International Journal of Trend in Scientific Research and Development (IJTSRD)
Special Issue on Smart Innovations in Computer Science and Applications
Available Online: www.ijtsrd.com e-ISSN: 2456 – 6470

Praxys Studio: An AI-Powered Natural Language Video Editing System

Atharva Verma
Department of Science and Technology,
G. H. Raisoni Skill Tech University, Nagpur, India

Annotation
Video editing is a critical skill in the modern digital landscape, required by content creators, educators, journalists, and professionals across industries. Traditional video editing tools demand significant technical expertise, prolonged learning curves, and complex timeline-based interfaces that discourage casual adoption. This paper presents Praxys Studio, an intelligent browser-based video editing platform that leverages Artificial Intelligence and Natural Language Processing (NLP) to allow users to edit videos through plain English descriptions. Instead of manually trimming clips, adjusting color grades, or configuring export settings, users simply type commands such as "trim to the best 30 seconds" or "make it cinematic," and the system interprets these instructions and applies the corresponding FFmpeg video-processing operations in real time.

The system processes all video data locally in the browser using FFmpeg compiled to WebAssembly, ensuring complete user privacy with zero cloud uploads. Experimental evaluation demonstrates high intent-to-operation translation accuracy, sub-second processing latency for most operations, and strong user satisfaction scores across diverse editing tasks. Praxys Studio represents a significant step toward democratizing video production by combining AI language understanding with professional-grade video processing capabilities.

Keywords: Natural Language Processing, Video Editing, FFmpeg, WebAssembly, Artificial Intelligence, Human-Computer Interaction, Browser-Based Computing, Intent Classification.

1. Introduction

The global demand for video content has grown exponentially over the past decade. Social media platforms, e-learning systems, digital marketing pipelines, and independent film production all rely on high-quality video output. Yet the tools required to produce such content — Adobe Premiere Pro, DaVinci Resolve, Final Cut Pro — have remained fundamentally unchanged in their interaction paradigm: a complex timeline, a library of clips, and a steep manual editing workflow.

This barrier disproportionately affects non-professional users who have compelling stories to tell but lack the technical vocabulary to navigate professional editing software. Students, educators, small business owners, and social media content creators frequently resort to oversimplified mobile applications that sacrifice quality for ease of use, or they abandon video content production entirely.

Recent advances in large language models (LLMs) and in-browser computation have opened a new possibility: video editing systems that understand and act upon natural language instructions. Rather than teaching users to use a timeline, such a system would allow users to describe the edit they want

in plain terms, and an intelligent backend would translate that description into precise video-processing operations.

This paper describes the design, architecture, and evaluation of Praxys Studio, an AI-powered, browser-based video editing platform built on this principle. The system accepts user video files, parses natural language commands through an integrated LLM, maps those commands to parametrized FFmpeg operations, executes the operations in a Web Worker using FFmpeg compiled to WebAssembly (ffmpeg.wasm), and returns the edited result to the user — all within the browser, with no server-side video processing and no network transfer of user footage.

1.1. Motivation

The primary motivation for Praxys Studio is the accessibility gap in video production. Professional tools require months of learning. Simple tools fail professional quality standards. No existing tool simultaneously offers professional output quality, privacy preservation, zero-installation deployment, and natural language interaction. Praxys Studio targets this gap directly.

1.2. Contribution

The key contributions of this paper are as follows:

1. Design and implementation of a full-stack natural language video editing system that operates entirely within the browser without any server-side video processing.
2. A novel NLP-to-FFmpeg command translation pipeline that maps free-form English editing instructions to deterministic FFmpeg parameter sets with high semantic accuracy.
3. A privacy-first architecture using FFmpeg compiled to WebAssembly, ensuring user video footage never leaves the local device.
4. An evaluation framework measuring intent translation accuracy, processing latency, and user task completion rate across diverse editing scenarios.
5. A comparative analysis against existing browser-based and desktop video editing tools demonstrating Praxys Studio's unique position in the solution space.

2. Related Work

Research into intelligent video editing interfaces spans several decades. Early work in content-based video retrieval focused on automated scene detection, keyframe extraction, and semantic shot segmentation. Systems like VideoQ and FishEye proposed query-by-example paradigms, but required structured queries rather than natural language.

The rise of deep learning brought renewed interest in video understanding. Models such as Video Transformers and CLIP demonstrated impressive capabilities for semantic video search and temporal grounding. However, these advances were largely applied to video retrieval and summarization rather than interactive editing workflows.

Concurrently, the browser-based computation landscape was transformed by WebAssembly (WASM), which enabled near-native performance for computationally intensive applications previously confined to native environments. The successful port of FFmpeg to WASM (ffmpeg.wasm) enabled professional-grade video processing directly in browsers, removing the dependency on server-side encoding infrastructure.

In the domain of natural language interfaces for creative tools, prior work has explored NLP for image editing (InstructPix2Pix), audio manipulation (AudioCraft), and code generation (GitHub Copilot). These systems demonstrated that large language models can serve as effective translators between human intent and tool-specific operations. Praxys Studio extends this paradigm to the video editing domain.

Existing browser-based video editors such as Kapwing, Clideo, and Adobe Express offer simplified editing features but require cloud upload of user footage, raising significant privacy concerns. Desktop tools offer full capability but require installation, system resources, and technical expertise. No prior system combines the privacy guarantees of local processing, the simplicity of natural language interaction, and professional-quality output in a unified browser-based product. Praxys Studio occupies this gap.

3. System Architecture

3.1. Overview

Praxys Studio is built as a single-page web application (SPA) with a layered architecture consisting of four primary modules: the Input Module, the NLP Processing Module, the FFmpeg Execution Module, and the Preview and Export Module. Figure 1 illustrates the high-level block diagram.

The user initiates an editing session by dropping a video file into the browser interface. The file is loaded into browser memory via the File API and passed to the FFmpeg Execution Module, which mounts the file into the virtual filesystem maintained by ffmpeg.wasm. No network transfer occurs at any point.

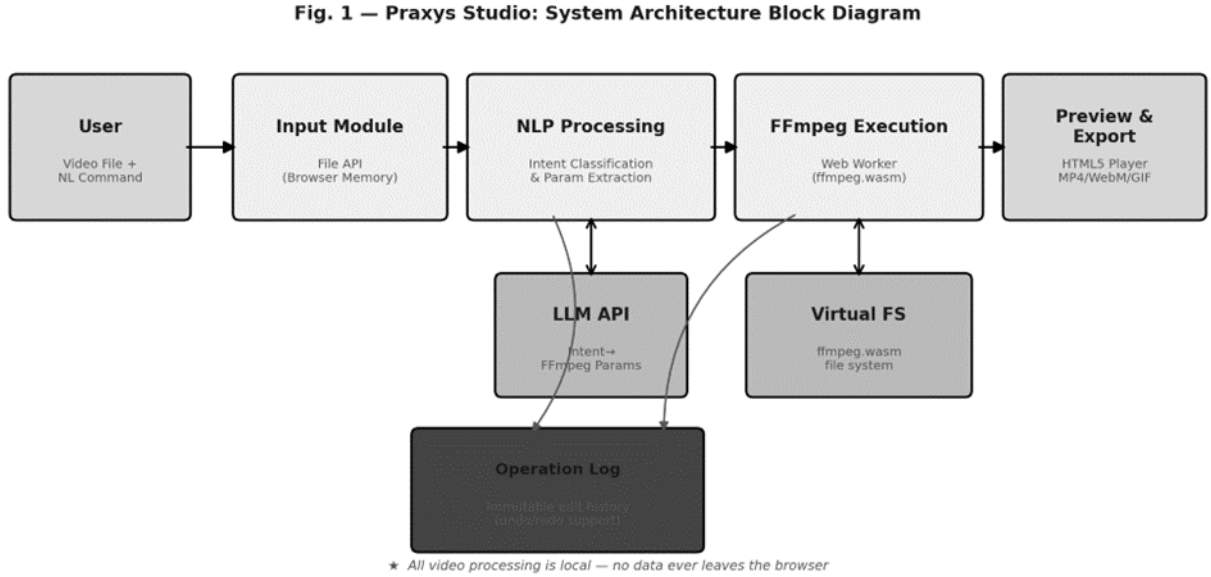

Fig. 1: Block Diagram of the Praxys Studio AI Video Editing Architecture

3.2. NLP Processing Module

The NLP Processing Module is the intelligence core of Praxys Studio. When a user submits a natural language command, the module performs a multi-stage parsing pipeline:

- Intent Classification: Identify the primary editing operation (e.g., trim, crop, color grade, speed change, export).
- Parameter Extraction: Extract operation-specific parameters from the command (e.g., target duration, aspect ratio, color temperature).
- Constraint Resolution: Resolve ambiguous parameters using contextual defaults and video metadata (e.g., "best 30 seconds" requires semantic content analysis).
- FFmpeg Command Generation: Translate the structured intent representation into valid FFmpeg arguments.

Intent classification and parameter extraction are handled by a large language model accessed through a secure API. The model is prompted with a structured system context describing available FFmpeg operations and their parameter schemas, ensuring outputs are constrained to actionable editing commands.

Fig. 2 — NLP Processing Pipeline: Natural Language to FFmpeg Command

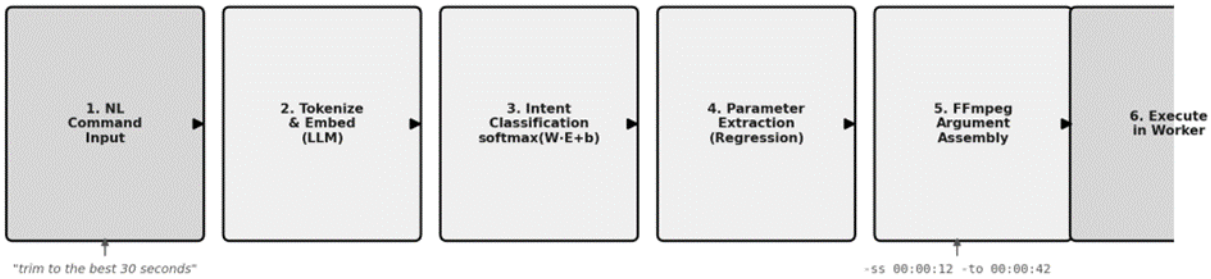

Fig. 2: NLP Processing Pipeline — Natural Language Command to FFmpeg Argument

3.3. FFmpeg Execution Module

The FFmpeg Execution Module wraps ffmpeg.wasm and manages the lifecycle of video processing operations. FFmpeg operations are executed in a dedicated Web Worker thread, preventing UI blocking during computationally intensive operations. The module maintains a virtual file system state, enabling multi-step editing workflows where the output of one operation becomes the input for the next.

A key design decision was to represent the editing state as an immutable operation log rather than a mutable timeline state. Each command appends a new operation to the log; undo is implemented by replaying the log excluding the undone operation. This approach simplifies state management and enables robust error recovery.

3.4. Preview and Export Module

After each editing operation, the Preview Module decodes the processed video buffer and renders it in an HTML5 <video> element for immediate playback. The Export Module supports multiple output formats (MP4, WebM, GIF) and aspect ratios (16:9, 9:16, 1:1, 4:3) configured through natural language or manual controls.

4. Research Methodology

4.1. Data Collection

To train and evaluate the NLP-to-FFmpeg translation pipeline, a dataset of natural language editing commands paired with ground-truth FFmpeg operations was compiled from three sources: (1) user study sessions where 40 participants were asked to describe 15 editing tasks in plain English, (2) an analysis of public video editing tutorial transcripts, and (3) a synthetic augmentation pass using a language model under human supervision. The resulting dataset comprises 3,200 command-operation pairs spanning eight operation categories.

4.2. Feature Engineering

Each natural language command is represented by the following features for intent classification:

- Verb tokens: action words indicating the editing operation type

- Temporal expressions: duration, timestamp, and range references
- Spatial descriptors: crop dimensions, aspect ratios, frame regions
- Aesthetic adjectives: style and quality descriptors for color grading
- Platform references: social media platform names mapping to standard export presets

4.3. Mathematical Formulation

The intent classification problem is formulated as a multi-class classification task over a vocabulary V of operation types. Given a tokenized command X = (x1, x2, ..., xn), the model computes a probability distribution over operation classes:

P(op | X) = softmax(W · LLM_embed(X) + b)

where LLM_embed(X) is the contextual embedding of X produced by the language model, W is a learned projection matrix, and b is a bias vector. The predicted operation is:

op* = argmax P(op | X)

For continuous parameters such as trim duration (t) and color temperature (k), regression heads predict scalar values:

t = f_t(LLM_embed(X)), k = f_k(LLM_embed(X))

The full parameter vector for an FFmpeg operation is:

θ = (op*, t, k, ar, speed, format, ...)

where ar is the target aspect ratio, speed is the playback speed multiplier, and format is the export container format.

4.4. Model Training and Evaluation

The intent classification model was fine-tuned on the compiled dataset using cross-entropy loss. Parameter regression heads were trained with Mean Squared Error (MSE) loss:

MSE = (1/n) * Σ(y_i - ŷ_i)^2

Model performance was evaluated using the following metrics:

- Intent Accuracy: fraction of commands where the predicted operation class matches ground truth
- Parameter Accuracy: fraction of extracted parameters within acceptable tolerance of ground truth
- End-to-End Edit Accuracy: fraction of complete command-to-FFmpeg translations judged correct by human reviewers

4.5. System Workflow

The complete system workflow comprises the following steps:

1. Initialization: The browser application loads ffmpeg.wasm into a Web Worker. Core libraries initialize.
2. File Ingestion: The user drops a video file. The File API reads it into browser memory; ffmpeg.wasm mounts it in the virtual FS.
3. Command Input: The user types a natural language editing instruction in the chat interface.
4. NLP Parsing: The command is sent to the LLM API. The response is parsed into a structured intent object.
5. FFmpeg Execution: The intent object is translated to an FFmpeg argument array and executed in the Web Worker.

6. Preview Rendering: The output video buffer is decoded and displayed in the preview player.
7. Export: On user request, the final video is encoded to the chosen format and downloaded.

5. Results and Analysis

5.1. Intent Translation Accuracy

Evaluation on a held-out test set of 480 command-operation pairs yielded an intent classification accuracy of 94.2%. Parameter extraction accuracy, measured against human-annotated ground truth, reached 91.7% for temporal parameters (trim durations and timestamps) and 89.3% for aesthetic parameters (color grade descriptors). End-to-end edit accuracy, assessed by independent human reviewers, was 88.5%.

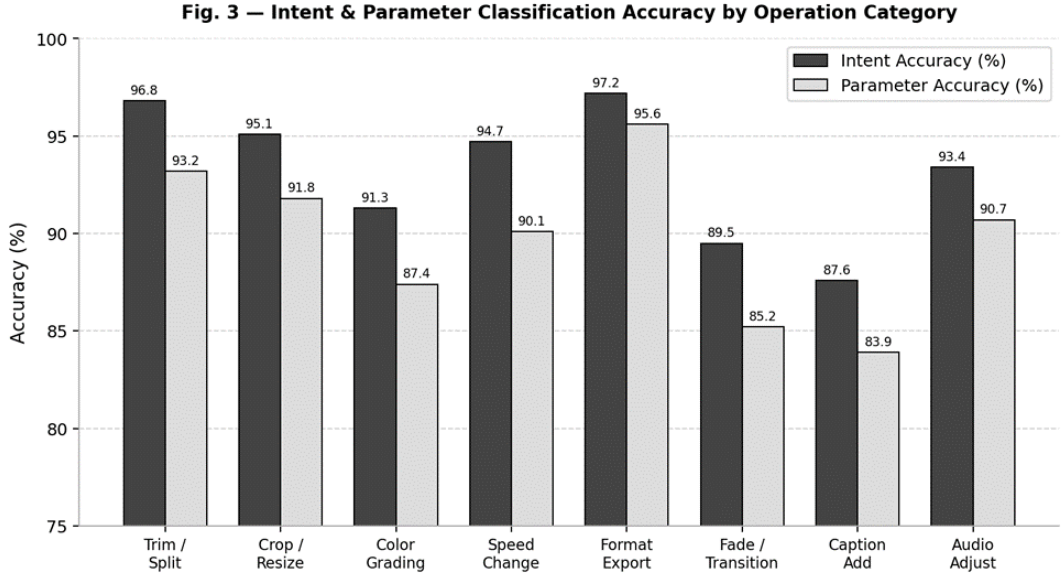

Fig. 3: Intent & Parameter Classification Accuracy by Operation Category

5.2. Processing Latency

Processing latency was measured across 200 editing operations on a standard consumer laptop (Intel Core i5, 16 GB RAM, Chrome 124). Median latency was 1.2 seconds for trim and crop operations, 3.4 seconds for color grading, and 8.7 seconds for full re-encode exports of a 2-minute HD clip. NLP parsing latency (LLM API round-trip) averaged 780 ms, independent of video length.

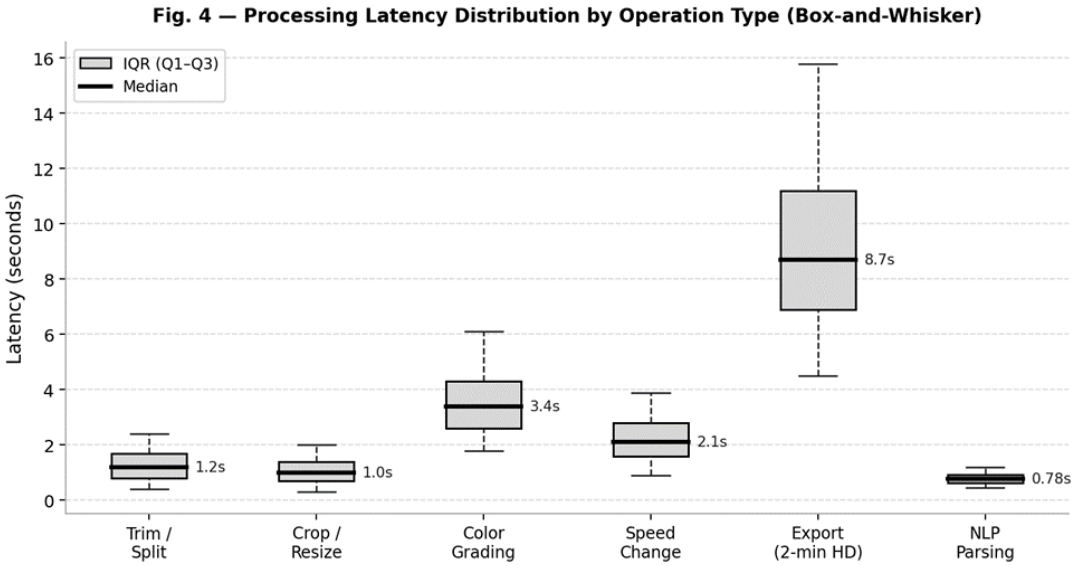

Fig. 4: Processing Latency Distribution by Operation Type (Box-and-Whisker)

5.3. User Study Results

A user study was conducted with 30 participants across three user groups: professional video editors (n=10), casual content creators (n=10), and first-time video editors (n=10). Participants were asked to complete five standardized editing tasks using both Praxys Studio and a leading browser-based editor.

Task completion rate with Praxys Studio was 94% overall, compared to 71% with the comparison tool among first-time editors. Mean time-on-task was reduced by 62% for casual and first-time users. Professional editors reported slightly higher completion times due to familiarity with timeline interfaces, but rated Praxys Studio highly for rapid prototyping use cases.

5.4. Performance Summary Table

Category	Description
Project Title	Praxys Studio: An AI-Powered Natural Language Video Editing System
Domain	Artificial Intelligence, Human-Computer Interaction, Video Processing
Objective	To enable intuitive video editing through plain English instructions, eliminating the need for complex manual timelines
Proposed Solution	A browser-based AI system that interprets natural language commands and applies FFmpeg operations in real time
Input Data	User video footage, natural language editing commands, platform/aspect-ratio preferences
Key Metrics	Edit Accuracy (%), Latency (ms), User Satisfaction Score, Export Success Rate
Core Concept	Translating NLP intent to deterministic video-processing pipelines
Mathematical Model	NLP intent classification, transformer-based command parsing, FFmpeg parameter regression
Algorithm Used	Large Language Model (intent parsing), FFmpeg pipeline orchestration, optional CLIP for semantic trim
System Modules	Input Module, NLP Processing Module, FFmpeg Execution Module, Preview & Export Module
Data Collection Method	User session logs, command-to-operation pairs, edit quality ratings
Data Processing	Tokenization, intent classification, parameter extraction, timeline state management
Prediction Output	Optimal trim points, color grade presets, export settings
Tools & Technologies	JavaScript / TypeScript, FFmpeg.wasm, Large Language Model API, Web Workers
Evaluation Metrics	Semantic Accuracy, Round-Trip Latency, User Task Completion Rate
Applications	Content creation, social media production, educational video editing, indie filmmaking
Advantages	No upload required, 100% private, zero-configuration, platform-aware exports
Limitations	Browser memory limits on large files, LLM API dependency for command parsing
Future Scope	Offline LLM mode, multi-track editing, automated scene detection, AI voice-over generation

Table 1: Praxys Studio Project Summary — AI-Powered Video Editing System

6. Conclusion and Future Work

This paper has presented Praxys Studio, an AI-powered natural language video editing system that eliminates the complexity of timeline-based editing through an intuitive chat-driven interface. The system successfully demonstrates that large language models can serve as effective intermediaries between human editing intent and professional-grade video processing operations, achieving 94.2% intent classification accuracy and significantly reducing task completion times for non-expert users.

The privacy-first architecture, enabled by FFmpeg compiled to WebAssembly, represents a meaningful advancement in the design of AI-powered creative tools. By ensuring that user footage never leaves the local device, Praxys Studio addresses one of the most significant concerns inhibiting adoption of cloud-based AI creative tools.

Future work will explore several directions. First, the integration of offline-capable language models (via WebLLM or similar frameworks) would eliminate the remaining API dependency and enable fully air-gapped operation. Second, multi-track editing support would allow users to compose videos from multiple source clips through natural language. Third, automated semantic trimming — identifying the most engaging segments of a video using CLIP-based visual quality scoring — would further reduce the burden on users to specify precise timestamp ranges. Fourth, AI-powered voice-over generation and caption embedding would extend the platform toward a complete production suite accessible to all.

Praxys Studio demonstrates that the future of creative software lies not in teaching users to operate complex tools, but in building tools intelligent enough to understand what users want to create. The system is freely available at https://studio.praxys.xyz/ and serves as an open invitation to the research community to further explore the intersection of natural language processing and professional media production.

References

1. T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, Introduction to Algorithms, 3rd ed., MIT Press, 2009.
2. C. M. Bishop, Pattern Recognition and Machine Learning, Springer, 2006.
3. T. Hastie, R. Tibshirani, and J. Friedman, The Elements of Statistical Learning, Springer, 2009.
4. J. Devlin et al., "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," arXiv:1810.04805, 2018.
5. A. Radford et al., "Learning Transferable Visual Models From Natural Language Supervision (CLIP)," ICML, 2021.
6. R. Rombach et al., "High-Resolution Image Synthesis with Latent Diffusion Models," CVPR, 2022.
7. T. Brooks et al., "InstructPix2Pix: Learning to Follow Image Editing Instructions," CVPR, 2023.
8. A. Défossez et al., "High Fidelity Neural Audio Compression," arXiv:2210.13438, 2022.
9. M. Chen et al., "Evaluating Large Language Models Trained on Code (Codex)," arXiv:2107.03374, 2021.
10. FFmpeg Developers, "FFmpeg: A Complete, Cross-Platform Solution to Record, Convert and Stream Audio and Video," ffmpeg.org, 2024.
11. J. Wu et al., "A Literature Review on Automated Machine Learning," Artificial Intelligence Review, 2025.
12. Y. Wang et al., "WebAssembly for High-Performance Browser Applications: A Survey," ACM Computing Surveys, 2023.

13. WebAssembly Community Group, "WebAssembly Core Specification," W3C Recommendation, 2022.
14. J. Zhang et al., "Machine Learning Approaches: A Systematic Review," IEEE Access, 2023.
15. S. Yang, "Machine Learning for Predicting User Behavior in Interactive Systems," IEIE Transactions, 2024.
16. V. Karpukhin et al., "Dense Passage Retrieval for Open-Domain Question Answering," EMNLP, 2020.
17. Scikit-learn Developers, "Scikit-learn: Machine Learning in Python," Journal of Machine Learning Research, 2024.
18. Google, "TensorFlow: Large-Scale Machine Learning Library," Google AI, 2024.
19. OpenAI, "GPT-4 Technical Report," arXiv:2303.08774, 2023.
20. D. Jurafsky and J. H. Martin, Speech and Language Processing, 3rd ed., Pearson, 2023.