Rapport projet PLDAC

Année universitaire 2018-2019

Analyse topologique de données encyclopédiques

Auteurs :
Salim MOULOUEL
Samy SADAOUI

Encadrants :
Hubert NAACKE
BAAZIZI MOHAMED-AMINE

Table des matières

1 Introduction et Problématique

1.1 Introduction

L'expansion quantitative des données numériques oblige de nombreux chercheurs à trouver de nouvelles manières de voir et d'analyser le monde. Les données sont recueillies de façons de plus en plus innovante et de manière continue, celles-ci fournissent des informations qu'il faut savoir stocker et traiter efficacement.

Le DataMining est une discipline qui s'inscrit très bien dans ce context, car elle vise à extraire les informations pertinentes d'un grand ensemble de données dans l'optique de les transformer en connaissances utiles, on peut distinguer deux grandes familles de tâches réalisées en DataMining :

- *Description* : consiste à trouver des caractéristiques générales relatives aux données fouillées et en déduire de nouvelles connaissances.
- *Prdiction* :consiste à faire de l'inférence à partir des données actuelles pour prédire des évolutions futures.

1.2 Problématique

L'enjeu de notre projet consiste à analyser et a explorer une grande base de connaissances (knowledge base), en commençant par l'extraction des motifs les plus fréquents, puis de compléter notre base de connaissances, en complétant certains motifs pour arriver à ces motifs déjà extrait.

1.3 Techniques et avancées dans le mining de graphe

1.3.1 AMIE[1] (Association Rule Mining under Incomplete Evidence)

Etat de l'art actuel cette méthode permet de raisonner même en OWA (open world assumption), elle ne requiert donc rien de plus que la base de connaissance elle.

- l'extraction de motifs : si on se met dans le contexte de Frequent Item Set Mining nous pouvons voir cette approche comme l'extraction des d'itemSet les plus frequent.
- la définition d'un seuil minimum de support pour ces motifs.
- utilisation de règle de monotonie du support des items pour élaguer les items c-a-d les motifs, en effet si un motif possède un support

inferieur au seuil ca ne sert a rien de calculer le support des motif dans lesquels il est inclus car il sera inférieur.
— utilisation de la monotonie sur les regle pour élaguer les règle d'inférence en effet la confiance une règle qui possède son corp inclus dans un une autre aura une confiance inférieur

1.4 Présentation des données

Avant d'entamer la procédure d'analyse, nous allons définir quelques notions qui nous permettent d'expliquer notre façons de procéder.

1.4.1 Qu'est ce que RDF ?

RDF qui signifie Ressource Description Framework est un modèle de données standardisées proposé par W3C et publié en 2014. Fondamentalement, les données sont représentées sous forme de graphe orienté, un graphe RDF est donc un ensemble de triplets(sujet, prédicat, objet), un triplet RDF peut également être vu comme un arc entre deux sommets d'un graphe. Deux triplets ayant un même sujet ou objet seront ainsi connectés dans ce graphe.

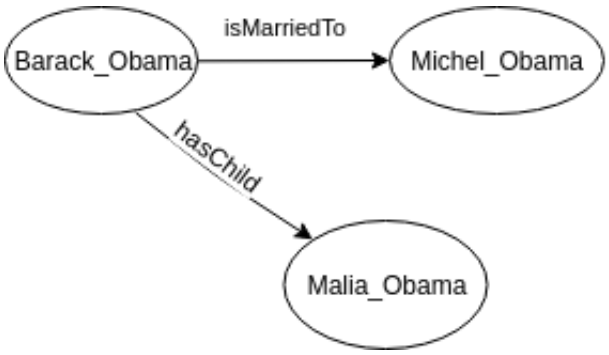

FIGURE 1 – Exemple de relations au format RDF

1.4.2 Qu'est ce que Yago ?

Yago [2] est une grande base de connaissances construite automatiquement à partir de Wikipedia, WordNet et GeoNames.

Le projet combine des informations de Wikipédia dans 10 langues différentes, donnant ainsi une connaissance de dimension multilingue, il attache également des informations spatiales et temporelles à de nombreux faits, et permet ainsi à l'utilisateur d'interroger les données dans l'espace et dans le temps. YAGO se concentre sur la qualité de l'extraction et possède une précision évaluée manuellement de 95

2 Contribution et méthodes

2.1 Motifs

2.1.1 Définitions

Tout au long du projet nous avons travaillé avec les motifs suivants :

— *Triangle* : Ce motif représente la plus petite correspondance de motifs non triviale, il fera donc un motif très intéressent pour effectuer nos analyses. Nous distinguons deux types de triangles :

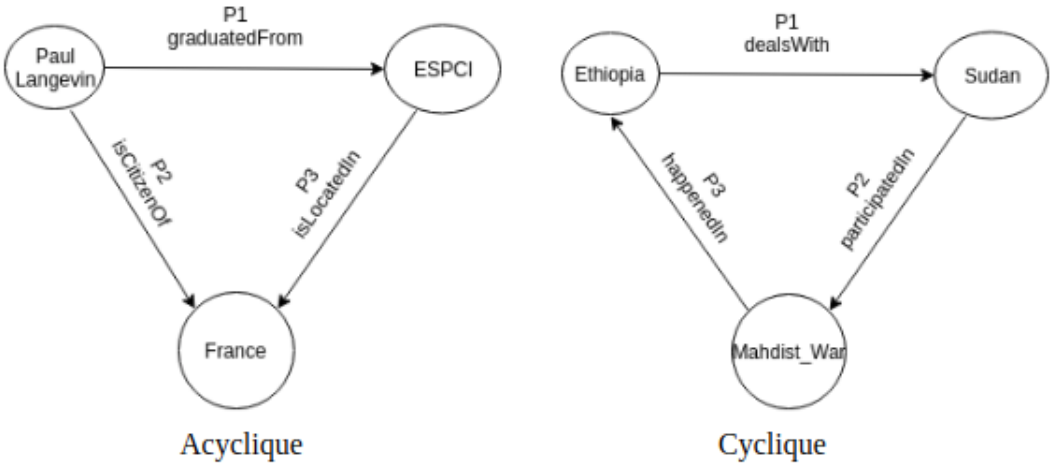

FIGURE 2 – Motif triangle

— *V* :Celui-ci est très intéressant car c'est un sous-motif du motif triangle et du motif N, l'objectif du projet et de réussir à fermer le plus de motifs possible pour en faire des motifs triangles.

— *N* :Celui-ci se distingue du motif triangle par une seul propriété qui au lieu de venir fermer un motif V en motif triangle, la propriété pointe vers une autre entité.

2.1.2 Appellation Canonique

Avant de se lancer dans le calcul du nombre d'instances de chaque motif, nous avons mis en place des conventions de nommage qui nous permettent de définir chaque motif et chaque type de triangle. Cette étape est notre brique de base pour l'ensemble de notre projet, car elle servira à construire nos motifs triangle de manière à pouvoir gérer la redondance et ainsi, calculer le nombre d'instances de chaque type de triangle.

— *Triangle cyclique* :

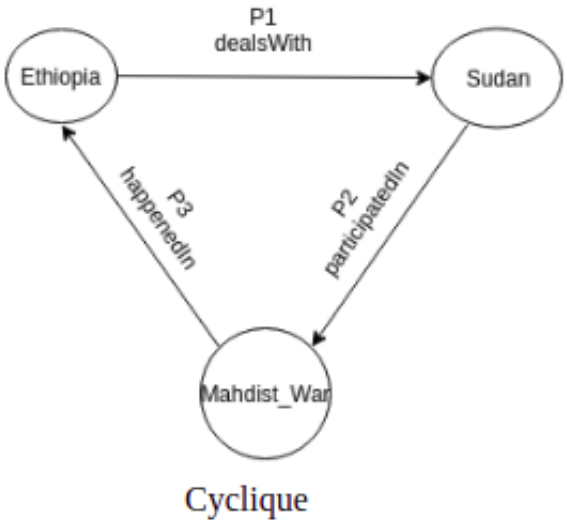

FIGURE 3 – Exemple de triangle cyclique

Pour construire ce motif, nous devions faire face à un problème de redondance. En effet, certaines relations se répètent, 2 ou 3 fois, de ce fait lors du calcul du nombre d'instances d'un type de triangle, il fallait prendre en considération le nombre de relations qui se répètent. Pour palier à ce problème nous avons mis en place les conventions suivantes : *Ordonner les proprits* :

1. Si les 3 propriétés sont différentes :(P1P2P3)
   On prend la plus petite des 3 (ordre alphabétique) comme étant P1, on continue ensuite la construction.
2. Si 2 propriétés sont identiques : (P1=P2 P3)
   — Si elles sont plus petites que la 3eme(P1=P2 < P3), alors on part du sujet de P1,puis on continue la construction.
   — Si elles sont plus grandes que la 3eme relation(P1=P2> P3), alors on part du sujet de P3 (pour faire une rotation), tel que P3 devient P1 et P1 devient P2 ,enfin P2 devient P3.
3. Si les 3 propriétés sont identiques : (P1=P2 P3)
   On s'intéresse alors à l'ordre alphabétique des sujets de chaque relation, et on prend le plus petit sujet, comme étant le sujet de P1 et ainsi de suite.

Nb : P1(respectivement P2, P3) signifie propriété 1(respectivement 2, 3) Type de triangle cyclique :

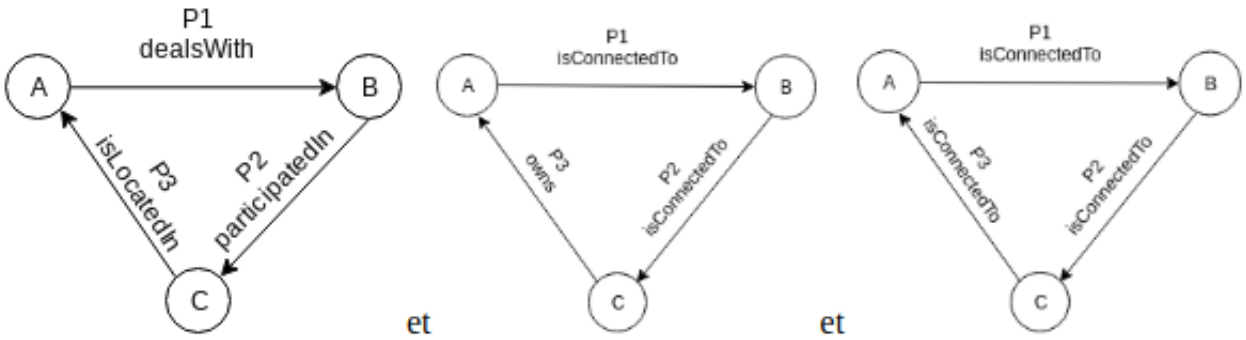

FIGURE 4 – Exemple de rotation

Sont trois types de triangles distincts. Sur ce motif nous avons identifié 31 types de triangles différents, nous avons ensuite compté le nombre d'instances de chaque type.

— *Triangle acyclique* :

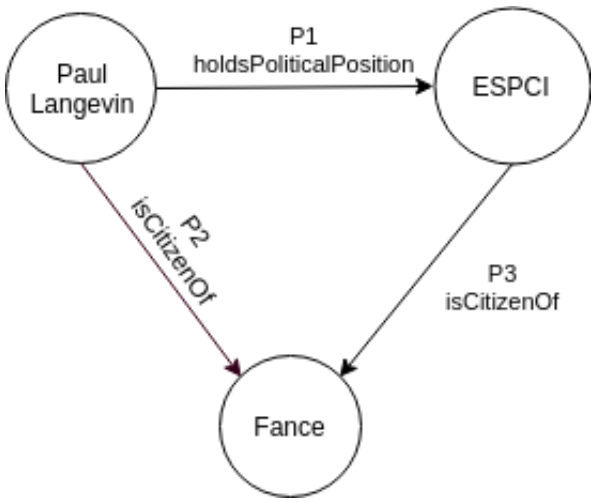

FIGURE 5 – Exemple de triangle acyclique

La construction de celui-ci fut beaucoup plus facile, du fait que les arcs sont orientés. Nous avons donc défini le nommage suivant :

1. Le sujet 1 est pris comme étant celui où nous avons deux arcs sortants
2. La propriété P1 est celle qui pointera sur un objet qui sera lui même sujet, de la propriété P3
3. La propriété P3 à donc pour origine (sujet), l'objet de la propriété P1

Type de triangle acyclique : Celui-ci est formé des 3 propriétés qui le compose comme pour le cyclique, par exemple :

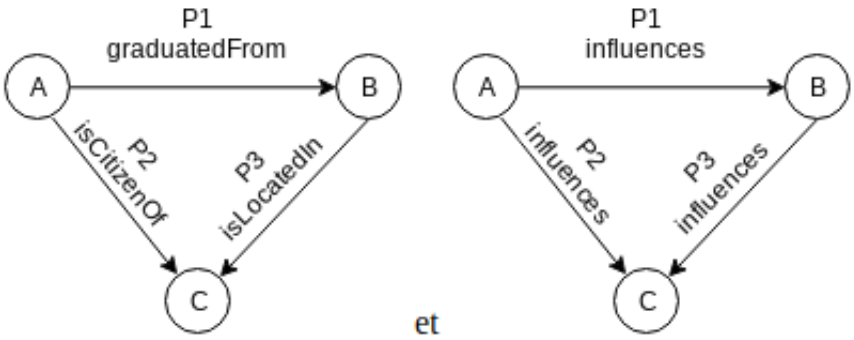

FIGURE 6 – Exemple de type de triangle acyclique

sont deux types distincts ,cette convention nous a permis d'extraire tous nos types de triangles acycliques. Pour ce motif nous avons extrait 528 types.

2.1.3 Construction

Les données du fichier Yago2 sont lues est stockées dans un dataFrame qu'on appellera yago2. Pour construire les deux motifs triangle nous devons tous d'abord commencer par les motifs V .

1. *motifs V*

   Pour former ce motif nous avons effectué une jointure sur yago2 et lui-même, cette opération à été reproduite 2 fois, une fois pour obtenir les V qui composent le motif cyclique (1), une autre pour obtenir les V qui composent le motif acyclique (2). Celui-ci sera utilisé pour la construction du motif triangle.

2. *motif triangle cyclique*

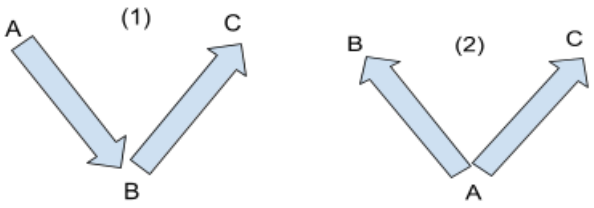

FIGURE 7 – motif construit

Nous commençons d'abord par faire une jointure entre Yago 2 et le dataframe qui contient les V de type cyclique () sur l'objet 2 du motif V et le sujet du dataframe, le sujet 1 du motif V et l'objet du dataframe et sur la propriété, le résultat de cette jointure est un dataframe qui contient toutes les instances qui respectent la configuration de ce motif, mais aussi beaucoup de redondances dû au cycle formé.

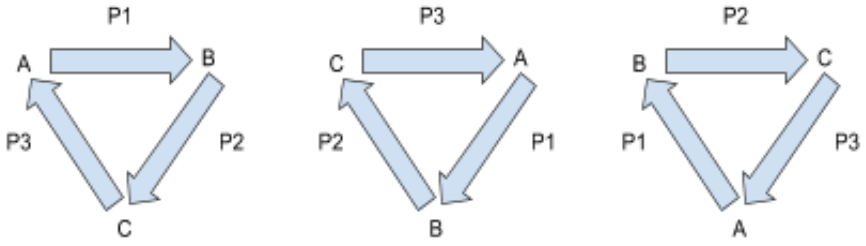

FIGURE 8 – motif construit

Pour faire face à ce problème nous avons dû créer une classe triangle dans laquelle nous restructurant les triangles pour ne former qu'un seul motif c-a-d le motif 1 de la nomination canonique, puis il suffit de faire un distinct sur le dataset pour éliminer toutes les redondances.

3. *motif acyclique*

Pour celui-ci, nous faisons cette fois une jointure entre yago2 et le dataframe qui contient les V de type acyclique() (nous appellerons cette configuration V sortant), cette jointure respecte les conventions de nommage de ce motif-ci et surtout nous n'avons pas de redondances ni de doublons ceci est dû au fait que les triangles acyclique sont différent dès qu'on change de position pour une propriété car

il y a le degré sortant d'un noeud qui diffère d'une position à l'autre du coup on ne peut pas permuter les noeuds ou les propriétés

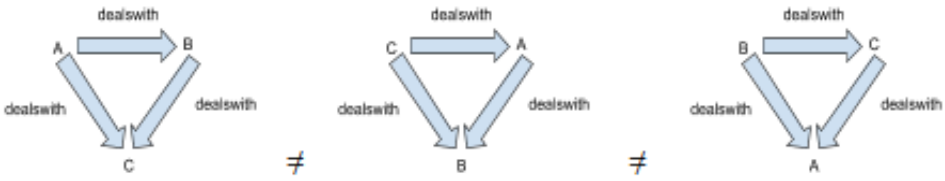

FIGURE 9 – motif construit

4. *motif N*

Pour obtenir ce motif on fait une jointure entre les dataframes contenant les motifs V, et yago2 à fin d'obtenir une forme de structure qu'on appellera N, celui-ci se distingue du motif triangle par la propriété P3 qui ne vient pas fermer un triangle et existe obligatoirement donc nous aurons 4 motifs différents en N c-a-d 3 pour les triangle Acyclique et 1 pour les triangles cycliques.
Acyclique

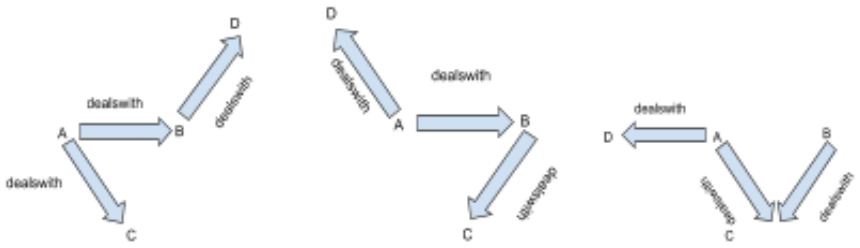

FIGURE 10 – motif construit

Cyclique

Pour le N cyclique cette fois nous n'aurons pas de redondance car c'était le triangle cyclique qui causait la redondance.
Bien évidemment nous faisons un calcul dynamique de ce motif N car avec un DataSet de 4 millions de lignes si on se met à calculer tous les motifs en N pour les stocker cela nous prendra énormément de temp de calcul mais aussi de la mémoire.
On compte ensuite directement le nombre d'instance de chaque type de triangle, grâce à une requête, SQL. Le résultat final est là aussi un dataframe qui est stocké dans un fichier json.

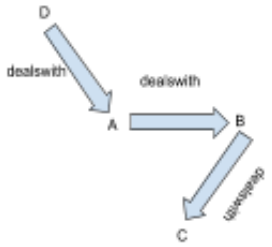

FIGURE 11 – motif construit

2.2 Score de pertinence

$$Score = \frac{Nb\ Triangle}{Nb\ V}$$

En effet à première vu ce score semble le plus adapté pour décider s'il faut oui ou non inférer la relation car il permet grâce à une évaluation statistique de prédire les relations qui peuvent éventuellement compléter nos motifs V candidats.

Si il faut donner une argumentation mathématique nous pouvons dire que ce score représente la confiance de la regle qui permet de fermer un motif , un score haut signifie que la proportion de triangle est supérieur comparé à celle des motifs N ou V. Les V candidats forment probablement un triangle dans yago3.

2.3 Inférence

2.3.1 Ne pas inférer P1

Sur le motif acyclique nous ne nous intéressons pas à la propriété P1, nous allons uniquement nous intéresser aux propriétés P2 et P3. Nous verrons par la suite que les résultats de l'inférence sur cette relation aboutit à de mauvais résultat

2.3.2 Candidats

Pour obtenir de bons résultats dans l'inférence de nouvelles relations, nous avons sélectionné les candidats potentiellement intéressant pour former une nouvelle relation qui n'existe pas dans notre **KB**.

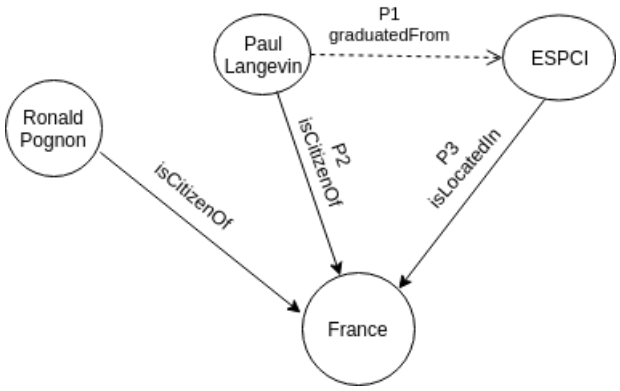

FIGURE 12 – Infernece P1

Pour nous ces candidats sont les instances qui forment un motif V sans pour autant former un motif triangle, ou N, autrement dit, la relation que nous voulons inférer n'est pas déjà existante même vers un autre noeud.

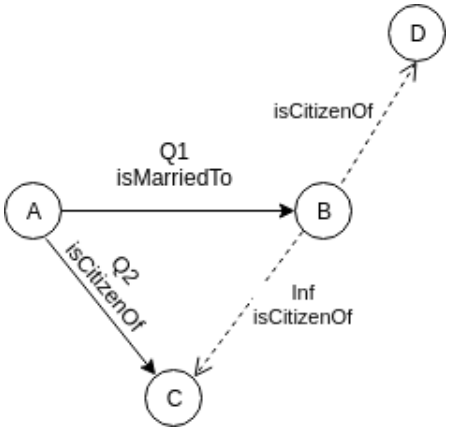

FIGURE 13 – Motif candidats

On utilise ici la même supposition que dans l'article AMIE[1] , qui dit que si l'information B isCitizenOf D est présente dans la base de connaissance alors on ne va pas ajouter l'information B isCitizenOf C ,car si cette relation existe elle aurait déjà été ajoutée.
Les instances A,B et C forment un motif V qui ne contient pas la 3eme propriété entre les instances B et C et cette relation n'existe pas vers un autre noeud. Ces instances forment donc des candidats potentiels pour former un motif triangle dans yago3.

2.3.3 Algorithme

Jusqu'à présent nous avons uniquement à disposition 2 datasets, l'un contient tous les types de triangles cycliques et leurs fréquence, le second, contient tous les types de triangles acycliques ainsi que leur fréquence de la forme suivante :

P1	P2	P3	nbT(Triangle)	NbV(Candidat)
isConnectedTo	isConnectedTo	isConnectedTo	91755	346294
happenedIn	isLocatedIn	participatedIn	1316	10938
dealsWith	participatedIn	happenedIn	578	23713

Le Tableau ci − dessus est un extrait du dataSet contenant le motif cyclique

Pseudo code de l'algorithme :

Algorithm 1 Cyclique

Require: *dataSetFrequence*
Ensure: *dataSetResultat*
  for *ligne* in dataSetFrequence **do**
    for *Pi* in (P1,P2,P3) **do**
      if *Pi* isFunction **then**
        Enlever Pi { pour obtenir un motif V formé des deux proprietes restantes}
        Calculer le score S pour ces motifs Triangle et V
        if *Si* < *Seuil* **then**
          Extraire tous les candidats
          Rajouter la propriete Pi aux candidats
        end if
      end if
    end for
  end for=0

2.3.4 Dépendance fonctionnel

Tous comme dans l'article AMIE[1] , on définie une fonction comme étant une relation (propriété) R qui pour chaque sujet a au plus 1 objet $\forall x : |y : R(x, y)|1$.

Nous appelons cela une dépendance fonctionnelle car la connaissance du sujet et de la propriété nous informe sur l'objet.
Example : hasCapital, wasBornIn sont des fonctions.
Pour améliorer nos résultats nous avons définis une fonction qui nous permet de vérifier si la relation (propriété) est oui ou non une fonction.

$$isFunction = \frac{|x| \ : \ \exists y : r(x,y)}{|r(x,y)|}$$

Celle-ci nous renvoi un taux qui nous permets de prendre cette propriété en considération ou non, nous avons choisi un seuil de 75%.
Ceci permet de réduire le nombre de propriétés à prendre en compte à 20 au lieu des 37 propriétés existantes. Nous avons testé l'apport de cette fonction en essayant d'abord de prendre en considération toutes les relations, ensuite nous avons appliqué le seuil.

3 Validation Expérimentale

3.1 Données

Durant ce projet nous avons travaillé avec deux jeux de données, l'un étant utilisé uniquement pour l'apprentissage, le second pour les tests

3.1.1 Train

Pour la partie analyse et exploration, nous avons travaillé avec la base de connaissance Yago2, qui a été construite en 2012 et qui contient environ 5 million de relations, elle est fournie dans la syntaxe Turtle au format tsv.

3.1.2 Test

Pour tester nos inférences nous avons utilisé la base de connaissance Yago3 qui est la dernière version de yago (données auquel n'avait pas acess la methode AMIE[1] car Yago3 est sorti aprés la publication de l'article), elle a été enrichie de plusieurs informations par rapport à yago2, pour atteindre 12 millions de relations, elle contient donc beaucoup plus d'informations que yago2 ce qui nous permettra de tester notre modèle.

3.2 Nettoyage des données

3.2.1 Nettoyage de Yago2

le dataset yago2 que nous avons trouvé posséde plusieurs relations en doublons donc il a fallu dans un premier temps les enlever pour avoir un yago2 qui ne contient pas de relation en double car cela va perturber nos calcul par la suite.

3.2.2 Construction de Yago3 vérité

Tout d'abord Yago 3 vérité représente le dataset qui ne contient que les instances et types de propriétés déjà présentes dans yago 2 avec potentiellement de nouvelles inférences.

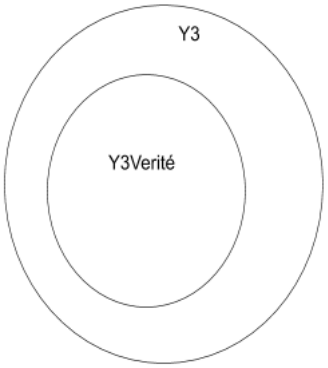

FIGURE 14 – Représentation de Yago3

3.2.3 construction de Y2 Y3

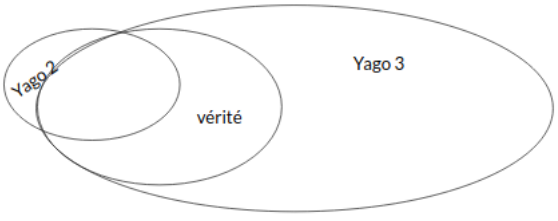

FIGURE 15 – Représentation de Yago2 et yago3

Ce dernier ne contient pas de relations qui seront supprimées dans Yago 3 donc nous ne ferons qu'ajouter des relations grace a Yago3, nous allons du coup utiliser cet ensemble en plus de Yago2 car il permet lors de la construction de nos motifs de ne pas les perdres dans Yago3 à cause d'une suppression de relation qui compose un de nos motifs.
Pour construire cet ensemble il a fallu faire une jointure interne entre Yago2 et Yago3 Vérité.

3.3 Résultats

3.3.1 Métrique d'évaluation

1. *Laprcision*

   C'est le nombre de relations pertinentes inférées sur une propriété d'un motif quelconque rapporté au nombre de relations total inférées. Le principe est le suivant : quand on infère une propriété, on souhaite obtenir un taux de pertinence maximum. Mais, toutes les relations non existant dans Yago3 constituent du bruit. la précision est calculée par la formule suivante

$$Prcision = \frac{Nb\ relations\ pertinentes\ infrs}{Nb\ relations\ total\ inferes}$$

2. *Lerappel* :

   Il est défini par le nombre de relations pertinentes inférées sur une propriété d'un motif quelconque, au regard du nombre total de relations pertinentes que contient Yago3.
   Cela signifie que lorsqu'on infère sur une propriété, notre objectif est de trouver toutes les relations existant dans Yago3, si cette adéquation entre l'inférence et le nombre de relations réellement existant dans yago3 et importante alors le taux de rappel est élevée. la formule est donnée ci-dessous :

$$Rappel = \frac{Nb\ relations\ pertinentes\ inferes}{Nb\ relations\ total\ pertinentes}$$

3.3.2 Comparaison des résultats sur les différents motif

1. *Motifacyclique*
   — Inférence sur P3

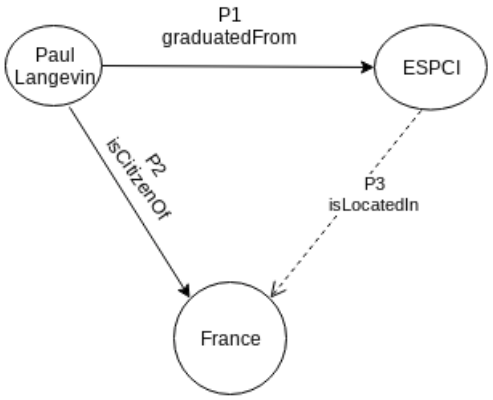

FIGURE 16 – Infernece P1

Le graphe ci-dessous(10 intervalles), nous montre bien que le taux de précision augmente avec l'augmentation du score S1.

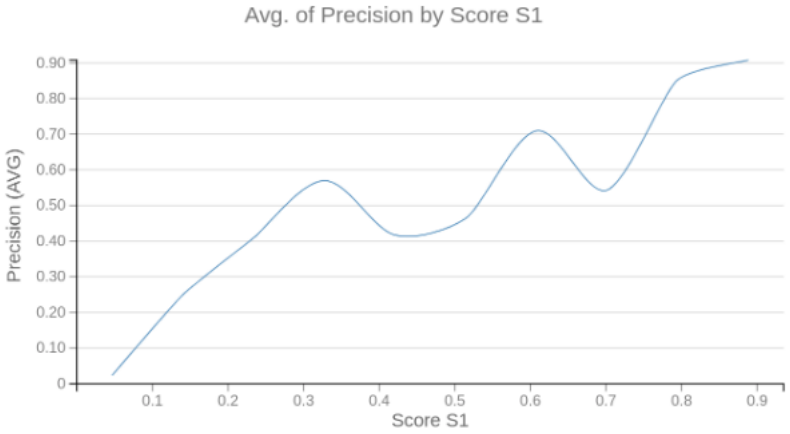

FIGURE 17 – evolution de la precision moyenne en fonction de S1

Les résultats pour cette propriété sont globalement intéressants, On remarque que si score > 0.5 il y a une grande covariance entre notre score et la précision S1> seuil, en effet plus ce seuil est grand plus la précision moyenne dans un interval de score est importante, cela confirme, que les motifs pour lesquelles la proportion de triangles sur le nombre de V est importante sont meilleurs pour effectuer une inférences sur la propriété P2.

— Inférence sur P2

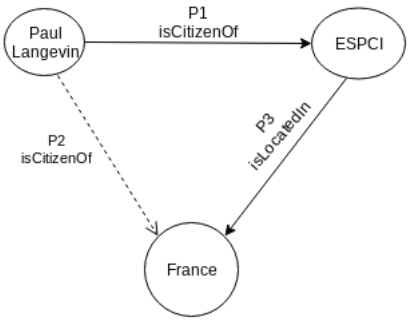

FIGURE 18 – Inferer P2

Le graphe ci-dessous(10 intervalles), nous montre bien que le taux de précision augmente avec l'augmentation du score S1.

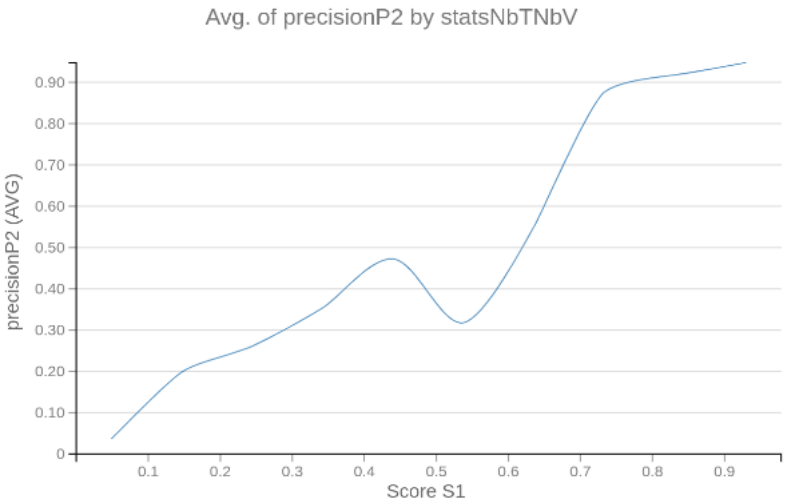

FIGURE 19 – evolution de la precision moyenne en fonction de S2

Les résultats pour cette propriété sont globalement intéressants pour un score S1> seuil, et plus ce seuil est grand plus la précision moyenne dans un interval de score est importante, cela confirme, que pour les motifs pour lesquelles la proportion de triangles sur le nombre de V est importante sont meilleurs pour effectuer une inférence sur la propriété P2.

— Inférence sur P1

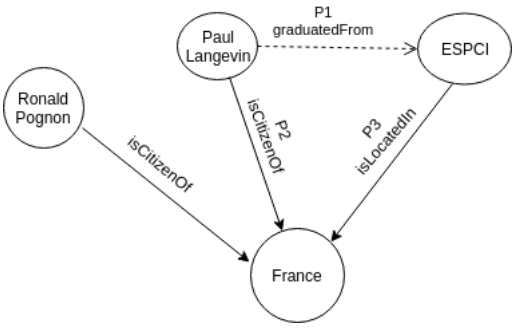

FIGURE 20 – inferer P1

le calculs est inutile pour la propriété P1 car d'un point de vue logique ca ne sert à rien d'infere P1 en effet dire que les personnes qui habitent dans le meme pays sont mariées serait abérrant et malheuresement la plupart des relations ont cette caracteristique, nous pouvons aussi justifier par le fait que les calculs soit trés long à réaliser meme en utilisant une plateforme de calcul paralléle en effet le nombre de V construit est C_N^2 est comme N est très grand ce calcul est trés long.
Le jeux de donnée fourni, contient un très grand nombre de motif V qui prennent cette forme, par exemple nous avons plusieurs entitées (sujets) qui pointes vers un même objet avec la même relation, dans notre exemple nous avons plusieurs sujets (personne) qui pointent vers l'objet France avec la relation "isCitizenOf", et plusieurs sujets (Écoles) qui pointe vers l'objet France avec la relation "isLocatedIn", nous avons donc un nombre de combinaison possible énorme.
Cela explique pourquoi nous avons pris la décision au début de ne pas inférer sur cette propriété

2. Motif cyclique

Sur ce motif, les résultats obtenus ne sont pas très satisfaisant, la précision dépasse rarement les 15%, et pour cause, il y'a très peu de types de triangle qui un nombre d'instance qui est représentatif, et sur les rares types ou le nombre d'instance est grand, le score S1 est la aussi très bas, on ne peut donc pas appliqué un seuil minimum pour S1.
Les résultats en terme de précision moyenne sont pratiquement nulles.

3. Cyclique vs Acyclique

   Le motif Acyclique est bien meilleur pour inférer, comparé au motif cyclique, cela peut s'expliquer par le fait que le motif cyclique possède un peu plus de 30 types de triangles distincts, là où le motif acyclique en possède plus de 500. Sur l'ensemble des types de triangle, le motif cyclique contient très peu de types de triangle qui ont un nombre d'instances important, le seul type qui possède un nombre d'instance significatif est celui ou P1=P2=P3='isConnectedTo', qui représente 90% du nombre totale d'instance pour ce motif, comparé au motif acyclique ou le nombre de types de triangles qui possède un nombre d'instance significatif est globalement haut .

4. P2 vs P3

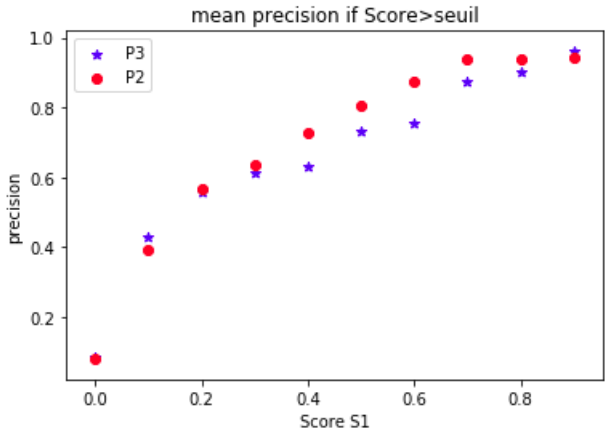

FIGURE 21 – P3 vs P2

   La figure ci-dessus montre une comparaison entre la qualité d'inférence de P3 et de P2 et nous remarquons en effet que la précision de P2 est legerement superieur à celle de P3 mais les deux offrent des résultats plutôt convaincant surtout à partir d'un seuil>0.4

5. Prise en Compte du fait que ca soit une fonction

   Nous pouvons voir que le fait que ca soit une fonction permet d'atteindre une précision très élevée même avec un seuil très bas pour le score de confiance par contre nous restreignons le nombre de relations inférées donc cela implique une augmentation de la précision accompagnée d'une diminution du rappel.

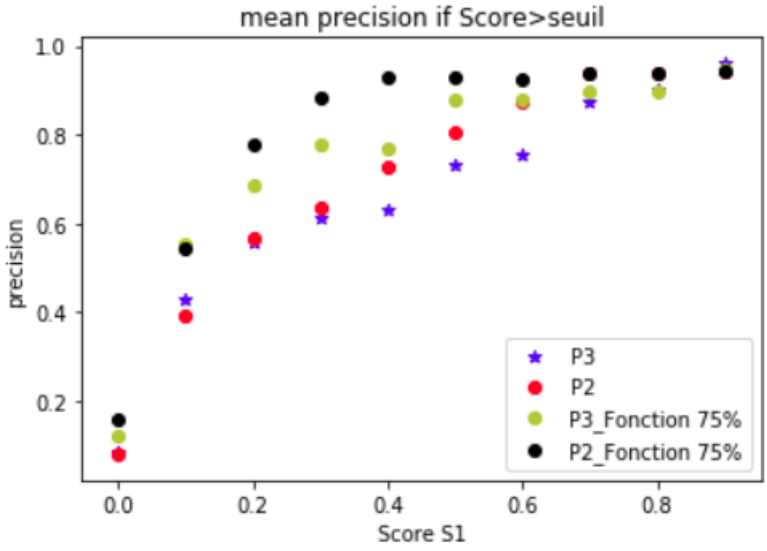

FIGURE 22 – P3 vs P2 amelioré

6. Remarque

   Pour la mesure de rappel, nous obtenons des résultats peu concluant, cela est dû notamment au fait que Yago soit basé sur l'hypothèse du monde ouvert et que du coup nous ne pouvons pas avoir le nombre total de relations existant(nombre qui représente le dénominateur du Rappel)

3.4 Réalisation

Durant ce projet, nous avons travaillé principalement en Scala[3] et Python, le premier est un langage qui intègre le paradigme de programmation objet ce qui nous a permis de construire nos motifs de manière concise. Python nous a permis d'obtenir les graphiques, pour une visualisation plus explicite. en tout nous avons écrit environs 1000 ligne de code en Scala.

3.5 Technologie de développement

Ce projet nous a permis de travailler sur la plateforme Spark.
Spark (ou Apache Spark2) est un framework open source de calcul distribué, il s'agit d'un ensemble d'outils et de composants logiciels structurés selon une architecture définie. Développé à l'université de Californie à

Berkeley par AMPLab3, Spark est aujourd'hui un projet de la fondation Apache. Ce produit est un cadre applicatif de traitements big data pour effectuer des analyses complexes à grande échelle.

4 Perspectives

4.1 Explorer des Motifs plus complexes

Jusqu'à présent nous avons uniquement pris en considération le motif triangle, celui-ci pourrait être utilisé comme motif de base pour d'éventuels constructions plus complexes, comme la construction de carrées ou de pentagones. La aussi une approche similaire a été essayer dans cet article[4]

4.2 Maximum à posteriori

On peut envisager une autre approche dans le cas ou on peut avoir un expert qui peut nous donner les probabilités des différentes inférences, et ainsi utiliser le maximum à posteriori d'une loi multinomiale pour trouver la meilleur relation à inférer sachant un motif donné :

4.3 Sémantique

Intégrer l'aspect sémantique serait un atouts intéressent, par exemple, extraire les paires de propriétés qui ne se combine dans aucun cas. Ceci aura déjà comme avantage d'éliminer un certain nombre de combinaisons et donc de réduire les temps de calculs, cela peut également contribuer à améliorer les résultats.

4.4 Frequent Items set

On peut également envisager d'appliquer les techniques de règles d'association, pour pouvoir extraire les propriétés les plus souvent liées entre elles. Cette technique peut être appliqué sur un motif triangle, par exemple chaque type de triangle représenterait un panier et les propriétés les items. Nous nous sommes d'ailleurs inspiré de cet article [5]

Références

[1] L. A. Galárraga, C. Teflioudi, K. Hose, and F. Suchanek, "AMIE : Association Rule Mining Under Incomplete Evidence in Ontological Knowledge Bases," in *Proceedings of the 22Nd International Conference on World Wide Web*, ser. WWW '13. New York, NY, USA : ACM, 2013, pp. 413–422, event-place : Rio de Janeiro, Brazil. [Online]. Available : http ://doi.acm.org/10.1145/2488388.2488425

[2] T. Rebele, F. Suchanek, J. Hoffart, J. Biega, E. Kuzey, and G. Weikum, "YAGO : A Multilingual Knowledge Base from Wikipedia, Wordnet, and Geonames," in *The Semantic Web – ISWC 2016*, P. Groth, E. Simperl, A. Gray, M. Sabou, M. Krötzsch, F. Lecue, F. Flöck, and Y. Gil, Eds. Cham : Springer International Publishing, 2016, vol. 9982, pp. 177–185. [Online]. Available : http ://link.springer.com/10.1007/978-3-319-46547-$0_1$9

[3] "Pattern Matching | Scala Documentation." [Online]. Available : https ://docs.scala-lang.org/tour/pattern-matching.html

[4] "Mining direct acyclic graphs to find frequent substructures — An experimental analysis on educational data | Elsevier Enhanced Reader." [Online]. Available : https ://reader.elsevier.com/reader/sd/pii/S0020025519300398 ?token=5373A27FF65ABC44

[5] "(PDF) Triangle Counting Approach for graph-based Association Rules Mining." [Online]. Available : https ://www.researchgate.net/publication/261461891