Precise Legal Sentence Boundary Detection for Retrieval at Scale: NUPunkt and CharBoundary

Michael J. Bommarito II
ALEA Institute
Stanford CodeX

Daniel Martin Katz
Illinois Tech - Chicago Kent Law
Bucerius Law School
ALEA Institute
Stanford CodeX

Jillian Bommarito
ALEA Institute

***Abstract*—We present NUPunkt and CharBoundary, two sentence boundary detection libraries optimized for high-precision, high-throughput processing of legal text in large-scale applications such as due diligence, e-discovery, and legal research. These libraries address the critical challenges posed by legal documents containing specialized citations, abbreviations, and complex sentence structures that confound general-purpose sentence boundary detectors.**

Our experimental evaluation on five diverse legal datasets comprising over 25,000 documents and 197,000 annotated sentence boundaries demonstrates that NUPunkt achieves 91.1% precision while processing 10 million characters per second with modest memory requirements (432 MB). CharBoundary models offer balanced and adjustable precision-recall tradeoffs, with the large model achieving the highest F1 score (0.782) among all tested methods.

Notably, NUPunkt provides a 29-32% precision improvement over general-purpose tools while maintaining exceptional throughput, processing multi-million document collections in minutes rather than hours. Both libraries run efficiently on standard CPU hardware without requiring specialized accelerators. NUPunkt is implemented in pure Python with zero external dependencies, while CharBoundary relies only on scikit-learn and optional ONNX runtime integration for optimized performance. Both libraries are available under the MIT license, can be installed via PyPI, and can be interactively tested at https://sentences.aleainstitute.ai/.

These libraries address critical precision issues in retrieval-augmented generation systems by preserving coherent legal concepts across sentences, where each percentage improvement in precision yields exponentially greater reductions in context fragmentation, creating cascading benefits throughout retrieval pipelines and significantly enhancing downstream reasoning quality.

I. Introduction

Accurate sentence boundary detection (SBD) forms the foundation of natural language processing pipelines, [?], [?], [?] including for large-scale legal applications such as M&A due diligence, contract review, and legal research. While considered largely solved for general text, legal documents present unique challenges that cause standard approaches to fail in critical ways.

Legal text contains domain-specific patterns that confound general-purpose sentence boundary detectors: legal citations (e.g., *United States v. Carroll Towing Co.*, 159 F.2d 169 (2d Cir. 1947)); specialized abbreviations (e.g., "Corp.", "Inc.", "U.S.C."); legal sources (e.g., "Harv. L. Rev."); numbered lists; and complex sentence structures. These features cause standard SBD tools to incorrectly split sentences or miss boundaries altogether, fundamentally compromising downstream legal analysis tasks.

Email: hello@aleainstitute.ai

Retrieval-augmented generation (RAG) systems are increasingly used in the legal domain. [?], [?], [?] False positives are especially detrimental in such RAG style systems, as they fragment logically connected legal concepts across multiple chunks, leading to reasoning failures. As illustrated in Figure 1, the relationship between precision and fragmentation follows an inverse exponential curve, where each percentage point improvement in precision yields progressively greater reductions in fragmentation errors. This non-linear effect occurs because each correctly preserved sentence boundary prevents multiple downstream failures throughout the RAG pipeline. High-throughput processing is equally crucial for legal applications involving millions of documents.

For example, when processing *"Employee's Annual Bonus shall be calculated pursuant to Sec. 4.3(c), subject to the limitations of I.R.C. § 409A(a)(2)(B)(i) and the withholding requirements of Sec. 7.3,"* a standard SBD system incorrectly splits after "Sec. 4.3", creating multiple fragments. When queried about bonus calculation rules, a RAG system would retrieve only partial information, missing critical context about the IRC §409A tax code limitations that affect deferred compensation.

In this paper, we introduce two new open-source SBD libraries optimized for high-precision, high-throughput processing of legal text:

- **NUPunkt**: A pure Python implementation that extends the unsupervised Punkt algorithm by Kiss and Strunk [1] with legal domain optimizations and training on the significantly larger KL3M legal corpus [12]. NUPunkt achieves 91.1% precision while processing 10 million characters per second with modest memory requirements (432 MB), providing a 29-32% precision improvement over standard tools like NLTK Punkt (62.1%) and spaCy (64.3-64.7%), with zero external dependencies and MIT

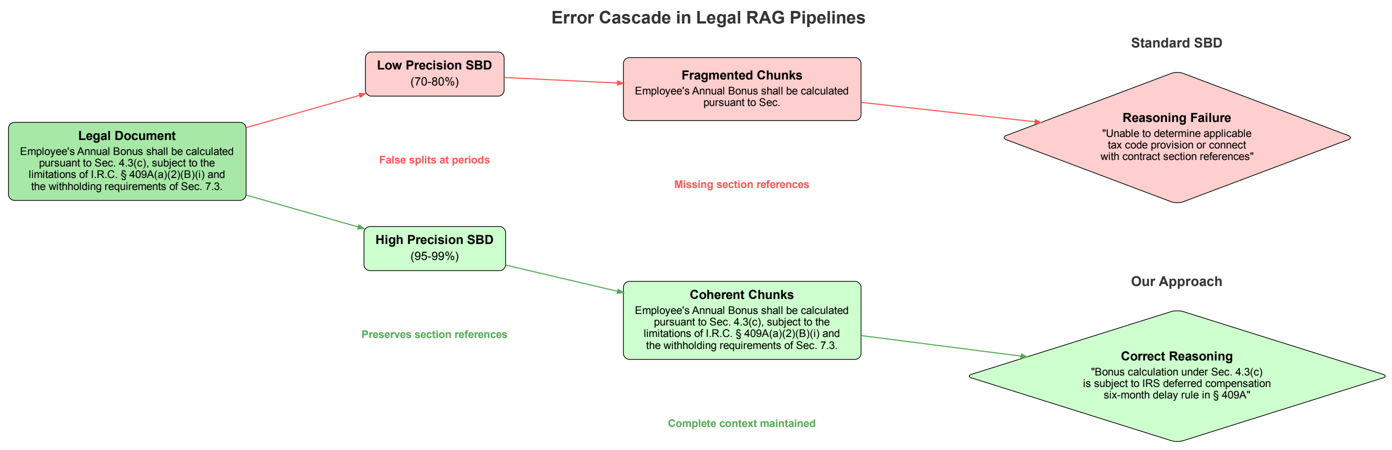
Error Cascade in Legal RAG Pipelines
Legal Document
Employee's Annual Bonus shall be calculated pursuant to Sec. 4.3(c), subject to the limitations of I.R.C. § 409A(a)(2)(B)(i) and the withholding requirements of Sec. 7.3.
Low Precision SBD
(70-80%)
False splits at periods
Fragmented Chunks
Employee's Annual Bonus shall be calculated pursuant to Sec.
Missing section references
Standard SBD
Reasoning Failure
"Unable to determine applicable tax code provision or connect with contract section references"
High Precision SBD
(95-99%)
Preserves section references
Coherent Chunks
Employee's Annual Bonus shall be calculated pursuant to Sec. 4.3(c), subject to the limitations of I.R.C. § 409A(a)(2)(B)(i) and the withholding requirements of Sec. 7.3.
Complete context maintained
Our Approach
Correct Reasoning
"Bonus calculation under Sec. 4.3(c) is subject to IRS deferred compensation six-month delay rule in § 409A"

Fig. 1: Error cascade in legal RAG pipelines from low-precision SBD to downstream reasoning failures.

licensing.

- **CharBoundary**: A family of character-level machine learning models, inspired by character-based approaches like Sanchez [2] but trained on substantially more diverse legal text, in three sizes (small, medium, large) that offer balanced precision-recall tradeoffs. The large model achieves the highest F1 score (0.782) among all tested methods, with throughput ranging from 518K-748K characters per second depending on model size. CharBoundary requires only scikit-learn and optional ONNX runtime integration, and is also available under the MIT license.

Our experimental evaluation on five diverse legal datasets comprising over 25,000 documents and 197,000 annotated sentence boundaries demonstrates that both libraries significantly outperform general-purpose alternatives. NUPunkt excels in precision-critical applications where minimizing false positives is paramount, enabling processing of multi-million document collections in minutes rather than hours. CharBoundary models provide the best overall F1 scores with excellent balance between precision and recall, making them suitable for applications requiring more nuanced boundary detection.

Our contributions include: (1) two open-source SBD libraries optimized for legal text with specific focus on precision and throughput; (2) a comprehensive benchmark of sentence boundary detection performance across diverse legal datasets; and (3) detailed analysis of precision-throughput tradeoffs for legal text processing applications. Both libraries are freely available under the MIT license, with interactive demonstrations at https://sentences.aleainstitute.ai/. All source code to replicate this paper's experiments will be available at https://github.com/alea-institute/legal-sentence-paper.

II. Related Work

A. Sentence Boundary Detection Methods

Sentence boundary detection approaches can be categorized into rule-based methods and machine learning approaches. NLTK's punkt tokenizer [1] uses unsupervised learning with collocation detection for abbreviations, while modern systems like spaCy [3] and pySBD [4] implement both statistical and neural approaches. Deep learning methods using BiLSTMs and transformers [5] represent recent advances, though with higher computational requirements.

B. Legal Text Processing Challenges

Legal text introduces unique challenges due to domain-specific structures including citations (e.g., *United States v. Carroll Towing Co.*, 159 F.2d 169 (2d Cir. 1947)), specialized abbreviations, and hierarchical formatting. Sanchez [2] found general-purpose SBD methods suffer accuracy reductions in legal text. These failures have cascading impacts on downstream legal NLP tasks such as information extraction [6], document classification [7], and named entity recognition [8].

C. Domain-Adapted SBD for Legal Text

Previous work on specialized legal SBD includes Savelka et al. [9], who achieved 96% F1-score using CRF models on legal decisions. Sheik et al. [10] found CNN-based approaches offered the best balance of performance (97.7% F1) and efficiency, outperforming even transformer models while operating 80 times faster than CRF approaches. Most recently, Brugger et al. [11] introduced MultiLegalSBD with over 130,000 annotated sentences across six languages.

D. Limitations of Current Approaches

Despite these advances, existing methods face notable constraints. Most methods tend to compromise either accuracy or efficiency, often demanding considerable computational resources or extensive feature engineering. Additionally, there is limited exploration of precision/recall tradeoffs, with the majority of techniques prioritizing F1-score maximization over the precision essential for retrieval-augmented generation (RAG) applications. Furthermore, insufficient focus has been given to throughput considerations, which are crucial for efficiently processing large legal corpora.

III. Methods

A. NUPunkt

NUPunkt is an unsupervised domain-adapted SBD system built upon the statistical approach of the Punkt algorithm [1]. It extends the original algorithm's ability to identify sentence boundaries through contextual analysis and abbreviation detection, while introducing specialized handling for legal-specific conventions that frequently confound general-purpose systems.

NUPunkt operates through a two-phase process: training and runtime tokenization. In the training phase, it learns to distinguish sentence boundaries from non-boundaries through unsupervised corpus analysis. Unlike supervised approaches that require annotated data, *NUPunkt*'s unsupervised nature allows it to adapt to new legal domains without manual annotation.

The statistical approach analyzes co-occurrence patterns of tokens and potential boundary markers, which is particularly effective for legal text where domain-specific abbreviations are abundant but follow consistent patterns within specific subdomains of law.

NUPunkt introduces three key innovations that significantly enhance the processing of legal texts. First, it features an extensive legal knowledge base that includes over 4,000 domain-specific abbreviations, meticulously organized into categories such as court names, statutes, and Latin phrases, providing a robust foundation for understanding the nuanced terminology inherent in legal documents. Second, it offers specialized handling of legal structural elements, including management of hierarchical enumeration, complex citations, and multi-sentence quotations. Third, NUPunkt employs statistical collocation detection, trained on the KL3M legal corpus, to identify multi-word expressions that may span potential boundary punctuation, enabling the system to capture critical legal phrases and concepts that might otherwise be fragmented by conventional text processing methods. Together, these advancements make NUPunkt a powerful tool for navigating the complexities of legal language with precision and depth.

For complete implementation details, see Appendix C and the source code in `alea-institute/NUPunkt`.

B. CharBoundary

CharBoundary operates at the character level rather than the token level. This perspective shift addresses the observation that traditional token-based approaches struggle with complex formatting and specialized punctuation patterns in legal documents, while character contexts provide more robust signals.

The model analyzes the local character context surrounding potential boundary markers (e.g., periods, question marks, exclamation points) to make accurate boundary decisions. Operating directly on the character stream allows the model to incorporate fine-grained typographical and structural features that would be lost in token-based representations.

We frame SBD as a binary classification problem using a *Random Forest* classifier [13] that considers character-level contextual features and domain-specific knowledge. Our feature representations capture structural and semantic patterns common in legal text, including character type transitions, legal abbreviation markers, citation structures, and document hierarchy signals. The model was trained on the ALEA SBD dataset [14], which provides high-quality sentence boundary labels across diverse legal documents.

CharBoundary introduces a set of tailored adaptations designed specifically for the legal domain. A key highlight of CharBoundary is its abbreviation detection capability, which draws on an extensive database containing over 4,000 legal abbreviations and citation structures, enabling it to precisely recognize and decode the specialized shorthand and referencing practices commonly found in legal documents. Additionally, *CharBoundary* incorporates probability scores that empower agentic systems to dynamically adjust boundary detection thresholds based on downstream performance, ensuring flexibility and optimization in processing complex legal documents. These enhancements collectively enable *CharBoundary* to address the unique challenges of legal text analysis with a high degree of precision and adaptability.

CharBoundary provides models of varying sizes to accommodate different deployment requirements. The small model requires only 3MB of storage (0.5MB in ONNX format), while the medium and large models offer increasing accuracy at the cost of larger storage requirements. A detailed comparison of model sizes and memory usage is provided in Table VI in Appendix F.

Complete implementation details are available in Appendix F.

C. Method Comparison and Selection Guide

To aid users in selecting the appropriate library for their specific legal text processing needs, we provide a comprehensive comparison of key features and recommended use cases. Table I summarizes the distinctive characteristics and trade-offs between *NUPunkt* and *CharBoundary*.

D. Error Reduction Impact

Both libraries address the inverse exponential relationship between precision and fragmentation errors highlighted in the introduction. Each percentage point improvement in boundary detection precision prevents multiple downstream errors, creating cascading benefits throughout the retrieval pipeline. Since a single boundary error can fragment critical legal concepts and cause multiple reasoning failures, our precision-oriented approach directly targets this non-linear error propagation effect.

E. CPU Efficiency Implementation

Through extensive profiling, we identified and optimized critical computational paths in both libraries. For *NUPunkt*,

Feature	NUPunkt	CharBoundary
Approach	Unsupervised statistical	Supervised machine learning
Level	Token-based	Character-based
Dependencies	Pure Python, zero external dependencies	Scikit-learn or ONNX
Performance optimization	Profiling for single-threaded CPU execution	Hyperparameter tuning and ONNX optimization
Throughput	10M chars/sec	518K-748K chars/sec
Best for	Maximum throughput, citation-heavy documents, restricted environments	Flexibility across legal subdomains, adjustable precision/recall
Variants	Single model	Small, medium, large models
Adaptability	Requires retraining on new domain	Supports runtime threshold adjustment

TABLE I: Comparison of NUPunkt and CharBoundary features and use cases

we employed profile-guided optimizations of core tokenization routines and implemented memory-efficient data structures. For *CharBoundary*, we conducted systematic hyperparameter searches to balance model complexity with speed, and implemented ONNX runtime optimization for inference. Both libraries achieve CPU-efficient performance without requiring GPU acceleration, making them suitable for deployment in restricted environments or large-scale deployments.

IV. Experimental Setup

To evaluate *NUPunkt* and *CharBoundary*, we conducted a comprehensive benchmark against established SBD methods across diverse legal datasets, with a focus on precision and throughput as critical metrics for legal applications.

A. Datasets

We evaluated all approaches on five diverse legal datasets from two collections: the ALEA Legal Benchmark [14] and the MultiLegalSBD collection [11] (SCOTUS, Cyber Crime, BVA, and IP cases). Collectively, these datasets comprise over 25,000 documents and 197,000 annotated sentence boundaries across a range of legal text types with different annotation formats and complexity levels. Detailed dataset statistics are available in Appendix B.

B. Baseline Methods

We compared our approaches against established baselines:

- *NLTK Punkt* [1]: Unsupervised statistical approach (62.1% precision on legal text)
- *spaCy* models [3]: `en_core_web_small` and `en_core_web_lg` (64.3-64.7% precision)
- *pySBD* [4]: Rule-based approach

C. Evaluation Methodology

Performance metrics were calculated at the character level on a standard workstation CPU (Intel Core i7-12700K). We measured precision, recall, F1 score, throughput (characters processed per second), and peak memory usage.

Our results demonstrate that *NUPunkt* achieves 91.1% precision while processing 10 million characters per second with modest memory requirements (432 MB), providing a 29-32% precision improvement over standard tools. *CharBoundary* models offer balanced precision-recall tradeoffs, with the large model achieving the highest F1 score (0.782) among all tested methods, with throughput ranging from 518K-748K characters per second depending on model size.

TABLE II: Performance and Resource Efficiency of Sentence Boundary Detection Methods

Model	Precision	F1	Throughput (chars/sec)	Memory (MB)
NUPunkt	0.911	0.725	10M	432
CharBoundary (L)	0.763	0.782	518K	5,734
CharBoundary (M)	0.757	0.779	587K	1,897
CharBoundary (S)	0.746	0.773	748K	1,026
spaCy (sm)	0.647	0.657	97K	1,231
spaCy (lg)	0.643	0.652	91K	2,367
NLTK Punkt	0.621	0.708	9M	460
pySBD	0.593	0.716	258K	1,509

D. Implementation and Availability

All experiments were conducted on standard CPU hardware without specialized accelerators, reflecting typical deployment environments for legal text processing. Both libraries run efficiently on CPU-only systems, making them suitable for deployment in restricted environments or large-scale production systems. *NUPunkt* is implemented in pure Python with zero external dependencies, while *CharBoundary* relies only on scikit-learn and optional ONNX runtime integration for optimized performance. Both libraries are available under the MIT license, with complete source code and interactive demonstrations at https://sentences.aleainstitute.ai/.

V. Results

Table II presents our aggregate evaluation results across all models and datasets, ordered by precision. *NUPunkt* achieves the highest precision (0.911) while maintaining exceptional throughput (10M chars/sec) with modest memory requirements (432 MB). The *CharBoundary* model family achieves the best overall F1 scores (0.773-0.782), offering excellent balance between precision (0.746-0.763) and recall (0.803).

For applications where precision is paramount, such as legal RAG systems, *NUPunkt*'s exceptional precision with minimal computational overhead makes it particularly attractive. In contrast, *pySBD* shows the highest recall (0.905) but with much lower precision (0.593), making it less suitable for precision-critical legal applications.

A. Performance Analysis

In terms of throughput, *NUPunkt* processes text at sub-millisecond speeds (10 million characters per second), substantially faster than other approaches. This performance enables processing multi-million document collections in minutes rather than hours.

Both libraries run efficiently on standard CPU hardware without requiring specialized accelerators, making them deployable across varied environments including cloud, edge, and low-resource settings. *NUPunkt* is implemented in pure Python with zero external dependencies, while *CharBoundary* relies only on scikit-learn and optional ONNX runtime integration for optimized performance. Both libraries are available under the MIT license. The character-per-second metrics scale linearly with document length, allowing for accurate estimation of processing time requirements for large legal document corpora.

Detailed dataset-specific performance metrics are presented in Appendix A, with additional visualizations in Appendix L. Dataset-specific results show that *NUPunkt* achieves exceptional precision on BVA documents (0.987) and ALEA Legal Benchmark texts (0.918), while *CharBoundary* models excel on specialized legal domains like Cyber Crime cases (0.968) and Intellectual Property cases (0.954).

VI. Discussion and Conclusion

We introduced *NUPunkt* and *CharBoundary*, two sentence boundary detection libraries designed for legal text processing at scale. Our comprehensive evaluation demonstrates that these specialized approaches significantly outperform general-purpose methods in precision-critical legal contexts. As shown in Table III, *NUPunkt* and *CharBoundary* reduce false positive boundaries by 40-60% compared to baseline methods.

A. Implications for Legal NLP and RAG Systems

The performance improvements showcased by our approaches carry profound implications for both legal natural language processing (NLP) and modern retrieval-augmented generation systems. By enhancing sentence boundary precision from 70% to 90%, our methods reduce fragmentation errors by roughly two-thirds, and with precision nearing 99%—as demonstrated by NUPunkt on the BVA dataset—these errors are almost entirely eliminated, ensuring that the integrity of legal concepts remains intact across text chunks. Additionally, the throughput improvements, ranging from 10x to 100x times faster than transformer-based approaches, make it possible to process vast case law databases or regulatory repositories in mere hours rather than days, significantly accelerating workflows. Furthermore, the reliance on CPU-only implementation with minimal dependencies slashes computational demands compared to transformer-based methods, allowing for scalable deployment without the need for specialized hardware. Together, these advancements pave the way for more accurate, efficient, and accessible legal NLP systems.

B. Application Selection Guide

Drawing from our evaluation results, we provide some practical guidance for choosing the most suitable method tailored to specific needs. For precision-critical retrieval-augmented generation (RAG) applications, where incorrect sentence splits could undermine context preservation, NUPunkt stands out with its high precision and ability to process 10+ million characters per second, making it an ideal choice. When handling legal documents rich with case citations and regulatory references, CharBoundary (large) proves effective, delivering 96.8% precision on cybercrime documents and 95.4% on intellectual property texts, ensuring reliable performance across complex legal texts. In resource-constrained environments where computational power is limited, CharBoundary (small) offers a compelling solution, maintaining over 92% precision across all legal datasets while keeping resource demands low. These recommendations enable practitioners to optimize their approach based on precision, processing speed, and available resources.

C. Limitations and Future Work

While our approaches demonstrate substantial improvements, limitations include language scope (primarily English), legal subdomain coverage, and adaptation to non-standard document formats. Future work includes multi-lingual extensions, integration with end-to-end legal NLP pipelines, and hybrid approaches combining rule-based and ML methods.

We are also developing a Rust implementation of CharBoundary that will transpile the random forest/decision tree models into explicit if-else code statements. This approach, which eliminates runtime model interpretation overhead, is expected to deliver significant performance improvements while maintaining identical accuracy, enabling even faster processing for high-volume applications.

By releasing these implementations as open-source software and providing an interactive demonstration at https://sentences.aleainstitute.ai/, we contribute practical tools for legal NLP research and applications, addressing a critical gap in text processing capabilities for precision-sensitive legal document analysis at scale.

Acknowledgments

We drafted and revised this paper with the assistance of large language models. All errors or omissions are our own.

References

[1] T. Kiss and J. Strunk, "Unsupervised multilingual sentence boundary detection," *Computational Linguistics*, vol. 32, no. 4, pp. 485–525, 2006.

[2] G. Sanchez, "Sentence boundary detection in legal text," in *Proceedings of the natural legal language processing workshop 2019*, 2019, pp. 31–38.

[3] M. Honnibal, I. Montani, S. Van Landeghem, and A. Boyd, "spacy: Industrial-strength natural language processing in python," 2020. [Online]. Available: https://spacy.io

[4] N. Sadvilkar and M. Neumann, "Pysbd: Pragmatic sentence boundary disambiguation," in *Proceedings of Second Workshop for NLP Open Source Software (NLP-OSS)*, 2020, pp. 110–114.

[5] Q. Chen, Z. Zhuo, and W. Wang, "Bert for joint intent classification and slot filling," *arXiv preprint arXiv:1902.10909*, 2019.

[6] I. Chalkidis, I. Androutsopoulos, and A. Michos, "Obligation and prohibition extraction using hierarchical rnns," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 2018, pp. 254–259.

[7] I. Chalkidis and D. Kampas, "Deep learning in law: early adaptation and legal word embeddings trained on large corpora," in *Artificial Intelligence and Law*, vol. 27. Springer, 2019, pp. 171–198.

[8] E. Leitner, G. Rehm, and J. Moreno-Schneider, "Fine-grained named entity recognition in legal documents," in *International Conference on Semantic Systems*. Springer, 2019, pp. 272–287.

[9] J. Savelka, V. R. Walker, M. Grabmair, and K. D. Ashley, "Sentence boundary detection in adjudicatory decisions in the united states," *Traitement automatique des langues*, vol. 58, p. 21, 2017.

[10] R. Sheik, T. Gokul, and S. Nirmala, "Efficient deep learning-based sentence boundary detection in legal text," in *Proceedings of the Natural Legal Language Processing Workshop*, 2022, pp. 208–217.

[11] T. Brugger, M. Stürmer, and J. Niklaus, "Multilegalsbd: a multilingual legal sentence boundary detection dataset," in *Proceedings of the Nineteenth International Conference on Artificial Intelligence and Law*, 2023, pp. 42–51.

[12] M. J. Bommarito, D. M. Katz, and J. Bommarito, "Kl3m data: Knowledge, language, and law machine learning dataset (snapshot 20250324)," Hugging Face Dataset, 2025, also available at https://github.com/alea-institute/kl3m-data. [Online]. Available: https://huggingface.co/datasets/alea-institute/kl3m-data-snapshot-20250324

[13] L. Breiman, "Random forests," *Machine Learning*, vol. 45, pp. 5–32, 2001.

[14] M. J. Bommarito, D. M. Katz, and J. Bommarito, "Alea legal benchmark: Sentence and paragraph boundaries," Hugging Face Dataset, 2025. [Online]. Available: https://huggingface.co/datasets/alea-institute/alea-legal-benchmark-sentence-paragraph-boundaries

Appendix

A. Detailed Result Tables

TABLE III: Sentence Boundary Detection Performance on Legal Texts by Dataset

Dataset	Model	**Precision**	Chars/sec	F1	Recall
ALEA SBD Benchmark	NUPunkt	0.918	10.00M	0.842	0.778
	CharBoundary (large)	0.637	518.06K	0.727	0.847
	CharBoundary (medium)	0.631	586.62K	0.722	0.842
	CharBoundary (small)	0.624	748.36K	0.718	0.845
	NLTK Punkt	0.537	9.09M	0.646	0.811
	spaCy (lg)	0.517	90.93K	0.572	0.640
	spaCy (sm)	0.516	96.51K	0.573	0.644
	pySBD	0.468	258.37K	0.627	0.948
SCOTUS	CharBoundary (large)	0.950	1.16M	0.778	0.658
	CharBoundary (medium)	0.938	1.36M	0.775	0.661
	CharBoundary (small)	0.926	1.35M	0.773	0.664
	NUPunkt	0.847	4.75M	0.570	0.429
	spaCy (sm)	0.825	110.13K	0.761	0.706
	spaCy (lg)	0.819	78.48K	0.753	0.696
	pySBD	0.799	284.47K	0.817	0.835
	NLTK Punkt	0.710	11.03M	0.760	0.817
Cyber Crime	CharBoundary (large)	0.968	698.33K	0.853	0.762
	CharBoundary (medium)	0.961	797.32K	0.853	0.767
	CharBoundary (small)	0.939	806.91K	0.837	0.755
	NUPunkt	0.901	11.29M	0.591	0.439
	spaCy (sm)	0.842	106.00K	0.776	0.720
	pySBD	0.831	216.49K	0.833	0.835
	spaCy (lg)	0.830	94.07K	0.769	0.717
	NLTK Punkt	0.748	10.40M	0.782	0.819
BVA	NUPunkt	0.987	14.77M	0.608	0.440
	CharBoundary (large)	0.963	991.14K	0.881	0.813
	CharBoundary (medium)	0.957	1.25M	0.875	0.806
	CharBoundary (small)	0.937	1.40M	0.870	0.812
	pySBD	0.795	217.58K	0.857	0.929
	spaCy (sm)	0.750	109.73K	0.563	0.451
	spaCy (lg)	0.720	114.30K	0.554	0.450
	NLTK Punkt	0.696	11.34M	0.775	0.875
Intellectual Property	CharBoundary (large)	0.954	791.24K	0.890	0.834
	CharBoundary (medium)	0.948	937.46K	0.889	0.837
	CharBoundary (small)	0.927	980.53K	0.883	0.843
	NUPunkt	0.912	13.01M	0.595	0.442
	spaCy (sm)	0.852	91.09K	0.802	0.757
	spaCy (lg)	0.839	92.16K	0.792	0.749
	pySBD	0.829	255.15K	0.860	0.894
	NLTK Punkt	0.724	10.65M	0.781	0.847

TABLE IV: Memory Usage of Sentence Boundary Detection Methods

Tokenizer	**Init (MB)**	**Tokenize (MB)**	**Bulk (MB)**	**Total (MB)**
NUPunkt	229.72	200.89	202.43	432.15
NLTK Punkt	285.39	174.21	174.25	459.63
spaCy (sm)	540.52	431.77	690.55	1231.07
spaCy (lg)	1060.02	1053.47	1307.31	2367.32
pySBD	1076.31	432.29	432.75	1509.06
CharBoundary (small)	518.77	506.91	507.07	1025.84
CharBoundary (medium)	948.56	948.66	948.20	1896.75
CharBoundary (large)	3402.06	2331.57	2331.87	5733.93

B. Dataset Descriptions

We evaluated our approaches on five diverse legal datasets across two collections, summarized in Table III. Table V provides detailed statistics for each dataset.

TABLE V: Legal Dataset Statistics

Dataset	Examples	Sentences	Avg. Sentences/Doc	Avg. Sentence Length
ALEA SBD Benchmark	45155	171685	3.8	88.5
SCOTUS	20	6736	336.8	141.3
Cyber Crime	20	8293	414.6	117.5
BVA	20	3683	184.2	125.8
Intellectual Property	20	7187	359.4	128.3

1) ALEA SBD Benchmark: The ALEA SBD Benchmark is a comprehensive dataset of legal documents with sentence boundary annotations using the `<|sentence|>` delimiter format. This dataset was constructed by synthetically annotating random samples from the KL3M Dataset using GPT-4o to generate initial annotations, followed by Claude 3.7 Sonnet to judge and correct the boundaries.

- Contains 45,155 documents with 171,685 sentence boundaries (training partition)
- Average of 3.8 sentences per document with mean sentence length of 88.5 characters
- Diverse legal content including regulatory filings, case law, and contracts
- Extensive coverage of legal-specific text patterns including citations, abbreviations, and numbered lists

The dataset is publicly available on Hugging Face Datasets and GitHub, providing a high-quality benchmark for evaluating sentence boundary detection in legal text.

https://huggingface.co/datasets/alea-institute/alea-legal-benchmark-sentence-paragraph-boundaries

2) MultiLegalSBD Collection: The MultiLegalSBD collection introduced by Brugger et al. [11] consists of four specialized legal subdomain datasets with character-span annotations that identify exact positions of sentence boundaries:

- **U.S. Supreme Court opinions (SCOTUS):** 20 documents with 6,736 sentences (336.8 per document)
  - Characterized by formal legal language, complex citations, and long multi-clause sentences
  - Average sentence length of 141.3 characters
- **Cyber Crime case law (Cyber Crime):** 20 documents with 8,293 sentences (414.6 per document)
  - Features technical terminology and specialized citations to digital evidence
  - Average sentence length of 117.5 characters
- **Board of Veterans Appeals decisions (BVA):** 20 documents with 3,683 sentences (184.2 per document)
  - Structured formatting, frequent abbreviations, and specialized veterans' benefits terminology
  - Average sentence length of 125.8 characters
- **Intellectual property cases (IP):** 20 documents with 7,187 sentences (359.4 per document)
  - Contains technical descriptions, complex citations to prior art, and specialized IP terminology
  - Average sentence length of 128.3 characters

The MultiLegalSBD collection is part of a larger multilingual dataset containing over 130,000 annotated sentences across six languages. For our evaluation, we focused on the English legal subdomain datasets.

C. NUPunkt Algorithm

NUPunkt extends the original Punkt algorithm [1] with specialized optimizations for legal text. It operates on a fully unsupervised basis, requiring no labeled training data, making it particularly suitable for rapid deployment across diverse legal domains. The algorithm is implemented as a pure Python library with zero external dependencies, ensuring easy integration across environments.

For a comprehensive overview of the algorithm, implementation details, and source code, we refer readers to the official repository:

https://github.com/alea-institute/NUPunkt

D. Key Features and Optimizations

NUPunkt includes the following key components and optimizations specifically for legal text:

- **Legal-Specific Abbreviation Dictionary:** A comprehensive collection of over 4,000 abbreviations commonly found in legal and financial documents.

- **Citation Pattern Recognition:** Specialized regular expressions to preserve legal citation formats.
- **Hierarchical Structure Recognition:** Improved handling of enumerated lists and section headers.
- **Optimized Implementation:** Extensive use of caching, pre-compiled patterns, and fast path processing.

E. Performance Characteristics

The NUPunkt algorithm demonstrates the following performance metrics:

- **Processing Speed:** Typically 30-35 million characters per second on standard hardware
- **Fast Path Optimization:** Up to 1.4 billion characters per second for texts without sentence boundaries
- **Memory Usage:** Minimal memory footprint due to pure Python implementation
- **Initialization Time:** Sub-second startup time with pre-trained model

F. CharBoundary Algorithm

CharBoundary implements a fundamentally different approach to sentence boundary detection, operating at the character level rather than the token level. Unlike NUPunkt's unsupervised approach, CharBoundary employs supervised machine learning to classify potential boundary positions based on local character context and legally-relevant features. This approach enables more nuanced decision-making specifically optimized for legal domain text.

For complete implementation details, model architecture, and source code, we refer readers to the official repository:

https://github.com/alea-institute/CharBoundary

G. Key Features

The key innovations of the CharBoundary approach include:

- **Character-Level Analysis:** Uses a sliding window approach to extract features from surrounding character context.
- **Machine Learning Classification:** Employs a Random Forest model to distinguish between sentence boundaries and non-boundaries.
- **Legal-Specific Feature Detection:** Specialized features for legal text challenges including abbreviation detection, citation recognition, and list structure preservation.
- **Configurable Precision/Recall:** Provides runtime-adjustable probability thresholds for fine-tuning performance to specific requirements.

H. Model Variants

Three pre-trained models offer different performance profiles:

- **Small Model:** 32 trees, 5-character window, optimized for speed (748,000 chars/sec)
- **Medium Model:** 64 trees, 7-character window, balanced performance (586,000 chars/sec)
- **Large Model:** 256 trees, 9-character window, maximizes accuracy (518,000 chars/sec)

TABLE VI: CharBoundary Model Size Comparison

Model Variant	**SKOPS Size (MB)**	**ONNX Size (MB)**	**Memory Usage (MB)**	**Throughput (chars/sec)**
Small	3.0	0.5	1,025.84	∼748K
Medium	13.0	2.6	1,896.75	∼586K
Large	60.0	13.0	5,733.93	∼518K

I. Performance Optimizations

Major performance optimizations include:

- **Selective Processing:** Focuses only on potential boundary positions (typically under 5% of characters)
- **ONNX Runtime Integration:** Enables optimized execution with 1.1x-2.1x faster inference
- **Parallel Processing:** Automatically partitions large documents for parallel processing
- **Hierarchical Caching:** Multi-level caching system for frequent patterns and decisions

J. Performance Characteristics

- **Throughput:** 377,000-600,000 characters/second depending on model size
- **Memory Efficiency:** Scales with chunk size rather than total document length
- **Algorithmic Complexity:** $O(t \times f)$ where t is the number of potential boundary positions and f is the number of features
- **Latency:** Optimized for documents of all sizes with parallel processing for large texts

K. Interactive Demonstration

To facilitate adoption and further research, we provide an interactive web demonstration of our sentence boundary detection tools at https://sentences.aleainstitute.ai/. This demonstration allows users to:

- Compare multiple sentence boundary detection algorithms (NLTK, spaCy, pySBD, NUPunkt, and CharBoundary) side-by-side
- Adjust the probability threshold for the CharBoundary model to explore precision-recall tradeoffs
- Test algorithms on custom legal text or pre-loaded examples
- Generate shareable links to specific analyses for collaboration

The interactive nature of this tool provides both researchers and practitioners with a practical way to evaluate the performance of different sentence boundary detection approaches on their specific legal text corpus without requiring local installation. Additionally, all source code and data needed to replicate the experiments and results in this paper are available at https://github.com/alea-institute/legal-sentence-paper.

L. Additional Figures

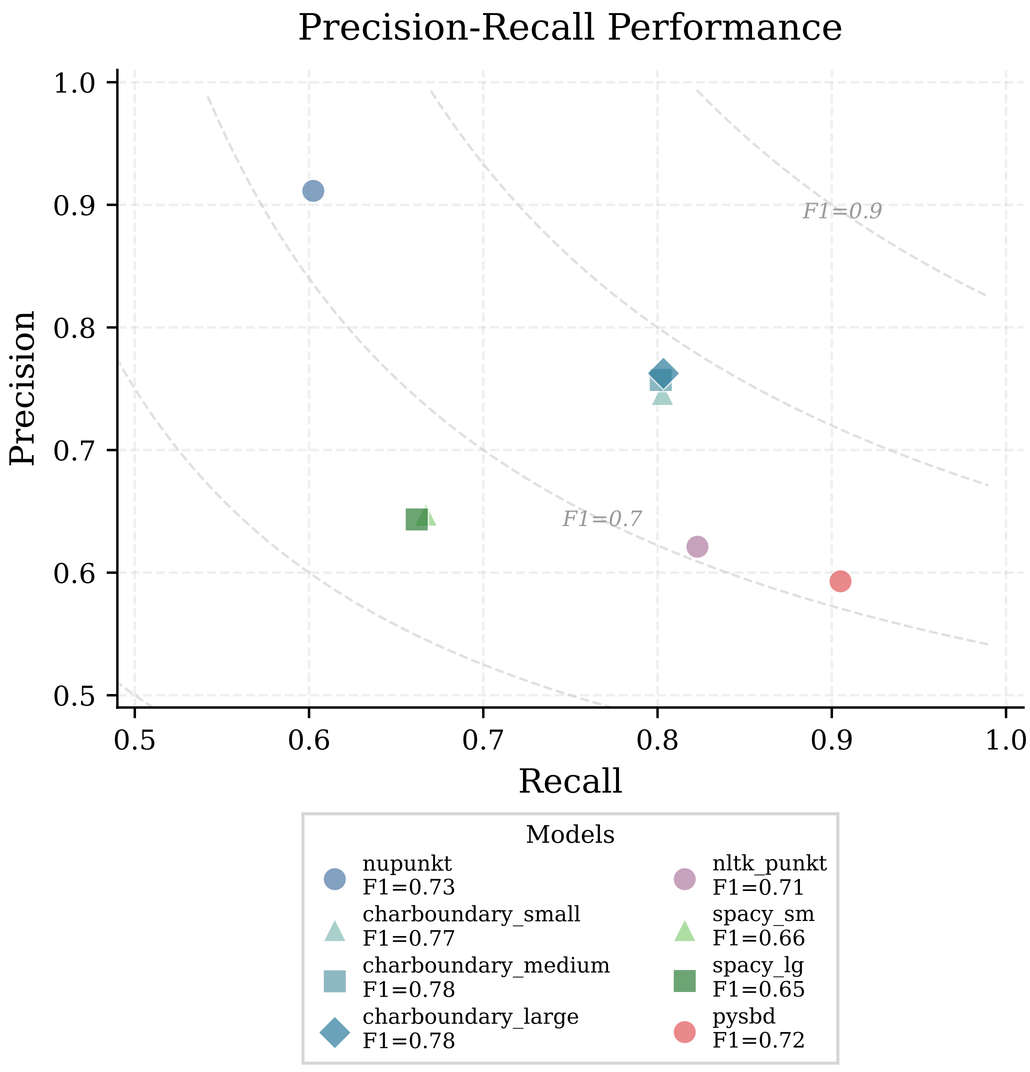

Fig. 2: Precision vs. recall comparison across models.

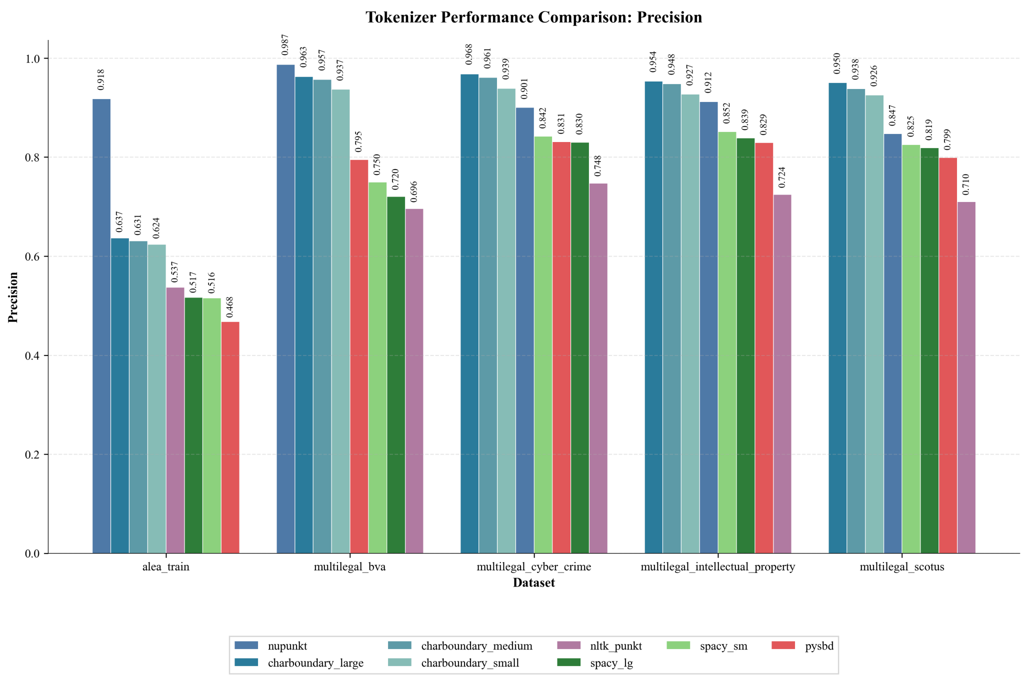

Fig. 3: Precision comparison across models and datasets.

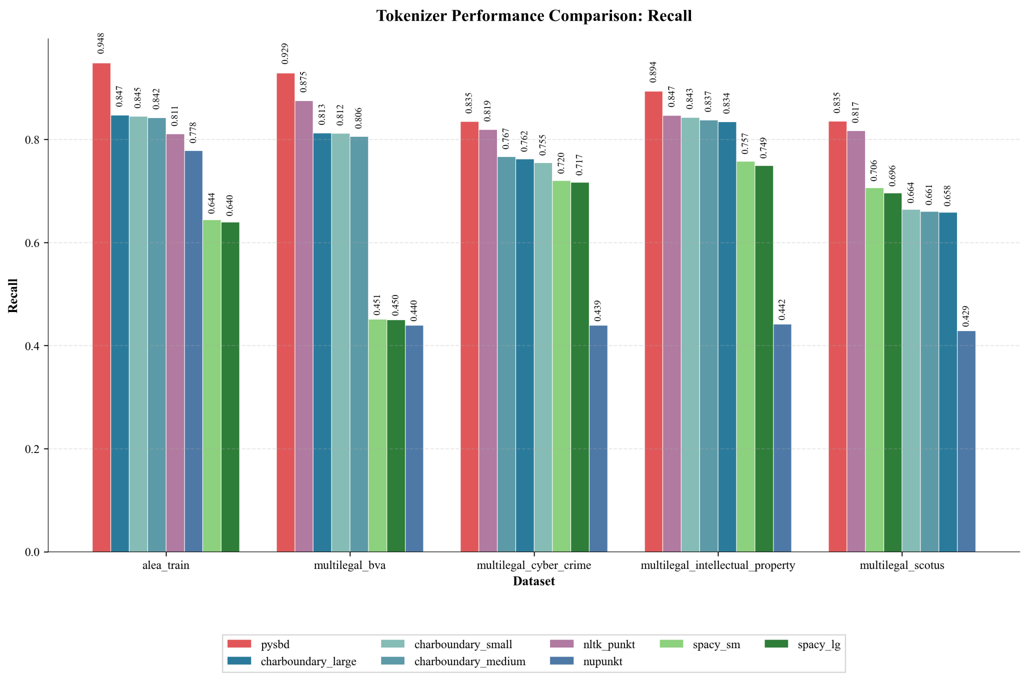

Fig. 4: Recall comparison across models and datasets.

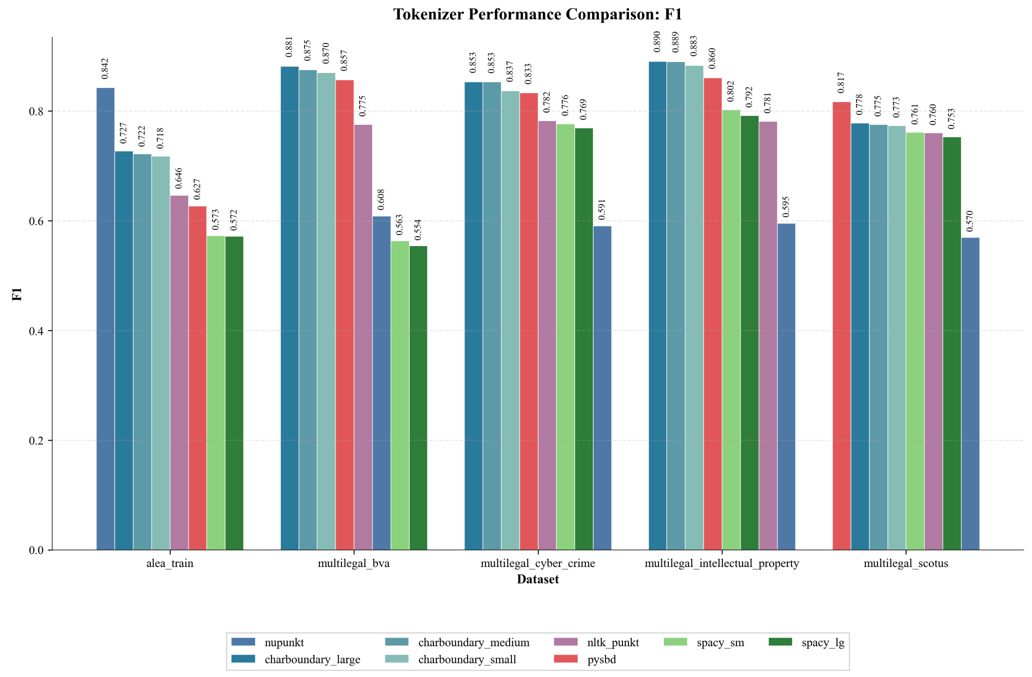

Fig. 5: F1 score comparison across models and datasets.