PARIS SACLAY
Communauté d'agglomération

Intégration d'informations de fiabilité à l'application Omnicrobe

Elora VIGO - BIBS Paris-Saclay

Mai 2023

Sous la direction de :

Sandra DEROZIER et Louise DELEGER

Institution de stage :

Unité MaIAGE d'INRAE

Mathématiques et Informatique Appliquées du Génome à l'Environement

INRAE
Service de Fédération

Table des matières

1 Introduction

L'Institut National de Recherche pour l'Agriculture, l'alimentation et l'Environnement (**INRAE**) réalise de la recherche dans les domaines comme l'agriculture, l'alimentation, l'environnement et la biodiversité. L'unité de recherche **MaIAGE** est rattachée aux départements de Mathématiques et Numérique et MIcrobiologie et Chaîne Alimentaire de l'INRAE. La thématique de MaIAGE est le développement de méthodes mathématiques et informatiques pour l'étude de la biologie et l'agro-écologie. L'unité est composée de mathématiciens, informaticiens, bioinformaticiens et de biologistes. Elle est composée de quatre équipes : **Bibliome** (acquisition et formalisation de connaissances à partir de textes), **BioSys** (biologie des systèmes), **Dynenvie** (modélisation dynamique et statistique pour les écosystèmes, l'épidémiologie et l'agronomie), **StatInfOmics** (bioinformatique et statistique des données omiques) ; et de la plateforme de bioinformatique **Migale**. Ce stage s'inscrit dans une collaboration entre les équipes StatInfOmics et Bibliome qui a notamment abouti au développement de l'application Omnicrobe.

De plus en plus de données sur les micro-organismes sont disponibles dans différentes sources. Cependant, chercher des informations spécifiques sur un micro-organisme peut prendre du temps, car il faut effectuer des recherches dans différentes bases de données et ressources bibliographiques. La base de données Omnicrobe [1] a été développée pour centraliser les informations issues de six sources différentes. Ces sources sont **PubMed** (résumés de littérature scientifique [2]), **GenBank** (base de données de séquences [3]) et des catalogues de ressources biologiques : **BacDive** (base de données sur la diversité bactérienne [4]) et trois catalogues **CIRM** (Centre international de ressources microbiennes [5, 6, 7]) gérés par INRAE. L'utilisateur peut ainsi interroger plusieurs sources via une même interface et répondre à des questions telles que "Où vit un micro-organisme donné ?". Pour cela, les données des sources sont traitées grâce à un workflow de fouille de texte. Celui-ci permet d'extraire les entités d'intérêt : **taxon** (les micro-organismes), **habitat**, **phénotype** et **usage**, mais également les relations entre ces entités. On nomme 'élément' les entités qui ne sont pas des taxons. L'extraction automatique des relations taxon-élément est composée de trois parties : la reconnaissance d'entité, leur standardisation et l'extraction de relation.

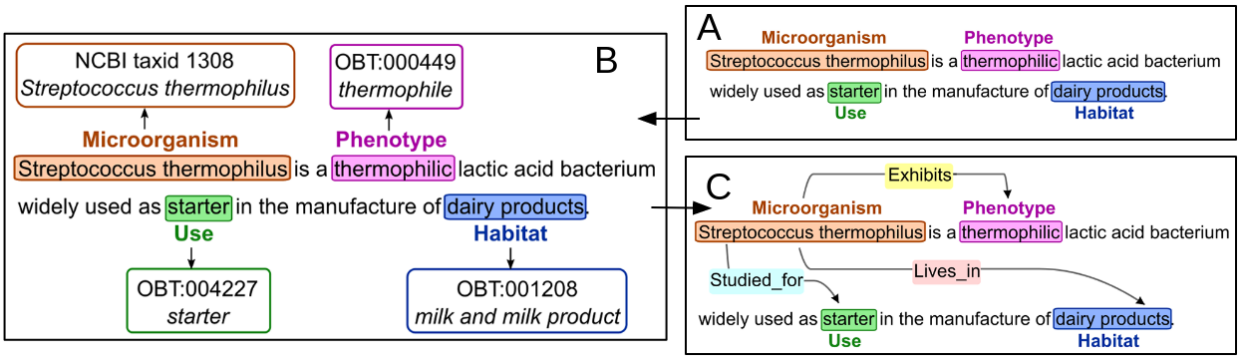


FIGURE 1 – Le pipeline de fouille de texte [1]

La reconnaissance d'une entité dans un document se fait grâce à des listes de vocabulaire, ou référentiel (cf. Figure 1.A). Une fois les entités reconnues, on peut leur rattacher l'identifiant associé dans ce référentiel, c'est l'étape de standardisation (cf. Figure 1.B). Ces référentiels sont la taxonomie du **NCBI** pour les taxons et l'ontologie **Ontobiotope** (crée par l'équipe Bibliome, cf. [8]) pour les éléments (`https://agroportal.lirmm.fr/ontologies/ONTOBIOTOPE`). Dans la figure 1, le mot "thermophilic" est similaire à "thermophile" du référentiel Ontobiotope, ce mot est donc reconnu comme entité et est standardisé par le nom "thermophile" et l'identifiant "OBT :000449". Enfin, le pipeline de fouille de texte prédit la présence

d'une relation entre le taxon et les autres éléments détectés (cf. Figure 1.C). Il détecte par exemple que le micro-organisme et le phénotype "thermophile" sont liés.

On nomme "occurrence" l'ensemble formé par une relation et ces deux entités. L'ensemble des occurrences liant les mêmes entités est nommé "relation agrégée". Les occurrences sont donc les mentions, dans les documents, de la relation. La figure 2 permet de mieux comprendre le vocabulaire utilisé.

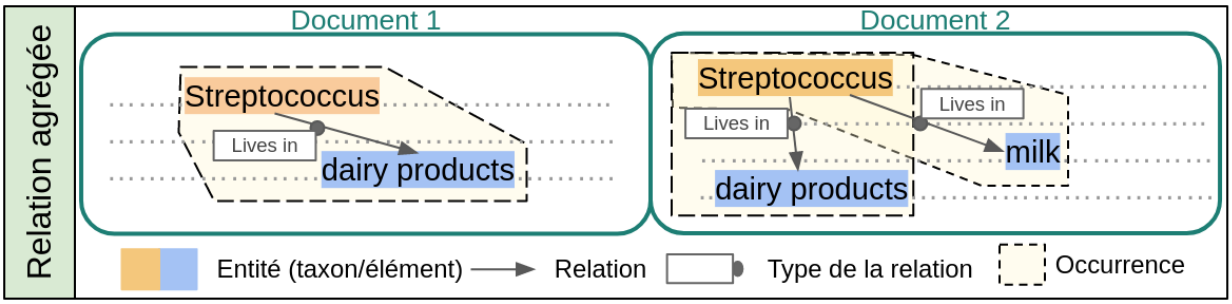


FIGURE 2 – Vocabulaire d'une relation

Les données en sortie de la fouille de texte sont stockées dans une base de données relationnelle et interrogeable via une interface web (cf. figure 3) et une interface programmatique (API). La figure 3 montre une recherche sur "acacia tree" pour répondre à "Quels micro-organismes peuvent être trouvés dans un habitat donné ?".

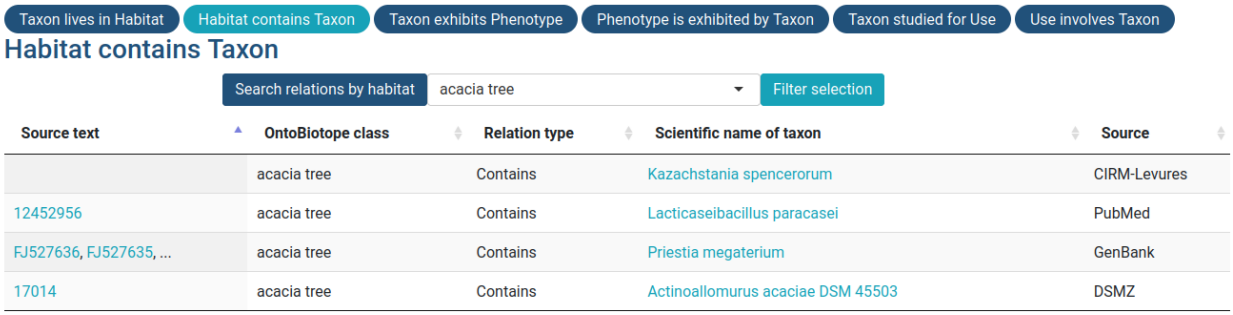


FIGURE 3 – L'interface d'Omnicrobe : résultat simplifié de la recherche "acacia tree"

Les résultats sont présentés dans un tableau exportable en format tabulé (cf. figure 3). Une ligne représente une relation agrégée et comprend le nom de l'élément de l'ontologie Ontobiotope (ici "acacia tree"), le taxon ("Kazachstania spencerorum"), le type de la relation ("contains") et la source ("CIRM-Levures"). De plus, on a l'identifiant des documents dans la source dans la première colonne (sauf pour la source CIRM), ce qui permet de revenir au texte d'origine. Ainsi, rechercher une information sur un micro-organisme devient plus rapide et facile. Le projet Omnicrobe est en conformité avec le principe FAIR (Findable, Accessible, Interoperable, Reusable), favorisant ainsi leur découverte, leur accessibilité, leur interopérabilité et leur réutilisation.

Le pipeline de fouille de texte est entièrement automatique et ne comporte pas d'étape de vérification manuelle. Les relations extraites par le pipeline de fouille de texte peuvent donc comporter des erreurs. Cependant, le pipeline fournit plusieurs scores qui indiquent la fiabilité des résultats obtenus. Le but de ce stage consiste à ajouter cette notion de qualité des données à l'interface web existante afin de guider l'utilisateur dans l'interprétation des résultats proposés. Pour cela, il faut récupérer les scores du pipeline de fouille de texte, les mettre dans la base de données et modifier l'interface web.

2 Matériels et Méthodes

Cette section présente le cadre de travail dans lequel j'ai effectué mon stage. Elle présente les méthodes et le matériel qui étaient déjà en place à mon arrivée. Tous les codes sont disponibles sur deux dépôts gitlab : "Omnicrobe database" et "Omnicrobe web".

2.1 Données générées par la fouille de texte

Le workflow de fouille de texte génère des fichiers au format texte contenant l'ensemble des informations détectées. Avant le début de mon stage, les fichiers étaient au format tabulé. Afin de faciliter les évolutions de l'application, mes encadrantes m'ont fourni ces informations au format JSON [9]. Le fichier contient les entités "brutes" détectées dans le texte ainsi que les entités standardisées par le référentiel utilisé. On y trouve également le type de relation entre ses entités et un identifiant permettant de revenir au document source à partir duquel a été extrait l'ensemble des informations. Les scores de fiabilité des données générés par le pipeline sont également présents dans ce fichier.

Le pipeline fournit des scores de fiabilité à différents niveaux :

— **Score d'une entité** : Il indique si la reconnaissance de l'entité par la fouille de texte est fiable. Une entité reconnue dans un document source a une certaine forme, celle-ci peut différer de la forme standardisée présente dans le référentiel. Par exemple, une forme de texte peut-être '*dairy product*', mais ce mot sera associé à '*milk and milk product*' dans le référentiel (exemple en Figure 1). Le score d'une entité reconnue est le score de similarité entre ces deux termes, calculé avec la distance de Jaccard.

— **Score d'une relation** : Il indique si la relation entre deux éléments est plus ou moins fiable (étape C de la figure 1). Son calcul n'est pas encore implémenté, une valeur empirique est utilisée actuellement.

— **Score d'une occurrence** : Il indique si la mention d'une relation dans un document est fiable. Il prend en compte les scores individuels des entités et de la relation. Il est calculé en faisant la moyenne de ces trois valeurs.

— **Score de document** : Il indique si les occurrences de la relation trouvées dans le document sont en moyenne fiables. C'est donc une moyenne des scores des occurrences du document.

— **Score global** : Il représente la fiabilité de la relation dans son ensemble. Il est calculé en faisant une moyenne des scores des documents de la relation agrégée.

2.2 Le pipeline de développement d'Omnicrobe

2.2.1 Les étapes du pipeline général

Les données relatives aux micro-organismes suivent un processus en plusieurs étapes, correspondants à différents pipelines (cf. figure 4). La première étape est la récupération des documents (comme les résumés PubMed). Les informations d'intérêt sont alors extraites via le pipeline de fouille de texte présenté en introduction, et exportées dans des fichiers JSON (cf. 2.1). Un pipeline intègre ensuite ces informations dans une base de données relationnelle (cf. 2.3). L'interface web et l'API permettent alors de faire des requêtes sur cette base de données (cf. 2.4).

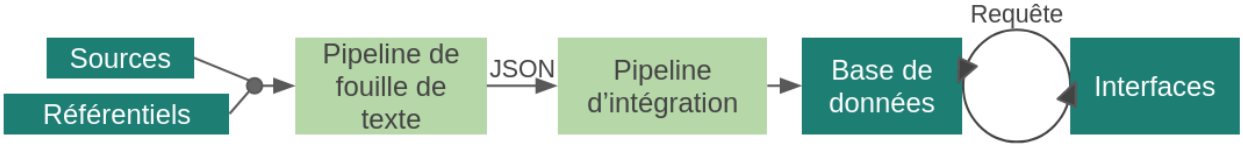


FIGURE 4 – Schéma simplifié du pipeline général

2.2.2 Langages des pipelines

Les différents pipelines ont été réalisés avec **Snakemake** (*version* $\geq$ *6.15*) (cf. [10]). Snakemake est un outil de gestion de pipeline utilisé en bioinformatique pour sa reproductibilité, sa gestion automatique des dépendances et sa reprise sur erreur. Un Snakefile est un fichier définissant les différentes règles à exécuter. C'est le nom des fichiers en entrée et sortie de ces différentes règles qui permettent de faire le lien entre elles. Le Snakefile est exécuté dans un environnement virtuel Python (librairie **venv** [11]), où sont installées toutes les librairies nécessaires. On a par exemple la librairie **psycopg2** (*version* $\geq$ *2.8.6*) de **Python3** (*version* $\geq$ *3.10*), elle permet de se connecter, de modifier et de lire une base de données. De plus, pour exécuter les pipelines de fouille de texte et d'intégration, on utilise le cluster de calcul Migale de MaIAGE.

2.3 La base de données

Les données sont stockées dans une base de données relationnelle **postgreSQL**. Les trois tables principales permettant de stocker les relations détectées sont celles définies dans la figure 5. La table "**taxon**" permet de stoker les micro-organismes et la table "**élément**" les habitats, les phénotypes et les usages. La table de jointure "**relation**" stocke les identifiants des deux entités, le type de relation, mais également l'identifiant du document source.

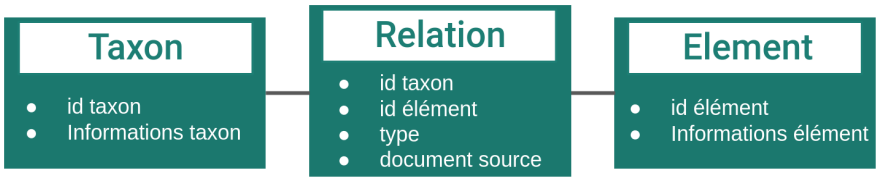


FIGURE 5 – Schéma simplifié de la base de données d'Omnicrobe

2.4 Interfaces

Les données sont accessibles via une interface web et une interface programmatique permettant de télécharger des données en masse. Le côté serveur de l'application est géré en Python via la librairie **Flask** (*version* $\geq$ *1.1.2*). Flask est une classe qui permet de créer une instance d'application via un script principal. Les requêtes sont faites dans un autre script Python, il fait appel à la librairie pyscopg2 permettant la communication avec la base de données. Le résultat d'une requête est au format JSON. Côté client, l'application est développée en **Javascript** et utilise des templates **HTML** ainsi que des fichiers de style **CSS**. Les fichiers HTML définissent l'architecture et le texte d'une page web. Les fichiers CSS définissent les styles graphiques des différentes parties de la page. Enfin, l'utilisation de JavaScript rend dynamique le site web. Le passage entre le côté client et serveur se fait grâce au script principal de Flask. Ce script associe des chemins (URL) à des fonctions (cf. [12]). Un chemin peut mener à une page HTML via une fonction de Flask, ou mener à une structure JSON via une fonction du script de requête. Seuls les chemins menant aux HTML sont visités par les utilisateurs sur le site. Les scripts Javascript modifient les pages HTML pour qu'elles affichent les résultats présents dans les structures JSON.

L'interface web d'Omnicrobe permet à l'utilisateur de faire différents types de recherche. En effet, il peut interroger la base de données en saisissant un terme d'intérêt, que ce soit le nom d'un microorganisme, un habitat, un phénotype ou un usage via des champs de complétion. Il peut également naviguer au sein de l'ontologie OntoBiotope représentée sous forme d'arborescence et cliquer sur un terme d'intérêt. Une fonctionnalité de recherche avancée permet aussi d'effectuer des requêtes plus complexes.

3 Résultats

Afin d'enrichir l'application Omnicrobe avec les informations de fiabilité, j'ai dû modifier la structure de la base de données, parser les fichiers JSON fournis, intégrer leur contenu dans la base et modifier l'interface web.

3.1 Base de données

3.1.1 Architecture de la base de données

J'ai enrichi le schéma relationnel de la base de données en ajoutant des tables et des nouveaux champs aux tables existantes (cf. figure 5). Le nouveau schéma est présenté en figure 6. Ces modifications ne sont pas toutes liées au score, certaines sont faites pour prendre en compte des évolutions prochaines de l'application.

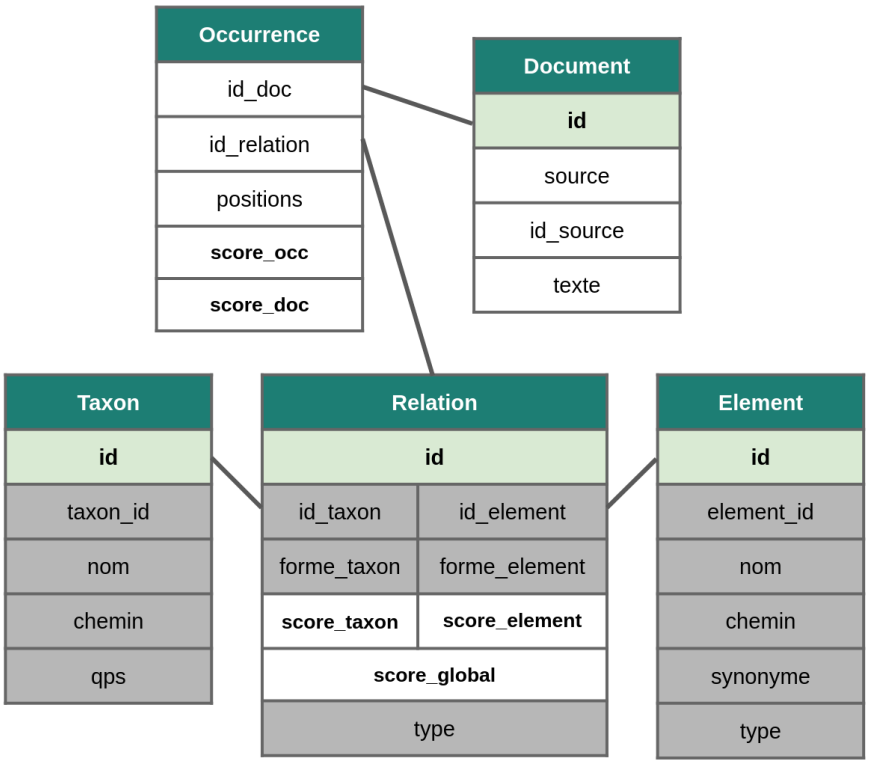


FIGURE 6 – Schéma de la base de données

Les clés primaires des tables sont en vert clair et en gras. Les cases grises sont celles non modifiées, les blanches sont les ajouts ou les champs ayant changé de table.

Les cases grises de la figure 6 sont les attributs inchangés. Un **taxon** ou un **élément** contient des informations sur son identifiant ('taxon_id' ou 'element_id'), son nom et son chemin dans le référentiel avec lequel il est standardisé. Les référentiels sont structurés en arbres, le chemin est ce qui permet de retrouver un concept donné depuis la racine de celui-ci. De plus, un taxon a une information de présomption d'innocuité reconnue (QPS) indiquant si le micro-organisme est autorisé à être utilisé dans l'alimentation. Un élément a une information sur ses synonymes et son type d'élément (habitat, phénotype ou usage). Les cases 'id' en vert sont les identifiants internes à la base de données, à différencier des identifiants dans les référentiels (comme "OBT :000449" dans l'ontologie Ontobiotope).

Une **relation** lie un taxon et un élément, les identifiants dans la base de données des deux entités sont indiqués dans cette table. Les formes des entités trouvées dans les documents sources sont stockées dans 'forme_taxon' et 'forme_element', leurs scores respectifs sont ajoutés ('score_taxon' et 'score_element'). Le score global d'une relation agrégée est placé dans cette table. Le type d'une relation est le type de l'élément de la relation. Les informations concernant les documents sources sont stockées dans une nouvelle table "**document**", et non plus dans la table "relation". La table "document" contient le nom de la source (GenBank par exemple) et l'identifiant du document dans cette source. Les résumés PubMed sont placés dans l'attribut 'texte'. On peut trouver plusieurs relations dans un document, et une même relation peut être trouvée dans plusieurs documents différents (différents résumés PubMed par exemple). C'est la nouvelle table "**occurrence**" qui conserve cette liaison entre la relation et les documents sources dans laquelle elle a été trouvée. De plus, pour les sources PubMed, cette table conserve les informations de position des entités reconnues dans le résumé. Enfin, les scores d'occurrences et de document dépendent à la fois de la relation et du document, ils sont placés dans la table occurrence.

3.1.2 Remplissage de la base de données

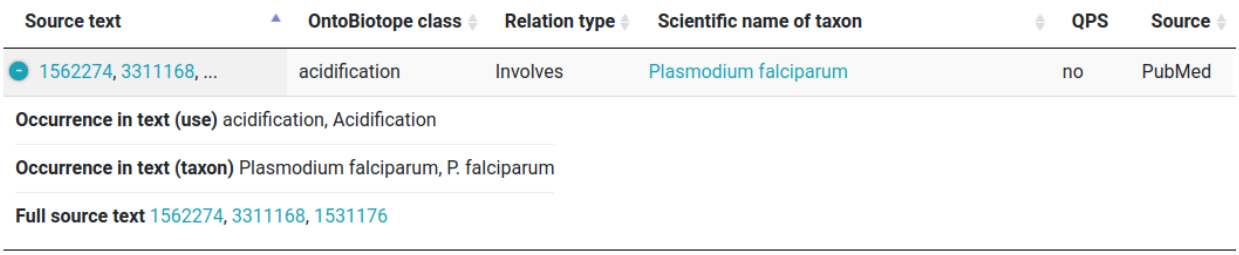
Afin d'intégrer les données dans la base de données, j'ai effectué des modifications dans plusieurs étapes du pipeline d'intégration (cf. 2.2.1). Cela est dû à la modification de l'architecture de la base de données et au changement du format du fichier généré par la fouille de texte. J'ai donc créé un script Python qui parse le fichier JSON et insère les informations dans la nouvelle base de données (cf. figure 6). J'ai de plus ajouté un script d'intégration des résumés PubMed dans la base de données. Enfin, j'ai effectué des modifications mineures concernant les changements de noms d'attributs et de tables dans les requêtes SQL des scripts préexistants mon arrivée.

Afin de tester ces modifications, j'ai travaillé sur une base de données en local. Je l'ai instanciée avec un sous-ensemble des données disponibles correspondant à 525 016 relations, 182 139 taxons et 4 237 éléments.

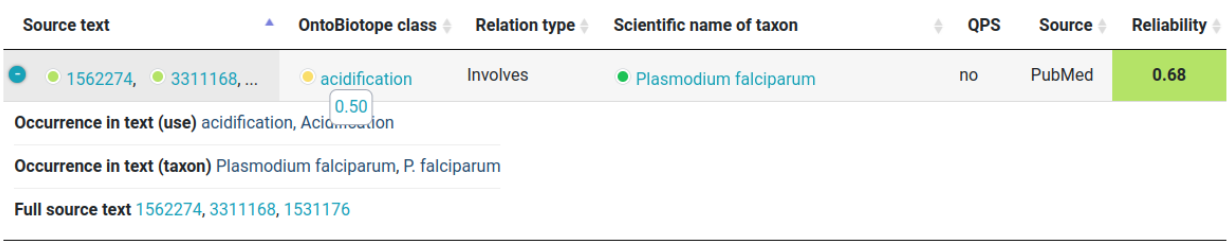
3.2 Interface web

J'ai effectué des modifications sur l'interface web pour visualiser les différents scores et filtrer les informations en fonction de leur fiabilité.

Les données récupérées à partir de la base de données sont présentées sous forme de tableau sur le site web (cf. Figure 3). Chaque ligne représente une relation agrégrée, et différentes informations sur celle-ci sont disponibles. Lorsque toutes les informations ne tiennent pas sur une ligne de l'écran, on peut déplier la ligne pour afficher celles manquantes. Une ligne est composée du type de relation ('Relation type') entre un élément spécifique ('Ontobiotope class') et un taxon ('Scientific name of taxon'). Les deux premiers documents sources sont accessibles via des liens dans la colonne 'Source text', et l'ensemble des documents dans la colonne 'Full source text'. On a de plus les informations sur le statut QPS du taxon, la source des documents et les formes des entités telles qu'elles apparaissent dans les textes ('Occurrence in text').



(a) Interface non modifiée



(b) Interface modifiée

FIGURE 7 – Une ligne du tableau des résultats dans l'interface web

Dans la nouvelle version, j'ai ajouté une colonne 'Reliability' (fiabilité) (cf. Figure 7b). Elle contient le score global de la relation agrégée, qui indique si de manière générale la relation est fiable. Le fond de la case est colorié en fonction de ce score pour le faire ressortir. La couleur varie du rouge (fiabilité de 0) au vert (fiabilité de 1), le jaune est donc le score médian (0.5). Des ronds de couleurs ont été ajoutés à côté du texte dans les colonnes 'Source text', 'Ontobiotope class' et 'Scientific name of taxon' (cf. Figure 7b). Ils permettent d'indiquer respectivement le score de fiabilité des premiers documents et des deux entités via le même code couleur que pour le score global. Une entité a plusieurs formes de textes, et donc plusieurs scores associés, c'est le meilleur qui est visible dans ces cases. Un passage de souris sur les mots des colonnes 'Source text', 'Ontobiotope class', 'Scientific name of taxon', 'Occurrence in text (use)', 'Occurrence in text (taxon)' et 'Full source text' change la couleur du texte et affiche les scores associés. Sur la figure 7b, la souris est placée sur le mot "acidification" de la colonne 'Ontobiotope class'. Ce mot passe alors d'un bleu marine à un bleu clair et la bulle affichant le score apparait. Dans le cas où la souris passe sur les mots de la colonne 'Scientific name of taxon', 'Source text' et 'Full source text', le texte passe cette fois du bleu clair au bleu marine. Cette différence est due au fait que, contrairement aux autres colonnes, ces trois dernières contiennent des liens vers les documents sources et la taxonomie du NCBI. Cela permet de différencier les termes avec des liens. Dans le cas où plusieurs formes de textes et documents sont présents, ceux-ci sont triés de sorte à montrer en premier ceux avec un meilleur score.

Le tri par ordre croissant ou décroissant du score de la colonne 'Reliability' est possible. De plus, j'ai ajouté une option de filtre à celles préexistantes : on peut ne récupérer que les relations avec un score supérieur à une certaine valeur saisie par l'utilisateur (cf. Figure 8).

FIGURE 8 – Les filtres sur les relations

De plus, j'ai modifié le code afin que l'URL de la page se mette à jour en fonction de la valeur saisie dans le filtre du score global (case 'Reliability'). Cette mise-à-jour de l'URL était déjà réalisée pour les entités saisi en barre de recherche et dans le filtre, ainsi que pour le statut QPS.

Parmi les différents scores proposés, les scores de relation et d'occurrence ne sont pas visualisés dans le tableau de résultat. Il n'est pas nécessaire d'afficher le score de relation sur le site puisque le score d'occurrence est plus représentatif de la relation entre les deux entités. En effet, il prend en compte à la fois la relation et ses entités. Pour que l'utilisateur ait accès aux scores d'occurrences, j'ai implémenté une nouvelle visualisation des résumés PubMed. Auparavant, un lien au niveau des identifiants des documents pointait vers l'outil AlvisIR, un outil de visualisation de résumé externe à l'application Omnicrobe. Cette nouvelle visualisation permet de ne pas dépendre d'outils extérieurs et d'ajouter les informations de fiabilités sur cette visualisation. Les entités reconnues sont surlignées et liées par une flèche symbolisant la relation (cf. figure 9). Cela a été réalisé avec la librairie Recogito.js (cf. [13]). Recogito est une librairie d'annotation qui permet de mettre des commentaires sur des mots, de les relier et d'ajouter des étiquettes sur les mots et les liaisons. Chaque type d'entité est associé à une couleur, dont l'opacité dépend du score de l'entité. Ainsi, les entités les plus sûres ressortent. Un passage de souris sur les entités affiche leur score. La couleur de l'étiquette de la flèche reliant les entités varie du rouge au vert, et représente le score de fiabilité de l'occurrence.

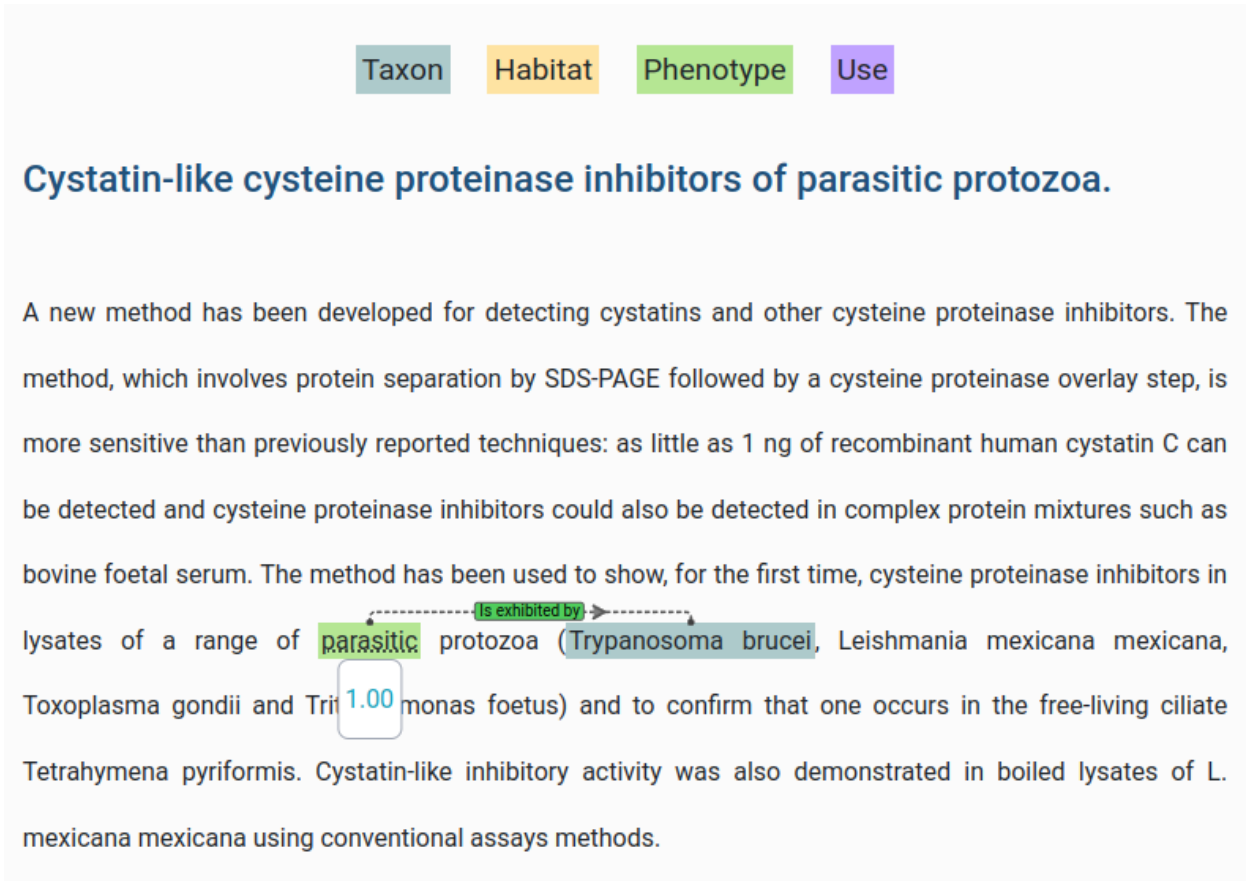


FIGURE 9 – Abstracts PubMed

4 Discussion

Durant mon stage, j'ai modifié l'application Omnicrobe afin d'ajouter la notion de fiabilité dans les résultats. Mes modifications ont concerné l'architecture de la base de données, le pipeline d'intégration et l'interface graphique. J'ai ajouté les scores de fiabilité pour les entités, les documents et les scores globaux des relations. Pour cela j'ai proposé une visualisation avec des couleurs et des passages de souris.

J'ai rencontré quelques obstacles lors des modifications de l'interface. Le but était d'afficher le score d'occurrence lors d'un passage de souris sur la flèche reliant les entités dans les résumés (cf. 10), en plus de l'indication de couleur. Je n'ai pas réussi à réaliser cet affichage. Pour mettre un évènement de souris sur une partie de la page, je récupère cette partie via JavaScript puis je lui associe un évènement. J'ai réussi à récupérer les éléments HTML de la flèche. Cependant, l'évènement ne s'enclenche pas malgré mes tests des différentes manières d'ajouter l'évènement. Cela reste donc un point à approfondir lors de futurs travaux.

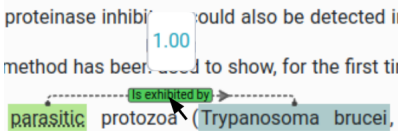


FIGURE 10 – Rendu voulu pour un passage de souris sur la relation

De plus, j'ai voulu mettre une fenêtre pop-up pour la visualisation du résumé, et non pas l'ouvrir dans une nouvelle page. J'ai réussi à créer la fenêtre pop-up, mais cela entraine un problème avec l'affichage des traits reliant les entités. Les entités sont surlignées avec la librairie Recogito, mais le trait n'apparait pas. Mon analyse est que le calcul de position des entités pour les relier entre elles n'a pas fonctionné dans la fenêtre pop-up. Il faudrait donc enquêter sur la façon dont les positions sont calculées pour pouvoir résoudre le problème.

J'ai également identifié quelques évolutions ou améliorations de l'application qui pourraient être intéressantes, que je décris ci-dessous.

4.1 Temps de chargement

Le chargement des résultats sur le site peut prendre de moins de 1s à 10s. J'ai fait plusieurs tests pour comparer le temps de réponse avec et sans les modifications que j'ai faites. Il est du même ordre de grandeur pour la majorité des requêtes. Comme la base de données que j'ai instanciée est plus petite que la base Omnicrobe complète, il est possible qu'à taille égale ma version soit plus lente. Pour certaines requêtes, dans certains navigateurs, le temps dépasse les 10s et le navigateur affiche un message indiquant que la réponse est lente.

Un temps de chargement légèrement plus long pourrait s'expliquer par l'ajout des tables "occurrence" et "document". En effet, pour toutes les requêtes nécessitant une information sur la source, je dois lier la table relation avec les tables occurrence et document. Cependant, je n'ai pas réussi à comprendre pourquoi certaines requêtes ne donnaient rien au bout de 3 min, sachant que ces mêmes requêtes peuvent parfois s'effectuer en moins de 10s sur un autre navigateur. D'autre part, une étude approfondie des requêtes permettrait de positionner des index adéquats facilitant l'exécution des requêtes et diminuant le temps de réponse.

4.2 Calcul de score

Les scores de document et scores globaux sont calculés par des moyennes, cette valeur est temporaire et permet d'avoir un score pour vérifier le fonctionnement du script parsant le JSON et tester l'affichage dans l'interface. Le score global est donc une moyenne des scores de différents documents d'une relation. Il va baisser si un seul score de document est mauvais. Or, on peut

supposer qu'avoir quelques bons scores donne un score global correct, et que celui-ci ne va pas trop diminuer à cause d'un document. J'ai donc réfléchi à une formule de calcul qui serait plus représentative de la fiabilité de la relation en fonction des scores de documents. Une première modification serait de faire une moyenne pondérée en donnant un poids plus important au haut score. Cela permettrait d'avoir un bon score dès que quelques documents sont très sûrs. Un deuxième aspect pourrait être pris en compte : le nombre de documents dans lequel apparaît une relation. En effet, plus une relation est fréquemment trouvée, plus elle a de chance d'être correcte. Ainsi, avoir quelques documents avec un bon score augmente le score global, et avoir quelques documents en plus, même avec des scores faibles, permet d'indiquer que l'on est encore plus sûr de cette relation. Un principe similaire pourrait s'appliquer pour le score de document.

Au niveau du tableau de résultats dans l'interface web, la colonne 'Ontobiotope class' indique le nom de l'élément dans le référentiel. La colonne 'Occurrence in text' indique toutes les formes possibles de l'élément trouvées dans les textes, chacune est associée à un score. Le score de la colonne 'Ontobiotope class' indiqué actuellement est le meilleur parmi ces scores. Ce choix est fait car il n'existe pas de score global au niveau de l'entité. Une évolution possible serait donc l'ajout d'un nouveau score qui pourrait être calculé avec un principe similaire à celui décrit précédemment. Le score global d'entité serait une moyenne pondérée des scores des formes de textes des entités. Cependant, la prise en compte du nombre de formes de textes semble moins cohérente dans ce cas.

4.3 Généricité de la base de données

Les modifications que j'ai apportées à la base de données permettent de gérer des relations entre un taxon et un élément (cf. figure 6), tout comme le schéma initial (cf. figure 5). Cependant, il est prévu d'utiliser l'application Omnicrobe sur des données issues de projets de recherche impliquant d'autres types d'entités. En parallèle du sujet de mon stage, j'ai pu faire quelques tests concernant les évolutions à apporter afin de rendre le schéma de la base de données plus générique.

La généralisation de la base se ferait en rassemblant les taxons et éléments en une table commune nommée "entité". Comme il y a de nombreuses instances de taxons et éléments, tout mettre dans une table pourrait ralentir le temps de réponse d'une requête. Deux versions du schéma de la base de données ont donc été proposées. La version 1 ne contient qu'une classe entité (cf. figure annexe 1), la version 2 contient deux classes entités (entity1, entity2) avec les mêmes attributs (cf. figure annexe 2). Cette deuxième version place les entités dans l'une des deux tables de manière à limiter les dédoublements d'information. J'ai instancié les deux versions, puis j'ai utilisé la clause "*EXPLAIN ANALYSE*" proposée par PostgreSQL pour effectuer des tests de performances sur des requêtes dans les deux bases. Cette partie n'étant pas le cœur de mon stage, je n'ai pu y consacrer qu'une petite partie de mon temps. Ce ne sont donc que des pistes d'évolution qu'il faudra approfondir.

4.4 Détection de relation

La détection de la présence d'une relation entre deux entités reconnues par la fouille de texte repose actuellement sur une méthode simple dont les performances sont assez limitées. En effet, d'après l'article sur Omnicrobe [1], dans la plupart des cas une relation est détectée si les deux entités sont dans la même phrase. La phrase " Le micro-organisme bacillus ne vit pas dans un accacia." contient deux entités : "bacillus" et "accacia". Le pipeline de fouille de texte détectera donc la relation bacillus-accacia. Ce stage avait pour but d'ajouter les scores, le pipeline de fouille de texte n'est donc pas une partie sur laquelle j'ai travaillé. Mais l'amélioration de la détection de relations serait une perspective intéressante. Des travaux sont d'ailleurs en cours dans l'équipe pour intégrer une méthode à base d'apprentissage profond (deep learning), dont les performances sont supérieures à la méthode actuelle.

Références

[1] Sandra Dérozier, Robert Bossy, Louise Deléger, Mouhamadou Ba, Estelle Chaix, Olivier Harlé, Valentin Loux, Hélène Falentin, and Claire Nédellec. Omnicrobe, an open-access database of microbial habitats and phenotypes using a comprehensive text mining and data fusion approach. *PloS one*, 18(1) :e0272473, 2023.

[2] Centre national pour l'information biotechnologique (NCBI). Pubmed. `https://pubmed.ncbi.nlm.nih.gov`, 1996.

[3] Centre national pour l'information biotechnologique (NCBI). Genbank. `https://www.ncbi.nlm.nih.gov/genbank/`, 1982.

[4] Leibniz Institute DSMZ German Collection of Microorganisms and Cell Cultures GmbH. Bacdive. `https://bacdive.dsmz.de`, 2012.

[5] INRAE. Cirm-bia. `https://collection-cirmbia.fr/`, 2020.

[6] INRAE. Cirm-levures. `https://cirm-levures.bio-aware.com/`, 2020.

[7] INRAE. Cirm-cfbp. `https://cirm-cfbp.fr/`.

[8] Claire Nédellec, Estelle Chaix, Robert Bossy, Louise Deléger, Sandra Dérozier, Jean-Baptiste Bohuon, and Valentin Loux. L'ontologie ontobiotope pour l'étude de la biodiversité microbienne. *Revue des Nouvelles Technologies de l'Information*, Extraction et Gestion des Connaissances, RNTI-E-34 :353–358, 2018.

[9] Yoan Blanc. Json. `https://he-arc.github.io/livre-python/json/index.html`, 2016-2017.

[10] Johannes Köster and Sven Rahmann. Snakemake—a scalable bioinformatics workflow engine. *Bioinformatics*, 28(19) :2520–2522, 08 2012.

[11] A Kenneth Reitz. Environnement virtuel en python. `https://python-guide-pt-br.readthedocs.io/fr/latest/dev/virtualenvs.html`, 2016.

[12] PS Lokhande, Fankar Aslam, Nabeel Hawa, Jumal Munir, and Murade Gulamgaus. Efficient way of web development using python and flask. 2015.

[13] Pelagios Commons. Recogito.js. `https://github.com/recogito/recogito-js`, 2022.

Annexes

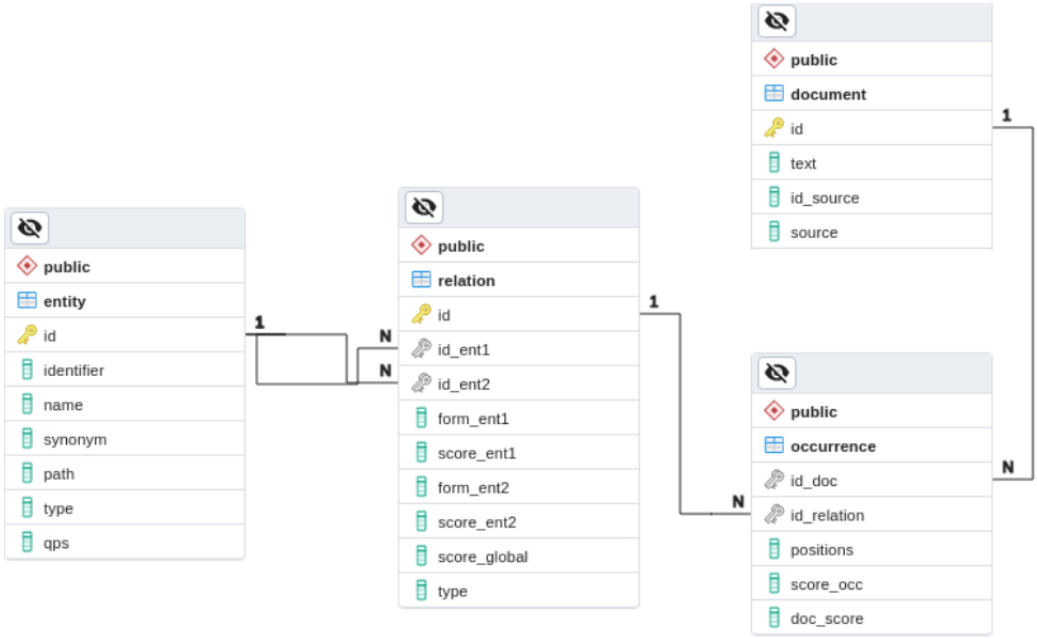


FIGURE 1 – [Annexe] Schéma de la version 1 de la base de données

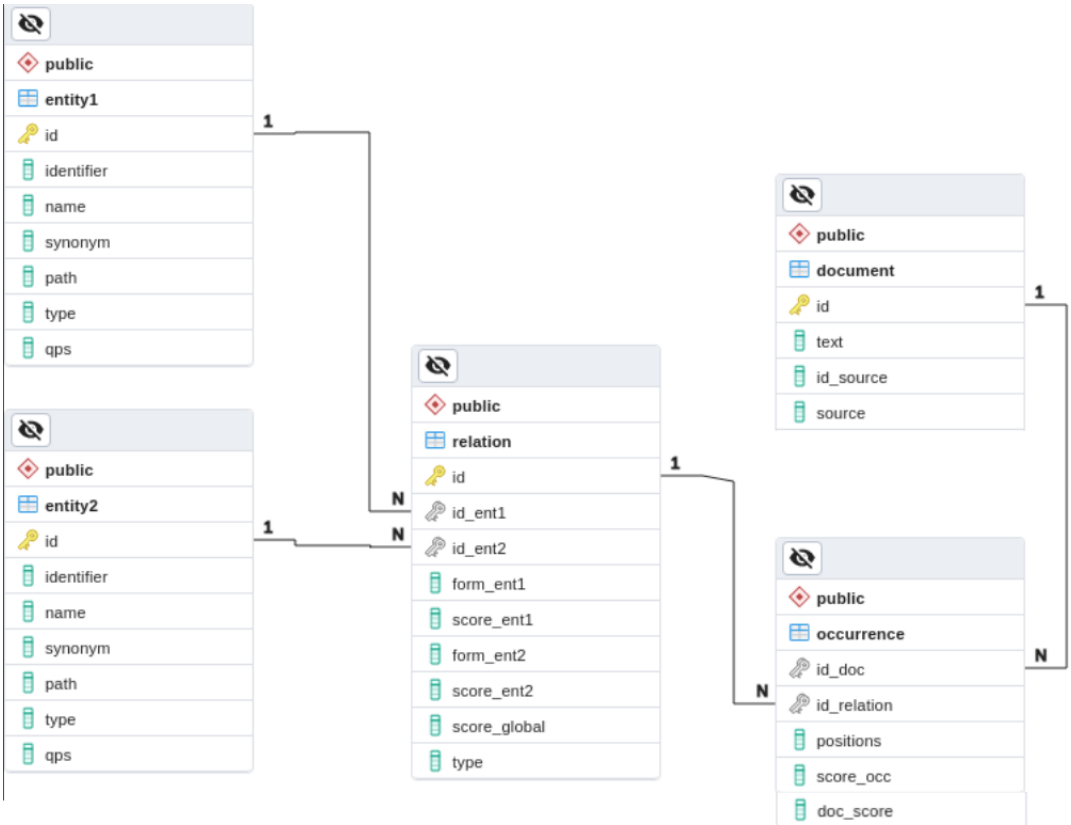


FIGURE 2 – [Annexe] Schéma de la version 2 de la base de données