TEXT MINING EXAM GUIDE

1. Introduction to the exam project

This exam project aims to guide students in the conceptual design and realization of Retrieval-Augmented Generation systems applied to the domain of civil law, with a specific focus on three national contexts: Italy, Estonia, and Slovenia. The reference setting is a curated dataset that collects civil code texts and past legal cases for each of these three countries. Students will work on this dataset with a design-oriented perspective and are required to critically compare two main architectural strategies: a single-agent system with ReAct-style reasoning and a multi-agent system with a supervising agent.

Alongside the design of these two solutions, the project requires the realization of a chat interface that users can employ to query the dataset, and a conceptual or implemented evaluation dashboard that uses metrics inspired by the RAGAS framework to assess the chatbot's performance. The overarching goal is not only to obtain correct answers, but also to ensure transparency regarding how the system uses its sources, to foster critical thinking around architectural choices, and, above all, to reflect on which routing strategy is best suited to handle three different legal systems in a scalable way.

1.1 General objectives

From a learning perspective, the project is designed to help students understand the complete pipeline of a RAG system: from document ingestion and indexing, through the construction of vector databases, to the generation of the final answer.

A central objective is to make students reason explicitly about routing decisions: they must understand when and how to select the most appropriate sub-corpus among Italy, Estonia, and Slovenia, and how to handle questions that may involve more than one jurisdiction.

Another objective is to compare different architectural approaches, especially a single ReAct-style agent versus a multi-agent system with a supervisor, in a scenario where the dataset is explicitly multinational. Finally, students are expected to assess system performance using explicit metrics, connecting design decisions to observed outcomes and preparing their systems for an automatic evaluation based on a hidden set of queries.

2. Reference dataset

The dataset provided for the project consists of three national subsets: Italy, Estonia, and Slovenia. For each of these three countries, the corpus is organized into three main content types.

The first type is Inheritance civil codes, that is, civil code provisions relating to inheritance and succession. The second type is Divorce civil codes, which includes civil law rules and related provisions governing separation, divorce, and family law. The third type is Past legal cases, a mixed collection of past cases related to both inheritance and divorce, including court decisions, settlements, mediation outcomes, and similar case material.

All documents are provided in JSON format. Each file contains a textual field representing the content of the legal norm or the concrete case, and a set of metadata. These metadata typically include the country (Italy, Estonia, or Slovenia), the legal area involved, for example inheritance or divorce, any referenced civil code articles, the document type, which can be "code" or "case", the source, and additional attributes such as the duration of the procedure, costs, the nature of the dispute, or other contextual information.

This dataset must be treated as the system's ground truth. All retrieval mechanisms must operate on these documents. A key point of the project is that both architectures, the single-agent system and the multi-agent system, must be designed to use the entire multinational dataset: the combined data for Italy, Estonia, and Slovenia, across all three content types. The goal is to push students to think carefully about how to design routing strategies that can navigate multiple jurisdictions and sub-corpora in a principled way.

3. Task A – Single-agent system with ReAct-style logic

3.1 General description

The first task requires students to design a system based on a single RAG agent equipped with ReAct-style logic. This agent receives natural language questions from the user and uses a language model to explicitly decide whether it needs to consult documents from the dataset to answer the question, or whether the question can be handled from the model's general knowledge, for example for very simple or definitional queries.

Whenever the agent decides that retrieval is necessary, it must also determine which part of the multinational corpus is most relevant to the question. This means the agent should be able, for example, to recognize whether a query refers to a divorce case in the Italian context, an inheritance dispute in Estonia, or an ambiguous scenario that might benefit from comparing normative solutions across the three countries.

3.2 Logical flow of the single-agent system

The logic of the single-agent system can be described through four conceptual steps. In the Thought step, the agent uses the language model as a classifier to decide whether the user's question requires access to external documents. This decision is expressed in a simple form, for instance as a yes or no answer regarding the necessity of retrieval.

In the Action step, if the answer is no, the system proceeds by generating a response based solely on the model's internal knowledge. In an exam setting this should be used only for genuinely generic questions. If the answer is yes, the system must select which portions of the dataset to query. At this stage the options include the three country blocks, Italy, Estonia, and Slovenia, and the three content types: Inheritance civil codes, Divorce civil codes, and Past legal cases. The selection may target a single combination, such as inheritance law in Italy, or multiple combinations, such as past divorce cases across all three nations.

Once the relevant knowledge bases are selected, the system performs vector retrieval. This involves computing embeddings for the texts, using an index such as FAISS, and typically retrieving more documents than will ultimately be used as context. The retrieved documents are then filtered and re-ranked based on cosine similarity with the user's question, so that only the most semantically pertinent passages are kept as context.

In the Observation step, the system internally summarizes what it did. This summary might include which countries and document types were queried, which civil code articles are prominent in the selected documents, and which past cases appear most similar to the situation described by the user.

Finally, in the Answer step, the agent builds a prompt that combines the user's question with the selected documents as context and submits it to the language model. The generated answer should be clear, consistent with the legal framework of the relevant countries, faithful to the sources, and, whenever possible, explicitly refer to relevant articles and cases.

4. Task B – Multi-agent system with a supervising agent

4.1 General description

The second task introduces a multi-agent architecture. In this scenario the system is no longer represented by a single agent performing all functions, but by a supervising agent and several specialized agents. Each specialized agent is responsible for a particular subset of the dataset. In this setting, specialization should not be limited to a single country only; it can also be organized more finely, for example by legal area or by nation–content-type combinations, as long as the union of all specialized agents covers the full dataset for Italy, Estonia, and Slovenia across all three content types.

Each specialized agent is associated with one or more vector databases that are coherent with its role. For instance, one agent might focus on past divorce cases, another on inheritance civil codes, another on inheritance cases in a specific country, and so on. In each case, the internal retrieval and answering logic of the specialized agent may reuse the same ideas developed for the single-agent system, but restricted to its own slice of the knowledge base.

4.2 Logical flow with the supervisor

The supervising agent is the entry point for user questions. When it receives a question, it has access to a short description of the content and domain expertise of each specialized agent, for example which combination of country, legal area, and document type each agent handles best. Based on these descriptions and the question content, the supervisor uses a language model to decide which agents to activate for that specific request.

The decision may involve a single agent, for example if the question clearly concerns a divorce case under Italian law, or multiple agents, for instance if the question asks for a comparison between several jurisdictions or requests similar cases across different countries. The supervisor forwards the question to the selected agents, each of which runs its own retrieval and answer-generation process against its vector database.

After the specialized agents have returned their partial answers, the supervisor is responsible for aggregating them. The language model can synthesize the different perspectives, compare normative solutions, and highlight possible divergences between Italy, Estonia, and Slovenia. The final output presented to the user is a single coherent answer that hides the internal complexity of the multi-agent architecture.

From a design perspective, students must clearly describe this flow, explaining how agent descriptions are constructed, how the supervisor makes routing decisions, how partial results are merged, and how the overall system leverages the multinational dataset to produce richer answers.

5. Chat interface and conversation logging

The chat interface is the user's main point of contact with the system. It must allow users to submit natural language questions, view the model's answers, and, in the interest of transparency, see the main documentary sources used to support each answer. Ideally, the chat should indicate which documents were retrieved, from which country they come, whether they are civil code provisions or past cases, which articles are mentioned, and a short excerpt of the most relevant content.

At the same time, the chat must log interactions in a structured format, such as JSON. For each turn in the conversation, it should store at least the user's question, the system's answer, and the set of retrieved documents, including key metadata like country, type, and source name. These logs will be used both for internal evaluation with RAGAS and for understanding how the system actually exploits the multinational dataset in practice.

6. Evaluation dashboard and exam assessment with RAGAS

The evaluation dashboard is intended to use conversation logs to compute, conceptually or in practice, the system's performance using metrics inspired by the RAGAS framework. The focus is on the relationship between the retrieved context and the quality of the generated answer.

The metrics of interest include context precision, which measures how many of the retrieved documents are truly useful and relevant to the question, and context recall, which measures how much of the necessary information is actually present in the context provided to the model. Faithfulness evaluates how well the answer is grounded in the retrieved documents. Answer relevancy quantifies how well the answer addresses the user's query, while answer correctness compares the model's answer to a reference answer considered as ground truth. For internal experimentation, students may define a small set of test questions with ideal reference answers and compute these metrics on them.

6.2 Official exam evaluation with hidden queries

The official exam evaluation will use RAGAS on a set of ten hidden queries, each with its own reference answer. This evaluation set will not be known to students in advance. The evaluation modality will be communicated a few days before the exam date and will POSSIBLY consist of a link to a dedicated web application.

In this web application, each group will be asked to enter the names of all group members, a short description of their system, both for the single-agent and for the multi-agent architecture, and to provide access to their application endpoint according to the technical instructions that will be specified. The web app will then use the ten hidden ground truth and automatically compute the RAGAS metrics. The results will be used as part of the final evaluation and may be returned as a report, which students can consult to understand the actual performance of their systems and their limitations.

The presence of this hidden evaluation is meant to encourage students to design robust architectures that do not overfit a small set of known examples, but instead generalize to realistic new questions concerning the legal systems of Italy, Estonia, and Slovenia.

The interpretation of the results for both the agentic and multi-agent system is strongly adviced and will be crucial part of the evaluation.

THE EXAM EVALUATION IS ONE SHOT, ONE SUBMISSION.

7. Project documentation and deliverables

7.1 Flowcharts for the tasks

Students must produce a set of flowcharts describing the main processes of the two tasks. The flowchart for the single-agent architecture should clearly illustrate how the question enters the system, how the decision to use retrieval is made, how the system selects which countries and document types to query, how documents are retrieved and filtered, and how the final answer is generated.

The flowchart for the multi-agent system should highlight the role of the supervisor, how the specialized agents are described and selected, how the supervisor forwards queries to them, how their partial answers are returned, and how they are merged into a single answer. In both diagrams, it should be clear where routing towards different knowledge bases occurs and how the final context is constructed for the generative model.

7.2 Architectural description, models used, and performance table

Beyond the flowcharts, students are required to write a textual description of the overall architecture. This description should present the main conceptual modules: configuration management, loading of documents for Italy, Estonia, and Slovenia, the components that compute embeddings and build vector databases, the functions implementing the different answer-generation pipelines for the single-agent and multi-agent systems, and the mechanisms for logging and evaluation.

Within the report, there must be a dedicated section for the models used. In this section, students must briefly indicate the type of embedding model adopted, specifying at least the model name, whether it is an open-source model or an external service, and the general motivations for this choice, such as language support, model size, semantic capability, or cost. Similarly, they must describe the generative model selected, indicating the provider, the model name, any relevant API parameters, and the main reasons behind this choice.

It’s suggested to also prepare a comparative performance table with students trials . In this table, they will report the RAGAS metrics for the two main architectures focusing on context precision, context recall, faithfulness, answer relevancy, and answer correctness.
The numerical merged with the official evaluation should be accompanied by short comments that highlight the strengths and weaknesses of each approach, helping to understand which routing strategy appears most suitable for handling the multinational dataset and for being scaled to broader contexts.

7.3 Presentation slides and choice of the best strategy

Students are required to prepare a short slide deck (8-12). These slides should describe the dataset structure and eventual data manipulation, the two main architectural strategies, the key design choices for each approach, the main results from the internal evaluation dashboard, and, when available, the outcome of the official evaluation based on hidden queries.

In the final part of the presentation, students must provide a critical assessment that takes into account both quantitative results and qualitative observations from system development and usage. They should justify which routing strategy, whether single-agent or multi-agent with a supervisor, appears more promising for future scaling to larger legal corpora, while maintaining transparency, robustness, and coherent behavior across Italy, Estonia, and Slovenia.

8. Possible extensions and further experimentation

In addition to the mandatory requirements, students are encouraged to explore extensions that deepen the analysis of system behavior and routing strategies.

- One possible extension is to further refine the segmentation of past legal cases according to the type of legislation or specific legal subdomains, and to assess whether this finer-grained partition improves the retrieval of analogous and relevant cases.
- Another extension concerns experimenting with different API settings for language and embedding models, testing alternative combinations of generative and embedding models, and varying parameters such as temperature, maximum answer length, repetition penalties, and other relevant settings. This allows students to observe how these configuration changes affect global performance and RAGAS metrics.
- A further line of experimentation is related to similarity functions. Students may compare cosine similarity with other approaches, such as dot product or different distance measures, to understand whether the choice of similarity metric has a significant impact on retrieval quality.
- Students may also introduce advanced post-retrieval optimization and query-handling strategies, such as automatic query reformulation, query expansion, secondary re-ranking using a more powerful model, or simple consistency checks on the final answer, for example verifying that it cites at least one relevant norm or that the country mentioned in the answer matches the source documents.
- Finally, even though it is not part of the core tasks, students can propose a hybrid system that more tightly integrates metadata and vector search. For example, they might use metadata to perform a first logical filtering of the multinational corpus, followed by vector similarity to

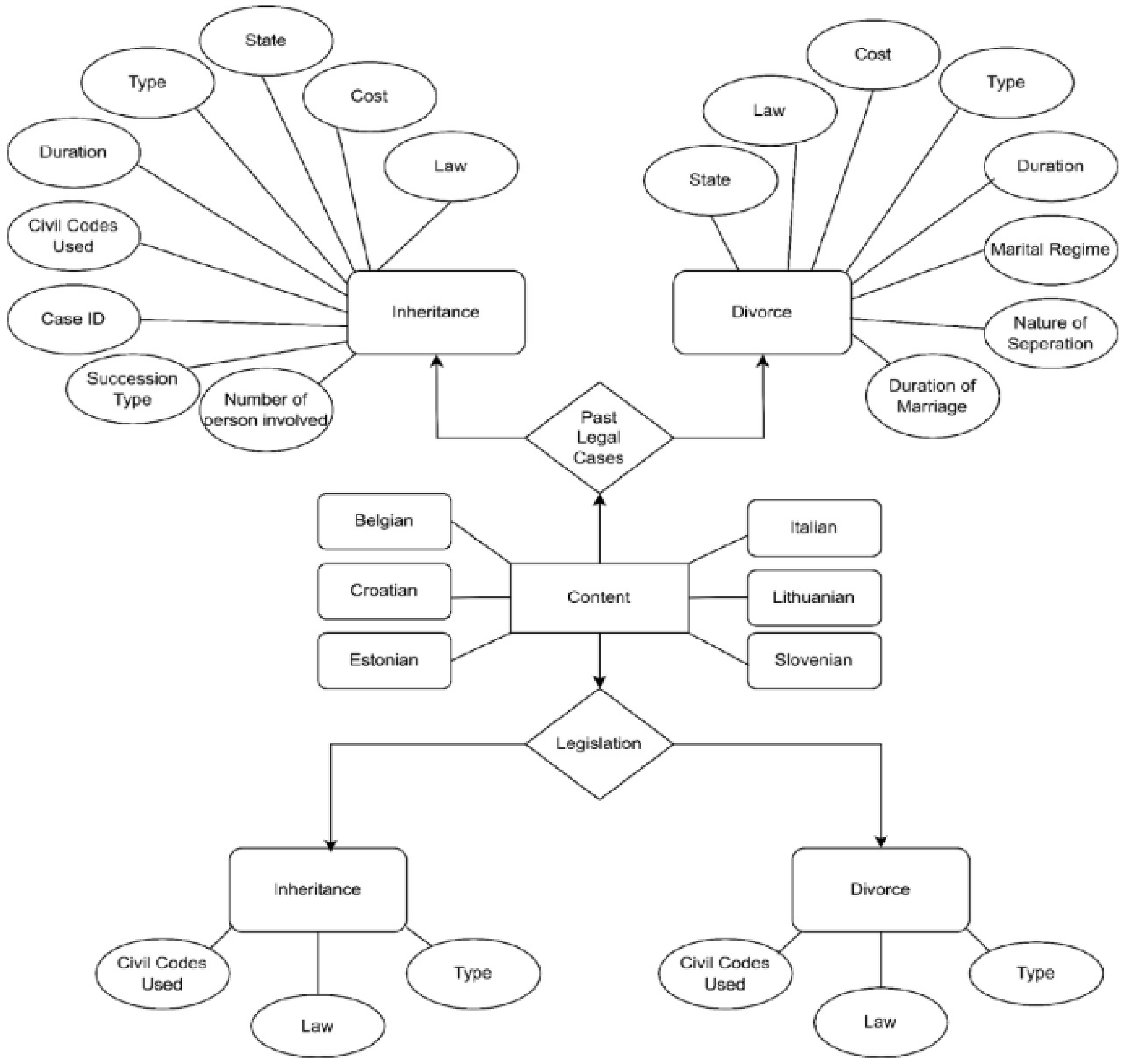
refine the selection. These extensions are not compulsory but can provide valuable insight into the trade-offs among routing strategies and contribute to identifying which approach is most suitable to be scaled to a larger legal setting.

9. Conclusion

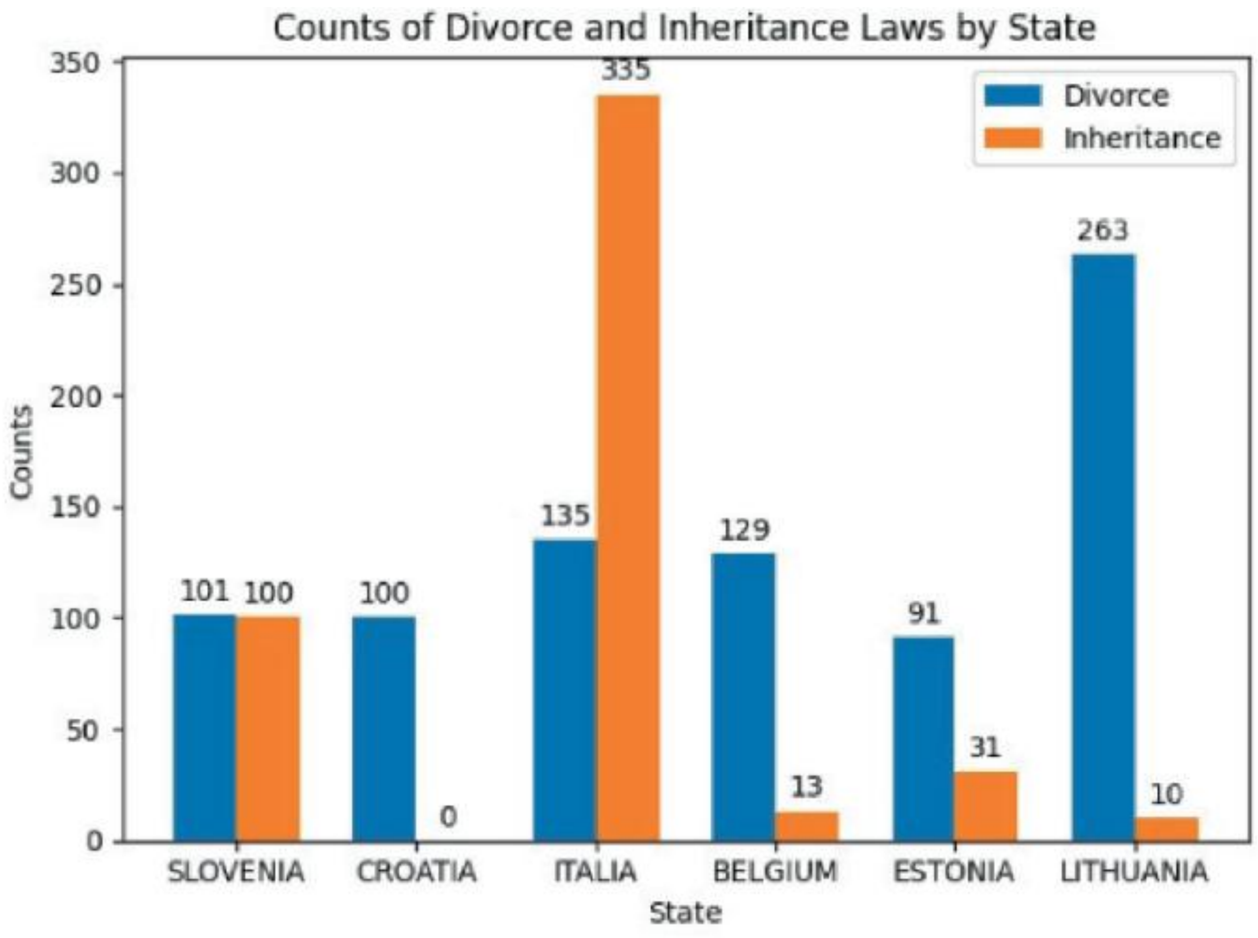
This document is intended as a complete guide for the exam project. While it leaves room for flexibility in the choice of models, parameters, and implementation details, it requires students to remain consistent with two distinct approaches to retrieval in the legal domain, to build a transparent chat interface, to use the full dataset covering Italy, Estonia, and Slovenia as the only knowledge source, and to adopt a rigorous perspective on performance evaluation.

Through the design of the single-agent system, the multi-agent supervisor architecture, and any additional experiments, students are invited to reflect not only on immediate answer quality, but also on routing structure, the choice of embedding and generative models, and the feasibility of scaling the application to broader contexts while maintaining controllability, transparency, and reliability.

Appendix



CREA2 Knowledge Base Schema: Inheritance and Divorce for six different Countries. This schema reflects the metadata in the raw dataset (sometimes they are not present or Null).



The bar chart presents a comparative analysis of the number of divorce and inheritance cases across six European states: Slovenia, Croatia, Italy, Belgium, Estonia, and Lithuania. The data indicate that Italy possesses the greatest number of divorce cases, totaling 335, whereas Lithuania leads in the category of inheritance laws with 263 instances.

CREA 2 References

https://link.springer.com/chapter/10.1007/978-3-031-86149-9_40

https://link.springer.com/chapter/10.1007/978-3-031-76462-2_1

https://link.springer.com/chapter/10.1007/978-3-031-96099-4_9

https://link.springer.com/chapter/10.1007/978-3-032-05772-3_26

CODE

https://github.com/Al-Moccardi/RAG_4_Scratch **(Handcrafted RAG system)**
https://rag_demo.idealunina.work/ **RAG Demo: use (.\data\ for folder ingestion)**