Law Smells Detection with Machine Learning

Law Smells Detection with Machine Learning

Moriya Dechtiar

A Thesis in the Field of Software Engineering

for the Degree of Master of Liberal Arts in Extension Studies

Harvard University

May 2024

Abstract

While major advancements have been achieved in many fields utilizing artificial intelligence for a variety of tasks, some specialized areas remain difficult to tackle. The legal domain is one such area. It is often said that legal language is a dialect of English and one that requires a Law degree to be fluent in. In this work we examined parallels between software engineering and legal drafting to develop definitions for contract smells, quick indications for potential issues with legal contracts. We created an auto labelled dataset of these contracts smells using engineered prompts and demonstrated how even a small set of human labels can significantly improve auto labelling results with few shots techniques. We demonstrated using bi-directional deep learning models that these contract smells can indeed be successfully detected automatically with high accuracy after further fine tuning BERT as well as LegalBert. This work underscores the feasibility of applying advanced NLP techniques to automate aspects of legal contract review and provides a strong foundation to further develop models for this purpose.

Acknowledgements

I would like to thank my thesis director, Prof. Daniel Martin Katz and my thesis advisor Prof. Hongming Wang, for their guidance and continuous support throughout this work. I am also deeply grateful for my husband's tremendous support during this journey.

Contents

List of Figures

List of Tables

List of Code

Chapter I.

Introduction

In software engineering there is a well known term “code smells” referring to easily detected code problems which could be a good indication for deeper problems in the system itself. In previous work done to port these concepts from software engineering to the legal world (Coupette et al., 2023), they were redefined for the legal context and named “Law Smells”, meaning easily detected issues in laws and regulations that can indicate deeper problems in the law. This work successfully identified several such smells and developed a toolkit to demonstrate how we can detect some of these smells in US laws. This thesis will visit the work done on Law Smells starting with definitions and going all the way to actual detection methods, to explore the law smells that can be applied to contracts and legal agreements in order to alleviate the load of the contract review process. Accomplishing detection of ”contract smells” will make contract reviewing a more automated process and even has the potential to stride in the direction of more standardizes contract drafting.

1.1. Background

1.1.1 Legal Contracts Background

A contract is a legally enforceable agreement between two or more parties that is entered into voluntarily, with the intent to create legal relations. Each party accepts a set of obligations, rights, and benefits established within the contract. Contracts are fundamental instruments within private law and serve as the basis for a vast range of commercial and personal transactions, relationships, and exchanges. In order for the contract to be legally binding it needs to meet several requirements. Some of these core requirements are: one of the parties must make a clear offer that the other parties will be accepting in this agreement , there must be a consideration made on each of the parties to supply some value to the other and that the transaction itself must be legal. The parties themselves must also have the legal capacity and competency to contract, meaning they are of sound mind and a certain minimum age depending on the type of contract. The specific terms within a contract articulate the responsibilities, entitlements, and commitments each party will be bound to. The obligations and benefits of all parties are explicitly stated, and failure to fulfill one's obligations can result in legal consequences imposed by the contract itself or through litigation, such as compensatory damages or court-ordered performance. The terms of a contract are interpreted strictly by the courts, and will be enforced as written unless found to be ambiguous or unconscionable. This is where contract agreements complexity become challenging as one can not cover every possible outcome of the contract. Edge cases, special circumstances and external impacts could be only a few of the hard to tackle clauses. Contracts establish a mechanism by which parties can rely on the force of law to guarantee confidence, equity, and performance in their dealings and exchanges. By voluntarily consenting to the mutual obligations and benefits set out within a contract, individuals, businesses, and organizations are able to engage in risk-taking by planning for the future, developing ongoing relationships, and participating in complex transactions—all with the assurance that legal recourse is available if either party fails to carry out their contractual responsibilities. Contracts thus provide an indispensable foundation for commerce and cooperation in an immense range of human affairs.

1.1.2 Importance of Legal Contracts

Legal contracts are at the heart of human society. They are the basis of instilling trust and eliminating risk in situations and transaction and in that they are in fact one of the biggest driver of economic progress. We can think of agreements as strings that connect people, businesses, organizations and countries. When focusing on the world of commerce we can see how contracts are very much indispensable. The create a framework for trade and felicitate business and how it is conducted between parties. Agreement clauses specifying each party's rights and obligations are placed in order to protect both sides, reduce uncertainty, reduce risk and guarantee the parties interests in a transaction. This provides trust on which parties can build on when collaborating and exchanging knowledge, goods, or any other valuable.

This is key to economic growth and development. The 2016 Nobel prize in the field of economics was awarded to researchers for their work in contract theory, demonstrating just how much contracts are crucial for our society. Looking at this from a social development perspective we can see here also contracts are pivotal in shaping societal behaviours and setting expectations. For example employment agreements are not only binding the employee and employer, they are also setting norms for fair compensation, reasonable work life balance and safe working environment. Furthermore, social stability and justice which are the pillar of democratic societies, rely on contracts as a fundamental mechanism for enforcing rule of law, individual rights and ensuring accountability. This also contributes to reinforcing trust in legal and institutional systems. One important characteristic of contracts is their evolution which happens in tandem with social development, technological development and so they are transforming over time along with us. Today, with the advent of artificial intelligence and machine learning, we stand on the brink of another major shift in contract management and analysis. This research aims to contribute to this evolving landscape by developing a deep learning model to detect law smells in legal contracts, thereby enhancing the efficiency and accuracy of contract analysis.

1.1.3 Current Challenges with Legal Contracts

As legal contracts are such a big part of our society, economic system and law, they are not without challenges. In fact, challenges in properly drafting a contract can be clearly observed through the need for interpretation that occurs after the contract is signed and in effect. Many times this can even require settling unclear contracts clauses disputes in court. Issues with contracts are expected to a certain level, as it is the contract's goal to cover as many possible future outcomes and scenarios as possible, however the future is inherently uncertain and covering all options is non realistic. The law itself, to which a contract is subject to, is not static. Rules and regulations change over time, evolve, and this has an impact on the context in which the contract was created. The above cause challenges both when drafting and interpreting contacts, however we can also describe several other causes that can be minimized with proper tools and procedures. The first comes from language. Any language can be open to different interpretations and sometimes even missing or incorrect punctuation can result in a completely different meaning for a sentence. This is even amplified in the legal language which incorporates the usage of many specific terms, and legal phrases. This can make a legal contract more complex to draft, resulting in many ambiguities. A second cause is the lack of drafting standards in the field in general. Each law firm will tend to have several templates they work with and edit them to fit the purpose of a newly drafted contract. A mistake in this template will persist in all contracts that were based of that template. Also the fact that these templates differ from law firm to another cause additional overhead when analyzing contracts as they can be very different and require much bigger effort to properly assess all the clauses. Lastly, we need to always remember the human factor. More prone to errors can be inexperienced law professionals especially with no guidelines or standards, they have a greater chance to make mistakes when drafting contracts. Also with contract length and complexity increase, the bigger the chance even very experienced law professionals will overlook inconsistencies, gaps or contract vulnerabilities. All these can be addressed and mitigated with software tools to help automate the process of assessing and detecting issues with legal contracts and lead the way to drafting better contracts. We can see the impact of these challenges when summing court cases for clarifying or interpreting contracts as they are accessible through CanLi, a non for profit organization providing web access to court judgments from all Canadian courts. In the past 3 years we can consistently see approximately 10% of court judgments were with regards to contracts and interpretation.

1.2. Software Development and Legal Drafting

1.2.1 Realm Similarities

The creation of both legal agreements and software code share intriguing commonalities, despite seeming disconnected on the surface. A deeper analysis illuminates striking parallels between these two rule-based systems, forming a compelling basis for applying insights on code smells from software engineering to the novel concept of law smells in contractual agreements. Legal contracts and software code both establish intricate rule systems intended to tightly control behaviors and outcomes. Both define the content world it governs, then a set of conditions and outcomes that are based on the branching of those conditions. Even minor oversights can propagate through such fastidious rule-based frameworks, magnifying unintended consequences. This similarity was noted by (Coupette et al., 2023), who characterized both domains as systems of instructions crafted to direct actions towards desired results. Furthermore, the languages involved require specialized expertise. Legal 'legalese' and programming syntax comprise complex vocabularies which only skilled practitioners can fluently employ. Errors in syntax or legal language usage, introduces vulnerabilities, manifesting as software bugs or legal loopholes. Additionally, contracts and code alike must manage multiple possible scenarios. Agreements account for potential disputes, just as programs handle varied inputs. This reality was highlighted by (Coupette et al., 2023), who observed both fields grappling with dynamic situations and abstraction of unknowable future complexities. Most critically, both domains suffer from detectable warning signs of latent defects, analogized as code smells and law smells. While not immediately dysfunctional, such symptoms often betray future complications. As (Coupette et al., 2023) suggested, identifying such contract weaknesses enables proactive quality assurance, much like how code smells flag software vulnerabilities. Given these marked similarities, applying software methods for smell detection to legal agreements shows a lot of promise. This research will develop a model leveraging machine learning to pinpoint law smells, working towards the quantitative contract analysis envisioned by (Coupette et al., 2023) By detecting problematic codifications early and improving drafting quality, this approach can enhance the effectiveness of legal agreements.

1.2.2 Software Development Code Smells

Code smells were initially introduced in the 1990s and became popular following a publication of a book on the topic (Fowler et al., 1999) focusing on code manifestations or patterns that could be good indications to an actual problem with the software system at hand. Below is a short definition for code smells: Long Method: A method that is too long or became too long with time, suggesting that it is doing too many things. Large Class: Similar for long method but referring to a class. A class that is simply too large, often containing many variables and methods. Primitive Obsession: Using primitive data types instead of using simple and small objects for basic tasks (for instance for currency, ranges, special strings for phone numbers, etc.). Long Parameter List: Methods that have more parameters than necessary, making them hard to understand and maintain. Data Clumps: Different sections of the software code that use the same groups of variables, suggesting that they should actually be combined into the same class. Switch Statements: Extensive use of switch statements or sequences of if statements, which could be replaced with polymorphism. Temporary Field: Occurs when a class has a field that is only set in certain circumstances. Such fields are puzzling and error-prone. Divergent Change: Class that is being changed in several different ways for a number of different reasons, suggesting that it has more than one responsibility and context. Shotgun Surgery: Similar to divergent change, but in this case, a change in one class requires additional changes to be done in many other classes. Feature Envy: A method that makes more calls to other class methods than it is making calls internally which is suggesting we may have misplaced functionality. Duplicate Code: When sections of identical code or very close to that, can be seen in more than one place in the software system. Lazy Class/Freeloader: A class that has insignificant responsibility and is not pulling its weight, increasing the cost of maintenance without offering much functionality. Speculative Generality: A class or method that has been created to support anticipated future requirements that never materialize. Inappropriate Intimacy: A class that is using some internal fields and methods of another class excessively despite these are meant to be for internal use. Message Chains: A pattern of method calls that form a chain, which can create a strong coupling between classes. Middle Man: A class that does nothing but delegate to another class, which can be eliminated to simplify the design. Inheritance vs. Delegation: Inappropriate use of inheritance, where delegation might be more appropriate. Data Class: A class with no actual functionality, that in fact only contains data fields and thegetters and setter for accessing them. Refused Bequest: If a subclass does not use the methods and properties inherited from its superclass, inheritance might not be the best approach. Comments: Excessive use of comments to explain complex code, instead of refactoring the code to be more understandable.

1.2.3 Code Smells and Legal Issues

Building on the concept of code smells, and as seen in the work by (Coupette et al., 2023) a mapping was defined between code smells and issues with law, legislation and regulations. The mapping focused on creating parallels between the realms but also defined law smells based on that mapping. Here are the law-smells the work focused on: Duplicated phrase: A phrase in a legal text that repeats in several occurrences. This indicates inefficiencies and create risks of inconsistencies. Long element: An element (e.g. section, chapter) of law containing text that exceeds an absolute length threshold or is an outlier by some relative measure. Indicates lack of structure/abstraction. Ambiguous syntax: Use of conditions or punctuation marks or confusing structure in a way that leaves room for interpretation and can be understood in more than one way. Creates legal uncertainty. Large reference tree: When the structure of the legal element is too complex and contains too many references to other elements. This can be confusing and may require to be traversed to understand it. Increases cognitive load. Natural language obsession: Representation of structured data solely as unstructured natural language text. Impedes analysis and maintenance. In this work we will perform a similar mapping, only under the focus of legal contracts.The focus here of our first step is to define the most common or impactful code smell concepts and apply them to the realm of common contract issues.

1.3. Machine Learning to detect code smells

Identification of code smells early on and automatically is extremely beneficial for software products development and so we can see than there has been work done to use machine learning in order to help with this task. (Gupta et al., 2021) evaluates the effectiveness of natural language processing (NLP) methods for predicting code smells. In this research, 629 open source packages were subjected to machine learning techniques in order to predict eight different code smells. Some of their findings suggest that different datasets, ML techniques, performed better for different types of code smells, indicating that there is no one model fits all approach that could be effective for catching all code smells. The work demonstrates the potential of AI/ML methods for automating code smell detection to aid developers. Another research that was done by (Sharma et al., 2019) was looking at he feasibility of transfer learning code smells using deep learning. This work demonstrated how using RNNs and CNNs for code smell detection is effective and can even be transferred between different code languages. Another approach that was examined is code smell detection using multilabel classification approach by (Guggulothu, 2019). Here researchers were taking into account the level in which code smells can be manifested and created a method-level data set to focus on that. The researchers also redefined code smell detection as a multilabel classification challenge, enabling the detection of multiple code smells within a singular code element. This is a significant shift from the traditional approach, which treated the problem as a single-label classification task. This successfully resolved the problem of inconsistencies that happened on the single code smell label approach. This raises a question if targeting several labels in parallel is increasing the accuracy of detecting code smells. Another related research work by (Soares et al., 2023) explored test smells which are smells that exist in software testing scripts. This tests are usually written in natural language and include a description on how to test a feature or functionality. The work included expanding the catalog defining test smells and then creating a tool to automatically detect these smells within a dataset of test scripts. The researchers created a test tool using Part-of-speech tagging, named entity recognition, dependency parsing, and syntactic analysis to identify test smells in the test scripts samples. A very recent research that is leveraging machine learning for automatic detection of smells, this time the focus is on defining API documentation smells and using ML for automatic detection of those smells (Khan et al., 2021). The team created the first benchmark dataset that is focused on API documentation smells. Five types of API documentation smells were defined: lazy, bloated, excess structure, tangled and fragmented. The datasett created included examples for all these smells, and several automated detection techniques were used. The top performer here was BERT which achieved an F1 score of 0.75-0.97.

1.4. Machine learning for legal analysis

Due to the nature of the legal domain, it has been challenging for machine learning researchers to achieve as much progress as we could see with other domains. However, some significant work was done and continues to be done to utilize the potential of artificial intelligence tools for helping legal professionals. Legal AI can be broken down to several main categories: Legal Judgment Prediction, Contract Review and Analysis, Discovery and Due Diligence, Legal Question Answering and Legal Document Generation. For legal judgment prediction we can see the paper (Yang et al., 2020), work aimed to deal with the main challenges of LJP such as multiple sub tasks that are involved when performing LJP, as well as the fact that some of the sub tasks are dependent on others. The researchers used graph theory to formulate dependencies relationships and used that to create a learning framework called TOP-JUDGE for legal judgment prediction. Unlike previous approaches, TOP-JUDGE represents the relationships between different subtasks as a Directed Acyclic Graph (DAG). This allows capturing of the dependencies among connected subtasks directly into the model architecture. By integrating multiple related subtasks and modelling their interdependencies, TOP-JUDGE provides a way to improve judgment prediction performance. The key innovation is the use of a DAG structure to capture how completion of certain subtasks influences and informs the overall judgment prediction task. This topological framework is at the core of the proposed TOP-JUDGE model. Another recent work for LJP can be seen in the paper Legal Judgment Prediction via Event Extraction with Constraints (Feng et al., 2022). The researchers here are making a claim that the most significant challenge for LJP models is their inability to identify the exact event or fact that are the determining factor for the judgment decision that is made. Therefor they are suggesting an event based prediction model to address this challenge. If the model is able to detect the event pattern in the facts of the case, it can match it to an event pattern that is detailed in a law article and then it can deduct the decision and applicable penalty. While there is a lot of work done on prediction models to assist legal professionals, legal language is at the center of any type of legal task and therefor we can see significant amount of effort directed to legal language understanding with language models. To evaluate and compare models performances we need a benchmark. Research by (Chalkidis et al., 2022) introduces LexGLUE, a benchmark dataset for this very task. Looking at state of the art in NLP, we can see that pre trained transformer based NLP models are leading in performance for almost all tasks and showing rapid improvements. The benchmark is providing both a generic and also a legal specific evaluation, and so it was able to show that legal-specific models are consistently performing better across multiple tasks. The evaluation benchmark datasets are available to use as well as the code used during this research. The legal tasks that were explored were judgment prediction, legal topic classification, and information extraction. The researchers looked at state of the art models such as BERT (Devlin et al., 2019), RoBERTa, Deberta (He et al., 2021), Longformer, BigBird, Legal-BERT and CaseLaw-BERT that was introduced at previous reseach works.

1.5. Natural language processing

Looking specifically at using NLP for legal purposes, the work by (Zhong et al., 2020) is reviewing how recent NLP work done is benefiting the legal industry. It defines that LegalAI purpose is to apply AI and NLP to assist with legal tasks, particularly in automating paperwork and streamlining legal processes. We can see two main approaches the field primarily utilizes: embedding methods that learn from data, and symbol methods that use legal knowledge. Drilling down on embedding-based methods using Character, Word, and Concept, we can see Word2Vec is being applied to legal texts, with the challenge being the learning of professional legal vocabulary and the aim being to capture grammar and legal knowledge within the embeddings. We can also drill down on pre-trained language models. Models like BERT have shown success in NLP, but legal terminology differences can lead to poor performance, as well as lack of large enough corpus as legal documents tend to be confidential or private. Pretraining on legal documents did shown better benchmark performances, with future work focusing on integrating knowledge into pretrained models for improved reasoning and actual tasks performed on legal texts as opposed to focusing on general understanding. In the realm of symbol-based methods we see the focus on extractions of information. This involves tasks like named entity recognition and relation extraction, crucial for legal tasks such as inheritance dispute resolution and criminal case judgment. Future applications include using extracted information for downstream legal tasks. We can identify several applications for LegalAI: Judgement prediction based on case facts, similar case matching, question answering which is both to help non professionals get legal help but also professionals who are looking to do legal research. Here the most recent models are still performing below human performances especially with multiple stage reasoning. Despite the progress, several areas require further improvement, knowledge representation, reasoning ability, and interpretability. It appears that by combining data-driven and knowledge-based methods, as well as few-shot learning techniques in prompting, we can expect better results.

1.6. Project Goals

The goal of this project is to first identify the possible contract smells, define them and categorize if they are a good candidate for detection by machine learning techniques. As well as identify which machine learning techniques are performing better at identifying contract smells in different types of contracts and different types of contracts. The goal is to test the feasibility of detecting such contract smells and provide a mechanism for both law professionals and the public to obtain an initial idea of areas in which a given contract might be lacking or requiring additional clarifications.

Chapter II.

Development of Contract Smells

2.1. Defining Contract Smells

Code smell is defined as an indication of a problem, suggesting a weakness in design that may increase the risk of future software failures or defects. Law smell is defined as a pattern in legal texts that pose threat to the comprehensibility and maintainability of the law. Contract smell can therefore be defined as a pattern or an indication in a legal contract document, that suggests a problem with the contract and increases the risk of a contract failure in comprehensibly and maintainability and may require mitigation. We can draw some parallels between contract legal drafting and software development. These will be used as the basis for identification and categorization of contract smells. 1. Logical structure: Both tasks require sticking to a logical structure. This structure has a significant role in making sure the expected result. In software the structure implements the control flow, conditions and outcomes. In legal contracts, the structure also allowing to organize the logical flow of the clauses but also to cover various conditions and contingencies. 2. Clarity: Since both rely on textual descriptions and in order for both to achieve exactly the desired results, software code and legal contracts need to be very clear and contain no ambiguity. 3. Maintenance: Software code and legal contract need to be adjusted while still in use. Software may need updates for new features, security, or efficiency improvements. Contracts might be updated to reflect changes in law, business arrangements, or other circumstances. 4. Interpretation for execution: In both cases, the text is being interpreted and then executed based on that interpretation. The impact of a possible inaccurate interpretation could be very significant in both case, resulting in major consequences and losses to the involved parties or software users.

2.2. Mapping Code to Contract

2.2.1 Long Method

Overly Long Clauses - In code, methods that carry on too long can become difficult to parse and manage. Similarly lengthy clauses in legal contracts tend to introduce complexity leading to potential misinterpretations or issues enforcing them.

2.2.2 Large Class

Overly Broad Scope - Software classes that take on too many responsibilities start to get unwieldy. Likewise, contracts with excessively wide scope likely address more than needed, resulting in ambiguity or disputes.

2.2.3 Primitive Obsession

Overuse of Basic Provisions - Leaning too heavily on primitive data types rather than rich objects creates problems in code. Contracts that are too reliant on generic boilerplate terms can require more tailored provisions to address specific needs.

2.2.4 Data Clumps

Repetitive Clauses - Groups of variables that appear together again and again in code signal opportunities to refactor. The same could be said for repetitive clauses in contracts that point to redundancies.

2.2.5 Switch Statements

Over-Reliance on Conditional Clauses - Chain after chain of switch statements usually indicate structural issues in code. Similarly, contracts that are long and contain many conditional clauses tend to be confusing.

2.2.6 Temporary Variable

Inconsistent Terms - variables or fields that are used only temporarily make code harder to follow. Terms or definitions in contracts that arbitrarily shift from section to section can cause confusion.

2.2.7 Feature Envy

Misplaced Responsibilities - When methods rely more on other classes, responsibilities are not placed where they should. Contracts could also include related responsibilities in mixed clauses causing confusion and make the contract more difficult to interpret.

2.2.8 Duplicate Code

Redundant Language cutting and pasting codes, or multiple hands drafting the same long contract could cause duplications go unnoticed. Repetitions cause not only an unnecessary increase in length and complexity , but can also increase the challenge of maintaining the contract when a clause that has many duplications requires updating.

2.2.9 Lazy Class

Code that lacks specificity or detail, or clause that add no real legal weight as it is not detailed enough to be enforceable or effective, will only make the contract longer, more complex to understand.

2.2.10 Speculative Generality

Coding for speculative future scenarios that can never apply adds unnecessary complexity. Similarly, overly broad contract provisions risk covering edge cases that are unlikely to appear.

2.2.11 Inappropriate Intimacy

In code, classes coupled too tightly run risks. Contracts often include sensitive information, overly detailed disclosures could cause a breach of confidentiality and an increased liability.

2.2.12 Comments

Over-reliance on comments inside code segments to explain complex code can be mapped to using excessive legalese or legal jargon terms can in fact make the contract harder to understand not only for the general public but also for less experienced law professionals.

2.3. Selecting Candidates

When assessing the suitability of different contract smells for detection by machine learning models, there are a few key factors to consider. The detection level, when the options are the clause, contract, or comparative level, impacts how easy it is to automate identification. If we require to assess a long clause, we would need to determine acceptable length for example. Also, the contextual understanding plays a major role. Smells that can be quantified or identified through pattern recognition tend to allow for more straightforward auto-labeling.

Consider something like overly long clauses, which can simply be measured by word or sentence count thresholds. The length of clauses provides clear numerical data for algorithms to parse. On the other end of the spectrum, you have issues like misplaced responsibilities that require deeper comprehension of the contractual relationships and objectives, which can be challenging for auto detection by AI.

In between, there are smells that have some mix of signals and subjective interpretation involved. For instance, repetitive clauses can certainly be spotted through duplicate text detection, but techniques like transformer-based language models may prove better suited to assess whether contractual provisions are basic.

As natural language understanding capabilities continue to progress, more smells may shift from moderate to high suitability for auto-labeling and detection. However is it likely that a hybrid approach that leverages the strengths of both automated flagging and expert human review is likely to produce the most robust results. However for this work we will mostly focus on a subset of contract smells that make the best candidates for auto detection based on their required level of context for successful detection. Below is a breakdown of our contract smells and an evaluation of their potential for auto detection. We will attempt to cover detection of the contract smells that will offer high suitably for detection by machine learning and auto labeling by large language models.

2.3.1 Overly Long Clauses (Long Method)

Detection Level: Clause Level Suitability for Auto-Labeling: High. Length of clauses can be measured quantitatively, making it suitable for auto-labeling at the individual clause level. This could be done against a pre determined threshold, and also potentially looking at the type of clause as some are proned to be longer than other by nature.

2.3.2 Overly Broad Scope (Large Class)

Detection Level: Contract Level Suitability for Auto-Labeling: Moderate. Requires understanding the overall context of the contract in order to evaluate the extent of scope which is covered in a specific clause. Additionally, comparative analysis with standard scopes in similar contracts could also provide significant information for the purpose of detection.

2.3.3 Overuse of Basic Provisions (Primitive Obsession)

Detection Level: Clause and Contract Level Suitability for Auto-Labeling: Moderate. Pattern recognition can help identify overused provisions within the clause however a comparative analysis with other contracts is also required to determine if the usage can indeed be considered excessive.

2.3.4 Repetitive Clauses (Data Clumps)

Detection Level: Contract Level Suitability for Auto-Labeling: High. Repetitive text within a contract is straightforward to identify, however the challenge here is the need to analyze the contract as a whole in order to achieve best results. As some contracts can reach a length of 500 pages, this is not easily achieved.

2.3.5 Over-Reliance on Conditional Clauses (Switch Statements)

Detection Level: Clause and Contract Level Suitability for Auto-Labeling: Moderate. Identifying multiple conditional clauses is feasible, but understanding their logical flow requires a broader view of the contract and as discussed above, is challenging with longer contracts.

2.3.6 Inconsistent Terms (Temporary Field)

Detection Level: Contract Level Suitability for Auto-Labeling: High. Inconsistencies can be detected by comparing terms within the same contract.

2.3.7 Misplaced Responsibilities (Feature Envy)

Detection Level: Contract Level Suitability for Auto-Labeling: Low. Requires deep understanding of the contract's context, not only its structure and therefore could be more difficult to automate.

2.3.8 Redundant Language (Duplicate Code)

Detection Level: Clause and Contract Level Suitability for Auto-Labeling: High. Redundant language can be identified by detecting similar phrases or sentences within a contract or within a clause.

2.3.9 Superfluous Clauses (Lazy Class)

Detection Level: Contract and Clause Level Suitability for Auto-Labeling: Moderate. Determining the the necessity for a given clause requires a contextual understanding of the entire contract, which is difficult for automated systems. However we can try to evaluate the amount of details in a specific clause to determine if it is sufficient in order for this clause to enforceable and effective.

2.3.10 Overly Broad Provisions (Speculative Generality)

Detection Level: Contract Level and Comparative Analysis Suitability for Auto-Labeling: Moderate. Broad language can be identified, but assessing its speculative nature might require comparison with typical provisions in similar contracts.

2.3.11 Overly Detailed Disclosures (Inappropriate Intimacy)

Detection Level: Clause and Contract Level Suitability for Auto-Labeling: Moderate. Detailed disclosures can be flagged, but evaluating their appropriateness requires understanding the contract's context.

2.3.12 Excessive Legalese (Comments)

Detection Level: Clause Level Suitability for Auto-Labeling: High. Machine learning can identify complex legal jargon at the clause level and contract level as well.

Name	Level	Suitability
Overly Long Clauses (Long Method)	Clause	High
Overly Broad Scope (Large Class)	Contract	Moderate
Overuse of Basic Provisions (Primitive Obsession)	Clause, Contract	Moderate
Repetitive Clauses (Data Clumps)	Contract	High
Over-Reliance on Conditional Clauses (Switch Statements)	Clause, Contract	Moderate
Inconsistent Terms (Temporary Field)	Contract	High
Misplaced Responsibilities (Feature Envy)	Contract	Low
Redundant Language (Duplicate Code)	Clause, Contract	High
Superfluous Clauses (Lazy Class)	Clause, Contract	Moderate
Overly Broad Provisions (Speculative Generality)	Contract, Comparative	Moderate
Overly Detailed Disclosures (Inappropriate Intimacy)	Clause, Contract	Moderate
Excessive Legalese (Comments)	Clause	High

Table II.1: Identified Contract Smells

Chapter III.

Methods And Experiments

3.1. Dataset

The dataset we are using is the CUAD dataset introduced in (Hendrycks et al., 2021). The dataset comprises over 500 legal contracts, encompassing 25 different types of contracts. These contracts vary in length, ranging from a few pages to over 100 pages. The work done here was to label these contracts for extracting significant pieces of information that law professionals usually look for when evaluating and reviewing contracts. The annotations produced during CUAD work, are not relevant for the purpose of this research, however the store of contracts that are publicly available are very useful as obtaining legal contracts can be challenging due to privacy and confidentiality considerations.

3.1.1 Labelling Contracts

In order to train models to detect contract smells that were introduced above, we need to create a labeled dataset of contracts. As manual labeling by law professionals is expensive and extremely time consuming, the plan is to use large language models to auto label the contracts. Auto labelling was performed in several ways in order to further discuss how best we can leverage LLMs for auto labeling. The current leading LLM model that also offer API access is GPT4-turbo by OpenAI and that is the engine we use. - Feeding whole contract to the model along with a subset of the selected contract smells as identified in table II.1 with high suitably for auto labelling and detection.

3.1.2 Labelling Methods

A total of 3 datasets were created when auto-labeling the contracts: In order to create the first database, GPT4 api inference is used, provided contract clauses and a list of 5 contract smells with their definitions, and produces a list of clauses to match with the contract smells, if such found. The dataset columns are: contract, clause, LongMethod, DataClumps, TempField, DuplicateCode, ExcessLegalese. This is an extractive zero shot approach as no examples are provided to the model. Only a descriptive textual definition of all the smells in question. This is referred to as the Textual-Oriented dataset. The second dataset is created in a similar manner, however the prompt to identify the contract smells within the contract is based on a coding oriented definition of the contract smells. This is referred to as the Coding-Oriented dataset. The third dataset is auto-labelled with extended (coding and textual) definitions of all seven contract smells selected. Since this is the most extensive dataset created, it is simply referred to as the Auto-Labeled dataset. A forth dataset we are using is the Human Labeled dataset that we were able to obtain by collecting human feedback. As human resources are limited we were only able to obtain a small set of contracts labeled only to cover two of our selected contract smells: LazyClass and InappropIntimacy.

3.2. Experiments

In order to achieve the best possible results for detecting contract smells, several experiments were be conducted:

Baseline BERT. Our baseline result is running BERT on each dataset and producing a prediction on each of the contract smells in question. This model has no prior knowledge of the domain, nor does it have a the full context of the contract. This model is trained and evaluated on all four datasets.

LegalBERT. This experiment is to determine the significance of the domain knowledge on the ability to detect contract smells. The legalbert model was trained on a variety of legal text and is able to capture the meaning of legal content. This model is trained and evaluated on all four datasets.

LegalBERT CUAD Fine Tuned. This experiment evaluating the ability to detect contract smells when the model has both pretaining of legal domain as well as training on the contracts from which the labelled data is taken from . This is expected to show the best results for detecting contract smells. This model is trained and evaluated on all four datasets.

Chapter IV.

Development

This chapter discusses the tools, libraries and APIs that are employed in the code development of this contract smells detection system.

4.1. Development Tools

OpenAI's APIs have been used in order to auto label the contracts using their latest GPT4 model. Google colab Jupyter notebooks are used as running environment to for inference and fine tuning of the models. huggingface is the repository used to access the models that are tested during this work as well as store the datasets created.

4.2. Project Structure

Auto labelling: $auto_label_descriptive.ipynb$ and $auto_label_by_clause.ipynb$ are created to parse the contracts, analyze them and produce the datasets. This is done when taking a descriptive prompt approach with two sets of contract smells definitions - coding oriented, and textual oriented. There are 3 colab notesbooks to match each of the experiments we run: Baseline BERT model with its specific data preparation and results analysis is captured in $BERT_contract_smells_detection_hl_autol.ipynb$. LegalBERT model is captured in $LegalBERT_contract_smells_detection_hl_autol.ipynb$. $CUADLegalBert_contract_smells_detection_hl_autol.ipynb$ is where we run detection task on a LegalBert model that was already trained on the contracts from the CUAD project, which are the source of the clauses that are analysed in this project.

4.2.1 Optimizing Definitions For Model Consumption

The definitions for the selected contract smells are not optimized for model consumption. The goal is to make them more concise and simple to make them more accessible for the model to find matches for. We will be experimenting with 2 sets of optimized definitions:

4.2.2 Set 1 - Focus on textual patterns:

LongMethod (Overly Long Clauses): Clauses in contracts that are exceptionally long and contain many complex sentences. These clauses can be difficult to understand and may lead to misinterpretation. DataClumps (Repetitive Clauses): Clauses that are very similar or identical to each other, appearing multiple times within the contract. This indicates potential redundancy and makes the contract unnecessarily long. TempField (Inconsistent Terms): Variations in the use of key terms or definitions throughout the contract. Inconsistent terminology can create confusion and ambiguity. DuplicateCode (Redundant Language): Phrases, sentences, or entire clauses that are repeated unnecessarily. This lengthens the contract without adding value and can make future updates more difficult. ExcessLegalese: Overuse of obscure legal terms and complex jargon. This makes the contract difficult to understand for non-specialists and may impede clarity even for legal professionals. LazyClass: An ambiguous clause that add no real legal weight as it is not detailed enough to be enforceable or effective, will only make the contract longer, more complex to understand. InappropIntimacy: Contracts often include dependencies between clauses or sentences that are tightly dependant within a clause.

4.2.3 Set 2 - Focus on parallels to coding:

LongMethod: Clauses that are excessively long introduce complexity and can lead to misinterpretation. Similar to how lengthy methods in code become hard to manage. DataClumps: Repetitive clauses in contracts, like repeated groups of variables in code, indicate redundancy that could be streamlined. TempField: Shifting terms or definitions within a contract create confusion, akin to the use of temporary fields in code that complicates understanding. DuplicateCode: Redundant language in contracts, resulting from multiple drafters or copy-pasting, mirrors code duplication, increasing length and complicating updates. ExcessLegalese: Using too much legal jargon makes contracts difficult to understand, similar to overusing comments to explain complex code segments. LazyClass: A code that is not specific enough and therefore not effective. InappropIntimacy: Classes or functions that are coupled too tightly.

Chapter V.

Results

This chapter presents the results obtained from auto labelling as well as results obtained after completing all fine tuning experiments that were run on the two auto labelled datasets across models tested.

5.1. Auto-labelling

5.1.1 Coding Oriented Dataset:

Auto labelling the contracts with the coding oriented definitions of the contract smells provided in the prompt resulted in a dataset of 14,177 smelly clauses detected across 474 contracts.

Percentage of instances for each class:

i. LongMethod 37.229315

ii. DataClumps 10.002116

iii. TempField 3.625591

iv. DuplicateCode 1.065105

v. ExcessLegalese 78.810750

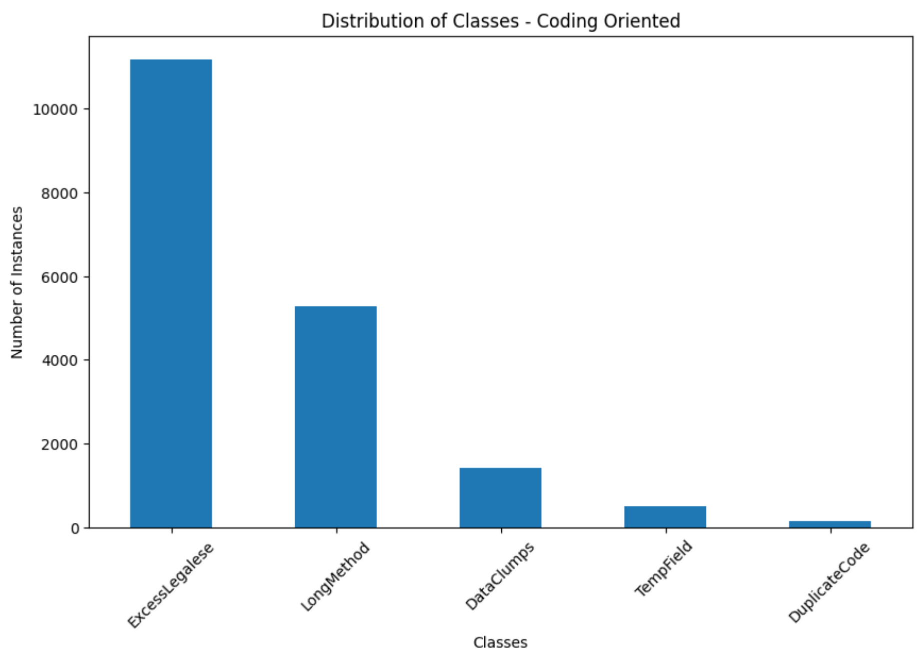

Figure V.1:

This dataset is clearly displaying imbalance of the classes when most of the smelly clauses are identified as ExcessLegalese which means they make excessive use in legal language. On the other hand we see classes that are extremely rare such as DuplicateCode or TempField.

It stands to reason that some classes are related to others and so we look at the co-occurrence of classes in the image below:

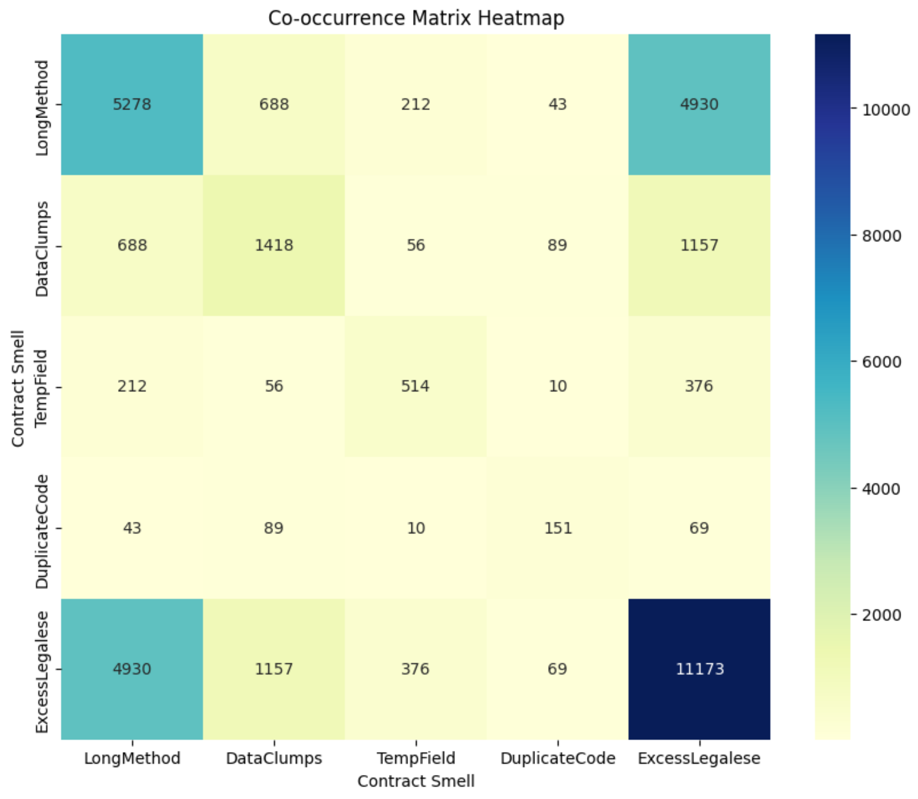

Figure V.2:

5.1.2 Textual Oriented Dataset:

Auto labelling the contracts with the textual oriented definitions of the contract smells provided in the prompt resulted in a dataset of 9,607 smelly clauses detected across 381 contracts.

Percentage of instances for each class:

i. LongMethod 23.857604

ii. DataClumps 5.214947

iii. TempField 1.405225

iv. DuplicateCode 1.301135

v. ExcessLegalese 79.046529

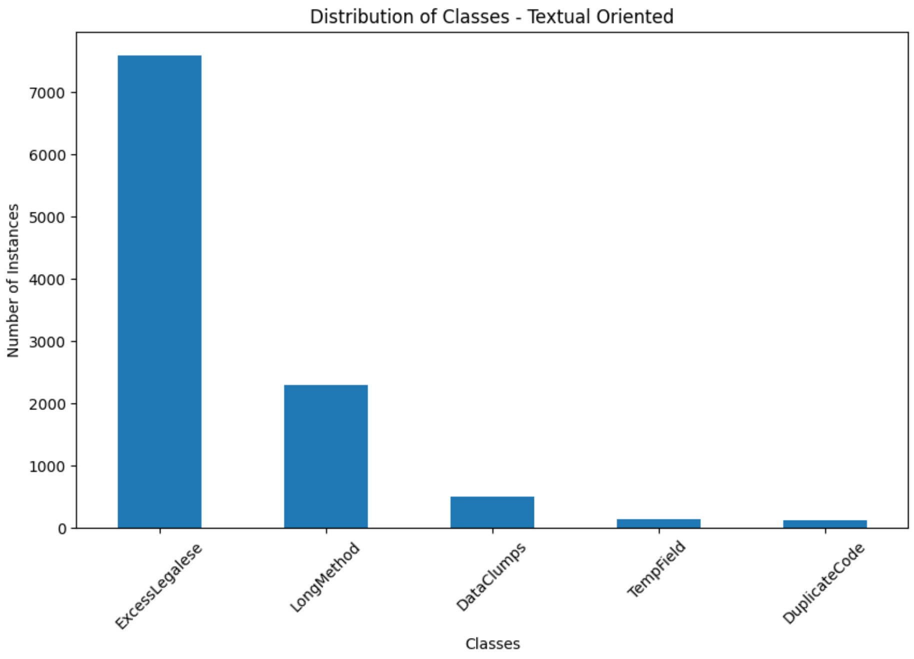

Figure V.3:

And similarly the co-occurrence heatmap is shown below:

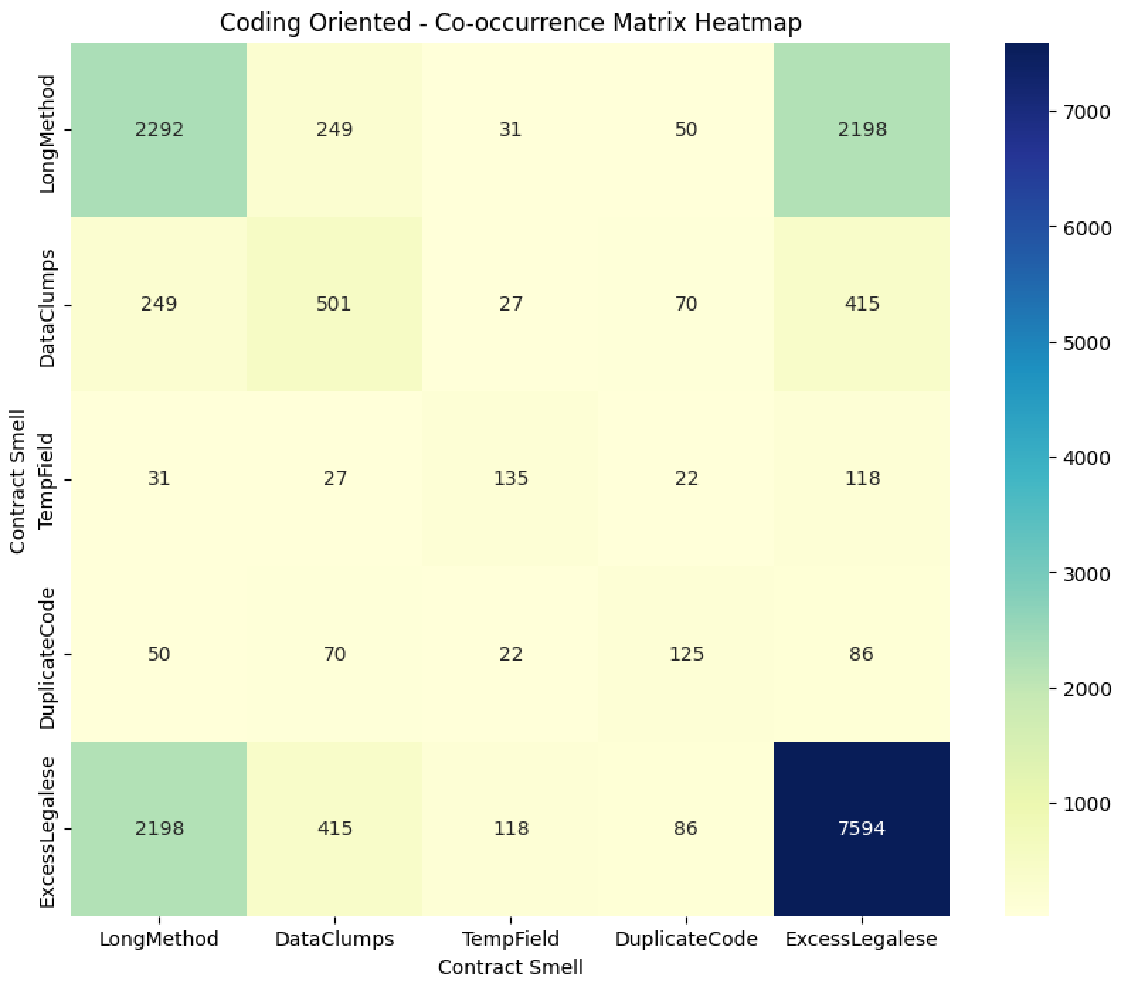

Figure V.4:

It appears we have similar distribution for the classes when again ExcessLegalese and LongMethod are very prevalent while TempField and DuplicateCode are rare. This is one of the reasons for the co-occurance we see between these two classes shown in the heatmap.

5.1.3 Auto Labeled Dataset:

Auto labelling the contracts with the extended definitions covering all 7 contract smells. A total of 15257 were identified.

Percentage of instances for each class:

i. LongMethod 29.586419

ii. DataClumps 1.356754

iii. TempField 3.172314

iv. DuplicateCode 2.772498

v. ExcessLegalese 28.957200

vi. LazyClass 3.198532

vii. InappropIntimacy 0.117979

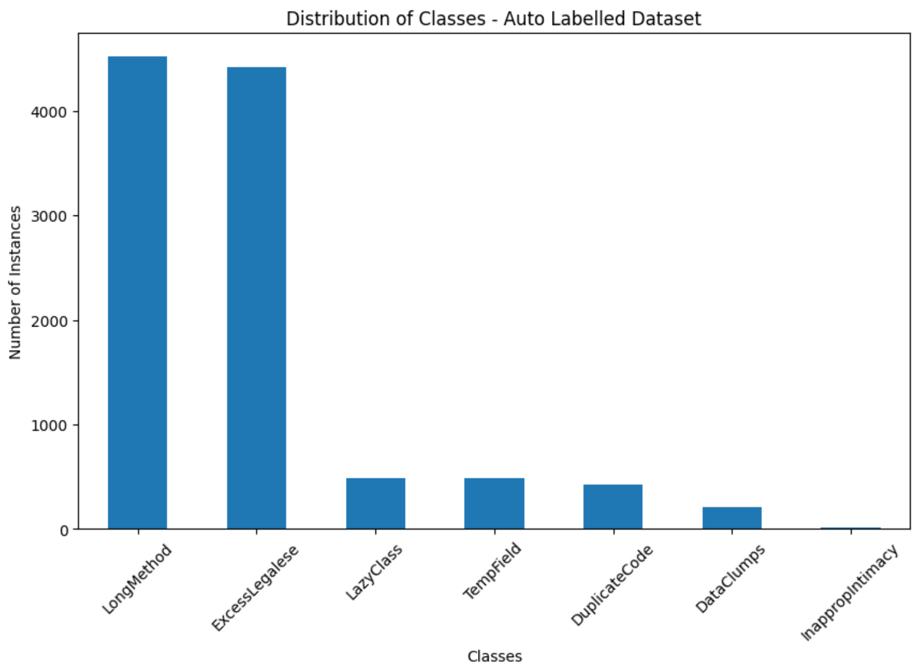

Figure V.5:

And similarly the co-occurrence heatmap is shown below:

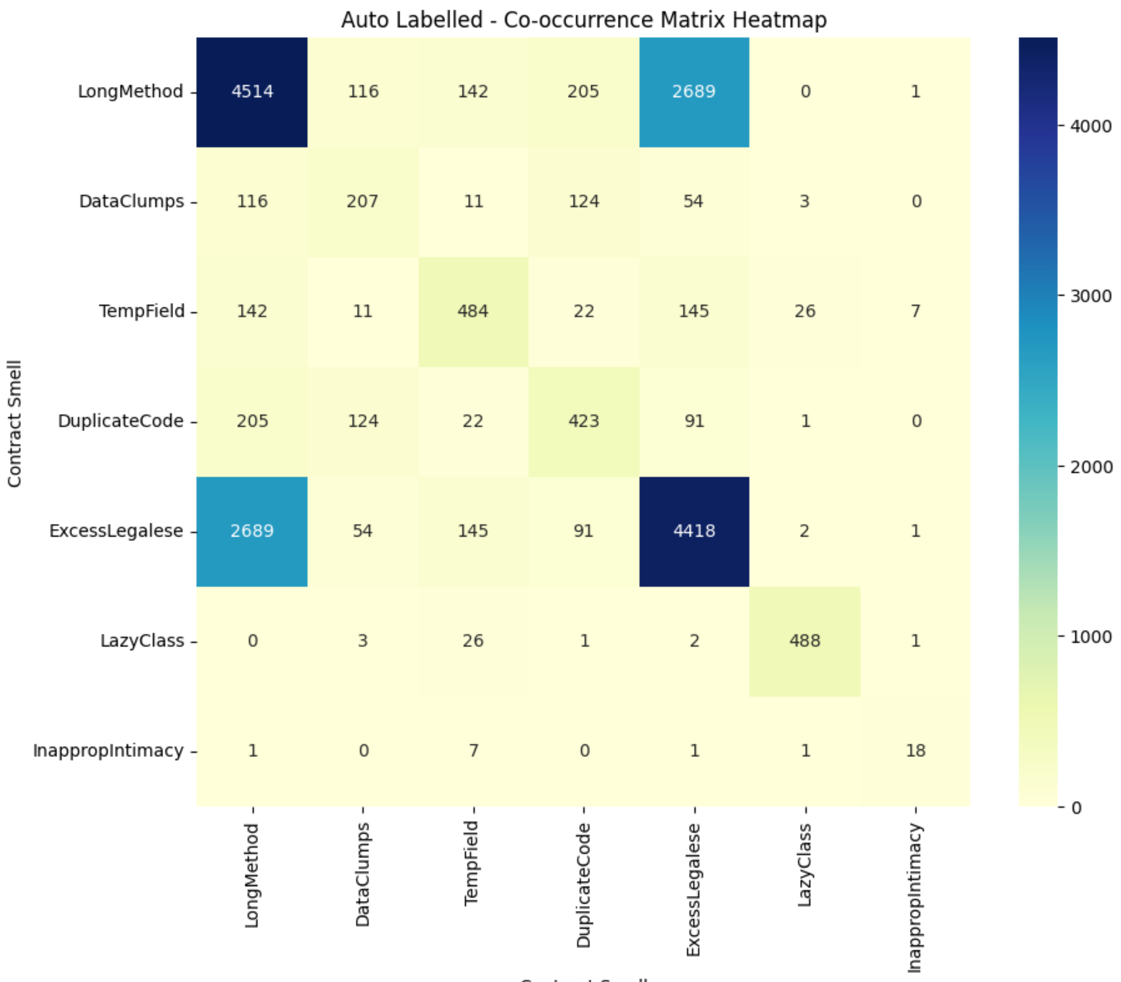

Figure V.6:

5.1.4 Human Labeled Dataset:

Human labels were done for several contracts for InappropIntimacy and LazyClass contract smells. A total of 529 clauses were identified.

Percentage of instances for each class:

i. LazyClass 61.625709

ii. InappropIntimacy 24.763705

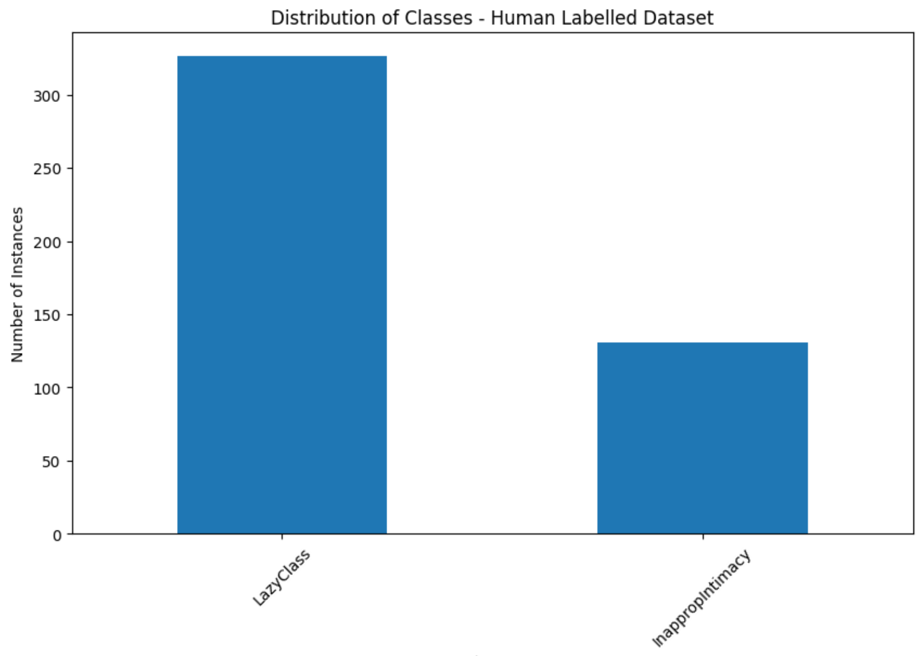

Figure V.7:

And similarly the co-occurrence heatmap is shown below:

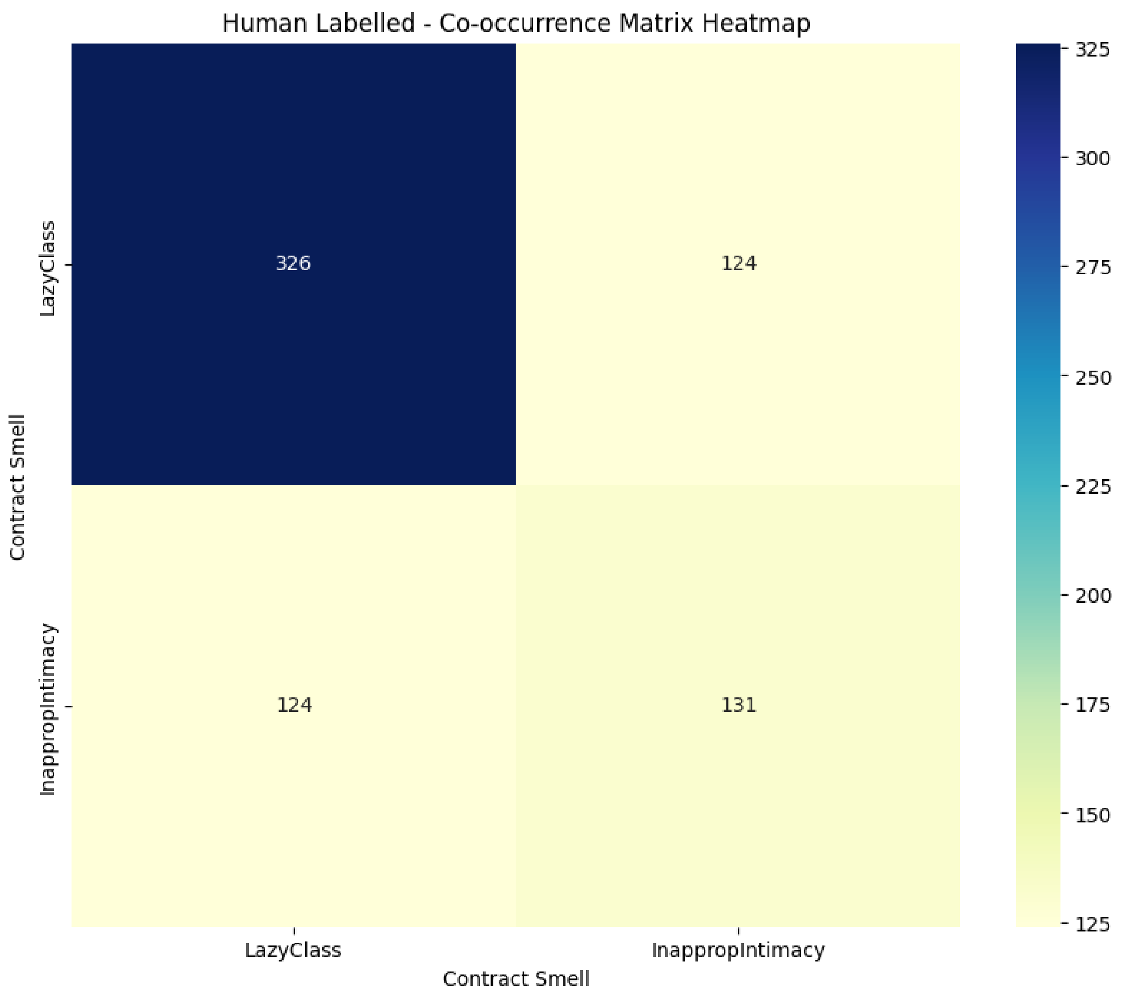

Figure V.8:

5.1.5 Datasets Highlights:

The 2 datasets that were created on specific prompt definitions are similar in class compositions.

LongMethod and ExcessLegalese are both classes that could be over represented in both datasets which impacts models tendency to classify clauses for these classes. There may be a need to undersample this class.

DataClumps, DuplicateCode, TempField may all be underrepresented in both datasets which can impact the models predictions.

InappropIntimacy and LazyClass are both even more rare in our auto-labelled dataset.

Differences between the auto labeller's results can be explained by the impact of rare events i.e. because contract smells are generally rare, and most contracts examined are of high quality, there is more impact to the prompt provided as there are simply less examples to learn from.

Since we have such imbalance in the datasets, it is more reasonable to train separately on each contract smell than it would be to train for a multi label problem.

5.2. Baseline BERT

BERT was trained on a train-test split of twenty percent as commonly used for train on task, and was run on each of the contract smells separately (single label scenario) Below is a table showing performance metrics that were evaluated on the performance of a baseline BERT model for detecting contract smells:

Category	**Dataset Type**	**Accuracy**	**F1 Score**	**Precision**	**Recall**
LongMethod	Text-Oriented	89.12%	84.70%	86.65%	83.19%
LongMethod	Coding-Oriented	91.39%	90.75%	91.21%	90.37%
LongMethod	Auto-Labelled	94.00%	92.74%	93.01%	92.47%
DataClumps	Text-Oriented	96.30%	82.66%	88.66%	78.46%
DataClumps	Coding-Oriented	96.12%	88.66%	91.49%	86.29%
DataClumps	Auto-Labelled	98.69%	71.10%	76.34%	67.64%
TempField	Text-Oriented	99.42%	87.06%	87.93%	86.23%
TempField	Coding-Oriented	98.76%	90.97%	93.26%	88.93%
TempField	Auto-Labelled	97.51%	77.01%	78.65%	75.55%
DuplicateCode	Text-Oriented	98.64%	49.65%	49.32%	50.00%
DuplicateCode	Coding-Oriented	99.43%	78.80%	95.56%	71.13%
DuplicateCode	Auto-Labelled	98.10%	73.62%	91.31%	66.79%
ExcessLegalese	Text-Oriented	77.99%	43.81%	38.99%	50.00%
ExcessLegalese	Coding-Oriented	88.54%	81.16%	84.69%	78.73%
ExcessLegalese	Auto-Labelled	87.35%	84.28%	84.85%	83.76%
InappropIntimacy	Auto-Labelled	99.87%	49.97%	49.93%	50.00%
InappropIntimacy	Human-Labelled	71.69%	41.75%	35.84%	50.00%
LazyClass	Auto-Labelled	96.85%	69.19%	72.86%	66.61%
LazyClass	Human-Labelled	76.41%	70.64%	85.95%	70.23%

Table V.1: BERT Baseline Model Results

It is clear that BERT baseline is able to produce strong overall results on most datasets especially for accuracy. When looking at the precision and recall however , it is clear that the model is facing challenges with false positives and false negatives. Better results are obtained on the Coding oriented dataset, and it appears that all are especially challenged when trying to detect DuplicateCode, ExcessLegalese and InappropIntimacy. This is reflected in the F1 score as well.

5.3. Legal BERT

LegalBERT was similarly trained on a train-test split of twenty percent as commonly used for train on task, and was run on each of the contract smells separately (single label scenario). Below is a table showing performance metrics that were evaluated on the performance of a the LegalBERT model for detecting contract smells:

Category	**Dataset Type**	**Accuracy**	**F1 Score**	**Precision**	**Recall**
LongMethod	Text-Oriented	83.71%	76.69%	78.26%	75.49%
LongMethod	Coding-Oriented	73.23%	67.29%	72.95%	66.54%
LongMethod	Auto-Labelled	93.84%	92.47%	92.92%	92.05%
DataClumps	Text-Oriented	96.98%	83.94%	90.05%	79.61%
DataClumps	Coding-Oriented	90.40%	59.66%	77.87%	57.06%
DataClumps	Auto-Labelled	98.66%	68.12%	78.04%	63.49%
TempField	Text-Oriented	98.64%	49.65%	49.32%	50.00%
TempField	Coding-Oriented	96.40%	49.08%	48.20%	50.00%
TempField	Auto-Labelled	97.90%	75.93%	88.23%	69.83%
DuplicateCode	Text-Oriented	99.06%	67.62%	99.53%	60.86%
DuplicateCode	Coding-Oriented	99.04%	49.76%	49.52%	50.00%
DuplicateCode	Auto-Labelled	98.39%	84.37%	85.44%	83.36%
ExcessLegalese	Text-Oriented	80.64%	52.45%	77.63%	54.03%
ExcessLegalese	Coding-Oriented	78.63%	44.01%	39.31%	50.00%
ExcessLegalese	Auto-Labelled	87.19%	84.11%	84.51%	83.75%
InappropIntimacy	Auto-Labelled	99.93%	49.98%	49.97%	50.00%
InappropIntimacy	Human-Labelled	71.70%	41.76%	35.85%	50.00%
LazyClass	Auto-Labelled	96.79%	60.29%	76.56%	56.74%
LazyClass	Human-Labelled	72.64%	63.52%	84.57%	64.63%

Table V.2: LegalBert Model Results

For this model we can actually see that accuracy scores for the textual dataset are better than the ones obtained on the coding oriented dataset. We can also see that precision and recall are better on the textual dataset as well. However these results are improved significantly when training on the auto-labelled dataset built using the extended definitions. This suggests the Legal context this model has, enabled it to detect contract smells better on a textual definition and improved balance.

5.4. CUAD trained Legal BERT

The last experiment on our two datasets is running with a LegalBERT model that was already trained on the contracts developed in the CUAD research work. This model is familiar with all the contracts used for creation of the two datasets.

Below is a summary of the results:

Category	**Dataset Type**	**Accuracy**	**F1 Score**	**Precision**	**Recall**
LongMethod	Text-Oriented	84.55%	79.39%	79.54%	79.25%
LongMethod	Coding-Oriented	89.31%	88.67%	88.47%	88.89%
LongMethod	Auto-Labelled	93.35%	92.18%	91.60%	92.83%
DataClumps	Text-Oriented	96.51%	81.54%	83.03%	80.20%
DataClumps	Coding-Oriented	95.20%	86.74%	87.93%	85.64%
DataClumps	Auto-Labelled	98.62%	66.84%	79.98%	61.84
TempField	Text-Oriented	98.69%	49.67%	49.35%	50.00%
TempField	Coding-Oriented	97.07%	77.05%	81.12%	73.99%
TempField	Auto-Labelled	97.08%	75.95%	78.74%	73.69%
DuplicateCode	Text-Oriented	98.69%	63.95%	80.67%	59.18%
DuplicateCode	Coding-Oriented	99.33%	78.72%	96.11%	70.95%
DuplicateCode	Auto-Labelled	98.39%	81.71%	85.53%	78.66%
ExcessLegalese	Text-Oriented	77.73%	43.73%	38.86%	50.00%
ExcessLegalese	Coding-Oriented	88.01%	81.40%	83.14%	80.01%
ExcessLegalese	Auto-Labelled	87.02%	84.45%	84.70%	84.21%
InappropIntimacy	Auto-Labelled	99.93%	49.98%	49.97%	50.00%
InappropIntimacy	Human-Labelled	71.70%	41.76%	35.85%	50.00%
LazyClass	Auto-Labelled	96.79%	60.29%	76.56%	56.74%
LazyClass	Human-Labelled	75.47%	70.37%	77.81%	69.47%

Table V.3: CUAD trained LegalBert Results on Text-Oriented and Coding-Oriented Datasets

We can see this model also performs well in terms of accuracy across all categories especially on the coding-oriented dataset. In previous models we have seen that while accuracy was high, F1 scores were not as good, however here we can see improvements on the F1 scores compared to the original LegalBert, and similar in performance to the baseline BERT model.

5.5. Models Summery

5.5.1 High Level Performance

The summary of average accuracy and F1 scores on the results emphasize the impact of dataset type in model performance, meaning the prompt used in the auto labelling process has a visible impact on the quality of the results. In terms of model performance, the average performance across all smells is greatest when using the Baseline BERT model.

	Bert	**LegalBert**	**CUADLegalBert**
Accuracy Textual	92.294	91.806	91.234
F1 Textual	69.576	66.070	63.656
Accuracy Coding	94.848	87.540	93.784
F1 Coding	86.068	53.960	82.516
Accuracy Auto-Labelled	95.13	95.196	94.892
F1 Auto-Labelled	79.75	81	80.226

Table V.4: Performance Metrics of Different Models

5.5.2 Smell Specific Analysis

Accuracy values are very strong for all models on most of the contract smell types. However we can see clearly that for LongMethod and ExcessLegalese the accuracy values are not as good. This is likely due to the fact that for both these classes, it is a challenge to identify them by the clause level alone.

Category	**BERT Baseline**			**LegalBert**			**CUAD-trained LegalBert**		
	Text	**Coding**	**Auto-Labelled**	**Text**	**Coding**	**Auto-Labelled**	**Text**	**Coding**	**Auto-Labelled**
LongMethod	89.12%	91.39%	94.00%	83.71%	73.23%	93.84%	84.55%	89.31%	93.34%
DataClumps	96.30%	96.12%	98.68%	96.98%	90.40%	98.65%	96.51%	95.20%	98.62%
TempField	99.42%	98.76%	97.50%	98.64%	96.40%	97.90 %	98.69%	97.07%	97.08%
DuplicateCode	98.64%	99.43%	98.09%	99.06%	99.04%	98.39%	98.69%	99.33%	98.39%
ExcessLegalese	77.99%	88.54%	87.35%	80.64%	78.63%	87.18%	77.73%	88.01%	87.02%

Table V.5: Accuracy Analysis Across Models for Contract Smells

Contract smells for which we also had human labelled data are analyzed below:

Category	**BERT Baseline**		**LegalBert**		**CUAD-trained LegalBert**	
	Human Labelled	**Auto Labelled**	**Human Labelled**	**Auto Labelled**	**Human Labelled**	**Auto Labelled**
LazyClass	76.41%	96.85%	72.64%	96.78%	75.47%	96.46%
InappropIntimacy	71.69%	99.86%	71.69%	99.93%	99.73%	86.74%

Table V.6: Analysis Across Models for Human Evaluated Contract Smells

5.5.3 Cross Predictions Results

Models Comparison. With human labels so difficult to obtain, there is great value in developing further the idea of auto labelling to allow models to better predict human labels. In this research we have two contract smells for which we can used automatically generated labels, to the human labels provided. Below we can see performance of our models that were trained on the auto labelled data, in predicting the human labels.

Auto-Labelled Training to Predict Human Labels:

Category	**Accuracy**	**F1 Score**	**Precision**	**Recall**
BERT LazyClass	39.62%	28.38%	19.81%	50.00%
BERT InappropIntimacy	71.70%	41.76%	35.85%	50.00%
LegalBERT LazyClass	38.68%	27.89%	19.34%	50.00%
LegalBERT InappropIntimacy	71.70%	41.76%	35.85%	50.00%
CUADLB LazyClass	37.74%	27.40%	18.87%	50.00%
CUADLB InappropIntimacy	71.70%	41.76%	35.85%	50.00%

Table V.7: Predictive Results

Improving Cross Prediction - Few Shot. In our case, and this is quite common, we only have a small sample of human labelled examples and these as we have seen are not enough to achieve good fine tuning results. However there is another way in which we can use our human labelled data. Capturing examples which our models incorrectly predicted their value, we are able to use those difficult to label examples in order to better our auto labeller. To demonstrate this, we have selected several such examples and recrafted the prompt used for auto labelling the data. After few-shots labelling of 106 examples, we can see the following correct labelling improvements: LazyClass 38% InappropIntimacy 0.9%

This improvement is definitely related to the type of contract smell but we can see that the bigger improvement was achieved on LazyClass which is the contract smell on which the auto trained models showed poor prediction performance in the table above.

Chapter VI.

Summary and Conclusions

As this work is showing us, NLP is demonstrating great promise when it comes to taking on the challenging task of legal contract review. There is no doubt this field in legal work can be effectively automated with properly trained models and more extensive datasets. In addition, people in their everyday lives are interacting with contracts and agreements written in legal language and are unable to fully understand the meaning due to this language barrier. The successful detection of issues with legal contracts that was displayed in this work, lays the foundation for a development of a fully automated system for contract evaluation by ML and AI that can help the public get a quick feedback to a legal document they are confronted with and be able to get an initial idea of how well the document is drafted while also provide them with a better ability to benefit from professional legal advice once they reach out for one. As well as demonstrating the great potential of using auto labelling for creating quality datasets.

6.1. Discussion Points

i. The successful mapping of code smells into contract smells that was completed during this work, is showing great promise for further transfer of ideas and concepts between software development and legal drafting.

ii. To mitigate imbalance in the datasets, the training was done separately on each contract smelland allowed the models to be specialized on each contract category.

iii. The summary of average accuracy and F1 scores on the results emphasize the impact of dataset type in model performance, meaning the prompt used in the auto labelling process has a visible impact on the quality of the results. In terms of model performance, the average performance across all smells is greatest when using the Baseline BERT model.

iv. As seen in the results of this research, the parallels between software development and legal drafting are in fact benefiting NLP models when analyzing legal contracts despite the fact that these domain seem to be non related on the face of it. We see this in the way BERT baseline model showed performance that surprisingly surpassed the LegalBert performance. Meaning general legal training is not as impactful for this specific task as well as the fact that our definitions for issues are taken from software engineering domain and therefore a more versatile training allowed BERT baseline to generalize better when detecting contract smells.

v. We can also see that CUAD contracts training did not improve model performance by much and therefore suggesting that for our selected contract smells, task based fine tuning is more effective compared to general context and familiarity with the used documents.

v. Using automatic labelling as means to save on human label costs and time, had proved to be extremely instrumental and the confirms a way to tackle the challenges in obtaining data for model training. This is only enhanced by our demonstration of how even a small subset of human data can create a stronger auto labellers with iteratively improving using difficult to label few shots examples in our auto labeller prompt.

6.2. Future Work

This work is laying a foundation for development of a set of tools to assist both professionals and non professionals with contract understanding and mass contract review process that is often required in a business scenario. There are however more ways to further explore and enhance the results that were achieved during this work. Below are examples of such ideas:

i. Combining other computational metrics and information such as regex matching or computed thresholds could be included in the prompt and in the labelling process for achieving better results.

ii. Further exploring of the other contract smells that were defined during this work and envision ways to automatically detect these as well.

iii. NLP models have gotten a boost of new approaches and advancements in the past several months with Generative AI models leading the way. Using even newer state of the art models can produce even more successful results.

References

Chalkidis, I., Jana, A., Hartung, D., Bommarito, M., Androutsopoulos, I., Katz, D., & Aletras, N. (2022). LexGLUE: A benchmark dataset for legal language understanding in English. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 4310–4330). Dublin, Ireland: Association for Computational Linguistics.

Coupette, C., Hartung, D., Beckedorf, J., Böther, M., & Katz, D. M. (2023). Law smells. *Artificial Intelligence and Law*, 31(2), 335–368.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)* (pp. 4171–4186). Minneapolis, Minnesota: Association for Computational Linguistics.

Feng, Y., Li, C., & Ng, V. (2022). Legal judgment prediction via event extraction with constraints. In *Proceedings of the 60th Annual Meeting of the Association* *for Computational Linguistics (Volume 1: Long Papers)* (pp. 648–664). Dublin, Ireland: Association for Computational Linguistics.

Fowler, M., Beck, K., Brant, J., Opdyke, W., & Roberts, D. (1999). *Refactoring: Improving the Design of Existing Code*. Addison-Wesley Professional.

Guggulothu, T. (2019). Code smell detection using multilabel classification approach.

Gupta, H., Gulanikar, A. A., Kumar, L., & Neti, L. B. M. (2021). Empirical analysis on effectiveness of nlp methods for predicting code smell.

He, P., Liu, X., Gao, J., & Chen, W. (2021). {DEBERTA}: {DECODING}-{enhanced} {bert} {with} {disentangled} {attention}. In *International Conference on Learning Representations*.

Hendrycks, D., Burns, C., Chen, A., & Ball, S. (2021). Cuad: An expert-annotated nlp dataset for legal contract review. *arXiv preprint arXiv:2103.06268*.

Khan, J. Y., Khondaker, M. T. I., Uddin, G., & Iqbal, A. (2021). Automatic detection of five api documentation smells: Practitioners' perspectives.

Sharma, T., Efstathiou, V., Louridas, P., & Spinellis, D. (2019). On the feasibility of transfer-learning code smells using deep learning.

Soares, E., Aranda, M., Oliveira, N., Ribeiro, M., Gheyi, R., Souza, E., Machado, I., Santos, A., Fonseca, B., & Bonifácio, R. (2023). Manual tests do smell! cataloging and identifying natural language test smells.

Yang, F. et al. (2020). Legal judgment prediction via topological learning. *AAAI*.

Zhong, H., Xiao, C., Tu, C., Zhang, T., Liu, Z., & Sun, M. (2020). How does nlp benefit legal system: A summary of legal artificial intelligence.

Appendix A.

Source Code

The source code for this work can be found in the following github repositories: *github repository*.