Image search based on a broker approach

Pere Torán, Jaime Delgado

Universitat Politècnica de Catalunya (UPC)
{ptoran, jaime.delgado}@ac.upc.edu

Abstract. This paper proposes a new way of searching resources, specifically images, through the web, following a broker approach. After studying some of the most important image repositories, Panoramio, Picasa and Flickr have been chosen as the resource databases where the application looks for images. Apart from the central application, which implements the search engine, a web portal has been developed in order to provide a user interface front-end where users can perform some searches in a friendly way.

Keywords: Metadata interoperability, broker, image search.

1 Introduction

Nowadays, there are many web sites that provide a huge amount of resources. Each resource has some metadata associated, which changes from one server to another. This causes some interoperability problems between each metadata model that makes difficult performing searches against different servers.

Our goal is to implement an image searcher that can handle, in the best possible way, the negative effects of this lack of interoperability, following a broker approach.

This paper is organized as follows: Section 2 is an overview of the state-of-the-art in the use of some standard metadata models and languages to request multimedia content, and in the use of broker approaches. Section 3 is an overview of which modules integrate the whole system we have developed and how they interact with each other.

The conclusions and future work are shown in section 4.

2 State-of-the-Art

Because this is not a new problem, a lot of research work has been already done on trying to solve it. This implies, from designing a broker architectural approach or metasearches [1][4][5] to specifying standard metadata models and languages [2][6][8], to accept and respond to requests for multimedia content without the concern about where they come from [3].

2.1 Designing a Broker Architectural Approach

There are many approaches to try to solve the interoperability problem [1]. One of them is based on a broker, also called query rewriting because its job is to receive search queries in a determined metadata format and to rewrite them once for every metadata format that is supported.

After the queries have been generated, they are sent to every external server and the resulting data is processed again and mapped back to the original metadata format.

This means that it is necessary to use a metadata model in order to establish some translations, or mappings, from this model to the rest.

2.2 Working on Standardizing a Multimedia Metadata Model

There is not a metadata model considered to be the standard to use. There are several models each one of them focusing on different needs and kinds of resources.

For example, the MPEG-7 standard is a complex model that aims to describe audiovisual material such as video, sound or images [6].

Another example is the Dublin Core model, which is oriented to multimedia and text content and whose core is composed by only 15 metadata elements, but has been highly extended by some working groups [8].

The Dublin Core model is widely used in many environments, such as the eXtensible Metadata Platform [9], which is a standard for processing and storing standardized and proprietary information relating to the content of a file. This is used by the Media Annotations Working Group [7], who wants to provide an ontology and an API designed to facilitate cross-community data integration of information related to media objects in the Web, such as video, audio and images.

An interesting model that has been recently standardized is the JPSearch Core, which is the metadata core of the JPSearch framework, which wants to provide a way to decouple the in general tightly coupled systems and to provide a better interoperability during image search [2].

JPSearch goal is to provide a standard for interoperability for image search and retrieval, which matches our objective.

2.3 Defining a Language to Request for Multimedia Content

The MPEG working group has defined a language, which became an ISO/IEC standard in December 2008, to accept and respond to requests for multimedia contents. It is called MPQF (MPEG Query Format) [3].

MPQF is an XML-based query language that defines the format of queries and responses to be interchanged between clients and servers in a distributed multimedia information search-and-retrieval context. It provides interoperability between parties, e.g. clients and content providers, and platform independency.

A subset of MPQF is also used in the JPSearch part 3, which is the part that provides a standardized message protocol for image retrieval; it is called JPQF (JPSearch Query Format).

3 The Developed System

The software we have developed provides a centralized place for searching images from different images servers. In our current version we are using Panoramio, Picasa and Flickr.

Our system is compliant with the current version of the JPSearch standards. From an architectural point of view, the system follows a broker architectural approach (see section 2.1), which means that there is a subsystem that receives JPQF queries in a metadata format and rewrites them once for every metadata format that is supported (Panoramio, Picasa and Flickr). This is sketched in Figure 1.

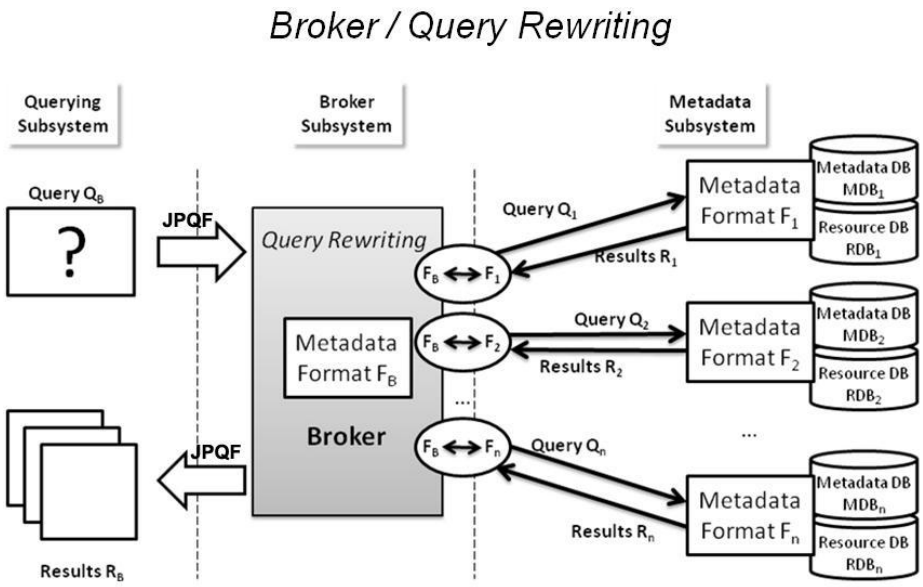

Fig 1. A broker architectural approach

The system is split into two modules: a central application that implements a broker-based metadata aggregator and a web portal as the user front-end. Modules communicate each other by using JPQF queries.

To understand how they all work together we first need to understand what each module does.

3.1 Central Application

This module takes as the input a JPQF query referring to metadata elements compliant with the JPSearch Core Schema (defined in ISO/IEC 24800-2) and returns a JPQF response.

3.1.1. Metadata Model

As mentioned before, the subsystem receives a query in a metadata format and rewrites it once for every metadata format that is supported. So, one metadata model must be used in order to establish the appropriated translations between that model and the rest.

The JPSearch Metadata Core has been chosen to be the default metadata model and has been extended with some image specific fields and with some EXIF fields, like the camera make and model.

The example in Figure 2 illustrates the model used by the application through a real search result. All the image dependent metadata fields like width, height or EXIF are included in the Photo element. Whenever it is possible, the other fields are grouped by meaning, for example, the Name and Nick fields, which are related with the Author, are included in the Author element.

```
<Metadata>
  <Title>Prenent un Bany. Taking a Bath.</Title>
  <Description>Bigui. beagle.</Description>
  <Keywords>landscape,yellow,orange,groc,taronja,Catalunya</Keywords>
  <URI>http://www.flickr.com/photos/arturdebat/1382400520/</URI>
  <Source>Flickr</Source>
  <License>1</License>
  <Author>
    <Name>Artur Debat</Name>
    <Nick>¡arturii!</Nick>
  </Author>
  <Date>
    <Created>2007-09-10 16:21:34</Created>
    <Published>2007-09-14 20:45:50</Published>
  </Date>
  <Location>
    <Latitude>41.5475158691406</Latitude>
    <Longitude>2.39347410202026</Longitude>
  </Location>
  <Photo>
    <URI>
      <Source>http://farm3.static.flickr.com/2/1892.jpg</Source>
      <Thumbnail>http://farm3.static.flickr.com/2/1892t.jpg</Thumbnail>
    </URI>
    <Width>500</Width>
    <Height>333</Height>
    <Orientation>1</Orientation>
    <EXIF>
      <Make>Canon</Make>
      <Model>Canon EOS 400D DIGITAL</Model>
    </EXIF>
  </Photo>
</Metadata>
```

Fig 2. Example of the metadata model in XML format.

3.1.2. The Broker Interface

In order to search on each server, an interface per-server must be provided. Each interface, depending on what kind of searches it receives, must know if it is possible to search on its associated server and, if so, how to launch the search and what to do with the results later.

There is a simple way to decide whenever it is possible or not to search on a server just knowing which metadata fields have their results and which of them it accepts to perform a search.

Knowing how to launch a search against a server is not only limited by using its API but, in order to expand the kind of searches that its server accepts, to properly modify the original query, for example by removing some conditions before launching it and then filtering the results by the conditions previously excluded.

For example, imagine that somebody wants to search images from Barcelona higher than 200 pixels in a server that does not allow searching by height. This search constraint will not be considered at searching time but after getting the results from Barcelona they will be reprocessed by height greater than 200 pixels.

3.1.3. Reprocessing Results

The reprocess can hence be thought as a filtering process that is done when there are search constraints that are not supported by a server.

This way of looking for images increases the search options that servers offer by default, but, on the other hand, from a user point of view, some searches can look slow because, internally, the application repeatedly launches several searches and reprocesses the results until all constraints are resolved, and that can take a while depending on how restrictive the given constraints are.

The time that these searches take can be minimized by scattering them across threads and launching them in parallel.

In order to be able to reprocess content obtained as the result of a search, they must be temporary kept in some way, for example inserted in a database. Since there is no need to keep them after being returned to the user, an in-memory database engine has been used in order to improve the efficiency.

3.1.4. Implementing a JPQF Interpreter

Our Web portal and the central application communicate each other by using JPQF queries but, internally, the application, which is programmed with Java, works with a set of Java classes that represent the JPQF query, not with JPQF expressed on XML.

This means that some kind of translation must be done from a JPQF query to the internal set of classes and vice versa. So, an interpreter that parses and transforms each incoming JPQF query must be implemented.

Since there is nothing to parse when the opposite transformation is realized, it is easier than the previous one. The interpreter only has to be aware of the JPQF query's OutputDescription, which is a part of the JPQF query, and to specify how the results

will be returned, in order to properly construct the JPQF response. For example, it is possible to specify which fields will be returned with the results, in which order they will be shown or how many of them will be returned. This is done by iterating each result's metadata field set and filling them into the JPQF.

3.2 Web Portal

The web portal is form-based and allows users to search images at different places without being concerned of how they are implemented as well as specifies some display options like sorting or the maximum number of results by page. When a user submits the form, a JPQF query representing the specified search and the display constraints is created and is sent to the application.

3.2.1 Another JPQF Interpreter

As well as we did before, a JPQF interpreter must be implemented on the client side in order to translate from HTML to JPQF and vice versa.

The HTML to JPQF translation can be done in a similar way the metadata model to JPQF translation does. Every form field becomes almost directly a JPQF condition that, concatenated with the rest using a properly Boolean operator (AND, OR), conforms the whole JPQF condition set.

Because JPQF is XML-based, the opposite translation, JPQF to HTML, can be accomplished just applying an XSLT on the JPQF response.

3.2.1 Web Design

The web is form-based, which means there are some input fields, which users can fill in with text, each one of them represents a different search field.

There is some javascript code that adds some dynamic behaviour to the search form. For example, because the form is too big, it is divided by sections (e.g. date, location, etc.) and each one of them can be collapsed resulting in a smaller form.

Also, it can be duplicated, as it is shown in figure 3, allowing launching multi-queries. In fact, they will be all assembled with an OR operand and sent to the application using a single JPQF query, but the application's JPQF interpreter will split it into several searches, one by each form, and the broker will launch them in parallel.

The search results are presented to the user as a table of thumbnails, as figure 4 shows. In order to see the information associated to each result, users must click on the appropriated thumbnail. A pop-up similar to which is shown in figure 5 will appear.

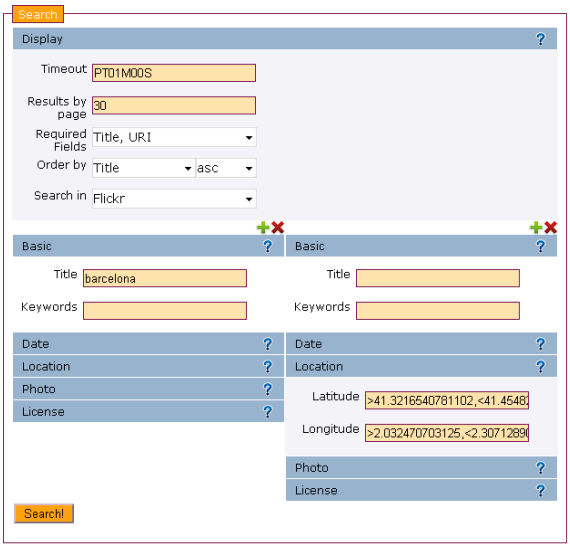

Fig 3. Example web form

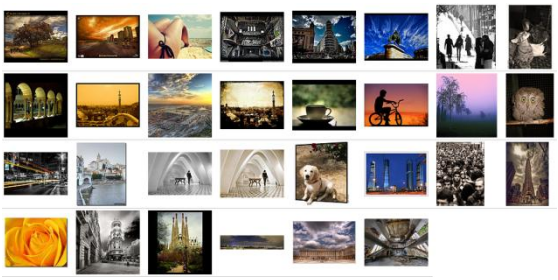

Fig 4. Result set shown in graphic mode

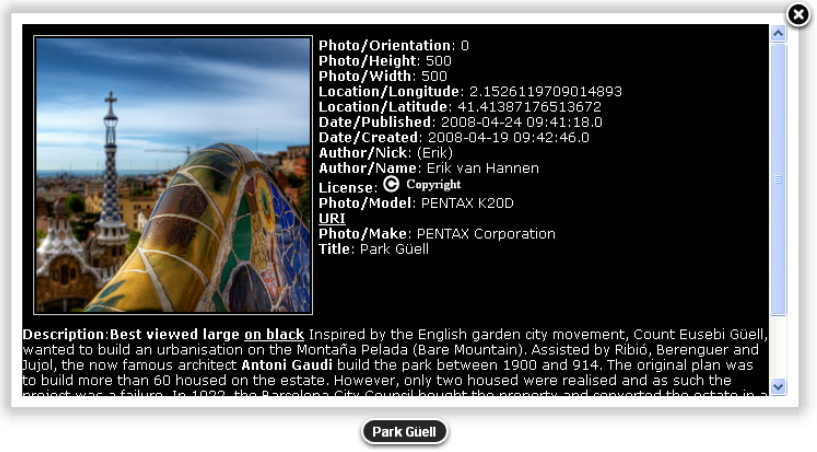

Fig 5. Result item in graphic mode

4 Conclusions and Future Work

This paper has described the design and the implementation of an image searcher that follows a broker architectural approach using some JPSearch standard parts like its metadata core and its language to retrieve images.

The main objective has been to be able to launch queries over several public image servers independently of the metadata elements used by them.

This system has been presented at the last ISO/IEC JTC1 SC29/WG1 [10] (JPEG) meeting in March 2010 and has been selected as an application to promote the use of JPEG standards. Nevertheless, our approach may work with different metadata and querying standards. For example, we have also available an implementation using the query standard from MPEG: the MPQF.

For the near future we will extend the application in order to support more content servers and more content types like audio or video.

5 Acknowledgments

This work has been partly supported by the Spanish government through the project MCM-LC (TEC 2008-06692-C02-01).

6 References

1. Toni Sala, Jaime Delgado, Rubén Tous: “Possible approaches for metadata interoperability in audiovisual content search”. DMAG Internal Report (2010). Available at http://dmag.ac.upc.edu/
2. JPSearch Metadata Core, http://www.jpeg.org/
3. MPEG Query Format standard, http://www.mpqf.org/
4. Rubén Tous, Jaime Delgado: “Advanced Meta-search of News in the Web”. Proceedings of the 6th International ICCC/IFIP International Conference on Electronic Publishing held in Karlovy Vary, Czech Republic, 6–8 November 2002.
5. Rubén Tous, Jaime Delgado, “Interoperability adaptors for distributed information search on the web”. Proceedings of the 7th ICCC/IFIP International Conference on Electronic Publishing held at the Universidade do Minho, Portugal, 25-28 June 2003.
6. MPEG-7 standard, http://mpeg.chiariglione.org/standards/mpeg-7/mpeg-7.htm
7. Media Annotations Working Group, http://www.w3.org/2008/WebVideo/Annotations/
8. The Dublin Core Metadata Initiative, http://dublincore.org/
9. eXtensible Metadata Platform, http://www.adobe.com/products/xmp/
10. Jaime Delgado, Ruben Tous, Pere Toran, Toni Sala: “Demonstration of two JPSearch (ISO/IEC 24800) Real Applications”, ISO/IEC JTC1 SC29/WG1 document wg1n5316, March 2010. Available at http://dmag.ac.upc.edu/