Galler-eCollection: An Online Mashup Game of Art

Chao Yan¹, Dong Han¹, Haimin Lu¹

¹University of Southern California, Computer Science Department
Los Angeles, CA, 90089
{chaoyan, han275, haiminlu}@usc.edu

Abstract. Galler-eCollection is an online multiplayer mashup game which is
aimed to provide people an entertaining way to learn and appreciate great art-
works hiding in the museums all over the world. It is a real mashup application
because it is built upon different data sources which are extracted, cleaned, in-
tegrated and can be easily queried by the application program. We first use Py-
thon program to crawl the data from three art museum websites. Then use Kar-
ma [6], Google Refine and Python scripts to clean and integrate our art data into
RDF triple store. To extend our data sources, we use SILK to link the artist data
with DBpedia [4], and use GeoNames.org [5] to geo-tag each artist with his/her
active location. Our mashup game runs fluently and can be scaled up to incor-
porate multiple new data sources.

Keywords: mashup game, artwork collecting, data cleaning, data integration,
RDF, record linkage, geo-tagging, SPARQL.

1 Introduction

Most people have a desire to know and appreciate great artworks from historic artists.
However, world-famous art museums seem to be far from people’s daily life, and
museum websites are so boring for people who prefer more entertaining websites. We
designed an online multiplayer mashup game for people to know thousands of art-
works from world-famous museums and to enjoy themselves by playing game. Every-
one, to some extent, has a desire to collect valuable artworks. The game creates a
virtual environment where players can collect virtual artworks which come from real
artworks displaying on museum websites. Players need to read the text hints of each
artwork carefully and match the picture of artwork to its text in order to collect an
artwork. They can also review the artworks they collected. The locations (e.g. birth
place, active place) of all the related artists are visualized on the Google map. Our
online mashup game needs multiple data sources which are distributed over the World
Wide Web. In our project, there are five data sources (See section 4) used for the
game application. They are cleaned, integrated as RDF and stored in triple store
which support SPARQL queries for application program. To geo-tagging each artist
with the longitude and latitude of his/her location, we call the GeoNames.org API to
get the associated geo data. Section 2 & 3 introduces the game functions and require-
ments. Section 4 describes the process of data extraction, data cleaning and data inte-
gration for the game. Section 5 demonstrates the architecture of our mashup game. Section 6 gives a brief evaluation of our work and points out the future work.

## 2 The Mashup Game

Galler-eCollection is a web-based mashup game. After signing in the game, players can see the main page of game (Fig.1). Users are initially assigned 3 albums. By selecting one “album” (e.g. “album1”, “album2”), players will see an album consist of 4 artworks. All the artworks are initially left blank but with clues (description, title, artist name and material). Players’ task is to match the artwork images given in “bag” with the corresponding artworks in albums. Players will be given a new “album” if they match all artworks in the “album”. Game is somewhat challenging because few irrelevant images are mixed in the bag. The “pool” is a public place where all users can throw unwanted artwork images from “bag” and drag desired images to “bag”. Players can review the artworks they have finished. Also the active location of artists related to the matched artworks will be visualized on Google map. The information of the artists will also be pinned on the map. Each artwork in new “album” is randomly generated from the global space of datasets. As players’ level grows up, they need to match more artworks in an “album” and complete more albums to level up. Players will enjoy their achievement and are motivated to discover more great artworks.

![Main page of Galler-eCollection](image)

Fig. 1 Main page of Galler-eCollection

## 3 Requirement

- Extensible data sources
Our online mashup game is built upon different data sources, and it is supposed to extend endlessly with more and more data sources being added. New data sources should be added independently from the web game service. And they should be queried in a uniform way.

- User experience

Mashup game should also ensure the data to be usable for game players. Since the data are extracted from multiple data sources (multiple art museums across the world), they may have missing data, error data and heterogeneous schemas. Data should be highly cleaned and integrated for the use of game. (e.g. players don’t want to see a blank attribute of artwork title)

- Performance

Querying the online RDF triple store incurs the cost of network latency. It would be a bad experience if every operation from player needs to query the triple store. A local database should be used to store cached data as well as dynamic data of players such as game progress.

4 Data Source Building

4.1 Data cleaning phase

There are three main art data sources that we extract: Getty museum [1], IMA museum [2] and OMCA collection [3]. In the web crawling phase, we used Python and Python “BeautifulSoup” package to extract more than 12,000 artworks and 3000 artist data. The retrieved datasets are quite heterogeneous: different schemas, different names for the same attribute, different data format (e.g. different representations of date, location and full name), missing data, incorrect data, ambiguous data and unstructured data. We faced a bunch of data cleaning problems, and now we describe some of them. One problem is that extracting birth-death date and birth-death location from unstructured short-bio snippet (i.e. to structure the biographic information). We first use Google Refine to separate birth information and death information by a common pattern that birth data always come before death data. Next, we wrote Python scripts along with regular expression to extract useful data from short bio snippets, and refined them into more structured fields: birth date, death date, birth location, death location. By this step there are still some irregular data, we used Karma transformation function to clean the data intelligently. The second challenge is relevant to string matching problem, i.e. same artist with different name presentation. We solved 46% string mess by tackling names and taking another relevant attributes into consideration using Google refine. Another thing we need to do is to add an additional enumerable artwork ID field which would be used by the game program for generating random artworks to players.
4.2 Data integration

In the data integration phase, we referred to the artwork source description given in class and built our own source description which is suitable for our datasets. Then we used Karma to turn our previous data into RDF triple store with ontologies defined by OWL. Moreover, in order to extend the information about artists, we established record linkage between our triple store and DBpedia (See Fig. 2).

![Fig. 2. Using Silk to build record linkage between our RDF triple store and DBpedia](image)

Then we geo-tagged each artist with the longitude and latitude comes from GeoNames.org. We queried the GeoNames ontology with artists’ specific location information. However, the problem is that most artists only have people information (e.g. American, Spanish). We solve this by building a mapping from people to its country name. Finally, we had all the required cleaned information from multiple sources integrated in a single RDF triple store which is stored in OpenRDF Sesame server. It executes SPARQL queries sent from game server.

5 Architecture

Fig. 3 shows the overall architecture of the application. Our mashup game is built upon REST web service which is implemented by server-side JavaScript (Node.js engine). Node.js is based on asynchronous request handling mode, which can scale up to handle multiple concurrent operations from players. Information related to players such as game progress and personal information are stored in MongoDB. MongoDB is built on JSON documents storage and provides NoSQL query, which supports efficient user data storage and retrieval. Large volumes of artworks and artists information are storing in RDF triple store (a Sesame server provided in class), the game program needs to send SPARQL query to the Sesame server to get data of artworks and artists. For the SPARQL query, the most common query is to get artwork and artist information for given artwork IDs. In order to reduce response time, the game application would send a batch of artwork IDs to the Sesame server and get results back. Since the server will encounter large numbers of user requests for artwork and
artist information, the query result got from Sesame server will be cached into MongoDB for the improvement of performance. Benefiting from the NoSQL feature of MongoDB, the schema of each datasets can have different number of attributes, which simplified the data integration in the local DB.

6 Evaluation and Future Work

The mashup game provides players abundant or even endless artworks to discover, to learn, and to enjoy. Most data are cleaned well and can be correctly displayed in game. And most artists are correctly pinned on the Google map with their active location. Data sources can be scaled up—if we add a new dataset from a new art museum, we do not need to modify the application and we simply clean and load it to our triple store. Our future work is to clustering the artworks by genre or material. Then instead of generating a artworks randomly, we generate artworks from the same genre cluster in to one “album”. And we will distribute our RDF triple store over the internet, so as to reduce the load of each Sesame server.

References

1. http://www.getty.edu/art/