Rule-based Extraction of Constraints from Regulatory Texts

Leo Holzhauer
leo.holzhauer@tum.de
Technical University Munich

Abstract

This study tackles the task of extracting and rebuilding constraints from regulatory texts to ensure compliance in business processes, with a focus on certifications within the voluntary carbon market. As the volume of regulatory requirements expands, the necessity for automated systems to interpret these documents becomes evident. A rule-based method, augmented by Generative Pre-trained Transformers (GPT) from OpenAI, is introduced for the systematic identification and reconstruction of constraint items from textual data. The developed data pipeline is modular and scalable, facilitating adaptation to a variety of use cases. Quantitative and qualitative evaluations demonstrate the approach's effectiveness, highlighting its potential benefits and pinpointing opportunities for future enhancements. Incorporating advanced language models adds a new dimension to constraint reconstruction, blending rule-based accuracy with the generative capabilities of machine learning.

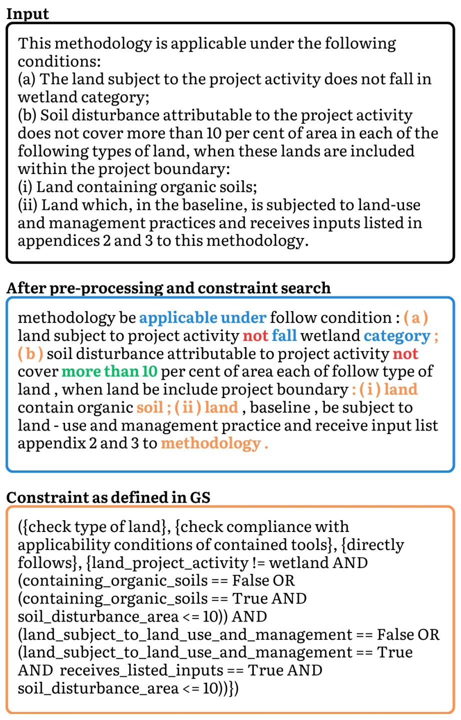
1 Introduction

Businesses that demonstrate resilience and efficiency are often underpinned by well-defined processes and regulatory frameworks (1). With the burgeoning volume of regulatory documents, particularly in domains such as ecological sustainability, the imperative for an automated mechanism to assure compliance with business processes has become increasingly evident (2). The growing complexity and scope of regulations within the voluntary carbon market exemplify the urgent need for such automated systems to ensure that businesses can navigate and comply with these evolving standards effectively.

A pivotal element within this context is the extraction and formalization of constraints from textual sources. According to (3), within the domain of business processes, constraints are characterized as sentences comprising at least one marker indicative of a constraint. These markers serve to identify constraint items which, in turn, can be amalgamated to formulate a comprehensive constraint. For an illustration of the complexity of these constraints, which might incorporate various types of constraint markers, refer to Figure 1.

The objective of this research was to aggregate a dataset encapsulating business processes and their associated constraints across a variety of real-world domains. The aim was to reconstruct this dataset through a rule-based methodology, striving for optimal accuracy while ensuring the system's modularity and scalability. This approach facilitates straightforward enhancements and modifications, enabling application across a broad spectrum of use cases.

The development of the data pipeline was undertaken using Python, integrating state-of-the art open-source Natural Language Processing (NLP) technologies such as spaCy (4), Sentence-BERT (S-BERT) (5), and Generative Pre-trained Transformers (GPT) from OpenAI. The entire dataset, along with the source code, has been made publicly available on GitHub under the MIT License.



Input

This methodology is applicable under the following conditions:
(a) The land subject to the project activity does not fall in wetland category;
(b) Soil disturbance attributable to the project activity does not cover more than 10 per cent of area in each of the following types of land, when these lands are included within the project boundary:
(i) Land containing organic soils;
(ii) Land which, in the baseline, is subjected to land-use and management practices and receives inputs listed in appendices 2 and 3 to this methodology.

After pre-processing and constraint search

methodology be **applicable under** follow condition : **(a)** land subject to project activity **not fall** wetland **category ;** **(b)** soil disturbance attributable to project activity **not** cover **more than 10** per cent of area each of follow type of land , when land be include project boundary **: (i) land** contain organic **soil ; (ii) land** , baseline , be subject to land - use and management practice and receive input list appendix 2 and 3 to **methodology .**

Constraint as defined in GS

```
({check type of land}, {check compliance with
applicability conditions of contained tools}, {directly
follows}, {land_project_activity != wetland AND
(containing_organic_soils == False OR
(containing_organic_soils == True AND
soil_disturbance_area <= 10)) AND
(land_subject_to_land_use_and_management == False OR
(land_subject_to_land_use_and_management == True
AND  receives_listed_inputs == True AND
soil_disturbance_area <= 10))})
```

Figure 1: Example for a complex constraint and different types of constraint markers

2 Dataset

To enhance the relevance and diversity of the dataset, descriptions of regulatory processes from multiple real-world sources were incorporated. To augment this diversity further, synthetic data were integrated into the dataset. Consequently, the dataset encompasses eight distinct use cases, detailed in Table 1. For each case, a pair of files was created: one input file that encapsulates the descriptive information on the process, and one output file that contains the gold standard (GS). The output file delineates the anticipated outcomes from the extraction of constraints.

3 Pre-processing

The preprocessing phase involves several key operations: segmenting text into sections, cleaning and modifying character sequences in the unprocessed text, dividing the text into smaller segments (“chunks”), tokenization, lemmatization, part-of-speech (POS) tagging, eliminating stop words, and identifying enumeration details.

Each use case is imported from the input files into the data pipeline. This process includes the elimination or modification of specific character sequences and the removal of tabular indentations. Sequences of line breaks and white spaces are consolidated into single occurrences of each, preparing the text for segmentation into chunks designed to encapsulate all relevant context for the extraction of no more than one constraint, which may comprise several interconnected components. These chunks are identified as either sentences or enumeration sequences, with the latter being treated as singular entities when all items are related to a preceding sentence. Following segmentation, line breaks are exCIed to prevent interference with tokenization and POS tagging, though this information is preserved in an additional feature vector corresponding to each segment’s token vector length, indicating the original presence of a line break after each token for later use in context analysis.

Utilizing one of spaCy’s pre-trained pipelines, each text segment undergoes tokenization and lemmatization, followed by the removal of stop words. Stop words, typically common and semantically non-significant, are often excluded in NLP to enhance the efficiency of language processing or information retrieval [(14),(15)]. The spaCy-provided default stop word list serves as the baseline, which is then refined by retaining stop words crucial for understanding constraints, including connectors ("and", "or"), negators ("no", "not"), and relational terms ("beyond", "least"). A comprehensive list of the excluded stop words is detailed in the appendix’s parameter section.

In the final step, the process identifies and extracts details related to enumeration items within each chunk, if applicable. This includes determining the location, type, and hierarchical structure of enumeration items, which is vital for the accurate construction of constraints.

4 Modelling

This section delineates the methodology employed in identifying constraint items within the pre-processed dataset and elaborates on the subsequent assembly of constraints from these identified items. The construction of each constraint involves the integration of one or more constituent components, referred to as constraint components.

An exploratory analysis conducted on the pre-processed dataset, inclusive of Part-of-Speech (POS) tagging, facilitated the identification of three distinct types of constraint items (CI): *inequality*, *equality*, and *meta*. These classifications are instrumental in the systematic extraction and categorization of constraint items. The identified constraint item types are detailed in Table 2.

Algorithm 1 Get Constraint

```
Require: $nlp$, $builder$, $chunk$, $eqParams$, $ineqParams$, $metaParams$, $enumSummary$, $linebreaks$
Ensure: $constraint$
 1: Initialize $inequality$, $equality$, $meta$ searchers with respective parameters
 2: Search for inequality CI in $chunk$
 3: Search for equality CI in $chunk$
 4: Search for meta enumeration CI in $chunk$ with $enumSummary$
 5: Search for meta if and for clauses in $chunk$
 6: Combine findings into $constraint$
 7: if $constraint$ found then
 8:     $constraint \leftarrow$ Determine context
 9:     $constraint \leftarrow$ Insert connections
10:     $constraint \leftarrow$ Insert boolean CI
11:     $constraint \leftarrow$ Sort and prune
12:     if $constraint$ left then
13:         Build $formattedConstraint$
14:     end if
15: end if
16: Add $formattedConstraint$ to $constraint$
17: return $constraint$
```

Algorithm 1, pivotal to the modeling process, initiates by setting up constraint searchers for inequality, equality, and meta categories. Subsequent steps involve executing sequential searches across these constraint item categories, capturing essential positional details such as the presence of constraint markers ("match") and negations. Upon identifying any constraint items during these initial searches, the context for each constraint is ascertained, facilitating the generation of further constraint items and the subsequent processes of sorting and pruning. Should the pruning process yield a chunk still containing viable constraints, a structured constraint is then constructed with the aid of a builder mechanism. This construction leverages both a rule-based approach and the capabilities of a model from OpenAI’s GPT series. The compiled data on constraint

Table 1: Data sources (RW: real-world, S: synthetic)

	Symbol	Name	Constraints	Type	Source
1	AKTG	German Stock Corporation Act	8	RW	(6)
2	CDM1	Afforestation and reforestation of degraded mangrove habitats (UNFCCC CDM)	5	RW	(7)
3	CDM2	Afforestation and reforestation project activities implemented on lands other than wetlands (UNFCCC CDM)	5	RW	(8)
4	CDM3	Cable cars for mass rapid transit system (UNFCCC CDM)	6	RW	(9)
5	CDM4	Energy efficiency and/or energy supply projects in commercial buildings (UNFCCC CDM)	10	RW	(10)
6	CDM5	Electricity generation by the user (UNFCCC CDM)	14	RW	(11)
7	COFFEE	Coffee Roasting Process	26	S	(12)
8	PATG	German Patent Act	4	RW	(13)

Table 2: Types of constraint items

Type	Operators	Example
Inequality	<, <=, >, >=	`x less than y`
Equality	==, !=	`x must be y`
Meta	`AND`, `OR`	`if x, then y`

items, alongside the structured constraint, are then aggregated into a comprehensive summary of constraint information for each analyzed use case.

4.1 Inequality Constraint Items

The identification of inequality constraint items is generally straightforward due to their distinctive structure. The analysis of sentences derived from the carbon emission avoidance calculations domain exemplifies this:

- `The emission value shall not exceed 60 kt` CO_2`e.`
- `The electricity is available for less than 36 hours.`

Examination of the dependency structure in these sentences, as illustrated in Figure 4 in the appendix, reveals variability in the part-of-speech (POS) tag sequences and syntactic dependencies, complicating the application of a rule-based matching schema. Nonetheless, with limited exceptions in synthetic data, inequality constraint items consistently adhere to a discernible pattern. Specifically, the constraint marker constitutes a relational phrase (e.g., "below", "lower than", "exceed") that corresponds to a relational mathematical operator, followed by a numerical token. The repository contains a detailed compilation of lemmatized expressions serving as inequality constraint markers and their equivalent operators.

The assortment of inequality constraint markers indicative of strict inequalities (<, >) is expanded by appending "or equal to" to each marker, thereby encompassing non-strict inequalities (<=, >=) as well.

Moreover, each identified constraint item undergoes a review for negation tokens ("not", "no", "none") within a specified token range adjacent to the marker. The presence of a negation token in connection with a constraint item inversely modifies the operator.

4.2 Equality Constraint Items

Equality constraint items adhere to principles similar to those observed with inequality constraints, with typical markers including terms such as "shall", "should", and "must" (3). Consider the following examples:

- `The by-laws must be established by way of being recorded by a notary.`
- `This methodology does not apply to large-scale project activities.`

Unlike inequality constraints, equality constraint items lack a distinct linguistic marker for the reference value, complicating their construction. Additionally, negations are identified for equality constraints, similarly to inequality constraints, where their presence inversely affects the operator.

Equality constraint markers may also exhibit domain specificity. In the context of CDM use cases, for example, the emphasis is on ensuring the methodology's applicability for emissions attribution, making some markers pertinent to concepts of applicability and eligibility. Such specificity, while potentially limiting cross-domain relevance, is deemed acceptable provided it does not lead to false positive identifications.

4.3 Enumeration Constraint Items

The complexity inherent to legal texts, notably those of German origin, presents significant challenges for conventional Natural Language Processing (NLP) systems, as outlined by (16). Among these complexities, enumerations stand out as particularly intricate structures. For the purposes of this research, enumerations are delineated as follows:

Definition 1 (Enumeration)**.** *An enumeration is identified as a hierarchically organized list of enumeration* *items contained within a textual segment, initiated by a colon preceding the first item. Such enumerations comprise a minimum of two items and potentially extend across several hierarchy levels. The introduction of each item is marked by enumerators, and every item encapsulates at least one constraint item in conjunction with the enumeration-specific constraint item.*

An exemplar enumeration, derived from the AKTG case study, is provided below:
`The deed is to set out the following particulars:`

1. `the founders;`
2. `for par-value shares: the nominal amount; for no-par-value shares: their number, the issue price and, if various stock classes exist, the class of stocks each founder will acquire;`
3. `the paid-in amount of the share capital.`

In this framework, enumeration items are incorporated as constraint items, with additional data concerning their hierarchical level and connecting expressions recorded. Such connecting expressions, identified in the preamble to the first enumeration item and at the conclusion of the initial constraint item at each level, are scrutinized for phrases indicative of logical connectors (`AND`, `OR`).

4.4 If- and For-Clauses

If- and for-clauses function as meta constraint items, influencing both the structure and content of constraints. Analyzing the second enumeration item from the previous example, it becomes apparent that if- and for-clauses lead to a consistent structure: `if` A `then` B or `for` A `do` B. This structure is formalized as:

$$A \implies B \equiv \neg A \vee (A \wedge B) \equiv \neg A \vee B$$

The diversity of syntactic forms for the implication marker (e.g., "then", ",", ":", "do") and the frequent presence of "if" and "for" markers necessitate that if- and for-clauses are considered only when they encompass an inequality or equality constraint or are part of an enumeration. This is further elaborated in the section on pruning.

4.5 Context Determination

Determining the context is a crucial preparatory step for incorporating connectors and boolean constraint items, as well as for the subsequent pruning process. Context is defined as follows:

Definition 2 (Context)**.** *The context encompasses a range of tokens within a chunk, marked by a starting and an ending point. The start of the context is defined by a token whose index is less than or equal to the start of the constraint marker, while the end of the context is determined by a token whose index is greater than or equal to the end of the constraint marker. For any inequality or equality constraint item i, the context is determined by the most extreme (for the start) or least extreme (for the end) of the following criteria:*

1. *Context limit markers (";", ".").*
2. *A line break.*
3. *The end (for start) or start (for end) of another constraint marker not encompassing i.*
4. *The start (for start) or end (for end) of encompassing constraint items, where the constraint marker's start is less than or equal to the start of i and its end is greater than or equal to the end of i.*

For all meta constraint items, the context's start aligns with the constraint marker's start. For enumeration constraint items, the context's end matches the constraint marker's end. The determination of the context's end for if- and for-clauses aligns with that for inequality and equality constraint items.

4.6 Inserting Connectors

Upon establishing the set of constraint items along with their respective contexts, the process seeks to identify connectors. Connectors are articulated as follows:

Definition 3 (Connector)**.** *A connector refers to any phrase or symbol within an approved lexicon that resides within the contexts of two distinct constraint items, provided these contexts do not overlap. Each connector corresponds to a logical operator, specifically* `AND` *or* `OR`*.*

4.7 Incorporation of Boolean Constraint Items

Boolean constraint items, despite being interpreted as equality constraints, fall under the category of meta constraint items. This classification stems from their lack of explicit constraint markers within the text, necessitating the prior identification of other constraint items. For instance, the initial enumeration item in the preceding example includes a boolean constraint item. Any enumeration CI and both segments of each if-/for-clause yield a boolean constraint item, provided the encompassing enumeration/if/for CI does not already include an equality or inequality constraint. In the context of boolean items, the markers' start and end points align with those of the context, as determined by the encompassing meta constraint.

4.8 Pruning Process

The final stage in identifying and refining constraint items within a specified chunk involves their organization and selective exclusion, or pruning. Initially, constraint items are organized according to the commencement point of their markers. Subsequent pruning aims to establish a coherent constraint structure, ensuring that each pair of constraint components within a singular constraint is interconnected by a logical operator. The pruning process encompasses several scenarios:

1. Boolean constraint items enveloping an equality or inequality CI are excluded.
2. If- and for-clauses not included within an enumeration CI and lacking an associated equality or inequality CI within their scope are eliminated, along with any contained boolean CI.
3. In instances of consecutive inequality CI lacking an intervening connector and residing within mutual context, the subsequent inequality CI is omitted.
4. Similarly, when consecutive equality CI are identified without a linking connector and share a context, the subsequent equality CI is disregarded.
5. Should an equality CI be immediately followed by an inequality CI without a connecting operator, the equality CI is removed.
6. If two consecutive instances of "and" connectors between CI are found, both are discarded.

These measures ensure the integrity and logical coherence of the constructed constraints, streamlining the constraint structure for clarity and efficacy.

4.9 Constraint Building

Post-pruning, surviving constraint items are utilized to formulate the constraint in the designated output structure. The foundational structure of the constraint, or its skeleton, is devised through the application of enumeration, for, if, and connector constraint items. The assembly of the constraint components is initiated by the presence of equality, inequality, and boolean constraint items, which are subsequently integrated into the pre-established skeleton. The component construction process is bifurcated into two methodologies: a rule-based approach and one augmented by a GPT.

In the rule-based construction method, each constraint component is delineated into three segments: a left part, an operator, and a right part, with the operator derived directly from the constraint item. The assembly of the left and right segments varies contingent on the constraint item type—equality, inequality, or boolean:

- For inequality constraint items, the right segment embodies the numerical token, while the left segment is a concatenation of up to k proximal tokens within the context, excluding the span of the constraint marker. These tokens are restricted to verbs, nouns, or adjectives, with k set to a maximum of 4, to balance conciseness with contextual adequacy. The disregard for syntactic dependencies arises from the absence of a uniform schema (as illustrated in Figure 4).
- For equality constraint items, both left and right segments are composed of the k nearest tokens within context, adhering to the same categories and constraints as mentioned.
- Boolean constraint items are treated similarly for their left segment, whereas the right segment is distinctly set to True or False.

Conversely, the GPT-supported approach amalgamates the constraint components into a singular entity, formulated as the GPT's response to a specifically tailored prompt. This research utilizes OpenAI's `gpt-3.5-turbo-0125` and `gpt-4-1106-preview` models. The design of these prompts incorporates one-shot learning principles, as introduced by Brown et al. (17), and integrates a templating approach as recommended by White et al. (18). Detailed prompt configurations are enumerated in the appendix.

5 Evaluation

The evaluation consists of a quantitative and a qualitative evaluation. Due to the focus of this work on constraint extraction, only the constraint items and not the business process steps were evaluated. For the quantitative evaluation, underscores were removed before calculating the similarity scores, since they tend to improve the score, as to be seen in Subsection 5.2.

5.1 Quantitative Evaluation

To quantitatively evaluate the accuracy of constraint reconstruction, we utilize precision and recall metrics. Precision calculates the ratio of correctly identified constraints within the extracted set C_E, while recall measures the proportion of correctly identified constraints out of the total in the gold standard set C_{GS}.

The assessment of whether an extracted constraint is correct relies on the cosine similarity score between the embeddings of matched constraint pairs—one from C_E and one from C_{GS}—using the state-of-the-art S-BERT model for sentence similarity estimation [cf. (5), (2)]. Following this approach, we compute the similarity scores for all possible pairs, sort them in descending order, and sequentially match and remove constraints from their respective sets based on these scores, until one set is depleted. A constraint from C_E is deemed correct if its similarity with a counterpart in C_{GS} exceeds a specified threshold θ, ranging between 0 and 1.

In summary: Let $S(c_e, c_{gs})$ denote the cosine similarity score between an extracted constraint $c_e \in C_E$ and a gold standard constraint $c_{gs} \in C_{GS}$, as determined by the S-BERT model. A constraint c_e is classified as correct if there exists a $c_{gs} \in C_{GS}$ for which $S(c_e, c_{gs}) > \theta$. The formulas for precision and recall are defined as follows:

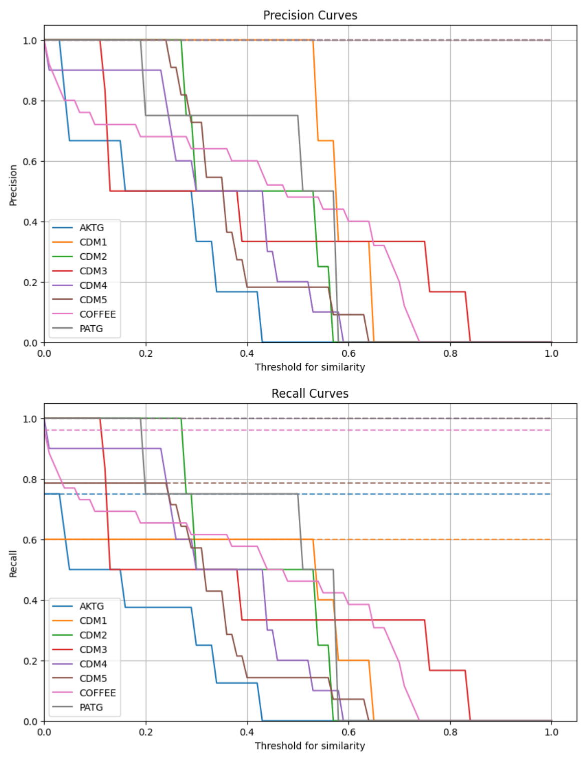


Figure 2: Precision and recall curves for a rule-based constraint component building with a variable similarity threshold θ. Dotted lines indicated the maximal achievable precision/recall ($\theta = 0$).

$$\text{Precision} = \frac{|c_e \in C_E \mid \exists c_{gs} \in C_{GS} : S(c_e, c_{gs}) > \theta|}{|C_E|} \tag{1}$$

$$\text{Recall} = \frac{|c_e \in C_E \mid \exists c_{gs} \in C_{GS} : S(c_e, c_{gs}) > \theta|}{|C_{GS}|} \tag{2}$$

These equations serve to measure the effectiveness of the constraint reconstruction process, employing cosine similarity scores to ascertain the correctness of the extracted constraints.

Utilizing the aforementioned equations with θ as a variable allows for the generation of precision and recall curves. Figure 2 presents a comprehensive plot summarizing the outcomes across all analyzed use cases, specifically highlighting the performance of the rule-based constraint component building functionality. The plot features dotted lines representing the maximal achievable precision and recall when θ is set to 0, serving as an indicator of the constraint detection's efficacy. Furthermore, the area beneath these curves offers a metric for evaluating the effectiveness of the constraint building process.

The analysis reveals that, across all use cases, the maximum achievable precision consistently reaches 1.0, whereas the maximum achievable recall varies between 0.8 and 1.0, except for the CDM1 use case, which is notably lower at 0.6. This suggests that the quantity of extracted constraints generally aligns well with the actual number of constraints, albeit occasionally underestimated. This demonstrates the efficacy of constraint detection. However, the success of constraint building varies significantly across different use cases. The AKTG case, in particular, poses challenges, likely due to the intricate nature of its legal text. Conversely, constraints identified in the CDM1 case consistently exhibit considerable similarity scores. Among all evaluated constraint pairs, only a single pair from CDM3 achieves a similarity score surpassing 0.8, highlighting a broad discrepancy with the manually crafted gold standard. Comparative analysis of results from two distinct GPT models, detailed in the appendix, shows comparable outcomes. It's important to acknowledge the impact of the probabilistic elements inherent in spaCy's POS tagging and OpenAI's GPT chat completion on the non-deterministic nature of constraint building, which compromises evaluation robustness. To enhance robustness, considering an average over multiple iterations may offer a viable approach for future evaluations.

5.2 Motivation for Qualitative Evaluation

The qualitative evaluation complements the quantitative analysis by addressing nuances not captured through precision, recall, and similarity scores. This approach is essential due to the limitations of a purely quantitative analysis, which overlooks semantic integrity and the variability in constraint formulations. The qualitative evaluation examines the semantic accuracy and contextual appropriateness of the extracted constraints, particularly when different phrasings convey the same requirement.

Examples of Constraint Formulations:

Identified categories of alignment between the extracted constraints and their references in the gold standard range from perfect matches to significant deviations:

- **Perfect:**
```
(co_fires_fossil_fuel == False OR
capacity_entire_unit <= 15)
co_fires_fossil_fuel == False OR
capacity_entire_unit < 15
```
- **Great:**
```
electricity_saving_single_project
<= 60
electricity_savings_per_year <=
60
```
- **Good:**
```
application ==
disclose_invention_manner_clear
application_dicslose_clear_and_complete
== True
```
- **OK:**
```
heat_power_cogeneration_system !=
category
combined_heat_and_power == False
```

- **Bad:**
```
application ==
relate_invention_group_invention)
application_relate_one_invention
== True OR linked == True
```

- **Garbage:**
```
law == establish_way
recorded_by_notary == True
```

Semantic Similarity Scores:

To demonstrate the impact of formulation variations, scores for selected constraint pairs are presented, contrasting versions with and without underscores:

Table 3: Semantic similarity scores for varying levels of constraint formulation alignment.

Reference	Score w/ _	Score w/o _
(1) Perfect	0.9647	0.9843
(2) Great	0.8502	0.7429
(3) Good	0.2545	0.2421
(4) OK	0.6139	0.4985
(5) Bad	0.6674	0.7079
(6) Garbage	0.2838	0.3194

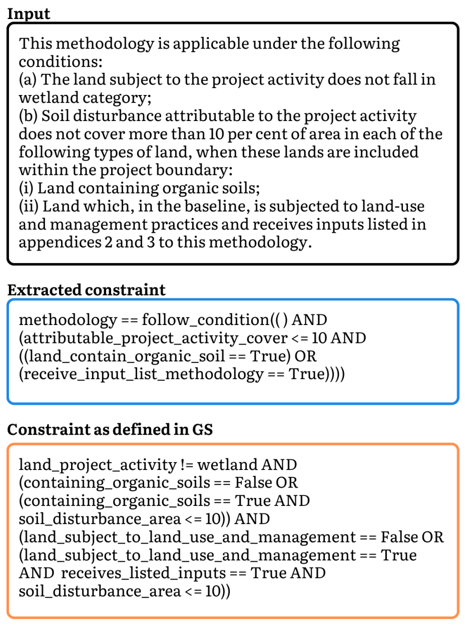
Discrepancies in semantic similarity scores highlight the complexity of evaluating constraint accuracy based on quantitative measures alone. Notably, cases like (3) and (5) illustrate that a high similarity score does not necessarily correlate with a correct or meaningful semantic match, due to variations in the construction of constraints or the use of synonymous terms that change the perceived meaning.

This emphasizes the necessity for human expertise in the assessment process, pointing out the inadequacies of automated, quantitative evaluations in fully capturing the semantic spectrum of constraint accuracy. Although quantitative measures offer a valuable initial estimation of the efficacy of constraint extraction and building, they must be complemented with detailed qualitative analysis to achieve a comprehensive evaluation of system performance.

5.3 Qualitative Evaluation

A comprehensive qualitative evaluation exceeds the scope of this study. Nonetheless, the example depicted in Figure 3 has been meticulously chosen to elucidate various facets of the constraint extraction process.

Observations indicate that the extracted constraint manifests with a notably reduced length for several reasons. Initially, the detection of the constraint corresponding to enumeration item (a) is evident, as denoted by the parentheses. However, the construction was unsuccessful due to an inaccurately defined context lacking in nouns, verbs, or adjectives. Furthermore, the disparity in the variety of case combinations between the gold standard and the extracted constraint can be attributed to the overarching connection between items (b) and subitems (i) and (ii) across different enumeration levels. On a positive note, the identification of the negated inequality constraint is commendable, although the representation of the symbol could be enhanced for clarity. Additionally, the depiction of organic soils and the incorporation of listed inputs are accurately captured. The condensed format of the extracted constraint arguably offers an initial impression of coherence, leveraging the logical equivalence of implication and the amalgamation of disparate logical scenarios to streamline the constraint. However, the constraint items pertaining to land-use and management, alongside the connector in (ii), were not recognized, attributed to their exclusive boolean nature (lacking an equality type constraint).


Input

This methodology is applicable under the following conditions:
(a) The land subject to the project activity does not fall in wetland category;
(b) Soil disturbance attributable to the project activity does not cover more than 10 per cent of area in each of the following types of land, when these lands are included within the project boundary:
(i) Land containing organic soils;
(ii) Land which, in the baseline, is subjected to land-use and management practices and receives inputs listed in appendices 2 and 3 to this methodology.

Extracted constraint

methodology == follow_condition(( ) AND
(attributable_project_activity_cover <= 10 AND
((land_contain_organic_soil == True) OR
(receive_input_list_methodology == True))))

Constraint as defined in GS

land_project_activity != wetland AND
(containing_organic_soils == False OR
(containing_organic_soils == True AND
soil_disturbance_area <= 10)) AND
(land_subject_to_land_use_and_management == False OR
(land_subject_to_land_use_and_management == True
AND receives_listed_inputs == True AND
soil_disturbance_area <= 10))

Figure 3: Illustration of the same constraint as in Figure 1, now with the extracted constraints.

6 Conclusion

This study's investigation into the rule-based reconstruction of constraint items for business processes elucidates a spectrum of potential benefits and inherent challenges. The process of identifying constraint items generally adheres to explicit rules, facilitating a relatively straightforward approximation. However, the subsequent phase of constraint construction presents additional complexities.

Drawing upon extant literature, the present work introduces a theoretical framework for categorizing constraints into three distinct types: inequality, equality, and meta. Additionally, the groundwork for an annotated real-world dataset spanning multiple domains was established. Moreover, it establishes a modular methodology for a comprehensive processing pipeline, laying the groundwork for facile, iterative enhancements aimed at accommodating an expanded diversity of use cases. This study also explores the integration of a state-of-the-art language model, specifically OpenAI's GPT 3.5 and 4, into the constraint component construction process. This hybrid approach merges the reliability of rule-based detection with the flexibility of language model-based construction. A novel methodological approach was developed to assess precision and recall quantitatively within the context of constraint extraction, complemented by a qualitative analysis. Preliminary findings indicate a promising capacity for constraint reconstruction, albeit contingent on human expertise to ensure accuracy and reliability.

Nonetheless, this approach is not without limitations. The detection of enumeration items necessitates a specific format reliant on line breaks, restricting the scope of identifiable enumeration items. The development of use case-specific rules for constraint identification may inadvertently generate false positives in novel datasets. Moreover, the system's current design only recognizes inequalities when a numerical token is referenced, limiting its ability to correctly categorize inequality constraints using only symbolic representations. Complex relationships extending across multiple sentences, such as those encountered in synthetic datasets like COFFEE, are only partially addressed by the rule-based methodology. While the infrastructure for future expansion has been established, the extraction of process steps was not incorporated into this study. Additionally, the quantitative evaluation's robustness is constrained by the probabilistic nature of the POS tagging and LLM responses, along with a limited capacity for semantic comparison of embeddings.

Future work will focus on a comprehensive analysis of general patterns versus use-case-specific patterns to enhance understanding of the pipeline's scalability. Enhancing the robustness of quantitative evaluations through averaging across multiple iterations represents another avenue for advancement. Furthermore, the integration of an LLM-based process step identifier, coupled with constraint data, offers the potential for aligning constraints with corresponding process steps and conducting sanity checks based on these associations.

Source Code Availability

The source code developed as part of this research is publicly accessible and provided under the MIT License. It can be found on GitHub at the following URL: https://github.com/holzhauerL/nlp-rule-extraction.

Acknowledgments

I would like to thank Catherine Sai for her supervision and mentoring and Jasper Schmieding for his help in collecting the data and creating the gold standard.

References

[1] I. S. Bajwa, M. G. Lee, and B. Bordbar, "Sbvr business rules generation from natural language specification," in *AAAI Spring Symposium: AI for Business Agility*, 2011.

[2] C. Sai, K. Winter, E. Fernanda, and S. Rinderle-Ma, "Detecting deviations between external and internal regulatory requirements for improved process compliance assessment," in *International Conference on Advanced Information Systems Engineering*, 2023.

[3] K. Winter and S. Rinderle-Ma, "Deriving and combining mixed graphs from regulatory documents based on constraint relations," pp. 430–445, 2019.

[4] M. Honnibal, I. Montani, S. V. Landeghem, and A. Boyd, "spacy: Industrial-strength natural language processing in python," 2020.

[5] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, 11 2019.

[6] "Stock Corporation Act of 6 September 1965 (Federal Law Gazette I, p. 1089), as last amended by Article 7 of the Act of 22 February 2023 (Federal Law Gazette 2023 I no. 51)." https://www.gesetze-im-internet.de/englisch_aktg/englisch_aktg.html. Accessed: Jan 15, 2024.

[7] "UNFCCC CDM, AR-AM0014: Afforestation and reforestation of degraded mangrove habitats — Version 3.0." https://cdm.unfccc.int/methodologies/DB/KMH6O8T6RL3P5XKNBQE2N359QG7KOE. Accessed: Jan 15, 2024.

[8] "UNFCCC CDM, AR-AMS0007: Afforestation and reforestation project activities implemented on lands other than wetlands — Version 3.1." https://cdm.unfccc.int/methodologies/DB/J6ZHLX1C3AEMSZ52PWIII6D2AOJZUB. Accessed: Jan 15, 2024.

[9] "UNFCCC CDM, AMS-III.U.: Cable Cars for Mass Rapid Transit System (MRTS) — Version 2.0." https://cdm.unfccc.int/methodologies/DB/I7O8EX3R0PA22GNGBJMH2FHCOIL03L. Accessed: Jan 15, 2024.

[10] "UNFCCC CDM, AMS-II.Q.: Energy efficiency and/or energy supply projects in commercial buildings — Version 1.0." https://cdm.unfccc.int/methodologies/DB/YCL1T3NURPHKSHBSR8TIHC2T543HTQ. Accessed: Jan 15, 2024.

[11] "UNFCCC CDM, AMS-I.A.: Electricity generation by the user — Version 19.0." https://cdm.unfccc.int/methodologies/DB/1TIFADHWTMIW25TAL778RLEFJ6AWBB. Accessed: Nov 11, 2023.

[12] E. Bre, “Repo for nlp bpm datasets.” https://github.com/EduardoBre/nlp-bpm-data/tree/main. Accessed: Nov 12, 2023.

[13] “Patent Act as published on 16 December 1980 (Federal Law Gazette 1981 I, p. 1), as last amended by Article 1 of the Act of 30 August 2021 (Federal Law Gazette I, p. 4074).” https://www.gesetze-im-internet.de/englisch_patg/englisch_patg.html. Accessed: Jan 15, 2024.

[14] D. J. Ladani and N. P. DESAI, “Stopword identification and removal techniques on tc and ir applications: A survey,” *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)*, pp. 466–472, 2020.

[15] J. K. Raulji and J. R. Saini, “Stop-word removal algorithm and its implementation for sanskrit language,” *International Journal of Computer Applications*, vol. 150, pp. 15–17, 2016.

[16] I. Glaser, S. Moser, and F. Matthes, “Sentence boundary detection in german legal documents,” pp. 812–821, 2021.

[17] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, “Language models are few-shot learners,” *CoRR*, vol. abs/2005.14165, 2020.

[18] J. White, Q. Fu, S. Hays, M. Sandborn, C. Olea, H. Gilbert, A. Elnashar, J. Spencer-Smith, and D. C. Schmidt, “A prompt pattern catalog to enhance prompt engineering with chatgpt,” 2023.

A Appendix

Boolean constraint items

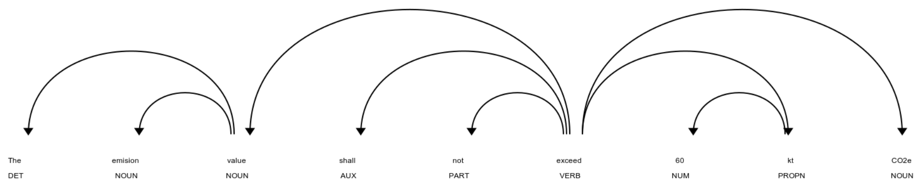
```
"role": "system", "content": "You
are a linguist. Your task is to fit
the output in a compact way into the
placeholder X while preserving the
formatting and overall template that
I provide. Keep the output very
brief, only using nouns, verbs and
adjectives. Use _ to connect the
parts of the output on each side of
the equation."
"role": "user", "content": "This
is the template: X == True.
Extract X from the following text:
land subject to project activity be
degrade mangrove habitat"
"role": "assistant", "content":
"degraded_mangrove_habitat == True"
"role": "user", "content": "This
is the template: X " + operator +
right part + " . Extract X from the
following text: " + context
```

Equality constraint items

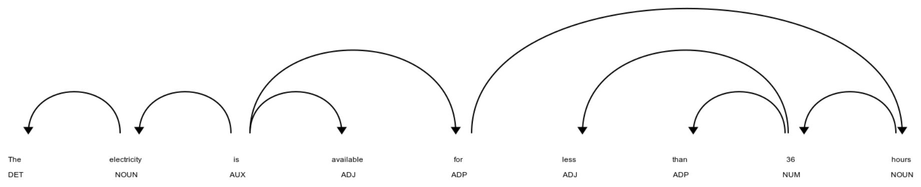
```
"role": "system", "content": "You
are a linguist. Your task is to fit
the output in a compact way into the
placeholder X while preserving the
formatting and overall template that
I provide. Keep the output very
brief, only using nouns, verbs and
adjectives. Use _ to connect the
parts of the output on each side of
the equation."
"role": "user", "content": "This
is the template: X == False.
Extract X from the following text:
not apply to large - scale a / r CDM
project activity"
"role": "assistant", "content":
"large_scale == False"
"role": "user", "content": "This
is the template: X " + operator +
right part + " . Extract X from the
following text: " + context
```

Inequality constraint items

```
"role": "system", "content": "You
are a linguist. Your task is to fit
the output in a compact way into the
placeholder X while preserving the
formatting and overall template that
I provide. Keep the output very
brief, only using nouns, verbs and
adjectives. Use _ to connect the
parts of the output on each side of
the equation."
"role": "user", "content": "This
is the template: X <= 15. Extract
X from the following text: to
qualify as a small - scale project
, total output of modify or retrofit
unit shall not exceed limit of 15
MW"
"role": "assistant", "content":
"total_output_modify_or_retrofit
<= 15"
"role": "user", "content": "This
is the template: X " + operator +
right part + " . Extract X from the
following text: " + context
```



(a) Sentence 1



(b) Sentence 2

Figure 4: Examples for sentences with Inequality constraint items

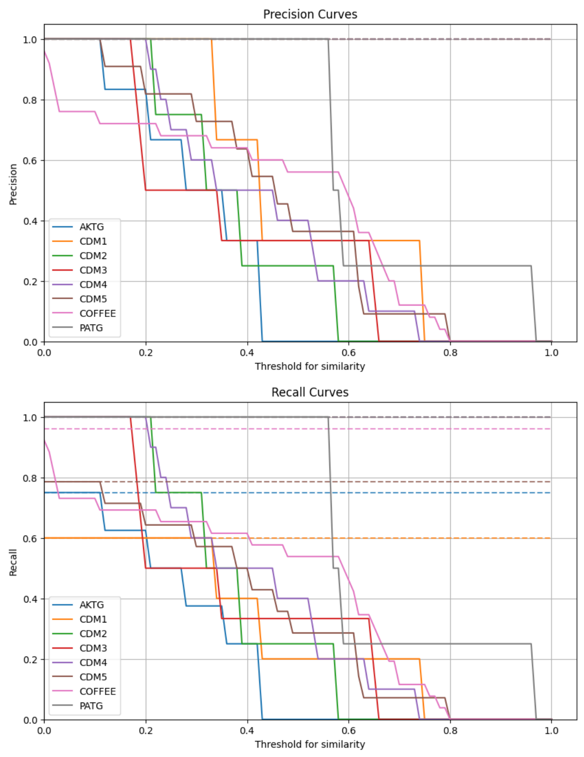


Figure 5: Precision and recall curves using `gpt-3.5-turbo-0125` component building with a variable similarity threshold θ. Dotted lines indicated the maximal achievable precision/recall ($\theta = 0$).

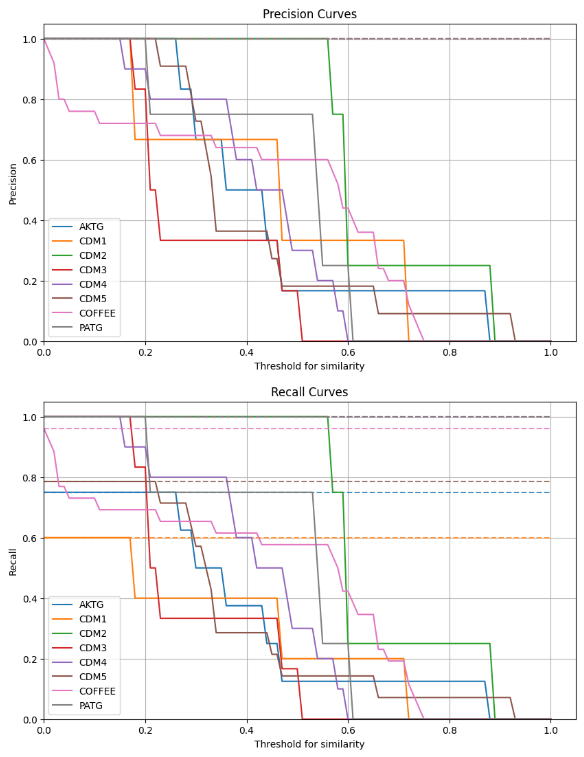


Figure 6: Precision and recall curves using `gpt-4-1106-preview` component building with a variable similarity threshold θ. Dotted lines indicated the maximal achievable precision/recall ($\theta = 0$).