(

Rapport projet AI11 – IA03

LLM IN E-COMMERCE

May 16, 2025

Contents

1 Introduction

1.1 Context and Motivation

Recommender Systems (RS) are software applications that support users in finding items of interest within extensive collections of objects, often in a personalized way [4]. Today, such systems are used in various application domains, including e-commerce and media streaming. Receiving automated recommendations in different forms has become a part of our daily online user experience. Internally, such systems analyze the past behavior of individual users or a user community as a whole to detect patterns in the data. On typical online sites, various relevant user actions can be recorded, such as viewing an item or making a purchase. Several actions by a single user may relate to the same item. These recorded actions and the detected patterns are then used to compute recommendations that match individual users' preference profiles.

1.2 Objectives

This ontology focuses on recommendation systems in the field of fashion applied to e-commerce. It aims to model the various methods and techniques used to provide personalized recommendations to users based on their preferences, interactions, and profiles.

The objective is to offer a comprehensive and reusable knowledge base for designers, developers, and researchers working on recommendation solutions in fashion. This ontology is also beneficial for e-commerce companies seeking to enhance customer experience and increase engagement through accurate and relevant recommendations.

It will be made publicly accessible through open platforms such as GitHub or LinkedIn to promote collaboration and innovation in this domain.

2 Survey of LLM-Based Recommendation Systems

2.1 Overview of Recommendation Systems

A recommender system (RecSys), or a recommendation system (sometimes replacing system with terms such as platform, engine, or algorithm), is a subclass of information filtering system that provides suggestions for items that are most pertinent to a particular user.Recommender systems are particularly useful when an individual needs to choose an item from a potentially overwhelming number of items that a service may offer.

Typically, the suggestions refer to various decision-making processes, such as what product to purchase, what music to listen to, or what online news to read.[1] Recommender systems are used in a variety of areas, with commonly recognised examples taking the form of playlist generators for video and music services, product recommenders for online stores, or content recommenders for social media platforms and open web content recommenders. [1]

2.2 Categorization

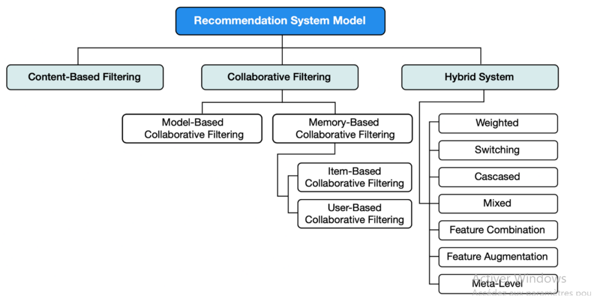

Recommendation System Model
Content-Based Filtering
Collaborative Filtering
Hybrid System
Model-Based Collaborative Filtering
Memory-Based Collaborative Filtering
Item-Based Collaborative Filtering
User-Based Collaborative Filtering
Weighted
Switching
Cascased
Mixed
Feature Combination
Feature Augmentation
Meta-Level

Figure 1: Overview of recommendation system categories .

A. **Collaborative Filtering**

The collaborative filtering method is based on gathering and analyzing data on user's behavior. This includes the user's online activities and predicting what they will like based on the similarity with other users. For example, if user A likes Apple, Banana, and Mango while user B likes Apple, Banana, and Jackfruit, they have similar interests. So, it is highly likely that A would like Jackfruit and B would enjoy Mango. This is how collaborative filtering takes place.

Two kinds of collaborative filtering techniques used are:

- User-User collaborative filtering
- Item-Item collaborative filtering

One of the main advantages of this recommendation system is that it can recommend complex items precisely without understanding the object itself. There is no reliance on machine analyzable content.

B. **Content-Based Filtering**

Content-based filtering methods are based on the description of a product and a profile of the user's preferred choices. In this recommendation system, products are described using keywords, and a user profile is built to express the kind of item this user likes.

For instance, if a user likes to watch movies such as Iron Man, the recommender system recommends movies of the superhero genre or films describing Tony Stark.

The central assumption of content-based filtering is that you will also like a similar item if you like a particular item.

C. **Hybrid Recommendation Systems**

In hybrid recommendation systems, products are recommended using both content-based and collaborative filtering simultaneously to suggest a broader range of products to customers. This recommendation system is up-and-coming and is said to

provide more accurate recommendations than other recommender systems.
Netflix is an excellent case in point of a hybrid recommendation system. It makes recommendations by juxtaposing users' watching and searching habits and finding similar users on that platform like the presents.This way, Netflix uses collaborative filtering.

The hybrid recommendation system has 7 approches.

1. Weighted:

   In a weighted recommendation system, multiple models can be defined to effectively interpret the dataset. This system combines the outputs of these models using fixed weights, meaning the weightings remain consistent across both the training and testing phases.
   For instance, a content-based model and an item-item collaborative filtering model can be combined, each contributing 50 percent to the final prediction. The static weighting ensures that each model's influence on the recommendation remains constant throughout the process.

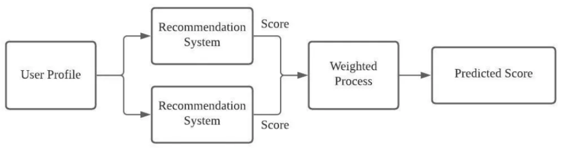


Figure 2: Weighted Hybrid Recommendation System

2. Switching:

   The switching hybrid approach dynamically selects a single recommendation system based on the specific context or situation. This method is particularly suitable for datasets that are sensitive at the item level. To implement this, selection criteria should be defined based on factors such as the user profile or other relevant features. This approach adds an extra decision-making layer on top of the recommendation model, determining which model to use in each scenario. It leverages the strengths and mitigates the weaknesses of the individual constituent models, ensuring the most appropriate system is applied for optimal results.

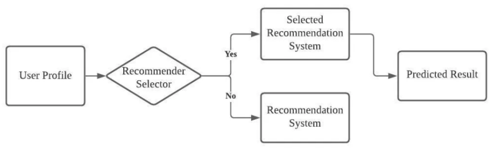


Figure 3: Switching Recommendation System

3. Mixed:

   The mixed hybrid approach begins by using the user profile and features to generate multiple sets of candidate datasets. These candidate sets are then fed into different recommendation models, which process them accordingly. The predictions from these models are subsequently combined to produce the final recommendation.

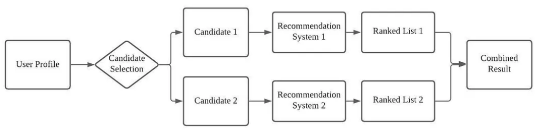


Figure 4: Mixed Recommendation System

4. Feature combination:

   In the feature combination hybrid approach, a virtual recommendation model is added to the system, functioning as a feature engineering layer that enhances the original user profile dataset.
   For instance, features from a collaborative filtering model can be integrated into a content-based recommendation model. This hybrid approach enables the system to incorporate collaborative data from the auxiliary model without relying solely on one recommendation model.

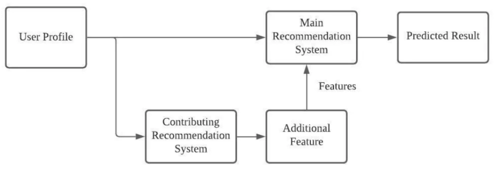

User Profile
Main Recommendation System
Predicted Result
Features
Contributing Recommendation System
Additional Feature

Figure 5: Feature combination Recommendation System

5. Feature augmentation:

   A contributing recommendation model is used to generate ratings or classifications for the user/item profile, which are then incorporated into the main recommendation system to produce the final prediction.
   The feature augmentation hybrid enhances the performance of the core system without altering the main recommendation model. For example, by applying association rules, the user profile dataset can be enriched. This augmented dataset improves the performance of the content-based recommendation model.

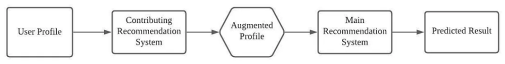

User Profile
Contributing Recommendation System
Augmented Profile
Main Recommendation System
Predicted Result

Figure 6: Feature augmentation Recommendation System

6. Cascade:

   The cascade hybrid approach establishes a strict hierarchical structure within the recommendation system, where the primary model generates the initial recommendation, and a secondary model is used to address minor issues with the primary result, such as breaking ties in the scoring.
   In practice, since most datasets are sparse, the secondary recommendation model can effectively handle issues like tie-breaking or missing data, improving the overall accuracy of the recommendations.

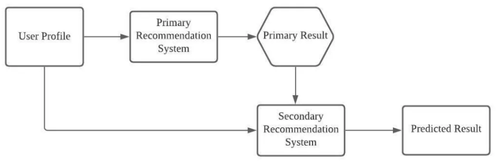

User Profile
Primary Recommendation System
Primary Result
Secondary Recommendation System
Predicted Result

Figure 7: Cascade Recommendation System

7. Meta-Level:
   The meta-level hybrid is similar to the feature augmentation hybrid in that the contributing model provides an augmented dataset to the main recommendation model. However, unlike feature augmentation, the meta-level hybrid replaces the original dataset with a model learned from the contributing system, which is then used as input to the main recommendation model.

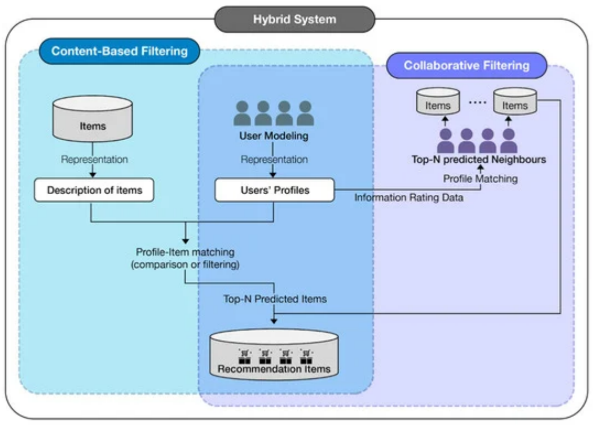


Figure 8: Process of general recommendation system models

2.3 Evolution of LLMs in Recommendations

With the advancement of understanding semantics and user behavior, recommendation systems have undergone significant evolution. Over time, various techniques and models have been developed to address different challenges in providing personalized recommendations. Among these, sequential recommendation systems have emerged as a key focus due to their ability to model user behavior over time and predict future interactions. This progression reflects the continuous refinement of methods, starting from early probabilistic models like Markov Chains to more advanced neural network-based approaches and multimodal frameworks, each designed to tackle specific recommendation scenarios.

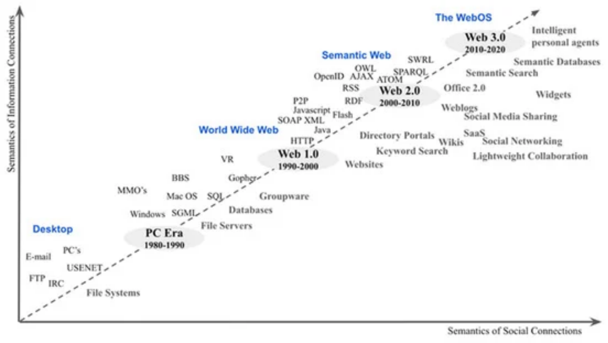


Figure 9: Web development

A. **Markov Chain-Based Models**

Markov chain-based recommender systems (RSs) adopt Markov chain models to model transitions over user–item interactions in a sequence to predict the next interaction. These methods represent one of the earliest approaches in recommendation systems, predating many modern techniques. According to the specific technique used, Markov chain-based RSs are divided into basic Markov Chain-based approaches and latent Markov embedding-based approaches. The former directly calculates the transition probability based on explicit observations, while the latter embeds the Markov chains into Euclidean space and then calculates the transition probabilities between interactions based on their Euclidean distance [26]. However, the shortcomings of Markov chain-based RSs are apparent; they can only capture short-term dependencies while ignoring long-term ones due to the Markov property, which assumes that the current interaction depends on one or several most recent interactions only. Additionally, they can capture point-wise dependencies while ignoring collective dependencies over user–item interactions. Consequently, as newer techniques have emerged to address these limitations, Markov chain-based methods have been less employed in SRSs in recent years.

B. **Sequential recommendation system**

Recommender systems have evolved significantly since the 1990s, starting with basic heuristics for content-based and collaborative filtering. In the late 2000s, matrix factorization emerged as a dominant approach, but its limitations in handling complex user interactions and intricate item features became apparent. The mid-2010s witnessed a revolution in machine learning with the rise of deep learning, whose success in areas like speech recognition and natural language processing sparked a new wave of research. Recognizing the potential of deep learning to handle complex data patterns, researchers began applying these techniques to recommender systems, leading to significant advancements in recent years.

Sequential recommendation systems focus on modeling the sequential behavior of users as they interact with items over time. Key concepts in SRS include:

- User–Item Interactions: Representing the historical interactions between users and items, such as clicks, views, purchases, and ratings.
- Sequence Modeling: Techniques for capturing the temporal dependencies in user interactions and predicting future behavior based on past interactions.
- Session-based vs. Long-term Preferences: Distinguishing between short-term preferences within a session and long-term preferences that evolve over multiple sessions.

C. **Multimodal Recommendation Systems**

To further enhance the capabilities of Sequential Recommendation Systems, Multimodal Large Language Models (MLLMs) have been integrated. These models combine the power of multimodal data (images, videos, text, etc.) and large language models (LLMs), which allow for better understanding and interpretation of user-item interactions across different types of media. MLLMs transform multimodal data into a unified textual semantic space, making it possible to leverage rich visual and textual information within the recommendation process. This integration addresses several challenges, such as capturing long-term temporal dynamics of user preferences and improving the personalization of recommendations.
The technique works by combining sequence modeling and multimodal fusion, where MLLMs enhance the understanding of both user behaviors and item features. These models are especially effective in Sequential Recommendation Systems (SRS) because they adapt to both short-term session-based preferences and long-term evolving interests, offering more personalized and accurate recommendations over time. By leveraging MLLMs, systems can handle the complexities of multimodal data while improving the interpretability and accuracy of the recommendations, especially in cases where sequential data is critical to understanding user preferences.

2.4 Techniques for LLM-Based Recommendation

A. **Text Mining**

Text mining is a technique for discovering useful text information by extracting text-related information from data. With the recent development of natural language processing technology, semantically important information has been extracted from the corresponding text [

B. **KNN (K-Nearest Neighbor)**

K-Nearest Neighbor (KNN) is an algorithm that classifies K-nearest neighbors of test tuple and train tuple to classify a dataset. KNN classifies datasets based on the closest distance by comparing the similarity between each item of data through distance-based weighting [**67**]. Euclidean distance, cosine similarity, and Pearson correlation are mainly used as measures to compare similarities. When the KNN algorithm is used in a recommendation system, the user's search pattern can be classified and the user's future preference can be predicted. After analyzing the patterns of user behavior data, such as users' web server logs and clickstream data, it can be used to classify items similar to users' tastes, and then use the results to recommend suitable items.

C. **Clustering**

Clustering is an algorithm that identifies finite categories or clusters to describe data, and is widely used in recommendation systems because of its low redundancy and ambiguity [**69**]. There are many different types of clustering techniques used in the recommendation system, but K-means clustering is mainly used. K-means clustering is an algorithm that clusters around the mean after setting the number of K clusters. After calculating the similarity between all the data in the recommendation system, it is assigned to the nearest cluster and the calculation is repeated in the order of calculating the cluster center [**70**]. However, if the number of clusters is small, K-means clustering is vulnerable to the scalability problem, in which the calculation speed decreases when the number of users and items increases while the recommendation system is servicing.

D. **Matrix Factorization**

Matrix Factorization recommendation system became widely known through the Netflix Prize, and especially solves the problem of scarcity in Collaborative Filtering . Matrix Factorization is a method to characterize items and user data after inferring elements from user evaluation data for items and storing them as vectors. The main purpose of this technique is to find the dimension of the latent factor that expresses the user information and the user's preference by storing the user's evaluation data in the Rating Matrix . In particular, it offers good scalability and flexibility as it can analyze users' tastes by using data such as mouse movement and search pattern, as well as explicit data that the user directly evaluates for a specific item.

E. **Neural Network**

In recent years, the use of neural networks has expanded in various fields, such as speech recognition, image recognition, photo search, and language translation. On

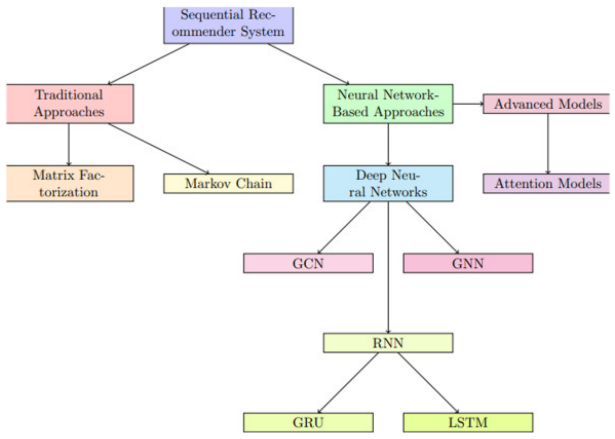


Figure 10: Taxonomy of sequential recommendation techniques

the other hand, although the introduction and use of neural networks in the recommendation system field is relatively small compared to other fields, many studies are being conducted as one of the main areas of interest in recommendation-system-related research. It has been widely used in research to obtain additional data in situations where it is difficult to understand user preferences with historical data [**81**,**82**]. In addition, He et al. [**83**] modeled noisy implicit feedback data using a deep neural network (DNN) to improve the performance of a recommendation system; deep learning offers the potential to improve the performance of recommendation systems. In other words, in developing a recommendation system, neural networks are used for modeling research to additionally secure and supplement data to solve the problem of sparsity and cold start in Collaborative Filtering, or for the purpose of improving the performance of the recommendation system itself.

Sequential recommendation techniques

This section categorizes and describes different approaches for modeling sequential user behavior in recommendation systems. As shown in 10, the taxonomy of sequential recommendation techniques is categorized into two main approaches: Traditional Approaches and Neural Network-Based Approaches. These categories encompass various models and techniques that enhance the recommendation process based on user interactions over time. Traditional approaches.

Traditional approaches consist of two primary methods: Matrix Factorization and Markov Chain. Matrix Factorization is a technique that decomposes the user–item interaction matrix into latent factors representing users and items, facilitating the prediction of user preferences. Markov Chain models capture the sequential dependencies between items, predicting the next item based on the previous items in the sequence.

Neural network-based approaches.

Neural network-based approaches are divided into Deep Neural Networks and Advanced Models. Deep Neural Networks include several models, such as Graph Convolutional Networks (GCN), Graph Neural Networks (GNN), and Recurrent Neural Networks (RNN). GCN and GNN leverage graph structures to capture complex relationships between items and users, enhancing the recommendation process by considering higher-order connections. RNNs are particularly effective in modeling sequential data due to their ability to maintain hidden states that capture information from previous time steps. Variants of RNNs include Gated Recurrent Units (GRU) and Long Short-Term Memory (LSTM), which address issues such as vanishing gradients and long-term dependencies.

(a) Markov chain-based models

Markov chain-based recommender systems (RSs) adopt Markov chain models to model transitions over user–item interactions in a sequence to predict the next interaction. According to the specific technique used, Markov chain-based RSs are divided into basic Markov Chain-based approaches and latent Markov embedding-based approaches. The former directly calculates the transition probability based on explicit observations, while the latter embeds the Markov chains into Euclidean space and then calculates the transition probabilities between interactions based on their Euclidean distance [26]. The shortcomings of Markov chain-based RSs are apparent; they can only capture short-term dependencies while ignoring long-term ones due to the Markov property, which assumes that the current interaction depends on one or several most recent interactions only. Additionally, they can capture point-wise dependencies while ignoring collective dependencies over user–item interactions. Consequently, they have been less employed in SRSs in recent years.

(b) Matrix factorization with sequence modeling

The iALS1 algorithm is designed for learning a matrix factorization model to recommend items to users by analyzing user interactions (implicit feedback) instead of explicit ratings to suggest the top-n items a user might be interested in. Furthermore, the study in introduces a method combining matrix factorization and Markov chains to enhance recommender systems. This method uses personalized transition graphs and factorizes a transition cube with a pairwise interaction model, outperforming traditional matrix factorization and unpersonalized Markov chain models. Fossil combines Matrix Factorization, Markov Chains, and similarity-based methods to predict personalized sequential behavior in recommender systems, showing improved performance on sparse, real-world datasets. [2]

Since our ontology topic focuses on LLMs in e-commerce, particularly in fashion recommendation, we observe that this method is applied in fashion recommendation systems. Below is an explanation of its implementation.

3 Methodology 101

The design of our ontology is inspired by the **Methodology 101** proposed by Natalya F. Noy and Deborah L. McGuinness from Stanford University. It consists of a series of steps outlined below:

(a) Define the domain and scope

(b) Consider potential reuse of existing ontologies

(c) Enumerate the important terms in the ontology

(d) Define the classes and the class hierarchy

(e) Define the properties of the classes – attributes

(f) Define the facets of the attributes

(g) Create the instances

(h) Refine the ontology – completeness

4 Ontology construction

4.1 Ontology purpose and domain

The Main objectif of the ontology is to perform a knowledge base in recommandation system using Large language model in ecommerce. we will particulary experiment the fashion world.

4.2 Reused ontologies

As a remarkable advantage of ontology is the interoperability of the taxonomies, we find existing own to reuse them.

- FOAF

As we are working on related persons domain, we used the well known Friend of a Friend ontology. FOAF is a project devoted to linking people and information using the Web. Regardless of whether information is in people's heads, in physical or digital documents, or in the form of factual data, it can be linked. FOAF integrates three kinds of network: *social networks* of human collaboration, friendship and association; *representational networks* that describe a simplified view of a cartoon universe in factual terms, and *information networks* that use Web-based linking to share independently published descriptions of this inter-connected world. This vocabulary 11 helps us to define all about users.

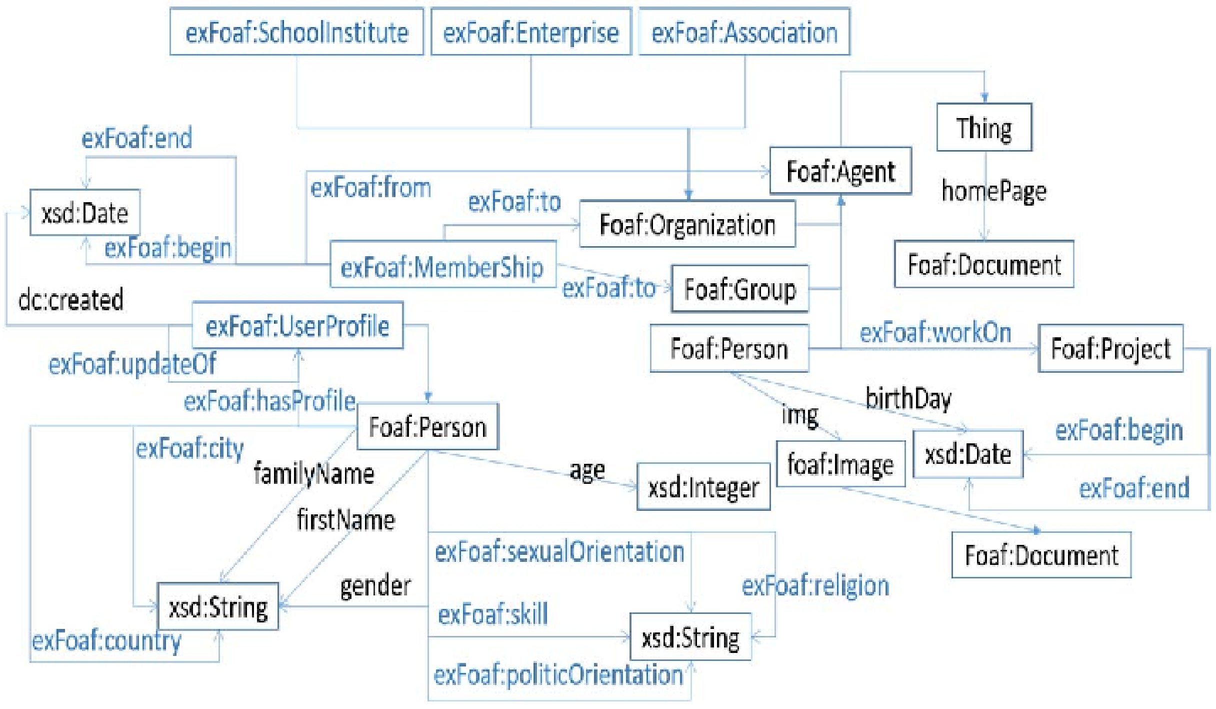


Fig. 3. Vue du module étendu exFOAF

Figure 11: FOAF title

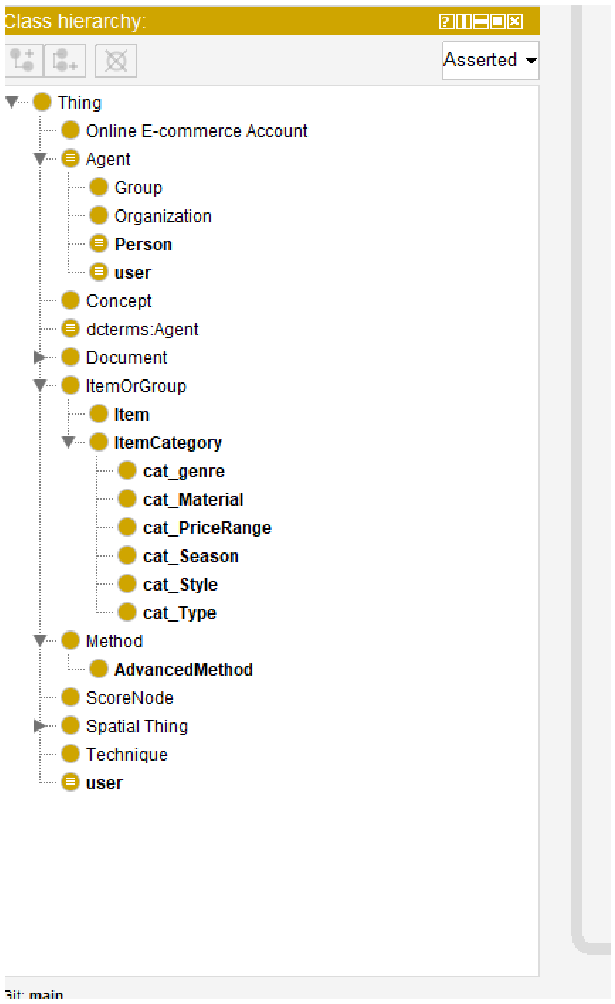
- dbpedia, Lov.linkeddata

Thus, we use the SPARQL end point of dbpedia to find a related ontology. Using key words like "user", "recommandation", "article", "e-commerce", and switching the language, but any response. In the same way, linked open vocabularies didn't show anything related.

4.3 Classes and class hierachy

For the development of the classes in our ontology, we adopted a top-down approach, starting from general concepts and gradually specializing them. This method aligns with sequential recommendation techniques used in recommendation systems. We identified a set of main classes and several related subclasses within the ontology, ensuring they capture the necessary functionalities to represent both content-based and collaborative filtering techniques.

The class hierarchy enables us to group related concepts under a common parent class, which shares a similar purpose. By doing this, we maintain a clear and organized structure in our ontology, making it easier to implement different recommendation techniques and facilitate further expansion.
Here is our classes and subclasses: 12



Class hierarchy:

Asserted

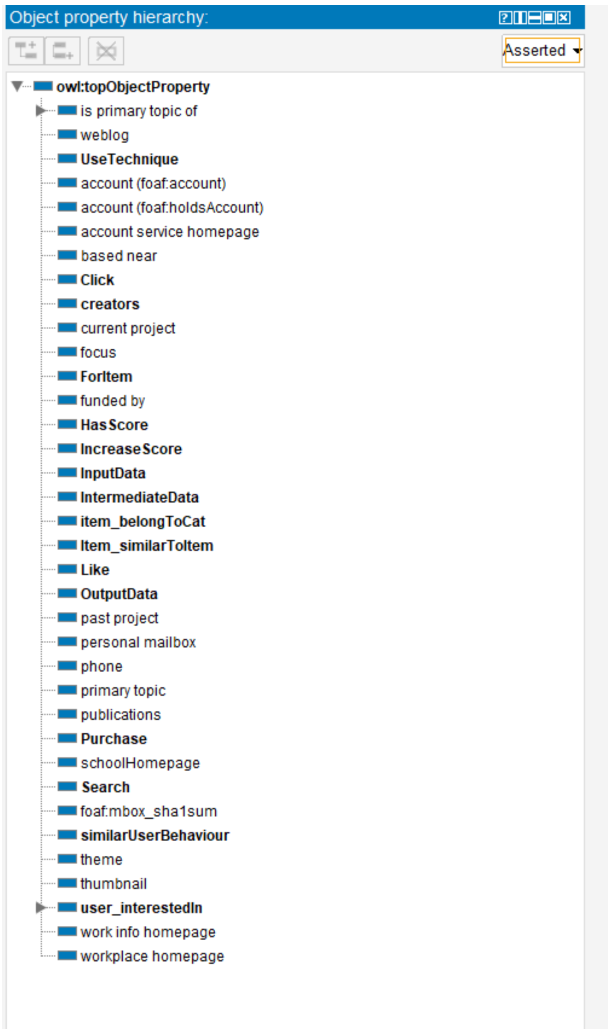
- Thing
  - Online E-commerce Account
  - Agent
    - Group
    - Organization
    - **Person**
    - **user**
  - Concept
  - dcterms:Agent
  - Document
  - ItemOrGroup
    - **Item**
    - **ItemCategory**
      - **cat_genre**
      - **cat_Material**
      - **cat_PriceRange**
      - **cat_Season**
      - **cat_Style**
      - **cat_Type**
  - Method
    - **AdvancedMethod**
  - ScoreNode
  - Spatial Thing
  - Technique
  - **user**

Git: main

Figure 12: Class hierachy

4.4 Properties

We then defined the internal structure of the ontology. To do this, we reviewed the specific characteristics of each class and the relationships between them from our list of elements. In this way, we defined the properties of our ontology, both for objects and for data. 13


Object property hierarchy:

Asserted

- owl:topObjectProperty
  - is primary topic of
  - weblog
  - UseTechnique
  - account (foaf:account)
  - account (foaf:holdsAccount)
  - account service homepage
  - based near
  - Click
  - creators
  - current project
  - focus
  - ForItem
  - funded by
  - HasScore
  - IncreaseScore
  - InputData
  - IntermediateData
  - item_belongToCat
  - Item_similarToItem
  - Like
  - OutputData
  - past project
  - personal mailbox
  - phone
  - primary topic
  - publications
  - Purchase
  - schoolHomepage
  - Search
  - foaf:mbox_sha1sum
  - similarUserBehaviour
  - theme
  - thumbnail
  - user_interestedIn
  - work info homepage
  - workplace homepage

Figure 13: Properties

4.5 Pattern Implemented

(a) **abstract class factory**
Is a design pattern that provides a way to create families of related objects without imposing their concrete classes, by encapsulating a group of individual factories that have a common theme without specifying their concrete classes. Thus, we have the class "ItemOrGroup", which allows us to represent iteraction between a user and either a single item, or a group of it. For instance, he can "search" for "Trouser" or "Winter shoes".14

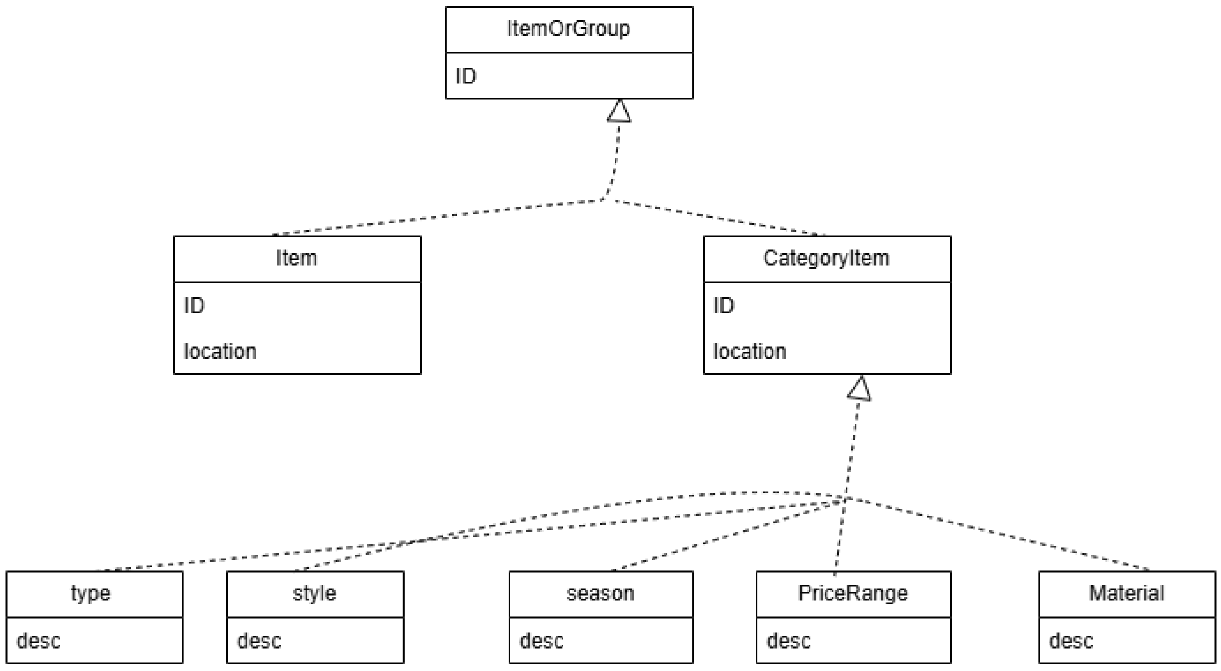


Figure 14: Abstract factory class

(b) **Blank node**
We faced an issue : a recommendation method at the end, increase a score associated to a user for an item. Meanwhile RDF language is the association of triple, we use a blank node. In RDF, a blank node is a node in an RDF graph representing a resource for which a URI or literal is not given.[1] The resource represented by a blank node is also called an anonymous resource. According to the RDF standard a blank node can only be used as subject or object of an RDF triple. So, here 15 is the blanck node : scoreNode, with three properties.

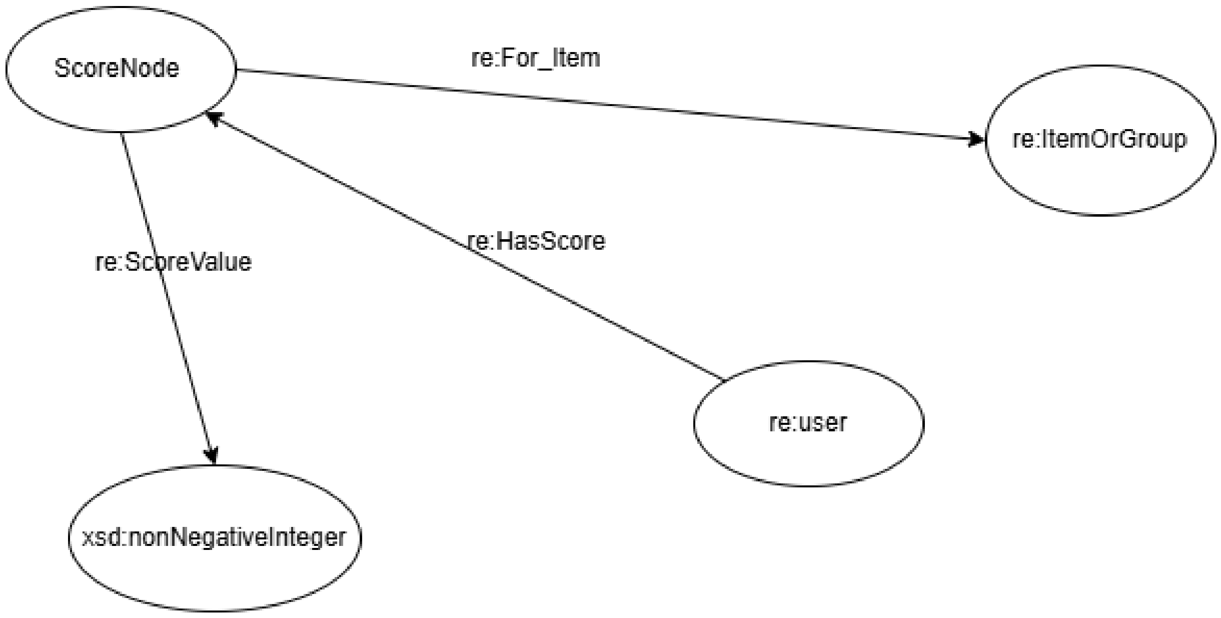


Figure 15: RDF blank node diagram

(c) **subproperties**
In the context of ontologies and the tool Protégé, subproperties are used to define hierarchical relationships between properties.
A subproperty is a property that is defined as a specialization of another property, known as the superproperty. This is expressed using the rdfs:subPropertyOf relationship in RDF and OWL. We express the fact that whether a user like, purchase, search or click on a item or a group of item, he is interested in. And we can also have a list of interested topic during the registering of the user, like many e-commerce sites do nowadays.
Then, the 16 shows the implementations

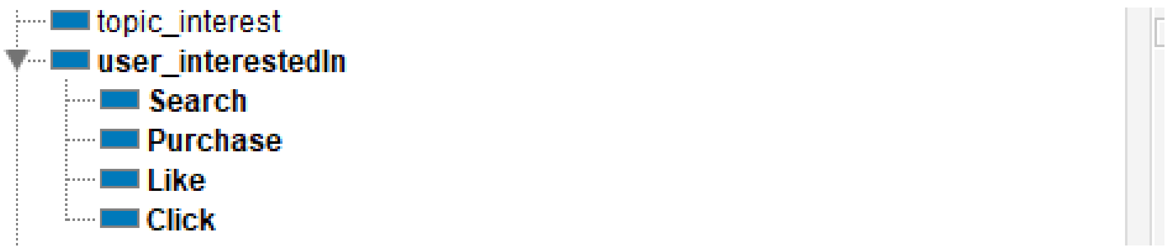
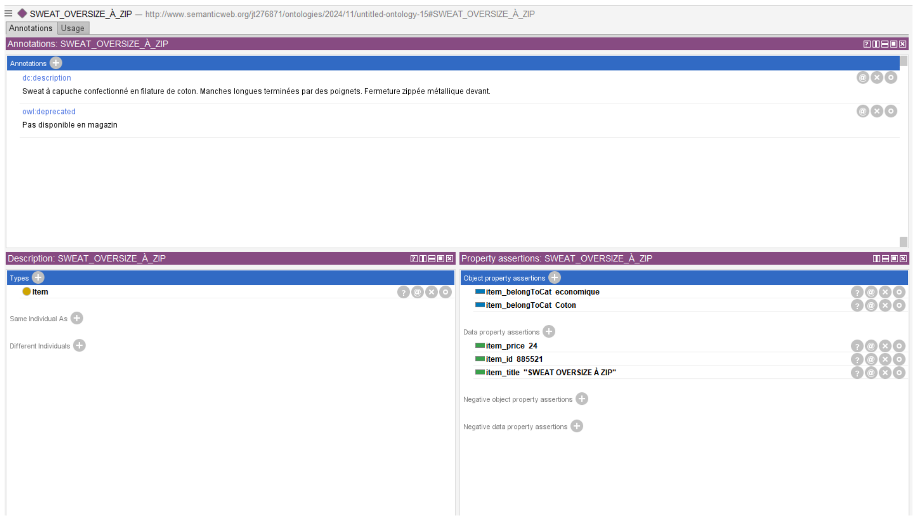


Figure 16: Sous-properties

4.6 Instances

In an ontology, instances (or individuals) represent specific data points or entities within the conceptual framework defined by the ontology. They play a crucial role in connecting the abstract structure of an ontology (classes and properties) to real-world data or concrete examples. We choose to feed our ontology with real data, from Zara website. This approach helps us to face real categorization problem, and

adapt the structure.
The 17 represent an item, while 18 is a registred user



SWEAT_OVERSIZE_À_ZIP — http://www.semanticweb.org/jt276871/ontologies/2024/11/untitled-ontology-15#SWEAT_OVERSIZE_À_ZIP

Annotations | Usage

Annotations: SWEAT_OVERSIZE_À_ZIP

Annotations

dc:description
Sweat à capuche confectionné en filature de coton. Manches longues terminées par des poignets. Fermeture zippée métallique devant.

owl:deprecated
Pas disponible en magazin

Description: SWEAT_OVERSIZE_À_ZIP

Types
Item

Same Individual As

Different Individuals

Property assertions: SWEAT_OVERSIZE_À_ZIP

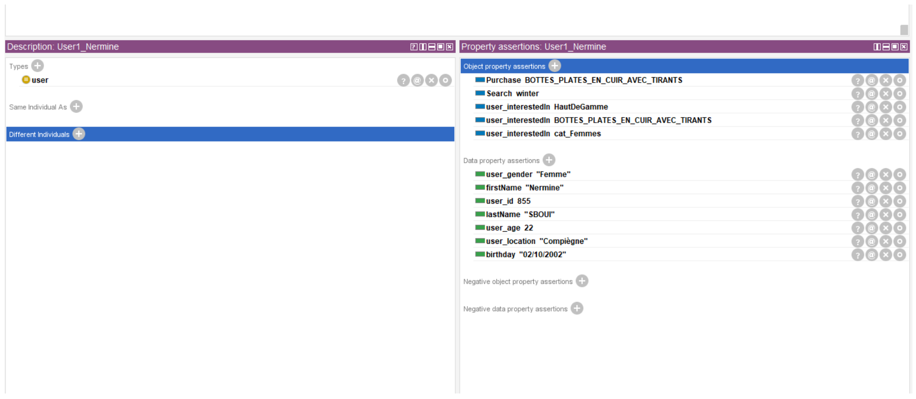
Object property assertions
item_belongToCat economique
item_belongToCat Coton

Data property assertions
item_price 24
item_id 885521
item_title "SWEAT OVERSIZE À ZIP"

Negative object property assertions

Negative data property assertions

Figure 17: An item



Description: User1_Nermine

Types
user

Same Individual As

Different Individuals

Property assertions: User1_Nermine

Object property assertions
Purchase BOTTES_PLATES_EN_CUIR_AVEC_TIRANTS
Search winter
user_interestedIn HautDeGamme
user_interestedIn BOTTES_PLATES_EN_CUIR_AVEC_TIRANTS
user_interestedIn cat_Femmes

Data property assertions
user_gender "Femme"
firstName "Nermine"
user_id 855
lastName "SBOUI"
user_age 22
user_location "Compiègne"
birthday "02/10/2002"

Negative object property assertions

Negative data property assertions

Figure 18: A user

5 Rules and queries

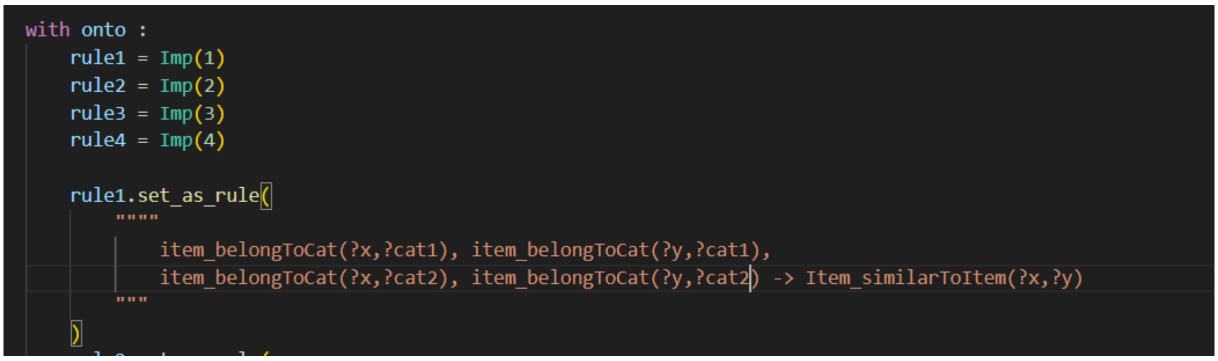
5.1 SWRL Rules inferences

SWRL (Semantic Web Rule Language) extends the reasoning capabilities of an ontology by enabling the expression of rules that go beyond the standard OWL (Web Ontology Language) reasoning framework. These rules operate on the classes, properties, and individuals (instances) in an ontology, enabling complex inferences. Thus, we add rules to complete our ontology and express the following facts :

- If two items are in at least two categories in common, they are similar.
- If two users are interested in at least three items, they have the same behavior.

- Each recommendation method increase the rate, following specifics criteria.
- At the end, the system, recommend to the user, item that has the highest rate, successively.

we choose to write these rules with the **owlready2** library, to keep a trace of them, and to offer the possibility to modify the parameters of the rules.



```
with onto :
    rule1 = Imp(1)
    rule2 = Imp(2)
    rule3 = Imp(3)
    rule4 = Imp(4)

    rule1.set_as_rule(
        """
            item_belongToCat(?x,?cat1), item_belongToCat(?y,?cat1),
            item_belongToCat(?x,?cat2), item_belongToCat(?y,?cat2) -> Item_similarToItem(?x,?y)
        """
    )
```

Figure 19: rule1

Notice that to make more flexible the parameter n which represent the number of commons categories items must have, we can use built-ins vocabularies. It support dynamic counting (more than three shared items), (swrlb:count).

5.2 RDF Graph

The RDF graph associated with our ontology is presented below:

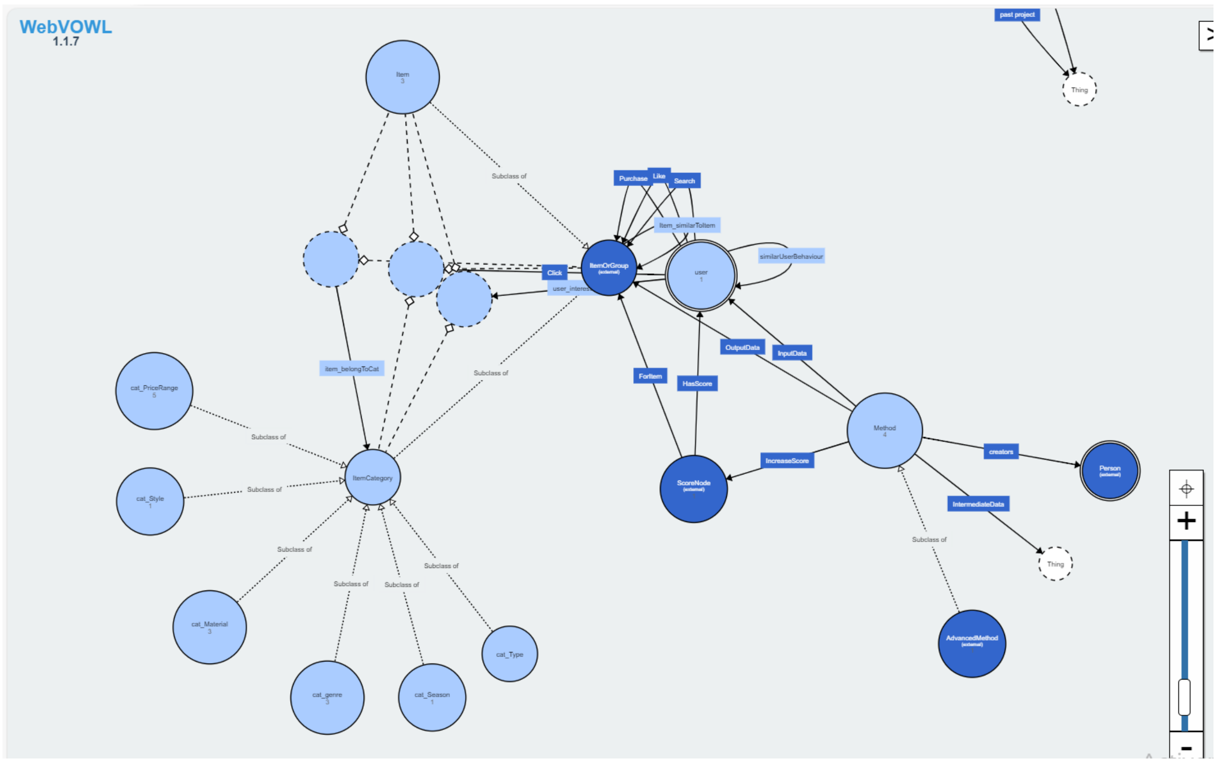


Figure 20: RDF Graph Associated with Our Ontology

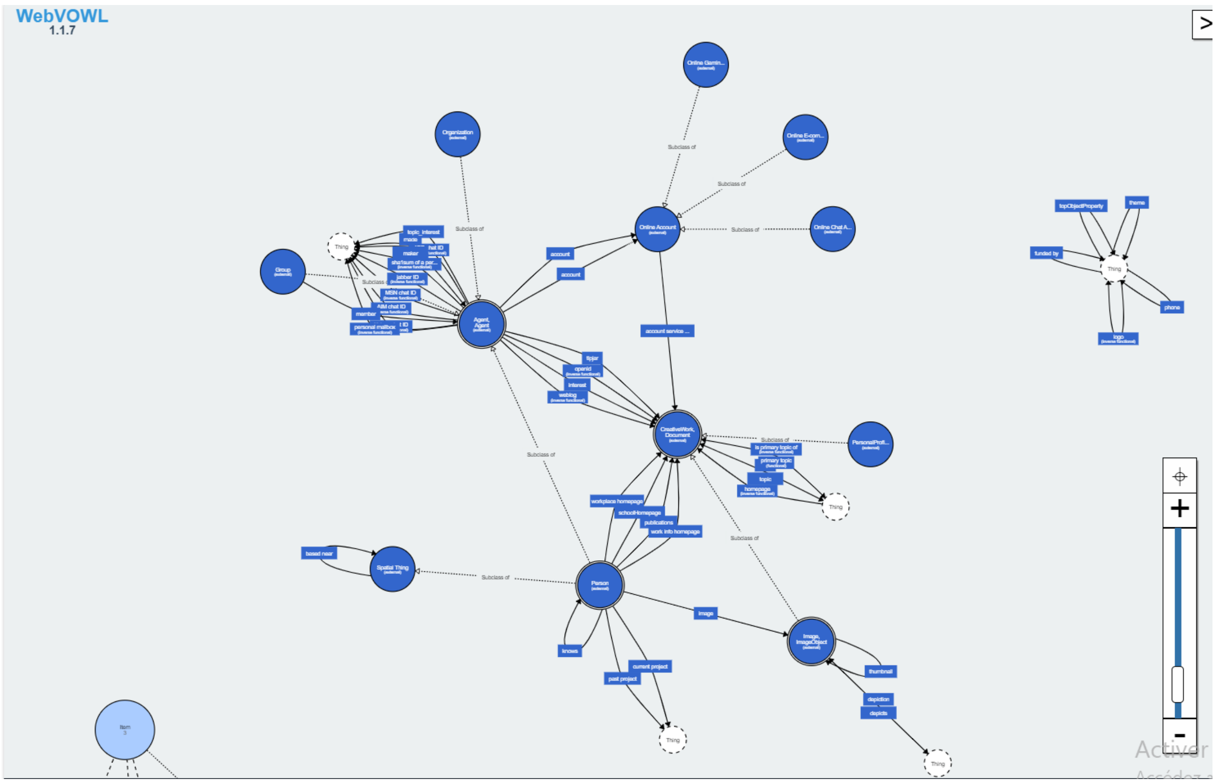


Figure 21: RDF Graph Associated with Our Ontology

Generated on the site: https://service.tib.eu/webvowl, it allows navigation between the different concepts, viewing.

5.3 Ontology Validation

After adding rules, we use **Hermit** reasoner, to infer class hierarchy and properties, to make sure nothing is nonsense.

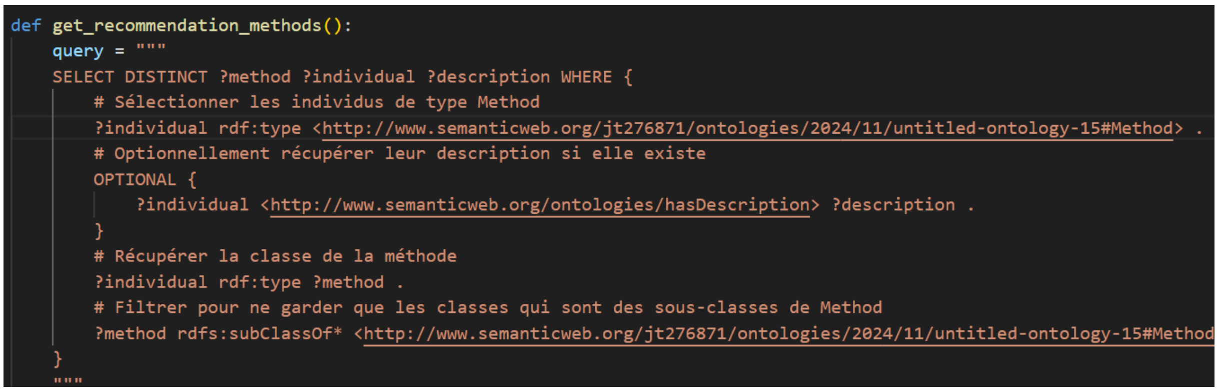
5.4 Queries

To assess the relevance and usefulness of our ontology, we can use SPARQL queries, a query language designed to retrieve data (or more specifically, triples) stored in RDF graphs.

The previously defined competency questions will guide these queries to verify if our ontology satisfactorily meets the specific needs we aim to address (in this case, the development of a recommendation system for fashion).

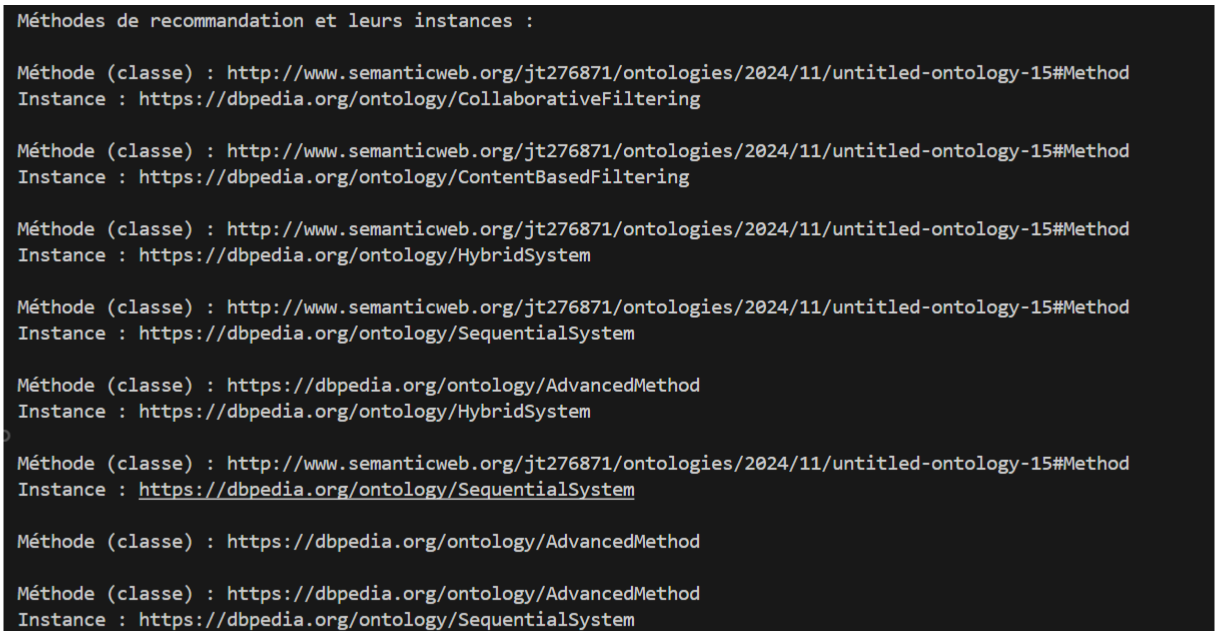
If the responses to the queries correctly reflect the expectations set in terms of competencies, it will confirm that the goal of the ontology has been achieved. Therefore, SPARQL queries serve as an excellent and powerful tool for evaluating, validating,

and optimizing the ontology, ensuring its quality and suitability for the development of the system.
First of all,We need to define a query about our recommendation systems:



```
def get_recommendation_methods():
    query = """
    SELECT DISTINCT ?method ?individual ?description WHERE {
        # Sélectionner les individus de type Method
        ?individual rdf:type <http://www.semanticweb.org/jt276871/ontologies/2024/11/untitled-ontology-15#Method> .
        # Optionnellement récupérer leur description si elle existe
        OPTIONAL {
            ?individual <http://www.semanticweb.org/ontologies/hasDescription> ?description .
        }
        # Récupérer la classe de la méthode
        ?individual rdf:type ?method .
        # Filtrer pour ne garder que les classes qui sont des sous-classes de Method
        ?method rdfs:subClassOf* <http://www.semanticweb.org/jt276871/ontologies/2024/11/untitled-ontology-15#Method
    }
    """
```

Figure 22: Enumerating our Recommendation Systems

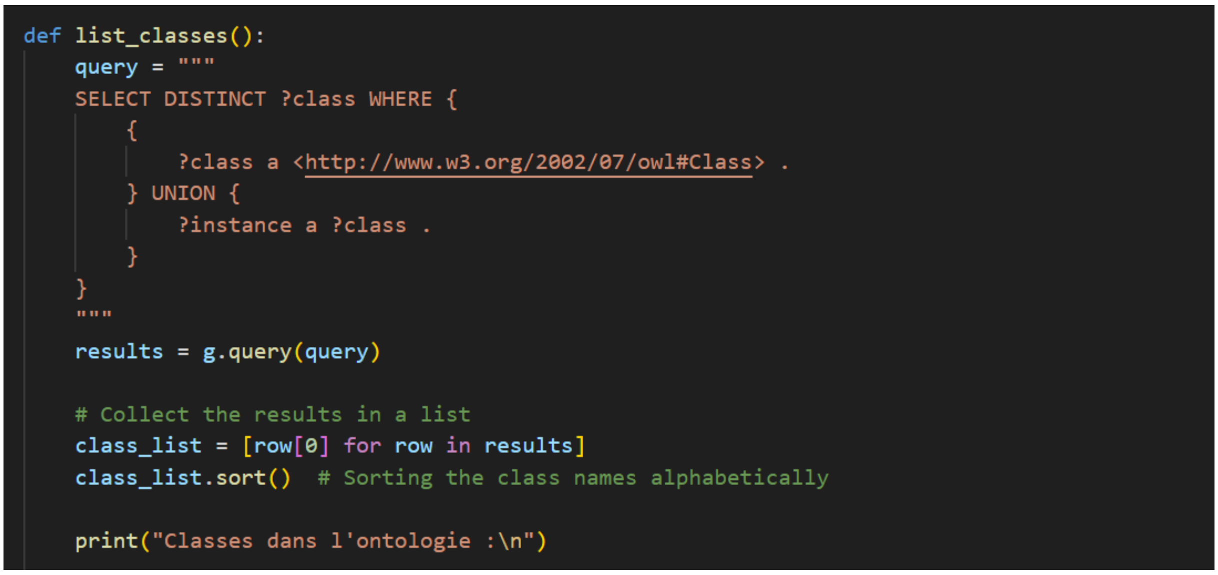


```
Méthodes de recommandation et leurs instances :

Méthode (classe) : http://www.semanticweb.org/jt276871/ontologies/2024/11/untitled-ontology-15#Method
Instance : https://dbpedia.org/ontology/CollaborativeFiltering

Méthode (classe) : http://www.semanticweb.org/jt276871/ontologies/2024/11/untitled-ontology-15#Method
Instance : https://dbpedia.org/ontology/ContentBasedFiltering

Méthode (classe) : http://www.semanticweb.org/jt276871/ontologies/2024/11/untitled-ontology-15#Method
Instance : https://dbpedia.org/ontology/HybridSystem

Méthode (classe) : http://www.semanticweb.org/jt276871/ontologies/2024/11/untitled-ontology-15#Method
Instance : https://dbpedia.org/ontology/SequentialSystem

Méthode (classe) : https://dbpedia.org/ontology/AdvancedMethod
Instance : https://dbpedia.org/ontology/HybridSystem

Méthode (classe) : http://www.semanticweb.org/jt276871/ontologies/2024/11/untitled-ontology-15#Method
Instance : https://dbpedia.org/ontology/SequentialSystem

Méthode (classe) : https://dbpedia.org/ontology/AdvancedMethod

Méthode (classe) : https://dbpedia.org/ontology/AdvancedMethod
Instance : https://dbpedia.org/ontology/SequentialSystem
```

Figure 23: Result of the query

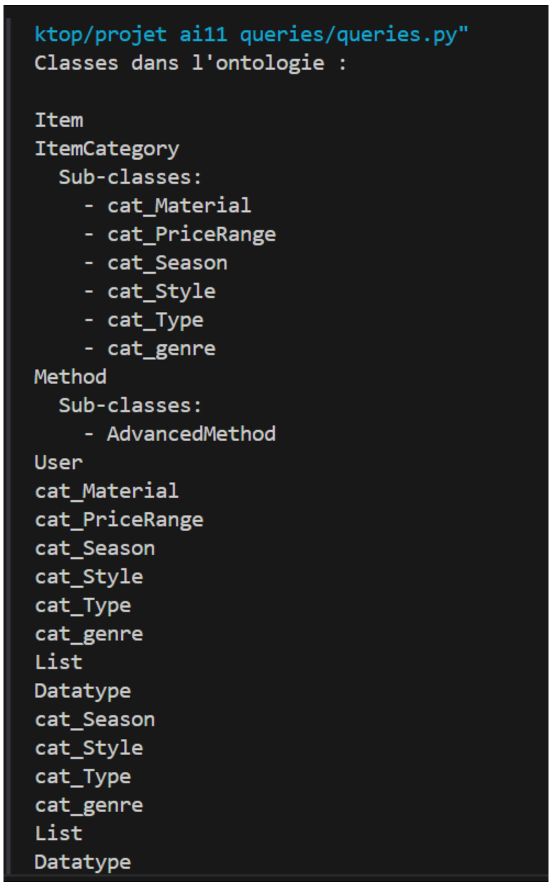
We will first present some general queries that we have implemented, followed by a deeper exploration of the specific queries related to each method.

- list_classes()



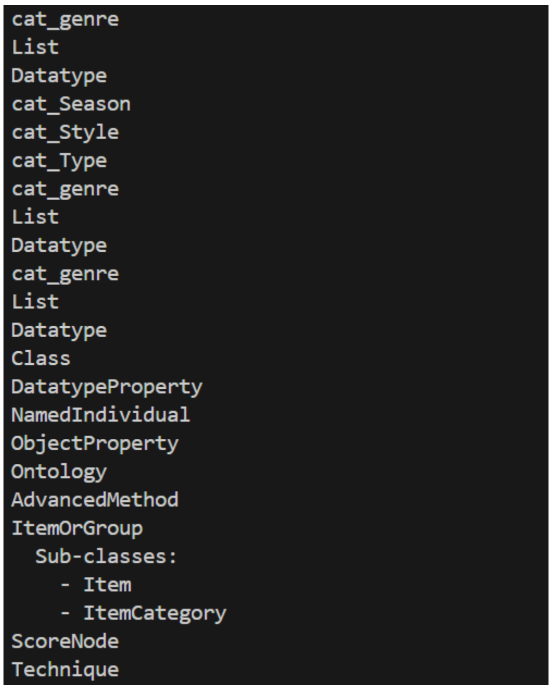
```python
def list_classes():
    query = """
    SELECT DISTINCT ?class WHERE {
        {
            ?class a <http://www.w3.org/2002/07/owl#Class> .
        } UNION {
            ?instance a ?class .
        }
    }
    """
    results = g.query(query)

    # Collect the results in a list
    class_list = [row[0] for row in results]
    class_list.sort()  # Sorting the class names alphabetically

    print("Classes dans l'ontologie :\n")
```

Figure 24: Enumerating our Recommendation Systems



```
ktop/projet ai11 queries/queries.py"
Classes dans l'ontologie :

Item
ItemCategory
  Sub-classes:
    - cat_Material
    - cat_PriceRange
    - cat_Season
    - cat_Style
    - cat_Type
    - cat_genre
Method
  Sub-classes:
    - AdvancedMethod
User
cat_Material
cat_PriceRange
cat_Season
cat_Style
cat_Type
cat_genre
List
Datatype
cat_Season
cat_Style
cat_Type
cat_genre
List
Datatype
```

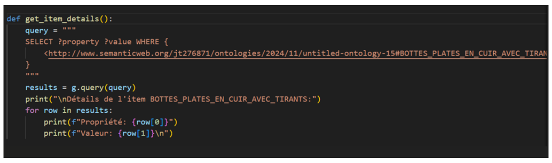
Figure 25: Ontolgy calsses



```
cat_genre
List
Datatype
cat_Season
cat_Style
cat_Type
cat_genre
List
Datatype
cat_genre
List
Datatype
Class
DatatypeProperty
NamedIndividual
ObjectProperty
Ontology
AdvancedMethod
ItemOrGroup
  Sub-classes:
    - Item
    - ItemCategory
ScoreNode
Technique
```

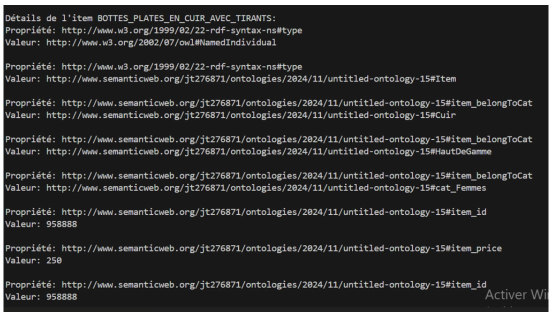
Figure 26: Ontology calsses

- get_item_details()



```
def get_item_details():
    query = """
    SELECT ?property ?value WHERE {
        <http://www.semanticweb.org/jt276871/ontologies/2024/11/untitled-ontology-15#BOTTES_PLATES_EN_CUIR_AVEC_TIRAN
    }
    """
    results = g.query(query)
    print("\nDétails de l'item BOTTES_PLATES_EN_CUIR_AVEC_TIRANTS:")
    for row in results:
        print(f"Propriété: {row[0]}")
        print(f"Valeur: {row[1]}\n")
```

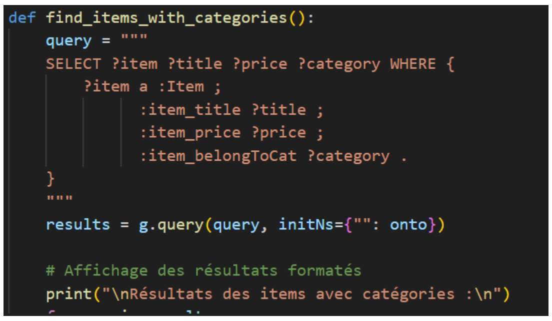
Figure 27: item details query



```
Détails de l'item BOTTES_PLATES_EN_CUIR_AVEC_TIRANTS:
Propriété: http://www.w3.org/1999/02/22-rdf-syntax-ns#type
Valeur: http://www.w3.org/2002/07/owl#NamedIndividual

Propriété: http://www.w3.org/1999/02/22-rdf-syntax-ns#type
Valeur: http://www.semanticweb.org/jt276871/ontologies/2024/11/untitled-ontology-15#Item

Propriété: http://www.semanticweb.org/jt276871/ontologies/2024/11/untitled-ontology-15#item_belongToCat
Valeur: http://www.semanticweb.org/jt276871/ontologies/2024/11/untitled-ontology-15#Cuir

Propriété: http://www.semanticweb.org/jt276871/ontologies/2024/11/untitled-ontology-15#item_belongToCat
Valeur: http://www.semanticweb.org/jt276871/ontologies/2024/11/untitled-ontology-15#HautDeGamme

Propriété: http://www.semanticweb.org/jt276871/ontologies/2024/11/untitled-ontology-15#item_belongToCat
Valeur: http://www.semanticweb.org/jt276871/ontologies/2024/11/untitled-ontology-15#cat_Femmes

Propriété: http://www.semanticweb.org/jt276871/ontologies/2024/11/untitled-ontology-15#item_id
Valeur: 958888

Propriété: http://www.semanticweb.org/jt276871/ontologies/2024/11/untitled-ontology-15#item_price
Valeur: 250

Propriété: http://www.semanticweb.org/jt276871/ontologies/2024/11/untitled-ontology-15#item_id
Valeur: 958888
```

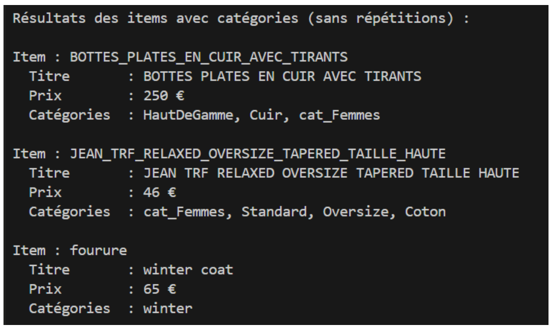
Figure 28: item details result

- find_item_withCategories()



```
def find_items_with_categories():
    query = """
    SELECT ?item ?title ?price ?category WHERE {
        ?item a :Item ;
              :item_title ?title ;
              :item_price ?price ;
              :item_belongToCat ?category .
    }
    """
    results = g.query(query, initNs={"": onto})

    # Affichage des résultats formatés
    print("\nRésultats des items avec catégories :\n")
```

Figure 29: item with category



```
Résultats des items avec catégories (sans répétitions) :

Item : BOTTES_PLATES_EN_CUIR_AVEC_TIRANTS
  Titre       : BOTTES PLATES EN CUIR AVEC TIRANTS
  Prix        : 250 €
  Catégories  : HautDeGamme, Cuir, cat_Femmes

Item : JEAN_TRF_RELAXED_OVERSIZE_TAPERED_TAILLE_HAUTE
  Titre       : JEAN TRF RELAXED OVERSIZE TAPERED TAILLE HAUTE
  Prix        : 46 €
  Catégories  : cat_Femmes, Standard, Oversize, Coton

Item : fourure
  Titre       : winter coat
  Prix        : 65 €
  Catégories  : winter
```

Figure 30: item with category

Content-based-filtring

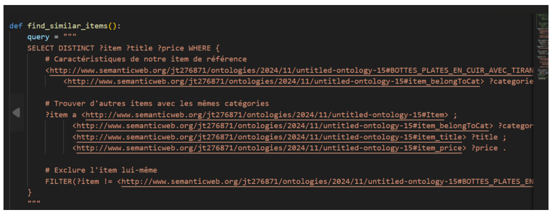
(a) Recommend products to a user based on their preferences.

(b) Find similar products based on characteristics (e.g., color, size).

(c) Find users with similar preferences to enhance recommendations.

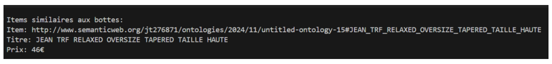
Examples of the queries for this method:

- find_items_for_user()
- find_similar_items()
- find_users_with_similar_preferences()



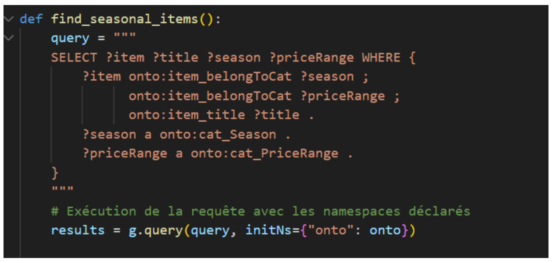
```
def find_similar_items():
    query = """
    SELECT DISTINCT ?item ?title ?price WHERE {
        # Caractéristiques de notre item de référence
        <http://www.semanticweb.org/jt276871/ontologies/2024/11/untitled-ontology-15#BOTTES_PLATES_EN_CUIR_AVEC_TIRAN
            <http://www.semanticweb.org/jt276871/ontologies/2024/11/untitled-ontology-15#item_belongToCat> ?categorie

        # Trouver d'autres items avec les mêmes catégories
        ?item a <http://www.semanticweb.org/jt276871/ontologies/2024/11/untitled-ontology-15#Item> ;
              <http://www.semanticweb.org/jt276871/ontologies/2024/11/untitled-ontology-15#item_belongToCat> ?categor
              <http://www.semanticweb.org/jt276871/ontologies/2024/11/untitled-ontology-15#item_title> ?title ;
              <http://www.semanticweb.org/jt276871/ontologies/2024/11/untitled-ontology-15#item_price> ?price .

        # Exclure l'item lui-même
        FILTER(?item != <http://www.semanticweb.org/jt276871/ontologies/2024/11/untitled-ontology-15#BOTTES_PLATES_EN
    }
    """
```

Figure 31: Similar items that belongs to the same category



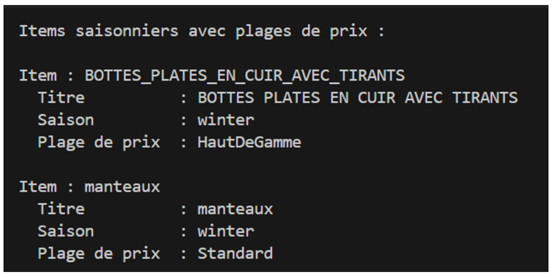
```
Items similaires aux bottes:
Item: http://www.semanticweb.org/jt276871/ontologies/2024/11/untitled-ontology-15#JEAN_TRF_RELAXED_OVERSIZE_TAPERED_TAILLE_HAUTE
Titre: JEAN TRF RELAXED OVERSIZE TAPERED TAILLE HAUTE
Prix: 46€
```

Figure 32: Similar items that belongs to the same category



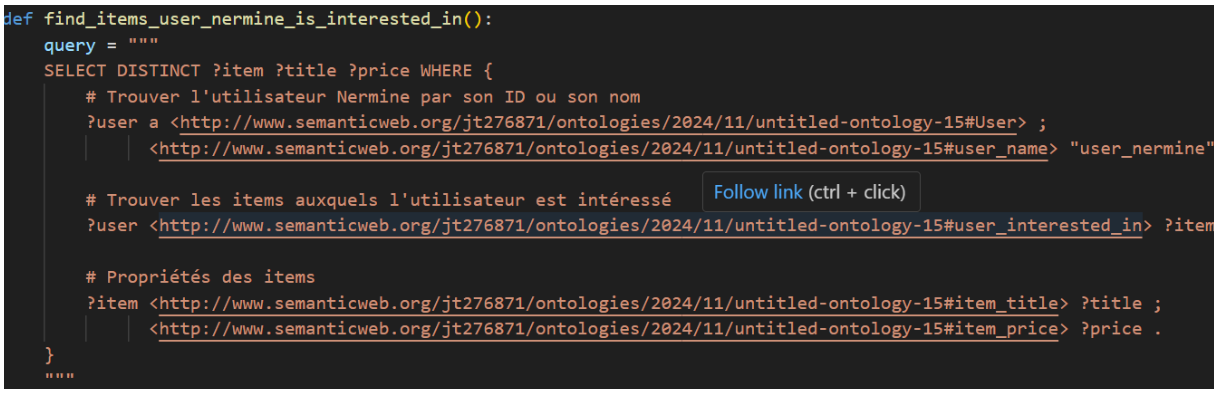
```
def find_seasonal_items():
    query = """
    SELECT ?item ?title ?season ?priceRange WHERE {
        ?item onto:item_belongToCat ?season ;
              onto:item_belongToCat ?priceRange ;
              onto:item_title ?title .
        ?season a onto:cat_Season .
        ?priceRange a onto:cat_PriceRange .
    }
    """
    # Exécution de la requête avec les namespaces déclarés
    results = g.query(query, initNs={"onto": onto})
```

Figure 33: Similar items that belongs to the same season



```
Items saisonniers avec plages de prix :

Item : BOTTES_PLATES_EN_CUIR_AVEC_TIRANTS
  Titre          : BOTTES PLATES EN CUIR AVEC TIRANTS
  Saison         : winter
  Plage de prix  : HautDeGamme

Item : manteaux
  Titre          : manteaux
  Saison         : winter
  Plage de prix  : Standard
```

Figure 34: similar item that belong to same season



```
def find_items_user_nermine_is_interested_in():
    query = """
    SELECT DISTINCT ?item ?title ?price WHERE {
        # Trouver l'utilisateur Nermine par son ID ou son nom
        ?user a <http://www.semanticweb.org/jt276871/ontologies/2024/11/untitled-ontology-15#User> ;
              <http://www.semanticweb.org/jt276871/ontologies/2024/11/untitled-ontology-15#user_name> "user_nermine"

        # Trouver les items auxquels l'utilisateur est intéressé
        ?user <http://www.semanticweb.org/jt276871/ontologies/2024/11/untitled-ontology-15#user_interested_in> ?item

        # Propriétés des items
        ?item <http://www.semanticweb.org/jt276871/ontologies/2024/11/untitled-ontology-15#item_title> ?title ;
              <http://www.semanticweb.org/jt276871/ontologies/2024/11/untitled-ontology-15#item_price> ?price .
    }
    """
```

Figure 35: user preference query

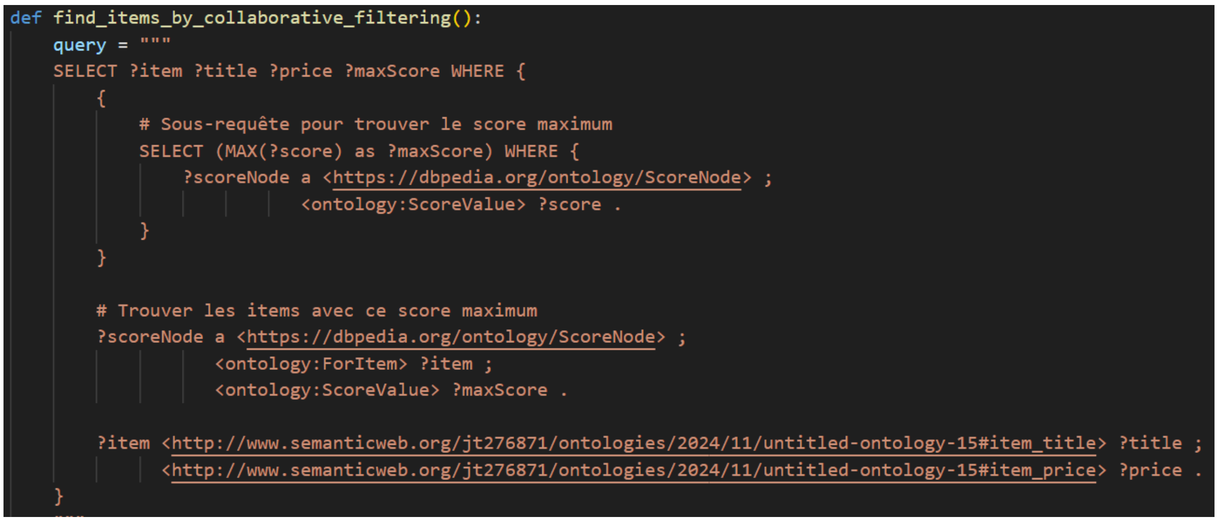
Collabrotive filtring

(a) Find users with similar ratings for a given product.

(b) Recommend products to a user based on ratings of similar products made by users with similar tastes.

(c) Analyze similarities between users to create groups of similar users.

- find_users_with_similar_ratings()
- find_items_by_collaborative_filtering()



```
def find_items_by_collaborative_filtering():
    query = """
    SELECT ?item ?title ?price ?maxScore WHERE {
        {
            # Sous-requête pour trouver le score maximum
            SELECT (MAX(?score) as ?maxScore) WHERE {
                ?scoreNode a <https://dbpedia.org/ontology/ScoreNode> ;
                           <ontology:ScoreValue> ?score .
            }
        }

        # Trouver les items avec ce score maximum
        ?scoreNode a <https://dbpedia.org/ontology/ScoreNode> ;
                   <ontology:ForItem> ?item ;
                   <ontology:ScoreValue> ?maxScore .

        ?item <http://www.semanticweb.org/jt276871/ontologies/2024/11/untitled-ontology-15#item_title> ?title ;
              <http://www.semanticweb.org/jt276871/ontologies/2024/11/untitled-ontology-15#item_price> ?price .
    }
```

Figure 36: find items by collaborative filtring query

- find_similar_users()

sequential recommendation:

(a) Recommend products based on a user's history.

(b) Find users with similar interactions.

(c) Recommend new products based on the history of interactions.

hybrid system

(a) Recommend products by combining Collaborative Filtering and Content-Based Filtering.

(b) Recommend products by combining Collaborative Filtering and Sequential Recommendation.

(c) Recommend products by combining all techniques (Content-Based, Collaborative, and Sequential) in a hybrid system.

6 Key words

A-General Key Terms for Recommendation Systems

- **Recommendation System**: A system designed to recommend items based on user preferences, behaviors, or other criteria.
- **Collaborative Filtering**: A recommendation technique based on user similarity and interactions with items.
- **Content-Based Filtering**: A recommendation technique that suggests items based on the features of items that a user has interacted with or liked in the past.
- **Sequential Recommendation**: A recommendation method that takes into account the sequence of user interactions or behaviors over time.
- **Hybrid Recommendation**: A recommendation system that combines multiple recommendation techniques (e.g., collaborative filtering, content-based filtering, sequential recommendation) to provide more accurate and personalized suggestions.

B-Key Terms Related to the Ontology Structure

- **Ontology**: A formal representation of knowledge as a set of concepts and the relationships between them, used to model information in a structured way.
- **Class**: A category of things in the ontology. For example, User, Product, Rating, etc.
- **Object Property**: A property that links an individual to another individual in the ontology. For example, hasRating, hasFeature, isRatedBy.
- **Datatype Property**: A property that links an individual to a data value (e.g., a score or date). For example, hasScore.
- **Instance (Individual)**: Specific examples or members of a class in the ontology. For example, UserA, Product1.
- **SubClass**: A class that is a more specialized version of a parent class. For example, CollaborativeFiltering is a subclass of RecommendationMethod.

C-Classes and Properties for Sequential Recommendation

- **SequentialRecommendation**: A class representing a method of recommendation based on the sequence of user interactions.
- **hasInteraction**: An object property linking a user to their interactions with products.

- **hasProduct**: An object property linking an interaction or a rating to a product.
- **hasSequence**: A property that could represent the sequence or order of user interactions.
- **basedOn**: An object property used to specify the recommendation method being used (e.g., sequential recommendation).

D-**Classes and Properties for Hybrid Recommendation**

- **HybridRecommendation**: A class representing the hybrid recommendation system that combines multiple methods.
- **hasFeature**: An object property linking a product to its features, used in content-based filtering.
- **recommendedFor**: An object property that links a recommendation to a specific user.
- **isSimilarTo**: An object property that indicates the similarity between users, useful for collaborative filtering.
- **hasRating**: An object property linking a user to a product rating.

6.1 Queries

7 Conclusion

8 bibliographic references

60 Link to Text Mining and NLP Application.

61 Link to Limitations of Word Frequency-based Analysis.

62 Link to Use of Ontology in Text Mining.

67 Link to KNN in Recommendation Systems.

69 Link to Clustering in Recommendation Systems.

70 Link to K-Means Clustering Algorithm.

81 Link to Neural Networks in Recommendation Systems.

82 Link to Research on Neural Networks for Recommendations.

83 Link to Deep Neural Networks for Modeling Implicit Feedback.

References

[1] In: ().

[2] Tesfaye Fenta Boka, Zhendong Niu, and Rama Bastola Neupane. "A survey of sequential recommendation systems: Techniques, evaluation, and future directions". In: *Information Systems* 125 (2024), p. 102427. ISSN: 0306-4379. DOI: https://doi.org/10.1016/j.is.2024.102427. URL: https://www.sciencedirect.com/science/article/pii/S0306437924000851.