PHENICX

Deliverable 3.5
Methods to extract web information from music piece as a whole

Grant Agreement nr	601166
Project title	Performances as Highly Enriched aNd Interactive Concert eXperiences
Project acronym	PHENICX
Start date of project (dur.)	Feb 1st, 2013 (3 years)
Document reference	PHENICX-WD-WP3-JKU-140715-Webinfo_Methods-1.1
Report availability	PU - Public
Document due Date	Jul 31st, 2014
Actual date of delivery	Jul 25th, 2014
Leader	JKU
Reply to	Markus Schedl (markus.schedl@jku.at)
Additional main contributors (author's name / partner acr.)	David Hauger (david.hauger@jku.at) Marko Tkalčič (marko.tkalcic@jku.at)
Document status	Final

Project funded by ICT-7th Framework Program from the European Commission

Table of Contents

EXECUTIVE SUMMARY 3

1 BACKGROUND 5

2 INTRODUCTION 6

2.1 OVERVIEW 6

2.2 METHODOLOGY 6

2.3 CONVENTION 7

3 EXTRACTING MULTIMEDIA MATERIAL FROM GENERAL AND MUSIC-SPECIFIC WEB SOURCES 8

3.1 STATE-OF-THE-ART IN MUSIC INFORMATION EXTRACTION 8

3.2 LIMITATIONS OF STATE-OF-THE-ART APPROACHES FOR CLASSICAL MUSIC 9

3.3 EXISTING MIE APPROACHES IN AN ORCHESTRA SETTING 10

3.4 RETRIEVING WEB INFORMATION FROM WIKIPEDIA 11

3.5 RETRIEVING WEB INFORMATION FROM LAST.FM 14

3.6 RETRIEVING WEB INFORMATION FROM FREEBASE 18

3.7 STORAGE AND USE OF THE WEB INFORMATION 25

4 EVALUATION OF RELEVANCE AND QUALITY OF THE RETRIEVED MATERIAL 26

5 CONCLUSION 31

6 REFERENCES 32

6.1 WRITTEN REFERENCES 32

6.2 ACRONYMS AND ABBREVIATIONS 32

Executive Summary

This deliverable presents **methods to mine multimedia material on composers, performers, pieces, and instruments from web sources**. It naturally relates to Music Information Extraction (MIE), which is the task of identifying and acquiring structured pieces of information about music from unstructured or semi-structured sources. The deliverable is a result of Task 3.5 and will be used in particular to support WP5 when it comes to **create personalized multimedia information systems for Classical music**.

We first **reviewed the state-of-the-art in MIE**, whose methods rely on machine learning and pattern recognition techniques, in particular rule-based analysis, to identify highly specific categories of information. We found that almost all **state-of-the-art methods seem not suited for Classical music**, as they have been developed for Rock music. Nevertheless, we extended a **rule-based approach to identify members and their instruments in orchestra**, based on web pages gathered via querying Google, which is the state-of-the-art in MIE. Manual inspection of the results showed, however, that this approach performs inferior than using only a few, albeit high-quality web sources.

As a result, we concentrated our efforts on **developing techniques to identify and acquire multimedia material from three dedicated web sources**: *Wikipedia/DBpedia*, *Last.fm*, and *Freebase*. To this end, we elaborated methods that take into account **structural information** of the resources, **disambiguation techniques** required for music entities with ambiguous names (e.g. "Bach", which also means "brook" in German), and methods to identify and acquire **different multimedia material** and **different versions of this material**. Applying these methods to input provided by the consortium member RCO, we created corpora of multimedia material on composers, pieces, performers, and instruments, parts of which have already been delivered in months 6 and 12 for deliverables D3.7 and D3.8, respectively. The developed methods and extensions thereof will further be required for deliverable D3.9[1].

In order to **validate the relevance and quality of the acquired material**, we performed several assessments:

Manual inspection of the acquired material showed that on the one hand, information from *Last.fm* and *Freebase* is more structured and more concise than the one from *Wikipedia*. On the other hand, *Wikipedia* usually provides much more information in higher detail and various modalities. These facts allow to create **different representations for different users**,

[1] Deliverables D3.7, D.3.8, and D3.9 refer to the creation of corpora containing supporting multimedia material of different amounts and specificity: D3.7: "initial corpus", D3.8 "standardized corpus", D3.9: "final corpus".

varying, for instance, length of biographies, amount of images or audio material shown[2], or the music entity focused on (composer, piece, performer, or instrument).

We further used a **crowd-sourced questionnaire** to assess how potential users of a multimedia information system perceive different categories of multimedia information provided. Investigated categories were **entity** (composer, performer, piece, and instrument), **modality** (text, image, audio), and **amount or length** of the presented material (few/short vs. many/long). We investigated all combinations of these three aspects, asking 167 participants to judge each combination according to whether it was (i) **interesting** and (ii) provided **novel information**. Through this questionnaire, we found that there are **no general preferences**, irrespective of the user, towards any specific multimedia material or categories of information. This in turn means that harvesting multimedia material of different nature (entity, length, and modality) is crucial to satisfy each user's individual information need, and it supports our approach to target the different information categories.

In conclusion, the methods developed for this deliverable can be used to mine relevant and high-quality multimedia material about the music entities of composer, piece, performer, and instrument. They will be used to support deliverable D3.9 ("final corpus of supporting multimedia material") and to enable the creation of personalized music information systems, which is part of WP5.

[2] A good example is given in D3.8 ("standardized corpus of supporting multimedia material") for the composer Debussy, in which not only images of himself, but also interesting additional material was identified; for instance, an image of his grave and an image of a French banknote depicting his portrait.

1 BACKGROUND

This deliverable **D3.5** responds to **WP3**, **Task 3.5** of the PHENICX project, as described in the Description of Work (DoW). The goal of this task is to extract **web information for different musical entities**. The developed methods are hence used to extract material found on the web, on musical pieces, as well as on composers, instruments, and performers. We examined both music-specific and general sources of information and performed a **user study** to assess different aspects of the material (such as general users' preference towards particular kinds of material).

The developed methods are used to retrieve multimodal web information, i.e. textual information as well as different types of image and audio material. Preliminary versions of the methods described in the deliverable at hand have already been used to create two datasets: **D3.7** (initial corpus of supporting multimedia material) and **D3.8** (standardized corpus of supporting multimedia material), which are available to all partners. Improved algorithms will be used in **D3.9** (final corpus of supporting multimedia material).

The results of this work support **WP5**, in particular **Task 5.5** (improving meta-data based matching of music items at different levels of specificity). Furthermore, the multimedia material acquired through the developed methods is crucial to **WP6** for **Task 6.2** (personalized multimodal information system), to build user-aware music information and recommendation systems. Finally, also in **WP7**, **Task 7.1** (demonstrator development and testing) the multimedia material mined through methods presented here will be integrated.

2 INTRODUCTION

2.1 Overview

Task 3.5 aims at elaborating methods to extract different kinds of multimedia material from the web, addressing various Classical music entities, more precisely composer, performer, piece, and instrument. In this document, we report on the work performed in the first 18 months of PHENICX towards this goal.

Section 3 describes the **methods developed to identify and extract multimedia data from different web sources**. We first provide in Section 3.1 an overview of the state-of-the-art in the corresponding research task of Music Information Extraction (MIE). Subsequently, we describe the methods used to acquire material related to the musical entities of interest from three specific web sources: *Wikipedia*[3], *Freebase*[4], and *Last.fm*[5]. We further describe how semantic information from online catalogues like *DBpedia*[6] or the application of basic heuristics can be used for entity disambiguation.

Section 4 describes the **evaluation of the extracted multimedia material**. To this end, we performed both a **quantitative and a qualitative assessment**, the latter including manual inspection by the authors as well as a crowd-sourced questionnaire.

Section 5 eventually summarizes the work performed in this deliverable.

2.2 Methodology

The goal of Task 3.5 is to extract web information about musical entities, such as composer, performer, piece, and instrument. **Implementing and investigating the state-of-the-art in MIE**, we found that current methods are not suited for Classical music. Indeed, it turned out that instead of crawling arbitrary, music-related web pages, **information extraction from specific sources**, such as *Wikipedia* or *Last.fm*, provides more accurate and higher quality material.

Given this insight, we refocused the task on elaborating **heuristical and rule-based methods to accurately extract pieces of information from dedicated web pages** on Classical music entities, addressing different (i) modalities, (ii) entities, and (ii) amount and detail of information. We further implemented strategies for automated quality assessment and to identify different versions of the same item (and in turn select the best version). All developed methods for web crawling and MIE, as related to the deliverable at hand, are available either as Java programs or Python scripts.

3 http://en.wikipedia.org
4 http://www.freebase.com
5 http://www.last.fm
6 http://wiki.dbpedia.org

Based on repertoire information from ESMUC and RCO, we then gathered multimedia material from the web, using the developed methods. We **evaluated the relevance and quality** of the mined material by quantitative and qualitative assessments. Given the absence of a "ground truth" for the task at hand, we performed **manual inspection** of samples of the returned material and **a user study** to figure out general preferences towards certain categories of pieces of information.

2.3 Convention

We use the following writing conventions:

- **bold** for emphasis
- *italics* for brand and product names
- `Courier New` for software design, in particular source code

3 EXTRACTING MULTIMEDIA MATERIAL FROM GENERAL AND MUSIC-SPECIFIC WEB SOURCES

Music Information Extraction (MIE), i.e. the automatic identification and extraction of music-related entities from unstructured or semi-structured data sources, is the central goal of this deliverable. In this section, we present our investigations of and methods to MIE for Classical music in the context of the PHENICX project.

We first review the state-of-the-art in MIE applied to arbitrary web pages, then investigate an extension to a rule-based state-of-the-art approach, at the same time motivating our choice to focus on web pages dedicated to music entities instead of crawling arbitrary web pages, and eventually we elaborate on the developed methods to extract relevant and high-quality multimedia material from *Wikipedia*, *Last.fm*, and *Freebase*.

3.1 State-of-the-art in Music Information Extraction

The current task aims at providing information on general music entities, including composers, performers, instruments, and pieces. Although the focus of PHENICX is Classical music, some of the existing methods, which have been developed for Rock and Pop music, may be generalized and used within the project.

Existing work on (web-based) MIE typically first tries to identify a set of web pages related to music, the most frequent approach being to query a search engine with music terms and fetch the resulting pages. Based on the corpus acquired this way, methods to extract a certain category of information are proposed. For instance, [Schedl and Widmer, 2007] aim at extracting **band members and their roles (instruments)** via a set of pre-defined rules. To this end, n-grams are extracted from the web pages and several filtering techniques are employed to construct a set of potential members. Subsequently, the authors use several pre-defined rules (similar to "Hearst patterns") and count the number of times each rule can be applied to each potential member and surrounding text. Examples of rules are "`[member] plays [instrument]`" or "`the [role] [member]`".
Other works, such as [Krenmair, 2010] and [Knees and Schedl, 2011] approach the same problem using classification techniques, in particular, Support Vector Machines (SVMs), instead of pre-defined rules. The authors make use of part-of-speech (PoS) tagging, gazetteer annotations to identify keywords, and named entity detection, to eventually create feature vectors from web pages. However, the SVM-based approach generally does not outperform the earlier approach of hand-crafted rules, proposed in [Schedl and Widmer, 2007].

Another category of information addressed by MIE research is **country of origin** of artists or bands. State-of-the-art approaches are given by [Govaerts and Duval, 2009] and [Schedl, Seyerlehner, Schnitzer, Widmer, Schiketanz, 2010]. While the former mines these pieces of information from specific web sites, the latter distills the country of origin from web pages

identified by a search engine. Govaerts and Duval search for occurrences of country names in biographies from *Wikipedia* and *Last.fm*, as well as in properties such as "origin", nationality", "birth place", and "residence" from *Freebase*. The authors then apply simple heuristics to predict the most probable country of origin for the artist or band under consideration, for instance, predicting the country that most frequently occurs in an artist's biography. When using *Freebase* as data source, the authors again predict the country that most frequently occurs in the related properties of the artist or band. In contrast, [Schedl, Seyerlehner, Schnitzer, Widmer, Schiketanz, 2010] propose three different approaches to country of origin detection: (i) a heuristic which compares the page count estimates returned by *Google* for queries of the form `"artist/band" "country"` and simply predicts the country with highest page count value for a given artist or band, (ii) computing term weights (tf-idf vectors) from up to 100 web pages retrieved via *Google* for each artist and predicting the country with highest tf-idf score, using the artist name as query, and (iii) predicting the country whose name appears closest to pre-defined keywords, such as "born" or "founded" in the artist's set of web pages.

The two categories of information discussed so far are textual ones. The only scientific work on MIE focusing on other multimedia material, as far as we aware of, is [Schedl, Knees, Pohle, Widmer, 2006] and [Schedl, Widmer, Knees, Pohle, 2011], which aims at mining **album cover artwork** from arbitrary web pages. The authors first use search engine results to crawl web pages of artists and albums under consideration. Subsequently, both the text and the HTML tags of the fetched pages are indexed at the word level. The distances at the level of words and at the level of characters between artist/album names and `<img>` tags are computed thereafter. If these distances are below a threshold, the image is considered an album cover and downloaded. In addition, a content-based filtering step is performed to discard non-square images as well as images showing scanned compact discs.

In contrast to these very specific techniques forming the state-of-the-art in MIE, the methods we developed as part of this deliverable should not result in a single piece of information for a certain query, but should provide a **diverse set of multimedia content for all entities of interest**, including, for instance, textual results of different lengths suited for different audiences, descriptive images as well as images providing extended additional information, related audio files, videos, etc. In PHENICX, we hence extend earlier work, focusing on extracting information about composers, performers, pieces, and instruments, and take a multimodal approach; enriching the presentation by videos and images.

3.2 Limitations of State-of-the-Art Approaches for Classical Music

All of the state-of-the-art methods to MIE are very limited when it comes to Classical music. The two most important reasons are:

- All **approaches were developed for Pop and Rock music**, which is reflected by the entities they are tailored to: bands, instruments (voice, guitar, bass, and percussion),

album covers, genre prototypicality, etc. Entities of interest in Classical music are, on the other hand, not considered.

- **Fans of Classical music are relatively reluctant to maintain web pages related to their preferred genre**, but also to use social media to talk about music or share multimedia material, as we showed in a very recent study, in the context of PHENICX [Schedl and Tkalčič; 2014]. Also commercial promoters of Classical music are sparse on the web, compared to their amount for Pop and Rock music. Both yield to a limited number of available web sites, especially on composers and pieces.

Given the latter reason and the fact that dedicated web sources already provide a high quality of information and multimedia material, we decided to refrain from web crawling arbitrary (music-related) web pages and instead tune our algorithms to identifying relevant pieces of information, considering a fixed set of web sources, namely *Wikipedia*, *Last.fm*, and *Freebase*.

3.3 Existing MIE approaches in an orchestra setting

Nevertheless, we investigated the state-of-the-art approaches to MIE in an orchestra setting, as demanded by PHENICX. In particular, we extended the rule-based approach for band member detection [Schedl and Widmer, 2007] to typical orchestra settings, using corresponding instrument names and roles, and we investigated its performance on web pages crawled for RCO, using queries `"Royal Concertgebouw Orchestra" members` and `"Royal Concertgebouw Orchestra" musicians`. However, as it turned out, performance was very low, due to the nature of returned web pages. In particular, this is due to the facts that (i) many more roles and sub-roles (e.g. first and second violin) exist in an orchestra setting than in a Rock band setting, (ii) there are temporal changes (e.g. current versus past members or temporary employments), and (iii) the different web sources frequently disagreed on who is a member and who is not.

Manual inspection of the crawled web pages showed that **dedicated web pages**, such as the official orchestra pages, but also **encyclopedias like *Wikipedia*, provide the most accurate results**; better than those yielded by the extended rule-based member detection approach. We hence decided to concentrate our efforts on mining a fixed set of data sources, rather than extending underperforming existing approaches.

First we aimed at extracting information from general web sources that are not explicitly created for providing information on music. One of the biggest sources for retrieving publicly available informational material is *Wikipedia*. Thorough empirical analysis has shown that in most cases *Wikipedia* offers more information than comparable sources. However, results are not necessarily related to the desired musical information, so we have to deal with **disambiguation** issues.

To obtain better structured information, we also tried to use *DBpedia*, which uses information from *Wikipedia* and presents them in a better organized way, i.e. adds structure and ontologies to the material available. However, this is true only in theory. Our experiments showed that this information is far too sparse to be used for retrieving valid text or multimedia material. Nevertheless, the ontologies can be used to gather additional disambiguation patterns and to derive heuristics for disambiguation.

3.4 Retrieving web information from Wikipedia

As stated in the DoW for Task 3.5, we start with a list of composers and pieces. As seed composers and pieces we used the repertoire of the RCO planned for the seasons 2014 and 2015, which has also been used to create the standardized corpus of supporting multimedia material, as part of D3.8.

The names of the seed composers and pieces were used as *Wikipedia* query. If there is more than one result, **disambiguation information from the *DBpedia* ontologies** and **basic heuristics** (such as taking the first page or the first page in which "music" occurs) can be used to determine the correct *Wikipedia* page. *DBpedia* is a project that extracts data from *Wikipedia* similar to the approach used for *wiki2rdf* [Meyer, 2013]. Although this approach only uses the infoboxes of *Wikipedia* to draw conclusions, it is well suited for simple tasks like disambiguation.

Disambiguation information for "Bach" on *DBpedia:*

```
<http://dbpedia.org/resource/Bach_(disambiguation)>
<http://dbpedia.org/ontology/wikiPageDisambiguates>
<http://dbpedia.org/resource/Bach_(surname)> .
<http://dbpedia.org/resource/Bach_(disambiguation)>
<http://dbpedia.org/ontology/wikiPageDisambiguates>
<http://dbpedia.org/resource/Johann_Sebastian_Bach> .
<http://dbpedia.org/resource/Bach_(disambiguation)>
<http://dbpedia.org/ontology/wikiPageDisambiguates>
<http://dbpedia.org/resource/Bach_family> .
<http://dbpedia.org/resource/Bach_(disambiguation)>
<http://dbpedia.org/ontology/wikiPageDisambiguates>
<http://dbpedia.org/resource/BACH_motif> .
<http://dbpedia.org/resource/Bach_(disambiguation)>
<http://dbpedia.org/ontology/wikiPageDisambiguates>
<http://dbpedia.org/resource/Carl_Philipp_Emanuel_Bach> .
<http://dbpedia.org/resource/Bach_(disambiguation)>
<http://dbpedia.org/ontology/wikiPageDisambiguates>
<http://dbpedia.org/resource/Johann_Christian_Bach> .
<http://dbpedia.org/resource/Bach_(disambiguation)>
<http://dbpedia.org/ontology/wikiPageDisambiguates>
<http://dbpedia.org/resource/Vincent_Bach> .
<http://dbpedia.org/resource/Bach_(disambiguation)>
<http://dbpedia.org/ontology/wikiPageDisambiguates>
<http://dbpedia.org/resource/Bach_(surname)> .
```

```
<http://dbpedia.org/resource/Bach_(disambiguation)>
<http://dbpedia.org/ontology/wikiPageDisambiguates>
<http://dbpedia.org/resource/Bach_an_der_Donau> .
<http://dbpedia.org/resource/Bach_(disambiguation)>
<http://dbpedia.org/ontology/wikiPageDisambiguates>
<http://dbpedia.org/resource/Bach,_Austria> .
<http://dbpedia.org/resource/Bach_(disambiguation)>
<http://dbpedia.org/ontology/wikiPageDisambiguates>
<http://dbpedia.org/resource/Bach,_Lot> .
<http://dbpedia.org/resource/Bach_(disambiguation)>
<http://dbpedia.org/ontology/wikiPageDisambiguates>
<http://dbpedia.org/resource/Bach_quadrangle> .
<http://dbpedia.org/resource/Bach_(disambiguation)>
<http://dbpedia.org/ontology/wikiPageDisambiguates>
<http://dbpedia.org/resource/Bach_(crater)> .
<http://dbpedia.org/resource/Bach_(disambiguation)>
<http://dbpedia.org/ontology/wikiPageDisambiguates>
<http://dbpedia.org/resource/Weesener_Bach> .
<http://dbpedia.org/resource/Bach_(disambiguation)>
<http://dbpedia.org/ontology/wikiPageDisambiguates>
<http://dbpedia.org/resource/Maybach> .
<http://dbpedia.org/resource/Bach_(disambiguation)>
<http://dbpedia.org/ontology/wikiPageDisambiguates>
<http://dbpedia.org/resource/Brown_Association_for_Cooperative_Housing> .
<http://dbpedia.org/resource/Bach_(disambiguation)>
<http://dbpedia.org/ontology/wikiPageDisambiguates>
<http://dbpedia.org/resource/Vincent_Bach_Corporation> .
<http://dbpedia.org/resource/Bach_(disambiguation)>
<http://dbpedia.org/ontology/wikiPageDisambiguates>
<http://dbpedia.org/resource/Bach_flower_remedies> .
<http://dbpedia.org/resource/Bach_(disambiguation)>
<http://dbpedia.org/ontology/wikiPageDisambiguates>
<http://dbpedia.org/resource/Bill_Bachrach> .
<http://dbpedia.org/resource/Bach_(disambiguation)>
<http://dbpedia.org/ontology/wikiPageDisambiguates>
<http://dbpedia.org/resource/Bach_(New_Zealand)> .
<http://dbpedia.org/resource/Bach_(disambiguation)>
<http://dbpedia.org/ontology/wikiPageDisambiguates>
<http://dbpedia.org/resource/Bach_(surname)> .
<http://dbpedia.org/resource/Bach_(disambiguation)>
<http://dbpedia.org/ontology/wikiPageDisambiguates>
<http://dbpedia.org/resource/Studio_apartment> .
<http://dbpedia.org/resource/Bach_(disambiguation)>
<http://dbpedia.org/ontology/wikiPageDisambiguates>
<http://dbpedia.org/resource/1814_Bach> .
```

Having found the URLs of the *Wikipedia* pages matching the given seed entities, the created methods and scripts download the respective HTML pages automatically. Furthermore, they **extract the text and parse the HTML content to identify images and audio files**. These multimedia files are then automatically downloaded. Although audio files and videos are

hardly included in general *Wikipedia* pages, for pieces and composers, multimedia content is available more frequently.

For **audio files**, this automatic multimedia retrieval can be performed as shown in the following example from a Python script:

Sample code for retrieving URLs of audio files from within a web page:

```
def store_AudioFromHTML(content, description, output_dir_root):
    if content in "":
        return []
    # ensure that output directory structure exists
    if not os.path.exists(output_dir_root):
        os.makedirs(output_dir_root)
    # parse HTML content, looking at audio links
    audio_urls = []
    tokens = content.split("<audio ")
    for tok in tokens:
        audio_tag = tok.split("</audio>")
        # get index of important attributes within tag
        idx_src =  audio_tag[0].find("src=\"")
        idx_type =  audio_tag[0].find("type=\"")
        # extract corresponding attributes
        # src
        src_attribute = audio_tag[0][idx_src+len("src=\""):]
        src_attribute = src_attribute[:src_attribute.find("\"")]
        # validity check and dimension check
        if src_attribute[0:1] in "//" and src_attribute[-4:] in ".ogg":
            audio_urls.append(src_attribute)
            url = "http:" + src_attribute
            print "Retrieving " + url + ": " + src_attribute
            before, separator, file_name = urlparse(url).path.rpartition('/')
            try:
                urllib.urlretrieve(url, output_dir_root + file_name)
            except IOError:
                print "Error retrieving " + url
    return audio_urls
```

For **images**, we applied two different approaches, taking into account the structure of *Wikipedia*. Sometimes, only smaller versions of the images are directly integrated into the web page, but larger ones are linked to. In a first step, we hence automatically analyze the size of such integrated images. If the image under consideration is sufficiently large, it is downloaded and stored in the multimedia repository. In addition to this (faster) method, we make use of the specific structure of *Wikipedia* pages to obtain the original sources in full size and typically much better quality.
We considered three different ways to retrieve the original image:

- Using the `srcset` attribute of the image. This attribute is part of a HTML5 W3C working draft[7] and used by *Wikipedia*. It lists a set of different image sources with different resolutions and consequently allows to retrieve images with higher quality.
- The `img` tag may be surrounded by a link leading to a URL. Usually the link has the format `href="/wiki/File:someFile.png"`. This URL leads to a page with all different versions of the given file. The list includes information on image sizes and on which image is the original one.
- In most cases, the description includes a tag: `<div class="magnify">`. This element also contains the link `href="/wiki/File:someFile.png"` and can be used if the image is not surrounded by a link. Usually, applying the first two approaches already leads to the original file.

For retrieving **textual information**, we parse the HTML file and remove all links, references, content tables, etc., as well as everything outside `<div id="content">`, i.e. the *Wikipedia* navigation and layout, etc. As the structure of the sections is stored, headlines provide a means of additional segmentation.

3.5 Retrieving web information from Last.fm

Wikipedia, as described in the last section, offers unstructured or semi-structured material with long texts for different purposes. As it provides information on a theoretically unlimited number of topics, it is hard to develop an overall structure that allows to retrieve topic-specific information, especially if the content is user-generated.
Contrastingly, *Last.fm* explicitly offers music-related material. This leads to some interesting new aspects. In addition to traditional web mining techniques like information extraction from tags, cf. [Hariri, Mobasher, Burke, 2013], focusing on a certain topic allows *Last.fm* to pre-define entities (e.g. the properties and relations of entities like songs and artists). This results in the possibility to provide music-specific APIs. Of course, restricting properties of music entities to the ones manually defined reduces the amount of available data. However, this approach results in a better structure. The structure itself contains valuable information and eases the semantic processing of data.

The following excerpt from our code shows how this structural information and the music-tailored API can be used to retrieve specific information on an artist, like the biography, tags added by users, pieces created by this artist and the fans on *Last.fm*:

Code excerpt for gathering artist information from *Last.fm:*

```
# retrieve various requested info via Last.fm API
# artist info
if RETRIEVE_LASTFM_ARTIST_INFO:
```

7 http://www.w3.org/html/wg/drafts/srcset/w3c-srcset/

```
        json_data_info = lastfm_artist_api_call("artist.getInfo", item, output_dir_root
+ output_dir_raw_data)
        data_info = json.loads(json_data_info)  # perform JSON processing

    # artist tags
    if RETRIEVE_LASTFM_ARTIST_TAGS:
        json_data_tags = lastfm_artist_api_call("artist.getTopTags", item,
output_dir_root + output_dir_raw_data)
        data_tags= json.loads(json_data_tags)   # perform JSON processing
        # process tags and write output
        tags_str = []       # list to hold tags and weights (to use a generic function
for writing to output file)
        tags_str.append("tag" + "\t" + "weight")
        try:
            no_tags = len(data_tags["toptags"]["tag"])
            for tag_no in xrange(0,no_tags):
                tag = data_tags["toptags"]["tag"][tag_no]["name"]
                weight = data_tags["toptags"]["tag"][tag_no]["count"]
                tags_str.append(tag + "\t" + weight)
        except KeyError:
            tags = {}
            tags_str = []
        writeToFile_info("tags", tags_str, item, output_dir_root +
output_dir_structured_data + "/" + composer_quoted + "/Last.fm/tags/")

    # artist biographies
    if RETRIEVE_LASTFM_ARTIST_BIO:
        try:
            json_data_info = lastfm_artist_api_call("artist.getInfo", item,
output_dir_root + output_dir_raw_data)
            data_info = json.loads(json_data_info)  # perform JSON processing
            # extract biographies
            bio = data_info["artist"]["bio"]["content"]
            bio = bio.split("\n\n")[0].strip()     # get rid of link to main Last.fm
artist page and trim whitespace
            # unescape (get rid of unicode HTML stuff
            parser = HTMLParser.HTMLParser()
            bio_unescaped = parser.unescape(bio)
            writeToFile_info("bio", bio_unescaped, item, output_dir_root +
output_dir_structured_data + "/" + composer_quoted + "/Last.fm/bio/")
        except KeyError, e:
            print 'KeyError while parsing JSON file - reason "%s"' % str(e)

    # artist pieces
    if RETRIEVE_LASTFM_ARTIST_PIECES:
        json_data_tags = lastfm_artist_api_call("artist.getTopTracks", item,
output_dir_root + output_dir_raw_data)
        data_pieces = json.loads(json_data_tags)   # perform JSON processing
        # process pieces and write output
        pieces_str = []       # list to hold pieces, playcount, listeners
        pieces_str.append("track" + "\t" + "duration" + "\t" + "playcount" + "\t" +
"listeners")
        try:
            no_tracks = len(data_pieces["toptracks"]["track"])
            for track_no in xrange(0,no_tracks):
                track = data_pieces["toptracks"]["track"][track_no]["name"]
                duration = data_pieces["toptracks"]["track"][track_no]["duration"]
```

```
                playcount = data_pieces["toptracks"]["track"][track_no]["playcount"]
                listeners = data_pieces["toptracks"]["track"][track_no]["listeners"]
                pieces_str.append(track + "\t" + duration + "\t" + playcount + "\t" +
listeners)
        except KeyError:
            pieces_str = []
        writeToFile_info("pieces", pieces_str, item, output_dir_root +
output_dir_structured_data + "/" + composer_quoted + "/Last.fm/pieces/")

    # fans
    if RETRIEVE_LASTFM_ARTIST_FANS:
        json_data_tags = lastfm_artist_api_call("artist.getTopFans", item,
output_dir_root + output_dir_raw_data)
        data_fans = json.loads(json_data_tags)   # perform JSON processing
        # process pieces and write output
        fans_str = []       # list to hold pieces, playcount, listeners
        fans_str.append("name" + "\t" + "realname" + "\t" + "url" + "\t" + "weight")
        try:
            no_fans = len(data_fans["topfans"]["user"])
            for fan_no in xrange(0,no_fans):
                name = data_fans["topfans"]["user"][fan_no]["name"]
                realname = data_fans["topfans"]["user"][fan_no]["realname"]
                url  = data_fans["topfans"]["user"][fan_no]["url"]
                weight = data_fans["topfans"]["user"][fan_no]["weight"]
                fans_str.append(name + "\t" + realname + "\t" + url + "\t" + weight)
        except KeyError:
            fans_str = []
        writeToFile_info("fans", fans_str, item, output_dir_root +
output_dir_structured_data + "/" + composer_quoted + "/Last.fm/fans/")
```

The information returned by the *Last.fm* API is not a web page that needs to be parsed and analyzed, but a JSON file including already structured information:

JSON file returned by *Last.fm* when querying for "Carl Maria von Weber":


```
{"artist":{"name":"Carl Maria von Weber","mbid":"c2d17829-1424-435b-9386-
c77d3a920abe","url":"http:\/\/www.last.fm\/music\/Carl+Maria+von+Weber","image":[{"#text
":"http:\/\/userserve-
ak.last.fm\/serve\/34\/71369818.png","size":"small"},{"#text":"http:\/\/userserve-
ak.last.fm\/serve\/64\/71369818.png","size":"medium"},{"#text":"http:\/\/userserve-
ak.last.fm\/serve\/126\/71369818.png","size":"large"},{"#text":"http:\/\/userserve-
ak.last.fm\/serve\/252\/71369818.png","size":"extralarge"},{"#text":"http:\/\/userserve-
ak.last.fm\/serve\/500\/71369818\/Carl+Maria+von+Weber.png","size":"mega"}],"streamable"
:"0","ontour":"0","stats":{"listeners":"123083","playcount":"543827"},"similar":{"artist
":[{"name":"Antonio
Salieri","url":"http:\/\/www.last.fm\/music\/Antonio+Salieri","image":[{"#text":"http:\/
\/userserve-
ak.last.fm\/serve\/34\/70896890.png","size":"small"},{"#text":"http:\/\/userserve-
ak.last.fm\/serve\/64\/70896890.png","size":"medium"},{"#text":"http:\/\/userserve-
ak.last.fm\/serve\/126\/70896890.png","size":"large"},{"#text":"http:\/\/userserve-
ak.last.fm\/serve\/252\/70896890.png","size":"extralarge"},{"#text":"http:\/\/userserve-
ak.last.fm\/serve\/500\/70896890\/Antonio+Salieri++1750++1825.png","size":"mega"}]},{"na
me":"Jules
Massenet","url":"http:\/\/www.last.fm\/music\/Jules+Massenet","image":[{"#text":"http:\/
\/userserve-
```

```
ak.last.fm\/serve\/34\/83465931.jpg","size":"small"},{"#text":"http:\/\/userserve-
ak.last.fm\/serve\/64\/83465931.jpg","size":"medium"},{"#text":"http:\/\/userserve-
ak.last.fm\/serve\/126\/83465931.jpg","size":"large"},{"#text":"http:\/\/userserve-
ak.last.fm\/serve\/252\/83465931.jpg","size":"extralarge"},{"#text":"http:\/\/userserve-
ak.last.fm\/serve\/500\/83465931\/Jules+Massenet+jules_massenet2.jpg","size":"mega"}]},{
"name":"Carl
Stamitz","url":"http:\/\/www.last.fm\/music\/Carl+Stamitz","image":[{"#text":"http:\/\/u
serserve-
ak.last.fm\/serve\/34\/46513657.jpg","size":"small"},{"#text":"http:\/\/userserve-
ak.last.fm\/serve\/64\/46513657.jpg","size":"medium"},{"#text":"http:\/\/userserve-
ak.last.fm\/serve\/126\/46513657.jpg","size":"large"},{"#text":"http:\/\/userserve-
ak.last.fm\/serve\/252\/46513657.jpg","size":"extralarge"},{"#text":"http:\/\/userserve-
ak.last.fm\/serve\/_\/46513657\/Carl+Stamitz+Stamitz.jpg","size":"mega"}]},{"name":"C\u0
0e9sar
Franck","url":"http:\/\/www.last.fm\/music\/C%C3%A9sar+Franck","image":[{"#text":"http:\
/\/userserve-
ak.last.fm\/serve\/34\/98867213.png","size":"small"},{"#text":"http:\/\/userserve-
ak.last.fm\/serve\/64\/98867213.png","size":"medium"},{"#text":"http:\/\/userserve-
ak.last.fm\/serve\/126\/98867213.png","size":"large"},{"#text":"http:\/\/userserve-
ak.last.fm\/serve\/252\/98867213.png","size":"extralarge"},{"#text":"http:\/\/userserve-
ak.last.fm\/serve\/_\/98867213\/Csar+Franck.png","size":"mega"}]},{"name":"Johann
Nepomuk
Hummel","url":"http:\/\/www.last.fm\/music\/Johann+Nepomuk+Hummel","image":[{"#text":"ht
tp:\/\/userserve-
ak.last.fm\/serve\/34\/11151669.jpg","size":"small"},{"#text":"http:\/\/userserve-
ak.last.fm\/serve\/64\/11151669.jpg","size":"medium"},{"#text":"http:\/\/userserve-
ak.last.fm\/serve\/126\/11151669.jpg","size":"large"},{"#text":"http:\/\/userserve-
ak.last.fm\/serve\/252\/11151669.jpg","size":"extralarge"},{"#text":"http:\/\/userserve-
ak.last.fm\/serve\/_\/11151669\/Johann+Nepomuk+Hummel+Hummel.jpg","size":"mega"}]}]},"ta
gs":{"tag":[{"name":"classical","url":"http:\/\/www.last.fm\/tag\/classical"},{"name":"r
omantic","url":"http:\/\/www.last.fm\/tag\/romantic"},{"name":"opera","url":"http:\/\/ww
w.last.fm\/tag\/opera"},{"name":"german","url":"http:\/\/www.last.fm\/tag\/german"},{"na
me":"classic","url":"http:\/\/www.last.fm\/tag\/classic"}]},"bio":{"links":{"link":{"#te
xt":"","rel":"original","href":"http:\/\/www.last.fm\/music\/Carl+Maria+von+Weber\/+wiki
"}},"published":"Fri, 19 Nov 2010 04:26:58 +0000","summary":"\n                Carl
Maria Friedrich Ernst, Freiherr von Weber (18th November 1786\u20135th June 1826) was a
German composer, conductor, pianist, and critic, one of the first significant composers
of the Romantic school.  Weber's works, especially his operas Der Freisch\u00fctz,
Euryanthe, and Oberon, greatly influenced the development of the Romantic opera in
Germany. He was also an innovative composer of instrumental music.\n\n        <a
href=\"http:\/\/www.last.fm\/music\/Carl+Maria+von+Weber\">Read more about Carl Maria
von Weber on Last.fm<\/a>.\n    \n    ","content":"\n                    Carl Maria
Friedrich Ernst, Freiherr von Weber (18th November 1786\u20135th June 1826) was a German
composer, conductor, pianist, and critic, one of the first significant composers of the
Romantic school.  Weber's works, especially his operas Der Freisch\u00fctz, Euryanthe,
and Oberon, greatly influenced the development of the Romantic opera in Germany. He was
also an innovative composer of instrumental music.\n\n        <a
href=\"http:\/\/www.last.fm\/music\/Carl+Maria+von+Weber\">Read more about Carl Maria
von Weber on Last.fm<\/a>.\n    \n    \nUser-contributed text is available under the
Creative Commons By-SA License and may also be available under the GNU FDL.\n    "}}}
```


Some pieces of information, like the biography of the artist, can directly be extracted from the JSON file. In addition to this textual information, linked multimedia material can be downloaded. For many images, different versions with different quality and resolution exist.

Heuristics like evaluating the size attribute of the respective JSON object, or analyzing the file URLs for contained size information, allows us to select the best image even before downloading it, which makes the method efficient.

3.6 Retrieving web information from Freebase

Among the selected sources, *Freebase* offers the most structured information. *Freebase* relies on a very **strong, generic, and sophisticated ontology**, which links all pieces of information semantically. Every item within their database is part of a semantic net and has semantic relations to other concepts, categories, and items.

As a matter of fact, our methods to retrieve web information from *Freebase* make use of this additional information. Contrary to the previously introduced methods, **we do not need an initial set of seed entities** like composers, pieces, or instruments, because e.g. "composer" exists as a concept within the *Freebase* ontology, which increases the precision of queries.

The *Freebase* API provides a query language called MQL, which allows to search for items matching a certain topic/category. Therefore, the scripts we developed to acquire pieces of information from *Freebase* can easily be used for any entity, returning all results from *Freebase* without having to deal with disambiguation problems.

It is sufficient to provide an entity type as input parameter, and all items of this type are automatically processed. The following code example shows how this information is downloaded and analyzed for additional images, which are then automatically and separately downloaded as well. By default, *Freebase* offers only thumbnail images and shrinks the original files. However, the thumbnail size can be overridden and if the requested size is at least as big as the original file, the original file is returned in full size and resolution.

Code snippet for retrieving data from *Freebase:*

```
query = [{PROPERTY_ID: None, 'name': None, 'type': TYPE_OF_OBJECTS_TO_RETRIEVE, 'limit':
MAX_ITEMS}]
params = {
        'query': json.dumps(query),
        'key': api_key
}
url = service_url + '?' + urllib.urlencode(params)
print url

response = json.loads(urllib.urlopen(url).read())

if not os.path.exists(PATH_FOR_CRAWLS):
  os.makedirs(PATH_FOR_CRAWLS)

if not os.path.exists(PATH_FOR_IMAGES):
  os.makedirs(PATH_FOR_IMAGES)
```

```
count = 0
service_url2 = 'https://www.googleapis.com/freebase/v1/topic'
flst_out = open("./"+NAME_OF_OBJECTS_TO_RETRIEVE+".txt","a")
flst_out_img = open("./"+NAME_OF_OBJECTS_TO_RETRIEVE+"_images.txt","a")

for res_object in response['result']:
  if res_object['name'] is None:
    continue
  count += 1
  str_count = str(count)

  line = str_count + '\t' + res_object[PROPERTY_ID] + '\t' + res_object['name'] + '\n'
  line = line.encode('utf-8')
  flst_out.write(line)

  topic_id = res_object[PROPERTY_ID]
  print str_count+ ': ' + res_object['name'] + ' - ' + res_object[PROPERTY_ID]
  params = {
    'key': api_key,
    'filter': 'all'
  }

  url = service_url2 + topic_id + '?' + urllib.urlencode(params)
  data = urllib.urlopen(url).read()
  fobj_out = open(PATH_FOR_CRAWLS + "/" + str_count + ".json","w")
  fobj_out.write(data)
  fobj_out.close()
  objectdata = json.loads(data)["property"]
  str_img_count = "0"

  if "/common/topic/image" in objectdata:
    images = objectdata["/common/topic/image"]["values"]
    img_count = 0
    for img in images:
      img_count += 1
      str_img_count = str(img_count)
      url = "https://www.googleapis.com/freebase/v1/image" + img["id"] + "?maxwidth=" +
MAXIMUM_IMAGE_WIDTH + "&key=" + api_key
      try:
        urllib.urlretrieve(url, PATH_FOR_IMAGES + "/img_" + str_count + "_" +
str_img_count + ".jpg")
        line = str_count + '\t' + str_img_count + '\t' + img['id'] + '\t' + img['text']
+ '\n'
        line = line.encode('utf-8')
        flst_out_img.write(line)
      except IOError:
        print "Error retrieving " + url
  print str_count + ": " + str_img_count + " images"

flst_out.close()
flst_out_img.close()
```

As this method uses the entity as the input parameter, it no longer relies on a given set of composers, pieces, performers, or instruments. This is especially useful as it helps to **detect new entities** that might not have been included in the original input files of music entities. If

the results should be restricted to match certain conditions, this can easily be done using the information provided by the semantic net. In any case, the exploitation of semantic information helps to retrieve and detect entities that are maybe not commonly known or included in predefined lists. For instance, searching for instruments, we could retrieve results for 1455 items. Our scripts retrieve JSON files as shown in the example below – in this case for the instrument Chitravina, an instrument that is probably not well known in the Western world.

Code excerpt of the JSON file for the instrument "Chitravina":


```
{
 "id": "/m/0546g0",
 "property": {
  "/common/topic/article": {
   "valuetype": "compound",
   "values": [
    {
     "text": "The chitravina (also known as chitra veena, chitraveena, chitra vina,
hanumad vina, or...",
     "lang": "en",
     "id": "/m/0546g5",
     "creator": "/user/mwcl_wikipedia_en",
     "timestamp": "2006-10-23T02:00:05.007Z",
     "property": {
      "/common/document/source_uri": {
       "valuetype": "uri",
       "values": [
        {
         "text": "http://wp/en/1472231",
         "lang": "",
         "value": "http://wp/en/1472231",
         "creator": "/user/mwcl_wikipedia_en",
         "timestamp": "2006-10-23T02:00:05.007Z"
        }
       ],
       "count": 1.0
      },
      "/common/document/text": {
       "valuetype": "string",
       "values": [
        {
         "text": "The chitravina (also known as chitra veena, chitraveena, chitra vina,
hanumad vina, or...",
         "lang": "en",
         "value": "The chitravina (also known as chitra veena, chitraveena, chitra vina,
hanumad vina, or mahanataka vina, is a 20 or 21-string fretless lute for Carnatic music
played mainly in South India today, though its origins can be traced back to Bharata's
Natya Shastra, where it is mentioned as a 7 string fretless instrument. It has undergone
numerous developments and is today among the more prominent solo instruments in Carnatic
music. It is also often seen in collaborative world music concerts and north-south
Indian jugalbandis.\nAround late 1800s and early 1900s, it had been bestowed another
name - "Gotuvadyam", Tamil:                    ) (often misspelt as gottuvadyam,
gottuvadhyam, kottuvadyam etc.) by Sakha Rama Rao, who was responsible for bringing it
back to the concert scene. The fretless nature of the instrument makes it the closest
instrument to vocal standards. There are six main strings used for melody that pass over
```

```
the top of the instrument, three drone strings, and about twelve sympathetic strings
running parallel and below the main strings.\nThe approach to tuning is in some ways
similar to the sitar; in other ways it is similar to the Saraswati veena, but in many
ways it is",
         "creator": "/user/mwcl_wikipedia_en",
         "timestamp": "2006-10-23T02:00:05.007Z"
        }
       ],
       "count": 1.0
      },
      "/type/object/attribution": {
       "valuetype": "object",
       "values": [
        {
         "text": "Freebase Data Team",
         "lang": "",
         "id": "/m/0gs8",
         "creator": "/user/mwcl_wikipedia_en",
         "timestamp": "2006-10-23T02:00:05.007Z"
        }
       ],
       "count": 1.0
      },
      "/type/object/type": {
       "valuetype": "object",
       "values": [
        {
         "text": "Document",
         "lang": "",
         "id": "/common/document",
         "creator": "/user/mwcl_wikipedia_en",
         "timestamp": "2006-10-23T02:00:05.007Z"
        }
       ],
       "count": 1.0
      }
     }
    }
   ],
   "count": 1.0
  },
  "/common/topic/description": {
   "valuetype": "string",
   "values": [
    {
     "text": "The chitravina often spelt as gottuvadyam, gottuvadhyam, kottuvadyam
etc.), which was bestowed...",
     "lang": "en",
     "value": "The chitravina often spelt as gottuvadyam, gottuvadhyam, kottuvadyam
etc.), which was bestowed upon it by Sakha Rama Rao from Thanjavur, who was responsible
for bringing it back to the concert scene. Today it is played mainly in South India,
though its origins can be traced back to Bharata's Natya Shastra, where it is mentioned
as a seven string fretless instrument.",
     "creator": "/user/wikirecon_bot",
     "project": "wikirecon",
     "dataset": "/m/0kj4zz_",
     "citation": {
```

```
      "provider": "Wikipedia",
      "statement": "Description licensed under the Creative Commons Attribution-
ShareAlike License
(http://en.wikipedia.org/wiki/Wikipedia:Text_of_Creative_Commons_Attribution-
ShareAlike_3.0_Unported_License).",
      "uri": "http://en.wikipedia.org/wiki/Gottuvadhyam"
     }
    }
   ],
   "count": 3.0
  },
  "/common/topic/image": {
   "valuetype": "object",
   "values": [
    {
     "text": "Ravikiran 25 A",
     "lang": "en",
     "id": "/m/0bn3y85",
     "creator": "/user/mwcl_images",
     "timestamp": "2011-07-31T13:28:49.001Z"
    },
    {
     "text": "Ch2",
     "lang": "en",
     "id": "/m/0h5dhbq",
     "creator": "/user/mwcl_images",
     "timestamp": "2011-09-22T03:03:36.000Z"
    }
   ],
   "count": 2.0
  },
  "/common/topic/notable_for": {
   "valuetype": "object",
   "values": [
    {
     "text": "Musical instrument",
     "lang": "en",
     "id": "/music/instrument"
    }
   ],
   "count": 1.0
  },
  "/common/topic/notable_types": {
   "valuetype": "object",
   "values": [
    {
     "text": "Musical instrument",
     "lang": "en",
     "id": "/music/instrument"
    }
   ],
   "count": 1.0
  },
  "/common/topic/topic_equivalent_webpage": {
   "valuetype": "uri",
   "values": [
    {
```

```
     "text": "http://es.wikipedia.org/wiki/Chitra_vina",
     "lang": "",
     "value": "http://es.wikipedia.org/wiki/Chitra_vina",
     "creator": "/user/wikirecon_bot",
     "timestamp": "2013-03-16T00:53:11.001Z"
    },
    {
     "text": "http://sv.wikipedia.org/wiki/Gottuvadhyam",
     "lang": "",
     "value": "http://sv.wikipedia.org/wiki/Gottuvadhyam",
     "creator": "/user/wikirecon_bot",
     "timestamp": "2013-03-16T00:53:30.003Z"
    },
    {
     "text": "http://es.wikipedia.org/wiki/index.html?curid=4650603",
     "lang": "",
     "value": "http://es.wikipedia.org/wiki/index.html?curid=4650603",
     "creator": "/user/wikirecon_bot",
     "timestamp": "2013-03-16T00:53:41.000Z"
    },
    {
     "text": "http://sv.wikipedia.org/wiki/index.html?curid=357207",
     "lang": "",
     "value": "http://sv.wikipedia.org/wiki/index.html?curid=357207",
     "creator": "/user/wikirecon_bot",
     "timestamp": "2013-03-16T00:53:44.003Z"
    },
    {
     "text": "http://en.wikipedia.org/wiki/index.html?curid=1472231",
     "lang": "",
     "value": "http://en.wikipedia.org/wiki/index.html?curid=1472231",
     "creator": "/user/wikirecon_bot",
     "timestamp": "2013-05-25T19:39:28.008Z"
    },
    {
     "text": "http://en.wikipedia.org/wiki/Gottuvadhyam",
     "lang": "",
     "value": "http://en.wikipedia.org/wiki/Gottuvadhyam",
     "creator": "/user/wikirecon_bot",
     "timestamp": "2013-05-25T19:40:01.003Z"
    }
   ],
   "count": 6.0
  },
  "/freebase/object_profile/object_generation_time": {
   "valuetype": "datetime",
   "values": [
    {
     "text": "2014-05-19T18:50:21.000Z",
     "lang": "",
     "value": "2014-05-19T18:50:21.000Z"
    }
   ],
   "count": 1.0
  },
  "/music/instrument/family": {
   "valuetype": "object",
```

```
   "values": [
    {
     "text": "Plucked string instruments",
     "lang": "en",
     "id": "/m/0fx80y",
     "creator": "/user/carmenmfenn1",
     "timestamp": "2009-03-05T09:41:32.012Z"
    }
   ],
   "count": 1.0
  },
...
```


[File truncated for this deliverable.]

The tools we developed automatically **parse the returned JSON files**. They extract the textual descriptions as well as references to additional multimedia material. Having extracted the IDs (and consequently the URLs) of the linked images, the crawlers fetch the original files for the respective multimedia items. They are automatically downloaded and the meta-information (e.g. the description of the images) is extracted and stored in the database. For the example of the Chitravina, the following two images were found:

In several cases, as in the given example, *Freebase* links to *Wikipedia* content. In these cases, the actual retrievable material is the same as for the *Wikipedia* crawls, but as it is better structured than the *Wikipedia* data, it still offers additional information, which makes this method an extraordinary valuable source of information.

3.7 Storage and use of the web information

All pieces of information and multimedia items we retrieved applying the described methods allowed us to set up different corpora of music-related multimedia content including descriptions of different lengths, images, and audio files for the various musical entities mentioned in the description of Task 3.5, i.e. composers, performers, pieces, and instruments. The resulting multimedia database provides the material required for the user study that was conducted as a joint endeavor for the D3.5 at hand and for D5.4. The acquired data is available to all partners upon request.

4 EVALUATION OF RELEVANCE AND QUALITY OF THE RETRIEVED MATERIAL

Having retrieved the multimedia contents from different web sources, as elaborated on in the previous section, the questions arose, whether the developed methods work, whether the automatically gathered pieces of information are correct, and which of these materials are suited to be presented to users, in terms of quality.

Freebase: image returned for "Cello".

At this point, it is important to note that there is obviously no "ground truth" for the task at hand. Addressing the questions mentioned above, we hence performed two types of assessments: a **quantitative investigation** using simple heuristics, and a **qualitative assessment**; the latter including (i) **manual inspection** by the authors and (ii) a **crowd-sourced questionnaire**.

The first, quantitative investigation included a set of rules and ensured, for instance, that images likely to be in bad quality are removed. To this end, we implemented several heuristics, tailored to the different data sources.
We could either retrieve actual image sizes (in pixels), categories of image sizes (small, medium, large, extra-large) or make sure that no thumbnail is returned. Especially for cases with several versions of the same picture, these assessments are important to retrieve the best version (i.e. the one with the highest resolution).

For the qualitative assessment, we **manually inspected the retrieved contents**, i.e. the extracted pieces of text as well as the multimedia files. All materials we manually checked were related to the given queries. However, the nature of the retrieved materials was quite different for the different sources. For instance, biographies and descriptions retrieved from *Last.fm* and *Freebase* are concise and of comparable length for most composers. *Wikipedia* instead provides textual descriptions of strongly varying lengths, depending on popularity and other factors.

Wikipedia: one of the 16 images for cello. The image caption reads "Rosin is applied to bow hairs to increase the "bite" of the bow on the strings."

The same is true for images. *Freebase* usually offers up to 3 images per entity, which are very closely related to the given query. *Wikipedia*, in contrast, offers strongly varying amounts of images, which in some cases might not be as closely connected to the topic. For instance, for "cello" there is only a single image provided by *Freebase*. However, this image shows quite well what we expect from a picture of a cello. *Wikipedia* instead provides 16 different images, including images of different types of cellos, chords, informational graphics on the parts of a cello, cellists, or even images related to the usage of a cello and the materials required for maintaining it, as shown in the second picture of the current section. This photograph, standing on its own, is probably not the best one to describe a cello, but in addition to the text and other images, it may provide interesting additional information.

As for the text modality, a short and concise description gives the reader a good first impression of a music entity, whereas a long text provides additional information, likely of high value someone interested in details. We assume that different users prefer different types of information with different levels of granularity and detail.

To verify this assumption, we performed a second qualitative assessment, a **crowd-sourced questionnaire**. This questionnaire, implemented via *Mechanical Turk*,[8] was a joint endeavor, relating to both the deliverable at hand D3.5 and D5.4.

For D3.5, the aim of the study was to **determine whether there exist general preferences** towards the following information categories:

1. **length or amount** of information (e.g. text), or number of items (e.g. images)
2. **modality** of material: text, image, and audio
3. **entity** of the item: composer, piece, performer, and instrument

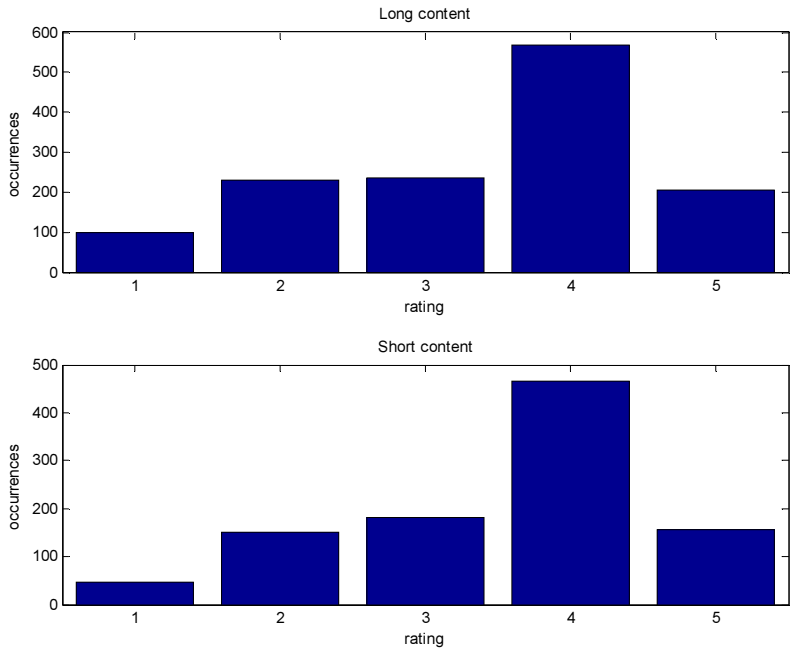
For D5.4, the questionnaire aimed at providing the data necessary to cluster users according to specific, individual preferences, given the categories mentioned above.

In total, 167 users participated in the questionnaire, which yielded statistically significant results. Users were asked for feedback on whether they regarded the presented material (i.e. the material gained through the methods described in the current deliverable) as (i) **interesting** and (ii) **providing novel information**. Furthermore, they could provide free-form feedback, a function used surprisingly frequently to indicate positive feelings towards the experiment and the music in general.

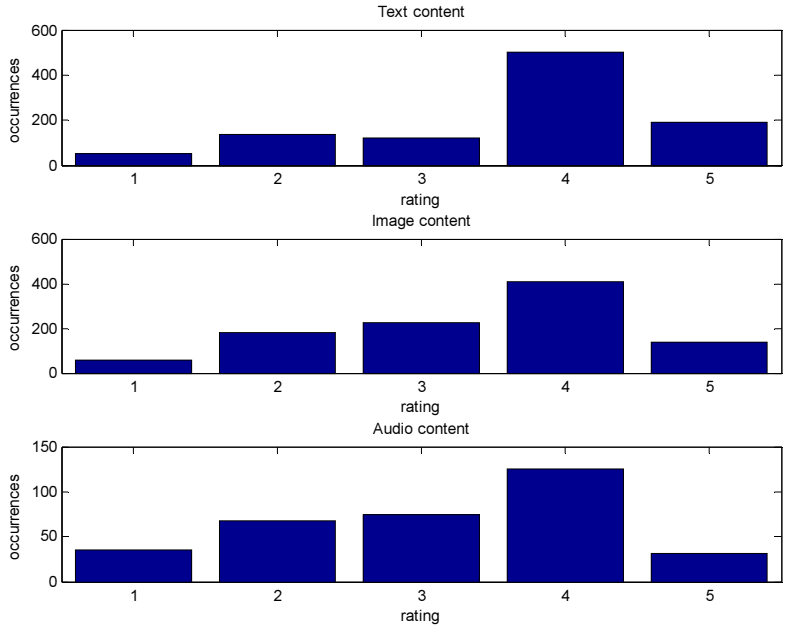
The material, grouped based on the length or amount (long/short or few/many items), modality (text/image/audio) and entity (composer/piece/orchestra), was rated by the subjects relating the statement »I find the content interesting« a Likert rating from 1 to 5 (stemming from »I strongly disagree to »I strongly agree«). Analyzing the results of the study showed no

[8] https://www.mturk.com

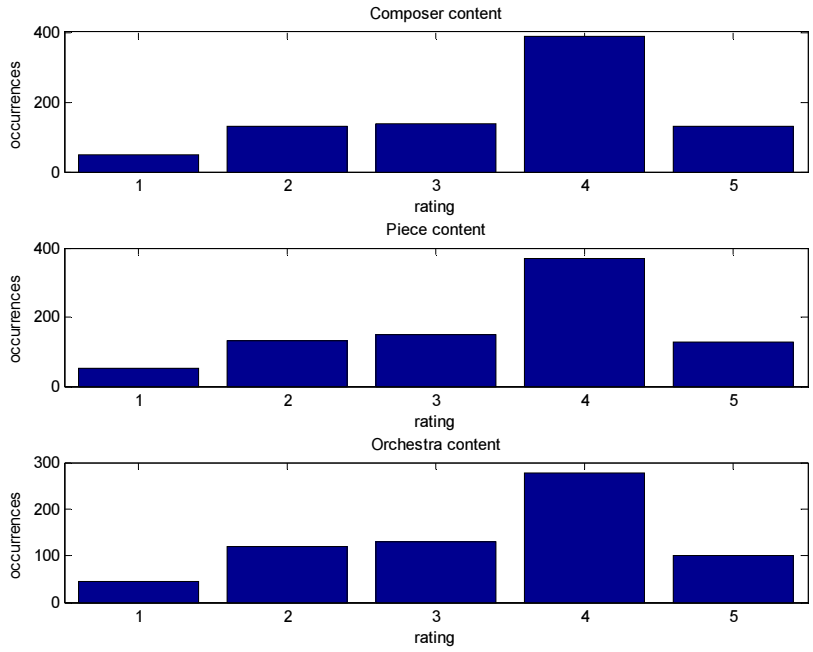
substantial differences in the distribution of the ratings among different groups. Most of the users gave a rating of 4 (i.e. »I agree«), as shown in the following histograms of the results:



Histograms of ratings (from 1 to 5) to the question how interesting the presented material is, analyzed for short vs. long content.



Histograms of ratings (from 1 to 5) to the question how interesting the presented material is, analyzed for various categories of multimedia (text, image, audio).



Histograms of ratings (from 1 to 5) to the question how interesting the presented material is, analyzed for various entities (composer, piece, performer).

The means and standard deviations of the ratings for each group are as follows, and show no statistically significant differences in the ratings. Solely, and surprisingly, users generally tend to prefer textual information over audio and image. However, we hypothesize that users anyways expect to hear music in a user study about Classical music; so this category of information becomes less important.

Length/Amount:	
Long:	3.41 +- 1.16
Short:	**3.53** +- 1.07
Modality:	
Text:	**3.65** +- 1.09
Image:	3.38 +- 1.10
Audio:	3.14 +- 1.17
Entity:	
Composer:	**3.51** +- 1.11
Piece:	3.46 +- 1.12
Performer:	3.41 +- 1.14

These results show that there are **no general preferences towards any of the categories of information**, which in turn means that different users prefer different types of material. It is hence necessary to store all kinds of information (on various entities, in various lengths/amounts, and in various modalities) in order to provide each user with the desired information, offering a personalized experience.

5 CONCLUSION

For the task addressed in the deliverable D3.5 at hand, i.e. Task 3.5., we developed a set of **methods to automatically retrieve multimedia items (text, image, and audio) on music entities, such as piece, composer, performer, and instrument**. Contrary to state-of-the-art approaches to Music Information Extraction (MIE), which are tailored exclusively to identifying textual pieces of information, we presented methods and heuristics to gather multimedia data from generic web pages as well as from specifically music-related sources and online catalogues, making use of **ontologies, structures, and semantic information** contained. Hence, our methods are not limited to textual information, such as composer's biographies; in addition, **multimedia material is automatically identified, downloaded, tagged, and categorized**. This material is used for the corpus of musical multimedia content that is continuously being extended as a part of WP3, and will eventually form D3.9 ("final corpus of supporting multimedia material").

To summarize the work conducted for this deliverable, we list the main tasks performed:

- Investigating the state-of-the-art in Music Information Extraction (MIE)
- Implementing and adapting state-of-the-art methods to detect members of a performing body (e.g. violin players of RCO), and evaluation thereof
- Identifying various web resources that provide multimedia material on composers, performers, pieces, and instruments
- Developing methods to automatically acquire pieces of information on the musical entities, from the general sources represented by *Wikipedia/DBpedia*, *Last.fm*, and *Freebase*. This includes elaborating disambiguation techniques and quality assessment techniques in order to identify relevant and high-quality material.
- Running the methods on input data from RCO and ESMUC (partially reported in deliverables D3.7 and D3.8) as well as on the level of music entities, when using *Freebase*.
- Conducting several quantitative and qualitative validation experiments: statistical figures of the gathered pieces of information (cf. D3.7 and D3.8), manual inspection, and crowd-sourced questionnaire.

Summing up we conclude that web resources can be categorized according to the amount of data available, the level of detail, the type of multimedia material (text, audio, image), and the semantic information provided. We presented methods that make use of different sources in order to set up a musical database providing different multimedia content for pieces, composers, performers, and instruments. We applied these methods to gather a diverse set of material, enabling user-adaptive information provision, which is one central goal of WP5. We further showed that our methods are able to extract semantically relevant information of different kinds.

6 REFERENCES

6.1 Written references

[Govaerts and Duval, 2009]: *A Web-based Approach to Determine the Origin of an Artist*. Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR), Kobe, Japan, October 2009.

[Hariri, Mobasher, Burke, 2013]: *Personalized Text-Based Music Retrieval*. Workshops at the Twenty-Seventh AAAI Conference on Artificial Intelligence, 2013.

[Knees and Schedl, 2011]: *Towards Semantic Music Information Extraction from the Web Using Rule Patterns and Supervised Learning*. Proceedings of the 2nd Workshop on Music Recommendation and Discovery (WOMRAD), Chicago, IL, USA, October 2011.

[Krenmair, 2010]: *Musikspezifische Informationsextraktion aus Webdokumenten*, MSc thesis, Johannes Kepler University Linz, Austria, 2010.

[Meyer, 2013]: *wiki2rdf: Automatische Extraktion von RDF-Tripeln aus Artikelvolltexten der Wikipedia*. Information - Wissenschaft & Praxis. 64(2-3): 69-172, 2013.

[Schedl, Knees, Pohle, Widmer, 2006]: *Towards Automatic Retrieval of Album Covers*. Proceedings of the 28th European Conference on Information Retrieval (ECIR), London, UK, April 2006.

[Schedl and Widmer, 2007]: *Automatically Detecting Members and Instrumentation of Music Bands via Web Content Mining*. Proceedings of the 5th Workshop on Adaptive Multimedia Retrieval (AMR), Paris, France, July 2007.

[Schedl, Seyerlehner, Schnitzer, Widmer, Schiketanz, 2010]: *Three Web-based Heuristics to Determine a Person's or Institution's Country of Origin*. Proceedings of the 33th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR), Geneva, Switzerland, July 2010.

[Schedl, Widmer, Knees, Pohle, 2011]: *A music information system automatically generated via web content mining techniques*. Information Processing & Management, 47, 2011.

[Schedl and Tkalčič; 2014]: *Genre-based Analysis of Social Media Data on Music Listening Behavior: Are Fans of Classical Music Really Averse to Social Media?*. First International Workshop on Internet-Scale Multimedia Management (ISMM 2014) @ ACM Multimedia (submitted), November 2014, Orlando, FL, USA.

6.2 Acronyms and abbreviations

DoW	Description of Work
MIE	Music Information Extraction
PoS	Part-of-Speech (tagging)
SVM	Support Vector Machine