Information Retrieval - Submission Report
Project N.14: Art For Sale

Jeferson Morales Mariciano, Filippo Piloni
Università della Svizzera italiana, Faculty of Informatics, Lugano, Switzerland

Contents

1 Introduction

The Information Retrieval course project *"ArtForSale"* aims to be a working prototype of an information retrieval system for a specific task: to display for-sale artworks from art selling related websites listed in Section 1.1. The system has around $\approx 8,500$ indexed documents served through a user-friendly and intuitive interface, further details in UI/UX Section 6.1, 7.1.

1.1 Websites

The following websites were chosen for the project based on availability of data, ease of scraping and low restrictions regarding policies and terms of use to scrape documents i.e. *robots.txt* rules:

- www.artsy.net

- www.artfinder.com

- www.saatchiart.com

The total cumulative number of documents gathered from the above websites is $\approx 8,500$ stored in *results.json* file containing the list of documents in JSON format. Further information concerning the scraping process and implementation is discussed in Crawler Section 3.

1.2 Features

For this project, the following features has been implemented:

- Result presentation

- Filtering

- Automatic recommendation

2 Technologies Used

2.1 Scrapy

Scrapy, a Python framework for web crawling and scraping, is utilized in to initiate the crawling/scraping process for a specified set of artworks. The extracted data is subsequently channeled into a .json file.

2.2 PyTerrier

PyTerrier is a Python library that provides a high-level interface for the Terrier information retrieval system. In the provided code snippet, PyTerrier is used for building an information retrieval system to search and index documents.
Our index.py file uses PyTerrier to create the index starting from the .json file

2.3 Django & SQLite

The main part of the project has been carried using a Django based backend. The backend handles the main fetches from the frontend. The backend is also the joint between the frontend and the index, making calls to a FASTApi that returns relevant documents given a query.
To return to the frontend the main information about the documents, the backend iterates through a database in SQLite that contains the main information for each artwork

2.4 Vue.js & PrimeVue

Finally, to develop the frontend, it has been used Vue.js, using the PrimeVue as UI library. Vue.js allows us to create a reactive and dynamic user interface, while PrimeVue provides us with a set of pre-built components and styles.

3 Crawling

4 Indexing

5 Backend

The backend relies on a SQLite database to store the main information about the documents, making possible to retrieve the relevant ones following a call to the index.
Django serves as the foundational framework for constructing the primary structure, bringing with it a multitude of advantages such as rapid development, a robust and secure architecture, built-in administrative features, and a thriving community support ecosystem.
The backend handles the two main fetches of the application:

- /api/documents/get-documents/"query"/: This endpoint facilitates the retrieval of relevant documents based on a provided query string.

- /api/recommendation/get-recommended/: This endpoint is tasked with retrieving recommended artworks. By inputting a set of tags, it conducts a search within the index, returning all pertinent documents associated with the specified criteria.

6 Frontend

The frontend is the main part of the application, where all the features are implemented.
For the interface, it has been chosen a simple layout, consisting of only a search-bar and a search button. Once the user performs the search, the aspect of the page changes, adding new components. We will examine each of them:

- First, under the search-bar will appear a new set of control to filter the obtained results according to four main characteristics: site of origin, minimum price, maximum price and tag.

- Second, there will be shown some carousels for the recommended results based on the most frequents tags of the resulted artworks. Essentially, once all the artworks related to the query have been retrieved, the frontend extracts from all these artworks the related tags, it chooses the ones with the higher frequency and, using the /api/recommendation/get-recommended/ fetch, it shows a maximum of 5 results related to each tag in a moving carousel

- Finally, the artworks retrieved from the query are displayed in a card-like style, showing an insight of the artwork, the author, the title, the price, a short description and the main tags, with a Visit button that will bring the user to the page of that artwork in the original site

6.1 User Interface

7 User Evaluation

7.1 User Experience

8 Running information

9 Conclusions