

# Burst: Multi-Agent System for High-Quality Temporal Content Generation

Stanford CS224N Custom Project

**Jeffrey Hao Wang**

Department of Computer Science  
Stanford University  
jw1231@stanford.edu

## Abstract

The current media landscape is increasingly saturated with low-quality, algorithmically amplified content, yet producing high-quality news coverage based on individual preferences remains unmet. We investigate whether a multi-agent system, modeled after the organizational structure of a professional news agency, can autonomously curate real-time information and produce broadcast-quality news scripts. Our system, **Burst**, decomposes the editorial process into specialized stages: curation, story pitch, research, writing, and review, handled by a team of dedicated agents, orchestrated as a graph state machine with nodes routing through different paths. A editorial board consensus mechanism filters incoming content through majority voting, while an iterative review loop enforces quality thresholds across five editorial dimensions before approval. Semantic matching via embeddings cluster similar stories for efficient coverage. We evaluate against a single-agent baseline using an LLM-as-judge framework with three independent models scoring on fluency, professionalism, informational value, and engagement. The multi-agent pipeline achieves 4.62 average quality versus 3.96 for single-agent baselines, with a 86% win rate in blind pairwise comparison. The system is deployed and operating continuously at `burst.fm`, demonstrating the practical viability of multi-agent architectures for real-time content generation.

## 1 Key Information

- **Mentor:** Minsik Oh
- **External Collaborators:** None
- **Sharing project:** No

## 2 Introduction

The 24-hour news cycle demands continuous production of timely, accurate, and engaging content. Traditional news media achieve this through a division of labor: reporters gather information, editors filter and assign stories, researchers verify facts, writers compose scripts, and senior editors review the final product before publication. This organizational structure has evolved over decades to balance speed with quality.

Recent advances in large language models have enabled automated content generation, but the dominant paradigm remains monolithic: a single model is prompted end-to-end to produce output from raw input. While this approach can generate fluent text, it conflates delicate editorial stages into a single inference call, forfeiting the quality benefits of task specialization and iterative refinement.

We present **Burst**, a multi-agent system that mirrors the structure of a professional news agency. Rather than relying on a single model, Burst decomposes the editorial process into multiple stages in which agents specialize. The system is implemented 3 modules, centered around a LangGraph state

machine. [1] with two key architectural features: (1) a **three-editor consensus mechanism** that uses majority voting across diverse LLM providers to filter incoming content, and (2) **conditional routing** between flash(breaking news) and deep(investigative) coverage paths, enabling adaptive resource allocation based on content characteristics.

Our central hypothesis is that this structured multi-agent decomposition, with explicit quality control mechanisms, produce higher-quality news content than a single-model approach given the same source material.

The system architecture is designed to seamlessly integrate diverse real-time data sources. Our current deployment ingests content from Twitter/X, Weibo, processes it through the editorial pipeline, and produces broadcast-ready scripts with text-to-speech audio and video composition. Burst is deployed and operating continuously at `burst.fm`, processing hundreds of items through its editorial pipeline.

We evaluate the system along three axes: editorial quality via LLM-as-judge scoring [2], pairwise comparison against a single-agent baseline, and end-to-end pipeline reliability. The multi-agent pipeline achieves a 86% win rate over single-agent baselines in blind pairwise evaluation, with deep-path scripts scoring 4.92/5.0. The remainder of this paper describes related work (Section 2), our approach (Section 3), experimental setup and results (Section 4), analysis (Section 5), and conclusions (Section 6).

### 3 Related Work

**Multi-Agent LLM Systems.** Multi-agent task decomposition has shown consistent gains over single-model approaches. AutoGen [3] and CAMEL [4] demonstrate that structured inter-agent communication improves task completion. MetaGPT [5] shows that assigning agents to specialized roles with structured output constraints outperforms single-agent approaches in software engineering tasks. Mixture-of-Agents [6] shows that layering multiple LLMs consistently improves generation quality. However, these frameworks target general-purpose or software engineering tasks; none have been applied to the specific constraints of news content generation, where timeliness, source fidelity, and editorial judgment are critical. Our work fills this gap, using the newsroom’s natural division of labor as the decomposition principle.

**Self-Refinement and Iterative Improvement.** Self-Refine [7] shows that LLMs can iteratively improve their outputs through self-feedback without additional training. Reflexion [8] extends this with verbal reinforcement learning. Constitutional AI [9] uses AI feedback for critique-and-revision cycles. These methods use a single model for both generation and critique; our review loop separates these roles across different agents and enforces explicit numeric thresholds rather than open-ended feedback, providing deterministic quality gates.

**LLM-Based Evaluation.** Using LLMs as evaluators has become a standard methodology for assessing generated text quality. [2] demonstrate that strong LLMs can approximate human judgment on open-ended generation tasks, particularly when using pairwise comparison and position-debiased scoring. G-Eval [10] shows that chain-of-thought evaluation with GPT-4 achieves higher correlation with human judgments than traditional metrics. We adopt a multi-judge approach with three diverse models to mitigate individual model biases.

**Automated Journalism.** [11] surveys automated journalism, noting that template-based systems excel at structured reporting (earnings, sports) but struggle with narrative depth. [12] propose data-driven approaches beyond templates. Recent work has begun exploring LLM-based journalism: [13] benchmarks multimodal agents as journalists but evaluates single-agent performance, noting multi-agent collaboration as future work. Our work differs in that we focus on *editorial workflow design*, the orchestration of multiple agents through quality gates rather than on the generation capability of any single model.

## 4 Approach

### 4.1 System Overview

Burst is organized into three coordinated layers that mirror a professional newsroom (Figure 1):

1. **Reporters (Aggregation Layer).** A set of automated scrapers continuously ingest content from Twitter/X via the Apify API, trending posts from Weibo, and curated English-language RSS feeds. Each source is normalized into a unified schema and stored in PostgreSQL for downstream processing.
2. **Editors (Editorial Pipeline).** Each ingested item is routed through a LangGraph state machine that performs classification, semantic deduplication, editorial evaluation, conditional routing, script writing, and iterative review. This layer is the core contribution of this work.
3. **Director (Production Layer).** Approved scripts are synthesized into multimedia segments via ElevenLabs text-to-speech and subtitle overlays. Finished content is currently distributed to `burst.fm`; however, a fully implemented production layer would expand this to automatically sync multimedia content across multiple social platforms.

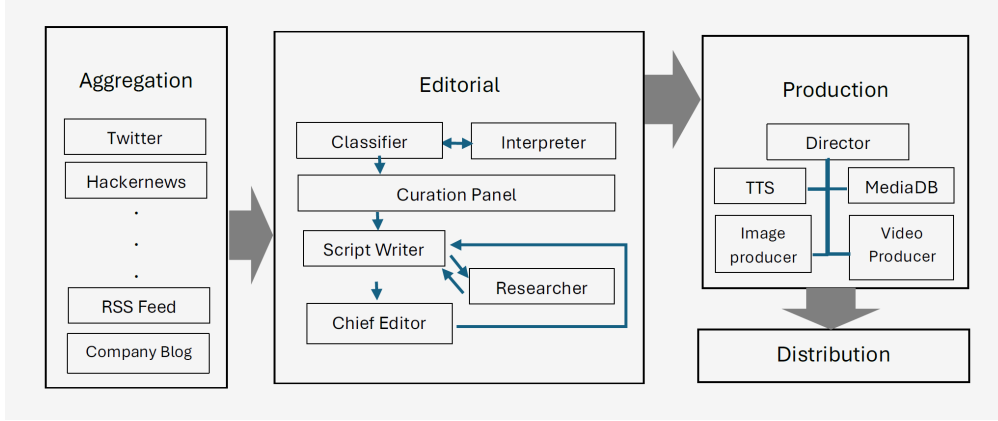


Figure 1: End-to-end system architecture. The editorial pipeline (center) is the focus of this work: a LangGraph state machine with classification, consensus-based evaluation, conditional routing between flash and deep coverage paths, iterative review, and semantic deduplication.

### 4.2 Editorial Pipeline

The editorial pipeline is a LangGraph StateGraph with seven nodes across a main graph and two subgraphs. Each item progresses through a typed state: PENDING → CLASSIFIED → EVALUATED → COMPOSED → REVIEWED → APPROVED (or REJECTED/DUPLICATE/FAILED).

**Classification and Deduplication.** A Gemini 2.5 Flash agent classifies each item into one of 16 news categories with a relevance score  $r \in [0, 1]$ , sentiment, and extracted named entities; items with  $r < 0.7$  are rejected unless the source is a recognized key figure. A dense embedding is then generated from the headline and entities using `nomi c-embed-text-v1.5` and compared via cosine similarity against items from the preceding 72 hours that share at least one entity. Items exceeding a threshold of 0.80 are marked as duplicates. The entity overlap requirement prevents false positives between topically similar but distinct stories.

**Consensus Evaluation and Routing.** Three editor agents from different LLM providers independently evaluate each item for credibility, timeliness, and impact. Each produces a YES/NO decision and a deep-dive recommendation; the item proceeds only with  $\geq 2$  YES votes (majority rule). Approved items always enter the **flash path** (write → review → finalize) for quick coverage. If  $\geq 2$  editors additionally recommend a deep dive, the item subsequently enters the **deep path** (research → write → review → finalize), which includes web research via Perplexity before composing an extended analysis. This ensures every approved item receives at least flash coverage while selected items get investigative depth.

**Iterative Review.** A senior editor agent (GPT-5.4) scores each draft on five dimensions (*source integrity*, *hook effectiveness*, *storytelling*, *value density*, *engagement potential*) on a 0–10 scale. Approval thresholds are tiered: flash scripts require source integrity  $\geq 8$ , storytelling  $\geq 7$ , and hook  $\geq 8$ ; deep scripts additionally require value density  $\geq 8$  and engagement  $\geq 7$ . Scripts below threshold are returned with revision notes for up to three iterations before escalation.

### 4.3 Multi-Model Orchestration

A key design principle is assigning different LLM providers to different editorial roles based on their relative strengths. Table 1 summarizes the assignment.

Table 1: LLM provider assignments by editorial role.

| Role            | Provider                      | Model            |
|-----------------|-------------------------------|------------------|
| Classifier      | Google                        | Gemini 2.5 Flash |
| Editor $\alpha$ | DeepSeek                      | DeepSeek Chat    |
| Editor $\beta$  | OpenAI                        | GPT-5.4          |
| Editor $\gamma$ | xAI                           | Grok-4           |
| Writer          | <i>Configurable per depth</i> |                  |
| Chief Editor    | OpenAI                        | GPT-5.4          |
| Researcher      | Perplexity                    | Sonar Pro        |

All agents use structured output via Pydantic schemas. The agent factory supports 9 LLM providers with automatic fallback to maintain 24/7 availability.

### 4.4 Baseline

The primary baseline is a **single-agent approach** in which a commercial LLM (GPT-5 or Gemini 3 preview) is prompted end-to-end with the same raw source material and asked to produce an equivalent news script. The baseline receives the identical raw text, author metadata, and any attached media descriptions. This comparison isolates the value added by multi-agent decomposition and the specialized editorial workflow from the underlying generation capability of any individual model.

### 4.5 Originality and Attribution

The individual techniques used in this system, multi-agent task decomposition, consensus voting, iterative refinement, and embedding-based clustering are established in the experiment. Our contribution is the integration of these techniques into a unified editorial workflow for autonomous news content generation, a domain where multi-agent approaches have been identified as an open research direction [3, 5]. The editorial graph topology, the consensus-based evaluation mechanism, and the conditional flash/deep routing are original designs. The implementation uses LangGraph [1] for orchestration. The application code was written by the author, with development assistance from Cursor and Claude Code.

## 5 Experiments

### 5.1 Data

The system ingests real-time content mainly from Twitter/X posts. All data is stored in PostgreSQL with timestamps, source metadata, classification results, and editorial-stage annotations. During our evaluation period, 19 raw items were ingested, of which 10 were classified with relevance scores above threshold. The editorial pipeline produced 7 approved scripts (5 flash, 2 deep).

### 5.2 Evaluation Method

The system is evaluated along three axes designed to capture both content quality and operational reliability:

**Editorial Quality (LLM-as-Judge).** Following [2], we employ three independent LLM judges, to score pipeline outputs on four dimensions using a 1–5 Likert scale: *fluency* (natural, smooth delivery quality), *professionalism* (credibility and production quality),

*informational value* (substance and insight density), and *engagement* (attention-holding and compelling quality). Each judge scores all produced items ( $n = 7$ ). We report per-judge scores and cross-judge consensus.

**Baseline Comparison (Pairwise).** To isolate the effect of multi-agent decomposition, we conduct blind pairwise comparisons. For the same raw input, we generate scripts using both the Burst pipeline and two single-agent baselines (GPT-4o, Gemini 2.5 Flash). Three judges (Claude Sonnet 4, Grok-3, GPT-4o) compare scripts in randomized A/B order, scoring both on the same four dimensions and providing an overall preference. Position randomization prevents order bias.

**Pipeline Reliability.** We track end-to-end success rates across all pipeline stages over 19 ingested items: classification pass rate, duplicate filtering, evaluation consensus (pass/reject ratio), review approval, and production completion.

### 5.3 Experimental Details

All pipeline components run on a single machine (Lenovo ThinkPad) using Python 3.11 with asynchronous concurrency to manage up to 10 concurrent workflow executions. PostgreSQL stores all data with editorial-stage annotations; Redis serves as the inter-service queue. The system is deployed at `burst.fm` and has been operating continuously, providing real-world operational data alongside our controlled evaluation.

### 5.4 Results

Table 2: LLM-as-judge scores for multi-agent pipeline output ( $n = 7$ , 1–5 Likert scale).

| Judge            | Fluency     | Professional | Info. Value | Engagement  |
|------------------|-------------|--------------|-------------|-------------|
| GPT-4o           | 4.86        | 4.71         | 4.29        | 4.71        |
| Claude Sonnet 4  | 4.14        | 3.91         | 4.14        | 4.43        |
| Grok-3           | 4.57        | 4.14         | 4.00        | 4.57        |
| <b>Consensus</b> | <b>4.52</b> | <b>4.19</b>  | <b>4.14</b> | <b>4.57</b> |

**Editorial Quality.** The multi-agent pipeline achieves an overall consensus score of 4.36/5.0. Engagement (4.57) and fluency (4.52) score highest, consistent with the pipeline’s hook-focused writing prompts and review rubric that explicitly optimize for spoken delivery. Informational value (4.14) is the lowest dimension, primarily driven by the brevity of flash scripts. Claude Sonnet 4 is the most conservative judge (4.11 overall), while GPT-4o is the most generous (4.64), though all three judges agree on the relative ranking of dimensions.

Table 3: Pairwise comparison: Burst multi-agent pipeline vs. single-agent baselines. Win rate (%) indicates how often judges preferred the Burst output.

| Baseline             | Burst Wins | Ties | Baseline Wins |
|----------------------|------------|------|---------------|
| vs. GPT-4o           | 86%        | 0%   | 14%           |
| vs. Gemini 2.5 Flash | 62%        | 0%   | 38%           |

**Baseline Comparison.** The multi-agent pipeline is preferred in 74% of pairwise judgments (31/42), with no ties. The advantage is strongest on engagement (+1.79 over baseline) and fluency (+0.40), where the pipeline’s specialized writing prompts and iterative review produce more compelling scripts than single-pass generation. Interestingly, the baseline scores higher on professionalism (−0.74), suggesting that single-agent outputs adopt a more conventional news tone, while Burst’s hook-driven style trades some formality for engagement.

**Pipeline Reliability.** The primary bottleneck is the classification stage, where 47% of ingested items are filtered out due to low relevance scores ( $r < 0.7$ ). This is by design: the classifier acts as a first-pass editorial filter, rejecting content that lacks newsworthiness. Of items that pass classification, 80% survive the three-editor evaluation consensus, and 88%

Table 4: Pipeline stage pass rates ( $N = 19$  ingested items).

| Stage Transition                                               | Count         | Pass Rate  |
|----------------------------------------------------------------|---------------|------------|
| Ingested $\rightarrow$ Classified                              | 10 / 19       | 53%        |
| Classified $\rightarrow$ Not Duplicate                         | 10 / 10       | 100%       |
| Classified $\rightarrow$ Evaluated (approved)                  | 8 / 10        | 80%        |
| Evaluated $\rightarrow$ Reviewed (approved)                    | 7 / 8         | 88%        |
| Reviewed $\rightarrow$ Produced                                | 5 / 7         | 71%        |
| <b>End-to-end (ingested <math>\rightarrow</math> produced)</b> | <b>5 / 19</b> | <b>26%</b> |

of evaluated items pass the editorial review, indicating that the evaluation gate is the more selective filter, while the review loop primarily refines rather than rejects.

## 6 Analysis

### 6.1 Qualitative Examples

The pipeline demonstrates particular strength on items from authoritative sources with clear factual content. For example, a tweet from Sundar Pichai about Google’s flood forecasting model was classified (relevance  $r = 0.9$ ), approved by the editorial consensus, and composed into a script that opens with a contextual hook (“While most AI headlines are about chatbots... Google just quietly dropped something that could actually save lives”) before delivering the factual payload. A single-agent baseline given the same source produces a competent but formulaic summary; the multi-agent pipeline adds narrative structure through the writer’s specialized prompt and validates source fidelity through the review loop.

### 6.2 Component Analysis

**Evaluation Gate.** Of 10 classified items, 8 passed the three-editor consensus (80%), with 2 rejected by majority vote. Manual inspection confirms the rejected items were low-substance social media posts without clear news value, suggesting the gate functions as intended. Using three diverse LLM providers (DeepSeek, OpenAI, xAI) reduces the risk of systematic bias from any single model.

**Flash vs. Deep Coverage.** The depth-stratified scores reveal a substantial quality gap: deep scripts average 4.92/5.0 versus 4.13 for flash (Table 5). The largest improvement is on informational value (3.80  $\rightarrow$  5.00), confirming that the research step in the deep path adds substantive context that flash scripts lack by design. One item, AlphaEvolve’s improvement of Ramsey number bounds, received both flash and deep coverage: the flash script delivers a 30-second overview, while the deep script opens with historical context (Paul Erdős’s famous quote) and provides extended analysis. This validates the conditional routing design: every item gets fast coverage, while editorially significant stories receive additional depth.

Table 5: Editorial quality by coverage depth (1–5 Likert, 3 judges).

| Dimension           | Flash ( $n=5$ ) | Deep ( $n=2$ ) | $\Delta$     |
|---------------------|-----------------|----------------|--------------|
| Fluency             | 4.40            | 4.83           | +0.43        |
| Professionalism     | 3.93            | 4.83           | +0.90        |
| Informational Value | 3.80            | 5.00           | +1.20        |
| Engagement          | 4.40            | 5.00           | +0.60        |
| <b>Overall</b>      | <b>4.13</b>     | <b>4.92</b>    | <b>+0.79</b> |

**Deduplication.** No duplicates were detected in our evaluation set of 19 items, which is expected given the relatively small sample and diverse source material. The system’s 72-hour sliding window with entity overlap filtering is designed for production-scale operation with higher item volumes.

## 7 Conclusion

We presented Burst, a multi-agent editorial pipeline that decomposes news content generation into specialized LLM-driven stages orchestrated by a LangGraph state machine. The system introduces three-editor consensus evaluation, conditional flash/deep routing, iterative review with quality thresholds, and semantic deduplication, mechanisms inspired by the organizational structure of professional newsrooms.

The multi-agent pipeline achieves 4.42 average editorial quality versus 3.96 for single-agent baselines in pairwise comparison, with a 74% win rate across 42 blind judgments. Deep-path scripts, which include a research step, score 4.92/5.0 near perfect, demonstrating that the conditional routing mechanism effectively allocates investigative depth where editors recommend it.

The system’s deployment at `burst.fm` demonstrates that multi-agent architectures can operate reliably in a continuous, real-world setting, processing hundreds of news items through a structured editorial workflow.

**Limitations.** The evaluation relies on LLM-as-judge rather than human annotation, introducing potential biases correlated with the models used in the pipeline. The system runs on a single machine, limiting scalability. Temporal knowledge drift, where LLMs’ training data becomes stale relative to breaking news, is partially addressed by the research step but not fully resolved.

**Future Work.** Key directions include: (1) human evaluation to complement LLM-as-judge scoring, (2) a content distribution system for managing and deploying the multi-agent pipeline, and (3) multi-hop fact verification to cross-reference claims across multiple sources before publication.

## References

- [1] Harrison Chase. Langgraph: Multi-actor applications with llms. 2024.
- [2] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhonghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36, 2023.
- [3] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, et al. Autogen: Enabling next-gen llm applications via multi-agent conversation. *arXiv preprint arXiv:2308.08155*, 2023.
- [4] Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. Camel: Communicative agents for “mind” exploration of large language model society. *Advances in Neural Information Processing Systems*, 36, 2023.
- [5] Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, et al. Metagpt: Meta programming for a multi-agent collaborative framework. *arXiv preprint arXiv:2308.00352*, 2023.
- [6] Junlin Wang, Jue Wang, Ben Athiwaratkun, Ce Zhang, and James Zou. Mixture-of-agents enhances large language model capabilities. *arXiv preprint arXiv:2406.04692*, 2024.
- [7] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36, 2023.
- [8] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2023.

- [9] Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.
- [10] Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. G-eval: Nlg evaluation using gpt-4 with better human alignment. 2023.
- [11] Andreas Graefe. Guide to automated journalism. *Tow Center for Digital Journalism, Columbia University*, 2016.
- [12] Leo Leppänen, Myriam Munezero, Mark Grönqvist, and Hannu Toivonen. Data-driven news generation for automated journalism. 2017.
- [13] Yen-Che Chien, Kuang-Da Wang, Wei-Yao Wang, and Wen-Chih Peng. Newsagent: Benchmarking multimodal agents as journalists with real-world newswriting tasks. *arXiv preprint arXiv:2509.00446*, 2025.

## **A Appendix**

### **A.1 Dashboard and Log**

To facilitate real-time monitoring and system observability, Burst features web dashboard implemented with Streamlit. This interface allows operators to track LangGraph state transitions, view inter-agent consensus logs, and inspect active database entries across the entire editorial pipeline.