Summary: Rule Extraction from Pharmaceutical Documents for Automated Consistency Checks on Clinical Trial Databases

Christian Fillip Pinderup Nielsen and Magnus Olesen

Summarised: 10-06-2024

Nielsen and Olesen's paper [3] address the challenge of automated consistency checking (ACC) on clinical trial databases within the pharmaceutical domain. Clinical trials are essential for pharmaceutical companies to get regulatory approval for new drugs, and the trial databases must comply with numerous regulations to get this approval. This process is typically labor-intensive and costly due to the complexity and volume of regulatory texts. While state-of-the-art software tools exist to automate the consistency checking process, they are manually implemented.

The authors therefore address the gap of automating rule extraction from regulatory documents and serializing them into machine-readable templates needed for automated consistency checking. The aim of the paper reads: *How can NLP algorithm(s) be used to extract regulatory rules from pharmaceutical documents and serialize them into CDISC conformance rules used for automated consistency checks?* They break down this question into three subproblems: identifying rules within regulatory documents, classifying these rules into relevant categories, and extracting necessary data elements from the rules for serialization.

The research focuses on the domain Study Data Tabulation Model (SDTM) and draws inspiration from the software tool CDISC Open Rule Engine (CORE) [2]. The authors construct an in-domain dataset from the SDTM Implementation Guide (SDTMIG) v3.4, preprocessing PDF sentences, and using ground-truth labels from the CDISC Library API [1]. Various machine learning models and text embedding methods are experimented with. Sentence classification is tackled using a Support Vector Machine (SVM) with TF-IDF embeddings, achieving an F2-score of 0.79, favoring recall over precision. For operator classification, a K-Nearest Neighbors (KNN) classifier with TF-IDF embeddings is used, achieving an F2-micro score of 0.71. Named entity recognition (NER) is addressed using a fine-tuned version of LegalBERT, which significantly outperformed a naive model, achieving an F2-score of 0.69 for extracting elements. Lastly, the outputs of these models are used to generate simple CDISC conformance rules. These experiments demonstrate that the proposed NLP pipeline effectively identifies and classifies rules and extracts relevant elements, showing the potential for automating the generation of CDISC conformance rules.

The proposed NLP pipeline demonstrates a feasible approach to automating rule extraction and serialization in the pharmaceutical domain. However, the process is not without limitations. The dataset was limited in size and variety, affecting model training and performance. The quality of ground-truth labels was inconsistent, and complex rules involving nested conditions were not fully addressed by the current pipeline. Despite these challenges, the study showcases significant progress toward efficient and accurate ACC in clinical trial data management.

Future research should focus on constructing a comprehensive in-domain dataset, enhancing data preprocessing steps, and improving the quality of ground-truth data. Additionally, investigating advanced methods for handling complex rules and incorporating

latent element extraction could further enhance the system's robustness. Addressing these areas could lead to more accurate and reliable automated consistency checks, ultimately improving the efficiency of regulatory compliance in the pharmaceutical industry.

In conclusion, the paper presents a promising approach to automating rule extraction and serialization using NLP techniques. By breaking down the problem into manageable subproblems and addressing each with tailored machine-learning models, the authors demonstrate the potential for automating the compliance process in clinical trial data management.

References

[1] CDISC. *CDISC Library API Documentation*. 2023. URL: `https://www.cdisc.org/cdisc-library/api-documentation#/Rule/api.products.rule.get` (visited on 10/24/2023).

[2] CDISC. *Core*. 2023. URL: `https://www.cdisc.org/core` (visited on 10/24/2023).

[3] Christian Fillip Pinderup Nielsen and Magnus Olesen. *Rule Extraction from Pharmaceutical Documents for Automated Consistency Checks on Clinical Trial Databases*. 2024. URL: `https://projekter.aau.dk/projekter/en/`.

Rule Extraction from Pharmaceutical Documents for Automated Consistency Checks on Clinical Trial Databases

Christian Fillip Pinderup Nielsen
Aalborg University
The Department of Computer Science
Aalborg Øst, Denmark
cfpn19@student.aau.dk

Magnus Olesen
Aalborg University
The Department of Computer Science
Aalborg Øst, Denmark
molese19@student.aau.dk

Abstract

For pharmaceutical companies to get new drugs to market, they first must get clinical studies approved. This entails following rigid rules defined in large regulatory documents. This is both a costly and time-intensive process when done manually. The field of automated consistency checking (ACC) can assist in automating this process.

As regulatory documents are large, complex, and contain rich natural language, implementing ACC solutions is complex. However, natural language processing (NLP) methods have become increasingly powerful in recent years, providing a better use case for ACC.

Thus, this paper investigates ACC in the pharmaceutical domain in collaboration with Novo Nordisk. The paper explores the problem of ACC by dividing it into multiple NLP subproblems and presents a pipeline for ACC. The pipeline consists of identifying sentences representing rules in regulatory documents and extracting relevant data from these rules needed to serialize them into CDISC Core rules.

This paper demonstrates how an in-domain dataset can be constructed needed to implement machine learning models. Using this dataset, we train multiple machine-learning models to solve each subproblem. For the first problem of identifying rules, an SVM classifier using TF-IDF embeddings obtains an F_2 score of 0.79, outperforming other baselines and fine-tuned versions of BERT models. To assign operators to the classified rules, an MLkNN classifier also using TF-IDF embeddings obtains an F_2-micro score of 0.71 . Lastly, to extract elements such as columns and values from the rule sentences, a fine-tuned version of LegalBERT can be used, obtaining an F_2 score of 0.69.

Utilizing the output of these three models, we show that it is possible to generate simple rules, which can be used to implement ACC on clinical trial study databases.

1 Introduction

Pharmaceutical companies like Novo Nordisk need regulatory approval to get new drugs approved. This requires clinical trial study databases to comply with numerous regulations established by the organization CDISC and other regulatory authorities. This compliance process is both labor-intensive and costly when done manually.

Automated consistency checking (ACC) is the field focusing on automating the consistency of data to regulatory rules. The unstructured and rich natural language found in regulatory documents adds to the complexity of ACC. In recent years, several natural language processing (NLP) techniques have advanced and found use cases within ACC.

NLP and ACC have been researched in regulatory fields such as law [35] and architecture, engineering, and construction [34]. Despite its significant impact, ACC within the pharmaceutical industry is largely unexplored.

To address this gap, we collaborate with Novo Nordisk and provide this feasibility study.

In this paper, we focus on two NLP tasks: text classification and named entity recognition.

Text classification (TC) can be used to identify relevant text, such as rules, in regulatory documents. As mentioned, these texts can be complex, and thus, state-of-the-art methods usually solve this task with semantic methods. This can be, incorporating semantic information such as ontologies into machine learning models [29] or utilizing deep learning and contextual embeddings [26].

Named entity recognition (NER) is used to find relevant entities within text. Similarly, state-of-the-art methods for NER also involve deep learning models [36] and the BIO-tagging scheme [38].

Currently, state-of-the-art methods within ACC in the pharmaceutical domain are software tools implemented by the company Pinnacle21 or the open-source tool CDISC Open Rule Engine (CORE). However, this software is developed manually, and the process of identifying sentences as regulatory rules, and converting them into machine-readable language is not automated. Thus, we focus on applying the above-mentioned NLP techniques to regulatory documents to address this problem.

In this thesis paper, we aim to answer the question of

How can NLP algorithm(s) be used to extract regulatory rules from pharmaceutical documents, and serialize them into CDISC conformance rules used for automated consistency checks?

Focusing on the domain of rules for SDTM defined by CDISC, and inspired by the rule conformance checking software CORE, we break down the task into three NLP-related subproblems. At a general level, the first subproblem deals with identifying rules in regulatory documents. The remaining two subproblems deal with extracting data from these rules, which is necessary for serializing them into CDISC conformance rules.

At the technical level, we construct an in-domain dataset consisting of preprocessed PDF sentences from SDTMIG v3.4. Using ground-truth labels from the CDISC API, we set up and experimented with sentence classification, multi-label operator classification, and named entity recognition models, combining them with different text embedding methods. We then use the output of these models to serialize simple CDISC conformance rules.

The proposed pipeline displays how NLP and machine learning can be used in the pharmaceutical domain for ACC.

The paper is organized as follows. Section 2 and 3 provide related work on the NLP topics, and the project background and data, respectively. Section 4 gives formal definitions of the three subproblems. In Section 5 we provide a relevant analysis of the CDISC data concerning the three subproblems. Section 6 presents the paper's methodologies, and in Section 7 we present the results from our experiments on these methodologies. Lastly, Section 8 gives discussion points and future directions ideas on our solutions, and finally, Section 9 presents the conclusion of this paper.

2 Related Work

To answer the problem statement of this paper, we need to investigate and experiment with different natural language processing (NLP) and machine learning methodologies.

The task of extracting rules from unstructured pharmaceutical documents can be handled using text classification (TC) methods. Furthermore, obtaining rule data necessary for serializing the rules can be handled using TC and named entity recognition (NER).

To perform machine learning on any kind of text data, it is first necessary to map the text to some feature space (embeddings). Text embeddings are a fixed-length numerical representation that captures the meaning and content of a given sequence. In recent years, researchers have used deep learning models, such as large language models (LLM), to solve these kinds of tasks [18]. Authors Zhe Zheng et al investigate different methods to obtain text embeddings (both static and contextual) and their performance on TC and NER tasks [37]. They found the contextual embeddings from BERT to produce the best results.

The paper "Classifying Free Texts Into Predefined Sections Using AI in Regulatory Documents: A Case Study with Drug Labeling Documents" [25] investigates drug-label classification on unstructed FDA documents. The study investigates the performance of fine-tuned BERT models and baseline SVM and RF models on binary and multiclass tasks. Both the tasks and the domain are therefore very similar to this study. However, the baseline models in this study are only fed TF-IDF embeddings and not contextual embeddings, making the comparisons unfair. The paper also investigates a somewhat limited multiclass setting of four classes. Lastly, the paper only reports accuracy metrics, a very limited way to evaluate models on these tasks.

In classification settings where the instances of interest are rare, or where many classes are of interest, but only a few are common, it is important to consider methods to deal with imbalance in classes. The paper "Learning from class-imbalanced data: Review of methods and applications" looks into this by reviewing methods of dealing with class imbalance [27]. The authors found that for areas associated with clinical data, resampling-based ensemble classifiers are widely used, but other areas focus more on feature engineering processes. Specifically, they found that the resampling method 'SMOTE' generally performed well [16].

In the construction domain, BERT has also been used successfully for TC and NER tasks. On the TC problem, Zhe Zheng et al investigate many different versions of BERT fine-tuned on their dataset [37]. Similarly, the authors of the ACC system for financial agreements iSyn also fine-tune ten models, mostly BERT, on their own labeled dataset [19]. For the NER problems, Zhou et al [38] showcase how BERT can be used to recognize entities in regulatory construction documents. A commonly used method for tagging entities is the BIO tagging scheme [20, 38].

So far, NER models are limited to extracting only those entities explicitly mentioned in the text. is that only entities explicitly mentioned in the text are caught. Authors Eylon Shoshan and Kira Radinsky approach this problem as latent entity extraction (LEE) [32]. In their paper "Latent Entities Extraction: How to Extract Entities that Do Not Appear in the Text?", the authors group together related entities based on co-occurrence in distinct classification tasks. Then, they use deep learning models to classify and assign entities to the text.

3 Project Background & Data

This background section is mostly repeated from the previous semester's report [28], as the project background has not changed.

3.1 Clinical Trials Overview

The primary objective of clinical trials is to facilitate the safe and effective introduction of new drugs into the market. To achieve this goal, authorities require complete transparency and traceability of all data throughout the process. The process starts with the invention of a new molecule in a lab and progresses through various stages of testing until a drug is

approved for public use. These stages are also referred to as clinical trials.

The clinical trials are split into two stages, preclinical and clinical. The following descriptions of these stages come from [22][p. 8-11].

Preclinical Trial. During the preclinical trial phase, researchers conduct studies exclusively on animals. This allows researchers to; identify any significant concerns before progressing to human trials, examine the drug's effect on pregnant subjects, and evaluate the toxicology [1] of the drug.

Clinical Trial. In the clinical trial stage, the subjects are now human, and the stage is divided into four phases.

- **Phase 1:** Safety, pharmacokinetics, or pharmacodynamics-focused testing on many healthy individuals or a selected population of patients with the disease.
- **Phase 2:** Efficacy and safety testing on patients to determine drug efficiency on diseases and provide a more robust safety profile.
- **Phase 3a:** Same objective as phase 2, but with more in-depth testing on a larger patient group for safety and rare side effects control.
- **Phase 3b:** Tests on specialized groups (e.g., children) parallel to Phase 3a.
- **Phase 4:** Post-market non-interventional trials, where the objective is further to understand the safety and efficacy of the drug, for example, collect data about the people taking the drug and use that for analysis.

To complete the clinical trial phase and move the drug to market, phase *3a* must be completed.

3.2 Data Flow in Clinical Trials

When conducting clinical trials a lot of data has to be collected, stored, and managed. The data utilized in this paper focuses on regulatory rules regarding data concerning phases *2* and *3*. Consequently, this dataflow is shown in Figure 1 and will now be explained.

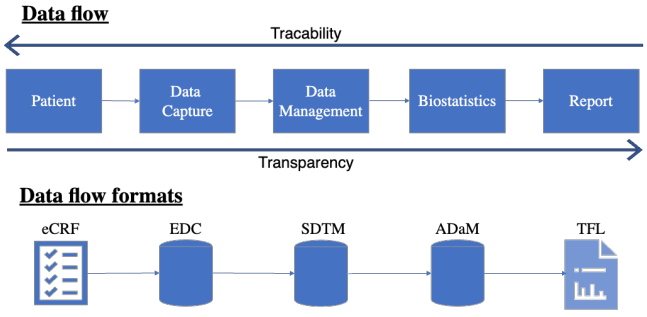


Figure 1. Data flow and relevant formats of the data in clinical trials from patient to final report.

[1] A glossary of terms such as toxicology can be found in Appendix A.1

Each observation or data point can come from multiple sources. One example of a data source is a doctor's visit, where the patient's data is collected in an Electronic Case Report Form (eCRF). Other sources include samples sent directly from labs or electronic data from wearable devices. The raw data from these sources is stored in an Electronic Data Capture (EDC) system.

The EDC data is mapped almost one-to-one, with minimal processing, to a data management system. This system is standardized by the Study Data Tabulation Model (STDTM). Depending on the subject of the observations, they will belong to different domains, which are stored in various datasets. Finally, these datasets can be processed by Biostatistics in an Analysis Data Model (ADaM) and further processed for internal reports in one of the following formats Table, Figure, or Listing (TFL).

Since the SDTM datasets are sent to authorities for clinical trial approval, traceability and transparency are crucial. Any data processing must be well-documented and based on sound motivation. To achieve this, SDTM is rigid in its rules and documentation requirements, thereby achieving transparency during the data flow. With transparency comes traceability, meaning data can be traced back to its source. Additional factors for this is the SDTM-annotated Case Report Form (CRF) document, which is a PDF document containing all mappings from CRF to SDTM, or the *Define-XML* file describing metadata; the origin, format, and possible transformations of every single data item.

Examples of eCRF, SDTM, and ADaM can be found in Appendix A.2.

3.3 SDTM Rules

The rules for SDTM and ADaM are defined by the Clinical Data Interchange Standards Consortium (CDISC), an international non-profit organization. However, CDSIC's rules can be overruled by regulatory bodies such as the U.S. Food and Drug Administration (FDA) or the European Medicines Agency (EMA). Because of this hierarchy, and in some cases contradictions, understanding what rules exist and which to follow in different situations is a complex task.

In this paper, the focus is on CDISC rules regulating SDTM [5]. As of 2024, CDISC's SDTM rules are divided into three concepts:

- The SDTM model. This model is available in PDF and describes a specific version of SDTM.
- The SDTM Implementation Guide (SDTMIG). These guides are also available in PDF and cover guidance on how to implement an SDTM version
- Conformance Rules (CR) covering both SDTM and SDTMIG. These documents are Excel documents and cover data on rules across SDTM and SDTMIG versions.

In Figure 2 we show the current structure of the three types of documents. The figure illustrates how each CR Excel document, can be based on several SDTM and SDTMIG documents. Furthermore, it also illustrates how, for example, *CR v1.1* does not contain rules from *SDTM v2.0* and *SDTMIG v3.4*, while *CR v2.0* covers all the shown documents.

Figure 3 shows two examples of rules concerning SDTM. The first is rules written in a free-text format and the second in a tabular format. Besides free text and tables, the SDTM and SDTMIG documents contain other structures such as images, bullet points, lists, and more.

the scientific subject matter of the data or to its role in the trial. Each domain dataset is distinguished by a unique 2-character code that should be used consistently throughout the submission. This code, which is stored in the SDTM variable named DOMAIN, is used in 4 ways: as the dataset name, as the value of the DOMAIN variable in that dataset, as a prefix for most variable names in that dataset, and as a value in the RDOMAIN variable in relationship tables.

All datasets are structured as flat files with rows representing observations and columns representing variables; each dataset is described by metadata definitions that provide information about the variables used in the dataset. The Define.XML specification provides additional information.

(a) Example of how rules in SDTM can be defined in free-text format [5].

#	Variable Name	Variable Label	Type	Role	Description
1	--MODIFY	Modified Treatment Name	Char	Synonym Qualifier of --TRT	If the value for --TRT is modified for coding purposes, then the modified text is placed here.
2	--DECOD	Standardized Treatment Name	Char	Synonym Qualifier of --TRT	Standardized or dictionary-derived name of the topic variable, --TRT, or the modified topic variable (--MODIFY), if applicable. Equivalent to the generic drug name in WHODrug, or a term in SNOMED, ICD-9, or other published or sponsor-defined dictionaries.
3	--MOOD	Mood	Char	Record Qualifier	Mode or condition of the record (e.g., "SCHEDULED", "PERFORMED").
4	--CAT	Category	Char	Grouping	Used to define a category of topic variable values.

(b) Example of how rules in SDTM can be written in tabular format [5].

Figure 3. Examples of the different structures used in CDISC documents.

The PDF documents can be found on CDISC's website, and the corresponding rulesets for the documents, are released through the CDISC Library API [8] as JSON files.

Lastly, this project also uses the FDA Validator Rules v1.6 ruleset as additional rules for training data. This ruleset contains descriptions of 728 FDA rules, and can be found on the FDA website [21].

CDISC Conformance Rules

The purpose of SDTM rules is to apply a certain industry standard to the clinical trial data in the data management step. Currently, there exists certain automated consistency checking (ACC) software that can apply the given rules on datasets, and check for consistency. One is the commercial software tool Pinnacle 21 [3], and more recently the open-source tool CORE [9].

To formalize the rules, P21 used XML standards [2] and CORE uses JSON [17]. However, the process of identifying sentences representing rules and converting them into machine-readable data in the specified format is currently not an automated process.

This is also the goal of this paper. We provide a feasibility study, going in-depth with the possibility of automatically identifying rules in regulatory documents, and extracting data necessary to formalize them.

As we have restricted this study to focus on SDTM rules by CDISC, we draw inspiration from CORE and its JSON structure. A snippet of this structure is shown in Template 1.

```
{"Rule_id": rule_id,
    { "Cited_Guidance": rule_text },
    { "Check": {
        "name": element,
        "operator": operator,
        "value": element
        }
    }
}
```

Template 1. Simplified JSON template from CORE showing a simple rule.

In this template, we have the unique identifier for the rule (*Rule_id*), the citation which is the reference text of the rule (*Cited_Guidance*), and data on how the rule check should be applied (*Check*). In this project, we focus on the citations, and the data under the 'Check' key, which will be defined later.

The CORE template shows simple rules. These only use a check condition to look for rule consistency. However, there are also rules in CORE that check for more complex logic. The structure of these rules is shown in Template 2

```
{"Rule_id": rule_id,
    { "Cited_Guidance": rule_text },
    { "Check": {
        "all": [
        {
        "name": element,
        "operator": operator,
        "value": element
        },
        {
        "name": element,
        "operator": operator,
        "value": element
        }
    ]}
}}
```

Template 2. Simplified JSON template from CORE showing a complex rule.

In this template, we can see that the rule uses multiple names, operators, and values. Additionally, we have the introduction of the 'all' keyword. This specifies that all checks must be consistent for the data to comply with the rule. The other possible keyword is 'any', indicating that if at least one of the checks fails, the data fails to comply with this rule. These keywords can be nested to multiple degrees.

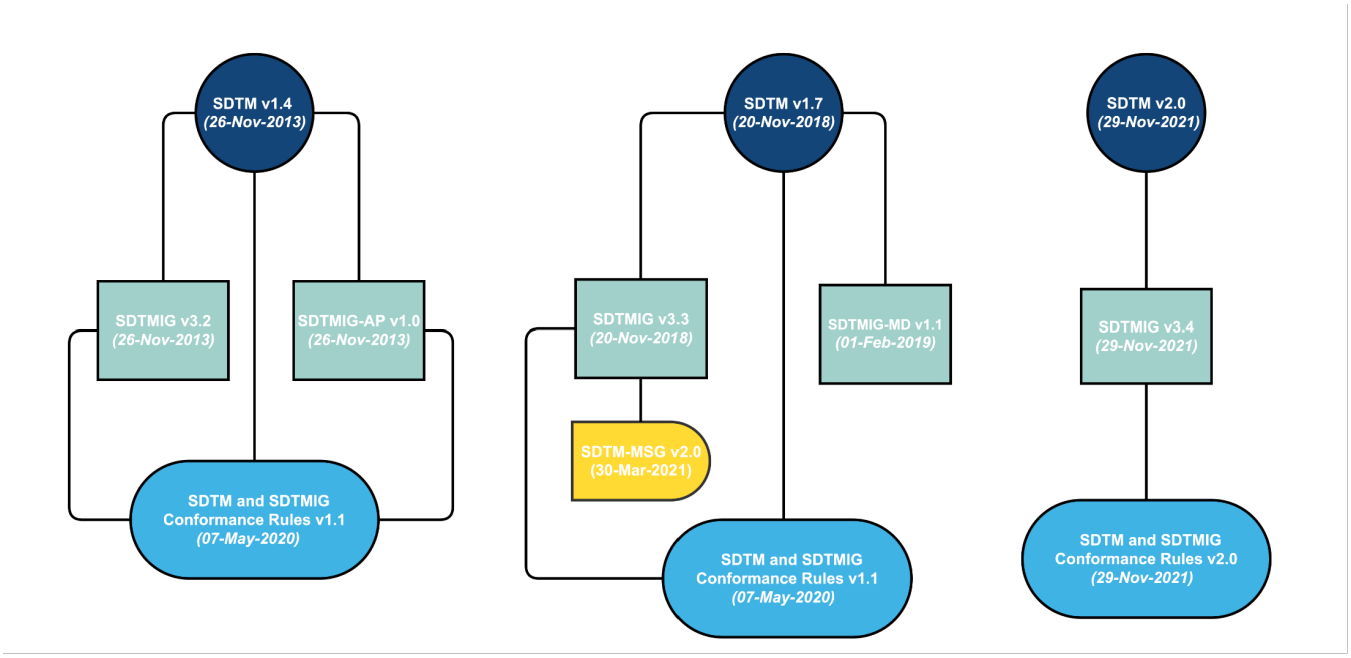


Figure 2. The structure of SDTM, SDTMIG, and Conformance Rules as described by CDISC [10].

4 Problem Definitions

In this section, we break down the aim of this paper, and formally define the problem of extracting rules from regulatory documents and serializing them for automated consistency checks.

This section is based on the previous semester's report [28] but with modifications.

In the following, we consider sentences as ordered set of words from a given vocabulary ending with a full stop. These words are also called dictionary terms, and in this project, we consider the set of English words, numbers, domain abbreviations, and punctuation symbols. Thus we define sentences as follows

Definition 4.1 (Sentence)**.** Let W be a dictionary of terms and S be a sequence of n terms, $|S| = n$, from W. A **sentence** can consist of all possible subsets of W, that is $S \in \mathcal{P}(W)$, where $\mathcal{P}$ represents the powerset.

These sentences can be found in documents. We consider the set of sentences in these documents to be ordered, and thus, we define a document as

Definition 4.2 (Document)**.** Let a **document** d be a sequence of m sentences, such that $d = (S_1, \ldots, S_m)$ and $|d| = m$.

Since we have more than one document, we also define a collection of documents (corpus) as $D = \{d_1, \ldots, d_k\}$.

In each document, a subset of sentences may contain information describing a regulatory rule, $R \subseteq \{S\}$. Rules stipulate how something should take place, or how data should behave in a certain situation. The remaining sentences provide context to rules, are examples of rules, or give otherwise necessary information regarding the domain. The problem then becomes finding a function to identify these rules. That is, for every sentence S in d, classify the sentence as a rule or not by assigning a label $\{0, 1\}$, where 1 represents a rule and 0 is everything not a rule.

Problem 1. Given $\{S\}$, the problem of **sentence classification** require to find a function f:

$$f : \{S\} \rightarrow \{0, 1\}$$

such that, $f(S) = 1$ iff S is a rule and $f(S) = 0$ otherwise.

For all sentences where $f(S) = 1$, we denote this set $\hat{R}$.

Example: *Figure 4 shows a snippet of a regulatory document containing sentences. In this example, we have highlighted three sentences of two rules such that* $d = (S_1, S_2, S_3)$ *and* $R = \{S_2, S_3\}$*. Using a sentence classification model, the goal is to label* $f(S_1) = 0, f(S_2) = 1, f(S_3) = 1$.

Besides classifying sentences as rules, it is also important to capture the essential information from a rule. Three pieces of information jointly describe a rule; what the rule applies to, how the rule checks for conformance, and what it checks for.

CDISC describes how the rule checks for conformance in the rulesets under the *operator* key. Examples of operators include *exists* and *equal_to*. To represent these categorical operators as binary labels we define operators as follows

Definition 4.3 (Operators)**.** Let $O = o_j : j = 1, \ldots, q$ be the set of all operators from the *operator* key. Furthermore, the operator set for a rule $r \in R$ is denoted by the indicator

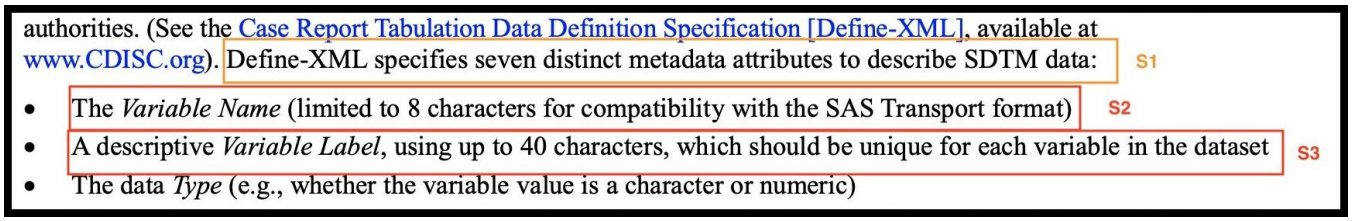


Figure 4. Snippet from a regulatory document containing sentences and rules.

vector Y_r, such that $Y_r = [y_{r1}, y_{r2}, \dots y_{rq}]$ where each y_{rj} is a binary indicator variable defined as follows

$$y_{rj} = \begin{cases} 1 & \text{if } o_j \text{ is in the operator set for rule } r \\ 0 & \text{otherwise} \end{cases}$$

CDISC similarly describes what the rule applies to and what it checks for in the ruleset under the *name* and *value* keys, respectively. respectively. The values found in these keys are complex and can consist of several forms of data types. Typically, the values for *name* are datasets or dataset columns that the rule applies to. *value* on the other hand, can consist of datasets, columns, numbers, or strings. We group values from both of these keys using the term **elements**.

Definition 4.4 (Elements)**.** Let $E = \{e_1, \dots, e_m\}$ be the set of possible **elements**, and $L = \{\text{name}, \text{value}, \text{None}\}$ the **element label** set.

We describe two additional problems to extract operators and elements. The first is to assign operators to the rule approximations $\hat{R}$ from the sentence classification model. In this case, a rule can have multiple operators assigned to it, making this a multilabel classification problem.

Problem 2. Given O and $\hat{R}$, the problem of **multilabel operator classification** require to find a classifier g:

$$g : \hat{R} \rightarrow Y \cup \{\epsilon\}$$

such that Y is the indicator vector representing the operators for $\hat{R}$ and ϵ is introduced as a special symbol to address scenarios where no valid operator from O is assigned.

We use ϵ to effectively capture the case where the output should explicitly indicate the absence of an assignment of operators. This is necesarry in the scenario when a sentence *not* representing a rule is passed from the sentence classification model f into the multilabel classification model g.

The second problem is to extract what elements a rule utilizes in the conformance check. This extraction can be done using named entity recognition (NER). NER is the process of locating and classifying the set of elements E in the rule into the predefined element categories L.

Problem 3. Given a rule $\hat{R}$ of n terms, the set of element labels L, and a set of elements E, the problem of **named entity recognition** requires finding a function h:

$$h : \hat{R} \rightarrow \mathcal{P}(E) \cup \epsilon$$

such that $h(\hat{R}) = E\hat{R}$ is the subset of elements found in the rule $\hat{R}$ that are tagged with the entity types in L. Additionally, ϵ is the set of terms not tagged with entity types from L.

Again, we use ϵ to specify the terms in rule sentences corresponding to no entity types.

Example: *Following the previous example, where sentence S_2 is classified as a rule, the first step is to assign operators to the rule. In this case, a single operator is needed, namely longer_than. In this case, the function $g(S_2)$ should return a* 1 *only for the entry of the longer_than operator, and* 0 *for all other entries. The second step is to extract the relevant elements for this rule. In this case, "Variable Name" should be tagged with the "name" label and "8" the "value" label, such that the two elements extracted from h are* $\{"VariableName", 8\}$.

We can utilize the combined output of classifying document sentences as rules and extracting essential information from each rule to validate the conformance of a rule if it is structured in the specified JSON template previously shown in Template **??**.

Lastly, an important part of this study is to reflect on the usefulness of the proposed solutions to the three defined problems. For the outputs of the models to be useful, and thus the pipeline a success, a level of trust in the system is needed.

To quantify the performance of a given model, we use three evaluation metrics. These metrics are *precision, recall* and F_2-score.

Definition 4.5 (Metrics)**.** Given the number of true positives TP, false negatives FN, and false positives FP from a model output, the evaluation metrics are defined as:

$$\text{Recall} = \frac{TP}{TP+FN}$$

$$\text{Precision} = \frac{TP}{TP+FP}$$

$$F_2 = (1+\beta^2) \cdot \frac{\text{Precision} \cdot \text{Recall}}{(\beta^2 \cdot \text{Precision}) + \text{Recall}}$$

We use the F_2 score, to indicate the importance of recall. Retrieving as many relevant instances as possible is important for the usefulness of the pipeline, even with a tradeoff in precision in mind.

For the multilabel classification problem in Problem 2, the above definitions extend into macro and micro versions [31]. The micro versions aggregate performance across all instances, treating each equally, emphasizing the model's overall accuracy. The macro metrics evaluate performance

for each class independently and then average these scores, ensuring each class is equally represented.

5 Data Analysis

In this section, we present an analysis of the operator classification problem and the named entity recognition task (Section 5.1 and 5.2). Analysis of the sentence classification problem is presented in our previous work [28].

The numbers in this analysis are calculated on the data found in the SDTMIG v3.4 JSON file containing data on the regulatory rules.

5.1 Operator Analysis

Figure 5 shows the frequency of each operator. We can see that a few operators are very frequent, such as *non_empty* and *equal_to*. We can also see that most of the operators are much less frequent, with some operators only appearing once in the dataset, such as *suffix_matches_regex* or *ends_with*.

Based on the operator frequencies, we further investigate the possible data imbalance, with the imbalance metrics MeanIR, MaxIR, and SCUMBLE, which are commonly used in multi-label classification tasks.

To understand MeanIR and MaxIR, we first define the *imbalance ratio per label (IRLbl)* [13].

Definition 5.1. [13] Given the set of operators O, and the true label set Y_r for rule r, IRLbl is defined as the ratio between each operator $o \in O$ and the majority operator o'.

$$IRLbl(o) = \frac{\max_{o' \in O} \left(\sum_{i=1}^{m} h(o', Y_r) \right)}{\sum_{r=1}^{m} h(o, Y_r)}, \quad h(o, Y_r) = \begin{cases} 1 & o \in Y_r \\ 0 & o \notin Y_r \end{cases}$$

Thus, for the most frequent operator, IRLbl equals 1. For other operators, the larger the value, the higher the imbalance level for the operator is.

As IRLbl is calculated for each operator, measures such as MeanIR and MaxIR are also reported. MeanIR is the average of all IRLbl values and is, therefore, the mean imbalance ratio across operators in the dataset. Similarly, MaxIR is the imbalance ratio of the majority operator against the minority operator.

In our case, MeanIR is 8.24 and MaxIR is 27.66. This means that, on average, the most frequent operator is 8.24 times more frequent than other operators. Similarly, the most frequent operator is 27.66 times more frequent than the most rare operators.

Lastly, we report a SCUMBLE score, representing the concurrence among frequent and infrequent operators. This lets us investigate how often rare operators are in the same operator set as frequent operators and is designed to assess whether certain data augmentation methods can be assumed to be appropriate for the dataset [14].

Definition 5.2. [14] SCUMBLE is defined as

$$SCUMBLE(d) = \frac{1}{m} \sum_{i=1}^{m} \left[1 - \frac{1}{IRLbl_i} \left(\prod_{o \in O} IRLbl_{i,o} \right)^{\frac{1}{q}} \right]$$

A small SCUMBLE score denotes a dataset where concurrence among imbalanced labels is rare, and on the other hand, a large score will mean infrequent and frequent labels appear together more often. Charte et. al [15] denote a SCUMBLE larger than 0.1 as high. In our case, the SCUMBLE score equals 3.28, indicating a very high level of concurrence between infrequent and frequent operators within the dataset.

Based on Figure 5, MeanIR, MaxIR, and SCUMBLE values, we can conclude that we are dealing with an imbalanced multi-label dataset. Such imbalance between labels can affect the performance of classification models as they may overfit on the more common operators while underperforming on the rare ones. To handle the skewness of operators in the classification task, we will investigate different data augmentation methods.

5.2 Element Analysis

To investigate the rule elements (*names* and *values*), which are relevant for the named entity extraction problem (Problem 3), we look at the *name* and *value* keys in the SDTMIG v3.4 ruleset. In this ruleset, there are 155 rules.

Figure 6 shows where the names and values occur in the text based on document level and sentence level. It can be seen that even on the document level, some rule elements are not mentioned. If we go further down in the figure and look at the sentence level, the amount of rule elements mentioned significantly drops. On the lowest filter, we can see that only 42% of rules mention *all* names in the rule sentence, and likewise, only 19% for the values.

For the elements not present on the document level, we further analyze how many unique elements this affects.

- Names: 25 out of 139 names (18 %)
- Values: 43 out of 106 values (41 %)

Out of these unique elements, we also analyze how many rules use them, and are thus affected by them not appearing in the document.

- Names: 22 out of 155 rules (22%)
- Values: 46 out of 155 rules (44%)

This will impact Problem 3 as the NER model can only extract elements present in the rule sentences.

A possible solution to this issue, is to set up the NER model as a latent entity extraction model as shown in the paper "Latent Entities Extraction: How to Extract Entities that Do Not Appear in the Text?" [32]. This LEE model involves setting up each element as a classification class, and grouping classes together in distinct tasks, based on the classes co-occurence.

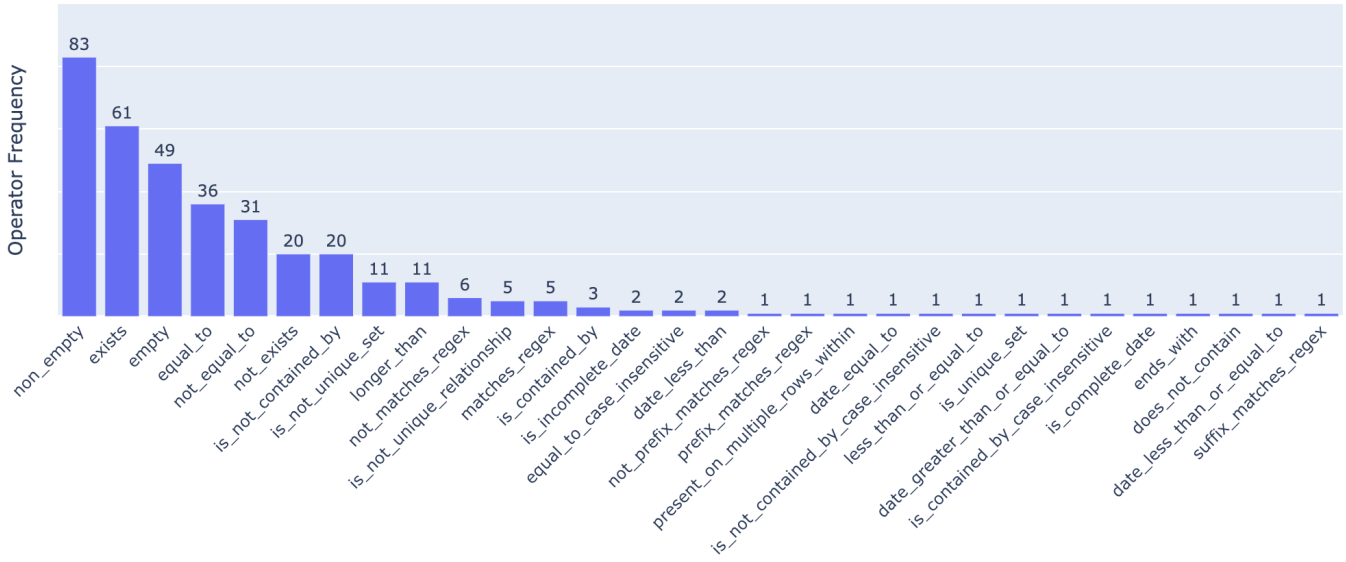


Figure 5. The distribution of operators.

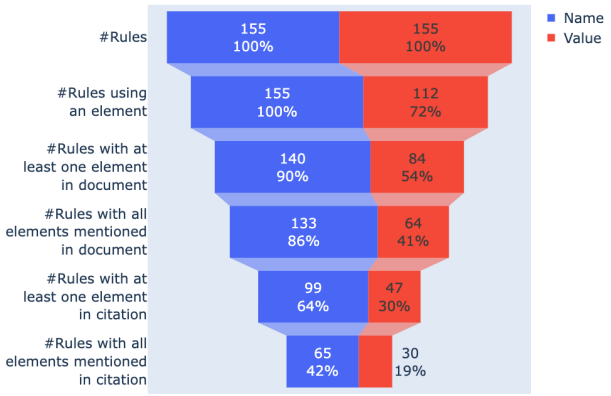


Figure 6. Funnel plot that shows the number of rules at each stage of the element occurrence analysis.

Table 1. The frequency of distinct and unique names and values across all versions of SDTMIG.

	Frequency
Distinct Names	211
Unique Names	50
Distinct Values	112
Unique Values	33

Table 1 shows the number of distinct and unique names and values across all versions of SDTMIG.

This analysis indicates that using the LEE model for our dataset is not applicable, as we would obtain an excessively high number of classes (211+122) even when grouped. Moreover, as one-third of the elements are unique, their infrequent appearances would pose significant challenges to learning these classes properly.

Therefore, we limit this Problem 3 by discarding elements not explicitly mentioned in the sentence. Instead, we focus on learning a NER model, which can only extract elements present in the rule sentences.

6 Problem Solution Methods

In this section, we first present the construction of our SDTM domain dataset consisting of sentences and labels (Section 6.1), and the embedding methods used to obtain numerical features of the sentences (Section 6.2). Then, we present the different methods used to solve the problems of sentence classification (Section 6.3), operator classification (Section 6.4), and named entity recognition (Section 6.5).

6.1 Construction of Dataset

As described in Section 3, the CDISC Library API can extract data for our domain. We use the API to obtain four files; the SDTMIG v3.4 PDF file, and three JSON files containing data regarding all rules for SDTMIG v3.2, v3.3, and v3.4 respectively.

We use these files to construct a dataset from which we can obtain sentence embeddings (features) and rule labels, operators, or elements (classes) to feed into supervised machine learning models in our experiments depending on the problem at hand.

The raw text sentences in the dataset are obtained and labeled from the PDF and JSON files as follows:

- Using Python, we process the SDTMIG v3.4 PDF and extract all relevant sentences. The sentences are thus

comprised of natural language text found in the PDF document.
- The processed sentences from the PDF are manually labeled by us, using ground-truth rule labels found under the 'Cited_Guidance' key in the SDTMIG v3.4 JSON file.
- To obtain more sentences labeled as rules, we incorporate rule sentences directly found in the 'Cited_Guidance' key in the SDTMIG v3.2 and v3.3 JSON files, as well as additional rule sentences found in the FDA Validator Rules v1.6 Excel file [21].

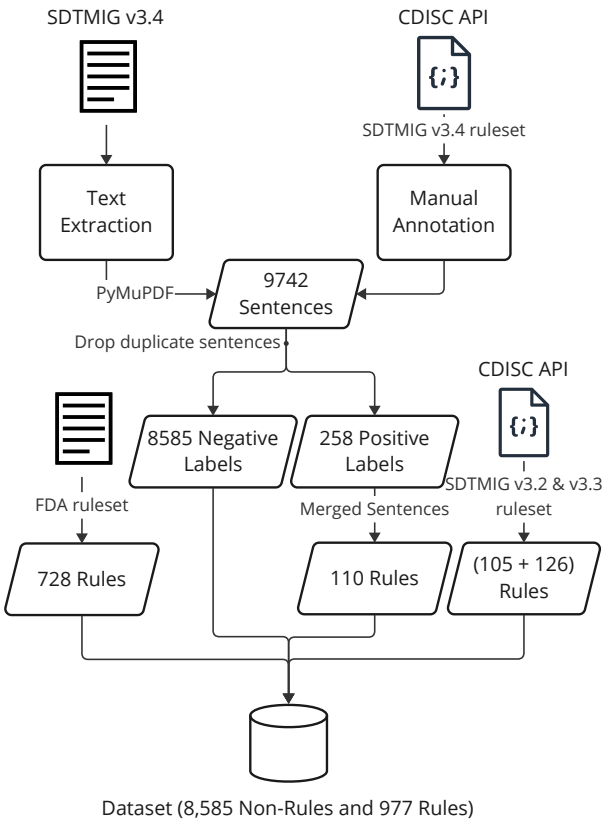


Figure 7. Visualization of the presented data preparation flow. We manually label the processed sentences from SDTMIG v3.4 PDF and augment them with additional rules from other rulesets. This results in a dataset with 8, 585 non-rules and 977 rules after dropping duplicate sentences.

Figure 7 showcases the dataset preparation flow, where we manually label the processed sentences from SDTMIG v3.4 PDF using ground-truth labels from the JSON files and incorporate additional rules from other sources. This process is described more in-depth in our previous work [28].

Similar to the rule labels, the operator and elements labels are also found in the JSON files under the 'Check' key. The FDA Excel file does not contain this data and is thus not included in the data for the problems of operator classification and element extraction (Problems 2 and 3). The operators are represented in the dataset as binary indicator variables, creating an indicator vector for each rule. The elements are represented in the dataset using the BIO-tagging scheme, where each word in the rule sentence is tagged with a label from the label set *O, B-name, I-name, B-value,* and *I-value*. We perform the tagging automatically using Python.

To simulate the process of obtaining a new document to extract data from, we set up the following constraint on the dataset splits: the test rules are the manually labeled sentences from SDTMIG v3.4 as this is the latest version, while the training and validation data are rules from older versions SDTMIG v3.2 and v3.3 (and FDA Validator Rules v1.6 rules for the sentence classification problem).

We ensured there were no duplicate sentences to prevent data leakage between different dataset splits. We defined duplicates based on their TF-IDF embeddings. This method is necessary because sentences in different versions of the SDTMIG often have very minor differences, such as an extra space or comma. Simple text matching would not catch all true duplicates. TF-IDF embeddings effectively identify duplicates. Before splitting the data into training and validation sets, we removed all duplicates from these sets. If a duplicate sentence was found between these sets and the test set, it was kept in the test set, to simulate the process of obtaining a new document, and because the test set is already limited in size.

6.2 Sentence Embedding Methods

To obtain numerical representations (embeddings) of the sentences in our dataset, we tokenize them and embed them using either static or contextual methods.

For the static embeddings, we use the three methods Bag of Words (BoW), Term Frequency-Inverse Document Frequency (TF-IDF) [1], and the pretrained embeddings Word2Vec (W2V) [24].

The BoW methodology employs a frequency-based approach wherein the initial step involves enumerating the unique terms, denoted as $|W|$, present across all m sentences. This enumeration influences the dimensionality of the feature vector associated with each sentence. To mitigate excessively lengthy feature vectors, we incorporate a term-sorting step based on frequency, limiting the maximum length of the feature vector to $2,000$. Following this preprocessing, the frequency of each term within a sentence is counted and inserted into the respective sentence's feature vector. The second static embedding method, TF-IDF captures the importance of terms in a document while keeping in mind that some terms are more frequent than others. The second static embedding method, TF-IDF, is designed to quantify the relevance of terms within a document, taking into account the variable frequency of terms across documents. Like BoW,

the length of the embedding is the number of unique terms, $|W|$, which we sort and limit to 2,000. Then, each sentence's feature vector is populated using the following definition:

Definition 6.1 (TF-IDF)**.** Given the frequency of term t within a document d denoted $tf(t,d)$, and the inverse document frequency defined as

$$idf(t,D) = log(\frac{|D|}{\{d \in D : t \in d\}})$$

Then, TF-IDF is defined as

$$tfidf(t,d,D) = tf(t,d) \cdot idf(t,D)$$

where the denominator is the number of documents the term t appears in.

The last static embedding method, W2V, is a group of models developed by Google and pretrained on large amounts of data. The models have two objectives: given a context window predict the target word or given a target word predict the context words. These objectives are learned using large neural networks, and the hidden layer weights are extracted as word embeddings.

For the contextual embeddings, we use both pretrained and finetuned versions of the LLM BERT, specifically DistilBERT [30] and LegalBERT [12].

Both of these are based on BERT, which is a large language model also developed by Google. BERT is based on the transformer architecture, which is bidirectional and considers both contexts of words from left and right are considered. It is pretrained on different tasks, resulting in contextualized embeddings that capture relationships between words. The text embeddings obtained through the BERT models are 768 dimensional feature vectors.

DistilBERT is a distilled version of BERT designed to be smaller and faster, while still retaining most capabilities of the larger version. LegalBERT is a BERT model finetuned on several large legal datasets.

Lastly, for DistilBERT and LegalBERT, we also use versions that are finetuned on our dataset.

The text embeddings obtained from these embedding methods are the features fed into the different machine learning models trained during our experiments.

6.3 Sentence Classification Method

For the sentence classification problem (Problem 1), we set up multiple binary classifiers to predict whether a sentence represents a rule. The following section on sentence classification is largely a repetition of the work done in the previous report [28].

The binary sentence classifiers we use are Support Vector Machines (SVM), Random Forest (RF), K-Nearest Neighbor (KNN), and Extreme Gradient Boosting (XGB). We use these four classifiers in combination with the static embedding methods, as well as the two BERT models that are *not* finetuned on our dataset. For the two BERT models finetuned on our dataset, we employ a simple linear neural network (NN) as a classification head on top of the BERT architecture. The weights of the classification layer are learned during the model finetuning.

Table 2. The number of non-rules (sentences not labeled as rules) and rules in each of the three dataset splits. The rules in the test set are manually labeled SDTMIG v3.4 rules. The rules in training and validation are extracted from rulesets not SDTMIG v3.4.

Split	Non-rules	Rules
Training	5845	792
Validation	664	74
Test	1100	110

Table 2 shows the size of each split constructed from the dataset for the sentence classification problem.

As each split is imbalanced in the form of non-rules versus rules, we ensure that each model weights each class during training based on the frequency of instances. We give the minority class a weight of 1, and the other class a weight equal to that class' frequency divided by the frequency of the minority class. Thus, the more frequent class gets a weight lower than 1.

6.4 Operator Classification Methods

For the operator classification problem (Problem 2) defined as a multi-label classification task, we employ several classifiers. Additionally, to handle the operator imbalance problem presented in Section 5.1, we use data augmentation techniques.

Data Augmentation Methods. The analysis of operators revealed a high imbalance in the frequency of the different operator sets. To handle this imbalance we focus on oversampling techniques to produce synthetic instances of each class. We do not undersample the majority operator set, as its frequency is only 83.

To generate synthetic instances of the less frequent operator sets, we use the KNN-based approach to oversampling, *Synthetic Minority Over-sampling Technique (SMOTE)* [16]. SMOTE augments our dataset by generating new samples through interpolating of existing minority classes.

Definition 6.2 (SMOTE)**.** Given an embedding x_i, and a sampled neighbor x_{nn} amongst k-nearest neighbors to x_i, a new synthetic embedding x_{new} is defined as

$$x_{new} = x_i + \lambda + (x_{nn} - x_i)$$

where λ is a number in the interval $[0,1]$.

Creating synthetic instances with SMOTE requires a minimum of two examples, and preferably more for diverse interpolations. Due to this requirement, we remove all operators from the dataset that appears less than three times.

Table 3 shows statistics after dropping the low-frequency operators.

Table 3. Statistics of the operator dataset after dropping low-frequency operators.

	Statistic
#Rules	274
#Operators	13
#Operator Sets	50
Average #Operators per Rule	1.6
Average #Rules per Operator	26

Furthermore, before augmenting the dataset with SMOTE, we consider the issue when the operator set consists of a frequent and less frequent operator. In Section 5.1 the co-occurrence of such sets in the whole dataset was measured using SCUMBLE. Handling this issue is important because oversampling a set with a high SCUMBLE value creates more instances of high-frequency operators. Therefore, before applying SMOTE, we use a method called 'REMEDIAL' [15]. This method examines the SCUMBLE value of each instance, and if it is larger than the mean SCUMBLE value for the dataset, the operator set of this instance is decoupled, meaning identical feature vectors are created for each operator in the set.

Example: *We have a rule with operators non_empty and not_matches_regex, a frequent and infrequent operator respectively. This operator set is rare, meaning we want to oversample this set. However, doing so would also create more instances of the frequent operator non_empty. Thus, we decouple the operator set. The rule is represented by an embedding* x_1 *and the label set* $y_1 = \{1, 1\}$*. Decoupling then produces two identical embeddings,* $x_{1,1}$ *with label set* $y_{1,1} = \{1, 0\}$*, and* $x_{1,2}$ *with label set* $y_{1,2} = \{0, 1\}$*. Then, the resample method only resamples the embedding for not_matches_regex* $x_{1,2}$*.*

With this method we reduce the impact of oversampled frequent operators, while still generating synthetic instances of the less frequent operators with SMOTE.

After decoupling the operator classification dataset described in Section 5, the number of rules increases from 244 to 279. Then, after augmenting the dataset with SMOTE, we have 924 rule representations. The dataset imbalance metrics are subsequently lowered with a MeanIR of 3.19, a MaxIR of 5.97, and a SCUMBLE value of 2.5, compared to the previous values of 8.24, 27.66, and 3.28, respectively.

Operator Classification Models. There are multiple ways to handle a multilabel classification problem. In this project, we try two common solutions: adapt traditional classifiers to work with multi-labels or transform the problem into multiple binary classifiers.

For the first class of solutions, we use classifiers KNN, SVM and a Hierarchical ARAM Neural Network (MLARAM) adapted to work for multilabel instances [33]. For the second class, we use classifiers KNN, RF, XGB, and MLP. These classifiers are binary classifiers. We use these binary classifiers to translate the problem into learning a classifier for each class. We investigate three different transformation methods to achieve this: learning a binary classifier for each operator (Binary Relevance), learning binary classifiers for each operator but in a conditioned chain (Classifier Chain), and learning a binary classifier for each operator set (Label Powerset).

For the operator problem, we employ the best-performing sentence embedding method from Problem 1 for a gien classification model. This approach supports the idea of an automated rule extraction framework, where Problem 2 happens as a continuation after Problem 1. Thus, we use the output of the first model as input to the second model.

Operator Datasets. To investigate how the operator classification model learns to identify operators, we construct different datasets for training. These datasets are based on the outputs of the sentence classification model.

Table 4. Each column represents a different dataset, with the size of the training and test split represented in the rows. For the validation split 20% of the training split is used.

	Only Rules	**Non-rules (FP)**	**Non-rules (TN)**
Train	169	190	190
Test	105	117	117

Table 4 shows the three different datasets. The 'only rules´ dataset only contains actual rules, which is also the dataset described in Section 5. This dataset is used to evaluate how the model can learn to assign operators to rules. Next, we have the dataset also containing non-rules representing hard sentences to differentiate from actual rules. These non-rules are the false positives from the first model sampled into both the training and test set. The last dataset instead contains sampled non-rules from the true negatives from the first model, and thus are easier sentences to differentiate from actual rules. For the last two datasets, we construct an additional class, 'no operator', to assign no operators to the sentences.

Based on which dataset is used for training and testing, we can more thoroughly investigate how each operator classification model performs. Models trained on the 'only rules' dataset serve as baselines to investigate the performance on only rule sentences For the dataset containing false positives, we evaluate model performance in a more realistic setting, where false positives are part of the output from the sentence classification model. Assigning the new 'no operator' class to

non-rules is essentially the same as the task for the sentence classification model, which could not be learned at that time. Therefore, we also include the third dataset containing true negatives, as these were correctly identified by the sentence classification model.

6.5 Named Entity Recognition Methods

To identify names and values (elements) mentioned within rule texts, we employ fine-tuned versions of LegalBERT dedicated to token classification. Each token in the rule sentence is tagged with a BIO-tag, and the objective is to accurately classify these tags for tokens within the test set rule sentences.

To tokenize the rule sentences, we use the LegalBERT tokenizer from HuggingFace [12]. When tokenized, the tags are also aligned. If a tagged word is tokenized into multiple tokens, the original tag is retained for the first token while the remaining tokens are assigned the label −100. This label ensures that the model disregards these tokens during both training and inference. The label is also assigned to the special tokens CLS and SEP.

The task of assigning elements to specific rules follows the classification of operators, a multilabel task where each rule may be associated with multiple operators (Template 2). Since elements are linked to rule operators rather than the rule text itself, a more complex challenge arises in mapping extracted elements to the appropriate rule operators. Currently, this aspect is beyond the scope of this project; our focus is confined to rules with a single operator, allowing a direct mapping of elements to that operator.

Thus, the test set for this task consists of rules from SDTMIG v3.4 with a single operator. The remaining rules from SDTMIG v3.4 and the previous versions constitute the training and validation sets. The size of each split is shown in Table 5.

Table 5. Size of NER dataset splits, number of distinct elements, and total number of elements in each split. For the validation split 20% of the training split is used.

	Train	**Test**
#Rules	350	47
#Distinct Names	201	201
#Names	723	81
#Distinct Values	102	102
#Values	634	53

7 Experimental Evaluation and Analysis

In this section, we evaluate and analyze the experimental results. We first present the setup on which the experiments are conducted, and then we present the results from each problem (Sections 7.1, 7.2 and 7.3). Lastly, we present the overall objective of serializing the model outputs into CDISC conformance rules (Section 7.4)

Experimental Setup. The experiments are conducted on a machine with a Tesla T4 GPU, 40GB of RAM, and 10 CPU cores using Python 3.11 [2].

Hyperparameter Tuning. We use hyperparameter tuning to evaluate different model configurations on a validation set. The model configuration that achieves the best performance on the validation set is the model evaluated on the test set. The tuning experiments are implemented using the Python library *Optuna*, which uses a probabilistic model to make educated guesses on new hyperparameter values to try out [4].

Depending on the computational load of the model, we perform either 250 or 100 trials. Any neural network training is conducted for 20 epochs. The hyperparameters tested for all models and their ranges are reported in Appendix A.3.

7.1 Sentence Classification Results

Figure 8 shows boxplots for the F_2 scores across embedding and classifier combinations described in the sentence classification methodology Section 6.3.

From the embedding boxplots, we observe that, except for LegalBERT, the scores across classifiers within the same embedding method are relatively close. Additionally, the pretrained Word2Vec embeddings achieve the lowest scores, while BoW, TF-IDF, and LegalBERT obtain the highest F_2 scores.

For the classifier boxplots, we first highlight that the SVM classifier exhibits a very high variation in performance based on the embedding method used, with F_2 scores ranging from 0.8 to 0.2. Lastly, we note that the NN models on average obtain high F_2 scores. However, it is important to reiterate that these results are based on only fine-tuned versions of DistilBERT and LegalBERT.

Table 6 further shows the precision, recall, and F_2 scores of each embedding and classifier combination, with the four highest F_2 scores highlighted.

For the highlighted methods, Table 6 shows while the recall scores are relatively close around 0.77, the precision varies the most between methods. TF-IDF embeddings with an SVM classifier and a finetuned LegalBERT embedding with NN classifier both obtain precision scores above 0.9, while the two other methods are under 0.77. However, since we report F_2 which prioritizes recall over precision, the fine-tuned LegalBERT model ranks lower than the other three.

It is also worth noting that besides NN with LegalBERT which had a high precision of 0.95, two other combinations achieved higher precision scores: the NN with DistilBERT and the RF with BoW with precision scores of 0.99 and 0.97 respectively.

[2]The full list of required packages can be found here.

Table 6. Precision, recall, and F_2 scores for all classifier/embedding combinations.

		Classifier				
	P/R/F_2	**SVM**	**KNN**	**NN**	**XGB**	**RF**
Embedding	*BoW*	0.88/0.61/0.65	0.90/0.70/0.73		**0.68/0.81/0.78**	0.97/0.55/0.60
	TF-IDF	**0.92/0.76/0.79**	**0.76/0.79/0.78**		0.61/0.75/0.71	0.74/0.60/0.62
	Word2vec	0.42/0.17/0.20	0.43/0.45/0.44		0.42/0.61/0.56	0.40/0.47/0.46
	DistilBERT	0.75/0.48/0.52	0.68/0.56/0.58	0.99/0.60/0.65	0.79/0.57/0.61	0.52/0.45/0.46
	LegalBERT	0.69/0.40/0.44	0.52/0.56/0.55	**0.95/0.74/0.77**	0.65/0.58/0.59	0.46/0.48/0.48

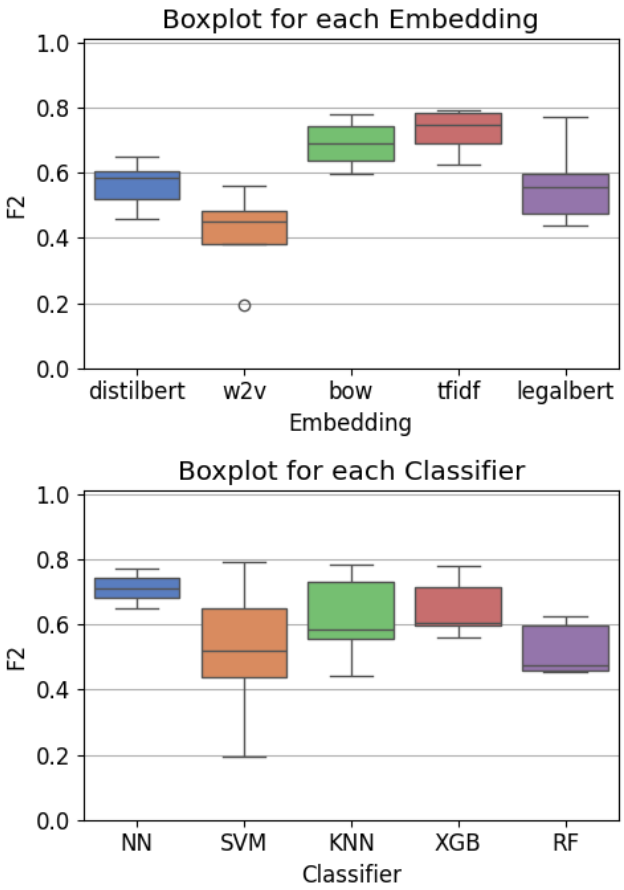


Figure 8. Boxplots for F_2 scores across embedding and classifier combination.

Based on these results, we can conclude that basing the models on pretrained contextual embeddings did not produce better results compared to the simpler static embeddings. However, for the two models in the NN column, where we finetuned the BERT embeddings instead of using only pretrained, performance was much better. This showcases the potential of finetuned contextual embeddings in this domain.

To illustrate how the four highlighted classifiers performed on the test set, we present the confusion matrices for each in Figure 9.

Figure 9a and Figure 9d have very few false positives, while Figure 9b and Figure 9c predict significantly more non-rules as rules. The number of true rules found consistently varies between 80 and 90, and likewise for false negatives in the 20 to 30 range. These observations indicate that some non-rules are hard to distinguish from actual rules, and it is the ability to make this distinction that separates the methods.

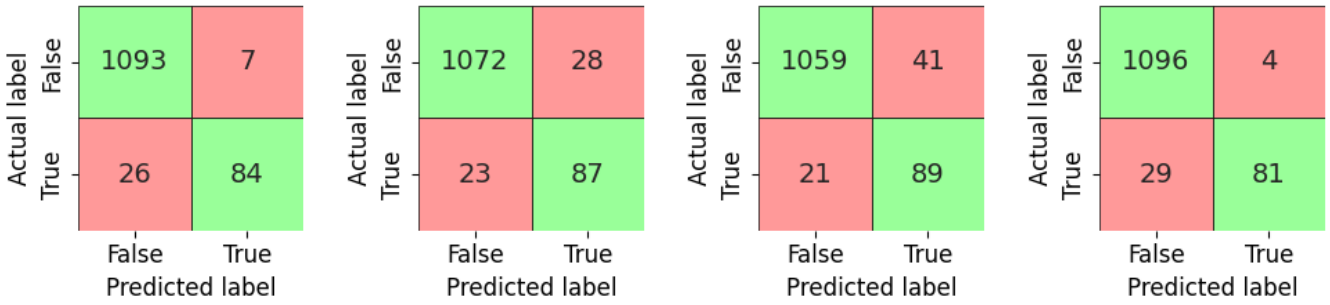
Precision/Recall Tradeoff. Based on the observation that using a unified framework for automated consistency checks, the sentence classification component would ideally achieve as high a recall score as possible. To illustrate the effect of recall versus precision for the four previously presented methods, ROC and Precision-Recall curves are shown in Figure 10.

From the ROC curves in Figure 10 we observe that all models perform better than random guessing with relatively high AUC scores. Of the four models, the KNN with TF-IDF (orange) is worth highlighting as it struggles to achieve a true positive rate higher than ≈ 0.8 without a drastic increase in the false positive rate.

For the Precision-Recall curves, we observe two different tendencies. The SVM with TF-IDF (blue) and NN with LegalBERT (pink) show high recall values for the most part, with the dip in precision scores happening rather late. For the remaining methods XGB with BoW (grey) and KNN with TF-IDF (orange), we can see that they cannot achieve a precision of 1 without having a recall of 0. The same can be observed for high recall values, where a steep drop-off in the precision values happen at recall values of ≈ 0.82. Based on this, we can highlight the advantages of the SVM and LegalBERT models.

Furthermore, based on the trade-off between precision and recall analyzed for the four models, we can conclude that an appropriate model for classifying sentences as rules is the SVM with TF-IDF method. Figure 10 shows that using the SVM model, it is possible to obtain a recall just above 0.9, while still achieving a precision value of 0.5.

Using the output of this model as input to the operator classification task ensures that almost all rules are caught, while keeping the number of false positives as low as possible.



(a) TF-IDF with SVM. **(b)** TF-IDF with KNN. **(c)** BoW with XGB. **(d)** LegalBERT with NN.

Figure 9. Confusion matrices for four best-performing embedding and classifier combinations based on results of F_2 scores on the test set.

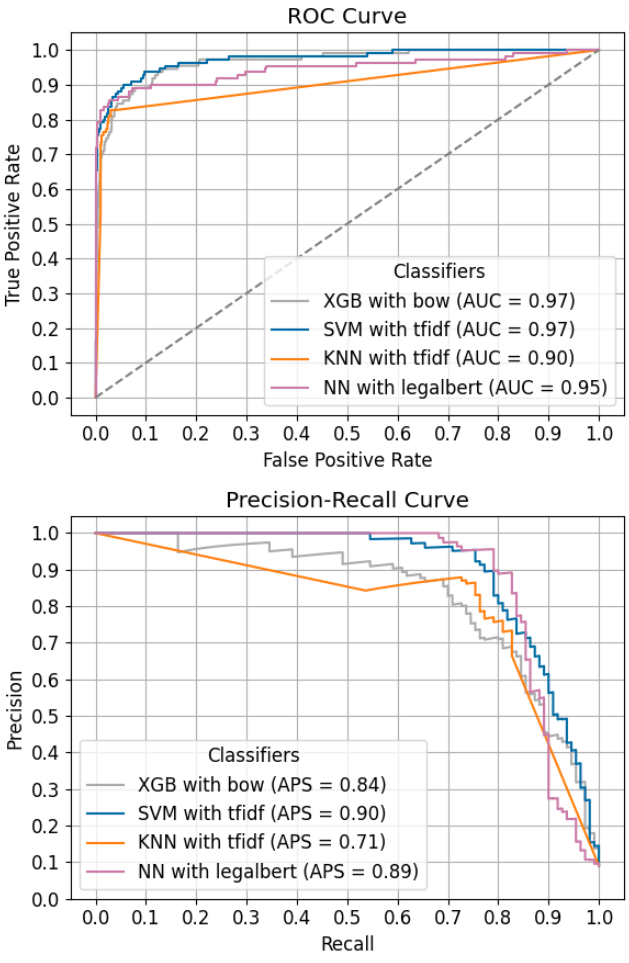


Figure 10. ROC and Precision-Recall curves for the four best performing methods.

7.2 Operator Classification Results

In this section, we present and analyze the findings from the experiments conducted on the operator classification methods described in Section 6.4. We will briefly summarize the methods again. In this experiment section, we investigate the performance of multiple different classifiers on a multi-label classification task. The classifiers are learned on three different datasets; one containing only rules, and two with additional sentences not representing rules. Lastly, we investigate the effect of decoupling the datasets with 'REMEDIAL' and augmenting the datasets with 'SMOTE'.

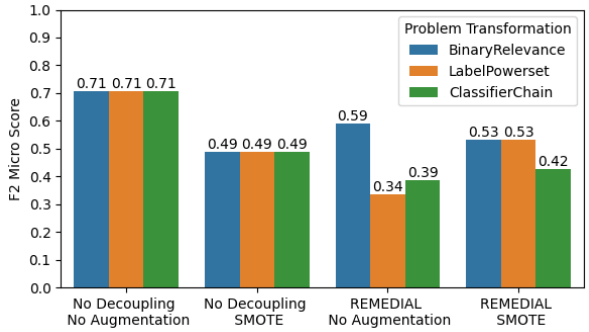
First, we will present and investigate the results of the classification method evaluated on the dataset containing only rules.

Afterwards, we compare those results to those obtained on similar models on the two other datasets which also contain false positives or true negatives from the sentence classification model.

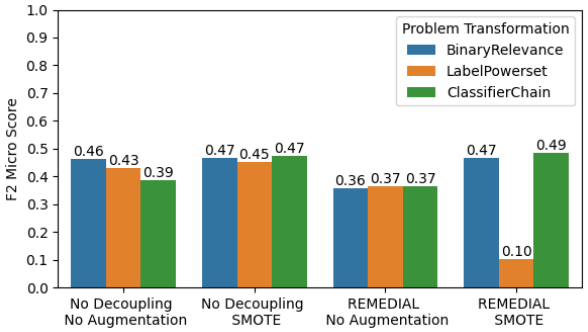
Experiments on Rules. Figure 11 presents the F_2-micro scores for various classification models on the test set of the 'only rules' dataset. Figures 11a, 11b, and 11c are the problem-transformed classifiers, while Figure 11d are the multilabel adapted classifiers.

From Figures 11a, 11b, and 11c it is clear that the primary influence on the metrics is the chosen data augmentation method, while the individual problem transformation methods usually obtain similar results. For the problem transformation classifiers in Figure 11a, KNN obtains the best results on the test set, with the highest F_2 micro score of 0.71 when no data augmentation is applied.

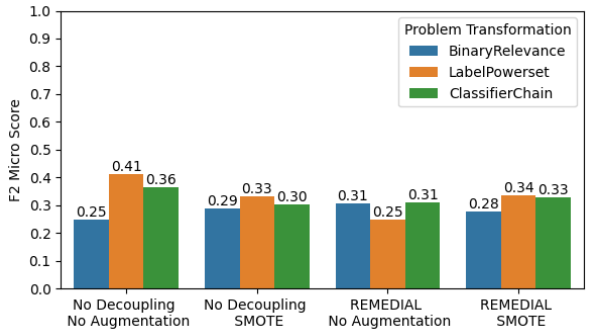
Figure 11d showing results on the adapted classification methods, exhibits more variation between the different classifiers and the different augmentation methods. MLkNN and MLTSVM consistently outperform MLARAM, with the best-performing model being MLkNN with an F_2-micro score of 0.71.



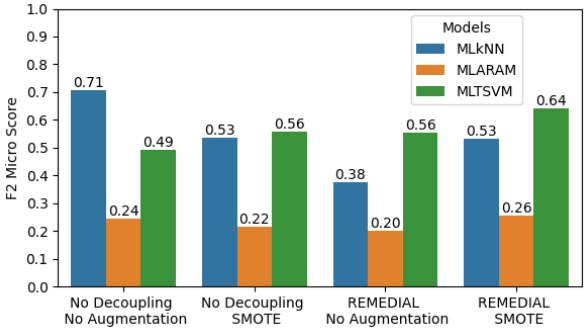
(a) KNN classifiers using TF-IDF embeddings.



(b) XGB classifiers using BoW embeddings.



(c) MLP classifiers using LegalBERT embeddings.



(d) Multilabel classifiers. KNN and SVM use TF-IDF embeddings and MLARAM uses pretrained LegalBERT embeddings.

Figure 11. Barplots of results from test set for operator classifiers. On the x-axis are different variations of data augmentation methods tested, while the y-axis shows the highest obtained F_2-macro score by the classifier. These results are from the dataset 'only rules´.

Contrary to our expectations, we observed that the data augmentation did not lead to better results, compared to doing nothing at all. This is likely due to the dataset imbalance, which can render the benefit of resampling non-existent [14].

Additionally, the results demonstrate that the simpler classifier, KNN, in both its problem-transformed and adapted variants, outperforms all other evaluated classifiers on the test set. Since KNN is the best-performing model, this indicates that for this problem a direct distance computation based on TF-IDF embeddings is sufficient to predict the test classes. Although the MLTSVM model also used TF-IDF embeddings, it is not guaranteed that the clusters of classes are easily linearly separable, as well as the limited training data can make the decision boundaries hard to learn.

To investigate the results more in-depth, we present evaluation metrics on each operator in the 'only rules' dataset in Table 7.

In Table 7 we have separated the operators in the test set based on the frequency. Those with a frequency lower than 10 are greyed out, as we will not draw any conclusions based on the metrics for those operators due to their low frequency.

For the high-frequency operators, the results show that the model is better at classifying operators such as *is_not_unique_set* and *non_empty* compared to *exists* and *empty*.

Furthermore, if we calculate the F_2-micro and macro scores on the high-frequency operators, we get 0.71 and 0.70 respectively. The same calculation done on a model that naively predicts everything as 1, thus getting a recall of 1 for every operator, yields F_2-micro and macro scores of 0.39 and 0.35 respectively.

Thus, we can conclude that classifying operators based on the distance between nearest neighbors in the TF-IDF embedding space is more accurate than a naive model.

The data presented so far is based on the 'only rules' dataset. We will compare these results with those from the two other datasets that sampled non-rules from the sentence classification models output.

Experiments including non-rules. Table 8 compares the MLkNN model on the 'only rules´ dataset with the two best-performing models on the two other datasets. Here, we can see that the 'only rules´ dataset obtains the best results. Across all three datasets, the best-performing models are the

Table 7. Evaluation metrics on each operator from the dataset containing only rules as obtained by the MLkNN classifier. The results are sorted according to the F_2-scores.

Operator	Precision	Recall	F2-Score	Frequency
is_not_unique_set	0.91	0.83	0.85	12
non_empty	0.71	0.89	0.85	36
equal_to	0.65	0.71	0.70	24
not_equal_to	0.58	0.73	0.70	15
is_not_contained_by	0.53	0.75	0.69	12
exists	0.44	0.64	0.58	11
empty	0.47	0.44	0.44	16
not_matches_regex	1.00	1.00	1.00	5
matches_regex	0.60	1.00	0.88	3
is_not_unique_relationship	0.60	0.75	0.71	4
not_exists	0.36	0.80	0.65	5
is_contained_by	1.00	0.50	0.56	4
longer_than	0.60	0.43	0.45	7

Table 8. F_2-macro and F_2-micro scores of the best-performing classifier, embedding, decoupling, and augmentation method for each of the three operator datasets.

Dataset	Model/ Embedding	Decoupling/ Augmentation	Precision-micro	Recall-micro	F_2-macro	F_2-micro
Only rules	MLkNN with TF-IDF	None and None	0.62	0.73	**0.70**	**0.71**
Non-rules (FP)	MLkNN with TF-IDF	None and None	0.60	0.62	0.66	0.67
Non-rules (TN)	MLkNN with TF-IDF	None and None	0.58	0.69	0.65	0.67

KNN models with no decoupling and augmentation done, which further supports the conclusions based on Figure 11 from the previous section.

The confusion matrices in Figure 12 show predictions on the no-operator class by the MLkNN classifiers. This class represents when no operators should be assigned to the sentence for those sentences not representing rules. Figure 12a shows results for the dataset containing true negatives from the sentence classification model and Figure 12b for the dataset containing false positives from the same model.

Comparing the two confusion matrices shows that the dataset containing false positive sentences is harder to classify, indicated by the five instances in Figure 12b incorrectly assigned the no-operator class.

The results for the datasets containing non-rules align with our expectations described in the operator datasets methodology Section 6.4.

To summarize the operator classification experiments, using an MLkNN classifier based on TF-IDF embeddings, we can assign operators to sentences classified as rules by the sentence classification model. Furthermore, the performance of this MLkNN worsens when false positive sentences are included in the test set. However, regardless of the underlying dataset, we can conclude that the model more accurately assigns operators compared to a naive approach.

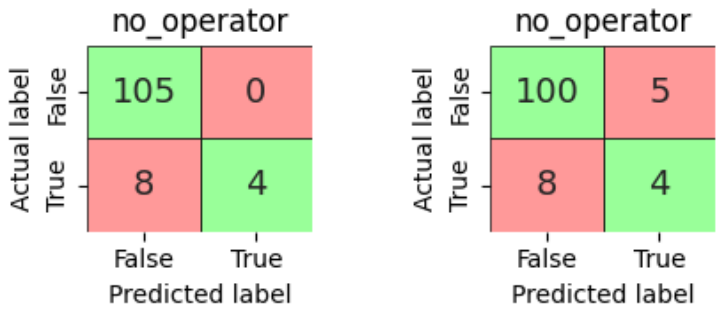
Lastly, we can conclude that while the dataset is very imbalanced, the efforts to incorporate 'REMEDIAL' and 'SMOTE' to provide a more balanced dataset did not yield better model results than the original dataset. We attribute this to the fact that even after performing the data augmentation methods, the dataset remains too imbalanced for these methods to have a positive effect on the model results.

TF-IDF Keyword Analysis. As observed in Table 8, the optimal classification methodology for all operator datasets is achieved using an MLkNN classifier with TF-IDF embeddings. Similarly, the best model to classify whether a sentence represents a rule or not was also an SVM classifier with TF-IDF embeddings.

Given that TF-IDF quantifies term relevance in the documents, the primary factor for our classification tasks appears to be linked to specific terms occurring in the sentences.

To explore this hypothesis, we conduct an in-depth analysis of the TF-IDF embeddings. We base our analysis on the task of operator classification and try to identify keywords for each operator. Subsequently, we use these keywords to assess their impact on correct operator classifications and to determine if misclassifications are correlated with these keywords.

To identify the keywords, we follow these steps:



(a) Dataset with true negatives.

(b) Dataset with false positives.

Figure 12. Confusion matrices on the no operator class for MLkNN classifiers.

1. Extract all embeddings where the label matches a predicted operator.
2. Each embedding includes a TF-IDF value for every term in the sentence, which is used to calculate the average TF-IDF value across embeddings.
3. Select the top k values, where $k = 10$ in our case.
4. These top k values represent the most significant keywords, based on their importance in the entire corpus and their frequency in the relevant sentences.

Table 9. Top ten keywords for the 'empty' operator and their mean TF-IDF embedding values.

Term	Mean TF-IDF
null	0.16
populated	0.11
must	0.10
arm	0.09
comments	0.06
tedur	0.06
teenrl	0.06
armcd	0.06
unplanned	0.05
armnrs	0.05

Table 9 displays the top 10 keywords for the 'empty' operator. Empirically investigating the rules that utilize the 'empty' operator shows that the keyword terms intuitively make sense as to why they are relevant for the 'empty' operator because:

- *null* and *populated* are terms relevant when checking for emptiness.
- *must* and *comments* are general terms that can be expected to be found in rule sentences.
- The specific domain terms *arm, tedur, teenrl, armcd* and *armnrs*, and *unplanned* are also often used in those sentences.

Figure 13a shows the overlap of words among the keywords for each operator. We can see that no operator has a unique keyword set that distinguishes it from all other operators. The largest keyword overlap is between the *is_not_unique_set* and the *equal_to* operators, with seven shared keywords.

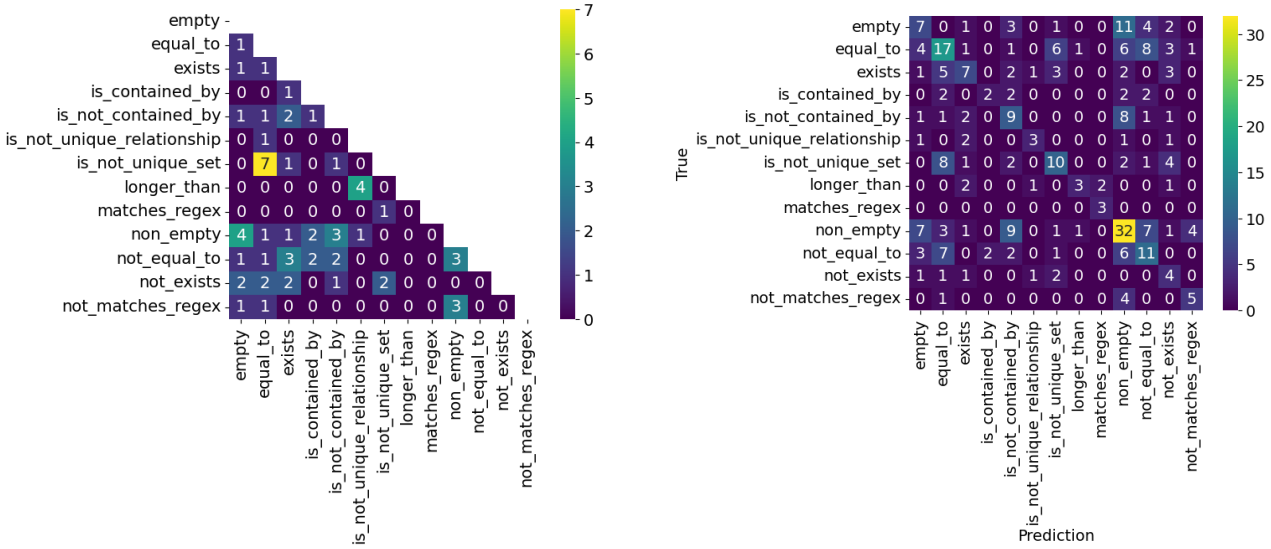
Figure 13b shows a confusion matrix for all operators. From this figure, we can see that rules utilizing the 'empty' operator, are often incorrectly classified with the 'non empty' operator. This can be explained by the overlapping keywords for the two operators, as seen in Figure 13a.

Table 10. The number of wrongly predicted operators, and the number of times the wrong operator's keywords overlap with the correct operator's keywords. The predictions are from the MLkNN classifier on the 'only rules' dataset.

#Misclassifications	68
#Misclassifications with keyword overlap	51
#Misclassifications w/o keyword overlap	17

More generally, Table 10 shows instances where an incorrectly predicted operator shares keywords with the actual operator. Of the 68 observed misclassifications, 51 involve cases where the misclassified operator shares keywords with the correct one. This suggests that the presence of specific keywords is a key factor in assigning operators to rules. When misclassifications occur, they tend to arise for operators sharing overlapping keywords. For the KNN classifier, misclassification thus occurs because rule sentences with different operators but overlapping keywords are in close proximity in the TF-IDF space. The nearest neighbor to base the classification on, can in that case have a different operator class.

These issues occur as TF-IDF embeddings are static embeddings, and do not learn any contextual information or



(a) Overlap of keywords for all operators.

(b) Confusion matrix of all operators.

Figure 13. (a) Heatmap showing the overlap in keywords between operators and po(b) a confusion matrix of all operator classes.

deeper representation of rules. To learn these deeper contextual embeddings, BERT models can be used. Due to limited resources, we did not fine-tune the BERT models for this task, and instead used the pre-trained versions. As shown in the sentence classification experiment results in Section 7.1, when we fine-tuned the BERT models, we also saw a significant improvement in the model results.

In conclusion, based on these observations it is possible to classify operators to rule sentences using TF-IDF embeddings, by effectively capturing the presence of keywords. However, we also show that this method is limited for complex rules sentence, for example, those where the keywords are present in the wrong operator class or are simply missing from the text.

7.3 Named Entity Recognition Results

Table 11. LegalBERT NER results for names, values, and overall averaged.

Element	Precision	Recall	F2-Score	Frequency
B-value	0.60	0.75	0.71	8
B-name	0.65	0.70	0.69	57
Overall	0.64	0.71	0.69	65

Table 11 shows the performance of the LegalBERT model on the named entity recognition task. The model achieves the highest recall and thus F_2-score on the value elements, although there are only eight of these in the test set. For the majority of elements (names), the model achieves an F_2-score of 0.69.

We can compare these results to a naive model. For the naive model, we randomly guess a class using uniform probabilities of 1/3. This is favorable compared to considering the number of instances of each class, as we are only interested in the two minority classes of names and values, and not the O class. Such a naive model would obtain an F_2 score of 0.08, indicating that the learned model is better at distinguishing between elements and other tokens.

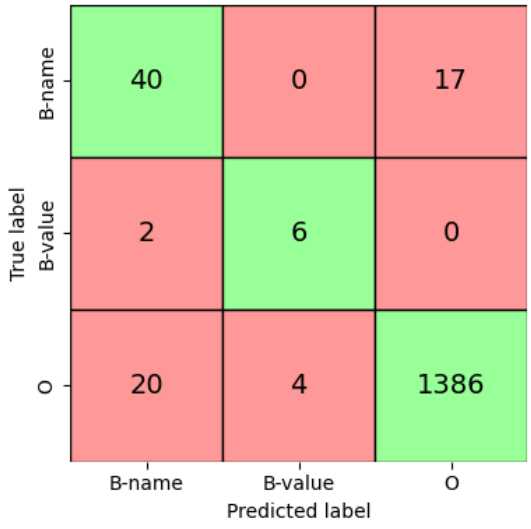


Figure 14. Confusion matrix on the names and values for the LegalBERT NER model.

The confusion matrix in Figure 14 displays the distribution of predictions across the elements. Names are not confused with values but are sometimes wrongly labeled as O. Conversely, the O label is most often wrongly labeled as a name.

We take a deeper look into the misclassifications of the O label in Table 12.

Table 12. The wrongly predicted names and values for the O label.

Token	True Label	Predicted Label
STRAIN	O	B-name
SBSTRAIN	O	B-name
–DECOD	O	B-name
–BODSYS	O	B-name
8	O	B-value
–	O	B-name
SUPPAPFAMH	O	B-name
–STAT	O	B-name
–EVAL	O	B-name
–EVALID	O	B-name
20	O	B-value
–TESTCD	O	B-name
TESTCD	O	B-name
ELEMENT	O	B-name
–PRESP	O	B-name
–ORRES	O	B-name
TSPARM	O	B-name
–TESTCD	O	B-name
IETEST	O	B-name
40-character	O	B-value
200	O	B-value
TSPARM	O	B-name
–STAT	O	B-name
–OCCUR	O	B-name

Table 12 shows that the elements where the true label is O, but has been classified as either a name or value empirically seem to be correct. The elements in the table appear to be domain column names or values, suggesting a data quality issue where the correct label is ambiguous.

Lastly, we investigate if the model can generalize to unseen elements. Unseen elements are those that did not appear in the training data, but is present in the test set. This is important because it is reasonable to assume that new versions of rule documents may have changed or new column names or values to check for. Essentially, any new rule would utilize either a new name or value.

Table 13 presents performance numbers for different elements in the test set, that were *not* seen during training. Although the set of these elements is small, we can based on the recall values conclude that it is possible for the model to correctly classify some of these unseen elements.

Table 13. The frequency and recall of elements *not* in the training set, but present in the test set.

Element	Test frequency	Recall
Name		
–TRT	2	0.5
–TERM	2	1.0
TSPARM	2	1.0
–DETECT	1	0.0
–USCHFL	1	1.0
–CAT	1	1.0
–SCAT	1	1.0
–TEST	1	1.0
–IMPLBL	1	0.0
Value		
TSPARMCD	2	0.0

The results presented for the NER model show that it is possible to use and fine-tune a LLM such as LegalBERT to identify elements within rule sentences. In our experiments, the finetuned model performs best on the name elements, which are also the most frequent elements.

In conclusion, the experiments presented in this Section 7, show that by dividing the problem statement of our thesis into smaller subproblems and solving these individually, we can identify sentences as rules using an SVM model, assign operators to those rules using an MLkNN classifier, and lastly, extract relevant elements from the rules using a fine-tuned LegalBERT classifier. The final step for automated consistency checking is to use the combined output of the models to serialize the data into CDISC conformance rules.

7.4 CDISC Conformance Rule Serialization

In this section, we investigate the possibility of serializing the outputs of each model into CDISC conformance rules.

Following the JSON template from CORE (Code 1), the output from the sentence classification model is serialized into the 'Cited_Guidance' key. The assigned operators and extracted elements are similarly serialized into their respective fields under the 'Check' key.

In this experiment, we focus on serializing simple rules that only utilize a single operator.

Table 14 shows the number of rules with only a single operator that were processed through all steps in the pipeline. Overall for these 55 rules, 28 of them had correctly assigned operators, 17 had correctly assigned names, and 26 had correctly assigned values. Combining all the outputs for each rule, in total 9 of the 55 rules had all outputs correctly predicted. This also means that we can serialize 9 of the simple rules into CDISC conformance rules.

The remaining 46 generated rules differ in the operators, names, and values compared to the corresponding ground-truth CDISC conformance rule. While these rules may not

Table 14. Frequency of matching outputs from each step of the pipeline. Matching means the output corresponds to the ground-truth labels.

	Frequency
Rules	55
Matching Operators	28
Matching Names	17
Matching Values	26
Matching All	**9**

be as expected, their correctness would need to be evaluated by rule experts.

To conclude this section, we have shown that the constructed pipeline for automated consistency checking can generate simple CDISC conformance rules, although only a small fraction are generated as expected.

8 Discussion and Future Direction

In this section, we discuss the insights gained from the experiments performed on the three problems: sentence classification, operator classification, and named entity recognition. We also provide comments on possible future directions for this study.

Embedding Methods

The embedding method that performed best in both the sentence classification and operator classification experiments was TF-IDF embeddings. We see both benefits and drawbacks of this.

One benefit of the TF-IDF embeddings are their explainability: one can analyze the embeddings and identify the terms that impact the classifications the most. This is important as transparency and traceability for the clinical study domain are crucial factors in the approval process. Therefore, the ability to understand the basis of predictions provides an advantage.

For the drawbacks, it is important to highlight that TF-IDF is static frequency-based embeddings, meaning that no deeper meaning or pattern is learned. This can be a problem since the ability to perform classifications is restricted to certain words. Similarly, this superficial understanding of words can also cause incorrect classifications, as we showed in Section 7.2 where certain operators shared keywords.

Sentence Classification

In the sentence classification experiments, we found the best model to be an SVM classifier based on TF-IDF embeddings, outperforming other methods such as fine-tuned BERT models.

This was surprising to us, as we did expect BERT models to perform better at this task given the success of LLMs in other NLP scenarios. However, it is worth discussing whether our experiments on the large language models were comprehensive enough. Due to limited resources, we did not investigate changes in the neural network architecture such as the number of layers, embedding sizes, and more. Of course, the best-performing model being an SVM classifier also has its benefits, as these algorithms are easier to train and are less of a black-box compared to neural networks.

Lastly, we want to highlight that errors from the sentence classification model propagate into the operator classification and NER models in the pipeline. Therefore, the performance of these models are highly dependent on the sentence classification model output. For the next iteration of this study, we therefore see significant value in improving this first model.

Operator Classification

In the operator multilabel classification experiments, we found the best embedding method to also be TF-IDF but with an MLkNN classifier for this task.

As explained in Section 5.1, the classes for this problem are very imbalanced, and thus we investigated augmenting the dataset with 'REMEDIAL' [15] and 'SMOTE' [16].

Our results showed that the classifier's performance on the unmodified dataset was better, compared to augmenting it with the above methods. This was surprising to us, but after calculating the dataset imbalance metrics on the augmented dataset, we saw that the dataset was still relatively imbalanced. Thus, the steps we took to addresss the data imbalance problem were not sufficient.

Instead of further investigating augmentation methods to deal with imbalanced datasets, we would look into collecting more rules using the CORE operators, and ideally, more rules using the less frequent operators. This would help in obtaining a more balanced dataset, and also solve the issue where we had to remove the operators with a frequency of less than three times.

Lastly, due to limited resources, we only had the time to investigate how pretrained versions of BERT performed on this task. We would have liked to investigate how a fine-tuned version of the BERT models with a multilabel classification layer on top would have performed, similarly to the ones we employed for the sentence classification task.

Named Entity Extraction

For the named entity extraction task, we opted to fine-tune a LegalBERT model. This model was chosen due to its pretraining on legal texts, its superior performance compared to DistilBERT in the sentence classification experiments, and its context-aware capabilities combined with pretrained language model knowledge, which we deemed essential for this problem.

This pretrained knowledge was evident in the experiments, as the model was able to correctly extract elements in the test set that were not present in the training set.

However, a notable limitation of this approach is the model's ability to handle latent elements. As discussed in Section 5.2, a significant portion of the elements referenced in the rules are not explicitly mentioned in the text, making them undetectable by the LegalBERT model. The most obvious way to mitigate this limitation is to improve the rule sentences in the data, by explicitly mentioning the relevant elements. However, this involves modifying the writing of the data source which is beyond the scope of this project.

Rule Generation

Looking beyond the individual models used to solve each subproblem and instead focusing on the pipeline, the problem involved converting unstructured natural language text to serialized CDISC conformance rules in CORE format.

We have demonstrated how this can be achieved by serializing the output of each individual model into the JSON format used by CORE. However, the rules we generated in this project are simple rules, as they only use a single operator.

The JSON format used in this project was inspired by CORE and does not fully capture every complexity in their implementation. Therefore, we want to highlight some missing pieces in our pipeline that are needed to fully automate the process of creating CDISC conformance rules.

First, in our pipeline, the sentence classification model identifies rules, and the second and third models then assign operators and extract elements for those rules. As specified, the operator classification model is multilabel. In the case of rules using multiple operators, the serialization of these rules has an additional complexity, in that the extracted elements need to be specified for the correct operator. This is a complexity that our pipeline does not solve.

Second, each CDISC rule has an additional detail to the way rules are checked. In their specification, rules are possibly nested using 'any' or 'all' checks. In practical terms, these checks specify whether only a single consistency check needs to return an error, or if all checks need to fail. These any or all checks can even be nested to multiple degrees. Handling 'any' and 'all' is also something that our pipeline currently does not handle, and would require a fundamental new solution to solve.

Data

In this paper, we specifically focused on data from CDISC regarding SDTM. This restriction naturally influenced the amount of data available to us.

Primarily, the amount of available SDTM rules in the JSON rulesets from the CDISC API was limited. To mitigate this issue, we included rules from the FDA. However, the total number of rules is still relatively low, and the number of rules for the later tasks of operator classification and named entity recognition is even lower, as only the rules from CDISC had these assigned. Limited data directly influences the machine learning models we train during the experiments and the patterns they can learn. Furthermore, we also saw the data used for the operator classification task was very imbalanced.

Additionally, the quality of the labels has also been questionable. For example this was previously shown in Table 12, where we highlighted the inconsistency of how elements are encoded.

Improving the data quality of the ground truth data, further enhancing our data preprocessing steps in extracting data from the PDF, and finding more usable high-quality data are all steps we believe will improve the results from our models.

Ideally, all of these problems would not exist if a large and reliable in-domain dataset for regulatory rules within pharmaceutical documents existed. Such an in-domain dataset could be the basis of scientific experiments within the topic of ACC and beyond.

9 Conclusions

This paper aimed to answer the question of *How can NLP algorithm(s) be used to extract rules from pharmaceutical documents and serialize them into CDISC conformance rules that can be used for automated consistency checks?* by dividing it into three distinct NLP subproblems.

To identify sentences as rules in the SDTMIG v3.4 document, the experiment showed that an appropriate model for this task is an SVM classifier based on TF-IDF embeddings. This model demonstrated a good tradeoff between precision and recall while favoring recall.

To assign operators to the identified rules, we concluded that using TF-IDF embeddings and a KNN approach, proves effective in this task. However, this method depends heavily on specific keywords for the correct assignment of operators. Additionally, we showed how this task becomes more challenging in a realistic setting where errors from the sentence classification model propagate into this model.

For the third problem of extracting elements from the rule sentences, we demonstrated the potential of fine-tuning a large language model for extracting these elements, as it significantly outperformed a naive model. Additionally, the large language model correctly extracted previously unseen elements in the test set.

Regarding the utilized models, this paper shows the potential of fine-tuning large language models. However, as their performance did not surpass TF-IDF embeddings and simpler classification methods, we concluded that either more training data to learn better representations is needed, or a more advanced and newer LLM than BERT is needed.

It was shown how the output of the models can be used to generate simple CDISC conformance rules. However, to generate the more complex rules additional complexity is needed, which the presented pipeline does not handle in the current iteration.

In conclusion, this paper showed how the presented pipeline can be used to obtain simple serialized CDISC conformance rules using NLP algorithms. This approach provides a structured and automated method for rule extraction, which can enhance the efficiency of consistency checks in the pharmaceutical domain.

In practice, the pipeline facilitate the ability to review new regulatory documents and identify all sentences representing a rule. For these rules, the pipeline can suggest which CORE operator to use or indicate if none of the current operators fit. Lastly, the pipeline can go through the rules and identify the relevant elements mentioned in the rule sentence, such as columns or values.

To work on this problem further in the future and expand upon the presented ideas, we recommended constructing an in-domain dataset to provide the basis of additional and more consistent examples. This would further improve the accuracy and robustness of the rule extraction and serialization process.

Acknowledgments

We would like to thank Novo Nordisk for collaborating with us and providing this interesting problem. We would also like to thank the people involved with this project. Specifically, we would like to thank our supervisors from Novo Nordisk Rasmus Stenholt and Henning Pontoppidan Föh for their expert guidance, support, and availability throughout these six months. We would also like to thank our supervisors Daniele Dell'Aglio, Associate Professor at the Department of Computer Science at Aalborg University, and Matteo Lissandrini, Associate Professor at the University of Verona, for their excellent guidance and feedback throughout the last year.

References

[1] 2011. Data Mining. In *Mining of Massive Datasets*, Anand Rajaraman and Jeffrey David Ullman (Eds.). Cambridge University Press, Cambridge, 1–17. https://doi.org/10.1017/CBO9781139058452.002

[2] Pinnacle 21. 2023. *OpenCDISC Validation Framework.* https://www.pinnacle21.com/projects/validator/opencdisc-validation-framework

[3] Pinnacle 21. 2023. *Pinnacle 21.* https://www.pinnacle21.com/

[4] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A Next-generation Hyperparameter Optimization Framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.*

[5] CDISC. 2019. *SDTM v1.8.* https://www.cdisc.org/standards/foundational/sdtm/sdtm-v1-8

[6] CDISC. 2023. *ADaM BDS for average glucose measurements.* https://www.cdisc.org/kb/examples/adam-bds-average-glucose-measurements-113590324

[7] CDISC. 2023. *Adverse Events.* https://www.cdisc.org/kb/ecrf/adverse-events

[8] CDISC. 2023. *CDISC Library API Documentation.* https://www.cdisc.org/cdisc-library/api-documentation#/Rule/api.products.rule.get

[9] CDISC. 2023. *Core.* https://www.cdisc.org/core

[10] CDISC. 2023. *SDTMIG Versions.* https://www.cdisc.org/standards/foundational/sdtmig

[11] CDISC. 2023. *Urine Protein 1.* https://www.cdisc.org/kb/examples/urine-protein-1-29105914

[12] Ilias Chalkidis, Manos Fergadiotis, Prodromos Malakasiotis, Nikolaos Aletras, and Ion Androutsopoulos. 2020. LEGAL-BERT: The Muppets straight out of Law School. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, Trevor Cohn, Yulan He, and Yang Liu (Eds.). Association for Computational Linguistics, Online, 2898–2904. https://doi.org/10.18653/v1/2020.findings-emnlp.261

[13] Francisco Charte, Antonio Rivera, María José del Jesus, and Francisco Herrera. 2013. A First Approach to Deal with Imbalance in Multilabel Datasets. In *Hybrid Artificial Intelligent Systems (Lecture Notes in Computer Science)*, Jeng-Shyang Pan, Marios M. Polycarpou, Michał Woźniak, André C. P. L. F. de Carvalho, Héctor Quintián, and Emilio Corchado (Eds.). Springer, Berlin, Heidelberg, 150–160. https://doi.org/10.1007/978-3-642-40846-5_16

[14] Francisco Charte, Antonio Rivera, María José del Jesus, and Francisco Herrera. 2014. Concurrence among Imbalanced Labels and Its Influence on Multilabel Resampling Algorithms. In *Hybrid Artificial Intelligence Systems (Lecture Notes in Computer Science)*, Marios Polycarpou, André C. P. L. F. de Carvalho, Jeng-Shyang Pan, Michał Woźniak, Héctor Quintian, and Emilio Corchado (Eds.). Springer International Publishing, Cham, 110–121. https://doi.org/10.1007/978-3-319-07617-1_10

[15] Francisco Charte, Antonio Rivera, María José del Jesus, and Francisco Herrera. 2015. Resampling Multilabel Datasets by Decoupling Highly Imbalanced Labels. In *Hybrid Artificial Intelligent Systems*, Enrique Onieva, Igor Santos, Eneko Osaba, Héctor Quintián, and Emilio Corchado (Eds.). Springer International Publishing, Cham, 489–501. https://doi.org/10.1007/978-3-319-19644-2_41

[16] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. 2002. SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research* 16 (June 2002), 321–357. https://doi.org/10.1613/jair.953

[17] CDISC Core. 2023. *cdisc-rules-engine.* https://github.com/cdisc-org/cdisc-rules-engine

[18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. https://doi.org/10.48550/arXiv.1810.04805 arXiv:1810.04805 [cs].

[19] Pengcheng Fang, Zhenhua Zou, Xusheng Xiao, and Zhuotao Liu. 2023. iSyn: Semi-automated Smart Contract Synthesis from Legal Financial Agreements. In *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA 2023).* Association for Computing Machinery, New York, NY, USA, 727–739. https://doi.org/10.1145/3597926.3598091

[20] Dan Feng and Hainan Chen. 2021. A small samples training framework for deep Learning-based automatic information extraction: Case study of construction accident news reports analysis. *Advanced Engineering Informatics* 47 (Jan. 2021), 101256. https://doi.org/10.1016/j.aei.2021.101256

[21] FDA U.S. Food and Drug Administration. 2023. *Study Data Standards Resources.* https://www.fda.gov/industry/fda-data-standards-advisory-board/study-data-standards-resources

[22] The International Council for Harmonisation of Technical Requirements for Pharmaceuticals for Human Use (ICH). 2023. *GENERAL CONSIDERATIONS FOR CLINICAL STUDIES E8(R1).* https://database.ich.org/sites/default/files/ICH_E8-R1_Guideline_Step4_2021_1006.pdf

[23] The International Council for Harmonisation of Technical Requirements for Pharmaceuticals for Human Use (ICH). 2023. *INTEGRATED ADDENDUM TO ICH E6(R1): GUIDELINE FOR GOOD CLINICAL PRACTICE E6(R2).* https://database.ich.org/sites/default/files/E6_R2_Addendum.pdf

[24] Google. 2024. *word2vec.* https://code.google.com/archive/p/word2vec/

[25] Magnus Gray, Joshua Xu, Weida Tong, and Leihong Wu. 2023. Classifying Free Texts Into Predefined Sections Using AI in Regulatory Documents: A Case Study with Drug Labeling Documents. *Chemical Research in Toxicology* 36, 8 (Aug. 2023), 1290–1299. https://doi.org/10.1021/acs.chemrestox.3c00028 Publisher: American Chemical Society.

[26] Maryam Habibi, Leon Weber, Mariana Neves, David Luis Wiegandt, and Ulf Leser. 2017. Deep learning with word embeddings improves biomedical named entity recognition. *Bioinformatics* 33, 14 (July 2017), i37–i48. https://doi.org/10.1093/bioinformatics/btx228

[27] Guo Haixiang, Li Yijing, Jennifer Shang, Gu Mingyun, Huang Yuanyue, and Gong Bing. 2017. Learning from class-imbalanced data: Review of methods and applications. *Expert Systems with Applications* 73 (May 2017), 220–239. https://doi.org/10.1016/j.eswa.2016.12.035

[28] C. Nielsen and M. Olesen. 2023. *Automated Consistency Checks for Pharmaceutical Industry Documents*. https://kbdk-aub.primo.exlibrisgroup.com/permalink/45KBDK_AUB/a7me0f/alma9921651458805762

[29] Dareen M. Salama and Nora M. El-Gohary. 2016. Semantic Text Classification for Supporting Automated Compliance Checking in Construction. *Journal of Computing in Civil Engineering* 30, 1 (Jan. 2016), 04014106. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000301 Publisher: American Society of Civil Engineers.

[30] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR* abs/1910.01108 (2019). arXiv:1910.01108 http://arxiv.org/abs/1910.01108

[31] Scikit-Learn. 2024. *Metrics and scoring: quantifying the quality of predictions*. https://scikit-learn.org/stable/modules/model_evaluation.html#precision-recall-f-measure-metrics

[32] Eylon Shoshan and Kira Radinsky. 2018. Latent Entities Extraction: How to Extract Entities that Do Not Appear in the Text?. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, Anna Korhonen and Ivan Titov (Eds.). Association for Computational Linguistics, Brussels, Belgium, 200–210. https://doi.org/10.18653/v1/K18-1020

[33] P. Szymański and T. Kajdanowicz. 2017. A scikit-based Python environment for performing multi-label classification. *ArXiv e-prints* (Feb. 2017). arXiv:1702.01460 [cs.LG]

[34] Chengke Wu, Xiao Li, Yuanjun Guo, Jun Wang, Zengle Ren, Meng Wang, and Zhile Yang. 2022. Natural language processing for smart construction: Current status and future directions. *Automation in Construction* 134 (Feb. 2022), 104059. https://doi.org/10.1016/j.autcon.2021.104059

[35] Adam Wyner and Wim Peters. 2011. On Rule Extraction from Regulations. *Frontiers in Artificial Intelligence and Applications* 235 (Jan. 2011). https://doi.org/10.3233/978-1-60750-981-3-113

[36] Ruichuan Zhang and Nora El-Gohary. 2021. A deep neural network-based method for deep information extraction using transfer learning strategies to support automated compliance checking. *Automation in Construction* 132 (Dec. 2021), 103834. https://doi.org/10.1016/j.autcon.2021.103834

[37] Zhe Zheng, Xin-Zheng Lu, Ke-Yin Chen, Yu-Cheng Zhou, and Jia-Rui Lin. 2022. Pretrained domain-specific language model for natural language processing tasks in the AEC domain. *Computers in Industry* 142 (Nov. 2022), 103733. https://doi.org/10.1016/j.compind.2022.103733

[38] Yu-Cheng Zhou, Zhe Zheng, Jia-Rui Lin, and Xin-Zheng Lu. 2022. Integrating NLP and context-free grammar for complex rule interpretation towards automated compliance checking. *Computers in Industry* 142 (Nov. 2022), 103746. https://doi.org/10.1016/j.compind.2022.103746

A Appendices

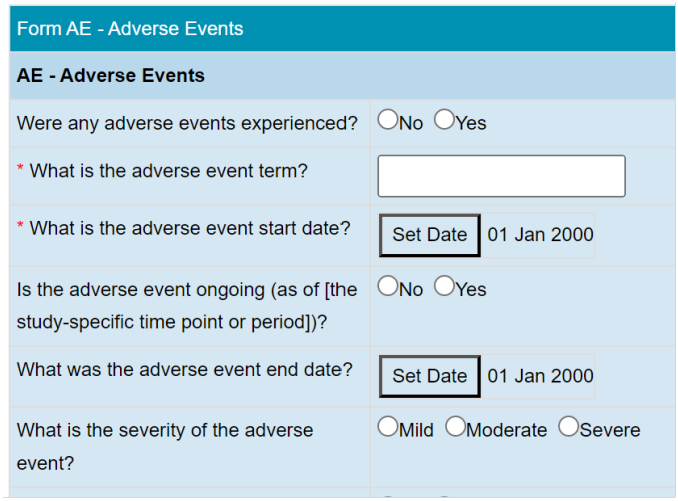
A.1 Glossary

Some definitions originate from [23].

Term	Definition
ADaM	Analysis Data Model. When clinical trial data flows to biostatistics ADaM determines how this data should be processed and structured.
AE	Adverse Effect. An adverse effect is an unintended and harmful reaction to a medical intervention ranging from mild to severe and potentially temporary or permanent. These effects are crucial for assessing patient safety and treatment efficacy.
CDISC	Clinical Data Interchange Standards Consortium. An international non-profit organization that develops standardized data formats for clinical research, enhancing the quality and efficiency of trials and regulatory submissions. CDISC standards are essential for data sharing and are often required by regulatory agencies like the FDA and EMA.
eCRF	Electronic Case Report Form.
Efficacy	Refers to the ability of a drug or treatment to produce a desired effect under controlled conditions.
Pharmacodynamics	Pharmacodynamics deals with the effects of drugs on the body and the mechanisms of their action.
Pharmacokinetics	The study of how substances administered to living things behave. This can include determining how much of the treatment gets into the bloodstream if it gets accumulated anywhere, or determining half-life points.
SDTM	Study Data Tabulation Model. When clinical trial data flows to data management, SDTM determines how this data should be processed and structured.
SDTMIG	Study Data Tabulation Model Implementation Guide. A type of document with in-depth explanations, guides and examples of how SDTM should be implemented.
Toxicology	Focuses on studying the adverse effects of chemicals, including drugs, on living organisms to ensure safety and guide dosage levels.
Traceability	Being able to find the source of the data point at any stage of the data flow
Transparency	Comply with data format standards and processes, and document each data processing step

A.2 Data Examples

Figure 15 shows examples of eCRF, SDTM, and ADaM from CDISC. For the ADaM example, besides the data displayed, it also consists of dataset metadata, variable metadata, and a parameter value list.



Form AE - Adverse Events	
AE - Adverse Events	
Were any adverse events experienced?	○No ○Yes
* What is the adverse event term?	
* What is the adverse event start date?	Set Date 01 Jan 2000
Is the adverse event ongoing (as of [the study-specific time point or period])?	○No ○Yes
What was the adverse event end date?	Set Date 01 Jan 2000
What is the severity of the adverse event?	○Mild ○Moderate ○Severe

(a) Example of an Electronic Case Report Form for adverse events [7].

Row	STUDYID	DOMAIN	USUBJID	LBSEQ	LBREFID	LBSPID	LBTESTCD	LBTEST	LBC
1	XQJ	LB	XQJ05	1	A9990	1	PROT	Protein	URINA
2	XQJ	LB	XQJ05	2	A9991	2	PROT	Protein	URINA
3	XQJ	LB	XQJ05	3	A9991	3	CREAT	Creatinine	URINA
4	XQJ	LB	XQJ05	4	A9991	4	PROTCRT	Protein/Creatinine	URINA
5	XQJ	LB	XQJ06	1	A8880	1	PROT	Protein	URINA

(b) Example of an SDTM database on urine protein patients [11].

SPDEVID	TRT01P	PARAMN	PARAMCD	PARAM	AVISITN	AVISIT	AVAL	BASE	CHG	ABLFL	ASTDT
CGM-001	DRUG A	1	AVGGL24H	Average Glucose (mg/dL) 24 Hours Prior to the End of the Analysis Interval	0	Baseline	113.7	113.7	.	Y	07OCT2016:1

(c) Example of an ADaM database consisting of aggregated data on average glucose measurements [6].

Figure 15. Examples of the different formats used in the data flow

A.3 Model Hyperparameters

Table 15. All hyperparameters that are tuned during an Optuna trial for each classifier. Range specifies what values Optuna can sample from, either a categorical range or a numerical range.

Classifier	Hyperparamter	Range	Note
SVM	Kernel	Linear, Polynomial. Gaussian (RBF) or Sigmoid	
	C	[1e-5, 1e1]	Controls strength of squared L2-penalty. Sampled from log distribution.
	γ	$\frac{1}{\#\text{features} \cdot \text{X.var()}}$ or $\frac{1}{\#\text{features}}$	Kernel coefficient.
RF	max_depth	[2, 100]	Maximum depth of the trees.
	max_features	$\sqrt{\#\text{features}}$ or $\log_2(\#\text{features})$	Maximum features to consider when looking for a split.
	min_leaf	[1, 4]	Minimum number of samples required for a leaf.
	min_samples	[2, 10]	Minimum number of samples required to split a node.
	n_estimators	[50, 500]	Number of trees in the ensemble.
KNN	n_neighbors	[1, 10]	Number of nearest neighbors to consider
	Weights	uniform or distance	How to weight points in the neighborhood in reference to the query point.
	s	[0.1, 2]	Smoothing parameter used in MLkNN
XGB	n_estimators	[50, 500]	Number of boosting rounds
	max_depth	[2, 50]	Maximum tree depth
	eval_metric	error	Evaluate model using binary classification error rate
	Objective	binary:logistic	The learning objective, in this case logistic regression for binary classification.
NN	Optimizer	AdamW, RMSprop or SGD	Different optimization algorithms
	Learning rate	[1e-5, 1e-1]	Parameter used in the optimization algorithms.
	Batch size	4, 6, 8, 16, or 32	≈ 32 was the maximum value for DistilBERT and ≈ 8 for LegalBERT due to limitations in memory on our machine.
MLARAM	Vigilance	[0.80, 0.99]	Responsible for the creation of prototypes during training of the network
	Threshold	[0.001, 0.05]	Controls how many prototypes are used in the prediction
MLTSVM	c_k	[0.03, 8]	Tradeoff between the loss terms in the model
	sor_omega	[0.1, 2]	Smoothing parameter
	Threshold	[0.0000001, 0.005]	Threshold for label predictions
	Lambda	[0.1, 2]	Regularization parameter
	max_iterations	[100, 1000]	Maximum number of iterations
MLP	hidden_layer_sizes	(50), (100), (50, 50), (100, 50)	Number of layers and their sizes
	Activation	identity, logistic, tanh or relu	
	Solver	LBFGS, SGDor AdaM	
	α	[1e-5, 1e-1]	Sampled from log distribution
	Batch size	[4, 128]	
	Learning rate	constant, invscaling or adaptive	
	max_iter	1000	