Dérivation d'ontologies à partir de données brutes et zoom sémantique par "analyse de concepts formels"

Sofiene Chamakhi

sofiene.chamakhi@etu.univ-lyon1.fr
Université Claude Bernard Lyon 1,
http://www.univ-lyon1.fr/

Résumé Ce document concerne l'analyse de données crues de capteurs afin de dériver des schémas terminologiques qui sont vérifiés par des triplets RDF. L'objectif est de catégoriser les données brutes en se basant sur des classes et leurs attributs dont la structure est inférée par analyse de concepts formels (ACF). L'objectif d'une telle analyse est de constituer un schéma terminologique pour ces données qui puisse être utilisé pour raisonner sur les propriétés implicites de ces données. Des approximations permettent de regrouper des données selon leur degré de ressemblance, avec comme objectif d'obtenir un échelonnage conceptuel ou un zoom sémantique.

Mots-clefs : Extraction de connaissances ; analyse de données ; analyse de concepts formels ; raisonnement automatique ; données massives ; gestion d'environnement ; villes intelligentes.

Abstract This document concerns the analysis of raw sensor data in order to derive terminological schemas that the generated RDF triples abide by. The objective is to categorize these raw data using classes and attributes fitting their structure inferred using Formal Concept Analysis (FCA). The outcome of such an analysis is to constitute a terminological schema for this data that could be used for reasoning about implicit properties of such data. Approximations are used to aggregate similar data in order to do conceptual scaling or semantic zoom.

Keywords: Knowledge Extraction; Data Analytics; Formal Concept Analysis; Automated Reasoning; Big Data; Environment Monitoring; Smart Cities.

Table des matières

1 Introduction

De nos jours, il existe plusieurs sources de données brutes qui ne respectent pas une classification ou un schéma terminologique. Par exemple, parmi les données publiques fournies par la métropole de Lyon, nous trouvons plusieurs données résultantes de mesures et de capteurs, mais qui manquent d'interprétation. Il existe plusieurs approches d'analyse qui permettent de dériver des ontologies et des connaissances. Nous avons passé en revue quelques approches, parmi les plus remarquables, afin de les adapter à notre besoin d'analyse des données publiques de la métropole de Lyon. L'objectif est d'extraire une définition intensionnelle qui permet de découvrir des propriétés logiques qui caractérisent ces éléments. Les données brutes sont alors mises dans une forme plus sémantique qui puisse être exploitée à des fins plus utiles.

Dans ce document, nous expliquons nos choix concernant les approches (notamment *Formal concept analysis*) et les technologies choisies. Nous détaillons aussi notre méthode de regroupement de données et le zoom sémantique qui permet d'interpréter intelligemment les données analysées.

Le présent document est structuré de la manière suivante :

Dans la Section 2, nous présentons quelques approches, parmi les plus remarquables, permettant de dériver des ontologies et des schémas terminologiques. Ensuite, dans la Section 3, nous justifions nos choix et nous décrirons notre approche globale. La Section 4 permet de détailler l'architecture générale de notre approche ainsi les implémentations réalisées, alors que la Section 5 est consacrée aux expérimentations. Enfin, nous trouvons dans la Section 6 une conclusion pour nos travaux ainsi que les perspectives.

2 Etat de l'art

Dans cette section, nous allons présenter quelques approches, parmi les plus remarquables, permettant de dériver des ontologies et des schémas terminologiques à partir d'un jeu de données. Le but est de déterminer l'approche qui convient le mieux à notre besoin d'extraction de connaissances à partir de données brutes.

2.1 Formal concept analysis (FCA)

Présentation et utilisation de FCA Formal Concept Analysis (FCA) [9] est une méthode mathématique d'inférence introduite par Rudolf Wille en 1982 en tant qu'une application de la théorie des treillis de Galois. FCA permet d'analyser des données brutes et de trouver des attributs communs entre objets, et ce, afin de pouvoir dériver des taxonomies de concepts.

Concrètement, FCA repose sur la notion de contexte formel. Un contexte peut être défini comme une matrice, ou comme un ensemble de triplets $C(O, A, R)$, où O est un ensemble d'objets, A un ensemble d'attributs et R une relation binaire entre les objets et les attributs. Des concepts formels sont dérivés à partir d'un contexte. Un concept est un ensemble d'objets et un ensemble d'attributs,

où chaque objet appartenant à ce concept possède tous les attributs, et où chaque attribut est partagé par tous les objets. Une hiérarchie de concepts est alors construite sous forme de treillis.

Le tableau 1 illustre un contexte formel. Chaque ligne représente un objet (dans ce cas un personne) et chaque colonne représente un attribut (le genre de la personne ou une passion). FCA a permis de construire le treillis de Figure 1. Chaque nœud (cercle) du treillis est un concept formel constitué d'un ensemble d'objets (tous les objets de ce nœud ou des nœuds fils) et d'un ensemble d'attributs (tous les attributs de ce nœud ou des nœuds parents). Le nom d'un objet est indiqué dans un rectangle avec un fond blanc, alors que le nom d'un attribut est indiqué dans un rectangle grisé.

Table 1. Un exemple de contexte formel

	Homme	**Femme**	**Voitures**	**Sport**	**Mode**	**Voyages**
Julie		X			X	
Jean	X					X
Marie		X		X		
Michel	X		X	X		

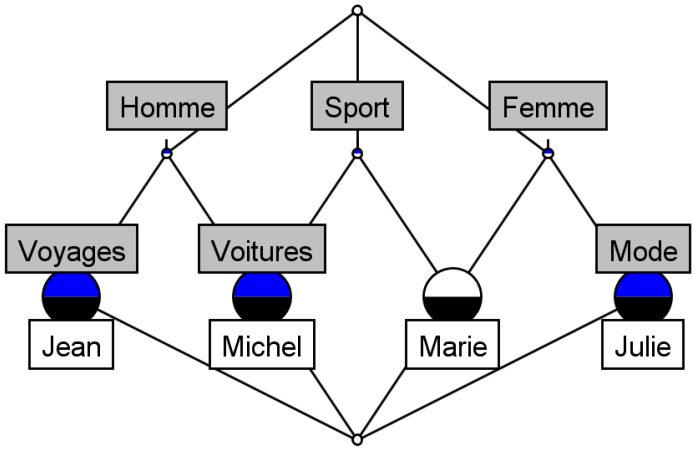

Figure 1. Treillis de concepts formels

Plusieurs techniques et outils se sont basés sur FCA afin d'inférer des ontologies.

Dans l'article [8] de Dau et Sertkaya, nous pouvons distinguer deux contributions majeures. La première consiste en l'application de FCA sur des triplets

RDF [1]. Cette application est intéressante vu l'utilisation prépondérante de RDF pour le web sémantique. La deuxième s'intéresse à l'utilisation de FCA sur des contextes multi-valeurs. En effet, à l'origine, FCA était destinée à manipuler uniquement des contextes binaires. Etendre l'utilisation de FCA sur des contextes multi-valeurs représente une évolution importante, dans la mesure où cela permet de supporter plusieurs types de données (chaînes de caractères, numériques, dates ...), de regrouper plusieurs attributs et ainsi minimiser le contexte et favoriser le passage à l'échelle. Une technique appelée "échelonnage conceptuel" (conceptual scaling) est l'une des étapes utilisées par les auteurs lors de la manipulation de contexte multi-valeurs. Elle consiste à définir des seuils permettant de regrouper ou de dissocier des attributs. Un contexte mutli-valeurs peut être ainsi converti vers un contexte binaire afin d'en extraire le treillis de concepts.

Figure 2 montre à gauche un treillis de concepts classique, où chaque objet a un attribut ayant le même numéro (l'objet numéro 1 a un attribut nommé 1 et ainsi de suite). Le treillis à droite est obtenu après l'application d'un échelonnage conceptuel, où une comparaison par rapport à l'entier 3 est effectuée, ce qui permet de regrouper certains objets dans le même concept. Le nombre total de concepts se trouve ainsi réduit.

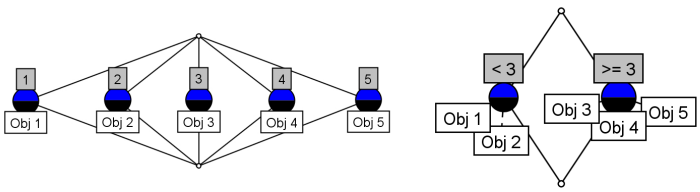

Figure 2. Treillis FCA classique (gauche) et Treillis FCA avec échelonnage conceptuel (droite)

D'autres auteurs se sont intéressés à l'application de FCA sur des données du web sémantique [14]. Contrairement aux données d'autres travaux (tels que [8]) basés sur FCA, les ensembles de données, provenant du web sémantique, ont la particularité d'être largement plus grands. Les auteurs discutent alors du passage à l'échelle et de le performance de FCA.

Des approches, plus spécifiques, utilisent FCA pour extraire des connaissances à partir de requêtes SPARQL [2] [5], [1]. FCA est utilisée pour organiser et hiérarchiser les réponses aux requêtes, ce que ne permettent pas de faire les formats classiques des réponses (généralement XML/RDF, JSON, CSV, HTML ...).

[1] http://www.w3.org/RDF/
[2] http://www.w3.org/TR/rdf-sparql-query/

FCA a été aussi utilisée pour fusionner des ontologies [23], [7]. Le but est de découvrir de nouveaux concepts qui n'existaient auparavant dans aucun des concepts d'origine. De la même manière, ceci permet de supprimer des concepts redondants.

Haav propose une méthode semi-automatique pour la construction d'ontologies [10]. En premier lieu, elle utilise FCA sur un contexte formel, puis, elle transforme l'ontologie obtenue en des expressions de logique du premier ordre. Le but est d'obtenir une nouvelle ontologie basée sur des clauses de Horn, pouvant être exécutée ensuite dans des progammes Prolog.

FCA a aussi été utilisée pour effectuer du raisonnement sur des textes. Selon [18], FCA peut être employée pour construire des ontologies linguistiques, ou encore, pour comparer ou fusionner des bases de données lexicales.

Une autre approche pour l'extraction de taxonomies à partir de textes est proposée par Cimiano *et al.* [6]. L'approche repose sur les composants TreeTagger [20] et LoPar [21] afin de parcourir le texte et marquer les mots afin d'obtenir un arbre pour chaque phrase. Des paires (sujet - verbe), (verbe - objet) ou (verbe - proposition) sont extraits. Tous les mots sont analysés lexicalement et mis sous forme normale correspondant à leurs racines lexicales. Enfin, FCA est appliquée sur les arbres obtenus afin d'obtenir des ontologies.

Outils FCA Dans cette section, nous présentons quelques outils de raisonnement basés sur FCA.

ToscanaJ [2] [3] est un outil basé sur FCA développé en Java, permettant de manipuler des contextes binaires, multi-valeurs ou même imbriqués. La particularité de ToscanaJ réside dans des composants lui permettant d'être connecté à un système de gestion de bases de données ou un moteur reposant sur un langage de requêtes SQL.

Lattice Miner [4] est lui aussi développé en Java, Open Source, et permet de manipuler tous les types de contextes, de retourner un rendu graphique du treillis mais aussi des règles d'association sur les données. L'utilisateur peut spécifier le support et la confiance pour ces règles.

D'autres logiciels sont similaires, à l'instar de Concept Explorer (ConExp) [5] ou de Galicia [24] [6]. Contrairement aux autres outils qui n'utilisent qu'un seul algorithme spécifique (à moins de modifier le code source), Galicia propose par défaut différents algorithmes FCA.

OpenFCA (Confexplorer) [7] est un outil en ligne développé en C# et en Flex permettant de lancer une analyse FCA directement sur un navigateur.

Il existe d'autres programmes, plus basiques, comme FCA Stone [8], développé en Perl, qui permet de convertir le format des fichiers FCA utilisés par les autres

[3] http://toscanaj.sourceforge.net/
[4] http://lattice-miner.sourceforge.net/
[5] http://conexp.sourceforge.net/
[6] http://galicia.sourceforge.net/
[7] https://code.google.com/p/openfca/
[8] http://fcastone.sourceforge.net/

outils, ou encore, In-Close [9] qui est un outil basique reposant uniquement sur une console de ligne de commande.

Après une période initiale de familiarisation avec les principaux outils FCA, nous avons pu dresser une classification utilisant la méthode FCA elle-même [10].

Une classification des outils FCA est proposée dans Figure 3 selon les principales fonctionnalités de ces outils, comme le support de contextes binaires ou multi-valeurs, la production de treillis sous forme graphique, la gestion des treillis imbriqués, la connexion avec une base de données, le langage du code source, le calcul des règles d'association ...

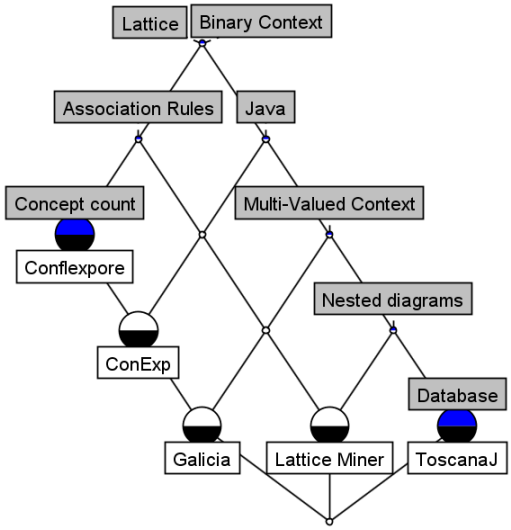

Figure 3. Classification des outils FCA par l'approche FCA

2.2 Relational Concept Analysis (RCA)

L'approche RCA [11], [4] est une extension de FCA. Elle vise à pallier une faiblesse de FCA qui ne prend en compte que le contexte et sa matrice binaire, alors que RCA prend en compte aussi les relations entre les objets eux-mêmes. Le contexte peut être alors considéré comme un modèle relationnel. RCA fournit des descriptions relationnelles permettant d'être exploitée par des formalismes de représentation comme la logique de description par exemple.

RCA repose sur une approche itérative. Un treillis est construit pour chaque contexte indépendamment de toute relation entre les objets. Les relations sont prises en compte une par une jusqu'à l'obtention du dernier treillis.

[9] http://inclose.sourceforge.net/

[10] Ceci nous a paru une démonstration intéressante de l'utilité de FCA en soi.

Comme applications possibles de RCA, les auteurs citent l'exemple de factorisation de diagrammes UML [11], ou encore, l'extraction d'architectures dans des logiciels orientés-objet.

2.3 Approches déterministes

La fouille de textes est une approche déterministe d'extraction de connaissances. Le sujet a été traité par Maedche et Staab qui proposent un mécanisme de découverte de relations entre des concepts [16]. L'outil repose sur les opérations d'extractions lexicales conventionnelles comme la *tokenization* (découpage du texte en mots), la mise des mots en forme de base ou encore sur une analyse syntaxique de surface, le tout afin d'obtenir des annotations XML sur les mots. Afin d'extraire une taxonomie, deux méthodes sont possibles :

- Utilisation d'algorithmes reposant sur les expressions régulières [12].
- Utilisation d'un algorithme [22] de découverte de règles d'association.

2.4 Approches non-déterministes

Dans ce qui suit, nous allons présenter quelques approches non-déterministes pour l'extraction de connaissances.

Ensembles flous (Fuzzy sets) Les ensembles flous [25] ont été introduits par Lotfi Zadeh en 1965. Cette approche permet de représenter mathématiquement l'imprécision relative aux données. Contrairement aux relations binaires où il existe deux valeurs possibles, les ensembles flous modélisent les données avec une précision variant dans un intervalle allant de 0 à 1.

Un état de l'art sur les différents travaux qui ont traité la logique floue a été réalisé par Belohlavek *et al.* [3].

D'autres auteurs s'intéressent à l'application de FCA sur la logique floue [19]. Cette approche est divisée en trois principales étapes :

- Obtenir un treillis de concepts à partir de données floues. Un seuil de confidence peut être utilisé afin d'éliminer les données avec un degré de précision faible.
- Effectuer un regroupement (*clustering*) sur les concepts.
- Hiérarchiser les concepts afin d'obtenir un treillis final.

Ensembles grossiers (Rough sets) Les ensembles grossiers [17] sont une approche mathématique d'extraction de connaissances qui a été introduite par Pawlak en 1982. Contrairement aux ensembles flous qui traitent des données imprécises, l'objectif des ensembles grossiers est d'analyser des données imparfaites et inconsistantes. Les ensembles grossiers reposent sur des approximations

[11] http://www.uml.org/

basées sur une borne supérieure et une borne inférieure. Les approximation permettent par exemple de vérifier l'égalité ou l'inclusion d'ensembles de données, ou encore, de pallier des incohérences au niveau des dépendances fonctionnelles dans une base de données relationnelle.

Les domaines d'application sont l'intelligence artificielle, le raisonnement inductif, la classification automatique, les algorithmes d'apprentissage ...

Bien qu'ils soient pertinents, les ensembles grossiers ne peuvent être utiles pour notre approche, dans la mesure où nous traitons des données brutes (issues de mesures de capteurs), non-structurées et qui contiennent pas d'incohérences.

2.5 Apprentissage automatique

L'apprentissage automatique (*Machine Learning*) est une discipline scientifique visant à mettre en place des méthodes d'analyse automatisables favorisant l'acquisition de connaissances par des systèmes d'analyse ainsi que leur évolution.

Dans [15], Lehmann présente DL-Learner [12], un framework d'apprentissage reposant sur le formalisme de Logique de Description. Il s'agit d'un outil d'apprentissage automatique supervisé, basé sur la logique de description et utilisant des algorithmes afin de résoudre des problèmes d'apprentissage dans OWL qui est un langage de représentation de connaissances du web sémantique, construit sur le modèle de données RDF et capable de décrire des ontologies. L'utilisation de plusieurs algorithmes permet de s'adapter à la nature et aux spécificités du problème à résoudre. L'outil peut être connecté à des systèmes de raisonnement.

Hellmann *et al.* évoquent l'utilisation des techniques d'apprentissage automatique des bases de connaissances comme les résultats de requêtes SPARQL ou sur des données structurées (linked data [13]) [13]. L'article met le point plus précisément sur l'application de cette approche sur une très large quantité de données. Afin d'améliorer la performance et de favoriser le passage à l'échelle de l'approche, les auteurs préconise un pré-traitement dont l'objectif est de limiter l'analyse uniquement sur les données les plus intéressantes.

3 Description de notre approche

3.1 Cadre du projet

Nos travaux s'inscrivent dans le cadre du projet *Living-Environment Monitoring Use Scenario with Intelligent Control (LivEMUSIC)* [14], un projet piloté par Prof. Hassan Aït-Kaci au sein du Laboratoire d'informatique en image et systèmes d'information (LIRIS) [15] et de l'Université Claude Bernard Lyon 1 [16]. Le projet fait partie du Programme Avenir Lyon Saint-Etienne (PALSE) [17].

[12] http://dl-learner.org/
[13] http://linkeddata.org/
[14] https://liris.cnrs.fr/actualites-en/generales/un-chercheur-du-liris-laureat-de-lappel-package-palse/
[15] http://liris.cnrs.fr/
[16] http://www.univ-lyon1.fr/
[17] http://palse.universite-lyon.fr/

L'objectif de ce projet est de développer des cas d'utilisation dans le domaine de la supervision intelligente de cadre de vie. Afin d'assurer une approche réaliste, nous avons exploité des données publiques fournies par la métropole de Lyon à travers la plateforme Data Grand Lyon [18]. Il s'agit d'un ensemble de 538 jeux de données, dont la plupart sont en accès libre, et qui concernent divers thèmes (environnement, santé, infrastructure ...).

3.2 Technologies et outils utilisés

Notre objectif consiste à dériver des taxonomies et des schémas terminologiques à partir de données brutes. Nous avons décidé d'opter pour une approche basée sur FCA, dans la mesure où c'est la technologie qui répond le plus à nos besoins. FCA permet de dériver des taxonomies, d'effectuer des regroupements d'attributs, de retourner un graphe sous forme de treillis de concepts, et il est possible de se servir des algorithmes et des outils disponibles et de les adapter et enrichir selon nos besoins. FCA est donc compréhensible, utile, et surtout modifiable.

Après l'utilisation des différents outils FCA, nous avons opté pour l'utilisation de Lattice Miner [19], qui est un outil Open Source, qui implémente déjà un algorithme FCA et qui permet de retourner un rendu graphique pour le treillis de concepts. Lattice Miner s'est distingué parmi tous les autres outils et environnements que nous avons explorés par la clarté et l'adaptabilité de son code, la simplicité de son utilisation, tout en offrant toutes les fonctionnalités métier qui existent dans les autres outils.

Toutefois, Lattice Miner présente quelques limites quant à nos propres besoins. Des modifications ont ainsi dû être apportées au code source de cet outil.

3.3 Approche globale

Le but consiste à dériver des taxonomies à partir de données brutes. Les données ne sont pas limitées à un domaine particulier, et peuvent être dérivées à partir de plusieurs sources (bases de données, capteurs, *etc.*). Elles doivent toutefois être mises dans un format lisible par l'outil d'analyse FCA.

Le processus commence par la lecture de données à partir d'un fichier texte. Grâce à un outil graphique, l'utilisateur peut choisir les objets et les attributs du contexte formel qu'il veut obtenir. Il peut aussi spécifier des approximations à effectuer sur ces attributs.

Après la mise en place de ses choix, l'outil Lattice Miner est exécuté et un contexte multi-valeurs est généré. Si aucune modification n'est apportée, l'utilisateur génère un contexte binaire permettant d'obtenir un treillis qui sera affiché sous forme graphique. Le treillis est converti en un fichier RDF pouvant être réutilisé dans d'autres outils de raisonnement.

[18] http://data.grandlyon.com/
[19] http://lattice-miner.sourceforge.net/

3.4 Fonctionnalités de l'approche

Dans ce qui suit, nous détaillons notre approche globale, en présentant le problème rencontré ainsi que la solution que nous avons apportée.

Lecture des données A l'instar de chaque logiciel prenant en entrée des fichiers textes, Lattice Miner a ses propres spécificités et formats d'entrée. Etant donné que nous avons effectué toutes nos expérimentations avec les données publiques de la métropole de Lyon, il a fallu adapter le code source de Lattice Miner afin de pouvoir prendre en entrée les fichiers téléchargés de la plateforme Data Grand Lyon. Le nouveau comportement permet de prendre en entrée des fichiers de format JSON. Il s'agit d'un format assez utilisé, ce qui favorise la réutilisation de la fonctionnalité développée.

Choix des données Les données de la plateforme Data Grand Lyon sont classées par thèmes, comme Transport (état du trafic en temps réel, aménagement cyclable, parkings, stations de métro, *etc.*), Santé, Environnement, *etc.* Chaque source de données peut contenir plusieurs attributs qui ne sont pas forcément utile pour l'utilisateur. L'utilisation de tous les attributs lors de l'application de FCA pourrait provoquer une explosion combinatoire à cause du très grand nombre de concepts obtenus. Pour faire face à ce problème, nous avons développé un nouvel outil graphique permettant à l'utilisateur de ne retenir que les attributs qui l'intéressent et d'ignorer le reste.

Regroupement de données Pour certains champs, notamment les numériques, il est possible de trouver plusieurs valeurs possibles pour le même attribut. Si dans le contexte binaire final, il y avait un attribut pour chaque valeur, nous nous trouverons avec un très grand nombre d'attributs, et donc, de concepts aussi. Nous avons développé des approximations permettant de regrouper les valeurs par intervalle afin de réduire considérablement le nombre d'attributs. Le regroupement se fait par échelonnage conceptuel et peut être appliqué à d'autres types de données que les numériques.

Zoom sémantique Le zoom sémantique est l'une des principales contributions de nos travaux.

Il s'agit d'un type de zoom plus abstrait que d'autres types de zoom. Alors qu'un zoom visuel ne fait qu'augmenter ou diminuer les dimensions d'un objet visualisé, sans modifier ses caractéristiques intrinsèques, et qu'un zoom structurel repose sur une structure multi-échelles qui permet d'afficher ou de cacher des détails de l'objet selon le niveau de précision choisi, le zoom sémantique permet de raisonner sur les attributs mêmes des objets et d'effectuer des opérations supplémentaires (des mises en relations entre les objets par exemple).

Dans notre cas, et comme pour le regroupement de données, nous avons développé d'autres approximations dont le but est d'augmenter ou de diminuer

la précision d'une donnée par rapport à une référence spatiale ou temporelle. Par exemple, des données géographiques sont comparées à des coordonnées géographiques de références (latitude et longitude, GPS, *etc.*) avec un seuil de tolérance fixé par l'utilisateur. Le seuil de zooming ou de regroupement peut donc passer de quelques mètres (rues) à quelques ou plusieurs kilomètres (commune, département, *etc.*). Idem pour les dates qui peuvent être comparées à une date référence.

Conversion de contexte Dans notre approche, nous utilisons des données brutes que nous exploitons avec FCA dans un contexte multi-valeurs. Le contexte multi-valeurs sera converti en un contexte binaire afin d'obtenir le treillis de concepts.

4 Implémentation

Dans cette section, nous présentons nos différents travaux et nous mettons l'accent sur la manière dont ils ont été implémentés.

4.1 Architecture générale

L'architecture générale de l'approche que nous proposons est illustrée dans Figure 4.

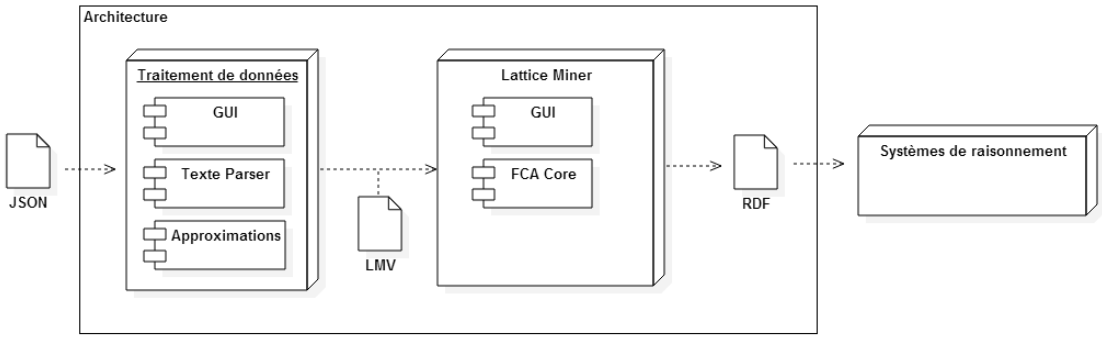

Figure 4. Architecture générale

Le processus d'analyse sera effectué sur des données brutes ayant le format JSON.

Un outil graphique a été développé en intégralité pour les besoins de nos travaux. Il a été développé en Java et la bibliothèque graphique Swing. Il

permet de sélectionner le fichier de données JSON, d'en faire l'analyse syntaxique et d'en afficher la liste des champs. L'utilisateur peut alors choisir les entrées (objets/attributs) de son analyse FCA. Cet outil possède un composant qui sert à gérer les approximations qui seront utilisées pour le zoom sémantique. Finalement, l'outil graphique génère un fichier d'extension `.lmv` qui contient un contexte multi-valeurs et qui sera ouvert dans Lattice Miner.

Lattice Miner a pour rôle de lire le contexte multi-valeurs, de l'éditer si nécessaire, de le transformer en un contexte binaire, puis en appliquant FCA, de générer une taxonomie pour des données.

Enfin, un nouveau module a été mis en place dans Lattice Miner afin de pouvoir générer un fichier RDF à partir du treillis de concept obtenu. Cette génération se fait avec la bibliothèque Apache Jena [20]. Ce fichier RDF sera utilisé à d'autres fins dans des systèmes de raisonnements.

4.2 Scénario d'utilisation

La section précédente a permis de présenter les différents composants de l'architecture de notre approche, mais elle a aussi servi à décrire un enchaînement standard d'utilisation. Le processus est illustré dans l'histogramme spécifié en Figure 5.

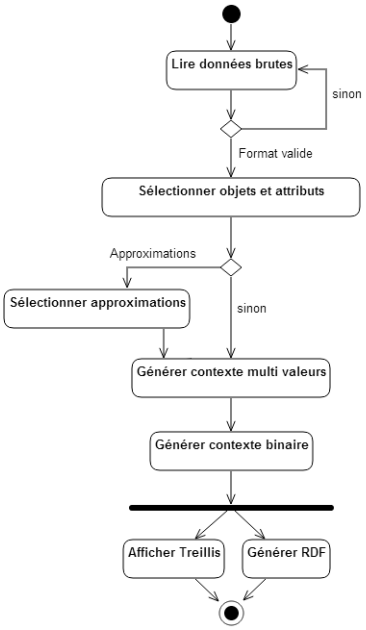

Figure 5. Histogramme de notre système

[20] https://jena.apache.org/

Le diagramme de séquence (Figure 6) lui aussi permet d'illustrer le processus général, mais en plus, il sert à montrer les interactions entre les intervenants ou les composants de notre architecture.

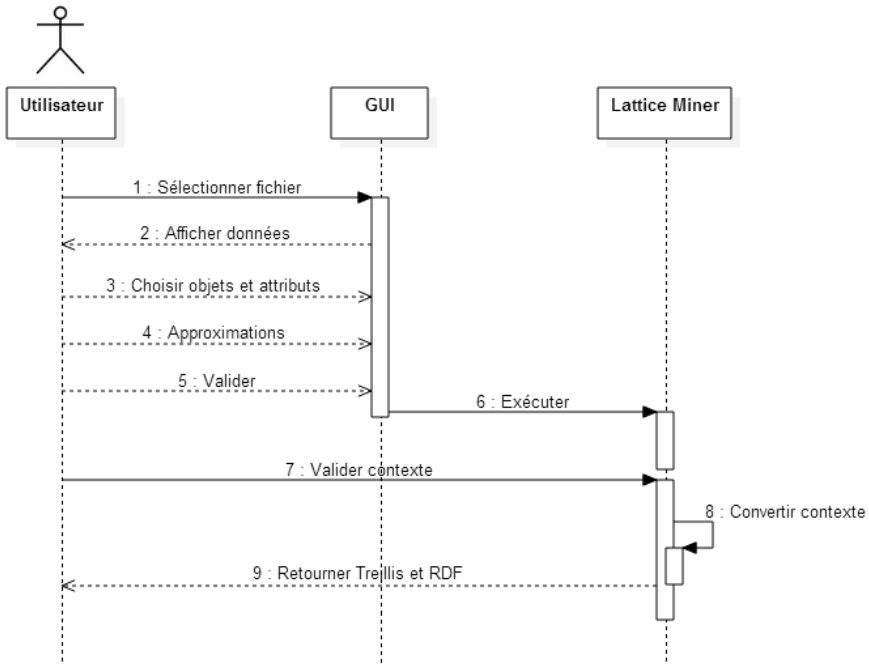

Figure 6. Diagramme de séquence

4.3 Composants et fonctionnalités

Ci-après les détails de l'implémentation des principaux composants et fonctionnalités de notre approche.

Lecture et choix des données Les données sur lesquelles s'est portée notre analyse sont issues de la plateforme Data Grand Lyon et ont le format JSON. Il a donc fallu développer un nouvel outil qui sert à analyser syntaxiquement les fichiers JSON afin de récupérer les données. Nous avons opté pour le développement d'un outil graphique (avec Java et la bibliothèque Swing) afin de rendre l'expérience de utilisateur plus intuitive.

L'interface graphique de cet outil faisant suite à la lecture d'un fichier JSON est illustrée dans Figure 7 . L'interface graphique liste tous les champs du fichier JSON, et l'utilisateur a la main pour spécifier les objets et les attributs qui seront

pris comme entrée pour son analyse FCA. Il s'agit d'une approche certes semi-automatique, mais elle a l'avantage de donner à l'utilisateur le contrôle de ses données et elle évite de se trouver avec des champs inutiles par rapport à l'analyse FCA.

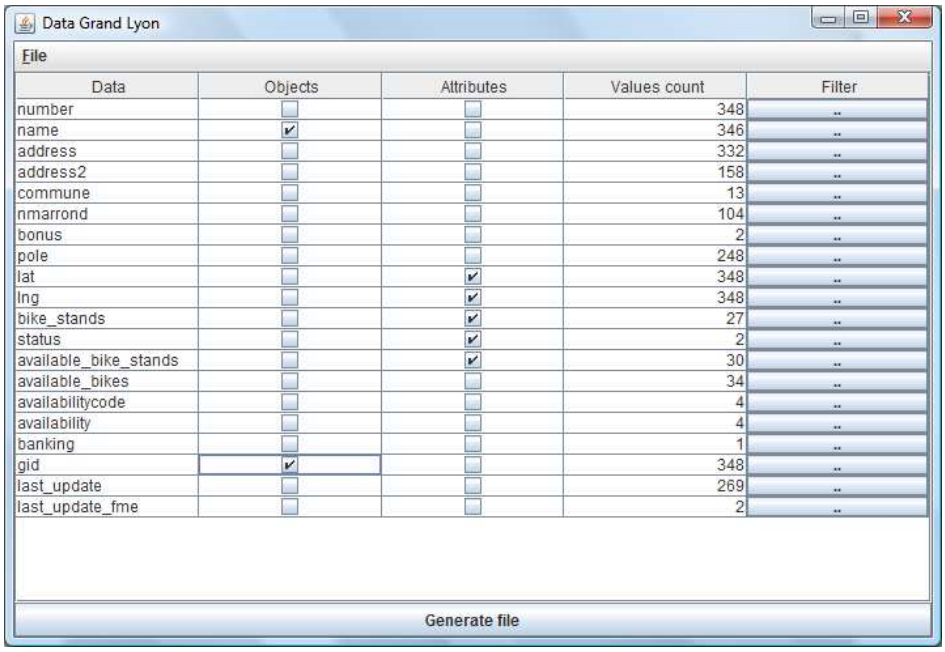

Figure 7. Outil de lecture et de choix de données

Lors du développement de cet outil, nous avons pensé à la réutilisabilité du code. Il est, par exemple, tout à fait possible d'intégrer des analyseurs syntaxiques pour d'autres formats que JSON (XML, CSV, RDF . . .), ou de modifier l'interface graphique en changeant les détails de données affichées.

L'interface actuelle affiche le nom du champ, des cases à cocher permettant de sélectionner les objets et attributs, un bouton pour appliquer des approximations ainsi que le nombre de valeurs qui existent pour chaque champ. Plus un champ a de valeurs, plus le nombre de concepts formels sera grand, d'où la nécessité de regrouper certaines valeurs en y effectuant des approximations.

Approximations et Zoom sémantique Le zoom sémantique est l'une des principales fonctionnalités que nous avons implémentées. Il permet d'augmenter ou de diminuer le degré de précision d'un attribut par rapport à une donnée référence. Pour ce faire, nous avons développé un module qui sert à appliquer un échelonnage conceptuel sur les données en utilisant des approximations spécifiables par l'utilisateur.

La mise en œuvre d'une approximation se fait dans l'outil graphique que nous avons vu précédemment 7. Pour chaque champ du fichier JSON, il existe un bouton (dernière colonne) qui permet de sélectionner, si nécessaire, une approximation.

Nous avons mis en place trois familles d'approximations :

- **Données géographiques :**
  Dans les données de la plateforme du Grand Lyon, nous trouvons plusieurs champs représentant des coordonnées géographiques (coordonnées d'une station de vélos, d'un arrêt de bus . . .). Les coordonnées géographiques utilisées dans la plateforme Grand Lyon sont la latitude et la longitude.
  Nous donnons à l'utilisateur deux possibilités d'échelonnage conceptuel sur les coordonnées géographiques :
  - **Par intervalle :** l'utilisateur spécifie les coordonnées de référence ainsi qu'une valeur qui servira à regrouper les objets selon des intervalles égaux.
    Prenons l'exemple d'un utilisateur qui applique cette approximation sur le jeu de données des stations de vélos. Il choisit l'Hôtel de Ville de Lyon comme point de référence et une valeur de 10 km pour l'intervalle. Toutes les stations de vélos se trouvant moins de 10 km du point central seront considérées similaires (intervalle $[0, 10]$). Les stations se trouvant au moins à 10 km du point de référence et au plus à 20 km de ce point seront considérées comme similaires (intervalle $[10, 20]$), et ainsi de suite.
  - **Par distance de tolérance :** l'utilisateur spécifie les coordonnées d'un point de référence ainsi qu'une distance de tolérance. Elles représentent respectivement le centre d'un cercle et le rayon de ce cercle. Tous les objets qui se trouvent à l'intérieur de ce cercle seront considérés comme similaires et seront regroupés grâce à l'échelonnage conceptuel.

  L'exemple des deux approximations mentionnées ci-dessus est illustré dans Figure 8.
  Afin d'appliquer une approximation géographique, l'utilisateur doit donc spécifier le champ du fichier JSON qui correspond à la latitude, le champ qui correspond à la longitude, le type de l'approximation et l'intervalle ou distance désirés.
- **Données temporelles :**
  Nous avons ajouté une approximation qui permet de regrouper toutes les données temporelles qui ont une date comprise entre deux dates références introduites par l'utilisateur.
- **Données numériques :**
  Les approximations des données numériques sont les mêmes que pour les données géographiques.
  Il est possible soit de regrouper les attributs par des intervalles égaux, soit de les regrouper selon leur appartenance ou non à un intervalle de centre r avec un seuil de tolérance s (intervalle $[r - s, r + s]$).

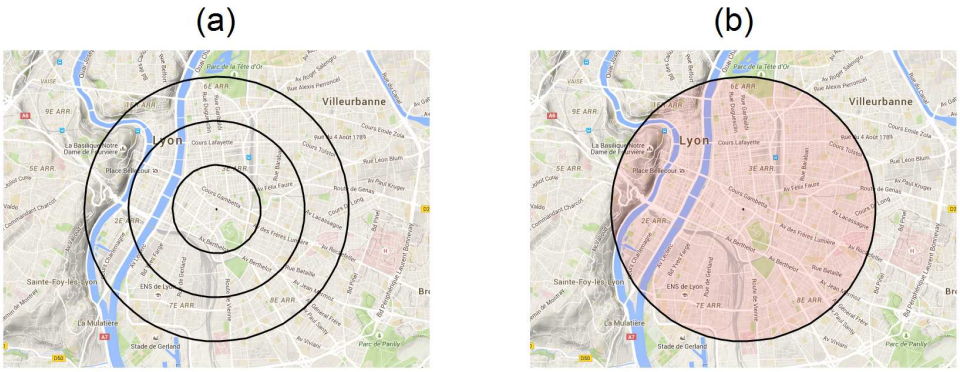

Figure 8. Approximation par intervalle (a) ou par distance de tolérance (b)

Les données regroupées respectent une distribution uniforme comme le montre Figure 9. L'utilisation d'autres distributions (par ex. : gaussienne, exponentielle, *etc.*) est envisageable comme mentionné dans les perspectives.

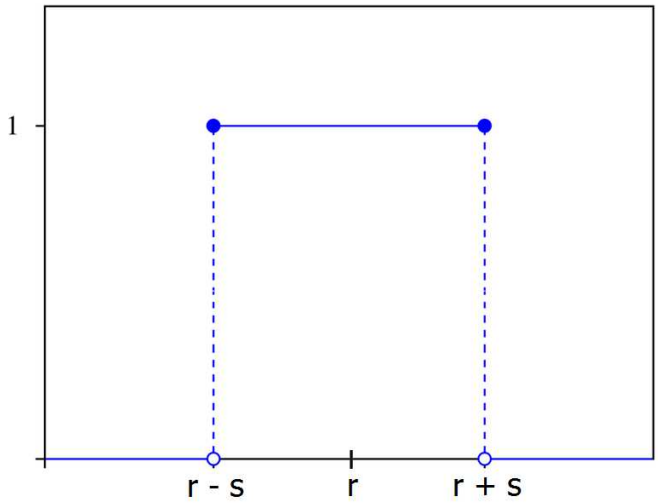

Figure 9. Distribution uniforme des données regroupées

Il est possible de faire des approximations sur les attributs grâce à un bouton relatif à chaque champ dans l'interface graphique. En cliquant sur le bouton, l'utilisateur sélectionne le type d'approximation qu'il veut appliquer, et selon le type, il introduit les valeurs nécessaires.

Un exemple d'application d'une approximation de type "regroupement de valeurs numériques par distance de tolérance" est montré dans Figure 10.

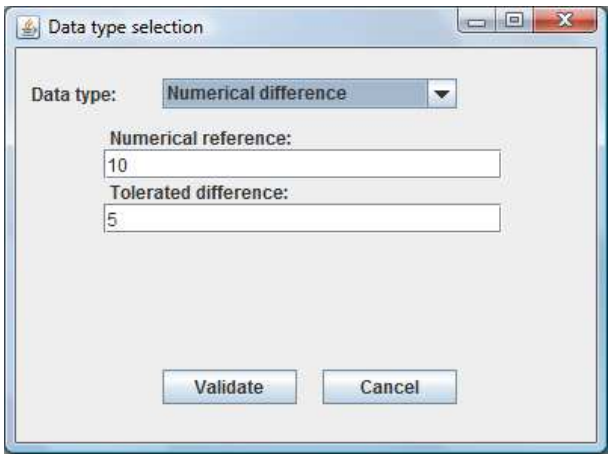

Figure 10. Fenêtre d'application d'approximation

Le code source du module d'approximations est réutilisable et il supporte l'ajout d'autres approximations dans le futur.

Lattice Miner Le code source de Lattice Miner a été modifié et augmenté afin que cet outil réponde au mieux à nos besoins. Par exemple, nous avons modifié le code source de Lattice Miner pour pouvoir l'exécuter dès la sélection des données dans l'outil graphique précédent, et ainsi manipuler directement le contexte multi-valeurs obtenu. Nous avons aussi modifié les classes de lecture de fichiers afin de les adapter à de nouveaux formats textuels que nous avons spécifiés. D'autre part, ayant trouvé que Lattice Miner génère quelques attributs redondants lors de la conversion d'un contexte multi-valeurs en un contexte binaire, nous avons changé le comportement de cet outil afin d'ignorer ces champs redondants. Le module d'écriture du treillis a aussi été enrichi pour pouvoir générer le treillis obtenu sous format RDF.

Fichier de sortie RDF Resource Description Framework (RDF) est un format qui permet de décrire, sous forme de graphe, des ressources web ainsi que leurs méta-données. Dans nos travaux, nous nous sommes servis de RDF afin de décrire formellement la taxonomie issue de l'analyse FCA. Le treillis de concepts est donc décrit dans un fichier RDF où nous trouvons les informations suivantes :

- Les objets FCA.
- Les attributs FCA.

– Les concepts formels : contenu (objets et attributs) et hiérarchie (par ex. : le Concept A est le parent du Concept B).

Le module de génération de fichiers RDF a été ajouté dans l'outil Lattice Miner. La génération du fichier RDF se fait simultanément avec l'obtention du treillis. L'implémentation s'est faite avec la bibliothèque Apache Jena, développée en Java.

Enfin, le fichier RDF ainsi obtenu peut être réutilisé dans des systèmes de raisonnement par exemple.

5 Expérimentations

Nous avons utilisé notre approche sur des données de la plateforme du Grand Lyon afin de vérifier qu'elle est applicable sur des cas d'utilisation réels. Le but de nos travaux est d'obtenir, à partir de données brutes, des ontologies qui soient personnalisées par l'utilisateur.

Le cas d'utilisation que nous avons testé est celui des données relatives aux stations de vélos de Lyon (Velo'V). Ce jeu de données contient 348 entrées (une entrée pour chaque station de Velo'V existante) et 20 champs. Parmi les champs, nous trouvons des chaînes de caractères (nom de la station, l'adresse, la commune, *etc.*), des entiers (nombre total de places, nombre de vélos disponibles, nombre de places vides, *etc.*), des valeurs binaires (possibilité de payer par carte bancaire), des dates (date de mise à jour), ou encore, des coordonnées géographiques (latitude et longitude).

Nous avons repris le jeu de test que montre Figure 7. Le contexte contiendra alors 348 objets (nombre de stations) et 4 attributs (la latitude et la longitude forment un seul attribut). Nous effectuons ces quatre séries de tests :

- Cas 1 : Obtention du contexte binaire sans aucune approximation.
- Cas 2 : Application d'une approximation de regroupement géographique par intervalle de 1000m.
- Cas 3 : Approximation de regroupement géographique par intervalle de 2500m.
- Cas 4 : Approximation de regroupement géographique par intervalle de 2500m, regroupement du nombre total de places par tranche de 20 et regroupement des stations ayant un nombre de places disponibles compris entre 10 et 20.

Le tableau 2 nous démontre que plus les approximations sont restrictives, moins il y aura des concepts dans le treillis final.

L'approche est efficace dans la mesure où elle permet de réduire considérablement le nombre de concepts en les regroupant. L'avantage, c'est que le regroupement se fait d'une manière cohérente selon des métriques définies par l'analyste lui-même.

Pour chaque cas de test, un fichier RDF est généré, contenant la liste des objets, la liste des attributs, la composition des concepts ainsi que leur hiérarchie.

Table 2. Nombre de concepts par cas de test

Cas de test	Nb. de concepts
Cas 1	418
Cas 2	266
Cas 3	181
Cas 4	44

Comme le montre ce bout de code, nous

Nous nous sommes servis des vocabulaires RDF et OWL afin de déclarer nos classes ainsi que leurs propriétés, comme le montre cet exemple :

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:ex="http://example.org/"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#">
  <owl:Class rdf:about="http://example.org/Attribute"/>
  <owl:Class rdf:about="http://example.org/Concept"/>
  <owl:Class rdf:about="http://example.org/Object"/>
  <owl:ObjectProperty rdf:about="http://example.org/hasValue"/>
```

Voici un exemple, en RDF, représentant un des objets du contexte final :

```
  <ex:Object rdf:about="http://example.org/Object/96">
    <ex:hasName>Gare Part-Dieu / Vivier Merle</ex:hasName>
  </ex:Object>
```

Ceci est un exemple d'un attribut de notre contexte :

```
  <ex:Attribute rdf:about="http://example.org/Attribute/10">
    <ex:hasName>position interval</ex:hasName>
    <ex:hasValue>1000-2000</ex:hasValue>
    <ex:hasDataType>DistanceInterval</ex:hasDataType>
    <ex:longitude>4.85</ex:longitude>
    <ex:latitude>45.75</ex:latitude>
    <ex:interval>1000</ex:interval>
  </ex:Attribute>
```

Nous retrouvons dans cet exemple les champs classiques d'un attribut, à savoir son nom et sa valeur, mais aussi des détails supplémentaires relatifs à l'approximation qui a été appliquée (dans ce cas, un regroupement de données géographiques par intervalle), ce qui favorise la réutilisation du fichier dans des systèmes de raisonnement.

6 Conclusion et perspectives

Dans ce papier, nous avons présenté une nouvelle approche basée sur FCA permettant de collecter des données brutes et non-structurées, de manière semi-

automatique (à la discrétion de l'utilisateur), et d'en extraire des schémas terminologiques et des ontologies. Un rendu graphique permet de visualiser les ontologies sous forme de treillis. Des triplets RDF seront générés et réutilisés dans des systèmes de raisonnement.

L'approche utilisée repose aussi sur une notion de zoom sémantique. Elle consiste à regrouper ou à dissocier des données selon leur degré de ressemblance.

Un prototype expérimental sous forme d'outil enrichissant le logiciel Open Source Lattice Miner a été développé afin de permettre aux utilisateurs de définir leurs préférences et les approximations qu'ils veulent mettre en place.

Plusieurs perspectives sont envisageables afin d'enrichir les travaux que nous avons menés.

D'abord, il serait intéressant d'enrichir les approximations qui ont été déjà mises en place avec des nouvelles métriques sur d'autres types de données (par ex. : couleurs, chaînes de caractères, *etc.*).

Aussi, dans nos cas d'utilisation, nous avons utilisé des métriques sur des données ayant une distribution uniforme, mais rien n'empêche d'utiliser dans le futur d'autres types de distributions non-uniformes (par ex. : gaussienne, exponentielle, poissonienne, *etc.*). Ces distributions permettent d'être plus restrictif ou plus souple dans le regroupement de données, comme le montre Figure 11.

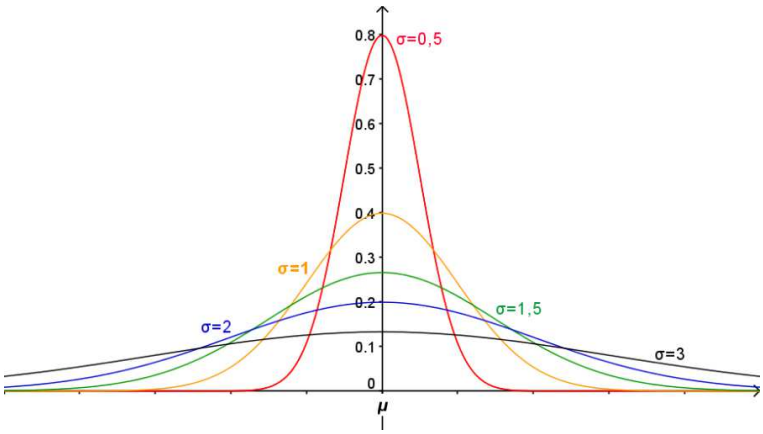

Figure 11. Distribution de données selon la loi de Gauss

La solution peut être connectée aussi à une base de données géographiques afin d'améliorer et d'automatiser la gestion de tous les types de coordonnées géographiques, ou pour pouvoir retourner des rendus graphiques (cartes) facilitant la compréhension de ce type de données. Le zoom sémantique sera plus précis, en passant par exemple d'approximations par commune ou par arrondissement à des approximations par rue.

L'outil peut être interfacé avec un système de gestion de bases de données, donnant plus de souplesse à l'utilisateur dans le choix de ses données et supportant plus de requêtes.

Une autre évolution serait de modifier l'outil Lattice Miner afin de pouvoir éditer graphiquement les treillis. L'utilisateur aura la possibilité de regrouper ou de supprimer graphiquement des concepts. La suppression de concept pourrait transformer le treillis en un ensemble partiellement ordonné, ce qui rend le graphe moins contraignant.

Enfin, il reste possible de connecter notre outil avec un système de raisonnement, afin d'automatiser l'utilisation des triplets RDF dès leur génération.

Références

1. Mehwish Alam and Amedeo Napoli. Lattice-based views over SPARQL query results. In Ilaria Tiddi, Mathieu d'Aquin, and Nicolas Jay, editors, *Proceedings of the 1st Workshop on Linked Data for Knowledge Discovery*, pages 49–58, Nancy, France, September 19, 2013. CEUR-WS.org, CEUR Workshop Proceedings. [Available online[21]].
2. Peter Becker and Joachim Hereth Correia. The toscanaj suite for implementing conceptual information systems. In Bernhard Ganter, Gerd Stumme, and Rudolf Wille, editors, *Formal Concept Analysis, Foundations and Applications*, pages 324–348. Springer, 2005. [Available online[22]].
3. R. Bělohlávek and V. Vychodil. What is a fuzzy concept lattice ? In *CLA 2005, Proceedings of the 3rd International Workshop*, pages 34–45, Olomouc, 2005. CEUR-WS.org. [Available online[23]].
4. Rokia Bendaoud, Amedeo Napoli, and Yannick Toussaint. Formal concept analysis : A unified framework for building and refining ontologies. In Aldo Gangemi and Jérôme Euzenat, editors, *Knowledge Engineering : Practice and Patterns, 16th International Conference, EKAW 2008, Acitrezza, Italy, September 29 - October 2, 2008. Proceedings*, pages 156–171. Springer, 2008. [Available online[24]].
5. Melisachew Wudage Chekol and Amedeo Napoli. An FCA framework for knowledge discovery in SPARQL query answers. In Eva Blomqvist and Tudor Groza, editors, *Proceedings of the 12th International Semantic Web Conference's Posters & Demonstrations Track, (ISWC 2013),*, pages 197–200, Sydney, Australia, October 23, 2013. CEUR-WS.org, CEUR Workshop Proceedings. [Available online[25]].
6. Philipp Cimiano, Andreas Hotho, and Steffen Staab. Learning concept hierarchies from text corpora using formal concept analysis. *J. Artif. Intell. Res. (JAIR)*, 24 :305–339, 2005. [Available online[26]].
7. Olivier Curé and Robert Jeansoulin. An FCA-based solution for ontology mediation. *Journal of Computing Science and Engineering*, 3(2) :90–108, June 2009. [Available online[27]].

[21] http://ceur-ws.org/Vol-1232/paper5.pdf

[22] Lien raccourci

[23] http://ceur-ws.org/Vol-162/paper4.pdf

[24] https://hal.inria.fr/inria-00344051/document

[25] http://ceur-ws.org/Vol-1035/iswc2013_poster_5.pdf

[26] http://www.aaai.org/Papers/JAIR/Vol24/JAIR-2409.pdf

[27] http://jcse.kiise.org/posting/3-2/jcse_3-2_45.pdf

8. Frithjof Dau and Barış Sertkaya. Formal concept analysis for qualitative data analysis over triple stores. In Olga De Troyer, Claudia Bauzer Medeiros, Roland Billen, Pierre Hallot, Alkis Simitsis, and Hans Van Mingroot, editors, *Proceedings of The 1st International Workshop on Modeling and Reasoning for Business Intelligence (MOREBI 2011)*, pages 45–54. LNCS 6999, Springer, November 2011. [Available online[28]].
9. Bernhard Ganter and Rudolf Wille. *Formal Concept Analysis—Mathematical Foundations*. Springer, 1999.
10. Hele-Mai Haav. A semi-automatic method to ontology design by using FCA. In Václav Snášel and Radim Bělohlávek, editors, *Proceedings of the CLA 2004 International Workshop on Concept Lattices and their Applications*, pages 13–24, Ostrava, Czech Republic, September 23–24, 2004. CEUR-WS.org, CEUR Workshop Proceedings. [Available online[29]].
11. Mohamed Rouane Hacene, Marianne Huchard, Amedeo Napoli, and Petko Valtchev. A proposal for combining formal concept analysis and description logics for mining relational data. In *Formal Concept Analysis, 5th International Conference, ICFCA 2007, Clermont-Ferrand, France, February 12-16, 2007, Proceedings*, pages 51–65, 2007. [Available online[30]].
12. Marti A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *14th International Conference on Computational Linguistics, COLING 1992, Nantes, France, August 23-28, 1992*, pages 539–545, 1992. [Available online[31]].
13. Sebastian Hellmann, Jens Lehmann, and Sören Auer. Learning of OWL class descriptions on very large knowledge bases. *Int. J. Semantic Web Inf. Syst.*, 5(2) :25–48, 2009. [Available online[32]].
14. Markus Kirchberg, Erwin Leonardi, Yu Shyang Tan, Sebastian Link, Ryan K. L. Ko, and Bu-Sung Lee. Formal concept discovery in semantic web data. In Florent Domenach, Dmitry I. Ignatov, and Jonas Poelmans, editors, *Proceddings of the 10th International Conference Formal Concept Analysis, (ICFCA 2012)*, pages 164–179, Leuven, Belgium, May 7–10, 2012. LNCS 3626, Springer. [Available online[33]].
15. Jens Lehmann. Dl-learner : Learning concepts in description logics. *Journal of Machine Learning Research*, 10 :2639–2642, 2009. [Available online[34]].
16. Alexander Maedche and Steffen Staab. Mining ontologies from text. In Rose Dieng and Olivier Corby, editors, *Knowledge Acquisition, Modeling and Management, 12th International Conference, EKAW 2000, Juan-les-Pins, France, October 2-6, 2000, Proceedings*, pages 189–202. Springer, 2000. [Available online[35]].
17. Zdzislaw Pawlak. Rough sets. *International Journal of Parallel Programming*, 11(5) :341–356, 1982. [Available online[36]].
18. Uta Priss. Linguistic applications of formal concept analysis. In Bernhard Ganter, Gerd Stumme, and Rudolf Wille, editors, *Formal Concept Analysis, Foundations and Applications*, pages 149–160. Springer, 2005. [Available online[37]].

[28] http://dr-dau.net/Papersneu/MoreBI_workshop_Dau_Sertkaya_final.pdf

[29] http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-110/paper2.pdf

[30] http://hal-lirmm.ccsd.cnrs.fr/lirmm-00163364/document

[31] http://www.aclweb.org/anthology/C92-2082.pdf

[32] http://www.informatik.uni-leipzig.de/ auer/publication/2009_dllearner_sparql.pdf

[33] http://slink.foiks.org/Articles/ICFCA12.pdf

[34] http://jmlr.org/papers/volume10/lehmann09a/lehmann09a.pdf

[35] http://aifb.kit.edu/images/9/93/2000_511_Maedche_Mining_Ontologi_1.pdf

[36] http://sierpes.cs.us.es/cursos/ia2/trabajos/RoughSets.pdf

[37] http://upriss.org.uk/papers/fcaic03.pdf

19. Thanh Tho Quan, Siu Cheung Hui, and Tru Hoang Cao. A fuzzy fca-based approach to conceptual clustering for automatic generation of concept hierarchy on uncertainty data. In Václav Snášel and Radim Bělohlávek, editors, *Proceedings of the CLA 2004 International Workshop on Concept Lattices and their Applications, Ostrava, Czech Republic, September 23-24, 2004.* CEUR-WS.org, 2004. [Available online[38]].
20. Helmut Schmid. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of International Conference on New Methods in Language Processing*, September 1994. [Available online[39]].
21. Helmut Schmid. LoPar : Design and implementation. Bericht des Sonderforschungsbereiches "Sprachtheoretische Grundlagen fr die Computerlinguistik" 149, Institute for Computational Linguistics, University of Stuttgart, Stuttgart, Germany, 2000. [Available online[40]].
22. Ramakrishnan Srikant and Rakesh Agrawal. Mining generalized association rules. In *VLDB'95, Proceedings of 21th International Conference on Very Large Data Bases, September 11-15, 1995, Zurich, Switzerland.*, pages 407–419, 1995. [Available online[41]].
23. Gerd Stumme and Alexander Maedche. FCA-MERGE : bottom-up merging of ontologies. In Bernhard Nebel, editor, *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI 2001)*, pages 225–234, Seattle, Washington, USA, August 4–10 2001. Morgan Kaufmann. [Available online[42]].
24. Petko Valtchev, David Grosser, Cyril Roume, and Mohamed Rouane Hacene. Galicia : An open platform for lattices. In *In Using Conceptual Structures : Contributions to the 11th Intl. Conference on Conceptual Structures (ICCS'03)*, pages 241–254. Shaker Verlag, 2003. [Available online[43]].
25. Lotfi A. Zadeh. Fuzzy sets. *Information and Control*, 8(3) :338–353, 1965. [Available online[44]].

[38] http://ceur-ws.org/Vol-110/paper3.pdf

[39] http://www.cis.uni-muenchen.de/ schmid/tools/TreeTagger/data/tree-tagger1.pdf

[40] http://www.cis.uni-muenchen.de/ schmid/papers/lopar.pdf

[41] http://www.rsrikant.com/papers/vldb95.pdf

[42] http://www.kde.cs.uni-kassel.de/stumme/papers/2001/IJCAI01.pdf

[43] Lien raccourci

[44] https://wiki.eecs.yorku.ca/course_archive/2014-15/F/4412/_media/zadeh65.pdf