LegalText: Next Word Prediction For Lawyers

Sneha Joshi
Marymount University | Arlington, VA 22207
snehajoshi1sj@gmail.com

Abstract

Legal document drafting is complex, time-consuming and prone to errors which could benefit from natural language processing (NLP) techniques. This study focuses on the application of next-word prediction models to enhance the efficiency and accuracy of legal writing. The approach utilizes Long Short-Term Memory (LSTM) deep learning networks, to capture the multifarious language patterns in legal texts trained from corpus of public court opinions of Virginia and other dispositive writings gotten from courtlistner dataset, totaling over one million words.

1 Introduction: Legal Literature

Law is a profession of legal language and writing. Lawyers have no choice but to write and they must write in a certain style for their writing to qualify. Legal writing often involves common parlance and approved terminology, with a limited lexicon further constrained by syntax and form. Studies suggest, "The process of legal writing is 85 percent analysis and 15 percent composition, yet law schools often treat legal writing as mere expository writing rather than as essential legal writing training." [1]

Next word prediction means predicting the most likely word or phrase that will come next in a sentence or text. What if there was a inbuilt feature on an lawyer application that suggests the next word as you type the legal document .

2 Literature Review

Legal text processing poses unique challenges due to its specialized vocabulary, rigid syntax, and the high level of precision required. While models like GPT and BERT have achieved remarkable success in general language modeling, their application within the legal domain remains underexplored and underutilized. Prominent datasets, such as CourtListener and the Harvard Caselaw Access Project, provide a foundation for studying legal texts. There is limited research on real-time text generation tools specifically designed for legal professionals. This study addresses this gap by utilizing character-level LSTMs to predict text in the context of legal drafting. The use of court decisions from the author's hometown Virginia ensures both relevance and comprehensibility in evaluating the model's performance.

3 Methodology

3.1 Sample

The dataset used for this study consists of court decisions from Virginia, manually downloaded from CourtListener. A total of 25 recent documented court decisions in PDF format were collected, ensuring relevance and diversity. These legal texts form the corpus for training and evaluation.

3.2 Data Preprocessing

The raw legal documents were preprocessed into a format suitable for character-level text generation. The steps include:

- Loading PDFs: Text from each PDF was extracted using the `PyPDF2` library. The text from all files was concatenated into a single corpus.
- Data Cleaning: Non-text elements (e.g., headers, footers) were removed, and the text was standardized by removing extra spaces and irrelevant characters.
- Tokenization: A character-level vocabulary was built. Each character was assigned a unique integer index, and the corpus was transformed into sequences of integers.

[1] Mary Ellen Gale, *Legal Writing: The Impossible Takes a Little Longer*, 44 ALB. L. REV. 298 (1980).

3.3 Model Design

The text generation model is a character-level Recurrent Neural Network (RNN) implemented using TensorFlow. Its architecture includes:

- Embedding Layer: Transforms character indices into dense 256-dimensional vectors.
- LSTM Layer: A single LSTM layer with 1024 units captures long-term dependencies in text.
- Dense Layer: Outputs probabilities for the next character in the sequence.

The model was compiled using the Adam optimizer and Sparse Categorical Crossentropy as the loss function.

3.4 Training Process

The preprocessed corpus was divided into overlapping sequences of 100 characters, batched, and shuffled for training. The training configuration was as follows:

- Sequence Length: 100 characters.
- Batch Size: 64 samples per batch.
- Epochs: 50 iterations over the dataset.

To optimize runtime, the model was first tested on a smaller dataset of 2 PDFs, followed by training on the full 25-PDF corpus. This incremental approach enabled rapid experimentation while accommodating computational constraints.

3.4.1 Training Time Calculations

The time for training is calculated based on the number of data points (N), batch size (B), and time per batch (T_b). The formulas used are:

$$TotalBatchesperEpoch = \frac{N}{B}$$

$$Time = TotalBatchesperEpoch \times T_b$$

For the full dataset of 25 PDFs:

$$N = 12,500, \quad B = 64, \quad T_b = 2\,seconds$$

$$TotalBatches = \frac{12,500}{64} \approx 195$$

$$TimePerEpoch = 195 \times 2 = 390\,seconds$$

$$TotalTime = 50 \times 390 = 19,500\,seconds$$

With limitation in computing hardware testing with 2 PDFs minimized initial training time, enabling faster debugging and validation. Scaling to 25 PDFs improved model generalization but required significantly more computation time.

3.4.2 Checkpointing

Model weights were saved after each epoch to enable recovery and performance comparisons across epochs. Early stopping was considered to prevent overfitting.

3.5 Text Generation

After training, the model was used to generate legal-style text. A seed phrase (e.g., `WHEREAS,`) was provided, and the model predicted the next character iteratively. The process included:

- Adjusting the **temperature** parameter to control creativity.
- Generating text samples of 1000 characters for evaluation.

The outputs were evaluated using automated metrics (e.g., BLEU score, perplexity) and manual review by legal professionals for coherence and legal accuracy.

4 Conclusion

Looking forward, this research opens several avenues for further exploration. Integrating more advanced models, such as transformer-based architectures like GPT or BERT, could enhance the quality and fluency of generated text while addressing limitations in handling longer contexts. Expanding the dataset to include legal texts from multiple jurisdictions and legal systems could increase the model's generalizability and utility. Additionally, incorporating user feedback loops into the model could create interactive, real-time drafting tools tailored to the specific needs of legal practitioners.

References

2024. Courtlistener. https://www.courtlistener.com/. Data Accessed: 2024-11-08.

Mary Ellen Gale. 1980. Legal writing: The impossible takes a little longer. *Albany Law Review*, 44:298. (Bluebook format).

Aman Kharwal. 2024. Text generation model using python.