# CityMUS: Music Recommendation When Exploring a City

Pasquale Lisena[1], Lorenzo Canale[1], Fabio Ellena[1], and Raphaël Troncy[1]

EURECOM, Sophia Antipolis, France
{lisena|canale|ellena|troncy}@eurecom.fr

**Abstract.** Linked Data makes possible the discovery of interesting connections between semantic entities that belong to different domains. This paper presents CityMUS, a web application that gives to the user the experience of a walk in the city with the most suitable soundtrack, on the base of the urban context. The application relies on a recommender system that searches for paths in a knowledge graph between nearby places and music composers, making use of a combination of DBpedia and domain-specific datasets.

## 1 Recommending Music in Urban Environments

The consumption of music content on the move is one of the hottest trends. Some projects have sought to combine the experience of exploring a city with the one of media consumption [2, 6]. Context-based recommendation of music can be obtained using Linked Data. A strategy consists in exploiting arbitrary semantic connection between *Points of Interest (PoIs)* and musicians using knowledge graphs such as DBpedia, defining in this way a similarity measure between these two different domains [4]. So far, the implementation of this strategy relies on domain experts' who select a subset of classes used in the graph and manually define the interesting connection (i.e. path of properties), giving them a weight that can be used in a recommendation algorithm.

In this paper, we propose a completely unsupervised approach for developing a music recommendation system based on nearby PoIs. This strategy relies on the combination of general and encyclopedic datasets (i.e. DBpedia) and specialised datasets. We chose the city of Nice as an example for performing our experiment[1].

## 2 Data Preparation and Path Finder

PoIs and artists can be described in specialized datasets without necessarily be directly connected. Our strategy is to rely on a general purpose knowledge graph such as DBpedia for finding connections between such entities while using richer descriptions of PoIs and musical works in specialized datasets. Data about

[1] The demo is publicly available at `https://citymus.doremus.org/` while the source code is at `https://github.com/MultimediaSemantics/CityMUS`

musical work can be relatively poor in DBpedia (e.g. classical music often misses information about the composer) and we therefore rely instead on the data about their composers for finding connections. The goal is therefore to find paths in the DBpedia graph that connect entities that are also described in rich datasets such as 3cixty [8] for PoIs and DOREMUS [1] for artists.

**Datasets Selection and Interlinking.** The **3cixty** knowledge base [8] contains data about events and places from a touristic point of view. In this demo, 3cixty is used as a trusted source for POIs in Nice, retrieved through the API[2]. These PoIs should be matched to **DBpedia**. In order to perform the matching, we retrieved all the resources geographically located in Nice or that have `dbr:Nice` as `dc:subject`. The labels are firstly transformed in alphabetical-only ASCII string. Then, we interlink 3cixty PoIs to DBpedia applying a set of similarity measures[3] on labels and coordinates and choosing the one which maximise the average score among the best 3 ones of each measure.

The **DOREMUS** knowledge base [1] contains rich information describing classical and contemporary music, accessible through a SPARQL endpoint[4]. Among all the artist in DOREMUS, we retrieve a list of artist candidates from an indexed full dump of DBpedia, performing a simple search by artist name. We perform the interlinking considering the name and the birth and death dates and checking if they belong to a class that identifies a person or an artist.

**Path Finding.** Retrieving paths between two entities has already been solved by tools such as Relfinder [3] that searches the graph for possible paths with a given depth. However, increasing the depth generates both many more possible paths and makes the computation time prohibitive. In our context, the average depth is quite high, and we need in average 5-6 edges to connect a PoI of Nice to an artist. Running a query for retrieving all paths with a depth $d$ of 6 means in fact to retrieve all the triples in DBpedia, which at the time of writing is composed of 8.8 billion triples. For this reason, we developed a simplified version of Relfinder, that implements a bidirectional Breath First Search (BFS) [7]. In practice, we search for all the paths with depth $d$=2 of 3 from both the source (PoIs) and the destination (artists) entities. Then, we intersect the two sets in order to find the common nodes, and we operate joins that recreate the full paths. This technique reduces the complexity from $O(b^d)$ to $O(b^{d=2})$, with an exponential reduction of computation times. Moreover, we decided to not consider changes in the direction of the edges until the common node. Finally, a pruning is performed in order to remove cycles (repetition of the same entity in the path) and to preserve only the shortest path for each couple of entities.

[2] `http://aplicaciones.localidata.com/apidocs/`

[3] Partial Ratio, Token Set Ratio, Token Sort Ratio, Partial Token Sort Ratio, and the weighted combination of those (WRatio), all coming from `https://pypi.python.org/pypi/fuzzywuzzy`

[4] `http://data.doremus.org/sparql`

**Path Scoring.** Among all the possible paths between each PoI and artist, we are interested not only in the short ones, but also in paths that involve exclusive intermediate nodes, defined as the ones that minimize the generality formula:

$$gen = \frac{1}{\mid N \mid} \sum_{i}^{N} occ(r_i)$$

where $r_i$ corresponds to the $i_{th}$ resource of a path of length N and $occ(r_i)$ to the number of its occurrences in all found paths. Given $deep_{max}$ as the biggest path depth (in our case, 7) and $len(artist; poi)$ as the considered path length, we define the similarity between a PoI and an artist using a combination of different similarity measures [5]:

$$sim(artist; poi) = 1 - k(\frac{\log(len(artist; poi) - 1)}{\log(2 * (deep_{max} - 1))}) - (1 - k)gen$$

We select 5 artists for each PoI with the highest similarity score.

## 3 CityMUS Application and Evaluation

**CityMUS Mobile Application.** *CityMUS* is a mobile web application available at `https://citymus.doremus.org`. The app uses the geo-location API for getting the user position. The server generates then a playlist of tracks from the artists connected to the closest 3 PoIs, with a different weights according to their distance. The Spotify APIs are used in order to display and play the tracks (Figure 1.a). The user can see the path of the song that is currently played (Figure 1.c) and navigate the map for discovering the songs related to other PoIs (Figure 1.b).

**Evaluation.** The artists interlinking is evaluated by using as ground truth the one done by ISNI[5], for which the mapping is already in the DOREMUS dataset (although not used in the interlinking task). The evaluation give us a *precision* of 0.987, a *recall* of 0.940 and a *f-measure* of 0.963. For PoIs interlinking, we consider as correct the ones in a 250 meters distance from the one retrieved through the Google Maps API (70% of the total).

About the path, we are interested to select specific paths, which means maximising the $gen$ function. We prune in this way lots of paths that contain very common resources (i.e. classes or reference to Nice as a city). In particular, the paths we select for *CityMUS* contain resources that occur in average in just 0.001% of all paths after the pruning, while the highest number of occurrences appears in the 0.04% of them.

Future developments will involve the interlinking of different nodes in the musical metadata (i.e. the work itself, the genre, the historical period) that can reveal more interesting connections. A user evaluation is also planned, by adding the possibility to rate the appropiateness of a song.

[5] International Standard Name Identifier, `http://www.isni.org`