Construction d'une base de données d'oeuvres culturelles en espéranto

DURY Alexandre

12 Mai 2022

Résumé

Ce projet a pour objectif le développement d'une base de données recensant un maximum d'œuvres en espéranto. Ces œuvres seront intégrées à partir de différentes sources de données, et nécessitent donc un processus de déduplication pour limiter les doublons. Un site web sera développé afin de rechercher et visualiser les œuvres culturelles espérantistes.

1. Introduction

Dans le cadre du second semestre du Master 1 Informatique, j'ai travaillé sur le sujet POM (Projet d'Orientation en Master) concernant la construction d'une base de données d'oeuvres culturelles en espéranto sous l'encadrement de Monsieur Fabien Duchateau, maître de conférences et membre de l'équipe Bases de Données au LIRIS[1] (Laboratoire d'InfoRmatique en Image et Système d'information).
L'espéranto est une langue internationale construite par le Docteur Zamenhof en 1887, parlée par plusieurs millions de personnes à travers le monde. Comme cette langue n'est rattachée à aucun État, l'une de ses critiques porte sur sa limitation au niveau culturel. Pourtant, de nombreuses oeuvres sont disponibles en espéranto, qui peuvent être soit des traductions d'oeuvres existantes (e.g., *la eta princo* pour le petit prince) soit des oeuvres originales (e.g., *Mr. Tot aĉetas mil okulojn*). À ce jour, il n'existe pas de liste la plus exhaustive possible rassemblant ces œuvres espérantistes. Les seuls sites disponibles proposant la consultation d'œuvres espérantistes sont généralement, en plus d'être incomplets, en langue espéranto (sans traduction), ce qui signifie que ce n'est pas facilement compréhensible par tout le

[1] https://liris.cnrs.fr/

monde. Des sites non dédiés à l'espéranto tels que Open library[2], Wikipédia ou Worldcat[3] permettent la visualisation de nombreuses œuvres, mais ils nécessitent que les internautes utilisent les fonctionnalités de recherche avancée (e.g., choix de la langue) pour trouver des résultats pertinents.

Pour remédier à ce problème, l'idée principale est de construire une base de données et un site qui rassemblent les œuvres espérantistes de plusieurs sources. Pour y parvenir, plusieurs étapes ont été effectuées tels que la modélisation d'une base de donnée, la création d'un jeu d'essai associé à la base de donnée créée, du web scraping pour récupérer les données qui nous intéressent sur les différentes sources, un processus de déduplication pour éviter les oeuvres en doublons et enfin le développement d'un site web.

Dans ce rapport, nous aborderons dans un premier temps le contexte de ce projet ainsi que ses problématiques. Dans un second temps, nous détaillerons les étapes réalisées pour atteindre l'objectif, mais aussi les outils utilisés dans le cadre de ce projet. Enfin, nous conclurons et terminerons sur les perspectives.

2. Problématiques

Lorsque l'on veut assembler des informations issues de différentes sources de données, c'est un problème d'**intégration de données**, qui est amplement étudié dans la littérature scientifique [1]. Le scénario qui nous concerne est celui d'une base de données centralisée, c'est à dire que nous avons un schéma global (pour cette base de données) et que nous la peuplons avec des données d'autres sources, qui ne respectent pas le même modèle. Par exemple, un auteur sera modélisé par un attribut "nom" et un attribut "prénom" dans le schéma global, alors qu'il peut être représenté par un seul attribut (regroupant le nom et prénom) dans une autre source. Il est donc nécessaire de trouver les concepts correspondants entre le schéma global et les sources de données utilisées. Dans notre contexte, la quantité d'informations reste limitée et ce travail peut être réalisé manuellement.

Une fois les correspondances établies, il faut **migrer les données** [2] de chaque source vers la base de données centralisée. Les modèles de données ne respectent pas les mêmes contraintes, et des transformations sont donc nécessaires. Par exemple, dans le cas du nom et du prénom, il faudra couper la chaîne de caractères pour séparer le nom du prénom et insérer ces deux données sous le bon attribut. De nombreux problèmes existent comme un type de données différent, des formats de données différents, etc. La définition de ces fonctions de transformations (mappings) est automatisable, mais c'est un problème complexe et nous avons choisi de développer ces mappings directement dans les scripts de migration.

Enfin, un dernier défi lié à l'intégration de données concerne la suppression des doublons, ou **déduplication [3]**. Différentes techniques permettent de détecter automatiquement des entités dupliquées, en comparant notamment leurs différentes propriétés. Un score de similarité est alors calculé, puis un mécanisme de décision (eg, un seuil) permet de décider si deux entités sont en doublon ou pas. La solution à apporter dans le cas d'un doublon détecté dépend du domaine d'application :

[2] https://openlibrary.org/
[3] https://www.worldcat.org/

parfois l'une des entités est sélectionnée pour être conservée et les autres sont supprimées, parfois les entités sont fusionnées en une seule afin de conserver un maximum d'informations. Pour traiter la déduplication, nous avons adapté les techniques existantes à notre contexte.

La section suivante détaille entre autres ces différentes étapes pour l'intégration d'œuvres culturelles liées à l'espéranto dans une seule base de données.

3. Travail réalisé

Le figure 1 fournit un aperçu global des différentes étapes du projet. La première étape est la modélisation de la base de données à partir des spécifications, qui permet de structurer correctement les données avec le minimum de redondance. En parallèle, le processus de scraping permet d'extraire des informations sur des œuvres, soit à partir de documents HTML soit à partir d'API. Ces deux processus permettent d'obtenir une base de données peuplée. Une étape de déduplication est exécutée pour supprimer les éventuels doublons (oeuvres ou agents) de la base. Enfin, la dernière partie consiste à développer une interface web pour visualiser les données de la base. Le reste de cette section détaille chacune de ces étapes.

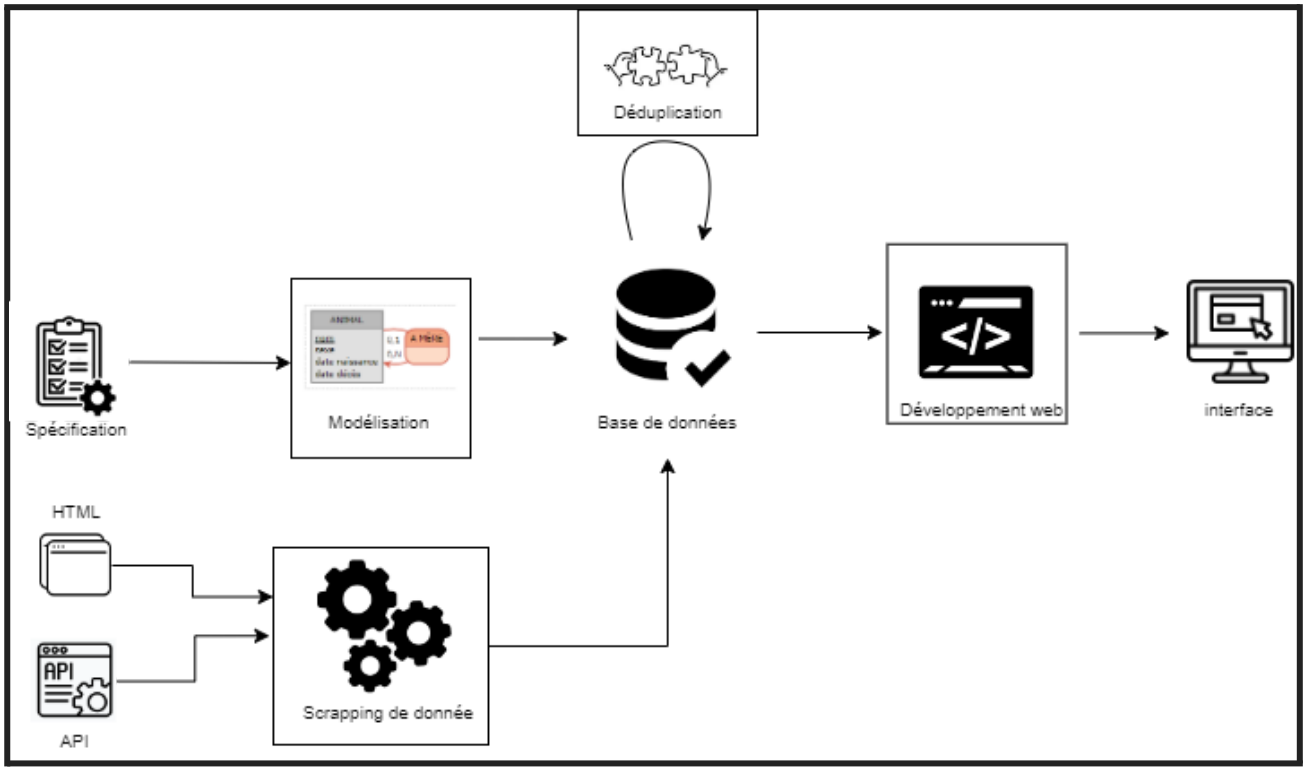


Figure 1 - Schéma représentatif du projet.

a. Modélisation de la base de donnée

Dans un premier temps, il a fallu réfléchir à un modèle conceptuel cohérent avec les données que l'on souhaitait enregistrer dans la base de données. Nous avons donc créé un diagramme entité-association (MCD, cf. Annexe 1) contenant peu d'entités. Ce diagramme se concentre autour des œuvres et des agents qui ont contribué à ces œuvres comme par exemple les auteurs, les réalisateurs, les compositeurs, etc., mais aussi des relations entre œuvres telles que des suites, des traductions ou encore des parodies. L'un des défis consistait à pouvoir intégrer de nouvelles sources de données sans avoir besoin de modifier le modèle. Pour répondre à ce problème, nous avons modélisé les propriétés de façon flexible en s'inspirant du modèle "entité-attribut-valeur"[4], c'est-à-dire qu'il est possible d'ajouter n'importe quelle propriété et d'affecter une valeur pour cette propriété à une œuvre ou à une relation.

Pour créer le diagramme, nous avons utilisé Mocodo[5], un outil pour la conception de données. Il permet à partir d'un format spécifique à Mocodo de générer le diagramme entité/association. Après validation par mon encadrant, Mocodo m'a permis de transformer ce diagramme en un schéma relationnel (MLD, cf. Annexe 2), en image SVG et en script SQL. Pour les attributs du script SQL, le type de chaque variable a été choisi de manière pertinente, avec une taille adaptée, afin de faciliter les requêtes et la compréhension par de futurs collaborateurs qui souhaiteraient améliorer ou ajouter des éléments à cette base de données. Enfin, le script SQL a été exécuté sur PHPMyAdmin en local, et un jeu d'essai a aussi été réalisé pour tester la base de données.

b. Scraping de données

Tout d'abord, le web scraping, c'est-à-dire l'extraction de données par le biais d'un script ou d'un programme, a pour objectif d'extraire des données à partir de documents (semi-)structurés et de les stocker de manière pérenne et mieux structurée. Le scraping de données permet de récupérer des éléments précis que nous souhaitons afin d'enrichir la base de données avec des œuvres espérantistes. Plusieurs sources de données peuvent être utilisées comme par exemple une page web en HTML, une API, ou encore des fichiers XML, JSON ou simplement textuelles. Ici le scraping a été effectué pour deux sources de données. Dans les deux cas nous avons pu utiliser l'outil Postman[6] afin d'enregistrer les différents liens qui ont servi à scraper ("attraper") les données.

Premièrement, nous avons interrogé l'API Open Library[7] à l'aide de la bibliothèque d'URL PHP (cURL[8]) afin de récupérer les données sous forme JSON, qui par la suite ont été traitées puis insérées dans la base de données. Idéalement, nous avons récupéré le titre, une description, l'URL, l'ISBN (Numéro d'identification attestant l'enregistrement international d'un livre), et une image de la couverture de celui-ci. Le nom de l'auteur, sa date de naissance, et de décès (si elle existe) sont aussi récupérés. De plus, certains champs ont nécessité des modifications avant d'être ajouté à la base de données, comme par exemple le lien vers la page de couverture d'une œuvre, on retrouve une partie qui reste inchangée pour chacune d'entre elles, il a donc fallu ajouter aux URL, l'identifiant spécifique à l'œuvre. On peut aussi noter l'utilisation de fonctions comme « addslash » qui selon un texte ajoute des barres obliques

[4] https://en.wikipedia.org/wiki/Entity%E2%80%93attribute%E2%80%93value_model
[5] http://mocodo.wingi.net/
[6] https://www.postman.com/
[7] https://openlibrary.org/developers/api
[8] https://www.php.net/manual/fr/book.curl.php

inversées (« \ ») avant chaque simple ou double guillemets ce qui permet d'éviter les erreurs d'intégration en base. Bien-sûr des informations peuvent être manquantes pour certaines œuvres. Grâce à Open Library, nous capturons les œuvres issues d'une recherche sur les œuvres contenant la requête *espéranto*, mais aussi sur les œuvres ayant une propriété langue valuée à *eo*. Au total, nous avons environ 1100 œuvres insérées dans la base de données pour Open Library.

Dans un second temps, le site Katalogo[9] a été utilisé pour recueillir d'autres œuvres listées par l'association internationale espérantiste. Ce site étant créé par la communauté, il regroupe un grand nombre d'œuvres variées, et il est donc logique d'intégrer ces données à notre base de données centralisée. Pour éviter d'interroger le site à la volée, un dump de leur base de données est mis à disposition. Cette ressource regroupant un ensemble d'œuvres par catégorie a été exploitée pour développer différents scripts d'extraction. En plus des attributs récupérés sur le site Open Library, la source Katalogo offre en plus des catégories associées aux œuvres (philosophie, histoire, politique, etc.). La librairie PHP DOMDocument[10] permet de décomposer la page HTML construite avec les différentes balises qui la composent pour ensuite scraper les données recherchées. Par exemple, pour accéder au chemin de l'image d'une œuvre on cherche les balises imbriquées td>div>a>img ce qui revient à faire une requête Xpath équivalente à "//td/div/a/img/@src". On récupère donc la valeur du champ source de la balise "img" qui correspond à l'URL de l'image de l'œuvre. Au total, cette source a permis l'ajout de près de 8200 œuvres dans notre base.

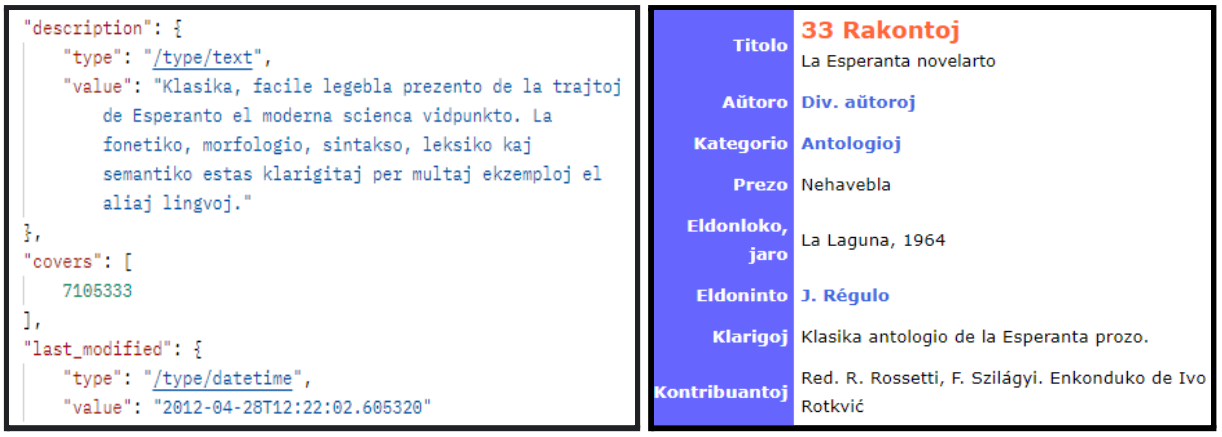


Figure 2 - Exemples de données extraites de Openlibrary en JSON (gauche) et de Katalogo en HTML (droite).

Au final, environ 9400 œuvres ont été intégrées à la base de données, avec en moyenne 5 attributs par source. La plupart des œuvres étant des sources bibliographiques, le contributeur principal correspond au rôle d'auteur renseigné dans la base. La figure 2 montre un extrait des données de Open Library et de Katalogo. L'œuvre issue de Open Library regroupe ici la description d'une œuvre espérantiste ainsi que l'identifiant correspondant à la couverture de l'image. Ces éléments ne sont pas toujours disponibles selon les œuvres il faut donc gérer cela, pour cela on peut spécifier dans la base de données que certains champs peuvent être vides. On remarque sur l'œuvre de droite (issue de Katalogo) que les

[9] https://katalogo.uea.org/
[10] https://www.php.net/manual/fr/class.domdocument.php

multiples auteurs ne sont pas mentionnés directement, et que Ivo Rotkvić a contribué en rédigeant l'introduction.
Néanmoins, le web scraping peut rapidement amener à des similitudes avec d'autres œuvres déjà existantes surtout si on récupère des données venant de sources différentes. C'est pourquoi il est nécessaire d'effectuer ce qu'on appelle un processus de déduplication afin de les détecter.

c. Déduplication

La déduplication est un processus qui élimine les copies excessives de données et réduit considérablement les besoins en capacité de stockage.
L'étape de déduplication peut s'effectuer de deux manières :

- soit au moment de l'insertion d'une entité. Dans ce cas, on vérifie au préalable si une entité de la base est équivalente avec celle que l'on veut insérer. L'avantage de cette approche est que l'on évite d'insérer des doublons, mais cela ralentit fortement l'intégration d'une nouvelle source, en particulier quand le volume de la base devient conséquent.
- soit après qu'une source ait été intégrée. Ici, on va parcourir les entités de la base et les comparer deux à deux pour détecter les éventuels doublons. Cette approche peut être relancée à tout moment.

Nous avons choisi la seconde approche, c'est-à-dire une déduplication indépendante de l'insertion.
Pour comparer deux entités, nous utilisons une fonction PHP levenshtein qui a pour but de calculer une distance d'édition, plus clairement de mesurer la différence entre deux chaînes de caractères (sensible à la casse). Elle est égale au nombre minimal de caractères qu'il faut supprimer, insérer ou remplacer pour passer d'une chaîne à l'autre, et le coût dépend des paramètres donnés à la fonction. Par exemple, on peut choisir comme coût 2 pour l'insertion et la suppression d'un caractère, mais seulement 1 pour le remplacement. Cette distance de Levenshtein [4] est appliquée sur le titre des œuvres mais pourrait être éventuellement appliquée à d'autres attributs d'une œuvre. Ici nous avons choisi uniquement les titres car contrairement aux titres les autres attributs peuvent être manquants. On effectue donc des tests pour savoir si une œuvre A correspond à une œuvre B.

Figure 3 - Exemple d'un doublon, avec une oeuvre de OpenLibrary (à gauche), et celle correspondante sur Katalogo (à droite).

Selon le pourcentage de différence entre A et B, on décide si elles sont équivalentes en paramétrant une valeur de seuil suite à différents tests. Lorsqu'un doublon est détecté, nous n'avons pas choisi de supprimer l'une des deux entités (risque de perte d'informations) ni de fusionner les deux entités (difficulté à décider si deux valeurs d'un même attribut doivent être conservées, ou si l'une est obsolète, etc.) : nous utilisons notre système de relations entre les œuvres pour indiquer que l'une est un doublon de l'autre, à l'aide d'une relation de la base de données nommée "doublon". Celle-ci regroupe l'ensemble des doublons qui ont été trouvés par l'algorithme. Pour effectuer cet algorithme, nous souhaitons donc calculer des valeurs pertinentes et précises afin de retrouver les œuvres très similaires présentes dans la base de données. Pour cela nous avons besoin de travailler sur un petit ensemble de tuples. Prenons par exemple les 500 premières œuvres de la base de données. Parmi ces éléments nous avons créé manuellement un fichier d'expertise d'environ 50 paires d'œuvres (Cf annexe 3). Ce fichier contiendra les doublons qui ont été trouvés parmi les 500 tuples. Maintenant nous allons considérer 4 cas différents (Cf annexe 4 et tableau ci-dessous) :

Nom du cas	**Abréviation**	**Description**
Vrai positif	VP	L'algorithme trouve **à raison** la paire d'oeuvres comme **appartenant** aux doublons
Faux positif	FP	L'algorithme trouve **à tort** la paire d'oeuvres comme **appartenant** aux doublons
Vrai négatif	VN	L'algorithme trouve **à raison** la paire d'oeuvres comme **n'appartenant pas** aux doublons
Faux négatif	FN	L'algorithme trouve **à tort** la paire d'oeuvres comme **n'appartenant pas** aux doublons

Grâce à l'algorithme on trouve au final le nombre total de chacun des cas cités. Les nombres issus de l'algorithme sont ensuite utilisés pour effectuer des calculs tel que le rappel, la précision et la F-mesure :

$$rappel = \frac{TP}{TP+FN} \qquad précision = \frac{TP}{TP+FP} \qquad Fmesure = \frac{précision * rappel}{précision+rappel}$$

La mesure précision permet de calculer le taux de correspondances découvertes qui sont corrects. À l'inverse, le rappel évalue la quantité de correspondances correctes parmi toutes celles attendues par l'expertise. Ces deux mesures sont généralement complétées par la f-mesure, un compromis entre précision et rappel. Ainsi, une fois les calculs automatisés, nous obtenons pour différentes configurations et seuils de différence par l'algorithme, les résultats suivants.

Tableau 1 - Résultats des algorithmes de déduplications.

	Algorithme 1	**Algorithme 2**	**Algorithme 3**	**Algorithme 4**
configuration (paramètres)	levenshtein($titre, $titre2, 1, 1, 1)	levenshtein($titre, $titre2, 1, 1, 1)	levenshtein($titre, $titre2, 3, 1, 1)	levenshtein($titre, $titre2, 3, 3, 3)
Pourcentage de différence entre les oeuvres (seuil)	< 20%	< 40%	< 40%	< 40%
précision	$\simeq 0.0069$	$\simeq 0.0010$	$\simeq 0.0018$	$\simeq 0.0103$
rappel	$\simeq 0.0833$	$\simeq 0.0770$	$\simeq 0.1$	$\simeq 0.125$
F-mesure	$\simeq 0.0064$	$\simeq 0.0010$	$\simeq 0.0017$	$\simeq 0.0095$

On a donc évalué et comparé les différents algorithmes avec des configurations et seuils différents. On retrouve donc des résultats peu cohérents. Pour avoir de meilleurs résultats il faudrait avoir un fichier d'expertise plus complet et une analyse plus fine sur différents champs d'une œuvre. De plus, on aurait pu effectuer d'autres tests avec encore d'autres seuils ou pourcentage de différence, mais ce serait encore mieux d'essayer avec une autre fonction de calcul de différence comme par exemple la fonction "similar_text". Finalement, nous insérons dans la table "doublon" uniquement les Vrai Positif (VP), c'est-à-dire les doublons.

Figure 4 - Exemple d'une paire d'œuvres similaires (Ici un FP) trouvé par l'algorithme Levenshtein.

Ainsi après avoir créé la base de données, intégré les données à celle-ci, et repérer les doublons, nous avons effectué le travail nécessaire qui va permettre de proposer un jeu de données complet et pertinent. Celui-ci va nous servir afin d'afficher le maximum d'attributs possible concernant une œuvre.

d. Développement d'un site web

L'objectif du développement d'un site web est tout d'abord de rendre visible les œuvres stockées dans la base de données, d'autant qu'il n'existe pas de liste la plus exhaustif possible rassemblant des œuvres espérantistes. Le site web a été développé en PHP version 8.1.1, ainsi qu'avec Bootstrap pour la mise en page et rendre le design responsif. Sur ce site nommé "EsperantOeuvres" on y retrouve différentes pages telles que :

- Une page d'accueil, dans laquelle on peut accéder à des œuvres proposées, et rechercher des œuvres espérantistes.
- Une page qui liste l'ensemble des œuvres, limités à une dizaine d'œuvres par page.
- Une page de description, dévoilant pour chaque œuvre le maximum d'informations récoltées, ainsi que le lien du site source.

Le site a été développé de manière à pouvoir gérer les traductions dans d'autres langues (anglais, français, espéranto) pour qu'à terme, les personnes non-francophones puissent aussi accéder à ces ressources. Ce site a donc pour but d'être épuré, simple d'utilisation, et accessible par tous.
Les annexes 5 et 6 donnent un aperçu des interfaces de notre site.

4. Conclusion

L'objectif initial de ce POM était la création d'une base de données contenant un grand nombre d'œuvres espérantistes, mais aussi la création d'un site permettant de consulter ces œuvres. Les points les plus complexes à réaliser dans ce projet ont été la récupération cohérente des données présentes sur les différentes sources traitées, à savoir Open Library et Katalogo. On peut noter l'importance de la déduplication, avec une estimation de la similarité de deux entités, qui permet de nettoyer les éventuels doublons qui se sont immiscés dans les données suite à l'intégration des deux sources. Enfin la création du site web permet de se rendre compte du travail effectué, qui est plutôt satisfaisant selon moi, et qui pourrait être facilement améliorable par la suite.

Ce projet a plusieurs perspectives. La première serait d'y intégrer de nouvelles sources de données afin d'avoir de plus amples informations sur les œuvres déjà existantes mais aussi de pouvoir enrichir le site d'œuvres espérantistes récentes. De plus, il serait intéressant de développer d'autres scripts pour la détection de doublons, qui seraient passés entre les mailles du filet, car la base de données commence à être relativement conséquente et donc la détection de plus en plus délicate. Actuellement la déduplication n'est pas optimisée (produit cartésien entre les entités, donc 100 millions de comparaisons pour les œuvres). Il sera nécessaire par la suite d'utiliser un système de blocking pour limiter le nombre de comparaisons.
L'ajout de fonctionnalités au site, est une perspective intéressante afin de s'y retrouver parmi ce grand nombre d'oeuvres, de nouvelles catégories plus précises pourraient être associées, la visualisation d'oeuvres par auteur, ou encore la possibilité d'ajouter de nouvelles sources directement sur le site, qui pourrait être enrichi directement par la communauté. Mais pour cela il faudrait l'ajout d'un

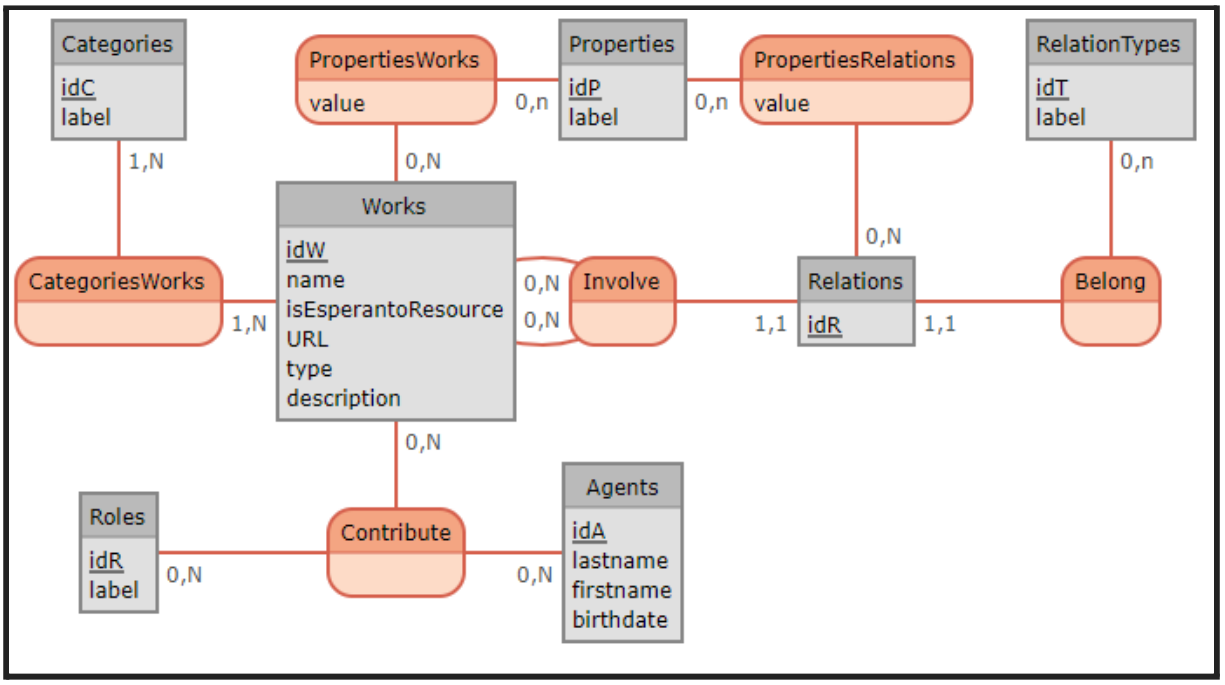
administrateur qui puisse valider les propositions d'ajout afin de ne pas ajouter des éléments qui n'ont aucun rapport avec la langue espéranto et le peu de culture qui lui est associée.

Pour ma part, ce projet fut un réel plaisir à réaliser. J'ai pu perfectionner mes connaissances de développement avec à la fois l'apprentissage de nouvelles fonctionnalités que propose PHP notamment sur toute la partie récupération de données sur des sources externes, mais aussi sur les fonctions de similarités (levenshtein,similar_text). Cela m'a également permis de me rendre compte que la manipulation de données n'est pas évidente et demande de la concentration car on peut très vite se retrouver perdu par toutes les données que l'on manipule.
Cette expérience a favorisé une certaine agilité d'esprit, un goût du challenge, qui confirme mon choix d'orientation en Master 2 Technologies de l'Information et Web (TIW).

Références

[1] Doan, A., Halevy, A., & Ives, Z. (2012). *Principles of data integration*. Elsevier.
[2] Golshan, B., Halevy, A., Mihaila, G., & Tan, W. C. (2017, May). Data integration: After the teenage years. In *Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI symposium on principles of database systems* (pp. 101-106).
[3] Christen, P. (2012). The data matching process. In *Data matching* (pp. 23-35). Springer, Berlin, Heidelberg.
[4] Levenshtein, V. I. (1966, February). Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady* (Vol. 10, No. 8, pp. 707-710).

Annexe 1 : Modèle conceptuel de données



Annexe 2 : Modèle Logique de Données (schéma relationnel)

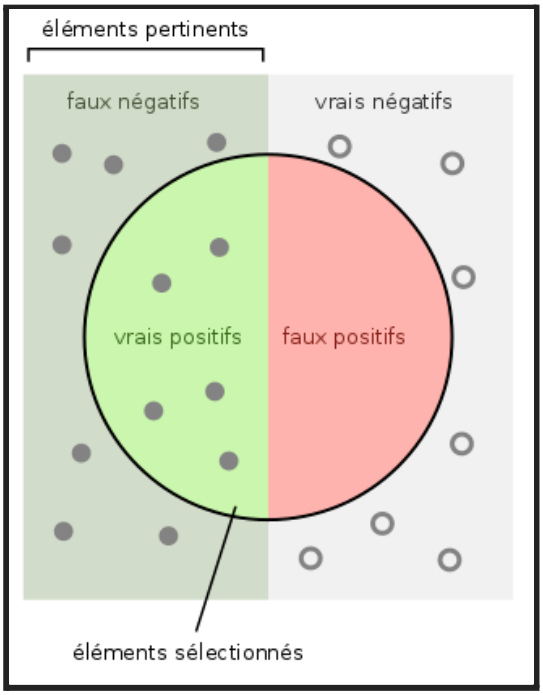
CATEGORIES (idC, label)
PROPERTIESWORKS (*idP*, *idW*, value)
PROPERTIES (idP, label)
PROPERTIESRELATIONS (*idP*, *idR*, value)
RELATIONTYPES (idT, label)
CATEGORIESWORKS (*idW*, *idC*)
WORKS (idW, name, isEsperantoResource, URL, type, description)
RELATIONS (idR, *idT*, *idW*, *idW.1*)
ROLES (idR, label)
CONTRIBUTE (*idW*, *idA*, *idR*)
AGENTS (idA, lastname, firstname, birthdate)

Annexe 3 : Extrait du fichier d'expertise regroupant les doublons.

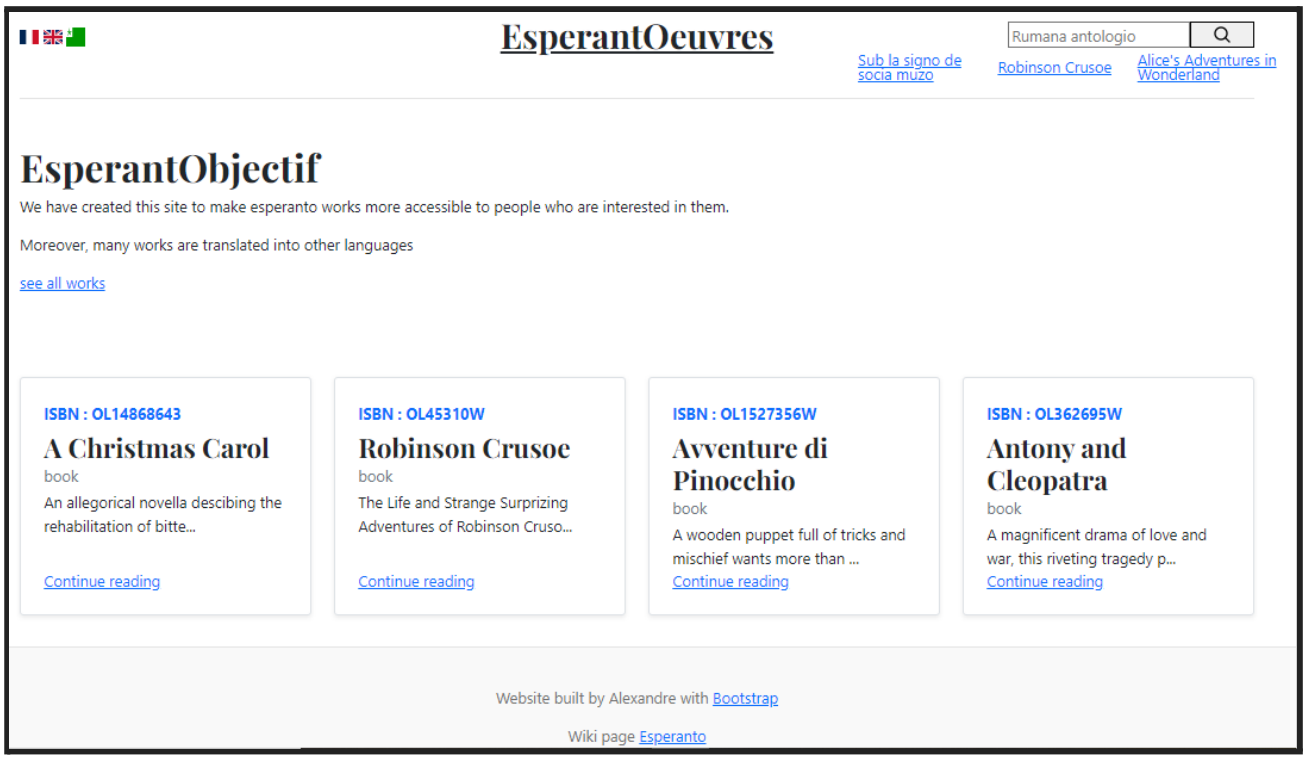
similarity.txt

```
1   6,4
2   705,4
3   703,6
4   237,17
5   297,17
6   690,27
7   163,141
8   263,145
9   163,147
10  613,191
11  263,232
12  441,263
13  270,264
```

Annexe 4 : schéma explicatif des différents cas présent dans un ensemble d'oeuvres



Annexe 5 : Capture d'écran de la page d'accueil du site



Annexe 6: Exemple d'une page de description d'une oeuvre

Work

Nom : Avventure di Pinocchio

Description : A wooden puppet full of tricks and mischief wants more than anything else to become a real boy.

Lien externe : voir la source

Type : book

Author

Nom de l'auteur : Hendrik Jan Bulthuis

Date de naissance : 1865

Date du décès :