EINDHOVEN UNIVERSITY OF TECHNOLOGY

Department of Mathematics and Computer Science

Information Extraction from Dutch Administrative Decisions: A Comparison of Supervised NLP Techniques and Unsupervised Large Language Models

Bachelor End Project Report

Giacomo Grazia

Supervisors:

Prof. Dr. Johan Wolswinkel, LL.M.

Harry Nan, PhD candidate

January 21, 2025

Abstract

Transparency policies have become a cornerstone of modern public administration, with governmental authorities increasingly mandated to proactively disclose their decisions. However, the effectiveness of these policies is often undermined by non-standard publication formats, insufficient metadata, and an ever-growing volume of documents.

As a result, the implementation of Information Extraction pipelines using Natural Language Processing remains largely unexplored, particularly for Dutch administrative decisions, which must be fully proactively disclosed by 2026. This research aims to explore methods for extracting key pieces of information, as specified in Article 3.3 of the Dutch Open Government Act, from a subset of Dutch administrative decisions (specifically energy permits), comparing the performance of supervised machine learning algorithms with unsupervised large language models.

While automated Information Extraction has been explored in the legal domain, as exemplified by projects like *LexNLP* for English, Spanish, and German, the application of such methods to Dutch administrative decisions remains unexplored.

This research investigated the feasibility of performing Information Extraction on Dutch administrative decisions and explored whether the reliance on labeled data—often incomplete, unavailable, and very costly to produce—can be overcome by employing unsupervised large language models instead of traditional supervised machine learning algorithms.

Experiments were conducted to evaluate various approaches tailored to each key piece of information, systematically comparing the performance of supervised and unsupervised algorithms to identify their respective strengths and weaknesses.

The results demonstrated that traditional supervised machine learning is almost always capable of extracting the targeted pieces of information and consistently outperformed unsupervised large language models. However, the large language model-based approaches still exhibited decent performance and represent a promising direction for future research, particularly in scenarios where obtaining labeled data is highly challenging.

The GitHub repository containing all the code used to produce the results of this research is available at: `https://github.com/giacomograzia/BEP_CITaDOG`.

Contents

Acknowledgements

I am profoundly grateful to my supervisors, Professor Dr. Johan Wolswinkel and his colleague Harry Nan, for giving me the opportunity to join them in the CITaDOG project and for providing me with invaluable feedback and guidance throughout my research. It has been an honor to work with you and (hopefully) offer new insights into how Open Government policies can be made more effective using NLP and, more broadly, Data Science.

Following that, I would also like to thank the team of developers of the *LinkExtractor* from the Dutch Publication Office (KOOP), especially Dr. Marc van Opijnen and Jan Vlug, for their patience and assistance in helping me understand and implement the *LinkExtractor*. I greatly appreciate the time they devoted to answering my questions, often extending beyond their regular office hours. I am also deeply grateful to Rens Jansen (currently Data Scientist & Legal Engineer at PwC) for his initial feedback on my research. Without his help, I would never have discovered the *LinkExtractor* repository.

Additionally, as this project marks the conclusion of a long and, at times, stormy bachelor's journey, I want to take this opportunity to thank everyone who, academically or otherwise, contributed to my journey and success in this Data Science program.

In particular, I want to express my deepest gratitude to Dr. Lina Ochoa Venegas and my friend and physicist, Silvia Ferri. Without their constant belief in me, I would not have been able to achieve this milestone. Thank you for guiding me and motivating me to persevere, even when I faced challenges I never thought I could overcome.

Next, I want to thank my family for unconditionally supporting my dream of studying in the Netherlands and always believing in me, even when all odds were against me. I will forever be grateful for the unique opportunity they gave me and for standing by me throughout this journey—whether in person or over countless FaceTime calls.

Finally, my gratitude extends to my friends: those with whom I shared this bachelor's journey over the past (almost) three years, exploring Europe together when we weren't in a lecture room, and those who welcomed me with open arms during every trip back home, always making it feel as if I had never been away. I am truly thankful to have you in my life and for always being there for me.

Chapter 1

Introduction

In recent years, government transparency has become a cornerstone of modern public administration. Policies such as the Dutch Open Government Act (*Wet open Overheid*, WOO) aim to increase public oversight by mandating the proactive disclosure of government documents. This initiative empowers citizens, researchers, and policymakers to assess governmental actions, fostering accountability and reducing opacity in administrative processes.

Currently, administrative decisions are typically made public only to the directly involved parties, with some exceptions, such as those by the *Autoriteit Consument & Markt* (Authority for Consumers and Markets) or the *Kansspelautoriteit* (Gambling Authority). As a result, citizens are rarely aware of cases other than their own. Nonetheless, modern open government legislation is increasingly pushing governments to disclose every single case decision to the public[1]. However, even with such measures, it remains challenging to strengthen overall transparency in a context where decisions lack standardization. This, combined with an ever-growing volume of administrative decisions, hinders large-scale analyses and reduces the potential impact of transparency initiatives.

Information extraction (IE) algorithms can be employed to improve the efficacy of such policies. However, the lack of standardized formatting (both within and across governmental authorities) poses a challenge for these tasks too. Therefore, this research will focus on implementing NLP-based IE (Natural Language Processing) on administrative decisions as a first step towards large-scale processing and enhancing transparency policies.

1.1 Challenges in Processing Legal Data from Administrative Decisions

Administrative decisions, particularly permits issued under the WOO, often lack a standardized structure across different issuing authorities. For instance, a permit issued by the *Autoriteit Consument en Markt* is not formatted the same way as one issued by the *Kansspelautoriteit*. Differences arise not only in document layouts but also in the associated metadata, such as dates, parties involved, or legal references. Metadata serves as the backbone for document retrieval, classification, and analysis. Its absence creates significant barriers. For example:

- with **missing parties** it is impossible to identify the party involved in a decision, and retrieving related permits or grouping decisions by involved entities becomes nearly impossible;
- on the other hand, with **missing dates** filtering decisions based on a specific time frame is very complex and can yield partial representations of the legal frameworks during certain time windows.

[1] An indicative schedule outlining which decision categories will need to comply and when is provided in this document published by the *Ministerie van Binnenlandse Zaken en Koninkrijksrelaties*: Bijlage 1: Indicatieve verdeling in tranches en planning, per informatiecategorie (art. 3.3 Woo).

These gaps make comparative studies between decisions labor-intensive and limit the ability to automate document analysis. While metadata could facilitate smarter searches, automated classification, and comparative legal studies, its inconsistency and absence in many documents undermine these possibilities. In fact, there are currently no standardized metadata schemas, such as XML frameworks, universally adopted for Dutch administrative decisions. Authorities often publish their documents in varying formats, including plain text on websites or as attached PDFs, further complicating efforts for consistent data extraction and analysis.

1.2 Leveraging NLP for Efficient Legal Data Processing

NLP offers a promising solution to address these challenges. As a subfield of artificial intelligence, NLP focuses on enabling computers to interpret and process human language. It provides powerful tools for automating the extraction and structuring of information embedded in unstructured legal texts. By employing techniques such as Named Entity Recognition (NER) and document classification, NLP systems can be used to extract key metadata elements (e.g., involved parties, decision dates, legal references, etc.) from administrative decisions. This will enable more efficient document retrieval, enhance search capabilities, and support comparative analyses within and across issuing authorities. Additionally, research on information extraction (IE) in the medical domain has validated that NLP systems are highly scalable [10, 19], making them well-suited to handle the rapidly growing datasets generated by transparency policies.

The primary objective of this project is to **extract metadata from administrative decisions**, focusing specifically on energy permits issued under the WOO or comparable legislation, such as the *Instellingswet* for the ACM[2]. Metadata extraction is valuable in itself, enabling improved document classification and retrieval while also allowing deeper insights into patterns across different authorities and decision types. For example, identifying recurring parties in decisions or analyzing patterns in permit issuance over time can uncover trends that might otherwise remain hidden in unstructured data. While the primary goal of this research is to extract metadata from administrative decisions, a significant challenge lies in addressing the lack of standardization across unstructured administrative data. This inconsistency complicates the application of NLP techniques across diverse governmental authorities, particularly when using supervised approaches that require pre-training on the targeted metadata before deployment. Therefore, this research not only focuses on metadata extraction but also explores strategies to mitigate these standardization challenges, bridging the gap between the theoretical potential of NLP and the practical constraints posed by unstructured administrative data. Ultimately, this work aims to lay the groundwork for scalable and consistent information extraction from energy permits across diverse governmental authorities.

1.3 State of the Art

The extraction of information from legal texts has been a longstanding challenge at the intersection of computer science and legal studies. Over time, various methods have been developed to tackle this issue, each with distinct strengths and limitations. These methods have evolved from rule-based systems to supervised Machine Learning (ML) approaches and, more recently, to Large Language Model (LLM)-based frameworks.

1.3.1 Evolution of NLP in the Legal Domain

Early works focusing on NLP applications in the legal domain heavily relied on hand-crafted rules or features due to computational limitations at the time [17, 23, 28, 32]. Even in more recent works (e.g., [8, 31]) and despite significant improvements in computational power, the creation of rules remains a complex and labor-intensive task, serving as a major bottleneck for this approach [27]. Domain experts are still required to define rules and patterns to identify and extract relevant data from legal documents. These rule-based approaches excel in explainability,

[2]More details can be found on the Overheid.nl website

as the rules are explicitly defined and easily traceable. However, they lack adaptability across domains and struggle to generalize across diverse document formats and unseen data—especially given the constantly evolving standards of legal documents [27]. This limitation significantly hinders their scalability and effectiveness in large-scale legal text analysis.

The introduction of supervised ML—and later deep learning—brought significant improvements in adaptability and performance. The need for manually crafted rules diminished as models became increasingly capable of learning how to carry out tasks autonomously. These approaches rely on training models using labeled datasets, where key metadata (e.g., involved parties, decision dates, legal references) are annotated manually or semi-automatically (e.g., [14, 21]). While supervised ML approaches can offer better generalization across document types compared to rule-based systems—mainly due to their rule-agnostic nature—their effectiveness heavily depends on the availability of context-relevant and high-quality labeled data [6, 20].

The emergence of LLMs, such as OpenAI's GPT series, marked a significant leap forward in NLP. LLMs are built on the foundation of deep learning, specifically transformer architectures, introduced in the seminal paper *"Attention is All You Need"* by Vaswani et al. in 2017 [36]. These models leverage self-attention mechanisms to process and understand the context within the text, enabling them to handle tasks ranging from simple text classification to generating human-like language. Trained on massive corpora spanning diverse domains and languages, LLMs eliminate the need for manual feature engineering and can generalize effectively across unstructured datasets and a wide range of tasks—from annotation (e.g., [9]) to document summarization (e.g., [40]) and classification (e.g., [39]). This versatility makes them a transformative tool for legal data processing, capable of extracting, structuring, and analyzing information at an unprecedented scale and efficiency.

1.3.2 NLP for Information Extraction in Administrative Decisions

In fields outside of legal information extraction, supervised ML models, when trained on high-quality labeled datasets, have shown strong potential for extracting key metadata elements from unstructured texts. For example, the Python library *LexNLP* (available in English, Spanish, and German only) leverages supervised ML and other NLP techniques to extract over eighteen types of structured information like distances and dates or named entities like companies and geopolitical entities [14]. This shows that such supervised models can deliver precise and reliable results. However, to achieve good performance, they require accurate and sufficient labeled data for training these algorithms. Nonetheless, in the context of administrative decisions, these ideal conditions are hardly achieved by relying only on the available metadata that accompanies the published documents. Given the lack of mandatory standards in terms of metadata disclosure[3] It remains up to each governmental authority to decide how they will publish their documents, as well as the amount of metadata that will accompany them. This is why the metadata is currently not always present, and when it is, it shows inconsistencies even within other publications from the same authority. Addressing these issues to enable the training of a supervised model requires labor-intensive manual validation and correction. This process is particularly challenging in the context of legal NLP tasks, as it demands expert knowledge to accurately and legally label the dataset. As a result, the production costs for high-quality labeled datasets become prohibitively high, ultimately disincentivizing the use of supervised ML models to enhance transparency.

In contrast, LLMs present a promising alternative. With their *zero-shot* and *few-shot* learning capabilities, LLMs can potentially bypass the need for extensive labeled datasets. Pre-trained on vast and diverse text corpora, they are capable of inferring metadata patterns directly from administrative decisions without requiring domain-specific fine-tuning. This flexibility offers a pragmatic workaround for the challenges posed by metadata scarcity, inconsistency, and incompleteness. However, while LLMs exhibit significant potential, their effectiveness in extracting metadata from administrative decisions remains largely untested and unexplored.

[3] Although a standard for metadata disclosure in the context of the WOO exists, its adoption is not (yet) mandatory. More details can be found on the website overheid.nl

Models such as GPT demonstrate impressive *zero-shot* and *few-shot* capabilities, enabling them to perform complex information extraction tasks with minimal task-specific fine-tuning [1, 38]. The strengths of LLMs include their scalability across diverse legal text formats and their adaptability to varying linguistic structures, even in the absence of extensive labeled datasets. However, these approaches come with notable limitations:

- **Hallucinations:** LLMs occasionally generate plausible-sounding but factually incorrect outputs, which can compromise the reliability of extracted metadata.
- **Lack of Explainability:** compared to traditional ML models, LLMs operate as black boxes, making it difficult to trace and interpret their reasoning processes. In a legal context, where transparency and accountability are critical, the inability to justify or explain decisions poses significant challenges to their adoption and trustworthiness.
- **Domain Expertise:** it is unclear whether LLMs qualify as domain experts. Since they are trained on an unknown set of data, the extent to which legal data is included—and the quality or quantity of this data—remains uncertain. This limitation raises questions about their ability to reliably handle specialized legal tasks.

Despite this, they have already been employed in prioir legal research (though not directly related to administrative decisions), demonstrating impressive capabilities in understanding and analyzing text with high accuracy, allowing the processing of vast quantities of data with minimal human intervention and enabling accurate and detailed analysis at scale [9, 11]. This evolution in NLP represents a viable solution for handling large datasets in the legal domain, aligning with the growing demand for transparency and accessibility in governmental decision-making.

1.4 The Research Objectives and Questions

This research focuses on a subset of permits, serving as a foundation for evaluating and comparing different approaches to metadata extraction from administrative decisions. While grounded in this specific subset, the study also considers the generalizability of its findings to a broader range of administrative decisions. The primary objective is to explore and assess the performance of supervised ML algorithms in extracting metadata—using labeled data, where present—and compare these results with those achieved using LLM-based approaches. Specifically, the study investigates whether the adaptability and *zero-shot* capabilities of LLMs can compensate for the absence of high-quality labeled datasets, offering a scalable and effective solution for large-scale metadata extraction.

By systematically evaluating these two paradigms, this research aims to highlight their respective strengths and limitations while identifying their practical applicability under the framework of the Dutch Wet Open overhead. Ultimately, the findings seek to contribute to both practical applications—by offering insights into scalable metadata extraction workflows—and academic research—by advancing our understanding of the strengths and weaknesses of different NLP approaches in the legal domain. To address these objectives, the research investigates two central questions:

- *How can NLP techniques be applied to extract key metadata from Dutch administrative decisions, specifically focusing on energy permits issued by the Autoriteit Consument & Markt (ACM)?*
- *Is it possible, leveraging large language models (LLMs), to overcome the lack of labeled data required for supervised ML algorithms without incurring significant performance losses?*

Ultimately, this research seeks to bridge the gap between theoretical advancements in NLP and the practical challenges posed by unstructured administrative datasets. The findings aim to contribute to ongoing efforts to improve transparency and accessibility in administrative decision-making under the Dutch Open Government Legislation.

1.5 Methodology

To address the research questions outlined in Sec. 1.4, this study employs a structured workflow combining NLP techniques, specifically supervised ML models and LLMs, to extract and structure metadata from Dutch administrative decisions. The focus lies on energy permits issued by the *Autoriteit Consument & Markt* (ACM).

1.5.1 Dataset Overview, Preprocessing and Labeling

The dataset used in this research consists of energy permits sourced from the ACM website (`www.acm.nl`), where they are publicly available. These documents were collected through web scraping and converted into machine-readable text format. Further details regarding data collection and dataset preparation are provided in Chapter 2.

Before metadata extraction, preprocessing techniques were applied to ensure data quality and consistency (see Sec. 2.2). Rule-based noise removal methods were used to clean the text, addressing artifacts introduced during PDF-to-text conversion. Additionally, an attempt was made to correct spelling errors using a large language model (see Sec. 2.4). These preprocessing steps aimed to reduce noise and improve the overall quality of the textual data for downstream processing.

Incomplete metadata elements were manually labeled due to the low quality of the original metadata obtained during scraping. The details of this procedure are provided in Sec. 2.3.

1.5.2 Metadata Extraction Tasks

This research aims to extract four key metadata elements from administrative decisions, which are expected to be present in all such decisions (though sometimes in different forms depending on their legal effect) and are based on Article 3.3 from the WOO:

1. *Recipient(s)*: parties the administrative decision is addressed to.
2. *Decision Date*: date on which the administrative decision was made.
3. *Legal References*: the laws, articles, or regulations present in an administrative decision and possibly underpinning it.
4. *Legal Effect*: the type of outcome resulting from the administrative decision based on a categorization specific to permits.

The rationale for selecting these specific metadata elements and their definitions are discussed in Chapter 6.

1.5.3 Supervised Approaches

Supervised ML models were employed to extract metadata elements from the dataset. Python libraries such as *scikit-learn* and *spaCy* were used for model training and implementation. The training process relied on the (partially) labeled permits[4] included in the dataset, ensuring task-specific adaptation of the models. However, supervised ML approaches were not suitable for extracting the *Legal Basis* metadata due to the lack of labeled data for each decision, which is highly unfeasible to manually annotate. Additionally, the extreme variability of legal references within the permits makes it impractical to implement a weak labeling approach effectively.

To address this limitation, a specialized rule-based tool, *linkeXtractor*, developed by the Dutch Publication Office (KOOP), was used instead [35]. This tool represents a highly refined and domain-specific solution, the result of years of research and optimization. *linkeXtractor* serves as a benchmark, embodying a highly fine-tuned system built upon extensive domain expertise and iterative refinement of its rules and datasets. This comparison enriches the scope of the research, as it allows for an evaluation of whether an approach requiring significantly

[4]Permits are considered labeled if they carry correct/reliable metadata or have undergone a process of manual annotation.

fewer resources—such as an LLM-based method—can achieve comparable results in the task of Legal Basis extraction, mirroring the comparison between supervised ML and LLMs for the other metadata tasks.

1.5.4 LLM-Based Approaches

LLMs were utilized as an alternative approach to metadata extraction, leveraging the OpenAI API for both *zero-shot* and *few-shot* learning. Task-specific prompts were tailored for each metadata type, guiding the LLM in extracting and structuring the required information from the permit texts.

The implementation of LLM-based approaches was carried out using the OpenAI API, which provides access to state-of-the-art models, such as Generative Pretrained Transformers (GPT). These models are designed to process and generate text based on given prompts, enabling applications such as text summarization, question-answering, metadata extraction, and spelling correction. The API operates by receiving a prompt that specifies the task to be performed and returns output text generated based on its pre-trained knowledge. For tasks like spelling correction, the prompt can be customized to include specific instructions, examples, or constraints, allowing the model to holistically analyze the text while considering context, syntax, and semantics.

To fine-tune the behavior of the OpenAI API, several configurable parameters are available. The `temperature` parameter controls the randomness of the output, with lower values (e.g., 0.2) resulting in more deterministic responses, while higher values encourage greater creativity. The `max_tokens` parameter limits the maximum length of the generated response, ensuring outputs remain concise and within the desired range. `top-p` (nucleus sampling) adjusts the probability distribution of tokens considered during generation, balancing precision and diversity. Additionally, `stop_sequences` can be specified to define points where the model should stop generating output, providing greater control over responses. These parameters allow users to tailor the model's behavior to suit specific tasks and requirements.

However, reliance on LLMs introduces challenges, such as API usage costs and potential latency for processing large-scale datasets, particularly when using advanced models. Despite these challenges, LLMs offer a flexible and powerful tool for metadata extraction and other text-processing tasks, demonstrating their adaptability to a wide range of applications.

1.5.5 Comparative Analysis and Evaluation Framework

The results from the supervised ML models, the KOOP rule-based approach (for *Legal Basis* extraction), and the LLM-based methods were systematically compared. Traditional evaluation metrics (e.g., precision, recall, F1-score) were utilized, and task-specific metrics (e.g., entity fuzzy matching) were designed to suit the unique requirements of specific metadata extraction tasks. Additionally, qualitative analysis was conducted in cases where numerical evaluation alone was insufficient to fully capture performance differences.

Task-specific evaluation frameworks are discussed in detail in each Chapter covering a specific type of metadata. This methodology sets the stage for a structured comparison between supervised ML and LLM-based approaches, offering insights into their applicability for metadata extraction in administrative decisions and their potential contributions to transparency under the Dutch WOO.

1.6 Findings

Results

The evaluation showed that supervised approaches generally outperformed LLMs in structured tasks like date extraction and legal effect classification, achieving F1-scores of 97% and 90%, respectively. The LLM-based methods, while less precise (e.g., 87% and 73% F1-scores for the

same tasks), demonstrated significant adaptability across different datasets and contexts without requiring retraining. Recipient extraction proved more challenging, with supervised models achieving an F1-score of 85% (93% precision, 78% recall), while the LLM-based approach outperformed them with an F1-score of 88% (84% precision, 92% recall). Legal reference extraction, performed using the rule-based *LinkExtractor*, delivered granular and precise outputs but struggled more with adaptability. Conversely, LLMs exhibited broader coverage but lacked the fine-grained detail necessary for specific legal references.

Interpretation

The results indicate that both supervised models and LLMs have strengths and weaknesses that align with different objectives. Supervised methods excel in accuracy and are resource-efficient once trained, but their effectiveness is constrained by the availability and quality of labeled data. In contrast, LLMs offer unmatched scalability and cross-context adaptability, particularly valuable in dynamic legal environments or large-scale transparency initiatives. However, their reliance on computational resources, susceptibility to hallucinations, and lack of interpretability can still present significant challenges. In addressing the research question, this study demonstrates that while supervised techniques remain (almost always) the gold standard for tasks requiring high accuracy, LLMs present a viable alternative in scenarios where labeled data is limited and adaptability is critical. The findings underscore the potential of LLMs to complement traditional supervised methods, advancing the broader objective of promoting transparency in administrative decision-making. By highlighting these trade-offs, this research contributes to both the scientific understanding of information extraction techniques and practical advancements in fostering a more digital and open government. For supervised approaches, fFor supervised approaches, future work should focus on fine-tuning models using larger datasets spanning multiple authorities to improve generalization and assess whether the high performance observed in this study persists when applied to broader, more diverse datasets, which may challenge the capabilities of supervised methods. Conversely, to enhance the application of LLMs, future work should refine prompting strategies, explore LLM fine-tuning in greater depth, and investigate the integration of rule-based or ML systems as a final verification step to address the reliability issues inherent in LLMs.

Chapter 2

The Data

The data used in this research originates from the Publications section of the website of the Autoriteit Consument & Markt (ACM)[1]. In compliance with the *Instellingswet ACM*, the ACM regularly publishes its decisions proactively, making them publicly accessible. Each decision page on the website includes metadata such as the publication date, type of decision, decision date, case number, and involved parties. Additionally, one or more PDF documents are attached to each decision, which may be either scans or original typeset files containing the full text of the decision. For this research, a filter was applied to focus specifically on energy permits (*energievergunningen*, in Dutch). The filtered publications were scraped and consolidated into a CSV file, retrieving all available metadata along with the textual content from the PDFs. Within the database, each decision was split into multiple rows—one row per page of the associated PDF—while retaining shared metadata such as the case number. This resulted in a CSV table containing 12 columns and 1964 rows, each representing a portion of a decision. By utilizing the `id` and `file_number` columns, it was possible to reconstruct the original decisions. This reconstruction produced a dataset of 309 unique decisions spanning 381 rows, as some decisions included multiple attached PDFs.

2.1 Exploratory Data Analysis

As shown in Table 2.1, NaN values are present in the `decision_date`, `case`, `text_pypdf2`, and `parties` columns. Notably, the `parties` column has a high proportion of missing values, with one-third of the rows lacking a party entry, despite decisions typically being addressed to specific entities. Closer inspection revealed further issues within the `parties` column. Some entries listed parties that could not be traced back to any part of the decision's headline or description (i.e., the mentioned party was absent from the text). Additionally, certain decisions involved multiple parties, but this representation lacked a consistent format. For instance, some entries used commas or semicolons to separate parties, while others simply listed them without any delimiter beyond the "B.V." suffix for one party. Attempts to process and standardize the `parties` column programmatically—such as splitting entries by "B.V.", commas, or other delimiters—proved unsuccessful. As a result, manually labeling the data for this column proved to be a better approach to address both the absence or inaccuracy of party entries and the inconsistent formatting when multiple parties were involved (see Sec. 2.3). Moreover, a significant number of decisions listed *Leveringsvergunningen Kleinverbruik Ex Art. 45 Gaswet* as the party, even though this phrase does not represent an actual party. This issue arose because a group of 26 decisions were aggregated under a single decision page on the ACM website without individual metadata, such as the addressed party. Due to this partial lack of metadata, these decisions were excluded from certain parts of the analysis (reducing the dataset to 283 decisions).

Figure 2.1 provides an overview of key characteristics of the dataset's textual and temporal features. It consists of three violin plots, which summarize the statistics for the length

[1]See `acm.nl/nl/publicaties`, filtering for *vergunningen* using the *trefword* drop-down list.

Table 2.1: Overview of the dataset columns, including their descriptions and the percentage of missing (NaN) values

Column Name	Description	NaN Values (%)
`id`	Identifier for each decision in the dataset	0.00
`headline`	Short title summarizing the decision	0.00
`description`	Short detailed description of the decision	0.00
`publication_date`	Date when the decision was published	0.00
`decision_date`	Date when the decision was made	14.97
`decision_type`	Type or category of the decision (here constant: *besluit*)	0.00
`case`	Case number or identifier associated with the decision	9.27
`parties`	Entities or individuals involved in the case	33.15
`file_link`	URL or link to the original decision page on the ACM website	0.00
`text_pypdf2`	Extracted text content from the decision PDF (here often referred to as "decision body")	2.95
`file_number`	File number associated with the PDF	0.00
`page_number`	Page number of the text extracted from the PDF (initially, a row of the dataset corresponds to a single page of a decision document, i.e. decisions are often split across multiple rows)	0.00

(in words) of the `headline`, `description`, and `text_pypdf2` columns. These plots visualize the distribution of word counts, highlighting the median and key percentiles (25th and 75th) for each column, along with the range of word counts (minimum and maximum). Notably, the `text_pypdf2` column contains significantly longer entries compared to the `headline` and `description` columns, as expected since it includes the full text of the decisions. Detailed summary statistics for these columns are available in Table A.2.1.1, provided in the appendix.

The `publication_date` spans from June 2, 2004, to August 27, 2024, while the `decision_date` covers a slightly narrower range from September 29, 2005, to August 13, 2024[2]. These ranges highlight the extensive temporal span of the dataset, providing valuable historical and recent decision records. A tabular summary of temporal information is available in Table A.2.1.2, also included in the appendix.

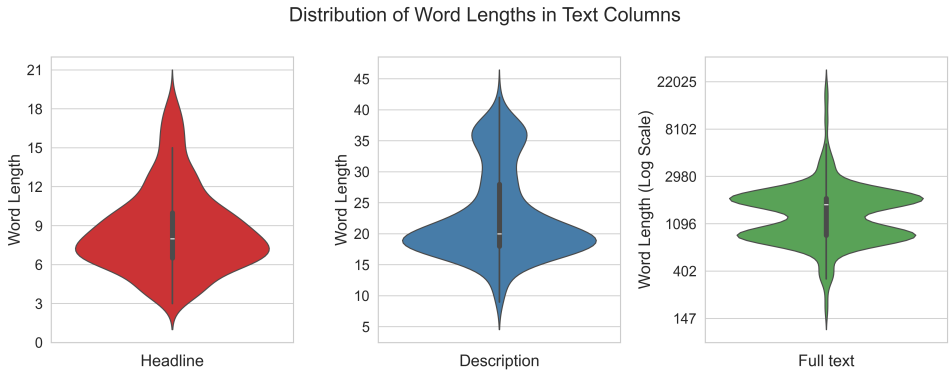


Figure 2.1: Distribution of textual columns. The y-axis for the full text is log-scaled to account for strong outliers that would otherwise skew the plot. For clarity, the y-axis ticks are presented in their original (unlogged) scale.

[2]The earlier start date for the `publication_date` column is due to a decision which does not have a decision date but only a publication date.

2.2 Data Cleaning

Upon inspection of the data, it was found that the text in a subset of 22 rows (5.77%) was irreversibly corrupted, necessitating their deletion. This issue arose due to the PDF-to-text conversion process (powered by the *pypdf2* library, [4]), which is used to extract text from the PDFs. This attempts to recognize characters in scanned images and convert them into machine-encoded text, but errors can occur due to poor scan quality or complex formatting in the original document. Doing so reduced the dataset to 359 rows and 288 unique decisions. Subsequently, rule-based methods were applied to clean the body of each permit and remove elements that did not contribute meaning but instead introduced noise into the data. These methods included character or word replacements and the use of regular expressions for more complex patterns. The following modifications were applied (see Table 2.2):

Table 2.2: Examples of text cleaning steps

Cleaning Step	**Original Text**	**Cleaned Text**
Replacing newline characters with spaces to unify text flow	`This is a line.\nThis is another line.`	`This is a line. This is another line.`
Removing repeated dots/ellipses	`This decision is final...`	`This decision is final.`
Removing dots separated by a space	`This . is problematic.`	`This is [not] problematic.`
Removing tab characters with spaces	`This \t is a tab with trailing spaces.`	`This is a tab with [no] trailing spaces.`
Retaining only ASCII[3] and accented characters	`financiële © besluit`	`financiële besluit`
Eliminating unnecessary spaces around punctuation	`This , is an example .`	`This, is an example.`
Consolidating multiple spaces into one	`This has too many spaces.`	`This has[n't] too many spaces.`

The choice of these modifications, which effectively removed nearly all unwanted noise in the text resulting from the PDF-to-text conversion, was informed by a careful manual inspection of the corpus. This review also identified unnecessary elements at the beginning of each decision, such as standard preambles, headers, and failed attempts to convert logos or images into text. To address this issue, only the text following the first mention of the case number was retained, as this point consistently marked the beginning of the actual decision content. The relevant starting points in the text were identified based on specific patterns, including text beginning with `ACM/` or `AGM/`, the phrase `Ons kenmerk`, and the phrase `BESLUIT Nummer:`. As a final step, all text was converted to lowercase to "harmonize" strings of text by generating a single token out of lowercase and uppercase variants of the same word. [5].

The publication and decision dates were converted into datetime objects to facilitate chronological analysis. The term *Zaaknummer* (Dutch for case number) was removed from entries in the `case` column to isolate the case number itself. Additionally, the `headline` and `description` columns were combined into a single feature representing the header of each decision.

2.3 Labeling

Although each decision included some metadata, it was often necessary to refine or extend it. Features like the list of parties involved in each decision were frequently missing, unreliable, or stored inconsistently. As a result, all 360 entries in the `parties` column—also referred to as *ontvanger(s)*—were manually re-labeled to ensure accuracy and consistency. Additionally, to enable the classification of decisions into categories, a legal effect category (*rechstgevolg*) was assigned to each decision based on a predefined set, if applicable. The creation of a gold

standard was essential for the research, as it provided a reliable benchmark against which to evaluate methods and results. This ensured that the analysis was grounded in well-defined and consistent data, supporting the validity of the findings. Details of the labeling protocol are provided in the appendix A.2.2 for further reference.

2.4 Spelling Correction

Early exploration of the corpus revealed numerous misspelled words, not due to errors by the decision writers but as a result of text extraction from PDFs using the *PyPDF2* library. This observation prompted an investigation into spelling correction prior to implementing any IE/NLP pipeline.

Proper preprocessing can, in some cases, enable simpler ML models to outperform transformers [30]. Preprocessing generally aims to remove sources of variation or noise[4] in a text hindering a model's ability to extract meaningful patterns from it. Among these techniques, spelling correction plays a significant role. Unlike humans, who possess an innate "auto-complete" functionality in their cognitive processes, computer systems often struggle to interpret misspelled words in context [13]. This limitation is particularly evident in traditional machine learning approaches, which treat words atomically and lose the structural relationships that exist between them.

While researchers do not universally agree on the choice and number of preprocessing techniques to use, it has been demonstrated in [30] that preprocessing decisions can result in differences of over 25% in classification accuracy when using the same model and dataset. Correcting misspellings helps standardize texts and reduce noise [16], both of which are beneficial for ML by improving the consistency and interpretability of input data.

Moreover, experiments have shown that spelling correction is often a likely cause of improved model accuracy, partly due to its ability to reduce data sparsity[5] [22, 37]. This demonstrates that spelling correction can be a crucial component of effective preprocessing pipelines.

2.4.1 Initial Approaches and Challenges

Early attempts at spelling correction relied on a dictionary-based approach using the Python library `pyspellchecker`, which matches words against a predefined frequency-based word list within a Levenshtein edit distance of two. While promising in concept, this method proved inefficient for the dataset, as it failed to complete even for small inputs. The inefficiency stemmed from several factors: out-of-dictionary words like proper nouns and domain-specific terms led to excessive candidate generation, tokenization errors caused fragmented words to be incorrectly processed, and the fixed edit distance significantly expanded the search space for longer words. These limitations highlighted the need for a more robust and scalable solution tailored to the dataset's unique characteristics.

2.4.2 LLM-Based Spelling Correction

Another approach explored for spelling correction involved leveraging an LLM. Unlike dictionary-based methods, this is inherently unsupervised and is performed by sending a prompt to an LLM, instructing it to correct the spelling of the provided text. One of the primary advantages of this approach is its ability to overcome the limitations of vocabulary and language specificity inherent in traditional dictionary-based methods. LLMs are pre-trained on extensive corpora across multiple languages and domains, making them capable of handling out-of-dictionary words, domain-specific terms, and linguistic nuances. Additionally, the computational complexity of traditional methods—such as generating permutations of words and comparing them

[4] 'Noise' is intended as defined by Siino et al. as "*anything related to any useless information for any text-based task to be performed after preprocessing a dataset*" [30].

[5] By correcting misspelled words, the dimensionality of the data matrix is minimized, as each unique misspelling would otherwise be treated as a new token despite having the same meaning. This reduction in sparsity allows models to focus on actual existing words, facilitating better generalization and enhancing overall NLP model performance.

against a dictionary—is avoided, as the LLM processes text holistically. Furthermore, context awareness is a significant benefit, as LLMs consider the surrounding text to make corrections, reducing the likelihood of incorrect substitutions and being able to recognize wrongly divided words. This makes LLMs particularly effective for complex or ambiguous corrections where context is critical. Two different implementations of this spelling correction approach were tried, differing only in the prompt used[6]. Since the OpenAI API imposes token limits, the implementations included a mechanism to split texts exceeding these limits into smaller chunks. A helper function divided the input text into chunks of up to 16,383 tokens, and the spelling correction function was applied recursively to each chunk. The corrected chunks were then reassembled to produce the final corrected text. Robust error-handling mechanisms were implemented to address potential issues too. If a token limit error occurred, the text was split into smaller chunks for reprocessing. In cases of API rate limits, a delay was introduced to prevent repeated failures. Persistent errors resulted in retaining the original text and logging the issue for further review.

The **first implementation** utilized a Dutch prompt[7] for spelling correction, explicitly instructing the model to correct only spelling errors in the provided text while maintaining its original structure, content, and formatting. Additionally, the model's output was required to strictly conform to the JSON schema specified in the API request.

Levenshtein distance was chosen to measure the changes made during spelling correction, as it quantifies the number of single-character edits (insertions, deletions, or substitutions) required to transform one string into another. This metric provides a clear and interpretable indication of the extent of corrections, enabling objective comparisons of outputs across different implementations.

After inspecting the histogram of Levenshtein distances (see Fig. 2.2), it was observed that there were more changes than what was expected. The median distance was also quite high, with a value of 94 per decision body. This is quite significant for a spelling correction task, where typical corrections would reasonably involve only a few edits (on average), and maybe more but only heavily misspelled texts. Such a large distance suggests that the model went far beyond simple spelling adjustments, likely introducing significant alterations to the text, such as removing, rephrasing, or hallucinating content.

This prompted a closer examination of some of the most altered decisions following the spelling correction. While the results were occasionally promising—such as correcting words like *"be paald"* to *"bepaald"* or accurately identifying and correcting entities like *wellsius energie b.v.* to *Wellsius Energie B.V.*—it became evident that the LLM was also removing parts of the text entirely without replacing them. This behavior highlighted the model's tendency for hallucinations, making it highly unreliable for consistent text correction. An example of this can be seen in Fig. 2.4.

To address the hallucination issue observed with the initial prompt, a **second implementation** was developed. This version featured a more explicit and detailed prompt designed to minimize unintended alterations by clearly specifying the task. The updated prompt included a concrete example of acceptable spelling corrections (*one-shot* learning) and explicitly instructed the model to leave style, grammar, and phrasing unchanged. Beyond the refined prompt, the overall methodology remained identical to the first implementation.

[6]API call parameters:

- **Model:** `gpt-4o-mini`: an efficient and cost-effective model with strong textual reasoning, a 128K context window, and optimized multilingual support. Its low latency and affordability make it perfect for real-time applications, ensuring high-quality outputs at scale.
- **Temperature:** 0 (deterministic behavior)
- **Max Tokens:** 16,383 (allowed maximum, to accommodate long texts)
- **Top-p:** 1 (consider all probable tokens)
- **Frequency Penalty:** 0 (no penalty for frequent tokens)
- **Presence Penalty:** 0 (no penalty for introducing new tokens)
- **Response Format:** JSON schema specifying the structure and strict validation of the output.

[7]The prompts are presented on the next page in English for consistency with the rest of the text. The original Dutch prompts can be found in the appendix in Sec. A.2.3.

This time, the median Levenshtein distance dropped to 45, less than half of what was observed with the previous prompt. Moreover, inspecting the updated histogram of Levenshtein distances (see Fig. 2.3) revealed that, although numerous changes still occurred during the spelling correction process, they were now more concentrated on the left side of the distribution. Despite this improvement, the median distance remains relatively high, indicating that further inspection of the data is necessary to fully understand the nature of these changes and assess whether the issue of hallucination persists.

Eventually, upon closer examination of the corrected text, it became apparent that the LLM still occasionally exceeded the task's intended scope, making modifications beyond simple spelling corrections (see, for instance, Fig. 2.5). In some cases, entire chunks of text were removed despite explicit instructions to preserve the original structure. While the median distance suggests these instances occurred less frequently compared to the first implementation, their continued presence undermines the reliability of the output from this process.

Prompt 1 for Spelling Correction

Correct exclusively the spelling mistakes in this Dutch text: `{decision body}`. Maintain the original structure, content, and formatting. Return only the corrected text in the `text_pypdf2_gpt` field of the JSON output. There should be no extra text before or after the JSON output. Only the provided JSON schema is allowed.

Prompt 2 for Spelling Correction

Correct only spelling mistakes in this Dutch text: `{text}`. Strictly preserve the original structure, content, and formatting. The API may adjust a maximum of one word at a time or correct compound word parts (e.g., "au to" to "auto"). Return only the corrected text in the `text_pypdf2_gpt` field of the JSON output. There should be no extra text before or after the JSON output, and only the provided JSON schema is allowed. Do not correct style, grammar, or phrasing—only spelling.

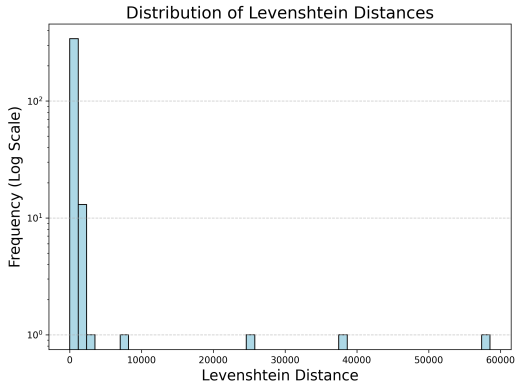


Figure 2.2: Distribution of Levenshtein distances after spelling correction (Prompt 1). The four outliers can be disregarded, as they result from the OpenAI API's output word limit and could be easily resolved.

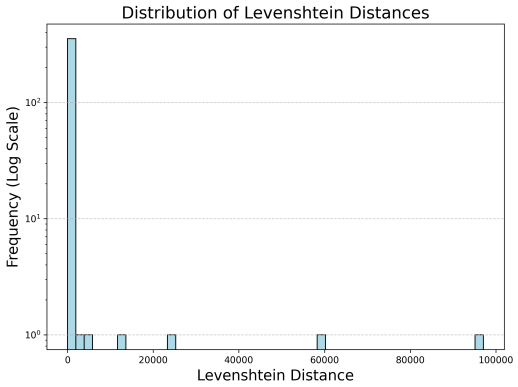


Figure 2.3: Distribution of Levenshtein distances after spelling correction (Prompt 2). The same outlier remark as in Fig. 2.2 holds.

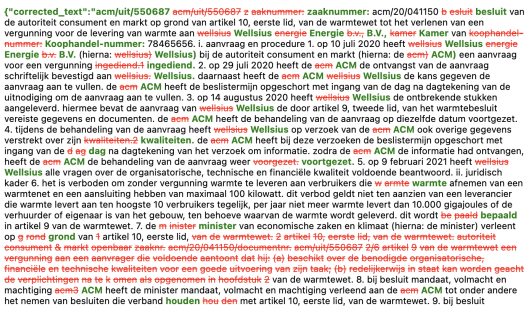


Figure 2.4: Hallucination example (truncated) after spelling correction with prompt 1

Figure 2.5: Hallucination example (truncated) after spelling correction with prompt 2

2.4.3 Conclusion

Dictionary-based spelling correction methods are a fundamental tool in text preprocessing, offering a straightforward way to address misspellings. However, their effectiveness diminishes significantly when applied to large, unstructured datasets with numerous out-of-dictionary words or frequent PDF conversion errors, leading to increased inefficiency and reduced accuracy. While optimizations such as refining tokenization or restricting edit distance for longer words could improve performance, these adjustments can compromise the simplicity of such approaches.

Conversely, LLM-based methods offer advantages such as context awareness, ease of implementation, and lower local computational requirements. However, hallucinations remain an unpredictable challenge that cannot be fully mitigated through prompt engineering. Notably, both LLM-based implementations sometimes agreed on corrections and text removals (see Figures 2.4 and 2.5), reinforcing the notion that prompt refinement alone is insufficient to address this issue.

Despite these challenges, research on LLM-based spelling correction remains promising, with studies highlighting their potential both as standalone tools and as enhancements to traditional models [2, 3]. Ultimately, while spelling correction could improve downstream tasks, it is not essential for this research. Priority was given to information extraction, leaving the evaluation of spelling correction's impact on algorithm performance as a future consideration.

Chapter 3

Recipient Extraction

Legal decisions are addressed to a specific recipient, often a company, organization, or another identifiable entity. Having the recipient explicitly labeled as metadata is invaluable for downstream tasks, such as efficiently searching for permits issued to a particular entity. This capability can serve various purposes, from enabling companies to identify decisions relevant to their operations to allowing stakeholders to examine administrative actions taken towards similar entities.

The permit dataset at hand predominantly consists of legal decisions addressed to specific recipients. However, a small subset of entries includes public announcements, which lack a defined recipient. This distinction underscores the importance of accurately extracting recipient information from decisions where it is present while simultaneously identifying and handling cases where no clear recipient exists.

Extracting the recipient (*ontvanger*, in Dutch) from legal texts is a quite challenging task. Company names, for instance, are often inconsistent in their representation. They might be shortened to acronyms, partially omitted, or written with slight variations across documents. For example, a full company name might sometimes be abbreviated or truncated in a way that preserves its recognizability but complicates automated extraction[1]. Furthermore, names can range from a single word followed by standard legal suffixes like B.V.[2] or N.V.[3], to multi-word names with ambiguous boundaries that make it difficult to identify where the entity's name begins and ends. In addition to this, then, the dynamic nature of corporate entities, including the continuous emergence of new companies, introduces further variability that cannot be simply memorized. Given this complexity, rule-based approaches—though straightforward in theory—are often insufficient for recipient extraction. They require extensive hand-crafted rules, require frequent updates, and struggle to generalize across unseen variations in naming conventions. This chapter explores two alternative methodologies for recipient extraction. First, a traditional NER approach (often used in the industry) is implemented and evaluated. Second, the capabilities of LLMs are investigated to assess their effectiveness in addressing the challenges posed by recipient extraction tasks.

3.1 EDA of Labeled Parties

Due to the initial lack of quality ground truth labels about the parties involved in each decision, exploratory data analysis was conducted prior to attempting party extraction. This analysis revealed that the dataset contains a total of 223 distinct parties, with 28 documents not addressed to any party (i.e., not permits). The average amount of parties per decision was slightly above one (1.17), with a standard deviation of 0.58. Decisions can have more than one recipient for various reasons. For example, in a name change decision (*naamswijziging*), both the old

[1] An *example* (link) from the dataset: "[...] Op 5 februari 2013 heeft *Kas Energie Nederland B.V.* (hierna: *KEN*) bij de *Nederlandse Mededingingsautoriteit* (hierna: *NMa*) [...]".

[2] *Besloten Vennootschap*, a private limited liability company in Dutch law.

[3] *Naamloze Vennootschap*, a public limited liability company.

and new company names are addressed, or in a license transfer decision (*overdracht*), both the former and new license holders are mentioned. A more detailed and insightful description of the distribution by number of parties addressed is illustrated in Figure 3.1.

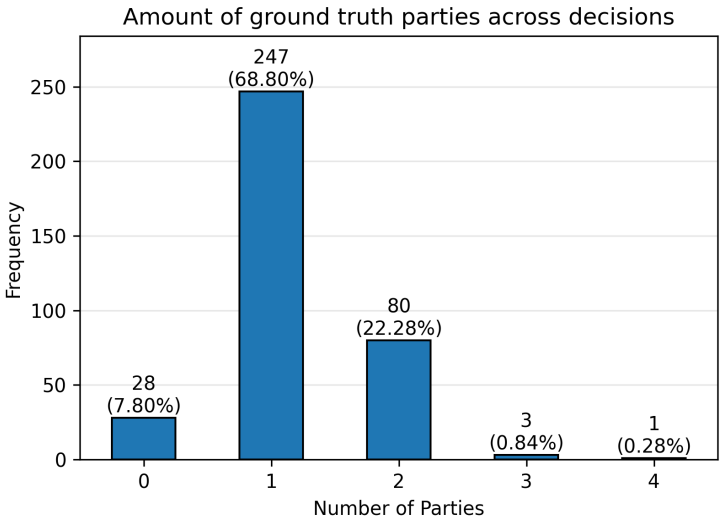


Figure 3.1: Amount of parties per decision

3.2 Supervised NER for Recipient Extraction

Named Entity Recognition is a task in NLP that aims to identify and classify named entities, i.e., real-world objects such as individuals, organizations, countries, or products—within a body of text. SpaCy is a widely used NLP library for this task, offering industrial-strength NLP through pre-trained models and customizable pipeline components. Its `EntityRecognizer` predicts entity types using statistical models trained on large corpora. However, as these models are statistical in nature, their performance heavily depends on the quality and domain relevance of their training data, often requiring fine-tuning for specialized tasks, as could be extracting recipients from legal texts.

The Dutch spaCy models were trained on a combination of multiple corpora to ensure broad linguistic coverage. These include *UD Dutch LassySmall v2.8*[4], containing Wikipedia sentences annotated with Universal Dependency labels, and *UD Dutch Alpino v2.8*[5], which consists of samples from various treebanks, including material from the Eindhoven corpus and the Dutch reference grammar ANS. Additionally, the *Dutch NER Annotations for UD LassySmall*[6] and *Explosion fastText Vectors* (cbow, OSCAR Common Crawl + Wikipedia)[7] were used to enhance semantic representation. Despite this variety, the datasets' origins are not entirely transparent, and none focuses specifically on the legal domain. The model names (`nl_core_news_sm`, `nl_core_news_md`, `nl_core_news_lg`) suggest that the training primarily relied on news data, likely limiting their out-of-the-box suitability for extracting recipients from legal texts.

To perform NER using this library, the `EntityRecognizer` must be used. This is a transition-based component designed to identify non-overlapping labeled spans of tokens, and it incrementally predicts a sequence of actions to determine entity boundaries and labels [12]. While effective for many NER tasks, this approach could have some limitations, namely because of:

- **Boundary sensitivity:** the recognizer performs best when entity boundaries are consistently annotated in the training data. Inconsistent annotations, such as labeling an

[4]By Bouma, Gosse; van Noord, Gertjan
[5]By Zeman, Daniel; Žabokrtský, Zdeněk; Bouma, Gosse; van Noord, Gertjan
[6]By NLP Town
[7]By Explosion

entity as *Conceptsnsolutions B.V.* versus *Conceptsnsolutions*, can reduce generalization and model reliability.

- **Token proximity assumption:** the model assumes that the most decisive cues for entity identification are located near the initial tokens of an entity. For example, in *Gas Supply Company B.V.*, if the word "Company" carries critical identifying information but appears in the middle, the model might misclassify or overlook the entity.

Despite these limitations, SpaCy's `EntityRecognizer` remains highly efficient, scalable, and adaptable. Its support for fine-tuning on domain-specific datasets allows it to address some of these challenges effectively, and, with careful annotation practices and tailored adjustments, SpaCy can still deliver reliable results in specialized tasks, such as recipient extraction from legal texts.

3.2.1 Baseline NER Implementation

As shown in Table 3.1, the out-of-the-box `nl_core_news_lg` spaCy model for NER performed poorly when applied to the concatenation of the headline and description columns. This result was largely anticipated given the model's training data, which (most likely) predominantly focuses on general news text where these entities are rare or even absent. In the sample from Table 3.1, the model consistently failed to identify entities of interest fully or accurately. In eight out of ten cases, the entity of interest is completely missed, and in the remaining two, it is only partially extracted. For example:

- The entity *HVC Energie B.V.* was detected inconsistently, with only fragments like *HVC* being recognized while omitting the rest of the entity name.
- Similarly, the entity *Holthausen Clean Energy B.V.* was reduced to partial detections such as *Clean Energy*, missing the remaining identifying components.

These examples highlight the model's limitations and the need for fine-tuning to improve NER performance for this task. Consequently, it was decided not to thoroughly evaluate this baseline implementation, as doing so would require a tailored definition of true positives. For example, an entity consisting of n parts might be considered correctly predicted if at least $n-m$ parts are included in the predicted entities list. Here, m represents the number of missing parts, and m must satisfy the condition $m < (n - x)$, where x is a similarity threshold designed to balance true positives and false positives. This threshold determines the maximum allowable mismatch while still counting the entity as a true positive. Carrying out such an evaluation in this context is impractical, as it requires careful reflection to establish appropriate thresholds and evaluation criteria. Even with these efforts, the results would likely remain insufficient, as observed during the qualitative comparison of the predicted entities against the ground truth. Instead, the focus was shifted toward developing a custom NER implementation, which is likely to be more applicable to real-world scenarios.

Table 3.1: Comparison of ground truth parties and detected ORG entities in `headline` & `description` by baseline NER

Parties (Ground Truth)	**Pred. ORG Entities**
TotalEnergies Gas & Power Nederland B.V.	ACM
Gemeente Hengelo	ACM
HVC Energie B.V.	HVC, ACM, HVC
Energie der Nederlanden	ACM
e-Energy Europe B.V.	ACM
TotalEnergies Gas & Power Nederland B.V.	ACM
Greenfoot Energy B.V., Energyhouse B.V.	(No Entities Detected)
Holthausen Clean Energy B.V.	Clean Energy, ACM
Duurzame Energie Veenendaal-oost (DEVO) B.V.	ACM
Greenfoot Energy B.V., Energyhouse B.V.	(No Entities Detected)

3.2.2 Custom NER Implementation

SpaCy offers the flexibility to retrain its NER models to adapt them to domain-specific datasets. This retraining process relies on high-quality labeled data, where the model learns to recognize entities based on annotated examples. Specifically, each labeled example must include:

- **Text:** the raw text in which entities are to be identified.
- **Entity Annotations:** start and end character offsets of each entity within the text, along with their corresponding entity type labels (e.g., ORG for organizations (current target), PERSON for people, etc.).

In this research, recipients were manually annotated (see `party_labeled` column). These labels are represented as a list containing one or more party names associated with the text. This structured labeling provides a foundation for retraining the SpaCy NER model to specialize in extracting recipients from legal decisions accurately.

Train-Test Split

Before training the custom NER model, the dataset was carefully divided into training and testing partitions. Given that some decisions comprised multiple attachments—each corresponding to different sections of the same permit but always linked to the same party—they shared identical IDs in the database. This overlap risked introducing data leakage if rows from the same decision appeared in both the training and testing sets. To address this, the split was performed at the level of unique decision IDs rather than individual rows. First, the unique IDs were randomly divided using an 80:20 ratio. Subsequently, all rows associated with each ID were grouped together to form the final training and testing sets[8]. While the 80:20 ratio was largely maintained, minor deviations occurred due to some IDs being associated with multiple rows. In total, the dataset contained 286 unique IDs, with 264 decisions allocated to the training set and 69 to the testing set[9].

[8]For example, if the decision ID `D123` was assigned to the training set, all associated data for decision ID `D123` would be included in the training set. This ensures that parts of the same decision (e.g., a document on gas and another on electricity for a company operating in both sectors) are not split between the training and test sets. Splitting such data would be problematic, as the recipient would remain the same, and the prediction task could become easier due to the model already encountering very similar documents. This method ensures a consistent split and prevents overlap between the training and test sets.

[9]This is expected, as some decision IDs arw associated with multiple documents. Adding 264 and 69 gives a total of 333, which corresponds to the initial number of cleaned entries in the dataset (359) minus the 26 rows excluded due to the issue with *Leveringsvergunningen Kleinverbruik Ex Art. 45 Gaswet.*

Data Preparation for Custom NER Training

To convert the ground truth labels into SpaCy-readable labels (with word start and end positions), the approach involved locating each label within the combined headline and description through reverse lookups. However, it was observed that the text from the labels did not always perfectly align with how the parties were mentioned in the dataset. This misalignment often stemmed from the ground truth labels being more complete. For example, a ground truth label might specify *TotalEnergies Gas & Power Nederland B.V.*, while the dataset text contains only *TotalEnergies*. Similarly, *Flexenergie B.V.* might appear as just *Flexenergie*. These variations introduced additional complexity in accurately mapping the labels to their corresponding text spans.

To address inconsistencies in how entities are represented, a preprocessing pipeline was developed to normalize, match, and annotate these entities within the text. First, each party label was normalized by removing or standardizing trailing suffixes such as "B.V." (in various formats, including "B. V.", "BV", or "b.v."), ensuring consistency across the dataset. This normalization was followed by constructing regex patterns capable of matching both the base entity name and its variants with suffixes. These patterns were applied to the corresponding text fields, identifying all occurrences of the entities and recording their start and end character offsets. An illustration of this can be found in Fig. 3.2.

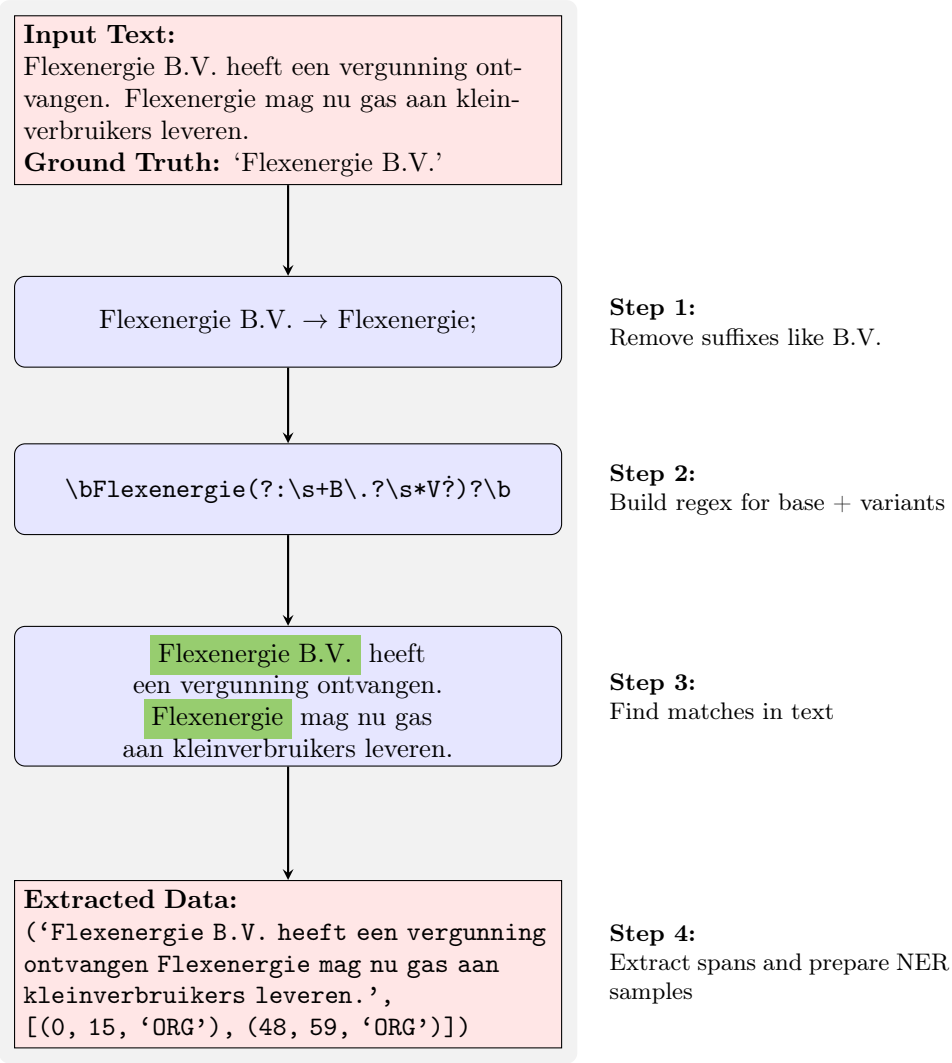


Figure 3.2: Flow chart of label preparation for NER

The resulting entities were then annotated with the label "ORG" to align with SpaCy's predefined entity schema. Each training example was structured as a tuple containing the text (`headline_and_description`), the list of annotated entities (with start and end offsets and entity labels), and the original party labels (`party_labeled`) for reference. Fig. 3.3 shows how, with this approach, it is possible to catch entities that slightly differ from the ground truth label.

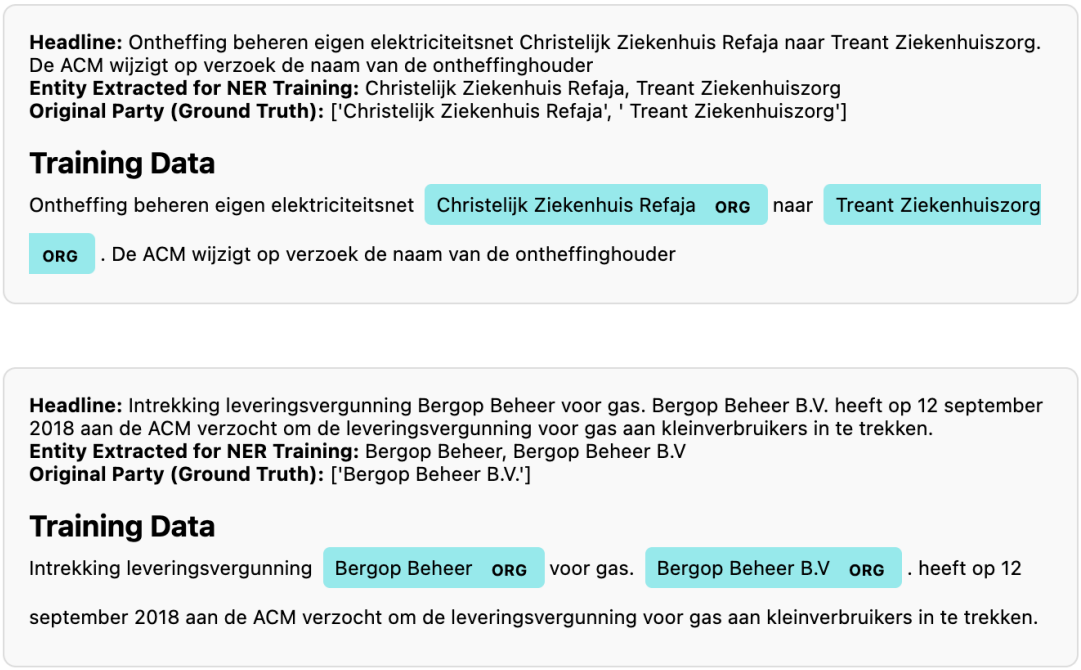


Figure 3.3: Example of training entity (flexible) matching

Custom NER Training Pipeline

SpaCy relies on token boundaries to identify entities accurately. If entity character offsets (*start, end*) do not align with the tokenizer's token indices, SpaCy raises warnings, and the training process may fail. Thus, entities first undergo an alignment process:

1. Each text sample is tokenized using SpaCy's tokenizer.
2. For every entity (*start, end, label*) in the dataset, the algorithm attempts to find the token boundaries that match the entity span.
3. If the alignment fails, a warning is logged for inspection.

SpaCy also does not support overlapping entity spans. Hence, overlaps are resolved by:

1. Sorting entities by their start index.
2. Retaining the longest entity span in overlapping cases.
3. Discarding shorter conflicting spans to ensure training consistency.
4. If overlaps are detected, they are logged for further inspection.

At this point, the NER model is trained on aligned and overlap-free data using the `nl_core_news_lg` base model. The training process is configured with a dropout rate of 0.2 to prevent overfitting and runs for a total of 30 epochs. These parameters aim to balance model robustness and training efficiency, enabling the model to generalize effectively to unseen data.

The convergence of training is monitored using a loss curve plotted across epochs (see Fig. 3.4), which shows a steady decrease, but never reaches extremely low values, stabilizing around a minimum of approximately 50. This suggests that while the model does learn from the data, its performance remains suboptimal and may require further fine-tuning or adjustments to the training data and hyperparameters.

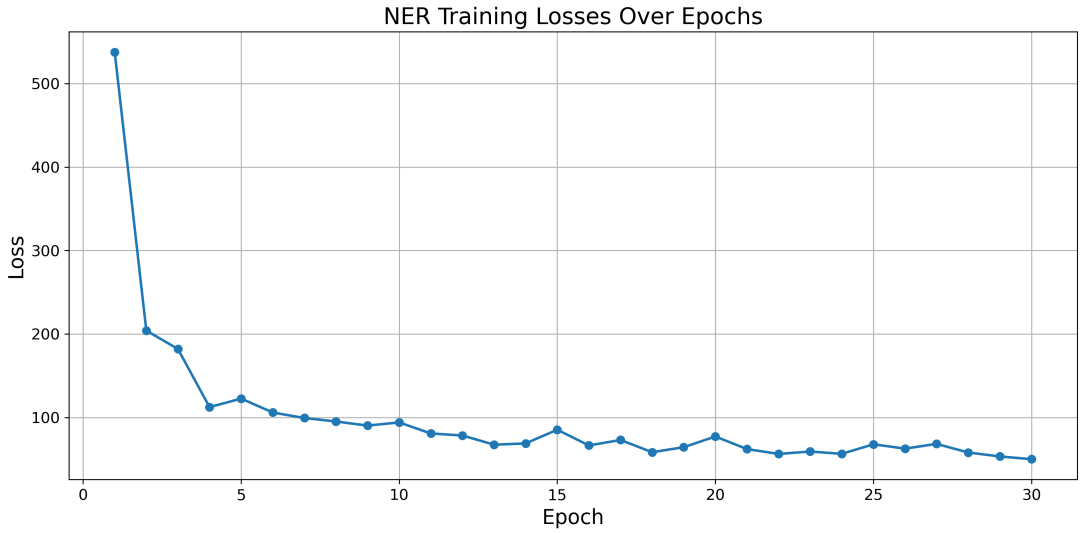


Figure 3.4: Training loss across 30 epochs

Evaluation

Elements from the test set are taken, and the trained custom NER model is applied to the text to extract the entities. As with the headlines and descriptions from the training data (see Fig. 3.3), it is possible for a party to appear multiple times within the same text. Consequently, there may be more predicted entities than ground truth labels, requiring an appropriate and tailored evaluation approach. A prediction is considered a **True Positive (TP)** if it correctly matches an entity from the ground truth party list at least once (it can be present more often, but matching it once is enough to find the addressee of a decision). Conversely:

- A **False Negative (FN)** occurs when an entity from the ground truth list is missed entirely by the model.
- A **False Positive (FP)** occurs when the model predicts an entity that is not present in the ground truth list.
- **True Negatives (TN)** are not applicable in this context, as they would imply non-entity spaces, which are irrelevant in Named Entity Recognition (NER) evaluation.

However, only using this evaluation framework overlooks two key complexities:

1. **Entity Completeness Mismatch:** labeled entities might include suffixes (e.g., "B.V.") that are missing in predicted entities, or vice versa.
2. **Entity Variations:** the model might predict variations of the same entity at the same time (e.g., a full company name and an abbreviated version). Any of these variations should not automatically be considered a False Positive if it aligns with the intent of the ground truth label.

To handle these challenges, the following strategies were implemented:

1. **Deduplication of Predicted Entities:**

- Predicted entities are analyzed to identify duplicates caused by variations in formatting, partial overlaps, or repeated patterns.
- Each pair of entities is compared using two main criteria:
  (a) **Substring Containment:** if one entity is fully contained within another (e.g., `Company XYZ` in `Company XYZ B.V.`), the shorter entity is considered redundant.
  (b) **Textual Similarity:** the `fuzzy.ratio` function from the `rapidfuzz` library is used to calculate a similarity score between two entities. If the score exceeds 70%, the entities are considered duplicates. This value has been chosen with the aim of striking a balance between precision and recall in entity deduplication. Higher thresholds, such as 85%, would overly prioritize precision, potentially missing meaningful partial matches. Conversely, lower thresholds, such as 60%, increase the risk of unrelated entities being incorrectly flagged as duplicates. At 70%, the similarity score effectively captures variations while minimizing excessive noise.
- In cases of duplication, the longer or more complete entity is retained, while shorter or less specific duplicates are removed.
- This deduplication process is applied recursively across all predicted entities until no further duplicates are detected, ensuring a refined and non-redundant final list.

2. **Fuzzy Matching Between Predicted and Ground Truth Entities:**
   - Each deduplicated predicted entity is compared against all ground truth entities using a fuzzy matching algorithm.
   - Similar to the deduplication step, a similarity threshold is established to determine whether a predicted (deduplicated) entity is present in the ground truth set.
   - A match is considered valid if it exceeds a predefined similarity threshold of 50%. While this value might seem low, an inspection of the prediction revealed it effectively accounts for variations in formatting, abbreviations, or missing suffixes (e.g., *Linthorst Energie Services B B.V.* (ground truth) vs. *Energie Services B B.V.*). A higher threshold risks overlooking valid matches, while a lower threshold may introduce false positives. At 50%, the threshold seems to nicely balance precision and recall, ensuring meaningful matches are captured without introducing excessive noise. Tables A.3.1.3, A.3.1.4 and A.3.1.5 show a random sample of ten true positives, FP and FN, respectively. Inspecting the mistakes, we can see that among the FP the models incorrectly split entities (e.g. *Cogas* and *Duurzaam* instead of *Cogas Duurzaam*), caught parts of them (e.g. *Clean Energy B.V.* instead of *Holthausen Clean Energy B.V.* or predicted entities which were not in the ground truth (*Delta Comfort B.V.*). While the first two mistakes are harder to address and likely due to the token proximity limitation of the model, the last one shows that the model can easily predict entities that are of the desired type (i.e. ORG), but it cannot distinguish between recipients and other parties mentioned (which is a problem if the goal is restricted to extracting those only). On the other hand, inspecting the FN is relatively less interesting, as we can only see that either all or a subset of the entities are missing in the predictions, which can be traced back to the model not being sufficiently exposed to them while training.

Results

The NER model achieved the following metrics, demonstrating strong performance across key evaluation criteria.

Table 3.2: Custom NER model performance

Metric	Score
Precision	0.93
Recall	0.78
F1-Score	0.85

The high precision suggests that the model reliably predicts entities with few false positives. However, the lower recall reveals that approximately 22% of ground-truth entities were missed, often due to variations in formatting, suffixes, or underrepresented patterns in the training data. This issue is particularly evident in Table A.3.1.5, where several entities were entirely missed. These omissions are likely due to insufficient representation of certain entity patterns or formatting variations in the training data, preventing the model from learning to recognize them effectively. While the results are promising, especially in terms of precision, future improvements could focus on better handling of entity variations, refining similarity thresholds, and expanding or diversifying the training dataset to boost recall without compromising precision.
Despite these positive outcomes, it is also important to consider that spaCy's NER performance might have been slightly overestimated due to the potential presence of some parties appearing in multiple decisions (with different IDs), potentially resulting in overlap between the train and test sets. This overlap could inflate performance metrics during evaluation, as the model might be tested on entities it has already encountered during training. Although this is not inherently problematic—training a model with comprehensive examples of all entities is ideal—it highlights a key limitation of the approach: the model struggles to generalize to out-of-train entities.
To enhance the approach's effectiveness and reliability, creating and maintaining a comprehensive, regularly updated list of companies (e.g., registered companies in the Netherlands) could be instrumental. This list, combined with rule-based techniques (e.g., generating variations of entity names by adding or removing suffixes, handling commonly omitted words in company names, or incorporating acronyms), and fuzzy matching, could support the development of a more robust labeled dataset for training a NER pipeline. Such a dataset could improve generalization by enhancing the model's ability to recognize entities, particularly in cases involving spelling variations, newly introduced entities, or variations in naming conventions. For instance, a company like *Siemens Energy* might appear in different forms, such as *Siemens Energy GmbH*[10], *Siemens Energy AG*[11], or simply *Siemens*. With a comprehensive database, these variations can be systematically generated and matched during the entity recognition process, ensuring better coverage and accuracy.
Despite these limitations, the results remain encouraging, given the inherent complexity of the task. Supervised NER approaches offer clear advantages over purely rule-based methods, particularly in their ability to adapt to linguistic and contextual variations—an essential feature when dealing with the significant variability in how company names are written. Combining NER with rule-based techniques to create a labeled dataset could yield even better outcomes. These findings suggest that a hybrid approach, leveraging the strengths of both supervised NER and rule-based methods, offers significant promise. By fine-tuning[12] the NER model and expanding the training dataset, further improvements in performance could be achieved. Future work could focus on addressing these challenges to unlock the full potential of this approach.

[10] *Gesellschaft mit beschränkter Haftung*, i.e. a company with limited liability.
[11] *Aktiengesellschaft*, i.e. public limited company.
[12] While fine-tuning was considered during this research, challenges in implementation and time constraints prevented its full exploration.

3.3 LLM-based Recipient Extraction

The LLM-based approach used earlier for spelling correction (Ch. 2.4) can also be exploited for extracting the parties involved in a decision by changing the prompt and asking for a search in the document of the decision recipients. This approach requires no training data and leverages the exposure of the model to a vast training set, which likely contains the names of the companies mentioned in the input decisions.

3.3.1 Implementation

This approach was implemented using the `gpt-4o` model by OpenAI, with a temperature setting of zero to ensure deterministic outputs. While `gpt-4o` was employed, a smaller variant like `gpt-4o-mini` would likely have sufficed, given the task's emphasis on entity extraction rather than complex reasoning. The task, this time formulated in English (as it was determined that language did not pose a barrier for the LLM), focused on identifying parties mentioned in the ACM permits, using information from the headline and description fields. A system prompt[13] was designed to define the model's role and overarching objective, while a detailed user prompt provided explicit guidelines for extraction. These instructions emphasized identifying names of public or private organizations in their most complete written form (see examples below in the prompt), excluding individuals, generic references, or regulatory mentions. Multiplicity in the output was permitted, as certain decisions involved more than one recipient. Additionally, reverse lookups were applied to validate that extracted names were explicitly present in the input text, trying to mitigate the risk of hallucinated outputs. The full prompt is presented below.

Decision Recipient Extraction Prompt

Given the following headline and description (or text, in case the headline and/or description are unavailable) of an administrative decision, return a list of strings representing each of the parties involved in the decision[14], as they are written in the headline, description, or text.
Names might be written in non-(fully)-standard forms, for instance:

- "Huismerk" instead of "Huismerk Energie N.V."
- "Treant Ziekenhuiszorg" instead of "Stichting Treant Ziekenhuiszorg"
- "GDF SUEZ" instead of "GDF SUEZ Energie Nederland N.V."

Guidelines for Extraction:

1. Entity Type: Extract only private or public companies and organizations (not individual persons). Ignore generic references to laws, procedures, or regulatory items (e.g., *Leveringsvergunningen Kleinverbruik Ex Art. 45 Gaswet*) unless no other suitable party is found. Ignore ACM, as it is always involved in these decisions and does not need to be listed as a party.
2. Entity Form: Each identified party must appear in its **most complete form** as it is written in the text. Include all suffixes, such as "B.V." or other relevant elements. Avoid repeating entities by selecting the most complete version (e.g., NO "Huismerk" and "Huismerk Energie N.V." — only "Huismerk Energie N.V." should be included).
3. Reverse Lookup: Ensure extracted names are **present in the text** by performing reverse lookups to validate their existence. Do not create or infer names that are not explicitly present.
4. Output Format: If parties are identified, return a list of strings in this format:
   output = [''party_1'', ''party_2'', ''party_3'']
5. If no suitable parties are identified, return a list containing only the string: ''no_party'':
   output = [''no_party'']
6. Restrictions:

[13] *You are a helpful assistant for entity identification and labeling.*

[13] It was later realized during the research that this formulation might be ambiguous, as "involved" parties could be interpreted as parties merely mentioned rather than those "directly addressed" in the decision. The next iteration of this experiment should improve this by rephrasing to a more unambiguous formulation.

(a) DO NOT modify or improve the names in any way.

(b) DO NOT output anything outside the specified formats.

Input Fields:

- Decision headline: `decision_headline`
- Decision description: `decision_description`
- Decision text: `decision_text`

Evaluation

In total, 232 distinct parties were extracted, which is nine more than the ground truth set. Additionally, 28 decisions were assigned no party, matching the number observed in the ground truth. While all 223 ground truth entities are included in the set of predicted entities, this does not guarantee strong model performance; for instance, the model might fail to predict a party where it should have (FN) and compensate by predicting it incorrectly in another location (FP). Evaluation was done by comparing the predicted parties to the ground truth ones. However, a strict 1-to-1 matching approach is overly pessimistic due to variations in how ground truth entities are written and found in the text. Therefore, a three-step evaluation strategy was employed:

1. An initial *exact match* attempt is made between predicted and ground truth entities.
2. If the first attempt fails, suffix variations such as *B.V.* or *N.V.* are added to predicted entities for a second matching attempt.
3. If both (1) and (2) fail, a *fuzzy matching* step is performed as a final effort to align predicted and ground truth entities.

→ If no match is found after the third step (based on a chosen similarity threshold), the prediction is marked as a false positive (FP).

The selection of the similarity threshold for fuzzy matching was conducted by inspecting unmatched predictions (65 out of 481 predictions) and comparing their similarity scores with unmatched ground truth entities. An analysis of these scores revealed that a threshold of 70 was optimal, as almost no false matches occurred above this value (only two out of thirty-one).

Results

According to this evaluation strategy, the final metrics/counts are summarized below in Tables 3.3 and 3.4.

Table 3.3: Proportion metrics

Metric	Value
Average proportion of (possibly fuzzy) matched entities	0.97
Precision	0.84
Recall	0.92
F1-Score	0.88

Table 3.4: Count metrics

Metric	Count
Total number of true positives	406
Total number of false positives	75
Total number of false negatives	34

The high average proportion of matched entities (0.97) suggests that the model generally aligns its predictions closely with the ground truth. The relatively low number of false positives (75) indicates that incorrect predictions are limited in scope, while the small count of false negatives (34) demonstrates that the model rarely overlooks relevant entities. With a precision score of 0.84, the model ensures that a significant majority of its predictions are accurate. Additionally,

the recall score of 0.92 underscores the model's capability to capture the most relevant entities effectively.
The results are particularly satisfactory given the *zero-shot* nature of the approach and the inherent difficulty of the task, especially when compared to more traditional or structured extraction methods. The high recall indicates that, of all entities present in the dataset, the majority are correctly identified by the model, which is a highly desirable characteristic for this task. On the other hand, the slightly lower precision score indicates that there is a risk of incorrectly predicted entities, which is still quite high.
Nonetheless, an important consideration regarding precision—also relevant to the previous approach—can be made. The precision metric, as calculated here, adheres to a strict definition specifically tailored to extracting only decision recipients. This approach distinguishes between a party formally designated as a recipient and one merely mentioned in the text (for any other reason). However, this distinction can sometimes become blurred in practical scenarios. In real-world applications, extracting all mentioned parties in legal decisions—regardless of whether they are formally designated as recipients—could provide substantial value. For example, identifying recurring entities across multiple decisions might reveal systemic patterns or highlight key stakeholders in regulatory processes.
Performing reverse lookups of the predicted parties in the headline, description, and text of each decision revealed that over 99% of the predictions could be traced back to one of these sources. This indicates that hallucination was not a significant issue—or at least not a major contributor to false positive predictions. If it were possible to validate not only that all predicted entities appear in the text, but also that they represent valid entities (e.g., company names), this approach could become even more promising for extracting parties—beyond just recipients—from legal decisions. However, the feasibility of this broader application ultimately depends on the specific task requirements and whether such outputs align with user needs. If so, NER systems could be more easily implemented, gradually refined, and tailored over time to effectively differentiate between recipients and non-recipients.

3.4 Comparison

Table 3.5: Model performance comparison

	SpaCy Custom NER	**LLM-based NER**
Precision	0.93	0.84
Recall	0.78	0.92
F1-Score	0.85	0.88

The two approaches demonstrated complementary strengths (see Table 3.5) in extracting recipient entities from legal decisions. While the supervised NER approach achieved higher precision, ensuring reliable identification of recipient entities with minimal false positives, the LLM-based approach exhibited higher recall, effectively capturing a larger proportion of relevant entities. This complementary behavior suggests that each approach leverages its inherent advantages: NER excels when trained on domain-specific datasets, while LLMs benefit from vast pre-trained knowledge. However, these observations should not be interpreted as a recommendation for combining the two approaches but rather as evidence of their distinct and context-dependent suitability.
The evaluation of the NER model revealed that its performance might have been slightly overestimated due to the presence of overlapping entities across training and test splits. While this overlap is a common challenge in supervised methods, particularly in datasets with recurring entities, it also reflects a realistic scenario where some entities will inevitably appear in both training data and real-world inference tasks. In supervised NER, this is not inherently problematic—ideally, models benefit from extensive exposure to entities. However, it underscores a key limitation: these models perform better on previously seen entities and require extensive,

up-to-date datasets to generalize effectively to unseen ones. Although domain adaptation techniques and periodic retraining could mitigate this challenge, they demand significant resources and careful dataset maintenance. In contrast, an LLM-based approach offers a more flexible and scalable solution, requiring neither large labeled datasets nor periodic retraining (which is taken care of by the LLM's developers themselves, offering an advantage in dynamic and rapidly changing domains).

3.5 Conclusion

Especially in the case of an LLM-based approach (given the positive results achieved in this research), expanding the task from recipient extraction to party extraction emerges as a promising future direction. Extracting all mentioned parties, rather than limiting results strictly to formal recipients, could enrich the metadata associated with each decision. This approach aligns with broader goals, such as identifying patterns across decisions or analyzing recurring stakeholders in regulatory processes. However, it introduces a trade-off: while the volume of extracted metadata would increase, distinguishing between primary decision recipients and secondary mentions might require additional post-processing to maintain interpretability and focus. Ultimately, this would depend on the goal being pursued. For example, if the aim is to develop a search tool for scholars investigating a party's involvement in a certain category of administrative decisions (e.g., uncovering patterns behind a party's mentions, regardless of recipient status), such a distinction between formal recipients and mentions would not be necessary.
On the one hand, fine-tuning SpaCy's pipeline could lead to improved performance, but this requires a significant initial investment in labeling entities, especially if they are not already available as metadata. On the other hand, LLMs remain an attractive and convenient option for recipient extraction due to their *zero-shot* (or *few-shot*) capabilities and independence from extensive domain-specific labeled datasets.
Hybrid approaches—where LLMs perform broad initial extractions and NER models refine results—could also be explored to combine the strengths of both methods. However, their value must be critically weighed against the potential of refining a fully LLM-based solution. Given the already promising results of the LLM approach, further improvements through refined prompts or fine-tuning might yield more substantial gains with fewer resource demands compared to integrating hybrid pipelines.

Chapter 4

Date Extraction

Dates play a crucial role in establishing timelines, identifying key events, and structuring information chronologically. Extracting and interpreting them accurately is essential for downstream tasks such as decision classification, event detection, and timeline reconstruction. While dates often appear multiple times in an administrative decision, not all occurrences are equally relevant. For example, a legal text might reference the issuance date of a law, a historical event, or other unrelated temporal markers. These dates, while valid, do not necessarily indicate the key temporal frame of the document itself. Identifying the correct decision date requires context and cannot be inferred merely from the presence of a date entity.
Unlike company names, dates can be easily extracted from text using an NER pipeline[1], such as the one discussed in Chapter 3. However, they lack the ability to determine whether a date is contextually relevant. For example, a NER model might correctly extract a date from a sentence but fail to assess if it represents the document's primary temporal reference. This limitation is addressed using supervised classification algorithms. In this chapter, the challenge was to identify, among an array of dates, the one referring to a document's issuance. By leveraging labeled data, ML classifiers were employed to predict valid dates based on features that capture the contextual information surrounding each detected date.
Subsequently, a zero-shot LLM-based approach was also applied to address the same task. This approach explores whether the model's ability to understand and interpret context allows it to distinguish between "relevant" and "irrelevant" dates within the text.

4.1 A Short Introduction to Supervised ML Cassifiers for NLP

Traditional supervised machine learning refers to pre-deep learning methods, often relying on algorithms such as logistic regression, support vector machines (SVMs), and decision trees, where models learn to predict outcomes based on labeled training data. In essence, these algorithms are provided with input-output pairs, where the input represents the features of the data, and the output represents the desired result. The model then learns to generalize from these examples, identifying patterns and relationships to make predictions on unseen data.
Supervised machine learning algorithms can be broadly categorized into two types: regressors and classifiers. While regressors predict continuous numerical values (e.g., predicting housing prices or temperature values), classifiers predict discrete categories or labels (e.g., determining whether an email is spam or not), which is what will be used in this Chapter to discriminate decision dates from irrelevant ones.
Originally, supervised machine learning models were primarily designed to handle quantitative, structured datasets. However, text data is inherently unstructured and non-numerical, requiring additional preprocessing steps to make it compatible with these models. Techniques such as

[1]Dates can be written in many formats but, unlike company names, not in an infinite variety of ways (e.g., `DD/MM/YYYY`, `YYYY/MM/DD`, `Month DD, YYYY`). Thus, a NER model like SpaCy already has the capability to extract dates effectively.

Bag-of-Words (BoW) or Term Frequency-Inverse Document Frequency (TF-IDF) are employed to transform textual data into numerical representations. These representations enable these models to process and learn from textual inputs effectively.
In this case, ML will build upon NER to perform date validation. NER will first extract dates from the decisions, and subsequently, in a second stage, date validation will classify the extracted dates based on whether they correspond to the decision date or not.

4.2 ML Based Date Extraction

4.2.1 Implementation

Obtaining Targets

Having chosen a supervised approach, the first step has been to separate decisions with a ground truth date, from those which did not (315 and 44 respectively). After this, the spaCy `nl_core_news_lg` model has been used to extract a list of all the date entities in each of those 315 documents. On average, each decision had between seven and eight dates, and no document had below one date. More statistics can be found in Tab. 4.1.

Table 4.1: Statistics of the number of NER dates per decision

NER Dates per Decision	Value
Median	7.00
Mean	7.49
Standard Deviation (SD)	5.39
Minimum	1.00
Maximum	71.00

Eventually, this resulted in a set of dates to classify. To get targets for the ML model, each date was compared against the ground truth and assigned a binary label:

$$target\_date = \begin{cases} 1, & \text{if date is equal to ground truth,} \\ 0, & \text{otherwise.} \end{cases}$$

This choice led to an increase in dimensionality compared to the starting datasets, as each date (and no longer decision) represented an item to classify, and each decision contained, on average, six dates (with one of them being the effective decision date in most of the cases). Moreover, an issue of class imbalance emerged, as there are only 367 correct dates against 9277 incorrect ones (3.8%). This will be addressed later in Sec. 4.2.1.

Obtaining Features for Prediction

To be able to carry out predictions, each date must also be provided with a set of features that a ML model can learn from. In this case, the choice was made to use a span of twenty words around each date, since the surrounding context often contains critical information that helps determine the relevance and role of the date in the document. This context can include keywords, entities, or linguistic cues that provide valuable semantic signals for the model to make accurate predictions. Figure 4.1 provides an example of a date and its corresponding twenty[2]-word context window.
To enable ML models to process and learn from the textual spans surrounding target dates, the text must first be transformed into a numerical format. This transformation is accomplished through vectorization, which converts text data into a structured, mathematical representation

[2]In cases where dates appear near the beginning or end of the document (and ten words are not available before or after the date), the algorithm will extract as many words as are available. Fig. 4.1 is an example of this, as can be observed by the words *ons kenmerk* at the start of the left span.

Figure 4.1: Example of context window around a sampled extracted date

that models can interpret. In this research, three commonly used vectorization techniques are implemented and compared. Each technique offers distinct advantages and limitations in representing text spans around dates. Their effectiveness will be evaluated empirically to determine which method provides the best performance for predicting the relevance of target dates.

1. **Bag of Words (BoW)**

   The Bag of Words model represents text as a collection (or "bag") of individual words, disregarding grammar and word order while focusing solely on word frequency. Each unique word in the text corpus becomes a dimension in the resulting feature space, and its value indicates how often it appears in the text span. An alternative focusing on word presence/absence (Binary BoW) is also present.

   - **Strengths:** simple and computationally efficient.
   - **Weaknesses:** ignores word order and semantic meaning and may result in a sparse feature matrix when working with large vocabularies (however, given the small number of words in each word span, this issue should be somewhat contained).

2. **Bag of N-Grams**

   The Bag of N-Grams builds upon the BoW, but instead of using words, it uses n-grams, which are continuous sequences of n words or tokens (e.g., pairs or triples of words). This approach helps capture local contextual relationships between words that are otherwise lost in traditional BoW representations.

   - **Strengths:** captures word frequency effectively and supports n-grams for better context representation.
   - **Weaknesses:** still treats all words with equal importance and does not handle rare words effectively.

3. **TF-IDF**

   TF-IDF (Term Frequency-Inverse Document Frequency) is an extension of the non-binary BoW vectorizer that not only considers word/n-gram frequency but also evaluates the importance of each word relative to the entire corpus.

   - **Term Frequency (TF):** measures how often a term appears in a specific text span.

   $$\mathrm{TF}(t, s) = \frac{\text{Number of occurrences of term } t \text{ in span } s}{\text{Total number of terms in the span } s} \tag{4.1}$$

   - **Inverse Document Frequency (IDF):** measures the importance of a term across the corpus: it weighs down the terms that are very common while it weighs up the rare terms.

   $$\mathrm{IDF}(t) = \log_e \left(\frac{\text{Total number of documents in the corpus}}{\text{Number of documents with term } t \text{ in them}} \right) \tag{4.2}$$

   Note: the "Total number of documents in the corpus" refers to all word spans.

The final TF-IDF value is calculated as the product of 4.1 and 4.2.

- **Strengths:** TF-IDF reduces the impact of common, non-informative words and emphasizes rare yet important words.
- **Weaknesses:** still lacks word order and semantic understanding.

Created Features

All of the above-mentioned vectorizers have been tried with unigrams and also n-grams ($n \leq 3$) to capture both granular and broader contextual patterns, jointly or alone, and see which combination can lead to better performance. Here below is an overview of the configurations used for each vectorizer (five for each of the three vectorizers).

1. Unigrams only
2. Bigrams only
3. Trigrams only
4. Unigrams and bigrams
5. Unigrams, bigrams and trigrams

Data Imbalance

The issue of data imbalance emerged after transitioning from decision-level to date-level classification (as discussed in Sec. 4.2.1). Each date now represents an individual item for classification rather than an entire decision. This significantly increased the dataset's dimensionality, with each decision containing, on average, six dates. More importantly, it introduced a stark imbalance in class representation: only 367 dates are labeled as correct, compared to 9,277 incorrect ones, resulting in a minority class proportion of just 3.8%.
Addressing data imbalance is essential because an overrepresentation of one class in the training data can introduce significant bias. Machine learning classifiers trained on imbalanced data are prone to favoring the majority class, often predicting instances of that class not because of their ability to generalize effectively but simply due to having seen more examples of it during training. This phenomenon reduces the model's ability to accurately classify the minority class, leading to skewed predictions and poor performance in real-world scenarios [18].
Many machine learning algorithms inherently optimize for overall accuracy, which exacerbates this problem. Errors in predicting the minority class are often overlooked in favor of achieving high accuracy on the dominant class. Downsampling of the majority class was applied to address the data imbalance issue, ensuring that both classes are adequately represented in the training set. Additionally, since ensemble methods like Random Forests are particularly effective in handling class imbalance [26], an imbalanced split was also made, to see if these models could overcome the imbalance.
In both cases, the splitting strategy mirrored the approach used in Sec. 3.2.2. Unique decision IDs were first partitioned into training and testing sets (80:20) to prevent overlap between dates originating from the same decision. Subsequently, all dates associated with these IDs were grouped into their respective partitions. This methodology ensures that decisions with shared contextual dependencies do not inadvertently appear in both training and testing sets.

ML Classifiers

The previously created features have been used as input to a range of classifiers to empirically determine which combination of classifier and feature set would yield the best performance. This approach ensures that multiple perspectives on the data are considered, leveraging the unique strengths of each algorithm.
A total of nine classifiers[3], each representing different modeling paradigms, were applied across all fifteen vectorization techniques and evaluated both with and without class imbalance adjustments (for a total of 270 combinations):

[3]Sources used for describing the algorithms: [7, 15, 25, 33].

1. **Logistic Regression:** a *discriminative* classifier that aims to learn the probability distribution over all classes. It "learns" the weights for individual features based on how important they are to make a classification decision. The goal is to learn a linear separator between classes in the training data with the aim of maximizing the probability of the data. This "learning" of feature weights and probability distribution over all classes is done through a function called the "logistic" function and (hence the name) logistic regression.

2. **Decision Tree:** a non-linear model that recursively splits data into subsets based on feature values. DT are models that predict the value of a target variable by learning simple decision rules inferred from the data features. Because of their non-parametric nature, they do not assume a specific distribution of the data, which allows them to model complex relationships between features and classes without being overly constrained by the underlying distribution of the classes, making them robust in the presence of class imbalance.

3. **SVM (Support Vector Machine):** a classifier that identifies the optimal hyperplane for separating data into distinct classes. The hyperplane is chosen to maximize the margin, which is the distance between the hyperplane and the nearest data points from each class, known as support vectors. This margin maximization helps improve the model's generalization to unseen data. SVMs are particularly effective in high-dimensional spaces and are versatile, as they can handle both linear and non-linear decision boundaries through the use of kernel functions.

4. **Complement Naive Bayes:** Naive Bayes is a *probabilistic* classifier that uses Bayes' theorem to classify texts based on the evidence seen in training data. It estimates the conditional probability of each feature of a given text for each class based on the occurrence of that feature in that class and multiplies the probabilities of all the features of a given text to compute the final probability of classification for each class. Finally, it chooses the class with maximum probability. **Complement NB** is an adaptation of the standard **Multinomial Naive Bayes (MNB)** algorithm that is particularly suited for imbalanced data sets. Specifically, CNB uses statistics from the complement of each class to compute the model's weights. The inventors of CNB show empirically that the parameter estimates for CNB are more stable than those for MNB. Further, CNB regularly outperforms MNB (often by a considerable margin) on text classification tasks.

5. **K-Nearest Neighbors (KNN):** a simple non-parametric classifier that assigns labels based on the majority class of neighboring points. It can become computationally expensive as the dataset size increases, especially in high-dimensional spaces. Additionally, the choice of k (the number of neighbors) can significantly impact performance.

6. **Random Forest:** an ensemble method that builds multiple decision trees to improve generalization and reduce overfitting. They can be effective on unbalanced datasets as they combine the predictions of multiple trees, which can help balance the influence of minority classes.

7. **Extra Trees:** similar to Random Forest, Extra Trees (Extremely Randomized Trees) is an ensemble method that constructs multiple decision trees. However, while Random Forest selects the best split among a subset of features at each node, Extra Trees chooses splits entirely at random, making it computationally faster and often less prone to overfitting.

8. **AdaBoost:** an adaptive boosting technique that combines multiple weak classifiers (typically decision trees with a single split) to form a strong predictive model. Each subsequent classifier focuses on the mistakes made by the previous ones by assigning higher weights to misclassified samples.

9. **Gradient Boosting:** just like AdaBoost, Gradient Boosting works by sequentially adding predictors to an ensemble, each one correcting its predecessor. However, instead

of improving the instance weights at every iteration as AdaBoost does, this method tries to fit the new predictor to the residual errors made by the previous predictor.

Choosing Vectorization and Classifier

Before training the models, an evaluation strategy was established to ensure alignment with the Chapter's goal. Aggregate metrics, such as accuracy, were avoided as they can obscure class-specific performance, particularly for class 1 (effective decision dates), which is more critical in this context as we don't want to miss out on good decision dates.
Moreover, class-specific metrics provide clearer insights into the model's effectiveness. Among these, recall for class 1 (i.e., the proportion of actual decision dates correctly identified by the model) was prioritized to minimize false negatives, ensuring that fewer valid dates are overlooked. Although achieving perfect precision might not always be feasible (always predicting relevant dates without any false positives), narrowing down candidate dates per document—such as retaining only the top two most likely dates based on their confidence scores or likelihood metrics—significantly simplifies downstream processing. If both dates are similarly likely according to the model, they are retained; otherwise, only the most likely date is kept. Then, if any candidate dates are present, they can either be manually verified or resolved through an automated two-step approach using a fine-tuned, efficient model, such as a large language model.
Figure 4.2 illustrates the forty classifiers maximizing recall for class 1. Surprisingly, the simple K-NN algorithm outperformed all other classifiers across all fifteen vectorizers (with balanced data), achieving 98.4% recall for class 1 regardless of the vectorizer. This means that even with fewer n-grams and without considering word counts, the algorithm can achieve comparable results. However, the heatmap clearly reveals a significant drop in recall for class 0 and precision for class 1. In practice, this means that while K-NN effectively captures most of the important decision dates (class 1), it struggles to correctly identify non-decision dates (low precision for class 0) and often misclassifies irrelevant dates as valid ones (low recall for class 1), reducing overall reliability. This prompted an examination of the "runner-up" models, which are Logistic Regression and Multinomial Naive Bayes. Both scored 96.9% in recall for class 1 (-1.5 percentage points compared to K-NN), but Multinomial Naive Bayes consistently scored less across the other metrics, justifying a preference for the former for later fine-tuning. This is also quite advantageous, as Logistic Regression offers insight into its features and respective coefficients, which makes it possible to understand what words are most impactful in determining the model's predictions.
Figure 4.3 illustrates the performance comparison of the ten models, specifically highlighting their differences in recall rates for class 1. The analysis reveals that ensemble models demonstrated lower performance compared to both the statistical (Logistic Regression) and probabilistic (Naive Bayes) models. Furthermore, a more comprehensive evaluation of the metrics (e.g. F1-score) indicates that KNN would rank at the bottom of the ranking due to its imbalanced metric distribution. In the figure, we can also see that TF-IDF with 1-grams is the best-performing vectorization technique across all models, followed by Binary BoW and traditional BoW (although this can only be seen by inspecting all 270 rows in the data).
Figure A.4.1.1 in the appendix provides an overview of the top ten classifiers evaluated under imbalanced conditions, sorted by their recall for class 1. This visualization highlights the relative performance of each algorithm, offering insights into their effectiveness at handling the minority class. Among these, Complement NB emerged as the best-performing classifier in terms of recall for class 1, achieving an F1-score of 98.4%. However, it exhibited notably low precision (69.9%) on the minority class.

Model Training and Fine-Tuning

Logistic Regression was fine-tuned using the same balanced data splits (approximately 65 dates each) as in Sec. 4.2.1, with features engineered through a TF-IDF vectorizer using 1-grams. No separate validation set was created; instead, 5-fold cross-validation (CV) within the training set ensured comprehensive data exposure and optimal hyperparameter (HP) selection. During

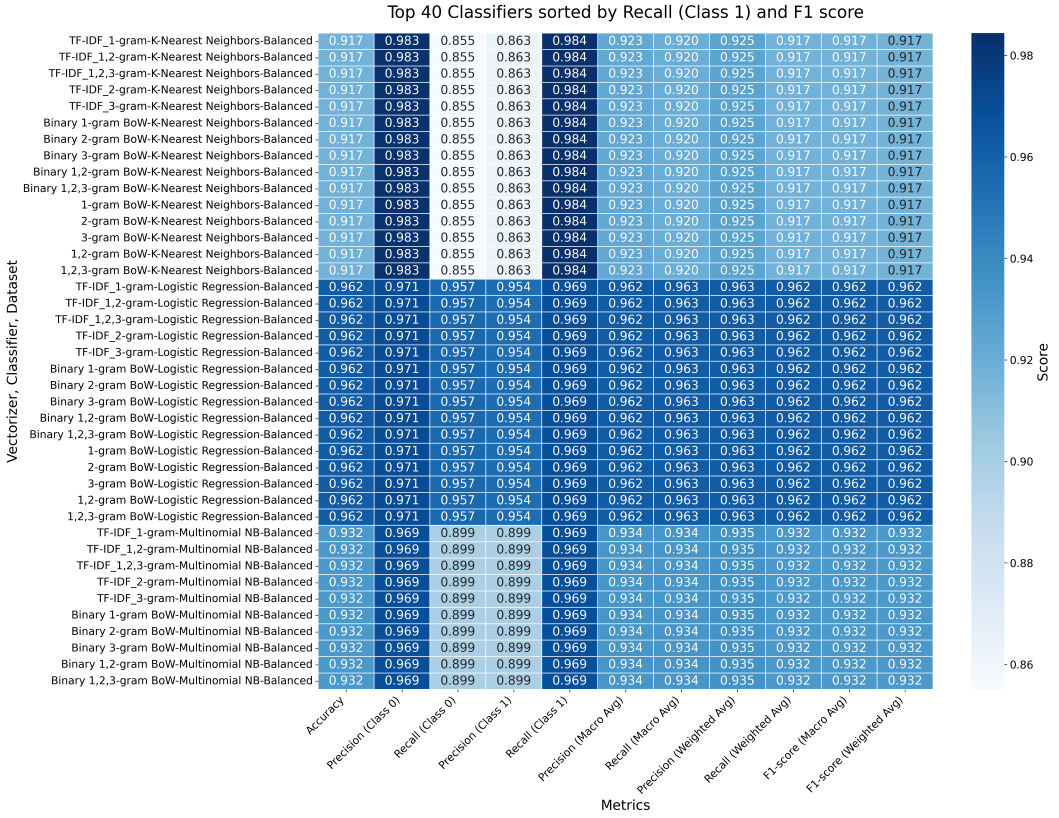


Figure 4.2: Heatmap illustrating the top 40 classifiers, sorted by Recall for Class 1

Figure 4.3: Comparison of the best ten classifiers (based on and sorted by Recall for Class 1)

training, CV systematically evaluated different HP configurations by training and validating on separate subsets of the data. Once the optimal HPs were identified, the model was retrained on the entire training set and subsequently evaluated on the holdout set. Model performance was assessed using the weighted F1 score, which balances precision and recall while accounting for the (very) small difference in sample size between the training (64) and testing (69) sets. Fine-tuning focused on three key HPs:

- C **(regularization strength):** determines the inverse strength of regularization, with

values 0.01, 0.1, 1, 10, and 100. Lower values of C correspond to stronger regularization, which helps reduce overfitting by limiting the influence of less important features. Under the ℓ_1 norm, this can also result in some coefficients being reduced to zero, effectively performing feature selection.

- **Solver:** optimization solver with options `liblinear` and `saga`', where the former is usually suitable for smaller datasets, while the latter tends to be more efficient for larger datasets and sparse data (likely the case as TF-IDF creates sparse matrices[4]).

- **Penalty:** Regularization penalty using either the ℓ_1 norm (Lasso) or the ℓ_2 norm (Ridge), each influencing how coefficients are constrained.

Before fine-tuning, the model achieved a weighted F1 score of 92.5%, with precision and recall at 92.8% for class 0 and 92.2% for class 1. The optimal hyperparameters were identified in a C of 10, ℓ_1 penalty, and `saga`. This configuration suggests that moderate regularization strength ($C = 10$) combined with an ℓ_1 penalty effectively improved performance, while the `saga` solver ensured efficient optimization and convergence, particularly given the sparse representation of TF-IDF features.
After fine-tuning, the weighted F1 score increased to 97%, marking an improvement of nearly five percentage points. Class-specific metrics also showed significant gains, with precision and recall reaching 97.1% for class 0 (+4.3) and 96.9% for class 1 (+4.7). These improvements highlight the model's enhanced ability to correctly identify both effective decision dates (class 1) and non-decision dates (class 0) while maintaining a balance between precision and recall. A summary of the before-after fine-tuning metrics can also be found in Table 4.2.

Table 4.2: Comparison of performance metrics before and after fine-tuning

Metric	Before Fine-Tuning	After Fine-Tuning
Class 0 Precision	0.928	0.971
Class 0 Recall	0.928	0.971
Class 0 F1-Score	0.928	0.971
Class 1 Precision	0.922	0.969
Class 1 Recall	0.922	0.969
Class 1 F1-Score	0.922	0.969
Accuracy	0.925	0.970
Macro Avg F1-Score	0.925	0.970
Weighted Avg F1-Score	0.925	0.970

4.2.2 Predicting Missing Dates

Logistic regression achieves predictions through a linear combination of the model's learned weights and the features extracted from the word span surrounding each date. Each term in the span contributes to the final prediction by either increasing or decreasing the probability of the date being classified as a decision date, depending on the sign and magnitude of its associated coefficient. The learned coefficients provide valuable insight into the importance of specific features. Positive coefficients indicate terms that increase the likelihood of a date being classified as a decision date, while negative coefficients suggest terms that reduce this likelihood. Fig. 4.4 visualizes these coefficients, highlighting the most influential features driving the model's predictions. This analysis helps interpret the model's behavior and assess whether

[4]This occurs because the vocabulary size of the vectorizer, which includes all unique words or n-grams across the entire dataset, is significantly larger than the set of words present in any single document (in this case, the word spans around target dates). As a result, the vector representation for each document contains many zero entries corresponding to words that do not appear in that specific document.

its predictions align with domain expectations. As can be seen, the word "datum" exhibits the highest positive coefficient, strongly indicating its relevance in signaling a decision date. Other positively weighted terms, such as *teammanager*, *af*[5], and *klimaat*[6], suggest that the model can partly capture contextual signals associated with decision-relevant language (especially with words like *datum*, *uit* or *zaaknummer)*. Conversely, negatively weighted features like *2014* and *00* decrease the likelihood of a span being classified as a decision date, possibly due to their frequent occurrence in non-decision contexts or as numerical artifacts. The overall distribution of coefficients highlights a reliance on clear semantic cues while also revealing the potential presence of noise or ambiguous patterns in the training data. These findings suggest that while the model captures meaningful features, further refinement in preprocessing or feature selection could enhance predictive performance.

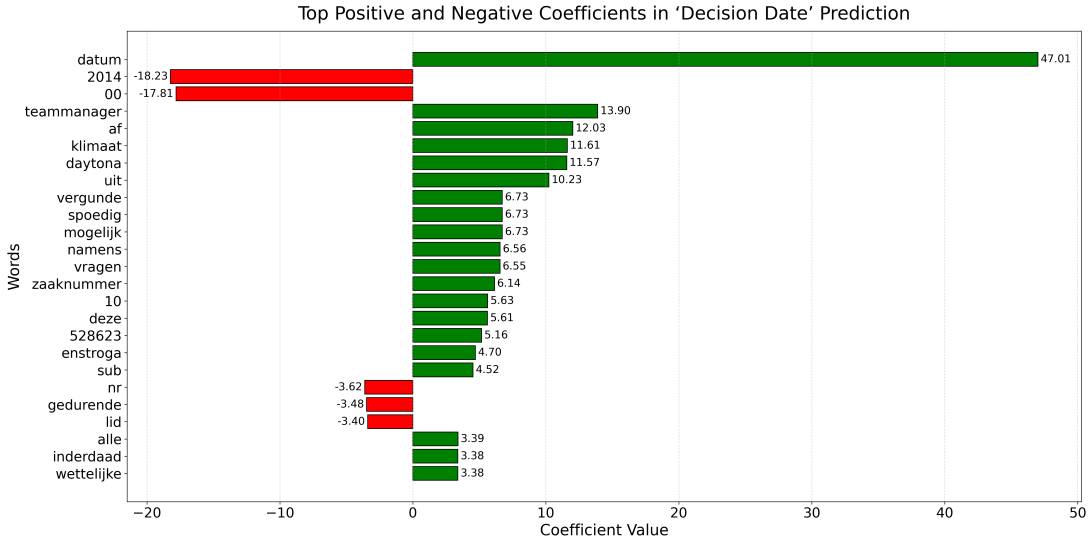


Figure 4.4: Top 25 coefficients in magnitude

Evaluation on Missing Ground Truth Dates

To further evaluate the model's performance, the set of 19 rows lacking ground truth for the decision date was utilized for prediction. It was acknowledged that this subset might exhibit different characteristics compared to the previously evaluated data. The absence of a date in these rows may not solely indicate that no date was present but could also stem from other factors, such as inconsistencies or ambiguities in the original documentation provided by the ACM.
The results were as follows:

- **Correctly Predicted Dates:** the model successfully identified 9 valid dates and correctly recognized 4 instances where a date was missing, resulting in a total of 13 accurate predictions.
- **Missed Dates:** 6 dates were not predicted correctly. Notably, all incorrect predictions exhibited very low likelihood scores, indicating a lack of confidence in the model's output. One of these failures was due to a typo in the date (in the text), which made it difficult for the NER system to recognize it correctly in the first place.

These findings align with prior expectations for lower performance but also demonstrate that, even under more challenging conditions, the model maintained a commendable performance, accurately identifying dates (or their absence) in nearly 7 out of 10 cases.

[5]Dutch preposition meaning *finished*, *completed*, *done*.
[6]Dutch noun meaning *climate* (it can refer to weather conditions or (metaphorically) the general atmosphere or environment in a certain context (e.g., political climate).

4.3 LLM-Based Date Extraction

4.3.1 Implementation

Unlike the previous approach, no train-test split was needed and it was possible to input all 315 permits having a ground truth date (allowing for evaluation) into the LLM. The approach was the same as that used previously for spelling correction and recipient extraction (see Sec. 2.4.2 and 3.3) employing a temperature setting of 0 and an English Language prompt. However, this time the model was switched to gpt-4o-mini. This decision was driven by the excessive costs incurred during recipient extraction and the observation that the current task does not demand the high-level reasoning capabilities of the gpt-4o model. Instead, it primarily requires identifying if a text belongs to an administrative decision (i.e., is addressed to a party, contains decision numbers, etc.) and if it is indeed an administrative decision, then performing date searches within the input text and validating them based on whether they are the decision's date or not.
Two prompts were provided to the model: a "system" prompt[7], which established the model's role and overall objective for the task, and a detailed "user" prompt, which included explicit instructions on how to process the provided text. The model was instructed to strictly output the decision date if any was found or the string "none" conversely. The prompt also asked the LLM to perform reverse searches of the output in the text and further validation to try to ensure its correctness and limit the risk of hallucination. The full prompt can be found here below for a more detailed inspection.

Date Extraction Prompt

You are given the following text. Your task is to determine if it is an administrative decision and, if so, extract the decision issuance date. Strictly follow these guidelines:

1. Identify if the text is an administrative decision or not (it could be a public announcement, for instance).
2. Extract the decision issuance date if available. Ignore irrelevant dates (e.g., dates mentioned in the text but not related to the decision issuance—e.g. law dates).
3. Do not predict a date based on other elements like laws and regulations.
4. Verify the identified date exists in the text (reverse look it up) and then ensure it is indeed the decision date.

Given the following text, analyze it and determine if it is an administrative decision. If so, output the decision issuance date, if available. Most decision texts contain multiple dates, with some of them being irrelevant (for example, dates mentioned in the text but not related to the date on which the decision was issued). If no date is found, don't try to predict it based on other decision elements like laws and regulations. This is highly deprecated. Input: `{decision_text}` Output Format:

- If a decision date is identified, return it as a string in the format 'yyyy-mm-dd'. DO NOT return any other information/justification/reasoning. ONLY THE DATE!
- If no decision date is identified, return the string 'none'. DO NOT return any other information/justification/reasoning. ONLY THE DATE!

Results

The evaluation compares the model's predictions against the labeled ground truth data, focusing on precision, recall, and F1 Score. The following categories were used to classify predictions:

- **True Positive (TP):** the predicted date matches the ground truth issuance date.
- **False Positive (FP):** the LLM predicts a date, but it does not match the ground truth.
- **False Negative (FN):** The LLM fails to predict a date where one exists in the ground truth.

[7] *You are a helpful assistant for extracting administrative decisions' issuance dates.*

Rows without ground truth dates were excluded from the evaluation to ensure consistency with the previous approach. The evaluation yielded the following results:

Table 4.3: LLM date extraction evaluation results

Metric	Value
TP	241
FP	38
FN	36
Precision	0.86
Recall	0.87
F1 Score	0.87

These metrics indicate strong overall performance by the LLM-based approach, with precision and recall values closely aligned, resulting in an F1 Score of 0.87.
The observed precision and recall scores suggest the model is generally able to discriminate dates and extract the correct ones. However, the presence of false positives suggests that the model occasionally struggles with contextual disambiguation when multiple temporal references are present in the same document. In addition, further analysis of the predicted dates against all dates present in the text could give insight into whether some of the false positives are due to hallucination or are mainly due to the model struggling with date multiplicity and ambiguity. Due to time constraints this was not explored in the current research, but proceeding in this direction would help understand the model's mistakes and reflect on ways to counter them.
Overall, the results suggest that LLMs are capable of performing date extraction tasks with commendable accuracy, even without task-specific fine-tuning. Their adaptability across diverse document structures and ability to infer contextually relevant dates offer significant advantages, particularly in scenarios where labeled data is scarce or prohibitively expensive to produce. However, the observed false positives and negatives indicate that the approach is not entirely foolproof. Structural variability and ambiguous contexts remain challenges that would highly benefit from further investigation. Further error analysis and refinements could enhance the performance of LLM-based date extraction workflows, ultimately contributing to more reliable metadata processing in large-scale legal document datasets.

4.3.2 Predicting Missing Dates

As in the previous approach, decisions lacking ground truth metadata about their issuance date were fed to the LLM to further analyze its behavior. The results were as follows:

- **Correctly predicted dates:** the LLM correctly predicted 11 decision dates and identified 4 documents with no date, resulting in a total of 15 accurate predictions (2 more than the ML-based approach).
- **Missed dates:** two decisions were not assigned a date, despite having one in the text.
- **Incorrect predictions:** this category was absent in the previous approach because decisions with a likelihood below 65% were immediately discarded, preventing incorrect predictions within the small sample used. However, since there is no notion of likelihood when using an LLM, such an approach was not applicable in this context. Thus, the LLM produced two incorrect predictions: one date was off by 20 days compared to the ground truth, while the other was entirely incorrect and did not correspond to any date present in the text (a potential hallucination).

These findings indicate that the LLM can still deliver strong performance even when the presence of a date is more uncertain, as is the case in this dataset, which contains more non-decisions than the one with ground truth dates. Compared to the previous approach and on this small dataset, the LLM appears to identify one more decision per ten analyzed (8 versus 7 in the ML-based approach).

However, the risk of hallucination cannot be overlooked. With limited insight into the model's prediction process, it is challenging to determine whether the date that was only 20 days off resulted from an oversight—perhaps caused by noise or ambiguities in the text—or if it represents a case of partial hallucination. The presence of a completely made-up date in the predictions further underscores the risk of such behavior.
The results of this small evaluation are, therefore, purely indicative and limited in scope. More extensive testing and analysis would be necessary to better understand the risks associated with this approach, which otherwise seems to perform the date extraction task quite effectively.

4.4 Comparison

Both approaches demonstrated strong performance (see Tab. 4.4), though with notable differences in their strengths and limitations. On the one hand, the fine-tuned ML-based approach delivered outstanding results, achieving an F1-score of 97%. This high performance highlights the effectiveness of supervised ML models when trained on context-relevant, high-quality labeled datasets. In contrast, the LLM-based approach achieved an F1-score of 87%. While this result is lower, it is still impressive, given that the LLM required no specific training on the dataset. This demonstrates that, despite their simplicity, traditional ML algorithms can still outperform more complex and modern models when applied in well-defined and consistent data environments.

Table 4.4: Comparison of ML-Based and LLM-based date extraction evaluation results

	ML-Based Approach	**LLM-Based Approach**
Precision	0.97	0.86
Recall	0.97	0.87
F1 Score	0.97	0.87

However, additional testing on a subset of decisions lacking ground truth metadata revealed an inverse trend (see Table 4.5). In this scenario, the LLM outperformed the ML-based approach in terms of prediction accuracy. This difference likely stems from the subset containing documents less similar to the training data used for the supervised ML model. The ML approach struggled to generalize to unfamiliar contexts, as it heavily relies on learned patterns tied to the structure and vocabulary seen during training.

Table 4.5: Performance comparison for predicting missing dates

	ML-Based Approach	**LLM-Based Approach**
Correctly Predicted Dates	13 (9 dates, 4 no dates)	15 (11 dates, 4 no dates)
Missed Dates	6	2
Incorrect Predictions	not applicable	2
Accuracy	68.4%	78.9%

This limitation highlights an important weakness of supervised ML approaches: their dependence on the similarity between training and deployment data. If training data predominantly represent decisions from a single authority or reflect formatting and writing styles that may evolve over time, the model's reliability and effectiveness are destined to decay unless periodic retraining or adaptation occurs. Nonetheless, as metadata associated with administrative decisions will become increasingly standardized, available, and accurate, the adaptation burden on supervised ML models is expected to diminish. Under such conditions, supervised ML approaches will continue to offer a reliable and efficient tool for metadata extraction, particularly in structured and predictable datasets.

Conversely, LLMs offer a ready-to-use alternative that delivers decent performance without the need for extensive labeled datasets. Their adaptability allows them to handle a variety of contexts and document structures, as demonstrated by their ability to make accurate predictions even where the supervised ML approach struggled. However, LLM-based approaches come with their own set of challenges. Hallucination remains a significant risk, as seen in the incorrect date predictions, including one that was entirely fabricated. Furthermore, interpreting the reasoning behind an LLM's predictions remains opaque, making it difficult to diagnose whether errors stem from noise in the text, ambiguities, or inherent weaknesses in the model's reasoning process.

4.5 Conclusion

When deciding between these two approaches, the specific characteristics of the dataset and extraction context should guide the choice. The ML-based approach is best suited for scenarios where date metadata is consistently present, accurate, and follows a relatively uniform format across documents. Under such conditions, the model can generalize effectively, delivering high precision and reliability. However, when these conditions are not met—such as in datasets with significant variability in formatting, inconsistent metadata availability, or a higher proportion of non-standard documents—the LLM-based approach becomes a more viable alternative. Its adaptability allows it to handle diverse and unstructured contexts, offering reasonable accuracy even without prior training. Nonetheless, the advantages of LLMs should be carefully weighed against their inherent risks, including hallucination and lower overall accuracy compared to a well-tuned supervised ML model in ideal conditions. Despite these limitations, both approaches have shown strong potential for addressing the metadata extraction challenge, contributing valuable insights to the broader field of legal document analysis.

Chapter 5

Legal References Extraction

A legal basis consists of the regulations, laws, and directives that form the foundation of a legal decision. It is a subset of legal references, which encompass all citations to legal texts within a decision. Extracting legal references is crucial as it enables deeper comparison and analysis of legal decisions, moving beyond surface-level metadata such as involved parties and dates. However, identifying the legal basis specifically within the broader set of legal references is particularly challenging, as it requires a comprehensive understanding of the legal text and its context. Therefore, this chapter focuses primarily on extracting all legal references, leaving to future research the task of discriminating which references constitute the legal basis of a decision.

At the heart of legal communication lie two principal components: words and citations [29]. Citations define how legal documents—and the propositions, facts, and arguments they contain—connect to the broader framework of legislation, jurisprudence, and legal doctrine. In the legal domain, these connections between concepts and resources are not only more intricate but also carry far greater significance than in many other fields. As a result, the ability to search by citation and navigate through this complex network of references is essential for meaningful legal analysis [34]. By focusing on these foundational legal principles and references, comparisons between cases can highlight the reasoning and legal grounds underlying each decision. This approach offers valuable insights into how similar legal bases can lead to different outcomes, benefiting scholars, legal practitioners, and other stakeholders. Ultimately, accurate extraction of legal references plays a critical role in enhancing transparency, fairness, and consistency across legal systems.

From an information extraction perspective, identifying legal references presents significant challenges. The legal domain encompasses an extensive and ever-growing body of laws, regulations, and directives, coupled with highly diverse citation styles and conventions. These citation styles often vary across jurisdictions and even within individual legal systems, complicating both the extraction and normalization of legal references. This makes traditional machine learning and deep learning approaches less effective—especially in the absence of labeled datasets, which are essential for supervised model training.

In scenarios where extensive domain expertise is available, a rule-based system could theoretically be developed. Such a system would need to account for the vast range of citation patterns and inconsistencies across legal texts. However, this approach requires substantial time investment, effort, and specialized expertise, making it difficult to pursue or maintain consistently.

On the other hand, the rise of LLMs trained on unprecedented volumes of text data offers an appealing alternative. These models have likely been exposed to diverse legal texts during their training, which suggests they might generalize well to varying citation styles and patterns. While identifying legal references is a task LLMs seem well-suited for, challenges remain, particularly in grouping references under consistent identifiers without access to domain-specific databases or implementing multi-step validation approaches.

This chapter aims to compare two contrasting approaches to legal references extraction:

1. A rule-based system developed by *Logius* and *KOOP*, which relies on pre-defined patterns

and rules to identify legal references.

2. An LLM-based system leveraging the OpenAI API only.

Through this comparison, the chapter seeks to evaluate the strengths, limitations, and practical applicability of these two methodologies in the context of legal reference extraction.

5.1 The LinkExtractor

The *LinkeXtractor* emerged as a key solution to address the persistent challenge of missing cross-references in legal documents. Initially developed as part of a PhD research project, the tool aimed to automate the detection of legal references to assess the importance of court decisions based on outgoing and incoming citations. While the prototype achieved satisfactory detection performance, it was not designed for scalability, adaptability, or long-term multi-user use.
However, recognizing its potential, the Linked Government Data Project (LiDO), launched in 2012, adopted the *LinkeXtractor* as a foundational tool to improve accessibility to legal information in the public sector. LiDO's primary goal was to create a technical infrastructure for storing, importing, cleansing, and distributing legal resources via linked data. It also introduced a "link tool"—a user-friendly web application for generating uniform and persistent hyperlinks in legal documents. Despite these advancements, manually linking references across various sources, such as court decisions, parliamentary documents, and policy guidelines, proved to be too time-consuming and required specialized knowledge. To overcome these limitations, the *LinkeXtractor* was redeveloped from scratch by KOOP[1] within the LiDO framework.
Eventually, in October 2017, the *LinkeXtractor* went live alongside an updated version of the LiDO website. Through these combined efforts, the LiDO project successfully established an ecosystem for automated legal reference detection and management. This initiative not only resolved the persistent issue of missing cross-references but also significantly improved the efficiency, reliability, and accessibility of legal information systems, creating a more cohesive and accessible legal data ecosystem [34].

5.1.1 LinkeXtractor Architecture and Documented Performance

The LinkeXtractor (LX) pipeline consists of several components designed to process different types of documents effectively. Its main components are presented in Sec. A.5.2 of the appendix and can also be found (though in Dutch only) in the LX repository, in the documentation section[2].
The precision and recall of LX are influenced by various challenges, as achieving a perfect F-score is unrealistic due to the wide range of variations in legal references. Issues such as mismatches, false positives, false negatives, ambiguity, and null results are common, and errors can arise from both the LX system itself (algorithms and reference repositories) and the text being processed (e.g., incomplete or incorrect references). The "80-20 rule" applies here, as LX is adept at recognizing standard references like ECLIs[3], but handling the many variations in case number spellings and sub-article references can complicate the process. Additionally, LX performs better with texts authored by legally trained professionals, while its performance may drop when processing documents written by non-legally trained individuals.

5.1.2 Implementation and Results

Thanks to the LX developers, access to the LiDO server was made possible, completing the LX pipeline and significantly improving the reliability of its results. This access allows the API to perform lookups, validate references, and provide additional metadata, such as URIs and identifiers. Using this, a script was executed to extract legal references from the decision texts.

[1] *Kennis en Exploitatiecentrum Officiële Overheidspublicaties*, i.e. Publications Office of the Netherlands
[2] See: `https://gitlab.com/koop/ld/lx/linkextractor`
[3] *European Case Law Identifier*: a standardized system for uniquely citing court judgments across Europe.

For each decision, the script produced a dictionary containing the extracted legal references (as written in the text) and their corresponding legal category: *wet*, *officiële publicatie*, *uitspraak*, or *europese-regelgeving*[4]. A total of 4,577 legal references were extracted, with a median of 12, a mean of 12.75 per document, and a standard deviation of 7.31, indicating considerable variation across different permits. Notably, in 4 out of 359 cases (1.11%), the LX was unable to extract any legal references. A histogram showing the number of legal references and a bar chart depicting the distribution of references across the four categories are presented in Figures 5.4 and 5.5.

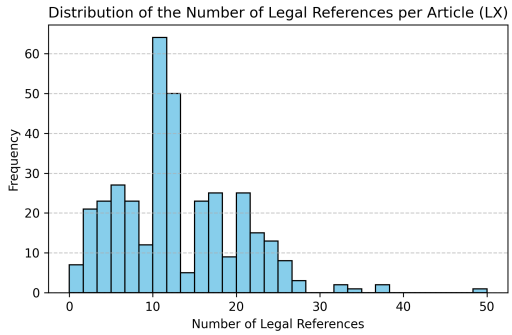


Figure 5.1: Distribution of Amount of Legal References per permit.

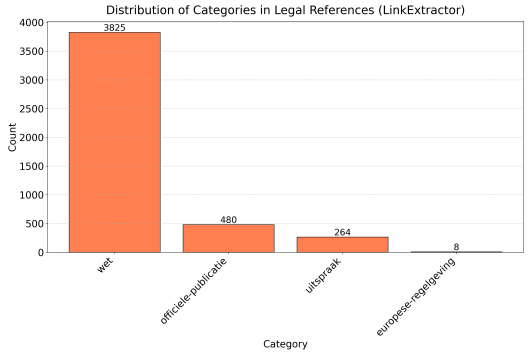


Figure 5.2: Distribution of Amount of Legal References per category.

5.2 LLM-Based Legal References Extraction

5.2.1 Implementation

This approach closely followed the methodology previously used for spelling correction, date, and recipient extraction (see Sec. 2.4.2, 3.3 and 4.3), employing a temperature setting of 0, an English-language prompt and `gpt-4o-mini`.
Two prompts were provided to the model: a "system" prompt, which established the model's role and overall objective for the task, and a detailed "user" prompt, which included explicit instructions on how to process the provided text. The model was instructed to produce output in a structure resembling that of the *LinkeXtractor* (i.e., the legal reference as it appears in the text, along with its category among the four used by LX). Additionally, the output was required to include the context in which each reference was found to minimize the risk of hallucinated results. To ensure clarity and alignment with expectations (and given the increased difficulty of this type of task), two illustrative examples of the desired output were carefully created and added to the prompt, leveraging a *few-shot* learning approach[5].

Legal Reference Extraction Prompt

Your task is to extract legal references and organize them in a Python dictionary.
The dictionary structure should adhere to the following format:

- Keys: Represent the specific names or identifiers of the legal articles (e.g., "Wet dieren", "Artikel 6:162 BW"), as written in the text.
- Values: Contain a dictionary with two keys:
  - "context": the exact text strings from the decision body where the legal article or

[4] *Law, official publication, decision or European regulation* respectively.
[5] It was acknowledged after completing the experiments and upon reviewing the literature on this topic that adopting *few-shot* learning for other IE tasks would likely have improved the LLM's performance. Due to time constraints, the current experiments were not revised; however, future work should consider this aspect and ensure it is not overlooked.

reference is mentioned.

- "category": the category of the legal article or reference out of this list: 'wet', 'officiele-publicatie', 'uitspraak', 'europese-regelgeving'.

If no legal references (of any of the four given categories) are found in the text, return an empty dictionary ({}). Example Output (only the dictionary!):

```
{
    ''Wet dieren'': {
        ''context'': ''Op grond van de Wet dieren is het verboden
                    om dieren te houden op een wijze die schade
                    toebrengt aan hun welzijn of gezondheid.'',
        ''category'': 'wet'
    },
    ''Burgerlijk Wetboek Artikel 6:162'': {
        ''context'': ''Volgens Artikel 6:162 van het Burgerlijk
                    Wetboek is er sprake van een onrechtmatige
                    daad wanneer iemand schade veroorzaakt door
                    een handeling die in strijd is met het
                    recht.'',
        ''category'': 'wet'
    }
}
```

Input (body of the decision): `{decision_text}`

5.2.2 Results

This approach extracted a total of 3,637 legal references, with a median of 10, an average of 10.13 per document, and a standard deviation of 4.55. To ensure that the model is not hallucinating, a reverse lookup procedure was implemented. This process compares each extracted legal reference against the text in which it was found by the LLM and calculates the proportion of "verified" legal references relative to the total number of extracted references.
The algorithm first attempts an exact match of the legal reference as it appears in the text. If unsuccessful, it proceeds to a fuzzy matching step with a similarity threshold of 80%. This two-step approach accounts for how the LLM extracts references: each key in the dictionary (i.e., legal reference) is a "refined string", where typos, extra spaces, or supplementary phrases necessary for textual flow—but not part of the legal reference—have been removed.
Two illustrative examples are shown in Table 5.1. Without accounting for fuzziness, the evaluation tends to be overly pessimistic, even when the LLM performs correctly. As demonstrated in the examples, minor discrepancies—such as extra spaces (e.g., *artikel-en*) or additional words (e.g., *van de*)—prevent an exact match, resulting in false negatives. Based on manual verification, a similarity threshold of 80% was selected to strike a balance between minimizing false positives and avoiding false negatives.

Table 5.1: Extracted legal reference, text in the decision body and similarity score

Text in Decision Body	**Extracted Legal Reference**	**Score**
artikel en 3:40 en 3:41 van de algemene wet bestuursrecht	*artikel 3:40 en 3:41 van de algemene wet bestuursrecht*	94.44
artikel 7:1a, eerste lid, van de algemene wet bestuursrecht	*artikel 7:1a, eerste lid, algemene wet bestuursrecht*	86.79

The distribution of the proportion of successfully reverse-looked-up references per document revealed a median proportion of 1, with an average proportion of 97.53% (SD = 0.10). This average was slightly influenced by the presence of a few outliers, which are also visible in the

histogram shown in Figure 5.3. Overall, however, it is clear that this time, no hallucinating behavior was encountered. A histogram illustrating the number of legal references per document and a bar chart showing the distribution of references across the four categories are presented in Figures 5.6 and 5.7, respectively. For ease of comparison, they are accompanied by the two plots showing the *LinkExtractor* results.

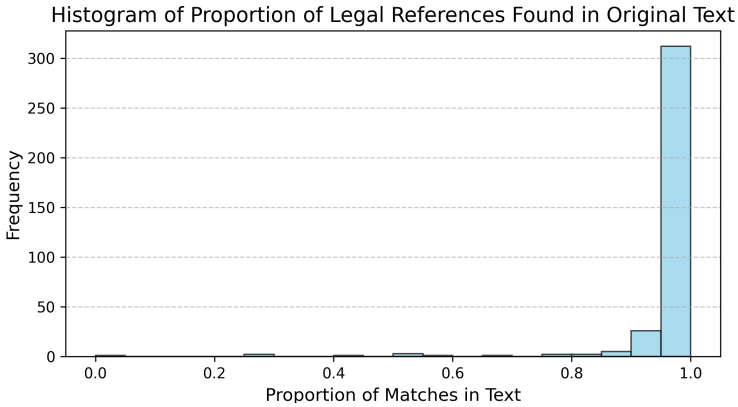


Figure 5.3: Distribution of the proportion of reverse-looked-up extracted references in the original text (similarity threshold: 80%).

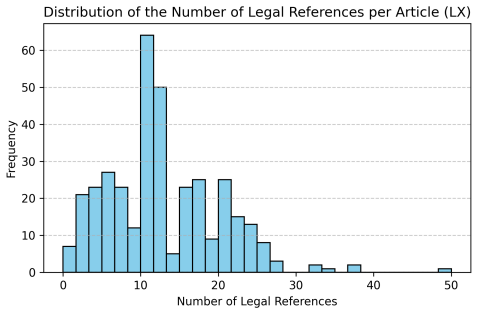


Figure 5.4: (Bis) Distribution of Amount of Legal References per permit

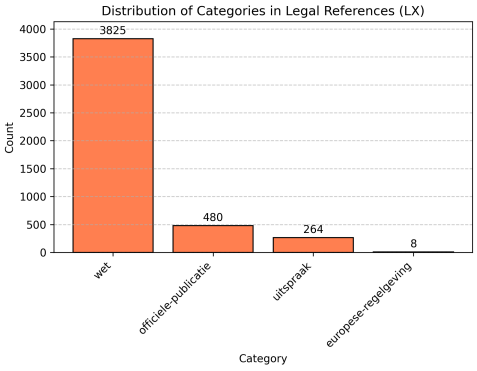


Figure 5.5: (Bis) Distribution of Amount of Legal References per category

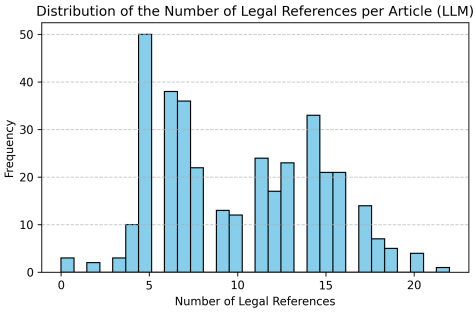


Figure 5.6: Distribution of amount of legal references per permit (LLM)

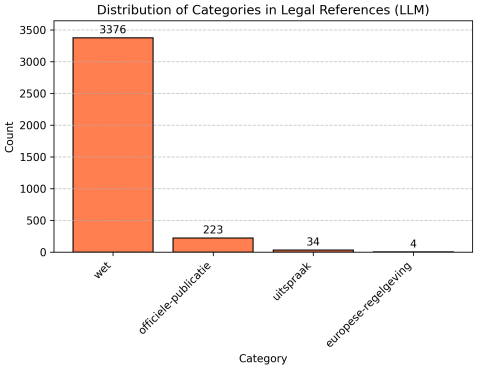


Figure 5.7: Distribution of amount of legal references per category (LLM)

5.3 Comparison

In this section, the performance and characteristics of the LX and the LLM-based legal reference extraction approach will be compared across several dimensions, including the quantity and distribution of extracted legal references, category-specific trends, and the relationship between the extraction performance and the length of the original text.

5.3.1 Quantity of Extracted Legal References

The median number of legal references per document differs slightly between the two approaches, with a median of 12 per document for the LX and 10 for the LLM-based approach. However, the LX exhibits a higher frequency of extreme positive outliers, with some documents containing close to 50 extracted references. In contrast, the LLM-based approach shows a maximum of slightly above 20 references per document. This suggests that LX is more exhaustive in its extraction process, while the LLM-based approach appears more conservative. These differences are visually represented in Figures 5.4 and 5.6, where the LX distribution shows a longer tail towards higher reference counts.

5.3.2 Distribution Across Categories

Analyzing the category-wise distribution of extracted references, the two approaches reveal similar but distinct patterns. The LLM-based approach extracts a higher proportion of legal references categorized as *wet* (laws), indicating a stronger focus on legislative references. Conversely, LX consistently extracts more references in the categories of *officiële publicatie*, *uitspraak*, and *europese-regelgeving*, demonstrating a broader coverage across different types of legal documents. If the categorization of laws by the LLM is accurate, this suggests that it tends to focus more heavily on laws while underperforming in identifying other categories compared to LX. Table 5.2 summarizes what can be found in Figures 5.5 and 5.7, illustrating these differences effectively.

Table 5.2: Comparison of legal reference counts and proportions between LX and LLM across different categories

Category	**LX count**	**LLM count**	**LX prop.**	**LLM prop.**
wet	3825	3376	0.836	0.928
officiële publicatie	480	223	0.105	0.061
uitspraak	264	34	0.058	0.009
europese-regelgeving	8	4	0.002	0.001

5.3.3 Document Length and Extraction Trends

To better understand the relationship between the number of extracted references and the length of the original document, a line plot was produced. This plot compares the number of references extracted by both approaches while overlaying the (scaled) length of the document (Fig. 5.8).
The number of extracted references from both approaches shows a positive correlation with document length, reflecting the natural relationship where longer texts tend to contain more legal references. However, closer inspection reveals that the two approaches are more strongly correlated with each other than with document length alone[6]. This alignment—validated by correlation calculations[7]—suggests that, despite their differing methodologies, both systems are

[6]Further data exploration revealed that this correlation appears to hold until documents reach around 2000 words in length, after which it declines (with the LLM struggling to extract all legal references, while the LX remains unaffected). Figure A.5.1.2, illustrating this, can be found in the appendix.

[7]Correlation between LX reference count and LLM reference count: 0.62; correlation between LX reference count and document length: 0.55; correlation between LLM reference count and document length: 0.29

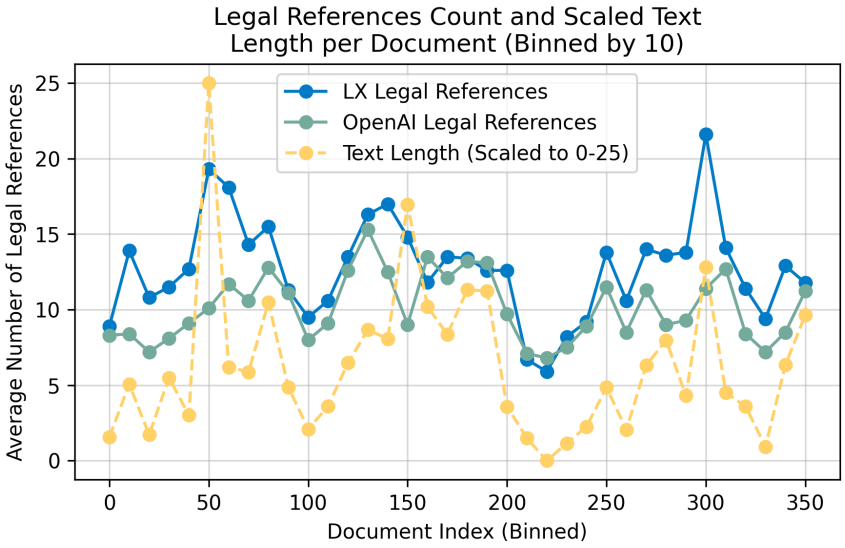


Figure 5.8: Comparison of the average number of extracted legal references by LX and OpenAI across binned document lengths, with scaled document length included

sensitive to similar textual cues indicative of legal references. This also raises an interesting question: is the LLM-based approach approximating the rule-based logic embedded in LX, or are both systems reacting to universal patterns commonly found in legal texts?
Despite this apparent alignment, differences remain noticeable. LX consistently extracts more references across most document bins, indicating a more exhaustive approach—especially as document length increases (see Fig. A.5.1.2 in the appendix). By contrast, the LLM-based method might focus more on salient references, potentially missing less obvious or borderline cases. These differences could become more pronounced in longer or structurally complex documents, where variations in citation patterns, textual density, and reference styles pose additional challenges. Furthermore, the relationship between document complexity and extraction performance deserves deeper analysis, as it might reveal whether one approach is inherently better suited for specific document types.

5.3.4 Evaluation and Validation

A critical difference between the two approaches lies in their evaluation and validation processes. The LX has been extensively evaluated and fine-tuned by its developers, ensuring its robustness and correctness. In contrast, the LLM-based approach lacks formal validation, as such evaluation requires careful human labeling by experts capable of accurately annotating a corpus of legal decisions. This evaluation process is inherently complex, as it demands expertise in both legal interpretation and technical assessment.
Nonetheless, the strong observed correlation between the two approaches lends credibility to the LLM-based method. Given LX's established reliability, here it will serve as a valuable benchmark for assessing the performance of the LLM model. This benchmarking, combined with reverse-lookup verification and similarity scoring, provides initial confidence in the LLM-based system's outputs. However, further validation—especially through manual annotation and comparison with authoritative legal datasets—would be highly valuable for drawing final conclusions about its accuracy and reliability.

5.3.5 Overlapping and Divergent Outputs

The comparison between the LX and LLM-based approaches raises a fundamental question: does the LLM-based system merely extract a subset of the references identified by LX, or does

it operate with a degree of independence in its extraction logic? Two possible hypotheses emerge from this consideration:

1. **Subset Hypothesis:** the LLM extracts only a subset of the references identified by LX, potentially making it useful in domains or jurisdictions where LX is not applicable (e.g., outside Dutch law).

2. **Independent Extraction Logic:** if the overlap is not near-perfect, then understanding where the two approaches agree or disagree becomes essential.

Analyzing overlaps and divergences will clarify whether LLM-based extraction can serve as a viable lightweight alternative to rule-based systems like the LX or if the effort required to build a robust, highly structured, rule-based system remains ultimately unavoidable for achieving optimal extraction performance. If the LLM-based approach demonstrates sufficient accuracy and reliability with minimal configuration, it could offer a more adaptable and resource-efficient solution in contexts where LX is not applicable (e.g. other jurisdictions). Conversely, if significant limitations persist, the results may reinforce the necessity of highly engineered, domain-specific rule-based systems to ensure comprehensive and precise legal reference extraction.

5.3.6 Output Analysis

To reduce the impact of small textual variations, all extracted legal references were normalized by converting them to lowercase and removing extra spaces. This step ensured that minor inconsistencies, such as differing capitalizations or irregular spacing, did not artificially inflate mismatches between the two systems. Once normalized, matches were analyzed within each decision, distinguishing references matched only by LX, those matched only by the OpenAI LLM, and those identified by both systems. A summary can be found in Table 5.3.

Table 5.3: Summary of intra-decision matches and unmatched references

	Matches	**LX only**	**OpenAI only**
Mean	4.16	8.59	5.97
Std	3.30	5.73	3.99
Min	0.00	0.00	0.00
25%	2.00	4.00	3.00
50%	4.00	8.00	5.00
75%	6.00	11.00	9.00
Max	14.00	42.00	18.00

When (non-)matches were aggregated across all decisions, overarching trends began to emerge. The analysis revealed that LX identified a total of 587 legal references, while OpenAI's GPT extracted 609. Of these, 232 references were common to both systems, while 355 were unique to LX and 377 were unique to OpenAI. These results can be visualized using a Venn diagram (Fig. 5.9), which offers an intuitive representation of the areas of overlap and divergence between the two systems. While the diagram indicates a significant overlap, the set differences—references identified by one system but not the other—are substantial, suggesting meaningful discrepancies in how the two approaches operated. At first glance, one might be tempted to conclude that the LLM extracts more references overall, as suggested by the larger set difference in the Venn diagram. However, this observation warrants deeper scrutiny. A closer examination of the individual elements in each segment of the diagram can provide better insights into the nature of these references and the reliability of this representation. While the diagram highlights differences, understanding the granularity and overlap of the extracted references is essential for a more accurate interpretation.
To verify whether the set differences between LX and LLM are genuine or merely artifacts of minor variations in textual representation, a Cartesian product of the set differences across all

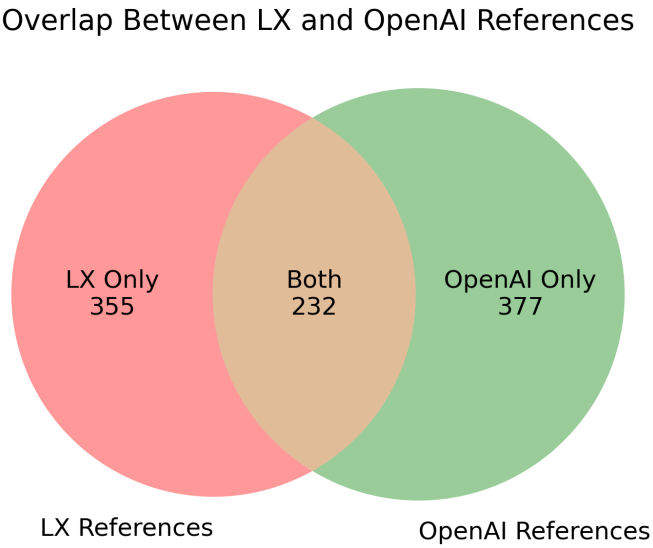


Figure 5.9: Venn diagram illustrating the overlap and differences between legal references extracted by LX and OpenAI

decisions was performed. For each pair of references in the set differences, similarity scores were calculated and ranked in descending order.

The analysis revealed an interesting pattern: while most highly similar references corresponded to genuinely distinct legal citations—such as *artikel 95b* versus *artikel 95* from the same law—there were also cases where both systems identified the same overarching legal framework (e.g., *Elektriciteitswet 1998*) but emphasized different levels of granularity. The LLM system often extracted broader references (e.g., *artikel 95*), whereas LX tended to focus on more specific clauses (e.g., *artikel 95b*).

Interestingly, despite the LLM prompt explicitly including examples with sub-articles and clauses, these mismatches persisted. This raises the question of whether the discrepancy stems from how the LLM processes or prioritizes textual patterns. If so, it might be that a minor adjustment to the extraction prompt could mitigate these inconsistencies in subsequent API calls, but further research is required to ascertain this. In any case, this observation highlights a nuanced limitation in the current comparison: while both systems demonstrate sensitivity to the same legal frameworks, their granular focus can diverge.

While this finding does not undermine the overall analysis, it adds a layer of complexity to interpreting the differences between the two systems. It suggests that both systems share an underlying sensitivity to identifying references within the same legal context, even if their focus diverges at a more granular level. These subtle variations should be carefully considered when assessing whether the two approaches are complementary or redundant. Consequently, the Venn diagram remains a reliable reflection of the relationship between the two systems. However, the interpretation of the set differences requires nuance. The LX system's set difference appears to represent a more precise reflection of the underlying legal references, as it includes more granular citations. In contrast, the OpenAI API's set difference seems to over-represent broader references, potentially inflating the perceived divergence between the two systems. This suggests that LX's approach might provide a more realistic representation of the differences in legal reference extraction.

Building upon this verification, the analysis then turned to identifying the most frequently recurring references within each set difference. This exploration revealed meaningful patterns. For example, LX consistently missed references to *Gaswet* articles, likely because these laws were not included in its predefined database. This underscores a fundamental limitation of the LX: its reliance on a more static set of rules and laws, constraining its ability to adapt to less common or newly introduced legal references.

Table 5.4: Similarity across most similar legal references from the set differences (red indicates the differences, green matching references)

LX Reference	**OpenAI Reference**	**Score**
artikel 95b, eerste lid, van de elektriciteitswet 1998	*artikel 95, eerste lid, van de elektriciteitswet 1998*	99.065
artikel 5, eerste lid, elektriciteitswet 1998	*artikel 45, eerste lid, elektriciteitswet 1998*	98.901
artikel 2, lid 2 sub g van het besluit	*artikel 2, lid 2, sub g van het besluit*	98.701
artikel 3 van de elektriciteitswet	*artikel 43 van de elektriciteitswet*	98.551
artikel 12r, eerste lid, instellingswet autoriteit consument en markt	*artikel 12w, eerste lid, instellingswet autoriteit consument en markt*	98.551
artikel 95a, eerste lid van de elektriciteitswet 1998	*artikel 95, eerste lid, van de elektriciteitswet 1998*	98.113
artikel 95d, eerste lid van de elektriciteitswet 1998	*artikel 95, eerste lid, van de elektriciteitswet 1998*	98.113
artikel 45, eerste lid, van de elektriciteitswet 1998	*artikel 95, eerste lid, van de elektriciteitswet 1998*	98.113

The comparison also examined how consistently each system identified the other's most frequently recurring references. The results revealed an asymmetry: LX reliably identified most of OpenAI's recurring references, while the LLM was notably less consistent in detecting LX's most frequently recurring citations. This pattern suggests that LX demonstrates greater stability and consistency in its extraction logic, likely due to its structured, rule-based architecture. In contrast, the LLM approach, driven by probabilistic language modeling, exhibits greater variability in its results, even when presented with frequently recurring patterns. Another noteworthy observation was the LX's extraction of meaningless references, such as *spencer* or *ervan*. These instances suggest the identification of local aliases (see A.5.2 for an explanation of what these are), particularly in the case of *ervan*, which in Dutch can mean *from/of that*, functioning as a the English possessive pronoun (*whose*). While this *issue* was not thoroughly discussed with the LX developers, it indicates that such cases should either be addressed by retrieving the legal reference under these local aliases or excluded altogether in applications where the focus is on a simpler and more straightforward legal reference extraction. An illustration of this comparison can be found in Fig. 5.10.

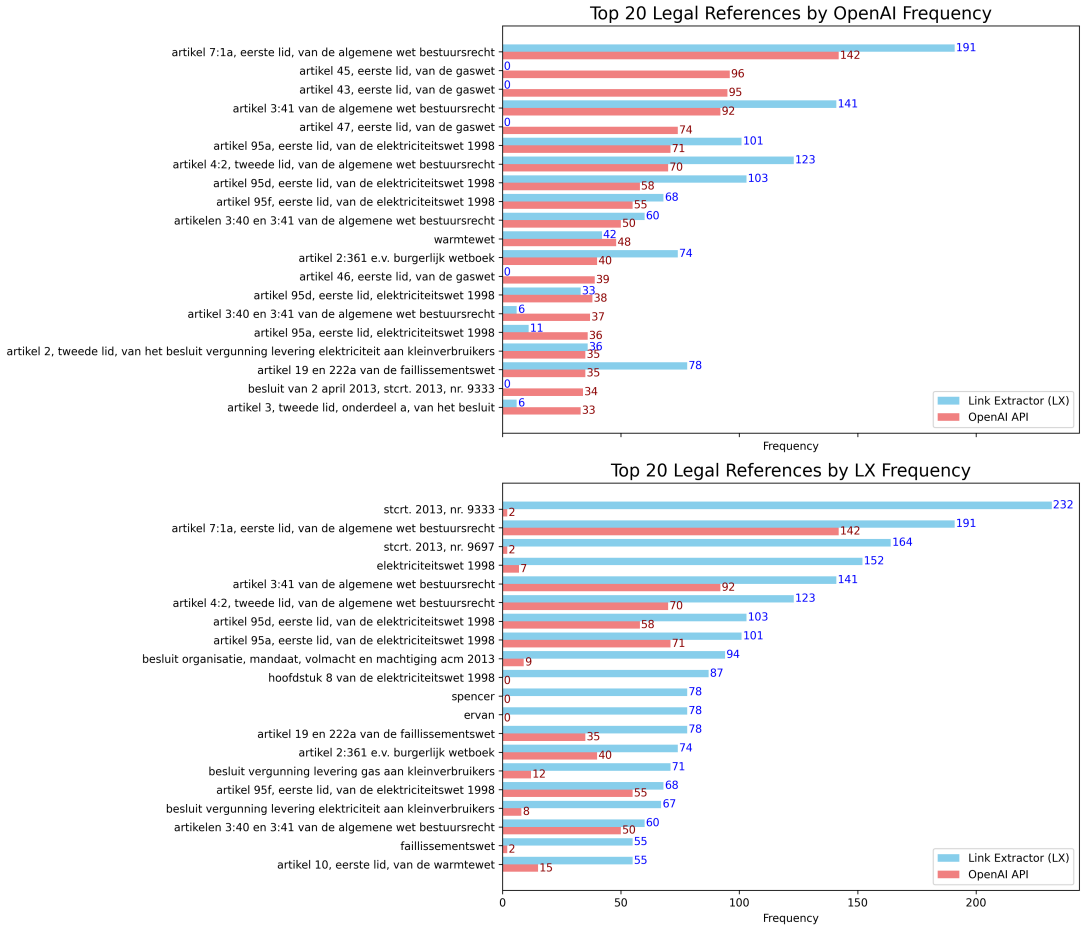


Figure 5.10: Comparison of the 20 most extracted legal references by each approach

5.4 Conclusion

The analysis reveals that while there seems to be significant overlap between the two systems, up to a certain level of granularity, the results do not fully support the *subset hypothesis*. The LLM does not merely extract a strict subset of LX references; instead, it occasionally identifies references missed by LX, particularly those outside LX's predefined database or rigid extraction rules. This observation supports the *independent extraction logic hypothesis*, suggesting that the LLM operates with a degree of independence in its extraction patterns.

In conclusion, the LX system proves to be more robust and reliable than the LLM-based approach. Its design prioritizes minimizing false positives [35], and while this experiment does not explicitly evaluate false positives or false negatives, LX's consistent performance suggests a carefully optimized system. However, LX's reliance on a predefined database and strict extraction rules also introduces limitations, particularly when encountering references outside its defined scope (which need to be manually added for the system to recognize them). The LLM, on the other hand, exhibits adaptability and flexibility but struggles with stability and consistency, especially when identifying frequently recurring references that LX reliably detects.

These findings suggest that an LLM may not yet serve as a standalone legal reference extraction tool but could complement the LX effectively. The LLM's strength lies in its ability to identify references overlooked by the LX, particularly those beyond the scope of predefined rules or datasets. While it is possible to refine the LLM's performance through improved prompting or domain-specific fine-tuning, achieving LX's level of precision would likely require substantial research and effort.

The most promising path forward probably lies in exploring a hybrid approach, where the LX's

structured and database-driven reliability is paired with an LLM's adaptability and flexibility. Such a combined system could leverage the strengths of both approaches, mitigating their individual weaknesses to achieve more comprehensive and accurate legal reference extraction. However, in scenarios where the LX—or a similar highly engineered tool—is unavailable, further exploration of an LLM-based approach remains worthwhile. Despite its limitations in granularity and stability, an LLM can still provide a baseline for legal reference extraction, offering a valuable starting point for comparative studies of legal decisions, even if it does not fully match the precision of LX.

Chapter 6

Legal Effect Classification

The Legal Effect, as the outcome of a decision, is not a piece of metadata that can be *extracted* from a decision's text, as it is not mentioned in a standardized form (e.g., "The legal effect of this decision is...") but must instead be inferred. For example, a decision stating, *"permit XYZ granted to company ABC"*, indicates a permit grant, but this could also appear in another decision's headline as *"ZYX permit grant, company CBA"*. This is not an information extraction task in the traditional sense of identifying a specific entity among many.
Therefore, the goal of this chapter has been reframed as a classification task, where NLP techniques are used to classify permits according to a predefined set of possible outcomes (i.e. legal effects). The categories used, along with their distribution, are presented in Table 6.1.

Table 6.1: Initial distribution of the (manually labeled) legal effects

Decision Category	**Count**	**Proportion**
Verguning verlening (Grant)	146	0.41
Wijziging of aanpassen (Amend/Modify)	102	0.28
Intrekking of beëindiging (Revoke/Withdraw)	64	0.18
None	36	0.10
Overdracht (Transfer)	11	0.03

The *Transfer* category, representing only 3% of the dataset, was removed as its limited representation reduces the overall classification performance and interpretability. Then, to mitigate the issue of multiple entries with identical id's—arising from almost identical legal decisions (e.g. grant of a gas and an electricity permit simultaneously to the same party)—it was decided to retain only the first occurrence of each unique identifier. This prevents data leakage during train-test splits. The refined category distribution is presented in Table 6.2.

Table 6.2: Refined distribution of legal effect categories

Decision Category	**Count**	**Proportion**
Verguning verlening (Grant)	136	0.49
Wijziging of aanpassen (Amend/Modify)	83	0.30
Intrekking of beëindiging (Revoke/Withdraw)	42	0.15
None	18	0.06

This chapter explores and compares two approaches to classifying legal decisions. The first approach employs traditional ML classifiers, similar to those used in Chapter 4. ML models are well-suited for classification tasks, particularly when labeled data is available. The second approach leverages an LLM-based method, building on the methodologies introduced in Chapter 3.

6.1 ML Based Legal Effect Classification

Data Imbalance and Splits

Unlike the *zero-shot*-LLM-based approach, supervised ML models require a train-test split to learn from the data and evaluate their performance. Given the imbalance in the dataset, particularly the under-representation of the *None* category (6% of the total instances), a stratified split was performed. This ensures that the class distribution in both training and test sets remains representative of the overall dataset. Furthermore, as previously done in the Date Extraction chapter (Ch. 4), the split was first performed on unique decision IDs. This approach prevents data leakage caused by having slightly modified versions of the same decision appear in both training and test sets. The resulting split produced 167 instances in the training set and 112 in the test set. The class proportions in each set are shown in Table 6.3.

Table 6.3: Class distribution in training and test sets (proportions)

Decision Category	**Train**	**Test**
Verguning verlening (Grant)	0.49	0.49
Wijziging of aanpassen (Amend/Modify)	0.30	0.29
Intrekking of beëindiging (Revoke/Withdraw)	0.15	0.15
None	0.07	0.06
Total Instances	167	112

Model Choice

The same pipeline used in Chapter 4 (Sec. 4.2.1) was utilized here to identify the best combination of vectorizer and ML model. In light of the previous results, only the Naive Bayes (Multinomial and Complement), Logistic Regression, and Random Forest algorithms were considered. Similar to the LLM-based classification, a distinction was made between using only the headline and description versus including the full decision text too. In total, 216 dataset-vectorizer-model combinations were assessed, and the results of the best-performing models (based on the macro-average F1-score, which gives equal weight to each class's F1-score regardless of their support) are presented in Table A.6.1.6 in the Appendix. Once again, Logistic Regression emerged as the best-performing model—though with a binary 1-gram BoW vectorizer, differing from the TF-IDF vectorizer that was optimal in the date extraction task—achieving a macro-average F1-score of 97.2%.

Fine Tuning and Evaluation

The model was further fine-tuned using 5-fold cross-validation, exploring different regularization strengths, solvers, and penalties. The optimal parameters were found to be: C=100, Lasso penalty (ℓ_1), and *saga* solver. The final classification report for the model with the best set of hyperparameters (and using the full text), presented in Table 6.4, highlights consistently high precision and recall across all categories except one—the minority class *None*, though easily predictable. The improvements are also reflected in the confusion matrix (Fig. 6.1), showing minimal misclassifications: only three out of the seven instances in the *None* category and one out of thirty-three instances in the *Wijziging of aanpassen* category.

Feature Importance

Analysis The feature importance analysis highlights how the model distinguishes between legal effect categories based on the presence and absence of specific terms, effectively capturing class boundaries through term presence/absence patterns. This is particularly evident in the categories Vergunning verlening, Wijziging of aanpassen, and Intrekking of beëindiging, where

Table 6.4: Classification report for fine-tuned logistic regression model

Category	Precision	Recall	F1-Score	Support
Intrekking of beëindiging	1.00	1.00	1.00	17
None	0.80	0.57	0.67	7
Verguning verlening	0.96	1.00	0.98	55
Wijziging of aanpassen	0.97	0.97	0.97	33
Accuracy			0.96	112
Macro Avg	0.93	0.88	0.90	112
Weighted Avg	0.96	0.96	0.96	112

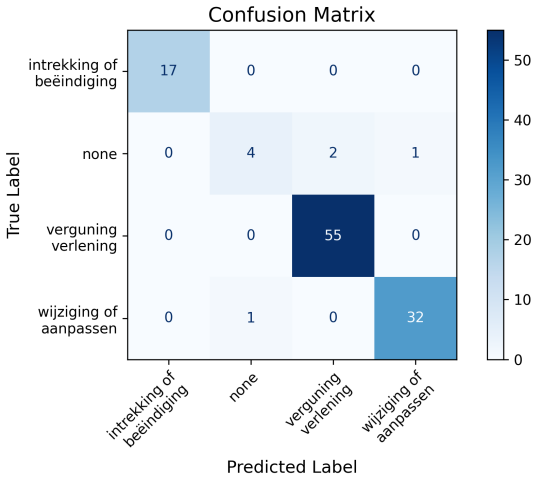


Figure 6.1: Confusion matrix for logistic regression legal effect classifier

terms strongly associated with one category are unlikely to appear in another. This juxtaposition effect suggests that the model not only identifies category-specific keywords but also relies on their absence as informative signals.

In the **Vergunning verlening** category, positive coefficients highlight terms like *ontbrekende* (missing), *administratie* (administration), and *geschillencommissie* (disputes committee). While these terms are often associated with procedural or administrative contexts, it is not entirely clear why they are so predictive for this category, suggesting that their importance might stem from recurring patterns or specific phrasing in permit-related documents. Conversely, words like *wijzigen* (to amend) and *wijziging* (amendment) have negative coefficients, indicating their weak association with permit grant decisions.

In the **Wijziging of aanpassen** category, the strongest positive features include *wijzigen* (to amend) and *wijziging* (amendment), which clearly signal modifications to an existing permit or regulation. Additionally, terms like *volgt* (follows) appear frequently, although their connection to amendments is less explicit (again, their importance may arise from recurring patterns in the structure or phrasing of amendment-related documents). On the other hand, negative coefficients highlight terms such as *intrekken* (to withdraw) and *onderdeel* (component). While *intrekken* strongly indicates withdrawal or termination, the negative association of *onderdeel* suggests that references to parts or components of a decision are more characteristic of other categories. This emphasizes the model's ability to identify patterns not just from individual terms, but also from their typical contextual associations across categories.

For the **Intrekking of beëindiging** category, positive coefficients emphasize terms like *ingetrokken*, *intrekken*, and *trekken*. These words are clear indicators of withdrawal or termina-

tion decisions, directly reflecting the legal language used in such contexts. In contrast, negative coefficients include *wijzigen* and *hoofdstuk* (chapter), terms more commonly associated with amendment or structural contexts. This distinction highlights the model's ability to effectively differentiate between language indicative of termination and that related to modifications.
In the **None** category, positive coefficients are dominated by terms like *2019*, *hierbij* (hierby), and *gas* (gas). These words suggest the absence of a specific legal context, reflecting the composition of the category, which mainly consists of administrative notices, announcements, or non-decision documents. Negative coefficients include terms such as *minister* and *nr* (an abbreviation for "number"), which typically precede decision numbers in legal decisions. Given the nature of the *None* category, here, the model appears to rely on temporal and contextual terms to identify these documents.

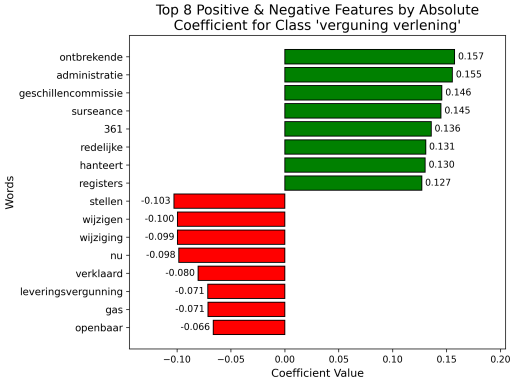


Figure 6.2: Largest/Smallest 8 coefficients for *verguning verlening* category

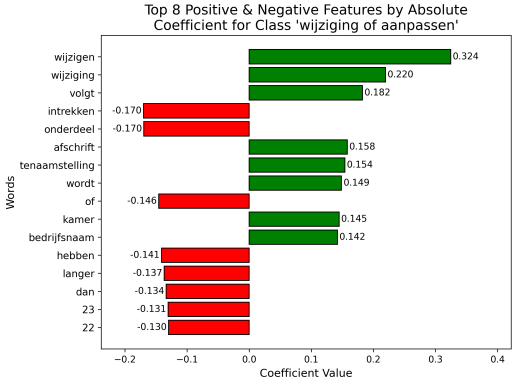


Figure 6.3: Largest/Smallest 8 coefficients for *wijziging of aanpassen* category

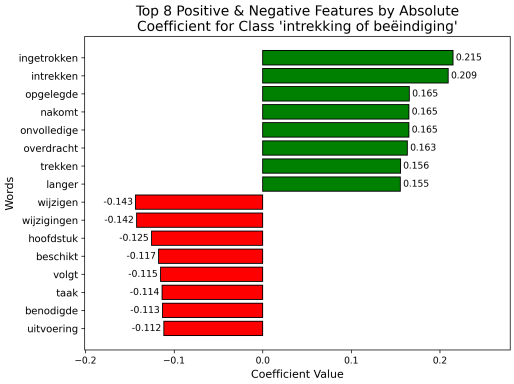


Figure 6.4: Largest/Smallest 8 coefficients for *intrekking of beëindiging* category

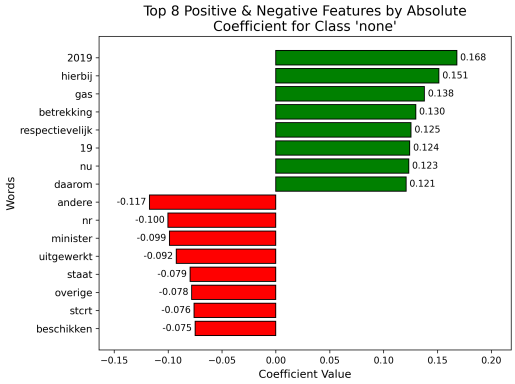


Figure 6.5: Largest/Smallest 8 coefficients for *None* category

6.2 LLM-based Classification

The overall approach closely followed the methodology used in the LLM implementations from the previous chapters, with the temperature set to 0, an English prompt, and `gpt-4o-mini`. Two prompts were used to test the model with both shorter and longer input texts. The system prompt[1] remained consistent across both implementations, defining the model's role and overarching objective for the task.
In one case, the model received only the combined decision headline and description, along with instructions to return a category from the predefined set (hereinafter referred to as *Prompt 1*). In the other case, the full text of the decision was also included in the prompt (hereinafter referred to as *Prompt 2*). For brevity, only Prompt 2 is presented below.

Legal Effect Classification Prompt

Given the following headline, description and text of an administrative decision, categorize it into one of the following 4 categories:

- "verguning verlening": decisions to grant a license
- "wijziging of aanpassen": decisions to amend/modify a license
- "intrekking of beëindiging": decisions to revoke/withdraw a license
- "none": if none of the previous categories is relevant

Guidelines for Categorization:

1. You may assign ONLY ONE category, if applicable.
2. If no category is relevant, return nothing.

Input:

- Headline and description: `decision_headline_description`
- Text: `decision_text`

Output Format:

- If a category is identified, return a string in this format: ''`assigned_category`''
- If no category is identified, return string like this: ''`none`''

Results

The results are summarized in Table 6.5. In the first approach, the model received the combined decision headline and description, while in the second approach, the full text of the decision was also included.

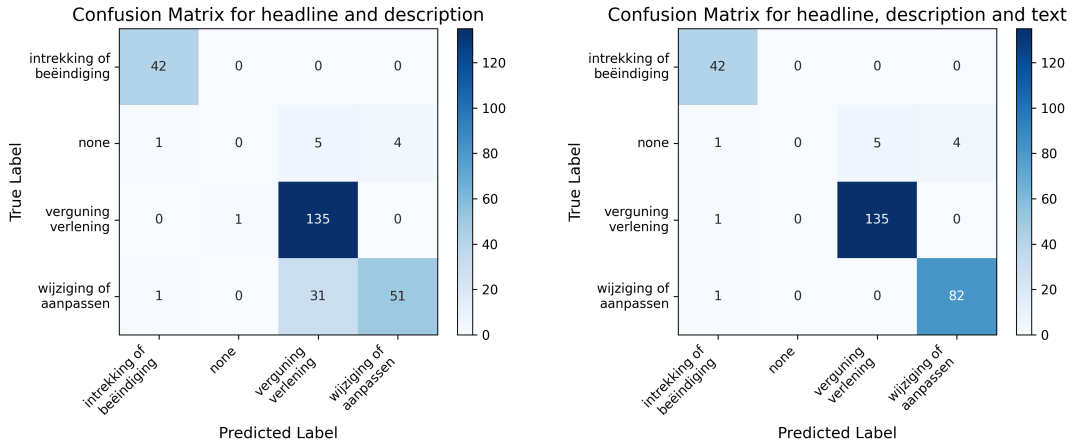
Table 6.5: Prediction distribution across two prompting approaches: headline and description (Prompt 1) or headline, description, and full text (Prompt 2)

Decision Category	Prompt 1	Prompt 2	G. Truth
Vergunning verlening (Grant)	171	140	136
Wijziging of aanpassen (Amend/Modify)	55	86	83
Intrekking of beëindiging (Revoke/Withdraw)	44	50	42
None	1	0	18
`Error: Expecting value: line 1 column 1 (char 0)`	8	3	-

The inclusion of the full text in Prompt 2 resulted in a noticeable shift in predictions, with fewer classifications as *Verguning verlening* and more as *Wijziging of aanpassen*. The occurrence

[1] *"You are an assistant for classifying legal decisions into 4 categories."*

of `[Error: Expecting value: line 1 column 1 (char 0)]` indicates issues in parsing or interpreting certain prompts, likely stemming from unexpected model outputs or edge-case scenarios in the input structure. This error frequency decreased in the second approach, although its occurrence should not be related to the prompt used.
Confusion matrices (Fig. 6.6) revealed distinct misclassification patterns between the two prompting approaches. The first approach, relying only on the headline and description, frequently misclassified *Wijziging of aanpassen* decisions as *Verguning verlening*. This suggests that the limited context provided in the headline and description is often insufficient to capture the nuanced differences between these categories, particularly in scenarios involving company name changes (*Naamswijziging*), where permits are simultaneously revoked and re-granted under a new name. Labeling such instances as a separate category might address this ambiguity and improve classification performance. Additionally, both approaches consistently struggled with the *None* category, which serves as a "catch-all" for announcements or other non-decision documents. The rarity and ambiguity of such cases in the dataset likely contributed to the model's poor performance in predicting them. Despite these challenges, the second approach, which included the full text, demonstrated improved differentiation between similar categories and reduced misclassifications overall. This improvement is reflected in the classification reports (Tables 6.6 and 6.7), where higher precision and recall scores were observed across most categories.



(a) Prompt with headline and description

(b) Prompt with headline, description, and text

Figure 6.6: Confusion matrices for LLM-based classifiers.

Table 6.6: Classification Report for prompt 1

Category	Precision	Recall	F1-Score	Support
Intrekking of beëindiging	0.95	1.00	0.98	42
None	0.00	0.00	0.00	10
Verguning verlening	0.79	0.99	0.88	136
Wijziging of aanpassen	0.93	0.61	0.74	83
Accuracy			0.84	271
Macro Avg	0.67	0.65	**0.65**	271
Weighted Avg	0.83	0.84	0.82	271

Table 6.7: Classification Report for prompt 2

Category	Precision	Recall	F1-Score	Support
Intrekking of beëindiging	0.93	1.00	0.97	42
None	0.00	0.00	0.00	10
Verguning verlening	0.96	0.99	0.98	136
Wijziging of aanpassen	0.95	0.99	0.97	83
Accuracy			0.96	271
Macro Avg	0.71	0.75	**0.73**	271
Weighted Avg	0.92	0.96	0.94	271

6.3 Comparison

Both the LLM-based and ML-based approaches demonstrated that larger textual input, including the full decision text, significantly improves classification performance. Headline and description alone did not provide sufficient context for the models to effectively distinguish between nuanced legal effect categories, underscoring the importance of richer textual information in this classification task.
Despite the lack of supervised training, the LLM-based approach achieved a macro-average F1-score of 0.73. While this result is commendable for a *zero-shot* approach, it remains insufficient for real-world applications, particularly in scenarios where additional legal effect categories could introduce further ambiguity. In contrast, the ML-based approach, trained on only 60% of the data, achieved superior classification performance. Although this split reduced the number of training instances per category, the model effectively handled the class imbalance and learned meaningful term-category associations—reaching a macro-average F1-score of 0.90. Inspecting the feature coefficients revealed the model's ability to identify terms strongly tied to specific legal effects while recognizing the absence of these terms as informative signals. This juxtaposition effect highlights the model's capacity to handle nuanced semantic boundaries, even with a relatively simple BoW representation.

	Macro-Average-F1-score
Fine-Tuned ML	0.90
LLM (Prompt 1)	0.65
LLM (Prompt 2)	0.73

Table 6.8: Comparison of ML-Based and LLM-Based Classification Approaches for Legal Effects

6.4 Conclusion

This experiment demonstrated that while LLMs offer significant flexibility, they may lack the precision required to accurately classify legal outcomes into predefined categories. In contrast, traditional machine learning, combined with longer input texts and straightforward feature engineering through text vectorization, appears to capture the nuances of each category effectively, resulting in superior performance in this context.
Future research should explore whether these findings hold true when the number of categories increases. Expanding the classification task to include more categories would introduce greater complexity, likely reducing performance for both approaches. This added complexity stems from increased semantic overlap between categories and the heightened potential for ambiguous classifications. Investigating whether the performance gap between ML and LLM approaches remains consistent, or whether one approach degrades more significantly, could provide valuable insights into their scalability and real-world applicability.

Additionally, traditional ML approaches could benefit from integrating more advanced text representations, such as word embeddings, which offer richer contextual understanding compared to simpler vectorization methods. This enhancement could address current limitations in representing word meanings. For LLM-based classification, experimenting with techniques like *few-shot* learning and other fine-tuning options may help narrow the performance gap, potentially making this approach more competitive and better suited for scenarios where annotated data is scarce.

Chapter 7

Conclusion

7.1 How NLP Techniques Can be Applied Extract Key Metadata from Dutch Administrative Decisions Issued by the ACM

Summary of Approaches and Results

Throughout this research, various supervised IE approaches were employed, each specifically tailored to the unique characteristics and challenges of the information being extracted. In parallel, an equivalent unsupervised approach leveraging an LLM was implemented for each task, allowing for comparative analysis with the supervised approach.
Extracting recipients (Chapter 3) proved highly challenging due to the significant variability in company names and their lengths, necessitating adjustments in the training and evaluation of a custom supervised NER model, primarily through fuzzy matching techniques. The resulting model achieved an F1-score of 85%, with 93% precision and 78% recall. However, it is acknowledged that these metrics may have been overestimated due to the presence of named entities in both the training and testing datasets, as they occasionally appeared in different decisions a few weeks apart. In contrast, the LLM-based NER approach achieved an F1-score of 88%, with 84% precision and 92% recall.
A supervised NER model was also employed as the basis for identifying date entities in decisions (Chapter 4), which proved highly successful due to the limited variation in date formatting and the fine-tuning of the baseline NER model, making identification relatively straightforward. However, to specifically focus on the decision date among the many mentioned within a document, all extracted dates were classified using a supervised ML classifier based on word spans surrounding each date. The high performance of the date classification algorithm was reflected in an F1-score of 97%, with both precision and recall at 97%. In contrast, extracting dates using an LLM resulted in an F1-score of 87%, with precision at 86% and recall at 87%.
To predict the legal effect of a decision (Chapter 6), a set of possible legal effects for the given dataset was defined *a priori* using domain knowledge. Each decision was then manually labeled to generate targets suitable for a supervised machine learning classification setting. Legal effects were predicted using the full decision text as input, resulting in a macro-average F1-score of 90%. In contrast, LLM-based legal effect classification achieved a maximum macro-average F1-score of 73% for the same task.
In addition to extracting (or determining, in the case of the legal effect) the three aforementioned key elements of administrative decisions, a fourth, slightly different experiment was conducted to extract legal references from administrative decisions (Chapter 5). Due to the lack of labeled data (i.e., annotated legal references within a decision) and the high cost of obtaining such labels, a supervised approach was not implemented. Instead, a highly engineered, rule-based approach developed by KOOP—the *LinkExtractor*—was utilized. While this approach does not fall under the umbrella of supervised ML, it shares a significant similarity: the requirement for substantial time and monetary investment before implementation. Specifically, it necessitates the creation

of a detailed and extensive set of rules, which faces comparable challenges as generating labeled data for supervised learning, including time, expertise, and cost. This implementation was compared to an LLM-based approach for legal reference extraction to evaluate whether the results would be comparable to those of the carefully designed and accurate *LinkExtractor*. The results demonstrated that the two approaches produced overlapping outcomes up to a certain level of granularity. However, the LLM often failed to capture finer details in legal references when compared to the rule-based system. Conversely, the LX exhibited an expected disadvantage: its reliance on predefined rules limited its ability to identify pieces of legislation not included in its rule/knowledge dataset.

Comparing Supervised and Unsupervised Approaches

NLP techniques can be employed to extract key metadata from Dutch administrative decisions, as it showed from this research on energy permits issued by the *Autoriteit Consument & Markt*. This research demonstrates that while supervised ML models generally achieve better predictive performance for structured tasks like date extraction and legal effect classification, their success is highly subject to the availability of specific, high-quality training data. For example, the supervised approaches outperformed LLM-based methods in date extraction and legal effect classification due to their fine-tuning on reliably annotated dataset. However, in more complex scenarios, such as recipient extraction, where variability in entity names posed a significant challenge, LLM-based NER proved more suited at capturing this complexity.
The rule-based *LinkExtractor* showcased a competitive edge over LLMs in legal reference extraction, leveraging its ability to process longer documents and its granular focus on legal details. Nonetheless, its reliance on predefined rules limited its adaptability to references that were not accounted for in the initial rule set. This limitation can hinder practical applications in dynamic legal contexts where new pieces of legislation or types of references emerge frequently, requiring ongoing updates to maintain system effectiveness. Conversely, while LLMs provided broader adaptability and cross-context application, their performance lagged in terms of precision and granularity (which are crucial in the legal domain when it comes to identifying specific articles). This gap could potentially be narrowed with future research into fine-tuning and advanced prompting strategies.
Another key distinction between the compared approaches lies in their computational demands and scalability. Once trained, supervised ML models are lightweight and efficient, delivering fast and reliable predictions with minimal resource requirements. In contrast, LLMs, particularly when accessed through outsourced APIs like OpenAI's, offer high-speed performance but incur significant operational costs. In-house deployment of LLMs, while reducing reliance on external providers, demands substantial computational resources and infrastructure investment, making scalability a complex consideration [24].
The adaptability of LLMs to diverse datasets and multilingual contexts without retraining is a significant advantage over supervised ML models, which can struggle in dynamic or varied environments without domain-specific retraining. However, this adaptability is not without challenges. LLMs require continuous oversight to ensure output reliability and to mitigate risks like hallucinations—issues that were repeatedly observed during the experiments. Hallucinated outputs can severely impact administrative metadata extraction by introducing incorrect dates, entities, or references, thereby undermining trust and the utility of the data. Addressing this requires thoughtful *ex-ante* design and strategic choices, including extensive research and experimentation on implementing robust prompting strategies or fine-tuning a model. Post-output, rigorous validation mechanisms or hybrid approaches, like combining LLMs with rule-based or supervised systems (e.g., reverse text lookups or supervised models trained to differentiate between hallucinated and reliable outputs), can help detect and correct errors. However, these measures involve substantial overhead, which could limit the appeal of straightforward LLM implementations. In contrast, supervised models often provide more reliable or deterministic behavior, such as with logistic regression or decision trees, producing consistent outputs for identical inputs and reducing the need for extensive post-deployment oversight. Additionally, supervised ML offers transparency by enabling insights into the features used in predictions, an essential aspect in the legal domain. On the other hand, the opaque, black-box nature of LLMs

makes it nearly impossible to trace back errors or understand the rationale behind specific outputs.

7.2 Overcoming the Lack of Labeled Data Using Large Language Models

The experiments conducted in this research demonstrate that it is possible to overcome the lack of labeled data in information extraction for Dutch administrative decisions using LLMs. Often, this approach yields impressive results given the context and domain specificity of administrative decisions. However, a significant performance gap remains when comparing LLMs to simpler but carefully fine-tuned supervised ML algorithms designed for specific tasks. While LLMs demonstrate satisfactory results, they still exhibit notable limitations, including reduced predictive power, a lack of explainability, and the need for constant oversight—an oversight that is less intensive for supervised ML systems.

A key consideration is that, in the current landscape, where administrative decisions often lack sufficient metadata and exhibit substantial variability, the effort required to label data for supervised training might outweigh the predictive performance benefits. In such scenarios, the trade-off of accepting a moderate (but possibly tolerable, depending on the application) loss in predictive accuracy with LLMs could be justified, especially if resources are instead directed toward oversight and gradual refinement of the LLM. Over time, this iterative improvement process could mitigate many of the current limitations (e.g., better prompting, fine-tuning...), enhancing reliability and consistency. Moreover, LLMs offer broader applicability across authorities with differing standards and even across jurisdictions, aligning well with large-scale transparency initiatives, such as those that might be implemented at the EU level.

Other available tools, such as *LexNLP*, currently fail to deliver such flexibility and applicability due to language limitations and a lack of recent updates—both of which are crucial for effective legal reference recognition and adaptation to evolving styles or contexts.[1] Achieving this level of cross-context adaptability is nearly impossible with simpler, highly domain-specific supervised ML systems unless constant updates and adjustments are made to prevent performance degradation over time.

To address the issue of external validity in supervised settings, one potential approach would be to extend the training of the algorithms to include data from multiple authorities. However, this strategy could quickly become prohibitively expensive to maintain, as the complexity of the task would increase significantly. This added complexity would likely result in diminished performance and reduced attractiveness of the supervised paradigm, further favoring an LLM-based approach for its scalability and adaptability.

Nonetheless, in narrower applications where precision and recall are critical—such as tasks requiring highly reliable outputs—investing in the creation of a high-quality labeled dataset for supervised ML training may remain a preferable choice. This is exemplified by the *LinkExtractor* system discussed in Chapter 5, which delivers very high performance and reliability within its intended scope. However, its domain specificity restricts its adaptability to contexts beyond Dutch legal texts.

In conclusion, while LLMs offer a scalable and adaptable solution to the challenges posed by insufficient labeled data, their effectiveness must be carefully weighed against their limitations (e.g., transparency and cost) and the specific goals of the task (e.g., achieving high accuracy or enabling extensive but less accurate—yet still reliable—information extraction). Depending on the scope, variability, and precision requirements of the task, both LLMs and supervised ML approaches have proven to be valuable tools in the context of Dutch administrative decision metadata extraction.

[1] At the time of writing, the last release of *LexNLP* is version 2.3.0, published on November 30, 2022, nearly three years ago. See `https://github.com/LexPredict/lexpredict-lexnlp`.

Bibliography

[1] Waad Alhoshan, Alessio Ferrari, and Liping Zhao. Zero-shot learning for requirements classification: An exploratory study. Information and Software Technology, 159:107202, 2023.

[2] Thao Do. Reference-based post-ocr processing with llm for diacritic languages, 2024.

[3] Tao Fang, Derek F. Wong, Lusheng Zhang, Keyan Jin, Qiang Zhang, Tianjiao Li, Jinlong Hou, and Lidia S. Chao. Llmcl-gec: Advancing grammatical error correction with llm-driven curriculum learning, 2024.

[4] Mathieu Fenniak, Matthew Stamy, pubpub zz, Martin Thoma, Matthew Peveler, exiled-kingcc, and PyPDF2 Contributors. The PyPDF2 library, 2022.

[5] Andrea Ferrario and Mara Naegelin. The art of natural language processing: Classical, modern and contemporary approaches to text document classification. SSRN Electronic Journal, March 2020. Available at SSRN: `https://ssrn.com/abstract=3547887` or `http://dx.doi.org/10.2139/ssrn.3547887`.

[6] R. Stuart Geiger, Dominique Cope, Jamie Ip, Marsha Lotosh, Aayush Shah, Jenny Weng, and Rebekah Tang. "garbage in, garbage out" revisited: What do machine learning application papers report about human-labeled training data? CoRR, abs/2107.02278, 2021.

[7] Aurélien Géron. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems. O'Reilly Media, 2019.

[8] P. Giorgini, F. Massacci, J. Mylopoulos, and N. Zannone. Modeling security requirements through ownership, permission and delegation. In 13th IEEE International Conference on Requirements Engineering (RE'05), pages 167–176, 2005.

[9] Morgan Gray, Jaromir Savelka, Wesley Oliver, and Kevin Ashley. Can GPT Alleviate the Burden of Annotation? 12 2023.

[10] Bowen Gu, Vivian Shao, Ziqian Liao, Valentina Carducci, Santiago Romero Brufau, Jie Yang, and Rishi J Desai. Scalable information extraction from free text electronic health records using large language models. medRxiv, 2024.

[11] Muhammad Usman Hadi, Qasem Al Tashi, Abbas Shah, Rizwan Qureshi, Amgad Muneer, Muhammad Irfan, Anas Zafar, Muhammad Bilal Shaikh, Naveed Akhtar, Jia Wu, Seyedali Mirjalili, and Mubarak Shah. Large language models: A comprehensive survey of its applications, challenges, limitations, and future prospects. August 2024.

[12] Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. spacy: Industrial-strength natural language processing in python. 2020.

[13] Yifei Hu, Xiaonan Jing, Youlim Ko, and Julia Taylor Rayz. Misspelling correction with pre-trained contextual language model. CoRR, abs/2101.03204, 2021.

[14] M. J. Bommarito II, D. M. Katz, and E. M. Detterman. Lexnlp: Natural language processing and information extraction for legal and regulatory texts. In Research Handbook on Big Data Law. Edward Elgar Publishing, 2021.

[15] Daniel Jurafsky and James H. Martin. Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models. 3rd edition, 2024. Online manuscript released August 20, 2024.

[16] Margaret L. Kern, Gregory Park, Johannes C. Eichstaedt, H. Andrew Schwartz, Maarten Sap, Laura K. Smith, and Lyle H. Ungar. Gaining insights from social media language: Methodologies and challenges. Psychological Methods, 21(4):507–525, 2016.

[17] Fred Kort. Predicting supreme court decisions mathematically: A quantitative analysis of the "right to counsel" cases. The American Political Science Review, 51(1):1–12, 1957.

[18] Rushi Longadge and Snehalata Dongre. Class imbalance problem in data mining review, 2013.

[19] Reed McEwan, Genevieve B. Melton, Benjamin C. Knoll, Yan Wang, Gretchen Hultman, Justin L. Dale, Tim Meyer, and Serguei V. Pakhomov. NLP-PIER: A Scalable Natural Language Processing, Indexing, and Searching Architecture for Clinical Notes. AMIA Joint Summits on Translational Science Proceedings, 2016:150–159, 2016. eCollection 2016.

[20] Sedir Mohammed, Lukas Budach, Moritz Feuerpfeil, Nina Ihde, Andrea Nathansen, Nele Noack, Hendrik Patzlaff, Felix Naumann, and Hazar Harmouch. The effects of data quality on machine learning performance, 2024.

[21] W. Y. Mok and J. R. Mok. Legal machine-learning analysis: First steps towards a.i. assisted legal research. In Proceedings of the Seventeenth International Conference on Artificial Intelligence and Law (ICAIL '19), pages 266–267, New York, NY, USA, 2019. Association for Computing Machinery.

[22] Tony Mullen and Robert Malouf. A preliminary investigation into sentiment analysis of informal political discourse. pages 159–162, 01 2006.

[23] Stuart Nagel. Predicting court cases quantitatively. Michigan Law Review, 63(8):1411, 1965.

[24] Humza Naveed, Asad Ullah Khan, Shi Qiu, Muhammad Saqib, Saeed Anwar, Muhammad Usman, Naveed Akhtar, Nick Barnes, and Ajmal Mian. A comprehensive overview of large language models, 2024.

[25] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12:2825–2830, 2011.

[26] Uma R. Salunkhe and Suresh N. Mali. Classifier ensemble design for imbalanced data classification: A hybrid approach. Procedia Computer Science, 85:725–732, 2016. International Conference on Computational Modelling and Security (CMS 2016).

[27] Carlo Sansone and Giancarlo Sperlí. Legal information retrieval systems: State-of-the-art and open issues. Information Systems, 106:101967, 2022.

[28] Jeffrey A. Segal. Predicting supreme court cases probabilistically: The search and seizure cases, 1962-1981. American Political Science Review, 78(4):891–900, 1984.

[29] Fred R. Shapiro. The most-cited articles from the yale law journal. Yale Law Journal, 100:1449, 1991.

[30] Marco Siino, Ilenia Tinnirello, and Marco La Cascia. Is text preprocessing still worth the time? a comparative survey on the influence of popular preprocessing methods on transformers and traditional classifiers. Information Systems, 121:102342, 2024.

[31] M. W. Vail T. D. Breaux and A. I. Anton. Towards regulatory compliance: Extracting rights and obligations to align requirements with regulations. In 14th IEEE International Requirements Engineering Conference (RE'06), pages 49–58, 2006.

[32] S. Sidney Ulmer. Quantitative analysis of judicial processes: Some practical and theoretical applications. Law and Contemporary Problems, 28(1):164–184, Winter 1963.

[33] Sowmya Vajjala, Bodhisattwa Majumder, Anuj Gupta, and Harshit Surana. Practical Natural Language Processing. O'Reilly Media, Inc., 2020.

[34] Marc van Opijnen. Search engines don't understand lawyers, but smart technologies can improve access to legal information. In ESCB Legal Conference 2018. European Central Bank, December 2018.

[35] Marc van Opijnen, Nico Verwer, and Jan Meijer. Beyond the experiment: The extendable legal link extractor. In Workshop on Automated Detection, Extraction and Analysis of Semantic Information in Legal Texts, held in conjunction with the 2015 International Conference on Artificial Intelligence and Law (ICAIL), San Diego, CA, USA, June 2015. Available at SSRN: `https://ssrn.com/abstract=2626521`.

[36] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

[37] Deepali Virmani and Shweta Taneja. A text preprocessing approach for efficacious information retrieval. In Bijaya Ketan Panigrahi, Munesh C. Trivedi, Krishn K. Mishra, Shailesh Tiwari, and Pradeep Kumar Singh, editors, Smart Innovations in Communication and Computational Sciences, pages 13–22, Singapore, 2019. Springer Singapore.

[38] Yaqing Wang, Quanming Yao, James T. Kwok, and Lionel M. Ni. Generalizing from a few examples: A survey on few-shot learning. ACM Comput. Surv., 53(3), June 2020.

[39] Fusheng Wei, Robert Keeling, Nathaniel Huber-Fliflet, Jianping Zhang, Adam Dabrowski, Jingchao Yang, Qiang Mao, and Han Qin. Empirical study of llm fine-tuning for text classification in legal document review. In 2023 IEEE International Conference on Big Data (BigData), pages 2786–2792, 2023.

[40] Huihui Xu and Kevin Ashley. Argumentative segmentation enhancement for legal summarization, 2023.

List of Figures

List of Tables

Appendix A

Additional Resources by Chapter

A.2 Chapter 2: The Data

A.2.1 Additional Tables

Table A.2.1.1: Summary Statistics for Text Columns (units are words)

Statistic	`headline`	`description`	`text_pypdf2`
Average Length	8.68	23.00	1679.67
Median Length	8.00	20.00	1684.00
Standard Deviation	3.25	7.22	1744.76
25th Percentile	6.50	18.00	892.00
75th Percentile	10.00	28.00	1933.50
Range	(3, 19)	(9, 42)	(178, 19926)

Table A.2.1.2: Date Ranges for Publication and Decision Dates

Statistic	Start Date	End Date
`publication_date`	2004-06-02	2024-08-27
`decision_date`	2005-09-29	2024-08-13

A.2.2 The Labeling Protocol

This section outlines the protocol followed to ensure consistent and accurate labeling of the dataset. Labeling was carried out using a spreadsheet that contained the following fields: `id`, `file_number`, `headline`, `description`, `parties`, `party_labeled`, `file_link`, `(English) Translation`, and a dedicated column for each labeler. The `Translation` field combined text from both the `headline` and `description` columns and was generated using Google Translate to provide a consistent English version for reference. A drop down menu in the spreadsheet allowed each labeler to select a decision category, ensuring standardization and consistency during the labeling process. This structured approach facilitated efficient and uniform labeling across all entries in the dataset.

- **Labeling Approach:**
  - **Ontvanger(s):** information about the decision recipient(s) was extracted using a combination of the *headline*, *description*, and *full text* fields.

  - **Rechtsgevolg:** for identifying the legal effect, only the *headline* and *description* fields were used.

- **Rechtsgevolg Categories:** each decision was assigned **exactly one category** from the following predefined list:

  1. *Verguning verlening* (Grant)
  2. *Wijziging of aanpassen* (Amend/Modify)
  3. *Overdracht* (Transfer)
  4. *Intrekking of beëindiging* (Revoke/Withdraw)
  5. *None* (If no category applies)

  The category selection was based on domain knowledge and expectation for the most frequent categories given the data at hand (ACM permits).

- **Company Names:** company names were extracted in their most complete and consistent form from the available fields (*headline*, *description*, or *body*). Abbreviations or partial names were avoided unless explicitly stated in the decision, ensuring uniformity and clarity in the dataset.

- **Randomized Decision Order:** to minimize bias and avoid linked decisions being labeled by the same individual (e.g. PDFs from the same decision page of the ACM website), the decisions were presented to the labelers in a randomized order.

- **Quality Assurance:** in cases of uncertainty or ambiguity, remarks were documented through comments on the spreadsheet, and unclear or difficult-to-categorize decisions were flagged for potential follow-up or discussion.

- **Labelers:** the decisions were labeled collaboratively by my supervisor (Dutch PhD candidate in Legal Data Science) and myself (Data Science Bachelor student), relying exclusively on the original Dutch text. The English translation of the headline and description generated for each decision was used by me only in cases of ambiguity or uncertainty regarding the understanding of the text, but never as the sole basis for choosing a label. My prior exploration and cleaning of the dataset exposed me to the language patterns commonly used in the *headline* and *description* columns, which, combined with my existing knowledge of Dutch, enabled me to confidently understand and label the data. Regarding party extraction, I encountered minimal difficulties as the parties were generally clearly stated in the text.

This labeling protocol ensured a structured and reproducible process, facilitating the creation of a reliable dataset for further analysis.

A.2.3 Spelling Correction Dutch Prompts

Prompt 1 for Spelling Correction (Dutch)

Corrigeer uitsluitend de spelfouten in deze Nederlandse tekst: `{decision body}`. Behoud de oorspronkelijke structuur, inhoud, en opmaak. Geef alleen de gecorrigeerde tekst terug in het `text_pypdf2_gpt`-veld van de JSON-output. Er mag geen extra tekst voor of na de JSON-output staan. Alleen het gegeven JSON schema is toegestaan.

Prompt 2 for Spelling Correction (Dutch)

Corrigeer alleen spelfouten in deze Nederlandse tekst: `{text}`. Behoud strikt de oorspronkelijke structuur, inhoud en opmaak. De API mag maximaal één woord per keer aanpassen of samengestelde woorddelen (bijv. "au to" naar "auto") corrigeren. Geef uitsluitend de gecorrigeerde tekst terug in het veld `text_pypdf2_gpt` van de JSON-output. Er mag geen extra tekst voor of na de JSON-output staan, en alleen het gegeven JSON-schema is toegestaan. Corrigeer geen stijl, grammatica of formulering, alleen spelling.

A.3 Chapter 3: Recipient Extraction

A.3.1 Additional Tables

Table A.3.1.3: Sample of NER predictions with ten TP

Index	Ground Truth Parties	Predicted Parties	Deduplicated Predictions	TP
266	Linthorst Energie Services B B.V.	Energie Services B B.V.	Energie Services B B.V.	1
92	Allure Energie B.V.	Allure Energie, Allure Energie B.V.	Allure Energie B.V.	1
37	Enstroga B.V.	Enstroga, Enstroga	Enstroga	1
87	ServiceHouse B.V.	ServiceHouse, ServiceHouse B.V.	ServiceHouse B.V.	1
342	ConceptsnSolutions B.V.	ConceptsnSolutions, ConceptsnSolutions B.V.	ConceptsnSolutions B.V.	1
7	Holthausen Clean Energy B.V.	Holthausen Clean Energy, Holthausen Clean Energy B.V.	Holthausen Clean Energy B.V.	1
267	All In Power B.V.	All In Power B.V.	All In Power B.V.	1
105	Republiq Community NL BV	Republiq Community NL B.V.	Republiq Community NL B.V.	1
278	GP Groot	GP Groot, GP Groot	GP Groot	1
253	Kernion Energie B.V.	Kernion, Kernion	Kernion	1

Table A.3.1.4: FP (all) NER test predictions showing FP (all)

Index	Ground Truth Parties	Predicted Parties	Deduplicated Predictions	FP
52	Cogas Duurzaam B.V.	Cogas, Duurzaam	Cogas, Duurzaam	1
56	GDF SUEZ Energie Nederland N.V., ENGIE Energie Nederland N.V.	Energievergunning GDF SUEZ	Energievergunning GDF SUEZ	1
261	Holthausen Clean Energy B.V.	Holthausen Clean Energy, Clean Energy, Clean Energy B.V., Clean Energy B.V.	Holthausen Clean Energy, Clean Energy B.V.	1
338	E.ON Benelux Levering B.V., Eneco Zuid Nederland B.V.	Energievergunning E.ON Benelux	Energievergunning E.ON Benelux	1
341	Delta Energie B.V.	Delta Comfort B.V., Delta Energie B.V, Delta Comfort B.V., Delta Energie B.V.	Delta Comfort B.V., Delta Energie B.V.	1

Table A.3.1.5: FN (all) among NER test predictions

Index	Ground Truth Parties	Predicted Parties	Deduplicated Predictions	FN
11	e-Energy Europe B.V.			1
25	InEnergie Levering B.V., Greenspread Energy B.V.	Greenspread Energy, Greenspread Energy B.V	Greenspread Energy B.V	1
72	Electrabel UnitedConsumers Energie B.V., ENGIE UnitedConsumers Energie B.V.	ENGIE UnitedConsumers	ENGIE UnitedConsumers	1
85	Innova Energie B.V.			1
125	Zonneplan Energie B.V.			1
138	Kas Energie Nederland B.V.			1
163	Hezelaer Energy B.V.			1
164	Hezelaer Energy B.V.			1
173	Intergas Levering B.V., Dong Energy Sales B.V.	Dong Energy Sales B.V, Dong Energy Sales B.V.	Dong Energy Sales B.V.	1
219	FENOR B.V., Budget Energie	FENOR, FENOR	FENOR	1
230	Essent Energie Verkoop Nederland B.V., Eneco Midzakelijk B.V.	Eneco Midzakelijk, Eneco Midzakelijk	Eneco Midzakelijk	1
231	Essent Energie Verkoop Nederland B.V., Eneco Midzakelijk B.V.	Eneco Midzakelijk, Eneco Midzakelijk	Eneco Midzakelijk	1
257	Wij Maken Energie B.V.			1
289	Kikker Energie B.V.			1

A.4 Chapter 4: Date Extraction

A.4.1 Additional Figures

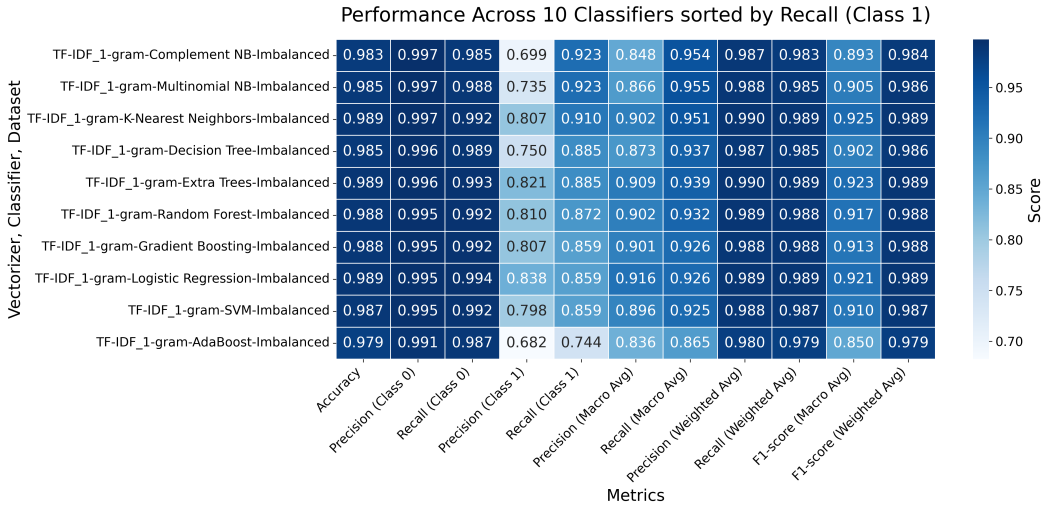


Figure A.4.1.1: Comparison of the best ten imbalanced classifiers (based on and sorted by Recall for Class 1)

A.5 Chapter 5: Legal References Extraction

A.5.1 Additional Figures and Tables

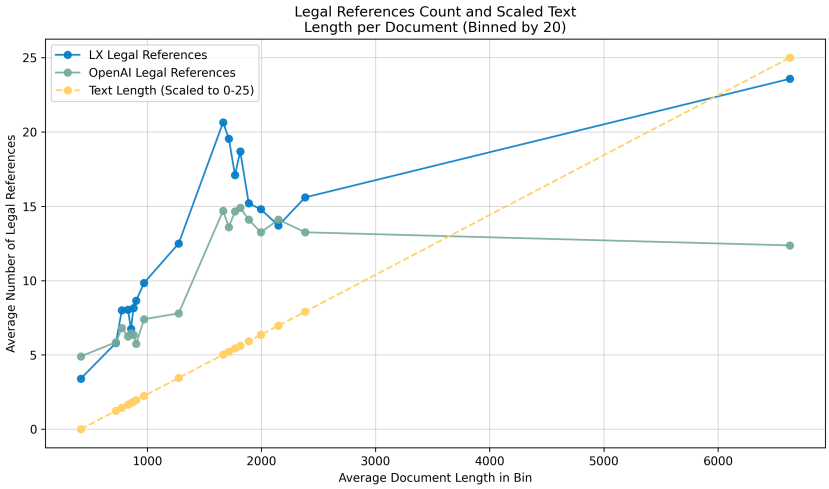


Figure A.5.1.2: Comparison of the average number of extracted legal references by LX and LLM across binned document lengths, with scaled document length included

A.5.2 LinkExtractor Architecture

1. **Input Document**: the pipeline begins with receiving the input document. This can be a complete file, such as a court ruling, or a short text fragment.

2. **Conversion and Fragment Selection:** the received documents are converted into the required format. If necessary, specific fragments are selected for further processing.

3. **NER:** this component is one of the most important in the pipeline: it's here that references to laws, regulations, and court rulings are identified. It uses a database containing approximately 200,000 entities with standardized URIs[1] for accurate recognition.

4. **Pattern Recognition:** extensive grammars are employed to search for legal source indicators, managing variations such as "NJ 2001, 34", "N.J. 2001/34" and "Ned Jur 2001-34."

5. **Local Alias Detection:** local aliases are document-specific references used to simplify citations of frequently mentioned sources. For example, if a document states, "Regulation (EC) 1408/71 (hereinafter: 'the Regulation')," then "the Regulation" becomes the local alias. Later, when the text mentions "Art. 5 of the Regulation" LX will, like a human reader, recognize it as Article 5 of Regulation (EC) 1408/71.

6. **False Positive Removal:** to ensure precision, the system filters out short abbreviations that could be ambiguous unless used in specific contexts. For instance, "RTL" can refer to both the *Regeling Toezicht Luchtvaart*[2] and a broadcasting company. Therefore, named entities of six characters or fewer are removed unless accompanied by a specific component (e.g., "Art. 5 RTL") or defined as a local alias.

7. **Civil Code Resolver:** this component resolves references to the Dutch Civil Code (BW[3]) by handling the distinct BWB [?] identifiers assigned to each book within the

[1] *Unique Resource Identifier*
[2] *Regulation on Aviation Supervision*
[3] Burgerlijk Wetboek

BWB system. LX can correctly recognize references when the specific book is mentioned (e.g., "Art. 3, Book 6 BW"), but it cannot handle incomplete references like "Art. 3 BW," even when the context suggests the intended book.

8. **Resolve Multiple References:** this step handles complex cases where multiple references appear within a single text segment, ensuring all are correctly interpreted.
9. **URI Generator:** the system generates canonical URIs for all resolved references, ensuring consistent and unique identification.
10. **Post-Processing:** additional document conversions are applied, and relevant metadata is added to the processed document during this phase.
11. **Output Document:** the final step produces the processed document, customized according to the input parameters and ready for use.
12. **Reference repositories:** LX uses the reference repositories listed in Fig. A.5.2.3 to recognize named entities and verify identifiers recognized through patterns.

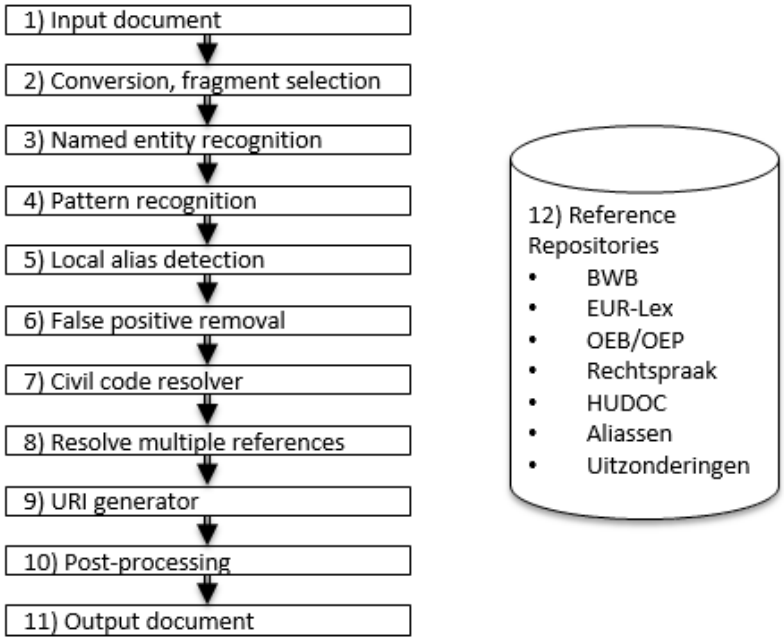


Figure A.5.2.3: Visual Summary of the LX Architecture (image by KOOP)

A.6 Chapter 6: Legal Effect Classification

A.6.1 Additional Tables

Table A.6.1.6: Classification report of the three best performing algorithms.
(1) Logistic Regression, (2) Complement NB
(*) Headline and description, (**) Headline, description, full text.

Classifier	**(1)**	**(2)**	**(1)**
Vectorizer	Binary 1-gram BoW	TF-IDF_1-gram	1,2,3,4-gram BoW
Dataset	(**)	(*)	(**)
Accuracy	0.973	0.973	0.964
Precision_intrekking_of_beëindiging	1.000	0.895	1.000
Recall_intrekking_of_beëindiging	1.000	1.000	1.000
F1_intrekking_of_beëindiging	1.000	0.944	1.000
Support_intrekking_of_beëindiging	17.000	17.000	17.000
Precision_none	0.833	1.000	0.800
Recall_none	0.714	0.571	0.571
F1_none	0.769	0.727	0.667
Support_none	7.000	7.000	7.000
Precision_verguning_verlening	0.965	1.000	0.948
Recall_verguning_verlening	1.000	1.000	1.000
F1_verguning_verlening	0.982	1.000	0.973
Support_verguning_verlening	55.000	55.000	55.000
Precision_wijziging_of_aanpassen	1.000	0.971	1.000
Recall_wijziging_of_aanpassen	0.970	1.000	0.970
F1_wijziging_of_aanpassen	0.985	0.985	0.985
Support_wijziging_of_aanpassen	33.000	33.000	33.000
Precision_macro_avg	0.950	0.966	0.937
Recall_macro_avg	0.921	0.893	0.885
F1_macro_avg	**0.934**	**0.914**	**0.906**
Support_macro_avg	112.000	112.000	112.000
Precision_weighted_avg	0.972	0.975	0.962
Recall_weighted_avg	0.973	0.973	0.964
F1_weighted_avg	0.972	0.970	0.962
Support_weighted_avg	112.000	112.000	112.000