Web Sémantique - Projet

Antoine-Marie MICHELOZZI, Jilian Lubrat
`github.com/mantoniu/MovieDB`

Janvier 2026

Table des matières

1 Introduction

Dans le cadre de ce projet, nous avons mis en place un graphe de connaissances centré sur le domaine du cinéma. L'objectif est de regrouper des informations sur les œuvres audiovisuelles, telles que les films, les séries et les documentaires, ainsi que sur les personnes qui y sont associées, notamment les acteurs et les réalisateurs. Ces informations permettent de contextualiser les œuvres et d'aider les utilisateurs à formuler leurs avis.

Le graphe intègre également des reviews d'utilisateurs, qui constituent un élément central du projet. Elles peuvent être consultées ou ajoutées par les utilisateurs, et servent de base à un système de recommandation visant à proposer des œuvres similaires à celles déjà appréciées.

2 Modélisation des Connaissances

Cette section détaille la structure logique de notre ontologie, conçue pour représenter l'univers cinématographique en s'appuyant sur les standards du Web Sémantique, comme l'utilisation de OWL et SKOS.

2.1 Classes

L'ontologie organise les œuvres en commençant par une catégorie très générale appelée `:Item`. Juste en dessous, nous trouvons la classe `:CreativeWork`, qui regroupe toutes les créations artistiques du graphe. Cette classe se spécialise en `:MotionPicture` pour l'audiovisuel, qui se divise ensuite en quatre types précis : les films (`:Movie`), les séries (`:TvSeries`), les téléfilms (`:TvMovie`) et les programmes spéciaux (`:TvSpecial`). Ces catégories sont séparées grâce à des règles de disjonction, le système empêche ainsi qu'une œuvre soit classée par erreur dans deux catégories incompatibles.

Pour les acteurs du monde du cinéma, nous utilisons la classe `:Agent` qui sépare les personnes (`:Person`) des organisations (`:Organization`). Les métiers comme `:Actor`, `:Director` ou `:Producer` sont des sous-classes de `:Person`. Cette organisation permet de lister précisément les rôles de chacun tout en gardant une structure simple.

Certaines classes professionnelles sont définies par ce que les personnes font réellement dans les données. Par exemple, une personne est automatiquement reconnue comme un `:Director` ou un `:Writer` si elle est liée à la direction ou à l'écriture d'au moins un film. De la même manière, les catégories `:Man` et `:Woman` sont liées à la valeur de la propriété `:gender`. Cela garantit que la classification d'une personne est toujours cohérente avec ses informations biologiques enregistrées.

2.2 Propriétés

L'utilisation de propriétés OWL permet d'inférer automatiquement de nouvelles connaissances à partir des relations définies dans le graphe. Ainsi, les propriétés telles que `:hasSpouse` et `:hasSibling` sont définies comme symétriques (*owl :SymmetricProperty*) et irréflexives (*owl :IrreflexiveProperty*), assurant la réciprocité des liens tout en interdisant qu'un individu soit son propre conjoint ou frère. Par ailleurs, la transitivité appliquée à la propriété `:hasDescendant` permet au raisonneur de déduire automatiquement des liens intergénérationnels complexes : si un individu A est le parent de B, et que B est le parent de C, alors C est formellement identifié comme un descendant de A.

La navigation au sein du graphe de connaissances est facilitée par la définition systématique d'inverses et de hiérarchies de propriétés. Des paires bidirectionnelles telles que `:directed` / `:hasDirector`, `:produced` / `:hasProducer` ou `:hasChild` / `:hasParent` permettent d'explorer les relations dans les deux sens. Cette structure est renforcée par une hiérarchie où les propriétés spécifiques (`:hasDirector`, `:hasActor`, etc.) sont déclarées comme des sous-propriétés de `:hasContributor`, autorisant ainsi des requêtes génériques sur l'ensemble des contributeurs d'une œuvre. Pour assurer l'interopérabilité avec des schémas externes, des équivalences de propriétés (*owl :equivalentProperty*) sont également établies, permettant l'intégration de données provenant de sources hétérogènes.

Enfin, l'intégrité et la profondeur sémantique du modèle reposent sur l'usage de propriétés fonctionnelles et de chaînes de propriétés. Les identifiants uniques et données biométriques, comme `:imdbId`, `:birthYear` ou `:gender`, sont définis comme fonctionnels (*owl :FunctionalProperty*) pour garantir qu'une instance ne possède qu'une seule valeur pour ces attributs critiques. En complément, l'utilisation de *Property Chains* (*owl :propertyChainAxiom*) permet des inférences avancées, comme la définition de la propriété `:actedIn` par la composition de `:hasRole` et `:inMotionPicture`. Grâce à ce mécanisme, le raisonneur peut automatiquement lier un interprète à une œuvre cinématographique dès lors qu'un rôle spécifique est créé, permettant une gestion fine des personnages tout en simplifiant les requêtes sur la filmographie des acteurs.

2.3 Thesaurus

Pour classer les œuvres par genre (action, comédie, drame, etc.), nous utilisons le standard SKOS. Chaque genre est modélisé comme un concept (`skos:Concept`) disposant d'un libellé et, lorsque cela est pertinent, d'une description. Les genres sont organisés selon une hiérarchie sémantique à l'aide des relations `skos:broader` et `skos:narrower`, ce qui permet de représenter des sous-genres et des spécialisations, par exemple lorsqu'un genre peut être considéré comme une déclinaison d'un genre plus général.

Cette approche offre une grande souplesse, car elle permet d'étendre ou d'affiner la classification sans modifier la structure globale de l'ontologie. Elle facilite également l'exploitation du graphe lors des requêtes, en permettant des recherches à différents niveaux de granularité. Les concepts de genre sont générés automatiquement lors de l'import des données, à partir des informations disponibles dans les sources utilisées.

2.4 Validation SHACL

L'utilisation de SHACL (*Shapes Constraint Language*) permet de vérifier que les données du graphe respectent une structure minimale et des contraintes cohérentes avec l'ontologie. Alors que l'ontologie décrit ce qui peut être déduit par raisonnement, les formes SHACL servent à valider ce qui doit effectivement être présent dans les données. Cette validation porte d'abord sur les identifiants et les littéraux, afin d'éviter les valeurs manquantes ou mal formées.

Par exemple, la shape `:NamePropertyShape` impose que chaque ressource principale possède au moins un nom non vide de type `xsd:string`. De la même manière, des contraintes de format sont appliquées à certains identifiants externes, comme `:imdbId`, qui doit respecter une expression régulière donnée, ou encore le code pays associé aux instances de `:Place`, qui doit suivre la norme ISO 3166-1 alpha-2.

Au-delà du simple formatage, le système exploite les cardinalités et les énumérations pour simuler des contraintes de fonctionnalité et d'intégrité. Les propriétés critiques telles que `:primaryTitle`, `:birthYear` ou `:gender` sont limitées par la contrainte `sh:maxCount 1`, interdisant toute ambiguïté sur ces attributs. La cohérence entre la classification des entités et les données est également vérifiée à l'aide de contraintes. Les shapes `:Actor` et `:Actress`, ainsi que `:Man` et `:Woman`, utilisent la contrainte `sh:hasValue` afin d'imposer une valeur de genre précise. Cela permet de s'assurer, par exemple, qu'une personne classée comme `:Actor` possède bien la valeur `"male"` pour la propriété correspondante.

Ces contraintes sont complétées par l'utilisation de `sh:not` pour exprimer des disjonctions explicites entre certaines classes, ce qui empêche notamment qu'une même instance soit à la fois un `:Movie` et une `:TvSeries`.

Enfin, certaines règles ne peuvent pas être exprimées uniquement à l'aide des contraintes SHACL classiques. Pour ces cas, nous utilisons des contraintes SPARQL (`sh:sparql`), qui permettent de vérifier des règles plus complexes. Elles sont notamment utilisées pour contrôler la cohérence temporelle des données, en empêchant par exemple qu'une année de fin de série ou de décès soit antérieure à l'année de lancement ou de naissance. Ces contraintes servent également à vérifier la cohérence des relations entre les entités. Lorsqu'une œuvre déclare un acteur via la propriété `:hasActor`, une validation croisée permet de s'assurer que cet acteur est bien relié à l'œuvre par un rôle ou une participation correspondante. De la même manière, les relations sociales telles que le mariage ou la fratrie sont vérifiées pour garantir leur symétrie, tandis que les relations entre parents et enfants sont contrôlées afin que chaque lien `:hasChild` possède bien son inverse `:hasParent`.

3 Construction du Graphe

3.1 Données structurées

Notre projet s'appuie sur des données structurées provenant de plusieurs sources publiques.

- Un jeu de données Kaggle regroupant un ensemble de critiques de films, issu du dataset *IMDb Review Dataset*. Les critiques sont fournies sous forme de fichiers JSON et contiennent notamment un identifiant unique de review, le nom du reviewer, l'œuvre concernée, une note sur 10 (lorsqu'elle est disponible), un résumé et le contenu détaillé de la critique, la date de publication, un indicateur de spoiler, ainsi qu'une information sur l'utilité de la review basée sur les votes des utilisateurs.
- Des données issues des jeux de données non commerciaux d'IMDb, disponibles via *IMDb Non-Commercial Datasets*. Ces données sont fournies sous forme de fichiers TSV interconnectés par des identifiants uniques, ce qui facilite la jointure entre les différents ensembles. Trois fichiers principaux sont utilisés :
  - les œuvres, regroupant les métadonnées descriptives des films, séries et téléfilms ;
  - les intervenants, recensant les professionnels du secteur (acteurs, réalisateurs, techniciens) ;
  - les contributions, servant de table d'association pour relier chaque personne à une œuvre, en précisant sa fonction et, pour les acteurs, le rôle interprété.

Le système repose sur un prétraitement des données en cascade : un échantillon d'œuvres (films, séries, téléfilms) est d'abord défini, permettant de filtrer les contributions et les intervenants (acteurs, réalisateurs, techniciens) associés, avant de filtrer les avis via une jointure sur les titres de films.

L'intégration des données dans le graphe RDF s'articule autour de trois processus de mapping distincts, s'appuyant sur les standards RML (*RDF Mapping Language*) et FnML (*Function Ontology*).

Le premier processus génère les instances de chaque classe conformément à l'ontologie définie, en renseignant leurs propriétés à partir des informations extraites des fichiers sources. Pour assurer la précision sémantique, des fonctions logiques telles que `grel:controls_if` et `grel:string_contains` sont utilisées pour déterminer dynamiquement le type d'une œuvre (ex : `:TvSeries` ou `:Movie`) selon les données sources. Par ailleurs, l'intégrité des données est maintenue grâce aux fonctions de la bibliothèque IDLab (`idlab:isNull`, `idlab:notEqual`), qui filtrent les entrées manquantes, tandis que `grel:string_split` permet de traiter les champs multi-valués comme les noms de personnages ou les genres des oeuvres.

Le second processus génère les genres cinématographiques qui ont été extrait du jeu de données et enrichis par des définitions durant le prétraitement. Ce mapping génère simplement l'URI et les triplet permettant de créer le concept SKOS associé au genre, avec sa définition. Cette approche permet de structurer un thésaurus sémantique avec le dataset, représentant au mieux les genres présents dans le dataset.

Le dernier processus de mapping traite les fonctions occupées par les personnes. Si la classe `:Person` et certaines sous-classes clés (telles que `:Director`, `:Producer`, `:Writer` ou `:Actor`) préexistent dans l'ontologie, les autres catégories de métiers sont générées dynamiquement en appliquant des transformations comme `grel:string_toTitlecase` et `grel:string_replace` sur les intitulés de postes. Cette approche permet au graphe de s'adapter à la diversité des rôles présents dans les données sources. En outre, des propriétés de contribution au format `:has[JOB]Contribution` sont générées dynamiquement via `grel:array_join` afin de lier directement chaque personne à une œuvre (`:MotionPicture`). Ces propriétés permettent également, par inférence, de définir des relations sémantiques plus directes (voir section 2).

3.2 Données non structurées

Pour l'intégration de données non structurées, nous avons fait le choix de récupérer la biographie des différentes personnes présentes dans le dataset d'IMDb, étant donné que l'on ne possède qu'un nombre restreint d'informations sur celles-ci. Ces informations sont obtenues à partir de Wikipédia, en extrayant automatiquement la section Biographie des pages correspondant aux personnes concernées. Dans l'ontologie, nous avons défini d'autres propriétés, telles que le lieu d'études, le conjoint, les enfants, ainsi que les frères et sœurs, dans le cadre d'un choix de modélisation des personnes, que l'intégration des biographies permet ensuite de renseigner.

Afin de récupérer les triplets, nous utilisons le modèle **gemini-2.5-flash** de Google avec un prompt lui demandant de respecter l'ontologie (limité aux Personnes), ainsi que les contraintes SHACL. Le prompt précise que seules les informations explicitement présentes dans la section biographie doivent être utilisées, sans inférence. Ensuite, pour être sûr que les données respectent bien les contraintes, nous avons utilisé **pyshacl** pour valider le Turtle généré. Si celui-ci n'est pas valide, il est renvoyé au LLM avec le rapport d'erreur afin qu'il corrige les triplets non conformes.

4 Alignement et Liage

Pour garantir l'interopérabilité et la richesse sémantique du graphe, nous avons mis en œuvre des processus d'alignement semi-automatisés, structurés selon trois axes :

- **Alignement d'ontologies** : Ce processus permet de lier nos classes et propriétés locales à celles issues de Schema.org, Wikidata ainsi qu'à une ontologie spécifique au domaine du cinéma (issue de semantics.id). Pour ce faire, nous utilisons AgreementMakerLight (AML), un système automatisé d'alignement d'ontologies auquel nous soumettons au format TTL les ontologies définies plus tôt. Ce traitement est suivi d'une vérification manuelle systématique afin de valider la pertinence et la précision sémantique des liens logiques établis.
- **Alignement des concepts SKOS** : L'alignement de notre thésaurus des genres est réalisé avec le référentiel ContentGenre issu de la base de données de l'EBU (European Broadcasting Union). Pour effectuer cette mise en correspondance, nous utilisons l'outil OnaGui. Ce logiciel facilite l'identification des liens entre les concepts sémantiques, un processus que nous complétons par une vérification manuelle rigoureuse. Nous procédons également à l'ajout manuel des termes omis par l'outil automatisé afin d'affiner la précision et la richesse de la hiérarchie conceptuelle.

- **Alignement des données** : Le liage des instances, qu'il s'agisse du nom des œuvres ou de celui des personnes, est réalisé vers Wikidata en s'appuyant sur les titres (ou noms) et leurs identifiants IMDb respectifs. Ce processus de réconciliation est effectué via l'écosystème Ontotext, incluant une vérification manuelle rapide pour garantir la précision des correspondances. Enfin, les fichiers finaux de liage des entités sont produits à l'aide d'un mapping généré liant l'URI de l'instance dans notre graphe avec l'URI wikidata.

5 Exploitation du Graphe

5.1 Requêtes SPARQL

5.1.1 Films polarisants

La requête SPARQL présentée en annexe (listing 1) vise à identifier les films les plus polarisants, c'est-à-dire ceux qui reçoivent simultanément un grand nombre de critiques très négatives ($notes \leq 3$) et très positives ($notes \geq 8$). La polarisation est calculée comme le produit des proportions de ces deux types de critiques, puis pondérée par le nombre total de critiques afin d'éviter les biais liés aux faibles volumes. Les résultats (tableau 1) montrent que des films grand public comme Charlie and the Chocolate Factory ou War of the Worlds apparaissent en tête du classement, ce qui suggère que leur forte exposition médiatique entraîne des réactions très contrastées du public. À l'inverse, certains films avec une polarisation brute élevée mais moins de critiques sont relégués plus bas par l'effet de la pondération.

5.1.2 Réalisateurs associés aux critiques les plus utiles

La seconde requête 2 s'intéresse à la relation entre les réalisateurs et l'utilité moyenne des critiques associées à leurs films, mesurée via les votes de pertinence des utilisateurs. En imposant un seuil minimal de films et de critiques, l'analyse se concentre sur des réalisateurs disposant d'un volume de données suffisant. Le tableau 2 met en évidence des réalisateurs relativement peu connus du grand public, tels que Brad Anderson ou Kurt Wimmer, dont les films génèrent néanmoins des critiques jugées très utiles. Cela suggère que certains styles ou genres cinématographiques encouragent des analyses plus détaillées et argumentées de la part des spectateurs, indépendamment de la notoriété du réalisateur.

5.1.3 Acteurs jouant en moyenne dans les films les mieux notés

La requête 3 calcule la note moyenne des critiques pour chaque acteur, en considérant uniquement ceux ayant participé à au moins cinq films et totalisant un nombre significatif de critiques. Les résultats présentés dans le tableau 3 montrent que des acteurs comme Kate Winslet, Christian Bale ou Morgan Freeman figurent parmi les mieux classés, ce qui reflète une filmographie globalement bien reçue par le public. Cette approche permet de lisser l'effet d'un succès ou d'un échec isolé et fournit un indicateur plus robuste de la qualité perçue des films associés à un acteur sur l'ensemble de sa carrière.

5.1.4 Comptage fédéré des récompenses d'acteurs via Wikidata

La dernière requête (listing 4, requête fédérée) illustre l'intérêt de l'utilisation de requêtes SPARQL fédérés en combinant les données locales de la base MovieDB avec celles de Wikidata. En reliant les identifiants IMDb des acteurs aux entités Wikidata correspondantes, il devient possible de compter automatiquement le nombre de récompenses reçues. Les résultats partiels (tableau 4) confirment la cohérence de l'approche, avec des figures emblématiques comme Meryl Streep ou Anthony Hopkins en tête du classement. Cette requête démontre la capacité du Web sémantique à enrichir une base de données locale avec des informations externes fiables, sans duplication des données.

5.2 Link Prediction

5.2.1 Mise en place de l'index

Nous avons choisi d'implémenter un système de recommandation de films pour mettre en œuvre la Link Prediction. Pour se faire nous avons tout d'abord enrichi notre graphe de connaissance à l'aide des synopsis disponibles sur Wikipedia, celui-ci n'ayant pas de données textuelles représentatives de chacun des films. Cela a été réalisé en utilisant WikiData, qui lie chaque élément à sa page Wikipedia et contenant l'id imdb des films, facilitant la jointure avec notre graphe.

Une fois les synopsis récupérés, ceux-ci ont été transformés en représentations vectorielles à l'aide d'un modèle d'embeddings sémantiques, **gemini-embedding-1.0**. Cette transformation permet de projeter chaque film dans un espace vectoriel où la proximité entre deux vecteurs reflète la similarité sémantique entre leurs synopsis.

Les vecteurs obtenus ont ensuite été normalisés afin de permettre l'utilisation de la similarité cosinus comme mesure de comparaison. Cette mesure permet d'évaluer dans quelle mesure deux films abordent des thématiques ou des contenus narratifs proches, indépendamment de la longueur des synopsis.

Enfin, l'ensemble des vecteurs a été indexé à l'aide de la bibliothèque **FAISS**, permettant une recherche efficace des films les plus similaires à un film donné. Cette recherche de similarité est utilisée pour identifier des films proches sémantiquement.

5.2.2 Recommandation

Pour recommander des films à un utilisateur, nous commençons par analyser son historique de notes à partir du graphe de connaissance. Les films ayant reçu une note supérieure à un seuil donné sont considérés comme appréciés par l'utilisateur, tandis que l'ensemble des films déjà vus est conservé afin d'éviter de les recommander à nouveau.

Les synopsis des films appréciés sont ensuite transformés en vecteurs à l'aide du même modèle d'embeddings que celui utilisé pour l'indexation globale. Ces vecteurs sont combinés en calculant une moyenne, ce qui permet de construire un profil représentant les préférences générales de l'utilisateur.

Ce profil est enfin comparé à l'ensemble des films indexés afin d'identifier les films les plus similaires sur le plan sémantique. Les films déjà vus sont exclus des résultats, et les plus proches restants constituent les recommandations finales proposées à l'utilisateur.

5.3 Graph RAG

5.3.1 Text-to-SPARQL

Pour mettre en place un chat capable de comprendre l'intention de l'utilisateur, executer une requête **SPARQL** et présenter les résultats, nous avons choisis de reposer sur la bibliothèque **LangChain**. Cette bibliothèque permet de mettre en place un agent et de lui mettre à dispositions différents tools, qu'il choisit selon le contexte d'utiliser ou non.

Cette approche présente de nombreux avantages. Tout d'abord, l'ontologie complète n'est pas fournie au LLM, car elle serait trop volumineuse et risquerait de dégrader la qualité des résultats. À la place, nous mettons à disposition de l'agent des outils d'exploration du graphe basés sur des requêtes **SPARQL**.

Deux tools principaux sont mis à disposition de l'agent pour explorer l'ontologie : le premier, **ontology_schema_tool**, permet d'obtenir une vue d'ensemble du schéma en listant les namespaces, les classes et les propriétés disponibles, et est utilisé par l'agent pour comprendre la structure globale du graphe avant de formuler une requête SPARQL ; le second, **property_details_tool**, fournit des informations détaillées sur une propriété donnée, notamment son domaine et sa portée (range), et est utilisé lorsque l'agent doit vérifier comment relier deux classes ou déterminer le type de valeurs associées à une propriété, afin de générer des requêtes **SPARQL** correctes et cohérentes avec l'ontologie.

Ensuite le tool **sparql_query_tool** permet d'exécuter la requête **SPARQL** générée par l'agent sur le graphe RDF de l'application. Il agit comme une interface entre le LLM et le graphe, les résultats sont formatés de manière lisible avant d'être renvoyés à l'agent. L'exécution est protégée par un mécanisme de timeout, évitant les requêtes trop complexes ou bloquantes, et garantissant la robustesse du système tout en permettant à l'agent d'explorer dynamiquement le graphe sans connaissance préalable complète de l'ontologie.

Enfin, l'agent peut exploiter le système de recommandation via le tool **user_recommendation_tool**, qui lui permet de proposer des films personnalisés à partir de l'historique et des préférences de l'utilisateur, tout en intégrant ces recommandations de manière naturelle dans le dialogue.

5.3.2 Embeddings

Dans le cadre du **Graph RAG**, nous avons développé un agent dédié à la recommandation de films reposant sur des embeddings appris à partir du graphe de connaissances. L'objectif de cet agent est d'exploiter la structure du graphe RDF afin d'identifier des films similaires en se basant sur leurs relations implicites. Cette approche vise à prédire des liens de similarité entre film.

Le graph embedding est construit à partir d'un graphe d'apprentissage regroupant les films et les entités qui leur sont associées, telles que les genres, les personnes impliquées dans la production et les interactions utilisateurs issues des critiques. Afin de limiter l'influence excessive de certains nœuds très connectés, un filtrage est appliqué pour supprimer les entités dont le degré dépasse un seuil fixé. Les représentations vectorielles des nœuds sont ensuite apprises à l'aide de l'algorithme **Node2Vec**, qui s'appuie sur des marches aléatoires biaisées pour capturer la structure locale et globale du graphe. À l'issue de l'apprentissage, seuls les embeddings correspondant aux films sont conservés et stockés pour la phase de recommandation.

Cependant, les expérimentations menées avec une recommandation reposant uniquement sur le graph embedding ont montré des limites. Les films proches dans l'espace vectoriel étaient souvent liés par des similarités structurelles, telles que le partage de mêmes personnes ou de mêmes schémas relationnels, sans nécessairement présenter une cohérence thématique ou narrative perceptible par un utilisateur. Cette observation met en évidence le fait que le graphe encode principalement des relations factuelles, mais ne capture pas directement le contenu sémantique des œuvres.

Pour pallier cette limitation, nous avons introduit une approche hybride combinant le graph embedding avec des embeddings textuels issus des synopsis des films. Les synopsis, préalablement encodés et indexés dans **FAISS**, sont récupérés à l'aide de leur identifiant IMDb. Pour chaque film disposant à la fois d'un embedding de graphe et d'un embedding de synopsis, les deux vecteurs sont normalisés puis concaténés afin de former une représentation hybride unique. Une normalisation finale est appliquée afin de permettre une comparaison robuste par similarité cosinus et d'équilibrer l'influence des deux sources d'information.

L'agent de recommandation s'appuie sur ces embeddings hybrides pour identifier les films les plus proches d'un film donné. Lorsqu'un utilisateur fournit un titre, le système retrouve le film correspondant dans le graphe, puis effectue une recherche des plus proches voisins dans l'espace vectoriel à l'aide d'un algorithme KNN basé sur la distance cosinus. Les résultats sont ensuite reformulés par l'agent conversationnel et présentés à l'utilisateur. Cette approche permet de combiner efficacement la proximité structurelle issue du graphe et la similarité sémantique issue du texte, produisant des recommandations plus cohérentes et plus pertinentes que l'utilisation du graph embedding seul.

5.3.3 Conclusion

Ces deux approches répondent à des besoins différents et se révèlent fortement complémentaires. L'approche *Text-to-SPARQL* permet d'interroger précisément le graphe de connaissances afin d'extraire des informations factuelles et littérales, telles que des relations explicites, des propriétés ou des valeurs numériques, tout en garantissant une parfaite traçabilité des résultats. Elle est particulièrement adaptée aux requêtes structurées et aux questions nécessitant une réponse exacte et interprétable.

À l'inverse, l'approche fondée sur les embeddings, qu'ils soient issus du graphe ou combinés à des représentations textuelles, permet de capturer des similarités implicites entre les entités. Elle ne repose pas uniquement sur les liens explicites du graphe, mais sur des proximités apprises, offrant ainsi la capacité de suggérer des relations ou des recommandations qui ne sont pas directement présentes dans les données structurées.

En combinant ces deux approches au sein d'un même système, il devient possible d'exploiter à la fois la précision du raisonnement symbolique et la flexibilité des méthodes vectorielles. Le *Text-to-SPARQL* fournit un accès contrôlé et explicable aux données du graphe, tandis que les embeddings enrichissent l'expérience utilisateur par des recommandations et des rapprochements sémantiques plus fins, aboutissant à un système hybride à la fois robuste, interprétable et performant.

6 Interface

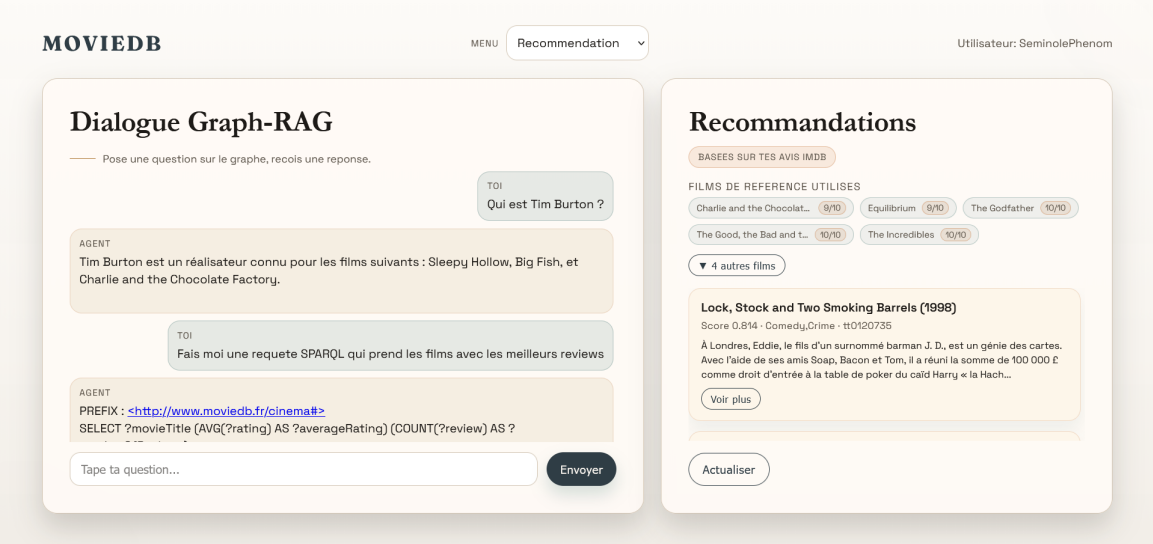

FIGURE 1 – Page principale de l'interface

Après avoir saisi son username sur la page d'accueil, l'utilisateur accède à la page principale de l'interface (voir Figure 1), composée de deux cartes. La première permet de dialoguer avec un agent conversationnel afin d'interroger le graphe RDF et d'obtenir des informations sur des acteurs, des films ou des séries. La seconde carte propose des recommandations personnalisées, basées sur les œuvres que l'utilisateur a le plus appréciées et sur ses avis précédents. Via le menu de navigation, l'utilisateur peut également rédiger un nouvel avis, une fois celui-ci soumis, le graphe est automatiquement mis à jour, ce qui permet d'actualiser les recommandations affichées.

7 Annexe

```
PREFIX : <http://www.moviedb.fr/cinema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT ?movie ?title ?nReviews ?polarization ?polarizationWeighted
WHERE {
  {
    SELECT ?movie ?title
           (COUNT(?r) as ?nReviews)
           (
             (xsd:decimal(SUM(?isLow)) / xsd:decimal(COUNT(?r)))
             * (xsd:decimal(SUM(?isHigh)) / xsd:decimal(COUNT(?r)))
             as ?polarization
           )
           (
             (
               (xsd:decimal(SUM(?isLow)) / xsd:decimal(COUNT(?r)))
               * (xsd:decimal(SUM(?isHigh)) / xsd:decimal(COUNT(?r)))
             )
             * xsd:decimal(COUNT(?r))
             as ?polarizationWeighted
           )
    WHERE {
      ?movie :primaryTitle ?title .
      ?r a :Review ; :isReviewOf ?movie ; :ratingValue ?rating .
      bind(if(?rating <= 3, 1, 0) as ?isLow)
      bind(if(?rating >= 8, 1, 0) as ?isHigh)
    }
    GROUP BY ?movie ?title
    HAVING (COUNT(?r) >= 50)
  }
}
ORDER BY DESC(?polarizationWeighted)
LIMIT 30
```

Listing 1: Films polarisants

Rang	Film	Critiques	Polarisation	Score Pondéré
1	Charlie and the Chocolate Factory	618	0.10	61.07
2	War of the Worlds	512	0.11	56.44
3	The Devil's Rejects	316	0.13	40.67
4	Sin City	262	0.13	35.36
5	Star Wars : Episode III - Revenge of the Sith	248	0.10	25.96
6	Wedding Crashers	252	0.09	22.33
7	Alexander	196	0.11	21.86
8	Batman Begins	349	0.06	21.18
9	The Island	397	0.05	20.13
10	Last Days	106	0.19	19.90
11	Stealth	215	0.08	17.76
12	Four Brothers	177	0.10	17.20
13	The Hitchhiker's Guide to the Galaxy	128	0.12	15.50
14	Land of the Dead	172	0.09	15.12
15	The Descent	132	0.11	14.58

TABLE 1 – Résultats de la requête 1 : Classement des films selon la « Polarisation Pondérée »

```
PREFIX : <http://www.moviedb.fr/cinema#>

SELECT ?directorName ?nMovies ?nReviews ?avgHelp
WHERE {
  {
    SELECT ?director ?directorName
           (COUNT(distinct ?movie) as ?nMovies)
           (COUNT(?r) as ?nReviews)
           (AVG(?help) as ?avgHelp)
    WHERE {
      ?movie a :MotionPicture ;
             :hasDirector ?director .
      ?director :name ?directorName .
      ?r a :Review ;
         :isReviewOf ?movie ;
         :helpfulnessVote ?help .
    }
    GROUP BY ?director ?directorName
    HAVING (COUNT(DISTINCT ?movie) >= 2 && COUNT(?r) >= 100)
  }
}
ORDER BY DESC(?avgHelp) DESC(?nReviews)
LIMIT 30
```

Listing 2: Réalisateurs dont les films ont les reviews les plus utiles

Rang	Réalisateur	Films	Critiques	Score Utilité
1	Brad Anderson	2	126	74.72
2	Kurt Wimmer	2	104	52.13
3	Breck Eisner	2	160	49.38
4	William Arntz	2	110	48.30
5	Betsy Chasse	2	110	48.30
6	Mark Vicente	2	110	48.30
7	Shane Carruth	2	102	48.30
8	Miranda July	2	152	48.28
9	Sam Raimi	8	144	47.53
10	Louis Leterrier	2	162	46.02
11	Mike Binder	2	114	44.95
12	Martin Scorsese	14	264	44.69
13	James Cameron	2	188	44.63
14	Christopher Nolan	2	746	42.44

TABLE 2 – Résultats de la requête 2 : Classement des réalisateurs selon l'utilité moyenne des critiques de leurs films

```
PREFIX : <http://www.moviedb.fr/cinema#>

SELECT ?actor ?actorName ?nMovies ?nReviews ?avgRating
WHERE {
  {
    SELECT ?actor
           (SAMPLE(?name0) as ?actorName)
           (COUNT(distinct ?movie) as ?nMovies)
           (COUNT(?r) as ?nReviews)
           (AVG(?rating) as ?avgRating)
    WHERE {
      ?movie a :MotionPicture ;
             :hasActor ?actor .

      ?r a :Review ;
         :isReviewOf ?movie ;
         :ratingValue ?rating .

      OPTIONAL { ?actor :name ?name0 . }
    }
    GROUP BY ?actor
    HAVING (COUNT(DISTINCT ?movie) >= 5 && COUNT(?r) >= 200)
  }
}
ORDER BY DESC(?avgRating) DESC(?nMovies) DESC(?nReviews)
LIMIT 50
```

Listing 3: Acteurs qui jouent en moyenne dans les films les mieux notés

Rang	Acteur	Films	Critiques	Note Moyenne
1	Kate Winslet	6	282	8.23
2	Simon Callow	6	342	8.18
3	Christian Bale	10	954	7.94
4	Kathy Bates	6	234	7.92
5	Morgan Freeman	6	568	7.80
6	Liam Neeson	10	844	7.74
7	Clint Eastwood	8	314	7.62
8	Gerard Butler	8	380	7.58
9	Leonardo DiCaprio	8	338	7.47
10	Emmy Rossum	6	434	7.41
11	Samuel L. Jackson	16	788	7.37
12	Bruce Willis	12	766	7.23
13	Helena Bonham Carter	8	1418	7.19
14	Mark Wahlberg	6	380	7.16
15	Johnny Depp	10	1484	7.16
16	Ian Holm	6	330	7.13
17	John Travolta	12	254	7.11
18	Nicolas Cage	14	214	7.07
19	Michael Caine	16	998	7.06
20	Freddie Highmore	6	1336	7.03
21	Christopher Walken	8	586	6.99
22	Jada Pinkett Smith	6	244	6.96
23	Natalie Portman	10	808	6.96
24	Uma Thurman	8	222	6.95
25	Jennifer Connelly	6	326	6.93

TABLE 3 – Résultats de la requête 3 : Classement des acteurs selon la note moyenne des critiques (Min. 5 films)

```
PREFIX wdt: <http://www.wikidata.org/prop/direct/>
PREFIX : <http://www.moviedb.fr/cinema#>

SELECT ?name (COUNT(?award) AS ?nbAwards)
WHERE {
  ?performer a :Performer ;
        :name ?name
  BIND(REPLACE(STR(?performer), ".*/", "") AS ?imdbId)

  SERVICE <https://query.wikidata.org/sparql> {
    ?wikidataActor wdt:P345 ?imdbId .
    ?wikidataActor wdt:P166 ?award .
  }
}
GROUP BY ?performer
ORDER BY DESC(?nbAwards)
```

Listing 4: Requête fédérée comptant les récompenses des acteurs via Wikidata

Rang	**Acteur**	**Récompenses (Wikidata)**
1	Meryl Streep	35
2	Woody Allen	34
3	Cicely Tyson	32
4	Anthony Hopkins	30
5	Sophia Loren	26
6	Sean Connery	26
7	Mel Brooks	26
8	Roberto Benigni	26
9	Jamie Foxx	26
10	Elizabeth Taylor	25
11	Helen Mirren	25
12	Jennifer Aniston	24
13	Laurence Olivier	23
14	Nicolas Cage	23
15	Dolly Parton	23
16	Vittorio De Sica	23
17	Audrey Hepburn	22
18	James Stewart	22
19	Clint Eastwood	22
...	...	...

TABLE 4 – Extrait des résultats de la requête 4 : Nombre de récompenses par acteur