An Extensible General-Purpose Data Gathering and Classification Platform: Maestro v2023

António Miguel de Sá Viana Valente Martins

Thesis to obtain the Master of Science Degree in

Computer Science and Engineering

Supervisors: Prof. Alberto Manuel Rodrigues da Silva
Prof. Jacinto Paulo Simões Estima

Examinating Committee

Chairperson: Prof. Pedro Tiago Gonçalves Monteiro
Supervisor: Prof. Alberto Manuel Rodrigues da Silva
Member of the Committee: Prof. João Carlos Amaro Ferreira

November 2023

Declaration

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Acknowledgments

Writing this thesis has been equal parts rewarding and frustrating. For many months I felt nothing short of adrift navigating this topic. Despite my innate interest in Maestro's concept, I struggled to decide on the path this new iteration should follow. If not for Prof. Alberto and Prof. Jacinto, I would still be questioning each choice that was made. I owe them immensely and would like to thank them for their constant input and support.

In many senses, this year proved difficult for me. Despite this, my mothers showed me unconditional support and patience, a testament to their own strength. My gratitude for them cannot be understated, nor can it be summarized here. Thank you, from the bottom of my heart.

My father portrayed a different role. He never once doubted my abilities, serving as a pillar of unwavering faith in me. Thank you for always believing in me.

To my aunt and cousin, I hope they understand how grateful I am for their patience and the ways in which they tried to make my life easier. Living with a student writing a thesis is not easy, but they never once complained about my antics during this time.

Finally, to Matej, my dearest friend. He was here for me every step of the way and listened intently with genuine interest and care. Talking to you is the highlight of my day. I hope I can always be there for you in the same way.

Abstract

The Maestro platform has been developed with the purpose of helping researchers automatically gather and classify data on specific topics, which has commonly been done using several purpose-specific tools in a tedious process. This thesis introduces and discusses Maestro v2023, an improvement over the original Maestro platform. This new version of the platform enhances its capabilities and extends its applicability in various scenarios. Notable improvements include the introduction of text data types, the addition of a new analysis stage to Maestro's pipeline, numerous usability-oriented enhancements, and the implementation of various other features. Furthermore, several conceptual and architectural aspects of Maestro were re-designed and extended. This research follows the design science research methodology, an iterative methodology that combines principles, practices, and procedures to provide guidance for research in information systems.

Maestro v2023’s capabilities are applied and demonstrated in two different scenarios: (i) gauging the "social climate" of corporations and businesses; (ii) streamlining literature reviews and semi-automatic abstract generation. Maestro v2023’s usability and relevance are also evaluated through a user assessment session and the results are compared to those obtained in a similar user assessment of Maestro’s original iteration. The results of this evaluation demonstrate a significant improvement in the perceived usability of the platform, as well as a continued interest in using Maestro v2023 in the future.

Keywords: Data gathering, Data classification, Data analysis, Text analytics, Machine learning.

Resumo

A plataforma Maestro foi desenvolvida com o propósito de auxiliar investigadores na recolha e classificação automática de dados, uma tarefa tediosa geralmente realizada com recurso a várias ferramentas específicas. Este trabalho apresenta e discute o Maestro v2023, uma melhoria sobre a versão original do Maestro. Essa nova versão da plataforma expande as suas capacidades e extende a sua aplicabilidade em diversos cenários. Estas melhorias incluem a introdução de tipos de dados textuais, a adição de uma nova etapa para análise de dados, diversas alterações com foco no aumento da usabilidade da plataforma, e a implementação de várias outras funcionalidades e conceitos. Diversos aspetos conceptuais na arquitetura do Maestro foram também re-estruturados e extendidos. Este trabalho segue a metodologia de design science research, uma metodologia iterativa que combina princípios, práticas e procedimentos para orientar a pesquisa em sistemas de informação.

As capacidades do Maestro v2023 são aplicadas e demonstradas em dois cenários distintos: (i) avaliação do "clima social" de corporações e empresas; (ii) agilização de revisões de literatura e geração semiautomática de resumos. Foi conduzida uma sessão de avaliação com utilizadores de forma a avaliar a usabilidade e relevância do Maestro v2023, e os resultados comparados com os obtidos na sessão de avaliação da iteração original da plataforma. Os resultados desta avaliação demonstram um aumento significativo na usabilidade da plataforma, bem como um interesse em utilizar o Maestro v2023 no futuro.

Palavras-chave: Recolha de dados, Classificação de dados, Análise de dados, Análise de Texto, Aprendizagem automática.

Table of Contents

List of Tables

List of Figures

List of Algorithms

List of Acronyms

API	Application Programming Interface
BPMN	Business Process Model and Notation
CTI	Cyber-Threat Intelligence
DOI	Digital Object Identifier
DOCX	Microsoft Word Text Document
DSR	Design Science Research
EXIF	Exchangeable image file format
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
IS	Information Systems
IT	Information technology
JSON	JavaScript Object Notation
MD5	Message-Digest Algorithm 5
MISP	Malware Information Sharing Platform
ML	Machine Learning
NLP	Natural Language Processing
ORM	Object-Relationship Mapping
PDF	Portable Document Format
RCP	RiverCure Project
RDBMS	Relational Database Management System
REST	Representational State Transfer
RG	Research goal
UI	User Interface
UML	Unified Modeling Language
URL	Uniform Resource Locator

1. Introduction

This chapter introduces the context, motivation, and the goals of this project. We also present the research methodology and the dissertation structure.

1.1. Context

The World Wide Web, commonly referred to as the Web, has found its way into the daily lives of most people in the world. Despite its relatively short lifespan, it has propelled humanity into a new age of distributed information. We now have access to vast amounts of specialized knowledge that was once arduous to share and discover.

The Web is home to an immense amount of information that is stored in various types of data, including text, sound, video, and other representative structures [1], [2]. While there are many ways to search through this wealth of information, some methods may not be immediately apparent to the average user. Fortunately, a great deal of this information has been indexed, allowing Web users to search through it by utilizing popular search engines such as Google and Bing [3], [4].

However, other methods of searching the contents of the Web exist and provide several benefits. Web crawlers, for example, allow for the recursive discovery of new resources [5], while Web scrapers can extract content from these resources [6]. These methods enable the bulk gathering of information, which is particularly relevant in many research areas like Machine Learning (ML) [7].

The field of ML has harnessed this ability to access vast amounts of digitized data, leading to the exponential growth and rapid improvement in major tasks. One notable recent achievement in ML was the development of GPT-3, an autoregressive language model [8] that uses neural networks to generate human-like text in response to a given textual prompt. Besides traditional natural language processing tasks, such as text completion, question answering, and language translation, GPT-3 can also be used to perform complex tasks like summarizing reports, generating programming code, and even developing games [9]. GPT-3 is a direct result of leveraging vast amounts of data available on the internet, trained on multiple sources, such as Wikipedia, books, and numerous websites.

The field of data classification stands out as a prime beneficiary of this approach, as complex classification algorithms require a large amount of quality-data to be trained effectively. However, it is crucial to approach this task with rigor to avoid any skewed or discriminatory results [10], and this trend is only expected to increase in the future. As the amount of digitized data continues to grow, their use in research will only become more prevalent, driving the development of new mechanisms to facilitate the process.

Maestro was developed to streamline some of these processes, serving as a modular, extensible, and configurable platform for data gathering and data classification [11]. Maestro allows users within an organization to create configurable search contexts that automatically gather and classify data for them. Although the platform's primary goal is data gathering and classification, it also allows for additional modular steps, such as data filtering and post-processing, further expanding its range of applications.

This dissertation presents Maestro v2023, an improved version of the original Maestro platform, achieved through expanding capabilities and refined functionality, namely the extension of the available data types to incorporate different forms of text data, the introduction of a new phase to the pipeline that allows users to analyse their data through modular graphs, as well as various updates to the internal behaviour, structure, features, and UI of the platform.

As with the original iteration of this research, the focus will not be on developing new methods of Web data gathering or data classification, but rather on developing an orchestration tool, much like a maestro does in an orchestra. Going forward, we will refer to the original Maestro platform as "Maestro v2022" and to the proposed revamped version resulting from the expansion as "Maestro v2023". The term "Maestro" will refer to the overall concept of the platform.

1.2. Motivation

With the increase in the amount of digitized data available on the web, the area of data gathering too has grown, as finding efficient ways of gathering these large amounts of data became a worthwhile endeavour. As previously mentioned, access to this data enables the development and study of various other areas of research, such as data classification technologies.

The fields of data gathering and data classification are commonly intertwined, with the gathered data serving as a catalyst for the training of classification algorithms. Classification algorithms serve little purpose if they have no data to classify, further exacerbating this symbiotic relationship. Despite this reality, the gap between these two areas is still scarcely bridged by existing studies. They are often done separately, with the data gathering process serving as a precursor to data classification, and not as sequential steps in a merged approach.

Maestro aims to bridge this gap. The Maestro v2022 platform was originally developed with the purpose of creating a service highly coupled with the RiverCure project (RCP), a research project aimed at reducing uncertainty and improving forecasting capabilities of hydrodynamic and morphodynamical mathematical models for flood simulation, water resources management, and habitat protection [12]. Maestro's purpose was to gather images of floods from social media, calculate the depth of the water at the moment the image was taken, and subsequently provide these images to RCP to help it forecast the evolution of the flood with a higher level of precision.

Understanding that this process could be generalized to other applications requiring Web data gathering, followed by a classification step, the researchers shifted the focus of the research. Thus, Maestro became a platform characterized by its extensibility, modularity, and configurability. It allows

organizations and its users to create search contexts in a pipeline that aggregates all the steps in creating a dataset, providing fresh labelled data at the end. The steps presented by the pipeline are highly customizable through the use of plugins and configurations, making the platform adaptable to various scenarios. It also allows for the distribution of the data to external services, iterative runs of the search context, as well as post-processing, filtering, and keyword enhancing mechanisms.

Maestro v2023 is the continuation of this research and the improvement of the original Maestro platform, both by expanding upon its capabilities and by refining the ones already implemented. As mentioned previously, the focus was not on developing new methods of Web data gathering or data classification, but to develop an orchestration tool to aggregate and coordinate different processes from these fields.

1.3. Problems Addressed

Maestro has been developed with the goal of tackling many, often unrelated, situations. Thanks to its configurable nature, researchers can adapt the platform to their specific needs. One particularly relevant aspect to be considered is the time sensitivity of the tasks Maestro can be used in. The platform can aid in tasks that bear real-time constraints, in which timely data is necessary, tasks that bear no time constraints, or even tasks to predict future events based on current data. However, Maestro v2022 still bears some limitations which we address in Maestro v2023, namely: limited types of data, inability to do in-depth analysis of the pipeline, and other limitations.

Limited types of data. Maestro v2022 was limited to two data types: images and sounds. Maestro v2023 extends the available data types to allow for the gathering and classification of general text data, as well as specialized types of text data (i.e., scientific papers and news articles).

Inability to do in-depth analysis of the pipeline. Maestro v2023 not only improves the amount and quality of the information shared by the system during the execution of the pipeline, but also adds a step to the pipeline itself, where, by using specific plugins, the users are able to make use of the information to perform a statistical analysis of their data. The type of information that is available to users, while useful, isn’t extensive enough to address certain situations. For instance, researchers might be interested in correlating the gathered data with the classified data, by better understanding which fetching API provided the system with data more likely to be classified with a particular label.

Other limitations. Other aspects were also considered in improving Maestro. A few were minor improvements or fixes, such as UI changes based on previous usability tests, while others can be seen as usability changes and features, such as the ability to utilize multiple classification plugins simultaneously, or the addition of a more robust search feature during the configuring stage of a new search context. Each of these individual aspects may not single-handedly make a big impact on the platform's quality, nor did they take up a significant amount of time during development, compared to

the previously mentioned improvements. However, once all their contributions are tallied up, we believe they increase the quality of the platform.

1.4. Research Goals

Addressing the issues in the Maestro v2022 platform should contribute to its original purpose of handling custom and specific Web data gathering and classification tasks. However, with these improvements, the number of applications and situations in which the platform can be used, as well as the ease in making use of it, should be greater. As such, this research was proposed with the following goals:

RG-1. Analyze and discuss the state of the art in the area(s) of: Data gathering utilizing Web Crawlers, Web Scraping APIs and other fetching tools, and how they can be integrated with classification and other machine learning algorithms.

RG-2. Design and implement Maestro v2023, an extensible and flexible platform that allows organizations to create search contexts in a pipeline intended to gather, classify and, optionally, analyze and distribute data.

RG-3. Evaluate Maestro v2023 according to its usability. This will be attained by designing several study cases demonstrating Maestro v2023's capabilities, and performing a test session where real users will follow the steps required to execute the proposed scenario(s).

1.5. Research Methodology

This research follows the Design Science Research (DSR) methodology. The DSR is an iterative methodology that combines principles, practices, and procedures. It provides guidance for research in Information Systems (IS) as well as other disciplines [13], [14]. Design Science emphasizes systematic, testable, and communicable methods [15].

For instance, Hevner et al. [13] propose a set of guidelines for the application of DSR in the information systems area, namely including the following aspects:

1. **Design as an Artifact**: The research shall produce a viable artifact in the form of a construct (e.g., software application or tool), a representation (e.g., new language or extension of a previous notation), a technique (e.g., a process or method), or an instantiation (e.g., a case study that applies such artifacts).
2. **The relevance of the problem**: The basic objective of DSR is to develop technology-based solutions to relevant and significant technology business problems.
3. **The design evaluation**: The quality (e.g., measures in terms of utility or efficacy) of the design artifact shall be demonstrated rigorously through a well-executed evaluation method.

4. **Research contribution**: Effective DSR shall offer a clear and demonstrable contribution in the area that the design artifact is applied such as design foundations and or design methodologies.
5. **Research Rigor**: The DSR depends upon rigorous methods application in both evaluation and the construction of the design artifact.
6. **Design as a search process**: The search for an effective artifact depends on the use of the available ways to reach desired outputs while the rules in the problem environment are still satisfied.
7. **Communication of the results**: The research presentation shall be effective from both the technology and the business perspective.

These guidelines drive the research but can be translated into the following phases [14]:

- **Problem identification and motivation**: Definition of the specific research problem in the areas of web data gathering and classification addressed by Maestro, justification of the value of the proposed improvements to the platform, as well as the motivation for the continued development of the platform and proposed solutions. (Oct. 2022 – Nov. 2022)
- **Define the objectives of a solution**: Define the objectives of an achievable solution to the identified problem. Namely, expanding and improving the Maestro platform, as well as the evaluation of the results. (Oct. 2022 – Dec. 2022)
- **Design and development**: Designing and creating Maestro v2023, an improvement and expansion of the original work on the Maestro platform. (Oct. 2022 – August 2023).
- **Demonstration**: Demonstrate the value of the Maestro platform, as well as the increase in quality introduced by the proposed solutions in the design of Maestro v2023, in the context of the presented study cases. (Feb. 2023 - August 2023)
- **Evaluation**: Perform a test session focused on the execution of the scenarios proposed during the demonstration stage to infer Maestro v2023's ability to address the identified research problem, and the participants' overall impressions of Maestro v2023's capabilities. (May 2023 – Sept. 2023).
- **Communication**: Write a research paper to submit to a conference, as well as a dissertation, demonstrating the relevance, utility, novelty, and effectiveness of Maestro v2023. (Dec. 2022 – Nov. 2023).

1.6. Dissertation Outline

The remainder of this document is organised as follows. Chapter 2 provides background on approaches and areas of research related to Maestro. Chapter 3 identifies and discusses the related work. Chapter 4 introduces Maestro v2023's key concepts and overviews its workflow from a conceptual view. Chapter 5 presents an in-depth analysis of the features and modifications introduced in Maestro v2023. Chapter 6 presents use cases designed to demonstrate Maestro v2023's

capabilities. Chapter 7 describes a test session performed to evaluate the system. Finally, Chapter 8 presents the conclusion and identifies future research goals.

2. Background

This chapter describes various aspects and concepts related to Maestro v2023, including some of the approaches and tools employed, as well as related areas of research.

2.1. Web Crawling

Web crawlers are a type of bot used for the discovery of web pages and links, often with the subsequent goal of indexing information. They traverse through a set of seed Uniform Resource Locators (URLs) and recursively continue their search through hyper-links found within these pages. Despite their simplicity, web crawlers present many inherent challenges, such as scalability, content selection trade-offs, and social obligations due to their potentially high burden on websites [16].

To mitigate web crawlers' challenges, different types have emerged, each applying certain techniques. A recent survey has attempted to discriminate between these different web crawler types, arriving at 5 major categories: universal crawlers, preferential crawlers, hidden web crawlers, mobile crawlers, and incremental crawlers [17].

Universal crawlers and preferential crawlers are similar in how they operate. However, unlike preferential crawlers, universal crawlers do not bound their search to specific categories and topics. They will recursively search through every hyperlink they find, thus maximizing the number of indexed pages.

Preferential crawlers also attempt to maximize the number of indexed pages. However, they limit themselves either completely to pages pertaining to a specific topic (topical crawlers) or partly, crawling closely related categories (focused crawlers) too. Preferential crawlers use a wide range of architectures and algorithms to increase their performance. A recent study by Gunawan et al. [18] researched the usage of a distributed architecture with focused crawlers, having one crawler serving as master, and other crawlers serving as slaves. By experimenting with different amounts of threads and search algorithms, they determined the optimal configuration to increase the crawler's performance.

Hidden web crawlers are specialized crawlers tasked with indexing pages that can't be accessed by traversing hyperlinks, instead focusing on retrieving content hidden behind search forms which might require prior authorization, registration, or configuration to access [19]. A group of researchers recently employed the use of a darknet crawler and were able to determine that, in reality, a big part of the visible darknet is well connected through the use of hub sites, such as wikis and forums [20].

Mobile crawlers attempt to minimize the network load they create by offloading the selection and filtration of websites to a server. On the other hand, incremental crawlers are used to maintain up-to-date versions of indexed webpages. Nonetheless, these types of crawlers are highly specialized and do not align with Maestro's objectives.

As for implementing web crawlers, there are various libraries and frameworks available. Scrapy [21] is one such framework for the python programming language, and the one that is most commonly used within Maestro itself. Crawler [22], a package to create Web crawlers for Node, is another example. One may also develop their own crawlers to satisfy specific requirements. A recent study, carried out in an attempt to categorize content available in the Deep Web, made use a of a custom-made crawler in python [23].

2.2. Data Classification

Data classification is a crucial process in ML projects. Essentially, it involves using a classification algorithm (i.e., a classifier) to assign a label to each object in a given dataset. To do this, the algorithm must undergo a training phase using a labeled dataset, in which each data item is paired with its respective label. Once trained, the classifier can then classify new, unlabeled data items [24] with a high degree of accuracy. In addition to the initial training phase, classification algorithms can be further adapted through a process called "fine-tuning", where the outputs are modified or extended to meet specific needs.

Data classification is a crucial process in various fields as it helps establish relationships between data and their corresponding classes. As Maestro serves as a bridge between data collection and classification, we place significant emphasis on this process.

There are many pre-trained classification algorithms that are readily available to researchers, which eliminates the need to create new ones from scratch. One such platform that offers pre-trained models and other ML tools is Hugging Face [25]. Researchers can either use these pre-trained classifiers directly or fine-tune them to meet their specific needs. For instance, Caraffini et al. [26] developed a classification algorithm that accurately categorizes various bird species based on their images.

2.3. Data Gathering

Gathering data is another process that Maestro handles directly. With the amount of available data on the web, it only makes sense that modern data collection techniques have evolved to collect it. This can be done in various ways. One such method would be through the usage of Web crawlers (described in section 2.1). They allow for the discovery of large amounts of data, which can be specialized through the usage of preferential crawlers. However, Web crawling is mostly focused on discovering URLs. To extract the data itself, another method that can be used is data scraping.

Web data scraping focuses on methods of extracting data from specific sources, such as websites or databases. One possibility revolves around using specialized APIs. The Twitter API [27] allows

users to retrieve data from the Twitter platform, such as media, tweets, or users. The Bing Images API [28] is another example of a scraping API - in this case specialized for the retrieval of image type data.

Some Web crawler frameworks recognize the data structure of a page automatically, removing the need to differentiate between the steps of Web crawling and scraping. This is the case for Scrapy [21], the python Web crawling framework previously mentioned. By utilizing the correct set of tools for data gathering, one can eliminate many of the intermediate steps in this process, enabling researchers and users to decrease the time and effort needed to produce datasets.

2.4. Text Analytics

Text analytics, or text mining, is the process of examining and extracting meaningful insights, patterns, and information from unstructured text data [51]. The fields of ML and Natural Language Processing (NLP) have become intimately tied to the field of text analytics, driving research in the advancement of several tasks associated with text analytics, such as text summarization and simplification. These tasks are relevant to the development of Maestro v2023, as it introduces new text data types and, consequently, may aid in scenarios that require these tasks to be pursued.

2.4.1. Text Summarization

The area of text summarization aims to allow the condensation of documents and publications. When done correctly, the produced summaries are expected to highlight the critical aspects of these artifacts, effectively undermining the need to sift through a large amount of redundant information.

Different trends and techniques form the basis for research within this field. A recent study by Widyassari et al. [52] systematically reviews automatic text summarization by analyzing different publications published from 2008 to 2019. They identified ML approaches as the most predominant technique, being used in more than half of the studies analyzed. Regarding trends, multi-document summarization was the most prevalent, in which the summary is generated based on a set of input documents and the target is to remove repetitive content in the input documents [53]. Extractive summarization followed closely behind, an approach focused on choosing the most important words, sentences, and paragraphs to produce a summary. The third most common trend was abstractive summarization, which aims to produce summaries consisting of sentences different from the original document(s). Abstractive approaches tend not to be as favored in research as extractive approaches, as they are highly complex and require extensive NLP [53].

2.4.2. Text Simplification

Text simplification aims to make complex language easier to understand by rephrasing it into simpler terms and typically involves making use of three core elements: splitting, deletion, and paraphrasing. Splitting involves breaking lengthy sentences into several smaller sentences that enhance the readability of the overall text. Deletion discards a sentence's extraneous and less consequential parts,

thereby reducing its complexity. Finally, paraphrasing is used to reorder, substitute, and, in some cases, expand sentence constructs to achieve a simplified version of the original text [54].

Research in this area has various practical applications, including assisting people with disabilities, low literacy, non-native language backgrounds, or limited expertise to comprehend written materials more easily [54]. Despite its complexity, the automation of this process has rapidly grown, spurred by the rise of both ML and NLP [55].

2.5. Technologies

Maestro is a web-based platform, developed using Python, Django, Celery, and PostgresSQL. This section succinctly introduces each of these technologies.

2.5.1. Python

Python is an interpreted, object-oriented, high-level programming language [58]. It is widely used in various fields, including web development, data analysis, scientific computing, artificial intelligence, automation, and more.

Python boasts one of the most extensive and highly engaged programming communities, consistently earning its place as one of the most cherished programming languages among developers for several consecutive years [59], [60]. Furthermore, Python is the number one choice when it comes to data science and ML, as several ML and data processing libraries (e.g., Numpy, Pandas, Pytorch, and Tensorflow) are available, often exclusively, in Python.

As Maestro is heavily associated with ML and data processing, the choice was made to use Python in the platform’s development. This choice is further supported by the fact that users are more likely to have some experience programming with Python, as plugin development is one of Maestro’s key features.

2.5.2. Django

Django is a high-level, open-source web framework for building web applications using the Python programming language, and facilitates “rapid development and clean, pragmatic design” [56]. It provides developers with useful out-of-the-box functionalities, such an ORM, admin panel, caching, logging, templating system, and security features, which is not true for other python-based web frameworks, like Flask. Furthermore, Django allows for packages developed by the community to be introduced into each project, significantly reducing development time. These packages include reusable apps, tools, models, or standalone functionalities.

Maestro was designed to be an extensible platform, something easily achieved using Django's framework and package system. Furthermore, the time spent on mundane tasks, such as form validation and webpage design, is highly reduced using Django's base features.

2.5.3. Celery & RabbitMQ

Celery is an open-source, distributed task queue system for asynchronous processing in Python [61]. This queue receives tasks delivered by a message broker and distributes them through its workers. More accurately, a client will send a message to the message broker with the details of the task to be performed. A worker process (managed by Celery) will then extract the message from the message broker and handle it. RabbitMQ [62] is the most popular choice of message broker when using Celery, as it is fault-tolerant, guarantees message delivery, and allows encrypted connections to be established.

Celery, with RabbitMQ serving as the message broker, is essential to Maestro's design, as it allows for asynchronous executions of tasks. Namely, the tasks that occur during the execution of a search context can be done asynchronously, with multiple workers working on different search context executions at a time. As the workers and message brokers do not need to run on the same machine as Maestro's client, any number of workers and message broking processes can be introduced, making the system highly scalable.

2.5.4. PostgreSQL

PostgreSQL is an open-source relational database management system (RDBMS) with an emphasis on performance, reliability, and robustness [63]. As Maestro's data is stored in relational structures, the choice of using an RDBMS like PostgreSQL was a sensible one.

Django also provides official support for PostgreSQL, allowing for the manipulation of the database's content using python objects directly. Systems like these, denominated object-relationship mappers (ORMs), allow developers to manipulate the database without recourse to manually written SQL code, enabling the development of systems with a layer of abstraction. However, ORMs require the generalization of queries to the database, which may prove inefficient in certain scenarios.

3. Related Work

This chapter introduces, analyses, and discusses research and work that relates to Maestro, as well as the work previously done on the development of the platform.

3.1. Maestro v2022

As this research work was focused on expanding upon the original research that led to the development of the Maestro platform, it is relevant to present some of the main aspects of the original work. Implemented using the Django framework for python [56], Maestro v2022 is the first iteration of the Maestro Platform, and was created with the purpose of gathering and classifying data as a service. It functions in a modular, extensible, and configurable fashion, enabling users within an organization to, through the usage of plugins, automatically collect and classify data.

3.1.1. Plugins

The system itself can be seen as a pipeline that, once run, results in a classified dataset that can be provided to external services. This pipeline comprises 6 different consecutive steps:

- **Fetching**. The system fetches URLs pointing to objects of the desired data type.
- **Gathering**. The system gathers the resources from the fetched URLs.
- **Post-processing**. Specific plugins can be applied to the gathered data, acquiring additional parameters for the subsequent steps (e.g., image metadata).
- **Filtering**. The data is filtered according to the specified plugins and parameters selected by the user.
- **Classification**. The finalized set of data is classified using the desired plugins.
- **Providing**. The resulting classified dataset is provided to external services.

Of these steps, the only two which cannot be configured to use plugins are the gathering and providing steps. Plugins are user-made scripts that follow a common interface that Maestro understands. For example, one may use the Pixabay API [29] to create a fetching plugin, thus enabling the system to retrieve images from the website during the fetching phase.

3.1.2. Organizations, Users, and Search contexts

Maestro's framework has three key concepts: organizations, users, and search contexts. Maestro users can be associated with one or more organizations, which exist to facilitate collaboration and simultaneous workflows. A user can define multiple workflows configured to gather, classify, and deliver the data they target. These workflows are named search contexts, and they function as declarative expressions of the tasks to be run through Maestro's pipeline.

Search contexts can initially be configured to determine the function of the internal components of the system, when running a specific task. As an example, a user might configure the search context to fetch images from the Pixabay API [29] that contain the keyword "bird" and select a classification plugin that classifies the retrieved images according to the species of the bird [26].

3.2. Data Gathering and Classification Research

Data gathering and data classification are highly researched areas of study. Several research projects and works on these areas exist, either focusing on a specific one or combining them.

Much research has been done on the process of training ML classification algorithms using both supervised and unsupervised [30] techniques. In these works, there is often a data gathering step, followed by a training phase, with the ultimate aim of training a classification algorithm. Dilrukshi et al. [31] follows this structure, where the Twitter API [27] was used to gather tweets containing news headlines from certain micro-blogs. They then manually labelled these short texts into different categories and, using a Support Vector Machine [32] method of supervised learning, trained a classifier with the purpose of labelling news articles into categories.

Various data gathering systems have also been developed and applied with the purpose of categorizing retrieved data for various purposes. In [33], [34], K. Yanai proposes an image collecting system, which allows for the gathering of web images employing keyword-based search engines. Later, the same author expanded upon his work by proposing a system [35] that used the gathered images to train a generic image classification algorithm, thus allowing images to be provided to the system for classification.

The InTime [36] platform is another project with relevant similarities. It allows users to configure web sources from where to identify and extract data related to the Cyber-Threat Intelligence (CTI) domain. This data can then be classified using ML algorithms to extract relevant CTI artifacts and export them to a MISP database [37], an open-source collaborative threat intelligence sharing platform.

Another popular tool is the "Publish or Perish" (PoP) software, developed by Anne-Wil Harzing [38]. This software program retrieves and analyzes academic citations from multiple sources, presenting academics with several citation metrics. Though it does not make use of classification algorithms, the analysis step derives additional information based on the retrieved citations, similarly to what is done in Maestro's pipeline during its post-processing and proposed analysis phase.

3.3. Discussion

Both data gathering and data classification are fields that continue to grow. New data gathering techniques often arise with the purpose of addressing different challenges, especially with the increasing need for high amounts of data in machine learning paradigms, data classification being no exception.

Research projects related to data gathering, data classification, and the development of systems that integrate both fields of study, show significant diversity and provide valuable contributions to these fields. Nonetheless, work aimed at achieving similar objectives as Maestro (i.e., bridging the fields of data gathering and classification in a scalable and adaptative manner) is still lacking. The showcased systems, while differing from Maestro in several aspects, still present several limitations. The ability to provide this data to external services, the ability to do statistical analysis on the data, and collaborative approaches, such as Maestro's organizational framework, are generally unavailable. Furthermore, all of these works are highly specialized, focusing on specific types of data and/or domains in which they can be applied. Maestro aims to go beyond individual use-cases, allowing organizations to apply the system in many research contexts. A summary of the comparison between these systems and both iterations of Maestro is provided in Table 3.1.

Maestro is unorthodox in the sense that it is not meant to train classification algorithms, nor is it solely focused on gathering and distributing data. Rather, it is aimed at bridging the gap between these processes, which are commonly done separately. Unlike most of the work done in these fields, it is not meant to expand these fields, but to aid researchers and other users to make use of these technologies and techniques, allowing for gathered data to be classified, the process to be adjusted and analysed, and the results distributed.

Table 3.1. Comparison between Maestro's iterations and related works.

	Processes Supported				**Features Supported**			
Work	Gather Data	Classify Data	Analyze Data	Automatically Provide Data	Collaborative Capabilities	Multiple Data types	Simultaneous use of Multiple Classifiers	Extensible Plugins
Maestro V2023	Yes	Yes	Yes	Yes	Yes	Yes[a]	Yes	Yes
Maestro V2022 [11]	Yes	Yes	No	Yes	Yes	Yes[b]	No	Yes
InTime [36]	Yes	Yes	Yes	No	No	No	No	Yes
PoP [38]	Yes	No	Yes	No	No	No	No	No
Extended image collector [35]	Yes	Yes	No	No	No	No	No	No
Image collector [33], [34]	Yes	No	No	No	No	No	No	No

[a] *Image data, sound data, and text data.*

[b] *Image data and sound data.*

4. Maestro Conceptual Aspects

Maestro v2023's conceptual structure is derived from the structure implemented by Maestro v2022. Nonetheless, several modifications and extensions have been made. This chapter presents Maestro v2023's main concepts, and the manner in which they differ from Maestro v2022. Figure 4.1 shows Maestro's domain model in UML, highlighting its top-level concepts. Figures presented in this chapter use the following color scheme: yellow represents classes or concepts which were altered in Maestro v2023; green represents new classes or concepts which have been added in Maestro v2023; white represents classes or concepts which have not been altered in Maestro v2023.

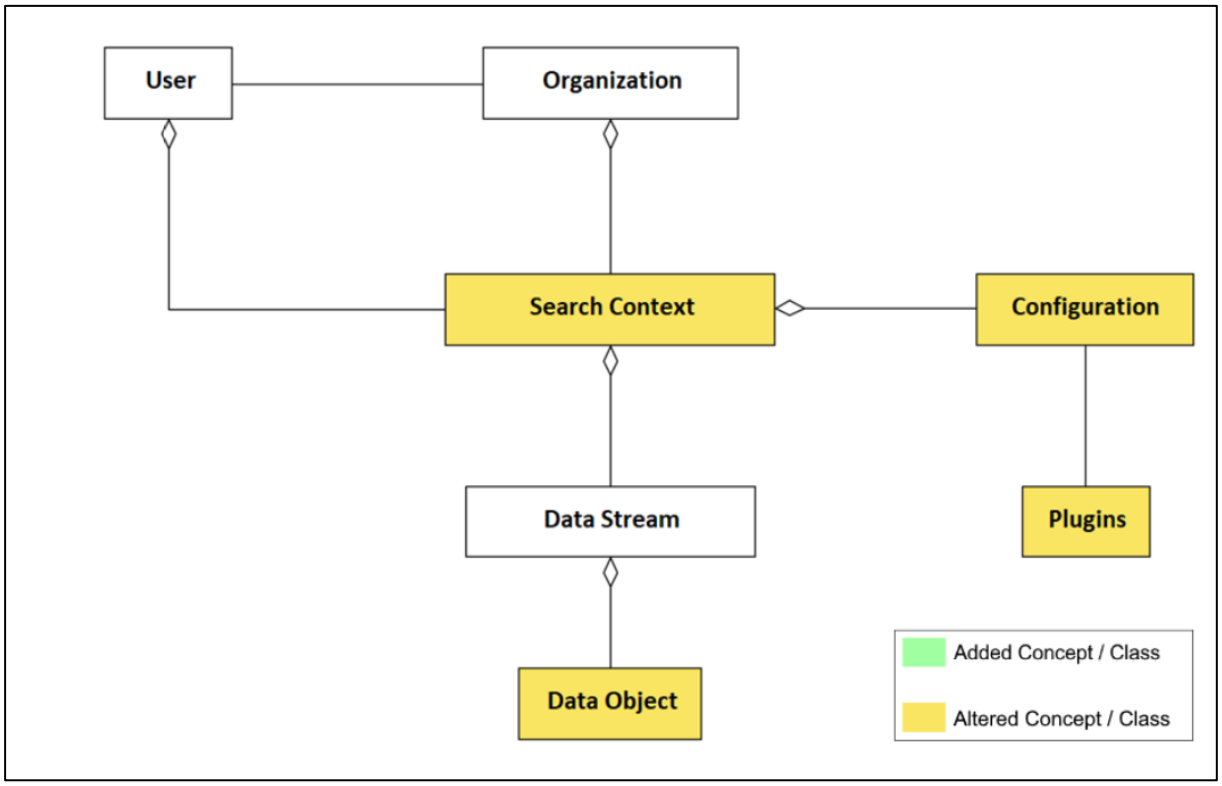


Figure 4.1. Maestro v2023 top-level concepts (UML diagram)

4.1. Search Contexts

Search Contexts function as declarative expressions of the tasks to be run through Maestro's pipeline, serving as a representation of the user's goals within the platform. In Maestro v2023, the concepts that comprise a Search Context have remained largely unchanged:

1. **What data type**: what data type will be used for this search context.
2. **What to search**: by defining the search string used to discover data.
3. **Where to search (optional)**: the data sources that are used.
4. **How to manipulate data (optional)**: what to extract from the data, or how to transform it.
5. **Filtering criteria (optional)**: what data objects should be removed from the dataset.
6. **How to classify data (optional)**: what classification mechanisms or other ML related tasks to perform on the data.
7. **Where to provide data (optional)**: where should the results be sent when finished.
8. **How to analyze data (optional)**: what attributes of the data should be analyzed.

When compared to Maestro v2022, major changes in behavior occur in point 6, as the ability to perform ML tasks beyond classification is new to Maestro v2023, and in point 8, as data analysis is also a new feature in Maestro v2023.

4.2. Users and Organizations

Users and organizations have not been changed in Maestro v2023, and function in the same manner as in the original iteration of Maestro. As before, an organization or user may have several search contexts associated with them.

In order to make use of Maestro's capabilities, a user must first register with an account. After registering, they are allowed to create and configure search contexts. Furthermore, users may create or join different organizations.

An organization is a group of different users that share search contexts, allowing for collaborative projects inside of Maestro. Once a user creates an organization, they may invite other users, remove users, change a user's permissions, or change the organization's settings.

4.3. Data Objects

Maestro stores data gathered throughout a search context's execution. Figure 4.2 depicts a domain model that demonstrates how data is represented inside Maestro.

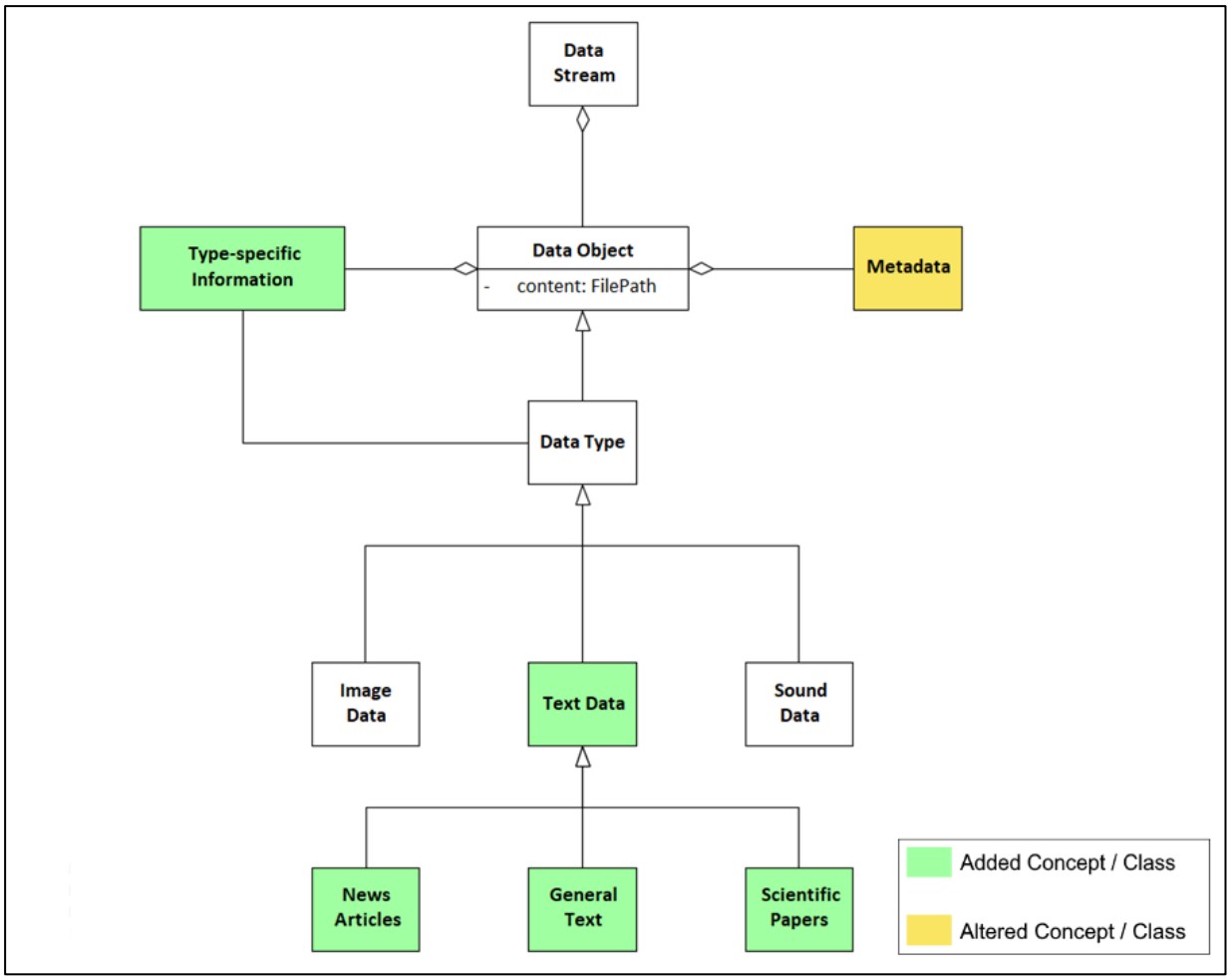


Figure 4.2. Maestro v2023 data hierarchy (UML diagram)

Data within maestro incorporates 4 main concepts:

- **Data stream**: the representation of the set of data objects associated with a given search context.
- **Data objects**: the conceptual representation of each individually gathered item, stored as a row within Maestro's database. It contains any universal information regarding the data (e.g., object identifier), serving as an aggregator for the following concepts.
- **Content**: the actual data (an image, sound, or text document). The content is a file stored within the filesystem, and the file extension can vary (e.g., mp3, jpeg, pdf, or txt).
- **Data type**: the type of the content. Currently, image, audio, general text, and specialized text (news articles and scientific papers) data types are supported. Each data type supports additional type-specific information regarding a data object (e.g., authors of a scientific paper).

Maestro v2023 differs from the previous iteration as it introduces new text data types. Furthermore, data type-specific information was introduced.

In Maestro v2022, any information that was specific to a particular data type would be stored in the metadata. For example, a post-processing plugin could extract the height and width of an image,

storing it in the metadata as a new key-value pair. However, this limited aspects of the platform that required that information to be present at all times, as we cannot be certain that certain fields will be present in the metadata, even if empty.

For text data types, this was particularly problematic, as one could not store information regarding the objects in an organized and accessible manner, preventing us from generating templates using this information. Figure 4.3 presents an overview of the information stored in each data type.

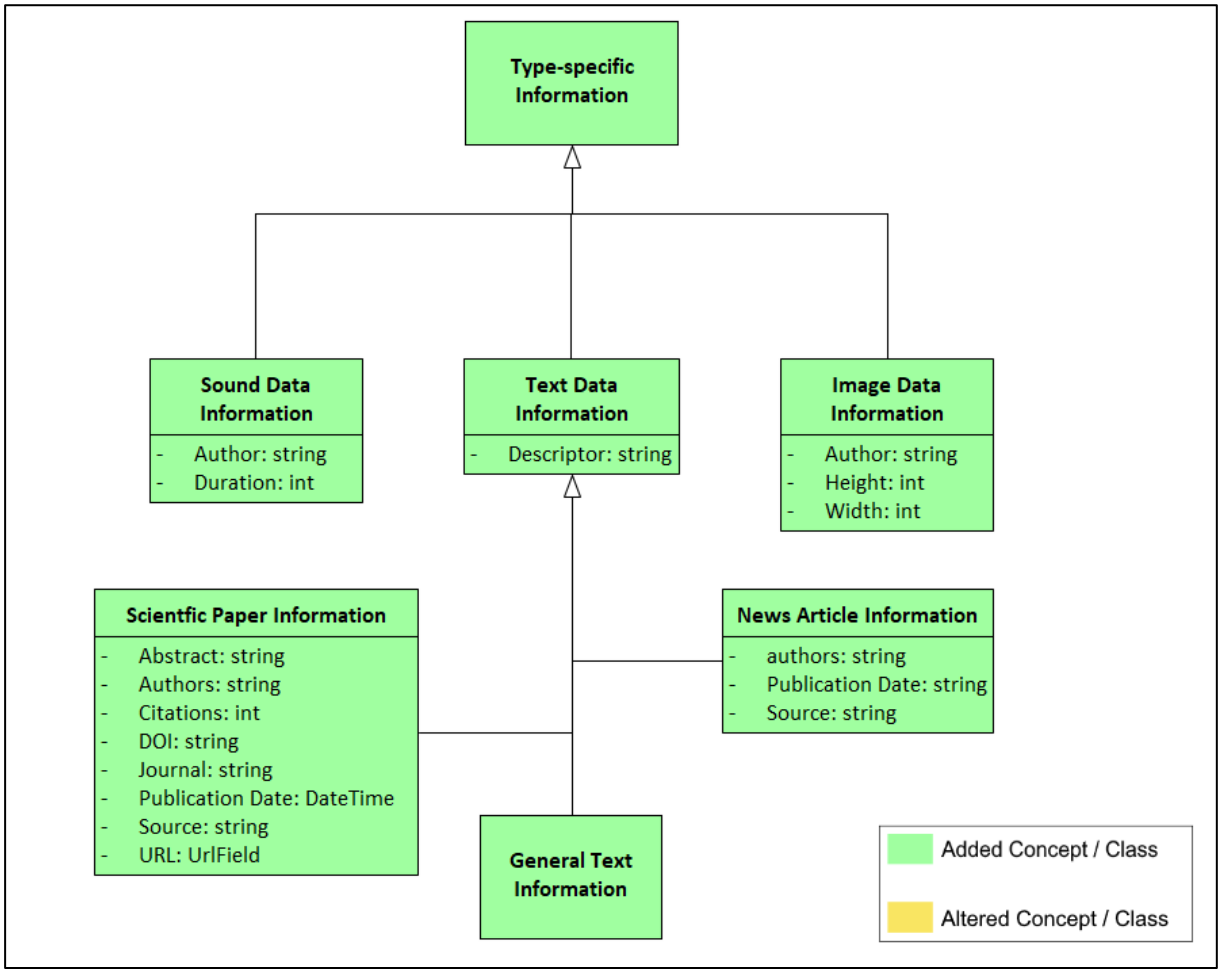


Figure 4.3. Data type-specific information overview (UML diagram)

The previous methodology utilizing metadata prevented Maestro from generating user friendly and comprehensive views of the results provided by a search context, as we would be unable to define what information should be presented, and in what order. Furthermore, plugin compatibility would become very difficult, as there would be no standardized information for a given data object. For example, a classification plugin that required the source for a given news article data object would need to presuppose the format in which this information was stored in the metadata, and either the news article fetching plugins, or a post-processing plugin would have to store this information in the presupposed format. As such, in Maestro v2023, metadata is reserved for storing dynamic information produced or extracted throughout a search context's execution (e.g., extracted EXIF information of an image), and the introduced fields for type-specific information are all optional and may be left blank.

4.4. Search context execution: Conceptual view

As illustrated in Figure 4.4, Maestro supports a pipeline that, once run, results in a classified dataset that can be provided to external services or directly analyzed within the platform. This pipeline is run asynchronously in the background, with the help of workers. This makes the process scalable, as more workers can be posteriorly added, allowing for a larger volume of tasks to be run simultaneously.

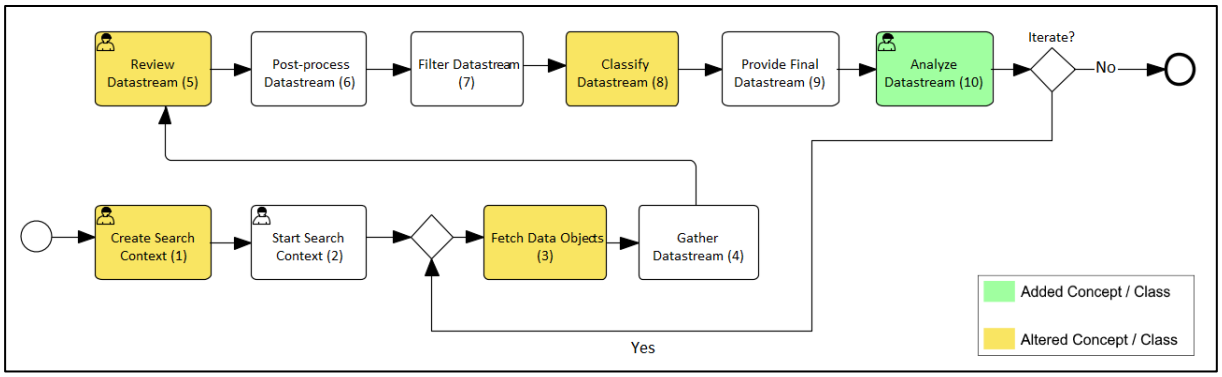


Figure 4.4. Maestro v2023's Workflow (BPMN Process diagram)

Maestro v2023's pipeline comprises ten essential steps or stages, namely:

Stage 1. Create / Configure a search context.

Stage 2. Start a configured search context.

Stage 3. Fetch URLs pointing to objects of the desired data type, as well as additional information regarding the object.

Stage 4. Gather the resources or data items from the fetched URLs.

Stage 5. Review the gathered data items and manually discard those deemed irrelevant.

Stage 6. Post-process the gathered data with the use of plugins, acquiring additional parameters for the subsequent stages (e.g., adding metadata to image data).

Stage 7. Filter the data according to the specified plugins and parameters defined by the user (e.g., filtering based on the date and location of a given data item).

Stage 8. Classify/Perform ML tasks on the dataset items using the desired plugins.

Stage 9. Provide the resulting classified dataset to external services.

Stage 10. Produce data regarding the data stream using the specified plugins and generate charts that allow the user to visually analyze their data.

Maestro 2023 introduces several modifications to this pipeline. Stage 1 was modified, as several usability changes were made to the configuration of search contexts. Stage 3 was extended to allow fetching plugins to fetch type-specific information for the data objects. Stage 5 also presents some usability changes, preventing the user from accidentally discarding their changes prior to completing this stage. Stage 8 was extended to allow for the usage of plugins that do not necessarily perform

classification tasks on the dataset objects. Finally, stage 10 is an addition to the pipeline which allows users to analyze their data stream using charts.

The initial portion of the pipeline, spanning stage 1 and stage 2, is entirely done by the user and focuses on creating, configuring, and starting the execution of the search context. The remaining steps are mostly done by Maestro v2023's workers in the background, with the exception of stage 5, which requires the user's intervention, and stage 8, in which the user may analyze their data stream using the generated charts.

4.5. Plugins

Similarly to Maestro v2022, plugins in Maestro v2023 can be applied to most stages of the pipeline (stage 3 and stages 6 to 10, in Figure 4.4), dictating how the system should handle the data objects when that stage is reached.

In Maestro v2023, fetchers, the plugins to be applied during the fetching step of the pipeline (stage 3 in Figure 4.4), are now able to fetch type-specific information regarding a data object, along with the URL pointing to said object's content. In Maestro v2022 fetchers were focused solely on discovering and providing URLs.

Classifiers, the plugins to be applied during the classification step of the pipeline (stage 8 in Figure 4.4), may now perform tasks that do not necessarily fall into the classification category of the ML field. While the term "classifier" and "classification stage" is still utilized, plugins that allow for different ML related tasks to be applied to the data have been added. For example, summarization plugins for both news article data and scientific publication data, as well as an abstract simplifier and keyword extraction plugins for scientific publication data, were added. By allowing for non-classification focused plugins to be used, the number of scenarios in which Maestro can assist its users increases exponentially. Furthermore, these changes integrate seamlessly with Maestro, providing no apparent drawbacks.

The possibility of applying multiple classification plugins to data objects was also added. Maestro v2022 only allowed a single classification plugin to be configured in each search context. This would mean that users looking to utilize many different classification plugins would have to perform multiple runs in Maestro, while ensuring that the classification results produced during a run were stored, as these would be overwritten by the classification results of a new run.

5. Maestro v2023: Key Features

This chapter provides an in-depth explanation of the major changes, extensions, improvements, and features introduced in Maestro v2023.

5.1. Text Data Types

Maestro is designed to be serviceable in many different scenarios and tasks. As such, it must be able to gather, classify, and provide datasets that differ in nature, characteristics, and purpose. Despite this, Maestro v2022 was limited to only two types of data: image and sound files, which represent a significant amount of the available data on the Web. However, text files and text data are not only abundant on the web but are also one of the major foci of classification and ML algorithms. As such, one of the main extensions added to Maestro v2023 was the introduction of text data types. This provides a much wider breadth of potential scenarios in which Maestro could be of use.

5.1.1. Conceptual Methodology

The addition of text data types into Maestro posed some challenges, as the manipulation of text data differs greatly from the manipulation of image and sound data. To make use of the content available in a text file, or a file that may contain text, the system must be aware of the category and format of the content.

Each file format requires different approaches to crawl through the text and extract the relevant content. For example, a news article, a scientific paper, and a raw text passage all tend to differ in these aspects: while news articles tend to be present inside HTML pages, scientific papers are often made available in PDF files, and plain text is often extracted and stored inside a TXT file (or a similar file extension). Furthermore, the available algorithms and mechanisms used in the development of plugins tend to be trained and synthesized with a focus on specific categories. Classification and ML learning algorithms generally focus on specific categories, such as a news article classifier or summarization algorithm. Though algorithms that can be applied to any text category exist, algorithms such as these are not ordinarily developed with the intention of generalizing them to different content categories and may provide inaccurate or irrelevant output as a result.

5.1.2. Design and Implementation Aspects

In light of the aforementioned challenges, we chose to develop three different text data types. Namely, we introduced a general text data type, which focuses on plain text, as well as two specialized text data types: scientific papers and news articles. The general text data types are stored inside TXT files, where as scientific papers are stored as PDF files, and news articles as HTML files. While these data types could potentially be present in files with different extensions, we found that the majority of them followed this pattern.

While Maestro v2022 only stored basic information regarding data objects in the database, news articles and scientific paper require extra information to be stored. As mentioned in section 4.3 of the previous chapter, dedicated database entries for information regarding these datatypes were introduced, as the previous method of storing metadata limited the platform in various ways. Figure 4.3 presents an overview of the information stored in each data object.

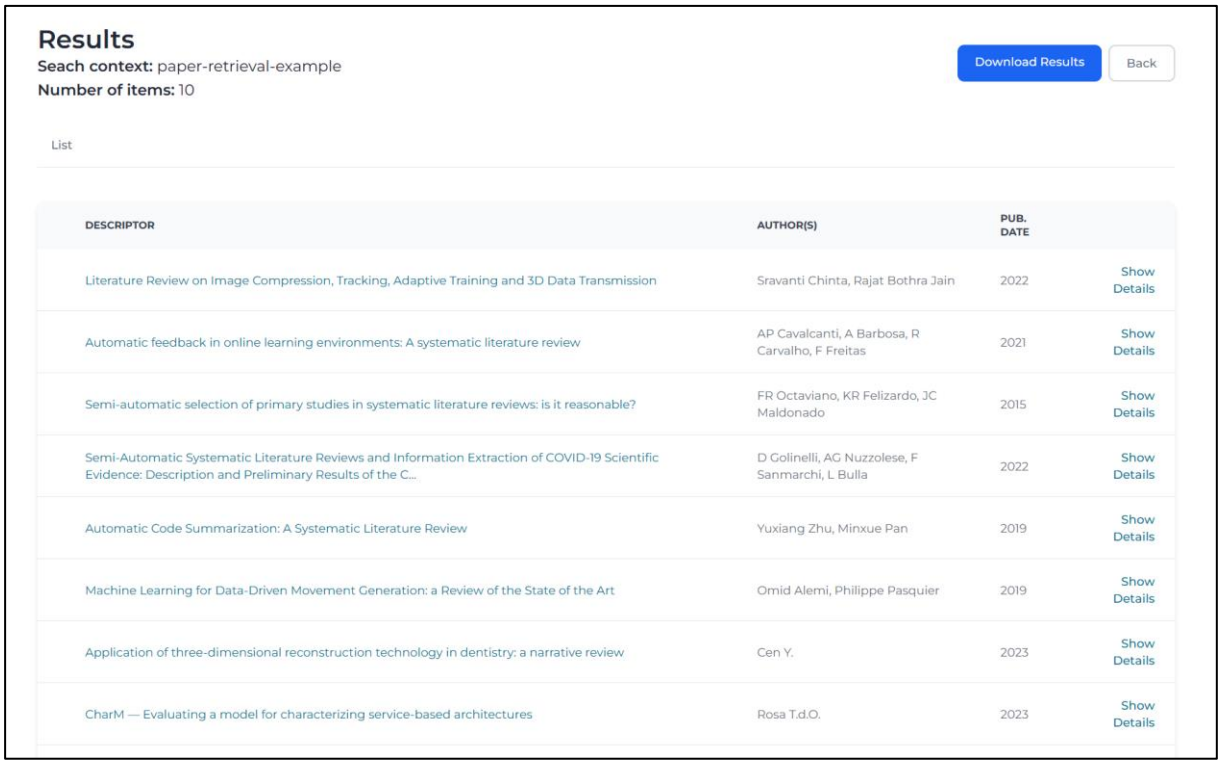
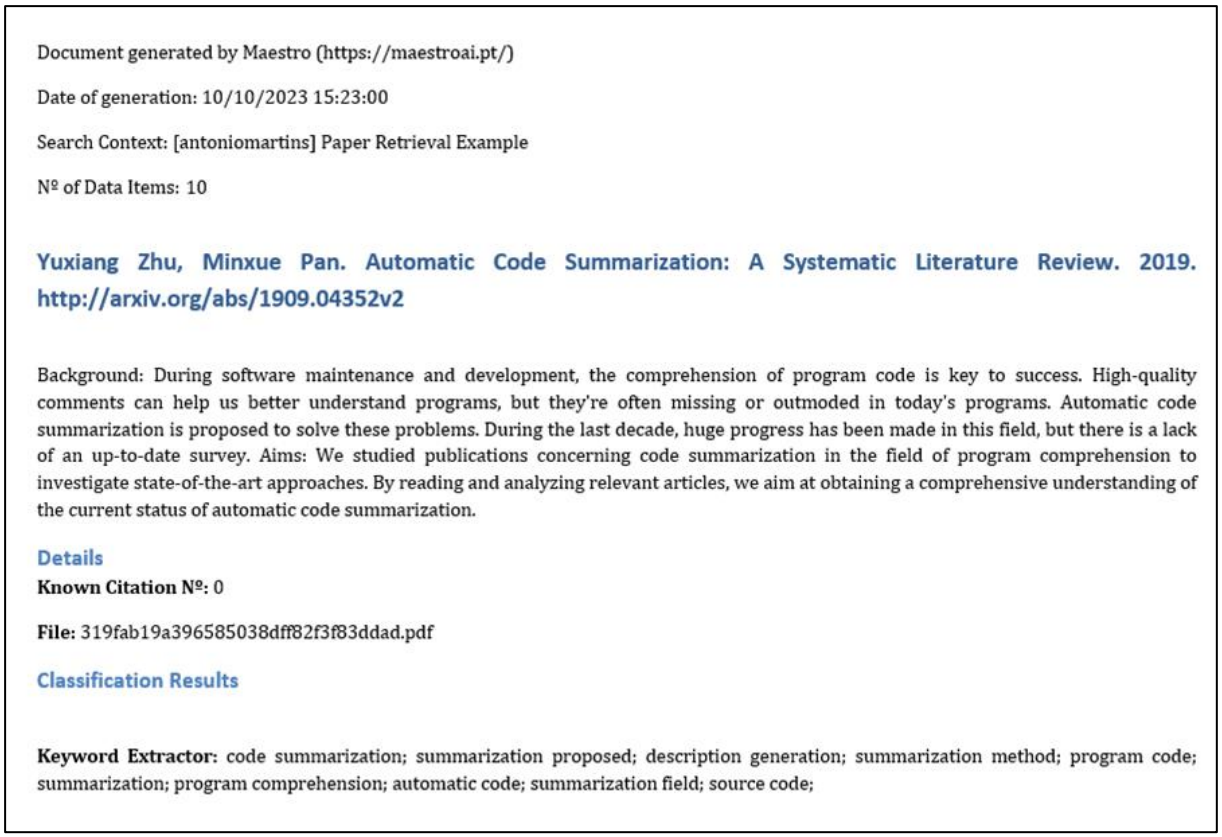


Figure 5.1. Maestro v2023's results page for a scientific paper data stream.

Scientific papers have the most amount of additional data stored. Some of this information is shown to the user in the results page, allowing them to discern the relevance and nature of each gathered object more quickly, as shown in Figure 5.1. The remaining information is shown when the user inspects a data object in more detail.



Document generated by Maestro (https://maestroai.pt/)

Date of generation: 10/10/2023 15:23:00

Search Context: [antoniomartins] Paper Retrieval Example

Nº of Data Items: 10

Yuxiang Zhu, Minxue Pan. Automatic Code Summarization: A Systematic Literature Review. 2019. http://arxiv.org/abs/1909.04352v2

Background: During software maintenance and development, the comprehension of program code is key to success. High-quality comments can help us better understand programs, but they're often missing or outmoded in today's programs. Automatic code summarization is proposed to solve these problems. During the last decade, huge progress has been made in this field, but there is a lack of an up-to-date survey. Aims: We studied publications concerning code summarization in the field of program comprehension to investigate state-of-the-art approaches. By reading and analyzing relevant articles, we aim at obtaining a comprehensive understanding of the current status of automatic code summarization.

Details

Known Citation Nº: 0

File: 319fab19a396585038dff82f3f83ddad.pdf

Classification Results

Keyword Extractor: code summarization; summarization proposed; description generation; summarization method; program code; summarization; program comprehension; automatic code; summarization field; source code;

Figure 5.2. Section of a generated DOCX file for gathered and classified scientific paper data.

Furthermore, extracting this information allowed us to implement a new feature, specifically for scientific papers: the automatic generation of a DOCX presenting each paper in an organized manner, along with a citation for said paper. As presented in Figure 5.2, this should allow users to perform certain tasks, such as literature reviews, with relative ease.

5.2. Analysis Stage

The analysis stage was introduced in Maestro v2023, located after the providing stage of the pipeline. As previously stated, the introduction of an analysis stage into Maestro's pipeline was a logical choice.

Both in Maestro v2023's evaluation stage, as well as the one for Maestro v2022, the usage of the Maestro platform as a medium for data analysis was given as one of the scenarios users found most interesting. Furthermore, the analysis of gathered data, as well as the remaining outputs provided by Maestro during a search context run, allows users to better understand the nature of their data, and how to improve the configurations for future runs. Despite this, this extension is not meant to be utilized to analyze the classification results, as several tools already exist to perform this process. By providing the data to an external endpoint, or manually downloading the datasets provided by Maestro, the data can easily be integrated into these tools. As such, the analysis stage mainly focuses on the analysis of information regarding the data objects themselves, as well as additional information produced by Maestro's pipeline.

5.2.1. Analyzers

As with most other pipeline stages, the analysis stage allows users to create plugins to determine the behaviour and output of the system. Plugins developed for the use in this stage of the pipeline are referred to as analyzers.

An analyzer makes use of an object's attributes and data in order produce labels and generate charts for visual data analysis. In other words, an analyzer will analyze the information regarding a given data object and assign a label to it. The set containing the labels produced for each data object is then utilized to generate a chart that allows users to visually examine their dataset. Analyzers can fall into two categories:

- **Object Analyzers:** analyzers that fall into this category receive a data object's information and make use of it to output a label**.**
- **Labelers:** analyzers that fall into this category receive the set of produced labels by an object analyzer and transform, augment, or otherwise correct it (e.g., add missing labels)**.**

Labelers are an auxiliary type of analyzer. In order for a labeler to function properly, an object analyzer must also be utilized. However, an object analyzer does not necessarily require a labeler to be applied. Rather, a labeler is only needed if transformations to the set of labels produced by an object analyzer are required.

5.2.2. Design Aspects

As illustrated in Figure 5.3, the analysis stage receives a search context's configuration and classified data stream, returning an analyzed version of the data stream.

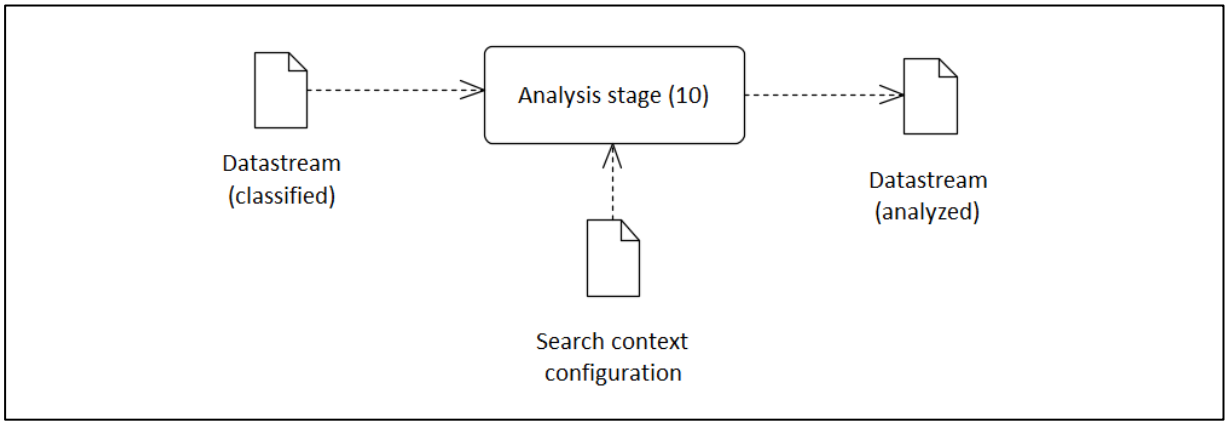


Figure 5.3. Analysis stage diagram.

The analysis stage proceeds as follows (Algorithm 5.1): for each object analyzer and labeler set, the data stream is analyzed. To do this, the object analyzer will run against each of the data objects, outputting a label. If the produced label has not been previously generated, it is added as a key to a

dictionary, with a corresponding value of 1, otherwise, the existing value is incremented by 1. The value for each key indicates the number of data objects for which that label was produced. After this is done for each data object, the dictionary is passed as an argument to the labeler (if available). The labeler will then make changes to the dictionary, outputting the transformed dictionary as a result. Finally, the dictionary keys, as well as the corresponding values, are stored in the database and attached to the search context. These results are then used to generate the desired chart.

```
   Input Final datastream of search context
   Output Analyzed datastream
 1: function ANALYSIS_STAGE(configuration, datastream)
 2:     datastream ← datastream.filtered = False
 3:     for AnalyzerSet ∈ configuration.analyzers do
 4:         Declare labels: type dictionary
 5:         for data ∈ datastream do
 6:             labels[AnalyzerSet.object_analyzer(data)] += 1
 7:         end for
 8:         if AnalyzerSet.labeler ≠ ∅ then
 9:             labels ← AnalyzerSet.labeler(labels)
10:         end if
11:         datastream.analysis_results[AnalyzerSet.name] = labels
12:     end for
13:     return datastream
14: end function
```

Algorithm 5.1. Algorithm executed the analysis stage.

The charts for the analysis stage are produced using the Chart.js library. This JavaScript library allows for a multitude of charts to be generated. Maestro v2023 allows for the generation of bar charts, line charts, pie charts, and bubble charts. When the plugin is selected in Maestro v2023, one of these chart categories must be selected. Once the analysis stage is finished, the user has access to the analysis page. When loaded, Maestro uses the produced output from the analysis stage as data to produce the charts dynamically. This means that the charts themselves are not stored, but rather generated on the go as the analysis page is loaded.

5.3. Developed Plugins

Throughout the development of Maestro v2023, several plugins focused on the newly introduced text data types were developed. Table 5.1 presents a summary of each plugin.

Four different fetching plugins were created, one for news articles, and three for scientific papers: the "News API Fetcher" plugin utilizes the News API [39] to gather different news articles based on the search string provided by the user; the "Elsevier Fetcher" will query the Elsevier API [41] for scientific papers based on the search string provided by the user; The "ArXiv Fetcher" plugin makes use of the ArXiv API [42] to discover scientific papers related to the provided search string; the "Google Scholar Fetcher" utilizes the Scholarly python library [43], along with ScraperAPI [44], in order to crawl and scrape scientific papers indexed by google scholar, based on the search string provided by the user.

Table 5.1. Overview of the developed plugins for Maestro v2023.

Plugin Type	Data Type	Name	Description
Fetcher	News Articles	News API Fetcher	Queries the News API [39] for news articles.
	Scientific Papers	Elsevier Fetcher	Queries the Elsevier API [41] for scientific papers.
		ArXiv Fetcher	Queries the ArXiv API [42] for scientific papers.
		Google Scholar Fetcher	Retrieves scientific papers indexed by google scholar using the Scholarly Python Library [43] and ScraperAPI [44].
Filter	Any	Duplicate Filter	Removes duplicates of data objects by comparing their identifiers.
Post-Processor	News Articles	News Metadata Extraction	Extracts the publication date of news articles.
Classifier	News Articles	News Classifier	Performs sentiment classification on news articles using a finetuned version of the roBERTa model [48].
		News Summarizer	Generates summaries of news articles using the Newspaper3k python library [40].
	Scientific Papers	Paper Summarizer	Generates summaries of scientific papers using the BARTxiv model [46].
		Abstract Simplifier	Rewrites difficult-to-understand scientific abstracts into simpler, easier-to-read versions, using the SAS model [45].
		Keyword Extractor	Extracts relevant keywords from paper abstracts using KeyBERT [47].
Analyzer	Scientific Papers	Citations Analyzer	Generates charts illustrating the distribution of known citations for scientific paper data streams.
		Source Analyzer	Generates charts presenting the distribution of sources for scientific paper data streams.
		Publication Date Analyzer	Generates charts presenting the distribution of publication date for scientific paper data streams.

One filtering plugin was introduced, and functions for all available data types. This plugin filters any duplicate data objects gathered by Maestro. It does this by comparing each data object's identifier or, in the case of scientific papers, the DOI, if available. For scientific papers and news articles, this identifier will be the title. In the case of images and sound data, the identifier is an MD5 hash, generated using the URL pointing to the data object's content, provided during the fetching phase of Maestro's pipeline. This plugin is particularly relevant when it comes to search contexts configured to utilize more than one fetching plugin, as different fetchers may provide the same data objects, or different versions of the same data object may be present in multiple databases.

One post-processing plugin named “News Metadata Extraction” was developed in order to extract the date of news articles.

Five classification plugins were created, two for news articles, and three for scientific papers: the "News Classifier" plugin utilizes an existing sentiment analysis algorithm [48] to label news articles as either "Positive", "Neutral", or "Negative", based on the their content; The "News Summarizer" plugin utilizes the Newspaper3k python library [40] to summarize the contents of a given news article; The "Paper Summarizer" plugin generates summaries of scientific papers using the BARTxiv model [46];

The "Abstract Simplifier" makes use of the SAS model [45] in order to rewrite the abstract for a given scientific paper into easier-to-read versions; The "Keyword Extractor" plugin uses KeyBERT [47] to extract relevant keywords from a scientific paper's abstract.

Finally, three analyzing plugins were developed for scientific paper data: the "Citations Analyzer" plugin is used to generate a bar chart presenting how many of the gathered papers have no citations, 1 to 9 citations, 10 to 49 citations, 50 to 99 citations, and 100 or more citations; the "Source Analyzer" plugin is used to generate a pie chart detailing what percentage of the gathered papers were retrieved by each fetching plugin; the "Publication Date Analyzer" plugin produces a line chart presenting the number of gathered papers published in a given year, starting from the publication year of the oldest gathered paper, to the publication year of the most recently published gathered paper.

5.4. Usability Improvements

Maestro's aim is to gather and classify data as a service. In other words, it serves as an intermediary for the gathering and classification of data. In order for it to be useful in this context, its design should present attributes that aid in this goal. Namely, some attributes it strives to incorporate are:

- **Flexibility**, as it is designed to be of use in many different scenarios, in order to aid in many different use-cases**.**
- **Customizability,** by allowing users to alter Maestro's behaviour through the usage of different plugins and configurations.
- **Extensibility**, as the platform itself allows for the addition of different plugins and modules in a simple and compact fashion.

Maestro v2022's design was envisioned with the focus incorporating these characteristics. However, in order for Maestro to be useful to as many different types of users and scenarios as possible, it ought to remain undemanding and intuitive. While Maestro v2022's usability was generally well rated, these attributes were not necessarily its focal point.

When interpreting the results from the external evaluation of Maestro v2022's usability, provided by test users, it is apparent that usability was one of the lower rated aspects of the platform, with the average score of 4.05 for usability being relatively low compared to the scores of 4.78 and 4.88 for flexibility and extensibility, respectively. Furthermore, this score could itself presents an inherent degree of bias, as all of the users involved in the test session had degrees in IT. Non-tech-savvy users, as well as those in different fields, could potentially have more difficulty navigating Maestro v2022.

By harnessing the feedback provided in the evaluation phase of Maestro v2022's development, as well as the feedback provided over the development of Maestro v2023, several changes to the UI and structure of Maestro were implemented in the new iteration. The following list presents the most significant usability changes introduced in Maestro v2023:

- Removal of several niche configurations that decreased the users' comfort during the configuration stage, while providing little benefit (e.g., tags and non-optional iteration settings for a search context).
- Removal of the "Roadmap" component present in Maestro v2022, which presented a roadmap of the development of the platform, as it did not provide a meaningful benefit to the users, and would have to be continuously updated, becoming increasingly larger with newer versions.
- Addition of a "Publications" component, listing any published articles, datasets, or documents related to Maestro.
- The description of different stages of a search context run were updated to better reflect their intended purpose (e.g., "Paused" rather than "Stopped", and "Concluded" rather than "Finished Providing")
- The layout / labels for different buttons were altered based on feedback.
- The option to proceed during the review stage was disabled until any changes are either saved or reverted, preventing users from accidentally discarding their changes.
- Re-writing and expanding the documentation for Maestro.
- Altered the logs provided by Maestro to the user in each stage.
- Fixed several bugs and visual defects.

5.5. Advanced Search Strings

Maestro v2023 allows for the use of advanced search strings in a search context. A search string can be simple (e.g., “Machine Learning”), or a compound search string that makes use of "and" and "or" tags, as well as brackets, in order to generate various search strings to be provided to the fetchers.

Tags allow users to make use of propositional logic to generate more complex search strings, similar to how one would create conditions in programming. To use "and" and "or" tags, the user should append a dollar sign before the tag: the "and" tag is written as “$AND”; the "or" tag is written as “$OR”; round Brackets can surround blocks of the search string to define the order and target of the tags.

As an example, if the user provided the proposition "(automatic $OR semi-automatic) $AND literature review $AND (system $OR program)" as the search string for a given search context, Maestro would decompose it into the following simple search strings: "automatic literature review system", "automatic literature review program", "semi-automatic literature review system", and "semi-automatic literature review program". Each of these search strings would, in turn, be used by the fetching plugins to find related data objects.

5.6. Data Object Submission

Maestro's manual data object submission feature was also reworked. Maestro v2022 allowed users to submit their own data objects, without having to solely rely on data gathered from a fetching plugin. In Maestro v2023, however, rather than simply allowing users to submit their data objects (e.g., a set of

images or scientific papers), users may also submit information for each data object (e.g., authors, title). This allows users to submit data gathered using different tools, without recourse to Maestro.

```
Input Data object files and information, datastream
Output Extended datastream
1:  function INITIAL_DATASTREAM(submission, datastream)
2:      for file ∈ submission.files do
3:          Declare item ← Create_Data_Object(file)
4:          if submission.object_details[file] ≠ ∅ then
5:              for detail ∈ submission.object_details[file] do
6:                  item[detail] = detail
7:              end for
8:          end if
9:          datastream.add(item)
10:     end for
11:     return datastream
12: end function
```

Algorithm 5.2. Algorithm to manually add data objects to a search context data stream.

To utilize this feature, the user submits a ZIP file containing each data object and, optionally, a JSON file containing the metadata for each of the data objects. The system then creates a new data object using these artifacts and add it to the search context's data stream (Algorithm 5.2).

This feature removes the need for seed URLs, a feature previously available in Maestro v2022. Seed URLs allowed users to manually add URLs to a search context as if fetched using a fetching plugin, which would then be gathered during the gathering stage. However, unlike Maestro v2022, fetchers in Maestro v2023 are not solely capable of fetching URLs to be gathered, as they may also provide additional information and metadata regarding a data object. As such, adding seed URLs would be a very limiting option, as the data objects gathered would be devoid of this type-specific information.

5.7. Other changes

The changes made to the information stored regarding a data object allowed for some particularly interesting changes. Namely, the addition of information regarding the fetching plugin utilized to gather a particular data object allows users to posteriorly analyze the quality of the results provided by a given fetcher. Moreover, storing the URL of the source for a particular data object allows users to visit the source when looking for more information, which is particularly relevant when the full data object is kept behind a paywall (as is the case for certain news articles and scientific papers).

When compared to Maestro v2022, Maestro v2023's download feature is more flexible. Rather than simply allowing users to either download the provided data, or the metadata and classification results for said data, users may now select what things they would like to retrieve from the platform: the object data, the object metadata, the classification results, or any combination of these choices.

Additionally, scientific paper datasets allow for the generation of DOCX file presenting each paper's metadata, classification results, and a generated citation.

6. Demonstration

This chapter presents some usage scenarios demonstrating Maestro v2023's capabilities, along with the framework of a search context execution from the user's point of view.

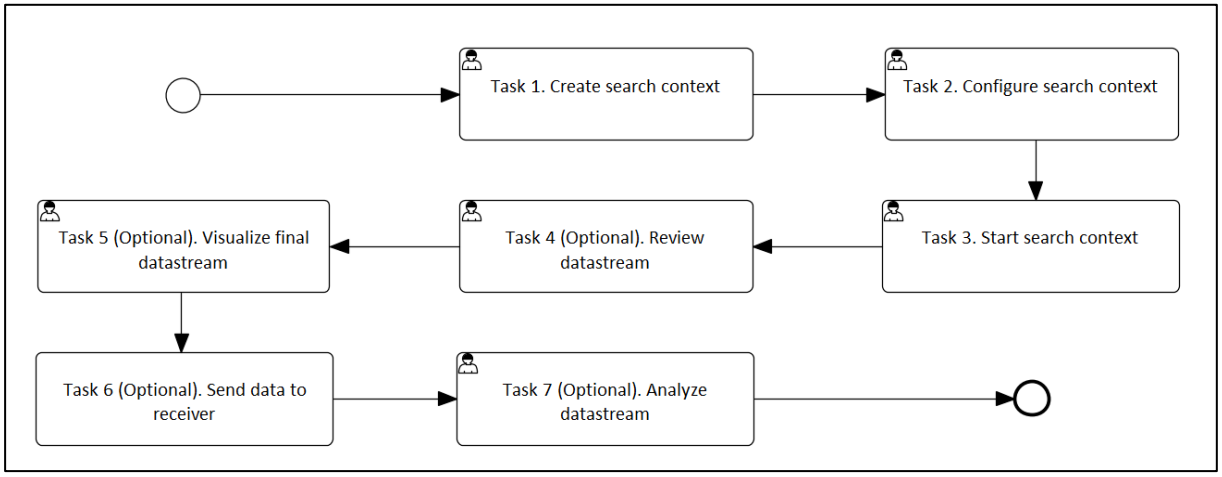


Figure 6.1. Sequence of tasks a user needs to perform during the execution of a search context (BPMN Process diagram).

Despite Maestro v2023's extensive pipeline, each stage progresses to the next one automatically. Because of this, a technical view of Maestro differs greatly from how a user perceives and interacts with the platform. As illustrated in Figure 6.1, from the user's point of view, executing a search context can be seen as a sequence of tasks:

Task 1. Create a search context. A search context has an owner (the current user or one of the organizations the user belongs to), a name, a unique code, and a description. This task is performed during the first stage of Maestro v2023's pipeline (see Figure 4.4).

Task 2. Configure the search context. There are two types of configurations: essential configuration and advanced configuration, where the former is mandatory, and the latter is optional. The essential configuration includes the minimum set of parameters required for a search context execution, namely a search string, the associated data type, and information regarding whether the run should repeat iteratively over time. The advanced configuration defines Maestro's behavior during each of the pipeline's stages. While the advanced configurations are optional, defining what should happen during

the fetching stage is recommended, as the system will perform no actions otherwise. This task is performed during the first stage of Maestro v2023's pipeline (see Figure 4.4).

Task 3. Start the search context execution. This task is performed during the second stage of Maestro v2023's pipeline (see Figure 4.4).

Task 4. (Optional) If the search context is configured to "Yield after gathering data", then the execution stops after the gather stage to allow the user to manually inspect the obtained data stream and remove data objects that are not relevant. When finished, the execution can be resumed. This task is performed during the fifth stage of Maestro v2023's pipeline (see Figure 4.4).

Task 5. (Optional) Inspect the resulting data stream which includes, for each data object, the classification result(s), as well as any additional information and metadata regarding the data object. Furthermore, the user is able to download the data stream files and/or the associated metadata and classification results to their device. This task is performed during the fifth stage of Maestro v2023's pipeline (see Figure 4.4).

Task 6. (Optional) If the user configured an HTTP Endpoint for the providing stage, the data stream is sent to that endpoint when the execution is finished. The user may then import the data stream into an external tool. This task is performed during the ninth stage of Maestro v2023's pipeline (see Figure 4.4).

Task 7. (Optional) Inspect the charts generated with recourse to any analyzers configured to be used in the analysis stage of the pipeline. The last task performed by the system is to gather the data used to generate these charts. This task is performed during the tenth stage of Maestro v2023's pipeline (see Figure 4.4).

This framework is used to demonstrate Maestro v2023's capabilities in the presented usage scenarios.

6.1. Demonstrative Scenarios

This section provides an overview of Maestro v2023's capabilities through two different usage scenarios: (i) gauging the "social climate" of corporations and businesses; (ii) streamlining literature reviews and semi-automatic abstract generation.

6.1.1. Usage Scenario 1

This scenario demonstrates how Maestro v2023 can be utilized by organizations to gauge their "social climate" (i.e., the public's general opinion and perceptions), in light of recent events and information that appears in the media or Internet. The monitorization of this "social climate" can be done in various ways, but Maestro serves as a useful tool to facilitate this process.

In this scenario, a user would like to utilize Maestro to gather recent news articles related to their company (in this example, Amazon), summarize them automatically, and apply a sentiment analysis classifier to label each article as either "Neutral", "Positive", or "Negative". Furthermore, the user wishes to execute this process periodically, every 20 days. This allows them to periodically identify the media's general opinion regarding their business, allowing them to make better informed decisions.

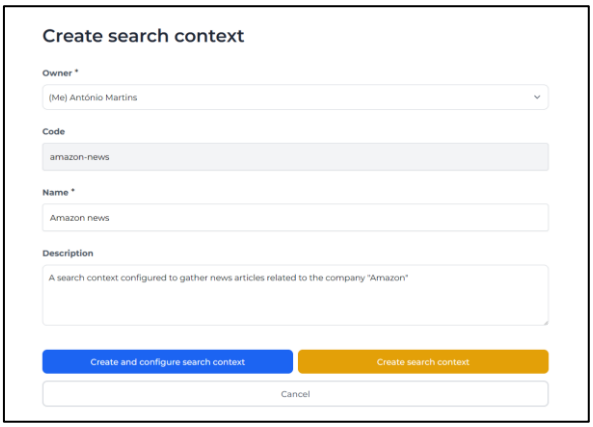


Figure 6.2. Creation of a new search context in Maestro v2023.

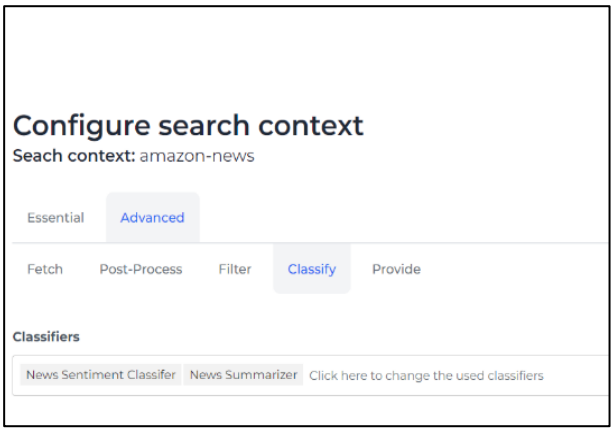


Figure 6.3. Configuration of two data classification plugins for a search context.

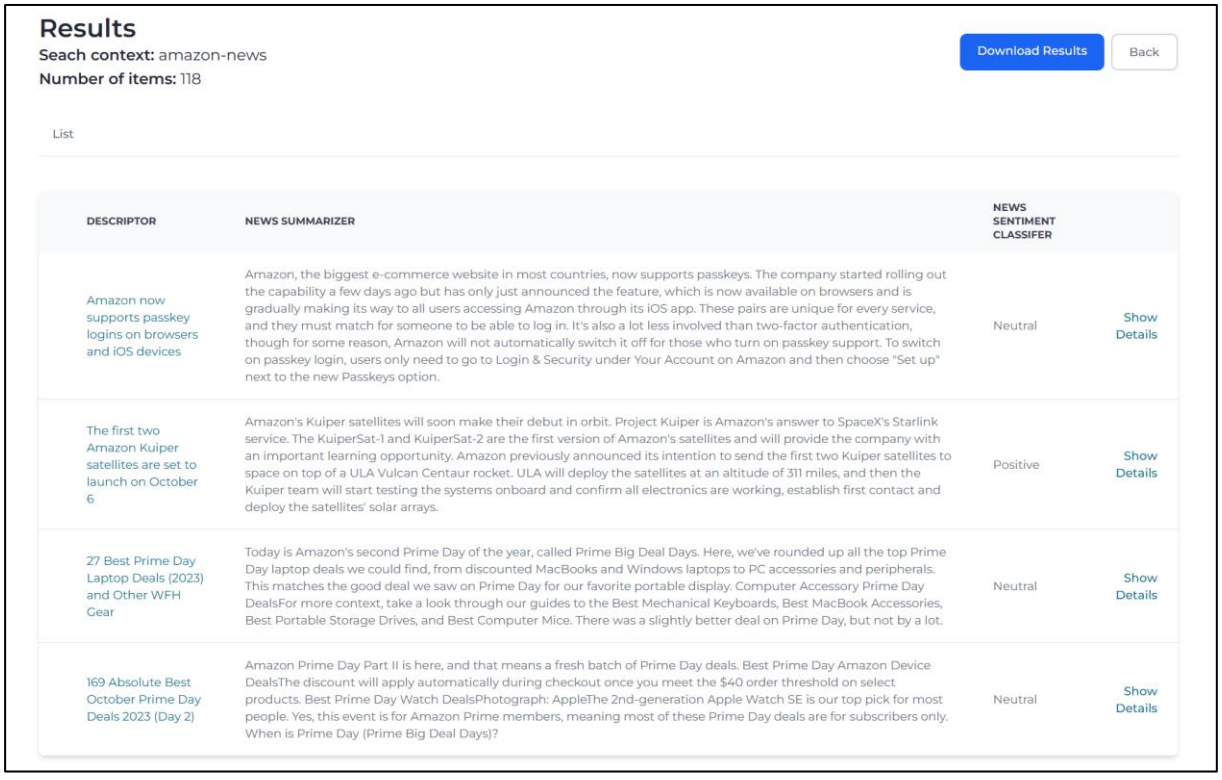


Figure 6.4. Results from a search context configured to gather and classify news articles.

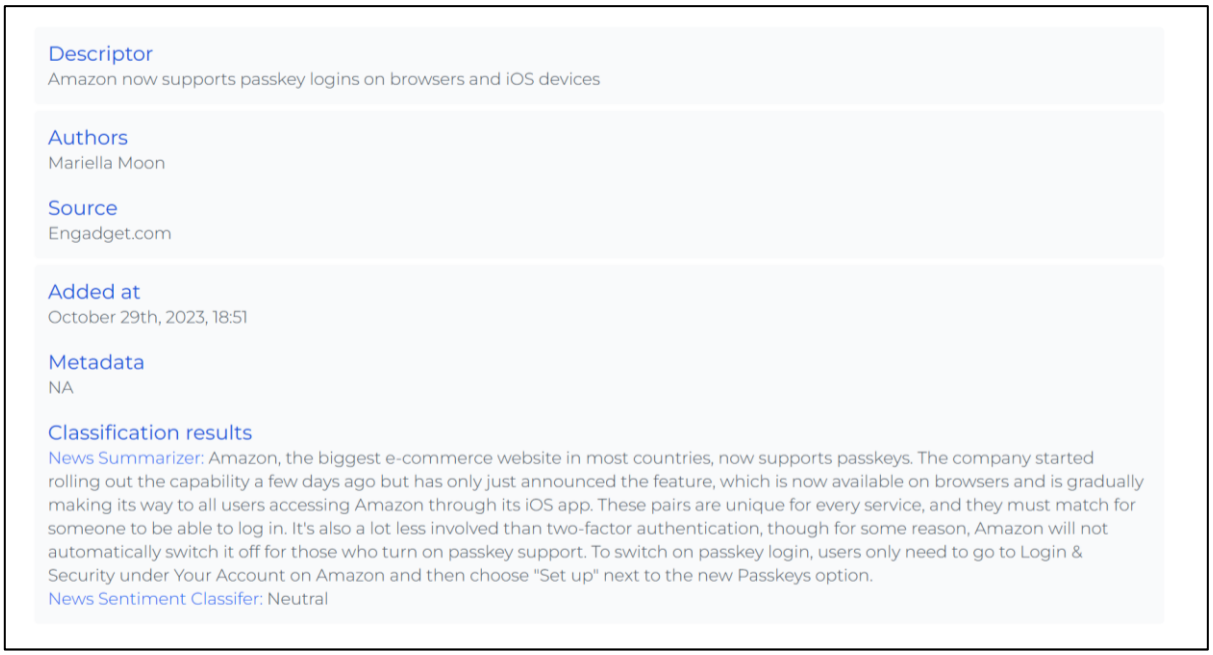


Figure 6.5. Detailed view of a news article data object's properties

To execute this scenario, the user would proceed by following the proposed framework:

1. The user creates a search context through Maestro's interface, as suggested in Figure 6.2. The user must define an owner, title, unique code, and a description for their search context.
2. Once created, the user must configure his search context.
   2.1. The user defines the essential configurations, which are mandatory. The user must define the search string for finding the data ("(Business $OR Brand) $AND Amazon"), the data type ("News Articles"), as well as other options that allow the search context to automatically run again after a certain amount of time (every 20 days).
   2.2. The user can then define the advanced configurations. Despite these settings not being mandatory, the system will do nothing if they are not configured. In this task, the user shall proceed by conducting the following tasks: select a plugin for fetching URLs of news articles related to their search string, using the News API. [Select "News API" fetch plugin]; select a classifier plugin for news summarization, and another for sentiment analysis, to be considered during the classification step, illustrated in Figure 6.3. [Select "News Summarizer" and "News Sentiment Classifier" classification plugins]; select a post-processing plugin to extract the date of each article. [Select "News Metadata Extraction" post-processing plugin]; apply filtering configurations in order to discard any gathered articles whose publication date (extracted in the previous step) is prior to the current month. [Configure data objects with a date prior to the current month to be discarded]; though optional, the user may also specify the configurations for an HTTP Rest endpoint to which the data will be sent during the providing step. [Configure the information for the desired HTTP Rest Endpoint].
3. The user presses the "Start" button in the search context menu to initiate their run.
4. As the user would like the search context to run iteratively, without having to access Maestro, manual inspection and filtering of the data stream is not configured.

5. Once the classification stage ends, the user accesses the "Results" page to inspect the data object that were not filtered, as well as the respective results provided by the classification plugins. As illustrated in Figure 6.4, the system provided the user with multiple news articles related to the company Amazon, summarized them, and classified them according to the aforementioned settings. Selecting the “Show Details” option also allows users to see each data object in finer detail (See Figure 6.5).
6. The system sends the results to the configured endpoint, which the user may then utilize as input for external tools and processes.
7. Since the user has no interest in analyzing the data through Maestro, no analysis plugins were selected. As such, no charts were generated for the user to inspect.

This scenario demonstrates how Maestro v2023 may aid businesses and corporations in their decision-making. Furthermore, it demonstrates how a search context can be configured to gather, classify, and provide results to an external source automatically and iteratively, serving as a long-term intermediary in a number of challenging scenarios.

6.1.2. Usage Scenario 2

This usage scenario, originally presented in a paper publish on ISD-2023 [49] about Maestro, demonstrates how Maestro v2023 can aid researchers in performing literature reviews. Namely, we showcase how the platform can be of use to the scientific community by automating several aspects of the literature review process.

In any type of scientific research, literature reviews play a vital role by gathering pertinent and current research on a particular subject and consolidating it into a comprehensive overview of the existing knowledge in the field. However, the process of writing literature reviews can be a dauting task, especially for less experienced members of the scientific and academic community, such as postgraduate students, who can find it particularly difficulty to engage in this subject, independently of their proficiency [50]. As such, research on the subject of automating literature reviews has emerged, with the aim of diminishing the hardships associated with this process. The field of ML has further propelled research on this topic, allowing for more complex systems for automating literature reviews to be developed.

Streamlining literature reviews can be achieved with the aid of Maestro v2023, as it can aggregate several of the tools and mechanisms used for literature review automation. Following the proposed framework, the user would go through the following tasks:

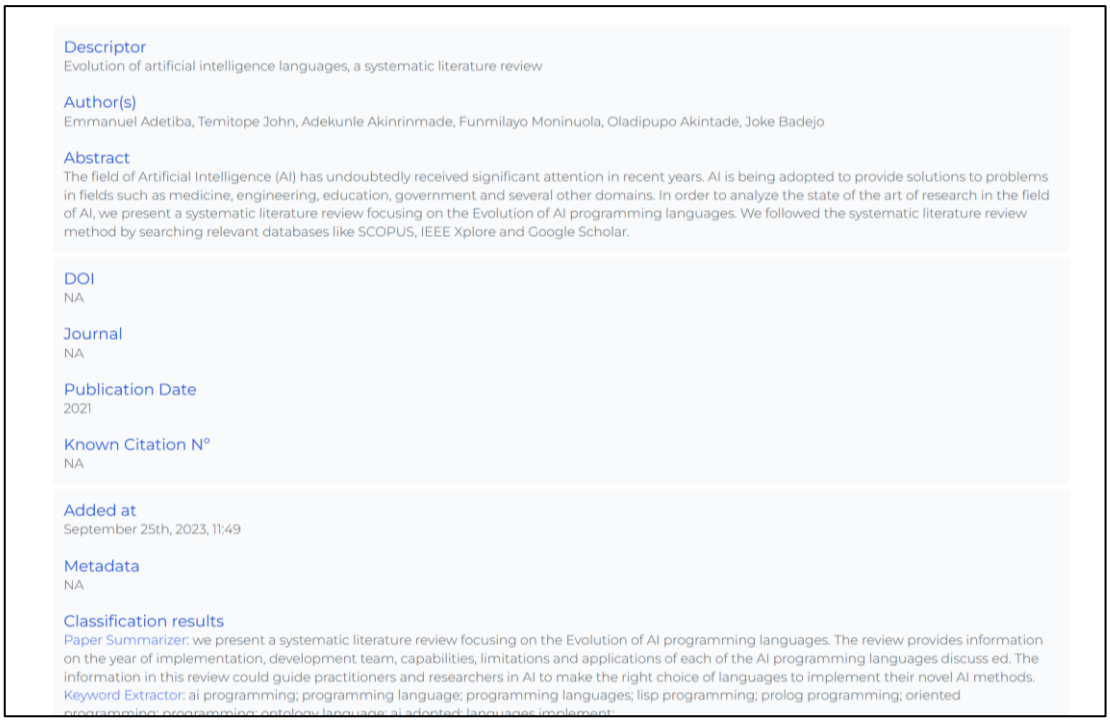


Figure 6.6. Detailed view of a scientific paper data object's properties.

1. The user creates a search context through Maestro's interface, as illustrated in Figure 6.2. The user must define an owner, title, unique code, and a description for their search context.
2. Once created, the user must configure his search context.
   2.1. The user defines the essential configurations, which are mandatory. They define the search string for finding the data as "(automatic $OR semi-automatic) $AND literature review $AND (system $OR program)", the data type ("Scientific Paper"), as well as other options that allow the search context to automatically run again after a certain amount of time (in this case, we set it to "Don't repeat").
   2.2. The user can then define the advanced configurations. In spite of these settings not being mandatory, the system will do nothing if they are not configured. In this phase, the user shall proceed by conducting the following tasks: select the "Elsevier API" and "ArXiv API" fetching plugins for fetching URLs of scientific papers related to the search string; manually submit any previously gathered data objects to be included in the data stream; select the "Yield data after gathering" option to allow for manual data stream review; select the "Paper Summarizer", "Abstract Simplifier", "Keyword Extractor" plugins, to be considered during the classification step; apply filtering configurations in order to discard any duplicates of gathered articles, by checking their DOI and/or Title; though optional, the user may also specify the configurations for an HTTP Rest endpoint to which the data will be sent during the providing step; finally, the user selects the "Citations Analyzer" and "Publication Date Analyzer" plugins to generate charts for data object analysis.
3. The user presses the "Start" button in the search context menu to initiate their run.
4. After the gathering stage, the user will manually review the dataset and discard any irrelevant object gathered by Maestro.

5. Once the classification stage ends, the user accesses the "Results" page to inspect the data object that were not filtered, as well as the respective results provided by the classification plugins. As illustrated in Figure 5.1, the system provided the user with multiple scientific papers related to the defined search string, summarized them, simplified the abstract, and extracted relevant keywords from the original abstract. Selecting the “Show Details” option also allows users to see each data object in finer detail (See Figure 6.6).
6. The system sends the results to the configured endpoint, which the user may then utilize as input for external tools and processes.
7. After the analysis stage ends, the user may then access the charts generated with the help of the data produced using the analysis plugins. As illustrated in Figure 6.7, the user is able to visually analyze information regarding the produced data stream. The generated charts present a distribution of the number of citations found for the gathered papers, as well as the gathered papers' publication year.

Figure 6.7. View of the charts produced using the “Citations Analyzer” and “Publications Date Analyzer” plugins.

While this scenario showcases how Maestro v2023 may be used to significantly decrease the time needed to discover relevant publications for a given topic, many different possibilities for the use of this platform exist. During the writing of the previously mentioned paper for ISD 23, Maestro v2023 was also used to generate the abstract and keywords.

To achieve this, we made use of Maestro v2023's manual dataset submission feature to run a draft of the paper through the pipeline, with the goal of summarizing it using the “Paper Summarizer” plugin. This summarization was then inserted into a separate run of our pipeline and, through the use of the “Abstract Simplifier” plugin, produced a more coherent version of this summarization. Finally, after some modification to the produced artifact, we passed the result to Maestro once again and

made use of the “Keyword Extractor” plugin to identify potential keywords to be used in the final version of the paper (See Appendix A).

7. Evaluation

This chapter presents the results of the user assessment performed during this work, serving as a tool to evaluate Maestro v2023's usability and perceived usefulness.

7.1. Assessment Methodology

To assess Maestro v2023's usability and relevance, we followed a similar approach to the one employed in Maestro v2022, in which a user test session was prepared.

To support this test session, a user guide (Appendix B) was produced, detailing the tasks necessary to perform the scenario described in section 6.2.1 of the previous chapter, as well as a questionnaire for the testers to fill out regarding their perceived experience when following the guide. Test sessions following the user guide were performed under the following circumstances:

- The test sessions were held remotely and done asynchronously, without the developer's intervention.
- The users had no prior experience with Maestro v2023 and were not allowed to ask for clarification when following the user guide.
- Access to a computer or tablet with a reliable internet connection and modern browser was required to follow the test guide accurately.

The preparation and execution of this test session consisted of the following tasks:

Task 1 (Preparation): A PDF document serving as a user guide was created, briefly introducing the concept of Maestro and its current iteration, Maestro v2023. Details regarding the goal of the test session were then provided, along with a step-by-step guide on how to utilize Maestro v2023 to perform the scenario described in section 6.2.1 of the previous chapter.

Task 2 (Preparation): A questionnaire was prepared using Google Forms, which consisted of 2 sections with a total of 17 questions (disclosed in Appendix C – User Assessment questionnaire):

- **Section 1:** 6 questions (Q1 to Q6) to determine the participant's profile, namely: age, gender, nationality, highest academic level, type of degree, and comfort with computers.

- **Section 2:** 11 questions (Q7 to Q17) aimed at evaluating the participant's experience using the platform, their thoughts on its qualities, and the usefulness it may have in real-world scenarios. The questions used a scale from 1 to 5 to perform the assessment, and most were interleaved with open questions to leave room for the opinions or comments of users. Furthermore, the users were asked to provide the downloaded results from the test session, as proof that it was completed beforehand.

Task 3 (Preparation): A call for testers was announced and shared through various sources. Those interested in participating were provided with a link to the user guide and questionnaire.

Task 4 (Execution): Each participant performed the experiment by following the instructions in the PDF and filling out the evaluation questionnaire.

Task 5 (Analysis): We collected all the responses submitted by the participants and analyzed the results.

7.2. Questionnaire Analysis

To gather participants for the test session, a call for testers was announced and shared through various sources, including social media platforms (e.g., LinkedIn), mailing lists, and in-person meetings. Furthermore, the test session was shared in an international conference, ISD2023 Proceedings in Lisbon, in which a paper regarding Maestro v2023 was published [49]. In total, 18 users participated in the test session and filled out the questionnaire.

As aforementioned, this questionnaire had a total of 17 questions divided into two sections. It is also important to point out that the questionnaire follows the same structure as the one used for the evaluation of Maestro v2022's usability. This allowed us to compare the results and determine whether the changes made in Maestro v2023 were beneficial to the users' experience.

7.2.1. Participant Profiles

The results from the first section gave us insight into the profiles of the participants. The results for this section are presented in Table 7.1.

We believe the general diversity of the participants to be a positive factor, especially regarding the highest academic level and experience Computer Science related fields. While most of the users self-assessed themselves as being very comfortable and experienced with computers, this is to be expected, as a degree of bias in those interested and capable of participating in this test session is likely to be present.

Table 7.1. Summary of participants' answers regarding their personal profiles.

Category	Response	Number of participants	Percentage of participants (%)
Age **(Q1)**	18 to 25	11	61.1%
	26 to 59	7	38.9%
Gender **(Q2)**	Male	12	66.7%
	Female	4	22.2%
	Non-binary	1	5.6%
	Unknown	1	5.6%
Nationality **(Q3)**	Portuguese	12	66.7%
	American	3	16.7%
	Other	3	16.7%
Highest Academic Level **(Q4)**	Bachelor's degree	10	55.6%
	Master's degree	5	27.8%
	PhD	1	5.6%
	Associate's degree	1	5.6%
	High School	1	5.6%
Field **(Q5)**	Degree or professional experience in CS related field	12	66.7%
	No degree or professional experience in CS related field	6	33.3%
Comfort with Computers **(Q6)** [a]	5 (very comfortable)	15	83.3%
	4 (comfortable)	2	11.1%

[a] *Using a Likert scale from 1 to 5 (where 1 is very uncomfortable and 5 very comfortable)*

7.2.2. Evaluation Results

The second section of the questionnaire can be subdivided into 4 dimensions:

Dimension 1 (Q8 to Q10): The difficulty of performing certain actions (e.g.: create an account, configure search context).

Dimension 2 (Q11 to Q12): The usefulness of Maestro capabilities (usability, flexibility, performance, usefulness).

Dimension 3 (Q13 to Q14): Maestro's serviceability in a real-world situation.

Dimension 4 (Q15 to Q17): The participants' overall experience and comments.

Analysis of the answers for the first dimension demonstrates that, in general, users found performing most actions to be very easy.

Q8 (see Table 7.2) used a Likert scale from 1 to 5 to gauge the ease in performing certain actions in Maestro v2023 (where 1 is very hard and 5 very easy). For this question, 5 was the most selected option in every category, except for search context configuration. Despite the changes made to simplify this process in Maestro v2023, this is still the hardest task to perform in Maestro v2023. Nonetheless, the average reported score for this task was 4.28.

Table 7.2. Summary of the responses for Q8 of the questionnaire, aimed at gauging the perceived difficulty in task completion.

Task	Number of responses for a given score, using a Likert scale from 1 to 5 (where 1 is very hard and 5 very easy), regarding task completion difficulty										Average Score
	1		2		3		4		5		
	Count	%	Count	%	Count	%	Count	%	Count	%	
Create user account	0	0%	0	0%	0	0%	4	22.2%	14	77.8%	4.78
Create search context	0	0%	0	0%	2	11.1%	4	22.2%	12	66.7%	4.56
Configure search context	0	0%	0	0%	2	11.1%	9	50%	7	38.9%	4.28
Start of search context	0	0%	0	0%	1	5.6%	2	11.1%	15	83.3%	4.78
Monitor progress of run	0	0%	1	5.6%	0	0%	3	16.7%	14	77.8%	4.67
Review Data	0	0%	1	5.6%	1	5.6%	3	16.7%	13	72.2%	4.56
View Results	0	0%	1	5.6%	0	0%	2	11.1%	15	83.3%	4.72
Download Results	0	0%	1	5.6%	0	0%	2	11.1%	15	83.3%	4.72

Table 7.3. Summary of the responses for quantitative close-ended questions of the questionnaire.

Question	Response	Number of participants	Percentage of participants (%)
Did you find the configuring stage challenging? **(Q9)**	Yes	0	0%
	A bit	6	33.3%
	No	12	66.7%
Do you see situations where you could use Maestro in your day-to-day life? **(Q13)**	Yes	13	72.2%
	Maybe	2	11.1%
	No	3	16.7%

In Q9 (see Table 7.3), 12 (66.6%) of the users reported that the configuring stage was not challenging, with the remaining 6 (33.3%) users reporting they found it somewhat challenging. For those that found it somewhat challenging, a common response when prompted for feedback in Q10 was that the configuring stage would have been hard to perform without the help of the user guide, suggesting that mechanisms such as "hints" to help guide users though the process might be a relevant change to consider in future work. For the remaining tasks, lower scores were uncommon, and were usually reported due to the presence of bugs that have since been fixed.

When comparing the answers provided in this category to the ones from Maestro v2022's test session questionnaire, we were able to determine that, overall, the results were either similar or improved. Namely, the scores given to the ease in reviewing the data increased. Furthermore, only 33.3% of users reported finding the configuration stage somewhat challenging, while the results from Maestro v2022's indicate that, at the time, 37% of users found it somewhat challenging, and 21% found it challenging. While there were some slight decreases in the score given to the difficulty in performing certain tasks in Maestro v2023, this is most likely due to the aforementioned bugs that

have since been resolved. Furthermore, all the participants in the Maestro v2022 test session reported having experience in the field of computer science, as well as the lowest reported academic level being a bachelor's degree, which may have skewed the results for Maestro v2022's test session toward more positive results.

The second dimension aimed at gathering the participants' opinions regarding Maestro v2023's capabilities, particularly its usability (ease of use/interaction), flexibility (ability to extend to multiple use cases), performance (how fast the interactions are), and usefulness (real world applications). Not all qualities could be perceived during the interaction with Maestro v2023, having the users been asked to read the paper if they wanted to know more.

Table 7.4. Summary of the responses for Q11 of the questionnaire, aimed at gauging the participants' opinion on Maestro v2023's capabilities.

Category	Number of responses for a given score, using a Likert scale from 1 to 5 (where 1 is very poor and 5 very good), regarding Maestro v2023's capabilities in different categories										Average Score
	1		2		3		4		5		
	Count	%	Count	%	Count	%	Count	%	Count	%	
Usability	0	0%	0	0%	1	5.6%	7	38.9%	10	55.6%	4.5
Flexibility	0	0%	0	0%	0	0%	8	44.4%	10	55.6%	4.56
Performance	0	0%	1	5.6%	0	0%	4	22.2%	13	72.2%	4.61
Usefulness	0	0%	1	5.6%	1	5.6%	5	27.8%	11	61.1%	4.44

Q11 (see Table 7.5) used a Likert scale from 1 to 5 (where 1 is Very Poor and 5 Very Good) to gauge each user's perceived opinion of Maestro v2023 in the proposed categories. In general, we found that the results were quite positive, with the average score for each of the categories ranging from 4.44, in terms of perceived usefulness, to 4.61, in terms of perceived performance, with flexibility and usability having an average score of 4.56 and 4.5, respectively.

When compared to Maestro v2022's test session results, we can see that the scores provided for flexibility of the system decreased slightly. As Maestro v2023 expands upon Maestro v2022, allowing for a wider number of use cases, we assume that the users might have either misunderstood the concepts underlying Maestro, or that the undiversified profiles of the users from Maestro v2022's test session might have skewed the results originally. There was also a significant increase in the usability score, with the average score reported for Maestro v2022 being 4.05. This indicates the changes made regarding usability in Maestro v2023 have had a positive impact in the users' overall experience when using Maestro. The usefulness and performance scores are not comparable to the results for Maestro v2022, as these categories were not present in the original questionnaire for Maestro v2022's test session.

The feedback provided in Q12 further reflects the users' overall positive impression when using Maestro v2023. "The UI is easy on the eyes", "I would definitely use Maestro", and "This was surprisingly fast" are all examples of the feedback provided by the users. One user suggested flagging automatically filtered results, rather than outright excluding them from the data stream, which could be a worthwhile change to be made in future iterations of Maestro.

The third dimension enquired users on whether they could see themselves using Maestro in their daily lives.

Unlike the questionnaire for Maestro v2022, where the users were asked to rate how likely they were to utilize Maestro in their daily lives using a 5-point Likert Scale, Q13 (see Table 7.3) asked the participants if they could see themselves using Maestro v2023 in their daily lives. here, 13 (72.2%) of the participants answered "Yes", 2 (11.1%) answered with "Maybe", and 3 (16.7%) said no. While the results for Maestro v2022 were more one sided in terms of positive outlook for this dimension, with all participants saying they had at least some interest in using the platform in their daily lives, we believe our own results are still positive and showcase a high degree of interest from the participants in using Maestro v2023. Some provided answers for Q14 regarding what situations the participants could see themselves using Maestro v2023 for were "Literature Reviews", "As a search engine", "Scientific research", and "Data analysis".

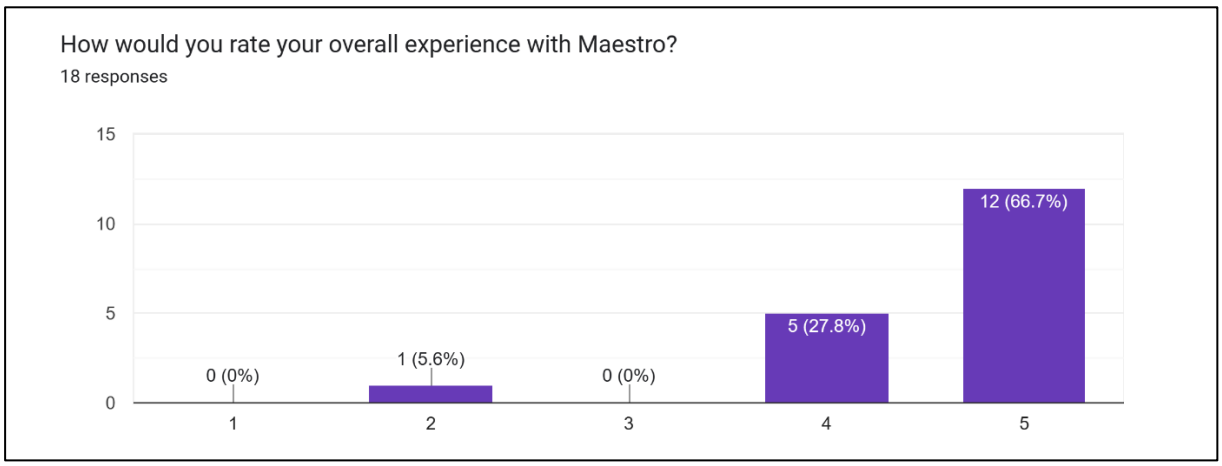


Figure 7.1. Results from the questionnaire on overall satisfaction with Maestro v2023.

The fourth and final dimension was aimed at gauging the participants' overall experience using Maestro v2023.

In Q15 (see Figure 7.1) participants were asked to rate their experience using the platform using a Likert scale from 1 to 5 (where 1 is Very Poor and 5 Very Good), and the results show an average score of 4.55, where only a single user rated their experience below as negative, with a score of 2. This is likely due to reported encounters with bugs that were since fixed, as well as some inconsistencies in the user guide, which have also been corrected. Feedback for Maestro v2022 achieved similar scores, with an average of 4.63, indicating that interest in using the Maestro platform is still present, despite an increasing number of ML platforms becoming available in recent times.

When prompted for general comments regarding their experience in Q16, users reported finding the application easy to interact with, responding: "You've done an excellent job on the layouts, UX and responsiveness of the application", "It was simple to use and navigate, and the GUI and looks very clean and fun to operate with", and "Very easily readable program, nothing was very difficult to comprehend or locate. It was very straightforward". In Q17, the respondents did provide bug reports and feedback regarding the UI, which were taken into account and fixed, posteriorly.

In summary, the results of this user assessment seemed positive and promising. Most participants seemed to like Maestro v2023, both in terms of usability, as well as usefulness. Though some negative outliers exist, we believe the results were still well within our expected outcomes. The biggest limitation we found in regard to this approach was the relatively small number of testers. For future iterations of Maestro, attracting a larger number of testers would surely be beneficial.

8. Conclusion

This chapter covers the conclusion attained after completing the development of Maestro v2023 and discusses some of the possible improvements that can be performed to further enhance the platform.

8.1. Main contributions

This work has achieved the research goals proposed in section 1.4. The first research goal was achieved by performing a thorough examination of the state of the art in several domains related to Maestro and the proposed alterations to the platform, as well as a comparison of related works with both iterations of the platform.

To achieve the second research goal, Maestro v2023, the second iteration of the Maestro platform, was developed, expanding the original version of the platform in multiple ways. A wide array of improvements to Maestro were added in this new iteration, greatly expanding the number of scenarios in which Maestro may be of use. We believe the addition of text data types, introduction of a new analysis stage to Maestro's pipeline, several usability-oriented improvements, along with numerous other modifications and features, have led to an increase in Maestro's quality and usefulness.

Finally, the third research goal was attained by performing an evaluation of Maestro v2023, during which several users participated in a test session, following the steps necessary to perform the scenario outlined by one of the usage scenarios proposed. Analysis of these results shows that, in general, some of Maestro v2022's biggest limitations were successfully addressed, as the perceived usability of the platform increased. Furthermore, several of the participants showed interest in Maestro v2023's potential to be of use in their daily lives.

During the development of this work we were also able to publish and present its progress in an international conference, ISD2023 Proceedings in Lisbon, under the track “Data Science and Machine Learning” [49].

The increase in Maestro's quality following this new iteration, as well as the positive results of our evaluation, leads us to believe in Maestro's potential to impact several different fields, such as business management and scientific research. Maestro v2023's flexibility and depth of functionalities allows it to serve as an intermediary tool for many different projects, bridging the gap between data gathering and the manipulation of data using different ML mechanisms.

8.2. Future work

The work done on Maestro v2023 addresses several of the limitations of the original iteration of the Maestro platform, allowing for future iterations to expand in a flexible manner. Nonetheless, the Maestro platform can still be enhanced in several ways. We propose the following limitations and improvements be considered in future iterations of the platform.

Programmatic interface. Though it streamlines the process of gathering, classifying, and manipulating data, Maestro nonetheless has an unnecessary overhead, caused by the necessity of utilizing the platform's web UI. For experienced users, having a programmatic interface that allows them to automate their access to Maestro could improve their experience. Furthermore, if a user is able to create a script that enables them to interact with Maestro, it should allow them to have a finer control over their desired configurations. For example, it could allow them to run their search contexts when desired, or automatically change the configurations based on the results.

Several approaches to introduce a programmatic interface are available. However, as Maestro is implemented using the Django framework for python, deploying a REST API should be a straightforward solution [57]. It should provide users with the ability to change the platform's database, where all the information regarding not only their search context's status and configurations, but also their profiles and organizational settings. To aid this feature, detailed documentation on how to use the API should be created, helping first-time users to make use of this alternative interface.

Simplification of the configuration process. While several improvements were made to increase Maestro's ease-of-use, the configuring step is still perceived by users as the most difficult task to be performed in Maestro v2023. This is likely due to the large number of configurations a user must define when creating a search context.

To decrease the complexity of the configuration process, we suggest several solutions. The introduction of a tutorial or guide for first time users, going in-depth regarding Maestro's functionalities, may aid users during this task. Adding "hints" suggesting what the user has yet to configure, may decrease the perceived complexity of the process. Finally, improving the UI of the platform further, as well as considering a different nomenclature for the plugins (e.g., fetching plugins would be called "Sources", rather than "Fetchers"), should make this process more intuitive to users.

Security concerns. Throughout the development of Maestro v2023, several cyber-security flaws were identified: the ability for organization members to access personal search contexts of other organization members; accessing the data gathered from other search contexts, by abusing the cache system, used to speed up a search context's execution; ability to bypass certain cross site scripting (XSS) protection mechanisms. The identified security concerns have since been fixed. Nonetheless, further flaws in Maestro's security may be present.

To tackle this issue, we propose an audit and thorough investigation of Maestro's architecture be performed, and that the major identified issues are subsequently mitigated.

Expansion of the analysis stage. The analysis stage introduced in Maestro v2023 allows users to analyze their data streams through the generation of informative charts. Nonetheless, the types of charts available, and the information that may be displayed, are limited. For example, it would not be possible to generate a chart illustrating the distribution of scientific paper's publication year, subdivided by fetcher used to discover said papers.

More robust methods of data analysis of the data streams may increase maestro's capabilities, despite a potential increase in the complexity of performing such a task. To broach this topic, we suggest allowing plugin developers to submit a file detailing the way a chart should be generated, rather than the current approach of generalizing the forms of charts available.

Dynamic data types. Currently, new data types must be manually inserted into the system, and new data models generated in Maestro's database. This limits the user to utilize Maestro in situations requiring the currently available data types. As such, allowing for dynamic data types to be introduced, bypassing the need for the platform's developers to implement a data type, would be particularly relevant to users.

Designing a solution for this limitation is not trivial, however. It would require careful design of a dynamic data type that could allow users to submit data types (similarly to how plugins function), taking into consideration the existence of unforeseeable or complex use cases. Nonetheless, we believe mitigating this limitation would be a major achievement.

Automatic addition of new plugins. In order to streamline the incorporation of new plugins, the current method, which involves the manual insertion of plugins by Maestro administrators, could be replaced with an automated process. This automated system would feature a designated portal for administrators to assess and approve new plugins, ensuring a more scalable and efficient way of adding them to the platform.

Dynamic configurations. Certain configurations and options regarding plugins in Maestro are not configurable. For example, API Keys must be defined in Maestro's configuration files after the plugin is approved. This prevents certain configurations and choices that may be relevant for the users from being configured. As such, we believe the addition of dynamic configurations, which the user would define during the configuring stage, would be an interesting addition to future iterations of Maestro.

In practice, this could be achieved by adding a JSON file associated with the plugins, which would inform Maestro of the dynamic settings and their nature. Users would then be prompted to define the value for these configurations when adding the plugin to their search contexts.

References

1. Internet Assigned Numbers Authority [IANA]. (2023, January 4). Media Types. Retrieved August 25, 2023, from https://www.iana.org/assignments/media-types/media-types.xhtml.
2. Nagel, S. (2022). Statistics of Common Crawl Monthly Archives by commoncrawl. Retrieved August 25, 2023, from https://commoncrawl.github.io/cc-crawl-statistics/plots/mimetypes.
3. Rasmussen, E. M. (2003). Indexing and retrieval for the Web. Annual Review of Information Science and Technology (ARIST), 37, 91-124.
4. Bar-Ilan, J. (2005). The use of web search engines in information science research. Annual Review of Information Science and Technology, 38(1), 231–288. doi: 10.1002/aris.1440380106.
5. Olston, C., & Najork, M. (2010). Web Crawling. Foundations and Trends® in Information Retrieval, 4(3), 175–246. doi: 10.1561/1500000017
6. Zhao, B. (2017). Web scraping. Encyclopedia of big data, 1-3.
7. Chen, H., & Chau, M. (2003). Web Mining: Machine Learning for Web. Annual Review of Information Science and Technology 2004, 38, 289
8. Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., … Amodei, D. (2020). Language Models are Few-Shot Learners. ArXiv: Computation and Language. doi: 10.48550/arXiv.2005.14165.
9. Zhang, M., & Li, J. (2021). A commentary of GPT-3 in MIT Technology Review 2021. Fundamental Research, 1(6), 831–833. doi: 10.1016/j.fmre.2021.11.011.
10. Chen, I. Y., Johansson, F. D., & Sontag, D. (2018). Why Is My Classifier Discriminatory. Neural Information Processing Systems, 31, 3543–3554. doi: 10.48550/arXiv.1805.12002.
11. Serra, Alexandre & Estima, Jacinto & Rodrigues da Silva, Alberto. (2022). Maestro: An Extensible General-Purpose Data Gathering and Classification System. Proceedings of ISD'2022. A15. doi: 10.13140/RG.2.2.26824.80646.
12. Gonzalez, M.G.B.: RiverCure Portal: Collaborative GeoPortal for Curatorship of Digital Resources in the Water Management Domain. Master's thesis, Instituto Superior Técnico (2020)
13. Hevner, S. March, J. Park, and S. Ram, "Design Science in Information Systems Research", MIS Quartly, vol. 28, no. 1, pp. 75–105, 2004, doi: 10.1007/BF01205282.
14. K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, "A Design Science Research Methodology for Information Systems Research", Journal of Management Information Systems, vol. 24, no. 3, pp. 45–77, 2007.
15. F. Gacenga, A. Cater-Steel, M. Toleman, and W. Tan, "A proposal and evaluation of a design method in design science research", Electronic Journal of Business Research Methods, no. 10, pp. 89–100, 2012.
16. Olston, C., & Najork, M. (2010). Web Crawling. Foundations and Trends® in Information Retrieval, 4(3), 175–246. doi: 10.1561/1500000017.
17. Kumar, M., Bhatia, R., & Rattan, D. (2017). A survey of Web crawlers for information retrieval. WIREs Data Mining and Knowledge Discovery, 7(6). doi: 10.1002/widm.1218.
18. Gunawan, D., Amalia, A., & Najwan, A. (2017). Improving Data Collection on Article Clustering by Using Distributed Focused Crawler. Data Science: Journal of Computing and Applied Informatics, 1(1), 1–12. doi: 10.32734/jocai.v1.i1-82.
19. Sriram Raghavan and Hector Garcia-Molina (2001). Crawling the Hidden Web. In Proceedings of the 27th International Conference on Very Large Data Bases (VLDB '01). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 129–138.
20. Avarikioti, G. (2018, November 4). Structure and Content of the Visible Darknet. arXiv.org. doi: 10.48550/arXiv.1811.01348.

21. Scrapy | A Fast and Powerful Scraping and Web Crawling Framework. (2022). Retrieved November 25, 2022, from https://scrapy.org/.
22. Bda-Research: Web crawler/spider for nodejs, Retrieved May 25, 2023, from https://github.com/bda-research/node-crawler.
23. Faizan, M., & Khan, R. A. (2019). Exploring and analyzing the dark Web: A new alchemy. First Monday. doi: 10.5210/fm.v24i5.9473.
24. Aggarwal, C. C. (2020). Data Classification: Algorithms and Applications (Chapman & Hall/CRC Data Mining and Knowledge Discovery Series) (1st ed.). Chapman and Hall/CRC.
25. Hugging Face – The AI community building the future. (2022). Available at https://huggingface.com.
26. Caraffini, M., & Landro, N. (2020). NTSNET: nicolalandro/ntsnet-cub200. GitHub. Available from https://github.com/nicolalandro/ntsnet-cub200.
27. Twitter API | Products. Twitter Developer Platform. Retrieved May 25, 2023, from https://developer.twitter.com/en/products/twitter-api.
28. Bing Image Search API | Microsoft Bing. Bingapis. Retrieved May 25, 2023, from https://www.microsoft.com/en-us/bing/apis/bing-image-search-api.
29. Pixabay API (2022). Pixabay. Retrieved December 20, 2022, from https://pixabay.com/api/docs/.
30. Badillo, S., Banfai, B., Birzele, F., Davydov, I. I., Hutchinson, L., Kam-Thong, T., Siebourg-Polster, J., Steiert, B., & Zhang, J. D. (2020). An Introduction to Machine Learning. Clinical pharmacology and therapeutics, 107(4), 871–885. doi: 10.1002/cpt.1796.
31. I. Dilrukshi, K. De Zoysa and A. Caldera, "Twitter news classification using SVM," 2013 8th International Conference on Computer Science & Education, 2013, pp. 287-291, doi: 10.1109/ICCSE.2013.6553926.
32. Han, J., Kamber, M., & Pei, J. (2006). Data Mining: Concepts and Techniques, Second Edition (The Morgan Kaufmann Series in Data Management Systems). Chapter 6 - Classification and Prediction. Morgan Kaufmann.
33. Yanai, K.: Image collector: an image-gathering system from the world-wide web employing keyword-based search engines. In: IEEE International Conference on Multimedia and Expo, 2001. ICME 2001. pp. 523–526 (2001).
34. Yanai, K.: Image collector ii: a system for gathering more than one thousand imagesfrom the web for one keyword. In: 2003 International Conference on Multimedia and Expo. ICME '03. Proceedings
35. Generic image classification using visual knowledge on the web. In: Proceedings of the eleventh ACM international conference on Multimedia. pp. 167–176 (2003)
36. P. Koloveas, T. Chantzios, S. Alevizopoulou, S. Skiadopoulos, and C. Tryfonopoulos, "intime: A machine learning-based framework for gathering and leveraging web data to cyber-threat intelligence," Electronics, vol. 10, no. 7, p. 818, 2021.
37. "MISP Open-Source Threat Intelligence Platform: Open Standards For Threat Information Sharing." Retrieved December 29, 2022, from https://www.misp-project.org.
38. Harzing, A.W. (2007) Publish or Perish. Retrieved May 20, 2023, from https://harzing.com/resources/publish-or-perish.
39. News API | Search News and Blog Articles on the Web. Retrieved May 18, https://newsapi.org/.
40. Ou-Yang, L. (2013). Newspaper3k Version (0.1.0.7). Newspaper3k Documentation. Retrieved May 18, 2023, from https://newspaper.readthedocs.io/en/latest/#.
41. Elsevier Developer Portal. (2023). Elsevier. Retrieved August 20, 2023, from https://dev.elsevier.com/.
42. ArXiv API (2022). ArXiv. Retrieved August 20, 2023, from https://info.arxiv.org/help/api/index.html.
43. Arun, K. (2021). Scholarly. Retrieved September 6, 2023, from https://github.com/scholarly-python-package/scholarly..
44. ScraperAPI. (2023). ScraperAPI. ScraperAPI - The Proxy API for Web Scraping. Retrieved September

6, 2023, from https://www.scraperapi.com/.

45. Wang, H. (2022). Scientific abstract simplification. Hugging Face. Retrieved August 10, 2023, from https://huggingface.co/haining/scientific_abstract_simplification.
46. Du, J. (2022, December). BARTxiv. Hugging Face. Retrieved August 10, 2023, from https://huggingface.co/kworts/BARTxiv.
47. Grootendorst, M. (2022). KeyBERT: Minimal keyword extraction with BERT. GitHub. Retrieved August 10, 2023, from https://github.com/MaartenGr/KeyBERT.
48. Srinath, G. (2022). News Sentiment Analysis. Hugging Face. Retrieved August 10, 2023, from https://huggingface.co/shashanksrinath/News_Sentiment_Analysis/.
49. Martins, A.M., Rodrigues da Silva, A., & Estima, J. (2023). Streamlining Literature Reviews Using an Automatic and Flexible Data Gathering and Classification Platform. In A. R. da Silva, M. M. da Silva, J. Estima, C. Barry, M. Lang, H. Linger, & C. Schneider (Eds.), Information Systems Development, Organizational Aspects and Societal Trends (ISD2023 Proceedings). Lisbon, Portugal: Instituto Superior Técnico.
50. Shahsavar, Z., & Kourepaz, H. M. (2020). Postgraduate students' difficulties in writing their theses literature review. Cogent Education, 7(1). https://doi.org/10.1080/2331186x.2020.1784620
51. Chakraborty, G., Pagolu, M., Garla, S. (2013). Introduction to Text Analytics. In Text Mining and Analysis: Practical Methods, Examples, and Case Studies Using SAS (pp. 1–17). essay, SAS Institute Inc.
52. Widyassari, A. P., Rustad, S., Shidik, G. F., Noersasongko, E., Syukur, A., Affandy, A., & Setiadi, D. R. I. M. (2020). Review of automatic text summarization techniques & methods. Journal of King Saud University - Computer and Information Sciences, 34(4), 1029–1046. doi: 10.1016/j.jksuci.2020.05.006.
53. El-Kassas, W. S., Salama, C., Rafea, A., & Mohamed, H. K. (2021). Automatic text summarization: A comprehensive survey. Expert Systems with Applications, 165, 113679. doi: 10.1016/j.eswa.2020.113679.
54. Xu, W., Callison-Burch, C., & Napoles, C. (2015). Problems in Current Text Simplification Research: New Data Can Help. Transactions of the Association for Computational Linguistics, 3, 283–297. doi: 10.1162/tacl_a_00139.
55. Al-Thanyyan, S., & Azmi, A. M. (2021). Automated Text Simplification. ACM Computing Surveys, 54(2), 1–36. doi: 10.1145/3442695.
56. "The web framework for perfectionists with deadlines | Django." Retrieved October 26, 2023, from https://www.djangoproject.com/.
57. Django REST framework. Retrieved Oct. 26, 2023, from https://www.django-rest-framework.org/.
58. "What is Python? Executive Summary". Python.org. Retrieved October 26, 2023, from https://www.python.org/doc/essays/blurb/.
59. Stack Overflow Developer Survey 2022. Stack Overflow. Retrieved October 26, 2023, from https://survey.stackoverflow.co/2022.
60. Stack Overflow Developer Survey 2023. Stack Overflow. Retrieved October 26, 2023, from https://survey.stackoverflow.co/2023.
61. Celery - Distributed Task Queue. Retrieved October 26, 2023, from https://docs.celeryq.dev/en/stable/.
62. RabbitMQ: easy to use, flexible messaging and streaming. Retrieved October 26, 2023, from https://www.rabbitmq.com/documentation.html.
63. P. G. D. Group, PostgreSQL, Oct. 14, 2023. Retrieved Oct. 26, 2023, from https://www.postgresql.org/.

Appendix A – Abstract Generation Outputs

Stage	Description	Output
Raw Summary	The original raw output provided by Maestro's pipeline when using the "Paper Summarizer" plugin, developed using the BARTxiv model [46], on the draft of the paper.	"we present how to streamline the process of writing literature reviews by automating several aspects of the process using an automatic and flexible data gathering and classification platform called Maestro v2023. The platform is designed to be modular and configurable, allowing users in an organization to create search contexts that automatically gather and classify data for them. \n the platform also allows for additional \nmodular steps, such as data filtering a nd post - processing, further expanding its range of applications and capabilities."
Sanitized Summary	A manually sanitized version of the raw summary above. This was done in order to remove any improper spelling or undesired text elements.	"We present how to streamline the process of writing literature reviews by automating several aspects of the process using an automatic and flexible data gathering and classification platform called Maestro v2023. The platform is designed to be modular and configurable, allowing users in an organization to create search contexts that automatically gather and classify data for them. The platform also allows for additional modular steps, such as data filtering and post-processing, further expanding its range of applications and capabilities."
Simplified Summary	The output provided by Maestro when using the "Abstract Simplifier" plugin, developed using the SAS model [45], on the sanitized summary above.	"We present how to streamline the process of writing literature reviews by automating several aspects of the process using an automatic and flexible data gathering and classification platform called Maestro v2023. The platform is designed to be modular and configurable, allowing users in an organization to create search contexts that automatically gather and classify data for them. The platform also allows for additional modular steps, such as data filtering and post-processing, further expanding its range of applications and capabilities."
Final Abstract	The finalized abstract for the paper, written based on the simplified summary above.	"Literature reviews are a crucial but time-consuming and complex task in scientific research. As such, interest in automating this process, using machine learning techniques, has increased over the last few years. In this paper, we present a method of streamlining the process of writing literature reviews by automating several aspects of the process using Maestro v2023, an automatic and flexible data gathering and classification platform. Maestro v2023 is a revamped version of the original Maestro platform, designed to be modular and configurable, allowing users in an organization to create search contexts that automatically gather and classify data for them. By analyzing the work related to literature review automation, we suggest how Maestro can contribute to this field, demonstrating how the system was utilized in order to streamline our own literature review process, as well aid us in formulating the abstract and extracting relevant keywords to this paper."
Extracted Keywords	The output provided by Maestro when using the "Keyword Extractor" plugin, developed using KeyBERT [47], on the final abstract.	"literature reviews; documents search; literature review; reviews using; literature; write literature; writing literature; machine learning; generation literature; automatic generation;"

Appendix B – Test Session User Guide

An Extensible General-Purpose Data Gathering and Classification Platform: Maestro v2023

Assessment Session – User Guide, v1.2a, 2023/October

António Valente Martins (MSc Student)
Alberto Manuel Rodrigues da Silva, Jacinto Paulo Simões Estima (Supervisors)

INESC-ID, Instituto Superior Técnico, Universidade de Lisboa, Portugal

DOI: 10.5281/zenodo.10003559

What is Maestro?
Maestro v2023 is a digital platform that gathers data from the web (social networks, web pages, APIs, etc.), transforms, extends, and classifies that data (usually through machine learning algorithms). The resulting dataset can afterwards be consulted in the Maestro platform and downloaded or sent to external services.
Maestro currently supports data types like images, sounds, and text. Both general text data types (i.e., plain text) and specialized text data types (e.g., news articles and scientific papers) are available.
Since Maestro has been designed with flexibility and extensibility, it allows for a wide array of use cases that involve data gathering, classifying, and performing other machine learning operations.

Goal: This user assessment session aims to evaluate Maestro v2023 by users with a "researcher" profile. This evaluation will be performed by developing a simple "Literature Review Automation" experiment. The gathered results will assess Maestro's usability level and determine its future enhancements. The experiment is described below.

More information: António Miguel Martins, Alberto Rodrigues da Silva, Jacinto Estima. Streamlining Literature Reviews Using an Automatic and Flexible Data Gathering and Classification Platform. 31st International Conference on Information Systems Development (ISD2023), AIS.
https://drive.google.com/file/d/1UtvS9VHY094_ilzJG0RnyBh8gI_4AgnE/view?usp=sharing

Conditions:
This user assessment session shall be conducted under the following conditions:

- All the tasks should be performed without further support or help.
- Use a computer with a reliable Internet connection and a modern browser (Google Chrome version 10+, Mozilla Firefox version 31+, Internet Explorer version 10+, Microsoft Edge version 18+, or Safari version 10+).
- The session is expected to last between 15 to 30 minutes.

Instructions:

1. Access https://maestroai.pt
2. Create a new user account and confirm your email.
3. Take a moment to explore Maestro's menus.
4. Create a new Search Context with a name of your choice.
5. Configure the Search Context to search for scientific papers related to automating literature reviews.
   5.1. Use the search string *"(automatic SOR semi-automatic) SAND literature review SAND (system SOR program)"* without the quotes ("").
   5.2. Select "Scientific Paper" from the "Data Type" field.
   5.3. Do NOT set it to iterate.
6. Advanced configurations:
   6.1. On the **fetch** tab:
   - Upload this zip file (https://drive.tecnico.ulisboa.pt/download/1414448696376725) as the initial datastream.
   - Set the maximum results to **5**.
   - Set the fetchers to "ArXiv Fetcher" and "Google Scholar Fetcher".
   - Make sure the "yield after gathering data" option is selected.
   - Do not forget to save these settings by pressing the save button.

   6.2. On the **classify** tab:
   - Select the classifiers "Keyword Extractor" and "Paper Summarizer".
   - Do not forget to save these settings.

   6.3. On the **filter** tab:
   - Make sure the "Duplicate filter" option is selected.

   6.4. On the **analyze** tab:
   - Select the analyzers "Source Analyzer" and "Citations Analyzer".
   - Do not forget to save these settings.

   6.5. Leave all other advanced configurations as default.
7. Review the configurations.
8. Start the Search Context execution by pressing the "Start" button.
   8.1. You can monitor its execution by pressing the "Monitor" button.
9. When the execution reaches the "Review" stage:
   9.1. Review the collected papers: if you think a paper seems unrelated to the search string, unselect it and click the "Save Changes" button. This option will remove it from the list of collected objects.
   9.2. Complete the review by clicking the "Complete review" button when you are done.
10. Monitor the progress and wait until all stages are finished.
    - **Note**: The "Classification" stage may take some time to complete. We recommend you leave it running in the background during this period. Furthermore, some data objects might not contain a result for a given plugin. This situation is expected and should only be concerning if the "Classification" stage fails.
11. When finished, check the classification results by clicking on the "Results" button on the Search Context details menu.
12. Download the results; you will be asked to submit this generated file later in the feedback form. This should include the data, metadata, results, and docx file.
13. Return to the Search Context details menu and press the "Analysis" button to view the generated charts using the selected plugins.
14. Congratulations! You have succeeded in creating and executing your first Maestro Search Context!

Please fill out the form below:

https://forms.gle/cNwPD7k3Mf65SpsT9

Thank you for your participation!

Appendix C – User Assessment Questionnaire

Survey on the Usability of "Maestro v2023"

Maestro is a versatile web application designed to gather, transform, and classify data from various online sources, including social networks, web pages, and APIs. Maestro supports a wide range of data types, including images, sounds, text, and scientific publications, making it suitable for diverse applications involving data processing and AI operations.

Developed by Alexandre Serra (v2022) and António Martins (v2023) as part of their Master's thesis at Instituto Superior Técnico, Universidade de Lisboa, supervised by Alberto Rodrigues da Silva and Jacinto Estima, Maestro aims to streamline data-related tasks efficiently.

To enhance your user experience, we kindly request your feedback through a short questionnaire.
Your insights are invaluable in helping us improve Maestro further.

Please take a moment to explore it based on the Assessment Guide available at https://doi.org/10.5281/zenodo.8407807. and then complete the questionnaire available below.

If you have any questions or additional feedback, don't hesitate to reach out to us at antoniomartins@protonmail.com.

Thank you for exploring Maestro v2023. We greatly appreciate your time and input!

For more information, you may read these recent papers:

António Miguel Martins, Alberto Rodrigues da Silva, Jacinto Estima. Streamlining Literature Reviews Using an Automatic and Flexible Data Gathering and Classification Platform, 31st International Conference on Information Systems Development (ISD2023), AIS Digital Library, 2023.

Alexandre Serra, Jacinto Estima, Alberto Rodrigues da Silva. Evaluation of Maestro, an extensible general-purpose data gathering and data classification platform. Information Processing & Management 60 (5), 103458, 2023.

* Indicates required question

Personal profile

1. Age

Mark only one oval.

- ◯ < 18
- ◯ 18-25
- ◯ 26-59
- ◯ > 60

2. Which of the following most accurately describes you?

Mark only one oval.

- ◯ Female
- ◯ Male
- ◯ Non-binary
- ◯ Transgender
- ◯ Intersex
- ◯ Prefer not to say
- ◯ Other: _____

3. What is your current nationality? (if you have more than one, please select the one in which you most identify)

Check all that apply.

- ☐ Portuguese
- ☐ Other: _____

4. What is your highest academic level?

Mark only one oval.

- ◯ PhD
- ◯ Master's Degree
- ◯ Bachelor's Degree
- ◯ High School
- ◯ Primary School
- ◯ Other: _____

5. Have you got a degree or professional experience in any field related to Computer Science (or equivalent)? *

Mark only one oval.

- ◯ Yes
- ◯ No

6. How would you rate your experience and comfort with computers? *

Mark only one oval.

	1	2	3	4	5	
Not	◯	◯	◯	◯	◯	Very comfortable

Experience with Maestro

7. Please provide the downloaded results from the test session

Files submitted:

8. How do you rate the easiness of performing these actions? *

Mark only one oval per row.

	1 (Very hard)	2	3	4	5 (Very easy)
Create user account	◯	◯	◯	◯	◯
Create Search Context	◯	◯	◯	◯	◯
Configure Search Context	◯	◯	◯	◯	◯
Start execution of Search Context	◯	◯	◯	◯	◯
Monitor the progress of the execution	◯	◯	◯	◯	◯
Review Data	◯	◯	◯	◯	◯
View Results	◯	◯	◯	◯	◯
Download Results	◯	◯	◯	◯	◯

9. Did you find the configuring stage challenging? *

Mark only one oval.

- Yes
- No
- A bit

10. If you answered "Yes" or "A bit" in the previous question, please tell us what you found challenging.

11. How would you rate Maestro in the following categories? *

Mark only one oval per row.

	1 (Very poor)	2	3	4	5 (Very Good)
Usability (ease of use)					
Flexibility (ability to extend to multiple use cases)					
Performance (how fast the interactions are)					
Usefulness (real world applications)					

12. Provide any comment or suggestion regarding the previous question on the Maestro capabilities.

13. Do you see situations where you could use Maestro in your day-to-day life? *

Mark only one oval.

- Yes
- No
- Maybe

14. If you answered "Yes" or "Maybe" in the previous question, please provide specific situations.

15. How would you rate your overall experience with Maestro?

Mark only one oval.

	1	2	3	4	5	
Very						Very positive

16. Provide any comment or suggestion regarding the previous question on the overall experience.

17. Provide any comments, suggestions, bug reports, or others here.