Université de Paris

UFR Mathématiques et Informatique

Évaluation des connaissances (contrôle continu) Module « Ontologie & Web Sémantique »

Master 2 Vision Machine Intelligente

Sommaire

1. Présentation de Musicontolgy

MusicOntology (MO) fournit un vocabulaire permettant de publier et de relier un large éventail de données relatives à la musique sur le Web. Les données de l'ontologie musicale peuvent être publiées par n'importe qui dans le cadre d'un site web ou d'une API et liées à des données existantes, créant ainsi un réseau de données sur la musique.
MO peut être utilisée dans les cas suivant (liste non exhaustive):

- construire une API lorsque nous souhaitons réaliser une page internet en lien avec la musique
- réaliser une base de données en lien avec les différents domaines de la musique
- construire une moteur de recherche pour retrouver des informations sur un artiste par exemple, ce que nous allons faire pour notre application

MO est écrite en RDF, ce format permet de décrire les données sous formes de triplets, a savoir sujet prédicat et objet. Par exemple : «*Light my fire*» - «est un titre de» – «Stevie Wonder». MO utilise également le format *owl (Ontology Web Language*), qui est un langage de représentation des connaissances construit sur le modèle de données de RDF. Il fournit les moyens pour définir des ontologies web structurées. MO présente à ce jour 54 classes et 153 propriétés et a été créé en 2006 par les auteurs suivants : Yves Raimond, Thomas Gängler, Frédérick Giasson, Kurt Jacobson, George Fazekas, Simon Reinhardt, Alexandre Passant.
En raison de son nombre important de classes et de propriétés, il est impossible de faire une représentation visuelle. Cependant il est possible d'avoir accès à l'intégralité de ces informations sur ce lien [1]

La raison pour laquelle MO utilise RDF comme modèle de représentation de graphe est parce que il lui permet de résoudre le problème des nombreuses composantes nécessaires pour établir une ontologie complète du domaine musical en un seul format de données.
RDF permet alors à MO d'obtenir davantage, de flexibilité , d'adaptabilité et d'extensibilité : l'ajout d'information issue de ressources externes dans MO sera facilité tant que nous connaissons l'URI de cette dite ressource. RDF étant un modèle de données sous forme de graphe, il ne présente pas véritablement de syntaxe: il existe plusieurs syntaxes possibles pour représenter une description RDF. La plus connue est la syntaxe RDF/XML qui, utilisant le méta-langage XML, permet de créer des descriptions facilement manipulables par la machine.

MO est donc le résultat de plusieurs ontologies assemblées , il existe 5 grandes catégories

- *FOAF, friend of a friend* : un vocabulaire destiné à décrire une personne, un groupe ou encore une entreprise
- L'ontologie FRBR, fonctionnalités requises des notices bibliographiques ,*Functional Requirements for Bibliographic Records*: vocabulaire destiné à fournir des informations contenues dans les notices bibliographiques des bibliothèques.

- L'ontologie de caractéristiques, *features ontology* : vocabulaire destiné à créer un framework générique pour décrire les caractéristiques d'un signal audio.
- L'ontologie d'événement , *event ontology* : un vocabulaire destiné à décrire des événements du passé ou du futur
- l'Ontologie chronologique/temporelle, *timeline ontology* : un vocabulaire destiné à décrire une intervalle temporelle

Nous allons dans la partie suivante, mieux présenter les deux dernières catégories mentionnées plus haut.

a. L'ontologie d'évènement(OE)

La carrière musicale d'un artiste inclut la publication de single, d'album ou encore de concerts et tournées. Il est donc nécessaire de préciser la date et les lieux de ces événements physiques. La représentation d'un *Event*, événement dans MO est basée sur une approche de type token-réification [2] : l'occurrence de l'*event* agit comme un crochet pour des informations supplémentaires relatives à un événement.

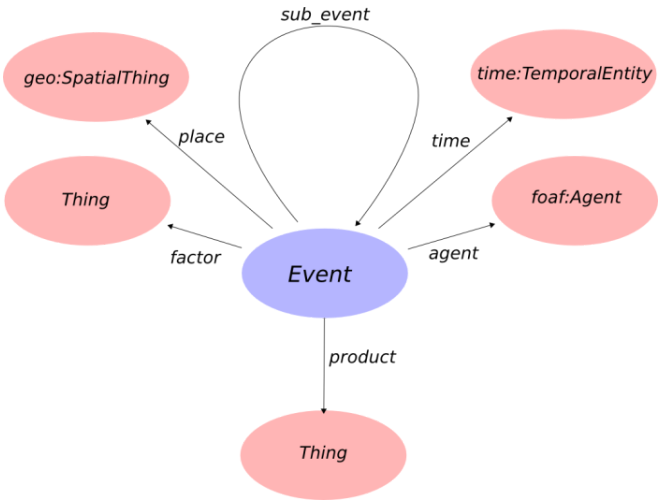

Figure 1 : Vue d'ensemble de l'OE

Le concept d'événement présente alors un certain nombre de facteurs (comme un instrument de musique e.g), d'agents (un interprète particulier, un musicien) et de produits (comme le son physique qu'une performance produit). Ce concept peut être lié à un lieu particulier par le prédicat *event:place* (reliant l'ontologie Event à l'ontologie Geonames) et à un temps particulier par le prédicat *event:time,* reliant l'ontologie *Event* à l'ontologie *Timeline* que nous allons présenter dans la partie suivante.

b. L'ontologie temporelle (OT)

L'information temporelle est la première chose que nous voulons exprimer lorsque nous traitons des connaissances liées à la musique. Il est nécessaire de couvrir plusieurs descriptions temporelles tout le long de la carrière d'un artiste : «Madonna a réalisé un concert à Paris le 30

mars 1990», «cette mélodie apparaît sur l'échantillon 8585». MO est donc construite au-dessus d'une ontologie capable d'exprimer de telles informations temporelles : l'ontologie ***Timeline*** [3]. Cette ontologie est elle-même construite au-dessus de deux concepts définis dans *OWL-Time* [4] : *Intervaland Instant*, représentant respectivement les intervalles de temps et les instants.

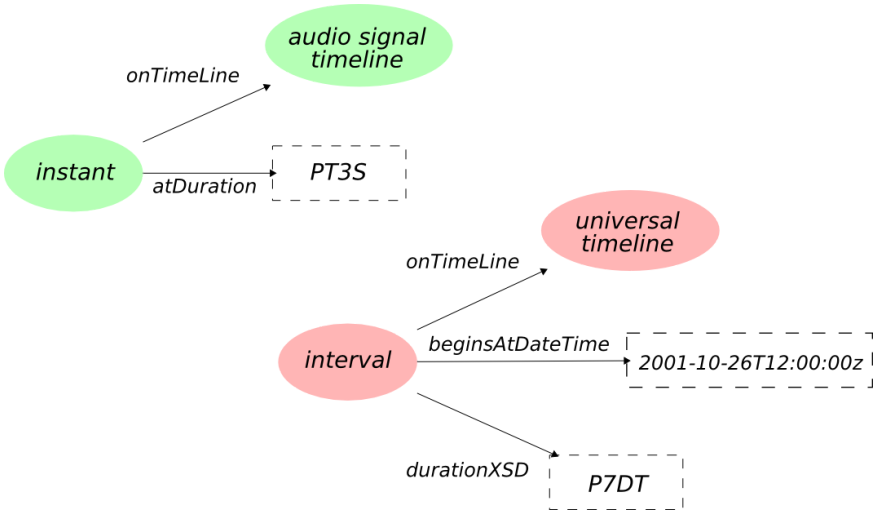

Figure 2 : Vue d'ensemble d'OT

OT définit un autre concept, *TimeLine*, représentant une colonne vertébrale cohérente pour traiter les informations temporelles. Par exemple, nous pouvons exprimer la date de sortie d'un disque particulier à l'aide de la ligne temporelle physique, et des informations telles que "de 0 à 3 secondes sur cette piste particulière" à l'aide de la ligne temporelle d'un signal audio.

2.Création de notre ontologie musicale(OM)

Dans le cadre de notre projet , il nous a été demandé de réaliser une ontologie avec les fonctionnalités suivantes :

- Permettre de représenter les artistes en trois catégories: auteur, compositeur et interprète
- Modéliser qu'il y ait des groupes , des chansons, des albums mais aussi des chansons dans des albums
- Représenter le genre musical : country, rap, variété française etc
- Créer les concepts adéquats, les relations nécessaires, leurs domaines/sous-domaines et leurs inverses.

Nous allons aborder ces trois points dans la partie les parties suivante

a. Représentation des artistes :

Classe :
Dans cette partie de notre OM, nous avons réalisé une classe Artiste qui est composé de 3 sous classe :

Figure 3 : modélisation d'un artiste sur Protégé

Un artiste peut alors présenter plusieurs fonctions , ou propriétés :

- être le compositeur d'une chanson
- être l'auteur d'une chanson
- être l'interprète d'une chanson

L'inverse également à modéliser , par exemple "Une chanson a pour [sous classe d'artiste]"
Ceci a été faite dans la partie Object Properties

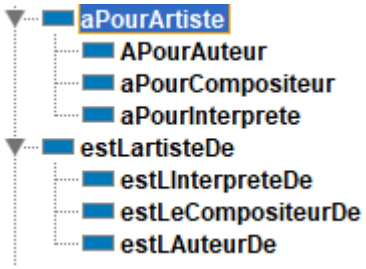

Figure 4 : Quelques propriété établies pour un artiste

b. Modélisation d'un groupe

Il nous a été demandé de construire dans notre OM une catégorie groupe. Nous avons décidé d'en faire une classe , intitulée Groupe. Il doit être composé d'au minimum de deux artistes. Nous avons établi deux relations possible dans ce cas de figure :

- un groupe est composé de membres qui sont des artistes : **aPourMembreDe**
  - son inverse est donc **estLeMembreDe**
- des membres composent un groupe : **aPourGroupe**
  - son inverse est donc **estLeGroupeDe**

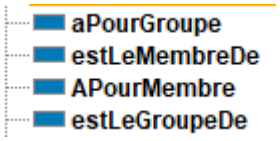

Figure 6 : Quelques propriétés pour la classe Groupe

Un groupe présente également d'autres propriétés telles qu'une année de formation , un nom de groupe etc.

c. Modélisation d'une chanson

La chanson représente l'élément clef de notre OM, nous lui avons donc dédié une classe. En effet , il constitue le produit final de notre ontologie, dans MO que nous avons expliqué en détail en tout début de notre rapport , il existe davantage de sous classes et de propriétés concernant cet élément : une chanson peut être un remix , elle peut être composée de plusieurs pistes ou l'inverse contenue dans des pistes différentes etc. Dans le cadre de ce projet nous nous sommes contentés de modéliser une chanson dans 3 cas de figure différents : lorsqu'elle compose un album , lorsqu'elle est produite par un artiste et son genre musical. Nous avons décidé de ne pas attribuer de genre musical à un album car il peut être composé de chansons de genres différents. Donc parmis les propriétés que nous avons listé pour la classe, certaines concernent la classe Chanson également , nous avons ajouté celles relatives à la composition d'un album et du genre musical :

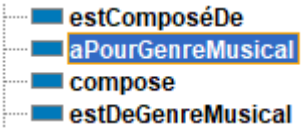

Figure 7 : Quelques propriétés de la classe chanson

Voici les données de propriétés que nous avons défini :

Figure 8 : Propriétés de données de la classe Chanson

d. Modélisation d'Album et des certifications discographiques

Dans cette partie nous allons présenter comment nous avons modélisé la classe Album. Il nous a été demandé de pouvoir préciser que lorsqu'un album atteint un certain nombre de ventes, il fallait préciser la certification discographiques qu'il pouvait atteindre : or ; platine, diamant etc. Ces certifications sont définies en tant que sous classe d'Album :

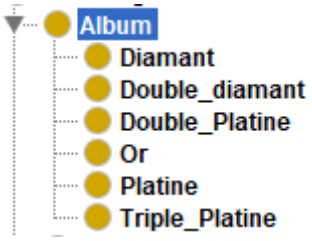

Figure 9 : Classe Album et ses sous classes

Voici les seuils à atteindre pour obtenir ces certifications (critères d'attribution français) :

- Double diamant : 1 million de ventes
- Diamant : 500 000 ventes
- Double platine : 200 000 ventes
- Platine : 100 000 ventes
- Or : 50 000 ventes

Il faut donc respecter ces conditions de ventes pour qu'un album soit catégorisé selon les sous classes établies .

Dans notre ontologie , un album présenté les propriétés de données suivantes :

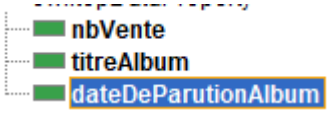

Figure 10 : Propriétés d'objets d'un album

e. Modélisation du genre musical

Lorsque nous avons décidé de modéliser notre OM nous pensions qu'utiliser uniquement une classe pour le genre musical serait suffisant, mais en utilisant le raisonneur tout le long de la phase débogage, il était nécessaire de rajouter une sous classe. Par ailleurs , sur le lien fourni dans le sujet , nous nous sommes rendus compte qu'il existait de grande catégorie de genre musicaux ie la *Retro soul* qui est une sous catégorie de la *Soul*.

3.Test de notre ontologie

a. Création d'une instance Album

Il nous a été demandé la consigne suivante :

«*Y a-t-il moyen de faire rentrer le nombre de ventes de disques comme un chiffre qui caractérise le disque ? Comment ? Peut-on modéliser le fait qu'un disque qui se vend à plus de 150 000 exemplaires, est disque d'or ? Modéliser les différentes certifications de disques possibles*»

Comme nous l'avons mentionné dans la partie modélisation d'un album et certification discographique, nous avons créé des classes qui peuvent répertorier selon le nombre de ventes

d'un album. Nous avons décidé de prendre comme exemple l'album de Stevie Wonder intitulé *Songs in the Key Of Life*, sorti en 1977. L'album aborde plusieurs genres musicaux : soul, rhythm and blues, pop et même jazz-rock Il est composé initialement de 5 titres :

- 1 - "I Wish",
  - apparu en décembre 1976, de genre Funk, d'une durée de 3mn55
- 2 - "Isn't She Lovely"
  - apparu en février 1977, de genre Smooth Soul, d'une durée de 6mn33
- 3 - "Sir Duke"
  - apparu en mars 1977, de genre Funk, d'une durée de 3mn52
- 4 - "Another Star"
  - apparu en août 1977, de genre Disco, d'une durée de 8mn28
- 5 - "As"
  - apparu en octobre 1977, de genre Soul, d'une durée de 7mn08

Concernant le nombre exact de ventes d'album nous n'avons pas pu le trouver. Toutefois, puisque l'album a été certifié Diamant aux Etats Unis (qui exige au minimum 10 millions de vente) , nous allons mettre cette valeur pour la propriété **nbVente**. Cet album est donc une instance de la sous classe **DoubleDiaman**t de notre OM.

Voici la liste des instances que nous avons fait pour cet exemple. Nous avons vérifié de façon régulière notre OM avec le raisonneur , nous avons obtenu aucune incohérence.

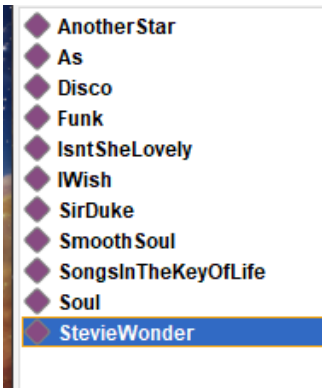

Figure 11 : Instances nécessaire pour modéliser l'album de Songs In the Key Of Life

Nous obtenons au final cet instance suivante :

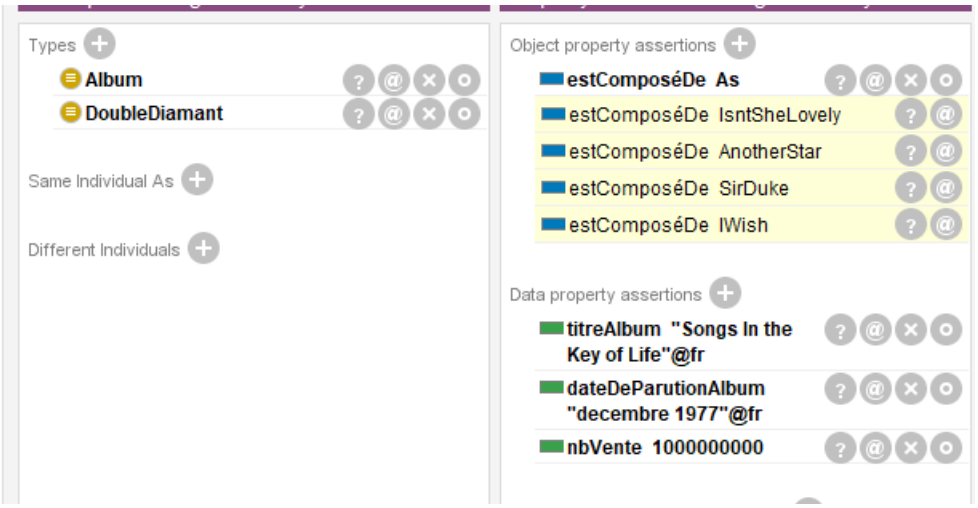

Figure 12 : Propriétés de l'instance Songs in The Key of Life

b. Création d'un groupe :

Nous avons pris comme exemple le groupe français Daft Punk composé de deux artistes : Thomas Bangalter et Guy-Manuel de Homem-Christo. Nous obtenons l'instance suivante

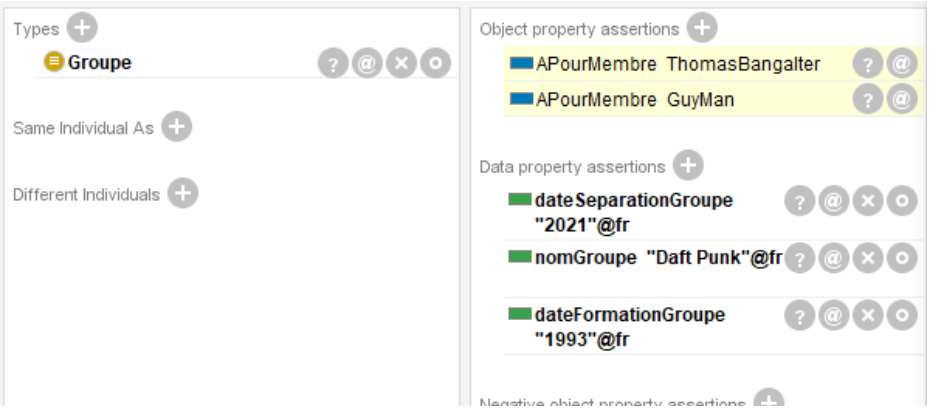

Figure 13 : Création de l'instance DaftPunk

4. Proposition d'architecture pour répertorier des disques de ventes :

Nous proposons l'architecture suivante :

➔ Utiliser une base de données pour stocker des informations sur les disques à vendre, y compris les attributs définis dans l'ontologie tels que l'artiste, le genre et le format, la certification ie.

- ➔ Utiliser un langage pour le backend tel que PHP ou encore javascript pour récupérer et filtrer les informations sur les disques à partir de la base de données en fonction des requêtes de recherche des utilisateurs.
- ➔ Se servir d'un framework pour le front-end tel que React ou Flutter pour construire l'interface utilisateur du site Web (il existe des formulaires prêts à être utilisés pour la recherche et l'affichage des résultats.
- ➔ Implémenter des fonctions pour interroger interroger avec une API la base de données et récupérer les disques pertinents en fonction de la saisie de l'utilisateur.
- ➔ Donner la possibilité aux utilisateurs de sauvegarder leurs recherches et de recevoir des notifications lors de nouvelles parutions de chansons ou d'album, ou s'ils ont obtenu de nouveaux certifications
- ➔ Utiliser le protocole HTTPS pour sécuriser les communications et le traitement des informations des utilisateurs.
- ➔ Sauvegarder et sécuriser régulièrement la base de données pour vous protéger contre la perte de données.

5. Application pour notre OM (cahier des charges)

Nous avons développé notre application à l'aide du langage de programmation Python. Nous avons utilisé la bibliothèque « rdflib »,ceci nous a permis de connecter notre ontologie à l'application. Concernant l'interface graphique, le GUI, nous avons *PysimpleGui.*

L'intérêt de notre application est de faciliter l'utilisateur de recherche d'entité dans notre OM. Cette l'application, consiste alors à être un moteur de recherche, il présente 4 boutons.

- ➔ Chercher un artiste
- ➔ Chercher un album
- ➔ Chercher une chanson
- ➔ Chercher un groupe.

Chercher un artiste : Ce bouton permet à partir du nom d'un artiste ,soit avec prénom ou le nom (qui existe dans les instances de notre OM). Il affiche les propriétés de l'artiste en question.Dans le cas où l'artiste n'existe pas dans notre base de données,l'application affiche « aucun résultat » dans une partie de résultat.

Chercher un album: Dans cette partie, si on cherche le nom d'un album (qui existe dans les instances de l'ontologie), il montre le résultat et s'il ne trouve aucun résultat, il dit « aucun résultat » dans une partie de résultat.

Chercher une chansonne: Ce bouton permet de chercher une chanson . Quand le user cherche une chanson, le résultat affiche le nom d'artiste relative de cette chanson.

Chercher un groupe: c'est le même principe pour cette partie. Quand on cherche le nom d'un groupe, il nous montre toutes les informations par rapport au groupe, à savoir sa composition. Si le groupe n'est pas dans notre base de données, il affiche "Aucun résultat"

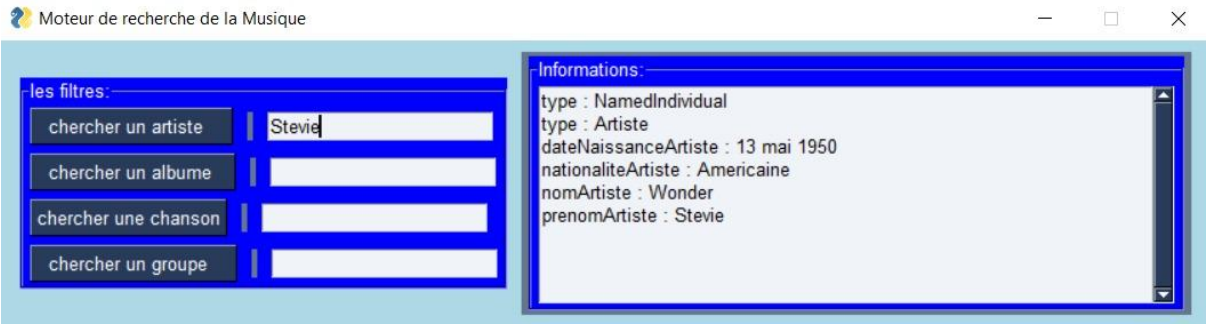

Figure 14 : GUI du moteur de recherche

6. Discussion, et avis sur notre ontologie

L'OM que nous avons réalisé respecte les fonctionnalités demandées par notre enseignant, toutefois elle présente quelques limites. En effet, lorsque nous comparons avec *MusicOntolgy*, nous présentons beaucoup de moins de classes et propriétés, par conséquent elle est beaucoup moins flexible et extensible.
Par exemple, il n'est pas possible d'indiquer les instruments utilisés lors de l'enregistrement de la chanson, ni d'introduire des informations sur le studio d'enregistrement. Pour ce faire, il faudrait rajouter d'avantages de propriétés et complexifierait plus notre projet.
Par ailleurs, la tâche qui a pris le plus de temps pour réaliser cette ontologie , fut la partie débogage avec le raisonneur. Sans celle-ci nous n'aurions pu obtenir d'OM cohérente.
Toutefois, retrouver les erreurs et les contradictions entre les règles et les propriétés peut être fastidieux et très long. Nous conseillons donc initialement d' utiliser le raisonneur sur OM vide d'instance. Une fois que nous obtenons plus d'erreurs, il faut instancier progressivement des objets: à ce moment-là nous affinons davantage notre ontologie. Dans notre cas nous avons découvert que des propriétés d'objets présentaient des relations inverses fausses, ce que nous avons corrigé à l'aide des explications fournies par le raisonneur.

6. Références :

1: Site web de MO : http://musicontology.com/specification/#sec-properties
2: M. P. Shanahan, "The event calculus explained," inArtificial Intelligence Today, Lecture Notes in AIno. 1600, M. J. Woolridge and M. Veloso, Eds.Springer, 1999, pp. 409–430
3: Y. Raimond and S. A. Abdallah, "The timelineontology," OWL-DL ontology, 2006. [Online].Available: http://purl.org/NET/c4dm/timeline.ow
4: J. R. Hobbs and F. Pan, "Time ontology in owl,"Working draft, September 2006. [Online]. Available:http://www.w3.org/TR/owl-time