information

Article

Streamlining Tax and Administrative Document Management with AI-Powered Intelligent Document Management System

Giovanna Di Marzo Serugendo *, Maria Assunta Cappelli, Gilles Falquet, Claudine Métral, Assane Wade, Sami Ghadfi, Anne-Françoise Cutting-Decelle, Ashley Caselli and Graham Cutting

Centre Universitaire d'Informatique (CUI), Université de Genève, 1205 Geneva, Switzerland; maria.cappelli@unige.ch (M.A.C.); gilles.falquet@unige.ch (G.F.); claudine.metral@unige.ch (C.M.); assane.wade@unige.ch (A.W.); ahmed.ghadfi@unige.ch (S.G.); anne-francoise.cutting-decelle@unige.ch (A.-F.C.-D.); ashley.caselli@unige.ch (A.C.); graham.ashley.cutting@gmail.com (G.C.)
* giovanna.dimarzo@unige.ch

Abstract: Organisations heavily dependent on paper documents still spend a significant amount of time managing a large volume of documents. An intelligent document management system (DMS) is presented to automate the processing of tax and administrative documents. The proposed system fills a gap in the landscape of practical tools in the field of DMS and advances the state of the art. This system represents a complex process of integrated AI-powered technologies that creates an ontology, extracts information from documents, defines profiles, maps the extracted data in RDF format, and applies inference through a reasoning engine. The DMS was designed to help all those companies that manage their clients' tax and administrative documents daily. Automation speeds up the management process so that companies can focus more on value-added services. The system was tested in a case study that focused on the preparation of tax returns. The results demonstrated the efficacy of the system in providing document management service.

Keywords: AI; administrative documents; data mapping to RDF; document classification; document management system; information extraction; knowledge engineering; machine learning approach; reasoning engine; tax documents

check for updates

Citation: Di Marzo Serugendo, G.; Cappelli, M.A.; Falquet, G.; Métral, C.; Wade, A.; Ghadfi, S.; Cutting-Decelle, A.F.; Caselli, A.; Cutting, G. Streamlining Tax and Administrative Document Management with AI-Powered Intelligent Document Management System. *Information* **2024**, *15*, 461. https://doi.org/10.3390/info15080461

Academic Editors: Polona Tominc and Maja Rožman

Received: 4 June 2024
Revised: 10 July 2024
Accepted: 15 July 2024
Published: 2 August 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Document management processes play a key role in the services that some companies offer to their customers, and companies are showing great interest in automating these document management processes. Particularly for companies operating in the document-intensive sector, such as insurance, medical, legal, and banking and finance, automation becomes a solution for managing large volumes of paper. Filling out invoices, drafting contracts, setting up insurance policies, etc., are burdensome and time-consuming practices that diminish the time spent on other services such as advising one's customers. The solution to this waste of time is the implementation of automated DMSs that store large amounts of textual information, enriched with metadata to support searchability [1]. The use of the DMSs leads to a reduction in the amount of paperwork to be managed, but above all, to a reduction in the time spent studying the documentation. The automation of work reduces the time needed for companies to access and review key documents. Although these automated DMSs have brought many benefits in time and cost, they still present challenges that need to be addressed. For instance, some proposed DMSs present a burdensome implementation process that requires the use of complex software or specific technical capabilities, or they are not flexible as they cannot be customised for specific professions in the sector. Moreover, the management of personal data remains an unsupported activity, and most solutions are targeted at companies rather than individual consumers. As a result, there is currently no "top-of-mind" solution for automated access

to key documents from individual consumers. There is also a lack of DMS solutions that can classify and understand customer profiles and documents to make better and faster decisions.

This study focuses on the fiduciary and insurance sector and proposes an intelligent DMS. We built this DMS through the combination of several specific modules. These modules encompass ontology, information extraction, and semantic reasoning, each of which was adopted using specific techniques. The combination of some techniques makes it possible to create a system that is autonomously able to analyse and understand the content of documents, extract meaningful knowledge from documents, and perform reasoning based on semantic rules. Furthermore, semantic reasoning also allows the DMS system to comply with tax regulations. In other words, the reasoning developed by the DMS will be in accordance with the rules in force in the Canton of Geneva for the filing of tax returns that are in the "Canton of Geneva's Tax Guide 2020". Thus, the DMS allows for the automation of compliance with these rules. Although our case study focuses on the Geneva Tax Guide, the same approach can be applied to other domains requiring compliance to regulations, as well as to other languages.

The proposed system fills a gap in the landscape of practical tools in the field of DMSs and advances the state of the art. The system represents a complex process of integrated AI-powered technologies with the aim of creating an ontological structure, extracting information from documents, defining profiles, mapping the extracted data in RDF format, and applying inference through a reasoning engine. The combination of already established techniques in the fields of NLP, knowledge engineering, and semantic data modelling allows our system to *(i)* be more efficient and accurate in the process of extracting information from documents; *(ii)* automatically define document schema and control profiles; *(iii)* use data mapping in RDF format, which enables a semantic representation of data, facilitating the integration and analysis of data from different sources; and *(iv)* dynamically adapt schema and profiles to the specific requirements of different contexts, representing an advance in the adaptability and flexibility of the system. Thus, the innovation proposed in this paper is the creation of an information service that allows for more efficient management of DMSs through the automation and integration of advanced technologies. This information service offers users a better experience and greater efficiency in the processing and organisation of fiscal and administrative documents. While this paper focuses on administrative tax related documents, the same approach can be applied to different domains. Indeed, the core aspect of our proposal lies in (1) the extraction and formalisation of regulations, (2) the development of an ontology of the domain, (3) the insertion of the data into a knowledge graph, and (4) the assessment of compliance of the data to the regulations. Our approach is then suitable to any use case in any domain related to compliance to regulations. As a matter of fact, we adopted it to various other projects related to underground data (e.g., underground water or gas pipes) verification and compliance to underground regulations [2], as well as automated vehicle safety regulations [3].

The paper is structured as follows: Section 2 provides an overview of the literature review. Section 3 discusses the materials and methodology, which includes the general overview of the global workflow and the architecture of the proposed DMS (Section 3.1), a detailed description and explanation of each module of the DMS (Section 3.2), and a discussion of the technological requirements and the technology used to develop the proposed solution (Section 3.3). Section 4 presents the results obtained from each module of our DMS as well as the results obtained from the proactive application of our system to three case studies (Sections 4.2.1–4.2.3). Finally, Section 5 concludes the paper by discussing the results, limitations of the approach, and future works.

2. Literature Review

The current literature explores various techniques for implementing DMSs, including the use of ontologies (for general details, see [4–6]), semantics, or information extrac-

tion [7–9]). This section includes a general review on Intelligent DMSs as well as a focus on a DMS for the financial sector presented by Bratarchuk and Milkina [10].

Some recent advances in AI have enabled the development of intelligent DMSs that aim to deliver benefits on a larger scale. Sambetbayeva et al. [11] proposed an intelligent DMS based on ML and multi-agent modelling of information retrieval processes to improve the efficiency and accuracy of document management. The system uses lexical agents and cognitive–linguistic agents to extract information from texts in the form of ontology-based structures, such as facts, objects, and relations. The authors argued that the use of a multi-agent approach is a viable alternative to text analysis systems with sequential architectures. This study used AI methods to enhance the capabilities of traditional DMSs, further improving efficiency, productivity, and customer satisfaction. The presented system is analogous to our system in that both focus on extracting information, through predictive models. The difference lies in the absence of ontology and semantic rules that establish regulatory compliance with tax rules.

Justina et al. [12] presented a cloud-based electronic DMS (SECEDOMAS) to address the inefficiencies and problems associated with manual document management. The system focuses on storing, retrieving and sharing documents and it uses the Advanced Encryption Standard (AES) algorithm for document security. The system was tested using the document of exams and records unit of Olusegun Agagu University of Science and Technology, Okitipupa. The system focuses on general document management and data security, unlike our DMS, which emphasises advanced analysis and tax regulation adaptability.

Ustenko et al. [13] presented Amazon Kendra to improve the work of electronic document flow in the banking sector. Amazon Kendra uses ML and natural language processing to understand the content of documents and provide accurate responses to user queries. The system combines Amazon Kendra as the search interface and the underlying DMS, which includes the creation of an up-to-date index of various types of documents from different repositories. Our work does not include the development of the interface but focuses on the design and implementation of the DMS.

Martiri et al. [14] introduced blockchain to a DMS by developing the DMS-XT system, which aims to automate the storage and retrieval of documents using AI algorithms. The DMS-XT system was used to simplify and coordinate the electronic management of students' diplomas to improve the quality of their content and to detect any plagiarism of previously stored theses. In other words, the system acts as an intermediary between students and lecturers, as students have access to the topics proposed by lecturers and can choose their own research topic. Unlike this approach, our study focuses on other dimensions of document management that do not prioritise security. However, citation of this work contributes to a more comprehensive understanding of the landscape of document management solutions, as it presents an innovative DMS that employs a combination of different techniques, including blockchain, to address security requirements.

Sladic et al. [15] proposed a DMS based on a two-layer document model. The first layer is generic and contains the characteristics common to all documents and has been designed following ISO 82045 standards [16]. The second layer is specific and contains the features that are specific to a specific domain. This model is represented as an OWL DL ontology. In particular, the layers of the model are developed as separate ontologies. In this way, the ontology acts as an intermediary between external document search queries and the set of archived documents. The adoption of this model allows the DMS to adapt to different specific domains. In fact, since the two levels are interdependent, the DMS can offer basic services common to all documents (such as the search function), and at the same time, it can also offer domain-specific services (by providing specific functions for the management of documents in the health sector). This research, although sharing some elements with our study, differs in function and main objective. In the cited case, the use of ontology and the semantic analysis of documents is functional for understanding the meaning of documents and facilitating interoperability between different systems through data semantics. In contrast, in our context, the semantic approach is mainly used for analysing and interpreting

documents through semantic reasoning. The presented system is based primarily on the ability to classify documents according to their content and retrieve relevant information using semantic techniques. In contrast, the objective is to develop an intelligent document management system that not only classifies and retrieves documents but also perceives the content of documents through advanced semantic analysis techniques.

Errico et al. [17] created a DMS using both a semantic classifier and a semantic search engine. First, paper documents are converted to editable and searchable digital texts. Then, they are analysed and assigned to appropriate categories and tags by means of the semantic classifier, which allows the content to be understood. At the end of this semantic analysis and classification, the documents are stored in the database in an appropriate manner. Information retrieval is performed via a semantic search engine that analyses user queries and searches for the most relevant documents. The semantic classification process is based on the use of ML algorithm called Latent Dirichlet Allocation (LDA).

With regard to the DMS in the tax domain, we cite the work by Bratarchuk and Milkina [10], discussing the importance of electronic document management systems (EDMS) within tax authorities. The text proposes implementing a modern domestic system based on the EDMS Logic platform, which is a software platform designed to manage the lifecycle of electronic documents, from creation to preservation, including aspects such as editing, digital signature, distribution, and storage. EDMS Logic is implemented on IBM Lotus Notes/Domino and is capable of leveraging the functionalities and features offered by IBM Lotus Notes/Domino. This includes tools and technologies that facilitate process automation and document management in compliance with tax regulations. Regulatory compliance is an aspect of our system that we pursue through technology and AI, through which we are able to evaluate regulatory violations in a more sophisticated manner compared to a standard document management platform like "EDMS Logic".

Although the examined DMSs offer a range of valuable functionalities, including document classification, semantic search, and security, they do not prioritise regulatory compliance with tax rules. Furthermore, these DMSs would not be adaptable to the tax sector, as they may need to be adapted to meet the specific requirements of the legal requirements of this sector. This could include semantic rules for verifying compliance and establishing relationships between profiles and documents for regulatory purposes. Therefore, further research and development may be needed to adapt these systems to effectively address tax sector requirements. Furthermore, our DMS is complex and needs the integration of multiple modules that require sophisticated functionality.

3. Materials and Methods

The intelligent DMS proposed in this study is the result of a research approach that involves various techniques implemented in five modules (see Table 1). We focused on the practical application and effective integration of existing solutions to address the specific needs of fiscal and administrative document management.

We performed a preliminary activity before developing the five modules that included the definition of key concepts for this project such as *classification*, *information extraction*, *user profile*, and *tagging*, as well as the definition of the basic terminology used in the system. For example, the "classification" term allows us to assign a label to an object. For the purposes of this research, classification assigned a class to the document. A hospital invoice was classified as an insurance document. The term "information extraction" refers to a module that processes the document. This module generates JSON files for each document and identifies its class (e.g., health insurance policy) as well as specific information extracted from the document (e.g., date, amount). The concept of "household profile" includes an individual's economic and personal status, such as marital status, dependants, employment status, etc. A label can be defined as "a small ticket giving information about something to which it is attached or intended to be attached" (Definition of label noun from the Oxford Advanced Learner's Dictionary https://www.oxfordlearnersdictionaries.com/definition/english/label_1 (accessed on 14 July 2024)). Each document can therefore be labelled or

tagged (from one to several for each document), allowing for a classification other than the fiscal classification. "Tagging" documents provides additional context and categorisation beyond their fiscal classification.

Once the basic concepts and terminology were established, the development of intelligent DMS proceeded through the five mentioned planning modules that involved the implementation of various techniques. Table 1 summarises them and provides an explanation and expected outcome for each module.

Table 1. Development modules for the proposed intelligent DMS.

Name	Description	Outcome
Section 3.2.1	Development of an ontology for fiduciary, insurance and user profiles	→ Representation of concepts of the fiduciary and insurance domains → Representation of concepts related to tax profiles
Section 3.2.2	Classification of documents into their respective categories. Extraction of relevant information using appropriate template based on the document category	→ File containing extracted information
Section 3.2.3	Defining documents schema and tax profiles	→ JSON schema for document schema and tax profiles
Section 3.2.4	Map extracted information to RDF	→ Convert JSON file to RDF triples by leveraging the ontology vocabulary
Section 3.2.5	Definition of SHACL shapes and SHACL rules	→ Document classification and recognition → Rules for multi-label classification → User profile classification rules

These modules are presented without any specific sequential numbering because only some modules are sequential, such as "Document schema and profile definition" and "Data mapping to RDF". Similarly, the "Ontology development" and "Reasoning engine development" follow this pattern. Consequently, the development of the ontology and semantic rules occurred concurrently with the development of the module related to data extraction and mapping into JSON files and RDF data. Once all modules were developed, the integration of RDF data into the ontology and semantic rules system was initiated.

The "Ontology development" module concerns the development of an **ontology** (defined for French terms) that includes concepts for representing fiscal and administrative documents as well as their keywords, user profiles, and changes to profiles (Section 3.2.1).

The use of ontologies in a DM offers considerable advantages, as demonstrated by various research studies in the field. Fuertes et al. [18] developed an ontology specifically for the construction sector, which allows documents to be classified along the project lifecycle. In this context, their ontology facilitates the navigation and organisation of documents within the specific context of construction activities. Doc2KG proposes a framework for transforming data into knowledge graphs, facilitating document management. The ontologies enable the representation of documents in a more detailed and related manner, thereby facilitating access to and comprehension of information. Furthermore, Sladić et al. [15] demonstrated that an ontology-based document model improves the efficiency of document management by enabling a better understanding of document content and facilitating more accurate analysis, classification, and retrieval.

The "Document classification and information extraction" module uses **information extraction** techniques for classifying documents into their respective categories; selecting the appropriate template for information extraction based on the document category; and extracting relevant information using selected templates (Section 3.2.2). The expected result of this module is to provide files containing extracted information including the categories assigned to the documents and their keywords, and the profiles retrieved from the documents.

The decision to employ technical information extraction is justified by the information and extraction technique's fundamental role in document analysis, classification, and organisation. This enables a more comprehensive understanding of the content of documents and facilitates more effective decision-making, as shown by a number of studies in the literature. Information extraction enables the combination of textual and visual data, as highlighted by the studies of Augereau et al. [7] and Ferrando et al. [19]. These studies succeed in capturing a wider range of information from documents, enhancing the understanding of the content, and enabling more accurate classification. Furthermore, information extraction automates the process of document analysis and classification, reducing the time and resources required for manual analysis of large volumes of documents [8]. In addition, information extraction facilitates the organised retrieval of information from documents. According to the approach proposed by Eswaraiah et al. [9], the use of optimised ontologies allows for the coherent organisation and retrieval of information through queries, thus enhancing the efficiency of DMS.

The "Document schema and profile" module focuses on the **generation of JSON schema files based on the extracted information.** These files contain information relevant to the documents' structure and the tax profile definitions (Section 3.2.3).

The "Data mapping to the RDF" module focuses on the **mapping** of the extracted information, represented as a JSON file, to the RDF triples (Section 3.2.4).

The "Reasoning engine" module develop **engine**, which leverages SHACL shapes and rules to *(i)* classify and recognise documents for tax returns; *(ii)* infer the labels to assign to the tax documents; *(iii)* classify users into different user profiles (Section 3.2.5).

SHACL is a language for describing constraints and validation rules for RDF graphs. SHACL is supported by TopBraid, developed by TopQuadrant, Inc., Raleigh, NC, USA. TopBraid Composer Maestro Edition version 7.1.1 was used for this project. We opted to use SHACL for several fundamental reasons, as described by Knublauch, and Kontokostas [20]. The SHACL language allows us to define constraints and conditions on RDF graphs in a precise and detailed manner, offering a robust mechanism for validating data against specific requirements. Furthermore, SHACL introduces the concepts of "shape graphs" and "data graphs", providing a clear and organised structure to define constraints and validate the RDF data. This distinction enables more effective management of constraints and validated data, facilitating data integration in various application contexts. Another key advantage of SHACL is its flexibility of use. In addition to validation, SHACL shape graphs can be used for a variety of purposes, such as creating user interfaces and code generation. Furthermore, the fact that the SHACL language is normatively defined provides clear and specific rules for defining constraints and validating RDF data, ensuring consistency and reliability in the data management process. Lastly, SHACL is compatible and integrable with a wide range of tools and technologies that support the RDF standard, offering an interoperable and scalable solution for data management.

3.1. Global Workflow and Architecture

We developed the proposed intelligent DMS using a methodology consisting of five modules (see Table 1), which we implemented according to the workflow shown in Figure 1.

The workflow illustrated in Figure 1 should be read from right to left, and it focuses on the following main activities:

- Within the component titled *Doc. Categorisation/Information Extraction*, documents (native PDFs or scanned documents) are processed by a Section 3.2.2. The module *(i)* classifies the type of document and *(ii)* extracts the information.
- Continuing from right to left, the two boxes labelled *JSON File (doc)* and *JSON File (Profiles)* demonstrate the generation of JSON files for each document with extracted information regarding document keywords and profiles (Section 3.2.3).
- Arrows labelled *RDF mapping* indicate that the JSON files are processed through *RDF mapping* and stored into the RDF data triple store (Section 3.2.4).
- Arrows connecting the RDF triple store component with the *Ontology* and *Rules* components indicate that the RDF data are integrated into the ontology and semantic rules.
- The *Ontology* component covers the domain of fiscal and administrative document management, describing fiscal budget, individuals, documents, profiles, and tax categories. The ontology is constructed from Swiss tax return data based on actual legal documents and required documents for tax return completion, as shown through the arrow connecting the tax declaration and legal documents to the *Ontology* component (Section 3.2.1).
- The *Rules* component includes a series of rules, including those related to document validation, profile updates, the identification of missing documents, and document labelling. The rules are derived from the documents provided by Addmin concerning the legal tax rules in the Canton of Geneva and Switzerland, as indicated by the arrow pointing left with the label *Rules modelling*. These rules are applied by the reasoning engine to RDF data (Section 3.2.5).
- The reasoner updates JSON profiles based on new information (e.g., a new child added to the household) and identifies missing documents based on existing profiles (e.g., missing health fees for a household member), as shown by the arrows labelled *update* and *identifies* at the bottom right of the figure.

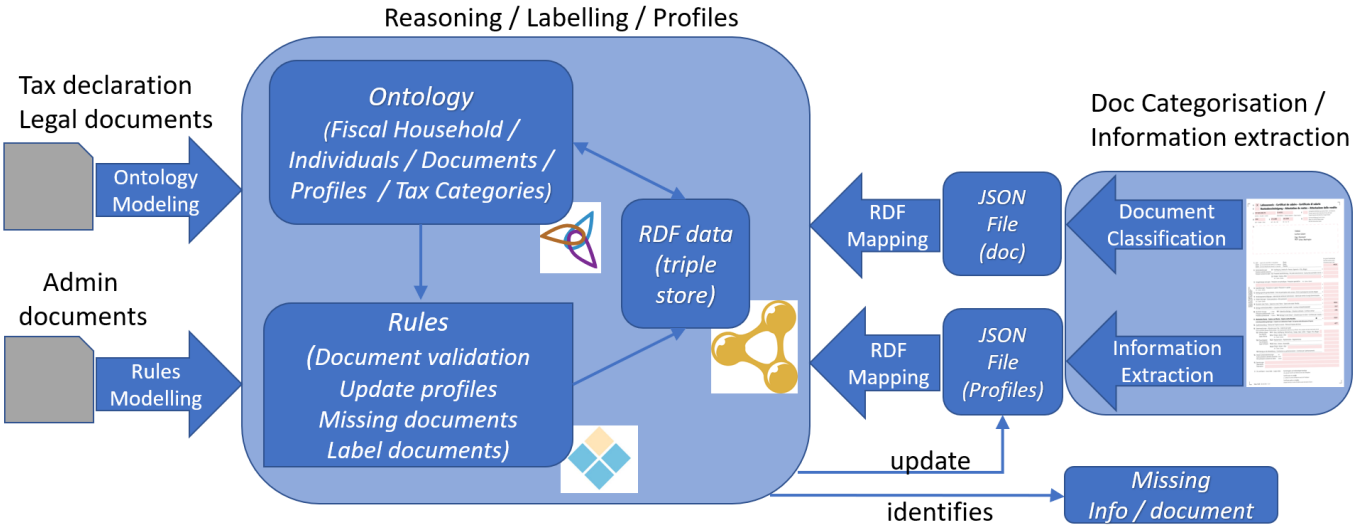


Figure 1. Global workflow of the DMS architecture.

3.2. Detailed Insight into DMS Architecture Modules

This section is dedicated to analysing the five modules that were used to design and implement an intelligent DMS, as described in Table 1.

3.2.1. Ontology Development

Ontologies provide a formal, explicit specification of a shared conceptualisation of a domain, providing a common vocabulary and definitions for the domain's concepts and relationships. In the context of financial and administrative documents and user profiles, an

ontology can ensure consistency in the representation and processing of these entities across different systems and applications. This ontology, built using the Canton of Geneva Tax Guide 2020 as a documentation source, describes financial and administrative documents, user profiles, and changes to profiles.

The ontology development process was guided by the phases proposed by Noy and McGuinness [6] to describe the document management domain, focusing on tax and administrative returns. A compendium of the steps in this process is described below.

- We defined the domain and purpose of the ontology, which allowed us to outline the context in which the ontology will be used and the main functions it must perform (the management of fiscal and administrative documents in our case). We clarified the main objectives of the ontology as well as its content (including document classification, user profile definition, and organisation of information regarding fiscal items and changes in marital status or domicile).
- Subsequently, we conducted an in-depth analysis of requirements and available data sources. We examined tax declaration forms, instructions for their completion issued by competent authorities, and other related documents, such as payroll statements and health insurance communications.
- We then established the classes and class hierarchy, which consisted of defining the main classes of the ontology including documents, user profiles, fiscal elements, and changes in marital status or domicile.
- We also defined class properties, which represent the attributes or characteristics of documents and user profiles. We defined keywords or attributes that documents can have in relation to the data we want to extract from them. These data are useful for classifying or labelling the document or for creating a profile of a person or a family. For example, in the case of tax returns, by defining attributes such as total income, deductions, and marital status, documents can be classified according to these attributes. The definition of attributes or keywords facilitates the structuring and organisation of data.
- Once the conceptual design of the ontology was completed, we proceeded with the technical implementation of the ontology using the Protege version Protégé-owl 5.5.0-beta-9 software.
- The ontology was validated by tax experts to ensure its validity and consistency over time. Corrections or adjustments were made based on the feedback received during this phase. In particular, the final validation phase was preceded by interim monitoring phases of the documentation used to define the classes and explain any doubts about the interpretation of the legal requirements. This process ensures that all the documents used in the system are adequate and comply with regulatory requirements. During the documentation monitoring phase, the team carefully reviewed all the documents used to define classes in the system. In addition, explanations were provided for any doubts regarding the interpretation of legal requirements that could affect the preparation of tax documents. Finally, the final validation was carried out by the project partner together with experts in the field of tax return preparation.

We followed the middle-out approach, which allowed us to gradually develop the ontology, starting from a limited set of data and expanding it as we gained more knowledge of the domain. This reduces the risk of instability and inconsistencies and allows one to quickly gather user feedback during incremental development [21,22].

Figure 2 shows a more general view of the hierarchy of classes, properties, and axioms that constitute our ontology.

`amount` some (`HealthInsurancePremium` or `ExpensesSicknessNotReimbursed`)

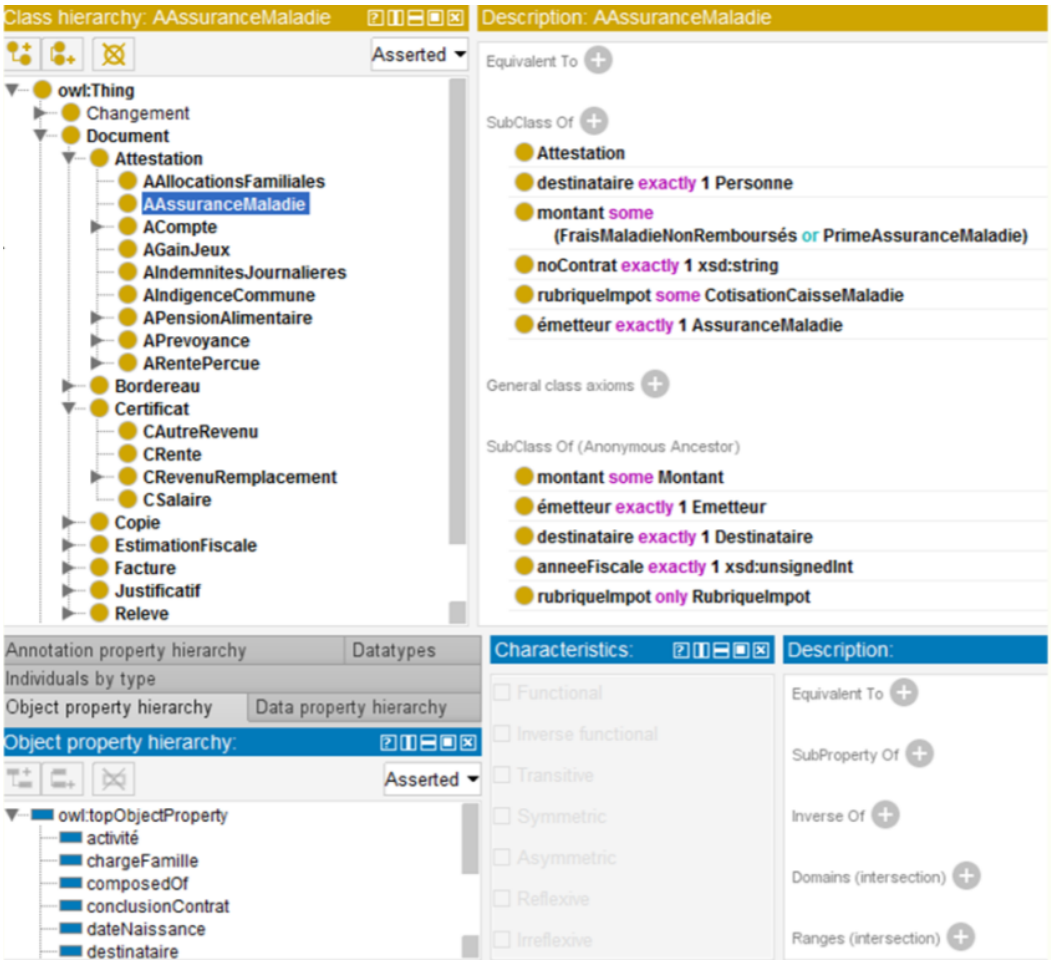


Figure 2. Ontology overview.

3.2.2. Document Classification and Information Extraction Module

Document classification and information extraction are two important tasks in DMSs. Document classification is the process of assigning predefined categories or classes based on the content of each document on which information extraction will be performed.

In our ML-based approach, we used *(i)* convolutional neural networks (CNNs), *(ii)* document type-specific templates for information extraction, and *(iii)* optical character recognition (OCR) to convert document images into digital text. CNNs are trained to automatically recognise and classify the significant features of documents, such as layout, structure, and distinctive content of each type of document. Once a document is correctly classified, the process of extracting relevant information begins. Key information is extracted from the document using OCR techniques to convert printed or scanned text to machine-readable form. OCR is performed using an open source tool such as Tesseract.

Figure 3 shows an overview of the workflow that combines classification (with CNNs) and information extraction.

The first line in Figure 3 shows that the first step is to insert the files containing the documents to be processed. The documents are then classified into specific categories. Once a document is classified into its correct category, the system selects the appropriate templates to extract the information (see Figure 4). The extracted data correspond to the value of the field type. The bottom line shows a concrete example of this extraction, where the document is classified as a *Salary Certificate*. The extracted data correspond to surname, first name, date of birth, and so on.

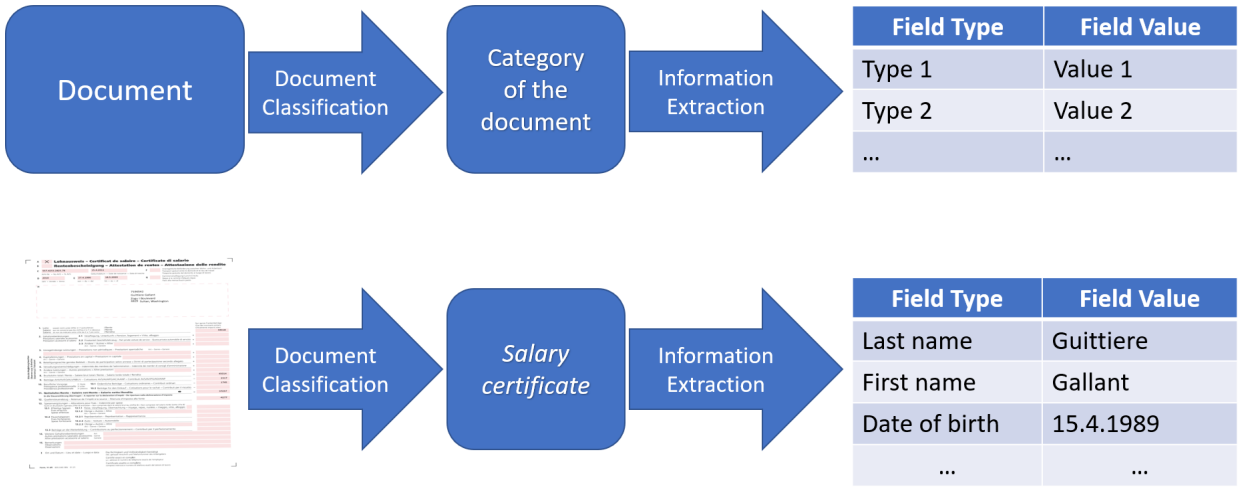


Figure 3. Workflow—document classification and information extraction.

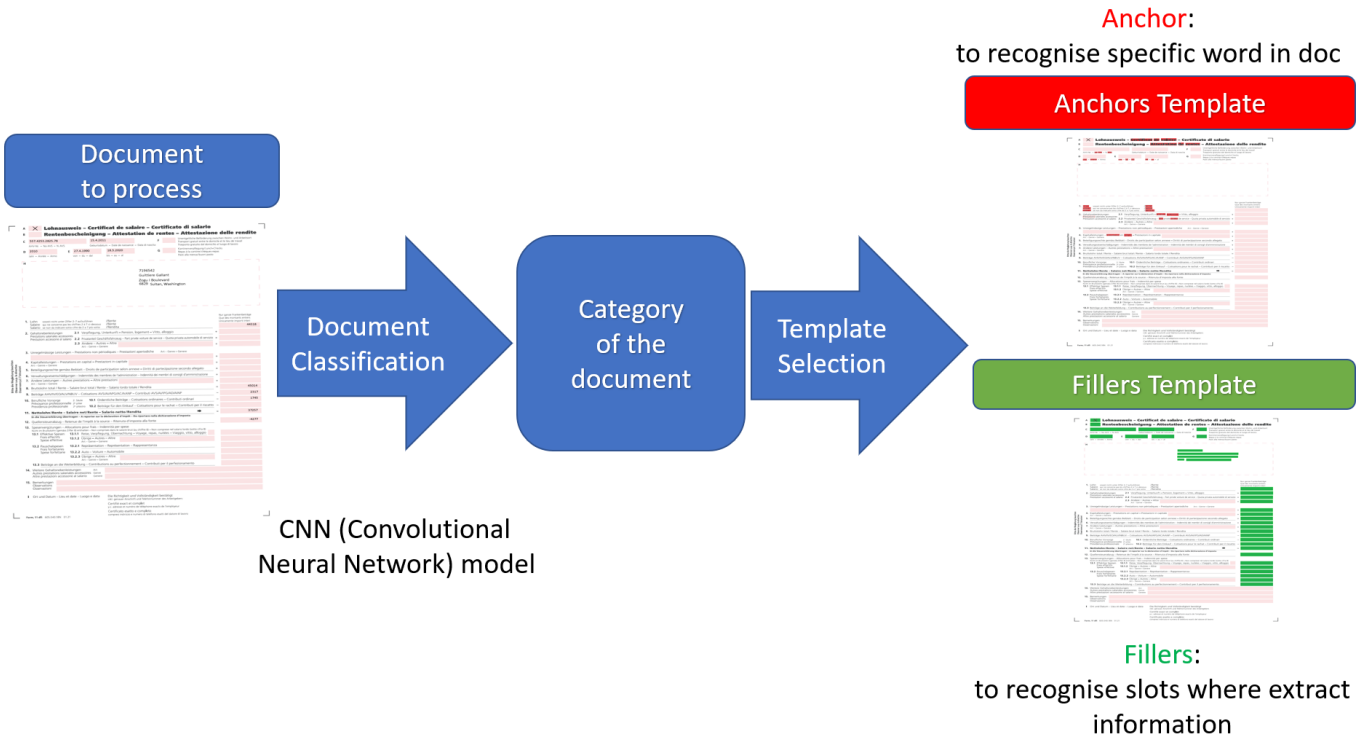


Figure 4. Information extraction.

The extracted information is then converted into the JSON format. This process is represented by the document schema and profile generation module (Section 3.2.3).

Concerning the classification task, we used *(i)* CNNs because they enable achieving accurate results for the analysis and recognition of documents. CNNs can extract specific features from text or images of documents, and both can be used to categorise documents or improve text quality. Regarding information extraction, we defined *(ii)* the template to automatically extract information from documents of that type. For each document type, we used specific models called "Anchors" and "Fillers". These templates provide pixel values to locate and extract the necessary information from the documents by identifying the type of field ("Anchors") and the value of the field ("Fillers"), as shown in Figure 4.

To train and evaluate AI models to accurately categorise tax documents, we developed a specific dataset. This complex task requires a detailed understanding of the different

types of tax documents and the ability to accurately distinguish between them. The dataset was organised into 11 main categories that represented the different types of tax documents, including receipts, invoices, tax returns, and so on such as *(i)* insurance benefit statement; *(ii)* insurance premiums; *(iii)* third-pillar contribution declaration; *(iv)* AVS (age and survivor insurance) annuity declaration; *(v)* salary statement; *(vi)* a combination of categories related to salary statement and AVS annuity declaration; *(vii)* tax declaration form; *(viii)* bank closing document; *(ix)* health insurance; *(x)* second-pillar pension document; and *(xi)* "Other Classes". This last category included 21 subcategories that were documents that did not fit into the other ten main categories. This ensured a broad and comprehensive coverage of the possible types of document that may be encountered in the real world. All documents were converted to images and resized to a uniform size of 56 × 56 pixels to facilitate analysis. This step not only simplifies data processing but also reduces the computational cost in both training the models and their practical application. Furthermore, considering that some documents may have multiple pages, it was decided to use only the first page of each document for classification. This simplification helps to keep the process efficient without compromising the quality of the classification.

To ensure a sufficient volume of training data and to cover a wide range of possible cases, summary documents were also generated. These summary documents mimic real tax forms, with fields such as first name, last name, and AVS number automatically populated. The generation process requires seed input to ensure the reproducibility of the results. Finally, documents unrelated to the Swiss tax return were included in the dataset to further improve the ability of the models to distinguish relevant documents from the mass of irrelevant documents. These included student records from the University of Geneva and documents from the US NIST tax dataset. This additional diversity helped the model to better identify and distinguish documents that were relevant to the Swiss tax return.

The dataset was divided for the training and evaluation of AI models. In total, it contained 12,090 documents, of which 9067 were relevant to the Swiss tax return, while 3023 were considered irrelevant. The dataset was split as follows: Overall, 75% of the data, i.e., 9067 documents, were reserved for model training, while the remaining 25%, i.e., 3023 documents, were included in the test and evaluation set. The evaluation of the dataset included two basic aspects: the analysis and evaluation of the performance of the models on synthetic, i.e., artificially generated, documents and the analysis of the performance of the models on real documents in the dataset [21].

3.2.3. Document Schema and Profiles Generation

JSON files were used to represent the structured information extracted from the documents, and these files were processed by the "Data mapping to RDF" module (Section 3.2.4). To facilitate the mapping process, two types of JSON schema were developed: one for each type of document and one for household tax returns. A JSON schema provides a vocabulary for annotating and validating JSON documents. In other words, a JSON schema defines the attributes that each document of a given type has to be possessed in order to be deemed valid.

To ensure that the extracted information is consistent and homogeneously structured within the information model, a JSON schema is created for each document type. This means that each extracted document is checked against the corresponding JSON schema to verify that it conforms to the specified properties. The extracted information is then mapped to the information model using the properties outlined in the corresponding JSON schema.

The generation of JSON files for the extracted information has to meet two prerequisites: *(i)* The documents have to be classified, and *(ii)* profiles have to be defined.

Regarding *(i)* the first prerequisite, the categories used to label the documents are defined using an ontology, with 28 different document types currently recognised. The classification process currently assigns only one category to each document even if a document may belong to more than one category in practice. For example, a hospital

invoice may be classified as a medical expense document, a tax return document, or an insurance document.

Table 2 provides a list of five example documents, where each document corresponds to a number (*N*), a name (*Document*), a set of features (*Features*) used to create the profile, a classification (*Classification*) corresponding to the document's class, and a label (*Label*) corresponding to the list of labels.

Table 2. List of 5 documents with their labels.

N	DOCUMENT	FEATURES	CLASSIFICATION	TAG
1	Tax return and/or ID (taxpayer number and declaration code)	Tax household	Tax return previous year	Tax Family
2	Employee's salary statement	Employee	Salary statement	Income Tax
3	Third-pillar A certificate and/or Second-pillar buy-back	Third-pillar	Third-pillar contributions	Insurance Miscellaneous Expenses Tax
4	Bank accounts, shares, bonds, participation, cryptocurrencies, lottery winnings, etc.	Securities	Bank statements	Securities Finance Tax
5	Bank account maintenance fees	Stocks	Bank account vouchers	Stocks Financial Tax

In relation to *(ii)* the second prerequisite, the household profiles were established by creating a list of 138 profiles. Each profile provided a clear description of a taxpayer using commonly understood language. Some of these profiles were "P1: a single person who has no children but owns property" and "P28: a single person who has no children and owns no property".

These profiles were designed to be easily identifiable and understandable, facilitating the process of classifying taxpayers into different household profiles based on their characteristics. This approach allows for more efficient analysis of tax data, enabling policymakers to identify patterns and trends in taxpayer behaviour and make informed decisions accordingly. Table 3 shows a selection of these profiles, each with a unique identification number.

Table 3. List of profiles.

PROFILE	NAME
P1	Single with property
P2	Single with property with children
P3	Single with a property with children and dependants
P28	Cohabitants owning a property without children
P29	Cohabiting with dependants
P31	Cohabitation with a self-employed person
P32	Cohabitation without property
P33	Cohabiting with children without property
P34	Living together without own property with children and other dependants
P36	Living together without a property with dependants
P38	Cohabiting with no children and no property
P39	Divorced with property

Table 3. *Cont.*

PROFILE	NAME
P40	Divorced with property with children
P41	Divorced with property with children and dependants
P43	Divorced with property and dependants
P72	Married with property and children
P74	Married with property with children and dependants
P135	Unmarried without property with children
P136	Unmarried with children and dependants
P138	Unmarried without property and with dependants

3.2.4. Data Mapping to RDF

The mapping process reads the JSON files containing the extracted data as input and applies the mapping rules to transform the records into RDF triples. The mapping process uses *(i) mapping files* to define custom mapping rules, *(ii) queues* to integrate with other tasks, and *(iii)* a *mapping processor* to transform the input data into RDF triples using the ontology vocabulary.

The *mapping files* contain the custom mapping rules that define how to transform the input data into RDF triples using the ontology vocabulary. The mapping rules specify how to extract data from the input sources, how to transform them, and how to create RDF triples from them using the ontology vocabulary.

The *queues* are used for communication purposes with other tasks such as information extraction or profiling. They are divided into input and output queues. The input data are provided by the input queues, and the RDF triples generated by the mapping processor are published to the output queues. This allows for easy integration with other parts of the system.

Finally, the *mapping processor* uses the custom mapping rules defined in the mapping files to generate RDF triples from the input data. The mapping processor reads the input data from the input queues and writes the RDF triples to the output queues. It uses the queues to read and write the data and the mapping rules to transform the data into RDF triples.

An overview of such an architecture is shown in Figure 5.

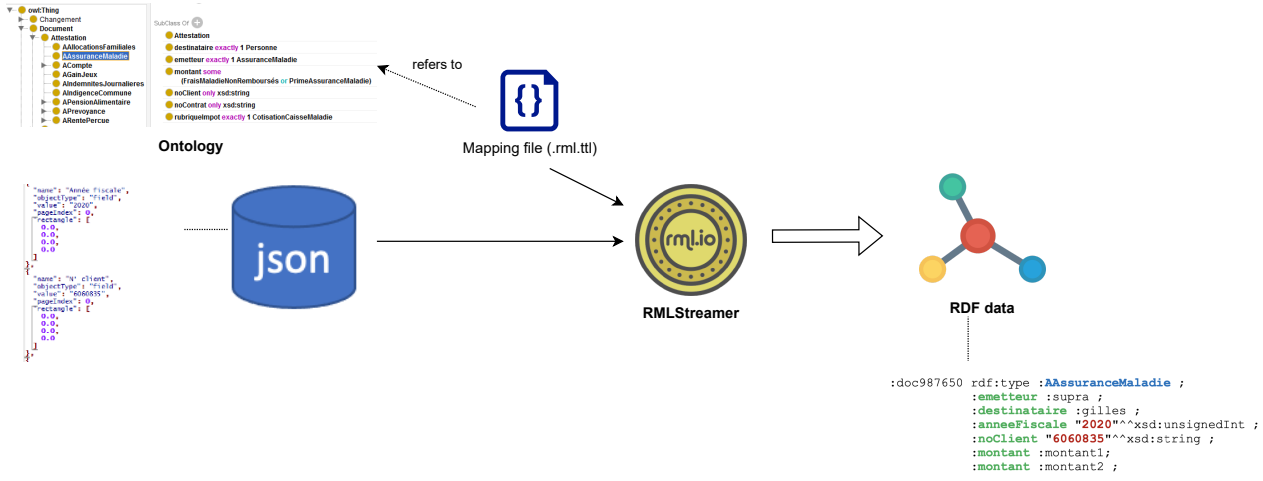


Figure 5. Architecture overview of the mapping process.

The two main steps in the implementation of the mapping process are *(i)* data preprocessing and the *(ii)* actual data mapping to RDF. During the preprocessing step, the input data are cleaned. The JSON files extracted during the information extraction task are

cleaned so that only the information that needs to be mapped to the RDF is retained. This involves removing extraneous fields to ensure that the data conform to the specific JSON schema. The purpose of this preprocessing step is to prepare the data so that they can be properly mapped to RDF. The *preprocessing* phase was carried out using an Apache Flink application deployed on a Flink cluster. The application read JSON data from a Kafka topic, performed cleaning operations by removing extraneous fields, and published the cleaned JSON data to Kafka topics, one for each document class, used for output purposes only.

The actual *data mapping* to RDF involves applying the RML mapping rules to transform the cleaned JSON data into RDF triples. The RML implementation used in this study was RMLStreamer, an RML processor built on Apache Flink and optimised for streaming and large datasets. RMLStreamer jobs were deployed for each document class, reading data from the corresponding output Kafka topic generated by the data preprocessing phase.

The RDF triples produced by each RMLStreamer job are written to a "output" Kafka topic. All of these "output" topics converge to a single `flinkOutput` topic from which the RDF triples can be consumed and loaded into a triple store. An overview of the steps involved in implementing the mapping process is shown in Figure 6.

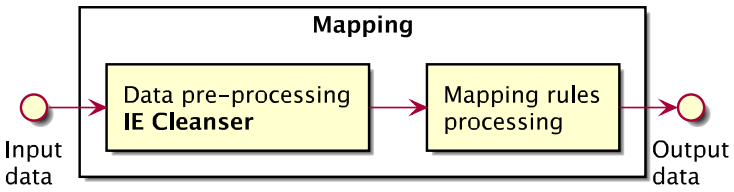


Figure 6. Overview of mapping steps.

3.2.5. Reasoning Engine Development

This section discusses the development of an inference engine for *(i)* classifying and recognising documents submitted by users, *(ii)* developing rules for the multi-label classification of documents, and *(iii)* developing rules for classifying users into different user profiles. The engine was implemented using SHACL, which is a language for describing constraints and validation rules for RDF graphs. SHACL is supported by TopBraid, developed by TopQuadrant, Inc. TopBraid Composer Maestro Edition version 7.1.1 was used for this project.

Concerning *(i)* classifying and recognising documents submitted by users, key elements that must be present for a document to qualify were identified and formalised as property SHACL shapes. using the properties that were used to define constraints on the values of nodes and edges within an RDF graph. For example, a valid salary certificate must contain certain information, including the salary amount and the first name and surname of the person who earned the salary. To ensure that all submitted salary certificates would meet these requirements, three SHACL properties (`impots:AmountShape`, `impots:PersonSurnameShape`, and `impots:PersonFirstNameShape`) were defined. For some of these properties, additional constraints describe the different values that can be assigned to each attribute shape; for example, the `impots:PersonSurnameShape` and `impots:PersonFirstNameShape` properties could specify that the values of the corresponding nodes must be of type `xsd:string` and must not be empty.

Regarding *(ii)* the developing rules for multi-label classification of documents, multiple labels were assigned to documents using SHACL rules. This enables automatic classification into multiple predefined categories. SHACL rules specify the conditions that an RDF graph must satisfy to be valid. To achieve this, a datatype property called `tag` was created to write triples and assign labels to documents. Thirteen different values were defined for the property to categorise the documents, as detailed in Table 4.

Table 4. Tags for multi-label classification.

TAG ID	TAG LABEL
1	Tax
2	Expenses/Other Costs
3	Pension
4	Children
5	Family
6	Finances
7	Income
8	Formation
9	Real Estate
10	Medical Expenses
11	Insurance
12	Job
13	Securities

Two types of rules were created for *(iii)* developing rules that classified users into different user profiles. The first type consists of customer profile rules, which infer user profiles by analysing the documents they have to provide. The documents provided by the users can be used as input to infer the user's profile. In other words, the documents provided by users can be analysed to infer their status or occupation, which constitutes their profile. These rules are called direct rules (where *Document* → *Profile*) because they proceed from the document provided by the user to their profile. So, if a user provides a salary certificate, their profile corresponds to the status of an employee. Table 5 describes some examples of how the documents provided by the users can be used to infer their profile (direct rules).

Table 5. The direct rules.

DOCUMENT → USER PROFILE		
Subject	*Predicate*	*Object*
User providing health insurance	is	Person
User providing salary statement	is	Employee
User delivering Training, Education, Retraining Document	is	Employee
User providing Third-pillar pension	is	Pensioner

The second type of rule consists of document delivery rules that infer which documents correspond to the user's status. These are called inverse rules (where *Profile* → *Document*) because they infer the document the user has provided from the user's profile. For example, if a person has the status of an employee, they provide their salary statement and proof of training and retraining. Table 6 describes some examples of how the user's profile can be used to determine which documents must be provided (inverse rules). These rules are described in natural language, which is then encoded using the SHACL language.

In total, 92 property forms and 3 sets of semantic rules were defined, giving a total of 120 rules. These rules consisted of 78 multi-label rules, 21 customer profile rules, and 21 document delivery rules. In addition, the reasoning engine step was the subject of a paper presented at the 29th International DMS Conference on Visualization and Visual Languages (DMSVIVA23) [23].

Table 6. The inverse rules.

USER PROFILE → DOCUMENT		
Subject	*Predicate*	*Object*
Person	delivers	Health insurance
Employee	delivers	Salary statement
Employee	delivers	Education, Training, Retraining Document
Retiree	delivers	Pension 3 Pillar

3.3. An Overview of Component Implementation

We implemented various components to design the intelligent DMS. The solution consisted of seven independent modules that interacted with each other. Each module was based on different technologies and had different architectural and technological requirements. They were designed to be independent and easily replaceable or removable if required. The following list describes each module in more detail:

- **Document classification and information extraction:** This module consists of sub-projects that perform *(i)* document classification on image documents, *(ii)* information extraction from image documents using templates, and *(iii)* generation of image documents for a document category/class (data augmentation).
- **Mapping rules:** This module defines custom mapping rules that are used to transform data from JSON to RDF. They are written using RML language and processed by an RMLStreamer processor. The module requires an Apache Kafka instance to run as the input data sources used by the mapping rules are Kafka topics. They are available at https://gitlab.unige.ch/addmin/rml-mappings (accessed on 14 July 2024).
- **Information and Extraction Merger:** This is a Python 3.9.1 module that performs a merge on the data derived from the Information and Extraction module. It is released as a Docker image. It requires a connection to a RabbitMQ and an Apache Kafka instance to work.
- **Information and Extraction Cleanser:** This is a Flink application used to clean, and merge when needed, the data derived from the Information and Extraction module. The application is released as a JAR that can be downloaded at https://gitlab.unige.ch/addmin/ie-cleanser/-/releases (accessed on 14 July 2024). This component must be deployed on an Apache Flink cluster.
- **Universal Unique Identifier (UUID) Generator:** This provides a set of REST APIs that generate a UUID. The component provides two entry points that return v1 and v4 UUIDs, respectively. It is released as a Docker.
- **Profiling:** This module contains the fiscal profiles and the minimum information to be extracted for each of them. The information is represented through the JSON schema, available at https://gitlab.unige.ch/addmin/profiles (accessed on 14 July 2024).
- **Constraints, Rules, and Validation:** This module contains the rules written using SHACL, which is an official W3C standard language for describing a set of conditions that data—specifically, data in the knowledge graphs—should satisfy. SHACL is supported by TopBraid, developed by TopQuadrant, Inc. We used TopBraid Composer Maestro Edition version 7.1.1 in our current research project. We also used SHACL to validate the graphs. A SHACL validation engine takes as input a data graph to be validated and a shape graph containing SHACL shape declarations and produces a validation report, also expressed as a graph. The result of SHACL validation describes whether the data graph matches the shape graph and, if it does not, describes any mismatch. In this way, SHACL can be used to validate that data conform to the desired requirements.

These seven components are integrated to form a complete system, and much of the process is automated. However, there are still a few human-in-the-loop steps.

Figure 7 shows the process that starts with document classification and information and extraction. This is a semi-automated process, as there are tasks that need to be performed manually. Information extraction is based on templates that have to be drawn. User

feedback is required to correct errors or misinterpretations of the extracted data. Then, the step of automating the creation of the JSON files in the format expected by the following components remains to be completed.

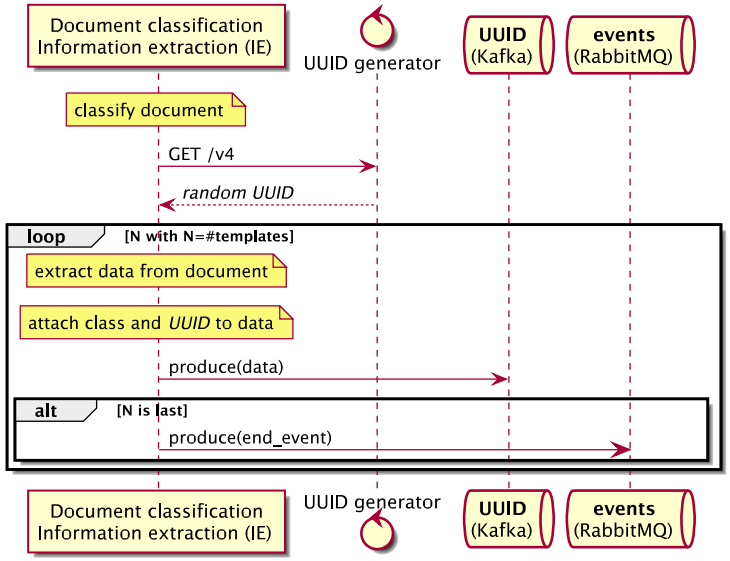


Figure 7. Interaction between the Document Classification/Information Extraction module and the UUID generator.

The information extraction component interacts with the UUID generator and the mapping component via message-oriented middleware (RabbitMQ) and a distributed event streaming platform (Apache Kafka). Specifically, each time a new document arrives, it is classified and assigned a random UUID retrieved from the UUID generator. This UUID must be unique throughout the process. The information extraction phase is performed, and the extracted data are published to a Kafka topic named with the UUID of the document used. Once information extraction is complete, this event is reported as a `ie_end` event message published to a RabbitMQ queue named *events*.

Figure 8 illustrates that the `ie_end` event message published in the queue is consumed by the information and extraction merger component, which was designed to respond to the publication of such an event and to consume the extracted data from the specific UUID Kafka topic in order to merge them into a single JSON object. Once the merge is complete, the data are published to a Kafka topic called *rml_streamer_in*, ready to be cleaned up and then mapped to RDF.

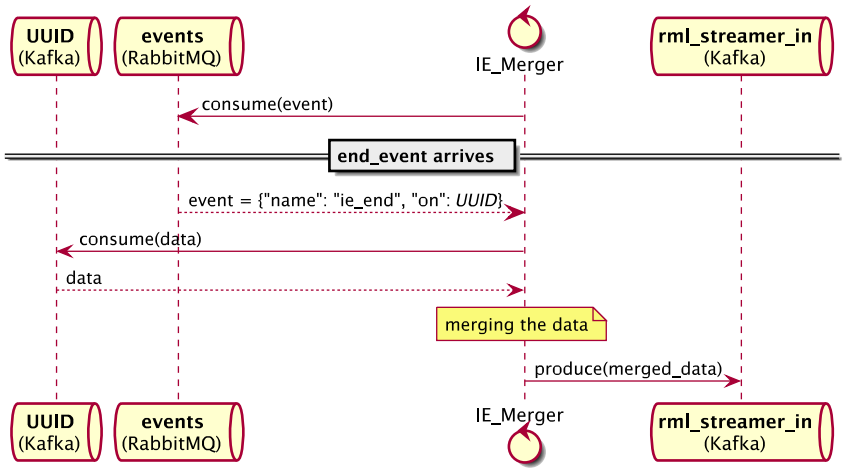


Figure 8. Interaction between the information and extraction merger component and RabbitMQ/Kafka queues/topics to handle events and data.

If no merging is required, the data are immediately published to the *rml_streamer_in* topic by the information extraction component.

Data mapping is followed by a data validation process. During this step, the RDF data are validated against the defined SHACL forms (Section 4.1.4). This phase is still carried out manually by importing the data into TopBraid Composer and running the SHACL validation. An alternative to importing the data into TopBraid Composer is to use the SHACL API (https://github.com/TopQuadrant/shacl, accessed on 14 July 2024)) in a Java application that reads the data and runs the validation rules against them.

The next phases involve profiling and reasoning, which are also carried out manually. Both rely on SHACL rules to infer new data about the profile and the documents the user is required to provide. Once the data are loaded into TopBraid Composer, the inference engine is started, and the rules are executed on the existing data (an example of the results is shown in Figures 12 and 13).

4. Results

In this section, we examine the results of modules of the five modules (Table 1). For each module, we present the results we obtained, highlighting the progress we made, the challenges we faced, and the implications for our overall work. In addition, to contextualise the results and demonstrate the applicability of our approach, we present a case study that illustrates the effectiveness of our system in a practical context.

4.1. Module Results

4.1.1. Ontology Outcome

The ontology is currently developed and contains 240 classes, 24 data type properties, 613 axioms, and 15 object properties. The complete ontology can be found at the following web site: https://gitlab.unige.ch/addmin/doc-onto/-/wikis/home (accessed on 14 July 2024).

4.1.2. Document Classification and Information Extraction Outcomes

In this module's context, we conducted three experiments to evaluate an AI model's performance in classifying tax documents in Switzerland. These experiments provide valuable information for optimising document management processes and improving tax operation efficiency.

During the data preparation phase, we processed the images/documents to create two versions: a high-resolution version for information extraction via OCR and a low-resolution version for classification purposes. This division allowed us to maintain the accuracy of the extracted information while optimising the performance of the classification model. The model was trained using the Adam optimiser and the categorical cross-entropy cost function. Special attention was given to performance evaluation using measures such as the F-measure (F1 score), which combines accuracy and model recall. Specifically, the average F1 score and macro-F1 score were considered to obtain a comprehensive assessment of the model's capabilities across all document classes [21].

All experiments used a classification model based on a CNN followed by two linear layers. The CNN enables the extraction of features from input images using convolutional filters that scan the image and identify visual patterns relevant to the classification task. The two linear layers process the features extracted by the convolutional layer to produce the final predictions on the document categories. The model underwent three epochs of training using the Adam optimiser and categorical cross-entropy as the cost function. Multiple iterations were conducted to optimise the performance of the model.

In our first experiment, we trained the model on a synthetic dataset of tax filing documents in Switzerland. Initially, we artificially created nine categories of documents related to Swiss tax reporting, along with "other classes" representing documents unrelated to this category. These were automatically generated and divided into specific categories, each corresponding to a specific type of document or Information and Extraction template. Next, after training the model on the training data split, the results of the model's classification performance on the test set (which contained only synthetically generated documents), were obtained, which are shown in Table 7.

Table 7. First experiment—per-class classification performance on the test split.

Class Index	Class	Number of Documents (in the Test Split)	F1 Score	Accuracy	Precision	Recall
0	InsuranceBenefitStatement_dataset	514	100%	100%	100%	100%
1	InsurancePremiums_dataset	548	100%	100%	100%	100%
2	3thPillarContributionDeclaration_dataset	248	100%	100%	100%	100%
3	AVSIncome Certification_dataset	17	53.96%	99.04%	36.95%	100%
4	SalaryStatement_dataset	29	0%	99.04%	0%	0%
5	TaxDeclaration_dataset	63	100%	100%	100%	100%
6	BankClosingDocument_dataset	70	100%	100%	100%	100%
7	HealthInsurance_dataset	70	100%	100%	100%	100%
8	2ndPillarPensionDocument_dataset	49	100%	100%	100%	100%
9	OTHER_CLASSES	1415	100%	100%	100%	100%

Following training, the model was evaluated on the test set, which contained only synthetically generated documents. The classification results for each class were as follows: macro-F1 score: 86.73%, average F1 score: 85.39%, micro-F1 score: 99.04%, and average accuracy: 99.80%. Then, the analysis of misclassifications between different document categories revealed the model's tendency to confuse certain similar categories. The performance on the [AVS Income Certification_dataset] and [Salary Statement_dataset] classes was weak, because the visual appearance of these two type of documents was very similar.

To overcome these challenges, in our second experiment, we merged two document categories and proceeded to merge the [Salary Statement_dataset] and [AVS Income Certification_dataset] classes, thus creating the new [Salary Statement Or AVS Income Certificate_dataset] class. These classes are almost identical, with the only difference being the presence of one checkbox. The terms *AVS Income Certification* and *Salary Statement* are essentially the same, with the only difference being the presence of an "A" or "B" checkbox. This indicates that the structure and content of the two document types are nearly identical, except for a single element represented by a checkbox that can be marked with either "A" or "B". The presence or absence of this checkbox distinguishes between the two document classes. We merged the similar classes and retrained the model using the updated training set.

After training the model on the training data split, the results of the model's classification performance on the test set were obtained, which are shown in Table 8.

Table 8. Second experiment—per-class classification performance on the test split.

Class Index	Class	Number of Documents (in the Test Split)	F1 Score	Accuracy	Precision	Recall
0	InsuranceBenefitStatement_dataset	4	100%	100%	100%	100%
1	InsurancePremiums_dataset	1	100%	100%	100%	100%
2	3thPillarContributionDeclaration_dataset	3	0%	88%	0%	0%
3	SalaryStatOrAVSIncomeCert_dataset	10	0%	60%	0%	0%
4	TaxDeclaration_dataset	5	100%	100%	100%	100%
5	BankClosingDocument_dataset	2	18.18%	64%	11.11%	50%
6	AHealthInsurance	0	-	-	-	-
7	2ndPillarPensionDocument_dataset	0	-	-	-	-
8	OTHER_CLASSES	0	-	-	-	-

The evaluation on the test set demonstrated excellent model performance, with an F1 score of 100% for each class. This resulted in better differentiation between similar classes, achieving a macro-F1 score of 92.17%, an average F1 score of 92.55%, a micro-F1 score of 96.00%, and an average accuracy of 96.15%.

In our third experiment, the dataset was further expanded with 21 new classes, bringing the total to 29. After training the model in the training set, evaluation was performed on the test set, resulting in an F1 score of 100% on each class. The results of the evaluation on the set of real documents are reported in Table 9.

Table 9. Third experiment—per-class classification performance on the dataset comprising real documents.

Class Index	Class	Number of Documents (in the Test Split)	F1 Score	Accuracy	Precision	Recall
0	InsuranceBenefitStatement_dataset	4	100%	100%	100%	100%
1	InsurancePremiums_dataset	1	100%	100%	100%	100%
2	3thPillarContributionDeclaration_dataset	3	50%	92%	100%	33.33%
3	SalaryStatOrAVSIncomeCert_dataset	10	94.73%	96%	100%	90%
4	TaxDeclaration_dataset	5	100%	100%	100%	100%
5	BankClosingDocument_dataset	2	66.66%	92%	50%	100%
6	HealtInsurance_dataset	0	-	-	-	-
7	2ndPillarPensionDocument_dataset	0	-	-	-	-
8	UNIGE_cards	0	-	-	-	-
9	NIST_TAX 1040_1	0	-	-	-	-
...	...	...	...	...	...	...
28	NIST_TAX_se_2	0	-	-	-	-

The evaluation on the test set demonstrated excellent model performance, with an F1 score of 100% for each class. This resulted in better differentiation between similar classes, achieving a macro-F1 score of 92.17%, an average F1 score of 92.55%, a micro-F1 score of 96.00%, and an average accuracy of 96.15%. However, despite the improvement in the synthetic data, we still encountered issues as a significant percentage of documents in the [AVS Income Certification_dataset] class were misclassified as [Bank Closing Document_dataset], with an error rate of 80%. This indicates that the model has difficulty distinguishing the differences between these two classes.The experiment demonstrated that documents within a single class were more similar to each other than to those in other classes. This was expected, as the synthetic documents were created from a single model, and the visual resolution for classification was limited.

As a result, a significant improvement in the model's ability to discriminate between the different classes was observed, with only 3 documents misclassified (compared to 14 in the previous experiments). The confusion matrix in Figure 9 confirms the improved model performance.

However, the third experiment's results suggest that there may be more widespread errors with a larger real dataset, despite only a few misclassified documents in the small sample examined. Therefore, adopting a multi-modal approach that considers multiple input modalities, such as images and text, and using a weighted voting system to make the final document class selection may be advantageous.

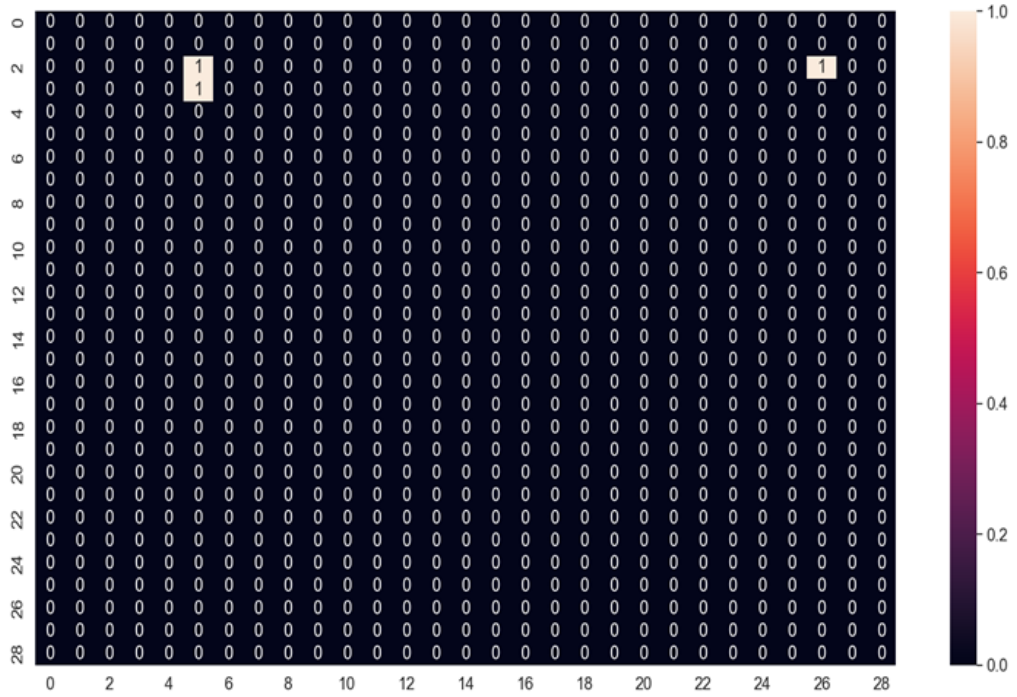


Figure 9. Third experiment—confusion matrix computed on the dataset comprising real documents.

In summary, for this module, we used the following evaluation criteria in relation to each class in the three experiments: F1 score, accuracy, precision, and recall. In Table 10, we summarise the macro-F1, average F1, micro-F1 and average accuracy scores that were obtained.

Table 10. Evaluation criteria and the results of the three experiments.

Experiment	F1 Score per Class	Accuracy per Class	Precision per Class	Recall per Class	Macro-F1 Score	Average F1 Score	Micro-F1 Score	Average Accuracy
First Experiment	✓	✓	✓	✓	86.73%	85.39%	99.04%	99.80%
Second Experiment	✓	✓	✓	✓	92.17%	92.55%	96.00%	96.15%
Third Experiment	✓	✓	✓	✓	92.17%	92.55%	96.00%	96.15%

4.1.3. Document Schema and Profile Generation Outcomes

We developed 31 JSON schema to annotate and validate the documents extracted during the tax filing process.

The schema contains all the relevant information extracted from documents, making data accessibility and management easy. It consists of two main parts: *(i)* the identification section and *(ii)* the information section. The identification section includes fields such as document category, document identifier, and tax household identifier. The information section includes details extracted from the document, such as the insured's name, AVS number, and other relevant information.

Below is an example of a JSON schema for the "Third-Pillar Attestation". A specific JSON schema was defined for this document, which included the necessary fields to store the information extracted from this type of document. The text describes each field and its expected type, and ends with a list of required fields (Listing 1).

The JSON schema for "Third-Pillar Attestation" was then instantiated using the data extracted from an example document. The resulting JSON file contains all identifying fields, along with the information extracted from the document itself. This approach organises all the necessary information for the tax declaration in a structured and understandable way (Listing 2).

Listing 1. The JSON schema corresponding to the Third-Pillar Attestation.

```
{
"$schema": "http://json-schema.org/draft-04/schema#",
"$id": "https://example.com/Attestation3emPillierACotisation.schema.json",
"title": "Record of the 3em pillier A",
"description": "This document records the 3ème pillier A",
"type": "object",
"properties": {
"id_document": {
"description": "L'identifiant du document",
"type": "string"
},
"id_foyerFiscal": {
"description": "L'identifiant du foyer fiscal",
"type": "string"
},
"NomAssure": {
"description": "Le nom de l'assuré",
"type": "string",
"minLength": 2
},
"NumAVS": {
"description": "L'assurance-vieillesse et survivants (AVS) est le numéro d'assurance sociale
↪  et un identifiant personnel.",
"type": "string"
}
},
"required": [
"PrenomAssure",
"DateNaissanceAssure",
"PolicesPrévoyanceAuprèsInstitutionsAssurances",
"Cotisations",
"TotalCotisationsPilier3A",
"NumAVS"
]
}
```

Listing 2. An instantiation of the JSON schema of the Third-Pillar Attestation.

```
{
"Categorie": "Attestation de 3ème pillier",
"id_document": "12.3.200",
"id_foyerFiscal": 111114543,
"AdresseAssure": "Boulevard Carl-vogt 56",
"AdresseInstitutionPrevoyance": "8401 Winterthur",
"AnneesContribution": 2019,
"ConventionsPrevoyanceAupresDeFondationsBancaires": null,
"Cotisations": 6826,
"DateAttestationCotisation": "10.01.2020",
"DateNaissanceAssure": "13.08.1956",
"DebutPolicePrevoyance": 2018,
"EcheancePolicePrevoyance": 2032,
"LieuAttestationCotisation": "Winterthur",
"NomAssure": "Zola",
"NomDeLInstitutionDePrévoyance": "AXA Vie SA",
"NumAVS": null,
"NumeroContratPrevoyance": null,
"NumeroPoliceAssurance": "G 3.654.645",
"PolicesPrévoyanceAuprèsInstitutionsAssurances": null,
"PrenomAssure": "Giovanna",
"TotalCotisationsPilier3A": 6826
}
```

4.1.4. Data Mapping and Reasoning Engine Outcome

The final critical step in our overall evaluation of the document classification system and user profiles was to examine the inference rules. These rules optimise document classification accuracy and personalise the user experience based on individual profiles. In this way, we analysed the effectiveness of these rules and how they contributed to the overall improvement in system performance.

Data generated as triples were integrated with rules in TopBraid to ensure the coherence and reliability of the data for further analysis and use.

The semantic reasoner analysed a set of 23 distinct entities, including individuals, tax household, salary statements, health insurances, gross salary, AVS contributions, and others shown in Table 11. Our DMS processed a total of 120 RDF triples, representing the relationships and characteristics of these entities within the RDF graph and the inferences we obtained.

Table 11. Number of instances per class.

Class	No. Instances
AVS Contributions and Others	2
Gross salary	2
Individuals	8
Insurance	6
Salary Certificate	2
TaxHousehold	3

We examine the results of integrating RDF in the reasoner with regard to two key factors: *(i)* compliance of the data with semantic rules and *(ii)* effectiveness of the rules themselves.

Concerning the *(i)* data compliance, SHACL validation language was used to validate RDF data. A SHACL validation engine takes a data graph and a shape graph containing SHACL shape declarations as input and produces a validation report, which is also expressed as a graph (See https://www.w3.org/TR/shacl/ accessed on 14 July 2024). The validation report describes whether the data graph conforms to the shapes graph and, if it does not, describes each non-conformity. In this way, SHACL allows us to validate that the data conform to the desired requirements (See https://allegrograph.com/shacl-shapes-constraint-language-in-allegrograph/ accessed on 14 July 2024).

The report indicates that 33 RDF data do not align with the semantic constraints. Figure 10 demonstrates the errors found in the salary certificate (*CSalaire/13.3.234*), in which there is no information regarding the employee's date of entry into service or the date of termination of employment (*DateDepartDeEmploye* and *DateEntreeEnServiceEmploye*). We defined this information as a requirement of the salary document. Consequently, the system detecte the absence of these requirements and reported that these property data were not present within the document (*"Property needed to have at least one value, but we found 0"*. Similarly, in other instances of salary statements (*CSalaire/13.3.234* and *CSalaire/13.3.334*, and so on), the system detected an error in the AVS format, as it did not correspond to the value established by the semantic rule (*"Values did not match pattern 756.\d{4}.\d{4}.\d{2}"*).

Property needs to have at least 1 values, but found 0	<http://www.cui.unige.ch/impots.owl#CSalaire/13.3.234>	impots:DateEntreeEnServiceEmploye
Property needs to have at least 1 values, but found 0	<http://www.cui.unige.ch/impots.owl#CSalaire/13.3.234>	impots:Lieu
Property needs to have at least 1 values, but found 0	<http://www.cui.unige.ch/impots.owl#CSalaire/13.3.234>	impots:RetenueDeLImpotALaSource
Property needs to have at least 1 values, but found 0	<http://www.cui.unige.ch/impots.owl#CSalaire/12.3.334>	impots:DateEntreeEnServiceEmploye
Property needs to have at least 1 values, but found 0	<http://www.cui.unige.ch/impots.owl#CSalaire/12.3.334>	impots:Lieu
Property needs to have at least 1 values, but found 0	<http://www.cui.unige.ch/impots.owl#CSalaire/13.3.234>	impots:DateDepartDeEmploye
Property needs to have at least 1 values, but found 0	<http://www.cui.unige.ch/impots.owl#CSalaire/12.3.334>	impots:DateDepartDeEmploye
Value does not match pattern "^756.\d{4}.\d{4}.\d{2}$"	<http://www.cui.unige.ch/impots.owl#Person/Ladoumegue:Jules>	impots:noAVS
Value does not match pattern "^756.\d{4}.\d{4}.\d{2}$"	<http://www.cui.unige.ch/impots.owl#Personne/Ladoumegue_Jules>	impots:noAVS
Value does not match pattern "^756.\d{4}.\d{4}.\d{2}$"	<http://www.cui.unige.ch/impots.owl#Person/zola:Chloe>	impots:noAVS
Value does not match pattern "^756.\d{4}.\d{4}.\d{2}$"	<http://www.cui.unige.ch/impots.owl#Person/zola:Giovanna>	impots:noAVS
Value does not match pattern "^756.\d{4}.\d{4}.\d{2}$"	<http://www.cui.unige.ch/impots.owl#Personne/Zola_Giovanna>	impots:noAVS
Value does not match pattern "^CH"	<http://www.cui.unige.ch/impots.owl#CSalaire/13.3.234>	impots:montant

Figure 10. Extraction of RDF data compliance with SHACL rules.

Once the data were validated against the SHACL forms, the next step was *(ii)* to test the rules to ensure that they could generate inferences from the data. It is crucial to verify that the rules are effective and can provide useful and relevant information to end users. In this project, (i) multi-labelling rules, (ii) direct rules, and (iii) inverse rules for user profiling

were tested. The reasoner engine was operated using TopBraid, and inferences are shown in Figures 11, 12 and 13, respectively.

- (i) Multi-labelling rules: Figure 11 shows that the reasoner assigned tags to document instances that were integrated. The different instances of health insurance (such as *AAssuranceMaladie/12.3.204* and so on) were classified under the categories of tax (*Impôt*, insurance *Assurance*, while the salary certificates (such as *CSalaire/13.3.234* were classified as tax (*Impôt*, income *Revenu*. The system generated a total of 14 such inferences.

impots.owl#AAssuranceMaladie/12.3.204>	impots:tag	Impot
impots.owl#AAssuranceMaladie/12.3.204>	impots:tag	Assurance
impots.owl#AAssuranceMaladie/12.3.205>	impots:tag	Assurance
impots.owl#AAssuranceMaladie/12.3.205>	impots:tag	Impot
impots.owl#AAssuranceMaladie/12.3.206>	impots:tag	Assurance
impots.owl#AAssuranceMaladie/12.3.206>	impots:tag	Impot
impots.owl#CSalaire/12.3.334>	impots:tag	Revenu
impots.owl#CSalaire/12.3.334>	impots:tag	Impot
impots.owl#AAssuranceMaladie/12.3.207>	impots:tag	Impot
impots.owl#AAssuranceMaladie/12.3.207>	impots:tag	Assurance
impots.owl#AAssuranceMaladie/13.3.2015>	impots:tag	Impot
impots.owl#AAssuranceMaladie/13.3.2015>	impots:tag	Assurance
impots.owl#CSalaire/13.3.234>	impots:tag	Revenu
impots.owl#CSalaire/13.3.234>	impots:tag	Impot

Figure 11. Inferences regarding multi-labelling rules in TopBraid.

- (ii) Direct rules: Figure 12 displays the inferences generated by the system regarding the direct relationships between documents and profiles. For example, the system knows that Mrs. Zola Giovanna has submitted a salary statement. Based on the type of document submitted by Mrs. Giovanna, the system infers that she is an employee. This means that the system uses the documents she uploaded to infer her profile, rather than relying on explicit user declarations. The information in Mrs. Zola Giovanna's user profile is useful for preparing her tax return in several ways. For example, as an employee, Mrs. Zola may be entitled to tax deductions for work-related expenses incurred during the tax year. These expenses can include transportation costs to get to work or the purchase of materials necessary to perform her job. Knowing that Mrs. Zola is an employee, the system may suggest that she includes these expenses as deductions in her tax return. Being an employee may also mean that she has received income from employment during the tax year. This income needs to be declared correctly on her tax return, and the system can help her calculate the exact amount to include in her return. The reasoner generated a total of two such inferences.

impots.owl#Personne/Zola_Giovanna>	rdf:type	impots:Salarie
impots.owl#Personne/Ladoumegue_Jules>	rdf:type	impots:Salarie

Figure 12. Inferences regarding direct rules in TopBraid.

- (iii) Inverse rules: Figure 13 shows the inferences related to the inverse relationships between documents and profiles. For example, Mr. Ladoumegue Jules has declared himself an employee. The system will ask Mr. Ladoumegue to upload either a salary certificate or a certificate of training, development, or conversion statement, depending on what is still missing in his profile. Furthermore, the DMS requests Mr. Jules to submit his health insurance, not as an employee but as an individual, in compliance with the rules that require every person to have insurance coverage. The system generated a total of 12 such inferences.

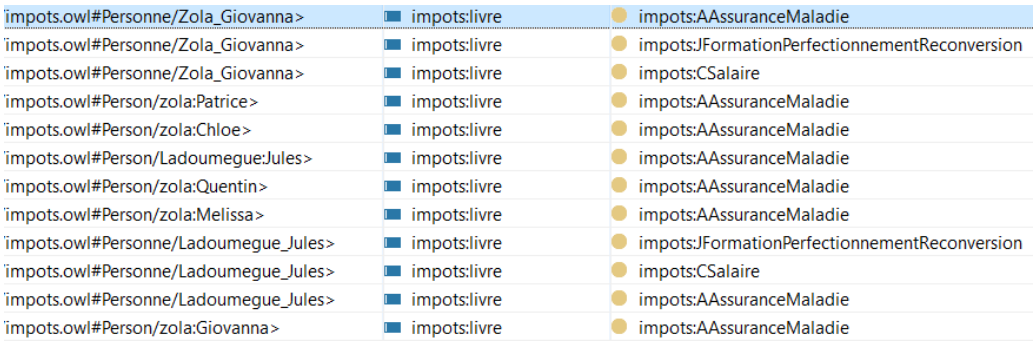


Figure 13. Inferences regarding inverse rules in TopBraid.

The results of the reasoner were evaluated manually, given the limited dataset available. This approach permitted a careful examination of all errors in the conformity of the data with respect to semantic constraints and the verification of the correctness of the inferences with respect to the established semantic rules.

In particular, we conducted a manual evaluation to ascertain the legitimacy of the compliance errors reported by the reasoner. For example, we verified that the salary certificate number 13.3.234 (represented in RDF format) indicates whether the employee's date of entry into service or the date of termination of employment is present. Furthermore, we analysed the system's capacity to accurately multi-classify and recognise the status of individuals. It was demonstrated that the system was capable of accurately identifying individuals such as Mrs. Zola Giovanna and Mr. Ladoumegue Jules as wage earners, given that they had submitted wage certificates. Finally, we examined which documents were required to be delivered by an individual with a specific status, reviewing the content of the rules that defined the type of document to be delivered. Upon completion of the evaluation process, it was confirmed that the semantic rules were functioning as intended and that the RDF data met the constraints and conditions defined by the rules.

By way of resuming, in Table 12, we outline the evaluation criteria adopted to ensure the reliability and accuracy of the document classification system, the profiling of users, and the conformity of RDF data to semantic constraints.

Table 12. Description of evaluation criteria, methods and tools, and results for data mapping and reasoning engine modules.

Evaluation Criteria	Description	Methods and Tools Used	Result
Data Compliance with Semantic Rules	Validation of RDF data against SHACL to ensure that the data conform to the desired requirements.	SHACL validation engine, SHACL shape statements	33 non-conformities, such as missing employee entry/exit dates and incorrect AVS format.
Effectiveness of Inference Rules	Evaluated rules to generate relevant and accurate inferences from data, including multi-labelling, direct and inverse rules.	Semantic reasoning engine in TopBraid	Inference including document classification, user profiling and document presentation requirements.
Manual Evaluation of Reasoning Engine Results	Manual examination of reported errors and the correctness of the inferences to verify the accuracy and legitimacy of the semantic rules and RDF data.	Manual inspection and verification	Confirmed accuracy of compliance errors, correct classification of individuals and document requirements.
Multi-Labelling Rules	Evaluate the system's ability to assign multiple categories to document instances based on the embedded data.	Multi-labelling rules in TopBraid	14 inferences, classifying documents such as health insurance and salary certificates.
Direct Rules	Evaluate direct relationships between documents and user profiles, inferring profile information based on submitted documents.	Direct Rules in TopBraid	2 inferences, such as classifying Mrs. Zola Giovanna as an employee.
Inverse Rules	Evaluate inverse relations where the user's profile information determines the requested documents.	Inverse Rules in TopBraid	12 inferences, requesting documents based on the user's declared status.

4.2. Use Cases

As mentioned in the Introduction, the approach described in this paper applies to any use case in any domain where the focus is on compliance checking to regulations. As we describe the process we developed for tax-related documents, the scenarios of these subsections focus on that domain. The intelligent DMS system used in this scenario was designed to help taxpayers to manage their documents for the preparation of the income tax return. This section presents three use cases that exemplify the functionalities of our DMS as described in the global workflow overview (Section 3.1). These include document management (Section 4.2.1), the identification of missing documents (Section 4.2.2), and the updating of profiles (Section 4.2.3).

4.2.1. Use Case No. 1—A Household with Working Parents and Children

This scenario concerns a household with working parents and children that manages its documents for the preparation of the family income tax return.

As both parents work, the data are extracted from their two salary certificates. Documents are taken as input by the information extraction module, which provides information as JSON data as output. An example of such data is shown in Listing 3. The following JSON file represents the results of extracting information from a tax document related to an employee's salary. The JSON object has a "uuid" field that contains a randomly generated unique identifier to distinguish this document from others. The "class" field specifies the class to which the document belongs, in this case `CSalaire`, which represents a salary tax document. The "id_foyerFiscal" field represents the fiscal identification number of the household to which the employee belongs. The "extractedFields" contains an array of key-value pairs representing the information extracted from the document. Each key-value pair contains a "type" field, which specifies the type of information extracted, and a "value" field, which contains the extracted value. For example, there are fields for the employee's first name and surname, the company's address, the start and end dates of employment, the amount of salary, various withholding taxes, fringe benefits, and other pertinent information related to the document.

Listing 3. JSON data extracted from the salary statement.


```
{
"uuid": ,
"class": "CSalaire",
"id_foyerFiscal": 5677777,
"extractedFields": [
{ "type": "id_document" , "value": "12.3.334"},
{ "type": "AdresseEmploye", "value": "Avenue des acacias 3, 1204 Genêve" },
{ "type": "AllocationsPourFrais", "value": null },
{ "type": "AnnéeCivileVersementSalaire", "value": 2021 },
{ "type": "AutresPrestations", "value": null },
{ "type": "AutresPrestationsSalarialesAccessoires", "value": null },
{ "type": "CotisationsAI", "value": null },
{ "type": "CotisationsAVS", "value":   },
{ "type": "CotisationsAvsAiApgAcAanp", "value": null },
{ "type": "Date", "value": null },
{ "type": "DateDeNaissanceSalarie", "value": "01.01.1986" },
{ "type": "DateDepartDeEmploye", "value": "31.12.2021" },
{ "type": "DateEntreeEnServiceEmploye", "value": "01.01.2021" },
{ "type": "DroitsDeParticipation", "value": null },
{ "type": "IndemnitésDesMembresDeLAdministration", "value": null },
{ "type": "Lieu", "value": null },
{ "type": "NomEmploye", "value": "Zola" },
{ "type": "NomEmployeur", "value": "UNIGE" },
{ "type": "NouveauNumAVS", "value": "457.3742.8881.74" },
{ "type": "NumAVS", "value": null },
{ "type": "Observation", "value": null },
{ "type": "PrenomEmploye", "value": "Giovanna" },
{ "type": "PrenomEmployeur", "value": null },
{ "type": "PrestationsEnCapital", "value": null },
{ "type": "PrestationsNonPériodiques", "value": null },
{ "type": "PrestationsSalairesAccessoires", "value": null },
{ "type": "PrevoyanceProfessionnelle2Pilier", "value": 3170 },
{ "type": "RepasALaCantineOuCèquesRepas", "value": null },
{ "type": "RetenueDeLImpotALaSource", "value": -6070 },
{ "type": "Salaire", "value": 49740 },
{ "type": "SalaireBrutTotal", "value": 44386 },
{ "type": "SalaireNet", "value": 46783 },
{ "type": "TransportGratuitEntreDomicileEtLieuTravail", "value": null }
]
}
```

The first part of the code defines the "impots:CSalaire_12.3.334" object. This object represents the tax document related to the employee's salary. The second part of the code defines the object "impots:Employeur_UNIGE". This object represents the company for which the employee works. The third part of the code defines the object "impots:Personne_Zola_Giovanna". This object represents the employee's personal details, such as her name, surname, date of birth and social security number. The fourth part of the code defines the "impots:FoyerFiscal_5677777" object. This object represents the employee's household, including the person represented by "impots:Personne_Zola_Giovanna" object.

Once the information extraction is complete, the data are sent to the mapping module, which outputs it as RDF triples as shown in Listing 4.

Listing 4. An example of the information extracted from a salary certificate represented using RDF.

```
@prefix impots: <http://www.cui.unige.ch/impots.owl#> .

impots:CSalaire_12.3.334
a impots:CSalaire ;
impots:anneeFiscale 2021 ;
impots:emetteur impots:Employeur_UNIGE ;
impots:destinataire impots:Personne_Zola_Giovanna ;
impots:montant [
a impots:SalaireBrut ;
impots:valeur 44386 ;
] ;
impots:montant [
a impots:impots:CotAVSetAutres ;
impots:valeur 3450 ;
] .


impots:Employeur_UNIGE
a impots:Employeur ;
impots:nomEmetteur "UNIGE" .


impots:Personne_Zola_Giovanna
a impots:Personne ;
impots:nomPersonne "Zola" ;
impots:prenomPersonne "Giovanna" ;
impots:dateNaissance "01.01.1966" ;
impots:noAVS "457.3742.8881.74" ;
impots:foyerFiscal impots:FoyerFiscal_5677777 .


impots:FoyerFiscal_5677777
a impots:FoyerFiscal ;
impots:composeDe impots:Personne_Zola_Giovanna .
```

Once data mapping is complete, the rules for profile classification and document labelling can be run on the mapped data . The results of the profile classification show that Mrs. Zola Giovanna is considered an employee (`impots:Salarie`) who has to provide several documents, such as health insurance (`impots:AAssuranceMaladie`) and salary statement (`impots:CSalaire`). In terms of the multi-label document, the salary certificate (*impots:CSalaire*) is labelled with two tags: "Impot" and "Revenu" (Figure 14).

impots:CSalaire_12.3.334	impots:tag	Impot
impots:CSalaire_12.3.334	impots:tag	Revenu
impots:Personne_Zola_Giovanna	impots:livre	impots:AAssuranceMaladie
impots:Personne_Zola_Giovanna	rdf:type	impots:Salarie
impots:Personne_Zola_Giovanna	impots:livre	impots:CSalaire
impots:Personne_Zola_Giovanna	impots:livre	impots:JFormationPerfectionnementReconversion

Figure 14. Results of applying the profile classification and document labelling rules to the mapped data.

4.2.2. Use Case No. 2—Detection of Missing Documents

Mrs. Zola Giovanna is part of a tax household and has two dependent children. As a type A taxpayer (*ContribuableA*), she has to submit documents related to family allocations (*AAllocationsFamiliales*) for her daughters. Our DMS is able to detect the missing documents and process an error message for Ms. Zola, requiring her to submit them.

The initial dataset comprises the data of Ms. Zola and her two daughters. We observe that the three individuals have the same tax household number (Listing 5).

Listing 5. The initial dataset for the use case n. 2.

```
@prefix impots: <http://www.cui.unige.ch/impots.owl#> .

#Data available for taxpayer Giovanna
<http://www.cui.unige.ch/impots.owl#Personne/Zola_Giovanna>
rdf:type impots:ContribuableA ;
rdf:type impots:Personne ;
impots:addresse "Avenue des acacias 3, 1204 Geneve" ;
impots:dateNaissance "01.01.1966" ;
impots:foyerFiscal <http://www.cui.unige.ch/impots.owl#FoyerFiscal/5677777.0> ;
impots:noAVS "854.6957.0961.94" ;
impots:nomPersonne "Zola" ;
impots:prenomPersonne "Giovanna" ;
.

#Data available for Giovanna's dependent daughter
<http://www.cui.unige.ch/impots.owl#Person/zola:Melissa>
rdf:type impots:Personne ;
impots:dateNaissance "31.10.2004" ;
impots:foyerFiscal <http://www.cui.unige.ch/impots.owl#FoyerFiscal/5677777.0> ;
impots:noAVS "756.6007.5116.06" ;
impots:nomPersonne "zola" ;
impots:prenomPersonne "Melissa" ;
.

#Data available for Giovanna's dependent daughter
<http://www.cui.unige.ch/impots.owl#Person/zola:Chloe>
rdf:type impots:Personne ;
impots:dateNaissance "23.07.1993" ;
impots:foyerFiscal <http://www.cui.unige.ch/impots.owl#FoyerFiscal/5677777.0> ;
impots:noAVS "756.4382.406.056" ;
impots:nomPersonne "zola" ;
impots:prenomPersonne "Chloe" ;
.
```

The first objective of this case study consists of illustrating the DMS's ability to identify members of the same tax household based on the same number of the *foyer fiscal*, as shown in Figure 15 for the three individuals.



Figure 15. Individuals belonging to the same tax household.

The second objective is defining the necessary documents that Ms. Zola, as the main taxpayer of the tax householder, has to deliver. In this context, Ms. Zola has to deliver family allowances, as shown in Figure 16.

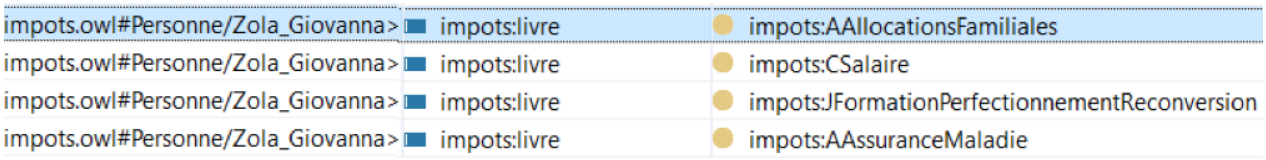


Figure 16. Documents that taxpayer A who belongs to a tax household has to deliver.

Lastly, the third objective consists of demonstrating that our DMS is able to ascertain whether Ms. Zola has submitted the required documentation. In this context, Ms. Zola neglected to submit the documentation for child benefit. Consequently, the DMS generated an error message, indicating that the child benefit documents were missing. This is illustrated in Figure 17.



Figure 17. Error message addressed to taxpayer for missed family allowances.

4.2.3. Use Case No. 3—Profile Update Following Status Change

This case study demonstrates the functionality of our DMS, which enables the updating of taxpayer profiles in the event of status changes. In particular, it illustrates how the system can detect a taxpayer's transition from wage earner to retired status. We consider the case of Mrs. Zola Giovanna, previously identified as a wage earner. Ms. Zola retires in May 2023, resulting in a change in status. Prior to May 2023, Ms. Zola held the status of employee. Consequently, Ms. Zola held two positions in 2023: as an employee from January to April 2023 and as a retiree from May to December 2023. The DMS detects this change and updates Ms. Zola's status, ensuring the delivery of all necessary documents for both types of profiles.

We have the following data in Listing 6.

Listing 6. The initial dataset for the use case no. 3.

```
@prefix impots: <http://www.cui.unige.ch/impots.owl#> .

# Giovanna's data
<http://www.cui.unige.ch/impots.owl#Personne/Zola_Giovanna>
rdf:type impots:ContribuableA ;
rdf:type impots:Personne ;
impots:addresse "Avenue avenue 2, 1223 Geneve" ;
impots:dateNaissance "01.01.1966" ;
impots:foyerFiscal <http://www.cui.unige.ch/impots.owl#FoyerFiscal/5677777.0> ;
impots:noAVS "854.2345.0961.94" ;
impots:nomPersonne "Zola" ;
impots:prenomPersonne "Giovanna" ;
.

# Giovanna's salary statement
<http://www.cui.unige.ch/impots.owl#CSalaire/12.3.334>
rdf:type impots:CSalaire ;
impots:anneeFiscale "2021.0" ;
impots:destinataire <http://www.cui.unige.ch/impots.owl#Personne/Zola_Giovanna> ;
impots:montant <http://www.cui.unige.ch/impots.owl#CotAVSetAutres/3450.0> ;
impots:montant <http://www.cui.unige.ch/impots.owl#SalaireBrut/49740.0> ;
.
```

The case study in question aims to achieve three objectives. The first goal consists of demonstrating the ability of the DMS to detect the change in status of the taxpayer from salaried to retired. Ms. Zola delivers a pension statement like the following in Listing 7.

Listing 7. An example of pension statement represented in RDF.

```
@prefix impots: <http://www.cui.unige.ch/impots.owl#> .

<http://www.cui.unige.ch/impots.owl#ARenteAVSOuAI/14.5.197>
rdf:type impots:ARenteAVSOuAI ;
impots:destinataire <http://www.cui.unige.ch/impots.owl#Personne/Zola_Giovanna> ;
impots:AVSShape "756.1234.5678.97" ;
impots:DateNaissanceShape "1950-10-14"^^xsd:date ;
impots:EmetteurShape "Caisse cantonale de compensation" ;
impots:MontantShape 35000.00 ;
impots:nom_PersonneShape "Zola" ;
impots:prenom_PersonneShape "Giovanna" ;
impots:AdresseRentier "Via Roma 123, 6900 Lugano" ;
impots:AnneeCivileVersementRentes 2023 ;
impots:DateAttestationRente "2023-05-31"^^xsd:date ;
impots:LieuAttestationRente "Lugano" ;
```

The DMS is able to ascertain, based on wage and pension documents, that Ms. Zola had two status categories in 2023: that of an employee (*Salarie*) and that of a pensioner (*Retraite*). This is illustrated in Figure 18.

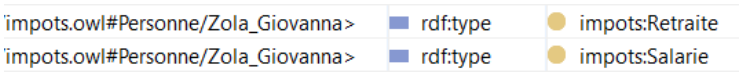


Figure 18. The taxpayer who has two profiles during a tax year.

The second objective of this use case is to demonstrate that the DMS is capable of ensuring that Ms. Zola delivers all tax documents related to the period of employment prior to retirement.

Finally, the third objective is to demonstrate that the DMS is capable of handling the new document phase related to retirement.

The two system abilities are illustrated in Figure 19, where the documents that Ms. Zola has to submit are defined. Some of these documents are related to her status as a retiree, such as *ARente3ePilier*, and *ARenteLPP*, while the rest of documents are linked to her employee status.

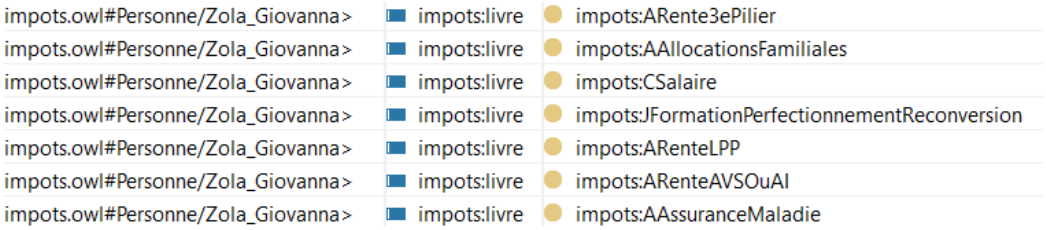


Figure 19. Types of documents to be delivered based on the status held by the taxpayer.

5. Conclusions and Discussion

This article presents a comprehensive solution to an intelligent DMS that offers an administrative and fiscal document management service to users.

The proposed system can automate the processing of tax and administrative documents for individuals, as well as automate the document classification process and user profile identification, reducing processing time and costs. Using inference rules, the system can suggest personalised recommendations based on the user's profile, improving the user experience and the effectiveness of the recommendations provided. Finally, leveraging the SHACL language, the solution ensures data integrity and consistency.

The proposed solution was developed in several steps: *(i)* the development of an ontology including documents, user profiles, and profile changes; *(ii)* the use of ML approach for document classification and information extraction; *(iii)* the generation of document schema and tax household profiles; *(iv)* data mapping from JSON data to RDF triples exploiting the ontology; *(v)* the development of a reasoning engine, which, through SHACL shapes and rules, allowed us to classify documents, multi-label them, and classify user profiles; and *(vi)* the integration of RDF triples and rules.

The project was tailored to the Swiss context and its tax legislation. Therefore, the adopted approach was designed to comply with Swiss laws and regulations, which served as the basis for defining the specific steps of the project. For example, Swiss tax documents formed the basis of the information exchange process used in this project.

Furthermore, the presented case studies demonstrated some strengths of the system, such as its ability to extract the relevant data from the parents' wage statements, eliminating the need for manual entry, and reducing the time required to prepare the tax return. In addition, the system accurately extracts critical financial information, such as salary, withheld taxes, and other employment-related benefits. It can then be configured to manage the tax documents of a family with working parents and children, allowing for customisation of features based on the specific needs of the family. In fact, the system is designed to be highly adaptable and can be easily customised to meet the specific needs of different contexts and industries. In particular, the system is interoperable with other systems through APIs that allow for the exchange of information with other systems or software. Then, the adaptability of our DMS also pertains to the flexibility of the data format. The system is able to manage many formats, such as JSON and RDF. This capability allows it to become more flexible and interoperable with other systems and software that use these formats.

The system facilitates integration with other document management systems or accounting software, improving the overall consistency and efficiency of the family tax process due to the data transformation according to the JSON and RDF format.

The system is capable of addressing a wide range of application scenarios. In fact, the three use cases are representative examples of the potential applications and benefits of the intelligent DMS.

Nevertheless, the proposed approach and implementation present a few limitations. Despite the broader scope of the initial research project, the proposed solution only targets Swiss tax return documents written in French. The solution does not handle any document provided in German, Italian, English, or Romansh. Consequently, our DMS cannot process contributor documents that are written in a language other than French. We should point out that although our system is currently limited to Swiss tax documents written in French, the potential application is broad. Many institutions such as fiduciary offices, audit firms, and insurance brokers can benefit from our solution. For example, our system can be implemented by institutions in Switzerland that handle tax and administrative documents. Then, each institution can process thousands of documents per month, including tax returns, financial reports, and other administrative documents. These estimates help to measure the usefulness of our DMS and demonstrate that despite the initial language and geographical limitations, the system has significant potential to improve efficiency and reduce operational costs for a wide range of users and institutions.

In addition, the solution does not take into account the privacy and confidentiality challenges that might arise. It is also important to note that a business-to-business (B2B) system for professionals integrating the proposed solution must carefully consider and address such concerns to ensure compliance with laws and regulations. Therefore, we worked with a limited availability of Swiss tax documents. This poses a significant challenge to the development of the system. As a large dataset of tax documents is required to perform ML tasks and related analyses, this was an obstacle to the progress of the project. Even though this paper focuses on the specific domain of tax return, the approach itself applies to any domain where regulations need to be extracted and verified within a digital service. Other research works we developed related to safety in automated vehicles and compliance to the regulation of underground geographic data (e.g., root trees, utility pipes). We are currently developing a general framework for compliance checking [24].

Finally, future work will explore the integration of generative AI, particularly large language models (LLMs), as a way to improve the various steps of our process flow. Specifically, information extraction from administrative documents into JSON files, ontology generation, and regulation (rule) extraction can be enhanced. We would then combine together the power of LLMs with the one of knowledge graphs and reasoning.Following this direction will lead to even more effective and innovative results, enabling organisations to further optimise document management and focus more on value-added activities.

Author Contributions: Conceptualisation, G.D.M.S.; Methodology, G.D.M.S., G.F., C.M., and A.-F.C.-D.; Software—Information Extraction, S.G.; Software—Data Mapping to RDF, A.C.; Ontology, C.M.; Reasoning Engine Development, M.A.C.; JSON files generation: A.W.; Validation, S.G., A.C., M.A.C., and G.C.; Writing of Individual Technical Parts, all authors; Writing—Draft Preparation, Writing—Review and Editing, M.A.C.; Supervision, G.D.M.S.; Project Administration, G.D.M.S.; Funding Acquisition, G.D.M.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Innosuisse in the framework of the innovation project 50606.1 IP-ICT entitled "Addmin–Private computing for consumers' online document access".

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data supporting the findings of this study are openly available at UNIGE GitLab, under the dataset titled "Document Ontology", at https://gitlab.unige.ch/addmin/doc-onto/-/wikis/home (accessed on 14 July 2024); under the dataset titled "Rules", at https://gitlab.unige.ch/addmin/rules (accessed on 14 July 2024); under the dataset titled "Profiles" https://gitlab.unige.ch/addmin/profiles (accessed on 14 July 2024); under the dataset titled "RML Mappings", at https://gitlab.unige.ch/addmin/rml-mappings, (accessed on 14 July 2024) and under the dataset titled "IE Cleanser", at https://gitlab.unige.ch/addmin/ie-cleanser, (accessed on 14 July 2024).

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AI	Artificial Intelligence
AVS	Assurance-vieillesse et survivants
CNN	Convolutional Neural Network
CNNs	Convolutional Neural Networks
DMS	Document Management System
DMSs	Document Management Systems
LLMs	Large Language Models
ML	Machine Learning
NLP	Natural Language Processing
OCR	Optical Character Recognition
OWL	Web Ontology Language
RDF	Resource Description Framework
RML	RDF Mapping Language
SPARQL	SPARQL Protocol and RDF Query Language
UUID	Universal Unique Identifier

References

1. Stylianou, N.; Vlachava, D.; Konstantinidis, I.; Bassiliades, N.; Peristeras, V. Doc2KG: Transforming Document Repositories to Knowledge Graphs. *Int. J. Semantic Web Inf. Syst.* **2022**, *18*, 1–20. [CrossRef]
2. Serugendo, G.D.M.; Cappelli, M.A.; Glass, P.; Caselli, A. *The Semantic Approach to Recognise the Components of the Underground Cadastre*; Technical Report; University of Geneva: Geneva, Switzerland, 2024. Available online: https://archive-ouverte.unige.ch/unige:175632 (accessed on 14 July 2024).
3. Cappelli, M.A.; Di Marzo Serugendo, G.; Cutting-Decelle, A.F.; Strohmeier, M. A semantic-based approach to analyze the link between security and safety for Internet of Vehicle (IoV) and Autonomous Vehicles (AVs). In *Proceedings of the CARS 2021 6th International Workshop on Critical Automotive Applications: Robustness & Safety*; HAL Inserm: Münich, Germany, 2021. Available online: https://hal.archives-ouvertes.fr/hal-03366378 (accessed on 14 July 2024).
4. Staab, S.; Studer, R. (Eds.) *Handbook on Ontologies*; Springer Science & Business Media: New York, NY, USA, 2010.
5. Staab, S.; Studer, R.; Schnurr, H.; Sure, Y. Knowledge processes and ontologies. *IEEE Intell. Syst.* **2001**, *16*, 26–34. [CrossRef]
6. Noy, N.F.; McGuinness, D.L. *Ontology Development 101: A Guide to Creating Your First Ontology*; Protege: Portland, OR, USA, 2001.
7. Augereau, O.; Journet, N.; Vialard, A.; Domenger, J.P. Improving classification of an industrial document image database by combining visual and textual features. In Proceedings of the 11th IAPR International Workshop on Document Analysis Systems, Tours, France, 7–10 April 2014; pp. 314–318.
8. Shovon, S.S.F.; Mohsin, M.M.A.B.; Tama, K.T.J.; Ferdaous, J.; Momen, S. CVR: An Automated CV Recommender System Using Machine Learning Techniques. In *Data Science and Algorithms in Systems: Proceedings of 6th Computational Methods in Systems and Software 2022, Vol. 2*; Springer: New York, NY, USA, 2023; pp. 312–325.
9. Eswaraiah, P.; Syed, H. An efficient ontology model with query execution for accurate document content extraction. *Indones. J. Electr. Eng. Comput. Sci.* **2023**, *29*, 981–989. [CrossRef]
10. Bratarchuk, T.; Milkina, I. Development of electronic document management system in tax authorities. *E-Management* **2021**, *3*, 37–48. [CrossRef]
11. Sambetbayeva, M.; Kuspanova, I.; Yerimbetova, A.; Serikbayeva, S.; Bauyrzhanova, S. Development of Intelligent Electronic Document Management System Model Based on Machine Learning Methods. *East.-Eur. J. Enterp. Technol.* **2022**, *1*, 115. [CrossRef]
12. Justina, I.A.; Abiodun, O.E.; Orogbemi, O.M. A Secured Cloud-Based Electronic Document Management System. *Int. J. Innov. Res. Dev.* **2022**, *11*, 38–45. [CrossRef]
13. Ustenko, S.; Ostapovych, T. Amazon Kendra at banking document management system. *Access J.* **2023**, *4*, 34–45. [CrossRef] [PubMed]
14. Martiri, E.; Muca, G.; Xhina, E.; Hoxha, K. DMS-XT: A Blockchain-based Document Management System for Secure and Intelligent Archival. In Proceedings of the RTA-CSIT, Tirana, Albania, 23–24 November 2018; pp. 70–74.
15. Sladić, G.; Cverdelj-Fogaraši, I.; Gostojić, S.; Savić, G.; Segedinac, M.; Zarić, M. Multilayer document model for semantic document management services. *J. Doc.* **2017**, *73*, 803–824. [CrossRef]
16. *ISO IEC 82045-1*; International Organization for Standardization (ISO). Document Management—Part 1: Principles and Methods. ISO: Geneva, Switzerland, 2001.
17. Errico, F.; Corallo, A.; Barriera, R.; Prato, M. Dematerialization, Archiving and Recovery of Documents: A Proposed Tool Based on a Semantic Classifier and a Semantic Search Engine. In Proceedings of the 2020 9th International Conference on Industrial Technology and Management (ICITM), Oxford, UK, 11–13 February 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 297–301.
18. Fuertes, A.; Forcada, N.; Casals, M.; Gangolells, M.; Roca, X. Development of an ontology for the document management systems for construction. In *Complex Systems Concurrent Engineering*; Springer: New York, NY, USA, 2007; pp. 529–536.

19. Ferrando, J.; Domínguez, J.L.; Torres, J.; García, R.; García, D.; Garrido, D.; Cortada, J.; Valero, M. Improving accuracy and speeding up document image classification through parallel systems. In Proceedings of the Computational Science–ICCS 2020: 20th International Conference, Amsterdam, The Netherlands, 3–5 June 2020; Proceedings, Part II 20; Springer: New York, NY, USA, 2020; pp. 387–400.
20. Knublauch, H.; Kontokostas, D. Shapes Constraint Language (SHACL), W3C Recommendation 20 July 2017. Available online: https://www.w3.org/TR/shacl (accessed on 14 July 2024).
21. Di Marzo Serugendo, G.; Falquet, G.; Metral, C.; Cappelli, M.A.; Wade, A.; Ghadfi, S.; Cutting-Decelle, A.F.; Caselli, A.; Cutting, G. *Addmin: Private Computing for Consumers' Online Documents Access: Scientific Technical Report*; University of Geneve: Geneve, Switzerland, 2022. Available online: https://archive-ouverte.unige.ch/unige:162549 (accessed on 14 July 2024).
22. Cappelli, M.A.; Caselli, A.; Di Marzo Serugendo, G. Designing an Efficient Document Management System (DMS) using Ontology and SHACL Shapes. *J. Vis. Lang. Comput.* **2023**, *2*, 15–28. Available online: https://ksiresearch.org/jvlc/journal/JVLC2023N2/paper034.pdf (accessed on 14 July 2024). [CrossRef]
23. Cappelli, M.A.; Caselli, A.; Di Marzo Serugendo, G. Enriching RDF-based Document Management System with Semantic-based Reasoning. In Proceedings of the The 29th International DMS Conference on Visualization and Visual Languages, DMSVIVA 2023, KSIR Virtual Conference Center, Pittsburgh, PA, USA, 29 June–3 July 2023; KSI Research Inc.: Pittsburgh, PA, USA, 2023; pp. 44–50. [CrossRef]
24. Caselli, A.; Serugendo, G.D.M.; Falquet, G. A Framework for Regulatory Compliance using Knowledge Graphs. In Proceedings of the 2023 Digital Sciences Day, Geneva, Switzerland, 28 June–1 July 2023; Centre Universitaire d'Informatique (CUI): Geneva, Switzerland, 2023.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.