Paper notes

- Only features available during trail
- help make predictions on the fly
- Uses information available right when the case starts, such as:
    - Who the attorney is
    - What type of case (tort vs. vehicle accident)
    - Where it’s being tried
    - The text of the complaint document itself

The researchers used Connecticut state civil court data (2004–2019):

- Over 900,000 total entries
- Focused on 7,904 motions to strike (the thing they’re predicting)
- Case types: tort (injury) and vehicular (car accident) cases

A. Court Administrative Features

Feature	Meaning
Juris number	Unique ID for the lawyer or law firm
Attorney specialization	How specialized a lawyer is, measured by **entropy** (low = very focused, high = works many case types)
Major code	Case type (tort or vehicular)
Case location	Which Connecticut court the case is in

They built formulas (using probability & Dirichlet smoothing) to calculate attorney specialization entropy, which basically measures whether a lawyer handles a narrow range of cases or a broad range.

B. Complaint Document Features

They used three main methods:

1. **Rule-based keywords** –

   Look for words like "car," "accident," "negligence" and learn if they tend to appear in granted or denied cases.

2. **Word embeddings (word2vec, doc2vec, law2vec)** –

   These are mathematical representations of text that capture meaning.

   - word2vec/doc2vec are general language models.
   - law2vec is trained on legal documents.

     They turned each complaint into a 200–300-dimensional numerical vector.

3. **TF-IDF weighting** –

   Weights important words higher if they appear often in one document but not across all documents.

six different classifiers:

1. **Decision Tree** – flowchart-style "if/then" rules
2. **Random Forest** – many trees averaged together
3. **AdaBoost** – builds multiple weak trees, focuses on errors
4. **Gradient Boosting** – improves on AdaBoost using gradients
5. **XGBoost** – optimized version of Gradient Boosting
6. **Support Vector Machine (SVM)** – finds the best line (or plane) to separate granted vs denied motions

AdaBoost. gave the best results using When using word2vec embeddings, TF-IDF weighting, and rule-based (FOIL) features,

grid search to find the best hyperparameters (the tuning knobs for each model).

Baseline

If you just guess using the majority class ("granted" ≈ 52%), accuracy = 0.50.

Using only administrative data

Accuracy around 0.55–0.58 (a bit better than guessing).

Adding complaint text (word embeddings)

Accuracy improved to around 0.60–0.64.

Best model: AdaBoost using word2vec + TF-IDF + rule-based features → 64.4% accuracy.

So the AI could correctly predict the motion outcome about 64 times out of 100, using only data available during the case.

Questions:

- Since the model in the paper is designed for real-time use, meaning it only uses features that a lawyer would have access to during the trial rather than information available afterward (like judicial opinions or final verdicts), would it make sense to I guess "enhance" the modle doing a secondary analysis? Specifically, could we first train and evaluate the model using only "real-time" features, and then run a post-trial version that includes features only available afterward to refine or compare predictions? This way, we could see whether any of the real-time features correlate with or help predict patterns found in the post-trial data.