An LLM-Based Assistant for Debt Waiver Court Procedures

Lluís Padró⋆, Daniel Ferrés†, Roser Saurí⋄, Mireia Artigot†
⋆Universitat Politècnica de Catalunya, ⋄Process Talks S.L., †Universitat Pompeu Fabra
lluis.padro@upc.edu, daniel.ferres@upf.edu, roser@processtalks.com, mireia.artigot@upf.edu

Abstract

Spanish Insolvency Law 1/2020 of the 5th of May enables individuals to apply for debt waiver under certain conditions. The large number of applications submitted each year places a heavy burden on judges and court officers, who must examine heterogeneous documentation before issuing a ruling. This paper presents an AI-based assistant designed to support the processing of debt waiver cases. The system integrates PDF-to-text conversion, rule-based document classification, large language model (LLM)-based information extraction, and post-processing to consolidate fragmented or duplicated records. A front-end interface provides structured summaries of the application content, and can automatically generate draft rulings. Evaluated on a set of real applications, the system achieves over 92% F1 in document classification and up to 91% F1 in personal data extraction, showing the potential of open-source LLMs to reduce administrative workload and accelerate judicial procedures, while keeping the final decision with the judge.

1. Introduction

Many individuals face situations where they cannot meet their credit obligations. Ongoing economic transformations have increased financial instability and personal uncertainty, creating the need for a legal "fresh start" mechanism allowing defaulting debtors to move forward. The European Union addressed this need in Directive 2019/1023 (EU Council, 2019). Spain incorporated this mechanism into national law in Royal Legislative Decree 1/2020 of May 5th on Insolvency (Spanish Government, 2015, 2020) (henceforth, RLD 1/2020).

Under this framework, a debtor may apply for a debt waiver during insolvency proceedings when certain conditions –such as a lack of assets and the presence of debts that can be legally discharged– are met. Since the introduction of this mechanism, the number of applications reaching Spanish courts has grown dramatically: According to the Spanish Judges General Council (Consejo General del Poder Judicial, 2025), it raised from 12,031 cases in 2020 to 57,507 in 2024, an increase over 40% per year. Although the procedure is relatively simple and allows only limited judicial discretion, its sheer volume has become difficult to manage.

This creates a clear opportunity for Artificial Intelligence (AI) to support the courts. Automatization of routine tasks such as document classification and information extraction can help judges and court officers access key facts more quickly and issue rulings with less administrative effort.

To address this challenge, we present an AI-based judicial assistant[1] that classifies the documents submitted in the debtor's application, extracts relevant information, and generates a structured summary of the case. The system can also draft a proposed resolution for judicial validation. By providing accurate and well-organized information, our approach aims to reduce the administrative burden on court staff while ensuring that final decisions remain under judicial control.

[1] Further information about the AI-based assistant is available at: `https://www.ai-insolvency.com`

2. Related Work

Court administration and decision making are fertile areas for the use of AI. Existing work usually addresses prediction of legal judgment and the retrieval of precedents. Cui et al. (2022) present a detailed survey on the topic.

Anonymization is another active research field, which has become essential for publishing court records responsibly and in compliance with the requirements of the General Data Protection Regulation (GDPR). Approaches range from rule-based pipelines (Niklaus et al., 2023) to transformer-based de-identification models such as GiusBERTo (Salierno et al., 2024).

Argument mining and explainability are also emerging topics. Al Zubaer et al. (2023) and Chlapanis et al. (2024) develop datasets for extracting argumentative components from judgments, allowing to augment document management systems with reasoning-centric metadata, or to develop intelligent search engines for assisting public defenders.

However, while predictive and retrieval-oriented AI attract academic attention, document management and information extraction are the most court-ready applications. Courts process large volumes of filings and rulings, making structured access to documents and their content critical. Early work by Nallapati and Manning (2008) demonstrated automated docket-entry classification for U.S. federal courts. More recent initiatives apply named-entity recognition (NER) and schema-based extraction to enrich metadata: Leitner et al. (2020) provide German legal NER corpora; Mathis (2022) extends

Proceedings of the Fifteenth Language Resources and Evaluation Conference (LREC 2026), pages 505–514
11–16 May 2026.

this with entity extraction for proceedings data; and Bellandi et al. (2024) propose an entity-centric infrastructure for managing court judgments at scale.

Practical adoption is steadily growing. For instance, the U.S. National Center for State Courts piloted NLP tools for civil filings, highlighting efficiency gains in triage and routing (Hannaford-Agor, 2023). In the UK context, Sargeant et al. (2025) show that large language models can accurately classify case topics, enabling automated docket organization. Complementary work by Mali et al. (2024) captures structured events from textual decisions, facilitating workflow integration.

See Ariai and Demartini (2024) for a comprehensive survey on AI for judicial applications.

Our proposal goes one step further by leveraging large language models for advanced information extraction to filter and normalize data needed for deciding debt-waiver applications, and by generating draft resolutions that reduce the workload of court staff while preserving judicial oversight.

3. System Overview

We present an AI assistant designed for assisting court officials, law clerks, and judges in fresh start insolvency proceedings of natural persons with no assets (regulated in articles 486 to 502 of RDL 1/2020). This proof of concept is developed within the EU project JuLIA[2], with the aim of assisting in managing the court's workload.

3.1. Design Goals

The original goals of the system were as follows:

- Verify that the legally and judicially required documents under RLD 1/2020 are filed with the debt waiver application.
- Classify them by their type (see Section 3.2).
- Extract document information: personal data, debt list, asset list, tax information.
- Classify each debt as waiveable or not, according to conditions defined by RLD 1/2020.
- Calculate the applicant's total asset value.
- Use the extracted information to suggest a ruling and provide it to the judge for validation.

However, after analyzing the dataset and observing the variability and often partial or scattered nature of the information across multiple documents as well as the heterogeneous interpretation of some legal provisions by judges, the last three items above were reformulated as:

- Extract the information (if any) about each of the debts and present it to the judge.
- Extract the information (if any) about each of the assets and present it to the judge, so that he/she can decide if it should be included in the applicant's total asset value,
- Extract relevant information from tax return forms (e.g., whether the applicant has children or whether any of the members of the household has a disability) that could be used by the judge when determining whether debt waiver should proceed forward.

Additional non-functional requirements include:

- No commercial LLMs may be used, for data privacy reasons. Used models must be open-access, and run on self-hosted hardware, which conditions the size and capabilities of the potentially usable LLMs.
- Input documents may include scanned documents with bad quality, which may result in extraction errors due to low OCR or visual model performance. Thus, original documents must be easily accessible to the final user for data verification.

3.2. Dataset

The dataset comprises all documents from 185 fresh start judicial proceedings that have been completed by the courts, with each case containing between 10 and 50 documents. The files were provided by the Catalan Department of Justice and Democratic Quality, upon judicial authorization, and duly anonymized by an independent specialized company - not part of the project - under the requirements set by the Spanish General Council of the Judiciary (CGPJ).

The dataset was split into three parts: 10 applications were reserved to extract the examples for the few-shot prompts, 82 applications were used as the development dataset, and 93 were kept to be used as the final test set.

For each application, a manually annotated gold-standard json structure was created, representing the information expected to be extracted. In addition, all of the documents were classified by hand in one of the classes listed below, or as "Others" if none applied.

Under RLD 1/2020, each application for a debt waiver in a fresh start proceeding has to include, at least, the following documents:

- **Form or Petition:** Either a form with personal data, debts to be waived, assets owned, etc., or a memo laying down that information in free text. Although only one of these is requested, most applicants include both.

[2] https://www.julia-project.eu/

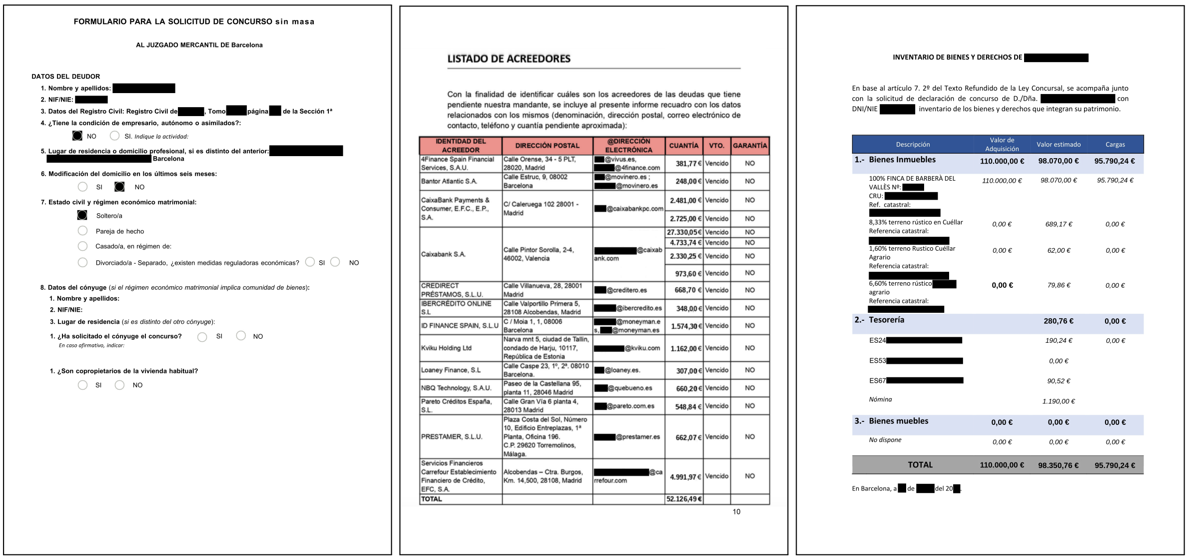

Figure 1: Document examples (anonymized): Form (left), list of creditors (center), list of assets (right).

- **Statement:** A text describing the applicant's personal situation, the reasons that led to their debt, actions taken so far to meet credit obligations, whether any of the debts are subject to court enforcement proceedings, etc.
- **Debt list:** including: name and contact details of each creditor, description and amount of the debt, existence of collateral.
- **Asset list:** List of income sources, bank accounts, labor contracts, vehicles, real estate, or any other income received or asset owned by the debtor.
- **Tax returns** from the last 3 years.
- **Identity card**.
- **Power of Attorney**.
- **Official criminal record**.

Despite this minimal set of required documents, applicants (or rather, their attorneys) often include many additional documents and information in their application –such as pay slips, bank statements, mortgage contracts, etc.– making it cumbersome to filter out the relevant information. The variability of the content of each application and the heterogeneity resulting from each lawyer's practices and writing style results in a rather challenging document classification.

Additionally, most applications include incomplete or scattered information: Debt lists often omit the nature of the debt (mortgage, credit card, taxes, etc.) or whether there is any collateral (real estate, a vehicle, etc.) securing it. Moreover, the list of assets may omit important details, such as whether the ownership of an asset (e.g., an apartment) is shared, and therefore not all of its value can be used to secure the debt. This forces court officials to sift through many documents to process the application.

Figure 1 shows an anonymized sample of some of the processed documents, and Figure 2 presents an example of the json expected after processing a whole application.

3.3. System Architecture

The system consists of a back-end component that performs the document classification and information extraction tasks, and a front-end that presents the information to the user in a convenient format, keeps track of the status of the application, and generates drafts of court documents that can be directly accepted or edited by the user.

The back-end information extraction system consists of four main components: a pdf-to-text converter, a document classifier, the actual information extraction module, and a post-processing module that filters and merges redundant or complementary information.

3.3.1. PDF Conversion

All documents are in PDF format but contain different types of content, such as textual data, tables, and, in some cases, images of scanned documents (e.g., tax returns, ID cards, etc.).

In order to process them either by the classifier or by the information extractor, they all first need to be converted into plain text. For that, we used open-source, state-of-the-art PDF-to-text conversion libraries: `unstructured`[3] (both in plain and OCR mode), `pdftotext`[4], and `PyMuPDF4llm`[5].

[3] https://unstructured.io/
[4] https://pypi.org/project/pdftotext/
[5] https://pypi.org/project/pymupdf4llm/

```
{
  "formulario/demanda": {
    "extracted": [
      {
        "ID": [redacted],
        "direccion": [redacted], Barcelona",
        "nombre": [redacted],
        "estado_civil": "soltera"
      }
    ]
  },
  "inventario_bienes": {
    "extracted": [
      {
        "valor": 98070.0 ,
        "identificador": "FINCA DE [redacted]",
        "tipo": "inmueble",
        "recurrente": false
      },
      {
        "valor": 689.17 ,
        "identificador": "8,33% terreno rustico en [redacted]",
        "tipo": "inmueble",
        "recurrente": false
      },
      {
        "valor": 62.0 ,
        "identificador": "1,60% Terreno rustico en [redacted]",
        "tipo": "inmueble",
        "recurrente": false
      },
      {
        "valor": 79.86 ,
        "identificador": "6,60% terreno rústico [redacted]",
        "tipo": "inmueble",
        "recurrente": false
      },
      {
        "valor": 190.24,
        "identificador": "ES24 [redacted]",
        "tipo": "cuenta bancaria",
        "recurrente": false
      },
      {
        "valor": 0.0,
        "identificador": "ES53 [redacted]",
        "tipo": "cuenta bancaria",
        "recurrente": false
      },
      {
        "valor": 90.52,
        "identificador": "ES67 [redacted]",
        "tipo": "cuenta bancaria",
        "recurrente": false
      },
      {
        "valor": 1190.0 ,
        "identificador": "n/a",
        "tipo": "salario",
        "recurrente": true
      }
    ]
  },
  "lista_acreedores": {
        "extracted": [
      {
        "importe": 381.77,
        "e-mail": "[redacted]@vivus.es, [redacted]@4finance.com",
        "direccion": "Calle Orense, 34 - 5 PLT, 28020, Madrid",
        "nombre": "4Finance Spain Financial Services, S.A.U."
      },
      {
        "importe": 248.0 ,
        "e-mail": "[redacted]@movinero.es : [redacted]@movinero.es",
        "direccion": "Calle Estruc, 9, 08002 Barcelona",
        "nombre": "Bantor Atlantic S.A."
      },

          (...)

      {
        "importe": 1162.0 ,
        "e-mail": "[redacted]@kviku.com",
        "direccion": "Narva mnt 5, Tallin, Harju, 10117 Estonia",
        "nombre": "Kviku Holding Ltd"
      },
      {
        "importe": 662.07,
        "e-mail": "[redacted]@prestamer.es",
        "direccion": "Plaza Costa del Sol, Número 10, 12 Planta, Oficina 196. 29620 Málaga.",
        "nombre": "PRESTAMER, S.L.U."
      },
      {
        "importe": 4991.97 ,
        "e-mail": "[redacted]@carrefour.com",
        "direccion": "Alcobendas – Ctra. Burgos, Km. 14,500, 28108, Madrid",
        "nombre": "Servicios Financieros Carrefour Establecimiento Financiero de Crédito."
      },
      {
        "importe": 52126.49 ,
        "nombre": "TOTAL"
      }
    ],
    "suma": 52126.49
  },
  "declaraciones_renta": {
    "extracted": [
      {
        "declarante_1": [redacted],
        "declarante_2": "",
        "hijos": false,
        "opción_de_tributación": "individual",
        "base_imponible": 7812.64 ,
        "ejercicio": 2020
      },
      {
        "declarante_1": [redacted],
        "declarante_2": "",
        "hijos": false,
        "opción_de_tributación": "individual",
        "base_imponible": "no presentada",
        "ejercicio": 2021
      },
      {
        "declarante_1": [redacted],
        "declarante_2": "",
        "hijos": false,
        "opción_de_tributación": "individual",
        "base_imponible": 18972.23 ,
        "ejercicio": 2022
      }
    ]
  }
}
```


Figure 2: Example of expected json for application in Figure1.

Given the diverse nature of the documents, a hybrid strategy has been implemented:

Document classification: We use `pdftotext` to convert the first page of the PDF document to text. If this conversion fails (no text or only very short or gibberish text is obtained), the `unstructured` in OCR mode is applied instead. The resulting text is then used to extract features for the classifier.

Information extraction: Taking place after classification. For that, the pdf-to-text conversion strategy is adapted based on document type:

- **Documents of type Form, Petition, or Statement**: Converted using `unstructured` in OCR mode, since they often contain scanned images of fill-in forms or similar.
- **Tax return documents**: First converted using `pdftotext` and, in case of failure, `unstructured` in OCR mode.
- **Documents containing debt or asset lists**: Processed with *both* `pdftotext` and `unstructured` in OCR mode. The former does a good job at extracting text strings (unless the document is a scanned image), while the latter is better at detecting the document layout, because these types of document contain mostly tables. Then, both results are sent to an LLM with the request of combining them into a single structured table.

3.3.2. Document Classification

The goal of the classifier is to label each document with one of the relevant classes described in Section 3.2, or with a default label *Others*.

The approach used is based on a classical feature-vector representation of the documents. The features encode the presence of selected keywords, phrases, or patterns, either in the document file name or in the document content. There are a total of 34 different features, each corresponding to a regular expression that matches a certain set of keywords, phrases, or phrase patterns. These features range from simple checks like whether the file name includes the word *IRPF*[6] to more sophisticated checks such as whether the first page of the document includes a line matching a regex like for example:

```
(formulario|impreso).*solicitud de concurso
```

Each feature is assigned a weight for each possible label, and the label obtaining the highest score given the active features in the target document is selected. Although ideally the feature weighting would be done with a machine learning algorithm such as Support Vector Machines or Maximum En-

[6] IRPF is the acronym for tax returns in Spain

tropy, the small size of the available dataset was not enough to train such a model, so, the feature weights were assigned manually on an iterative process over the development dataset.

Despite this simple approach, the file names and document content are consistent enough to allow the classifier to reach a performance over 95% F_1 (see Section 4).

3.3.3. Information Extraction

Once documents are classified, relevant information is extracted from each document in one of the targeted classes. Extraction is performed by an LLM, using a specific prompt and a set of few-shot examples tailored for each document type.

Given the sensitive nature of the data, the use of third-party LLM providers was not allowed, and only open-source, in-house models were employed. A further constraint was the available computing power, limited to an RTX 4090 GPU with 24 GB of memory. We experimented with Mistral-v0.3-7B (Jiang et al., 2023) and with Llama-3.1-8B (Grattafiori et al., 2024), finally selecting the latter given its much larger context window. The larger window was necessary to process some applications where the few-shot document examples were lengthy, resulting in prompts that exceeded Mistral's maximum context length.

The information extracted from each document type is the following:

- **Debtor's personal data** (name, ID number, address, marital status): Extracted from documents of type Petition, Form, and Statement. The needed data may be scattered across those different documents within the same application. Therefore, it is searched in all of them and the results obtained are combined afterwards (see Section 3.3.4).
- **Debt list**: Extracted from documents classified as such, which usually contain tables with a row for each debt. In many cases, the tables contain multirow or multicolumn cells, which pose a challenge to both the OCR and the LLM performing information extraction.
  Data extracted for each debt are: creditor's name, address, and e-mail; debt type and amount; reason or description of the debt. Not every debtor provides all these defining variables so extraction is often partial.
- **Asset list**: Extracted from documents classified as such, which typically contain tables with a row for each asset. The same problems as with debt lists (multirow/multicolumn cells or partial information) are observed in this case. Extracted fields here are: asset description, its type (vehicle, real estate, salary, etc.), its value, and whether it is a recurrent income or a static asset.
- **Last-3-year tax returns**: Processed to extract, for each fiscal year: year, taxable income, whether the debtor has children, whether the return is single or joint, and in the latter case, the spouse's name.

Pre-processing

Before being fed to the extraction LLM, text documents obtained from PDF files need to be adjusted, depending on the type of document:

- **Forms** are cut after the debtors' address because subsequent information in the document is not relevant for extraction. This reduces the context length, improving efficiency and extraction accuracy.
- **Statements and Petitions** are cut at a fixed length of 2,500 and 5,000 tokens respectively, since the requested information is usually in the first pages of the document.
- **Debt and asset lists** are cut at a fixed length of 7,000 and 5,000 tokens respectively, which typically grasps the relevant tables while avoiding to feed the LLM with certificates, bank extracts, or further supporting data that are often included in the document.
- **Tax return documents** are the most challenging. They are very long and cannot be used in a few-shot scenario given that they fill up the LLM context and/or the available GPU capacity. Thus, a set of regular expressions and context windows around them is used to extract parts of the documents where the searched information is expected to be. A shorter document is created with these parts collected and fed to the LLM.

Few-shot Prompts

For each document type –and corresponding gold extraction information– the appropriate shortened text is used to create a prompt for an LLM, consisting of:

- A *system* utterance describing the required task, the information to extract, applicable constraints (e.g. do not extract spouse's name if tax return is individual), and how to deal with special cases (missing information, wrong document, etc).
- A sequence of 3 to 8 solved task examples. Each example includes:
  - A *user* utterance requesting to extract the needed information, with less details than in the *system* utterance.

- A chunk of text of the same type of document.
- The json structure to be extracted for that document.

The examples used to create the few-shot prompt for each type of document were selected from a pool of 10 hand-annotated applications, preprocessed (pdf-to-text conversion + text shortening) with the criteria described above.

3.3.4. Post-processing

The LLM extracts information from all documents of relevant types, which often results in duplicate records. For example, personal data is searched for in documents of type Form, Petition, and Statement. Not all documents contain all information in all applications, so the result is usually three different records with partial information that must be unified into a single entry.

Creditor and asset lists are also often found across multiple files, with information repeated but sometimes differing –for example, variations in creditor names or addresses, or differing asset descriptions. Thus, each extracted record is compared with all previously processed records and either merged with an existing one or added as a new one, depending on a similarity metric.

The record similarity metric used combines the number of matching field names, and the similarity of their values, which is computed differently for each field type: numeric fields (asset values, credit amounts, etc.) are matched strictly, while names and addresses are compared using a weighted combination of Levenshtein string edit distance and word-wise Jaccard or Overlap similarities. In addition, each field has a scaling factor that increases or decreases its contribution to the total similarity of the compared records.

3.3.5. User Interface

All the information extracted by the backend is displayed in an intuitive, well-organized layout, which allows the judge or court official to find the relevant information easily.

Figure 3 shows the layout presented to the user. The left column shows the documents uploaded, each with the class assigned by the classifier. If some required document is missing, the system issues a warning and offers the option to draft a court order (in Spanish, *Diligencia de Ordenación* or depending on the court management, *Providencia judicial*) requesting the applicant to provide the missing documents to amend the application.

In the center, the extracted information is presented: the applicant's personal data (name, address, ID number, marital status), the list of assets (each with description and value), the list of outstanding debts (each with creditor name, amount, and description), and the income received in the previous three year tax returns.

Finally, on the right-hand side, the interface presents buttons to draft a court order admitting or rejecting the application, the notice of that order to the parties and the court order completing the judicial proceeding and waiving the debtor's waivable debts. This order, when notified to creditors, completes the debt waiver judicial procedure. Against this final decision, the applicant or the creditors may appeal before higher courts.

4. Experiments and Results

A set of experiments were conducted to establish the best prompt configuration for the extraction process for each type of document.

4.1. Experimental Setup

The dataset was divided into three parts: 10 applications were used as examples for the few-shot prompts; 82 applications were employed as development set, to explore the best example combination for each document type; and finally, 93 applications were reserved for final test set once the best configuration had been established.

For each document type, over 150 random subsets of the 10 example documents were used to create a few-shot prompt, and each combination was evaluated against the development dataset.

The *system* and *user* utterances of each prompt were also iteratively refined to improve results on the development dataset. Note that there is a different prompt for each document type, requesting the model to extract different information, and providing a different set of few-shot examples. Finally, the best few-shot example subset for each document type was used to create the final set of prompts, to be evaluated on the test dataset.

The classifier was evaluated computing precision and recall of per-document predicted class, regarding class "Others" as "no-class".

Information extraction output was evaluated at two levels: field level, and application level.

Field level. For each type of document, Precision, Recall and F1 at the field level are computed. Precision is the ratio of correctly extracted values over the number of extracted values, Recall is the ratio of correctly extracted values over the number of expected values, and F1 is the harmonic mean of Precision and Recall.

The criteria for determining whether an extracted field value is correct depend on the type of values being compared. For numeric fields (such as asset

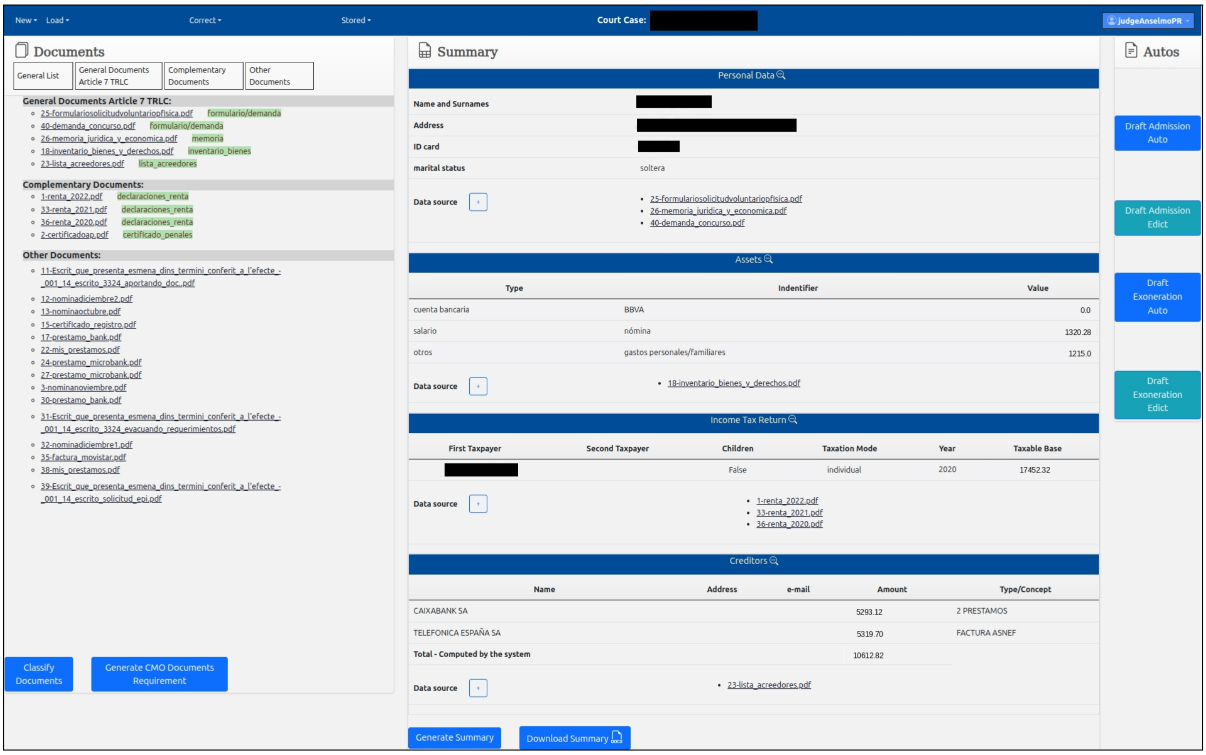

Figure 3: Front-end application layout presented to the user.

values or debt amounts), a strict numerical comparison is applied. However, for textual fields like names and addresses, a more flexible comparison is used: being extracted by an LLM, variations may occur in capitalization, punctuation, spacing, word order, etc., making strict string equality checks overly rigid. Therefore, comparisons are performed on normalized versions of the values (lowercased, with punctuations removed) and several distance metrics are used (Levenshtein string distance, Jaccard/overlap word set distance). An acceptance threshold is set, below which the answer is considered correct.

Another evaluation challenge concerns the lists of assets and debts. The order in which the LLM extracts list elements is not necessarily the same as that found in the gold standard. Thus, an alignment procedure must be performed, for which we use the Hungarian algorithm (Kuhn, 1955) and a distance metric for the whole element (debt or asset). This metric is based on the number of coincident fields and the similarity of their values with the same rules described above.

Application level. We also computed the average F1 for each application in order to evaluate whether errors are clustered in a few applications or are more evenly scattered.

4.2. Results

Tables 1 and 2 show the results for the classifier and the field-level scores for the best performing few-shot examples subsets. Figure 4 shows the distribution of applications according to their individual F1 score.

	Prec.	Recall	F1
Development	92.7	97.9	95.3
Test	90.3	94.0	92.1

Table 1: Classifier results.

Table 2 shows the combinations of few-shot examples that produce the best F1 results for each type of document on the development data set. Rows labeled *BEST* show the results of extracting data from each document type with the best example combination for that type.

Note that the results are not always identical to the original run (e.g. *personal data* achieves 92.7% F1 both in the exploration run and when that example set is used in the *BEST* configuration, but *asset list* achieves 85.2% F1 on the exploration run, but 84% when used in the *BEST* run. This is due to the inherent randomness of LLMs: Even with low temperature settings, LLM outputs vary slightly between runs, thus, in different runs of the same *BEST* configurations, we would obtain different results, which may vary in a range over one percentage point.

Few-shot examples	Personal data			Asset list			Creditor list			Tax return			Average		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
1-3-4-5-6-8-10	92.7	92.7	**92.7**	78.7	80.4	79.6	86	75.8	80.6	100	0	0	85.7	55.5	67.4
1-2-3-6-7-8-10	90.3	91.2	90.7	89.1	81.6	**85.2**	84.9	74.6	79.4	100	0	0	86	54.9	67
1-2-5-6	88.9	90.5	89.7	80.4	78.3	79.3	85.2	80.5	**82.8**	93.3	86.9	90	87.3	82.8	85
1-2-4-5-6	89.5	91.2	90.3	83.1	78.5	80.7	85.3	78.2	81.6	95.1	87.8	**91.3**	88.2	81.8	84.9
BEST (devel)	92.7	92.7	92.7	86.0	82.1	84.0	85.9	78.5	82.0	95.1	87.8	91.3	89.0	82.4	85.5
BEST (test)	88.8	93.5	91.1	73.9	76.2	75.0	85.8	70.3	77.3	92.1	72.9	81.4	86.3	72.9	79.0

Table 2: Extractor field-wise results for each type of document.

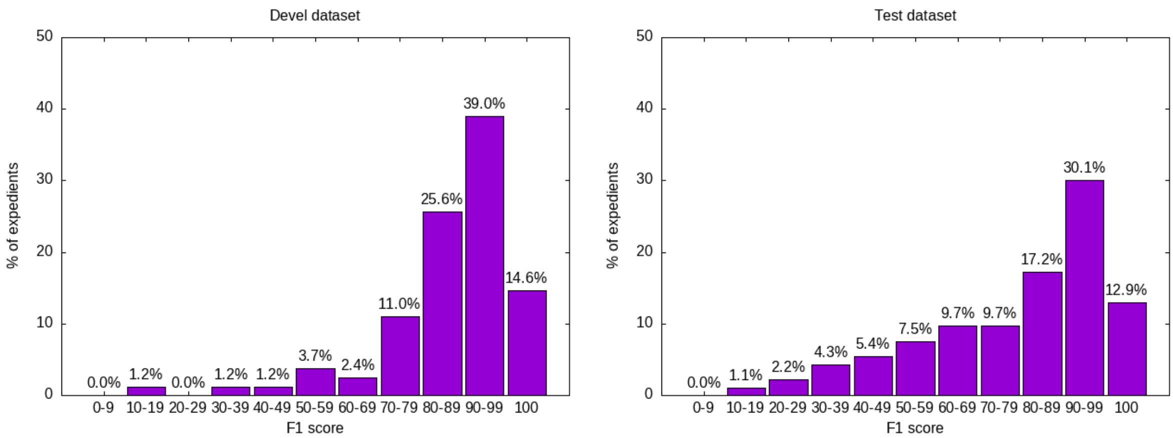

Figure 4: Distribution of F1 score over all applications for development (left) and test (right) dataset.

Figure 4 shows the distribution of the F1 scores among the applications used. In the development dataset, 14.6% applications had a perfect score (that is, all fields correctly extracted, with no additional spurious information), and 79.2% had an F1 of at least 80%. In the test dataset, the amount of perfect applications is 12.9% and the amount of applications over 80% F1 drops to 60%.

The performance drop when moving from development to test dataset may be explained by several factors. First, there is the obvious fact that the prompts and few-shot examples were optimized on the development dataset, so some performance loss is to be expected. Second, a relevant factor is the lower performance of the classifier: since extraction relies on the classifier output, a misclassified document will not be processed by the extractor, leading to a loss of recall (as it can be seen in Table 2, the drop in recall is much larger than in precision). Finally, the applications in the development dataset were more thoroughly selected, discarding very noisy documents or infrequent types of applications (e.g. companies or married couples applying for a debt waiver, with 95% of the cases being submitted by individuals), while the test dataset was cleaned less thoroughly, so some noisy applications remain, posing additional difficulties to the model.

5. Conclusion and Further Work

This paper presents an AI-based assistant developed to simplify and accelerate the processing of debt exoneration applications under RLD 1/2020. The system relies on LLMs to extract relevant information from documents and present it to the judge or court official in a structured manner. A front-end interface enables judges and court officers to review the extracted content and generate draft rulings, thereby reducing repetitive clerical work while preserving judicial oversight and decision-making.

Evaluation on real applications demonstrates that the system achieves over 92% F1 in document classification and up to 91% F1 in personal data extraction, with competitive results for asset and debt lists. These findings confirm the potential of self-hosted open-source LLMs to support court procedures in a practical and efficient manner.

Future work will focus on three directions. First, conducting user studies with court officials and judges to assess the usability and perceived reliability of the system. Second, improving PDF-to-text conversion and exploring larger extraction models, which may enhance performance on complex documents. Third, leveraging the collected data to develop predictive models capable of providing reasoned recommendations on debt waiver outcomes, further assisting judicial decision-making while ensuring transparency and accountability.

Ethical Considerations

All personal data has been used with the authorization of the judicial authorities. All case files were stored on a single secure server, no copies have been made, no local work has been conducted and access to the server has been secured via VPN and strict firewall rules. Numerical data has been modified in order to prevent re-identification.

Acknowledgments

This work was carried out within the framework of the JuLIA project, funded by the Justice Programme of the European Union (101046631, JUST–2021 JTRA), Artificial Intelligence, automated decision making and justice systems CNS2023-145543 Consolidación Investigadora (2023) and Protecting consumers in digital platforms PID2021-127258NA-I00/AEI/10.13039/501100011033. The authors thank the Scientific Computing Core Facility (MELIS-UPF) for their support in the configuration of the computing infrastructure.

References

Abdullah Al Zubaer, Michael Granitzer, and Jelena Mitrović. 2023. Performance analysis of large language models in the domain of legal argument mining. *Frontiers in Artificial Intelligence*, Volume 6 - 2023.

Farid Ariai and Gianluca Demartini. 2024. Natural language processing for the legal domain: A survey of tasks, datasets, models, and challenges.

Valerio Bellandi, Christian Bernasconi, Fausto Lodi, Matteo Palmonari, Riccardo Pozzi, Marco Ripamonti, and Stefano Siccardi. 2024. An entity-centric approach to manage court judgments based on natural language processing. *Computer Law & Security Review*, 52:105904.

Odysseas Chlapanis, Dimitris Galanis, and Ion Androutsopoulos. 2024. Lar-echr: A new legal argument reasoning task and dataset for cases of the european court of human rights. pages 267–279.

Consejo General del Poder Judicial. 2025. Efecto de la crisis en los órganos judiciales. https://www.poderjudicial.es/cgpj/es/Temas/Estadistica-Judicial/Estudios-e-Informes/Efecto-de-la-Crisis-en-los-organos-judiciales.

Junyun Cui, Xiaoyu Shen, Feiping Nie, Zheng Wang, Jinglong Wang, and Yulong Chen. 2022. A survey on legal judgment prediction: Datasets, metrics, models and challenges.

EU Council. 2019. Directive (EU) 2019/1023 of the European Parliamment and of the Council. https://eur-lex.europa.eu/eli/dir/2019/1023/oj/eng.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, and 557 other authors. 2024. The llama 3 herd of models. https://arxiv.org/abs/2407.21783.

Paula L. Hannaford-Agor. 2023. Use of natural language processing (nlp) in civil case management: A report on three proof of concept projects.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, and 14 other authors. 2023. Mistral 7b. https://arxiv.org/abs/2310.06825.

H. W. Kuhn. 1955. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97.

Elena Leitner, Georg Rehm, and Julian Moreno-Schneider. 2020. A dataset of German legal documents for named entity recognition. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 4478–4485, Marseille, France. European Language Resources Association.

Drish Mali, Rubash Mali, and Claire Barale. 2024. Information extraction for planning court cases. In *Proceedings of the Natural Legal Language Processing Workshop 2024*, pages 97–114, Miami, FL, USA. Association for Computational Linguistics.

Bruno Mathis. 2022. Extracting proceedings data from court cases with machine learning. *Stats*, 5(4):1305–1320.

Ramesh Nallapati and Christopher D. Manning. 2008. Legal docket-entry classification: where machine learning stumbles. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP'08, page 438–446. Association for Computational Linguistics.

Joel Niklaus, Robin Mamié, Matthias Stürmer, Daniel Brunner, and Marcel Gygli. 2023. Automatic anonymization of Swiss federal Supreme Court rulings. In *Proceedings of the Natural Legal Language Processing Workshop 2023*, pages 159–165, Singapore. Association for Computational Linguistics.

Giulio Salierno, Rosamaria Bertè, Luca Attias, Carla Morrone, Dario Pettazzoni, and Daniela Battisti. 2024. Giusberto: A legal language model for personal data de-identification in italian court of auditors decisions.

Holli Sargeant, Ahmed Izzidien, and Felix Steffek. 2025. Topic classification of case law using a large language model and a new taxonomy for uk law: Ai insights into summary judgment. *Artificial Intelligence and Law*.

Spanish Government. 2015. Ley 25/2015, de 28 de julio, de mecanismo de segunda oportunidad, reducción de la carga financiera y otras medidas de orden social [Law 25/2015, on the second chance mechanism, reduction of financial burden and other social measures]. Official State Gazette (BOE) No. 180 of 29th of July 2015.

Spanish Government. 2020. Real Decreto Legislativo 1/2020, de 5 de mayo, por el que se aprueba el texto refundido de la Ley Concursal [Royal Legislative Decree 1/2020 of 5 May 2020 approving the Consolidated Text of the Insolvency Law]. Official State Gazette (BOE) No. 127 of 7th of May 2020.