الجمهورية الجزائرية الديمقراطية الشعبية

The People's Democratic Republic of Algeria

وزارة التعليم العالي والبحث العلمي

Ministry of Higher Education and Scientific Research

جامعة محمد بوضياف بالمسيلة

University Mohamed Boudiaf of M'sila

كلية الرياضيات والإعلام الآلي

Faculty of Mathematics and Informatics

قسم الإعلام الآلي

Department of Computer Science

Domain: Mathematics and Computer Science

Thesis Presented to Fulfill the Partial Requirement

for an **Academic Master's Degree** in Computer Science

Speciality: Information Systems and Software Engineering

Prepared By: Rahmani Ouissal and Ramli Safia

Supervised By:

Dr. Mahmoud BRAHIMI

ENTITLED

An Intelligent System for Automatic Classification of Academic Titles Using AI and NLP

Jury Members

Lamri SAYAD	President
Mahmoud BRAHIMI	Supervisor
Amel MELIOUH	Examiner

Academic Year 2024/2025

Dedication

To the one who raised me and supported me with prayers and supplications, the one with the great heart who illuminated my path with love — to the most precious woman in this world… my beloved mother, may Allah protect and preserve her.

To the one who Allah has graced with dignity and reverence, the one who taught me to give without expecting in return, the one whose name I carry with pride, and who gave everything he had for my sake… to the dearest of all, my beloved father.

To the one I wished could share my joy today, my precious sister Rima — may Allah have mercy on her soul and grant her the highest place in Paradise — and to her two dear daughters, Mayar and Layan, Rima's blossoms and the extension of her soul, who keep her memory alive in our hearts every single day.

To the two upon whom I rely and find strength, who have touched my soul and captured my heart, who taught me the lessons of life — to the candles that lit my path… my dearest sisters Chaima and Asma.

And to my beloved brothers, Mohamed and Amine — the heartbeat and cornerstone of our family — who have never hesitated to support and stand by me with their ever-present love and constant encouragement. I also dedicate this work to their gracious wives and their dear children, who have filled my heart and home with light through their presence.

And to my dear friends, who were a source of support and companionship throughout my journey, filling my days with joy — especially my loyal friend Radja, who has always been the closest to my heart, the most sincere in her prayers, and the most faithful in her presence.

Ouissal

Dedication

With profound pride and heartfelt gratitude, and a soul filled with praise to the One True God, I dedicate this graduation—not merely a certificate, but a journey of perseverance, patience, and faith— to those whose presence was a blessing, and whose absence is an unhealable ache…

To the pure soul of my late father, who may have vanished from sight yet remains ever-present in my heart. You instilled in me the values and wisdom that continue to guide my mind and path. May God grant you boundless mercy and eternal paradise. This achievement is a solemn vow I made to myself—to become all that you dreamed I would be and more. May you boast of me in the heavens, just as you once proudly did on earth, and may you tell the angels that your daughter kept her promise...

And made it. To my mother, who endured, persevered, kept vigil, and nurtured. You have been my unwavering support, my haven, and the prayer whispered behind every success. Every accomplishment is a reflection of your sacrifices, and every triumph is yours to claim.

To my siblings and family, you were the warm embrace in trying times and the light that illuminated my darkest roads. You stood as my armour and unshakeable support. Thank you from the deepest depths of my heart.

To my husband, my companion in this journey and partner through life's trials, who stood steadfast by my side with wisdom, patience, and quiet strength. Your presence was a solid foundation and an irreplaceable pillar in helping us reach this milestone.

And finally, to myself— to the girl who never surrendered, despite storms and betrayals, who carried her dreams in her heart and stayed awake through countless nights, who believed—first in God, then in herself—and who truly earned this moment. This graduation is not an end but a new beginning—a greater path awaits, one paved with resolve and blessed by God.

Safia

Acknowledgement

In the name of Allah, the Most Gracious, the Most Merciful, All praise is due to Allah, by whose grace good deeds are completed and through whose favour this work has reached its fulfilment and purpose.

We want to express our sincere gratitude to everyone who contributed to the completion of this thesis, as well as to all who supported our academic and research journey from its inception to its conclusion.

We are especially grateful to our esteemed supervisor, **Dr. Brahimi Mahmoud**, for his continuous support, guidance, and generosity with his time and knowledge. He has truly been a mentor and a guiding light throughout this endeavour.

We also extend our heartfelt appreciation to the respected members of the examination committee for their valuable time and insightful remarks, which significantly enriched this work.

Our deepest gratitude goes to our dear families for their moral and financial support, their constant encouragement, and their enduring patience, which have formed the foundation of our progress.

Finally, we offer our gratitude to everyone who contributed, directly or indirectly, to the completion of this thesis.

May Allah reward you all on our behalf and bless your knowledge and efforts.

Ouissal & Safia

ملخص :

يهدف هذا المشروع إلى اقتراح نظام لتصنيف عناوين الكتب والرسائل الجامعية باستخدام تقنيات الذكاء الاصطناعي (AI) ومعالجة اللغة الطبيعية (NLP) .يعتمد النظام على نموذج DeBERTaV3 المدرب على مجموعة بيانات أكاديمية متعددة الفئات، ويدعم الإدخال متعدد اللغات من خلال تقنيات التعرف الضوئي على الحروف(OCR) ، واكتشاف اللغة(langdetect) ، والترجمة التلقائية (googletrans) وقد حقق النظام دقة عالية في تصنيف العناوين ضمن مجالات مثل العلوم الاقتصادية، القانون، والعلوم التقنية. تم تطوير النظام باستخدام لغة البرمجةPython ، ودمج مكتبات مثلTransformers ، Pandas، PyTorch، وPySide6،

مع تخزين النتائج في قاعدة بيانات Excel.

الكلمات المفتاحية: الذكاء الاصطناعي، تصنيف العناوين، المعالجة متعددة اللغات، DeBERTaV3

Abstract:

This project aims to propose a system for classifying book and thesis titles using Artificial Intelligence (AI) and Natural Language Processing (NLP). It employs the DeBERTaV3 model, trained on a multi-category academic dataset, and supports multilingual input through OCR, language detection (langdetect), and automatic translation (googletrans). The system achieved high accuracy in classifying titles into fields such as Economic Sciences, Law, and Technical Sciences. Developed in Python, it integrates libraries such as Transformers, Pandas, PyTorch, and PySide6, with results stored in an Excel database.

Keywords: AI , Title Classification, Multilingual Processing, DeBERTaV3

Résumé: **:**

Ce projet vise à proposer un système de classification des titres de livres et de thèses en utilisant l'Intelligence Artificielle (IA) et le Traitement Automatique du Langage Naturel (TALN). Il repose sur le modèle DeBERTaV3, entraîné sur un ensemble de données académiques multicatégories, et prend en charge les entrées multilingues grâce à la reconnaissance optique de caractères (OCR), la détection de la langue (langdetect) et la traduction automatique (googletrans). Le système a obtenu une grande précision dans la classification des titres dans des domaines tels que les sciences économiques, le droit et les sciences techniques. Développé en Python, il intègre des bibliothèques telles que Transformers, Pandas, PyTorch et PySide6, avec un stockage des résultats dans une base de données Excel.

Mots-clés : IA, Classification des titres, Traitement multilingue, DeBERTaV3

List of tables

List of Figures

Table of content

CHAPTER 1: AI & Text Classification

CHAPTER 2: Design and Methodology of the Intelligent Classification System

CHAPTER 3: Development and Deployment of the Classification System

General Introduction

In today's digital age, the exponential growth of scientific and academic content has brought forth new challenges in organising, managing, and retrieving information efficiently. Libraries, as repositories of knowledge, face increasing pressure to keep up with the vast and diverse influx of books, theses, and academic articles. Traditional classification systems—though foundational—are no longer sufficient to meet the needs of modern users who demand faster and more accurate access to relevant information.

This has led to the emergence of intelligent systems powered by Artificial Intelligence (AI), particularly in the fields of Machine Learning (ML), Deep Learning (DL), and Natural Language Processing (NLP). These technologies provide innovative solutions for automating complex tasks, including text extraction, translation, and semantic classification. Among their many applications, one of the most promising is **automated text classification**. This technique enables libraries to categorise academic content based on the meaning and context of titles rather than relying solely on human input or static rules.

This dissertation examines the design and implementation of an intelligent classification system that utilises AI to categorise academic book and thesis titles into predefined academic categories. The proposed system is built around a pipeline that combines Optical Character Recognition (OCR) for title extraction, language detection, and translation, as well as a deep learning-based classification model (DeBERTaV3) trained on a rich and multilingual dataset.

This dissertation is organised around three chapters:

The first chapter provides a theoretical overview of library classification, ranging from traditional methods, such as the Dewey Decimal and Library of Congress schemes, to modern AI-based techniques, including machine learning and natural language processing (NLP). It also discusses prior studies and real-world implementations that highlight the advantages and challenges of AI in classification tasks.

The second chapter presents the core development process of the proposed system. It details the architecture, tools, and methodologies used—from data collection and model training to text processing and prediction. The integration of multiple technologies enables the system to process multilingual inputs and deliver accurate classifications, thus improving the accessibility and usability of library resources.

The third chapter dives into the technical infrastructure behind the system, including the development environment, libraries, and user interface. It illustrates how the combination of

Google Colab, PySide6, and Python libraries, such as Transformers and EasyOCR, enables the creation of a seamless and interactive application.

Ultimately, this work aims to contribute to the modernisation of library classification systems by introducing a practical and efficient AI-driven solution. Through this project, we demonstrate how advanced technologies can enhance academic resource organisation, reduce human workload, and pave the way for more innovative and more accessible digital libraries.

CHAPTER 1

AI & Text Classification

1.1 Introduction

This chapter presents the foundational concepts of library classification, beginning with traditional classification methods and their historical importance in organising academic materials, such as books, theses, and scholarly studies. It then explores the evolution of classification techniques through the integration of artificial intelligence, with a particular focus on the applications of machine learning, deep learning, and natural language processing.

These advanced technologies are utilised to automate and enhance the classification process, especially when extracting and analysing text from scanned documents. Additionally, the chapter highlights the role of text mining in uncovering hidden patterns from unstructured content, thereby improving the accuracy and efficiency of classification systems.

Ultimately, this combination of traditional and modern approaches contributes to more effective information organisation and improved accessibility to academic resources, thereby enhancing research efficiency and enabling users to quickly and accurately find the content they need.

1.2 Library Classification Definition

Library classification is the process of categorising books by subject using a specific classification scheme. Books are arranged on shelves according to this scheme to facilitate quick access to the desired books and to identify their locations easily [1].

Sayers defined library classification as the arrangement of books on shelves, or description of them, in the manner which is most useful to those who read [2].

1.3 Benefits of Library Classification

The importance of library classification is reflected in many key benefits, such as[3] :

- It helps readers access the necessary materials quickly and easily.
- It sets clear boundaries between different fields and branches of knowledge, thereby preventing the mixing or overlapping of library materials.
- It provides an ideal method for organising books, making them easy to use and return to their proper places after use.
- It facilitates the inventory process, as books dealing with the same subject are arranged.

1.4 Library Classification Techniques

1.4.1 Traditional Techniques

Classification systems have dominated traditional libraries for centuries, enabling them to arrange books and facilitate easier retrieval systematically. Classification not only aids memory but also expresses the relationships between things, reflecting the cognitive structure of subjects [2]. Several prominent classification systems have been applied in libraries around the world, most notably:

1.4.1.1 Dewey Decimal Classification (DDC)

At its simplest, the Dewey system categorises books into 10 broad subject areas, each coded by a number. Particular subject areas are given a range of code numbers recognised throughout the world [4].

1.4.1.2 Using the Junior Color Code system

In the Junior Color Code system, books are categorised into the same subject areas as the Dewey Decimal system. However, each subject area is given both a Dewey code number and a special colour [4].

1.4.1.3 Library of Congress Classification Scheme (LCC)

The whole knowledge of this classification scheme is divided into 21 primary classes. Roman capital alphabet has been used for the main topics and their first sections. There is a mixed sign system in this classification. But, for the class number, there is a limit of 2 letters and 4 Arabian figures [2].

1.4.1.4 Colon Classification System (CC)

The Colon Classification system, developed by Ranganathan, is based on analysing subjects into their fundamental components (facets) and classifying them using five basic categories known as **PMEST**: **Personality, Matter, Energy, Space, and Time**. Each subject is broken down sequentially into these facets and then reassembled to create a unique and customised classification number for each document. This enables precise classification, even of complex and compound topics [14].

To illustrate the differences and overlaps among traditional classification systems, Table 1.1 presents a set of practical examples that demonstrate how the same knowledge categories are classified within each of the following systems: the Dewey Decimal Classification (DDC), the Junior Colour Code system, and the Library of Congress Classification (LCC).

General Category	Dewey Decimal Classification (DDC)	Library of Congress Classification (LCC)	Junior Colour Code	Colon Classification (CC)
General Knowledge	0	A	Red	A
Philosophy & Psychology	100	B	Blue	Q
Religion	200	BL-BX	Green	R
Social Sciences	300	H	Yellow	T-Z
Language	400	P	Orange	O/P
Science	500	Q	Purple	B-L (includes Math to Medicine)

Table 1.1 Comparison of Library Classification Systems by General Category

1.4.2 Artificial Intelligence Techniques in Library Classification

The integration of Artificial Intelligence (AI) into cataloguing and classification systems is transforming modern libraries, revolutionising traditional practices and paving the way for more efficient and sophisticated information management. As libraries increasingly adopt AI technologies, the impact on cataloguing and classification processes is becoming a critical area of study [5].

This impact extends beyond merely accelerating tasks or reducing human effort—it also improves the quality of classification through a deeper understanding of content and the ability to intelligently and dynamically handle large volumes of textual data.

The following diagram illustrates a simplified view of the text classification process, which lies at the core of modern AI-based classification systems:

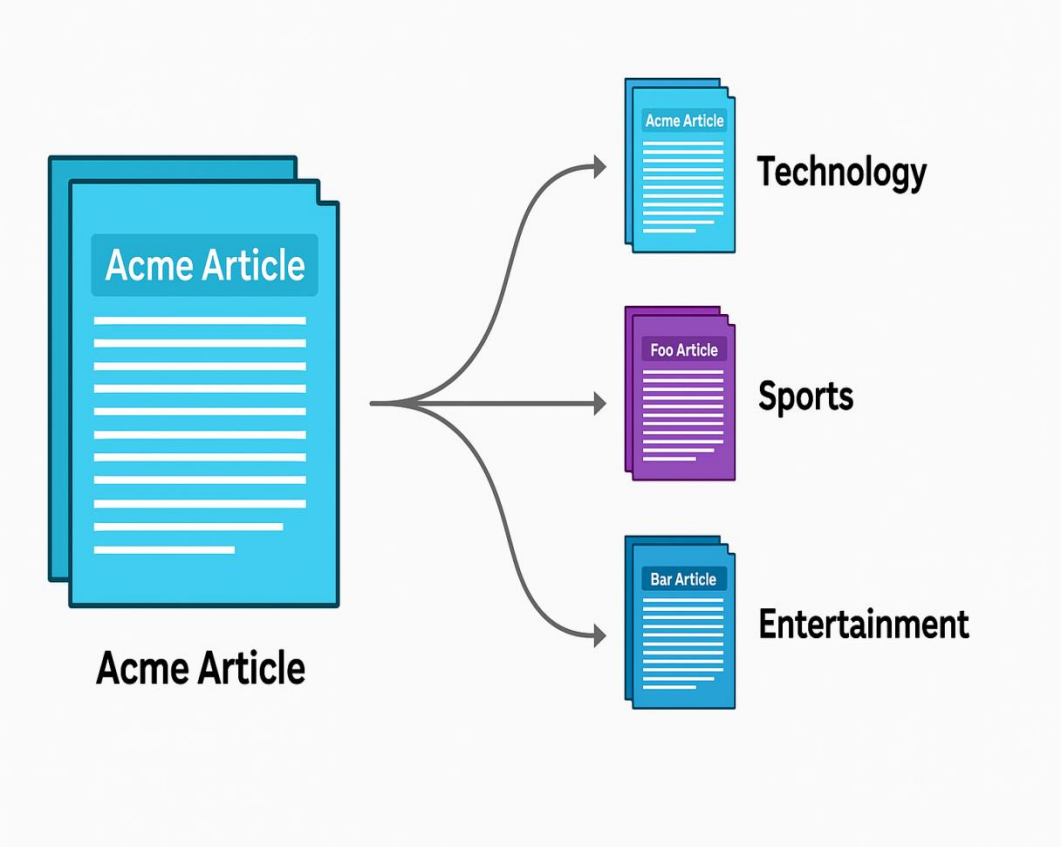

Fig 1.1 Text Classification Process in Modern AI-Powered Library Systems [30]

In this context, modern library classification relies on several key fields within artificial intelligence, most notably:

1.4.2.1 Machine Learning (ML)

Machine learning algorithms, for instance, are now employed to automate the classification of library materials. These algorithms can analyse text, images, and metadata to categorise resources accurately. The ability of machine learning models to learn from vast amounts of data enables them to continually improve classification accuracy. AI-driven classification systems have demonstrated a 20% increase in accuracy over traditional methods, reducing the time required for cataloguing by half [6].

This significant performance improvement is attributed to the use of powerful classification algorithms that form the core of machine learning systems. Among the most commonly used algorithms in text classification tasks are:

- **Linear Models**
  - Logistic Regression
  - Support Vector Machines (SVM)
  - Perceptron

    - Ridge Classifier
- **Non-linear Models**
    - K-Nearest Neighbours (KNN)
    - Kernel SVM
    - Naïve Bayes
    - Decision Tree Classification
    - Random Forest Classification

1.4.2.2 Deep learning (DL)

Deep learning technologies have also made strides in this field. For example, neural networks are now used to analyse and categorise multimedia content, such as images and videos, which were previously challenging for traditional cataloguing systems. These models have achieved a classification accuracy of up to 90% for visual content, representing a significant improvement over manual methods [6].

This advancement is attributed to the adoption of powerful models that can handle textual data in a sophisticated manner. Among the most prominent models used in text classification are:

- Recurrent Neural Networks (RNNs)
- Long Short-Term Memory units (LSTM)
- Transformer models (such as BERT, RoBERTa, DistilBERT)

1.4.2.3 Natural Language Processing (NLP)

Natural Language Processing (NLP) has further enhanced cataloguing by enabling better understanding and extraction of information from unstructured data. NLP tools can parse complex bibliographic data and convert it into structured formats suitable for library catalogues. This ability to process and interpret natural language text enables the creation of more comprehensive and precise metadata records, thereby improving resource discoverability [6].

To achieve these outcomes, NLP systems typically operate in two main phases [7]:

- Preprocessing:
    - Tokenisation
    - Stopwords Removal
    - Stemming
    - Lemmatisation
    - Text Representation Numerically (TF-IDF, Count Vectorization)

- **Analysis and Understanding:**
    - Machine Translation
    - Text Generation
    - Sentiment Analysis
    - Email Filtering
    - Text Classification

1.4.2.4 Text Classification

Text classification, also known as text tagging or text categorisation, is the process of categorising text into organised groups. By using Natural Language Processing (NLP), text classifiers can automatically analyse text and then assign a set of predefined tags or categories based on its content [8].

The following diagram illustrates the relationship between the main AI fields, Machine Learning (ML), Deep Learning (DL), and Natural Language Processing (NLP) and their roles in supporting automated text classification:

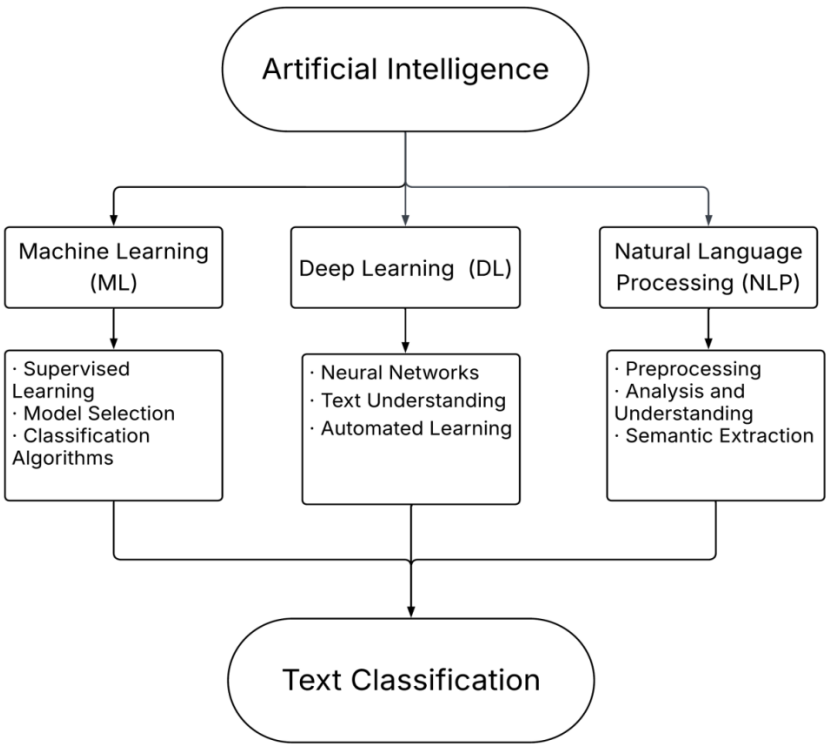

Fig 1.2 Relationship between AI technologies and Text Classification

This figure provides a visual overview of how each component contributes to different stages of processing, understanding, and classifying academic or textual content.

These classification techniques play a vital role in helping organisations extract valuable insights from vast amounts of textual data and streamline a wide range of business operations.
There are several types of classification approaches [9]:

- **Multi-Class Classification :**In this scenario, more than two classes are involved, and each document is assigned to only one class, such as categorising news articles into topics like politics, sports, or entertainment.
- **Multi-label Classification**: In this scenario, a document can belong to multiple categories simultaneously. For example, an academic paper may be categorised under multiple disciplines, such as biology and technology.
- **Hierarchical Classification**: Documents are classified in a supervised hierarchical format, meaning that broad categories are divided into more specific subcategories at multiple levels.
- **Single-Label Classification**: Often approached using binary classification methods, where documents are classified into distinct categories without overlap.
- **Binary Classification:** This involves two classes, such as academic and non-academic books, where each document belongs to one of the two categories.

1.4.2.5 Challenges and Limitations of AI in Library Classification

The integration of artificial intelligence (AI) into cataloguing and classification systems in modern libraries presents several challenges and limitations. Recent literature highlights concern about AI's effectiveness in handling complex and nuanced cataloguing tasks, such as interpreting ambiguous metadata or addressing diverse language requirements. AI systems often struggle with integrating heterogeneous data sources, leading to inconsistencies in classification and potential gaps in catalogue accuracy [6].
Additionally, there are issues related to the adaptability of AI tools to evolving library standards and practices. AI's reliance on historical data can perpetuate existing biases, impacting the fairness and inclusivity of cataloguing processes. Furthermore, the high cost and complexity of implementing advanced AI technologies can be prohibitive for many libraries. These challenges underscore the need for ongoing evaluation and refinement of AI systems to ensure they meet the dynamic needs of modern library environments [6].

1.5 Studies and Examples of Using Text Classification in Library Classification

1.5.1 Study by Momma Ali Shah et al. (2023)

In this study, authors propose a fine-tuned Bidirectional Encoder Representation from Transformers (BERT) architecture for text classification. The model has achieved state-of-the-art performance on text classification problems. The proposed framework is trained and tested using cutting-edge text datasets, such as the UCI email dataset, which comprises both spam and non-spam emails, and the BBC News dataset, which encompasses multiple categories, including tech, sports, politics, business, and entertainment. The system achieved an accuracy of 91.4% and can be utilised by various organisations to classify text-based data with high performance [10].

1.5.2 Study by Jadon & Sharma (2017)

Naïve Bayes approach is used to deal with the problem of document classification via a deceptively simplistic model. The Naïve Bayes approach is applied in a Flat (linear) and hierarchical manner to improve the efficiency of the classification model. It has been found that the Hierarchical Classification technique is more effective than Flat classification. It also performs better in the case of multi-label document classification [11].

1.5.3 Study by Fristi Riandari et al. (2022)

This study aims to develop a classification model to categorise book types using the Support Vector Machine (SVM) method. The dataset, collected from a library, undergoes several preprocessing steps, including text segmentation, case folding, tokenisation, stopword removal, and stemming. Feature extraction is then applied to convert the text into numerical vectors. After splitting the dataset into training and testing sets, classification is performed using the SVM multi-class method. The model's performance is evaluated based on the resulting accuracy, and the feasibility of implementing the classification model is analysed [12].

1.5.4 Study by Giannopoulou & Mitrou (2020)

In this study, authors aim at automatically classifying a multi-class corpus that was created from ebooks from the Springer collection by utilising an unsupervised neural network (self-organising maps, SOM) and two deep neural network (DNN) architectures, namely, a long short-term memory (LSTM) and a convolutional neural network (CNN) combined with an LSTM (CNN+LSTM) under various configuration scenarios. The vector construction leverages

information that was extracted from the table of contents (ToC) of each book using the TF-IDF weighting scheme (for the first case) and the Keras tokeniser (for the second). The subsystem that utilised the DNN (LSTM) performed the best, with F1-scores of 67% for the 26 categories and 80% for the five general categories [13].

1.6 Project Objective: Improving Library Classification Systems

The issue addressed by this project revolves around the intelligent and automated classification of academic documents in libraries, such as books and university theses. With the growing volume and diversity of scientific content, traditional classification methods have become insufficient in terms of speed and accuracy. This has led to the need for alternative technological solutions based on artificial intelligence.

In this context, the project proposes a system that relies on text classification as a primary method for organising documents within the library, using modern techniques such as Optical Character Recognition (OCR), Natural Language Processing (NLP), and the DeBERTaV3 classification model from Deep Learning technologies.

This system represents an important step toward digitising libraries and enhancing indexing and search methods, as it contributes to facilitating access to academic resources and organising them in a way that reflects the actual scientific content of each document rather than just its superficial or formal information.

Moreover, its reliance on advanced techniques makes it adaptable to different languages and scientific domains, enhancing its usefulness in various educational and academic environments.

1.7 Conclusion

In this chapter, we reviewed the foundations of library classification, covering both traditional and modern techniques. We explored how artificial intelligence, particularly machine learning, deep learning, and natural language processing, has significantly improved the automation and accuracy of classification processes. Furthermore, we examined real-world studies and applications that demonstrate the effectiveness of these technologies in organising academic materials.

Ultimately, the integration of AI in library systems provides scalable, efficient, and accurate solutions for managing scholarly content, paving the way for smarter, more accessible libraries.

CHAPTER 2

Design and Methodology of the Intelligent Classification System

2.1 Introduction

This chapter presents the development stages of the intelligent system designed for the automatic classification of books and theses. The system aims to automate the classification process by scanning the cover page, extracting the title of the work, and then determining the appropriate category to which it belongs. Additionally, the system offers an alternative option that enables users to manually enter the title in cases where scanning fails or when speed is required.

The project focuses on integrating modern technologies, including Optical Character Recognition (OCR), Natural Language Processing (NLP), and machine learning algorithms.

The importance of this system lies in its ability to facilitate the organisation of academic resources with greater accuracy and efficiency, thereby contributing to faster access to information and its classification in a systematic manner that supports research and educational environments.

2.2 Overview of the proposed solution

The proposed solution is based on an integrated sequence of steps that employ modern Natural Language Processing (NLP) techniques and deep learning to categorise book titles into their corresponding academic categories (see Figure 2.1). The system allows title input to be entered either manually by the user or extracted directly from a live camera feed using Optical Character Recognition (OCR) technology, with support for multiple languages, including Arabic, French, and English.

After extracting the text, the system automatically detects the language. If the language is not English, Google Translate is used to translate the text into English to ensure compatibility with the classification model. This step helps standardise the system's inputs and improve classification accuracy.

The translated text (or the original if already in English) is converted into a numerical representation using the tokeniser specific to the **DeBERTaV3** model. It is then passed to a pre-trained DeBERTaV3 model specialised in classifying book titles or academic thesis titles. This model is trained on a dataset of pre-labelled titles categorised into academic fields, and it identifies the most appropriate categories for the input title, displaying the top two predicted categories along with their associated confidence scores.

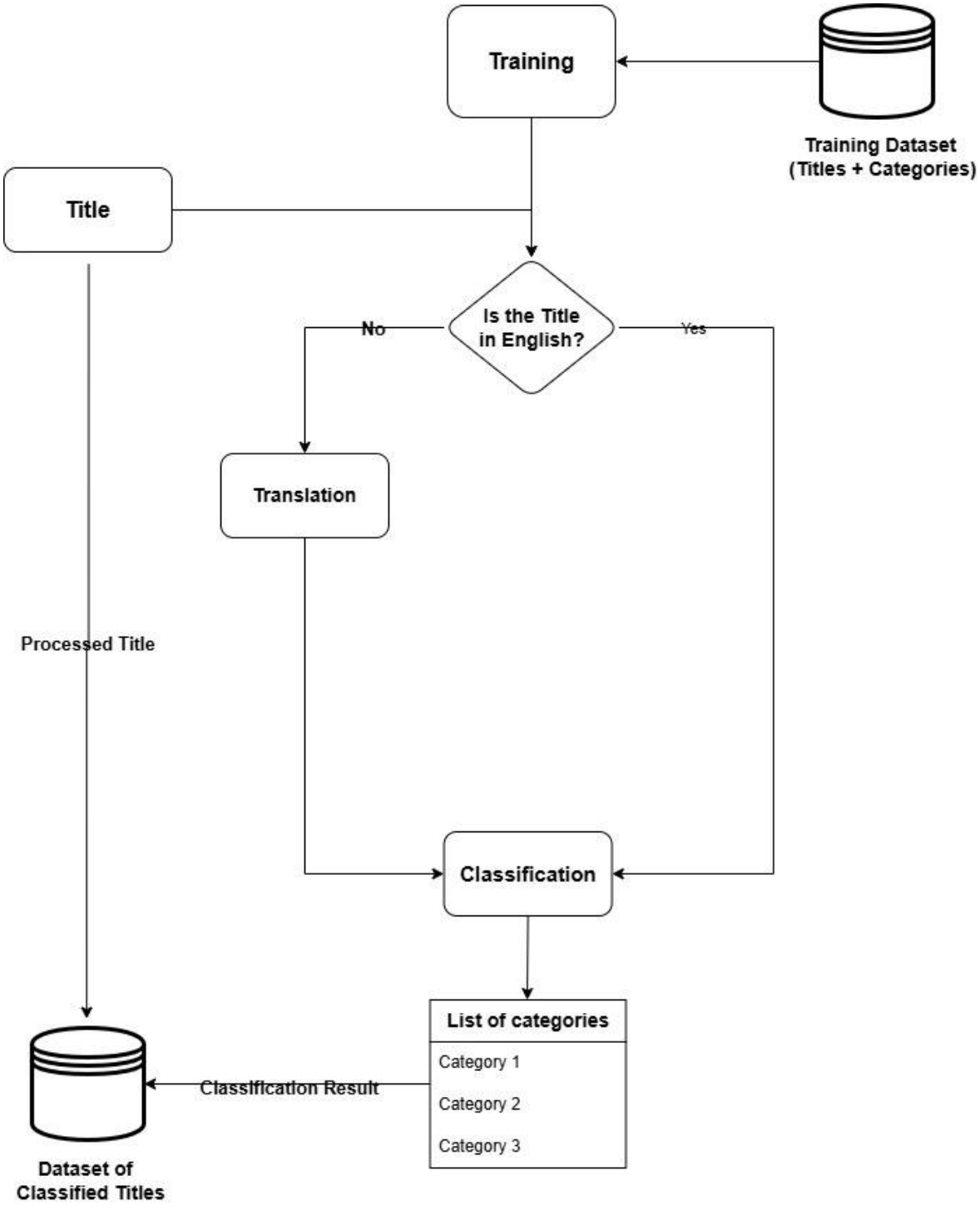

Fig 2.1 Classification Steps

The system benefits from the deep contextual understanding provided by the DeBERTaV3 architecture, enabling it to grasp the nuanced meanings of titles and effectively classify them into

fields such as programming, physics, Chemistry, and more. The training and validation accuracy reflect the model's efficiency and reliability in handling varied and potentially noisy inputs.

This solution is fully implemented in the **Python** programming language, leveraging powerful libraries such as **EasyOCR** for multilingual text recognition, **Google translate** for translation, **Transformers** (via Hugging Face) for deploying pre-trained deep learning models, and **Qt designer** to build an interactive graphical user interface. The result is an integrated, user-friendly, and technically robust system that enables accurate and efficient classification and organisation of book and academic thesis collection.

The training and validation accuracy reflect the model's efficiency and reliability in handling varied and potentially noisy inputs.

2.2.1 Training Dataset (Title + Categories)

The dataset used in this project is a CSV file containing the titles of books, theses, and academic articles, along with their subject classifications. This data was manually collected to create a reliable and comprehensive dataset that encompasses a broad range of academic fields. Special attention was given to ensuring accurate categorisation, particularly in broad disciplines such as Mathematics and Computer Science, which were broken down into several subfields to enhance classification accuracy and model performance. Examples include:

- **computer science:** Information Security & Cybersecurity, Networking & Communications, Algorithms & Data Structures.
- **In Mathematics:** Algebra, Mathematical Analysis, Probability and Statistics, Number Theory.

Titles were gathered from reputable sources, such as Amazon, OpenLibrary, and other academic repositories, which contributes to the credibility and quality of the dataset. The dataset consists of two columns: the first column, titled "Title," represents the title of the work (whether a book, thesis, or article) and reflects its content; the second column, titled "Category," indicates the academic field to which each title belongs, as illustrated in Figure 2.2.

	Title	Sub Genre
12293	Data Structures for GIS Applications	Geography
12168	Borderlands: Comparing Border Security in Nort...	Geography
35559	Postcolonial Ecocriticism: Nature and Environm...	Literature
487	User Acceptance Testing From Requirements to...	Programming, Software Development & Engineering
22321	Ordinal Topology and Convergence Structures	Set Theory
28571	The Law of Labour Disputes	Law
35270	The Representation of National Identity in His...	Literature
2445	Cybersecurity: A Tactical Guide for Incident R...	Information Security & Cybersecurity
33490	The Role of Locally Compact Spaces in Homotopy...	topology
20804	The Symmetries of Things	Geometry

Fig 2.2 The used dataset

This dataset is a fundamental component in developing an effective model for classifying academic book titles, as it provides the model with the opportunity to learn from a rich and diverse set of accurately labelled titles across 36 different categories. This variety in content and classifications enables the model to recognise the linguistic patterns associated.

With each academic field, thereby enhancing its accuracy in classifying future titles. The dataset includes a total of 36,000 titles, with an average of 1,000 titles per category, offering a solid foundation for training a reliable content-based text classification model.

Below is the full list of categories used in this project:

- Programming, Software Development & Engineering
- Databases & Data Management
- Geometry
- Number Theory
- Set Theory
- Financial Mathematics
- Mathematical Logic
- Engineering & Technology
- Medical & Health Sciences
- Religion & Islamic Jurisprudence
- Law
- Media & Journalism
- Tourism & Travel

- Sports
- Arts
- topology
- Arithmetic
- Probability and Statistics
- Mathematical Analysis
- algebra
- Physics
- Information Security & Cybersecurity
- Networking & Communications
- Operating Systems & Server Management
- Internet of Things (IoT) & Embedded Systems
- Algorithms & Data Structures
- Artificial Intelligence & Machine Learning
- Chemistry
- Languages & Linguistic
- Life and Earth & Planetary Sciences
- Economics & Management
- Geography
- Social Sciences
- History
- Philosophy
- Literature

2.2.2 Model Evaluation and Selection

During the early stages of system development, we explored and evaluated several classification models to determine the most suitable one for categorising academic titles. We began with traditional machine learning models, including Naïve Bayes, Logistic Regression, Support Vector Machines (SVM), and Random Forest. These models were characterised by their speed and ease of implementation, yielding acceptable initial results in specific categories. However, they struggled with representing and understanding textual context, which negatively affected their accuracy, especially when dealing with complex or semantically ambiguous titles.

To enhance performance, we developed an ensemble model using a Soft Voting Classifier that combined the outputs of Naïve Bayes, Logistic Regression, and SVM. This approach improved robustness and accuracy compared to using each model individually by leveraging their respective strengths. Nevertheless, challenges remained in distinguishing between semantically similar titles or those requiring deep contextual understanding.

In response to these limitations, we transitioned to modern transformer-based models such as BERT, RoBERTa, DistilBERT, and DeBERTa. Each model underwent fine-tuning using our dataset to enhance its ability to classify academic titles accurately.

Experimental results showed that DeBERTaV3 consistently outperformed the other model's thanks to its superior capacity to understand the full linguistic context of titles and efficiently handle complex structures.

Based on these findings, DeBERTaV3 was selected as the final classification model in the system due to its high contextual comprehension and strong performance on the validation data. This iterative model selection strategy enabled the development of a robust and scientifically grounded technical solution.

2.2.3 Training

This section describes the steps for training the DeBERTaV3 model to classify academic books and theses titles. It relies on specialised Python libraries such as Transformers, Scikit-learn, and PyTorch. Below is a detailed breakdown of the process:

A- Importing essential libraries:

- Libraries such as torch, transformers, pandas, and scikit-learn are imported. These are necessary for data processing, implementing deep learning, and model evaluation.

B- Loading and reading the dataset:

- A CSV file containing book titles and their sub-genre classifications is uploaded using Google Colab.
- The data is read using "pandas.read_csv()" with the appropriate encoding.
- Rows with missing values in the main columns ("Title" and "Sub Genre") are removed to ensure data cleanliness.

C- Splitting data into training and validation sets:

- The dataset is split into 80% training and 20% validation using the "train_test_split() "with the stratify option to maintain balanced class distribution.

D- Encoding the labels:

- Textual class labels are transformed into numeric format using LabelEncoder.

- Two mapping dictionaries (label2id and id2label) are created to link label IDs with their original names for use within the model and later interpretation.

E- Tokenising the texts:

- The DebertaV2TokenizerFast tokeniser is used to convert raw text into numerical input features for the model.
- Padding and truncation are applied to limit the input length to a maximum of 256 tokens.

F- Building a custom Dataset class:

- A custom class named BooksDataset is created by subclassing the torch.utils.data class.Dataset, which links tokenised inputs with corresponding labels for training.

G- Loading the pre-trained model:

- The DebertaV2ForSequenceClassification model is loaded from the Transformers library.
- The model is configured with the correct number of output classes and is linked with the label mapping dictionaries.

H- Configuring training settings:

- TrainingArguments is used to specify training parameters such as the number of epochs, batch size, evaluation steps, and checkpoint saving policy.
- The EarlyStoppingCallback is enabled to stop training automatically if no performance improvement is detected after a certain number of steps.

I- Starting the training process:

- The Trainer class is used to assemble the model, datasets, tokeniser, and evaluation function.
- It automatically handles training on the GPU (if available) without requiring manual intervention.

2.2.4 Training Results

Before presenting the accuracy results and training performance, the trained DeBERTaV3 model is evaluated using the built-in "evaluate()" function from the Hugging Face Trainer class. This function assesses the model's performance on both the training and validation datasets by predicting the labels of the input texts and then computing the accuracy using the "compute_metrics" function, which relies on the "accuracy_score "metric from the scikit-learn library. These metrics offer a clear quantitative assessment of the model's ability to classify academic book titles into their respective categories accurately.

A. Model Accuracy:

The results indicate that the model achieved strong performance on both training and validation datasets (see figure 2.3):

- Accuracy on training data: **98.55%**
- Accuracy on validation data: **93.64%**

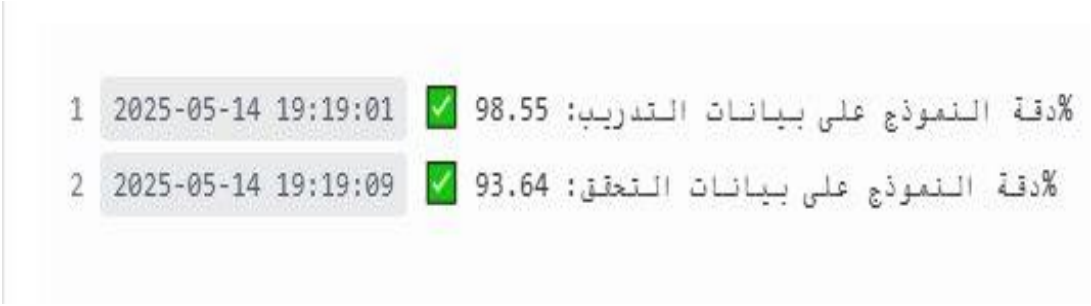

Fig 2.3 Model Accuracy Results on Training and Validation Data

B. Training Curves:

The following figure presents a set of training curves that track the model's performance over time, including the loss, learning rate, global steps, and epochs:

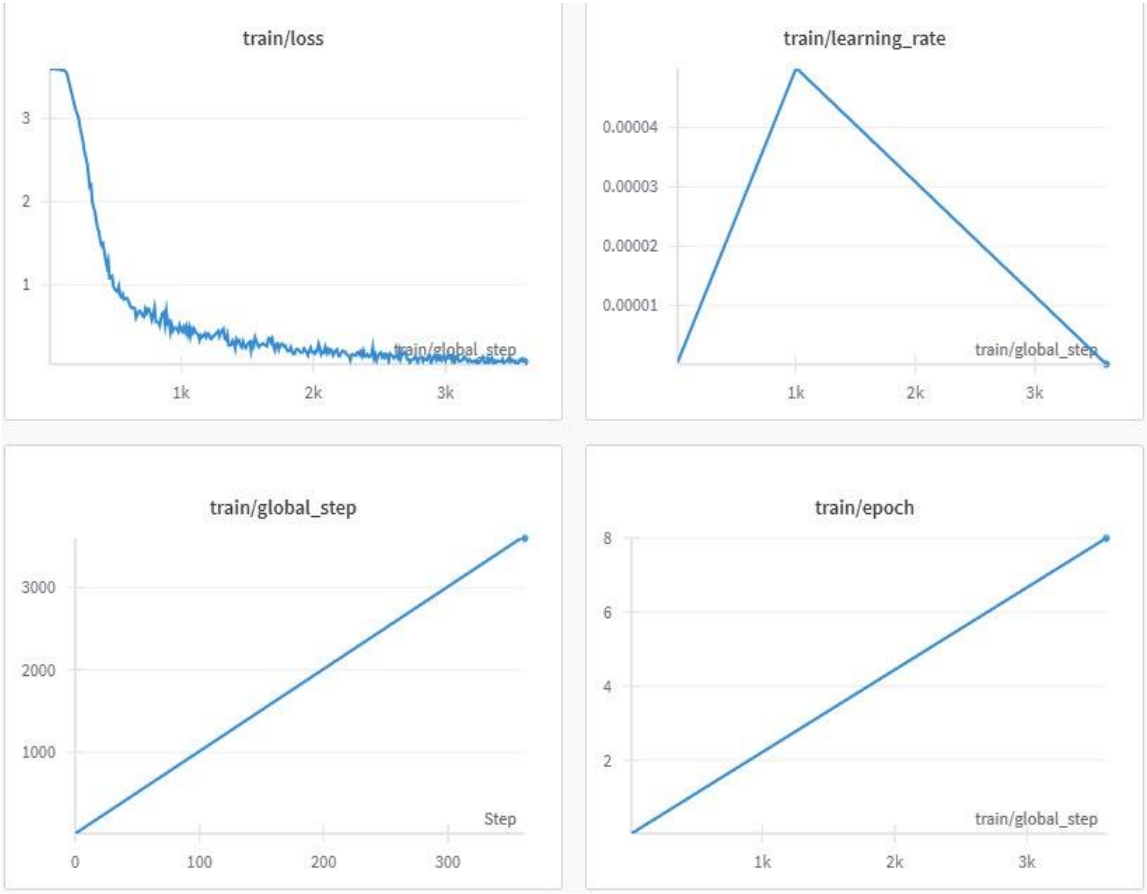

Fig 2.4 Model Training Performance Curves (Training Curves)

Loss Curve:

The curve shows that the loss started high (above 3.5) at the beginning of training, then gradually decreased over time, eventually stabilising at a low value. This indicates a steady improvement in model performance.

Learning Rate:

A dynamic learning rate scheduling strategy was employed. The learning rate started low, increased to a peak, and then gradually decreased. This helps the model explore optimal values early and refine its performance over time.

Global Steps:

The graph shows that the number of training steps exceeded 3400, reflecting the large volume of training performed and contributing to the model's generalisation capability.

Epochs:

The model was trained over eight whole epochs, allowing it to iterate over the data multiple times and progressively improve learning accuracy.

These combined indicators suggest that the training process was stable and effective, as the model successfully reduced the loss gradually while maintaining a stable learning rate and consistent training step progression. Additionally, the number of epochs used appears to be appropriate, as the model demonstrated good performance without clear signs of overfitting.

2.2.5 Title Extraction

The system captures an image of the book cover using a camera and processes it with the EasyOCR library to extract text. It also allows the user to enter the title in case scanning manually fails or if preferred. The extracted text goes through enhancement and cleaning before being displayed in the interface.

Detailed explanation of the code:

A. Capturing the image from the camera:

- A single image is captured when the scan button is pressed using self.camera.read().
- If the capture fails, a warning message is displayed in the user interface.

B. Enhancing image brightness and contrast:

- The function enhance_image_light() is used to convert the image to the LAB colour space.
- CLAHE (Contrast Limited Adaptive Histogram Equalization) is applied to the L (lightness) channel to highlight text features.

C. Resizing the image:

- The image is enlarged using "cv2.resize()" with a scaling factor (fx=1.5, fy=1.5) and INTER_CUBIC interpolation.
- This scaling improves OCR accuracy by making small text more distinguishable.

D. Extracting text with EasyOCR:

The image is converted to RGB format and passed to the appropriate EasyOCR reader depending on the selected language from comboBox_3:

- If the language is Arabic ("Ar"), the Arabic reader "reader_ar" is used.
- For other languages (English or French), the "reader" is used.

E. Filtering and cleaning the extracted text:

- The text segments are combined into a single string.
- A regular expression is used to filter out irrelevant or short tokens: \b[\w\u00C0-\u017F2}[ي-ء,}\b.
- Only meaningful words (minimum two characters) are retained and joined into a cleaned string.

F. Displaying or using the cleaned text:

- The cleaned text is inserted into the input field in the interface for the user to review or edit.
- If no text is extracted, a warning message is shown, allowing the user to retry scanning or manually enter the title.

G. User interaction:

- The user can press the "Scan Text" button to start the extraction process.
- If extraction fails, the user is allowed to enter the title manually.

2.2.6 Translation

If the extracted title is written in a language other than English, the system automatically translates it into English using available translation services. This is done to ensure that all data is standardised in English before the classification process.

Here is a detailed explanation of the code:

A. Detecting the original language of the text:

- The langdetect library is used to identify the original language of the extracted text.
- The "detect()" function determines the language based on the input text, such as Arabic, French, or any other language.

B. Checking if the language is English:

- If the detected language is English (i.e., lang == 'en'), the text remains unchanged, and no translation is needed.
- If the language is not English, the system proceeds to the translation step.

C. Translating the text into English:

- If the language is not English, the googletrans library is used to perform the translation.
- A "Translator()" object is created, and its "translate()" function is called with the source text.
- The source language is automatically detected, and the target language is set to English (dest='en').

D. Returning the translated text:

- If translation is necessary, the text is returned after being translated into English.
- If translation is not required (because the text is already in English), the original text is returned as-is.

E. Ensuring data language consistency before classification:

- This process ensures that all titles submitted for classification are in English, thereby enhancing the accuracy of models trained on English-language data.

2.2.7 Classification

Once the title has been extracted and, if necessary, translated, the system proceeds to classify the title using a pre-trained deep learning model based on the DeBERTaV3 architecture. This step aims to predict the most relevant academic categories to which the title belongs based on its semantic content. The process involves several sub-steps, as explained below:

A. Preparing the text for classification:

- The text (whether original or translated) is passed to the tokeniser, which is explicitly designed for the DeBERTaV3 model.
- The tokeniser converts the raw string into a structured tensor format suitable for input into the deep learning model.
- Padding and truncation are applied to ensure the text does not exceed the model's maximum length limit (512 tokens).

B. Passing the input to the classification model:

- The prepared input is fed into the DeBERTaV3 model for prediction.
- The model returns raw values known as logits, which represent unactivated outputs for each class.

- The "torch.no_grad()" context is used to disable gradient computation, improving performance during inference

C. Calculating classification probabilities:

- The softmax function is applied to the logits to convert them into normalised probabilities.
- These probabilities represent the model's confidence in assigning the input text to each of the possible classes.

D. Extracting the top three predicted classes:

- The top two classes are selected based on the highest probability scores using the "torch.topk()" function.
- The resulting indices are mapped to human-readable labels using the id2label dictionary.
- Each label is then paired with its corresponding confidence score.

E. Displaying classification results:

- The top two predicted classes are displayed in the graphical user interface (GUI), along with their confidence percentages.
- This presentation helps the user understand and validate the model's predictions.

F. Finalising the classification step:

- The classification results are supplemented with additional information, such as training and validation accuracy, to enhance transparency.
- This information enables the user to assess the reliability of the predictions in light of the model's overall performance.

2.2.8 Saving Data to the Database (save_to_database)

This function saves the book title and its associated list of categories to a CSV file, checking for duplicate titles to prevent redundancy, and updates the user interface by displaying appropriate status messages. The process consists of several stages as follows:

A. Reading Input Data from the User Interface:

- The text from the title input field (lineEdit) is retrieved and stripped of any extra spaces using "strip()".
- The text from the categories input field (lineEdit_2), which may contain multiple lines, is also retrieved and cleaned.

- If either field is empty, the saving process is aborted.

B. Defining the CSV File Path:

- The path to the file where data will be saved is set "D:\databasebooks\save.csv".
- This file serves as a simple database for storing titles and categories.

C. Loading Existing Data or Creating a New Structure:

- If the CSV file exists, it is loaded into a pandas DataFrame.
- If it does not exist, a new DataFrame is created with two columns: "title" and "list of categories".

D. Checking for Duplicate Titles:

- It is checked whether the new title already exists in the titles column.
- If the title is found, a warning message is displayed in the user interface, notifying the user that the title already exists.
- The input fields are cleared after displaying the message, and the save operation is cancelled.

E. Adding the New Row:

- If the title is not a duplicate, a new row is created containing the title and categories.
- This new row is appended to the current data, updating the index.

F. Saving Data to the CSV File:

- The updated DataFrame is saved back to the CSV file using UTF-8-SIG encoding to support multiple languages.
- The file is saved without including the index.

G. Updating the User Interface and Displaying Success Messages:

- The input fields are cleared to allow new data entry.
- A "Saved successfully" message is displayed to inform the user that the save was successful.
- A checkmark icon (image) is shown next to the message.
- After 5 seconds, both the message and the icon are automatically cleared.

H. Refreshing the Data Display Table in the Interface:

- The function "load_table_from_csv()" is called to reload the data from the CSV file and display it in the relevant table or UI element.
- This ensures the updated data is immediately visible to the user.

2.3 Conclusion

In this chapter, we addressed the fundamental components upon which the project is based through a systematic presentation of its development stages and operational mechanisms. The focus was placed on the practical aspects that reflect the system's effectiveness and its usability in real-world contexts. The achieved results have shown that this system can serve as a reliable tool that contributes to improving the organisation of academic resources and accelerating access to them. This work stands out as a practical step toward developing tools that enhance the quality of work and enable efficient and structured classification.

CHAPTER 3

Development and Deployment of the Classification System

3.1 Introduction

This chapter represents the technical foundation upon which the intelligent system for classifying books and academic theses is built. It highlights the adopted development environment and the advanced programming libraries used to carry out complex tasks such as text extraction, language processing, and automatic classification. The chapter also addresses the design of the graphical user interface, which plays a key role in facilitating user interaction with the system through a smooth and practical user experience.

3.2 Programming Language

In this project, we have adopted Python as a programming language. Python is a high-level, interpreted programming language known for its simplicity, readability, and versatility. It is widely used in fields such as artificial intelligence, machine learning, and natural language processing due to its extensive ecosystem of libraries and tools.

Python's clear syntax facilitates rapid development and supports complex system design. In this project, Python was utilised across all stages, from data preprocessing and model training to user interface development. The availability of powerful libraries, such as Transformers, Pandas, PyTorch, and EasyOCR, has significantly accelerated development, enhancing system performance and maintainability [15].

3.3 Development Environment Adopted

This section introduces the tools and platforms used for writing, testing, and deploying code.

3.3.1 Google Colab

In addition to choosing Python, we conducted our training using Google Colab. This cloud-based development environment enables us to write and run Python code directly in the browser without the need to install any software or perform complex setups on our local machines. Our decision to use Google Colab was based on several important reasons:

Firstly, Google Colab provides powerful computing resources, including Graphics Processing Units (GPUs) and Tensor Processing Units (TPUs), either for free or at low cost. This is extremely valuable for accelerating the training of large and complex machine learning models. This aspect was crucial for us, as deep learning models typically require high computational power that is not usually available on personal computers.

Secondly, Google Colab offers an interactive environment that simplifies the process of writing and testing code quickly and flexibly. It supports breaking the code into individual cells that can be executed independently, enhancing the development experience and speeding up debugging and modification.

Thirdly, the platform supports collaborative work, allowing team members to easily share notebooks, which facilitates idea exchange, code review, and effective teamwork even when working from different locations.

Fourthly, Google Colab comes pre-equipped with many essential libraries and tools, reducing the time and effort needed to set up the working environment. This allowed us to focus on model development and result analysis rather than dealing with installation and compatibility issues.

Finally, since the DeBERTaV3Base CodeTraining algorithm we adopted for training the model relies heavily on GPU acceleration, Google Colab provided us with the ideal solution to carry out the training efficiently without the need to own high-end hardware or make significant financial investments in infrastructure.

In conclusion, by combining the power of Python with the flexibility of Google Colab, we were able to execute our model training smoothly and efficiently, achieving accurate results for our project [16].

3.3.2 Visual Studio Code

In this project, Visual Studio Code (VS Code) was selected as the primary development environment to connect the interfaces designed using Qt Designer with the AI code that had been previously trained on Google Colab. This choice was based on several technical factors that made VS Code the optimal option for this task.

Firstly, VS Code provides a lightweight yet powerful development environment that offers excellent support for the Python language through a rich collection of extensions, such as support for PySide6, virtual environments, and debugging tools.

Secondly, VS Code facilitated the process of linking interface files in .ui or .py formats with other project components, enabling us to write the application logic code and seamlessly integrate it with the graphical interface.

Thirdly, VS Code offers robust support for virtual environments, helping to organise different Python packages and manage dependencies independently for each project, along with built-in advanced debugging tools that allow step-by-step code tracing and execution, making it easier to detect issues and improve software quality.

Fourthly, VS Code enables work on Python files containing the trained model (Model Inference) and allows calling the model trained on Google Colab through files or links, providing a flexible environment to consolidate the various project elements in one place.

Finally, VS Code is an open-source and free option that runs efficiently across various operating systems, enhancing accessibility and collaboration during project implementation [17].

3.4 Graphical User Interface (GUI) Design Tools

This section covers the tools used to design the visual interface that interacts with the user.

3.4.1 Qt Designer

In designing the user interface for our application, we relied on Qt Designer, an advanced graphical environment used to design graphical user interfaces (GUIs) for Python applications using the PyQt or PySide libraries. Qt Designer enables the creation of interactive interfaces easily and quickly, eliminating the need to manually write interface code, which saves time and reduces programming errors associated with UI design.

We chose Qt Designer for several important reasons:

Firstly, the tool offers a flexible drag-and-drop interface that enables us to arrange interface elements (such as buttons, menus, text boxes, and tables) visually and directly, making the UI design process smoother and more intuitive.

Secondly, designs created with Qt Designer can be saved in the .ui format and then converted into Python files using tools such as pyside6-uic, enabling seamless integration with the rest of the project written in Python.

Thirdly, Qt Designer supports the creation of professional and responsive interfaces, with the ability to customise them precisely to meet user needs. This enabled us to create a straightforward and user-friendly interface that fosters a positive user experience in our training project.

Finally, our use of Qt Designer contributed to the separation between the application logic (Back-End) and the user interface (Front-End), making development and modification easier and providing greater flexibility for maintaining and improving the application [18].

3.5 Used Programming Libraries

Software libraries are fundamental components in modern software development, as they provide ready-to-use. These well-tested functions enable developers to accelerate the development process and reduce the amount of code required. Additionally, they contribute to enhancing software quality and expanding the capabilities of the programming language in use. In the context of our project, several essential libraries were utilised to efficiently and effectively perform various tasks. In this section, we will highlight the most important of these libraries and their vital role in supporting the project's functionalities and achieving its goals:

3.5.1 Regular expression operations (RE)

The re.module in Python is a powerful tool for text processing using regular expressions. It offers functionality similar to that found in Perl. The module supports both Unicode strings (str) and byte strings (bytes), but the two types cannot be mixed in a single operation—i.e., a Unicode string cannot be matched with a pattern of type bytes, and vice versa. Among the key features and capabilities provided by the module:

- **Pattern matching with regular expressions:** The module allows the use of flexible and complex patterns within text, making it easier to search, extract, and replace content.Core functions:
    - re.search(): Searches for the first occurrence of a pattern in the text and returns a Match object if found.
    - re.match(): Checks if the pattern matches only at the beginning of the text.
    - re.findall(): Returns a list of all non-overlapping matches of the pattern in the text.
    - re.sub(): Replaces all matches of the pattern in the text with another string.
- **Pattern objects:** Regular expressions can be compiled into pattern objects using "re.compile()", allowing for pattern reuse and improving performance, especially in repetitive tasks.
- **Match objects:** Provide detailed information about the match, including the start and end positions, as well as the matched text.

The **re** library in Python provides robust and flexible functions for text processing using regular expressions, making searching, matching, editing, and extracting easier. In our project, we used functions such as "**re.findall()**" to extract titles matching a specific pattern, and "**re.sub()**"to clean the text of unwanted characters, which helped us improve result accuracy and reduce processing time [19].

3.5.2 Miscellaneous Operating System Interfaces (OS)

The **os** module in Python provides a unified interface for interacting with the operating system, enabling the execution of tasks such as file and directory management, path handling, environment control, and running system commands. The module supports compatibility across different operating systems, such as Windows, Linux, and macOS, and offers a variety of functions, including:

- **File and directory management**: Such as creating, deleting, and renaming files and folders using functions like "os.mkdir()"," os.remove()", and "os.rename()".
- **Path handling**: Via "os.path", which allows checking for file existence, obtaining absolute or relative paths, and safely joining path components.
- **Retrieving system information**: This includes reading environment variables (os.environ), getting the current working directory (os.getcwd()), and changing directories (os.chdir()).
- **Executing system commands**: Using "os.system() "to run external commands from within a Python program.

The module was used to manage and organise project files efficiently. It enabled operations such as categorising data, checking file existence, and handling file paths across different platforms, thereby improving the system's portability and robustness [20].

3.5.3 Image Processing and Computer Vision (cv2 / OpenCV-Python)

The **cv2** library in OpenCV-Python provides a comprehensive framework for image and video processing, computer vision, and machine learning applications. The library provides a comprehensive set of functions that enable tasks such as image processing, geometric transformations, feature detection, video analysis, and camera calibration. It is cross-platform and widely used in both simple and advanced computer vision projects. Key capabilities provided by the library include:

- **Image handling:** reading, displaying, and saving images using functions like "cv2.imread() "," cv2.imshow()", and "cv2.imwrite()".
- **Image processing and transformations:** Performing operations such as resizing, rotation, colour space conversions, and filtering (e.g., blurring and edge detection).

- **Feature detection and description:** Detecting key points and computing their descriptors for object recognition and tracking using modules like ORB, SIFT, and FAST.
- **Video capture and analysis:** Accessing video streams from files or cameras via "cv2.VideoCapture()", allowing frame-by-frame processing.
- **Camera calibration and 3D reconstruction:** functions for calibrating cameras, correcting distortions, and estimating 3D poses.

In this project, OpenCV (cv2) was used for real-time image capture and preprocessing. It enabled resizing and enhancement of camera input to optimie text extraction and improve OCR accuracy, contributing to a more reliable and responsive classification system [21].

3.5.4 Language Detection (langdetect)

The **langdetect** library is an effective tool for automatically detecting the language of texts in natural language processing applications. It is based on Google's original library algorithms and supports more than 55 languages. It employs statistical analysis based on n-gram models to determine the most likely language of the given input text, without requiring prior knowledge of the language.

Key features provided by the library include:

- **Automatic language detection:** The "detect()" function identifies the language of the text and returns the corresponding language code (e.g., "en" for English or "ar" for Arabic).
- **Multilingual detection:** The "detect_langs()" function returns a list of possible languages with their respective probabilities.
- **Ease of use:** The library provides a straightforward API, facilitating seamless integration into text-processing applications.
- **Wide language support:** The library supports a variety of languages, making it suitable for multilingual applications.

In this project, the **langdetect** library was used to automatically determine the language of input titles, enabling correct routing to translation or classification modules. This enhanced the system's ability to handle multilingual content accurately and efficiently [22].

3.5.5 Googletrans

The **Googletrans** library is a free and unofficial Python library used for machine translation via the Google Translate service. It relies on the **Google Translate Ajax API** and provides translation

and language detection functions directly without requiring a paid subscription to the official API. The library is designed to be fast and easy to use, supporting both single and bulk translations.
The core features offered by the library include:

- **Automatic text translation**: Using the "translate()" function, you can translate text from one language to another. The library supports translating individual texts or batches of texts (Bulk Translation).
- **Automatic language detection**: The "detect()" function automatically identifies the language of a given text, returning the language code (e.g., "en" for English or "ar" for Arabic) along with a confidence score.
- **Ease of use**: The library offers a straightforward API based on the Translator object, facilitating seamless integration into text processing applications.
- **Support for bulk translation**: The "translate()" function accepts a list of texts as input and returns their translations in a single operation, reducing the number of requests and improving efficiency.

In this project, the **googletrans** library was used to translate non-English titles into English, ensuring compatibility with the English-trained classification model. This allowed the system to support multilingual inputs without compromising classification accuracy [23].

3.5.6 Transformers

Transformers is a comprehensive library of pre-trained models spanning natural language processing, computer vision, audio, and multimodal domains. This library is used for inference and training tasks, enabling developers to generate text using large language models, train their own models on their data, or build intelligent applications based on these models.
Key features offered by the Transformers library include:

- **Pipeline:** An easy and fast inference tool that supports a wide range of tasks such as text generation, image segmentation, automatic speech recognition, document question answering, and more.
- **Trainer:** A comprehensive tool for training models that supports advanced features such as mixed precision compilation using the torch. Compile training acceleration with FlashAttention, as well as distributed training across multiple devices.
- **Generate:** Enables fast and efficient text generation using large language models (LLMs) or vision-language models (VLMs), supporting streaming and multiple decoding strategies.

- **Speed and ease of use:** Each model is built from only three main classes: configuration, model, and preprocessor. These models can be used directly with interfaces such as Pipeline or Trainer.
- **Reliance on pre-trained models:** To reduce carbon footprint and computational costs, the library provides ready-to-use pre-trained models without the need to train a new model from scratch, while maintaining state-of-the-art performance.

In this project, the Transformers library was used to implement a multilingual text classification system. Leveraging state-of-the-art pre-trained models significantly improved accuracy, while the pipeline-interface accelerated development, testing, and deployment [24].

3.5.7 EasyOCR

The **EasyOCR** library is an open-source Python library used for optical character recognition (OCR) and extracting text from images such as scanned documents or photographs. The library supports more than 80 languages, making it suitable for multilingual applications.

Key features of the library include:

- **Text extraction from images:** Using the Reader object, which is initialised with the specified language, for example, "easyocr.Reader(['en']) ".
- **Support for process acceleration using GPU:** GPU usage can be enabled when initialising the reader with "easyocr.Reader(['en'], gpu=True) ", which significantly improves image processing speed, especially with large image datasets.
- **Text recognition steps:** include feature extraction from images using deep models, such as ResNet, sequential classification with LSTM networks, and text decoding via the CTC algorithm.
- **Ease of use and integration:** The "readtext()" function allows for easy text extraction from images, returning the results in a structured format.

In this project, the **EasyOCR** library was utilised to extract text quickly and accurately from images, with GPU support enabled to accelerate the analysis process, thereby enhancing system performance and increasing efficiency in handling multilingual documents [25].

3.5.8 Torch (PyTorch) Library

PyTorch is an open-source deep learning framework developed by the FAIR lab at Facebook. It provides a flexible and easy-to-use environment for building and training neural networks, making it popular in both research and industrial applications.

Key features include:

- **Tensors:** Enable fast mathematical operations on the CPU or GPU, with support for gradient tracking.
- **Automatic differentiation:** Provided through the autograd module, module, which computes gradients automatically during training.
- **Neural network design:** Via the "torch.nn" module, allowing modular construction of models using layers and loss functions.
- **GPU acceleration:** Using ".cuda()" or ".to('cuda')" for faster training.
- **Extensive integration:** Works well with tools like TorchVision and Hugging Face

I chose **PyTorch** for this project because it combines ease of use with flexibility in building deep learning models. Its support for dynamic computation graphs allows interactive and seamless model modifications. The simple Pythonic interface speeds up development, which is essential for research and experimentation. Additionally, **PyTorch** supports GPU acceleration, making it efficient when working with large datasets [26].

3.5.9 Scikit-learn Library

Scikit-learn is an open-source Python library that provides simple and efficient tools for predictive data analysis and applying machine learning algorithms. The library supports both supervised and unsupervised learning and integrates easily with other scientific Python libraries, such as NumPy and pandas.

Key features of the library include:

- **Supervised learning algorithms:** A wide range of classification and regression algorithms such as logistic regression, support vector machines (SVM), random forests, and ensemble methods like AdaBoost.
- **Unsupervised learning algorithms:** Tools for clustering (e.g., K-Means), dimensionality reduction (e.g., PCA and ICA), and anomaly detection.
- **Model selection and evaluation:** Provides tools for evaluation using techniques such as cross-validation, grid search, and random search for hyperparameter tuning, along with various performance metrics, including accuracy, recall, and F1-score.
- **Data preprocessing:** Supports techniques such as feature expansion, scaling, encoding categorical variables, handling missing values, and building processing pipelines.
- **Pipelines:** Allow the combination of preprocessing steps and models into a single workflow, simplifying training and evaluation processes.

Scikit-learn was chosen for this project due to its maturity and reliability in the field of machine learning, providing a comprehensive set of tools for efficient data processing, preparation, and

model evaluation. The library is distinguished by its ease of use and compatibility with other libraries such as Transformers and PyTorch, making it ideally suited to the project's working environment [27].

3.5.10 Pandas

Pandas is a powerful and open-source Python library designed to simplify working with structured data. It is widely used for data analysis and manipulation.

Its key features and functionalities include:

- **Flexible data representation:** Through data structures such as two-dimensional tables (DataFrames), which resemble database tables or Excel files.
- **Support for diverse data types:** It handles numerical data, text, dates, and more, making it suitable for analysing complex datasets.
- **Performing statistical analyses:** Provides tools for descriptive statistics, data aggregation, filtering, and sorting.
- **Data cleaning and preprocessing: such as handling missing values and modifying column types,** is essential before applying machine learning models.
- **Seamless integration with other libraries:** Like NumPy, scikit-learn, and Matplotlib, making it a central tool in Python's data science ecosystem.
- **High performance:** Capable of efficiently handling large datasets.
- **Support for multiple data formats:** Including CSV, Excel, JSON, and more.

In this project, the Pandas library was used to facilitate the import, organisation, and processing of data related to book titles. It helped us load data from an external file, handle missing values, and generally improve data quality. The data structure provided by Pandas enabled us to carry out data analysis and preparation operations flexibly and efficiently, paving the way for accurate and effective text classification model applications [28].

3.5.11 PySide6

PySide6 is a powerful, open-source library that enables the development of desktop applications with graphical user interfaces (GUIs) using Python. Also known as "Qt for Python," it serves as the official Python binding for the Qt framework. It is widely used to create cross-platform applications with rich and interactive interfaces.

Key features include:

- **Flexible GUI development**: Offers a wide range of widgets like buttons, menus, and text fields.
- **User interaction support**: Employs a signal and slot mechanism to link user actions to functions.
- **Layout management**: Provides tools like QVBoxLayout and QHBoxLayout for organising the interface.
- **Cross-platform compatibility**: Supports Windows, macOS, and Linux with minimal code changes.
- **Design tool integration**: Compatible with Qt Designer for drag-and-drop UI creation.

In this project, PySide6 was used to build the graphical interface of the title classification system. It facilitated the creation of windows, the addition of UI elements, and the connection of user interactions to functionality, enhancing user experience and interactivity [29].

3.6 Graphical User Interfaces

In this section, we present the design of the system's graphical user interfaces, which were developed using the PySide6 library. These interfaces are intended to facilitate data entry, display results, and enhance the overall user experience. Main Interface

3.6.1 Main Interface

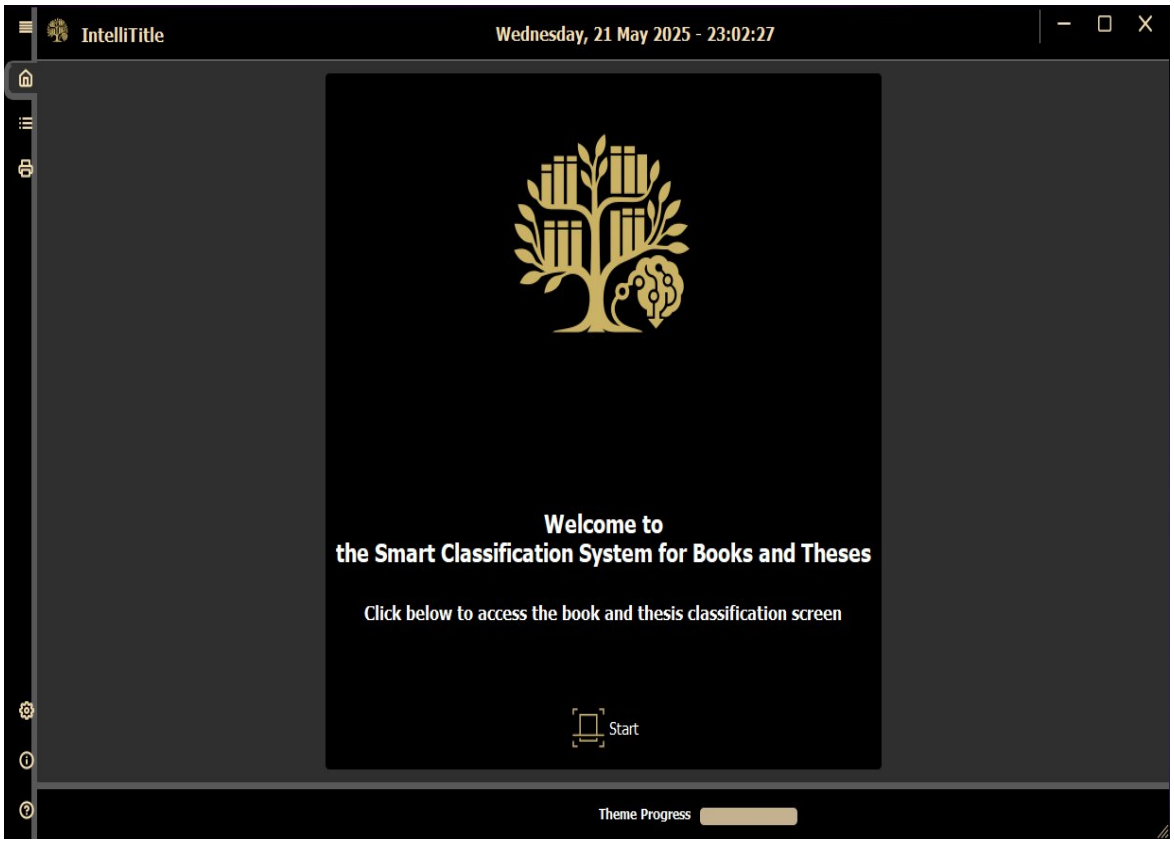

Fig 3.1 Main Interface of the System

When the system starts, the user is presented with the main interface, which includes the system's logo and the project name "IntelliTitle". The interface features a welcome message and a "Start" button that directs the user to the title classification screen.

3.6.2 Classification Interface

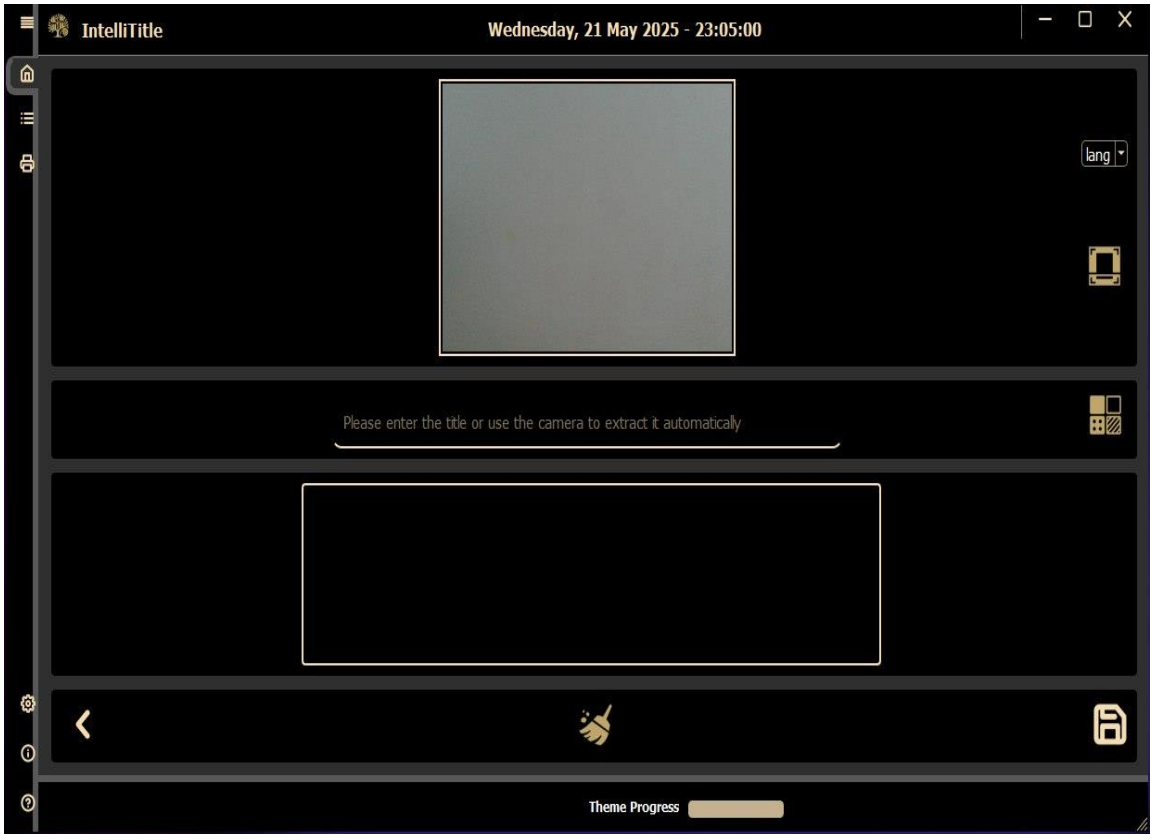

Fig 3.2 Classification Interface

This interface provides the user with the essential tools to classify the title of a book or academic thesis. The user can either capture an image of the book cover or manually enter the title, then select the language and start the classification process. The classification result is displayed in the designated box below, with options to return to the home page, clear the fields, or save the results. The interface is designed to be simple and user-friendly.

3.6.3 Report Interface

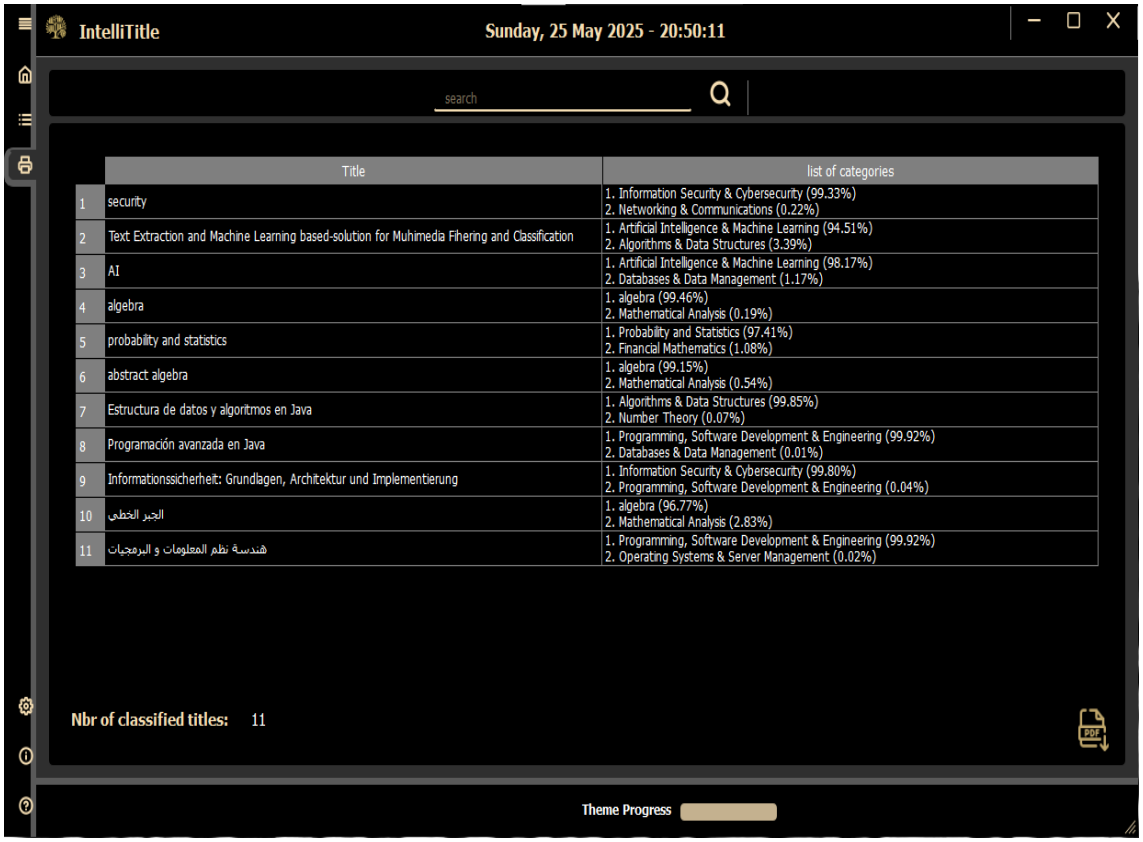

Fig 3.3 Report Interface

This interface displays a list of titles that have been classified and saved in the system. It features a detailed table that includes the book or thesis title, along with the suggested classifications and the confidence scores for each classification.

Additionally, the interface features a dedicated search area, enabling users to search for books or titles by various categories. This makes it easier for users to quickly and easily find books classified under a specific category.

At the bottom of the interface, a "Print" button is available, enabling users to print the displayed list and facilitate the process of keeping a hard copy of the results or sharing them.

3.6.4 Data Analysis Interface

Fig 3.4 Data Analysis Interface

This interface displays system performance statistics, including training accuracy, testing accuracy, the number of titles, and the number of classifications. It also shows the most important keywords for each classification, which helps in understanding the classification mechanism and analysing the results.

3.6.5 Information Interface

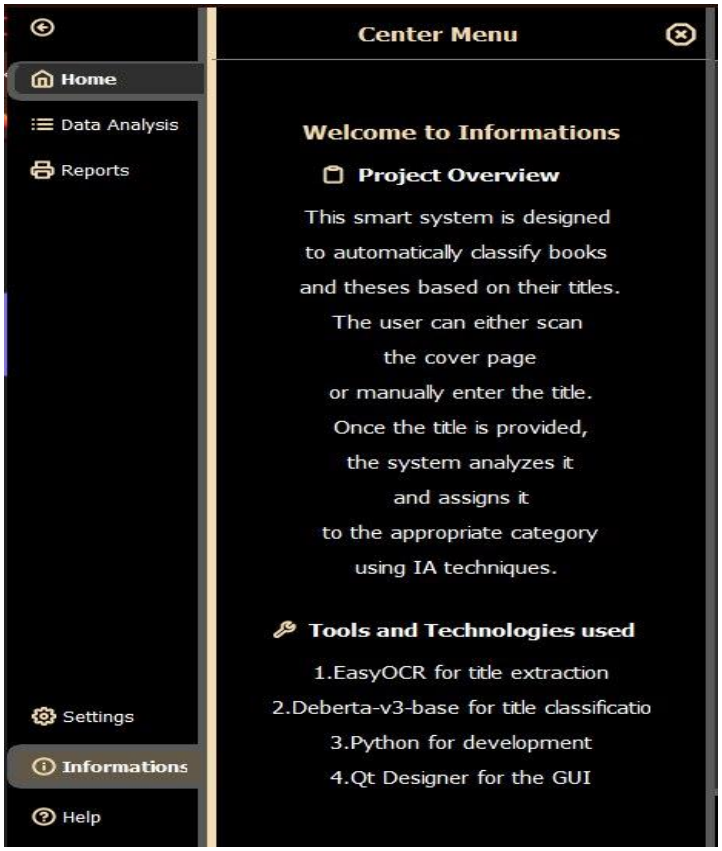

Fig 3.5 Information Interface

This interface provides the user with an overview of the system, offering a brief explanation of the innovative system's functionality and the main tools and technologies used. Its purpose is to introduce the user or reader to the concept of the system and its core functions simply and clearly without delving into technical details.

3.6.6 Help Interface

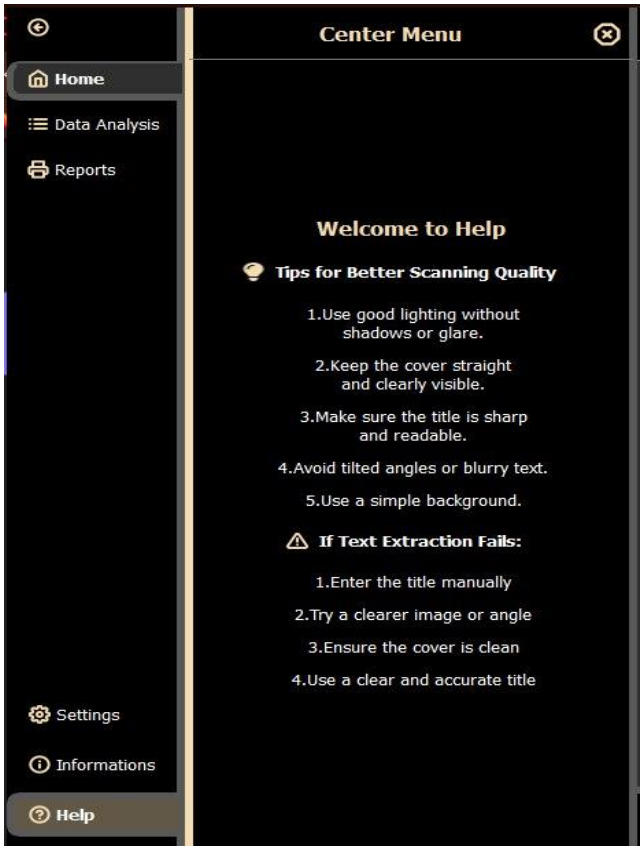

Fig 3.6 Help Interface

This interface offers a set of tips and guidelines to help users enhance the quality of scanning and text extraction from images. It also offers practical solutions in case the user encounters difficulties in extracting text, thereby enhancing the user experience and ensuring efficient interaction with the system.

3.6.7 Settings Interface

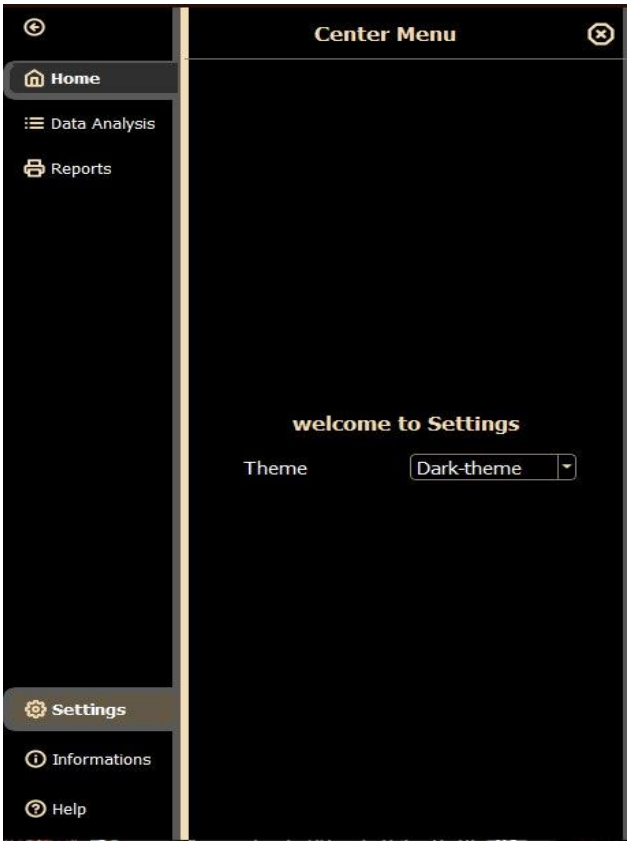

Fig 3.7 Settings Interface

This interface allows users to customise the system's appearance by changing the theme (such as enabling dark mode) according to their personal preferences. It is designed to be user-friendly and straightforward, offering flexibility and comfort during system use.

Additionally, the system supports both dark and light modes to enhance visual comfort in different environments. Figure 3.8 below shows an example of the main interface in light mode.

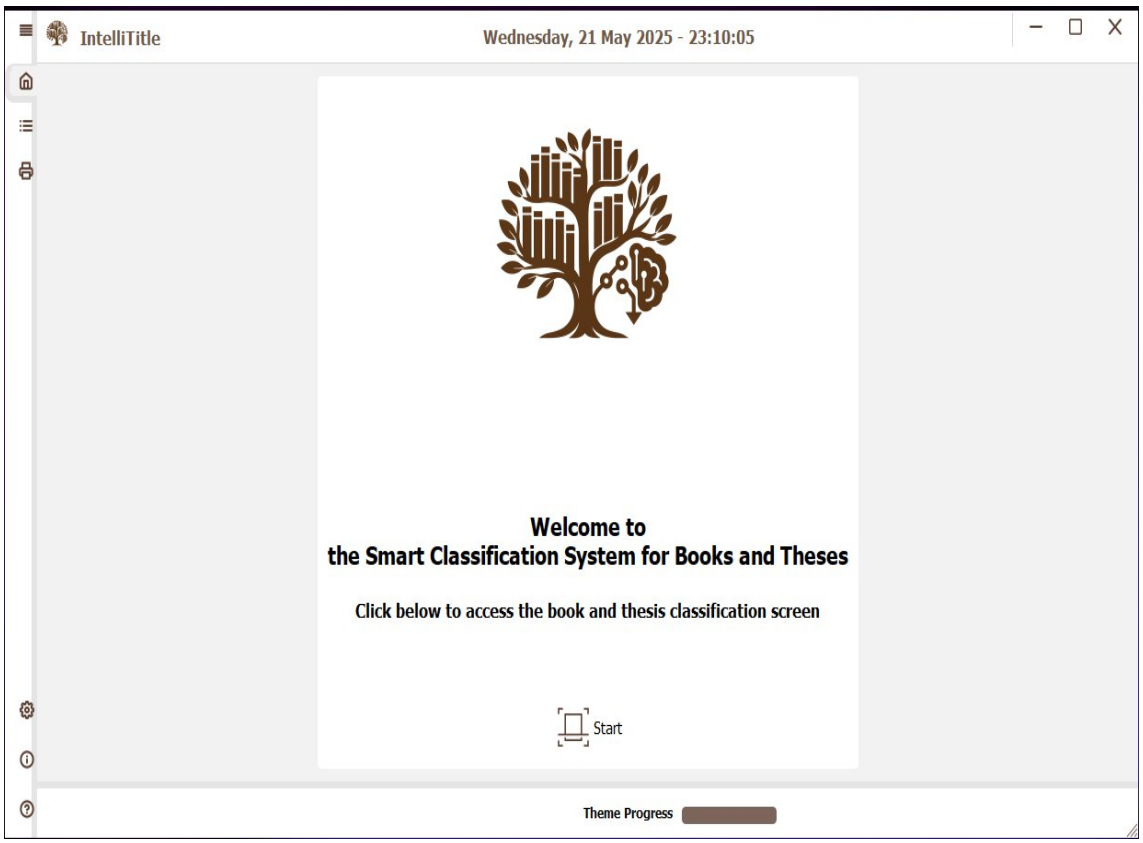

Fig 3.8 Main Interface of the System

3.7 Conclusion

This chapter presented the technical environment, tools, and graphical interfaces used in the development of the intelligent classification system. By highlighting the main features and design choices, it demonstrated how the integration of modern libraries and user-friendly interfaces contributes to the system's efficiency and usability.

General Conclusion

In this dissertation, we present a practical proposal for classifying book and thesis titles using Artificial Intelligence (AI) and Natural Language Processing (NLP) techniques within the context of developing intelligent systems that keep pace with the digital transformation in managing and organising academic content.

The project implementation results demonstrated the vast potential of these technologies, particularly the deep classification model DeBERTaV3, which was trained on a multi-category academic dataset and showed high capability in handling titles in different languages after automatic translation. The use of Optical Character Recognition (OCR) techniques, facilitated by the EasyOCR library, enabled effective text extraction from images. The langdetect library helped identify the language of the text, and the googletrans library automatically translated it into English before the classification phase.

The system was evaluated through classification experiments on actual titles, with results showing high accuracy in categorising academic fields such as "Economic Sciences," "Law," and "Technical Sciences," reflecting the model's effectiveness in analysing short texts with focused meanings.

The Python programming language, along with its diverse libraries, played a fundamental role in implementing this project. Libraries such as Hugging Face's Transformers facilitated model loading and fine-tuning. Pandas and Torch handled data processing and model execution. Meanwhile, PySide6 enabled the building of an interactive graphical user interface that eases user interaction with the system and stores results in an integrated Excel database.

Regarding future improvements, we recommend expanding the training dataset to include more specialised disciplines, supporting full-text analysis instead of limiting classification to titles only, and integrating the system with digital library platforms to enable full automation between classification and academic content management in educational institutions.

References

[1] M. Kessasra, *Le classement dans les bibliothèques universitaires algériennes*, Mémoire de Master, Université Mentouri de Constantine, 2007.

[2] M. Jamdade, P. Jamdade, B. Panage, and V. Mugade, "Classification and Its Development: A Study," *Review Of Research*, vol. 1, no. 12, pp. 1–4, 2012.

[3] Ministère de la Culture, *Résumé du classement dans les bibliothèques et le système décimal Dewey*, Direction des centres culturels, Damas, 2011.

[4] African Library Project, *Organising Information Books – Classification (ALP Manual, Version 7)*, African Library Project, 2012.

[5] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 4th ed., Global Edition, Pearson Education Limited, Harlow, 2022.

[6] S. Roy, S. V. Mallikraj, S. Moradia, G. Shantha, S. Aravind, and S. Shivaprakash, "The Impact of Artificial Intelligence on Cataloging and Classification Systems in Modern Libraries," *Library Progress International*, vol. 44, no. 3, pp. 769–773, 2024.

[7] V. Honavar, *Artificial Intelligence: An Overview*, Artificial Intelligence Research Laboratory, Iowa State University, Ames, 2006.

[8] G. L. N. Jayaprada, *Natural Language Processing (R20A6609)*, Department of Computational Intelligence, Malla Reddy College of Engineering & Technology, Hyderabad, 2024.

[9] H. Allam, L. Makubvure, B. Gyamfi, K. N. Graham, and K. Akinwolere, "Text Classification: How Machine Learning Is Revolutionizing Text Categorization," *Information*, vol. 16, no. 2, p. 130, 2025.

[10] M. A. Shah, M. J. Iqbal, N. Noreen, and I. Ahmed, "An Automated Text Document Classification Framework using BERT," *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 14, no. 3, pp. 279–285, 2023.

[11] E. Jadon and R. Sharma, "Data Mining: Document Classification using Naïve Bayes Classifier," *International Journal of Computer Applications (IJCA)*, vol. 167, no. 6, pp. 13–16, 2017.

[12] F. Riandari, H. T. Sihotang, T. Tarigan, and M. Rafli, "Classification of Book Types Using the Support Vector Machine (SVM) Method," *Jurnal Mantik*, vol. 6, no. 1, pp. 43–49, 2022.

[13] E. Giannopoulou and N. Mitrou, "An AI-Based Methodology for the Automatic Classification of a Multi-class Ebook Collection Using Information From the Tables of Contents," *IEEE Access*, vol. 8, pp. 218658–218672, 2020.

[14] M. P. Satija, "Colon Classification (CC)," *ISKO Encyclopedia of Knowledge Organization*, 2017.

[15] Python documentation, "What is Python?" [Online]. Available: https://docs.python.org/3/faq/general.html#what-is-python. Accessed: May 26, 2025.

[16] P. Raja, "What is Google Colab?" *Scaler Topics*. [Online]. Available: https://www.scaler.com/topics/what-is-google-colab/. Accessed: May 19, 2025.

[17] Visual Studio Code, "Documentation." [Online]. Available: https://code.visualstudio.com/docs. Accessed: May 21, 2025.

[18] L. Pozo Ramos, "Qt Designer and Python: Build Your GUI Applications Faster," *Real Python*. [Online]. Available: https://www.realpython.com/qt-designer-python/. Accessed: May 19, 2025.

[19] Python Software Foundation, "re — Regular expression operations," *Python 3.13.3 Documentation*. [Online]. Available: https://docs.python.org/3/library/re.html. Accessed: May 19, 2025.

[20] Python Software Foundation, "os — Miscellaneous operating system interfaces," *Python 3.13.3 Documentation*. [Online]. Available: https://docs.python.org/3/library/os.html. Accessed: May 19, 2025.

[21] OpenCV, "OpenCV-Python Tutorials." [Online]. Available: https://docs.opencv.org/4.x/d6/d00/tutorial_py_root.html. Accessed: May 20, 2025.

[22] GeeksforGeeks, "Detect an Unknown Language using Python." [Online]. Available: https://www.geeksforgeeks.org/detect-an-unknown-language-using-python/. Accessed: May 20, 2025.

[23] readthedocs, "py-googletrans Documentation." [Online]. Available: https://py-googletrans.readthedocs.io/en/latest/. Accessed: May 20, 2025.

[24] Hugging Face, "Transformers Documentation." [Online]. Available: https://huggingface.co/docs/transformers/en/index. Accessed: May 20, 2025.

[25] L. Ueno, "How to Use EasyOCR," *Roboflow Blog*. [Online]. Available: https://blog.roboflow.com/how-to-use-easyocr/. Accessed: May 20, 2025.

[26] PyTorch. [Online]. Available: https://docs.pytorch.org/docs/stable/index.html. Accessed: May 21, 2025.

[27] Scikit-learn, "User Guide." [Online]. Available: https://scikit-learn.org/stable/user_guide.html. Accessed: May 21, 2025.

[28] Pandas, "API Reference." [Online].

Available: https://pandas.pydata.org/docs/reference/index.html. Accessed: May 21, 2025.

[29] Sibadil.com, "How to Create Desktop Applications Using PySide in Python," [Online]. Available: https://sibadil.com/البرمجة/1833/كيفية-إنشاء-تطبيقات-سطح-المكتب-باست. Accessed: May 21, 2025.

[30] B. Sen, “NLP API for Text Classification (Natural Language Processing),” *TextCortex*, 7 Apr. 2023. [Online]. Available: https://textcortex.com/post/nlp-api-for-text-classification.