RankCut: A Ranking-Based LLM Approach to Extractive Summarization for Transcript-Based Video Editing

Sana Shah
Department of Informatics
University of Hamburg
Hamburg, Hamburg, Germany
Adobe Research
Adobe
Hamburg, Hamburg, Germany
sanashah1302@gmail.com

Mackenzie Leake
Adobe Research
San Francisco, California, USA
leake@adobe.com

Kun Chu
Informatics
Knowledge Technology
Hamburg, Hamburg, Germany
kun.chu@uni-hamburg.de

Cornelius Weber
Department of Informatics
University of Hamburg
Hamburg, Germany
cornelius.weber@uni-hamburg.de

Nico Becherer
Adobe
Hamburg, Hamburg, Germany
nbechere@adobe.com

Stefan Wermter
Department of Informatics
University of Hamburg
Hamburg, Hamburg, Germany
stefan.wermter@uni-hamburg.de

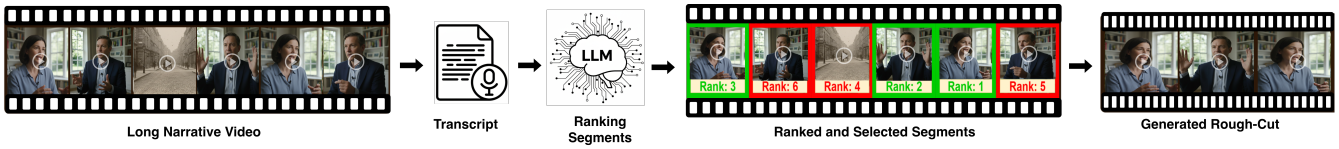

Figure 1: Overview of the RankCut pipeline. Given an existing transcript of a long narrative video, a large language model (LLM) ranks candidate segments under a target duration. A duration-aware selector then chooses segments and assembles them in source-chronological order to produce a concise, time-aligned rough cut.

Abstract

Video recordings of interviews, lectures, and meetings contain valuable moments surrounded by less essential talk. Making a shareable and meaningful shorter version of this content requires significant effort because it combines tedious, repeated operations with personal editorial decisions, which require human judgment. We introduce an editing approach that operates on video transcripts and combines a three-stage large language model pipeline with a timeline-anchored, marker-based interface so editors can inspect and refine suggestions before final assembly. The pipeline first produces an overview summary to maximize content coverage, then induces plain-language selection rules that encode editorial intent, and finally applies rule-conditioned ranking on small transcript windows to mitigate long-context limits, yielding strictly extractive, time-aligned spans under duration constraints. The interface displays groupings of short excerpts using markers with priorities and confidence cues, converting opaque model output into verifiable units within standard video editing workflows. On MeetingBank and MeetingBank-QA datasets, our method outperforms practical extractive baselines at matched lengths. In a within-subjects study with experienced video editors familiar with Premiere Pro video editing software, we found that our marker-based interface provided editors higher efficiency, control, and satisfaction than both a manual editing baseline and an opaque auto-cut condition.

CCS Concepts

• **Human-centered computing** → **User interface design**; **Empirical studies in HCI**; • **Computing methodologies** → *Natural language processing*.

Keywords

Video editing, Text-based video editing, Video transcript, Human–AI collaboration

ACM Reference Format:
Sana Shah, Mackenzie Leake, Kun Chu, Cornelius Weber, Nico Becherer, and Stefan Wermter. 2026. RankCut: A Ranking-Based LLM Approach to Extractive Summarization for Transcript-Based Video Editing. In *31st International Conference on Intelligent User Interfaces (IUI '26), March 23–26, 2026, Paphos, Cyprus.* ACM, New York, NY, USA, 20 pages. https://doi.org/10.1145/3742413.3789115

IUI '26, Paphos, Cyprus

ACM ISBN 979-8-4007-1984-4/26/03
https://doi.org/10.1145/3742413.3789115

1 Introduction

Spoken-content videos, such as recordings of interviews, lectures, and meetings, are one of the most popular forms of media online. These videos can range from minutes to hours and span a wide

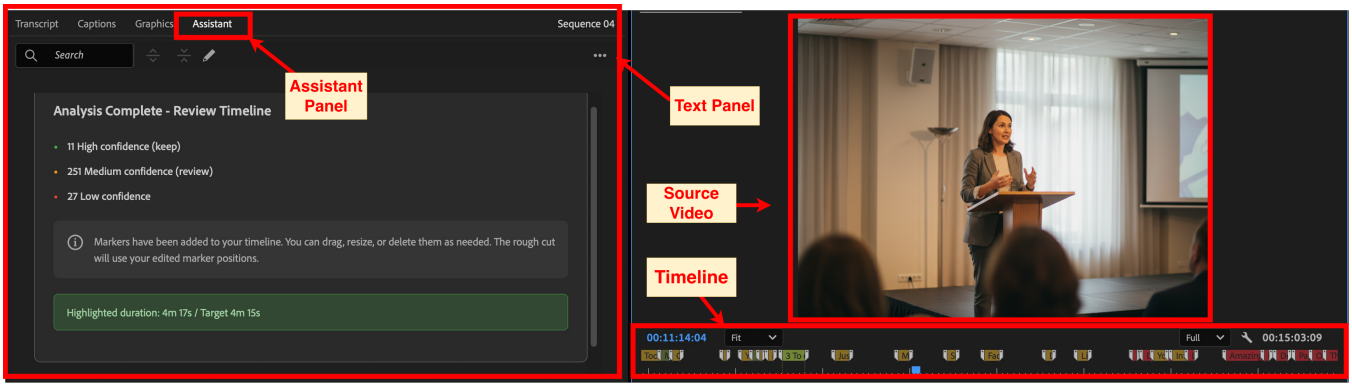

Figure 2: System overview showing how our research prototype could be surfaced within a video NLE. The Assistant panel (left) hosts the UXP extension. The Text panel provides the transcript used by the pipeline. The Source video (right) shows playback. The Timeline (bottom) displays color-coded markers aligned to sentence timecodes. (Adobe product screenshot(s) reprinted with permission from Adobe.)

range of topics. After recording, this type of content typically follows one of several paths before being shared on social media sites like YouTube: videos can simply be posted in their entirety, lightly edited to remove unnecessary setup time, or more heavily edited to produce shorter shareable highlights. These types of recordings contain valuable information, yet their length, and, in some cases, logical dependencies between parts of the content, make them difficult to navigate and edit efficiently.

In these domains, most communicative value lies in spoken dialogue rather than visual change, meaning transcript-driven methods can capture salient information without explicit visual analysis while still producing professionally useful edits. Professional editors often spend more time locating material than shaping it into deliverables [45], and real-world studies comparing manual vs. automated workflows report noticeable time savings from automation in educational content editing [34]. Webinars, lectures, and virtual meetings are some of the most common types of content, but they are also among the least favorite types of content for editors. They require disproportionately more time to sift through and index compared to narrative or visually engaging media [24].

Recent AI-assisted video editing tools, both in research and commercial products, have focused on social media and narrative content, producing different types of outputs, such as social media reels and narrative shorts [3, 7, 36, 50]. However, domains, such as webinars and meetings, differ from social media and narrative content in a few key ways. First, such content does not necessarily have as much of a narrative arc. Additionally, it can be information dense, with multiple interconnected topics discussed across a long time scale within the video, making high-quality highlight selection more nuanced. And finally, the ratio of input to output content can be quite large and highly variable [19].

Large language models (LLMs) present opportunities for assisting the editing of long speech-driven videos via transcript-based editing, but simple extractive summarization is insufficient for professional video editing workflows. In particular, extractive summaries often lack transparency, since editors cannot verify why specific segments were chosen; they may also disrupt temporal consistency and fail to preserve narrative flow. Moreover, they provide limited factual control, as models can still hallucinate or omit important contextual cues. To address these issues, our method prompts an LLM to explicitly select transcript sentences as visible, verifiable, and controllable units of editing, ensuring temporal coherence and explainable selection behavior.

We introduce a transcript-driven rough-cut workflow that makes model choices *visible, verifiable, and controllable* prior to timeline assembly. This workflow builds upon recent advances in LLM-assisted video editing [3, 25, 44]. We implement a user-in-the-loop, LLM-driven editing assistant within a widely used professional video editing tool (Adobe Premiere Pro [21]), combining transcript-based and traditional timeline-based editing to support the shortening of informational videos. Our work makes the following contributions:

(1) **Markerization of AI selections.** Building on prior work in timeline-based transparency [25], we convert model outputs into *timeline-anchored range markers* with short excerpts, confidence-normalized scores, and priority bands (Green/Yellow/Red), enabling editors to accept, trim, or reject spans *locally* rather than auditing a full rough cut.
(2) **Deterministic, budget-safe, source-ordered content selector.** Our assembly step uses a simple selector that is deterministic, budget-respecting, and source-ordered: with fixed inputs it always returns the same cut; the total length stays within the requested duration (up to a small rounding tolerance from sentence/clip granularity); and selected segments keep their original timeline order. These guarantees make results predictable, easy to audit, and quick to revise.
(3) **Quantitative and qualitative evaluation of summary quality in the context of video editing.** On MeetingBank/MeetingBank-QA, two common summarization datasets, we outperform length-matched extractive baselines; in a within-subjects study with experienced editors, the marker workflow raised perceived efficiency over manual and opaque auto-cut alternatives.

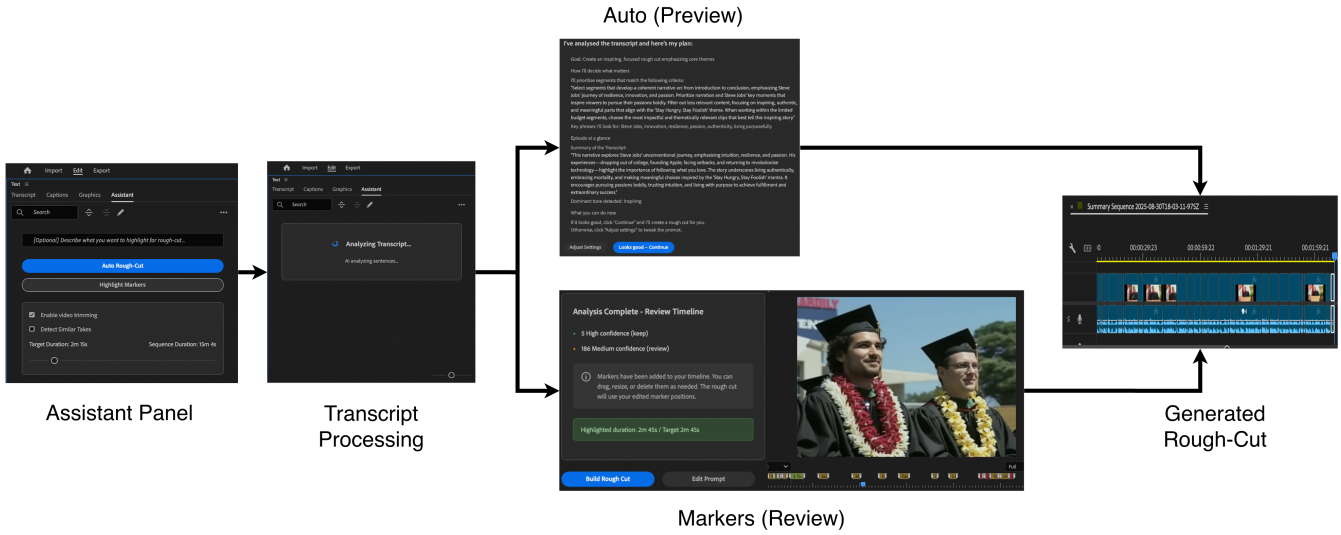

Figure 3: Two workflows for rough-cut generation. This illustrative example shows how our research prototype could be surfaced within a video NLE. *Auto* shows a prompt-driven *plan preview* and then compiles a cut without exposing span-level timeline markers before assembly. *Markers* first shows ranked, transcript-aligned spans as timeline markers (with priority and confidence) that editors can accept, reject, or trim, and then assembles the accepted set. Both workflows use the same deterministic, order-preserving, duration-bounded selector. (Adobe product screenshot(s) reprinted with permission from Adobe.)

2 Related Work

Our work builds upon prior research in computational tools for summarizing and editing video, as well as broader efforts to improve transparency and control in AI-assisted creative workflows. We situate our approach at the intersection of three key domains: text-based video editing interfaces that leverage transcripts for navigation, recent advances in LLM-driven content creation, and automated summarization techniques developed for long-form spoken dialogue.

2.1 Text-Based Video Editing

Text-based video editing relies on textual representations of content to make editing decisions. While most of these systems rely on text transcripts time-aligned with videos, in some cases, different forms of structured texts, such as markdown [10], HTML [11], or other documents have been used [9]. Early systems showed that aligning time-aligned transcripts with video segments could assist video search [37], rough cut creation [4, 46], and feedback [38]. While some approaches utilize narration to drive content selection for narrative shorts [46], physical demonstration videos [12], and video slideshows [27], other approaches have focused on social videos [2] and scripted scenes with multiple video takes [26]. Several text-based video editing interfaces have focused on accessibility considerations by assisting with audio descriptions [39] and using visual descriptions of videos to help blind and low vision users edit videos [20] and consume social media videos [47]. Our work focuses on a different type of input video: long, informational content, which poses distinct challenges in information selection and appropriate levels of time compression to achieve target output durations.

2.2 LLM-Based Video Editing

Recent work has explored the use of LLMs and VLMs for assisting parts of the video editing process. ChunkyEdit [25] used LLMs on video transcripts to group related clips and select video highlights, focusing on thematic chunking for interview footage. While ChunkyEdit organizes content into topic-coherent chunks, our work extends this by providing fine-grained, sentence-level extractive ranking with explicit confidence scoring and duration-bounded selection under user-specified constraints. LAVE used natural language to support several key video editing steps, such as providing a video overview, brainstorming, storyboarding [49]. EditDuet [44] proposes an agentic editing model, which uses natural language feedback generated by an LLM to iteratively improve a video rough cut through Editor and Critic agents. In contrast to EditDuet's iterative refinement of a full rough cut, our approach exposes intermediate ranked selections as timeline markers *before* assembly, enabling editors to inspect and modify individual segments rather than evaluating entire sequences. Commercial tools, such as OpusClip [36], CapCut [7], and WiseCut [52], use LLMs to create social media shorts. Some research, such as PodReels [50] has similarly focused on automating the end-to-end editing process for social media content. Most similar to our work in algorithmic approach, Lotus [3] helped users create short videos using LLM-based abstractive and extractive summarization. In contrast to these approaches, our work addresses long, information-dense, speech-driven videos (e.g., meetings, lectures, webinars) and focuses on timeline-faithful assembly optimized for professional editing where factual grounding, predictable duration control, and transparent intermediate outputs are needed. Our design differs by exposing edit decisions as timestamped, confidence-graded markers, where each marker directly

references exact transcript sentences and timecodes, allowing editors to inspect, trim, and reorder precise source material before committing to assembly.

2.3 Text-Based Summarization

Video summarization is a broad area of study that spans extracting highlights, providing representative segments of a video, and producing video trailers. While some approaches to video summarization rely on visual data [1] or viewership statistics [23], speech-driven content lends itself to transcript-based approaches, which can borrow from existing NLP techniques. Extractive (i.e., selecting verbatim segments from the input to include in the output) and abstractive (i.e., paraphrasing input segments when needed) are two distinct approaches to text summarization that offer different strengths. Classic extractive text summarization methods, such as LexRank and TextRank, established graph-based sentence scoring as a foundation for selecting salient segments [14, 32]. To overcome the strengths and weaknesses of each approach, some recent NLP systems have combined extractive and abstractive techniques and introduced human-in-the-loop steps. Recent work on interactive summarization systems has shown that giving users the ability to steer or constrain summaries can yield outputs that better match their goals [22] and assist skimming [28]. Abstractive methods generate paraphrased segments of transcript that may improve readability, but they break the one-to-one mapping between transcript text and video. However, segments from extractive summaries of speech-driven content provide indices into the input video clips. In our work, we use abstractive methods to reduce the search space of feasible clips and then extractive summarization to identify video clips for the output summary.

3 System Overview

Our system is implemented as a custom extension for Adobe Premiere Pro and appears as a custom *Assistant* panel in the software. It helps editors transform speech-driven source material into a reviewable rough cut by surfacing transcript-aligned candidate clips directly inside a familiar editing interface. Building upon the Text panel, a transcription and caption editing tool in the application, we run a lightweight three-stage algorithmic pipeline over the transcript to select candidate clips and return this set via markers (i.e., color-coded timeline aligned annotations) (Figure 2). Users can specify an optional text prompt and desired duration, and the result is a set of color-coded markers displayed on the timeline and an automatic rough cut (Figure 13). All selections are strictly extractive and preserve temporal order (Figure 2). The **Timeline** displays the output color-coded markers with normalized confidence scores and short excerpts. Editors can accept, trim, merge, or demote markers; the deterministic selector enforces the duration constraint while preserving source order.

3.1 Design Goals: Visible, Verifiable, Controllable

RankCut's interface design addresses a critical gap in AI-assisted video editing: the lack of interpretable intermediate outputs. Existing systems present end-to-end results without exposing selection criteria or enabling granular refinement. Our three design goals operationalize principles from explainable AI and human-AI collaboration research: **(1) Visible**—timeline markers expose selected spans with priorities and confidence scores; **(2) Verifiable**—transcript-to-timeline mapping enables efficient validation; **(3) Controllable**—local accept/reject/modify operations preserve editor agency. This approach supports calibrated trust through inspectable, reversible decisions [6, 13].

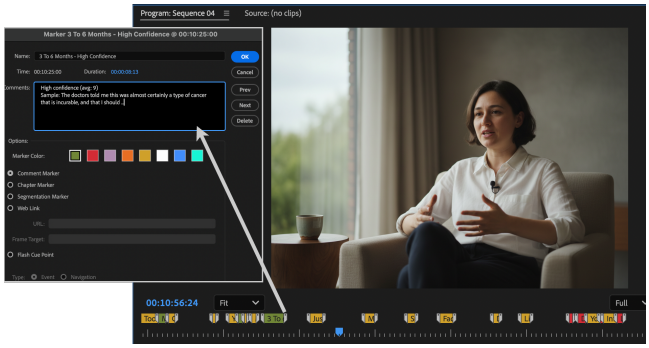

Figure 4: Visualization of timeline markers. This illustrative example shows how our research prototype could be surfaced within a video NLE. Each selection is shown as a color-coded span with priority label, normalized confidence score, and a transcript excerpt. Labels (GREEN, YELLOW, RED) are included to indicate whether the sentences are good to include, need human review, or should be excluded. These markers make the AI's decisions visible and editable: editors can inspect, trim, or relabel spans directly on the timeline before building the rough cut. (Adobe product screenshot(s) reprinted with permission from Adobe.)

3.2 Assembling a Rough Cut

Editors reach assembly via two entry points (Fig. 3). We use the term *span* to mean a contiguous, transcript-aligned time range $(t_{\text{start}}, t_{\text{end}})$ in the source video. *Auto* emphasizes speed by compiling a cut directly from a prompt-driven plan *without span-level controls*, whereas the *Markers* option exposes the model's candidate spans on the timeline for *span-level review and adjustment* before assembly.

When editors choose *Auto Rough Cut* in the Assistant Panel, the system analyzes the transcript and displays a text-based *AI Preview* of the plan, summarizing the inferred selection criteria and overall intent (Fig. 3, top path). Editors can re-run with a revised prompt or accept the plan. Span-level choices are not shown prior to assembly; continuing compiles a playable summary sequence directly from the model's selections.

When editors choose *Highlight Markers*, the system highlights candidate spans directly on the editing timeline (Fig. 3, bottom path). Each marker references a specific transcript span and displays a priority label (Green, Yellow, or Red), a normalized confidence score, and a short excerpt in the marker's comment field. **Green** indicates a high-confidence inclusion recommended by the AI, **Yellow** indicates medium confidence and invites editor review, and **Red** indicates low priority. Editors can accept or reject markers, trim boundaries, and merge or split adjacent spans that represent

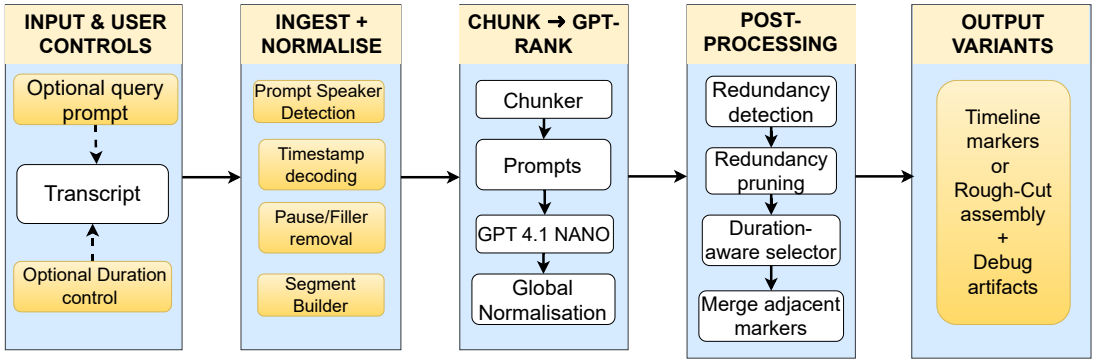

Figure 5: Runtime pipeline for RankCut. End-to-end process from transcript input to ranked, duration-bounded outputs. Steps in yellow occur within the host video editor (Premiere Pro plug-in); white steps are model-agnostic algorithmic components shared with offline evaluation. The five stages are: (1) *Input & User Controls* – optional query prompt and duration target with transcript input; (2) *Ingest & Normalize* – speaker detection, timestamp decoding, pause/filler removal, and segment building; (3) *Chunk → GPT-Rank* – windowing, prompting, ranking via GPT-4.1 Nano, and global normalization; (4) *Post-Processing* – redundancy detection and pruning, duration-aware selection, and marker merging; (5) *Output Variants* – timeline markers or auto-assembled rough-cut sequence.

distinct ideas. After review, the accepted set, now editor-curated, *specifies the clips that are assembled into the final rough cut.* This workflow provides visibility into the model's intermediate decisions and supports fine-grained editorial adjustments before assembly.

4 Algorithms

Our tool relies on an editing algorithm that ranks sentences in the transcript for inclusion in the output edit.

4.1 Transcript Processing

Figure 5 illustrates the processing stages conceptually. Underneath, our algorithms operate on individual transcript sentences parsed from Adobe Premiere Pro, each represented with start/end timecodes and, when available, a speaker label. All selections are strictly extractive and preserve the original timecodes, at a per-frame level. The transcript is prepared for ranking through four steps.

(1) Prompt-driven speaker filtering. Given transcript segments $X = \{(s_t, \ell_t)\}_{t=1}^{N}$ with diarization labels $\mathcal{S}$ and an optional user prompt q, we compute $\mathcal{S}_q = \{\, s \in \mathcal{S} : \texttt{lower}(s) \subseteq \texttt{lower}(q) \,\}$. If $\mathcal{S}_q$ is not empty, we restrict scoring and ranking to segments where $\ell_t \in \mathcal{S}_q$. If no speaker match is found or the user does not provide a speaker label, all segments are retained.

(2) Timestamp decoding. We read the transcript's per-segment time spans and keep them unchanged through all intermediate stages. This provides a duration for each segment, which is useful for checking whether the target output duration is achieved.

(3) Segment building with unsupervised topic boundaries. We detect boundaries between consecutive utterances in a sentence-aligned transcript, then form multi-sentence segments between successive boundaries (preserving original order and per-sentence timecodes). Each utterance u_t is embedded with `all-MiniLM-L6-v2` (384-d) from `sentence-transformers` and L2-normalized; proximity is cosine similarity [43, 51]. For each gap i we compute four cues and later fuse them: (i) **semantic change** $S(i,K) = 1 - \cos\big(\text{mean}(u_{i-K+1..i}),\ \text{mean}(u_{i+1..i+K})\big)$ for $K \in \{2,3,4\}$; per meeting, we z-score each $S(\cdot,K)$, smooth with a moving average (window = 3), then average over K to obtain $\text{Sz}(i)$; (ii) **Foote-style novelty** via a checkerboard kernel on the utterance–utterance cosine matrix with widths $\{8,12,16,20\}$, z-scored per width and averaged [17]; (iii) a binary **speaker-change** prior; and (iv) a **pause-density** prior given by the z-scored count of pause markers near the gap. We fuse

$$\text{Score}(i) = \alpha\,\text{Sz}(i) + \beta\,\text{Novelty}(i) + \gamma\,\text{Speaker}(i) + \delta\,\text{Pause}(i),$$

z-score the fused series per meeting, detect local maxima, apply non-maximum suppression with window w_{nms}=2 [33], and retain peaks above percentile $\theta \in \{70,80,90\}$ in a non-calibrated regime (unknown boundary count). Unless noted otherwise we use *Balanced Sz+Novelty*: $(\alpha,\beta,\gamma,\delta)$=$(0.5,0.5,0,0)$ with θ=80; a stricter setting adds discourse priors *(Best NONCAL, subset)*: $(0.4,0.3,0.1,0.1)$ with kernels $\{8,12,16\}$ and θ=90. Full implementation details and evaluation protocol appear in App. D and §D.2, with configurations summarized in Table 6.

4.2 RankCut Topic-Aware Extractive Pipeline

Long, multi-speaker transcripts create two common failure modes for LLMs: very large input context windows encourage truncation and position bias [29, 48], while very small windows provide narrow context. To balance these context-related considerations, our approach uses a three-stage pipeline. All three stages use a compact LLM (GPT-4.1 nano [35]). For concrete inputs and outputs of all three stages, see the worked example in §4.3.

Design rationale for three-stage architecture. We designed RankCut's three-stage pipeline to address fundamental challenges in applying LLMs to transcript-based video editing. A naive single-stage approach faces critical limitations: positional bias (models favor content near prompt boundaries while neglecting middle

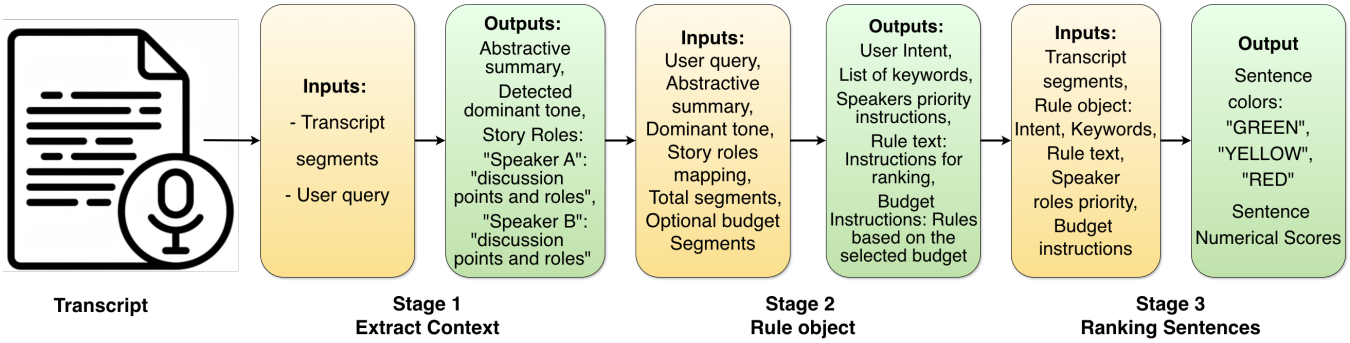

Figure 6: Three-prompt architecture. Stage 1 builds high-recall coverage summaries; Stage 2 induces a reusable `RULE_OBJECT` that formalizes and stabilizes the model's selection criteria; Stage 3 performs windowed extractive ranking with percentile normalization, producing timestamp-faithful spans for markers and the planning view.

sections [29, 30, 48]), structured output degradation with long inputs [53], and lack of transparency in selection criteria. Our ablations (App. C.1, C.2) empirically validate that very large windows (≥400 sentences) induce early-position preference and degrade accuracy, while very small windows inflate local winners with poor precision. The three-stage design mitigates these issues: Stage 1 exploits small-window recall for broad coverage; Stage 2 compiles explicit, inspectable rules to stabilize criteria; Stage 3 applies extractive ranking on intermediate windows with global rule conditioning, balancing scalability, transparency, and controllability.

Hyperparameters. Each stage uses a temperature setting aligned with its role: Stage 1 (T=0.7) encourages diverse generation to maximize recall of important topics during abstractive summarization; Stage 2 (T=0.6) balances flexibility with structure to yield stable JSON-based RuleObjects; Stage 3 (T=0.2) ensures deterministic extractive scoring, supporting reproducibility and comparability across chunks. These values were selected based on established prompt engineering practices [8] and held fixed after initial tuning. Window sizes (App. C.1) and post-processing thresholds were similarly fixed after development-phase tuning to ensure cross-run consistency.

4.2.1 Phase 1: Abstractive Coverage. This stage exploits the strength of LLMs at *abstractive compression* to encourage high coverage of salient content prior to extractive selection [3, 30, 42, 53]. Given a transcript X, our algorithm produces compact, human-readable artifacts that enumerate entities, decisions, actions, and topical anchors to be preserved by later stages (compact example in Listing 1).

Inputs are processed as a *single* sentence-aligned chunk whenever the transcript fits within the model's effective context budget. For longer inputs, the transcript is partitioned into smaller chunks $\{C_k\}$ solely to avoid exceeding the model's context limit. Each chunk satisfies

$$\text{len}(C_k) \leq B_{\text{ctx}} - g,$$

where B_{ctx} is the effective model context capacity (tokens) and g is reserved for prompts and structural headers.

Each chunk C_k is summarized with temperature = 0.8 to encourage diverse generation and maximize recall. The model returns a JSON with fields `summary`, `dominant tone`, and `speaker_roles`. The labeled chunks are merged into a single *fact pool* used in rule induction (Phase 2). Stage 1 is designed to mitigate two context-related failure modes in long, multi-speaker transcripts: loss of salient items when windows are too small, and positional/truncation effects when windows are too large. For more details about our parameter selections, please see App. C.1.

4.2.2 Phase 2: Rule Creation. In the second stage, the Stage 1 outputs (`summary`, `dominant_tone`, `speaker_roles`) together with any editor-provided focus (query prompt, speaker preferences, target duration) are compiled into a transcript-specific `RULE_OBJECT` (example schema/text in Listing 2). This object allows for uniform ranking criteria across transcript segments.

The abstractive `summary` is condensed into a single global rule with *intent* and a compact set of salient terms (entities, numbers, names) that help guide selection. This is combined with the user inputs of topics, speakers, and duration to also help guide selection. Stage 2 is generated with temperature = 0.4 to balance flexibility with structure, yielding stable JSON-based RuleObjects while preserving lexical expressiveness for salient criteria. The resulting `RULE_OBJECT` is emitted as strict JSON and retained with the job so it can be inspected or edited before Stage 3.

4.2.3 Phase 3: Extractive Summarization. In the third stage, our algorithm applies extractive ranking on *short* transcript windows (≈50 sentences) conditioned on the `RULE_OBJECT`. The `RULE_OBJECT` provides global context and intent to every window, stabilizes criteria across segments, and carries editor priorities, so small windows can be used without losing the bigger picture. Using smaller windows keeps each prompt within context limits and reduces truncation and positional bias observed with larger windows (App. C.1). Local ordinal ranks are mapped to percentile scores to obtain window-internal scores, which we normalize to $s \in [0, 1]$ for comparability across windows (App. C.3).

Highly similar content is often similarly ranked in our algorithm. We remove redundant content, which can occur when a speaker repeats themselves in the input. Each sentence u_t is embedded with Sentence-BERT (all-MiniLM-L6-v2, 384-d). For a candidate span A that covers sentences $[t_a, t_b]$, we define its embedding as the mean

of constituent sentence embeddings:

$$\mathbf{e}(A) \;=\; \frac{1}{(t_b - t_a + 1)} \sum_{t=t_a}^{t_b} \mathbf{e}(u_t), \qquad \|\mathbf{e}(\cdot)\|_2 = 1.$$

Span similarity is cosine similarity $\cos(\mathbf{e}(A), \mathbf{e}(B))$. We apply non-maximum suppression over the set of candidate spans using (priority, score) as primary keys and cosine similarity as the overlap test. A span B is suppressed if there exists A selected earlier such that

$$\cos(\mathbf{e}(A), \mathbf{e}(B)) \geq \tau \quad \text{and} \quad \Delta t(A, B) \leq \Delta t_{\max},$$

with defaults τ=0.85 and $\Delta t_{\max}$=30 s (to avoid conflating semantically similar but far-apart mentions).

After pruning, we greedily merge neighbors if they are (a) contiguous or separated by a gap $\leq g_{\max}$ seconds (default $g_{\max}$=2 s), (b) share the same speaker block or topic segment (if available), and (c) do not cross editor-locked boundaries. Merged spans are trimmed to sentence limits; any zero-length remnants created by trimming are dropped.

Within each window of N sentences, the model returns an ordered list. We convert the local rank $r \in \{1, \ldots, N\}$ to a scale-free score $\tilde{s} = 1 - \frac{r-1}{N-1} \in [0, 1]$, making scores comparable across windows of different sizes. Further implementation choices and ablations for this normalization are provided in App. C.3.

Priority labels $p_t \in \{\textsc{Green}, \textsc{Yellow}, \textsc{Red}\}$ are *emitted by the model* in Stage 3 under the same `RULE_OBJECT`. These categorical labels provide marker colors in the UI and define the selector's lexicographic ordering (partition by p, then sort stably by $-s$ and start time). Given (p_t, s_t) and time spans, the deterministic duration-constrained policy assembles a rough cut under a target T with a small overshoot tolerance, while preserving source order and timestamp fidelity. Formal guarantees and pseudocode appear in App. E.

4.3 Worked Example: Algorithm Walkthrough (Podcast)

To make the three-stage pipeline concrete, we show a short two-speaker podcast covering wireless headphones, performance numbers, an optimization tip, and regulations. Figures 1–3 show the machine-readable artifacts that drive selection.

Listing 1: Stage 1 (High-recall coverage).

```
{
  "summary": "The podcast introduces new wireless
     headphones, shares battery life results under
     different modes, highlights an optimization tip that
     extends usage, and closes with a short note on
     listening regulations.",
  "dominant_tone": "informational",
  "story_roles": {
    "Speaker 1": "Introduces the product, provides framing,
     and recaps key points.",
    "Speaker 2": "Gives performance figures, shares the
     optimization tip, and explains the regulations."
  }
}
```

Listing 2: Stage 2 (`RULE_OBJECT`).

```
{
  "intent": "Create a concise highlight reel focused on the
     product launch, concrete performance numbers, one
     optimization tip, and a short regulations note.",
  "keywords": ["headphones", "battery life",
     "noise-canceling", "optimization", "regulations"],
  "speaker_roles_priority": [
    "Speaker 1: Include opening and closing framing
     statements.",
    "Speaker 2: Include performance numbers, the
     optimization tip, and regulation details."
  ],
  "rule_text": "Select segments that state the product
     launch, report usage numbers, include one optimization
     with figures, and briefly mention regulations.",
  "budget_instructions": "If constrained, prioritize the
     launch statement, one performance figure, and the
     optimization tip."
}
```

Context. **Speaker 1** (host) frames topics; **Speaker 2** (guest) provides specifics. Target: 60–90 second social media highlight.

Stage 1 (coverage). Stage 1 runs at temperature T=0.7 to encourage diverse generation and maximize recall. It produces an abstractive summary enumerating salient topics and speaker roles without enforcing duration constraints. This high-recall scaffold surfaces all potentially relevant ideas, providing an interpretable overview that later stages reference when ranking concrete transcript spans.

Bridge to Stage 2. From this coverage and any editor focus (query, speaker preferences, duration), the system compiles a `RULE_OBJECT` that consolidates intent and stabilizes selection criteria across transcript segments.

Bridge to Stage 3. Stage 3 applies extractive ranking in short windows, conditioning on the `RULE_OBJECT`. Local ranks are converted to percentile scores ($s \in [0, 1]$) for cross-window comparability, then mapped to categorical priorities (Green/Yellow/Red) for robust triage. The deterministic selector (Sec. E) admits items by priority/score to meet the requested duration, preserves source order, merges adjacencies, and drops zero-length remnants.

Listing 3: Stage 3 (Local ranking output).

```
{
  "sentence_colors": {
    "0": "GREEN", "1": "GREEN", "2": "YELLOW",
    "3": "GREEN", "4": "YELLOW", "5": "RED"
  },
  "sentence_relevance": {
    "0": 0.88, "1": 0.91, "2": 0.55,
    "3": 0.83, "4": 0.47, "5": 0.20
  }
}
```

Editor-facing result description. The assembled 75 s cut opens with **Speaker 1** introducing the headphones, continues with **Speaker 2** reporting battery-life figures and the optimization tip, and closes with a short regulations note. Greens provide the backbone; Yellows extend context as budget permits; source order is preserved for coherence.

Outputs. The rightmost column of Figure 5 shows the two export variants that use the same ranked selections: (i) *Highlight Markers* writes color-coded range markers (Green/Yellow/Red) to the timeline for review; and (ii) *Auto Rough-Cut* directly creates a new sequence from the accepted spans in source order. The deterministic selector enforces a small overshoot bound relative to T and maintains timecode alignment; see Section 3.2 for assembly details.

5 Technical Evaluation

We evaluate our algorithm both on existing text-based summarization metrics and a smaller dataset of video transcripts to explore performance in the context of video editing.

5.1 Information Retention Across Output Constraints

5.1.1 NLP Datasets. **MeetingBank** is a benchmark of long, spoken city council meetings paired with human created *extractive* reference summaries [18]. It contains 6,892 segment-level instances in which each long transcript segment is aligned to a human-written reference constructed from verbatim excerpts of the source. We adhere to the released train/validation/test splits and report all metrics on the test split. We report ROUGE-1/2/L, BLEU, and METEOR, common NLP metrics that evaluate the overlap between input text and output summaries.

MeetingBank-QA is an augmented version of MeetingBank that adds, for each meeting, the full transcript, an abstractive summary, and three curated question–answer pairs targeting concrete facts such as decisions, names, dates, and counts [16]. We generate a strictly extractive summary at the chosen retention budget (i.e., the percentage of input content in the output) and use it as the sole context for answering the curated questions, enabling exact-match evaluation of factual retention. Details regarding setup and runtime are provided in Appx. A.

5.1.2 Time Budgets and Units. We bound how much source content a system may return with a retention budget $r \in (0, 1]$ and a unit $u \in \{\text{seconds}, \text{tokens}, \text{sentences}\}$. Given the input content C and a selection S, budget feasibility requires

$$\text{len}_u(S) \leq r \cdot \text{len}_u(C).$$

Unless noted otherwise, all systems are evaluated at matched budgets using the same unit u, with u set to sentences to mirror the editor's workflow. We evaluate at $r \in \{0.05, 0.10, 0.25, 0.50, 0.75, 0.80, 1.00\}$ to cover tight-budget operation, practical editing, and high-retention behavior. The r=1.00 setting is an upper-bound baseline that returns the entire transcript.

Table 1: MeetingBank extractive summarization at comparable lengths. Higher is better. *Sent.* is the average number of sentences retained.

Model	R-1	R-2	R-L	BLEU	METEOR	Sent.
Oracle (UB)	61.8	46.6	52.6	23.0	52.4	64.9
Lead-3	28.1	19.5	25.8	7.9	23.5	40.8
LexRank	24.6	10.7	19.1	5.9	17.7	53.7
TextRank	30.3	16.0	24.4	9.2	22.1	61.8
Single-prompt	41.4	21.4	30.6	11.1	31.5	52.1
Three-stage	**43.1**	**21.7**	**31.1**	**11.4**	**32.8**	**56.2**

5.1.3 Evaluation Metrics. In MeetingBank, we focus on summarization metrics that include ROUGE-1, ROUGE-2, ROUGE-L, BLEU, and METEOR between each system's linearized extract (selected sentences in source order) and the human extractive reference. ROUGE is case-insensitive with stemming enabled. BLEU is computed with SacreBLEU [41] using default smoothing and case-insensitive scoring. METEOR uses the standard English module with case-insensitive matching. Scores are averaged over instances at each retention budget r.

On MeetingBank-QA we focus on common question-answering metrics, specifically Exact Match (EM). Each curated question is answered by *GPT-4.1 nano* prompted with the system's extractive summary as its only context; the original transcript is not provided. Decoding is deterministic (temperature = 0) and no external tools are used. The model's answer is scored with Exact Match (EM): an item is correct when the predicted string matches the gold answer or any listed alias after lowercasing, punctuation stripping, and whitespace normalization. EM is averaged over QA items at each budget r.

5.1.4 Results on MeetingBank and MeetingBank-QA. Experiments on MeetingBank compare extractive summarization at matched lengths against practical baselines, including Lead-3, LexRank [14], and TextRank [32], together with a single-prompt variant of our own ranker. All variants of our system are strictly extractive and timestamp-faithful to support direct reuse on the editing timeline. We report ROUGE-1/2/L, BLEU, and METEOR. As shown in Table 1, the three-stage pipeline attains higher scores than the practical extractive baselines while preserving sentence-level alignment. It also achieves consistently higher scores than the single-prompt variant at comparable lengths. This improvement indicates that conditioning local ranking on a global machine-readable rule helps maintain stable selection behavior across windows.

To test whether selected content preserves information needed for downstream retrieval under high compression, we evaluate

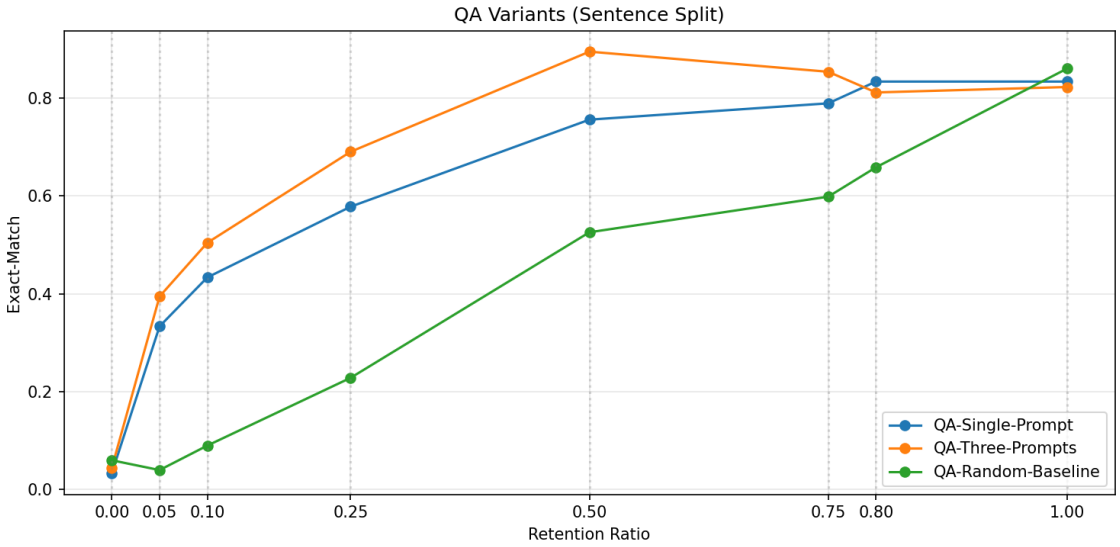

Figure 7: QA exact-match on MeetingBank-QA across retention ratios r. The three-prompt variant leads at low–mid budgets and converges with the single-prompt near full retention; a random baseline improves only at very high retention.

exact-match question answering on MeetingBank-QA across retention ratios r (the fraction of transcript text kept). The three-stage variant maintains higher exact-match than the single-prompt ranker in the low-to-mid retention regime ($r \in [0.05, 0.50]$), which is the range most relevant for short cuts; as retention increases toward full coverage, curves converge as expected (Figure 7). Together, these results suggest that conditioning local ranking on a globally induced rule improves coverage of salient content without sacrificing timestamp fidelity.

5.2 Video Summarization Generation Performance

We evaluate the quality of the *generated* rough cut independent of text overlap metrics by checking that assembled outputs satisfy correctness properties and practical constraints required by editors. For each input transcript and retention budget r, the system produces an extract (Stage 3) and then assembles a rough cut using the deterministic selector with trimming-to-sentence-boundaries enabled. We record (i) duration adherence, (ii) timecode fidelity, (iii) temporal order, (iv) redundancy handling, and (v) model-side assembly latency.

- **Duration error** $\Delta_T = \frac{|\text{len}_{\text{sec}}(S) - T|}{T}$, where T is the target duration. By construction the selector enforces a small bound on overshoot; we verify that $\Delta_T \leq 0.02$ holds for all assembled cuts (Appendix E).
- **Timecode fidelity** (sentence alignment): fraction of clip boundaries that coincide with sentence timecodes when trimming is enabled. The assembly procedure snaps boundaries to sentence limits, so this is guaranteed at 100% under the default setting.
- **Source-order preservation.** Checks that clip start times are nondecreasing in their source timestamps (i.e., no reordering or crossings). Guaranteed by the selector (Appendix E).
- **Assembly latency**: model-side time to produce Stage 1–3 artifacts and apply the selector.

Across evaluated transcripts, assembled cuts satisfy the deterministic properties by design: source-order preservation and sentence-boundary alignment hold, and duration error remains within the $\leq 2\%$ overshoot bound enforced by the selector (Appendix E). For reference dataset sizes and the QA subset, see Appendix B (*Dataset Statistics*). Content retention and faithfulness—whether the kept material answers downstream questions under tight budgets—are captured by MeetingBank/MeetingBank–QA, where the three-stage variant outperforms baselines at low-to-mid retention while remaining strictly extractive and timestamp-faithful (§5.1.4, Fig. 7, Table 1). Details regarding setup and runtime are provided in Appx. A.

6 User Evaluation

Our user evaluation moves beyond automated metrics, such as ROUGE, to assess human-centric factors crucial for real-world adoption: usability, trust, perceived control, and efficiency. We compare three distinct editing workflows to isolate the impact of AI assistance and, specifically, the value of transparent, marker-based interfaces. The findings provide concrete evidence for how AI can best integrate into a creative workflow.

6.1 Ethics and Participant Consent

The user study was approved by the institutional ethics review board at the authors' institution. All participants were contacted through internal mailing lists for professional media editors and provided informed consent before participation. Sessions were conducted remotely via video conferencing with screen sharing, and participants were informed of their right to withdraw at any time without penalty. Each session lasted approximately one hour, and participants were compensated for their time in accordance with institutional standard practices for user research.

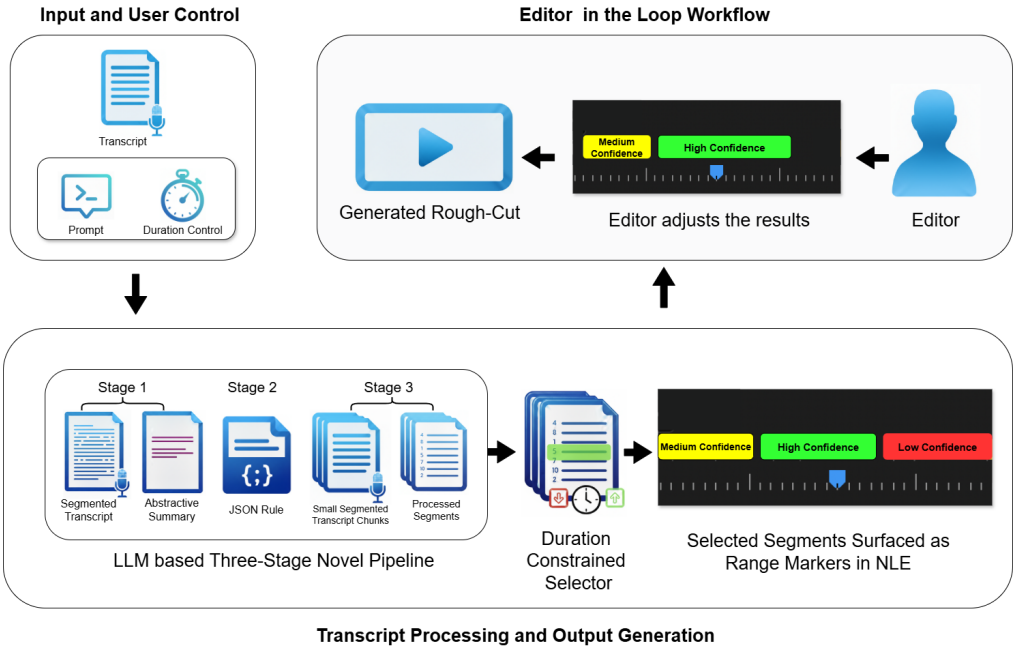

Figure 8: End-to-end human–AI workflow. A three-stage LLM pipeline surfaces confidence-graded, transcript-aligned markers; editors inspect and adjust these, then assemble the rough cut.

Participant identifiers were linked to interaction logs (task timing, workflow actions) and survey responses during data collection to enable within-subjects analysis, then anonymized prior to reporting. Sessions were conducted on the researcher's device via screen sharing. All data was stored securely in compliance with institutional data protection policies.

6.2 Participants

We recruited 11 experienced video editors (P1–P11) via a general-purpose mailing list for media editors. All participants had used *Adobe Premiere Pro* within the last year (self-reported) and were comfortable editing from text transcripts.

Figure 9 shows years of professional editing experience, binned into four levels: *Early Career* (1–5 years; 1/11, 9.1%), *Mid-Level* (6–10 years; 3/11, 27.3%), *Experienced* (11–15 years; 5/11, 45.5%), and *Veteran* (16+ years; 2/11, 18.2%). This skew toward Experienced/Veteran editors reflects our goal of testing workflows with practitioners who routinely deliver client-facing cuts.

Figure 10 summarizes self-reported primary genres. The most common were *Interviews/Talking-head* (8), *Trailer/Sizzle reels* (8), and *Social-media short-form* (7), followed by *Event recaps/highlights* (6), *Narrative/scripted* (6), and *Documentary/journalistic* (5). Genres aligned with our target use cases—dialogue-centric, information-dense content such as interviews, webinars, and explainers.

6.3 Study Design and Procedure

We conducted a study comparing three workflows for transcript-guided video editing. Each participant completed three edits—one per workflow—to enable within-participant comparisons while limiting variance in skill and experience.

Conditions (Independent Variable). Participants experienced all three workflows:

- **Manual (baseline).** Editors created a one-minute cut using standard timeline tools without AI support.
- **Auto Rough Cut (AI-opaque).** The system generated a full draft from a target duration with minimal visibility into selection rationale.
- **Markers-First Assist (AI-transparent).** The system proposed time-aligned segments as confidence-scored markers. Editors inspected these markers, then accepted, trimmed, merged, or rejected them to assemble the final minute; short excerpts and scores surfaced selection cues directly on the timeline.

Stimuli and Task. Each source video was approximately 5–6 minutes long. We used three speech-centric inputs that differ in topic:

- **Video A:** current-events segment on a high-profile public dispute and its implications for politics and social media.
- **Video B:** lifestyle segment comparing solo travel versus group travel, weighing cost, safety, and experience trade-offs.
- **Video C:** etiquette segment on whether political conversations are appropriate at family holiday gatherings.

In every condition, the task was identical: produce a **60-second** summary cut that captures the disagreements and opposing views in the content. To mitigate order effects and learning biases, the

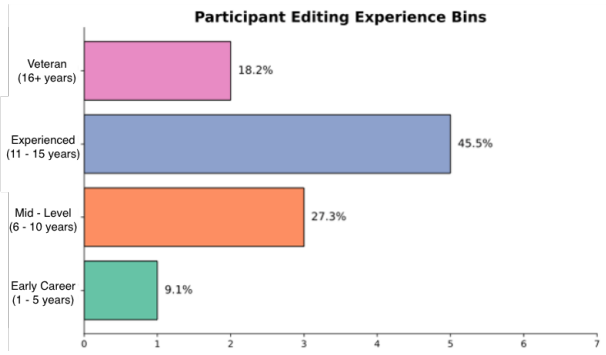

Figure 9: Participant background (n=11). Editing experience (years, binned).

sequence of the three workflows was counterbalanced across participants using a Latin square design.

Procedure. Sessions were conducted remotely with screen sharing using the following process:

(1) **Orientation.** Brief overview of the three workflows; consent and recording confirmation.
(2) **Per-condition edit**: *Familiarize* with the source by reading full transcript, then *edit* for up to **15 min** to produce a 60 s cut, then complete a post-task survey for that condition.
(3) **Comparative survey.** After all three edits, participants compared workflows (forced-choice + short free text).

We then evaluated:

- **Post-task ratings (5-point Likert).** *Efficiency*, *Perceived Quality*, *Trust*, and *Coherence*. For AI workflows, we additionally measured *Control*. Full item wording is in Appendix F.
- **Comparative preferences.** Forced-choice and brief free-text questions after all tasks.
- **Behavioral traces (where available).** Edit time within the 15 min cap, counts of accept/trim/merge actions on markers, and final output duration.

The opaque and marker-based workflows share the same ranking engine and duration algorithm; they only differ in transparency of intermediate outputs. Holding the algorithmic core constant isolates the effect of exposing intermediate evidence (markers) on editor experience and outcomes.

6.4 Results

We collected 5-point Likert ratings after each workflow condition to assess perceived *efficiency*, *quality*, *trust*, *coherence*, *satisfaction*, and *control*.

As summarized in Table 2, both AI-supported workflows improved perceived *efficiency* relative to manual editing. The fully automatic workflow achieved a high median efficiency of 4 [4–5], confirming substantial time savings (Time-Savings median = 4, range = 3–5). The marker-based condition further increased efficiency to 5 [4.5–5], reflecting smoother discovery and navigation when confidence-graded markers were available.

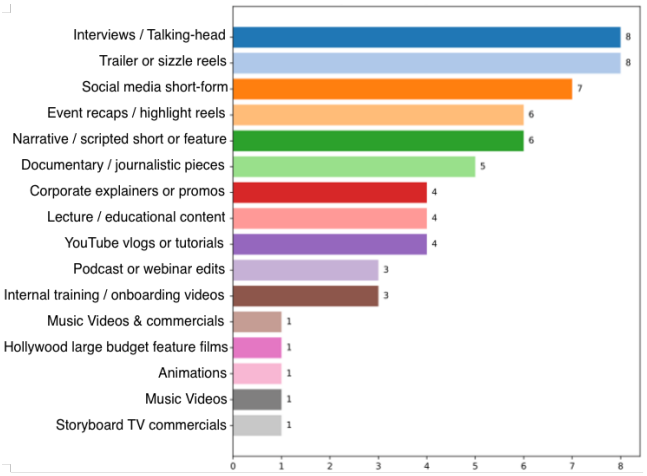

Figure 10: Participant background (n=11). Primary video genres edited, dominated by dialogue-centric work (e.g., interviews/talking-head, trailers/sizzles, short-form social, event recaps).

Manual editing maintained the strongest *coherence* (median = 4 [3–4.5]), consistent with participants' craft-based sequencing, whereas Auto AI coherence dropped to 3 [2.5–4], often due to omitted contextual lead-ins when prompts filtered by speaker.

The marker-assisted workflow achieved comparable coherence while markedly reducing cognitive load.

6.4.1 Perceived Trust and Control. Perceived *trust* and *control* highlight the importance of transparency. Several editors remarked that the Auto AI cut was on par with their own manual edit. **P3** said it was "so good—it's better than what I cut." The marker-based system produced higher medians for both trust (3 [2.5–4.5]) and control (4 [4–5]) compared to the opaque auto condition (trust = 3 [3–4], control = 4 [3–4]). Editors valued the ability to inspect and selectively accept candidate clips, underscoring that agency outweighed raw output quality.

6.4.2 Perceived Output Quality. *Quality* ratings were stable across conditions (Manual = 4 [3–4.5]; Auto = 4 [3–4]; Markers = 4 [2–4]). Despite differences in workflow mechanics, participants felt that all three approaches could produce comparable quality rough cuts when given sufficient attention. However, editors noted that Auto occasionally produced incoherent transitions or omitted important context, requiring more post-hoc verification. In contrast, Manual and Markers offered more predictable quality through direct editorial control and transparent AI suggestions, respectively.

6.4.3 Usage Preferences. Participants also indicated which workflow they would choose under different production scenarios (Table 3). Auto AI dominated perceived speed (*Felt Fastest*: 8/11) and tight-deadline use (4/11). The marker-based workflow was preferred for creating the *best starting cut* (4–6 participants including mixed choices), offering a balance between speed and editorial oversight. Manual editing remained the clear choice for *most control* (6/11) but was also rated as the *most mentally demanding* (7/11).

Table 2: Medians [IQR] of 5-point Likert ratings ($n = 11$). Dashes (–) indicate metrics not collected for specific conditions (e.g., Control was only assessed for AI-assisted workflows).

Metric	Manual	Auto	Markers
Efficiency	3 [2.5–4]	4 [4–5]	5 [4.5–5]
Quality	4 [3–4.5]	4 [3–4]	4 [2–4]
Trust	4 [2–4.5]	3 [3–4]	3 [2.5–4.5]
Coherence	4 [3–4.5]	3 [2.5–4]	–
Satisfaction	3 [2.5–4]	–	–
Control	–	4 [3–4]	4 [4–5]

Table 3: Participant workflow preferences by scenario (counts; $n = 11$).

Scenario	Manual	Auto	Markers	Manual & Markers	Auto & Markers
Felt Fastest	0	8	3	0	0
Tight Deadline	1	4	4	2	0
Best Starting Cut	2	3	4	1	1
Most Control	6	0	2	1	2
Most Mentally Demanding	7	1	1	2	0

6.4.4 Implications for Editing Systems. Overall, we find that the fully automatic approach offers substantial speed gains but limited transparency; manual editing ensures coherence at the cost of effort; and the marker-based approach provides the best compromise—near-automatic efficiency with manual-level control and verifiability. This balance suggests that **confidence-graded, human-in-the-loop workflows may yield the most usable form of AI assistance** for professional editors.

Transparent assistance—via confidence-graded markers, visible excerpts, and provenance—*increased perceived control*, supported *trust calibration*, and improved *practical usability*. Although many editors found the opaque auto-cut as fast or even stronger than their first manual pass, they *still preferred* markers because the rationale was legible and easy to correct. Under tight deadlines, nearly half selected the marker-based workflow (Fig. 11), aligning with higher efficiency for markers (Table 2). In short, editors valued *guidance with control*: transparency made them more willing to rely on AI outputs and integrate them into their workflow.

6.5 Example User Edits

To illustrate how editors used the system in practice, we present two representative examples from the study. Figure 8 summarizes the human–AI editing flow: the model proposes confidence-graded markers, and the editor assembles the final minute by accepting, trimming, merging, or rejecting them.

In the *Markers-First Assist* condition, P4 edited a 5–6 min clip from **Video A**. The system proposed 18 color-coded, time-aligned markers (7 GREEN, 6 YELLOW, 5 RED). P4 treated GREEN spans as the structural backbone (opening setup, key claims, one concrete consequence), promoted two YELLOW spans for brief connective context, and removed a redundant lead-in plus minor pauses. Editing time was under 8 minutes. The resulting 64 s rough cut opens with the dispute summary, includes one evidential detail and one implication for platforms/civics, and closes with a concise takeaway. P4 reported that confidence cues made it obvious what to keep first and what to skim for pacing; markers felt like a reliable shortlist rather than a black box. This example edit is reflective behavior across participants: editors typically *accept* most GREEN markers, *adjust* YELLOW for pacing or glue, and rarely retain RED.

In contrast to P4's confidence-led editing, P2 approached the same toolset almost inversely. They began by manually assembling what they felt was a coherent storyline from memory and note-taking, *then* turned to the AI markers to cross-check coverage and completeness. For **Video A**, the system had proposed 22 markers (8 GREEN, 7 YELLOW, 7 RED). P2 largely ignored these during the initial pass, saying they "didn't want to be biased by colors," and only afterward overlaid them to test whether their cut missed any "green" facts or "yellow" connective logic.

This second-stage comparison led them to restore one green-labeled consequence clip they had overlooked and to trim redundant phrasing around a yellow transition. Editing time was longer (about 13 minutes), but P2 reported feeling "more ownership" over the sequence and described the markers as "a good audit lens, not a recipe."

Whereas P4 leveraged markers as proactive scaffolding (accepting greens, adjusting yellows), P2 used them retrospectively as a post-hoc check for omissions and pacing. This pattern typified a minority of editors who valued interpretive autonomy over efficiency, favoring creative control even at the cost of time.

6.6 Open-Ended User Feedback

Throughout the sessions participants reflected on their current practice and how our tool could influence their editing workflows.

6.6.1 Pain Points of Manual Rough Cuts. Editors consistently framed the rough cut as a first draft whose primary job is to establish structure. Practices varied by project: interview-driven pieces typically began with transcript triage and line pulls, whereas trailers or sizzles were organized around beats, motif repetition, and rhythm. Several editors distinguished between internal and external drafts—producer-facing cuts leaned toward completeness and argumentation, while client-facing versions aimed to minimize open

questions and present a coherent storyline. The metaphors used reinforce this first-draft mentality: one participant likened a rough cut to *"beta software—functional and testable, not feature-complete"* (P1), while another underscored the tedium of manual locating work: *"I have to keep reviewing the same thing over and over again"* (P2). Participants evaluated coherence through the lens of flow, intent, and viewer experience. Ordering and local transitions were the first checkpoints: does the sequence progress logically, and do adjacent moments belong together? Editors also watched for basic alignment between sound and picture—audio–video timing issues or abrupt visual shifts can undermine comprehension even when the transcript reads well. Beyond mechanics, coherence was judged against the story the editor is trying to tell: does the cut advance that intent, or does it drift? Several described a pragmatic rubric rooted in the "five W's," and, where relevant, "how." Others emphasized the need for an arc: *"I expect the beginning, middle, and end"* (P5). In parallel, participants articulated a pragmatic stance toward AI help: *"trust, but verify"* (P6)—confidence rises when intermediate evidence (e.g., markers with excerpts and confidence) makes the output auditable. Editors described manual rough cutting as reliable for coherence but time-consuming and mentally taxing. The dominant friction points were locating material and repeated timeline navigation, which more automated solutions can overcome.

6.6.2 Automatic Rough Cut Greatly Speeds Process, but Markers Provide Explainability. Fully automatic cuts were valued for speed and strong coverage, and several participants noted they could outperform a first manual pass. However, overall participants wanted better control over prompting, speaker/scene handling, and quick preview/correction paths. P6 shared: "It's fast, but I don't know why it chose that clip," indicating a preference for the marker-based solution. Confidence-graded markers were preferred as a starting point under real deadlines because they made acceptance, trimming, and rejection quick and reversible. Editors emphasized perceived control and auditability over raw output quality. "Seeing GREEN/YELLOW with the snippet lets me accept or toss in seconds" (P3). Requiring a brief transcript skim before using AI grounded expectations and sharpened evaluation: even when the auto-cut was "better than my first draft," participants favored markers for verifiable reasoning and easy correction (P5, P8). Editors described the marker-based approach as the most compatible with their day-to-day practice in NLEs: it slots into familiar transcript–timeline routines while reducing rummaging. According to P3, "Markers give me a shortlist I can trust; I'm not hunting all over the place" (P3). P8 said, "It feels like my normal workflow, just pre-sorted—less rework later" (P8). Participants consistently preferred beginning with confidence-graded markers over a blank timeline or a fully assembled opaque cut. They framed the output as scaffolding that accelerates the first hour of work. P2 said, "It's a skeleton I can flesh out; way better than staring at an empty sequence" (P2). P6 shared, "Auto-cut is impressive, but the markers are the part that actually gets me moving" (P6). P9 said, "I can build around the greens, add a couple yellows for context, and I'm 70% there" (P9). Editors highlighted faster early momentum and fewer dead-ends, especially when asked to deliver a 60 s cut on a tight turnaround.

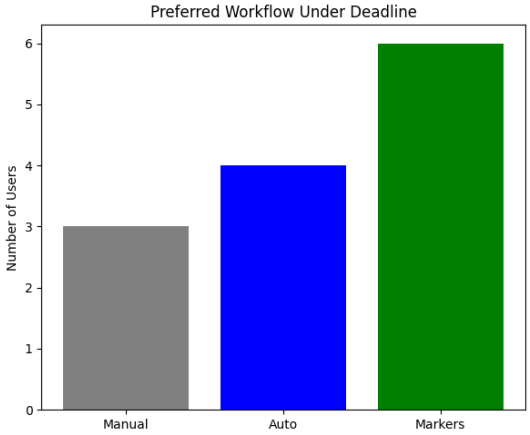

Figure 11: Preferred workflow under deadline (n=11). Most editors preferred *Markers*, followed by *Auto*; *Manual* was least preferred.

6.6.3 Feature Requests and Extensions. Editors consistently asked for ways to deepen control without sacrificing speed or transparency. Several called for lightweight batch operations and persistent context: "Let me demote all YELLOWs at once and keep only GREENs to hit time" (P4), and "I'd like to modify the result rather than start from scratch every time—keep the context, just tweak it" (P7). Others proposed richer feedback and explainability layers: a short text rationale or "shot sheet" for each AI selection (P1), and a fuzzy duration setting to generate slightly longer outputs for later trimming (P5). Participants also emphasized customization and creative prompting support. Requests included a "prompt assistant" that could propose potential storylines or focus options before generation (P6), and the ability to reuse markers as thematic tags for later edits (P8). Across these suggestions, the recurring design goal was reversible automation—AI assistance that remains editable, interpretable, and responsive to individual editorial habits.

7 Discussion

Transparency is often assumed to slow users down; our results suggest that when transparency is *task-aligned* and *tiered*, it can *increase* efficiency by stabilizing expectations and reducing rework. In the markers workflow, explanations are embedded in actionable, timeline-native structure—range markers, priorities, normalized scores, and a plain-language rule surfaced *before* edits—so verification becomes a quick scan rather than a full audit.

7.1 The "Markers vs. Auto" Paradox

Editors frequently praised the speed and coverage of the opaque Auto rough cut (P2, P6, P8, P10), yet preferred *Markers* when stakes or accountability were high (P1, P2, P3, P4, P5, P7). Quantitatively, Markers yielded higher perceived control and slightly higher trust than Auto. Interviews framed Auto as "fastest under deadline, but I still double-check" (P2, P6), and Markers as an "in-between that *preserves agency yet accelerates discovery*" (P1, P2, P3, P4, P5, P7). This aligns with findings on algorithm aversion and calibrated trust: users hesitate to accept uninspectable automation, but confidence

rises when they can see and influence intermediate choices [5, 6, 13, 40].

7.2 Predictability and Trust

Trust hinges not just on accuracy but on predictable, verifiable behavior. Deterministic guarantees—same input yields the same cut, bounded overshoot, and source-order preservation—make behavior legible in advance. Confidence-graded priorities and normalized scores help editors triage and budget review time without implying ground-truth probabilities. In practice, predictability shifted effort from open-ended hunting to targeted verification.

7.3 Design Implications

Expose decisions as anchored units. Turn opaque choices into small, time-anchored, confidence-graded markers so professionals can approve or correct locally rather than redoing an entire pass.

Make transparency tiered and optional. Surface the rule, priorities, and scores before assembly; allow collapsing detail once satisfied. This keeps routine work fast while preserving depth for higher-stakes edits.

Budget-first controls fit editors' mental models. Editors reason in minutes and trade-offs, not parameters. A duration-bounded, order-preserving selector matched how they plan summaries and prevented destabilizing surprises.

Reversible oversight by default. Accept/reject/promote/trim should remain reversible and localized. Combined with determinism, this supports accountability and efficient auditing.

7.4 Limitations and Future Work

7.4.1 Model Choice and Evaluation Methodology. Our pipeline uses GPT-4.1 nano for all three stages and for QA evaluation on MeetingBank. This model was selected for its balance of performance, latency, and deployment cost in production settings where the system must scale to multiple concurrent users. While using the same model family for both generation and evaluation introduces potential bias toward that model's style, we note that: (1) our extractive approach constrains outputs to verbatim transcript spans, limiting generative artifacts; (2) MeetingBank QA primarily tests factual recall via exact-match scoring, which is less sensitive to stylistic preference; and (3) our user study with professional editors provides independent human validation of practical utility. Future work should validate these results with diverse model families and larger-scale human evaluation.

7.4.2 Supporting Multimodal Workflows. Our pipeline is transcript-first. As a next step, we plan to incorporate lightweight audio and visual signals—e.g., slide or scene changes and music onsets—into the same marker grammar in order to support a broader range of content. Our evaluation focuses on long, speech-centric content. Applying the design to adjacent domains (lectures, podcasts, panels) may require minor schema extensions and domain priors.

7.4.3 Personalization and Integration into Professional Workflows. We do not currently model individual preferences. With consent, interaction signals (accept, reject, promote, trim) could adapt the rule or weights while preserving repeatability and privacy so that the system can better adapt to users' individual needs or their common editing preferences across projects. In addition, further in-the-wild deployments and longitudinal studies could test durability of perceived efficiency and control and how markerization shapes real workflows over time.

7.4.4 Ranking-Based Refinement and Alternative Suggestions. A key advantage of our rankingbased approach—currently underutilized—is its natural support for *fallback suggestions*. When an editor rejects a segment, the system could automatically surface the next highest-ranked alternative from the same topical region or time window. This "suggest next best" capability leverages the full ranking rather than treating selection as a binary threshold. Future work should explore interaction patterns for presenting ranked alternatives (e.g., "show 3 runner-up segments near this gap") and investigate whether editors prefer explicit fallback prompts or implicit re-ranking after rejections. This would further justify the ranking paradigm over simpler binary classification approaches, as the full rank order provides a graded utility function for iterative refinement rather than a single accept/reject decision.

7.5 Broader Implications

Embedding generative AI into professional editing tools raises important questions of fairness, bias, and representation. Studies show that summarization systems can under-represent certain speakers or reinforce stereotypes present in training data [15, 31]. For editors, such imbalances affect whose contributions are visible and valued in the final cut. Our system addresses part of this challenge by exposing its selections as transparent, auditable markers, making omissions legible rather than hidden. Nonetheless, ensuring equity in multi-speaker settings remains an open direction for future work, and fairness-aware selection strategies will be critical for responsible deployment.

8 Conclusion

We presented a transparent, marker-based assistant for spoken-video editing that integrates a three-stage prompting pipeline with a Premiere Pro interface. Our system demonstrates that transparency, when delivered as timeline-native markers with confidence and priority cues, can increase efficiency rather than slow it down. In our study with professional editors, participants reported higher trust, perceived control, and satisfaction compared to both manual rough cutting and opaque automatic baselines. Lightweight benchmarks on public datasets further showed that the three-prompt design is competitive with established extractive methods, while uniquely providing predictable, timestamp-faithful selections suitable for professional workflows.

The core contribution lies not only in technical design but also in the interaction pattern of *markerization*: transforming opaque model outputs into anchored, interactive, and auditable units. This pattern points toward broader design opportunities in IUI, where predictability, order preservation, and reversibility safeguard user agency. Looking ahead, larger-scale studies and real-world deployments are needed to validate adoption in production environments. Future directions also include personalization from interaction, multimodal extensions that combine transcript with audio–visual cues, and fairness-aware selection strategies to ensure balanced representation in collaborative editing.

In sum, our work shows that intelligent systems for creative work can move beyond output quality alone. By embedding transparent, predictable, and controllable evidence into existing tools, we can design interfaces that editors not only use, but also trust.

Acknowledgments

We thank the participants in our user study for their time and feedback. We also thank Adobe for travel support and for providing software/equipment used in this work.

Societal & Ethical Impact

Fairness. Transcript-driven selection can under-represent quieter or interrupted speakers and can reflect ASR/LLM biases. Our design mitigates these risks by encouraging editors to inspect intermediate outputs before outputting a final version. We recommend auditing speaker coverage and topic balance during editing and treating markers as suggestions rather than authoritative judgments.

Editor Agency & Accountability. The interface centers human control: editors inspect rationales, adjust budgets, and accept or reject markers. LLM decoding is configurable (e.g., temperature); for reproducible editing we default to deterministic settings with fixed prompt templates and response caching, while allowing higher temperatures when exploratory drafts are desired. Within a fixed model version and these deterministic settings, outputs are effectively repeatable under our length-bounded, timestamp-faithful, extractive policy. All edits remain reversible, and final assembly requires explicit editor approval rather than automatic export.

Privacy & Consent. Inputs may contain personal or sensitive speech. We process only transcript text and timestamps; generated outputs remain extractive and traceable to source segments. Deployments should respect recording consent, data-handling policies, and dataset licenses, and avoid uploading confidential media to third-party services without approval.

Intended Use & Misuse. The system is designed to assist professional and amateur editors in rough-cutting long, spoken recordings. It is not a fact-checking tool and should not be used to summarize without review. Future work includes fairness-aware constraints in the rule specification and tools that visualize speaker/topic coverage to further support equitable selections.

GenAI Usage Disclosure

We used large language models (LLMs) in the system pipeline to generate high-recall summaries over transcript, to induce a machine-readable `RULE_OBJECT` encoding, and to score sentences in short windows conditioned on that `RULE_OBJECT`. Outputs are presented as auditable, timeline-anchored markers; final selections are strictly extractive and timestamp-faithful. LLMs were not used to generate or alter any ground-truth summaries or question-answer pairs used in evaluation. For manuscript preparation, grammar and style assistance tools were used for copy-editing only; all research questions, system design, analyses, and claims were devised by the researchers. Exact prompts are provided in the supplemental material.

References

[1] Evlampios Apostolidis, Eleni Adamantidou, Alexandros I Metsai, Vasileios Mezaris, and Ioannis Patras. 2021. Video summarization using deep neural networks: A survey. *Proc. IEEE* 109, 11 (2021), 1838–1863.

[2] Ido Arev, Hyun Soo Park, Yaser Sheikh, Jessica Hodgins, and Ariel Shamir. 2014. Automatic editing of footage from multiple social cameras. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 1–11.

[3] Aadit Barua, Karim Benharrak, Meng Chen, Mina Huh, and Amy Pavel. 2025. Lotus: Creating Short Videos From Long Videos With Abstractive and Extractive Summarization. In *Proceedings of the 30th International Conference on Intelligent User Interfaces (IUI '25)*. Association for Computing Machinery, Cagliari, Italy, 967–981.

[4] Floraine Berthouzoz, Wilmot Li, and Maneesh Agrawala. 2012. Tools for placing cuts and transitions in interview video. *ACM Transactions on Graphics (TOG)* 31, 4 (2012), 1–8.

[5] Lasse Bohlen, Sven Kruschel, Julian Rosenberger, Patrick Zschech, and Mathias Kraus. 2025. Overcoming Algorithm Aversion with Transparency: Can Transparent Predictions Change User Behavior? arXiv:2508.03168 [cs.LG] https://arxiv.org/abs/2508.03168

[6] Ryan W. Buell and Michael I. Norton. 2011. The Labor Illusion: How Operational Transparency Increases Perceived Value. *Management Science* 57, 9 (Feb 2011), 1564–1579. doi:10.1287/mnsc.1110.1376

[7] ByteDance. 2025. CapCut. Software. https://www.capcut.com

[8] Banghao Chen, Zhaofeng Zhang, Nicolas Langrené, and Shengxin Zhu. 2024. Unleashing the Potential of Prompt Engineering in Large Language Models: A Comprehensive Review. *Patterns* 5, 10 (2024). doi:10.1016/j.patter.2024.101029

[9] Peggy Chi, Tao Dong, Christian Frueh, Brian Colonna, Vivek Kwatra, and Irfan Essa. 2022. Synthesis-Assisted Video Prototyping From a Document. In *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology*. Association for Computing Machinery, New York, NY, USA, 1–10.

[10] Peggy Chi, Nathan Frey, Katrina Panovich, and Irfan Essa. 2021. Automatic Instructional Video Creation from a Markdown-Formatted Tutorial. In *The 34th Annual ACM Symposium on User Interface Software and Technology*. Association for Computing Machinery, New York, NY, USA, 677–690.

[11] Peggy Chi, Zheng Sun, Katrina Panovich, and Irfan Essa. 2020. Automatic video creation from a web page. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*. Association for Computing Machinery, New York, NY, USA, 279–292.

[12] Pei-Yu Chi, Joyce Liu, Jason Linder, Mira Dontcheva, Wilmot Li, and Björn Hartmann. 2013. DemoCut: Generating Concise Instructional Videos for Physical Demonstrations. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology (UIST '13)*. ACM, New York, NY USA, 143–152. doi:10.1145/2501988.2502052

[13] Berkeley J. Dietvorst, Joseph P. Simmons, and Cade Massey. 2018. Overcoming algorithm aversion: People will use imperfect algorithms if they can (even slightly) modify them. *Management Science* 64, 3 (2018), 1155–1170. doi:10.1287/mnsc.2016.2643

[14] Günes Erkan and Dragomir R. Radev. 2004. LexRank: Graph-based Lexical Centrality as Salience in Text Summarization. *Journal of Artificial Intelligence Research* 22 (2004), 457–479. doi:10.1613/jair.1523

[15] Angela Fan, Claire Gardent, Chloé Braud, and Antoine Bordes. 2019. Fair Summarization: A Benchmark for Equity in Natural Language Generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 4055–4066.

[16] et al. Feng. 2023. *MeetingBank-QA-Summary: A Dataset for Evaluating Factual Consistency in Meeting Summarization.* Derived from the MeetingBank corpus with curated question–answer pairs for factual consistency evaluation.

[17] Jonathan Foote. 2000. Automatic Audio Segmentation Using a Measure of Audio Novelty. In *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, New York, USA, 452–455.

[18] Yebowen Hu, Timothy Ganter, Hanieh Deilamsalehy, Franck Dernoncourt, Hassan Foroosh, and Fei Liu. 2023. MeetingBank: A Benchmark Dataset for Meeting Summarization. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Toronto, Canada, 16409–16423. doi:10.18653/v1/2023.acl-long.906

[19] Hang Hua, Yunlong Tang, Chenliang Xu, and Jiebo Luo. 2025. V2Xum-LLM: Cross-Modal Video Summarization with Temporal Prompt Instruction Tuning. In *Proceedings of the AAAI Conference on Artificial Intelligence*. AAAI Press, Washington, DC, USA, 3599–3607. https://ojs.aaai.org/index.php/AAAI/article/view/32374/34529 Reports an average summarization/compression ratio of 16.39% for the Instruct-V2Xum dataset..

[20] Mina Huh, Saelyne Yang, Yi-Hao Peng, Xiang 'Anthony' Chen, Young-Ho Kim, and Amy Pavel. 2023. AVscript: Accessible Video Editing with Audio-Visual Scripts. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (Hamburg, Germany) *(CHI '23)*. Association for Computing Machinery, New York, NY, USA, Article 796, 17 pages. doi:10.1145/3544548.3581494

[21] Adobe Inc. 2025. Adobe Premiere Pro. Software. https://www.adobe.com/products/premiere.html

[22] Jeesu Jung, Hyein Seo, Sangkeun Jung, Riwoo Chung, Hwijung Ryu, and Du-Seong Chang. 2023. Interactive User Interface for Dialogue Summarization. In *Proceedings of the 28th International Conference on Intelligent User Interfaces (IUI*

'23) (Sydney, NSW, Australia). Association for Computing Machinery, New York, NY, USA, 934–957. doi:10.1145/3581641.3584057
[23] Minsun Kim, Dawon Lee, and Junyong Noh. 2025. Generating highlight videos of a user-specified length using most replayed data. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, 1–13.
[24] Eric Laurier, Ignaz Strebel, and Barry Brown. 2008. Video Analysis: Lessons from Professional Video Editing Practice. *Forum Qualitative Sozialforschung / Forum: Qualitative Social Research* 9, 3, Article 37 (Sept. 2008), 21 pages. doi:10.17169/fqs-9.3.1168
[25] Mackenzie Leake and Wilmot Li. 2024. ChunkyEdit: Text-first video interview editing via chunking. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, 1–16.
[26] Mackenzie Leake, Wilmot Li, and Maneesh Agrawala. 2017. Generating Audio-Visual Slideshows from Text Articles. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology (UIST '17)*. ACM, New York, NY, USA, 1–12.
[27] Mackenzie Leake, Hijung Valentina Shin, Joy O. Kim, and Maneesh Agrawala. 2020. Generating Audio-Visual Slideshows from Text Articles Using Word Concreteness. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems (CHI '20)* (Honolulu, HI, USA). Association for Computing Machinery, New York, NY, USA, 1–11. doi:10.1145/3313831.3376519
[28] Daniel Li, Thomas Chen, Albert Tung, and Lydia B Chilton. 2021. Hierarchical summarization for longform spoken dialog. In *The 34th Annual ACM Symposium on User Interface Software and Technology*. Association for Computing Machinery, New York, NY, USA, 582–597.
[29] Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2023. Lost in the Middle: How Language Models Use Long Contexts. arXiv:2301.11079 [cs.CL] https://arxiv.org/abs/2301.11079
[30] Yixin Liu, Kejian Shi, Katherine S. He, Longtian Ye, Alexander R. Fabbri, Liu Pengfei, Dragomir Radev, and Arman Cohan. 2024. On Learning to Summarize with Large Language Models as References. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 8647–8664. https://aclanthology.org/2024.naacl-long.478.pdf
[31] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. 2021. A Survey on Bias and Fairness in Machine Learning. *Comput. Surveys* 54, 6 (2021), 1–35. doi:10.1145/3457607
[32] Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing Order into Text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Barcelona, Spain, 404–411.
[33] Alexander Neubeck and Luc Van Gool. 2006. Efficient Non-Maximum Suppression. In *Proceedings of the 18th International Conference on Pattern Recognition (ICPR)*. IEEE, Hong Kong, China, 850–855.
[34] David Nußbaumer, Bettina Mair, Sandra Schön, Sarah Edelsbrunner, and Martin Ebner. 2024. Evaluating the Efficacy of Automated Video Editing in Educational Content Production: A Time Efficiency and Learner Perspective Study. In *Learning and Collaboration Technologies. HCII 2024 (Lecture Notes in Computer Science, Vol. 14722)*, Panayiotis Zaphiris and Andri Ioannou (Eds.). Springer, Cham, Switzerland, 234–246. doi:10.1007/978-3-031-61672-3_15
[35] OpenAI. 2025. Introducing GPT-4.1 in the API. https://openai.com/index/gpt-4-1/.
[36] OpusClip. 2025. OpusClip. Software. https://www.opusclip.ai
[37] Amy Pavel, Dan B Goldman, Björn Hartmann, and Maneesh Agrawala. 2015. Sceneskim: Searching and browsing movies using synchronized captions, scripts and plot summaries. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. Association for Computing Machinery, New York, NY, USA, 181–190.
[38] Amy Pavel, Dan B Goldman, Björn Hartmann, and Maneesh Agrawala. 2016. VidCrit: video-based asynchronous video review. In *Proceedings of the 29th annual symposium on user interface software and technology*. Association for Computing Machinery, New York, NY, USA, 517–528.
[39] Amy Pavel, Gabriel Reyes, and Jeffrey P. Bigham. 2020. Rescribe: Authoring and Automatically Editing Audio Descriptions. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology (UIST '20)* (Virtual Event, USA). Association for Computing Machinery, New York, NY, USA, 747–759. doi:10.1145/3379337.3415864
[40] Jon L. Pierce, Tatiana Kostova, and Kurt T. Dirks. 2001. Toward a Theory of Psychological Ownership in Organizations. *Academy of Management Review* 26, 2 (2001), 298–310. doi:10.2307/259124
[41] Matt Post. 2018. A Call for Clarity in Reporting BLEU Scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*. Association for Computational Linguistics, Brussels, Belgium, 186–191. doi:10.18653/v1/W18-6319
[42] Mathieu Ravaut et al. 2024. On Context Utilization in Summarization with Large Language Models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 2764–2781. https://aclanthology.org/2024.acl-long.153.pdf
[43] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, Hong Kong, China, 3982–3992. doi:10.18653/v1/D19-1410
[44] Marcelo Sandoval-Castañeda, Bryan Russell, Josef Sivic, Gregory Shakhnarovich, and Fabian Caba Heilbron. 2025. EditDuet: A Multi-Agent System for Video Non-Linear Editing. In *Proceedings of the ACM SIGGRAPH Conference Papers (SIGGRAPH '25)*. Association for Computing Machinery, New York, NY, USA, Article 2, 11 pages. doi:10.1145/3721238.3730761
[45] Than Htut Soe. 2021. AI video editing tools. What editors want and how far is AI from delivering? arXiv:2109.07809 [cs.HC] doi:10.48550/arXiv.2109.07809
[46] Anh Truong, Floraine Berthouzoz, Wilmot Li, and Maneesh Agrawala. 2016. Quickcut: An interactive tool for editing narrated video. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*. Association for Computing Machinery, New York, NY, USA, 497–507.
[47] Tess Van Daele, Akhil Iyer, Yuning Zhang, Jalyn C Derry, Mina Huh, and Amy Pavel. 2024. Making Short-Form Videos Accessible with Hierarchical Video Summaries. In *Proceedings of the CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '24)*. Association for Computing Machinery, New York, NY, USA, Article 895, 17 pages. doi:10.1145/3613904.3642839
[48] David Wan, Jesse Vig, Mohit Bansal, and Shafiq Joty. 2024. On Positional Bias of Faithfulness for Long-form Summarization. arXiv:2410.23609 [cs.CL] doi:10.48550/arXiv.2410.23609
[49] Bryan Wang, Yuliang Li, Zhaoyang Lv, Haijun Xia, Yan Xu, and Raj Sodhi. 2024. LAVE: LLM-Powered Agent Assistance and Language Augmentation for Video Editing. In *Proceedings of the 29th International Conference on Intelligent User Interfaces (IUI '24)* (Greenville, SC, USA). Association for Computing Machinery, New York, NY, USA, 699–714. doi:10.1145/3640543.3645143
[50] Sitong Wang, Zheng Ning, Anh Truong, Mira Dontcheva, Dingzeyu Li, and Lydia B Chilton. 2024. PodReels: Human-AI Co-Creation of Video Podcast Teasers. In *Proceedings of the 2024 ACM Designing Interactive Systems Conference*. Association for Computing Machinery, New York, NY, USA, 958–974.
[51] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers. arXiv:2002.10957 [cs.CL]
[52] Wisecut. 2025. Wisecut: AI Video Editor. https://www.wisecut.ai/. Accessed: 2025-09-29.
[53] Tianyi Zhang, Faisal Ladhak, Esin Durmus, Percy Liang, Kathleen McKeown, and Tatsunori B. Hashimoto. 2024. Benchmarking Large Language Models for News Summarization. *Transactions of the Association for Computational Linguistics* 12 (2024), 39–57. doi:10.1162/tacl_a_00632
[54] Ming Zhong, Da Yin, Tao Yu, Tao Zhang, Abhinav Sinha, Ahmed Hassan Zaidi, Ahmed Hassan Awadallah, Asli Celikyilmaz, Wen-tau Yih, and Dragomir Radev. 2021. QMSum: A New Benchmark for Query-based Multi-domain Meeting Summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*. Association for Computational Linguistics, Online, 5905–5921.

A Runtime Evaluation

We report model-side runtime for the end-to-end pipeline under our default settings (three-stage prompting with ranking on short windows). Times are total model compute times returned by the API. In our setup, short windows correspond to chunks of 50 transcript segments processed independently for local ranking.

Table 4: Model-side runtime for an hour-long video transcript (single-call abstractive summary).

Stage / Call	Input tokens	Time (s)
Abstractive summary	15,803	6.426
Prompt refiner (rule)	488	3.024
Rule eval — ranks (5 calls)	17,264	23.896
Total	33,555	**33.346**

Under the same chunking/window policy, runtime scales approximately linearly with transcript length. A 10-minute input (targeting a 1-minute output) is therefore tokens $\approx$ 5,660 and model time $\approx$ 5.9 s. These times reflect model compute only. End-to-end wall clock in the editing application additionally includes transcript I/O and NLE operations (sequence duplication, marker writes), which depend on the workstation and project, but do not affect the model-side scaling.

B Reproducibility Details

Metric Configurations

ROUGE–1, ROUGE–2, and ROUGE–L are reported with stemming enabled. BLEU uses SacreBLEU with default smoothing and case–insensitive scoring. METEOR uses the standard English module with case–insensitive matching.

Dataset Statistics

We evaluate on MeetingBank and MeetingBankQASummary. MeetingBank contains 6,892 segmentlevel instances across the official train, development, and test partitions. MeetingBankQASummary associates each meeting with three curated questionanswer pairs. QA analyses in this paper use the first 100 meetings from the test split.

Table 5: Dataset overview.

Corpus	Units	Size	Notes
MeetingBank	Segm.-level items	6,892	Official split
MeetingBankQA-Summary	QA pairs per mtg.	3	Verbatim answers
QA evaluation subset	Meetings	100	Test subset

Budget Grid and Units

Experiments evaluate retention budgets at $r \in \{0.05, 0.10, 0.25, 0.50, 0.75, 0.80, 1.00\}$. The primary unit is sentences to align with video–editing operations. All baselines and ablations are length–matched to the same pair (r, u).

Length Matching Procedure

Let C be the full content and S a system selection. Budget feasibility is enforced by $\text{len}_u(S) \leq r \cdot \text{len}_u(C)$ with the same unit u for all systems. When needed, we truncate or extend selections by the smallest additional unit to satisfy the constraint without reordering content.

Segmentation and Timecodes

Selections operate at utterance or clause granularity with preserved start and end timecodes so that every extracted unit maps back to the original media timeline without reinterpretation or paraphrase.

Baselines and Prompts

Baselines include Lead–3, LexRank, TextRank, and a single–prompt variant of the ranker. All prompts used for ranking and scoring, together with command–line arguments for metric computation, are provided in the supplemental material.

QA Construction Policy

QA items target facts such as decisions, names, dates, and counts. An item is correct under Exact Match when the gold answer string, or a listed alias, appears verbatim in the extract used as context.

Implementation Notes

Our preprocessing pipeline comprises ASR alignment, utterance segmentation, lowercasing, and punctuation normalization. Hyperparameters (e.g., learning rate, prompt weights, window size) were tuned on the development split and then held fixed for test evaluation. The ASR and transcript generation modules are hosted internally (i.e. under the control of our system), not via external services. For further implementation details and full hyperparameter tables, see Appendix A.

C Additional Sensitivity Analyses and Implementation Details

C.1 Context Window Size vs. QA Exact-Match

We sweep Stage-3 window sizes $\{5, 10, 50, 100, 200, 400\}$ across retention budgets $r \in \{0.10, 0.25, 0.50, 0.80, 1.00\}$ on MeetingBank–QA. Accuracy is non-monotonic: intermediate windows (50–200 sentences) peak, while very large windows ($\geq$ 400) degrade even at r=1.00, implicating a ranking–horizon limit rather than compression. These trends motivate a default 100–200 sentence window as a robust setting across budgets and meeting lengths.

- **Very small windows** (e.g., 5 sentences) are myopic: they inflate many local winners within each tiny context, boosting recall but harming precision.
- **Very large windows** ($\geq$ 400 sentences) induce early-position preference and truncation effects, depressing accuracy even when r=1.00.
- **Intermediate windows** (100–200) balance cross-topic context against horizon limits, producing the highest EM across budgets.

Precision–Recall Trade-off. Across retention ratios, chunk size induces a stark precision–recall trade-off (Fig. 12). For c=5, recall is high ($\approx$ 0.44) but precision collapses ($\approx$ 0.005), yielding $F_1 \approx 0.01$; for c=200, precision rises $\sim 16\times$ ($\approx$ 0.08) with workable recall ($\approx$ 0.21), improving F_1 to $\approx$ 0.116.

Rationale for a Multi-Stage Architecture. No single window size simultaneously maximizes recall and precision. We therefore decompose: **Stage 1** exploits small-window recall to build a broad fact pool; **Stage 2** compiles explicit selection rules to stabilize criteria; **Stage 3** restores precision with larger (topic-aligned) windows and global competition, exporting confidence-graded, timestamped markers for editing.

Practical Recommendation and Limitations. For a single-pass ranker, use **200 sentences** by default; avoid $\geq$ **400** due to position bias and horizon failure. Complement EM with semantic metrics; consider prompt/order controls to reduce position bias; validate window

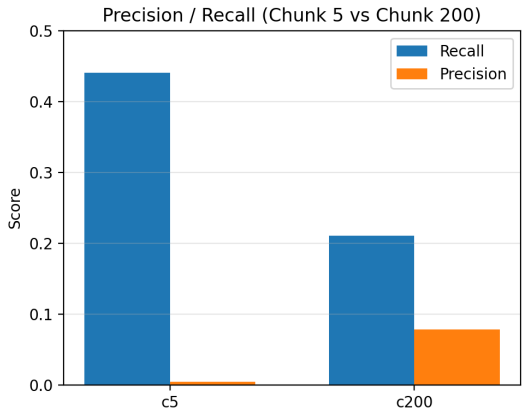

Figure 12: Precision–recall trade-off by window size. Small windows (c=5) boost recall but crush precision; larger windows (c=200) recover precision under similar budgets, yielding a much higher F_1.

size on new domains. The three-stage design remains superior for balancing recall and precision.

C.2 Mechanistic Signs of Failure Modes

Temporal diagnostics of within–window ranks reveal two distinct failure modes. With very small windows (e.g., 5 sentences), the ranker inflates local winners, yielding high recall but poor precision. With very large windows (250–400 sentences), a global position bias emerges: early positions are favored independent of content, effectively truncating the tail of the window. The three–stage design mitigates these issues by (i) capturing high–recall coverage first, (ii) stabilizing criteria via rule induction, and (iii) applying extractive ranking on short, normalized windows to reduce positional effects while preserving timestamp fidelity.

C.3 Cross–Window Rank Normalization

Stage 3 produces per–window relevance values for N candidate sentences. To make these comparable across heterogeneous windows, we convert local ordinal ranks to scale–free scores in $[0, 1]$ prior to any thresholding, averaging, redundancy pruning, or duration–aware selection.

Percentile mapping. Let $r \in \{1, \ldots, N\}$ denote the (lower–is–better) rank of a sentence within its window. We map

$$s_{\text{pct}} = 1 - \frac{r-1}{N-1} \in [0, 1],$$

so that the top–ranked item attains 1, the bottom attains 0, and equal relative standing yields equal scores across windows of different sizes.

MRR–style variant. When emphasizing the very top of the list is desirable, we optionally use a harmonic weighting:

$$s_{\text{mrr}} = \frac{\frac{1}{r}}{\sum_{k=1}^{N} \frac{1}{k}} \in [0, 1],$$

which preserves scale–freedom while allocating more mass to the top ranks.

Placement in the pipeline. Normalization is performed immediately after the model emits per–sentence scores for a window and before any operations that combine information across windows. This ordering ensures (i) cross–window comparability for global thresholding or averaging, (ii) stable behavior under duration constraints, and (iii) consistent visualization/triage when presenting confidence to editors.

Practical effect. Across budgets, percentile normalization reduces volatility introduced by variable window sizes and sentence densities, yielding more predictable inclusion thresholds and smoother budget–performance curves without altering timestamp fidelity.

D Unsupervised Topic-Boundary Detection: Reproducibility & Evaluation

D.1 Method

Encoder and similarity. We use `all-MiniLM-L6-v2` (384-dim) from `sentence-transformers` [43, 51] off-the-shelf. Each utterance u_t is mean-pooled over tokens and L2-normalized; proximity is cosine similarity.

Signals. For each gap i between utterances, we compute four cues and later fuse them. **Semantic change (Sz):** For $K \in \{2, 3, 4\}$, $S(i, K) = 1 - \cos\big(\text{mean}(u_{i-K+1..i}), \text{mean}(u_{i+1..i+K})\big)$. Per meeting: z-score each $S(\cdot, K)$, smooth with a moving average (window = 3), then average over K to obtain $\text{Sz}(i)$. **Foote-style novelty [17]:** From the utterance–utterance cosine matrix, apply a checkerboard kernel at widths $\{8, 12, 16, 20\}$ (subset uses $\{8, 12, 16\}$). Per width: z-score; then average across widths. **Speaker-change prior:** Binary indicator (1 if speakers differ across the gap; else 0). **Pause-density prior:** z-scored count of {pause}/<pause> markers in a small neighborhood of the gap.

Fusion and selection. Per meeting, z-score Score, detect local maxima, apply non-maximum suppression [33] with window $w_{\text{nms}} = 2$, and retain peaks above percentile $\theta \in \{70, 80, 90\}$. We operate in a non-calibrated (**NONCAL**) regime (unknown boundary count).

Representative settings (match Table 6). Baseline Sz: $(\alpha, \beta, \gamma, \delta) = (1, 0, 0, 0)$, $\theta = 80$. *Balanced Sz+Novelty:* $(0.5, 0.5, 0, 0)$, $\theta = 80$. *Macro-oriented:* $(0.5, 0.5, 0, 0.2)$, $\theta = 70$. *Micro-oriented:* $(0.6, 0.4, 0, 0)$, kernels $\{8, 12, 16\}$, $\theta = 90$. *Best NONCAL (subset, spk+pause):* $(0.4, 0.3, 0.1, 0.1)$, kernels $\{8, 12, 16\}$, $\theta = 90$.

Implementation notes. Per-meeting z-scoring normalizes across talks; smoothing window = 3 reduces spurious peaks; $w_{\text{nms}} = 2$ prevents duplicates. All operations are deterministic. Development used `sentence-transformers` (MiniLM), `numpy`, `scipy`, `scikit-learn`, and `pandas`.

D.2 Evaluation Protocol

Data. QMSum-derived corpus of 232 multi-domain meetings [54]. Each item provides a linearized utterance transcript with speaker labels and reference topic spans (utterance index ranges). We treat each span start as a gold boundary. Utterances may contain pause/filler markers used by our priors.

Task and matching. Predict boundaries at gaps between consecutive utterances (gap i corresponds to boundary index i+1). A prediction is a TP if it falls within ±2 utterances of an *unused* gold boundary (greedy 1–1). Unmatched predictions are FP; unmatched golds are FN.

Metrics and operating points. Report Precision/Recall/F1 with Macro F1 (mean of per-meeting F1) and Micro F1 (pooled TP/FP/FN). Unless specified: $K \in \{2, 3, 4\}$, novelty kernels $\in \{8, 12, 16, 20\}$, smoothing window = 3, $w_{\text{nms}} = 2$, and percentile $\theta \in \{70, 80, 90\}$. We sweep $(\alpha, \beta, \gamma, \delta)$ and θ to study component contributions.

D.3 Results

Component contributions. Adding Novelty to Sz is the main gain on the full set (Macro +0.023, Micro +0.018), indicating structural contrast captures boundary evidence beyond local semantic drift. Speaker/pause priors have domain-dependent benefits; with stricter gating they yield the best subset NONCAL result (Macro 0.125, Micro 0.102).

Precision–recall control. Operating point is governed by θ, w_{nms}, smoothing, and context sizes: higher values (and larger contexts) increase precision; lower values increase recall. Macro-oriented settings favor modest pause priors and lower thresholds to capture editor-salient transitions; Micro-oriented settings prefer stricter thresholds and no pause to concentrate on the most confident cuts corpus-wide.

Conclusion. Unsupervised fusion of Sz and Foote-style novelty improves boundary detection; discourse priors help in stricter settings. The method is lightweight, deterministic, and ensures cuts align with natural topic shifts, preserving narrative coherence in downstream rough-cut assembly.

E Duration-Constrained Selection: Policy, Guarantees, and Pseudocode

Setting and notation. Each candidate segment $j = 1..M$ has start/end timecodes, duration d_j, a priority $p_j \in \{\text{G}, \text{Y}, \text{R}\}$, and a normalized score $s_j \in [0, 1]$. Let T be the target duration and $\varepsilon = 0.02\,T$ the overshoot tolerance (2%). Source order is the order of start timecodes.

Selection policy. **(1) Partition & sort:** Partition candidates by priority (G, Y, R). Within each partition, apply a *stable* sort by $(-s_j,\ \text{start}_j)$. **(2) Greens first:** Scan Greens in sorted order and tentatively include each segment. If the cumulative duration exceeds T, iteratively remove the lowest-scoring included Green (tie-break by latest start); if removing any Green would create an avoidable gap near T, shorten the currently last included Green's *tail* to meet T exactly where possible. **(3) Then Yellows, then Reds:** Continue with Yellows (then Reds), in each case appending in sorted order while the cumulative duration $\le T + \varepsilon$. If the next segment would exceed $T + \varepsilon$, shorten that segment's tail so that the final total is in $[T - \min(d_j),\ T + \varepsilon]$. **(4) Order & merge:** Restore strict source order, merge adjacent/overlapping ranges, and drop zero-length remnants. Record selected indices, pre/post trims, and cumulative duration as the selection provenance.

Properties. **Determinism:** Fixed tie-breaks and stable sorting yield a single output for fixed inputs (candidates, T, ε), independent of runtime factors. **Budget feasibility:** The algorithm guarantees a final duration $\le T + \varepsilon$ via the tail-shortening step; when T cannot be met exactly due to granularity, the shortfall is bounded by the smallest unit size (utterance/clip granularity). **Order preservation:** Selections are emitted in source order; the policy never reorders content. **Idempotence:** Re-running the selector on its own output (as candidates) returns the same cut: scores/priority are unchanged; trimming removes any residual slack. **Complexity:** Sorting dominates at $O(M \log M)$; scanning, trimming, and merging are $O(M)$.

Pseudocode. **(1) Input:** candidates $\{(p_j, s_j, d_j, \text{start}_j, \text{end}_j)\}_{j=1}^{M}$, target T, tolerance $\varepsilon = 0.02\,T$. **(2) Partition** into G, Y, R; **stable_sort** each by $(-s_j,\ \text{start}_j)$. **(3) Init** $S \leftarrow [\ ]$, $D \leftarrow 0$. **(4) Phase(G):** for $j \in G$: if $D + d_j \le T$: append j to S, $D \leftarrow D + d_j$; else: repeatedly remove from S the included Green with smallest s (tie: latest start) while $D > T$. If $D < T$ and no removable Green remains, shorten the tail of the last included Green to make $D = T$. **(5) Phase(Y then R):** for $P \in \{Y, R\}$: for $j \in P$: if $D + d_j \le T + \varepsilon$: append j, $D \leftarrow D + d_j$; else if $D < T + \varepsilon$: set $d'_j \leftarrow \min(d_j,\ T + \varepsilon - D)$; if $d'_j > 0$ append a trimmed j with updated end, $D \leftarrow D + d'_j$; break. **(6) Restore order** of S by start; **merge** adjacent/overlapping; **return** S and provenance (selected ids, trims, D).

Outputs and provenance. For each returned range we log: (id, start, end, p_j, s_j, trim_from_end, phase), plus (T, ε) and the cumulative duration. This supports auditability, reproducibility, and user-visible diffs when retargeting durations.

F Post-Study Survey Instrument

Participants completed post-workflow micro-surveys after each condition and a final comparative questionnaire. All rating items used 5-point Likert scales (1 = Strongly Disagree, 5 = Strongly Agree). For AI-assisted workflows, additional items on *Control*, *Duration Accuracy*, and *Time Savings* were included.

Pre-Study Background

Questions covered: video types typically produced, years of editing experience, current rough-cut workflow, definition of a "good" rough cut, and personal criteria for judging narrative coherence.

Manual Workflow

Likert items: Efficiency, Quality, Trust, Coherence, Satisfaction. **Open questions:** What was the most frustrating part of this workflow?

Auto-Generated AI Workflow

Likert items: Efficiency, Quality, Trust, Coherence, Control, Time Savings, Duration Accuracy. **Open questions:** Notes on the Auto-Generated Rough Cut; one improvement you would suggest.

F.1 Implementation Details: Assistant Panel (Premiere UXP)

Marker-Based AI-Assisted Workflow

Likert items: Confidence-Label Usefulness, Locating Key Moments, Quality, Trust, Efficiency, Control, Duration Accuracy. **Open**

Setting	ks	kernels	sw	nms	α	β	γ	δ	θ	Macro F1	Micro F1
Baseline Sz (full)	2,3,4	8,12,16,20	3	2	1.0	0.0	0.0	0.0	80	0.098	0.073
+ Novelty (balanced, full)	2,3,4	8,12,16,20	3	2	0.5	0.5	0.0	0.0	80	0.121	0.091
Macro-oriented (full)	2,3,4	8,12,16,20	3	2	0.5	0.5	0.0	0.2	70	0.120	0.087
Micro-oriented (full)	2,3,4	8,12,16	3	2	0.6	0.4	0.0	0.0	90	0.115	0.095
Best NONCAL (subset, spk+pause)	2,3,4	8,12,16	3	2	0.4	0.3	0.1	0.1	90	0.125	0.102

Table 6: Unsupervised topic-boundary detection under NONCAL F1@±2 on the full dataset (232 meetings), plus the strongest subset configuration.

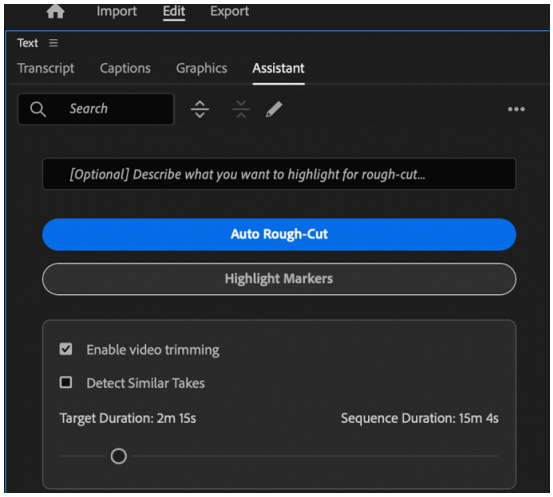

Figure 13: Assistant panel. This illustrative example shows how our research prototype could be surfaced within a video NLE. Enter an optional prompt, set a target duration, then choose *Auto Rough-Cut* or *Highlight Markers*. (Adobe product screenshot(s) reprinted with permission from Adobe.)

questions: Notes on the Marker-Based Workflow; one improvement you would suggest.

Comparative Wrap-Up

Participants compared workflows on perceived speed, preferred method under deadline, best starting cut, editorial control, mental effort, platform expectations, strengths of each workflow, and real-world usage intentions. Full Likert item wording and scales are available upon request for replication.

UXP architecture. The Assistant panel is a dockable UXP panel implemented in React/TypeScript with a thin controller that orchestrates (i) ingest/normalization, (ii) chunking and Stage 3 ranking conditioned on the `RULE_OBJECT`, and (iii) duration-constrained selection (Appendix E). Long-running steps execute in a local Python backend via a lightweight RPC bridge. Processing operates on transcript text and sentence-level timecodes; media never leaves the host application. Panel state (focus text, budgets, thresholds) is serialized alongside outputs for reproducibility.

Marker payload. Each marker stores `id`, `start`/`end` (timecodes), verbatim `text`, `priority` $\in$ {Green, Yellow, Red}, normalized `score` $s \in [0, 1]$, `phase` (e.g., rank/trim), and `provenance`


```
{
 "name": "Decision on Permit — High Confidence",
 "tickTime": 9472256640000,
 "duration": 2949125760000,
 "start": 9472256640000,
 "end": 12421382400000,
 "comments": "High confidence (avg: 86)\nSample: The council approved the permit on May 4th …",
 "sentence_count": 2,
 "avg_confidence_score": 86,
 "rank": 86,
 "query": "overall",
 "options": { "color": "MARKER_COLOR_GREEN", "type": "range" }
},
{
 "name": "Voiceover— Low Confidence",
 "tickTime": 15690568320000,
 "duration": 777288960000,
 "start": 15690568320000,
 "end": 16467857280000,
 "comments": "Low confidence (avg: 5)\nSample: Okay, so I'm just going to say this for the voiceover …",
 "sentence_count": 1,
 "avg_confidence_score": 5,
 "rank": 5,
 "query": "overall",
 "options": { "color": "MARKER_COLOR_RED", "type": "range" }
}
```


Figure 14: Representative range-marker payload emitted by the Assistant panel.

(RULE_OBJECT hash, prompt version). Payloads are strictly extractive and timestamp-faithful. Adjusting the target duration re-invokes the selector with the current T and ε (see Appendix E), updating markers in place. Adjacent or overlapping ranges are merged in source order; zero-length remnants are dropped.

Timeline materialization. Selected units are written back as *range markers* on the active sequence. The extension reads sentence/utterance timecodes from the host's text-based editing API and emits timeline-anchored annotations; sources remain untouched (assembly occurs on a duplicate sequence when requested).