Graph-Based Modelling of the Italian Criminal Code as a Knowledge Base for AI-Assisted Inference

Samuele Yves Cerini[1,2,*,†], Nicola Deidda[1,3,*,†], Yilian Huang[1,*,†], Sara Lilli[1,4,*,†], Luca Lobascio[1,5,*,†], Lavinia Russo[1,6,*,†], Maria Vittoria Zucca[1,4,*,†], Giorgio Fumera[3], Giorgio Giacinto[2,3], Paolo Prinetto[2,*] and Salvatore Marsocci[7]

[1]Scuola IMT Alti Studi Lucca, Piazza S. Ponziano 6, 55100 Lucca, LU, Italy
[2]CINI Cybersecurity National Lab, Via Ariosto 25, 00185 Roma, RM, Italy
[3]Università di Cagliari, Via Università 40, 09124 Cagliari, CA, Italy
[4]Scuola Superiore Sant'Anna, Piazza Martiri della Libertà 33, 56127 Pisa, PI, Italy
[5]Università degli Studi di Bari "Aldo Moro", Piazza Umberto I, 70121 Bari, BA, Italy
[6]Università degli Studi di Napoli Federico II, Corso Umberto I 40, 80138 Napoli, NA, Italy
[7]Procura della Repubblica presso il Tribunale di Torino, Corso Vittorio Emanuele II 130, 10138 Torino, TO, Italy

Abstract

This paper investigates the feasibility of representing the Italian Criminal Code as a graph knowledge base and of using Artificial Intelligence (AI), particularly Large Language Models (LLMs), to query this structure. We propose a Directed Acyclic Graph (DAG) where the nodes correspond to the structural elements of the code (Book, Title, Article, Comma), and the edges encode both hierarchical and selected semantic relationships, such as textual references and dependencies. Nodes and edges are enriched with properties that link them back to the text and record identifiers, as well as temporal information and provenance metadata. Treating this graph as a knowledge base enables a range of graph-theoretic operations to be interpreted as legal tasks such as structural navigation, dependency analysis, and localised impact assessment of legislative changes. We outline how LLMs can be used on top of this representation to support natural-language querying, semantically enriched search, and comparative analysis between provisions. In this design, LLMs act as an interface layer that translates natural-language queries into formal graph queries and translates graph-structured results back into explanations. All retrieval and dependency computation are performed by exact algorithms on the DAG. This proposal does not focus on devising a methodology to substitute the human expert. Indeed, a Human-in-the-Loop approach is crucial: the human legal expert can either accept, modify, or reject the response proposed by the AI system.

Keywords

Criminal Code, Artificial Intelligence, Large Language Models, Graph Representation, Human-in-the-Loop,

1. Introduction

The increasing availability of advanced automated reasoning techniques has renewed interest in computational approaches in the legal field. Knowledge representation and Artificial Intelligence (AI), particularly Large Language Models (LLMs), promise to support legal professionals in tasks such as norm navigation, consistency checking, and preliminary fact qualification [1], supporting, for instance, the semi-automatic extraction of information from statutory text, provide natural-language explanations of generated responses, and facilitate interactive querying in legal terminology. However, the efficacy of these technologies in criminal law is subject to the availability of explicit, structured representations of the underlying legal *corpus*. Moreover, it is crucial to recognize the vital role of human legal experts in

Joint National Conference on Cybersecurity (ITASEC & SERICS 2026), February 09-13, 2026, Cagliari, IT
samuele.cerini@imtlucca.it (S. Y. Cerini); nicola.deidda@imtlucca.it (N. Deidda); yilian.huang@imtlucca.it (Y. Huang); sara.lilli@santannapisa.it (S. Lilli); luca.lobascio@imtlucca.it (L. Lobascio); lavinia.russo@imtlucca.it (L. Russo); maria.zucca@santannapisa.it (M. V. Zucca); giorgio.fumera@unica.it (G. Fumera); giorgio.giacinto@unica.it (G. Giacinto); paolo.prinetto@cybersecnatlab.it (P. Prinetto); salvatore.marsocci@giustizia.it (S. Marsocci)
0009-0001-4961-8915 (S. Y. Cerini); 0009-0005-7309-5703 (N. Deidda); 0009-0000-2796-3067 (S. Lilli); 0009-0000-3011-906X (L. Lobascio); 0009-0002-3725-1092 (L. Russo); 0009-0004-0049-9611 (M. V. Zucca); 0000-0001-5300-226X (G. Fumera); 0000-0002-5759-3017 (G. Giacinto); 0000-0003-2400-8245 (P. Prinetto)

any interpretation, validation, and decision-making process, in accordance with the *human-in-the-loop* paradigm [2].

In this work, we examine the Italian Criminal Code (*Codice Penale*) [3] and propose its modelling as a graph-based structure that can serve as the foundation for various forms of automated inference. Existing digital tools for the Criminal Code are text-centric, offering support for keyword searches, citation-based navigation, and simple taxonomies. However, these tools do not expose the deeper relational structure of the code. Consequently, while their implementation considerably streamlines human processes, their capacity to support higher-level automated inferences appears to be constrained. Examples of such higher-level inferences include structural representations of the interaction between general and special norms, the cumulative or alternative application of offences, and the propagation of legislative amendments across related provisions.

In the proposed approach, the Criminal Code is represented as a knowledge graph [4], in which nodes represent legal entities of the Criminal Code at various granularity (e.g., books, titles, articles), while edges capture legally significant relationships among them. Additional attributes may encode metadata (e.g., source of the norm, interpretative notes, category). This graph is not merely a navigation aid; in fact, it is intended to serve as a knowledge representation layer over which more expressive rule-based or logical formalisms can be defined, enabling the explicit modelling of offence structures, applicability conditions, and normative constraints.

In this paper, we do not present a full-scale implementation nor an empirical evaluation. Instead, we focus on the conceptual and technical design choices required to make such an approach viable and explainable in practice. In particular, we address: (i) how to define an adequate graph schema for the Italian Criminal Code that balances expressiveness with maintainability; (ii) how to leverage graph properties and reasoning formalisms over this graph; and (iii) how LLMs can assist in the interaction with the model while ensuring that human jurists remain central in interpreting both the model and its outputs. Throughout the discussion, we emphasize the importante of leveraging a *human-in-the-loop* paradigm. Legal experts should be involved in designing the graph ontology, curating the extracted entities and relationships, specifying and reviewing formal rules, finally assessing the outputs of automated reasoning tools. The objective is not to mechanize legal judgment, but to provide structured, machine-processable artifacts and inference services that augment human analysis, increase transparency of assumptions, and facilitate rigorous inspection of normative interactions within the Italian Criminal Code.

The remainder of the paper is organized as follows. Section 2 provides background notions on the Criminal Code structure, on linguistics and modern language models; Section 3 outlines the analysis of the Criminal Code structure and the methodology used in developing the paper's proposal; Section 4 further explores the possibilities allowed by the graph representation of the Criminal Code; Section 5 concludes the paper summarizing its main motivations and the methodology applied, finally proposing future expansions.

2. Background and State-of-the-Art

Before presenting our methodology, we recall the legal, linguistic, and technical premises that provide the foundations for this proposal.

2.1. Legal Premises

This research is situated within the framework of Civil Law systems, which are characterized by the predominance of codified statutes as primary sources of law. In contrast to Common Law systems, where judicial decisions and case law form the bedrock of legal development, Civil Law systems emphasize the systematic codification of legal norms. Codified legal systems, prevalent in many European jurisdictions, such as Italy, France and Germany, aim to encompass all areas of legal regulation within a coherent structure of written codes. In particular, this study focuses on the Italian Criminal Code, a highly structured and articulated normative system that regulates criminal conduct and defines the conditions

of criminal liability. Its organization is based on a codified body of law, arranged according to a hierarchy of books, titles, chapters, and articles, which enables not only a formal but also a conceptual systematization of criminal norms. Within this framework, the Code is divided into three books, each of which fulfills a distinct yet complementary function, as follows:

- The General Part (Book I): outlines the foundational principles of criminal law, defining criminal liability, the structure of criminal conduct (i.e., *actus reus* and *mens rea*), and the application of sanctions. It also establishes defenses and identifies circumstances that may mitigate or aggravate penalties;
- Serious Crimes (Book II): regulates serious offenses (*delitti*), prescribing penalties and categorizing crimes according to the legal interests they safeguard (e.g., life, property, public order);
- Minor Offences (Book III): governs minor infractions (*contravvenzioni*), prescribing less severe sanctions, commensurate with the reduced social harm posed by such conduct.

A central element of this system is the dynamic relationship between general (Book I) and special provisions (Book II-III). The general provisions establish the foundational legal framework through which the specific provisions must be interpreted and applied, while the special provisions may introduce rules that either complement or restrict the scope of the general provisions (i.e., principle of specialty). The result is a stratified normative structure in which the interpretation of individual rules frequently requires coordination among multiple provisions located in different parts of the Code. The systemic nature of the Italian Criminal Code further emerges through the widespread use of cross-references and textual links, which interweave the various provisions in a non-linear manner. Understanding the prescriptive content of a rule is therefore often contingent upon analyzing the relationships it maintains with other provisions, both structurally and conceptually. In this sense, the Criminal Code may be described as a semantic network of interdependent provisions rather than as a mere collection of autonomous articles. It is precisely this relational nature that makes it a particularly suitable domain for structured modeling and representation as a system of formal relationships.

2.2. Linguistic Premise

Legal languages are generally considered specialized languages, employed in highly professional domains for specific purposes. The Italian Criminal Code is also included in this subset, and it is clearly distinguishable from standard Italian.

At the level of the written text, provisions are characterized by the presence of long and syntactically complex sentences in formal expressions, with extensive use of nested subordinate clauses, nominalization, and passive voice constructions. At the discourse level, legal documents are often densely interconnected. For instance, provisions are often referenced or associated with other articles and books in the general section. Cross-references of this nature are frequently important in determining the scope and conditions of an offense. Finally, it is evident that these documents are characterized by "legalese", incorporating multi-word expressions and formulaic phrases that are more complex and professional than those employed in everyday discourse.

Moreover, this domain is characterized by high stakes and low tolerance for errors. Minor variations in terminology or expression can result in significantly divergent legal outcomes. From a Natural Language Processing (NLP) perspective [5], the complexity and legal structures of the Italian Criminal Code imply that these legal documents cannot be treated as a collection of independent sentences. The necessity for a methodology to represent the connections within the codes is thus apparent, and this is precisely what a graph model fulfils. Such a model must be capable of supporting automated reasoning, providing adequate boundaries, thereby helping to avoid reasoning drifts and inaccuracies to the greatest extent possible. Furthermore, we consider it crucial to implement an auditing process performed by human legal experts: the purpose of this process would be to address errors caused by an automatic and unsupervised construction of the graph model.

2.3. Technical Premises

From a technical perspective, this research lies at the intersection of legal NLP [6] and knowledge representation. The following sections will highlight recent advances in domain-specific language models, as well as explore how the linguistic analysis of legal Italian can contribute to the design of a graph of the Italian Criminal Code that is both structurally correct and semantically rich.

2.3.1. Graphs

Graphs [7] provide a simple yet powerful mathematical framework for representing relational structures. Formally, a graph consists of two basic components: nodes (also called vertices), which represent individual elements, and edges, which are connections linking pairs of nodes together.

By assigning types or labels to nodes and edges, and by attaching attributes to them, it is possible to distinguish different kinds of entities and relationships within the same structure. These features make graphs a natural choice for modeling complex, interconnected systems, such as legal corpora, where one needs to represent both hierarchical organization and cross-cutting semantic links in a way that remains usable for algorithmic manipulation and inspection.

2.3.2. Natural Language Processing and Large Language Models

Natural Language Processing (NLP) studies computational methods for analysing, interpreting, and generating human language. Classical NLP techniques include syntactic parsing and information extraction, which break texts into structured units and identify entities and relations among them. Recent advances in neural models, particularly LLMs, have significantly expanded the range of tasks that can be tackled, from semantic similarity and question answering to summarization and dialogue.

Large Language Models (LLMs) are Artificial Intelligence (AI) models trained on massive text datasets to predict and generate sequences of words. Based on the Transformer [8] architecture, LLMs have demonstrated the ability to capture long-range dependencies, handle diverse linguistic patterns, and perform tasks based on carefully designed prompts. In practice, LLMs can answer questions and summarize documents [9], perform correlation of different inputs and generate responses, albeit with no formal guarantees of correctness [10]. Within a graph-based legal knowledge base, LLMs are particularly useful as an interface layer: they map natural-language queries to formal graph operations and turn graph-structured results into coherent explanations, allowing human users to benefit from the structure and precision of the graph without being exposed to its technical details.

Concurrent with these advancements in LLMs for general use, significant developments have also been made in LLMs for legal documents. Chalkidis *et al.* introduced LEGAL-BERT [11], a model based on BERT [12] that has been pre-trained on approximately 12 GB of English legal texts, including legislation and contracts. Specialized models, such as LEGAL-BERT, outperform general-purpose models on legal tasks. To properly evaluate them, a benchmark dataset called LexGLUE [13] was designed, combining different legal datasets to test and evaluate the models' capabilities on case prediction and classification tasks.

Similar developments can be observed in civil law languages. For Italian, Licari *et al.* introduced "Italian legal BERT", which outperforms general models by pre-training on civil law documents [14]. Other previous works have also successfully adapted transformer architectures for Italian legal tasks, such as retrieving civil code articles and summarizing judicial decisions [15]. More recently, the focus has shifted to generative models and Retrieval-Augmented Generation (RAG), which can combine LLMs with an external legal knowledge base to assist legal question answering and drafting [16]. These studies demonstrate that training models on a specific domain, such as legal texts, enhances models' performance, and benchmarks like LexGLUE can help identify which types of legal reasoning tasks existing models can or cannot handle.

Nevertheless, a fundamental issue persists: the majority of methodologies approach legal texts as unstructured, linear sequences of words, disregarding the underlying logical structure of the law itself. The graph-based representation method proposed in this work complements the current state of the art.

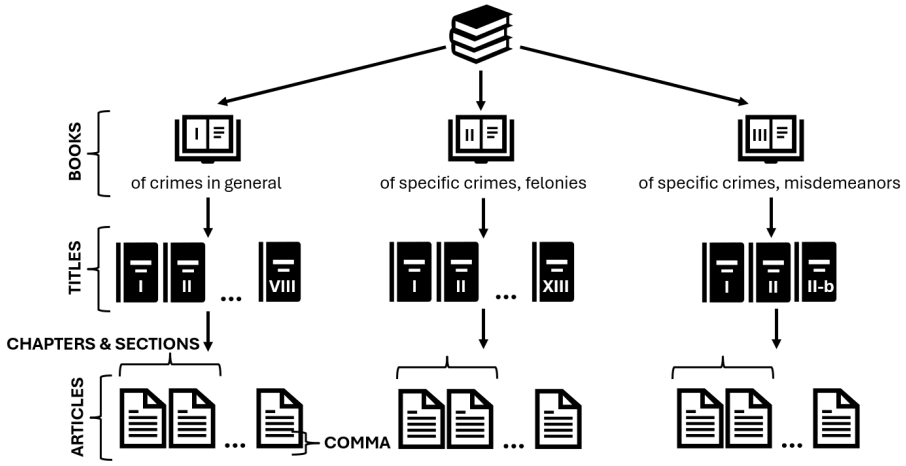

Figure 1: Criminal Code Structure

First, such a proposal can provide a structured, machine-readable version of the Italian Criminal Code, which can serve as an external knowledge base for specialized LLMs. Second, this method improves explainability by explicitly modelling the relationships between laws, ensuring that the outputs of legal LLMs are grounded in an interpretable network of legal dependencies.

3. Methodology

This methodological discussion defines the need for a structured, machine-readable representation of the Italian Criminal Code. As long as the code is available only as unstructured text, the systematic application of automatic reasoning techniques remains limited to ad-hoc heuristics or manual expert interpretation. To support more rigorous and transparent forms of automation, we need a representation that makes the internal organization of the code explicit and that can serve as a stable substrate for further formalization.

Given the inherently hierarchical nature of the Criminal Code, organised into books, titles, chapters, articles, and *commi*, as illustrated in Figure 1, we evaluate the use of a graph-like representation. In such a representation, nodes correspond to structural elements of the code, and edges capture the relationships between them, including both the hierarchical links and the selected semantic relations (e.g., textual references or applicability constraints). This choice is motivated by the flexibility of graph models in expressing heterogeneous relations while preserving a clear and navigable structure. Before detailing how the graph is constructed, we make explicit the high-level objectives that guide the modelization process. The representation should:

- Preserve the hierarchical structure of the Criminal Code in a form that can be traversed and queried algorithmically;
- Expose, through suitably labelled edges, a first layer of semantic relations that are relevant for reasoning tasks (e.g., cross-references, preconditions for applicability);
- Remain sufficiently simple and transparent to be inspected, validated, and maintained by legal experts, who retain control over the interpretation of the model;
- Be suitable as a basis for the later integration of formal methods and LLM-based tools, without requiring any specific reasoning technology;
- Support further integrations and modifications to mimic the future evolution of the Italian Criminal Code.

In the remainder of this section, we will progressively refine this methodological proposal, describing how the graph is defined, which kinds of nodes and edges it contains, and how properties and metadata are attached to its elements to support both human inspection and automated processing.

3.1. Evaluating Different Hierarchical Models for the Italian Criminal Code

In this section, we discuss which kind of model is most appropriate for the Italian Criminal Code. The choice determines how the hierarchical structure of the code and its internal links can be expressed, and how easily the resulting representation can be inspected and maintained by legal experts.

3.1.1. Tree-like Structures

The starting point is the idea of a tree-like structure. In this view, the code is represented as a rooted tree that mirrors its editorial hierarchy: books contain titles, titles contain articles, and articles contain *commi*. Each node has a *unique parent*, and the resulting structure coincides with the table of contents of the code. This representation has the advantage of being extremely intuitive: it is easy to visualise, traverse, and is closely aligned with how legal practitioners already think about the structure of the Code. However, a pure tree is too restrictive for our purposes. The Criminal Code is rich in cross-references and substantive links that cut across the strict hierarchy, such as provisions in the general part that condition the applicability of offences in the special part, or articles that refer explicitly to other articles. A tree, by construction, cannot directly express these lateral connections without resorting to ad-hoc annotations that fall outside the tree structure.

3.1.2. Graphs

Given the aforementioned limitations, the consequential choice is to move from trees to graphs. We retain the idea that nodes correspond to structural elements of the code, but we no longer restrict ourselves to a single *contains* relation (i.e., a single class of edges). Instead, we consider a graph in which the hierarchical links form a backbone, and additional edges connect nodes to represent semantic relations of interest, such as textual references, preconditions of applicability, or exclusions. In such a graph, the hierarchy that was previously captured by parent–child relations in a tree becomes one distinguished class of edges, and semantic links form other classes, all coexisting within the same formalism. This approach preserves the intuitive structural view provided by the tree, while adding the flexibility needed to encode the internal cross-references of the code. Within this graph-based view, two further design questions arise: the *direction of edges* and the *presence of cycles*. Directionality is natural in our setting. Hierarchical links have an inherent direction, from container to contained element (e.g., from a book to a title or from an article to a *comma*). Semantic relations also admit a meaningful orientation: a reference typically goes from the provision that cites to the provision that is cited; a precondition can be oriented from the structural element that expresses the condition to the structural element whose applicability depends on it. Adopting directed edges, therefore, allows us to reflect the asymmetry that is already present in the legal text.

Allowing arbitrary directed edges, however, raises the issue of cyclicity. In principle, nothing prevents two provisions from referring to each other, or a chain of preconditions from forming a cycle, and a general directed graph may contain several cycles. While this is not problematic for representing the raw structure of the code, it complicates several reasoning tasks that we envisage, such as propagating conditions along the graph or performing topological analyses to understand dependencies. Cycles can obscure the applicability of a given article and make it harder to explain why it is considered active in a particular scenario.

In this feasibility study, we therefore constrain the model to a *Directed Acyclic Graph (DAG)* [17]. The hierarchical edges already form a tree-like acyclic backbone. Semantic edges are oriented in a way that respects this acyclicity: references and preconditions are directed downstream along a chosen notion of dependency (e.g., from more general provisions to more specific ones, or from earlier interpretative clauses to later offences), and potential symmetric or circular relations are resolved by selecting a legally meaningful direction. When genuinely mutual relationships are encountered, they can be represented using pairs of edges at a meta-level or handled through additional annotations, but the core structural–semantic graph that we use as a basis for reasoning remains *acyclic*.

The resulting model is a directed acyclic graph whose nodes are the structural elements of the *Codice Penale*, and whose edges capture both hierarchical containment and a controlled set of semantic relations. Metadata is attached as *properties* to nodes and edges, rather than being encoded in the graph structure itself. This enforces a balance between faithfulness to the code's structure, the expressiveness of internal links, and the simplicity of the dependency structure required for subsequent automated reasoning.

More expressive models, such as hypergraphs or higher-order graphs, could, in principle, capture complex relationships involving more than two structural elements at once. However, their added complexity is not strictly necessary for the level of structure we aim to capture in this feasibility study, and would make the representation harder to communicate to non-technical stakeholders.

3.2. Graph Formalization

We introduce a formal description of the graph model. The objective at this stage is to specify its abstract structure, independently of any implementation choice. We denote by $G = \langle V, E \rangle$ the graph representing the Italian *Codice Penale*. The set V contains the nodes of the graph and corresponds to the structural elements of the code (e.g., books, titles, articles, and *commi*). The set E contains the edges of the graph and represents the relationships between such structural elements. Each edge $e \in E$ is associated with a source node $s(e) \in V$ and a target node $t(e) \in V$, so that $E \subseteq V \times V$ can be seen as a set of ordered pairs with explicit source and target functions $s, t : E \to V$.

Nodes and edges are typed. We write T_V for the finite set of node types and T_E for the finite set of edge types (or labels). The typing functions $\tau_V : V \to T_V$ and $\tau_E : E \to T_E$ assign to each node a structural type and to each edge a relation type, respectively. In our setting, T_V will later be instantiated with the structural categories of the *Codice Penale*, while T_E will distinguish between hierarchical and semantic relations.

Both nodes and edges may carry metadata. We denote by $\mathcal{M}_V$ the space of node properties and by $\mathcal{M}_E$ the space of edge properties. The functions $\alpha_V : V \to \mathcal{M}_V$ and $\alpha_E : E \to \mathcal{M}_E$ associate to each node and edge a finite map of attributes, such as textual content, identifiers, or temporal information. The specific choice of attributes is not fixed at the level of the formal definition: it can evolve as long as the typing and structural constraints of the graph are preserved. The acyclic requirement can now be expressed by imposing that the underlying directed graph is a DAG. This property will be exploited in subsequent reasoning tasks, for instance when propagating information along the graph or analysing the activation of provisions based on their preconditions.

This abstract formalization isolates the core components of the model: a finite set of nodes representing the structural elements of the Criminal Code, a finite set of directed, typed edges representing the relations between these elements, and two families of metadata functions.

3.2.1. Node Types

We now discuss the abstract node space V and the typing function τ_V. Considering the aforementioned objectives, we restrict node types to those elements that are explicitly present in the code's structure. No additional conceptual or logical entities are introduced at the node level: all higher-level semantics will be expressed via edge labels and metadata. This choice keeps the model close to the legal source and facilitates inspection by human experts.
Concretely, the set of node types T_V is defined as:

$$T_V = \{\mathsf{Book}, \mathsf{Title}, \mathsf{Article}, \mathsf{Comma}\}.$$

Each node $v \in V$ is assigned exactly one of these types by τ_V, so that $\tau_V(v) \in T_V$. A node of type Book represents one of the books into which the code is divided; a node of type Title represents a title within a given book. The level of granularity that is most directly connected to the operative content of the criminal provisions is captured by nodes of type Article and Comma. An Article node stands for a

single numbered article of the *Codice Penale*, while a Comma node represents a specific *comma* within that article.

The explicit introduction of Comma nodes is motivated by the internal structure of many criminal provisions, where different *commi* may contain distinct hypotheses, conditions, or consequences. By modelling *commi* as separate nodes, we are able to attach edges and metadata at this finer level of granularity, e.g., when expressing that a particular *comma* is a precondition for the applicability of another provision. When an article has no explicit subdivision, we use single Article node whose textual content coincides with the full text of the article.

This instantiation of T_V yields a node space that coincides with the visible structure of the *Codice Penale*. Every node can be mapped back to a well-identified fragment of the code, and conversely every such fragment has a corresponding node in V.

3.2.2. Edge Types and Semantics

While nodes are deliberately limited to the structural elements of the Italian *Codice Penale*, edges express how these elements are related to one another. In particular, edges play a dual role: they reconstruct the hierarchy of the code and encode selected semantic relations, such as textual references or preconditions for applicability. Formally, this is captured by the edge typing function

$$\tau_E : E \to T_E,$$

where T_E is a finite set of edge types. Intuitively, each type in T_E corresponds to a distinct legal relation that can hold between two structural elements.

It is convenient to conceptually distinguish between hierarchical edges and semantic edges. Hierarchical edges represent the relations between structural units. For instance, an edge from a node of type Book to a node of type Title indicates that the title is contained in that book; similarly, a node of type Article is connected to its *commi*. At the level of edge types, this can be expressed by including in T_E a type, denoted by contains, which is used whenever a higher-level structural element directly contains a lower-level one. The resulting set of contains-edges forms a tree-like backbone of the graph, reflecting exactly the editorial structure of the Criminal Code.

Semantic edges, by contrast, connect structural elements in ways that are not captured by simple containment. Their types express legally meaningful relations extracted from the text. A fundamental example is that of textual references: when a comma or article explicitly refers to another provision, this is represented by a directed edge from the node corresponding to the referring fragment to the node corresponding to the cited provision. The edge is typed by a label such as refersTo, so that the semantic content "this provision cites that provision" is recorded in the graph without introducing any additional non-structural nodes.

Another central use of semantic edges is the representation of preconditions of applicability. Many provisions in the general part of the Criminal Code define conditions or general rules that affect when a specific offence can be applied. In the graph, such relations are expressed by edges whose type indicates that one structural element must be taken into account for the activation of another. The label preconditionOf captures, at the level of the graph, the idea that the content of one structural fragment operates as a precondition for the operative effect of another. The direction of the edge reflects the dependency: it goes from the condition-bearing fragment to the fragment whose applicability depends on it.

The requirement that the overall graph be a DAG constrains how these edges are oriented. Hierarchical edges are naturally acyclic, as containment does not form cycles. For semantic edges, we adopt conventions that respect acyclicity: references are oriented in a way that avoids circular chains of dependence for the purposes of the core model, and preconditions are always directed from the fragment that sets the condition to the fragment that depends on it. An example of circular dependence in the Italian Criminal Code is Article 624, which refers to Article 625 for particular circumstances. Then, Article 625 refers to Article 624. With our convention, only one edge will be present in the graph (e.g., from Article 624 to Article 625). Moreover, with this convention, it is possible to trace an *activation*

path from more general or earlier interpretative clauses to more specific operative provisions without encountering directed cycles. In situations where the legal text suggests genuinely mutual or circular relations, these can be handled by choosing a canonical orientation at the structural–semantic level and, if needed, by recording the symmetry or circularity in additional annotations, rather than by introducing cycles in the core graph.

3.2.3. Properties of Nodes and Edges

We conclude the description of the graph model by discussing the properties attached to nodes and edges. While node and edge types determine the abstract structure of the graph, properties provide the concrete information that ties the model back to the legal text, supporting both human inspection and automated processing. Formally, properties are captured by the functions $\alpha_V : V \to \mathcal{M}_V$ and $\alpha_E : E \to \mathcal{M}_E$, where $\mathcal{M}_V$ and $\mathcal{M}_E$ denote the spaces of node and edge attributes, respectively. For each node $v \in V$, $\alpha_V(v)$ is a finite collection of key–value pairs describing that structural element; for each edge $e \in E$, $\alpha_E(e)$ collects analogous information about the corresponding relation.

For nodes, the primary role of properties is to anchor each structural element of the graph to its counterpart in the Criminal Code. A node of type Book, Title, Article or Comma is typically associated with its official textual content or the relevant excerpt, and a set of identifiers such as the book or article number, the index of the comma, and any stable URN/URI used in external legal information systems. Additional classification tags can be used to place nodes within broad macro-areas of criminal law (e.g., crime against a person or crime against a property), whenever such information is useful for navigation or query formulation. These properties do not alter the graph structure: they enrich it with the information needed for legal interpretation and for cross-linking with other resources.

For edges, properties serve a complementary purpose. An edge of type contains or partOf may only require minimal metadata, such as a reference to the consolidated version of the code from which the hierarchy was derived. By contrast, semantic edges such as refersTo, preconditionOf or excludes often benefit from more detailed annotations. In particular, $\alpha_E(e)$ can include a pointer to the precise textual fragment that justifies the relation (e.g., the sentence in which a reference is made or a precondition is stated), as well as any qualifiers that restrict its scope (such as a brief description of the context in which an exclusion applies). Temporal information can also be attached to edges, for instance, when a particular precondition or exclusion relation is introduced or modified by a specific legislative act.

Overall, the separation between types and properties reflects the model's design philosophy. The essential legal structure and the main semantic relations are encoded in node and edge types, ensuring a clear and stable backbone for reasoning. Properties provide the necessary detail to connect this backbone to the underlying legal text, to express auxiliary information such as identifiers and temporal data, and to document the provenance and validation status of each element. This combination makes the representation both technically usable to automatic reasoning and sufficiently transparent for legal experts to inspect, contest, and maintain over time. Figure 2 illustrates an example of the proposed graph model. The figure highlights several node types – namely Book, Title, Chapter, Article, and Comma – as well as two categories of edges. The first category represents the hierarchical organization of the graph and consists exclusively of edges labeled contains, while the second category comprises semantic relationships encoded by edges labeled refersTo. It is important to note that an Article is formally defined as a collection of *at least* one or more *commi*. Consequently, any semantic relationship involving an Article, such as the one shown in Figure 2, should be interpreted as linking a specific Comma within that article (in the example, the first *comma*) to a Comma belonging to another Article (e.g., *comma B.3* in the figure).

3.3. Leveraging Graph Properties on the Italian Criminal Code

Once the Italian Criminal Code has been represented as a directed acyclic graph G, the formal properties of this structure can be exploited directly. These provide a mathematical foundation that can be leveraged in different ways to support legal questions about structure, dependency, and impact within the Criminal

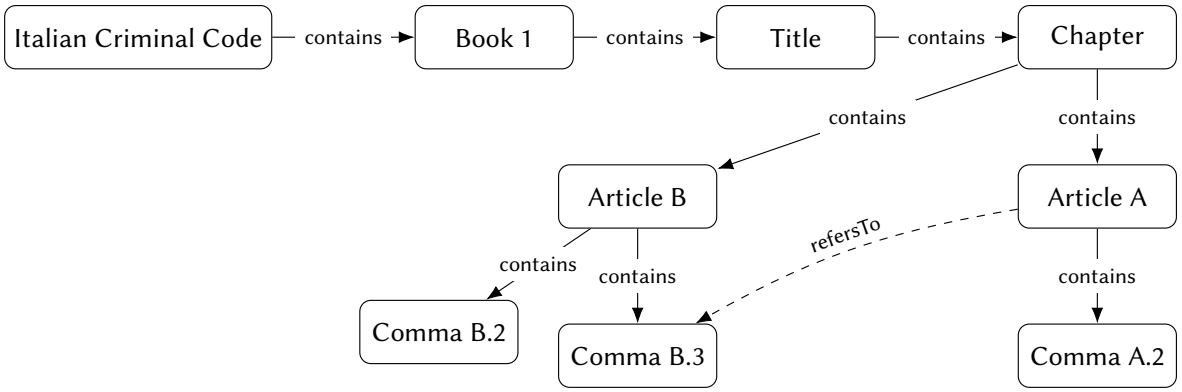

Figure 2: Schematic graph representation of the Italian Criminal Code.

Code, while keeping all intermediate artifacts inspectable by human jurists.

A first class of operations relies on *reachability* in G. Given a node $v \in V$, the set of nodes reachable from v along contains edges corresponds to the structural fragment of the code that is syntactically subordinated to v (e.g., all *commi* under a given article). Likewise, restricting reachability to semantic edges of a given type yields dependency sets: reachability along preconditionOf encodes all provisions on which a given fragment depends, and reachability along refersTo captures chains of explicit references.

Then, because node and edge properties are attached via functions α_V and α_E without altering the underlying DAG structure, all these operations can be combined with filtering and projection steps. For example, one can compute reachability by just exploiting edges satisfying certain property predicates (such as a given temporal regime), or restrict induced subgraphs to nodes annotated as belonging to a particular macro-area of criminal law. Formally, this corresponds to working with subgraphs of G defined by predicates on V and E, while preserving acyclicity and the associated benefits (existence of topological orders, well-founded dependency relations).

3.4. Using AI and LLMs over the Graph as a Knowledge Base

A crucial application of the DAG G is its exploitation as a structured knowledge base for the Italian Criminal Code. AI techniques, particularly LLMs, are not intended to replace formal representation, but to support tasks such as querying provisions in a manner accessible to human users.

A primary role of LLMs is as an interface for *natural language querying* of the graph. Given a user query formulated in natural language, the LLM interprets the question and translates it into graph operations: selecting nodes whose textual properties in α_V match the relevant concepts, restricting the search to certain macro-areas, and following specific edge types (e.g., preconditionOf or refersTo) to retrieve related provisions. The actual selection and traversal are performed on G using formal graph operations (such as reachability or induced subgraphs), while the LLM is used to map natural language into the structured query space. The result of the query (a set of nodes, edges, and their properties) is then summarised back into natural language for the user.

Then, the graph supports *semantically enriched search* [18]. Since each node is embedded in a network of hierarchical and semantic edges, search can be extended beyond simple keyword matching. Starting from nodes whose textual properties match a term of interest, the system can follow edges of selected types to include related provisions (e.g., all articles whose applicability depends on a given general clause, or all commi that are excluded by a particular rule). LLMs can help specify, in natural language, which relations should be taken into account, and then interpret the graph-based results. The underlying retrieval [19], however, remains an operation over the typed DAG G.

Finally, the graph serves as a knowledge base, enabling *structured exploration* driven by user queries. Users can request views such as "all provisions that must be considered before applying Article X", or "the set of provisions potentially impacted by changes to this comma". These correspond to graph-theoretic constructions over the DAG. LLMs can support this exploration by helping users formulate

such requests in natural language and by presenting the resulting subgraphs in an interpretable way, but the core knowledge base remains the formally defined graph G with its nodes, edges, and properties.

In all these scenarios, the methodological emphasis is the same: the graph is the primary, structured repository of knowledge about the Criminal Code; AI and LLMs are used to query within this repository, translating between natural-language interactions and well-defined operations on the underlying graph, while human jurists retain control over the interpretation and use of the retrieved information.

3.5. Explainability of AI Solutions and Legal Frameworks

The aforementioned knowledge base is interpretable by design, thanks to its graph structure, which describes real-world entities and their relationships. In this study, the choice of modeling the Italian Criminal Code by a DAG, following the structure shown in Figure 2, is the most intuitive approach to providing a knowledge base to a LLM. Such a structure enables the usage of advanced techniques, such as RAG, to improve the inference performances, as well as to minimize potential issues, such as inaccuracies, which are unacceptable in a law enforcement scenario. On the other hand, LLMs are deep neural networks with billions of parameters, which makes it difficult to implement interpretation mechanisms. Within the scope of this work, the computational modelling of the Italian Criminal Code can be understood as a foundational tool for the development of AI support systems in the judicial domain, rather than as a substitute for human legal judgment. This perspective should be situated within the supranational regulatory framework established by the Artificial Intelligence Act (Reg. EU 2024/1689) [20]. Pursuant to Article 6(2), AI applications listed in Annex III are classified as "high-risk", with point 8 covering systems used by or on behalf of judicial authorities to assist in the identification and interpretation of facts and law, or in the application of the law to a specific set of facts, including similar uses in alternative dispute resolution. At the same time, Article 6(3) introduces derogations, providing that AI systems listed in Annex III are not considered high-risk where they do not pose a significant risk to health, safety or fundamental rights, including where they do not materially influence the outcome of the decision-making process. In particular, this applies where the AI system is: limited to performing preparatory or procedural tasks, improving the outcome of a previously completed human activity, or identifying patterns without replacing or influencing human assessment, provided that appropriate human oversight is ensured. From this standpoint, computational representations of criminal law employed for analytical, structuring, or preparatory purposes may fall outside the high-risk category, although the scope of concepts such as "preparatory tasks" and "narrow procedural tasks" remains open to interpretation and warrants further regulatory clarification.

4. Discussion

In this section, we discuss the methodology to model the Italian Criminal Code as a knowledge graph. In particular, we address the advantages of this modelization and its main limitations.

4.1. Legal Perspective

From a legal-comparative perspective, the model proposed in this paper, while focused on the Italian Criminal Code, offers significant potential for adaptation to other civil law jurisdictions with similarly codified legal systems. Legal systems that feature a hierarchical structure, with laws organized into books, titles, and articles (e.g., the French *Code pénal* or the German *Strafgesetzbuch*) would find this graph-based approach particularly relevant. In these systems, the interrelations between general and special provisions, as well as the extensive use of cross-references, mirror the structure of the Italian Criminal Code, making the model a suitable framework for supporting automated reasoning and legal analysis. However, the application of this model in jurisdictions with structural differences would necessitate some modifications. For instance, legal systems with distinct classifications or hierarchical structures (e.g., the Danish *Straffeloven*) might require adaptations in how relationships are represented within the graph. Adjusting the ontology to accommodate the features of each code would ensure the

model's compatibility with the respective legal frameworks. Nonetheless, the fundamental idea of using a graph to represent the relational nature of legal norms remains highly transferable across codified Civil Law systems.

4.2. Technical Perspective

A central technical choice of this work is to keep nodes purely structural and to push semantic content into edge types and properties. This yields a relatively small and stable node type space while allowing the edge type space T_E and attribute spaces $\mathcal{M}_V, \mathcal{M}_E$ to carry the interpretative richness required by legal analysis. From a modelling perspective, this reduces the risk of overfitting the representation to a particular logical formalism or application scenario, since introducing a new semantic relation typically amounts to adding a new edge type and properties rather than redesigning the node ontology.

The main cost of this choice is that certain legal constructs that might naturally be seen as entities (e.g., offences, circumstances, cross-cutting legal concepts) are not represented as first-class nodes in the core graph. Instead, they must be reconstructed from patterns of structural nodes and edges, or introduced at a higher, derived layer. Technically, this yields a clean separation between the *structural* knowledge base and any *conceptual* or *logical* abstractions built on top of it, at the price of an additional mapping step for more sophisticated reasoning.

The imposition of directed acyclicity is another key design decision. On the one hand, the DAG constraint simplifies reasoning: algorithms for reachability, dependency analysis, and rule evaluation become straightforward when the underlying graph is acyclic, and the dependency relation is well-founded. On the other hand, some parts of the legal text may naturally give rise to cycles (e.g., mutual references or circular exclusions). Our approach addresses these cases by resolving directionality at modelling time or by representing potentially cyclic aspects in metadata rather than in the core edge structure. This keeps the main dependency relation acyclic, but means that some mutual relationships are treated asymmetrically and must be recovered by considering both directions when needed.

From a complexity perspective, the aforementioned graph properties can be evaluated with standard algorithms on a DAG. Given the size of the Italian Criminal Code, these are easily handled by off-the-shelf graph engines and can be executed interactively. Scalability issues arise not from the number of structural nodes, but from the growth of semantic edges and the richness of their properties. If multiple semantic relations are encoded and annotated in detail, efficient indexing (by edge type and key properties) becomes essential to restrict queries to relevant subgraphs. Here, acyclicity helps again: it bounds path lengths and thus the cost of dependency and impact computations.

Incremental updates are also simplified by the DAG structure. Legislative amendments typically affect a limited subset of articles and *commi*; because the graph is structurally aligned with the Criminal Code, updates can be localised to the incident edges and the relevant semantic neighbourhood. Cycle detection on new semantic edges can be performed incrementally by checking whether a proposed edge introduces a cycle, rather than re-analysing the entire graph.

Treating G as a knowledge base has direct implications for the use of AI and LLMs. In our evaluation, LLMs do not operate solely on raw text; instead, they leverage the graph as a structured input and output space. This enables hybrid query pipelines in which a natural-language query is first interpreted by an LLM into a formal query over G, then executed by a graph engine using exact algorithms, and finally summarised back to the user by the LLM. This separation is crucial: correctness and completeness of query results are guaranteed by deterministic operations on the DAG, while the non-deterministic interpretation of natural language is confined to the *interface* layer. Technically, this also allows formal queries to be logged and replayed independently of their natural-language formulations, which is important for reproducibility and auditability.

4.3. DAG Construction

A practical advantage of the proposed representation is that the DAG G can be constructed in a largely systematic way starting from a machine-readable version of the Italian Criminal Code. The key

technical point is that the code's hierarchy already induces an acyclic structure, which can be extracted deterministically and then used as the backbone on top of which additional semantic edges are added under explicit acyclicity constraints. The construction process starts with a consolidated, machine-readable text that is ingested and normalised so that structural markers are explicit and consistent. The output of this step is a sequence of textual blocks, each of which represents a structural unit of the code.

From this text, the structural nodes V and their types τ_V are generated by segmentation, thus each detected unit becomes a node. Hierarchical edges can be added deterministically, since whenever a unit is nested within another, an edge is created from the container to the contained element. By construction, the subgraph induced by these edges is a rooted, tree-like structure and therefore acyclic. This phase yields an initial DAG that captures the full navigable hierarchy of the Criminal Code.

The second phase enriches the backbone with semantic edges while preserving acyclicity. Candidate relations can be extracted from the text by identifying explicit linguistic patterns and mapping them to edges between the corresponding structural nodes. Each relation is represented as a typed directed edge and annotated with edge properties. At this stage, directionality encodes a dependency order that must remain compatible with the DAG constraint. Maintaining acyclicity requires an explicit enforcement mechanism when semantic edges are introduced. When a cycle is detected, proper strategies must be applied (e.g., selecting a canonical orientation).

Since amendments typically modify a bounded fragment of the code, updates can be implemented by re-segmenting only the affected structural region, updating the associated nodes and edges, and then re-evaluating only the required semantic edges. Because acyclicity is enforced locally at edge insertion time, the global DAG invariant can be preserved without recomputing the entire structure. In technical terms, this supports a maintainable workflow in which the structural backbone remains stable, while semantic relations evolve through controlled, auditable updates tied to changes in the source text.

5. Conclusions

In this paper, we have investigated the feasibility of representing the Italian Criminal Code as a structured knowledge base and of using this representation as a foundation for automated inference. Starting from the observation that purely textual access to the code limits the systematic application of formal methods and AI techniques, we argued for a graph-based model in which nodes correspond to the structural elements of the code (books, titles, articles, and commi) and edges encode both hierarchical and selected semantic relations. The resulting object is a Directed Acyclic Graph (DAG) with typed nodes and edges, enriched by properties that tie each graph element back to the underlying legal text.

We formalised this model using standard graph-theoretic notation, distinguishing the node and edge type spaces from the attribute layers that carry metadata. Then, we emphasised the role of the graph as a knowledge base for AI methods, particularly LLMs. LLMs are used to assist in querying, searching, and comparing provisions over the graph, by translating between natural-language requests and well-defined graph operations, and by summarising the results for human users.

Some directions for future work follow. On the legal side, the model could be extended to incorporate richer semantic relations or temporal layers for successive reforms. On the technical side, more formalisms could be explored on top of the DAG, and concrete query languages tailored to criminal-law tasks could be defined. From an AI perspective, systematic studies are needed to assess how reliably LLMs can assist in formulating graph queries and in supporting comparative analysis, and how their suggestions can be best integrated into a human-in-the-loop workflow.

Despite these limitations, the proposed framework suggests that a graph-based, formally specified representation of the Italian Criminal Code can provide a coherent bridge between doctrinal legal knowledge, formal reasoning techniques, and AI-assisted interaction. By keeping the structure explicit, localizing the semantics in edge types and properties, and keeping human jurists firmly in the loop, it offers a promising foundation for the development of transparent, controllable tools that augment legal reasoning in the criminal domain.

6. Acknowledgements

This work was partially supported by the SERICS project (PE00000014) under the NRRP MUR program, funded by the EU - NGEU.

Declaration on Generative AI

During the preparation of this work, the authors used ChatGPT-5 and DeepL in order to: Grammar and spelling check, text translation and to improve the writing style. After using these tools, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] M. Siino, M. Falco, D. Croce, P. Rosso, Exploring LLMs applications in law: A literature review on current legal NLP approaches, IEEE Access (2025). doi:10.1109/ACCESS.2025.3533217.

[2] X. Wu, L. Xiao, Y. Sun, J. Zhang, T. Ma, L. He, A survey of human-in-the-loop for machine learning, Future Generation Computer Systems 135 (2022) 364–381. doi:10.1016/j.future.2022.05.014.

[3] G. Ufficiale, Codice penale, https://www.gazzettaufficiale.it/sommario/codici/codicePenale, 2025. Gazzetta Ufficiale della Repubblica Italiana.

[4] A. Hogan, E. Blomqvist, M. Cochez, C. d'Amato, G. D. Melo, C. Gutierrez, S. Kirrane, J. E. L. Gayo, R. Navigli, S. Neumaier, et al., Knowledge graphs, ACM Computing Surveys (Csur) 54 (2021) 1–37. doi:10.1145/3447772.

[5] N. Patwardhan, S. Marrone, C. Sansone, Transformers in the real world: A survey on nlp applications, Information 14 (2023) 242. doi:10.3390/info14040242.

[6] A. A. Khaliq, S. Montanelli, Language models for legal NLP: A literature review, in: International Conference on Advanced Information Systems Engineering, Springer, 2025, pp. 326–337. doi:10.1007/978-3-031-94931-9_27.

[7] W. T. Tutte, Graph theory, volume 21, Cambridge university press, 2001.

[8] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, arXiv preprint arXiv:1706.03762 (2017). doi:10.48550/arXiv.1706.03762.

[9] T. Rehman, S. Ghosh, K. Das, S. Bhattacharjee, D. K. Sanyal, S. Chattopadhyay, Evaluating LLMs and pre-trained models for text summarization across diverse datasets, arXiv preprint arXiv:2502.19339 (2025). doi:10.48550/arXiv.2502.19339.

[10] Y. Zhang, S. Li, C. Qian, J. Liu, P. Yu, C. Han, Y. R. Fung, K. McKeown, C. Zhai, M. Li, et al., The law of knowledge overshadowing: Towards understanding, predicting, and preventing LLM hallucination, arXiv preprint arXiv:2502.16143 (2025). doi:10.48550/arXiv.2502.16143.

[11] I. Chalkidis, M. Fergadiotis, P. Malakasiotis, N. Aletras, I. Androutsopoulos, LEGAL-BERT: The muppets straight out of law school, arXiv preprint arXiv:2010.02559 (2020). doi:10.48550/arXiv.2010.02559.

[12] N. M. Gardazi, A. Daud, M. K. Malik, A. Bukhari, T. Alsahfi, B. Alshemaimri, BERT applications in natural language processing: a review, Artificial Intelligence Review 58 (2025) 1–49. doi:10.1007/s10462-025-11162-5.

[13] I. Chalkidis, A. Jana, D. Hartung, M. Bommarito, I. Androutsopoulos, D. Katz, N. Aletras, LexGLUE: A benchmark dataset for legal language understanding in english, in: Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2022, pp. 4310–4330. doi:10.18653/v1/2022.acl-long.297.

[14] D. Licari, G. Comandè, ITALIAN-LEGAL-BERT: A pre-trained transformer language model for italian law., Ekaw (companion) 3256 (2022). doi:10.1016/j.clsr.2023.105908.

[15] A. Tagarelli, A. Simeri, Unsupervised law article mining based on deep pre-trained language representation models with application to the Italian civil code, Artificial Intelligence and Law 30 (2022) 417–473. doi:10.1007/s10506-021-09301-8.

[16] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, et al., Retrieval-augmented generation for knowledge-intensive NLP tasks, Advances in neural information processing systems 33 (2020) 9459–9474. doi:10.48550/arXiv.2005.11401.

[17] M. Fiore, M. Devesas Campos, The algebra of directed acyclic graphs, in: Computation, Logic, Games, and Quantum Foundations. The Many Facets of Samson Abramsky: Essays Dedicated to Samson Abramsky on the Occasion of His 60th Birthday, Springer, 2013, pp. 37–51. doi:10.1007/978-3-642-38164-5_4.

[18] F. Maoro, B. Vehmeyer, M. Geierhos, Leveraging semantic search and LLMs for domain-adaptive information retrieval, in: International Conference on Information and Software Technologies, Springer, 2023, pp. 148–159. doi:10.1007/978-3-031-48981-5_12.

[19] M. Arslan, H. Ghanem, S. Munawar, C. Cruz, A survey on RAG with LLMs, Procedia computer science 246 (2024) 3781–3790. doi:10.1016/j.procs.2024.09.178.

[20] E. Parliament, Regulation (EU) 2024/1689 of the european parliament and of the council of 13 june 2024 laying down harmonised rules on artificial intelligence and amending regulations (EC) no 300/2008, (EU) no 167/2013, (EU) no 168/2013, (EU) 2018/858, (EU) 2018/1139 and (EU) 2019/2144 and directives 2014/90/EU, (EU) 2016/797 and (EU) 2020/1828 (Artificial Intelligence Act) text with EEA relevance., https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32024R1689, 2024.