ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

A DIGITAL CATALOGUE OF JERUSALEM'S AYYUBID AND MAMLUK ARCHITECTURES

EPFL DHLAB

Author: Youssef Mamlouk

Supervisor: Beatrice Vaienti
Professor: Pr. Frederic Kaplan

9th June 2023

ABSTRACT

This report presents a comprehensive approach to creating a digital catalog of Jerusalem's Ayyubid and Mamluk architectures.

The project utilizes computer vision, machine learning, natural language processing, and database management techniques to convert the book *Mamluk Jerusalem: An Architectural Study* into a digital database. The database is then visualized through an interactive leaflet map, enabling users to explore the buildings and their historical information.

The project showcases the successful extraction of architectural elements, accurate OCR results, and efficient database management.
The digital catalog serves as a valuable resource for researchers, enthusiasts, and preservationists interested in studying and appreciating the architectural heritage of Jerusalem.

Contents

CHAPTER 1

INTRODUCTION

1.1 CONTEXT

Jerusalem, the ancient and historically significant city, has been a treasure trove of architectural wonders from different periods in history.
Among these, the Ayyubid and Mamluk eras stand out as a testament to the rich cultural and artistic heritage of the region. These periods, spanning from the 12th to the 16th centuries, witnessed the construction of numerous architectural masterpieces that shaped the city's skyline and influenced architectural styles throughout the Islamic world.
Preserving and studying these architectural gems is of paramount importance to our understanding of Jerusalem's historical development and its cultural significance.

1.2 MOTIVATION

The study and documentation of architectural heritage have traditionally relied on manual methods, such as extensive field surveys, handwritten notes, and photographs.
While these approaches have provided valuable insights, they often suffer from limitations in terms of scalability, accessibility, and ease of analysis. In the case of the Ayyubid and Mamluk architectures of Jerusalem, there is a wealth of information locked away in books and documents that are not easily searchable or accessible for in-depth analysis.
Therefore, there is a pressing need to leverage modern technologies, such as computer vision, Machine Learning, Natural Language Processing, and database management to transform this valuable information into a digital format and make it readily available for researchers, scholars, and the general public.

1.3 GOALS

The primary goal of this project is to create a comprehensive and interactive digital catalogue of Jerusalem's Ayyubid and Mamluk architectures. By leveraging computer vision techniques, we aim to extract architectural plans, drawings, photographs, and textual descriptions from the book *Mamluk Jerusalem: An Architectural Study.* (Burgoyne, Richards and Archaeology in Jerusalem 1987)

Through optical character recognition (OCR), layout recognition, and NLP algorithms, we will convert these images and texts into structured and searchable data. Furthermore, by utilizing machine learning algorithms, we will enhance the accuracy and efficiency of the data extraction process.

Additionally, we will develop a robust and scalable database management system to store, organize, and manage the extracted data. This will facilitate easy access, efficient retrieval, and cross-referencing of architectural elements, styles, and historical contexts. Finally, we will leverage data visualization techniques to create an intuitive and interactive interface that allows users to explore the digital catalogue, visualize architectural features, and delve into the historical narratives embedded within Jerusalem's Mamluk architectures.

By achieving these goals, this project aims to bridge the gap between traditional architectural studies and modern technological advancements. The resulting digital catalogue will provide an invaluable resource for researchers, historians, architects, and anyone interested in the architectural heritage of Jerusalem. It will enable new avenues of research, facilitate comparative analysis, and contribute to the preservation and understanding of Jerusalem's rich cultural history.

CHAPTER 2

METHODS

2.1 LAYOUT RECOGNITION AND OCR

The first step in transforming the acquired book into a digital catalogue involved the identification and extraction of the article's structure and layout. This subsection outlines the process followed to achieve this goal.

2.1.1 STRUCTURE IDENTIFICATION

Initially, the book was obtained in a numeric scanned version, which presented the challenge of accurately identifying the structure and layout of the articles within its pages. Understanding the organization of the content was crucial for subsequent steps, such as extracting images, captions, and text blocks.

To tackle this task, an analysis of the book's structure and patterns was conducted. By visually examining the scanned pages, recurring patterns and common layouts were identified. These patterns included the arrangement of images, text blocks, headings, and other visual elements within the pages.

2.1.2 LAYOUT MODEL TRAINING

Various approaches were explored to automate the layout recognition process.

TRANSKRIBUS (*Public Models in Transkribus* 2021)

Transkribus, a widely used tool for text recognition in historical documents, was initially considered for layout recognition.

Transkribus employs state-of-the-art machine learning techniques, such as convolutional neural networks (CNNs) and long short-term memory (LSTM) networks, to recognize and transcribe text from scanned documents.

In the case of the book's layout, Transkribus struggled to accurately identify and differentiate between images, captions, headings, and other visual components. The complexity and variation in layout structures posed a challenge to its inherent capabilities.

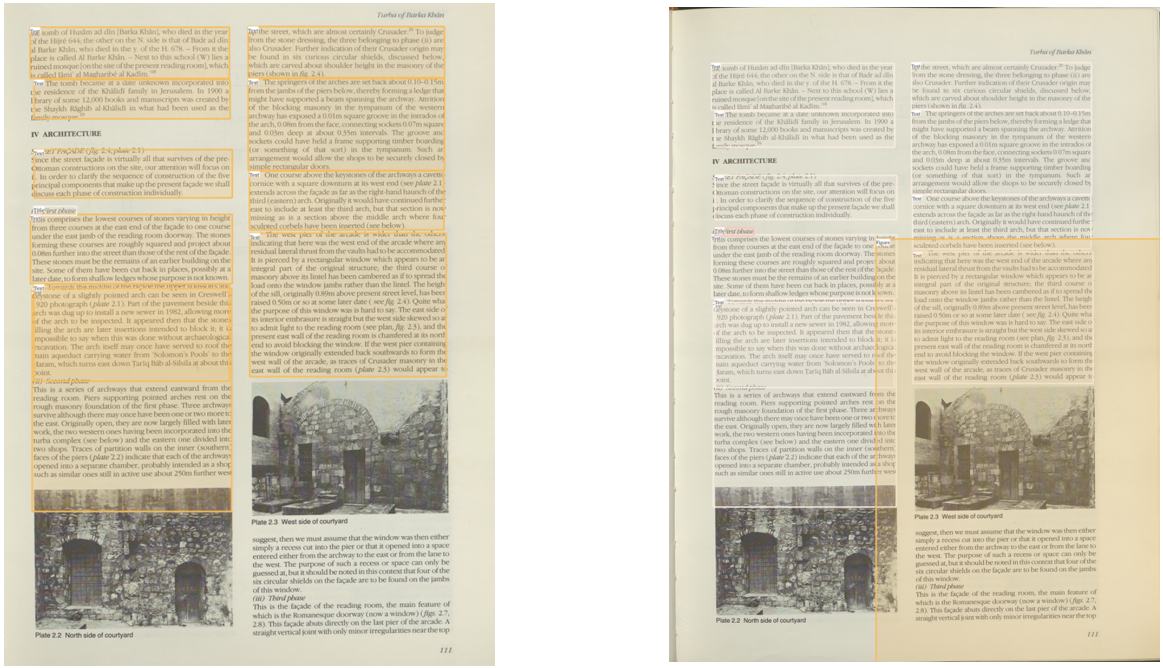
While Transkribus proved valuable for text recognition tasks, its limitations in layout recognition prompted the exploration of alternative methods. By transitioning to a more tailored solution, the goal was to achieve accurate and comprehensive extraction of the book's layout elements, including both textual and non-textual components.

LAYOUT PARSER

To address the limitations encountered with Transkribus, the Layout Parser library was adopted as an alternative approach for layout recognition.

Layout Parser (Shen et al. 2021) is a Python library that provides a comprehensive set of tools and functionalities specifically designed for Deep Learning Based Document Image Analysis. It allows for the detection, extraction, and interpretation of various layout elements within scanned documents.

In the context of our book's layout recognition, the Layout Parser library offers a collection of pre-trained models on different datasets among which we chose PubLayNet. On top of being the largest dataset ever for document layout analysis, PubLayNet was the most adapted one to the book's structure. Among the models PubLayNet has been trained on, we chose Faster R-CNN: a deep convolutional network used for object detection.



(A) Confidence: 0.8

(B) Confidence: 0.4

FIGURE 2.1
Layout Recognition with Faster R-CNN trained on PubLayNet

The **Figure 2.1** shows that The results Layout Recognition with Faster R-CNN trained on PubLayNet. At a confidence threshold of 0.8, images were not even recognized, leading to incomplete or missing annotations. Similarly, at a confidence threshold of 0.4, the recognition of text and images was poor, resulting in inaccurate or imprecise bounding box placements. These limitations highlight the challenges faced in accurately identifying and extracting the layout using a pre-trained model.

As we will see later, training a layout model using additional annotated data was necessary for the model to learn the specific layout patterns and structures present in the book.

The goal was to accurately identify and extract different layout components, such as images, captions, headings, and text blocks, from the scanned book pages.

LABEL STUDIO

Label Studio (Tkachenko et al. 2020-2022) is an open-source data annotation tool that provides a user-friendly interface allowing manual annotation of layout elements on images. A total of 30 pages were selected, each page loaded individually into the annotation tool.

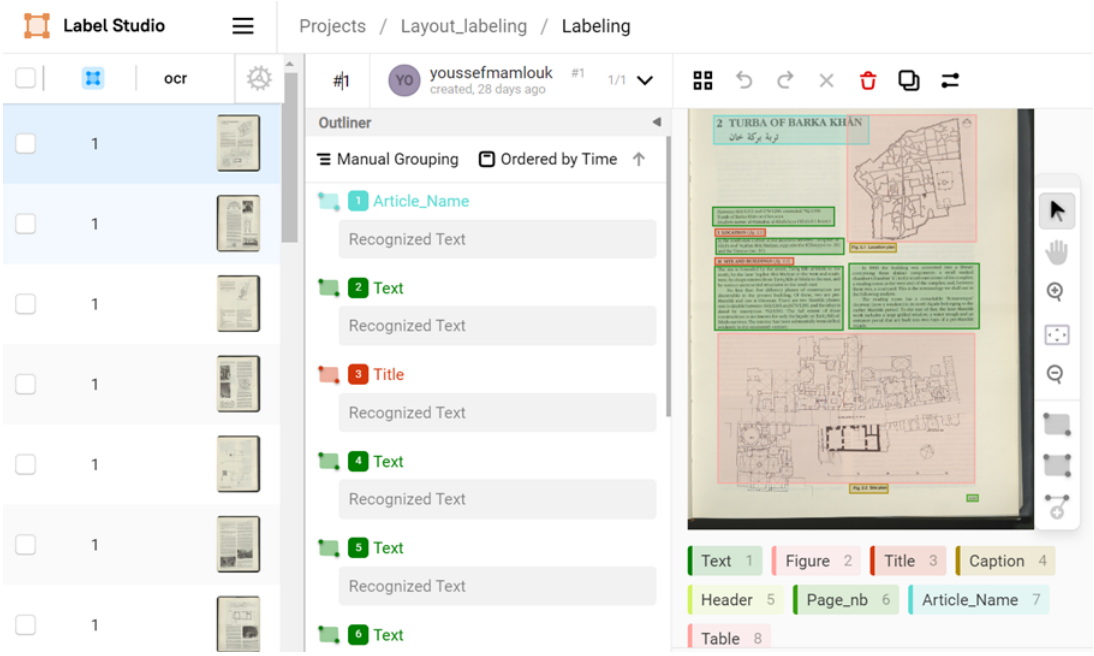


FIGURE 2.2
Manual annotation in Label Studio

As we can see in the **Figure 2.2**, we were able to mark bounding boxes around various block components. The chosen labels for annotation included:

- "Article_Name"
- "Text"
- "Title"
- "Caption"
- "Figure"
- "Header"
- "Page_nb"
- "Table"

We wanted to be as precise as possible to facilitate data processing later.

The annotation process required careful attention to detail, consistency to ensure accurate labeling, and a good understanding of the book's structure and layout to minimize errors and inconsistencies in the annotations.

Label Studio allowed for more fine-grained control and customization of the layout recognition process, better suited to the book's specific layout complexities. The annotated data was then exported in a COCO JSON (Lin et al. 2014), a specific JSON structure (popular in Machine Learning) dictating how labels and metadata are saved for an image dataset, for further processing and training of the layout recognition model.

TRAINING

The annotated data was split into two sets: a training set and a test set. The training set contained 85% of the annotated pages, while the test set consisted of the rest used for evaluation.

The Layout Parser library, along with the annotated data, was used to train the layout recognition model. The model was trained using the already pre-trained weights of PubLayNet/Faster R-CNN.

The training workflow consisted thus of fine-tuning the pre-trained model using the annotated data.

fast_rcnn/cls_accuracy	0.030273	0.036133	0.586914	...	0.987305	0.987305	0.990234	0.990234
fast_rcnn/false_negative	0.000000	0.000000	0.697083	...	0.015625	0.023438	0.015625	0.015625
fast_rcnn/fg_cls_accuracy	0.043103	0.147613	0.003906	...	0.984375	0.976562	0.984375	0.984375
iteration	0.000000	5.000000	39.000000	...	47699.000000	47719.000000	47739.000000	47759.000000
roi_head/num_bg_samples	454.000000	192.000000	198.250000	...	192.000000	192.000000	192.000000	192.000000
roi_head/num_fg_samples	58.000000	64.000000	57.750000	...	64.000000	64.000000	64.000000	64.000000
rpn/num_neg_anchors	217.000000	208.750000	211.250000	...	209.250000	205.500000	210.750000	209.750000
rpn/num_pos_anchors	39.000000	47.250000	44.750000	...	46.750000	50.500000	45.250000	46.250000
data_time	NaN	0.005414	0.004684	...	0.005703	0.005341	0.005709	0.005871
loss_box_reg	NaN	0.888959	0.808452	...	0.067150	0.067301	0.066970	0.059139
loss_cls	NaN	2.706443	1.814849	...	0.033690	0.029952	0.025339	0.026815
loss_rpn_cls	NaN	0.731621	0.546455	...	0.000198	0.000309	0.000380	0.000351
loss_rpn_loc	NaN	0.134820	0.115320	...	0.011165	0.014941	0.013248	0.012330
lr	NaN	0.000001	0.000010	...	0.000250	0.000250	0.000250	0.000250
time	NaN	1.571938	1.516439	...	1.384454	1.380754	1.381420	1.447890
total_loss	NaN	4.480943	3.106154	...	0.113467	0.113073	0.103984	0.097361
eta_seconds	NaN	NaN	37177.125169	...	46582.566943	46368.903291	46190.626935	46043.261088

FIGURE 2.3
Training metrics over 47759 iterations

As we can see in the **Figure 2.3**, this process was typically performed over multiple iterations for about 20 hours, where the model's parameters were updated based on the annotated data. Each epoch involved feeding the training data through the network, computing the loss, and updating the model's parameters to improve its performance.

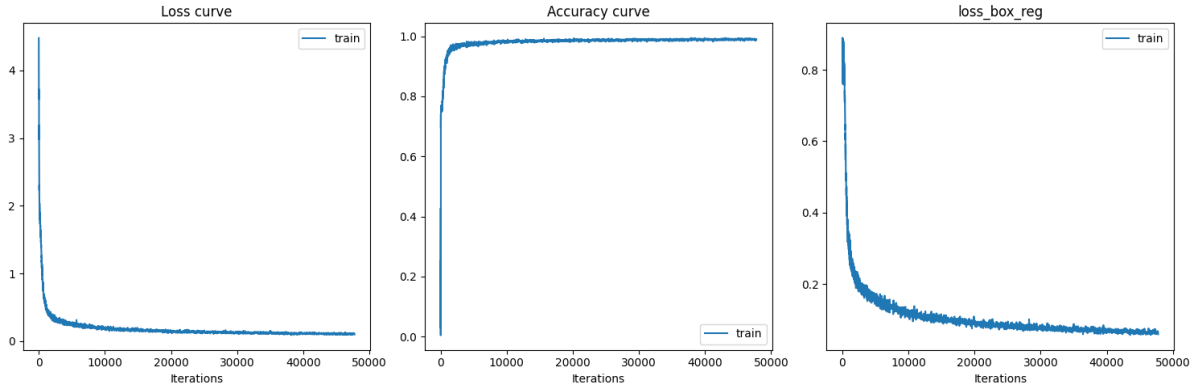


FIGURE 2.4
Evolution of Total Loss, Accuracy and Bounding Box Loss through iterations

The training process was interrupted after 47,759 iterations as the loss and accuracy metrics reached a stable state, as evident from the plotted **Figures 2.4**:

- **Total Loss:** 0.097361
- **Accuracy:** 0.990234
- **Bounding Box Loss:** 0.059139

Continuing the training beyond this point would have provided minimal improvements to the model's performance. The observed stability in the loss curve and accuracy curve suggests that the model had already converged and further iterations would not yield significant enhancements.

The goal of the training phase was to optimize the layout recognition model to accurately identify and extract layout components from the scanned book pages. The trained model would serve as a crucial component for subsequent steps in the transformation process, enabling the creation of a comprehensive digital catalogue.

2.1.3 OCR

After the layout recognition process, the next step was to initiate the content extraction phase. For this project, text extraction was the primary goal, as it provided the foundation for creating a comprehensive digital catalogue of the book's architectural content. The extracted text would be further processed, analyzed, and organized to enable efficient search, retrieval, and exploration of the architectural information. However, during the extraction process, it was also possible to identify and extract images present in the scanned book pages. While these images contained valuable visual representations of the architectural elements, layout recognition did not perform as well as for textual content (see Layout Evaluation section). They would have thus required additional post-training processing. Therefore, we decided to not use the extraction of images for the specific purpose of creating the digital catalogue.

Instead, the emphasis was placed on extracting and processing the textual information using Optical Character Recognition (OCR) techniques. For this purpose, Tesseract, an open-source OCR engine, available within the Layout Parser library, was utilized.

Tesseract is a powerful OCR engine known for its accuracy and wide range of language support. It employs deep learning algorithms to recognize and extract text from images. The previously trained and fine-tuned layout recognition model, which accurately identified and localized text regions, played a crucial role in guiding the OCR process.

The OCR workflow involved the following steps:

- **Preprocessing:** Before passing the extracted text regions to Tesseract for OCR, preprocessing techniques were applied to enhance the OCR accuracy. This mainly involved indexing text boxes in a properly sorted way so that they can be retrieved later in the correct order.
- **Text Region Extraction:** The layout recognition model, trained on the annotated data, provided precise bounding boxes around the text regions on the book pages. These bounding boxes were used to extract the text regions, ensuring that only the relevant text areas were processed.
- **OCR with Tesseract:** Text regions were then passed through Tesseract for OCR. Tesseract used deep learning models to analyze the images and convert them into machine-readable text. The recognized text was extracted along with its spatial coordinates to maintain its association with the layout structure.

The OCR process, combined with the layout recognition step, enabled the extraction of the textual content from the scanned book pages. The integration of Tesseract within the Layout Parser library facilitated a seamless and efficient workflow, allowing for accurate and reliable text extraction.

The extracted text data served as a crucial component for subsequent analysis, NLP (Natural Language Processing), and database management steps in the transformation process, enabling the creation of a comprehensive digital catalogue of the book's architectural content.

2.2 DATABASE MANAGEMENT

The database management phase involved designing the database schema, cleaning the extracted text data, and inserting it into the database.

2.2.1 ER MODEL

The first step in designing the database was to create an Entity-Relationship (ER) model. The ER model served as a visual representation of the entities, attributes, and relationships within the data. It provided a high-level overview of the database structure, helping to identify the key entities and their associations.

During the ER modeling process (see **Figure 2.5**), the entities relevant to the architectural content and relationships between these were identified capturing the associations and dependencies among them.

We have chosen to structure articles into 6 entities:

- **Building:** this is the main entity that has relations with every other one in the model. It represents each of the 64 Mamluk buildings and has an id as a primary key, a foreign key with every other entity (other than Inscription), a name, modern_name and type attributes.
- **Location:** keeps track of the building's coordinates (longitude and latitude), the CRS (Coordinate Reference System), and id as a primary key and a description attribute that gives further textual details about the building's location.
- **Date:** that an id as a primary key, both hijri and gregorian dates as attributes, an explanation for the dates meaning (built, endowed,...) and 'others' attribute for further details about the date (taken from the history section).
- **History:** has id as primary key, identification, founder, endowments, and subsequent_history information as attributes, a list of all subtitles of this section and others which loads the entire text of the History section in the article.

- **Architecture:** because of the complexity of this part's structure, we chose to simply store the entire text as well as the list of subtitles of the section in 'others' and 'subtitles' attributes respectively, and of course id as a primary key.
- **Inscription:** finally identifies all the inscriptions cited in the building's article. It has id as a primary key, the building reference stored in the foreign key 'building_id', the inscription 'text', and a 'reference' for the section containing the inscription as attributes.

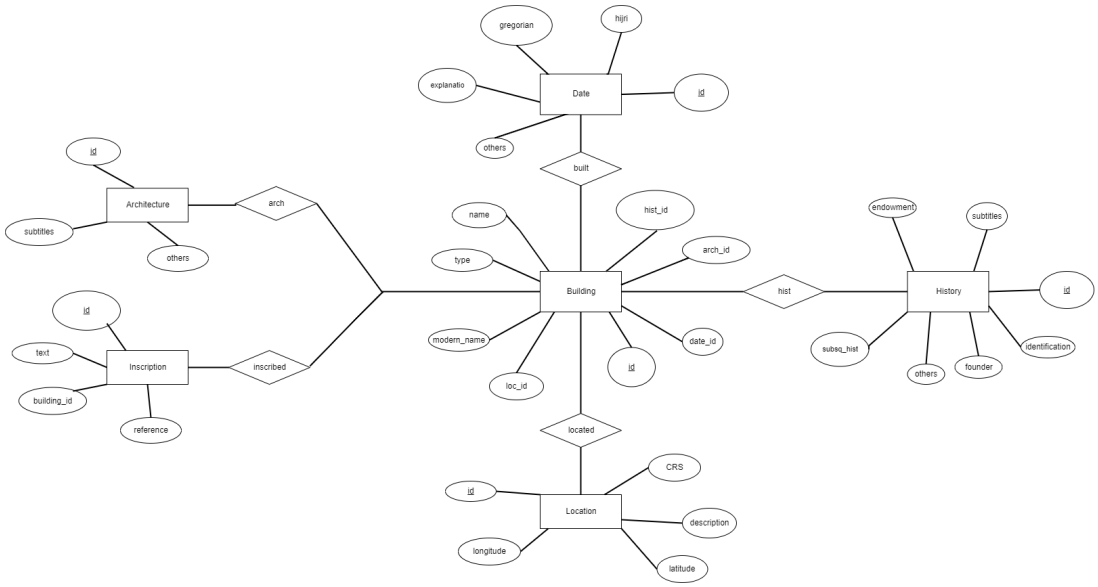


FIGURE 2.5
ER Model of Mamluk buildings in Jerusalem

The ER model acted as a blueprint for the subsequent creation of the relational model, which provided a more detailed representation of the database structure.

2.2.2 RELATIONAL MODEL

Based on the ER model, the next step was to convert it into a Relational Model. The Relational Model represented the database schema in the form of tables, with each table representing an entity and its corresponding attributes.

Using the identified entities and relationships from the ER model, tables were created in accordance with the principles of database normalization. The tables were designed to eliminate redundancy, ensure data integrity, and facilitate efficient data retrieval.

The attributes extracted from the OCR text were assigned to their respective tables and columns in the relational model we can observe in **Figure 2.6**. The appropriate data types and constraints were defined to ensure the accuracy and consistency of the stored data.

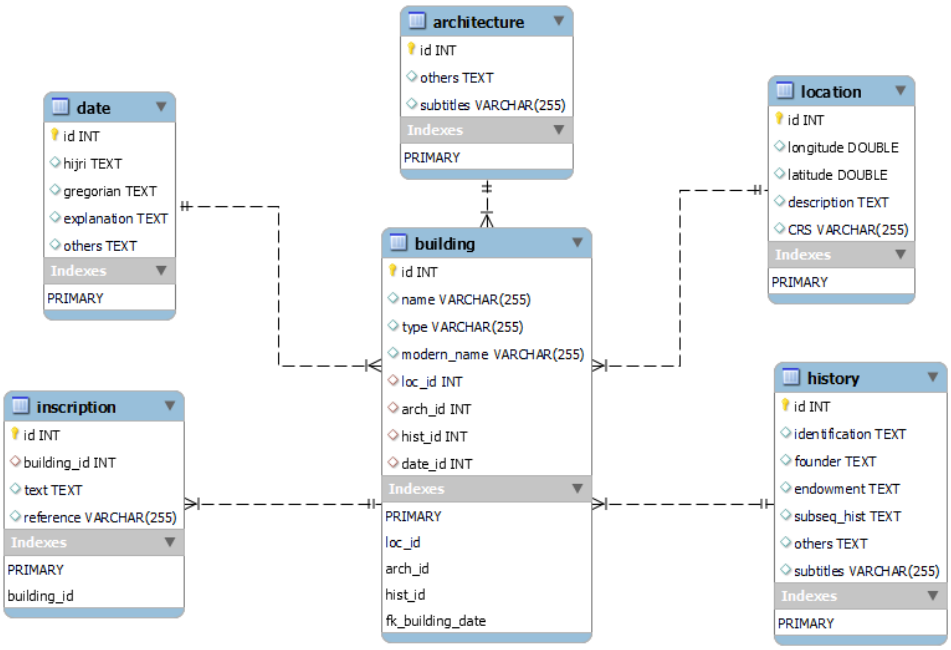


FIGURE 2.6
Relational Model

1:n : representation of a 1 to n relation

The Relational Model provided a structured and organized representation of the extracted text data, laying the foundation for the subsequent database insertion process.

2.2.3 DATABASE INSERTION

DATA CLEANING

Before inserting the extracted text data into the database, a data cleaning process was performed. This process aimed to preprocess and refine the raw text data to improve its quality and consistency.

Pandas, a powerful data manipulation library in Python, were used for data-cleaning tasks. The extracted text data was loaded into data frames, enabling efficient data transformation and manipulation.

NLP techniques from the NLTK (Natural Language Toolkit) library, were applied to perform tasks such as removing stopwords, punctuation, and special characters. Processes like stemming, using tools like the NLTK PortStemmer, were employed in some cases to reduce words to their root form, thereby reducing redundancy and improving data consistency.

The data cleaning process ensured that the text data was in a clean and standardized format, ready for insertion into the database.

MYSQL

After data cleaning, the cleaned text data was inserted into a MySQL database. MySQL is an open-source relational database management system (RDBMS) widely used for storing, managing, and retrieving structured data. It has been chosen because it provides a reliable and efficient platform for data storage and offers robust features for data manipulation, querying, and indexing.

The insertion process involved converting the data frames to CSV files, which served as an intermediate format for transferring the data to the database.

Using the csv python library, the data frames were converted to CSV files, ensuring that the data was structured in a tabular format suitable for insertion into the relational database. The CSV files were then used to generate SQL queries for database insertion.

The SQL queries were designed to create the necessary tables within the database schema and populate them with the cleaned text data. The INSERT statements were executed to insert the data into the appropriate tables and columns.

By leveraging the capabilities of Pandas, Python, and MySQL, the text data was effectively transformed and loaded into the database. This allowed for efficient storage, management, and retrieval of the architectural content within the digital catalogue.

The successful completion of the database insertion process established a solid foundation for further data management and facilitated the seamless integration of the extracted text data into the digital catalogue system.

2.3 DATA VISUALIZATION

The data visualization phase aimed to present the extracted architectural data in an interactive and visually appealing manner. The chosen approach involved creating an interactive leaflet map that displayed Mamluk buildings and provided various ways to interact with each building. The section below provides further details on the data visualization process.

The initial approach to implementing the interactive map was to utilize the Folium Python library (python-visualization 2020). Folium offers map visualization capabilities, allowing for the integration of geographical data with interactive elements. However, during the implementation phase, limitations in interactivity were encountered, prompting a switch to the ipyleaflet library.

The ipyleaflet library (QuantStack 2021), a Jupyter-friendly Python library based on the Leaflet JavaScript library, was chosen as an alternative due to its enhanced interactivity features and compatibility with Jupyter notebooks. ipyleaflet provided a powerful framework for creating interactive maps with customizable markers and interactive controls.

By leveraging the ipyleaflet library, the focus was on developing an interactive leaflet map that showcased the Mamluk buildings. Each building would be represented as a marker on the map, visually indicating its location. Additionally, various interactive elements were incorporated to enhance the user experience and provide additional information about each building.

The map's interactivity allowed users to interact with the markers and access detailed information about specific buildings. For example, clicking on a marker could trigger a pop-up window displaying information such as the building's name, historical significance, architectural features, and related inscriptions. Zooming and panning functionality enabled users to explore the map at different scales and navigate through different regions of interest.

The results section will provide a comprehensive demonstration and analysis of the interactive features and functionalities of the leaflet map. It will showcase how map visualization and interactivity contribute to a better understanding and exploration of the Mamluk buildings in Jerusalem, enriching the overall digital catalogue experience.

CHAPTER 3

RESULTS

3.1 LAYOUT EVALUATION

The layout evaluation process aimed to assess the performance of the layout recognition system in accurately identifying and extracting different elements within the document layout.

3.1.1 IOU SCORE

The evaluation was conducted with all labels included, followed by a refinement process where caption and figure labels were filtered out. The rationale behind this refinement was to focus specifically on the recognition of text elements within the layout.

Upon evaluation, the system achieved an average IOU score of 0.66 when considering all labels. This score indicates the degree of overlap between the predicted bounding boxes and the ground truth bounding boxes for all elements in the layout. However, after filtering out caption and figure labels, the average IOU score increased to 0.81 This improvement demonstrates the effectiveness of the filtering process in refining the recognition accuracy for text components.

3.1.2 F1-SCORE

To further analyze the performance, F1 scores were calculated both with and without figure labels. The F1 score considers both precision and recall and provides an overall assessment of the system's performance. By comparing the F1 scores, we were able to observe the impact of including or excluding figure labels on the system's recognition performance. The results were plotted to visualize the difference in F1 scores between the two scenarios, shedding light on the effectiveness of focusing on text-only recognition.

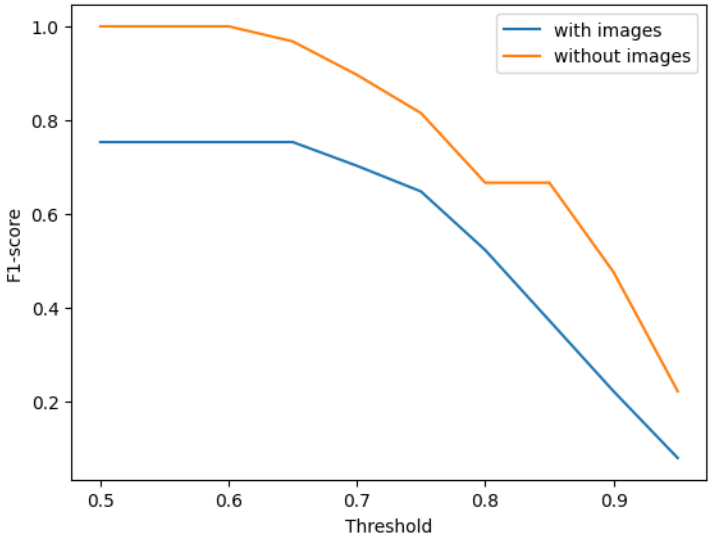


FIGURE 3.1
F1 scores across different thresholds

The **Figure 3.1** of the F1 score plot across different thresholds provides a clear visualization of the performance comparison between the two scenarios: with and without images.

As the threshold increases, the recognition system becomes more selective in identifying layout elements. When focusing solely on text recognition by excluding images, the F1 score consistently outperforms the scenario with images across all threshold values. This observation highlights the benefits of filtering out images and solely concentrating on text components in terms of achieving higher overall accuracy and performance.

Additionally, a precision-recall curve was plotted to provide a comprehensive understanding of the system's performance at various thresholds.

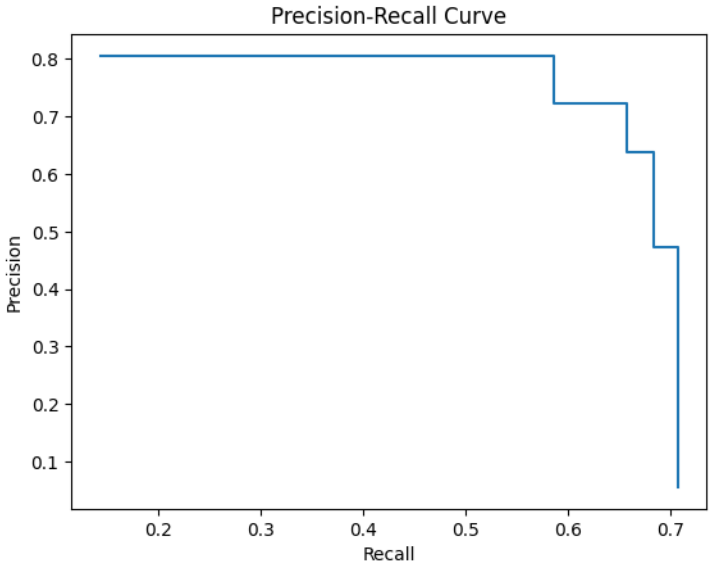


FIGURE 3.2
Precision-Recall curve across multiple thresholds

The curve in **Figure 3.2** visually represents the trade-off between precision and recall, allowing for the identification of an optimal threshold based on the desired precision-recall trade-off. A higher precision indicates a lower number of false positives (layouts wrongly identified), while a higher recall indicates a lower number of false negatives (missed boxes).

At lower thresholds, where the system is more permissive in accepting predictions, both precision and recall tend to be higher. This indicates that more elements are correctly identified (higher recall), but there is also a higher chance of including false positives (lower precision).

Conversely, at higher thresholds, the system becomes more selective, resulting in lower recall as some correct elements may be missed. However, the precision increases since the accepted predictions are more likely to be correct.

3.2 OCR RESULTS

In the OCR results section, the focus was on evaluating the quality of the Optical Character Recognition (OCR) process applied to the scanned pages of the book.

To ensure optimal OCR results, a high DPI (dots per inch) value of 300 was specified when loading the scanned pages using the lp.io.load_pdf() method. This higher DPI value significantly improved the quality of the loaded pages, resulting in better OCR performance. It is worth noting that when the DPI value is lower, the scanned pages' loading quality tends to degrade, impacting the accuracy of the OCR process.

While various metrics, such as word-level or character-level accuracy, could have been employed to quantitatively evaluate the OCR results, it was determined that conducting such evaluations would be time-consuming and unnecessary. This decision was based on the understanding that the combination of good layout parsing and the high scanning quality of the book inherently implies good OCR accuracy. As a result, visual inspection was deemed sufficient to gain confidence in the overall quality of the OCR output.

Visual inspection involved manually reviewing the OCR results alongside the original scanned pages to ensure accurate and faithful transcription of the text. This qualitative evaluation method enabled the identification of any discrepancies or errors, providing a comprehensive understanding of the OCR performance.

The evaluation of OCR results through visual inspection confirmed the high quality of the OCR output. The accurate layout parsing and the high scanning quality of the book contributed to the overall fidelity of the transcribed text.

3.3 DATABASE

3.3.1 INSERTION PRECISION

The evaluation of the database insertion process involved assessing the precision of the insertion for the main columns of the database. The precision was determined by examining the percentage of non-NaN (not inserted correctly) values in each column.

The evaluation results are as follows:

Column	Precision
Name	1.0
Date_id	0.84375
Type	0.859375
Modern_name	0.859375
Hist_id	0.90625
Arch_id	0.828125

TABLE 3.1
Database Insertion Evaluation

It is important to consider that the presence of missing values in these columns is not solely attributable to the database insertion process. The missing values may also stem from earlier stages of the workflow, such as layout recognition and OCR. Therefore, it is crucial to acknowledge that the precision results are influenced by the quality and accuracy of the data obtained from these preceding steps.

The database insertion evaluation provides insights into the precision of inserting data into the main columns of the database. While some columns exhibit higher precision, others have a relatively lower precision due to missing values.

3.3.2 EXECUTION TIME

Moreover, an evaluation was conducted to assess the execution time of several sample queries. These queries were designed to retrieve specific information from the database and provide insights into its performance. The purpose of this evaluation was to analyze the efficiency and responsiveness of the database system when executing different types of queries.

The sample queries included:

- Query 1: *"SELECT * FROM building WHERE name = ''".format(name)*
  Retrieving all columns from the "building" table based on a randomly selected building name.
- Query 2: *"SELECT * FROM building"*
  Retrieving all columns from the "building" table.
- Query 3: *"SELECT name, modern_name FROM building"*
  Retrieving only the "name" and "modern_name" columns from the "building" table.
- Query 4: *"SELECT COUNT(*) FROM date"*
  Counting the total number of records in the "date" table.
- Query 5: *"SELECT COUNT(DISTINCT subtitles) FROM architecture"*
  Counting the number of distinct values in the "subtitles" column of the "architecture" table.
- Query 6: *"SELECT * FROM building JOIN date ON building.date_id = date.id"*
  Performing a join operation between the "building" and "date" tables, retrieving all columns.

The execution times of these queries were measured and the results were plotted in a histogram, as shown below:

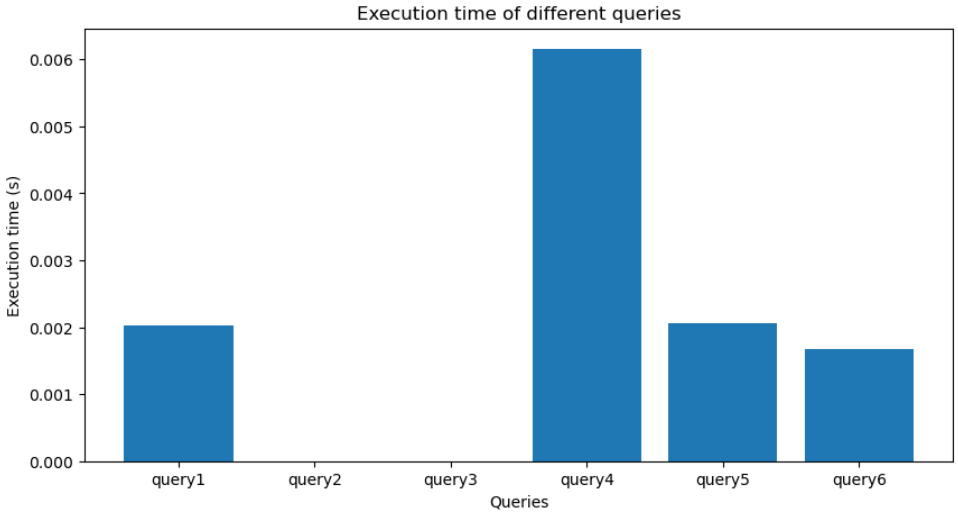


FIGURE 3.3
Execution Time of Sample Queries

Analyzing the histogram in **Figure 3.3**, we observe that the majority of the queries are executed within a very short time, with execution times close to zero. This indicates that the database system is highly efficient in handling these specific queries, resulting in fast response times.

3.4 LEAFLET MAP

The final result of the project is a leaflet map of Jerusalem displaying the locations of Mamluk buildings. The map incorporates various interactive features to enhance user experience and exploration of the architectural heritage.

SEARCH BAR

The leaflet map includes a search bar that enables users to search for a specific building by its name. This feature enhances the accessibility and convenience of locating a particular building within the map. Users can simply select the building's name in the search bar, and the map will automatically navigate to the corresponding location.

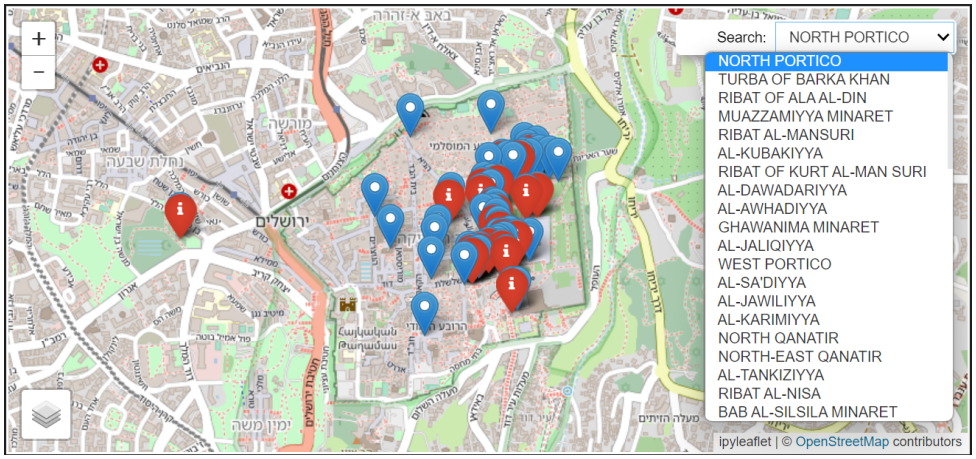


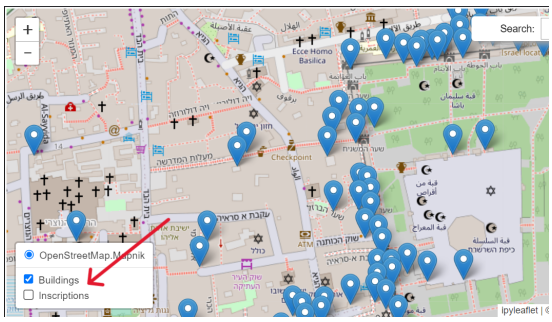
FIGURE 3.4
Search Bar

ZOOM IN/OUT

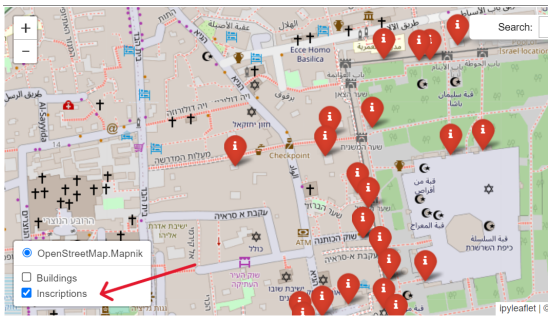
The leaflet map provides users with the ability to zoom in and out, allowing for a closer look at specific areas or a broader view of the entire map.

LAYERS

The map incorporates a layers feature that provides two distinct map layers for better organization and visualization. The first layer (Figure 3.5a) displays information about the Mamluk buildings, while the second layer (Figure 3.5b) focuses on inscriptions found within these buildings.



(A) Buildings Layer

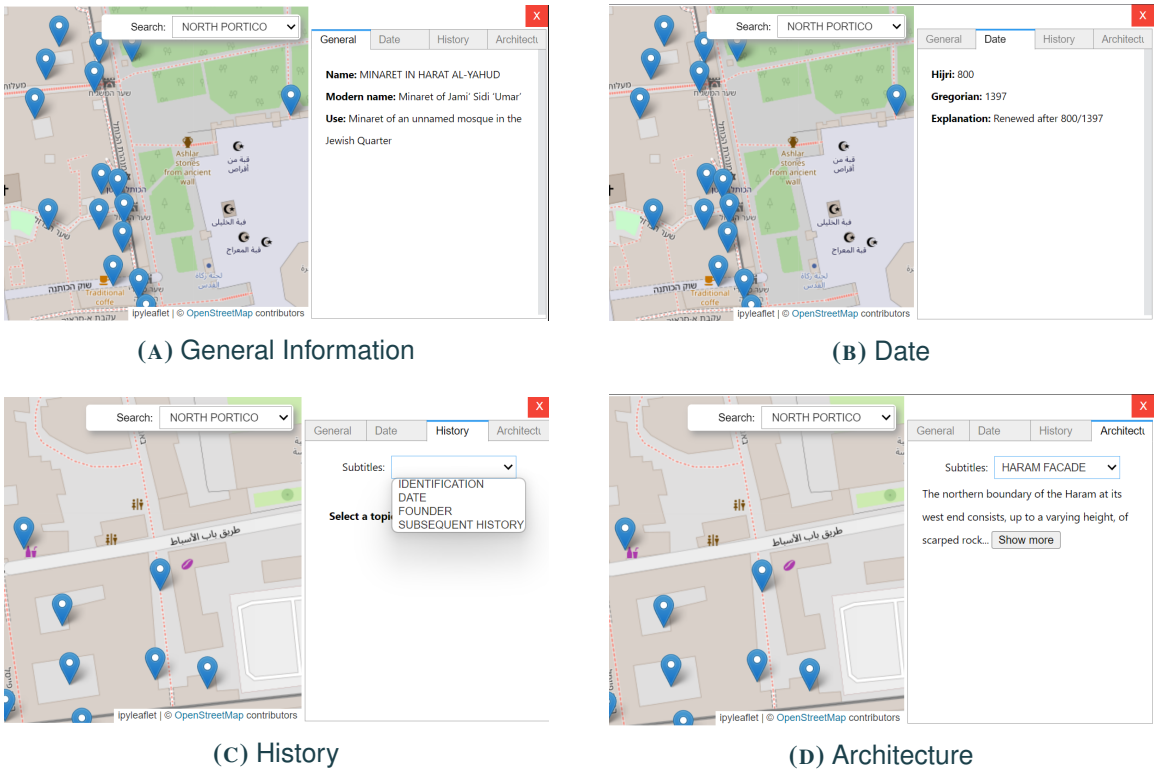


(B) Inscriptions Layer

FIGURE 3.5
Layers Feature

BUILDING MARKER POPUPS

Each building marker on the map includes a popup or side pane that provides detailed information about the selected building. The popup is divided into four sections: General, Date, History, and Architecture.



(A) General Information

(B) Date

(C) History

(D) Architecture

FIGURE 3.6
Buildings Popup

General: This section (Figure 3.6a) presents key information about the building, including its name, modern name (if applicable), and use or purpose.

Date: The Date section (Figure 3.6b) includes the building's historical dates in both the Hijri (Islamic calendar) and Gregorian calendars. Additionally, an explanation of the date is provided to offer further context.

History: In the History section (Figure 3.6c), users can access a dropdown menu of subtitles associated with the building. When a subtitle is selected from the dropdown, the corresponding content from the book article is displayed, providing insights into the building's historical significance.

Architecture: Similar to the History section (Figure 3.6d), the Architecture section allows users to explore specific architectural aspects of the building by selecting different subtitles from the dropdown menu.

These interactive building marker popups offer a comprehensive understanding of each Mamluk building's details, history, and architectural features.

INSCRIPTION POPUP

In addition to the building marker popups, the leaflet map includes a special feature for inscriptions. When a building marker on the map is clicked within the inscription layer, a side pane or popup displays all the inscriptions cited in the book specifically related to that building.

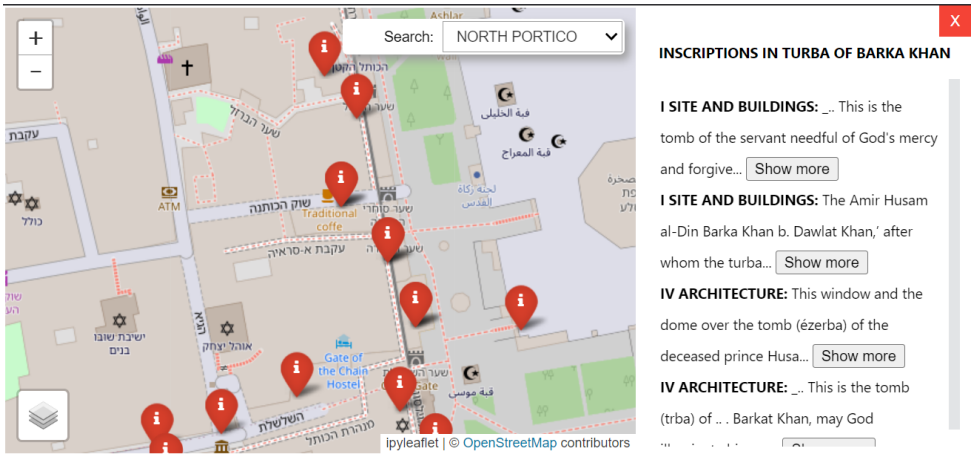


FIGURE 3.7
Inscription Popup

The Inscription Popup in **Figure 3.7** provides users with a comprehensive view of the inscriptions found within the selected Mamluk building. Users can explore the textual content, calligraphy, and other details of these inscriptions, contributing to a deeper understanding of the historical and cultural significance associated with each building.

The Leaflet map, along with its interactive features and informative marker popups, serves as an engaging platform for users to explore and learn about the Mamluk architectural and cultural heritage in Jerusalem.

CHAPTER 4

DISCUSSION

4.1 DISCUSSION

The results of the project, including layout recognition, OCR, database management, and leaflet map visualization, provide valuable insights into the digital cataloging of Jerusalem's Ayyubid and Mamluk architectures.

4.1.1 RESULTS ANALYSIS

LAYOUT RECOGNITION

The layout recognition process using the Layout Parser library proved to be effective in identifying and extracting various components of the book's layout. By training an existing model on annotated images from the book, we achieved accurate identification of the articles' layouts. However, it should be noted that certain limitations were encountered, particularly in cases where the layout structure was complex or varied. This suggests that further improvements and refinements could be made to enhance the accuracy and robustness of the layout recognition process.

OCR RESULTS

The OCR results demonstrated satisfactory quality, supported by visual inspection and the contextual understanding of the book's layout and scanning quality. Although word-level or character-level accuracy metrics were not evaluated, the combination of good layout parsing and high-quality scanning contributed to reliable OCR results. This highlights the importance of considering the overall context and characteristics of the document when assessing OCR performance.

DATABASE MANAGEMENT

The database management phase involved the design and implementation of the database schema, data cleaning, and insertion into a MySQL database. The evaluation of database insertion revealed high precision in the main columns, indicating successful data transfer from the extracted sources. However, it should be noted that missing values in certain columns could be attributed to earlier stages such as layout recognition and OCR. This highlights the impact of preceding processes on subsequent stages of the project.

LEAFLET MAP

The leaflet map visualization provided an interactive and informative platform for exploring Jerusalem's Mamluk buildings. The inclusion of interactive features enhanced the usability and user experience. The building marker popups offered comprehensive information about each building, including general details, historical dates, and architectural and historical aspects. The inscription popup further enriched the user experience by providing access to inscriptions specific to each building, contributing to a deeper understanding of their cultural significance.

4.1.2 LIMITATIONS AND FUTURE DIRECTIONS

Despite the overall success of the project, certain limitations should be acknowledged. The layout recognition process could benefit from further refinement to improve accuracy, particularly in complex layout structures, especially for images. Additionally, while the OCR results were satisfactory, future evaluation using word-level or character-level accuracy metrics could provide a more comprehensive assessment.

Furthermore, future work can focus on enhancing the processing and inclusion of images within the digital catalog. This involves developing techniques for image analysis, integrating image data into the database schema, and incorporating image visualization within the leaflet map. This expansion will provide a more comprehensive and immersive experience for exploring Jerusalem's Mamluk architectures.

The developed process for creating a digital catalog of Jerusalem's Ayyubid and Mamluk architectures is highly scalable and can be easily applied to other books with similar structures. By modifying the input PDF file, the layout recognition, OCR, and database management steps can be adapted to capture the architectural information from a different book, such as the Ayubid architecture book. The flexibility of the process allows for the expansion of the digital catalog to include additional architectural works

Lastly, while the leaflet map provided an interactive and informative visualization, future enhancements could include additional features such as advanced filtering options, user contributions, and integration of buildings' images and schemas to enrich the exploration and understanding of Jerusalem's architectural heritage.

CHAPTER 5

CONCLUSION

In conclusion, this project successfully implemented a digital catalog of Jerusalem's Ayyubid and Mamluk architectures.

The combination of computer vision, machine learning, database management, and data visualization created a comprehensive and interactive platform for exploring and understanding the architectural heritage of Jerusalem.

The findings and insights gained from this project provide a solid foundation for further research and preservation efforts in the field of digital cataloging and architectural studies.

BIBLIOGRAPHY

Burgoyne, M.H., D.S. Richards and British School of Archaeology in Jerusalem (1987). *Mamluk Jerusalem: An Architectural Study*. British School of Archaeology in Jerusalem by the World of Islam Festival Trust. ISBN: 9780905035338. URL: https://books.google.ch/books?id=qR%5C_qAAAAMAAJ.

Public Models in Transkribus (2021). en-GB. URL: https://readcoop.eu/transkribus/public-models/ (visited on 26th May 2021).

Shen, Zejiang et al. (2021). 'LayoutParser: A Unified Toolkit for Deep Learning Based Document Image Analysis'. In: *arXiv preprint arXiv:2103.15348*.

Tkachenko, Maxim et al. (2020-2022). *Label Studio: Data labeling software*. Open source software available from https://github.com/heartexlabs/label-studio. URL: https://github.com/heartexlabs/label-studio.

Lin, Tsung-Yi et al. (2014). 'Microsoft COCO: Common Objects in Context'. In: *CoRR* abs/1405.0312. arXiv: 1405.0312. URL: http://arxiv.org/abs/1405.0312.

python-visualization (28th Dec. 2020). *Folium*. Version 0.11.0. URL: https://python-visualization.github.io/folium/.

QuantStack (2021). *ipyleaflet*. URL: https://github.com/voila-dashboards/ipyleaflet.