Appendix D: Preprocessing and Segmentation Algorithms

D.1 Overview

CourtShadow's preprocessing pipeline transforms raw court transcripts into a standardized collection of short, linguistically meaningful segments. This is crucial since raw transcripts vary greatly in terms of structure, formatting, quality (i.e., Optical Character Recognition (OCR)), and completeness. This appendix provides pseudocode and algorithmic detail for each step:

1. Transcript normalization,
2. Speaker-tag identification,
3. Segmentation into linguistic units,
4. Metadata assignment,
5. Feature extraction routines.

All steps use deterministic rule-based processing for transparency and reproducibility.

D.2 Transcript Normalization Algorithm

The normalization stage removes formatting noise and aligns the text into a consistent structure.

Algorithm D.2: NormalizeTranscript

Input: raw transcript text T
Output: cleaned text T'

1. Remove page headers/footers using regex for patterns like `Page X`, `---`, `COURT REPORTER`.
2. Replace multiple spaces with single spaces.
3. Collapse multiple newlines into a single newline.
4. Remove hard linebreaks mid-sentence (e.g., OCR artifacts).
5. Trim leading/trailing whitespace on all lines.
6. Return cleaned transcript T'.

Examples of artifacts removed include: reporter timestamps, case numbers repeated at page tops, and stray indentation.

D.3 Speaker Identification Algorithm

Speaker turns are crucial for segmentation. A speaker label is defined as any line matching:

$$^{\lfloor}A-Z][A-Z\dot{]}+:{}^{\lfloor}A-Z][A-Z\dot{]}+:{}^{\lfloor A-Z][A-Z\dot{]}+:{}^{\lfloor A-Z][A-Z\dot{]}+:}}$$

(e.g., `THE COURT:`, `MR. SMITH:`, `DEFENDANT:`).

Algorithm D.3: IdentifySpeakers

Input: cleaned transcript T'
Output: list of (speaker, utterance) pairs

Initialize empty list U.
For each line L in T':
    If L matches speaker tag regex:
        Set speaker = extracted speaker label.
        Set currentUtterance = empty string.
    Else:
        Append L to currentUtterance.
When encountering the next speaker tag, push (speaker, currentUtterance) to U.
Return U.

This recovers a turn-by-turn structure consistent across transcripts.

D.4 Segmentation Algorithm

Segments are short units preserving local linguistic structure while enabling robust feature extraction.

Target length: **6–70 tokens**. Shorter turns are merged; longer turns are split at punctuation.

Algorithm D.4: SegmentUtterance

Input: utterance text U
Output: list of segments S

1. Split U at `.`, `?`, `!`.
2. For each raw chunk c:
    Trim whitespace; compute token count t.
    If $t < 6$: merge with the next chunk if possible.
    If $t > 70$: recursively split at midpoints or commas.
3. Discard empty segments.
4. Return final list of segments S.

This procedure ensures consistent granularity across transcripts regardless of original formatting.

D.5 Metadata Assignment

Each segment is associated with:

- Case ID
- Speaker role (judge, prosecutor, defense, witness, defendant, unknown)
- Line index
- Inherited group label

Speaker roles are assigned using a dictionary of cues (e.g., `THE COURT` → judge; `MR./MS.` → attorney unless cross-examination context indicates witness).

D.6 Feature Extraction Algorithms

This section details the exact computational rules for CourtShadow's 38 features.

Structure

Algorithm D.6.1: StructureFeatures

Input: segment s
Output: {chunk_tokens, chunk_sentences, chunk_question_marks}
1. t = tokenize(s); count tokens.
2. Count occurrences of `.`, `?`, `!`.
3. Count occurrences of `?`.

Discursive Framing

Framing relies on dictionary-based counts and normalized rates.

Algorithm D.6.2: FramingFeatures

Input: segment s, token list t
Output: 10 framing features
1. For each lexicon category (politeness, harshness, mitigation, strong certainty, weak certainty, positive/negative evaluation):
    Count occurrences of lexicon items in t.
2. Compute rates = counts / $|t|$.
3. Compute `sentiment_imbalance` = positive_count − negative_count.

Pronouns & Voice

Simple dictionary lookups:

$$\text{first-person} = |\{I, me, my, we, us, our\}|, \qquad \text{second-person} = |\{you, your, yours\}|.$$

Rates are computed by dividing by token count.

Topic Indicators

21 binary flags — if any keyword in the category appears, the feature fires.

Algorithm D.6.3: TopicIndicators

Input: segment token list t
Output: 21 binary topic features
For each topic category C:
    If any keyword in C appears in t, set $C = 1$; else set $C = 0$.

D.7 Standardization

Continuous features are standardized using training-set statistics:

$$x' = \frac{x - \mu}{\sigma}.$$

Binary topic indicators are left unchanged.

D.8 Final Feature Vector Assembly

Each segment's final feature vector is:

$$x = \big[\text{structure}, \text{framing}, \text{pronouns}, \text{topics}\big] \in R^{38}.$$

This vector is then fed into the logistic regression model described in Appendix B.

D.9 Quality Checks

Prior to modeling, the following types of segements are either merged or removed following deterministic rules. This prevents downstream noise. These include segments with:

- Fewer than 4 tokens,
- Non-ASCII artifacts,
- Corrupted speaker tags