Detecting Transitional Provisions in EU Legislation with Hybrid AI for Better Regulation

Detecting Transitional Provisions in EU Legislation with Hybrid AI for Better Regulation

Simone Vagnoni
CIRSFID-ALMA AI
University of Bologna
Bologna, Italy
simone.vagnoni3@unibo.it

Monica Palmirani
CIRSFID-ALMA AI
University of Bologna
Bologna, Italy
monica.palmirani@unibo.it

***Abstract*—Maintaining an accessible and up-to-date legislative framework is crucial for ensuring transparent governance and enabling effective citizen participation in democratic processes. This paper examines methodological approaches for the automated detection and modeling of textual modifications within EU Regulations and Directives, with particular attention to transitional provisions and temporal concepts (retroactivity, postponement, relative dates), as well as legislative ones (entry into force, application, sunset clause). We present a comparative analysis of four distinct approaches: (1) a rule-based pipeline utilizing regular expressions to capture standardized legal expressions, (2) a SpaCy-based pipeline leveraging advanced NLP capabilities for entity recognition and pattern matching, (3) a Large Language Model (LLM)-based pipeline using the OpenRouter API for contextual extraction, and (4) RDF/Cellar extraction serving as the benchmark. Each methodology is designed to generate standardized Akoma Ntoso (AKN) metadata annotations, facilitating point-in-time understanding of legislative modifications and temporal validity. The rule-based and SpaCy-based approaches offer computational efficiency and procedural transparency, while the LLM-based method demonstrates potential for handling complex linguistic variations and contextual dependencies. RDF extraction provides authoritative temporal information for evaluation purposes. This comparative framework provides insights into the relative strengths and limitations of each approach, contributing to the ongoing development of advanced legal informatics tools for legislative drafting and eParticipation initiatives.**

***Index Terms*—Better Regulation, Akoma Ntoso, LegalXML, AI and Law, LLM, NLP, EU Legislation**

I. Introduction

In European legislation, but also at the national level, we often assist, especially in innovative regulations (e.g., AI, privacy, security), in using transitional norms to permit society to take the necessary measures to comply with the new commands. Public administrations struggle to track at which moment the law should be applied to a specific concrete case or process (e.g., authorization, permission). The transitional norms define when some legislative provisions enter into force or operate according to some conditions (e.g., in the AIA we have four dates of applications: August 2026, but for Chapter, I and II shall apply from Feb. 2025, Chapter III, Section 4, V, VII, XII and Art. 78 shall apply from 2 August 2025, with exception of Art. 101; the Art. 6(1) and the corresponding obligation shall apply from 2 August 2027). *Transitional* or *Final Provisions* are key partitions in legislative documents that guarantee the smoothly introduction of the norms in the legal corpora. They address legal validity timespans, including entry into force, application dates, postponement, and retroactivity, among others [1]–[3]. Making the law transparent, applicable, and consumable also by eGov services is fundamental to detecting these dates and connecting them to the corresponding provisions. Maintaining an accessible and up-to-date legislative framework is crucial for ensuring transparent governance and enabling effective citizen participation in democratic processes.

This paper focuses on the automated detection and markup of temporal transitional and final clauses in EU legislation. We present four different yet complementary approaches:

- A **Regex-based pipeline**(regular expression) driven patterns and rules that search for typical legislative expressions (e.g., 'shall enter into force"), specifically designed to detect standard language indicating modifications in European legislative texts, as described in the Handbook for EU legal drafting [1].
- A **SpaCy-based pipeline** that leverages *SpaCy*'s Python library NLP capabilities for sophisticated entity recognition and pattern matching tailored to legal texts.
- A **Large Language Model (LLM)-based pipeline** implemented through an external API (OpenRouter), utilizing advanced language understanding for contextual extraction.
- **RDF/Cellar Extraction** serving as the authoritative benchmark by directly extracting temporal metadata from EU Cellar RDF[2].

All pipelines ultimately are aimed to generate Akoma Ntoso (**AKN**) markup, adding structured semantic data to each transitional or final provision, thus facilitating point-in-time or prospective analysis.

A. Motivation and Theoretical Principles

A robust solution for identifying transitional or final provisions offers manifold benefits:

[1]https://www.consilium.europa.eu/media/67390/joint_handbook_en_01-october-2023_clean_def_final.pdf

[2]https://op.europa.eu/en/web/cellar/cellar-data

1) **Enhanced Access and Transparency:** Point-in-time versions of legislation become easier to generate, ensuring that citizens and legal practitioners always access the correct text in effect on a particular date.
2) **Improved Drafting:** Legal drafters can automatically integrate discovered transitional or final clauses from previous legislation, thus reducing drafting errors and improving consistency.
3) **Better eParticipation:** Automating the consolidation process and facilitating the analysis of transitional clauses allows for more citizen-centric tools, potentially enabling more informed public consultations and discussions.

   To achieve these goals, we need to analyse the anatomy of these specific provisions. A trasitional provisions includes temporal modifications that have the following elements:

   - reference of the destination of the temporal modification (e.g., Chapter I);
   - action (e.g., shall apply, enter into force);
   - temporal parameters (e.g., from 2 February 2025);
   - conditions (e.g., with the exception of Article 101).

   In Akoma Ntoso there are metadata for modelling all the temporal modifications that are usually stated in the transitional and final provisions. Usually the taxonomy caputres the temporal modifications according the starting and endig moment of the enter into force or entery into application.

```
<xsd:restriction base="xsd:string">
    <xsd:enumeration
    value="entryIntoForce"/>
    <xsd:enumeration
    value="endOfEnactment"/>
    <xsd:enumeration
    value="postponementOfEntryIntoForce"/>
    <xsd:enumeration
    value="prorogationOfForce"/>
    <xsd:enumeration
    value="reEnactment"/>
    <xsd:enumeration
    value="unconstitutionality"/>
 </xsd:restriction>
```

Listing 1: List of ”forceMods” modifications.

```
<xsd:restriction base="xsd:string">
  <xsd:enumeration
  value="entryIntoEfficacy"/>
  <xsd:enumeration
  value="endOfEfficacy"/>
  <xsd:enumeration
  value="inapplication"/>
  <xsd:enumeration
  value="retroactivity"/>
  <xsd:enumeration
  value="extraefficacy"/>
  <xsd:enumeration
  value="postponementOfEfficacy"/>
  <xsd:enumeration
  value="prorogationOfEfficacy"/>
</xsd:restriction>
```

Listing 2: List of ”efficacyMods” modifications.

B. Paper Structure

The remainder of this paper is organized as follows: Section II discusses relevant background and related work. Section III introduces the overall methodology, and Sections IV to VII describe each of the four approaches: Regex-based, SpaCy-based, LLM-based, and RDF/Cellar Extraction. Section VIII details our dataset and experiment settings. Section IX presents evaluation results and analyses. Finally, Section X concludes and suggests directions for future work.

II. Related Work

The detection of textual modifications in legislative acts is a well-explored topic [1], [3], with various Natural Language Processing (NLP) tools successfully applied across different languages [2], [4]. However, the specific identification and analysis of temporal provisions, such as entry into force, application, end of validity, pose unique challenges due to their complex formulations (e.g. *”This Regulation shall enter into force on the day following that of its publication [...] It shall apply from 1 January 2020. However, Article 9 shall apply from 1 February 2020*”) and the presence of additional conditions (e.g., temporal postponement, retroactivity, and relative dates).

Building upon this foundation, our work aims to delve deeper into understanding the main elements within textual modifications, especially in scenarios involving intricate legal language. By leveraging *Akoma Ntoso (AKN) XML* - the current standard in European institutions [5] - we seek to improve the existing body of research on European legislative document analysis. Our study provides a detailed, quantitative analysis of temporal provisions over a decade of EU legislation, offering comprehensive classification and quantification of legislative wordings. Unlike previous studies [6], [7], which focused primarily on detecting and interpreting changes, our research extends to a systematic analysis of the verbs used on the temporal legal significance of such.

Furthermore, we incorporate insights from [8], who introduced advanced tools for processing temporal information in legal texts, enhancing our ability to capture and model temporal provisions accurately. This enriched approach not only deepens the understanding of legal drafting practices but also facilitates the development of specialized editors for better management of amending provisions [9], [10]. Furthermore, we introduce a methodological framework for future computational analyzes of European legislative modifications, supported by a data set annotated in AKN and derived from a baseline of ground truth [11]. This dataset serves as a

valuable resource for subsequent research, particularly those leveraging Artificial Intelligence techniques.
Akoma Ntoso (AKN) [5], [12], [13] provides an XML schema to capture legislative semantics, including `<efficacyMod>` for temporal changes, but bridging the gap between unstructured text and structured markup remains challenging. This work addresses that gap by evaluating four pipelines for extracting and annotating transitional or final provisions with temporal metadata, using RDF extraction as the authoritative benchmark.

III. Methodology

We define a multi-stage methodology to identify, extract, classify, and annotate transitional or final provisions. This approach embodies principles of hybrid AI, combining rule-based techniques, statistical NLP, and deep learning to leverage the complementary strengths of different artificial intelligence paradigms. The main steps include:

a) **Document Parsing**: Transform original Formex v4[3] EU acts into Akoma Ntoso XML, preserving article and paragraph boundaries.
b) **Provisional Filtering**: Identify potential transitional or final articles/paragraphs by scanning headings (e.g., "Final Provisions", "Entry into Force") or by detecting known expressions (e.g., "shall enter into force").
c) **Content Analysis**: Extract relevant dates, references, conditions, and exceptions.
d) **Classification**: Determine the specific effect type (the primary ones: entry_into_force, application, sunset_clause, as well as other attributes: postponed_application, retroactive_application, etc.).
e) **AKN Annotation**: Generate `<analysis>` elements in AKN, referencing `<efficacyMod>` or appropriate markup.

Each pipeline (Regex-based, SpaCy-based, LLM-based) implements steps 3 and 4 differently, leveraging their respective strengths. RDF/Cellar Extraction directly provides the ground truth data for evaluation.

IV. RDF/Cellar Extraction (Ground Truth)

A. Overview

RDF/Cellar Extraction serves as the authoritative benchmark by directly extracting temporal metadata from EU Cellar RDF. This method utilizes standardized properties to obtain precise temporal information through well-defined ontologies and controlled vocabularies.

B. Ontologies and Vocabularies

The extraction relies on two core ontologies and several controlled vocabularies maintained by the EU Publications Office:

1) Core Ontologies:

- **CDM (Common Data Model)** [4]:
  The primary ontology defining EU legal documents' structure, including:
  - Document types and classifications
  - Legal dates and temporal properties
  - Inter-document references
  - Standard metadata elements
- **Annotation Ontology** [5]:
  Manages document annotations, specifically:
  - Temporal annotations and date comments
  - Document structure references
  - Modification types and roles

2) Controlled Vocabularies: [6] Three key authority tables support temporal and structural annotations:

- **Type of Date (fd_335)**:
  Standardizes temporal classifications:
  - EV (Entry into Force)
  - MA (Application)
  - MA/PART (Partial Application)
  - DATPUB (Publication Date)
- **Role/Type of Modification (fd_375)**:
  Defines modification categories:
  - R (Replacement)
  - A (Repeal)
  - J (Insertion)
  - T (Postponement)
- **Location References (fd_370)**:
  Provides location reference codes:
  - ART (Article)
  - PAR (Paragraph)
  - AN (Annex)
  - SEC (Section)

C. Implementation

Temporal metadata is extracted using standardized RDF properties such as:

- `resource_legal_date_entry-into-force`
- `resource_legal_date_application`
- `resource_legal_date_end-of-validity`

These properties provide authoritative temporal information that can be used to evaluate the accuracy and reliability of the other three extraction approaches. The metadata is accessible in multiple formats (RDF, JSON-LD, SKOS/XML) through the EU Publications Office's vocabulary service.
The "end of validity" dates extraction undergo additional filtering to exclude those associated with specific legal relationships i.e:

- `amends`

[3]https://op.europa.eu/it/web/eu-vocabularies/formex

[4]https://op.europa.eu/en/web/eu-vocabularies/dataset/-/resource?uri=http://publications.europa.eu/resource/dataset/cdm

[5]http://publications.europa.eu/ontology/annotation#

[6]https://op.europa.eu/en/web/eu-vocabularies/authority-tables

- `resource_legal_amends_resource_legal`
- `work_is_repealed_by_work`
- `resource_legal_based_on_resource_legal`

These relationships indicate that the legal basis for the end of validity is expressed in another document, making them not relevant for the temporal provisions extraction.

D. Strengths and Limitations

RDF/Cellar Extraction offers high precision and reliability as it leverages official metadata through standardized ontologies and vocabularies. However, it is limited to the extent of the available RDF data and may not cover all nuances present in the legislative texts. For instance, complex date references like "shall enter into force when Union law ceases to apply" (Regulation 2019/592) cannot be adequately represented in fixed ontologies, and some documents (particularly corrigenda) contain no temporal annotations at all. The structured nature of the vocabularies ensures consistency but may not capture more complex or implicit temporal relationships.

The vocabularies are continuously maintained and updated by the EU Publications Office, ensuring their relevance and accuracy for legal document analysis. They can be accessed and explored through various interfaces, including web-based tools and direct API access, facilitating integration into automated extraction systems.

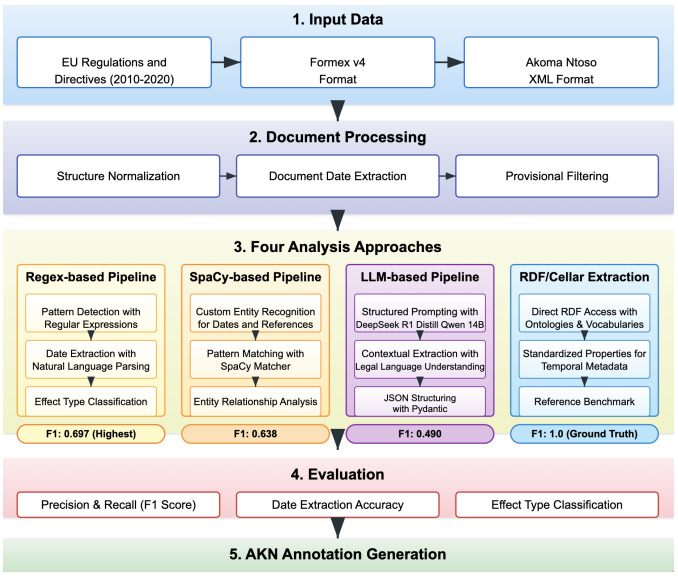

Fig. 1: Overall workflow of the transitional provisions detection process. The pipeline starts with document parsing, followed by provisional filtering, content analysis, classification, and finally AKN annotation.

V. Regex-based Pipeline

A. Overview

We first developed a *Regex-based approach* that seeks known textual patterns and triggers within each legislative provision. The pipeline uses Python's `re` library, along with date parsing (via `dateutil`) and references extraction for both internal references (e.g., "Article 5", "This Regulation") and external references (e.g., "Regulation (EU) No 1308/2013").

B. Implementation

A simplified core snippet is shown in Listing 3.

```
temporal_patterns = {
    'entry_into_force': re.compile(
      r'\b(?:shall|will)\s+enter␣into␣force'
      r'|\benters\s+into\s+force\b',
      re.IGNORECASE
    ),
    'application': re.compile(
      r'\b(?:shall|will)\s+apply'
      r'(?!.*until)'
      r'|\bapplies\b',
      re.IGNORECASE
    ),
    'sunset_clause': re.compile(
      r'\b(?:expire|cease␣to␣apply)'
      r'|\b(?:expires)\b',
      re.IGNORECASE
    )
}
```

Listing 3: Regex-based Temporal Provision Analysis

The algorithm performs:

- **Detection:** Identifies relevant expressions using handcrafted regular expressions.
- **Date Extraction:** Uses natural language date parsing to find explicit or relative dates (e.g., "on 1 January 2020" or "20 days following publication").
- **Classification:** Infers whether it is a *entry into force*, *application*, or *end of validity* (also known as *sunset clause*) following the drafting wording, as well as *postponed* or *retroactive* application by comparing the extracted date with the official publication date (`doc_date`).
- **AKN Markup Generation:** Constructs the appropriate `<efficacyMod>` or `<textualMod>` block, referencing the extracted textual nodes.

C. Strengths and Limitations

This rule-based pipeline is transparent and computationally efficient. However, it may not generalize easily to novel or complex phrasing in transitional provisions. It also requires regular updates of the patterns for new legislative language conventions or unforeseen styles.

VI. SpaCy-based Pipeline

A. Overview

The *SpaCy-based approach* leverages SpaCy's advanced NLP capabilities for more sophisticated entity recognition and pattern matching tailored to legal texts. This pipeline utilizes custom entity recognition for dates, references, and temporal expressions, combining both

rule-based patterns and machine learning-based entity detection.

B. Implementation

The pipeline performs:

- **Detection:** Uses SpaCy's Matcher to identify temporal expressions based on defined patterns.
- **Entity Recognition:** Utilizes SpaCy's NER functionality with the en_core_web_sm model, enhanced with a custom EntityRuler component. The NER component was configured to recognize DATE, TIME entities, while the EntityRuler was trained to identify specialized legal references like "Article X" or "Paragraph Y". This hybrid approach combines pre-trained NER capabilities with domain-specific patterns.
- **Classification:** Maps detected patterns to specific provision types.
- **AKN Markup Generation:** Constructs the appropriate AKN elements based on extracted information.

C. Strengths and Limitations

The SpaCy-based approach offers improved handling of linguistic variations and contextual dependencies compared to purely regex-based methods. It benefits from SpaCy's robust NLP capabilities, allowing for more accurate entity recognition and pattern matching. However, it requires more computational resources and may still face challenges with highly complex or ambiguous legal language without further customization.

VII. LLM-BASED PIPELINE

A. Overview

The *LLM-based approach* leverages the OpenRouter API in conjunction with the *DeepSeek R1 Distill Qwen 14B* model, a distilled large language model based on Qwen 2.5 14B, using outputs from DeepSeek R1. This state-of-the-art model incorporates chain-of-thought reasoning, enabling advanced language understanding essential for the contextual extraction of temporal provisions. Our pipeline employs structured prompting to extract temporal information, returning standardized JSON data that is subsequently structured using *Pydantic*, a data validation and settings management library for Python.

B. Implementation

A sample implementation snippet is shown in Listing 4.

```
"""Extract all temporal provisions from
the following EU legislative text.
For each provision, provide:
- 'main_effect_type' ('entry_into_force',
  'application', or 'sunset_clause')
- 'provision_date' (YYYY-MM-DD)
- 'text_snippet' (only the sentence that
  motivated the decision)
- 'references' (list)
- 'secondary_attributes' (i.e., 'retroactivity',
  'relative_date', 'conditions', 'exceptions')
- 'scope' ('general' or 'specific')
- 'target_of_mod' (string)
- 'is_partial' (boolean)
- 'is_retroactive' (boolean)
- 'relative_date' (boolean)
Ensure all dates are in 'YYYY-MM-DD' format.
If a date is relative, calculate based
on 'document_date'.
Document publication date:
{document_date.strftime('%Y-%m-%d')}
Text: {text}
Respond only with a JSON array of objects
with the specified fields."""
```

Listing 4: LLM-based Temporal Provision Analysis - Example Prompt

C. Pipeline Workflow

The LLM-based pipeline encompasses the following steps:

a) **Prompt Engineering**: We design structured prompts that guide the LLM to focus on extracting relevant temporal information related to transitional or final provisions. This ensures that the responses are both comprehensive and standardized.

b) **Minimizing the data to proces**: We made the LLM process only the last part of the legal text, which in almost all the cases contains the temporal provisions, for a matter of efficiency.

c) **Contextual Extraction**: Utilizing the LLM's deep understanding of legal language, the pipeline accurately identifies and extracts temporal provisions, even within complex and nuanced contexts.

d) **JSON Structuring with Pydantic**: Extracted data is validated and structured using *Pydantic*, ensuring that the JSON output adheres to the predefined schema. This step enhances data integrity and facilitates seamless integration with downstream processes.

e) **AKN Markup Generation**: The structured JSON data is then transformed into appropriate *Akoma Ntoso (AKN)* markup, embedding the extracted temporal provisions within the legislative document's semantic framework.

D. Strengths and Limitations

The LLM-based approach offers significant advantages in handling a wide range of linguistic variations and complex contexts, providing unmatched flexibility and adaptability compared to traditional methods. Its ability to comprehend and interpret nuanced legal language allows for more accurate extraction of temporal provisions. Additionally, the integration of *Pydantic* ensures that the extracted data is well-structured and validated, standardizing the outputs in order to be processed and compared.

However, this approach is computationally intensive and requires substantial resources, which may limit its scalability. In fact we used a free model API, but with limited tokens available daily. Moreover, meticulous prompt engineering is essential to mitigate potential hallucinations or inaccuracies in the LLM's responses. Dependence on external APIs also introduces considerations regarding data privacy and latency, which must be addressed in large-scale implementations.

VIII. Dataset and Experiment Setup

We tested all four pipelines on a dataset consisting of **15,286** EU **Regulations and Directives** (2010–2020), acquired from the European Publication Office in Formex v4, then converted into AKN [5]. We excluded corrigenda or purely minor editorial documents.

A. Data Preprocessing

- **Structure Normalization:** We aligned article, paragraph, and point eIDs in the AKN representation.
- **Document Date Extraction:** We parsed the `publication/@date` or `FRBRdate[@name="generation"]/@date` for referencing in retroactive/postponement checks.
- **Ground Truth:** We used RDF/Cellar extraction as the authoritative benchmark. Additionally, we performed a manual review of 200 random documents to validate the presence of transitional/final provisions and ensure the accuracy of RDF metadata.
- **Removing of the placeholder dates:** Approximately a hundred RDF entries have application dates set to `1001-01-01` or `9999-99-99`, indicating no start or end date, so we removed them from the dataset since there are no wordings for them in the legal text.

B. Implementation Details

- **Integration with AKN XML Processing:** All pipelines were integrated with the AKN XML processing framework to ensure consistent handling of legislative documents.
- **Common Preprocessing and Text Cleaning:** Standardized text preprocessing steps, including tokenization, normalization, and removal of irrelevant sections.
- **Standardized Output Format:** All approaches were configured to output data in a common schema to facilitate comparative analysis, taking in count in particular the main provision type, the date, and secondary attributes, such as if the date is relative rather then explicit, retroactive, etc.
- **Performance Monitoring:** Tracked computational resources, processing time, and scalability metrics across different approaches.

IX. Results and Discussion

A. Evaluation Metrics

We measured:

- **Precision, Recall, F1** for correctly detecting transitional/final provisions (binary classification).
- **Date Accuracy**: Validity of extracted dates (ISO 8601 format).
- **Effect Type Classification**: Accuracy in classifying effect types (entry_into_force, application, sunset_clause, etc.).
- **Handling Complex Cases**: Ability to manage relative dates, retroactive provisions, partial applications, multiple dates, and external references.

B. Comparative Results

TABLE I: Detection Performance of Different Approaches

Approach	Precision	Recall	F1 Score	Comments
Regex-based	0.6906	0.7028	0.0.6966	Limited by pattern specificity
SpaCy-based	0.6301	0.6587	0.6376	Better context handling with entity recognition
LLM-based	0.6098	0.4088	0.4895	Balances precision and recall, but lower F1 score compared to others
RDF/Cellar (GT)	1.00	1.00	1.00	Benchmark for evaluation

1) Detection Performance: The **Regex-based Approach** achieved a precision of 69,06% and recall of 70.28%, as shown in Table I, indicating a balanced performance constrained by the specificity of predefined patterns.

The **SpaCy-based Approach** demonstrated a precision of 63.01% and recall of 65.87%, benefiting from SpaCy's advanced entity recognition and context handling capabilities.

The **LLM-based Approach** reached a precision of 60.98% and recall of 40.88%, resulting in a lower F1 score compared to Regex and SpaCy-based methods due to its recall performance.

TABLE II: Date Extraction Accuracy

Approach	Date Accuracy	Comments
Regex-based	48.52% (12807/26397)	Suffers from pattern limitations affecting date extraction
SpaCy-based	31.16% (4385/14073)	Improved extraction with NLP techniques, but lower exact match rate
LLM-based	32.61% (598/1834)	Best extraction accuracy among limited LLM evaluations
RDF/Cellar (GT)	100%	Considered as benchmark

2) Date Accuracy: As illustrated in Table II, all approaches were evaluated based on the validity of extracted dates in ISO 8601 format. The **Regex-based Approach** achieved the highest date accuracy at 45.99%, followed by the **LLM-based Approach** with 33.71%, and the **SpaCy-based Approach** with 31.16%.

TABLE III: Effect Type Classification Accuracy

Approach	Accuracy	Comments
Regex-based	0.6656	Effective in classifying entry into force and application types, but struggles with sunset clauses
SpaCy-based	0.6587	Handles entry into force and application better, limited performance on sunset clauses
LLM-based	0.4088	Superior contextual understanding but lower overall accuracy in classification tasks
RDF/Cellar (GT)	1.0	Benchmark

3) Effect Type Classification: Table III shows that the **Regex-based Approach** achieved an accuracy of 66.56% in classifying effect types, effectively distinguishing between entry into force and application but struggling with sunset clauses. The **SpaCy-based Approach** showed a similar accuracy of 65.87%, benefiting from better context handling. The **LLM-based Approach** had a lower accuracy of 40.88%, indicating challenges in classification despite its contextual capabilities. The **RDF/Cellar (GT)** serves as the perfect benchmark with 100% accuracy.

4) Handling Complex Cases: All approaches were evaluated on their ability to manage complex temporal scenarios such as relative dates, retroactive provisions, partial applications, multiple dates, and external references.

- **Regex-based Approach**: Limited by rigid patterns, struggles with relative dates and multiple date scenarios.
- **SpaCy-based Approach**: Better at handling relative dates and partial applications due to flexible entity recognition, but less effective with retroactive provisions.
- **LLM-based Approach**: Demonstrated superior performance in managing retroactive provisions and multiple dates, leveraging advanced language understanding capabilities.

C. Performance and Resource Utilization

TABLE IV: Processing Time and Resource Requirements of Different Approaches

Approach	Processing Time	Hardware Requirements
Regex-based	$\bar{5}$ minutes	Workstation (32GB RAM, 6 cores)
SpaCy-based	$\bar{3}$ hours	Standard (32GB RAM, 6 cores)
LLM-based	$\bar{4}$ hours (for 5% of the documents)	Cloud API access
RDF/Cellar (GT)	N/A	N/A

As summarized in Table IV, while the **LLM-based Approach** offers enhanced capabilities in handling complex cases, it requires significantly more computational resources and longer processing times compared to the other approaches. The **Regex-based Approach** remains the most efficient in terms of speed and resource usage but compromises on flexibility and recall. The **SpaCy-based Approach** strikes a balance between performance and resource utilization, making it suitable for applications requiring moderate computational resources.

1) Handling Complex Cases: All approaches were evaluated on their ability to manage complex temporal scenarios such as relative dates, retroactive provisions, partial applications, multiple dates, and external references. The **LLM-based Approach** demonstrated superior performance in these areas due to its advanced language understanding capabilities. In contrast, the **Regex-based** and **SpaCy-based Approaches** showed varying degrees of success depending on the complexity of the cases. Specific metrics for handling complex cases were not provided but are qualitatively assessed based on the overall performance metrics.

D. Error Analysis

Common errors across approaches included:

- **Regex-based**: Missed provisions with unconventional phrasing and false positives from ambiguous patterns. Specific cases include:
  - Dates in regulation titles[7].
  - Dates within annex tables[8].
  - Provisions where references are unclear or not explicitly stated[9].
  - External Regulation References: Phrases like:

    "This Regulation shall enter into force on the day following that on which Union law ceases to apply to the United Kingdom."

    [10] pose challenges for external reference extraction.
- **SpaCy-based**: Misclassification of effect types in highly complex sentences and occasional missed entities. Examples include:
  - Supplementary amendments with complex phrasings. [11]
  - External regulation references that lack clear temporal markers.
- **LLM-based**: Occasional hallucinations or incorrect extractions when dealing with highly ambiguous contexts. Specific instances involve:
  - Phrases referencing external regulations without explicit temporal information.
- **RDF/Cellar Extraction**:
  - Corrigenda like `32012R0392R(01)`[12] do not present annotations of provisions, leading to missing entries.

[7]https://eur-lex.europa.eu/legal-content/en/ALL/?uri=CELEX:32020R0383
[8]https://eur-lex.europa.eu/legal-content/en/ALL/?uri=CELEX:32014R0820
[9]https://eur-lex.europa.eu/legal-content/en/ALL/?uri=CELEX:32018R1899
[10]https://eur-lex.europa.eu/legal-content/en/ALL/?uri=CELEX:32019R0592
[11]https://eur-lex.europa.eu/legal-content/en/ALL/?uri=CELEX:32020R2220
[12]https://eur-lex.europa.eu/legal-content/en/TXT/?uri=CELEX:32012R0392R(01)

X. Conclusion and Future Work

We have presented and compared four pipelines for detecting and modeling transitional or final provisions in EU legislation: Regex-based, SpaCy-based, LLM-based, and RDF/Cellar Extraction as Benchmark. The **Regex-based** pipeline provides transparency, simplicity, and efficiency but can fail on complex or unforeseen expressions. The **SpaCy-based** pipeline offers improved flexibility and entity recognition capabilities, balancing performance and resource utilization. The **LLM-based** pipeline using OpenRouter and the **distilled** model excels in adaptability and handling complex linguistic variations, though it demands higher computational resources and careful prompt engineering to mitigate hallucinations. The **RDF/Cellar Extraction** serves as a reliable Benchmark for evaluating the other approaches. While our pipelines demonstrate technical feasibility, practical adoption in public administrations faces challenges such as integration with legacy legal information systems, lack of staff training in AI-based tools, concerns around explainability and reliability, and regulatory procurement constraints. These barriers must be addressed through human-in-the-loop systems and strong institutional collaboration. Moreover, the use of AI in processing legal texts raises ethical and legal concerns, particularly regarding accountability in case of misclassification, transparency of decision-making, and the potential for bias in large language models—factors that must be considered in public-sector deployments

Future work includes merging Regex- and LLM-based approaches into a hybrid system, formalizing prompts (e.g., with DSPy), extending analysis to other types of modifications, and scaling expert-validated evaluations. Automated modeling of transitional or final provisions offers tangible benefits in legal informatics and eParticipation, making legislative documents more transparent, consistent, and easily navigable. Our research serves as a foundation for scaling such approaches across broader legal corpora and languages, ultimately contributing to more efficient legislative processes and enhanced public engagement.

Acknowledgment

This project is conducted with the support of the European Commission funds within ERC HyperModeLex. Grant agreement ID: 101055185.

References

[1] T. Arnold-Moore, "Automatic generation of amendment legislation," in *In ICAIL '97*. ACM, 1997, p. 56–62.

[2] P.-L. Spinosa, G. Giardiello, M. Cherubini, S. Marchi, G. Venturi, and S. Montemagni, "Nlp-based metadata extraction for legal text consolidation," in *International Conference on Artificial Intelligence and Law*, 2009. [Online]. Available: https://api.semanticscholar.org/CorpusID:8444356

[3] M. Palmirani and R. Brighi, "Model regularity of legal language in active modifications," in *Proceedings of the 2009 International Conference on AI Approaches to the Complexity of Legal Systems: Complex Systems, the Semantic Web, Ontologies, Argumentation, and Dialogue*, ser. AICOL-I/IVR-XXIV'09. Berlin, Heidelberg: Springer-Verlag, 2009, p. 54–73.

[4] D. Gianfelice, L. Lesmo, M. Palmirani, D. Perlo, and D. P. Radicioni, "Modificatory provisions detection: a hybrid nlp approach," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Law*, ser. ICAIL '13. New York, NY, USA: Association for Computing Machinery, 2013, p. 43–52. [Online]. Available: https://doi.org/10.1145/2514601.2514607

[5] E. Commission, "Akn4eu – a common structured format for eu legislative documents," EU institutions, Tech. Rep., 2023. [Online]. Available: https://op.europa.eu/it/web/eu-vocabularies/akn4eu

[6] R. Van Gog and T. Van Engers, "Modeling legislation using natural language processing," in *2001 IEEE International Conference on Systems, Man and Cybernetics. e-Systems and e-Man for Cybernetics in Cyberspace (Cat.No.01CH37236)*, vol. 1, 2001, pp. 561–566 vol.1.

[7] M. Palmirani, F. Sovrano, D. Liga, S. Sapienza, and F. Vitali, "Hybrid AI framework for legal analysis of the EU legislation corrigenda," in *Legal Knowledge and Information Systems – JURIX 2021: The Thirty-fourth Annual Conference, Vilnius, Lithuania, 8-10 December 2021*, ser. Frontiers in Artificial Intelligence and Applications, E. Schweighofer, Ed., vol. 346. IOS Press, 2021, pp. 68–75. [Online]. Available: https://doi.org/10.3233/FAIA210319

[8] M. Navas-Loro and V. Rodríguez-Doncel, "Timelex: A suite of tools for processing temporal information in legal texts," in *AI Approaches to the Complexity of Legal Systems XI-XII - AICOL International Workshops 2018 and 2020: AICOL-XI@JURIX 2018, AICOL-XII@JURIX 2020, XAILA@JURIX 2020, Revised Selected Papers*, ser. Lecture Notes in Computer Science, V. Rodríguez-Doncel, M. Palmirani, M. Araszkiewicz, P. Casanovas, U. Pagallo, and G. Sartor, Eds., vol. 13048. Springer, 2020, pp. 260–266. [Online]. Available: https://doi.org/10.1007/978-3-030-89811-3_18

[9] M. Palmirani, F. Vitali, W. V. Puymbroeck, and F. N. Durango, "Legal drafting in the era of artificial intelligence and digitisation," European Commission, Directorate-General for Informatics Solutions for Legislation, Policy HR, Tech. Rep., 2022. [Online]. Available: https://joinup.ec.europa.eu/sites/default/files/document/2022-06/Drafting%20legislation%20in%20the%20era%20of%20AI%20and%20digitisation%20%E2%80%93%20study.pdf

[10] S. Mador-Haim and A. Hershowitz, "Executing united states bills into law: A working application in the united states house," in *Legal Knowledge and Information Systems – JURIX 2023: The Thirty-sixth Annual Conference, Maastricht, The Netherlands, 18-20 December 2023*, ser. Frontiers in Artificial Intelligence and Applications, G. Sileno, J. Spanakis, and G. van Dijck, Eds., vol. 379. IOS Press, 2023, pp. 237–246. [Online]. Available: https://doi.org/10.3233/FAIA230969

[11] J. von Lucke and F. Fitsilis, "Using artificial intelligence in parliament – the hellenic case," in *Electronic Government*, I. Lindgren, C. Csáki, E. Kalampokis, M. Janssen, G. Viale Pereira, S. Virkar, E. Tambouris, and A. Zuiderwijk, Eds. Cham: Springer Nature Switzerland, 2023, pp. 174–191. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-031-41138-0_12

[12] M. Palmirani, *Legislative Change Management with Akoma-Ntoso*. Dordrecht: Springer Netherlands, 2011, pp. 101–130. [Online]. Available: https://doi.org/10.1007/978-94-007-1887-6_7

[13] M. Palmirani and F. Vitali, *Akoma-Ntoso for Legal Documents*. Dordrecht: Springer Netherlands, 2011, pp. 75–100. [Online]. Available: https://doi.org/10.1007/978-94-007-1887-6_6