

Resource-Efficient LLM Application for Structured Transformation of Unstructured Financial Contracts

Maruf Ahmed Mridul
Rensselaer Polytechnic Institute
Troy, New York, USA
mridum@rpi.edu

Oshani Seneviratne
Rensselaer Polytechnic Institute
Troy, New York, USA
senevo@rpi.edu

Abstract

The transformation of unstructured legal contracts into standardized, machine-readable formats is essential for automating financial workflows. The Common Domain Model (CDM) provides a standardized framework for this purpose, but converting complex legal documents like Credit Support Annexes (CSAs) into CDM representations remains a significant challenge. In this paper, we present an extension of the CDMizer framework, a template-driven solution that ensures syntactic correctness and adherence to the CDM schema during contract-to-CDM conversion. We apply this extended framework to a real-world task, comparing its performance with a benchmark developed by the International Swaps and Derivatives Association (ISDA) for CSA clause extraction. Our results show that CDMizer, when integrated with a significantly smaller, open-source Large Language Model (LLM), achieves competitive performance in terms of accuracy and efficiency against larger, proprietary models. This work underscores the potential of resource-efficient solutions to automate legal contract transformation, offering a cost-effective and scalable approach that can meet the needs of financial institutions with constrained resources or strict data privacy requirements.

CCS Concepts

• **Computing methodologies** → **Information extraction; Machine learning.**

Keywords

Common Domain Model (CDM), CDMizer, contract-to-CDM conversion, Credit Support Annexes (CSAs), Large Language Models (LLMs), Retrieval-Augmented Generation (RAG), resource-efficient models, legal contract digitization.

ACM Reference Format:

Maruf Ahmed Mridul and Oshani Seneviratne. 2025. Resource-Efficient LLM Application for Structured Transformation of Unstructured Financial Contracts. In *The 2nd Workshop on LLMs and Generative AI for Finance at the 6th ACM International Conference on AI in Finance (ICAIF '25)*, November 15–18, 2025, Singapore, Singapore. ACM, New York, NY, USA, 5 pages. <https://doi.org/10.1145/nnnnnnn.nnnnnnn>

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
Conference'17, Washington, DC, USA

© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-x-xxxx-xxxx-x/YYYY/MM
<https://doi.org/10.1145/nnnnnnn.nnnnnnn>

1 Introduction

The digitization and standardization of legal agreements are fundamental steps toward automating complex financial workflows, such as collateral management, regulatory reporting, and trade processing. In this paper, we focus on Credit Support Annexes (CSAs), which are integral components of ISDA Master Agreements that govern collateral management in over-the-counter (OTC) derivatives markets. They define critical terms such as eligible collateral, thresholds, and minimum transfer amounts, directly shaping counterparty risk exposure. However, CSAs are drafted in unstructured legal text, making them difficult to interpret consistently across institutions. This lack of standardization hinders automation, slows regulatory reporting, and increases operational risk. Digitizing CSAs into machine-readable, schema-adherent formats is therefore essential for enabling scalable collateral workflows, ensuring compliance, and improving transparency in financial markets.

The industry standard for this digital transformation is the CDM, a standardized, machine-readable blueprint for financial products, trades, and their lifecycle events [2]. Although the CDM provides a clear structure, converting unstructured legal agreements like CSAs from the over-the-counter (OTC) derivatives market into accurate CDM representations is still a major challenge. Significant industry efforts, including recent benchmarking work [4] by ISDA, are currently dedicated to addressing this complex conversion challenge.

Additionally, recent advancements in LLMs and Retrieval-Augmented Generation (RAG) offer promising avenues for automating this conversion. However, two major challenges persist, particularly for financial institutions:

- **Schema and Syntactical Adherence:** Direct generation by LLMs often struggles to consistently produce complex, deeply nested JSON outputs that adhere strictly to the precise CDM schema, leading to validation errors and integration issues.
- **Accessibility and Cost:** Industry-leading benchmarks for contract digitization have demonstrated high accuracy predominantly through the use of very large, proprietary LLMs (e.g., GPT-4o, Claude Opus) [4]. These models pose significant cost, data privacy, and deployment challenges that restrict their immediate adoption by many institutions, underscoring the need for competitive, open-source alternatives.

To address the adherence challenge, we previously introduced CDMizer [7], a template-driven framework that enforces syntactic correctness and schema adherence during the contract-to-CDM conversion process. CDMizer employs a recursive traversal approach, integrating RAG, to populate pre-generated, minimal CDM templates based on hierarchical context. In its initial application,

CDMizer was used for converting OTC derivative contracts, demonstrating its ability to handle complex financial agreements and structure them within the CDM framework.

In this work, we build upon the CDMizer framework to tackle the accessibility and cost challenge by applying it to a critical, real-world task defined by ISDA. Specifically, we perform a rigorous comparative analysis using the **Qwen3-30B**¹, which is a state-of-the-art, yet significantly smaller and open-source-compatible model, against the published performance of proprietary LLMs on the ISDA CSA benchmark.

The core contributions of this paper are:

- **Methodology Generalization:** We validate the application of the CDMizer framework for accurately encoding ISDA CSA clauses, extending its use to a new document type beyond its initial application to OTC derivative contracts.
- **Resource-Efficient Benchmarking:** We quantify the performance of CDMizer when integrated with the highly resource-efficient Qwen3-30B model, providing a benchmark for accessible, internal deployment in comparison to larger models.
- **Performance vs. Accessibility Trade-offs:** We analyze the semantic accuracy gap between our resource-constrained approach and commercial State-of-the-Art (SOTA) models, highlighting a practical approach to CDM conversion that balances cost and data privacy constraints.

2 Related Work

The automation of legal contract transformation into standardized formats has been explored with the use of LLMs and RAG. However, most research has focused on structured contracts, leaving the conversion of unstructured financial documents, like CSAs, largely unaddressed.

Early efforts in automating OTC derivatives, such as [3] and [1], relied on structured inputs, addressing only specific tasks like automating termination procedures or transforming standardized agreements into smart contracts. These works did not explore the challenges of handling unstructured legal text in complex contracts.

In contract generation, LLMs have been applied to simpler contracts, like those in [5] and [10], but these studies focused on domains like health insurance, which lack the complexity of financial agreements. [6] explored contract translation for blockchain using structured representations but did not tackle unstructured legal language.

The integration of RAG has proven useful in improving LLM-generated content, as demonstrated by [8], which enhanced the accuracy and coherence of contract generation. This approach is particularly valuable for financial contracts, where extracting structured information from unstructured text is critical. However, existing evaluation methods, such as those used in [5] and [9], fall short when it comes to assessing the complex and nuanced nature of financial contracts.

In our previous work [7], we introduced the CDMizer framework to address these challenges by applying a template-driven approach to convert OTC derivative contracts into CDM representations. This method ensured syntactic correctness and schema adherence, making it effective for simpler derivative contracts. This paper

extends CDMizer by applying it to a new dataset of CSA contracts and evaluating its performance against an industry benchmark.

3 Experimental Setup

Our experimental approach leverages the core architecture of **CDMizer** [7] to execute the contract-to-CDM conversion task. As outlined in the workflow diagram shown in Figure 1, the process follows three distinct stages - 1) creating templates specific to the target CSA clauses from the CDM schema and relevant examples, 2) populating these templates using the chosen LLM, and 3) evaluating the generated CDM outputs.

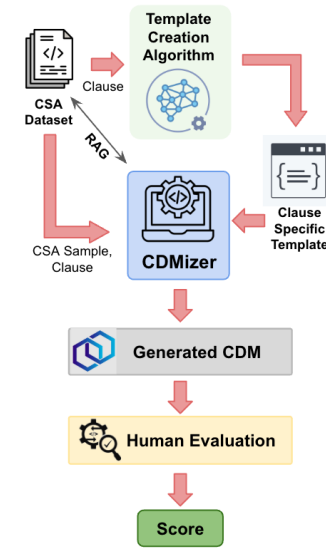


Figure 1: Overall Experimental Workflow

3.1 Template Creation and Scope

We focused on the five critical CSA clauses identified in the ISDA benchmarking study [4]: **Base Currency**, **Eligible Currency**, **Rounding**, **Minimum Transfer Amount (MTA)**, and **Threshold**.

As detailed in the methodology of the CDMizer framework [7], template creation is a deterministic process (Algorithm 1 in the cited work). It involves recursively traversing the full CDM schema and pruning all fields not relevant to the target clauses, resulting in a minimal, syntactically correct JSON structure. This process yielded four unique templates: one for the combined **Base Currency** and **Eligible Currency** fields, and one each for **Rounding**, **MTA**, and **Threshold**. This template-driven structure guarantees that the output JSON will always be compliant with the defined CDM schema, thus achieving 100% syntactical correctness and schema adherence, an advantage over direct generation methods.

3.2 LLM and Inference Setup

We selected the **Qwen3-30B** LLM for this study. This model represents a highly capable, modern open-source architecture with 30.5

¹<https://huggingface.co/Qwen/Qwen3-30B-A3B-Instruct-2507>

Table 1: Comparative scores of different models (With RAG)

Model / Method	Base and Eligible Currency	MTA	Threshold	Rounding
GPT-4o (~200B)	98.00	93.00	92.00	98.00
GPT-o1 (~200B)	100.00	93.00	90.00	100.00
Claude 3.5 Sonnet v2 (175B+)	100.00	85.00	88.00	100.00
Claude 3.7 Sonnet (100B+)	100.00	90.00	90.00	100.00
Claude 3 Opus (100B+)	98.00	87.00	90.00	95.00
DeepSeek R1 (671B, 37B active)	88.00	83.00	85.00	98.00
Nova Pro (~90B)	95.00	72.00	90.00	95.00
Llama 3.3 (70B)	98.00	70.00	82.00	98.00
CDMizer with Qwen3 (30.5B)	97.88	79.15	88.24	93.39
Rank (out of 9 models)	7th	7th	6th	9th

Table 2: Comparative scores of different models (Without RAG)

Model / Method	Base and Eligible Currency	MTA	Threshold	Rounding
GPT-4o (~200B)	98.00	37.00	87.00	97.00
GPT-o1 (~200B)	97.00	37.00	80.00	98.00
Claude 3.5 Sonnet v2 (175B+)	98.00	30.00	40.00	98.00
Claude 3.7 Sonnet (100B+)	98.00	38.00	40.00	90.00
Claude 3 Opus (100B+)	95.00	35.00	88.00	92.00
DeepSeek R1 (671B, 37B active)	85.00	37.00	68.00	90.00
Nova Pro (~90B)	88.00	37.00	88.00	88.00
Llama 3.3 70B (70B)	93.00	37.00	77.00	93.00
CDMizer with Qwen3 (30.5B)	88.81	58.31	46.22	82.37
Rank (out of 9 models)	7th	1st	7th	9th

billion total parameters (and approximately 3.3 billion active parameters per inference), classifying it as a *significantly smaller* model compared to the proprietary SOTA models used in the ISDA benchmark (e.g., those estimated to be over 100 billion parameters). Our primary goal is to assess the utility of this highly accessible model through the CDMizer framework for this complex, domain-specific task.

We tested two configurations for each clause:

- (1) **Without RAG:** The LLM receives only the clause-specific template, the schema definition, and the natural language contract text.
- (2) **With RAG:** The LLM receives the prompt augmented with contextual examples retrieved from a RAG knowledge base.

For RAG implementation, the knowledge base consisted of the 60 ground truth CSA examples. We employed a *leave-one-out* approach: when processing a given CSA, we excluded it from the retrieval pool and instead allowed the RAG system to draw contextual examples only from the remaining 59. In other words, each contract was evaluated as if it were a “new” unseen document, while the model could still learn patterns from similar but different CSAs. This ensured that no CSA was ever evaluated against its own ground truth, preserving fairness and preventing information leakage.

3.3 Dataset and Evaluation

Dataset: We used a dataset of 60 real-world CSA samples, which are the same source contracts leveraged in the ISDA benchmarking study [4]. Using this dataset ensures direct comparability with industry-standard benchmarks and provides a realistic assessment of performance on authentic legal documents. Note that the **Threshold** clause was applicable to 37 of the samples, resulting in a smaller test count for that clause.

Evaluation Metrics: The accuracy of the generated CDM outputs was assessed through manual, human evaluation, mirroring the ISDA benchmark’s methodology to ensure consistency and relevance. We manually compared the generated CDM JSON against the ground truth for each clause and assigned a final score on a scale of 0 to 100. The ground truth data for these CSA samples was human-annotated by ISDA experts, making it a reliable source of reference for evaluating the performance of our model. The evaluation score reflects:

- Correct schema usage (which CDMizer already guarantees).
- Correct extraction of the target information from the contract text.
- Correct placement and representation of the extracted values (e.g., currency codes, numeric values, party names) within the structured CDM fields.

To illustrate the nature of the dataset, following is an example of an MTA clause and its corresponding CDM representation [4].

An MTA Clause Excerpt from a CSA

Paragraph 12. Definitions “Minimum Transfer Amount” means, with respect to a party, the amount specified as such for that party in Paragraph 13; if no amount is specified, zero. Paragraph 13(vii) Minimum Transfer Amount (A) “Minimum Transfer Amount” means with respect to Party A: US Dollars 5,000,000. “Minimum Transfer Amount” means with respect to Party B: US Dollars 5,000,000.

CDM JSON Representation for the MTA Clause Excerpt

```
"agreementTerms": {
  "agreement": {
    "creditSupportAgreementElections": {
      "minimumTransferAmount": [
        {
          "mtaType": {
            "fixedAmount": {
              "amount": 5000000,
              "currency": "USD",
              "party": "PARTY_1"
            }
          }
        },
        {
          "mtaType": {
            "fixedAmount": {
              "amount": 5000000,
              "currency": "USD",
              "party": "PARTY_2"
            }
          }
        }
      ]
    }
  }
}
```

4 Results and Discussion

The results of our evaluation are benchmarked against models tested by ISDA’s benchmarking work [4], ranging from large proprietary models (e.g., GPT-4o, ~200B) to open-source alternatives (e.g., Llama 3.3, 70B). The full comparison is detailed in Table 1 and Table 2.

Despite being significantly smaller, the Qwen3-30B model demonstrated competitive performance in CSA-to-CDM conversion. A key strength of the CDMizer framework is its template-driven approach, which guarantees strict schema adherence. This was particularly advantageous when using a smaller model like Qwen3-30B, as it ensured accurate, structured outputs, which are often challenging for direct LLM generation, especially with complex documents like CSAs.

A noteworthy result was seen in the MTA clause: when RAG was not used, Qwen3-30B ranked first in performance. This was primarily due to the CDMizer framework’s strict schema adherence,

which ensured the correct structure of the output, boosting its score even when compared to larger models. For more standardized clauses like Base and Eligible Currency and Rounding, Qwen3-30B also performed strongly, although larger models like GPT-4o and Claude achieved higher scores across most of the clauses.

When RAG was incorporated, it enhanced the performance of all models, with the most significant improvement observed in Qwen3-30B. RAG helped increase semantic coverage and extraction accuracy, especially for more complex clauses. However, even without RAG, the Qwen3-30B model was able to achieve competitive results, showcasing the potential of smaller open-source models for contract digitization.

Overall, CDMizer with Qwen3-30B, while not ranking at the top in all aspects, achieved scores close to the highest-performing models, demonstrating its ability to generate syntactically correct and semantically meaningful CDM representations on par with larger models, all while remaining computationally efficient. This positions CDMizer as a resource-efficient solution for CSA-to-CDM conversion, demonstrating the framework’s broader applicability beyond its initial use for derivatives contracts.

5 Conclusion

In this work, we extended the CDMizer framework [7] to address the CSA-to-CDM conversion, utilizing the Qwen3-30B model and benchmarking its performance against the ISDA standard. Despite its smaller size, Qwen3-30B with CDMizer showed competitive results, especially when integrated with RAG, achieving accuracy levels close to those of larger models. This highlights the potential of resource-efficient solutions for automating contract digitization at scale without compromising on performance. Future work could focus on expanding CDMizer to handle a broader range of legal documents, enhancing the framework’s generalization capabilities. Additionally, exploring advanced fine-tuning techniques for domain-specific tasks and incorporating additional contextual knowledge could enhance the framework’s robustness, further improving the accuracy and reliability of the conversion process.

Acknowledgments

We acknowledge the support from NSF IUCRC CRAFT center research grant (CRAFT Grant #22018) for this research. The opinions expressed in this publication do not necessarily represent the views of NSF IUCRC CRAFT. We are also grateful for the advice and resources from our CRAFT Industry Board members, and David Lee from ISDA in shaping this work.

References

- [1] Matthew Armitage. 2022. Trust, Confidence, and Automation: The ISDA Master Agreement as a Smart Contract. *Business Law Review* 43, 2 (2022).
- [2] FINOS. 2024. Overview of the FINOS CDM. <https://cdm.finos.org/docs/cdm-overview/#purpose>.
- [3] Christian P Fries and Peter Kohl-Landgraf. 2018. Smart Derivative Contracts (Detaching Transactions from Counterparty Credit Risk: Specification, Parametrisation, Valuation). Available at SSRN 3163074 (2018).
- [4] ISDA. 2025. Benchmarking Generative AI for CSA Clause Extraction and CDM Representation. <https://www.isda.org/2025/05/15/benchmarking-generative-ai-for-csa-clause-extraction-and-cdm-representation/>.
- [5] Inwon Kang, William Van Woensel, and Oshani Seneviratne. 2024. Using Large Language Models for Generating Smart Contracts for Health Insurance from Textual Policies. In *AI for Health Equity and Fairness: Leveraging AI to Address Social Determinants of Health*. Springer, 129–146.

- [6] Rabimba Karanjai, Lei Xudagger, and Weidong Shi. 2024. SolMover: Feasibility of Using LLMs for Translating Smart Contracts. In *2024 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*. IEEE, 1–3.
- [7] Maruf Ahmed Mridul, Ian Sloyan, Aparna Gupta, and Oshani Seneviratne. 2025. AI4Contracts: LLM & RAG-Powered Encoding of Financial Derivative Contracts. arXiv:2506.01063 [cs.IR] <https://arxiv.org/abs/2506.01063>
- [8] Md Rizwan Parvez, Wasi Uddin Ahmad, Saikat Chakraborty, Baishakhi Ray, and Kai-Wei Chang. 2021. Retrieval augmented code generation and summarization. *arXiv preprint arXiv:2108.11601* (2021).
- [9] Priyan Vaithilingam, Tianyi Zhang, and Elena L Glassman. 2022. Expectation vs. experience: Evaluating the usability of code generation tools powered by large language models. In *Chi conference on human factors in computing systems extended abstracts*. 1–7.
- [10] William Van Woensel, Manan Shukla, and Oshani Seneviratne. 2023. Translating Clinical Decision Logic Within Knowledge Graphs to Smart Contracts.. In *SeWeBMeDA@ESWC*.