'autres sont évidentes.

De plus, ces règles sont efficaces seulement sur notre jeu de données. Nous ne pouvons pas les généraliser car les données ne sont pas exhaustives, il est donc possible de trouver un contre-exemple qui enfreindrait une ou plusieurs de nos règles.

Conclusion

En résumé, notre processus de fouille de données nous a permis de mettre en lumière les relations entre nos sujets d'étude de manière claire et visuelle via la création de différents supports tels que le modèle UML, le feature model ainsi que le treillis de concepts (RCA) et la base de règles grâce à FCA4J.

Nous nous attendions à trouver des liens plus subtils entre les données, par exemple qu'un système d'exploitation implique le choix d'une licence. Cela peut s'expliquer par la nature de nos données qui sont en réalité très peu discriminantes. On remarque notamment que presque tous les logiciels tournent sur les 3 systèmes d'exploitation.

Maintenant le processus de fouille établie, on pourrait, pour aller plus loin et obtenir davantage d'information, élargir notre base de données grâce à celles proposées dans le sujet.