Semantic Rule-based Approach for an Intelligent Document Management System

Giovanna Di Marzo Serugendo a, Gilles Falquet a, Claudine Métal a, Maria Assunta Cappelli a, Assane Wade a, Sami Ghadfi a, Anne-Françoise Cutting-Decelle a, Ashley Caselli a and Graham Cutting a
Centre Universitaire d’Informatique (CUI), Université de Genève, Genève
E-mail: 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. A combination of artificial intelligence methods has been adopted to create a system that can perform multiple functions, including the classification of Swiss insurance and fiduciary documents, the definition of tax profiles, and the extraction of inferences from the application of SHACL rules on tax profiles. 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 such as consulting. The creation of the DMS is carried out in several steps, including the development of an ontology for Swiss tax returns; the use of two alternative approaches for document classification and information extraction; the definition of tax family profiles; the representation of extracted data using RDF triples; and the classification of users into distinct profiles based on the tax documents provided. The system was tested in a case study consisting of tax return preparation.

Keywords: AI, Document Classification, Data mapping to RDF, Information Extraction, Machine Learning Approach, Reasoning Engine, Knowledge Engineering

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, 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 has been found in the implementation of automated DMSs that store large amounts of textual information, enriched with metadata to support searchability [1].

The use of the DMS leads to a reduction in the amount of paperwork to be managed but above all 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 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 of 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 Document Management System (DMS). This DMS is built through the combination of several specific modules. These modules encompass ontology, information extraction, and semantic reasoning, for each of which specific techniques were adopted. In this case, semantic reasoning also allows the DMS system to comply with tax regulations. In other words, the reasoning that the DMS develops will be compliant with the rules in force in the canton of Geneva for filing tax returns that are present in the “Canton of Geneva’s Tax Guide 2020”. Thus, the DMS allows for the automation of compliance with these rules. 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 on the basis of semantic rules.

The paper is organised as follows. Section 2 provides an overview of related work. Section 3 discusses the research approach and methodology. The global workflow and the architecture of the proposed intelligent DMS are described in Section 4. Section 5 and 6 provide an explanation of each step developed within the research and a use case that exploits the proposed DMS. A discussion of the technological requirements and technology used to develop the proposed solution is provided in Section 7. Finally, Section 8 concludes the article by discussing the limitations of the approach and future work.

2. Related works

This section is organised into three subsections, each providing an overview of related work of the following topics: Intelligent DMS (section 2.1), Information Extraction (section 2.2), and Semantic Reasoning (section 2.3).

Firstly, current studies on intelligent DMS will be described. Some current studies on the specific methods used to implement intelligent DMS are then analysed. This in-depth analysis examines the techniques leveraged in this paper in order to compare them with the current usages.

2.1. Intelligent DMS

Some recent advances in AI have enabled the development of intelligent DMSs that aim to deliver benefits on a larger scale. Ustenko et al. [2] present Amazon Kendra to improve the work of electronic document flow in the banking sector. Amazon Kendra is an intelligent semantic search system and an indexing interface. The service uses ML and natural language processing to understand the content of documents and provide accurate responses to user queries. Amazon’s intelligence lies in the type of search it performs, in the sense that it is able to search for relevant answers to queries even on the basis of keywords alone since it is able to understand the content of the query and search through data distributed in different locations within the institution.

Pasieka et al. [3] describe an advanced medical information management system that manages business documents, including patient records. This system uses AI to extract and analyse large amounts of data and select important information about patients. The system enables the structured organisation of information through the use of specialised databases and data warehouses that are document-oriented. The system handles administrative tasks, including managing medical records, scheduling patient visits and updating medical records according to international standards. In addition, the system can perform detailed searches and in-depth analysis of patient data. This allows trends to be identified and treatment protocols to be improved.

Martiri et al. [4] introduced blockchain to DMS by developing the DMS-XT system, which aims to automate the storage and retrieval of documents using artificial intelligence algorithms. DMS-XT extracts part of the unstructured content of PDF documents using information extraction techniques. The result of the extraction is stored in the blockchain, which allows the ownership and content of the document to be verified by retrieving and decrypting it. The DMS-XT system was used to simplify and coordinate the electronic management of students’ theses’ 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. Lecturers, in turn, manage the research proposals submitted by...
students and can check for plagiarism in the drafts submitted by students and ultimately approve the final theses. The system also communicates with the department secretary, who receives the verification emails generated by the system and takes appropriate action, which consists of assigning tutors to students and monitoring the whole process.

Sambetbayeva et al. [5] propose 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 is presented as a solution for the management of a large number of documents in an organisation or company. In particular, multi-agent text analysis and information extraction involve two types of agents: lexical agents and cognitive-linguistic agents. The former corresponds to the subject matter objects found in the text, while the latter type of agents describe these objects and make connections between them. Both types act in parallel and independently, extracting information from the text in the form of ontology-based structures, such as facts, objects, and relations. The authors argue that the use of a multi-agent approach is a viable alternative to text analysis systems with sequential architectures. These studies used AI methods to enhance the capabilities of traditional DMSs, further improving efficiency, productivity and customer satisfaction. However, these intelligent DMSs have focused on one or two techniques, but none have combined all the techniques.

2.2. Information Extraction

Information extraction is a technique that has developed over time through the use of more sophisticated extraction techniques. Information extraction occurs not only in the context of DMS but in various sectors. Augereau et al. [6] propose a combined method for classifying document images. This method consists of combining the Bag of Words (BoW) technique and the Bag of Visual Words (BoVW) technique. The former makes it possible to extract the keywords of a document, while the latter makes it possible to extract the characteristics of the images of a document. This information is used to train learning algorithms to detect correlations between the textual and visual features that have been extracted and categories of documents. In other words, the algorithms can detect that certain keywords combined with certain image features correlate with certain document categories. Once the algorithms have learned the pattern, they will be used to categorise new documents. This combined method was tested on an industrial database of document images from 1925.

Ferrando et al. [7] propose a method for document classification based on the use of visual and textual features. The predictions from both models are combined to produce the final prediction of the document class. Shovon et al. [8] propose an automated personnel selection system based on the match between the candidates’ competencies, declared in their CVs, and the job description. The study presents three phases corresponding to the use of three different techniques. The first phase consists of extracting relevant information from CVs and job descriptions. The second phase is to compare the extracted information in order to identify the CVs that meet the criteria of the job description. Of the various metrics tested, conditional probability is considered to be the most suitable for finding similarities. Finally, the third stage consists of recommending candidates that match the job description through a recommendation technique.

Eswaraiah et al. [9] propose a process for extracting information from documents by exploiting ontology. The authors propose an optimised ontology model called “OOM-QE-CE” that facilitates information retrieval. The ontology facilitates information retrieval as it organises knowledge in a specific domain in a structured manner. This ordered knowledge allows information to be retrieved through queries that the user addresses to the system. Consequently, user queries are converted into ontology queries that are executed on the set of indexed documents. Information extraction represents a specific method using techniques for extracting textual and visual features from documents, ML, ontologies and ontological queries, etc. These represent only some of the works that have been presented on information extraction.

2.3. Semantic Reasoning

Semantic techniques such as ontologies and multi-level document representations have been used in previous DMSs to improve the search, classification, and identification of document relationships.

Sladic et al. [10] 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 the ISO 82045 standards. This standard
offers a more detailed representation of documents together with their generalised life cycle. The choice of these standards guarantees a high degree of interoperability with other DMSs. 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). Some of the studies concerning the use of the semantic method for the DMS are limited to the use of ontologies but do not go as far as the combined application of ontology and semantics.

Errico et al. [11] create 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 takes place 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 a machine learning algorithm called Latent Dirichlet Allocation (LDA) that “allows the association of each new document to one or more clusters and to a set of tags useful for the subsequent phase of research and retrieval of documents”.

Existing semantic DMSs mainly rely on ontologies and classifications but do not use semantic reasoning via rules as the proposed DMS does.

3. Research approach and methodology

The proposed intelligent DMS is the result of a research approach involving various techniques implemented in six phases (see Table 1). The preliminary phase is the initial stage of the Intelligent DMS development process, which includes the definition of key concepts such as classification, information extraction, user profile, and tagging, as well as the definition of the basic terminology used in the system. The “classification” term allows to assign a label to an object. For the purposes of this research, classification assigns a class to the document. For example, a hospital invoice is 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”¹. Each document can therefore be labelled, or tagged (from one to several for each document), allowing a classification other than the fiscal classification. Tagging documents provides additional context and categorisation beyond their fiscal classification.

Once the basic concepts and terminology have been established in the preliminary phase, the development of the intelligent DMS can proceed through the six mentioned planning phases. These planning phases involve the implementation of various techniques for developing an intelligent DMS. Table 1 summarises them and provides a list of the steps with an explanation and expected outcome for each.

¹Definition of label noun from the Oxford Advanced Learner’s Dictionary https://www.oxfordlearnersdictionaries.com/definition/english/label_1
Table 1
Development phases for the proposed intelligent DMS

<table>
<thead>
<tr>
<th>PHASE</th>
<th>NAME</th>
<th>DESCRIPTION</th>
<th>OUTCOME</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>ONTOLOGY DEVELOPMENT</td>
<td>Development of an ontology for fiduciary, insurance, and user profiles</td>
<td>→ Representation of concepts of the fiduciary and insurance domains</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>→ Representation of concepts related to the tax profiles</td>
</tr>
<tr>
<td>II</td>
<td>DOCUMENT INFORMATION EXTRACTION</td>
<td>The information is extracted using an information extraction module that can process PDFs or scanned documents</td>
<td>→ JSON file containing extracted information</td>
</tr>
<tr>
<td>III</td>
<td>DOCUMENT SCHEMA AND PROFILES DEFINITION</td>
<td>Defining documents schemas and tax profiles through their relevant information</td>
<td>→ JSON schema for documents schema and tax profiles</td>
</tr>
<tr>
<td>IV</td>
<td>DATA MAPPING TO RDF</td>
<td>Map extracted information to RDF</td>
<td>→ JSON file to RDF triples by leveraging the ontology vocabulary</td>
</tr>
<tr>
<td>V</td>
<td>REASONING ENGINE</td>
<td>Defining SHACL shapes and SHACL rules</td>
<td>→ Document classification and recognition</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>→ Rules for multi-label classification</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>→ User profile classification rules</td>
</tr>
<tr>
<td>VI</td>
<td>DATA INTEGRATION</td>
<td>Integration of RDF triples and rules</td>
<td>→ SHACL validation of data rules</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>→ Evaluating the effectiveness of reasoning rules</td>
</tr>
</tbody>
</table>

The first phase involves the development of an ontology (defined for French terms) that includes concepts for representing fiscal and administrative documents as well as their features, user profiles, and changes to profiles (see Section 5.2).

The second phase uses information extraction techniques for extracting information from documents (see Section 5.1). The expected outcome of this phase is the provision of JSON documents containing extracted information.

The third phase focuses on the definition of the document’s relevant information and profiles. As the expected outcome, the phase provides JSON schema files that contain the documents’ structure relevant information and the tax profile definitions (see Section 5.3).

The fourth phase focuses on the mapping of the extracted information, represented as JSON file, to RDF triples by following the defined ontology (see Section 5.4).

During the fifth phase, a reasoning engine is developed. The engine leverages SHACL shapes and rules to (i) classify and recognise documents for tax returns; (ii) infer the labels to assign to the tax documents; and (iii) classify users into different user profiles. More details are available in Section 5.5.

Finally, the sixth phase focuses on integrating the data that have been extracted, mapped, and inferred during the previous steps. The data integration is done in two parts: (i) a SHACL validation of the RDF data is performed. This allows one to validate that the extracted data are syntactically and semantically valid. The aim is to ensure that the data is complete, correct, and consistent with the document’s relevant information; and (ii) evaluate the effectiveness of the reasoning rules developed in the previous phase. This is done by testing the rules on sample data and evaluating the inferences made by the inference engine. The more correct and relevant inferences the engine provides, the more effective the rules are considered to be (see Section 5.6). The inferences made by the engine (e.g. classifying documents, users and identifying missing documents) are compared to expected or known results to determine accuracy and relevance. This provides a measure of the effectiveness of the rules in accomplishing the intended tasks.

4. Global workflow and architecture

The proposed intelligent semantic-based DMS is developed using a methodology consisting of six phases (Table 1) that are implemented according to the workflow shown in Figure 1.
The workflow integrates Ontology development (phase I), Information Extraction (phase II), Document schema and profile definition (phase III), Data mapping to RDF (phase IV), Reasoning Engine (phase V), and Data integration (phase VI).

- The documents (native PDFs or scanned documents) are processed by a document classification and information extraction module. This module (i) classifies the type of document, (ii) extracts the information, and (iii) generates a JSON file for each document with the extracted information.
- The JSON files are processed by a component containing (i) an ontology of the Swiss Tax declaration, (ii) household profiles, (iii) a rule-based reasoner. The reasoner updates JSON profiles based on new information (e.g. a new child is in the household) and identifies missing documents based on existing profiles (e.g. health fees are missing for a person identified as part of the household).

Figure 2 shows that Ontology (Phase II), Data Mapping to RDF (Phase IV), and the Reasoning Engine (Phase V) are tightly integrated in the upgrade process. Together, they enable: (i) ontology development to represent documents and profiles, (ii) rule writing to update profiles and classify documents, and (iii) RDF mapping to store and reason about the extracted information.

The workflow in Figure 2 focuses on three main activities:

- Building the ontology of the Swiss tax return based on the actual legal documents of the tax return and the actual documents required to complete the tax return. The ontology component describes the Fiscal Budget, Individuals, Documents, Profiles, and Tax Categories.
- Writing rules for tasks such as document validation, profile updates, identification of missing documents, and document labelling. These rules are applied by the reasoning engine.
- Storing the RDF representation of JSON files containing extracted information and user profiles. The JSON files are processed by information extraction and updated by rule applications.
The process of mapping JSON files to RDF using an ontology vocabulary involves two main phases: data mapping and ontology-based transformation (see Section 5.4).

5. Planning stages for an intelligent DMS

This section is dedicated to analysing the six steps that were taken to design and implement an intelligent DMS, as described in Table 1.

5.1. Document classification and Information extraction

Document classification and information extraction are two important tasks in DMS. Document classification is the process that enables to assign to predefined categories or classes based on their content. This helps to organise large volumes of documents and enables more targeted search and retrieval. Information extraction involves finding relevant information within documents and structuring it in a standardised format.

Document classification and information extraction tasks can be challenging due to the diversity of real-world documents encountered in specific contexts, such as the Swiss financial context.

The general workflow showing these different approaches is illustrated in Figure 3.
This diagram shows the flow of document classification and information extraction. The first step is to input the files containing the documents to be processed. The documents are then classified into specific categories through ML techniques. The categories assigned to the documents by this process are represented by the “Document Category” section. The next step is to extract information from the documents. This information is stored in the Extracted Information section. Finally, the extracted information is mapped into JSON format. This process is represented by the “JSON Mapping” section.

The limited number of real documents available during the project made the task even more difficult. For this task, ML, rules-based approaches, and natural language processing are used. One of these techniques, or a combination of them, could handle most forms or documents, but a more diverse dataset would be required to adapt the methodology to a real-world situation.

**Machine Learning (ML) Approach** The first approach was to use ML for document classification and information extraction. The main research effort for Step ML has been based on three aspects, such as (i) the architecture using features learned by deep convolutional neural networks (CNNs); (ii) templates developed for each type of document selected according to the class of the document; (iii) the key information extracted after applying an optical character recognition (OCR) process.

The three techniques - CNNs, templates, and OCR - were chosen to address the challenges of document classification and information extraction in a real-world environment where documents of different types and formats are expected. CNNs were chosen because they have been used successfully in the past for document analysis and recognition, as evidenced by good results in specific competitions and rankings. CNNs can extract specific features from text or images of documents. Both text and images can be used for purposes such as categorising documents or improving text quality. ML models that have been trained have been preferred as a tool because they have proven to be able to achieve very accurate results in document classification and information extraction. To create a template, some initial effort is required to identify the fields of interest and their location within the document. The template can then be used to automatically extract information from documents of that type. To generate machine-readable text, OCR was chosen because it was expected that real documents would be a combination of PDF forms and scanned images. OCR translates is a technology that allows the translation of an image document into text that a machine can understand. This text can then be used to organise and categorise documents and extract relevant information from them.

Using these various document classification and information extraction techniques, documents are captured and converted to digital format. Subsequently, the images are subjected to pre-processing techniques to make them more suitable for OCR processing. OCR then converts the images into digital text, exploiting features extracted from a CNN to improve the quality of the resulting text. This process results in a more accurate and readable digital version of the text in the images. The information extracted by OCR is then used for document classification and information extraction, using specific templates for each type of document.
There were two essential requirements for the setup to be effective. The first was to obtain a sufficient number of relevant and representative documents to train the ML system. The project required around 30-40 documents of each type, and up to 50-60 documents depending on the personal circumstances of the taxpayer. The second requirement was that the documents had to follow a fixed-format business form to enable the use of templates for information extraction. However, both requirements have not been fully achieved, and the results obtained so far are only partial or limited. Obtaining tax, financial and banking documents proved challenging due to confidentiality and privacy concerns. While government forms followed prescribed formats, documents from private companies such as banks and insurance companies were in different formats, more like letters with some structured information. There were hundreds of such institutions, so a representative sample of documents was needed to study the formats and terminology.

**Complementary methods** The main method for classifying documents, at least in the long term, is through AI-based systems. However, implementing an AI solution requires a large volume of diverse documents, considerable computing power and significant resources from the organisation. In situations where these requirements are not met, complementary methods can be used, such as a lexical approach using natural language processing, rule-based approaches using keywords, and visual methods.

These methods can also be combined with hand-crafted rules and templates specific to the use case, document type or industry. While these complementary methods may not be as flexible as AI solutions, they can provide good and fast results, especially when dealing with a small number of available documents. In the case of Swiss tax documents, which have a wide variety of formats, alternative approaches such as rules based on keywords and lexical annotation using the open source tool GATE (General Architecture for Text Engineering) and the tax ontology can be explored to provide efficient solutions.

GATE is an open-source natural language processing software that provides tools and resources for document classification using predefined rules and lexical annotation. GATE can be used to classify and/or extract information from documents based on predefined rules. These rules are manually created and based on specific criteria, such as the use of keywords or regular expressions. In addition, GATE can also be used for lexical annotation, which consists of identifying key or relevant terms in documents. Finally, GATE can be integrated with a taxonomy ontology to define semantic relationships between annotated terms and ontology concepts.

Information were extracted from 43 different documents, falling under 28 document types and categorised into 19 document types, such as (ii) Salary Certificate; (iii) Copy of the notary deed for your property; (iv) Childcare costs (crèches, after-school programmes); (v) Gross rental value document for your property; (vi) Donations (organisations based in Switzerland only); (vii) Documents relating to benefits paid (sickness); (viii) Documents relating to health insurance premiums (various funds); (ix) Life insurance; (x) Maintenance costs for your property; (xi) Mortgage insurance premiums (summary/annual certificate); (xii) Second and Third pillar pension funds; (xiii) Pensions; (xiv) Profit and loss account and balance sheet for the tax year; (xv) Statement of assets and liabilities as at 31 December; (xvi) Tax return and/or identification data; (xvii) Training, retraining, conversion or reintegration costs; (xviii) Trade union dues (employment); (xix) Union dues (property).

The documents’ information was then categorised with regard to the type of document to which it belonged. This allowed the information to be sorted into the relevant categories for further analysis, “bank account documents”, “salary statements”, “health insurance”, etc.

**5.2. 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’s Tax Guide 2020 as a documentation source, describes financial and administrative documents, user profiles, and changes to profiles.

The vocabulary is structured using classes, properties and axioms, which facilitates analysis through automated deductive processes. For example, a health insurance certificate is associated with the class HealthInsurance and is defined, among other things, by at least one amount representing a health insurance premium (class...
HealthInsurancePremium) or unreimbursed health expenses (class ExpensesSicknessNotReimbursed). These two classes, together with the property amount, are used to define the following axiom:

\[
\text{amount some (HealthInsurancePremium or ExpensesSicknessNotReimbursed)}
\]

Figure 4 shows a more general view of the hierarchy of classes, properties and axioms that use these classes and properties.

The ontology is used to represent the documents and the information they contain, thus forming a knowledge base that can be used for querying and reasoning purposes. For example, Figure 5 shows how a health insurance certificate is represented. Document information represented as RDF data is realised by a data mapping process that leverages the defined ontology. More details about data mapping can be found in section 5.4.

The ontology is currently developed and contains 241 classes, 24 data type properties, 615 axioms, and 15 object properties. The ontology is designed to support document classification and tax return profiling. The ontology has also been used to associate labels with different types of documents and to create a hierarchical classification system. The ontology was implemented using the OWL ontology language and the Protégé ontology editor. Further information about the ontology can be found in [12, 13].
5.3. JSON files

JSON files have been used to represent structured information extracted from documents, and these files were processed by the mapping process (section 5.4).

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.

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.

In order to create JSON files for the extracted information, two prerequisites must be met: (i) the documents must be classified, and (ii) budget profiles must be defined.

Regarding (i), 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 (Tag) corresponding to the list of labels.

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

In relation to point (ii), the household profiles have been established by creating a list of 138 profiles. Each profile provides a clear description of a taxpayer using commonly understood language. Some of these profiles are ‘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 have been designed to be easily identifiable and understandable, which facilitates 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

<table>
<thead>
<tr>
<th>PROFILE</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>Single with property</td>
</tr>
<tr>
<td>P2</td>
<td>Single with property with children</td>
</tr>
<tr>
<td>P3</td>
<td>Single with a property with dependants</td>
</tr>
<tr>
<td>P28</td>
<td>Cohabitants owning a property without children</td>
</tr>
<tr>
<td>P29</td>
<td>Cohabitating with dependants</td>
</tr>
<tr>
<td>P31</td>
<td>Cohabitation with a self-employed person</td>
</tr>
<tr>
<td>P32</td>
<td>Cohabitation without property</td>
</tr>
<tr>
<td>P33</td>
<td>Cohabiting with children without property</td>
</tr>
<tr>
<td>P34</td>
<td>Living together without own property with children and other dependants</td>
</tr>
<tr>
<td>P36</td>
<td>Living together without a property with dependants</td>
</tr>
<tr>
<td>P38</td>
<td>Cohabitting with no children and no property</td>
</tr>
<tr>
<td>P39</td>
<td>Divorced with property</td>
</tr>
<tr>
<td>P40</td>
<td>Divorced with property with children</td>
</tr>
<tr>
<td>P41</td>
<td>Divorced with property with dependants</td>
</tr>
<tr>
<td>P43</td>
<td>Divorced with property and dependants</td>
</tr>
<tr>
<td>P72</td>
<td>Married with property and children</td>
</tr>
<tr>
<td>P74</td>
<td>Married with property with children and dependants</td>
</tr>
<tr>
<td>P135</td>
<td>Unmarried without property with children</td>
</tr>
<tr>
<td>P136</td>
<td>Unmarried with children and dependants</td>
</tr>
<tr>
<td>P138</td>
<td>Unmarried without property and with dependants</td>
</tr>
</tbody>
</table>

### 5.4. 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 it, and how to create RDF triples from it using the ontology vocabulary defined in Sect. 5.2.

The **queues** are used for communication purposes with other tasks such as information extraction or profiling. The queues are divided into input and output queues. The input data is provided by the input queues and the RDF triples generated by the mapping processor are published to the output queues. This allows 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 6.
The two main steps in the implementation of the mapping process are (i) data pre-processing, and the (ii) actual data mapping to RDF. During the pre-processing step, the input data is cleaned. The JSON files extracted during the information extraction task are cleaned so that only the information that needs to be mapped to RDF is retained. This involves removing extraneous fields to ensure that data conforms to the specific JSON schema. The purpose of this pre-processing step is to prepare the data so that it can be properly mapped to RDF. The pre-processing is done using an Apache Flink application deployed on a Flink cluster. The application reads JSON data from a Kafka topic, performs cleansing operations by removing extraneous fields, and publishes 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 is RMLStreamer, an RML processor built on Apache Flink and optimised for streaming and large data sets. RMLStreamer jobs are deployed for each document class, reading data from the corresponding output Kafka topic generated by the data pre-processing phase.

The RDF triples produced by each RMLStreamer job are written to an “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 7.

5.5. Reasoning Engine

This section discusses the development of an inference engine for (i) classifying and recognising documents submitted by users; (ii) developing rules for 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 were identified that must be present for a document to qualify and formalised them as property SHACL shapes using the properties that are 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 meet these requirements, three SHACL properties \((\text{impots:AmountShape}, \text{impots:PersonSurnameShape},\) and \(\text{impots:PersonFirstNameShape}\)) have been defined. For some of these properties, additional constraints describe the different values that can be assigned to each attribute shape, for example, the \(\text{impots:PersonSurnameShape}\) and \(\text{impots:PersonFirstNameShape}\) properties could specify that the values of the corresponding nodes must be of type \(\text{xsd:string}\) and must not be empty.

Regarding (ii) developing rules for multi-label classification of documents, multiple labels have been 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 \(\text{tag}\) was created to write triples and assign labels to documents. Thirteen different values have been defined for the property to categorise the documents, as detailed in Table 4.

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

Two types of rules are created for (iii) developing rules that classify users into different user profiles. The first type consists of customer profile rules, which infer users’ profiles by analysing the documents they have to provide. The documents provided by 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: \(\text{Document} \rightarrow \text{Profile}\)) because they go from the document provided by the user to their profile. So, if a user provides a salary certificate, his profile corresponds to the status of an employee. Table 5 describes some examples of how the documents provided by users can be used to infer their profile (direct rules).

| DOCUMENT \(\rightarrow\) 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: \(\text{Profile} \rightarrow \text{Document}\)) because they infer from the user’s profile the document that the user has provided. For example, if a person has the status of 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.
Ninety-two property forms and three 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) [14].

5.6. Data Integration

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

There are two main aspects to data integration: (i) SHACL validation and (ii) testing the effectiveness of the rules. SHACL validation language is used (i) 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. 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 one to validate that data conforms to the desired requirements. In the frame of this project, the validation of RDF data is performed by six graph validation test cases, which are related to: 3rdPillar Contribution Shape Attestation, Deposit Account Shape Attestation, Annuity AVS/AI Shape Attestation, LPP Annuity Shape Attestation Salary Certificate Shape Attestation, and Person Shape Attestation. These test cases perform SHACL constraint validation on the entire graph and compare the results with the expected validation results stored with the test case. By performing these validation tests, it is ensured that the data conforms to the expected SHACL shapes and is consistent with the rules that will be applied later.

Once the data has been validated against the SHACL forms, the next step is (ii) to test the rules to ensure that they can generate inferences from the data. This is crucial to verify that the rules are effective and can provide useful and relevant information to end users. In this project, multi-labelling rules, direct rules and inverse rules for user profiling are tested. The reasoner engine runs on TopBraid, and some inferences are shown in Figure 8 and Figure 9. For example, the system knows that Mrs. Zola Giovanna has submitted a salary statement. Based on the type of document submitted by Zola 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 Giovanna’s user profile is useful for preparing her tax return in several ways. For example, as an employee, Giovanna 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 Giovanna is an employee, the system could 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 in her tax return, and the system could help her calculate the exact amount to include in her return.

![Fig. 8. Example of some inferences in TopBraid](https://www.w3.org/TR/shacl/)

On the other hand, Ladoumegue Jules has declared himself as 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 to Mrs. 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 novelty of the proposed intelligent DMS lies precisely in the fact that it offers a capability that is not present in all current DMS systems, namely the ability to infer the user’s profile from the uploaded documents and vice versa using SHACL rules. Current DMS systems typically focus on document management, organisation, and archiving.

6. Use case: A household with working parents and children

The intelligent DMS system used in this scenario is designed to help a household with working parents and children manage their documents for the preparation of the family’s income tax return. Below is an example use case that illustrates how the intelligent DMS system can help with document management:

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 1. The following JSON file represents the results of extracting information from a tax document relating 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 tax document related to salary. The field “id_foyerFiscal” represents the fiscal identification number of the household to which the employee belongs. The “extractedFields” field 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.
Once the information extraction is complete, the data is sent to the mapping module, which outputs it as RDF triples. An example of such triples is provided by Listing 2.

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, made up of the person represented by the “impots:Personne_Zola_Giovanna” object.
Once the 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 the person Giovanna Zola is considered an employee (impots:Salarie) who has to provide several documents, such as the health insurance (impots:AAssuranceMaladie) and the salary statement (impots:CSalaire). As far as the multi-label document is concerned, the salary certificate (impots:CSalaire) is labelled with two tags: “Impot” and “Revenu”.

Fig. 10. Results of applying the profile classification and document labelling rules to the mapped data

7. From Proof-of-concept to implementation

To design the intelligent DMS, various components have been implemented. The solution consists of seven independent modules that interact with each other. Each module is based on different technologies and has different architectural and technological requirements. They are 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 (IE).** The module consists of sub-projects that perform (i) document classification on image documents, (ii) IE from image documents using templates, and (iii) generation of image documents for a document category/class (data augmentation). The module is made available as source code at https://gitlab.unige.ch/addmin/ImageDocumentsAnalysis.
- **Mapping rules.** The 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

---

```sparql
@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_5677777
  a impots:FoyerFiscal ;
  impots:composeDe impots:Personne_Zola_Giovanna .
```

Listing 2: An example of the information extracted from a Salary certificate represented using RDF
The following phases are Profiling and Reasoning, which are also done manually. Both rely on SHACL rules to infer new data and, if it does not, describes any mismatch. In this way, SHACL can be used to validate that data conforms to the desired requirements.

These seven components described above are integrated to form a complete system. Much of the process shown in Figure 1 and Figure 2 is automated, but there are still a few human-in-the-loop steps. The process starts with document classification and IE. This is a semi-automated process as there are tasks that need to be done manually. The IE is based on templates that have to be drawn. User feedback is required to correct any 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 (Figure 11) remains to be completed.

The IE 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 is published to a Kafka topic named with the UUID of the document used. Once the information extraction is complete, this event is reported as a \texttt{ie\_end} event message published to a RabbitMQ queue named \texttt{events}. Figure 11 illustrates such interactions. The \texttt{ie\_end} event message published in the queue is consumed by the IE Merger component, which is 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 is published to a Kafka topic called \texttt{rml\_streamer\_in}, ready to be cleaned up and then mapped to RDF. The sequence of these operations is shown in Figure 12. If no merge is required, the data is immediately published to the \texttt{rml\_streamer\_in} topic by the information extraction component.

Data mapping is followed by a data validation process. During this step, the RDF data is validated against the defined SHACL forms (see Section 5.6). This phase is still done manually by importing the data into TopBraid Componser and running the SHACL validation. An alternative to importing the data into TopBraid Componser is to use the SHACL API (https://github.com/TopQuadrant/shacl) in a Java application that reads the data and runs the validation rules against it.

The following phases are Profiling and Reasoning, which are also done 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 is loaded into TopBraid Componser, the inference engine is started and the rules are executed on the existing data. An example of the results is shown in Figure 8 and Figure 9.

\footnotesize{\textsuperscript{4}registry.gitlab.unige.ch/addmin/ie-merger:latest
\footnotesuperscript{5}registry.gitlab.unige.ch/addmin/uuidgenerator:latest}
8. Conclusion

The article presents a full-fledged solution to an intelligent semantic-based document management system (DMS). 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, by 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 two alternative approaches for document classification and information extraction; (iii) definition of document schema and tax household profiles; (iv) data mapping from JSON data to RDF triples exploiting the ontology; (v) development of a reasoning engine, that through SHACL shapes and rules, allows to classify documents, multi-label them, and
classify user profiles; (vi) integration of RDF triples and rules. The steps were implemented through modules that are publicly available (on request) 6.

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

The limited availability of Swiss tax documents posed 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. Despite the challenges encountered, a pilot solution is proposed and validated through a use case that can be readily used by trustees or insurance professionals.

Still, the proposed approach and implementation present a few limitations. Although 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. 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 laws and regulations compliance.

Acknowledgement

This research was supported by Innosuisse in the framework of the innovation project 50606.1 IP-ICT entitled “Addmin – Private computing for consumers’ online document access”.

References


6See https://gitlab.unige.ch/addmin


