PROJET TUTORÉ

Application de l'analogie sur des partitions K-means des entités de deux graphes de connaissances.

Master 1 Sciences Cognitives
IDMC - Université de Lorraine
2022-2023

Auteur : Chadi MORSI

Encadrants :
Miguel COUCEIRO
Mthieu D'AQUIN

Mots clés : Graphes de connaissances, Analogie proportionnelle, Partitions K-means, PCA, Modèle de distance translationnelle.

Table des matières

1 Introduction

Le graphe de connaissance (GC) est un terme beaucoup utilisé dans le domaine de web sémantique. Il constitue d'une base de données RDF sous forme de triplets, un graphe de connaissance est alors formé des nœuds et des flèches, les différents nœuds sont des entités, alors que les flèches représentent la relation entre les différentes entités. Un triplet est donc formé d'un objet, une prédicat et un sujet, par exemple dans la déclaration *Da Vinci a dessiné la Mona Liza* nous trouvons qu'il y a deux entités (Da Vinci et Mona Liza) et la relation (a dessiné), donc <objet, prédicat, sujet>. Un Graphe de connaissance est alors contient plusieurs triplets. Les GCs facilitent les recherches sur l'internet en construisant un lien entre les différentes entités, ils sont aussi utilisés pour les taches recommandations [4],[6].

Malgré tout, ces graphes de connaissances ne sont pas complets, ils manquent des entités et de relations, et il y a encore de l'ambiguïté. La tâche de complétion de graphe permet de résoudre ce problème, cela peut être fait par l'alignements des entités ou des relations ou plus loin l'alignement des graphes. Pour faire cet alignement il faut alors que nous ayons des informations sémantiques sur les entités que nous voulons aligner. Une des informations que nous pouvons savoir sur les triplets d'un graphe ce sont leurs représentations vectorielles en utilisant l'apprentissage de représentation du graphe (KRL). Le KRL nous permet de savoir la distribution des vecteurs de basse dimension des entités et des relations dans une espace. Il y a des différents espaces tel qu'espace multiple, espace vectoriel et espace vectoriel complexe comme cité dans [4], je parlerai des modèles plus en détail dans section 2. Ces vecteurs peuvent être donc classifier ce qui va nous donner des informations en plus sur ces entités ou ces relations et aussi nous permet de comprendre comment ces vecteurs sont représentés et qu'est ce qu'elles signifient.

Dans le rapport bibliographique, nous avons trouvé que nous pouvons arriver à compléter les graphes en utilisant l'analogie proportionnel. Nous trouvons aussi que par l'analogie, nous pouvons déduire des liens entre les différents graphes et c'est ce que nous allons voir dans la première section. Ce rapport présente alors la partie réalisation, où j'applique une méthode d'apprentissage automatique en utilisant des différents modèles pour convertir les graphes de connaissances a des vecteurs qui peuvent être utiliser pour construire des partitions, ce qui me permet d'aligner des différents graphes à partir d'analogie sur les partitions des entités.

Dans la première section, je présente ce qu'est l'alignement de graphe et pourquoi l'analogie peut être intéressante pour cette tâche et par rapport à ça je pose mon hypothèse. Dans la deuxième partie je développe mon hypothèse, je présente aussi la méthode, les outils utilisés et l'application de l'hypothèse, je commencerai par les données choisis et la méthode de requête, je parlerai ensuite de l'application d'apprentissage automatique pour faire les embeddings des graphes et les modèles utilisés, et je finirai par la méthode de construction des partitions et l'application de l'analogie sur ces partitions. Ensuite, je présente les résultats obtenus par la réalisation de la tâche, nous voyons les résultats des partitions et des analogies appliquées par des différents modèles d'embeddings et sur les différents graphes. Enfin, Je finirai par discuter le travail en tout, les résultats et comment ces observations peuvent nous aider à avancer dans les recherches sur les graphes de connaissances et ce que nous pouvons faire pour aller plus loin.

2 Alignement des graphes

L'alignement des entités est un domaine qui a bien attiré le monde de recherche, il permet de compléter les graphes non complets ou qui contiennent de l'ambiguïté. Cet alignement peut être

distingué par l'équivalence de relation pour deux entités appartenant à deux graphes différents par rapport à une ontologie ou par des modèles calculant cette équivalence.
Il a alors des différentes méthodes pour trouver cette équivalence, il y a la méthode de voisinage cité dans l'article [3] dont nous pouvons distinguer l'équivalence des entités en prenant en compte leurs voisins, théoriquement, nous disons que les entités contenant beaucoup des entités voisines communs sont similaires. Par cette méthode nous pouvons trouver les entités et les relations manquantes dans un graphe et nous pouvons aussi l'utiliser pour la translation linguistique des graphes.
Une autre méthode est de représenter les entités sous formes des vecteurs de basse dimension basant sur ses structures sémantiques et structurales [10]. Cela permet à comparer les vecteurs des entités et trouver les entités équivalentes en calculant la similarité entre les entités. Dans la première méthode, les chercheurs utilisent des modèles probabilistes, ils supposent que si deux entités a et b, l'entité a du graphe $G1$ et l'entité b du graphe $G2$ ont des voisines similaires, alors il y a une forte probabilité que les deux entités sont les mêmes[10]. Alors que dans la deuxième méthode nous basons sur la sémantique des entités, ils comparent alors les vecteurs basant sur les caractéristiques importantes des entités, en conséquence, les entités qui ont des caractéristiques similaires sont similaires [10].

2.1 L'alignement et l'analogie

Un graphe de connaissance est un ensemble d'entités, des relations et des faits, il est défini comme $G = E, R, F$ dont E est un ensemble des entités, R est un ensemble de relation et F est un ensemble de faits. Dans le cas d'alignement des entités, nous cherchons dans un graphe G_1 et le graphe G_2 les entités qui sont équivalents. Si nous avons E_1 qui est l'ensemble des entités du grpahe $G1$ et E_2 qui est l'ensemble des entités d'un autre graphe G_2 nous cherchons alors $e_1 \in \mathcal{E}_1 \equiv e_2 \in \mathcal{E}_2$. Si nous trouvons $e_1 \equiv e_2$ nous disons alors que ces deux entités sont alignées [10].

L'analogie est une méthode pour dire que « A est à B comme C est à D » [5], par exemple nous pouvons dire que Chien est a chienne comme chat est à chatte ». L'analogie est formé de deux paires et la déclaration « A est à B comme C est à D » est noté $A : B :: C : D$. Une analogie est valide si les deux paires (a, b) et (c, d) font référence à la même situation et ils sont en correspondances [7]. En utilisant l'analogie proportionnelle nous pouvons trouver alors une autre forme d'alignement. cette forme d'alignement est alors basée sur la relation entre les entités et non pas l'équivalence des entités. En citant que l'analogie signifie que $A : B :: C : D$ nous pouvons dire que deux paires des entités sont en alignement s'ils possèdent la même relation. Si alors e_1 et $e_2 \in E1$, e_3 et $e_4 \in E_2$ et $e_1 : e_2 :: e_3 : e_4$ nous pouvons alors dire que e_1 et e_3 sont alignées et e_2 et e_4 sont alignées. Cet alignement est alors le résultat d'une similarité des relations entre le premier et le deuxième paire des entités. Dans ce cas, ces deux entités sont alignées sans être équivalents, mais ils ont la même relation. L'analogie proportionnelle pour l'alignement des graphes est utilisé dans des différents domaine comme la complétion des graphes ou la recommandation [5].

Dans ce rapport, basant sur les anciens travaux je propose une nouvelle méthode pour détecter des analogies entre les différents graphes de connaissance. Cette méthode a alors comme objectif de détecter des relations entre deux paires d'ensemble d'entités de deux graphes en utilisant l'analogie proportionnelle. Pour réussir à faire appliquer l'analogie entre deux paires d'ensemble d'entités je commence par utiliser les méthodes de représentation des graphes pour convertir

les entités à des vecteurs de basse dimension. Ensuite, j'utilise la méthode des partitions K-means pour construire K ensembles des entités en utilisant sur la similarité entre les vecteurs d'embeddings représentant les entités. Enfin, je prends les ensembles de partitions des graphes et essayer de voir si une analogie peut être détectée entre deux paires de partitions.

2.2 Analogie entre les partitions

Le *Patitions K-means* est une méthode d'apprentissage automatique non-supervisé qui permet de construire des classes (partitions) par rapport à la similarité entre les éléments, plus les éléments ont des caractéristiques similaires et sont plus proche dans le plan vectoriel alors c'est plus probable qu'il appartenant dans la même classe. Le « K » signifie le nombre des partitions construits, ce nombre est alors manipulé jusqu'à ce qu'on arrive à un nombre optimal dont nous trouvons un équilibre entre la variance et le biais des partitions. Pour décider quel élément appartient à quel groupe, l'algorithme de K-means clustering choisit par hasard des points qui sont appellés *centroïdes* [8]. En calculant la distance euclidienne entre un élément et le centroïde, la distance la plus courte indique que l'élément appartient au groupe de ce centroïde. Un bon partition contient alors des éléments qui sont homogènes [8].

Dans le cas des graphes de connaissances, nous pouvons en utilisant des méthodes de représentation des graphes tel que *les modèles de distance translationnelle* de construire des partitions des entités qui permet de faire des groupements des entités les plus similaires et qui ont les mêmes caractéristiques. Cela signifie qu'une partition contient alors une partie des entités du graphe qui ont des caractéristiques similaires et spécifiques, qui peuvent être leurs relations, leur signification tel que « page », « étiquette », etc. Par exemple, si nous avons un GC sur les différents livres écrits par des écrivains français, nous attendons qu'il y ait par exemple des partitions qui groupent les livres et d'autres qui groupent les auteurs car ils ont des relations et des significations différents.
Par conséquence, en distinguant les relations entre les partitions fourni dans un graphe nous pouvons arriver à détecter des analogies entre deux paires des partitions entre deux graphes.

Pour distinguer et comprendre la relation entre les partitions d'un graphe, une des méthodes c'est de calculer la distance entre deux centroïdes, cela permet de comprendre les différences entre les deux partitions. L'idée principale c'est que si on cite C_1 et C_2 deux partitions de $G1$, C_3 et C_4 deux partitions de G_2 alors $C_1 : C_2 :: C_3 : C_4$ si les distance entre les deux partitions de chaque est proportionnelle.

Il y a alors des différentes méthodes pour appliquer l'analogie. Nous pouvons dire que deux paires sont proportionnelles s'ils ont la plus grande différence ou la plus petite différence. Ça veut dire que si C_1 et C_2 ont la distance euclidienne la plus grande et C_3 et C_4 ont la distance euclidienne la plus grande donc $C_1 : C_2 :: C_3 : C_4$. Cela nous permet alors de trouver des relations entre les groupes des entités dans un graphe et les comparer a des autres groupes des entités d'un autre graphe.

Alors que seulement la distance ne sera pas possible pour faire l'analogie, j'ai décidé de faire des classements des partitions par rapport au partition central, la partition le plus proche est le premier dans le classement, alors que celui qui est plus loin est le dernier.

Pour choisir la partition central, je calcule la distance euclidienne entre tous les partitions et pour chaque partition je calcule la moyenne des distances et celui qui a la moyenne la plus faible

est alors la partition centrale, je note la partition central N. Ensuite, à propos de cette partition centrale je commence a construire le classement $C_1, \ldots, C_z$ dont C_1 est la partition le plus proche au partition central alors que C_z est la partition le plus loin de partition central.

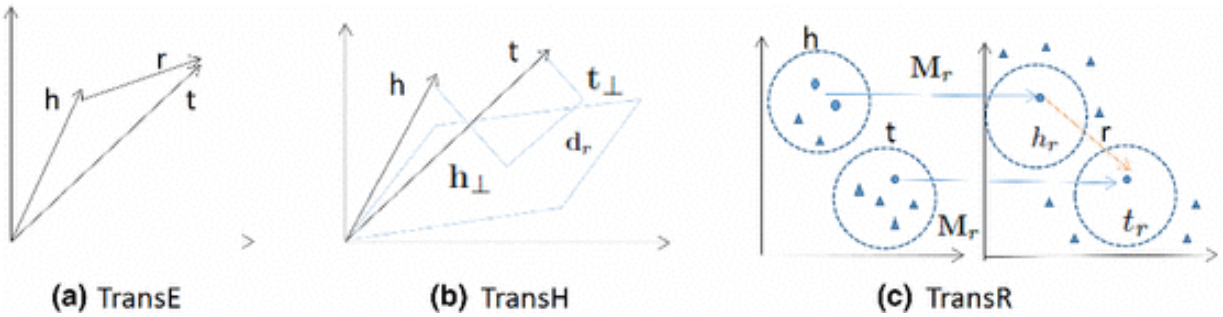
En basant sur ces partitions je propose l'analogie. Nous notons N_1 la partition centrale du G_1 et N_2 la partition centrale du G_2. Dans ce cas, nous disons que $N_1 : C_n :: N_2 : P_m$ si $n = m$, dont C_n est un partition de G_1, n son ordre de classement et P_m est un partition de G_2 et m son ordre de classement. Donc deux paires sont en en analogie si la relation entre la partition centrale N et la partition C est la même.

2.3 modèles de distance translationnelle

2.3.1 TransE

TransE (fig 1, a) est un modèle de représentation de graphe qui a comme objectif de représenter les entités et les relations sous formes des vecteurs. C'est un modèle qui représente les triplets d'un graphe dans une espace vectoriel réel. TransE représente tous les triplets dans une espace R^k des basses dimensions. TransE est principalement un modèle translationnelle, il présente le triplet h, r, t (head, relation, tail) dans l'espace si le triplet est vrai. Si le triplet est vrai, l'entité t doit être le plus proche possible de l'entité h. Dans ce cas, la relation est alors une translation de l'entité h vers l'entité t dont $h + r \approx t$ [4] [2]. Alors que TransE est un bon modèle de distance translationnelle, mais il n'a pas des résultats significatifs pour les graphes dont il y a des multi-relation, dont une entité est liée à plusieurs entités avec la même relation.

FIGURE 1 – les modèles de distance translationnelle [9]



2.3.2 TransH

TransH (fig 1, b) est alors une extension du TransE qui permet de résoudre la limite de multi-relation. TransH suppose que chaque relation fait partie d'un hyperplan déférent et projette les entités h et t dans l'hyperplan dont ils font partie. La plausibilité d'un triplet est alors noté $h^\perp = h - W_r^\top h w_r, t^\perp = t - W_r^\top t w_r$ [2]. Chaque hyperplan contient alors les caractéristiques et les propriétés d'une relation spécifique, cela permet d'attribuer à chaque entité t un *« embedding »* différent même s'ils contiennent la même relation ou la même entité h. La limite de TransH est qu'il suppose encore que les vecteurs des entités et de relation font partie de la même espace vectorielle.

2.3.3 TransR

TransR (fig 1, b) applique les *embeddings* des relations et des entités dans des espaces différentes. Les vecteurs des entités sont alors fait dans une espace R^k dont $(h,t) \in R^k$ [2] alors que les relations font partie d'une autre espace R^d dont $r \in R^d$. La fonction du score est alors notée $s(h,r,t) = -||h^{\perp} + r - t^{\perp}||_2^2$.

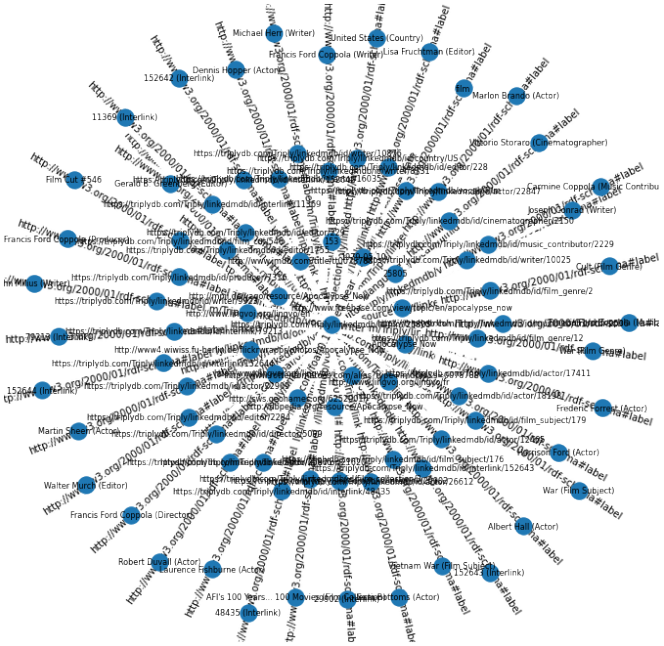
Dans ce projet, j'utilise les différents modèles de distance translationnelle pour voir quel modèle est le plus efficace pour construire les partitions et appliquer l'analogie proportionnelle. L'objectif est alors de récupérer des données RDF et les convertir à des vecteurs en utilisant les différents modèles. Après ça je choisis le modèle le plus efficace et l'utiliser pour détecter des analogies entre deux paires de partitions de deux GCs différents.

3 Méthode

3.1 Traitement des données

Pour récupérer les données j'ai utilisé la langage SPARQL sur les éditeurs SPARQL de *DBpedia* et *TriplyDB* pour récupérer des données RDF sur des films. Pour tester les graphes sur des modèles de représentation de graphes j'ai commencé par un test sur le GC du film *Logan*, ce graphe contient 91 triplets (fig 2). J'ai testé l'embedding de ce graphe et nous trouvons que c'était possible en utilisant le modèle TransE et que je peux aussi avoir la graphe pour le taux de perte.

FIGURE 2 – Graph de connaissance du film Apocalypse Now du TriplyDB



Pour passer vers le vrai application, j'ai récupéré des données contenant des différents film de TriplyDB en utilisant un *requête SPARQL* ent utilisant la bibliothèque *RDFlib* sur *Python*. Les résultats de requête sont alors enregistrés dans un fichier CSV pour que je l'utilise dans l'embedding, le fichier contient alors 2000 triplets. Pour utiliser les modèles de représentation de graphe, j'ai utilisé la bibliothèque *Pykeen*. Je divise en premier temps les données en 80% pour l'entrainement, 10% pour le test et 10% pour la validation. J'ai commencé en utilisant le modèle TransH, la longueur de vecteurs (n) est mise une fois a 50 et une fois a 75 [1]. Le nombre d'époque était décidé par la manipulation pour trouver le nombre optimum, le nombre d'époque (e) choisit était 15 parce que le taux de perte commence à augmenter après ce nombre. Le taux d'apprentissage (α) de modèle était 0.01 par défaut (constante pour tous les essayes) .

En utilisant cette méthode, ce n'était pas possibles d'entraîner le modèle. L'entrainement du modèle sur des données contenant des triplets des films seulement n'était pas suffisante pour que le modèle requérir toutes les informations importantes pour l'utiliser dans les étapes de test et de validation. Pour résoudre ce problème, j'ai récupéré des données RDF sur 10000 films et utiliser le même modele pour faire les embeddings avec les mêmes valeurs pour n, α et e. Mais même avec 10000 triplets ça n'était pas encore suffisant et j'ai rencontré le même problème. Le problème n'était pas alors le nombre de triplets, mais la complexité et la quantité des données.

Avec l'aide de mon encadrant, j'ai utilisé un SPARQL requête pour récupérer des données du TriplyDB qui contiennent des informations diversifiées tel que des films, des acteurs, performance, etc. Le fichier CSV contient alors 17416 triplets, tous en anglais (fig 3). En utilisant la bibliothèque *Pandas* je verifie qu'il n'y a pas des valeurs nulle, s'il y aura je l'efface mais dans ce cas aucune valeur a été effacé. En utilisant *Pykeen* le tableau est alors convertir aux triplets qui peuvent être lire par les modèles du *Pykeen*.

FIGURE 3 – Les dix premières lignes des données TriplyDB utilisés

	Head	Relation	Tail
0	https://triplydb.com/Triply/linkedmdb/id/film/...	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	https://triplydb.com/Triply/linkedmdb/vocab/Film
1	https://triplydb.com/Triply/linkedmdb/id/film/...	https://triplydb.com/Triply/linkedmdb/vocab/actor	https://triplydb.com/Triply/linkedmdb/id/actor...
2	https://triplydb.com/Triply/linkedmdb/id/film/...	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	https://triplydb.com/Triply/linkedmdb/vocab/Film
3	https://triplydb.com/Triply/linkedmdb/id/film/...	https://triplydb.com/Triply/linkedmdb/vocab/actor	https://triplydb.com/Triply/linkedmdb/id/actor...
4	https://triplydb.com/Triply/linkedmdb/id/film/...	https://triplydb.com/Triply/linkedmdb/vocab/actor	https://triplydb.com/Triply/linkedmdb/id/actor...
5	https://triplydb.com/Triply/linkedmdb/id/film/...	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	https://triplydb.com/Triply/linkedmdb/vocab/Film
6	https://triplydb.com/Triply/linkedmdb/id/film/...	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	https://triplydb.com/Triply/linkedmdb/vocab/Film
7	https://triplydb.com/Triply/linkedmdb/id/film/...	https://triplydb.com/Triply/linkedmdb/vocab/actor	https://triplydb.com/Triply/linkedmdb/id/actor...
8	https://triplydb.com/Triply/linkedmdb/id/film/...	https://triplydb.com/Triply/linkedmdb/vocab/actor	https://triplydb.com/Triply/linkedmdb/id/actor...
9	https://triplydb.com/Triply/linkedmdb/id/film/...	https://triplydb.com/Triply/linkedmdb/vocab/actor	https://triplydb.com/Triply/linkedmdb/id/actor...
10	https://triplydb.com/Triply/linkedmdb/id/film/...	https://triplydb.com/Triply/linkedmdb/vocab/actor	https://triplydb.com/Triply/linkedmdb/id/actor...

Pour les embeddings j'ai utilisé en premier temps TransE avec $l = 75$ et $e = 15$. En utilisant la bibliothèque *Pytorch* je récupère les vecteurs et leur étiquette (l'entité qui correspond à ce vecteur), cet étape va permettre d'appliquer les partitions K-means et les visualiser. Pour TransH

FIGURE 4 – Les dix premières lignes des données DBpedia utilisés

	subject	predicate	object
0	http://dbpedia.org/ontology/Place	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#Class
1	http://dbpedia.org/ontology/Place	http://www.w3.org/2000/01/rdf-schema#subClassOf	http://www.w3.org/2002/07/owl#Thing
2	http://dbpedia.org/ontology/Place	http://www.w3.org/2002/07/owl#equivalentClass	http://dbpedia.org/ontology/Location
3	http://dbpedia.org/ontology/Place	http://www.w3.org/2002/07/owl#equivalentClass	http://schema.org/Place
4	http://dbpedia.org/ontology/Place	http://www.w3.org/2000/01/rdf-schema#label	περιοχή
5	http://dbpedia.org/ontology/Place	http://www.w3.org/2000/01/rdf-schema#label	مكان
6	http://dbpedia.org/ontology/Place	http://www.w3.org/2000/01/rdf-schema#label	place
7	http://dbpedia.org/ontology/Place	http://www.w3.org/2000/01/rdf-schema#label	Ort
8	http://dbpedia.org/ontology/Place	http://www.w3.org/2000/01/rdf-schema#label	lugar
9	http://dbpedia.org/ontology/Place	http://www.w3.org/2000/01/rdf-schema#label	miejsce
10	http://dbpedia.org/ontology/Place	http://www.w3.org/2000/01/rdf-schema#label	sted

$l = 150$ et $e = 30$. Alors que pour le modèle TransR $l = 150$ et $e = 15$. Pour les trois modèles les donnees sont divisés en 80% pour l'entrainnement, 10% pour le test et 10% pour la validatione, et $\alpha = 0.01$.

Le deuxième graphe a été construit en utilisant les données de DBpedia. En utilisant SPARQL et la bibliothèque RDFlib, j'ai construit un sous-graphe contenant des données tel que des films, les noms des films et des différents informations comme des places,genre, type,etc. Le graphe contient 10000 triplets et il était enregistré dans un fichier CSV (fig 4). Au contraire des données récupérées de TriplyDB, les données récupérées de DBpedia contiennent des différentes langues que l'anglais, le cause de cette différence est que si les données sont seulement en anglais ils ne seront pas efficaces pour l'entraînement du modèle.
La même méthode d'entraînement des modèles a été utilisé pour ces données autant que les données du TriplyDB. J'ai appliqué les trois modèles sur ces données avec les mêmes valeurs de l et e utilisés pour le premier graphe.

3.2 K-means

Après avoir les différents vecteurs par les différents modèles, j'utilise les bibliothèques *Pytorch* et *tensorflow* pour attribuer les étiquettes des entités à leur vecteur spécifique. Cette information est ensuite enregistrée dans un dictionnaire qui contient l'étiquette et son propre vecteur. Cela permet alors d'appliquer les partitions K-means sans perdre les étiquettes des vecteurs. Les partitions sont alors construits en utilisant la bibliotheque *Scikit Learn*, les centroïdes initiaux sont choisis par hasard, alors que le nombre de partitions (k) est choisi par les essais. Après la manipulation de K, je trouve que les données sont assez représentables pour les deux graphes et peuvent-être utilisé pour faire l'analogie quand $K = 3$. La valeur de K diffère selon le modèle mais en général $K = 3$ est utilisé pour TransH et nous voyons dans la partie suivante pourquoi cela est le plus efficace. Pour visualiser les partitions, j'applique *L'analyse en composantes principales (PCA)* sur les vecteurs pour réduire la dimension des vecteurs, le nombre des composants

est alors fixé à deux. La bibliotheque *Matplotlib* a été utilisé pour tracer le graphe et visualiser les partitions.

Tandis que les partitions sont visibles, mais la visualisation seulement n'était pas toujours efficace pour déduire la partition central. Alors en utilisant la bibliotheque *Scipy* j'ai calculé la distance euclidienne entre tous les partitions, et celui qui contient la moyenne de distance la plus petite est alors la partition central. Le classement des partitions est alors fait à partir de ce partition central, dont C_1 est la partition qui est la partition le plus proche au partition central par rapport a la distance euclidienne calculé, et la partition C_7 est alors la partition le plus loin du partition central par rapport a la distance euclidienne calculé.

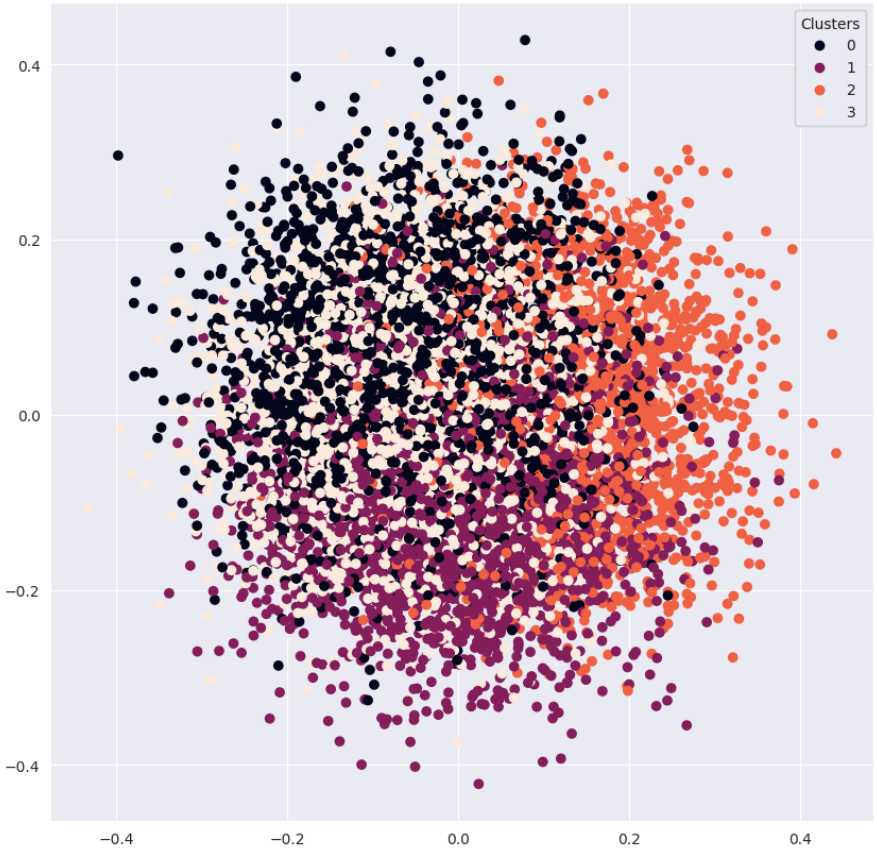
Les classements des deux graphes sont alors utilisés pour faire les analogies entre deux paires de partitions. La première paire contient la partition central du premier graphe et un autre partition, la deuxième paire contient la partition central du deuxième graphe et un autre partition. Les deux partitions utilisés pour détecter l'analogie doivent avoir le même *rang* dans le classement. C'est-à-dire que $N1 : C_m :: N2 : C_n$ si $n = m$.

Dans ce cas la relation entre nous pouvons détecter des analogies entre les différents ensembles des entités dans des différents graphes. Nous disons que dans un graphe $G1$ et $G2$ nous trouvons une analogie entre deux paires d'ensemble d'entités. Dans la partie suivante nous voyons les résultats des partitions K-means en utilisant les différents modèles de représentation du graphe et les résultats des analogies.

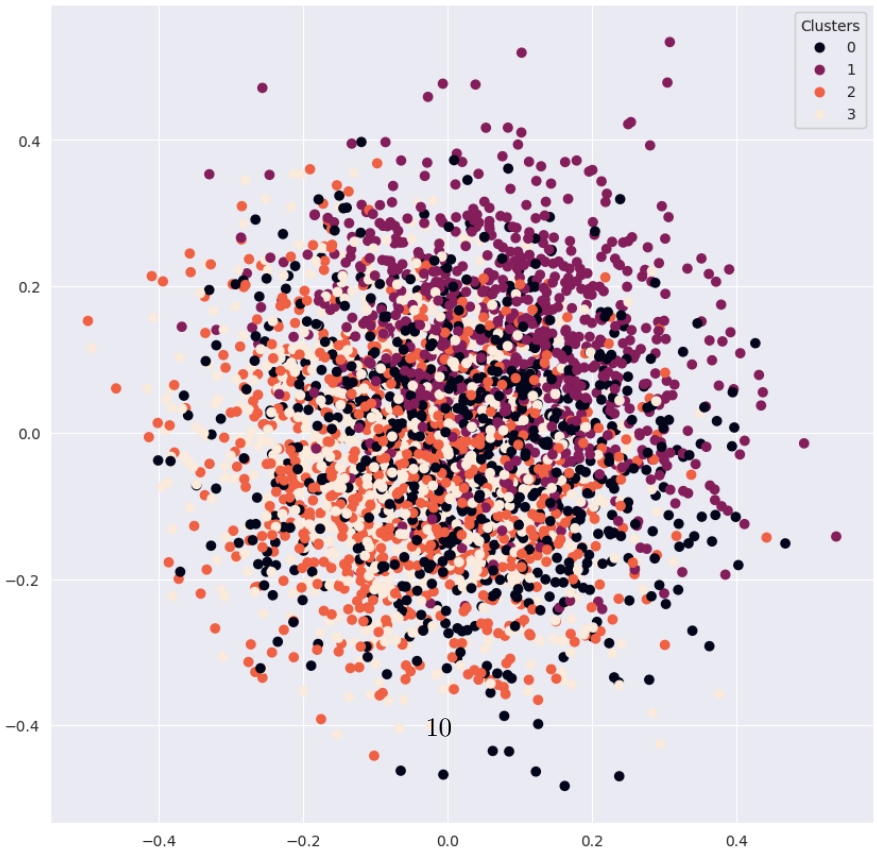
4 Résultats

Dans cette section je présente les différentes partitions construites en utilisant les différents modèles de représentation de graphe, nous voyons que les modèles ne sont pas tous utiles pour appliquer les partitions K-means et qu'il y a des modèles qui donnent des meilleurs résultats que les autres, au moins pour les données utilisées. Nous trouvons que pour le modèle TransE, les embeddings n'étaient pas efficace pour construire des partitions significatives. Quand nous regardons le graphe contenant les vecteurs des entités du graphe de TriplyDB (fig 5,a), nous trouvons qu'il n'y a pas une grande différence entre les partitions, dont la construction des partitions k-means n'a pas vraiment de signification. Une partition contient alors des entités différentes et nous voyons que toutes les partitions contiennent des entités similaires. Nous trouvons aussi que dans le graphe du modèle TransE pour les données du TriplyDB (fig 5,a) et DBpedia (fig 5,b) les embeddings des vecteurs et les partitions sont assez similaire. En conséquence, quand je calcule les distances euclidiennes entre les partitions je trouve que la partition ayant la moyenne de distance la plus petite était la partition 3, avec une moyenne de 0.188, alors que la partition 2 a la moyenne la plus grande avec une moyenne de 0.197. Cette grande variance dans les partitions mais qui est petite entre eux ne permet pas de faire des analogies fiables. Alors que pour les données de DBpedia la partition ayant la moyenne la plus petite était la partition 2, avec une moyenne de 0.232, alors que la partition ayant la moyenne la plus grande était la partition 1, avec une moyenne de 0.236. Cette grande variance dans les partitions mais qui est petite entre eux ne permet pas de faire des analogies fiables.

Le modèle TransH a attribué les embeddings les plus utiles pour construire les partitions K-means et pour appliquer l'analogie. Nous trouvons que les embeddings sont différents, et qu'il y a des différents ensembles des entités dans des espaces différents ce qui rend les partitions



(a) Résultats du modèle TransE pour les données de TriplyDB.



(b) Résultats du modèle TransE pour les données de DBpedia.

FIGURE 5 – Résultats du modèle TransE Avec la partition K-means avec $K = 4$.

plus claires. Au contraire du modele TransE, nous pouvons facilement la différence entre les embeddings du graphe de TriplyDB (fig 6, a) et les embeddings du graphe de DBpedia (fig 6, b). Alors que les deux graphes ne contiennent pas les mêmes ensembles des entités, le nombre des partitions ne sera pas aussi la même. Par conséquence, j'ai décidé de manipuler les nombres des partitions K pour trouver la meilleure valeur du K pour que les partitions soit au minimum représentable des ensembles des entités et aussi peut être utilisé pour appliquer l'analogie, la meilleure valeur du K était 3.

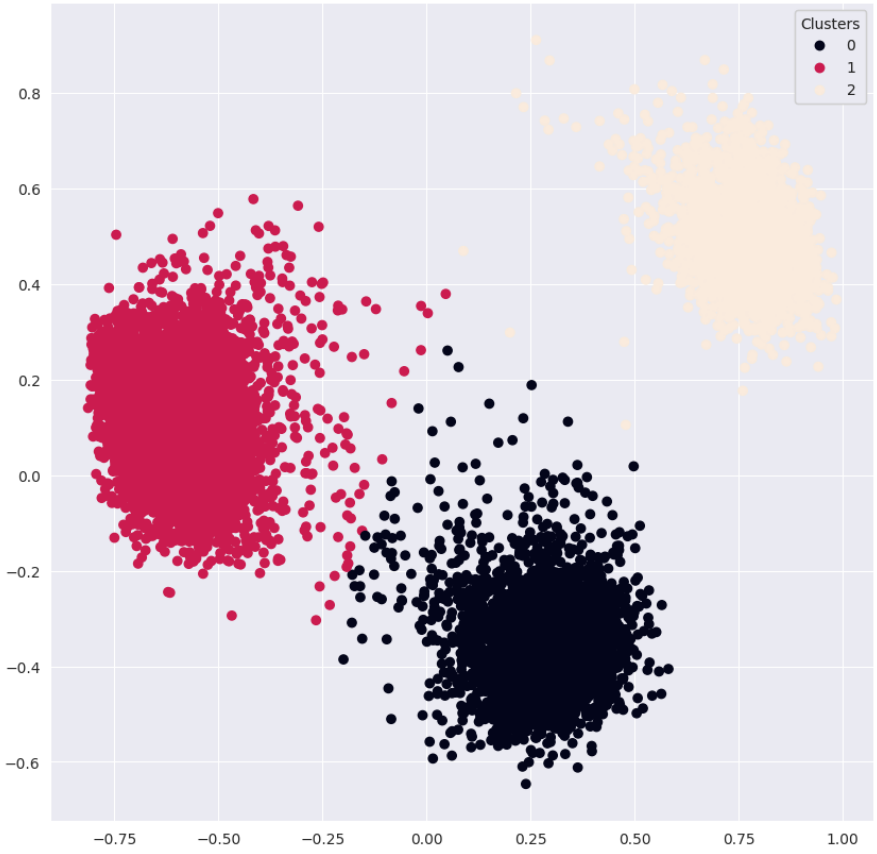
Si nous regardons les éléments des partitions K-means (fig 7) faits en utilisant les embeddings du modèle TransH, nous trouvons que les partitions contiennent des éléments spécifiques. Nous trouvons que la variance dans les partitions et la variance entre les partitions sont plus importantes que le modèle TransE, alors que la variance entre les partitions est plus importante. Pour les partitions K-means faits sur le graphe du TriplyDB, la partition 0 avait la moyenne de distance la plus petite avec une distance de 0.663, alors que la partition 1 avait la moyenne la plus grande avec une distance de 0.803. Alors que pour les partitions faites sur le graphe de DBpedia, la partition qui avait la moyenne de distance la plus petite était la partition 1 avec une moyenne de 0.521, alors que la partition ayant la moyenne la plus importante était la partition 1 avec une moyenne de 0.630. Nous distinguons alors que la variance entre les partitions est plus importante pour le modèle TransH que le modèle TransE ce qui lui rend plus fiable pour construire des partitions K-means qui seront utiles pour faire des analogies.

Le modèle TransR attribue des résultats qui sont meilleurs que TransE mais moins interpretable que le TransH. La variance dans les partitions étaient plus petites que dans TransE mais plus grande que dans les partitions du TransH. Alors que la variance entre les partitions était plus important que TransE mais moins important que pour les partitions de TransH. Pour les données de TriplyDB, la partition 2 avait la moyenne de distance la plus petite avec une moyenne de 0.201, alors que la partition 0 avait la moyenne de distance la plus petite avec une moyenne de 0.219. Pour les données du DBpedia, la partition 2 avait la moyenne de distance la plus petite avec une moyenne de 0.182, alors que la partition 0 avait la moyenne de distance la plus importante avec une moyenne de 0.213.

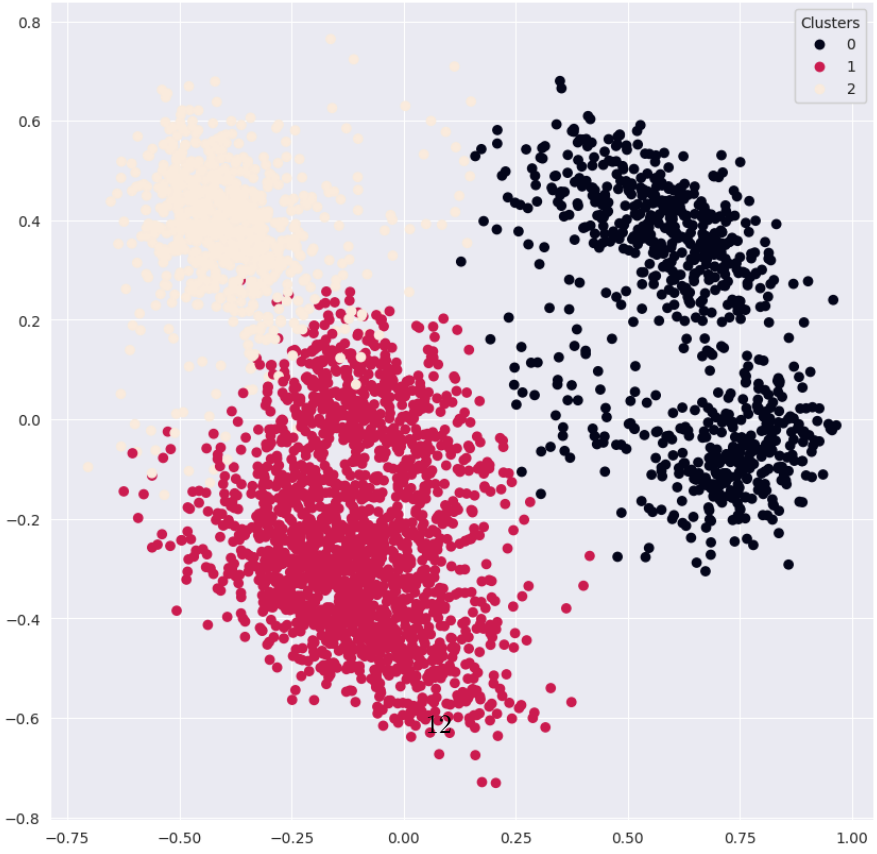
D'après ces résultats, j'ai décidé de faire l'analogie sur les partitions construites en utilisant les embeddings du modèle TransH. La forme d'analogie sera alors $N_1 : C_1 :: N2 : P_1$. Nous prenons comme partitions centrales la partition 0 pour pour le premier graphe et la partition 1 pour le deuxième graphe. Pour le deuxième élément des paires nous prenons les partitions les plus proche à la partition centrale, donc les partitions qui sont les premiers dans le classement, ces éléments sont alors la partition 2 pour le premier graphe et la partition 2 pour le deuxième graphe. Les deux partitions centrales contiennent des entités qui sont des films ou des titres des films. Alors que la partition 1 du premier graphe contient des entités concernant les acteurs, et la partition 2 du deuxième graphe contient des différentes entités concernant des informations sur des films, comme les pays, la place, mais aussi une grande partie des éléments contient des informations sur des *personnes* (fig 8) tel que les directeurs, compositeurs des musiques.

5 Conclusion

Si nous regardons les résultats, nous pouvons distinguer les différences entre les différents modèles. Le modèle TransH est alors le modèle le plus efficace pour construire des partitions K-means sur les entités d'un graphe de connaissance. Cela peut être expliqué par la capacité du TransH à attribuer les entités dans des espaces différents a propos de leurs relations[2]. Nous



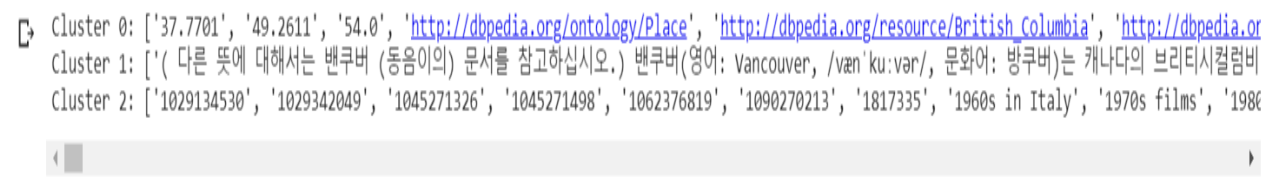
(a) Résultats du modèle TransH pour les données de TriplyDB.



(b) Résultats du modèle TransH pour les données de DBpedia.

FIGURE 6 – Résultats du modèle TransH Avec la partition K-means avec $K = 3$.

(a) Les différentes partitions et un exemple de leurs éléments, TriplyDB.



(b) Les différentes partitions et un exemple de leurs éléments, DBpedia.

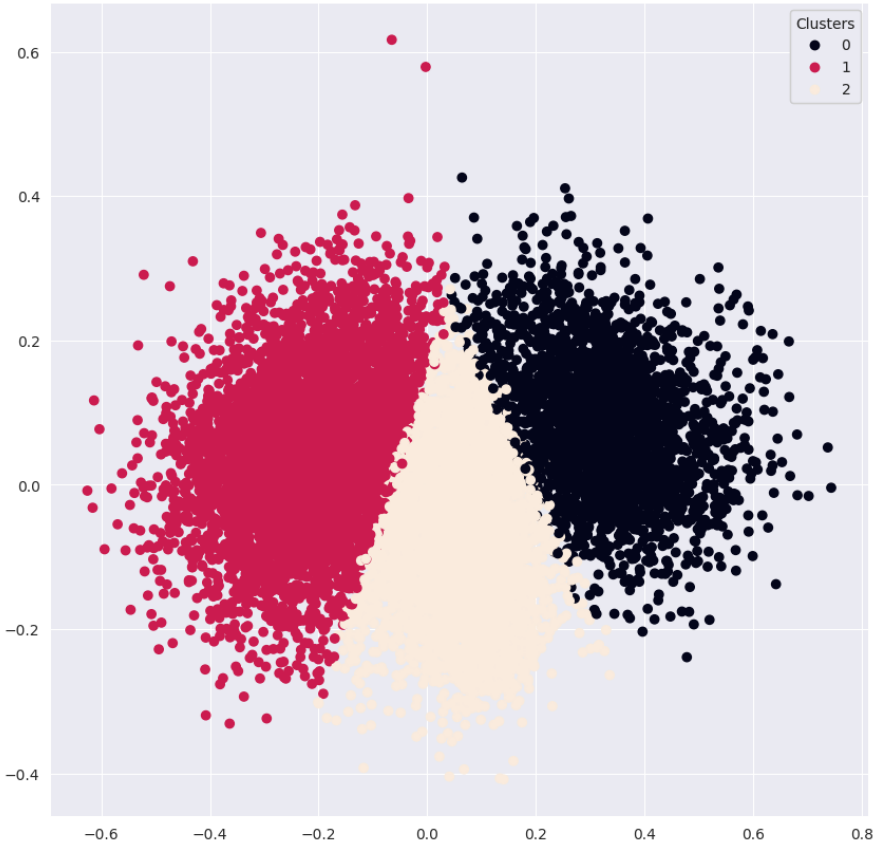
FIGURE 7 – Les différentes partitions K-means pour les deux graphes et un exemple de leurs éléments, $K = 3$.

trouvons aussi que la partitions K-means construites en utilisant les embeddings du TransH ont réussit à diviser les ensembles des entités films et l'ensemble d'entité acteur, ce que nous ne le trouvons pas en utilisant les embeddings du TransE ou TransR fig(9).

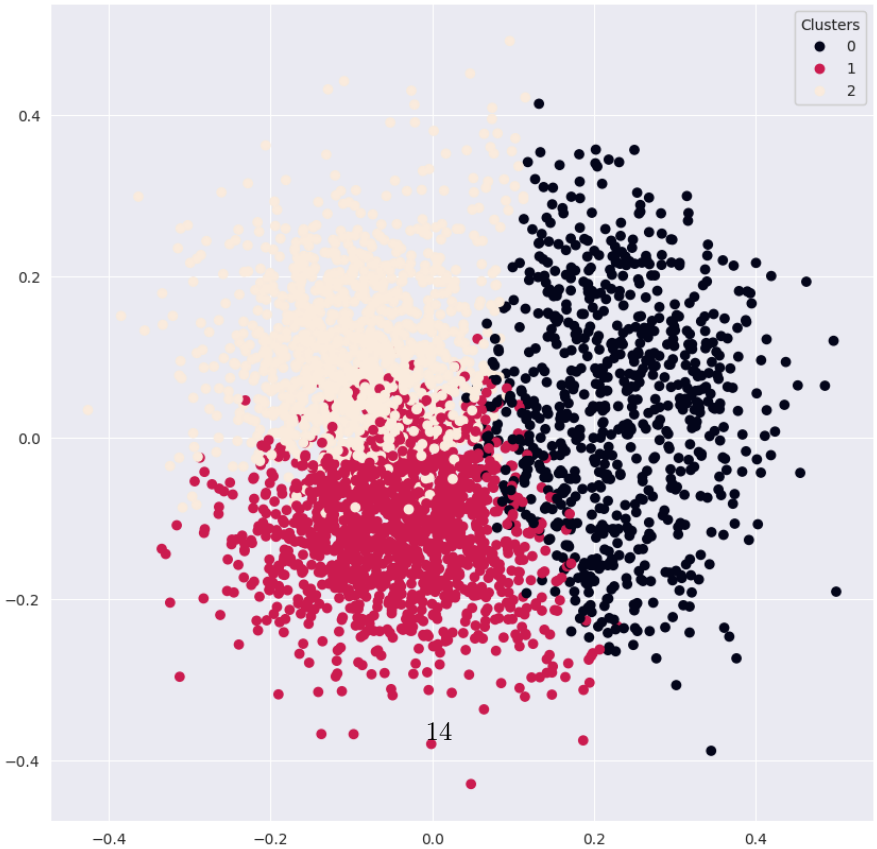
Concernant l'analogie, pour les partitions appartenant au deuxième rang (partition 1 pour les données de TriplyDB et 0 pour DBpedia) ça ne sera pas possible de dire qu'il y a une analogie a cause de la grande différence entre les éléments de deux partitions. Alors que pour les partitions de première rang (partition 2 pour les données de TriplyDB et la partition 2 pour les données de DBpedia) les deux partitions contiennent des entités qui sont de types *personne* tel que les acteurs, les directeurs, les cinéastes,etc. Mais en même, ça reste encore difficile de dire que l'analogie est valide a cause de la grande différence entre les deux graphes et la variance existante dans la partition 2 du deuxième graphe.

L'application des partitions K-means appliqué sur deux graphes seulement avec a peu près 15000 et 10000 triplets peut être non efficace pour appliquer l'analogie. Si les données sont plus complexes et aussi les deux graphes contiennent informations similaires tel que acteurs, films, cinéastes, l'analogie peut être appliqué en utilisant les embeddings du modèle TransH.

L'application de l'analogie sur des partitions K-means des entités construites en utilisant les embeddings du TransH peut nous donner une meilleure compréhension sur la relation entre les entités et peut être peut-on détecter des informations manquantes dans les graphes. Cela peut nous permettre de déduire les ensembles des entités manquantes dans un graphe et percevoir différemment les ensembles des entités et leurs relations. Dans ce projet j'ai juste appliqué l'analogie sur des partitions construites sur les *entités*, ce qui nous met à proposer la question si cette application peut être aussi utile si on l'applique sur les relations.



(a) Résultats du modèle TransR pour les données de TriplyDB.



(b) Résultats du modèle TransR pour les données de DBpedia.

FIGURE 8 – Résultats du modèle TransR Avec la partition K-means avec $K = 3$.

(a)

(b)

(c)

FIGURE 9 – Exemple des éléments appartenant à la partition 2 de données Dbpedia qui sont des personnes.

Références

[1] Antoine BORDES et al. "Translating embeddings for modeling multi-relational data". In : *Advances in neural information processing systems* 26 (2013).

[2] Jiahang CAO et al. "Knowledge Graph Embedding: A Survey from the Perspective of Representation Spaces". In : *arXiv preprint arXiv:2211.03536* (2022).

[3] Deepak CHAURASIYA et al. "Entity Alignment For Knowledge Graphs: Progress, Challenges, and Empirical Studies". In : *arXiv preprint arXiv:2205.08777* (2022).

[4] Shaoxiong JI et al. "A survey on knowledge graphs: Representation, acquisition, and applications". In : *IEEE transactions on neural networks and learning systems* 33.2 (2021), p. 494-514.

[5] Pierre MONNIN et Miguel COUCEIRO. "Interactions Between Knowledge Graph-Related Tasks and Analogical Reasoning: A Discussion". en. In : (2022).

[6] Maximilian NICKEL et al. "A review of relational machine learning for knowledge graphs". In : *Proceedings of the IEEE* 104.1 (2015), p. 11-33.

[7] Henri PRADE et Gilles RICHARD. "Analogical Proportions: Why They Are Useful in AI". en. In : *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*. Montreal, Canada : International Joint Conferences on Artificial Intelligence Organization, août 2021, p. 4568-4576. ISBN : 978-0-9992411-9-6. DOI : 10.24963/ijcai.2021/621. URL : https://www.ijcai.org/proceedings/2021/621 (visité le 16/12/2022).

[8] Avgoustinos VOUROS et al. "An empirical comparison between stochastic and deterministic centroid initialisation for K-means variations". In : *Machine Learning* 110 (2021), p. 1975-2003.

[9] Jihong YAN et al. "Optimizing model parameter for entity summarization across knowledge graphs". In : *Journal of Combinatorial Optimization* 37 (2019), p. 293-318.

[10] Kaisheng ZENG et al. "A comprehensive survey of entity alignment for knowledge graphs". In : *AI Open* 2 (2021), p. 1-13.