ENSA
ÉCOLE NATIONALE DES SCIENCES APPLIQUÉES
KHOURIBGA

REPORT

2nd year Master Big Data et Aide la Dcision

Data Integration Project

Written by

Nizar BENNANI Oussama QOUTI
Abdelouhab BELLA Abdessamad HAIDA

Supervised by

Pr. Imad HAFIDI

Academic Year: 2023-2024

Chapter 1

Introduction

Data integration is a critical aspect of managing and analyzing information in todays data-driven world. It enables the consolidation of diverse data sources into a unified structure, facilitating meaningful analysis and decision-making. This report details the experiences and outcomes of a workshop conducted as part of the Data Integration module. The workshop focused on equipping participants with practical skills in using existing tools and techniques to extract, integrate, and utilize academic data effectively.

Unlike research endeavors aimed at introducing novel methodologies, the primary objective of this workshop was to familiarize participants with established tools and frameworks. This learning experience emphasized hands-on application, allowing participants to understand the practical aspects of data integration using proven techniques. The workshops goals were threefold:

- **Data Extraction:** Participants learned to use scripting tools to automatically retrieve detailed information about authors and their publications from academic databases such as Scopus, Web of Science, and Google Scholar. The emphasis was on understanding the capabilities of these tools and applying them effectively.

- **Data Integration:** The workshop addressed common challenges in integrating data from diverse sources. Using mapping and schema matching techniques, participants learned how to handle disparities in data formats and structures, enabling the creation of a cohesive and structured dataset.

- **Journal Recommendation:** By utilizing the extracted metadata, participants explored techniques for recommending suitable scientific journals based on key attributes such as title, keywords, and abstracts. The focus was on lever-

aging existing algorithms and natural language processing (NLP) methods to achieve meaningful results.

Through these objectives, the workshop provided participants with a deeper understanding of practical tools and processes in data integration. It did not aim to push the boundaries of innovation in this field but rather to impart a solid foundation in utilizing existing technologies effectively. This report delves into the methodologies, challenges encountered, and insights gained during the workshop, offering a comprehensive overview of its activities and outcomes.

Chapter 2

Data Extraction

2.1 Overview of Databases: Scopus, Web of Science, and Google Scholar, SCImago, ORCID

In the realm of academic research, databases such as **Scopus**, **Web of Science**, and **Google Scholar** serve as essential tools for accessing scholarly literature. Each database has distinct features, strengths, and weaknesses that cater to various research needs.

2.1.1 Scopus

Scopus is recognized as one of the largest abstract and citation databases, encompassing over 21,000 peer-reviewed journals across diverse fields such as health sciences, physical sciences, social sciences, and humanities. It provides comprehensive indexing with more than 27 million abstracts and extensive citation data dating back to 1966. Scopus employs a rigorous selection process for journal inclusion, ensuring high-quality content. Its advanced search capabilities allow users to utilize Boolean operators and filter results by document type, subject area, and date range. Additionally, Scopus offers metrics like CiteScore and Scimago Journal Rank for assessing journal impact, making it a valuable resource for researchers seeking reliable bibliometric data.

2.1.2 Web of Science

Web of Science (WoS) offers a similarly curated collection of scholarly content but is particularly noted for its strength in the sciences and social sciences. It indexes

approximately 13,100 journals and provides access to over 10 million conference proceedings. WoS utilizes a selective approach to source inclusion based on expert editorial criteria, which helps maintain the quality of indexed materials. The database allows for cited reference searches, enabling researchers to track citations backward and forward in time. Its citation analysis tools are highly regarded in academic evaluations, although its coverage in the humanities may be more limited compared to Scopus.

2.1.3 Google Scholar

In contrast, **Google Scholar** adopts a more inclusive approach by automatically indexing a wide array of scholarly documents available on the internet. This includes not only peer-reviewed articles but also gray literature such as theses, conference papers, and technical reports. While Google Scholar's broad coverage allows it to capture more interdisciplinary content and non-English publications than its counterparts, it lacks the rigorous quality control seen in Scopus and WoS. Consequently, users may encounter issues such as duplicate entries or incomplete bibliographic information. Despite these drawbacks, Google Scholar's user-friendly interface and comprehensive search capabilities make it an accessible tool for researchers at all levels.

2.1.4 SCImago

To obtain critical information about journals that was not available on other platforms, we relied on SCImago. The SCImago Journal Rank (SJR) is a widely recognized metric developed by the SCImago research group to assess the prestige and impact of academic journals. Based on citation data from the Scopus database, it evaluates how frequently content from a journal is cited in other journals over a three-year period, considering the prestige of the citing journals. SJR covers all academic disciplines, offering a comprehensive performance overview of journals. It reflects both the average number of weighted citations per document and quartile classifications, which divide journals into four groups (Q1 to Q4) based on their SJR values. Additionally, an H-index can be derived for journals, providing insights into their productivity and citation impact. SJR serves as an alternative to traditional metrics like the Impact Factor, enabling fairer comparisons across disciplines by factoring in citation quality and the prestige of the citing sources.

2.1.5 ORCID

To ensure accurate attribution of researchers' work, we relied on ORCID to retrieve their unique identifiers. ORCID (Open Researcher and Contributor ID) is a unique identifier for researchers, which ensures their work is properly attributed and easily discoverable. Published by ORCID, Inc., it integrates with research databases and funding agencies, linking researchers' profiles to their publications, grants, and professional activities. By reducing name ambiguity and increasing visibility, ORCID plays a critical role in improving attribution and ensuring compliance with funder requirements. Together, SJR and ORCID enhance transparency and efficiency in academic publishing, supporting journal evaluation and ensuring proper acknowledgment of scholarly contributions.

2.2 Tools and Methods Used for Data Extraction

In this scraping project, we utilized several powerful tools, each offering unique capabilities to streamline the process.

2.2.1 Burp Suite

We used **Burp Suite** as a comprehensive platform for web application security testing. It provided a suite of tools for tasks such as web crawling, vulnerability scanning, and manual testing. The Proxy tool allowed us to intercept and modify HTTP/S traffic between the browser and the server, enabling detailed analysis of requests and responses. Additionally, Burp Suite offered tools like the Repeater for manual request manipulation, the Intruder for automated attack testing, and the Scanner for identifying vulnerabilities. These tools collectively enabled us to perform thorough security assessments of the web applications we targeted.

2.2.2 BeautifulSoup

We relied on **BeautifulSoup** for parsing HTML and XML documents. This Python library allowed us to navigate, search, and modify the parse tree easily, making it ideal for extracting data from web pages. BeautifulSoup supports different parsers, such as the built-in HTML parser and lxml, giving us flexibility in how we parsed the content. Its intuitive syntax and powerful search features made it an essential tool for scraping static web pages.

2.2.3 Selenium

We used **Selenium** to automate interactions with dynamic websites. Unlike other tools, Selenium allows us to simulate user actions such as clicking buttons, entering text, and navigating pages. This was particularly useful for websites that rely on JavaScript to load content. Selenium's support for multiple programming languages, including Python, and its compatibility with various browsers, made it the ideal tool for automating tasks and scraping content from complex, JavaScript-heavy web applications.

Each of these tools played a crucial role in the success of our scraping project, providing specialized functionalities that addressed different aspects of the task.

2.3 Results of the Extraction

2.3.1 Scopus

In this section, we explore the metadata extracted from Scopus, a prominent database of academic publications. The data is provided in XML format, which includes detailed information about the author, their publications, and co-authors. Each record is organized in a hierarchical structure, with the root element containing the authors information and a list of their associated publications. The metadata offers a variety of attributes, such as author details (name, affiliation, H-index), publication specifics (title, publication year, citation count, DOI), and additional metadata related to co-authors. Below is a general outline of the structure of this data.

Structure Breakdown:

1. **Root Element:** `<Authors>`
    - This is the main container for all author-related data.
2. **Author Information**
    - `<Nom_complet_de_l_auteur>`: The full name of the author (type: string).
    - `<Pays_d_affiliation>`: The country of affiliation (type: string).
    - `<H_index_de_l_auteur>`: The H-index of the author (type: integer).
    - `<FWCI_Field_Weighted_Citation_Impact>`: The Field Weighted Citation Impact (type: float).

- `<Co-auteurs>`: The number of co-authors (type: integer).
- `<Citations_totales>`: The total number of citations for the author (type: integer).

3. **Documents Information**
   - `<Documents>`: A container for a list of the authors documents (articles, book chapters, etc.).
   - **Item (Document)**
     - `<url>`: The URL of the document (type: string).
     - `<eid>`: The Scopus EID (identifier) for the document (type: string).
     - `<Authors>`: A list of authors involved in the document.
       - Each author is listed within `<item>` tags (type: string).
     - `<Article_Title>`: The title of the article (type: string).
     - `<Publication_Year>`: The publication year (type: string).
     - `<Source_Title>`: The source or journal title (type: string).
     - `<Citation_Count>`: The number of citations for this specific document (type: integer).
     - `<DOI>`: The Digital Object Identifier for the document (type: string).
     - `<Abstract>`: A brief abstract of the document (type: string).
     - `<Author_Keywords>`: Keywords associated with the document (type: string).
     - `<Editors>`: Information about the editors, if applicable (type: string, empty if not available).
     - `<Document_Type>`: The type of document (e.g., book chapter, journal article) (type: string).

4. **Co-authors Information**
   - `<Co_auteurs>`: A list of co-authors.
     - **Item (Co-author)**
       - `<name>`: The name of the co-author (type: string).
       - `<link>`: A link to the co-author's profile or more information (type: string).

Example Data Structure in XML Format:

```
<Authors>
    <Nom_complet_de_l_auteur type="str">Author Name</Nom_complet_de_l_auteur>
    <Pays_d_affiliation type="str">Country</Pays_d_affiliation>
    <H_index_de_l_auteur type="int">14</H_index_de_l_auteur>
    <FWCI_Field_Weighted_Citation_Impact type="float">0.90</FWCI_Field_Weighted_Citation_Impact>
    <Co-auteurs type="int">Number of Co-authors</Co-auteurs>
    <Citations_totales type="int">Total Citations</Citations_totales>
    <Documents>
        <item>
            <url type="str">Document URL</url>
            <eid type="str">Document EID</eid>
            <Authors>
                <item type="str">Author1</item>
                <item type="str">Author2</item>
            </Authors>
            <Article_Title type="str">Document Title</Article_Title>
            <Publication_Year type="str">Year</Publication_Year>
            <Source_Title type="str">Source Title</Source_Title>
            <Citation_Count type="int">Citation Count</Citation_Count>
            <DOI type="str">DOI</DOI>
            <Abstract type="str">Document Abstract</Abstract>
            <Author_Keywords type="str">Keywords</Author_Keywords>
            <Editors type="str"/>
            <Document_Type type="str">Document Type</Document_Type>
        </item>
    </Documents>
    <Co_auteurs>
        <item>
            <name type="str">Co-author Name</name>
            <link type="str">Link to Co-author Profile</link>
        </item>
    </Co_auteurs>
</Authors>
```

2.3.2 Web of Science

We now present the metadata extracted from Web of Science, another major academic database. The data is structured in a JSON format, providing details about

authors, their publications, and co-authors. Each entry includes the author's name, H-index, total citations, and a list of co-authors. The articles are also listed with metadata such as the authors, title, publication year, citation count, keywords, and document type. The information about the source of the article includes additional details, such as the journal name, publisher, and impact factor when available. Below is the general structure of the data.

Structure Breakdown:

1. **Root Element**: `<Auteur>`
    - This is the main container for all author-related data.
2. **Author Information:**
    - **Nom de l'auteur**: The full name of the author (type: string).
    - **H-index**: The H-index of the author (type: string).
    - **Co-auteurs**: A list of co-authors (type: array of strings).
    - **Nombre total de citations**: The total number of citations for the author (type: string).
3. **Articles Information:**
    - **Articles**: A container for a list of the author's publications (articles, conference papers, etc.).
    - **Item (Article):**
        - **Auteurs**: A list of authors involved in the article (type: array of strings).
        - **Titre de l'article**: The title of the article (type: string).
        - **Anne de publication**: The publication year (type: string).
        - **Titre de la source**: The title of the source or journal (type: string, empty if not available).
        - **Nombre de citations**: The number of citations for this article (type: string).
        - **DOI**: The Digital Object Identifier for the article (type: string, empty if not available).
        - **Rsum**: A brief abstract or description of the article (type: string).
        - **Mots-cls de l'auteur**: Keywords associated with the article (type: array of strings).

- **Type de document**: The type of document (e.g., journal article, conference paper) (type: string).
- **Informations sur la revue**: Information about the journal where the article was published.

4. **Information about the Journal:**

   - **Nom de la revue**: The name of the journal (type: string, empty if not available).
   - **diteur**: The publisher of the journal (type: string).
   - **ISSN**: The International Standard Serial Number for the journal (type: string, empty if not available).
   - **Porte thmatique**: The thematic scope of the journal (type: string).
   - **Index**: The index in which the journal is included (e.g., Web of Science) (type: string).
   - **H-index de la revue**: The H-index of the journal (type: string).
   - **Quartile de la revue**: The quartile ranking of the journal (type: string, empty if not available).
   - **Score SJR**: The SCImago Journal Rank (SJR) score (type: string).
   - **Impact factor**: The impact factor of the journal (type: string, empty if not available).

Example Data Structure in JSON Format:

```
[{
    "Auteur": {
        "Nom de l'auteur": "Author Name",
        "H-index": "Author H-index",
        "Co-auteurs": [
            "Co-author 1",
            "Co-author 2",
            "Co-author 3",
            "Co-author 4",
            "Co-author 5"
        ],
        "Nombre total de citations": "Total Citations"
    },
    "Articles": [
        {
```

```
            "Auteurs": [
                "Author 1",
                "Co-author 1",
                "Co-author 2",
                "Co-author 3",
                "Journal Name"
            ],
            "Titre de l'article": "Article Title",
            "Ann e de publication": "Publication Year",
            "Titre de la source": "Source Title",
            "Nombre de citations": "Citation Count",
            "DOI": "DOI Link",
            "R sum ": "Abstract of the article",
            "Mots-cl s de l'auteur": [
                "Keyword 1",
                "Keyword 2",
                "Keyword 3",
                "Keyword 4"
            ],
            "Type de document": "Document Type",
            "Informations sur la revue": {
                "Nom de la revue": "Journal Name",
                " diteur": "Publisher Name",
                "ISSN": "Journal ISSN",
                "Port e th matique": "Thematic Scope",
                "Index": "Index Name",
                "H-index de la revue": "Journal H-index",
                "Quartile de la revue": "Journal Quartile",
                "Score SJR": "SJR Score",
                "Impact factor": "Impact Factor"
            }
        }
    ]
}]
```

2.3.3 Google Scholar

The following JSON structure represents metadata about a specific author retrieved from Google Scholar. It includes essential information about the author's profile, their publications, and associated details like co-authors and journal metrics. This format is organized to provide an easy way to access key academic data for further analysis or integration.

Structure Breakdown:

- **Author Information**:
  - `"author_name"`: The name of the author.
  - `"profile_url"`: The URL linking to the author's Google Scholar profile.
  - `"h_index"`: The author's H-index, a measure of their citation impact.
  - `"total_citations"`: The total number of citations the author has received.
  - `"keywords"`: A list of keywords relevant to the author's research area.
- **Publications**:
  - Each publication contains:
    - `"title"`: The title of the publication.
    - `"type_of_document"`: The type of the document (e.g., article, conference paper).
    - **Journal Information**:
      `"Name"`: The journal name where the publication appears.
      `"Publisher"`: The publisher of the journal.
      `"ISSN"`: The ISSN (International Standard Serial Number) of the journal.
      `"Thematic Scope"`: The field or area of research the journal focuses on.
      `"H-index"`: The journals H-index.
      `"Quartile"`: The journal's quartile ranking by year.
      `"SJR"`: The SJR (SCImago Journal Rank) indicator for the journal.
      `"Impact factor"`: The impact factor of the journal.
    - `"year"`: The publication year.
    - `"citations"`: The number of citations the publication has received.
    - `"authors"`: A list of authors who contributed to the publication.
    - `"abstract"`: A brief abstract summarizing the publication.
    - `"doi"`: The DOI (Digital Object Identifier) for the publication.
- **Co-authors**:

– "co_authors": A list of co-authors who worked with the main author on their publications.

Example Data Structure in JSON Format:

```
{
    "https://scholar.google.com/citations?user=EXAMPLE_ID" or 'author name': {
        "author_name": "Author Name",
        "profile_url": "https://scholar.google.com/citations?user=EXAMPLE_ID",
        "h_index": "Author H-index",
        "total_citations": "Total Citations",
        "keywords": ["Keyword 1", "Keyword 2", "Keyword 3"],
        "publications": [
            {
                "title": "Publication Title",
                "type_of_document": "Document Type",
                "journal": {
                    "Name": "Journal Name",
                    "Publisher": "Publisher Name",
                    "ISSN": "Journal ISSN",
                    "Thematic Scope": "Thematic Scope",
                    "H-index": "Journal H-index",
                    "Quartile": {
                        "Field": {
                            "Year": "Quartile"
                        }
                    },
                    "SJR": {
                        "Year": "SJR Score"
                    },
                    "Impact factor": "Impact Factor"
                },
                "year": "Publication Year",
                "citations": "Citation Count",
                "authors": ["Author 1", "Author 2", "Author 3"],
                "abstract": "Publication Abstract",
                "doi": "DOI Link"
            }
        ],
        "co_authors": ["Co-author 1", "Co-author 2", "Co-author 3"]
    }
}

```

```
39 // co-authors data
```

Chapter 3

Data Integration

3.1 Introduction to Data Integration

Data integration refers to the process of combining data from multiple sources into a single, coherent dataset that can be used for analysis or decision-making. This process is essential in various fields, including academic research, business intelligence, and data science, where information is often scattered across different platforms with varying formats and structures. Effective data integration ensures that data inconsistencies, redundancies, and conflicts are resolved, enabling seamless usage of the integrated information.

In the context of this workshop, participants decided on the schema for data integration based on the objectives of scrapping metadata from three major academic databases: Scopus, Web of Science, and Google Scholar. The integration process involved working with two distinct data structures, XML and JSON, to better understand and practice the challenges of handling diverse formats. Each of these sources provides unique and valuable information, but their differing schemas and data formats present significant challenges for integration.

By addressing these challenges through schema mapping, attribute alignment, and conflict resolution, participants were able to produce a unified dataset that served as the foundation for further analysis, such as journal recommendations. This section explores the key concepts, methodologies, and tools used in achieving effective data integration during the workshop.

3.2 Schema Mapping and Matching

Schema mapping and matching is a crucial step in the data integration process, as it addresses the disparities between data sources by aligning their structures into a unified schema. During the workshop, the process involved defining and aligning schemas to integrate data from Scopus, Web of Science, and Google Scholar. This section outlines the key steps involved in the schema mapping and matching process.

3.2.1 Identification of Source and Target Schemas

- Each data source (e.g., Scopus, Web of Science, Google Scholar) presents its unique schema, with distinct field names and structures. The process began by examining the extracted metadata and documenting the attributes provided by each source.

- A target schema was then defined to serve as the standard for integration. This schema encompassed attributes such as Author, Title, Year, Keywords, DOI, and Publication Venue, ensuring compatibility across all sources.

3.2.2 Alignment of Attributes

- To achieve uniformity, attributes from the source schemas were mapped to the target schema. For instance, variations in field names like Author (Scopus) and Authors (Google Scholar) were identified and aligned to a single field in the target schema.

- Attributes with differing formats (e.g., date formats or citation counts) were standardized to maintain consistency.

3.2.3 Resolving Conflicts

- Conflicts such as duplicate entries, missing values, or inconsistent data formats were addressed. Techniques such as deduplication, data imputation, and validation checks were used to resolve these issues.

- For duplicate detection and resolution, tools like Dedupe were employed to identify and merge similar records, ensuring data integrity.

3.2.4 Fusion of Data

- After resolving conflicts, the aligned data from all sources was merged into a single unified dataset. This dataset adhered to the target schema, allowing for seamless analysis and further processing.

3.2.5 Tools and Techniques Used

- **Pandas:** Used extensively for data manipulation, cleaning, and schema mapping.
- **OpenRefine:** Utilized for standardizing data formats and resolving inconsistencies.
- **Dedupe:** Applied to identify and resolve duplicate entries in the datasets.

By following this structured approach, the integration of metadata from multiple sources successfully overcame the challenges posed by heterogeneous schemas. The unified dataset served as the foundation for subsequent tasks, such as journal recommendation, demonstrating the importance of schema mapping and matching in the broader context of data integration.

3.3 Integration Process

The integration process was designed to consolidate data from multiple sources, including Scopus, Web of Science, and Google Scholar, into a unified dataset. This involved several key steps, each meticulously executed to ensure data consistency and accuracy.

3.3.1 Data Ingestion

The first step involved ingesting data from various formats. Scopus data was retrieved from an XML file, while Web of Science and Google Scholar data were sourced from JSON files. Handling these different formats was crucial, as it allowed us to gather comprehensive information from each platform.

3.3.2 Data Parsing

Once the data was ingested, the next step was to parse it. For Scopus, we extracted author details such as full name, h-index, total citations, co-authors, and documents. Each document included URL, authors, title, publication year, citation count, DOI, abstract, keywords, document type, and journal information when available. A similar approach was taken for Web of Science and Google Scholar, with adjustments made to accommodate the JSON structure.

3.3.3 Data Collection

After parsing, all authors from the three sources were aggregated into a single list. This step was pivotal as it prepared the data for the subsequent matching and merging processes.

3.3.4 Author Matching

To ensure that authors from different sources were correctly identified as the same individual, a fuzzy matching algorithm with a threshold of 80 was employed. This method compared author names and validated matches by cross-referencing h-index and total citation counts, ensuring data consistency.

3.3.5 Record Merging

Once authors were matched, their records were merged. Co-authors and documents were combined, with measures in place to avoid duplicates. Additionally, metrics such as h-index and citation counts were updated to reflect the highest values found across all sources, ensuring that the integrated data was as accurate and up-to-date as possible.

3.3.6 Output Generation

The final step involved generating an output file. The integrated data was saved into a JSON file named `integrated_data.json`, providing a structured format for future analysis.

3.3.7 Error Handling and Logging

Throughout the process, robust error handling mechanisms were implemented to manage exceptions during data loading and processing. Logging was used to record errors and track the processing steps, facilitating easier debugging and transparency.

In summary, this integration process successfully consolidated data from multiple sources, providing a unified view of author profiles and publications. This unified dataset is essential for comprehensive analysis and serves as a foundational resource for the project.

3.4 Process of Suggesting Journals

The process of recommending journals in this system involves several stages, from data preprocessing to model training and inference. Here's a step-by-step breakdown of the key components:

1. **Data Collection and Preprocessing:** We begin by reading data from a JSON file containing research papers, each with relevant fields like title, abstract, and associated journal information. The primary goal here is to clean, validate, and format the data for further use in the recommendation model.

2. **Data Cleaning:** The preprocessing functions ensure that missing or malformed data entries are handled properly. Specifically:

   - **Missing Fields:** Any research paper that lacks key information, such as a title, abstract, or journal details, is discarded.
   - **Invalid Journal Entries:** If journal data is incomplete or not structured properly, the entry is flagged and skipped.
   - **Data Transformation:** For journal information, the h-index, SJR (Scimago Journal Rank), and impact factor are extracted and used to calculate a weighted journal score.

3. **Journal Score Calculation:** Each journal is assigned a score based on its h-index, SJR, and impact factor. These scores are used to rank journals during the recommendation process. This is achieved through the `calculate_journal_score()` function, which ensures that journals are evaluated based on these three crucial metrics, each contributing equally to the final score.

$$\text{Journal Score} = 0.33\text{H-index} + 0.33\text{SJR} + 0.33\text{Impact Factor}$$

4. **Tokenization and Text Preprocessing:** The title and abstract of each paper are concatenated and cleaned. The `clean_text()` function removes unwanted characters, lowers the case, and lemmatizes the text while eliminating stop words. This ensures that the text input to the model is clean and consistent, which is essential for building accurate embeddings.

5. **Data Structuring:** After preprocessing, the data is structured into a format suitable for training the model. This includes combining the paper's title and abstract, calculating the journal's score, and encoding journal names into numerical labels. The result is stored in a dataframe, which is then split into training and validation sets.

6. **Model Training and Embedding Generation:** A custom BERT-based model is used to generate embeddings for the paper titles and abstracts. These embeddings are fine-tuned using contrastive loss to learn the relationships between papers and their associated journals. This embedding space allows the system to measure the similarity between a query paper and potential journal candidates efficiently.

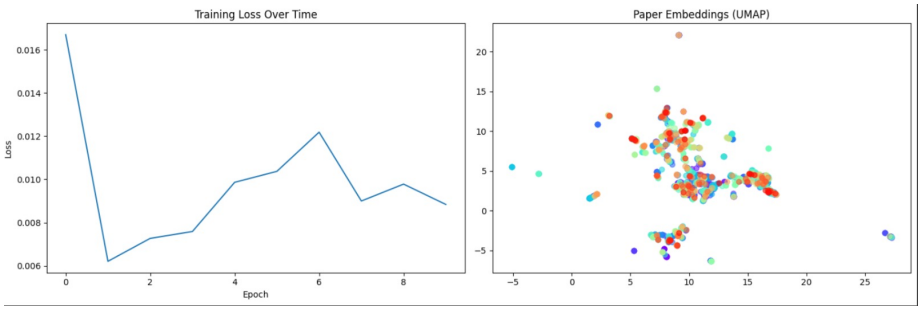

Figure 3.1: BERT Model Training and Embedding Generation

7. **Recommendation Generation:** To generate journal recommendations for a given paper, the system performs a two-step process:

    - **Query Embedding:** The title and abstract of the query paper are cleaned and passed through the trained BERT model to obtain an embedding.
    - **Similarity Measurement:** The system then computes the cosine similarity between the query papers embedding and all stored paper embeddings. The top-k most similar papers are selected, and their corresponding journals are recommended, ranked by their journal score.

8. **Final Score Adjustment:** Recommendations are further filtered and sorted by the journal score to ensure that papers are matched with journals that not only have a high similarity but also demonstrate strong academic impact.

3.5 Examples of Recommendations

To demonstrate the recommendation capabilities of the system, consider the following example:

- **Application interface:**

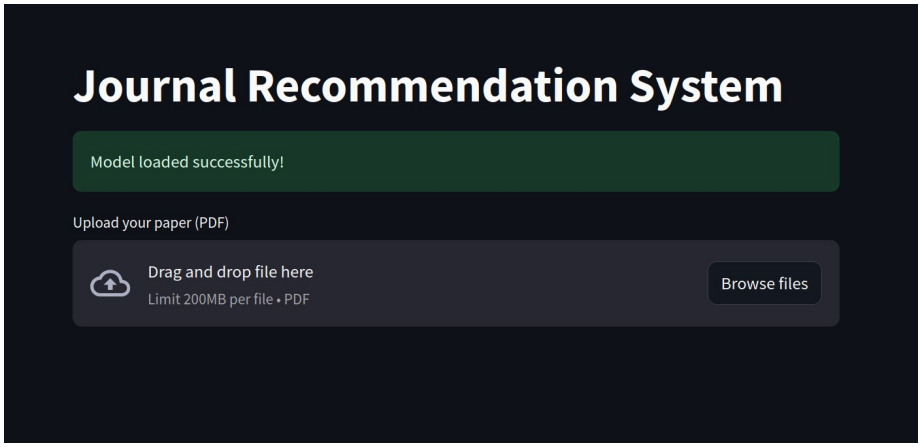

Figure 3.2: Application Interface

- **Input Paper:**
  - **Title:** "Deep Learning in Neural Networks: An Overview"
  - **Abstract:** "In recent years, deep artificial neural networks (including recurrent ones) have won numerous contests in pattern recognition and machine learning. This historical survey compactly summarizes relevant work, much of it from the previous millennium. Shallow and deep learners are distinguished by the depth of their credit assignment paths, which are chains of possibly learnable, causal links between actions and effects. I review deep supervised learning (also recapitulating the history of back-propagation), unsupervised learning, reinforcement learning & evolutionary computation, and indirect search for short programs encoding deep and large networks."

- **Recommended Journals:**

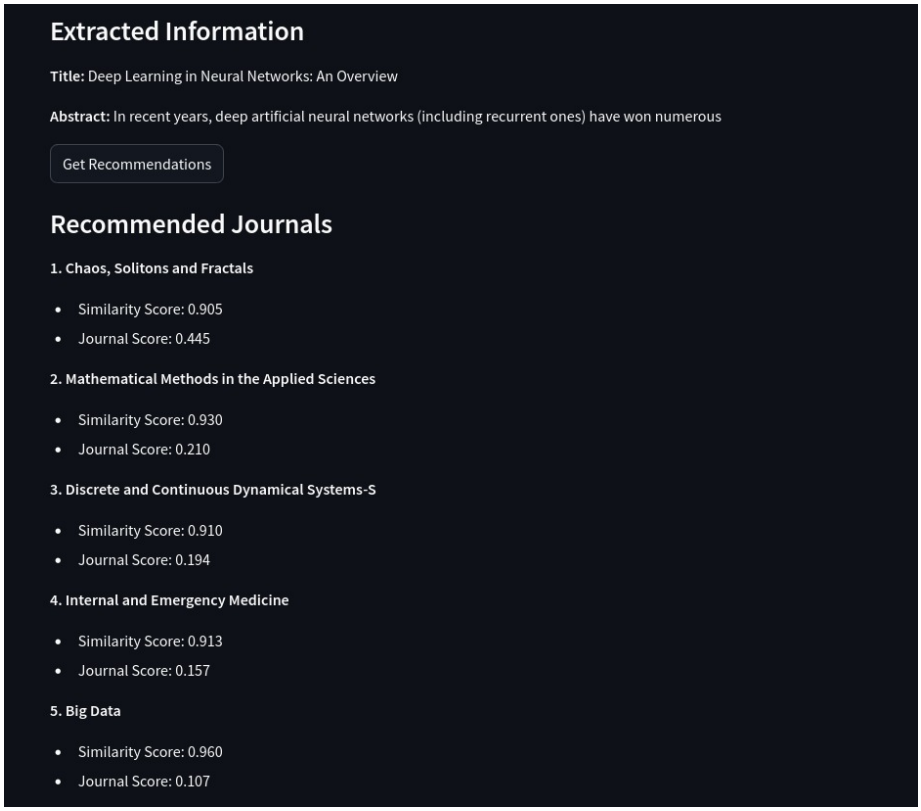

Figure 3.3: Recommended Journals

These recommendations are based on the similarity between the input papers embedding and the embeddings of all other papers in the dataset. The top recommendations are selected by ensuring both high similarity and high journal scores, making the system both efficient and accurate in its suggestions.

3.5.1 Conclusion

The preprocessing steps outlined above are critical for ensuring that the data fed into the model is of high quality and that the recommendation process is as efficient as possible. By cleaning the text, validating journal data, and using BERT embeddings to measure similarity, the system is able to provide meaningful and relevant journal recommendations.