Master 1 Informatique - Web Sémantique

TP2 - Construction d'un outil d'intégration de données

Hugo Maitre
Adrien Linares
Mitra Aelami
Amirhossein Nasri

Enseignant :
Pr. Konstantin Todorov

20 mars 2022

Table des matières

1 Introduction

L'intégration des données sémantiques est une composante fondamentale dans le domaine du web sémantique. Pour ce faire nous avons besoin d'ontologies, elles décrivent des concepts et des relations d'un domaine particulier.

Pour réaliser ce projet, il nous a été attribuées deux ontologies sous la forme de graphes rdf. Le graphe source rassemblait des oeuvres musicales de la Bibliothèque Nationale de France (BnF), le graphe target, quant à lui ceux de la Philarmonie de Paris. Enfin un benchmark (fichier d'alignements [1] de références) nous a été donné pour évaluer notre système.

Nous avons travaillé avec Python qui était le langage avec lequel le groupe se sentait le plus à l'aise pour ce type de travail. Le travail à été effectué sur Google Colab un environnement python en ligne qui permet l'utilisation de GPU à distance, pour certains calculs très longs cela s'est revelé être très utile. Notre travail, rendu sous le format d'un notebook, se trouve sur le dépot git ainsi que l'ensemble des fichiers nécessaires. ici.

1. ensemble de correspondances entre les éléments de deux ontologies hétérogènes

2 Développement

2.1 Preprocessing

Pour faciliter toute manipulation des graphes rdf, les graphes sont lus à l'aide de la librairie rdflib puis convertie en numpy array. Il est ensuite très aisé de parcourir le graphe puisque chaque triplet occupe une ligne de notre tableau nous manipulons donc des matrices de taille (n,3) où n (nombre de lignes) représente le nombre de triplet.

En ce qui concerne les prétraitements, on a décidé d'enlever tous les triplets comportant des objets de type noeuds comportant aucune information utile ainsi que les triplets ayant des objets sous forme d'URL. En faisant cela on est passé d'environ 10000 triplets dans chaque graphe à environ 2000.

J'ai également voulu traduire les données qui étaient en allemand italien où autre langues présentes ? vers le français à l'aide de DeepGoogleTranslator. Le coût de cette opération était très couteux donc pour éviter d'être effectué à chaque fois les données traduites étaient sauvegardées dans un fichier csv. Cependant cela n'a pas apporté de résultats significativement meilleurs ce prétraitement ne fais donc pas parti des prétraitements finaux.

Certains des prétraitements effectués pour l'UE Machine Learning 1 ont été réutilisés ici, notamment à l'aide de la librairie NLTK : tokénisation de chaque mot, suppression de la ponctuation et des stopwords, conversion en minuscule, lemmatisation de chaque mot, suppression des pronoms, déterminants verbes à l'infinitif...

Tous ces prétraitements ont été effectué sur les deux graphes.

2.2 Mesures de similarité utilisées

Nous avons pu retenir les similarités suivantes :

1. Identity similarity
2. Normalized Levenshtein similarity
3. JaroWinkler similarity
4. Jaccard similarity
5. Cosine similarity
6. Overlap coefficient similarity

7. Sørensen–Dice coefficient similarity
8. Sift4 similarity

SIFT4 est un algorithme de distance de chaîne à usage général inspiré de JaroWinkler et de Longest Common Subsequence. Il a été développé pour produire une mesure de distance qui correspond le plus possible à la perception humaine de la distance entre 2 mots. Plus de détail ici.

2.3 Comparaison et matching strategy

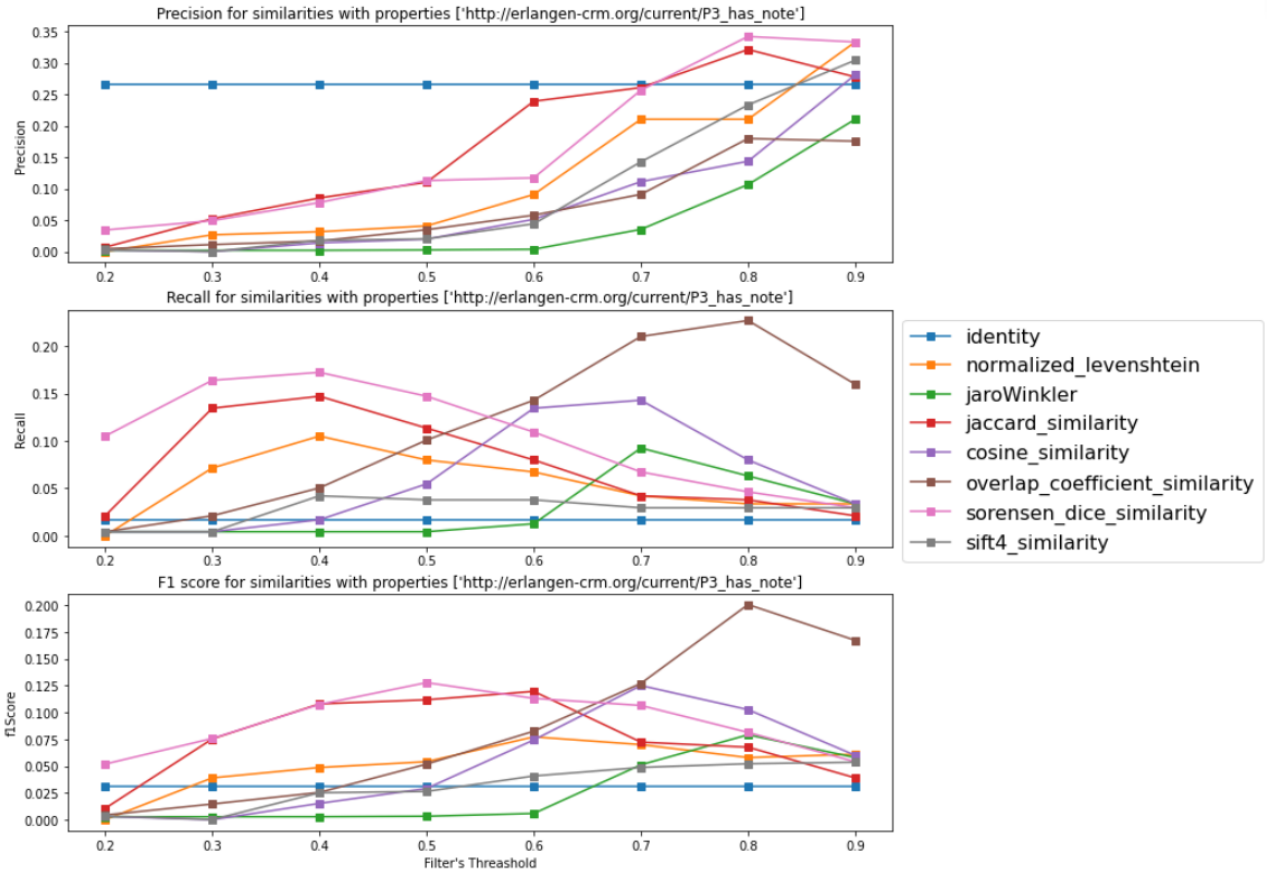
Premièrement les propriétés communes aux 2 graphes sont classées selon leurs occurrences. On en compte environ une trentaine. Toutefois après les filtres sur les URL et les noeuds (c'est-à-dire avec environ 2000 triplets restants après preprocessing) seulement 4 d'entre elles apparaissent. Lors de l'exécution de cette section du notebook une cellule va attendre des inputs utilisateur, en premier une entrée précisant le nombre de propriétés à inclure (habituellement seulement une) dans notre comparaison, puis en second les indices dans le tableau classé par nombre d'occurences de ces propriétés 0 étant la propriété ayant l'occurence la plus élevée et 3 la moins élevée. A noter qu'il est deconseillé d'utiliser trop de propriétés surtout pour les premières étant données le temps d'execution complètement prohibitif. Une fois la propriété choisie par l'utilisateur on sélectionne seulement les triplets avec cette propriété puis on effectue le produit cartésien entre les objet des triplets sources et des objets target. À noter aussi que la 4e est en fait inexploitable étant présente uniquement parmi des triplets du graphe source.

2.4 Évaluation

Pour l'évaluation on s'est aidés de la précision mesurant le nombre d'alignements corrects parmi ceux trouvés, le rappel mesurant le nombre d'alignements trouvés parmi ceux existants et du f1-score qui est une combinaison des 2 mesures précédentes.

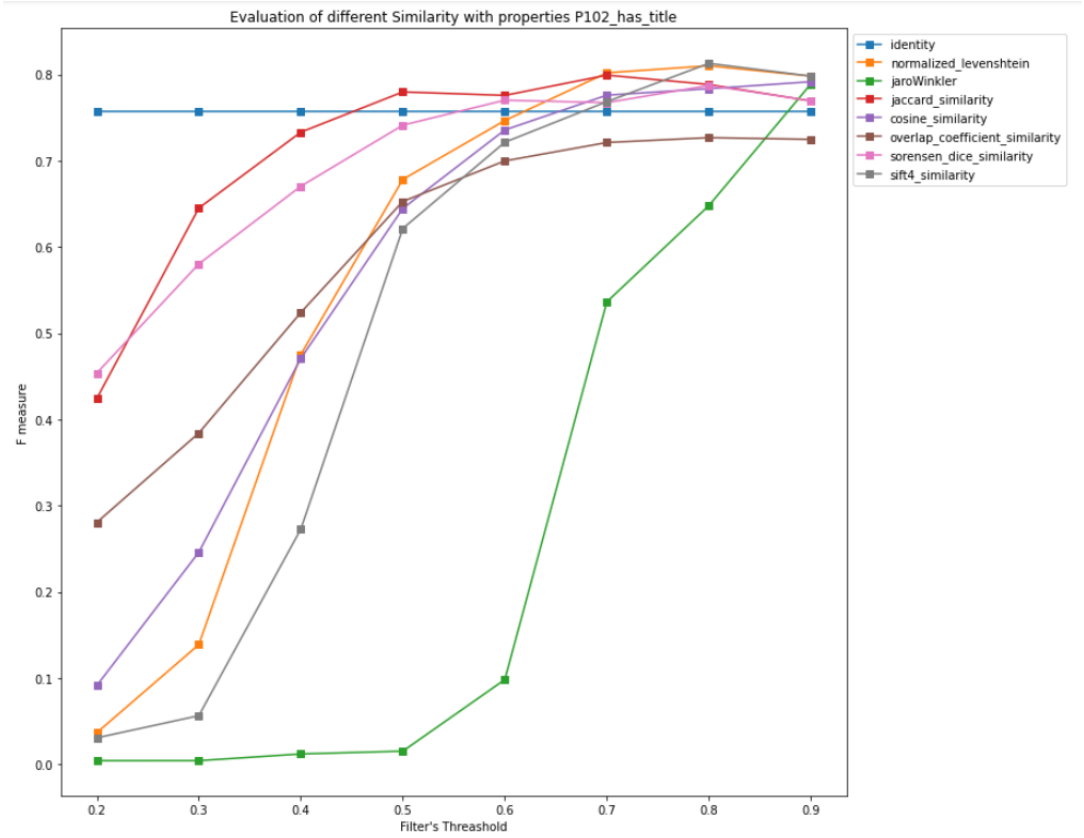
2.5 Résultats

Les résultats pour la première propriété P3_has_note ont été très décevants en plus des temps de calcul très longs il faut compter environ 20 min pour produire le graphique suivant.

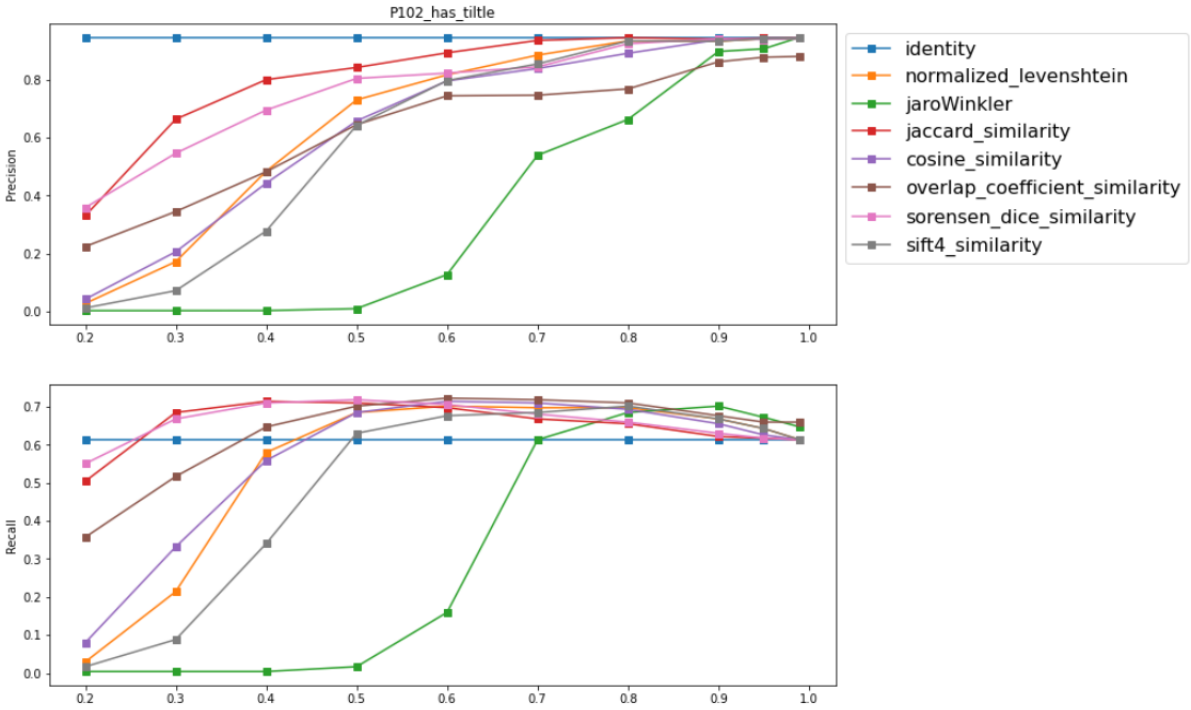


On remarque une augmentation de la précision au prix d'une baisse conséquente du rappel lorsque l'on augment la valeur de seuil. Les methodes de comparaison ont du mal à trouver de bon alignement parmi le nombre important trouvé mais en plus beaucoup d'alignements ne sont pas trouvés.

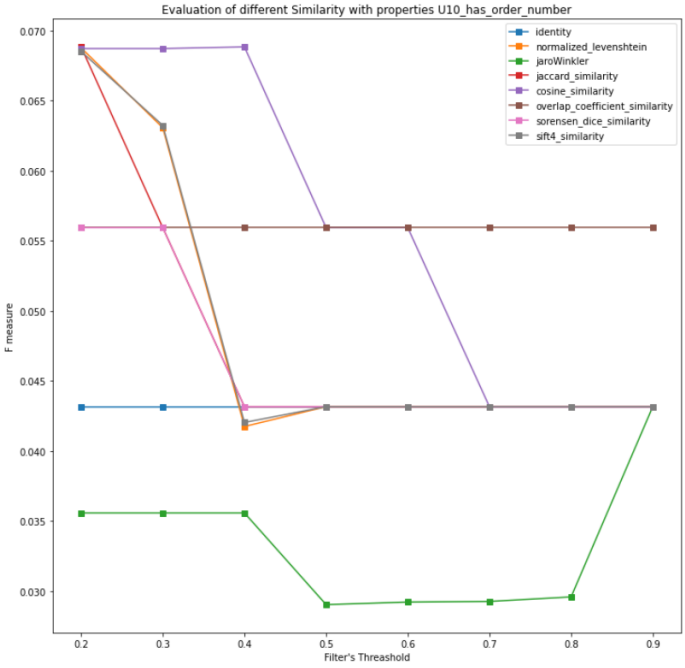
Les meilleurs résultats avec une nette différence par rapport aux autres ont été trouvés avec la propriété "P102_has_title", cf graphique ci dessous est représenté le f1score on voit que la majorité des propriétés donne de très bons résultats dès que le seuil est supérieur à 0.5. L'identité offre toujours le même résultat ce qui est normal puisque l'image de cette fonction est restreint à {0,1}. JaroWinkler performe très bien seulement pour des seuils assez élevés. On remarquera aussi que ces similarités performent le mieux autour de 0.8 et non sur des seuils très élevés ce qui est assez normal puisqu' un seuil trop élevé a tendance à enlever trop d'alignements et à faire baisser drastiquement le rappel



Comme on le voit sur le graphique ci dessous le rappel est impacté lorsque l'on augment le seuil au prix d'une précision pas forcément supérieur.



La troisième propriété offre des résultats assez mauvais ceci étant dû presque intégralement au fait que celle-ci est présente dans très peu de triplets.



3 Conclusion

3.1 Principales difficultés

Il a été difficile de prendre assez de recul sur le projet pour implémenter un système d'alignement vraiment intelligent et applicable à d'autres situations. Nous avons donc du nous resigner après beaucoup de difficultés à appliquer une approche gloutonne de comparaison sur les textes.

3.2 Perspectives d'amélioration

On pourrait imaginer une interface graphique encore plus intuitive pour l'utilisateur. Prendre en compte en plus des comparaisons par similarités les connaissances contextuelles d'un texte. Reflechir à d'autres prétraitements. Prendre en compte tous les noeuds associés à un élément donné et effectuer des comparaisons en parcourant le graphe (par exemple en considérant tous les noeuds à distance 1 puis à distance 2 etc)...

3.3 Bilan Appréciation

Nous avons apprécié travailler sur ce projet qui a été un complément au premier et nous a enseigné beaucoup sur le web sémantique. Le contexte était d'autant plus intéressant pour certains de nous puisqu'il portait sur le domaine de la musique.