Université Montpellier II

Facultés des sciences

MASTER 2 INFORMATIQUE

SPÉCIALITÉ À FINALITÉ PROFESSIONNELLE

&RECHERCHE UNIFIÉE

RAPPORT DE STAGE

effectué à

MDG Web (Seevl)

du **2 Avril** au **3 Septembre 2012**

par

JULIEN PLU

Directeur de stage de l'entreprise

ALEXANDRE PASSANT

Directeur de stage de l'Université

FRANÇOIS SCHARFFE

WEB SÉMANTIQUE, APPRENTISSAGE ET TALN POUR LE DOMAINE DES MÉTA-DONNÉES MUSICALES

REMERCIEMENTS

Je tiens avant tout à remercier les deux seules personnes qui m'ont vraiment poussé et soutenues dans le choix de ce stage, mon père et François Scharffe. Je veux aussi remercier mes deux employeurs, Alexandre Passant et Julie Letierce, ainsi que mes deux colocataires durant ce stage Simon Scerri et Lukasz Porwol, pour avoir fait en sorte que je me sente comme chez moi malgré le fait que j'étais loin de chez moi. Un énorme remerciement aux deux personnes du DERI qui m'ont beaucoup aidé et été présents quand j'avais des questions sur certaines de mes tâches, Richard Cyganiak et Brian Davis. Et bien entendu toutes les autres personnes que j'ai pu rencontrer durant cette formidable expérience.

RAPPORT

1. INTRODUCTION

1.1. Contexte du stage

La société MDG Web est une start-up issue du DERI (Digital Enterprise Research Insitute), laboratoire de recherche dans le domaine du Web Sémantique associé a l'Université de Galway. Le produit principal de la société est seevl, une extension pour navigateur Chrome qui ajoute des fonctionnalités de découverte musicales (recommandations, moteur de recherche, informations au sujet d'artistes) à YouTube. Pour ce faire, le système utilise un certain nombre de technologies Web Sémantique pour mettre en place son infrastructure et agréger des données musicales depuis des sources Web proposant des donnés ouvertes sur le domaine, comme Wikipedia, Musicbrainz ou et Freebase.

1.2. Objectifs détaillés du stage

L'objectif de ce stage se divise en trois tâches bien distinctes :

- Conversion de données : extraire certaines informations au sujet d'un artiste contenues dans la base de données de Musicbrainz et les exposer en RDF. RDF est le format de données utilisées par la base de donnés Seevl, qui les transmet ensuite au plug-in associé ;
- Extraction de connaissances liées aux récompenses : extraction d'entités dans une phrase correspondant à une catégorie Wikipedia d'un artiste, plus précisément une catégorie concernant une cérémonie qu'il a remportée ainsi que la catégorie si elle existe. Puis convertir ces informations en RDF via une ontologie spécialement réalisé à cet effet. Un papier scientifique pour le workshop DeRiVE (associé à la conférence ISWC2012) a été rédigé pour ce projet et accepté en temps qu'article de démonstration. Cet article est aussi en annexe 5.2 ;
- Extraction de connaissances liées à un artiste : extraction d'entités correspondant aux instruments, genres, locations, artistes et labels dans un texte concernant un artiste. Il s'agit ici de passer de données non structurées (texte brute) à des données RDF.

Bien que ces 3 taches soient distinctes, elles correspondent à un « socle technique » commun (RDF et donc Web sémantique) et s'intègrent à la même plate-forme, Seevl.

1.3. Planning prévisionnel

Le planning n'a pas été défini à l'avance, chaque tâche étant bien distincte. Les tâches ont été données au début du stage et l'organisation de ces tâches a été laissée à l'initiative de l'étudiant. Ainsi le travail fournit a été réparti dans l'ordre des tâches. La première tâche a été réalisée en premier, jusqu'à l'apparition des problèmes pour continuer sur la seconde tâche, une fois la seconde tâche terminée passée à la troisième. Beaucoup d'aller-retour ont été fait entre la première tâche et les deux autres. Mais la majorité du travail a été sur l'extraction d'entité dans un texte et des récompenses, qui étaient les tâches principales, l'autre étant là pour tester les possibilités des outils utilisés tout en améliorant le contenu de la base de connaissance de Seevl.

2. PROBLÈME – MÉTHODOLOGIE – OUTILS POUR CHACUNE DES TÂCHES

2.1. Conversion de données

2.1.1. Problématique

La base de connaissance de Seevl est dépourvue de certaines informations concernant les artistes musicaux. La base de connaissance contient pour le moment des informations sur les artistes eux-mêmes (biographies, influences, genres, etc.) mais rien au sujet des albums qu'ils ont enregistrée. La plupart de ces informations se trouvent dans diverses bases de données, dont l'une d'entre elles est Musicbrainz. MusicBrainz est la base de données de référence pour ce genre d'informations, avec plus de 10 000 000 d'œuvres musicales référencées et disponibles sous licence libre. Le format des données telles quelles sont dans cette base de données (PostgreSQL) n'est pas utilisables par le plugin. Il est donc important de convertir ces données dans le format de données utilisé par le plugin, RDF. Les informations contenues dans la base de données et qu'il fallait convertir sont :

- la liste des albums d'un artiste ;
- la liste des musiques d'un album ;
- la correspondance d'un code barre à un album ;
- la liste des tous les artistes qui ont joués ou repris la même musique ;
- la correspondance d'un code barre à un artiste.

2.1.2. Méthodologie adoptée et outils utilisés

Le but était donc de transformer ces données disponibles dans une base de données en RDF. Le langage R2RML[1] est utilisé pour réaliser cette tâche. R2RML (voir 5.1), encore en cours de standardisation par le W3C (au moment de l'écriture de ce rapport en Proposed Recommendation) est un langage de mapping permettant de convertir les données d'une base de données relationnelle en RDF. Le plugin Seevl utilise aussi Virtuoso comme magasin de triplets, l'équivalent d'une base de données mais pour RDF, ce format représentant les données sous forme de triplets (sujet – prédicat – objet) d'où le nom de magasin de triplets. Virtuoso permet d'utiliser le langage R2RML. Pour Musicbrainz, la base de données est au format PostgreSQL ; il a donc fallu créer une connexion entre Virtuoso et PostgreSQL afin que Virtuoso puisse utiliser les données de Musicbrainz contenues dans PostgreSQL et ainsi les exposer en RDF grâce à R2RML afin qu'elles soient utilisable par le plugin.

2.2. Extraction de connaissances liées aux récompenses

2.2.1. Problématique

La base de connaissance de Seevl contient des phrases en texte brut, associées aux artistes en tnt que categories (crées depuis les categories wikipedia) contenant beaucoup d'informations sur les artistes mais qui sont inexploitables. Comme « Best song MTV Music award winner », chacune de ces phrases est associée à un ou plusieurs artistes. Pour résoudre ce problème il faut donc extraire les informations importantes contenues dans ces phrases. Plus précisément faire en sorte que le plugin puisse répondre à une simple question, qu'elles sont les récompenses et les catégories qu'un artiste a remportées, en extrayant ainsi la catégorie et la récompense de ce genre de phrases.

[1] http://www.w3.org/TR/r2rml/

2.2.2. Méthodologie adoptée et outils utilisés

A ce jour, la base de connaissance de Seevl contient exactement 81 phrases concernant les récompenses associées à 2419 artistes. Tous les artistes de la base de connaissances n'ont pas une telle phrase associée. La tâche est donc d'effectuer une extraction des entités qui nous intéresse. L'extraction d'entité dans une phrase n'est jamais à 100% exacte. Il a donc fallu trouver une solution acceptable pour une mise en production de cette tâche. Le principal langage utilisé par le plugin est Python, il était donc important d'utiliser des outils utilisant ce langage, ainsi le framework NLTK (Natural Language Toolkit) a été utilisé pour le côté traitement du langage naturel. NLTK permet d'extraire des parties d'une phrase par une méthode appelée « chunking », cette méthode consiste à créer une ou plusieurs expressions régulières correspondant aux étiquettes des mots (adjectif, nom, nom personnel,…) ou suites de mots à extraire. Cette tâche ce déroule en deux parties, la première est l'extraction d'une cérémonie et la seconde est l'extraction d'une catégorie. Pour améliorer les résultats de cette extraction nous avons utilisé une API d'apprentissage, Google Prediction, qui a permis de choisir, statistiquement, si une extraction était juste ou non. Une fois ces données extraites, les faires correspondre à une ontologie développé spécialement pour décrire un évènement lié aux récompenses dans n'importe quel domaine.

2.3. Extraction de connaissances liées à un artiste

2.3.1. Problématique

Actuellement, la majeure partie des données sur Internet est sous forme de texte brute et donc non structurées, malgré des efforts comme Facebook OpenGraph ou schema.org. Dans cette forme non structurée, ces données sont très difficilement exploitables de manière automatique par une machine et donc les services comme Seevl se privent de cette masse de données. Cette tâche consiste donc à pouvoir extraire les données contenues dans des textes dédiés à la musique. Plus spécialement :

- les artistes du texte ;
- les localisations ;
- les instruments ;
- les genres de musiques ;
- les labels.

Il faut aussi extraire les relations entre ces entités contenues dans un texte, par exemple associer un artiste a son lieu de naissance ou de mort, et pas juste indiquer qu'il existe une relation entre un artiste et un lieu.

2.3.2. Méthodologie adoptée et outils utilisés

Pour cette tâche un seul outil est utilisé, GATE, qui est un framework Java pour le traitement de la langue naturelle. Cette tâche a été développée en plusieurs étapes. La première a été de développer une méthode d'extraction des entités voulues via une suite de différentes phases. Pour la seconde étape, afin de passer outre les limitations de l'interface graphique (et utiliser nos scripts sur des serveurs), il a fallu développer une application Java avec L'API de GATE pour automatiser cette tâche et nettoyer certaines extractions non voulues via un système règles en post-processing. Cette application est réalisée avec un système de plugins, plus précisément chaque tâche est faite par un plugin de cette application. Une fois les extractions faite, l'application renvoi ces extractions en JSON

pour un traitement dans un script Python dans le format pris en compte par les scripts de Seevl, ceci est la dernière étape.

3. SYNTHÈSE DE LA SOLUTION APPORTÉE POUR CHACUNE DES TÂCHES

3.1. Solution de la tâche numéro 1

La solution consistait à installer, avec les outils mit à disposition par Musicbrainz, la base de données du même nom sous PostgreSQL. Une fois cette base de données installé, il fallait ajouter une possibilité de communication entre Virtuoso et la base de données de Musicbrainz via l'outil ODBC fournit par Virtuoso. L'étape suivante fut la création des vues dans Virtuoso permettant de rapatrier juste les données souhaitées pour enfin créer le fichier de mapping R2RML permettant d'avoir une partie des données de Musicbrainz en RDF et aussi interrogeables en SPARQL via le point d'accès SPARQL de Virtuoso.

3.2. Solution de la tâche numéro 2

Tous les résultats de cette tâche sont décrits dans la section 4.2. Pour la première partie, la méthode de chunking à elle seule n'a pas permis d'avoir un pourcentage de résultats corrects. L'étape suivante était d'établir un point de comparaison avec toutes les cérémonies existantes, il a donc fallu extraire de DBPedia toutes les occurrences de nom de cérémonies et ainsi les comparer avec une méthode de comparaison de chaîne de caractère, Levenshtein. Avec cette étape supplémentaire, les résultats positifs ont été augmentés, mais reste insatisfaisant.

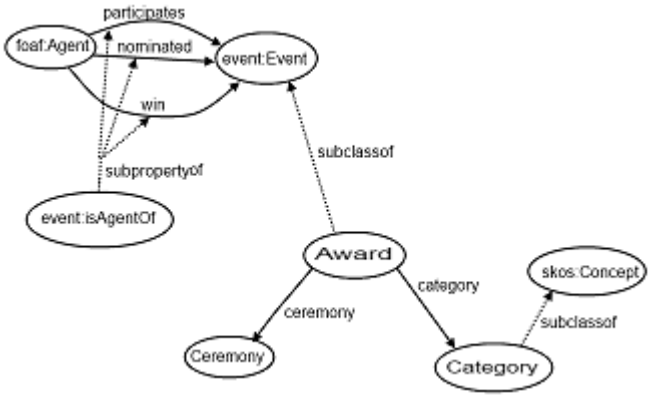
L'étape suivante consiste à donner une réponse à la question « est-ce que le résultat trouvé est juste ou non ? », ou a minima trouver un moyen d'y répondre le plus justement possible. Un système d'apprentissage a donc été introduit dans le processus, ce système est l'API Google Prediction. Cette API permet avec une liste d'exemples annotés (phrase extraite ; nom de la meilleure cérémonie retournée par la méthode de comparaison ; oui ou non) de retourner une réponse à la question. Cela a imposé un changement dans l'étape de comparaison: au lieu de choisir la meilleure chaîne de caractères, il a été décidé de prendre une liste des meilleures cérémonies comparées et ensuite d'étudier le pourcentage de réponse renvoyé pour chacune de ces cérémonies et prendre celle ayant le meilleur pourcentage. Une réponse est considérée comme correcte si l'API renvoi oui et est supérieur à un certain pourcentage.

Malgré une amélioration des résultats, nous avons décidé d'aller plus loin pour avoir des résultats plus satisfaisants. Une solution pour tenter d'améliorer le résultat était de garder toutes les extractions qui ont échouées et de refaire un « chunking » dessus afin d'éliminer un peu plus des mots inutiles de la première extraction. Ensuite les mêmes étapes de comparaisons et de l'API sont appliquées sur cette nouvelle extraction et seuls les résultats positifs sont ajoutés dans la liste finale. Cette étape donne un résultat satisfaisant, mais il n'était plus possible d'avancer à cause d'un problème qui sera expliqué dans la partie « problèmes rencontrés » de ce rapport.

Pour la seconde partie, l'extraction des catégories, la même méthode de chunking a été appliqué, mais cette fois aucun points de comparaisons dans DBPedia, et pas d'utilisation de l'API, il a donc

fallu trouver la meilleure grammaire possible pour les phrases qui inclus une catégorie, ce n'est pas le cas de toutes.

Pour finir, l'ontologie, pour modéliser ces données est celle-ci :



- Award : représente l'évènement décris par le vocabulaire Event[2] ;
- Ceremony : représente la cérémonie correspondante ;
- Category : représente la catégorie et/ou la sous-catégorie de la récompense.

L'ontologie au format N3 est disponible sur le site de Seevl[3].

3.3. Solution de la tâche numéro 3

Pour la première étape, le but était tout simplement de s'occuper de l'extraction des entités voulues via une suite de phases qui sont :

- localiser les entités nommées via un Gazetteer (liste des mots recherchés) ;
- manipuler ces entités nommées extraites via des grammaires ;
- catégoriser les entités nommées trouvées validées par les grammaires (artiste, label, etc…) ;
- lemmatiser les verbes reliant les entités nommées trouvées.

Une fois ces étapes implémentées, l'extraction n'était pas entièrement complète (causé par problème de coréférences (anaphores) voir 4.3.2). Il était donc nécessaire de créer une application pour trois raisons :

- la version graphique de GATE était utilisé, ce type de représentation est inutile pour Seevl, il fallait donc une application programmé via l'API de GATE fournie ;
- faire du prétraitement afin d'améliorer la qualité de l'extraction ;
- faire du posttraitement afin, aussi, d'améliorer la qualité de l'extraction.

Cette application a été conçue spécialement pour être étendue via des plugins. Ainsi il sera non seulement possible de créer des plugins GATE mais aussi d'autres plugins pour n'importe quel autre usage que de l'extraction d'entités nommées tout en utilisant une seule et unique application. L'application en elle-même ne fait strictement rien, elle sert juste à appeler les plugins, seul les plugins apportent une quelconque fonctionnalité. L'extraction d'entités nommées via GATE est donc

[2] http://motools.sourceforge.net/event/event.html
[3] http://seevl.net/schema/awards

faite via un plugin. La phase de prétraitement de ce plugin est nécessaire afin de connaître le genre de l'artiste (groupe, homme ou femme) via le pronom correspondant à l'artiste principal du texte. Statistiquement, dans un texte, la première apparition de ce type de pronom fait référence au sujet principal du texte. La solution était donc de trouver la première occurrence de ce type de pronom pour ensuite rajouter ce pronom au Gazetteer pour qu'il soit trouvé par GATE lors de l'analyse. Ensuite l'extraction ce déroule de la même manière qu'avec l'outil graphique de GATE. Un second problème (voir 4.3.2) cause de mauvaises extractions et notamment des doublons, c'est-à-dire qu'une entité est extraite plus d'une fois, par exemple pour l'artiste « Donna Summer » les entités « Donna », « Summer » et « Donna Summer » sont extraits, mais dans ce cas seul « Donna Summer » soit être pris en compte. Ce problème est résolu via un algorithme de détection de superposition qui prend seulement en compte l'entité nommée contenant le plus de caractères.

Ensuite pour les relations entre ces entités, il a fallu extraire aussi le verbe liant les deux entités intéressé et trouver le sens de ce verbe.

4. Résultats

4.1. Pour la tâche numéro 1

4.1.1. Résultats obtenus

Aucuns.

4.1.2. Difficultés rencontrées

Virtuoso contenait beaucoup de bugs dans son implémentation du standard R2RML, de long échanges ont eu lieu entre Seevl et le support de Virtuoso afin de réparer ces bugs. Ces bugs ont beaucoup ralenti le déroulement de cette tâche. Un autre problème étant que Virtuoso n'implémente pas entièrement le standard R2RML, il ne permet pas d'utiliser la propriété « rr:sqlQuery » qui permet d'effectuer une requête SQL directement dans le mapping R2RML, il fallait donc créer les vues correspondant à ces requêtes dans Virtuoso afin d'utiliser les données qui nous intéresse. Ces bugs ont aussi provoqués des problèmes au niveau de l'installation de Virtuoso, car il fallait réinstaller (du moins en grande partie) Virtuoso à chaque fois qu'un patch était réalisé pour un bug. Bien évidemment ce constat vaut aussi pour l'outil ODBC permettant la connexion de PostgreSQL à Virtuoso. Ces problèmes ce sont étalés sur une durée de 5 mois et n'est toujours pas résolu à la fin du stage. L'installation de Musicbrainz a aussi posé quelques problèmes (bugs au niveau du processus d'installation), mais la communauté a été très réactive et les problèmes résolues très rapidement.

4.1.3 Apports

4.1.3.1 Pour Seevl

Cette tâche a permis d'avoir en RDF toutes les informations citées dans la partie 2.1.1 sur tous les artistes contenus dans Musicbrainz et ainsi augmenter le nombre d'informations proposées par le plugin Seevl sur les artistes écoutés sur Youtube. Ceci est actuellement en intégration.

4.1.3.2 Personnel

Cette partie m'a permis de me familiariser avec le langage R2RML et l'outil Virtuoso.

4.2. Pour la tâche numéro 2

4.2.1. Résultats obtenus

Pour l'étape un de la première partie seulement 23 résultats corrects soit 28%, pour l'étape deux il y a 39 résultats corrects soit 48%, pour l'étape trois il y a maintenant 49 résultats corrects soit 60% et avec la dernière étape il y a 56 résultats corrects soit 69%. Pour la seconde partie, sur les 56 résultats 91% ont une bonne extraction de leur catégorie, pour celles qui en contenaient.

4.2.2. Difficultés rencontrées

La principale difficulté de cette tâche a rédidé dans le besoin de faire du cas par cas pour certaines categories, et en consequence d'implémenter un algorithme qui n'est absolument pas générique. Un exemple parmi d'autre pour lequel la méthode générique adoptée ne fonctionne pas peut être : « Best Music Score Golden Orange Award Winners » car la cérémonie correspondante est « International Antalaya Golden Orange Film Festival ». Ainsi, à cause de ce type de problème il n'était plus possible d'améliorer le pourcentage de réussite de cette méthode à moins de mettre dans le code, au cas par cas, toutes ces associations à la main.

4.2.3. Apports

4.2.3.1 Pour Seevl

Cette tâche a permis d'ajouter au plugin Seevl les informations sur les récompenses gagnées par certains des artistes que l'utilisateur écoute sur Youtube et dans certains cas la catégorie.

4.2.3.2 Personnel

Cette tâche m'a permis d'apprendre à utiliser NLTK ainsi que l'API Prediction de Google.

4.3. Pour la tâche numéro 3

4.3.1. Résultats obtenus

Les résultats pour cette tâche tournent autour de 73% à la fois pour le rappel et pour la précision.

4.3.2. Difficultés rencontrées

Le problème de coréférences (anaphores) était le premier important problème rencontré, en effet souvent dans un texte il ait fait référence au sujet principal du texte (ici un artiste) via un pronom (il, elle, etc…). Le second problème est la superposition des entités nommées extraites, par exemple en plus de l'extraction de « New York » il y a aussi l'entité nommée « York » qui est extraite, ce qui augmente le nombre d'extractions inutiles.

4.3.3. Apports

4.3.3.1 Pour Seevl

Cette tâche a permis à l'entreprise d'avoir une idée de la méthode à utiliser pour réaliser la tâche voulue. Néanmoins il reste encore un travail de recherche à faire sur certaines parties pour arriver à un résultat satisfaisant qui entraînerait sa mise en production.

4.3.3.2 Personnel

Cette tâche m'a permis d'apprendre à utiliser différentes méthodes d'extractions d'entités et aussi l'utilisation du framework GATE.

Rapport technique

5. ANNEXES DIVERSES

5.1. Annexe de la tâche 1

Définition de R2RML (cette définition est tirée de la page W3C) :

Une association R2RML se réfère à des tables logiques pour rapatrier les données venant de la base de données associée. Une table logique peut être une des trois choses suivantes :

- une table de la base de données ;
- une vue ;
- le résultat d'une requête SQL.

Chaque table logique est associée à RDF en utilisant une association de triplets. Une association de triplets est une règle qui associe chaque ligne d'une table logique à un nombre de triplets RDF. La règle a deux parties principales :

- une association de sujet qui génère le sujet de tous les triplets RDF qui sera généré à partir d'une ligne de la table logique. Les sujets sont souvent des IRI (International Resource Identifier, qui est une version généralisée des URI) qui sont générés à partir de la (les) clé(s) de la (des) colonne(s) de la table.
- de multiples associations prédicat-objet qui à son tour se compose d'associations de prédicats et d'associations d'objets (ou référençant les associations d'objets).

Les triplets sont produits en combinant l'association du sujet avec l'association du prédicat et l'objet d'association, et en appliquant ces trois sur chaque ligne de la table logique. Par exemple, la règle complète pour générer un ensemble de triplets devrait être :

- sujets : un modèle http://data.example.com/employee/{empno} est utilisé pour générer les IRI du sujet à partir de la colonne « empno » ;
- prédicats : l'IRI du vocabulaire constant « ex :name » est utilisé ;
- objets : la valeur de la colonne « ename » est utilisé pour produire un littéral RDF.

Par défaut, tous les triplets RDF sont dans le graphe par défaut du jeu de données de sortie. Une association de triplets peut contenir des associations de graphe qui placent quelques ou tous les triplets dans les graphes nommées à la place.

5.2. Annexe de la tâche 2

"And the winner is..." – Representing awards on the Web of Data

Julien Plu[1,2] and Alexandre Passant[1]

[1] seevl.net, MDG Web Limited
Unit 201, Business Innovation Centre,
NUI Galway, Galway, Ireland
julien,alex@seevl.net – http://seevl.net
[2] Université Montpellier 2, Sciences et Techniques,
Place Eugène Bataillon,
34095 Montpellier Cedex 5, France
julien.plu@etud.univ-montp2.fr – http://www.univ-montp2.fr/

Abstract. Awards are a particular kind of events happening in domains as various as entertainement, sport, and even scientific conferences. Here, we present an approach to model events on the Web of Data, and to extract event information about music artists from Wikipedia categories, combining (1) data extraction using background knowledge, and (2) machine learning with the Google Prediction API.

Keywords: Awards, events, music, Wikipedia, machine learning, natural language processing, SPARQL

1 Introduction

Whether it is in music, sport or academic conferences, awards are frequently granted to people for a particular accomplishment, such as the best live performance or the best album in the case of music. Yet, there is currently no easy way to query this information on the Web of Data[2]. In this paper, or goal is to provide an infrastructure to find who won such-and-such award, and in which category, while having the results provided in a machine readable data, i.e. RDF.

While a knowledge base like Wikipedia provide this information through categories (as people can be assigned to categorie describing an award, e.g. "Academy Award winner", the category does not explitely represent an award not its category. These Wikipedia categories can indeed include the name of the award, and sometimes the name of the category, but this information has a poor semantic, and a better structure is needed. To overcome this, we designed (1) an ontology to represent awards, and (2) a tool to extract this data from Wikipedia categories.

The remainder of this paper is organised as follows. First, we present a state of the art of ontologies represent events and, more particularly, awards. Then, we detail our *Award ontology*, from its design rationale to instances representation. In the third part, we present our approach for extracting such information from Wikipedia using wikipedia categories, etc.

2 State of the art

2.1 Generic event ontologies

An event can be described as a public gathering for the purpose of celebration, education, marketing or reunion. Events can be classified on the basis of their size, type and context. For instance, we consider the following as being events:

- *social and lifecycle* events: birthday parties, graduation days, weddings;
- *education and career* events: workshops, conferences, debates;
- *sports* events: football tournaments, Olympics Games, Roland Garros;
- *entertainment* events: music awards, concerts, festivals;
- *political* events: debates, summer schools, assembly meetings;

Hence, we consider award ceremonies as being events as they represent an important fact which brings together many people for a particular focus.

In order to model such events in a machine-readable way, ontologies are an obvious candidate. The field of event ontologies has been widely studied, and the following models can be considered to represent events on the Web of Data:

- The Event Ontology[3]: the most appropriate ontology to describe the representation of an event. It has a simple model, and has been extended in several domains, such as by the Press Association with their own event ontology[4];
- LODE: An ontology for Linking Open Descriptions of Events[5][5]: a complete event ontology, based on the previous Event Ontology and DOLCE;
- KMI ontology[6]: an ontology used for event extraction, providing very specific details, such as the event duration, sub-events, meeting organisers, etc;
- Event-F model[4]: a very detailed ontology for describing an event, and all their related properties;
- CIDOC-CRM [1]: an ontology specifically made for historical events.

While the Event Ontology and LODE provide enough details to represent events, they lack features to represent awards, for example their category (sport, music, etc.). Similarly, the KMI ontology and Event-F do not provide this support, and were too specific for our use-case. Finally, by being tailored for historical events, CIDOC-CRM was not directly appropriate neither for awards representation.

2.2 Award-related ontologies

In addition to the previous models, the following ontologies can more specifically describe awards:

- BBC sport ontology[6]: tailored for sport-related events, and awards, but not adapted for other kind of awards, such as music;

[3] `http://motools.sourceforge.net/event/event.html`
[4] `http://data.press.net/ontology/event/`
[5] `http://linkedevents.org/ontology/`
[6] `http://www.bbc.co.uk/ontologies/sport/2011-02-17.shtml`

- SWPortal ontology[3]: a perfect model to academic conferences awards to represent a conference award, but not appropriate to describe a sport award by example;
- Sport-ontology[7]: similar to the BBC sport ontology, but with a simplier model (and less specific) for describing a sport event;
- Baseball Ontology[8]: an ontology describing baseball events, but not appropriate for other domains.

3 Award Ontology

The main purpose of this ontology was how to describe an award event for a music artist. Our needs were to describe the ceremony where an artist won an award (e.g. "MTV Music Awards") as well as the category of this award (e.g. "Best song"). Not only it is useful to describe the event, but it can help to answer the following queries:

- list all artists that have been nominated and won in a category for a given ceremony;
- list all the awards and nominations of an artist on the same year;
- identify the career path of an artist, e.g. winning the "Best song" the year after winning the "Best your talent" award.

To do so, we extended the event ontology and developed a lightweight awards ontology, available at `http://seevl.net/schema/awards`, described in the following schema. The ontology classes are:

- *Award*: a subclass of *Event:Event*, representing the Award itself, as an event;
- *Ceremony*: a class representing the ceremony corresponding to an event (e.g. "MTV Music Awards");
- *Category*: a subclass of *skos:Concept*, describing a category in a ceremony (e.g. "Best song").

and its properties:

- *time*: date of the award described (e.g. "The award took place the 20/07/2012");
- *place*: place of the award described (e.g. "The award took place in Paris");
- *ceremony*: corresponding to the ceremony of an award (e.g. "The ceremony is MTV Music Awards");
- *category*: corresponding to one or many categories of an award (e.g. "The category is Best Song");
- *win*: a subproperty of *isAgentOf*, meaning that the artist won the award in a single or many categories;
- *participates*: a subproperty of *isAgentOf*, meaning that the artist is a participant in one or many categories;

[7] `http://code.google.com/p/sport-ontology/`
[8] `www.daml.org/2001/08/baseball/`

– *nominated*: a subproperty of *isAgentOf*, meaning that the artist is nominated in an award in one or many categories;

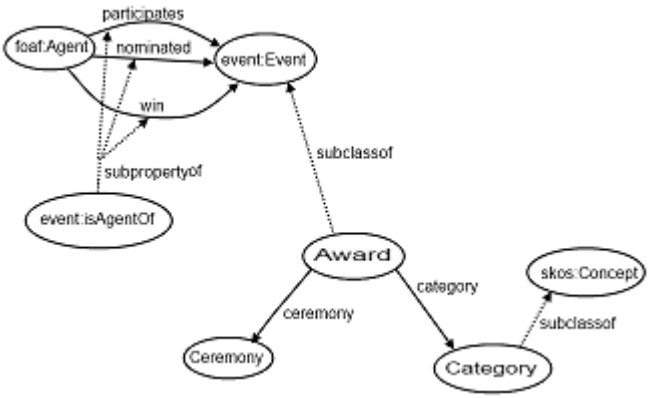


Fig. 1. award ontology schema

We extensively reused the Event Ontology to describe an event corresponding to an award, and we redefined the *isAgentOf* property which links a *foaf:Agent* to an *Event:Event* in order to identify who won, is nominated or participates in a ceremony, and more precisely at an award.

We also identified the need to organise categories in a hierarchical fashion, for instance considering that "Best album" is a top-category of both "Best world-music album" and "Best alternative rock album". We rely on SKOS to represent these hierarchies, representing *Category* as a sub-class of a *skos:Concept*[9]:

```
ex:best_album rdf:type awards:Category .

ex:best_world_music_album> rdf:type awards:Category ;
  skos:broader ex:best_album .

ex:best_alternative_rock_album rdf:type awards:Category ;
  skos:broader ex:best_album .
```

A simple complete example of award representation can be:

[9] Prefixes ommitted for space reason

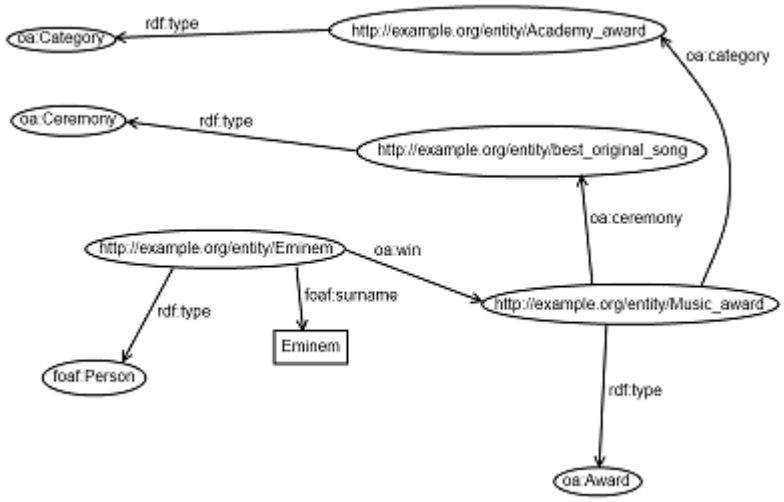


Fig. 2. Music example schema

```
ex:eminem rdf:type foaf:Person ;
    foaf:surname "Eminem";
    awards:win ex:Music_award .

ex:Music_award> rdf:type awards:Award ;
    awards:ceremony ex:Academy_award ;
    awards:category ex:best_original_song .

ex:Academy_award rdf:type awards:Ceremony .

ex:best_original_song rdf:type awards:Category .
```

While the ontology has been specifically designed for music, we realised that it could be used for describing awards in other domain (sport, best paper conference, film, etc.), as demonstrated by the following example and its corresponding Turtle code.

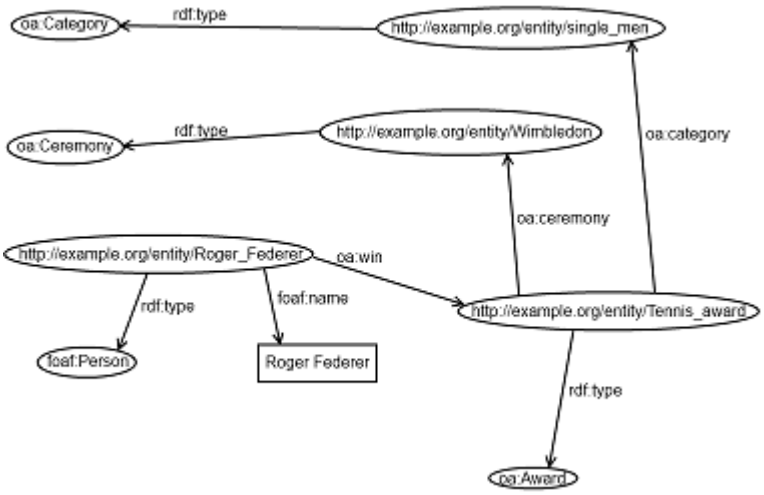


Fig. 3. Sport example schema

```
ex:Roger_Federer rdf:type foaf:Person ;
    foaf:name "Roger Federer";
    awards:win ex:Tennis_award .

ex:Tennis_award rdf:type awards:Award ;
    awards:ceremony ex:Wimbledon ;
    awards:category ex:single_men .

ex:Wimbledon rdf:type awards:Ceremony .

ex:single_men rdf:type awards:Category .
```

4 Award extraction

While this ontology can be used on top of any data, we also focused on extracting relevant awards information from Wikipedia mapped to this model. On Wikipedia, artists are assigned several categories, some of them containing the word “award”. Our goal here was to transform these categories (as sentences) into the corresponding award information (As structured data), using the previous ontology.

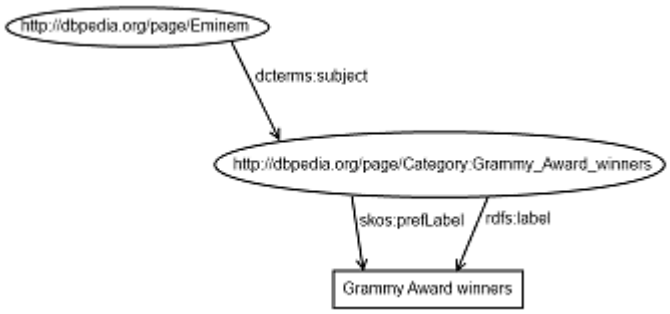


Fig. 4. Schema representing this link between a person and a string category

The main task was to turn the information embedded in these sentence categories into triples, designed with the award ontology. We used Python, and in particular NLTK (Natural Language ToolKit) as a NLP framework.

NLTK allowed us to extract a piece of a sentence with a method called *chunking*, by creating several regular expressions corresponding to tags associated to each words (noun, adjective, etc.) or words sequence to extract. We split the task in two parts: (1) cremony extraction, and (2) category extraction. Yet, taken alone, the chunking part was not enough sufficient to have a good percentage of correct answer for the extracted ceremonies. To improve the results, we then compared the results with existing ceremonies (defined in Wikipedia), by retrieving those from DBpedia and computing the Levenshtein distance between our results (for each ceremony name retrieved) and the list of ceremonies in Wikipedia/DBpedia, taking the best matching as a result. Finally, to improve the results that were only about 50% using this pipeline, we used Machine Learning, with the Google Prediction API[10], to train the algorithms to differentiate between correct and wrong answers. Our training set as defined as a set of mappings between the NLTK extraction and the correct ceremony name, as follows:

```
NLTK extracted ceremony;dbpedia ceremony;yes or no
```

We were eventually able to reach a score of 69% accuracy for the mappings after using this technique[11]. Here is an example of successful extraction with the process.

```
sentence : ARIA Award winners
extracted ceremony : ARIA Music Awards
```

And another more complete example, including both ceremony and category extraction:

[10] `https://developers.google.com/prediction/`

[11] The tests have been run on all the corpus of the Wikipedia categories containing the term *award*

```
sentence: worst actress golden raspberry award winners
extracted ceremony: golden raspberry awards
extracted category: worst actress
```

Corresponding to the following representation with the awards ontology:

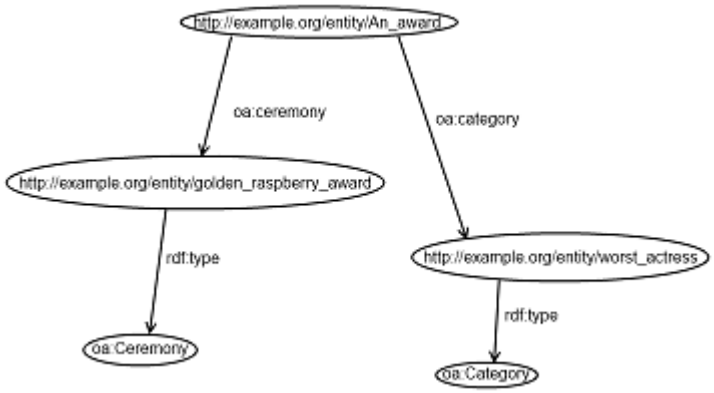


Fig. 5. Schema of the example

```
ex:An_award rdf:type awards:Award ;
    awards:ceremony ex:golden_raspberry_award ;
    awards:category ex:worst_actress .

ex:golden_raspberry_award rdf:type awards:Ceremony .

ex:worst_actress rdf:type awards:Category .
```

It's easy after to know at which artist this extraction can be associated.

5 Conclusion

In this paper, we addressed two main questions regarding events, with a particular focus on the concepts of awards:

- how to represent awards, a particular kind of events, on the Web of Data;
- how to extract award information from semi-structured sources, and to represent it as structured data on the Web

This extraction is currently integrated into seevl, a Chrome extension for music discovery on YouTube. Music fans will consequently be able to identify videos of artists having won particular awards, and to generate related playlists. In the future, we plan to extend the extraction to identify the year related to

the awards, and to focus on Wikipedia pages content (and not only category information), to extract this data.

References

1. Martin Doerr. The CIDOC Conceptual Reference Module: An Ontological Approach to Semantic Interoperability of Metadata. *AI Magazine*, 24(3):75–92, 2003.
2. Tom Heath and Christian Bizer. *Linked Data: Evolving the Web into a Global Data Space*. Synthesis Lectures on the Semantic Web. Morgan & Claypool Publishers, 2011.
3. Knud Mller, Livia Predoiu, and Daniel Bachlechner. Portal ontology. Technical report, DERI, 2004.
4. Ansgar Scherp, Thomas Franz, Carsten Saathoff, and Steffen Staab. F–a model of events based on the foundational ontology dolce+DnS ultralight. In Yolanda Gil and Natasha Fridman Noy, editors, *K-CAP*, pages 137–144. ACM, 2009.
5. Ryan Shaw, Raphaël Troncy, and Lynda Hardman. LODE: Linking Open Descriptions of Events. In Asunción Gómez-Pérez, Yong Yu, and Ying Ding, editors, *ASWC*, volume 5926 of *Lecture Notes in Computer Science*, pages 153–167. Springer, 2009.
6. Maria Vargas-Vera and David Celjuska. Ontology-driven Event Recognition on Stories. Technical report, KMI, the Open University, 2003.