Knowledge extraction from audio content service providers' API descriptions

Damir Juric, György Fazekas

Queen Mary University of London, London, UK
{d.juric, g.fazekas}@qmul.ac.uk

Abstract. Creating an ecosystem that will tie together the content, technologies and tools in the field of digital music and audio is possible if all the entities of the ecosystem share the same vocabulary and high quality metadata. Creation of such metadata will allow the creative industries to retrieve and reuse the content of Creative Commons audio in innovative new ways. In this paper we present a highly automated method capable of exploiting already existing API (Application Programming Interface) descriptions about audio content and turning it into a knowledge base that can be used as a building block for ontologies describing audio related entities and services.

Keywords: metadata, audio content, ontologies, natural language processing, knowledge extraction.

1 Introduction

The field of digital music or more general digital audio content (content that does not include only songs, but sounds or soundscapes) is very propulsive one, but still creatives who work in the industries that are using digital audio content in their daily work face with some basic problems. One of those problems is a lack of technologies for accessing and easily incorporating audio content directly into a creative workflow[1]. In this paper we will not deal with problems of accessing the data or workflow enhancements but with a first and very important step that will eventually allow us to tackle those problems in the future. That first step is conducting the task of knowledge and metadata extraction for potentially very large amount of unstructured data that already exists in this domain. In the particular case of sound and music, a huge amount of audio materials like sound samples, soundscapes and music pieces, is available online and released under Creative

content

mmons

[1] Deliverable D2.1: Requirements Report and Use Cases: http://www.audiocommons.org/materials/

is large and very diverse (and with time that dataset will increase considerably) it is problematic for creatives to grasp the potential benefits of such a dataset, exactly because of those characteristics. There is a need for tools and methods that will allow the assessments and extraction of Audio Commons data from its existing poorly structured and fragmented form into something more formal. A perfect candidate for that would be an ontology. However, building the Audio Commons Ontology (and ontology in general) requires extensive knowledge about the domain that the ontology will describe. Knowledge about the domain includes the vocabulary of the domain and knowledge about the workflows (processes) that are being carried out by various roles involved in them. This paper will describe the method for gathering the formalized knowledge (knowledge database) from unstructured data of audio content providing services that already exist. This should be only the first step in a much bigger european project that will try to offer creative users more integrated ways of searching for and using audio content. There is a lack of globally unique and interoperable identifiers that creators could easilly get familiar with and use it in their creative process. The aim of the Audio Commons project[2] is to create an ecosystem of content[3], technologies and tools to bring the Audio Commons to the creative industries, enabling creation, access, retrieval and reuse of Creative Commons audio content in innovative ways that fit the requirements of the use cases considered (e.g., audio-visual, music and video games production). Currently creative users can access various audio content by using existing APIs for programmatically accessing content from sites like Jamendo[4], Freesound[5], Europeana[6], etc. Despite the number of providers and large and their libraries are extensive users face with problems such as limited access to data due to the lack of high quality and unified metadata [1], there is no unified access mechanism that will connect the different APIs (APIs have different specifications), retrieval tools are inadequate and that all leads to the fact that audio commons content is not frequently used in the professional environment. The biggest challenge left to solve is to define the metadata requirements in creative applications, to design the appropriate ontologies for data representation and finally to provide reliable metadata to facilitate access to Audio Commons content. In this paper, we will present a highly automated method for harvesting the API descriptions of audio content providers to build the knowledge database and vocabulary that will be used as a basic building block for the Audio Commons ontology. In Section 2 we will mention the work that has been already done on the creation of music and audio related ontologies. In Section 3 we will describe the music API dataset and present the method that will conduct machine reading task on the dataset and the creation of knowledge database (implemented as a graph database). Finally, we evaluate the method on an audio content provider dataset in Section 4 and conclude in Section 5.

---

[2] Audio Commons Project - http://www.audiocommons.org/

[3] The Audio Commons Ecosystem (ACE) referred to as ACE in the rest of the paper.

[4] Jamendo - https://developer.jamendo.com/v3.0

[5] Freesound - https://www.freesound.org/help/developers/

[6] Europeana - http://www.europeana.eu/portal/

2 Related Work

Ontology construction is normally carried out manually but in recent years automated approaches have emerged. Most of these approaches deal with raw text, but some also use other sources such as Wikipedia pages and HTML (HyperText Mark-Up Language) forms. Manually developed general ontologies are still the most widely used type [2]. The construction of such ontologies is a very expensive and time-consuming process. Moreover, the process of acquiring new knowledge is always needed and it requires ongoing work by human experts, even after the ontology has been released. In order to solve the problem of reliance on a cumbersome manual construction, some techniques propose broader collaboration during the ontology construction process, as in the case of Semantic Wikipedia [3], where facts are created and incorporated into an ontology by many volunteers. As for automated approaches, Zhou [4] gives a typical scenario of an ontology learning process (which can either be manual or automated) and it consists of: creating concepts, creating relations, ontology population and ontology evaluation. Wiszniewski [5] introduces a metamodel for ontology learning from text and presents an extensive survey of ontology learning models. As for the audio and music domain, there have been research carried out on the construction of music related ontologies and metadata. The Music Ontology [6] allows for describing the music production workflow from composition to delivery, while the Studio Ontology is for capturing the nuances of record production by providing an explicit, application and situation independent conceptualisation of the studio environment [7]. Both are presented as a modular framework of ontologies using core elements that allow for the representation of time-based events (using the Event and Timeline ontologies), and the workflow of music production in an editorial context subsumed under broader terms defined by the Functional Requirements for Bibliographic Records (FRBR) [8]. Expressed as a layered entity-relationship (ER) model, FRBR distinguishes three types of things, entities, attributes and relationships. An entity can be anything from a physical object to an abstract concept, relationships specify interactions between entities and attributes are properties or characteristics of entities or relationships. A particular group of entities represent the products of intellectual or artistic works are of specific interest. In the music domain *Work* may represent a certain musical composition such as a Beethoven violin sonata. A respective *Expression* may be the recording by Itzhak Perlman, a *Manifestation* may refer to the CD release by the record label Naxos and finally an *Item* represents a specific physical CD copy of this release. The Audio Feature Ontology [9] provides a descriptive framework for expressing different conceptualisations of the audio feature extraction domain and enables designing linked data formats for representing feature data. There is an ongoing work on Europeana Data Model for sounds[7].

[7] Europeana profile for sound - http://pro.europeana.eu/get-involved/europeana-tech/europeanatech-task-forces/edm-profile-for-sound

3 Music API descriptions

3.1 Ontology and Web Services

The Audio Commons project intends to include different Web Services providing music related metadata into its ecosystem. The project started with well-established Web Services for music data retrieval like Freesound offering its content. Freesound is a sound sharing site with more than 300 000 sound samples (including sound effects, instrument samples and field recordings). The content is released under several types of licences. The service has built its own API that is available for users (Fig. 1). From the aspect of a creative industry user there is an ongoing problem in accessing that content (or content from similar services) because there is no unifying ontology that is describing metadata from different services in the music domain, making it difficult to query these services consistently using unified terms. Building such ontology can be time demanding and cumbersome task for domain experts but it is a necessary task that will be used later on as a basis for building semantic web services and orchestrating user queries.

One of the first challenges that needs to be overcome is the problem of knowledge acquisition. The ontology should be capable of describing entities and actions that are already defined in the data models of various service providers and that number can be potentially large. This is the reason why it is necessary to provide support to knowledge engineers at an early stage of the knowledge acquisition task. Following well defined methodology that describes the process of ontology building can be helpful and there is considerable amount of work addressing this issue [10]. These methodologies tend to be generic and they often can't be scaled for the specific problem. In a project like the one described in this paper there is a need for an automated or semi-automated tool that can be used by a knowledge engineer to analyse different API dictionaries (including the one that will be joining the AC ecosystem in the future).

3.2 Music API dictionaries

Music service APIs allow users to browse, search, and retrieve information about other users interacting with the service, but most importantly the collections of sounds and music pieces or particular sounds from their extensive databases. For example, it is possible to retrieve similar sounds to a given target (based on content analysis) or retrieve automatically extracted features from audio files. The Freesound API allows users to perform advanced queries combining content analysis features and various metadata. The API also allows different actions for manipulating the music data like upload action, writing comments, rating, etc.

Name	Type	Description
id	number	The sound's unique identifier.
url	URI	The URI for this sound on the Freesound website.
name	string	The name user gave to the sound.
tags	array[strings]	An array of tags the user gave to the sound.
description	string	The description the user gave to the sound.
geotag	string	Latitude and longitude of the geotag separated by spaces (e.g. "41.00823
created	string	The date when the sound was uploaded (e.g. "2014-04-16T20:07:11.145")
license	string	The license under which the sound is available to you.
type	string	The type of sound (wav, aif, aiff, mp3, or flac).
channels	number	The number of channels.
filesize	number	The size of the file in bytes.
bitrate	number	The bit rate of the sound in kbps.
bitdepth	number	The bit depth of the sound.
duration	number	The duration of the sound in seconds.
samplerate	number	The samplerate of the sound.

Fig. 1. Example of the Freesound API metadata and service descriptions

When looking into descriptions for each of the parameter in the API dictionary it is possible to distinguish different types of information that are implied in the text. For example, parameter named *channels* is described using a simple description: *the number of channels*. On the other hand, parameter like *tags* is described as: *an array of tags the user gave to the sound* or parameter *descriptions* described as: *the description the user gave to the sound* are describing parameter and its context (caused by a certain action). These descriptions bring valuable insight into a set of actions that are being carried out by various entities connected with the service. Having the capability to identify different contexts where various entities can take different (or the same) roles can help us to model the ontology that will describe specific contexts in a more general way. We can say that Web Service API parameters and their descriptions could serve as basis for a bottom-up approach of building the ontology. These descriptions are an important source of knowledge about the entities existing in the service for a knowledge engineer with a task of building the ontology that will exist on top of those services.

3.3 Method overview

As mentioned in the previous section each service that allows its content to be queried (using REST (REpresentational State Transfer) API) needs to build the dictionary that will be made of parameters names, parameter types and parameter descriptions. Those dictionaries are exposed publicly on the Internet for users or developers for retrieving the content they want by calling specific parameters.

We are proposing a set of tasks that will allow the knowledge engineer to better understand the concepts and relationships between entities that exist in a web service data model that will potentially be mapped to an ontology as classes and properties. The method is using various natural language processing techniques to analyse the parameter descriptions and create a collection of facts that will be represented in a graph database. The pipeline for the proposed method can be seen on Fig 2. It consists of the following tasks:

1. Web Scraping (Harvesting) – technique of extracting information from websites
2. Repository – of extracted parameter descriptions
3. Information Extraction - extraction of structured relation triples from plain text

4. Semantic Role Labelling - detection of the semantic arguments associated with the predicate or verb of a sentence and their classification into their specific roles
5. Visualisation and Manipulation – one of the possible usages would be annotation of WDSL (Web Services Description Language) or OWL-S (Semantic Markup for Web Services) service descriptions

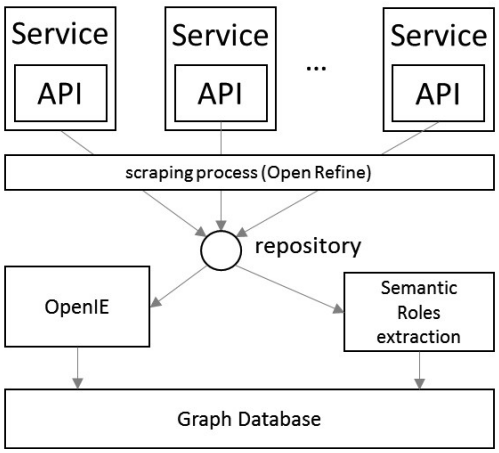

Fig. 2. Pipeline overview

3.3 Information extraction

Information extraction systems are often used for tasks like question answering, relation extraction, and information retrieval, because they can produce relation triples from unstructured text. IE systems search for a collection of patterns over either the surface form or dependency tree of a sentence. Although a small set of patterns cover most simple sentences (e.g., subject verb object constructions), relevant relations are often spread across clauses or presented in a non-canonical form [11]. For example, the parameter *id* described as: *the sound's unique identifier* will be transformed into a triplet: *sound*, *have*, *unique identifier*.

All produced triplets have high confidence C=1.0 but the produced statements are not canonical. For example, entity URI (Uniform Resource Identifier) came in four different variants: *(1.0, URI, point to, complete analysis result sound*), (*1.0, URI, point to, complete analysis result*), (*1.0, URI, point to, analysis result sound*), (*1.0, URI, point to, analysis result*). A solution to this problem is to implement a simple algorithm that will keep the statement with minimum numbers of terms constituting an object (Fig. 3) as a candidate for ontology. Since our goal is not to create an ontology in a completely automated fashion (manual refinement by domain expert is important) we decided to include all statements as a candidate for the ontology and decide which one to keep in a later stage (visualisation).

```
for each triple T(S, V, O)
    for triple T(S, V, O) == T'(S, V, O')
        numObjectTerms = count(split(O'))
        minObjectTerms = numObjecTerms
        if numObjectTerms < minObjectTerm
                candinateT = T'(S, V, O)
```

Fig. 3. Choosing a statement

3.4 Semantic Role Labelling

As a parallel task with information extraction we use an NLP technique called dependency parsing to analyse music service API descriptions. The Stanford dependencies provide a representation of grammatical relations between words in a sentence that are designed to be easily understood and effectively used by people who want to extract textual relations. Stanford dependencies (SD) are triplets: name of the relation, governor and dependent [12]. This approach was adopted as it is generally accepted as the best way forward when one does not know what is being looked for *a priori*. Entities extracted from textual descriptions should correspond to the lexical pattern shown in Fig. 4. The number of identified patterns depends on a number of identified Subject-Verb-Object (S-V-O) patterns – also one activity can contain more than one S-V-O pattern (e.g., *If the sound is part of a pack, this URI points to that pack's API resource* contains two S-V-O patterns). Since the result of the information extraction task described in previous section produces triples in S-V-O form we will include those triples to the semantic role labelling process to get the ontological pattern that correspond to the lexical pattern shown on Fig. 4. Our decision to create an ontological representation shown in Fig. 4 is influenced by the Provenance Ontology[8] and the Media Value Chain Ontology (MVCO)[9]. Both the Provenance (PROV) ontology and the MVCO ontology use action/activity entity that is connected with role and object entities through relationships as shown on Fig. 6. This kind of representation will be reproduced with the algorithm described below.

We can describe the algorithm as follows (Fig. 5): If A is a set of elements describing one particular action A then, for every A, we have $A_N, I_1,...,I_n, O_1,...,O_k, R$ where N is a string containing a description of an action, I_n is a set of strings describing elements that are part of the action. Set O_k is a set of strings describing the entities that are created as a result of the action, and R is an entity that started the action A (subject).

The following rules are implemented:

- Rule1: For every action A implied in the parameter description there should be a corresponding class A_{ont} in the ontology.
- Rule2: For every parameter description P_{verb} containing verb V_1, $V_{2, ..., } V_n$ create A_{ont1}, $A_{ont2, ...,}$ A_{ontn} class by splitting P_{verb} into a set {$P_{verbpart1, ...,}$

[8] Provenance ontology - https://www.w3.org/TR/prov-o/

[9] Media Value Chain ontology: http://dmag.ac.upc.edu/ontologies/mvco/

$P_{verbpartN}$} (e.g., *If the sound is part of a pack, this URI points to that pack's API resource* should create two action classes A_{ont1} = *being* and A_{ont2} = *pointer*.

- Rule3: For A_{ont} class there is a set of elements E_{ont} = {AG_{ont}, S_{ont}, A_{ont}}. Relationship S_{ont} between AG_{ont} and A_{ont} classes is called agent relationship. For example: E_{ont} = {*URI*, *pointer*, *pointingAction*}.
- Rule4: For A_{ont} class there is a set of elements T_{ont} = {PT_{ont}, O_{ont}, A_{ont}}. Relationship O_{ont} between PT_{ont} and A_{ont} classes is called patient\theme relationship. For example: T_{ont} = {*API_Resource, thing_pointed, pointingAction*}.
- Rule5: For some parameter descriptions P_{verb} containing verb V_n there is a set of elements Z_{ont} = {A_{ont}, V_{ont}, PO_{ont}} where V_{ont} is a relationship between action class and the object of a preposition. For example: Z_{ont} = {*giveAction*, *to*, *sound*}.

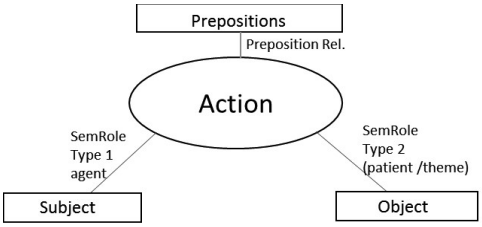

Fig. 4. Lexical Pattern

We use the Stanford dependency parser [12] to uncover the dependency tree for each parameter description. Our method is focusing on the following dependency relations:

- *nsubj*(V, S) - a nominal subject is a nominal phrase which is the syntactic subject and the proto-agent of a clause. (Implies subject-verb relationship.)
- *nsubjpass*(V, S) - a passive nominal subject is a noun phrase which is the syntactic subject of a passive clause. (Implies verb-object relationship.)
- *dobj*(V, O) - the direct object of a verb is the second most core argument of a verb after the subject. (Implies verb-object relationship.)
- *nn*(N, S/O) - a noun compound modifier of an NP is any noun that serves to modify the head noun. This pattern is used to expand the subject or object (example: *information descriptors* instead *descriptors*).
- *prep*(A, S/O) - A prepositional modifier of a verb, adjective, or noun is any prepositional phrase that serves to modify the meaning of the verb, adjective, noun, or even another preposition.

To automate the process of creation of the ontological classes described by the rules method uses various lexical repositories. The Unified Verb Index[10] is large list of English language verbs and a system that merges links and Web pages from four different natural language processing projects that are providing lexical information about verbs. One of those projects is PropBank [13] - a corpus of text annotated with information about basic semantic propositions. Verbs in the PropBank corpus can have a semantic role, also called argument, associated to them, which connects the verb with

[10] Unified Vverb Index - http://verbs.colorado.edu/verb-index/

the subject (the agent) and the object called patient/theme (some authors differentiate between patient and theme but the Penn Treebank regards it as a single patient/theme argument). Each argument is given a number. The agent and the patient are always given the argument numbers 0 (Arg0) and 1 (Arg1), respectively. In some cases Arg0 does not exist so the role of the agent is given by Arg1. The next lexical resource we use is WordNet [2]. WordNet is a large lexical database for English language and it's often used in the NLP domain. WordNet is used here to transform the verb found in the parameter description into its infinitive form (*c*). The main reason for conducting this task is the fact that a verb denoting one meaning or implying one specific action can appear in different representations. Also, acquiring the infinitive form of the verb allows the method to query for the derivationally related form of the verb so the action that the verb is implying can be labelled more naturally (example: V = gave, $V_{infinitive}$ = give, V_{drf} = giving). Since a verb can have a large number of different senses of derivationally related forms, gloss or dictionary definition is searched for predefined clues (a list of words with meanings similar to words *act* or *event*).

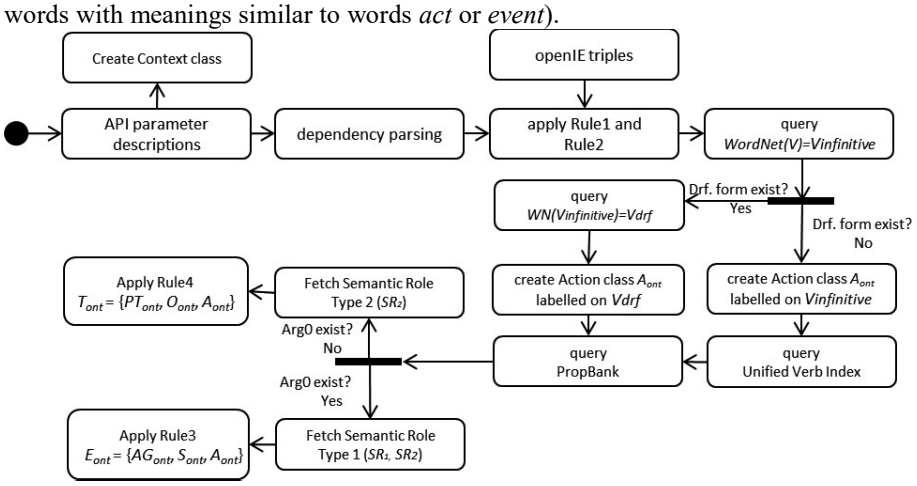

Fig. 5. Algorithm for automated creation of ontological patterns from API descriptions

For example, verb *gave* have three derivationally rated forms in the WordNet dictionary (*giver*, *giving*, *giving*) but only the third one is describing the act (gloss: *the act of giving*). The Unified Verb Index is used to find mappings between the $V_{infinitive}$ and the representation or the verb in PropBank (containing arguments Arg_0 and Arg_1). The values of arguments Arg_0 and Arg_1 are used to label the agent and the patient roles that are representing S_{ont} and O_{ont} in triples $E_{ont} = \{AG_{ont}, S_{ont}, A_{ont}\}$ and $T_{ont} = \{PT_{ont}, O_{ont}, A_{ont}\}$. Ontological patterns created from Music API descriptions should give insight into what kind of actions we can expect to deal with as well as objects (or possible IP entities) and roles (users of the system).

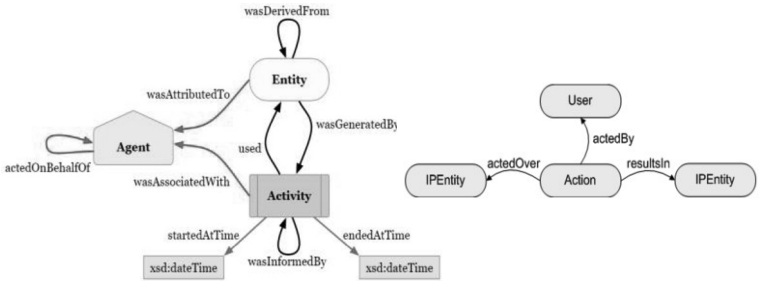

Fig. 6. Activities in PROVO and MVCO ontologies

4. Use case: The Freesound API

To assess the method discussed in Section 3 we deployed it over the Freesound API. We chose a graph database[11] as a repository for storing the ontological patterns extracted with the pipeline described in previous sections. In graph database every graph is a collection of two elements: vertices and edges, or it's a set of nodes and the relationships that connects them. Graphs represent entities as nodes and the ways in which those entities relate to the world as relationships. We are using the popular variant of a graph model called the *property graph.* The property graph is made up of nodes, relationships and properties. Nodes can contain properties. Nodes are entities that can store properties in the form of arbitrary key-value pairs. The keys are strings and the values are arbitrary data types. Relationships act as connectors between nodes. Having a properly labelled relationship is very important because it adds semantic clarity to the nodes structure. The graph database is used not just for storing the ontological patterns but is also a tool for visualising them. It supports Cypher a SPARQL like query language that can be used to manipulate the patterns and views of the patterns. Our information extraction system has been running over the collection of Freesound API parameter descriptions. The result of this action is a set of facts produced by the system with a certain confidence C (Table 1.).

Table 1. Facts produced by information extraction

	Freesound
nr. of param. desc.	81
nr. of facts	75
nr. of dist. fact	25

The algorithm took as an input 81 parameters (each parameter is coming with a short parameter description). The information extraction system suggested 75 facts (S-V-O patterns) from the input data. Since the IE system can produce variants of the same fact we used the algorithm from the Fig. 3 to keep only distinct facts and discarded the rest. Figure 7a. is showing how certain entities are connected with identical object through

[11] Graph database - https://neo4j.com/

different roles (contexts). For example, URI entity is used as a bookmark for a sound, as a pointer to a comment about the sound and simply as a link for downloading the sound. On the same input data, the semantic role labelling algorithm described on Fig. 5. created 10 action nodes (*download*, *rate*, *part*, *give*, *comment*, *retrieve*, *contain*, *point* and *upload*). The action nodes occupy the centre of the pattern that is shown on Fig. 4.

Table 2. Semantic role labelling

	Freesound
nr. of actions/events	10
nr. of agent/themes	31/20
nr. of semantic roles	51

The example on Fig 7b. is showing action nodes representing *retrieve* and *upload* actions that were identified in the parameter descriptions. Each action that denotes uploading of something (like *sound* in our example) will be merged with one unique node *uploadAction* and object/theme node of the action will be connected with a relationship *data, thing uploaded*. Relationship between *uploadAction* and agent/subject node is labelled as *agent, uploader*. The agent/subject node is labelled as *X* because there was not enough data in the parameter description to extract the actual label for that node (this can be done manually in the knowledge-base).

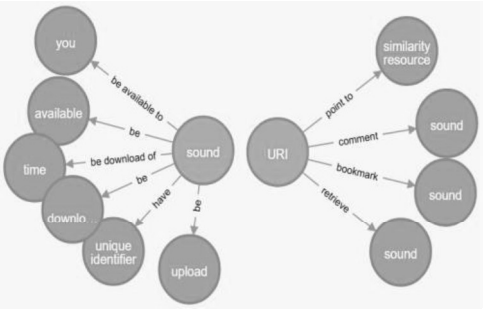

Fig. 7a. Subjects pointing to different objects

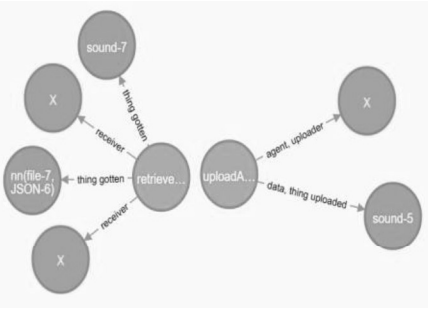

Fig. 7b. Action node with semantic roles

5 Conclusion and the future work

In this paper we presented an approach for extracting ontological patterns from music service API descriptions. Web service API descriptions represent a valuable source of knowledge about the capabilities of the service. Analysing these descriptions can give us insights about the actions/activities that are being carried out by different roles (usually users of the service). We presented an approach that is highly automated. While the construction of ontologies may never be completely automated, there is a need for methods proposed in this paper, that will ease the cumbersome task of ontology creation from scratch. Additionally, considering the amount of different services that the Audio Commons ecosystem can potentially have in the future, the "ontology bottleneck" could become a serious problem. Ontological patterns extracted from service API descriptions will be used by knowledge engineers as the next step towards the creation of the Audio

Commons ontology (an ontology that will describe various music and audio content entities and services). Our method proved successful in extracting ontological patterns from the Freesound API, thus it presents a promising direction. Future work will involve the creation of an interface that allows the knowledge engineer to build on the patterns extracted in an automated fashion providing an easy and convenient way of improving the quality of labels or filing the labels that could not be extracted from the descriptions. Recognizing important entities and actions from music service API descriptions will also assist in future tasks of the Audio Commons project that involves building semantic web services and orchestrating user queries.

References

1. Aslam, N., Ullah, I., Rohullah, B. S., Akram, T., & Shabir, M. Tracking the Progression of Multimedia Semantics: from Text Based Retrieval to Semantic Based Retrieval. World Applied Sciences Journal, 20, 4 (2012), 549-553.
2. Fellbaum C, editor. WordNet: An Electronic Lexical Database. Cambridge, MA: MIT Press; 1998
3. Krotzsch M, Vrandecic D, Volkel M, Haller H, Studer R. Semantic Wikipedia. Journal of Web Semantics 2007;5(4):251–61.
4. Zhou L. Ontology learning: state of the art and open issues. Information Technology and Management 2007;8(3):241–52
5. Wisniewski M. Metamodel of ontology learning from text. In: Emergent Web Intelligence: Advanced Semantic Technologies. Advanced Information and Knowledge Processing; Springer; 2010, p. 245–76.
6. Raimond, Y., Abdallah, S., Sandler, M., and Giasson, F. The Music Ontology. In International Society for Music Information Retrieval Conference (2007), 417–422.
7. Fazekas, G., and Sandler, M. B. The Studio Ontology Framework. In 12th International Society for Music Information Retrieval Conference (2011).
8. Saur., K.G. Functional requirements for bibliographic records: final report 1998., 136 p. (UBCIM publications; new series, vol. 19). ISBN 978-3-598-11382-6.
9. Allik, A., Fazekas, G., and Sandler, M. B. An Ontology for Audio Features. In 17th International Society for Music Information Retrieval Conference (2016).
10. De Nicola, A., Missikoff,cM., Navigli, R. A software engineering approach to ontology building; Information Systems 34 (2), 258-275.
11. Angeli, G., Premkumar, M. J., and Manning., C. D. Leveraging Linguistic Structure for Open Domain Information Extraction. In Proceedings of the Association of Computational Linguistics (ACL), 2015.
12. de Marneffe, C.-M., MacCartney, B., Manning, C.D.: Generating Typed Dependency Parses from Phrase Structure Parses. Proc. LREC 2006, 449-454.
13. Palmer M, Gildea D, Kingsbury P. The Proposition Bank: An annotated corpus of semantic roles. Computational Linguistics 2005;31(1):71–106