International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 10 Issue: 03 | March - 2026 SJIF Rating: 8.659 ISSN: 2582-3930

An Intelligent AI-Powered Meeting Analytics Extension with Automated Calendar Synchronization, Productivity Scoring, Visual Insight Modeling, and Custom Task Export Framework

Srimathi R , Monica K , Rithanya S, Vidhurshavarshini S

Computer Science and Business System, Rajalakshmi Institute of Technology

Computer Science and Business System, Rajalakshmi Institute of Technology

Computer Science and Business System, Rajalakshmi Institute of Technology

Computer Science and Business System, Rajalakshmi Institute of Technology**Abstract** - This paper presents a production-oriented intelligent meeting analytics extension that transforms conversational data into structured, actionable outputs within a unified Chrome-based system. Unlike conventional meeting assistants limited to transcription, the proposed approach introduces a semantic execution layer that extracts action items, aligns them with deadlines, and synchronizes them with calendar platforms. The methodology integrates streaming speech understanding, role-aware dialogue segmentation, transformer-based hybrid extraction with constrained decoding, and temporal conflict optimization. Additionally, a dynamic productivity scoring model evaluates engagement quality, task follow-through, and discussion efficiency. A visualization module enables longitudinal analysis of organizational patterns such as speaker contribution imbalance and workload drift. The system also supports interoperable task export across multiple platforms using a canonical task ontology. Experimental evaluation demonstrates strong performance, achieving a macro F1 score of 0.89 for task extraction, 0.93 accuracy in intent-to-calendar mapping, low latency (p95 of 1.8 seconds), and high correlation (r = 0.81) with human productivity assessments. These results highlight the effectiveness of treating meeting intelligence as a closed-loop productivity system.

Key Words: meeting analytics, action item extraction, productivity scoring, calendar synchronization, conversational AI, workflow interoperability.

I.INTRODUCTION

Knowledge work organizations primarily use synchronous virtual meetings for strategic, operational, and technical decision-making. While the overall meeting platforms have become more accessible with better recording options, the conversion of dialogue into accountable execution has been an ongoing challenge. There is often a lack of action ownership, failure to propagate deadlines into personal planners, and difficulty in retrospectively determining the overall value of the meeting. This has created a hidden productivity tax where commitments decay between the meeting and the execution, with the same issues often resurfacing at the next meeting with increasing coordination overheads.

Significant advancements have been made by the research community on automatic speech recognition, dialogue summarization, and meeting understanding [1]–[3]. The overall effectiveness of a meeting AI system, however, is less dependent on the overall linguistics of the system and more dependent on the system's ability to provide structured actions, actors, execution windows, and traceability into overall project management tools.

This paper is a response to the following research question: How can a browser-native meeting intelligence extension facilitate real-time and end-to-end productivity orchestration across conversation understanding, calendar decision support, behavioral scoring, and multi-platform task export?

The gap in the current system is threefold:

1) current systems do not sufficiently model task operationalization: current systems do not consider

ownership confidence, deadline semantics, and scheduling constraints when operationalizing tasks;

2) current productivity proxies such as speaking time and meeting duration are not good predictors of execution outcomes and do not consider post-meeting completion dynamics;

3) current export pathways are platform-specific and brittle and do not have a canonical schema that preserves semantics across transformation.

In light of these limitations, we propose a comprehensive architecture that revolves around a dynamic productivity model and a canonical task ontology. The major contributions of the proposed architecture are:

- A multi-stage AI pipeline for the online extraction of action items with uncertainty-based role attribution and time grounding.
- A calendar synchronization engine for slot optimization, given the availability of participants, priority constraints, and time zone diversity.
- A mathematically derived productivity score, incorporating the quality of engagement, execution follow-through, and conversational efficiency.
- A visualization pipeline for the conversion of raw meeting telemetry into insight layers at the team level.
- A customized export engine for the conversion of canonical tasks into JSON, CSV, and board-based formats like Trello, Notion, Jira, without loss of semantics.

The rest of the paper will discuss the architecture, methodology, implementation, and experimental validation of a comprehensive AI-based meeting productivity infrastructure.

II.RELATED WORK

A.Meeting Understanding and Conversational Intelligence

Previous works such as AMI and ICSI were used as a foundation for initial investigations into meeting structure, topics, and dialogue acts [1], [2]. Other works used neural techniques to perform abstractive summarization and action-oriented generation [4], [5]. Recent transformer-based models focus on semantic coherency but usually maximize text quality instead of execution utility.

Our system differs from previous works in that we focus on operational actionability: every extracted item must include a normalized intent type, an owner probability distribution, deadline confidence, dependency hints, and exportable metadata.

B.Productivity and Collaboration Analytics

Collaboration analytics research has focused on the analysis of participation balance, interruption behavior, and social signals as a measure of the effectiveness of a meeting. However, there has been a lack of a direct connection to the completion of the work. In a real-world setting, a productive meeting is not just a matter of balancing the level of participation, but it is a matter of a meeting where there is a high level of completion of the work within a given timeframe.

Our model improves on existing analytics by connecting the signals from the execution of the work to the execution trace. This provides a real-time score to help a manager separate the “engaging” from the “non-executing” meetings.

C.Scheduling and Workflow Automation

Calendar APIs and workflow tools offer programmatic interfaces, and integration is also subject to manual curation of tasks and schedule windows. Existing automation tools also heavily rely on brittle keyword triggers and cannot handle ambiguous time expressions such as “early next week,” dependencies that span multiple time zones, and overload prevention.

The proposed engine formalizes scheduling as a constrained optimization problem over availability and fairness. This creates a robust mapping between conversational intent and calendar blocks.

D.Limitations in Existing Systems

We identify four common shortcomings: (i) transcript-centric design, (ii) limited uncertainty modeling for inference of owners and deadline, (iii) static productivity metrics, and (iv) limited export interoperability. Our architecture is specifically designed to address such shortcomings.

A further shortcoming is the decoupling between time and other variables: extraction is often carried out in one system and scheduling in another, without a shared model of uncertainty. If action confidence and schedule confidence are separately optimized, a system might commit too many uncertain tasks into high-priority calendar blocks or delay high-confidence tasks because of overly conservative slot allocation heuristics. Our

framework maintains uncertainty through all stages to enable risk-informed decision-making in downstream modules.

Lastly, few existing systems have a machine-accessible provenance layer. In a regulated enterprise, every newly generated task must be traceable to its original utterances, transformations, and operator interventions. The absence of a provenance layer undermines reproducibility, error analysis, and compliance. Our proposed canonical task ontology defines explicit pointers to provenance and version stamps on models.

III.PROPOSED SYSTEM ARCHITECTURE

A. Overall System Design

The platform is designed to have a layered architecture with low-latency edge capture and cloud-native intelligence services. It consists of:

- Edge Layer (Chrome Extension): It is used to capture audio streams (user authorized), receive live transcript segments, monitor speaker turns, and render contextual UI cards.
- Inference Layer: It is used to execute streaming NLP, entity linking, temporal normalization, and action item extraction.
- Orchestration Layer: It is used to execute calendar synchronization, productivity score updates, and export transformations.
- Insight Layer: It is used to aggregate metrics to visualization-ready cubes and dashboard APIs.

The state is event-sourced, and each meeting artifact (such as utterance, candidate action, assignment update, and completion event) is stored as immutable events plus materialized views for fast querying. This supports auditability, rollback, and retrospective model evaluation.

Suggested Fig. 1 (Architecture Diagram): A four-layer architecture with data flow from browser capture to AI services, scheduling optimizer, analytics store, and export adapters

B. AI Processing Pipeline

Let a meeting stream be represented by ordered utterances

$$U = \{u_1, u_2, \ldots, u_T\}, u_t = (s_t, x_t, \tau_t), (1)$$

where s_t is speaker identity, x_t, token sequence, and τ_t timestamp.

The pipeline consists of:

1) **Segmentation and Diarization:** speaker turn correction and overlap resolution.

2) **Intent Span Detection:** token-level tagging for action-able commitments with a transformer encoder.

3) **Argument Filling:** extraction of {owner, task, due_date, dependency, priority}.

4) **Temporal Grounding:** conversion of relative expressions to absolute ISO timestamps.

5) **Confidence Calibration:** temperature-scaled probabilities.

For a candidate task ai, field inference produces:

$$a_i = \langle y_i, o_i, d_i, p_i, c_i \rangle, \quad (2)$$

with type y_i, owner distribution o_i, due-date d_i, priority p_i, and confidence vector c_i.

To preserve online responsiveness, inference uses a bounded-context strategy with a rolling memory window. Let W denote the active token window; each update pro- cesses $x_t - |W| + 1:t$ and attaches summary state $h_t - 1$ from prior windows. This enables long-range contextualization without quadratic growth in memory cost. For speaker disambiguation, we maintain an interaction graph $G_t = (V, E_t)$ where vertices are participants and edge weights encode delegation likelihood estimated from historical handoff patterns. Owner posterior becomes

$$P(o = k \mid a_i) \propto P_{text}(k \mid a_i) \cdot P_{graph}(k \mid G_t) \cdot P_{role}(k) \quad (3)$$

This factorized inference reduces false assignments in meetings with overlapping responsibilities.

C. Calendar Synchronization Engine

Calendar synchronization maps extracted tasks into schedulable blocks. Each task a_i is associated with an estimated effort e_i and deadline d_i. Let S_k be available time slots for participant k. We solve:

$$\min \sum i \sum j \, z_{ij} \, (\alpha \Delta_{ij} + \beta L_{i(\omega_j)} + \gamma C_{ij})$$

$$\sum j \, z_{ij} = 1 \; \forall_i$$

$$z_{ij} = 0 \; if \; j \notin S_{i(\omega)}$$

$$start_j + e_j \leq d_i \; \forall_i with \; z_{ij} = 1$$

$$z_{ij} \in \{0,1\}$$

where Δ_{ij} is slack penalty to deadline, $L_{k(j)}$ workload pressure for assignee, and C_{ij} context-switch cost. Binary $z_{i,j}$indicates assignment of task i to slot j.

If optimization fails under hard deadlines, the engine triggers negotiation policies: split task, reassign candidate owner with second-highest confidence, or propose deadline adjustment.

In distributed teams, time-zone and work-hour policies are first-class constraints. Let Z_k denote allowable local intervals for participant k, excluding protected hours and policy blocks (e.g., focus periods). Feasible slots are filtered through

$$j \in \Omega_i \Leftrightarrow j \subseteq S_{k(j)} \cap Z_{k(j)} \cap H_{\{org\}} \quad (8)$$

where H_{org} captures organization-level restrictions such as no- meeting windows.

The engine also enforces workload smoothing. If cumulative planned effort for assignee k in horizon h exceeds threshold ρ_k, overflow penalty grows superlinearly:

$$L_k = \frac{E_k^{h!_2}}{\delta_k} \quad (9)$$

This discourages pathological concentration of tasks and improves completion reliability.

D. Productivity Scoring Algorithm

We define a dynamic productivity score $P_{m(t)}$ for meeting m at time t:

$$P_{m(t)} = w_E E_{m(t)} + w_X X_{m(t)} + w_D D_{m(t)} - w_R R_{m(t)} \quad (10)$$

Subject to $w_E + w_X + w_D + w_R = 1 \; and \; w_i \geq 0$

Components:

•$E_{m(t)}$: engagement quality (balanced participation, low derailment, constructive response latency).

•$X_{m(t)}$: execution follow-through (fraction of tasks com- pleted on or before scheduled blocks).

•$D_{m(t)}$: discussion efficiency (decision density and low repetition ratio).

•$R_{m(t)}$: risk term (owner ambiguity, unresolved depen- dencies, deadline volatility).

Engagement sub-score:

$$E_m = \lambda_1(1 - G) + \lambda_2 Q + \lambda_3(1 - I) \quad (11)$$

where G is Gini coefficient over speaking contributions, Q semantic relevance ratio, and I interruption rate. Execution sub-score uses discounted completion:

$$X_{m(t)} = \left(\frac{1}{N}\right) \sum_{i=1}^{N} exp - \eta \; max(0, t_i^c - d_i) \; (12)$$

where $t\hat{}_i$ is completion time.

The score is recomputed whenever status events arrive, making productivity a trajectory rather than a static label.

Weight learning is data-driven. Given expert labels $P\tilde{}_m$, we fit weights by constrained regression:

$$min \sum_m \left(\hat{P}_m - w^T f_m\right)^2 + \lambda \left||w|\right|^2 \quad (13)$$

$$s.t \; 1^T w = 1, w \geq 0 \quad (14)$$

where $f_m = [E_m, X_m, D_m, -R_m]$. Temporal smoothing avoids score oscillation with exponential moving average:

$$\hat{P}_{m(t)} = \kappa P_{m(t)} + (1 - \kappa)\hat{P}_{m(t-1)} \quad (15)$$

Interpretability is enforced by exposing per-factor contributions $w_i f_i$ and confidence intervals estimated via bootstrap over event segments.

Suggested Fig. 2 (Scoring Model Diagram): Causal graph from conversational features and completion events into latent factors E, X, D, R and final score P_m.

E.Visualization Engine (Data Transformation Pipeline)

The raw data is transformed through a three-stage pipeline:

1) Normalization: schema harmonization and imputation of missing values.

2) Feature Cubing: aggregates over time windows (daily, weekly, sprint-level).

3) Rendering API: pre-computed data for trend lines, heatmaps, workload plots, and speaker analytics.

For metric v and window ω, aggregation is:

$$\overline{\{v\}} = \left(\frac{1}{|\omega|}\right) \sum_{t \in \omega} v_t \quad (16)$$

And drift measured by

$$\nabla v_w = v_w - \bar{v}_w \quad (17)$$

The engine supports anomaly overlays when $|\nabla v_\omega|$ exceeds robust thresholds (median absolute deviation).

Visualization primitives are intended to support direct answers to operational questions such as "Which teams over-commit relative to completion throughput?", "Where are ownership bottlenecks emerging?", and "How does speaker asymmetry correlate with unresolved task carryover?" For this purpose, the dashboard service materializes composite indicators like backlog pressure index, decision to action ratio, and reassignment churn.

Late-binding query templates enable us to support comparisons of cohorts like teams, projects, or sprint phases without retraining the model. Query latency is reduced through Suggested Fig. 3 (Visualization Pipeline Diagram): Event stream → feature store → analytical transformations → dash- board tiles.

F. Custom Task Export Framework

To prevent vendor lock-in, we define a Canonical Task

Object (CTO):

$CTO = \{id, title, owner, due, priority, tags, s\}$
Platform adapters compile CTO into target schemas:

- JSON/CSV for archival and analytics.
- Trello card payloads (list mapping, label encoding).
- Notion database rows (property typing, rich text support).
- Jira issue templates (project keys, issue type, sprint la- bels).

The constraint validator ensures that mandatory fields are present prior to export, while remediation prompts are provided if confidence is low.

Semantic preservation is achieved via adapter contracts. Adapters are implemented to provide bidirectional data types for status states, priority scales, and dependency references. For example, priority scale {P 0, P 1, P 2, P 3} is mapped to CTO custom fields or Jira custom fields or Trello labels while preserving reversible metadata. Failed exports are addressed via dead-letter queues. The framework also includes a versioned schema registry to prevent fragmentation between tool ecosystems. Changes to CTO fields are backward compatible via migration descriptors to ensure historical tasks remain queryable under unified semantics.

IV. METHODOLOGY

A.NLP Models Used

The extraction backbone combines:

- A domain-adapted encoder (RoBERTa-family) for token- level action-span tagging.
- A sequence-to-structure decoder for argument completion.
- A temporal parser for relative date normalization.
- A speaker-aware resolution module for owner disam- biguation.

ASR is delivered as a streaming text service over meeting platform UIs. Our contribution begins at semantic processing and orchestration.

Model training employs multi-task supervision over action de-tection, owner classification, and deadline parsing with total loss

$$L = \mu_1 L_{span} + \mu_2 L_{owner} + \mu_3 L_{time} + \mu_4 L_{priority}. \quad (19)$$

B.Feature Extraction Techniques

The features are categorized into four families:

1) Lexico-semantic: obligation verbs, commitment modals, temporal noun phrases.
2) Conversational structure: adjacency pairs, response latency, turn-transition entropy.
3) Role and authority: historical ownership graph, manager/individual-contributor priors.
4) Execution context: unresolved backlog count, partic- pant calendar load, sprint phase.

Speaker contribution entropy is

$$H_s = -\sum_{k=1}^{K} p_k \log p_k \quad (20)$$

where p_k is proportion of contribution by speaker k. Low entropy with single-speaker dominance may reduce meeting quality unless role-specific constraints justify asymmetry.

We additionally compute discussion recurrence vectors to measure repeated unresolved topics. Let z_t be cluster ID of utterance t in semantic topic space; recurrence over lag window ℓ is

$$R_{rec} = \left(\frac{1}{T}\right) \sum_{t=l+1}^{T} I[z_t \epsilon \{z_{t-l}, \dots, z_{t-1}\}] \cdot I[\neg\, resolved(z_t)] \quad (21)$$

High recurrence penalizes discussion efficiency unless accompanied by new evidence signals.

Another feature family models commitment strength. Utterances containing explicit accountability frames ("I will deliver by Friday") receive higher commitment priors than weak state- ments ("we should maybe consider..."). These priors improve calibration of extracted tasks and reduce false positives from speculative brainstorming.

C.Engagement Scoring Metrics

Engagement includes participation, topic adherence, and interruption discipline. Topic adherence uses embedding similarity between each utterance and active agenda embeddings. Let ϕ_{xt} be utterance embedding and ψ_A agenda embedding set:

$$q_t = \max_{\{a \in A\}} \cos\big(\phi(x_t), \psi(a)\big) \quad (22)$$

Then,

$$Q = \frac{1}{T}\sum_{t=1}^{T} 1\ [q_t > \theta_q] \quad (23)$$

The interruption rate is normalized by turn count. The constructive interruption exception, or urgent correction signals, is distinguished by intent tags.

To counter speaker count bias, engagement components are normalized by expected interaction topology. In small meetings, if K ≤ 4, turn concentration is normal. In larger meetings, turn concentration is a sign of exclusion. Topology-aware normalization is used:

$$E_m' = E_m \cdot \xi(K, \pi), \quad (24)$$

where π encodes planned meeting format (briefing, workshop, stand-up, review). This prevents unfair penalization of legitimate facilitation styles.

Algorithm 1 Streaming Action-Item Synthesis and Validation

Require: Utterance stream U, model ensemble M, threshold τc
Ensure: Validated task set A
1: Initialize A ← ∅, context buffer B ← ∅
2: for each utterance ut in U do
3: Append ut to B and update speaker context graph
4: St ← DETECTACTIONSPANS(B,Mspan)
5: for each span s ∈ St do
6: a ← FILLARGUMENTS(s, B, Marg)
7: a.due ← NORMALIZETIME(a.due, meeting timezone)
8: a.owner ← RESOLVEOWNER(a, speaker graph)
9: a.conf ← CALIBRATECONFIDENCE(a)
10: if min(a.conf) ≥ τc then:
11: A ← A ∪ {a}
12: else:
13: Emit clarification prompt to UI card
14: end if
15: end for
16: **end for**
17: **return** DEDUPLICATEANDLINK(A)

D. Data Flow Diagram Explanation

Data flow proceeds as follows:

1) Browser extension takes the live meeting segments and metadata.

2) Streaming queue batches segments into micro-windows.

3) AI pipeline outputs candidate tasks with associated uncertainty vectors.

4) Calendar engine performs scheduling with associated constraints to produce booking recommendations.

5) Productivity engine updates the score trajectory with associated meeting and post-meeting events.

6) Visualization and export services use the canonical events for the purpose of dashboards and third-party APIs

This is done to independently scale the inference-heavy modules as well as the I/O-heavy modules.

E. Algorithm 1: Online Action Item Synthesis

F. Algorithm 2: Calendar Slot Optimization and Export Dis- patch

V. IMPLEMENTATION DETAILS

A. Frontend: Chrome Extension Architecture

The extension is built with the extension pattern extension V3:

• Service Worker: session life cycle, token refresh, routing events

• Content Scripts: page instrumentation for the meeting, capture hooks with permissions.

• Popup/Side Panel UI: live action cards, confidence, and clarification UI.

• Local Cache Layer: temporary caching with retry queue to handle network intermittency

Algorithm 2 Schedule-Aware Task Dispatch

Require: Validated tasks A, availability matrix V, export targets T
Ensure: Scheduled tasks and platform-specific payloads
1: for each task $a_i \in$ A do
2: Generate feasible slots Ωi from V and effort estimate e_i
3: Score each slot j $\in \Omega_i$ using cost $J_{i,j}$
4: j∗ ← arg minj∈Ω_i $J_{i,j}$
5: if Ωi = ∅ then
6: Trigger fallback policy: split/reassign/deadline ne- gotiation
7: **else**
8: Reserve slot j^* and update participant load vectors
9: **end if**
10:**end for**
11:**for** each target platform $t \in$ T **do**
12: payloads ← COMPILEFROMCTO(A, t)
13: PUSHVIAADAPTER(payloads, t)
14: **end for**

The UI interactions are event-driven. On detection of low-confidence fields, the panel quickly prompts the user to confirm the result to maximize precision without interrupting the meeting.

The software engineering approach to this extension is to encapsulate privileged operations in the service worker and minimize content scripts' invasiveness. This minimizes security risk in case of changes in browser contexts and permission audits. The streaming updates are rendered with optimistic UI states, with eventual reconciliation after backend acknowledgement.

For robustness, an offline-first queue is used to store unsent events with encryption per record. When disconnected, the user can still see analytics snapshots, and synchronization resumes automatically without duplicates with sequence IDs.

B. Backend AI Integration

Backend The backend services are further decomposed to containerized mi- croservices:

• Inference Gateway (gRPC): batching, model routing, and timeout.

• Task Orchestrator: finite state machine for candidate to validated task transition.

• Scoring Service: real-time score refresh and historical recalibration.

• Analytics Warehouse: columnar storage for longitudinal metrics.

A message broker acts as an abstraction between inference and external API throughput constraints. Idempotence of event IDs prevents export duplicates on retries.

Service-level objectives are enforced with circuit breakers and adaptive timeouts. If a secondary model stage degrades, the orchestrator can fallback to lightweight extraction while flagging reduced confidence. This graceful degradation ensures that the essential productivity functionality remains available during failures. Mixed-precision inference and adaptive batching are the two main techniques that model serving employs. The gateway balances latency and throughput through adaptive batching based on queue pressure. During increased loads, micro-batches maximize GPU usage without violating real-time response constraints.

C.API Integrations

The system integrates with major calendar and task APIs using OAuth 2.0 and least privilege scope support. The integration patterns include:

•**Calendar Read/Write:** freebusy query, event creation, update conflicts.

•**Task Board Sync:** project/list discovery, issue/card creation, status notifications.

•**Webhook Intake:** post-meeting completion notifications for score recalculation.

A provenance tracker maps each artifact ID to internal CTO IDs to provide consistent updates.

Given the changing nature of third-party APIs, contract tests are employed to verify the payload semantics of the API, mocking the schema of the provider. Version pinning and compatibility matrices minimize the likelihood of breakage with updates to the provider.

Webhook signatures are validated, and replay detection prevents unauthorized notifications.

Rate limit schedulers are employed to ensure high-urgency tasks are executed when the budget is reduced.

Guaranteed delivery is provided to defer low-priority exports.

EXPERIMENTAL RESULTS AND PERFORMANCE EVALUATION

A. Experimental Setup

The evaluation utilizes public meeting data sets (AMI/ICSI style dialogues for structural validity) and anonymous enterprise pilot meeting data in software, consulting, and research teams. The testbed has 1240 meetings (average 43 min), 18,600 action items, and 52,000 status events for post-meeting completion analysis.

The evaluation metrics include extraction precision, recall, F1, owner attribution, temporal normalization, scheduling feasibility, score human correlation, and latency percentiles.

For robust generalization, the data is split along organizations and time periods to avoid leakage of meeting template and participant phrasing. Hyperparameter tuning is performed using a development set with Bayesian optimization over learning rates, dropout, and decoding constraints. The final evaluation is performed using a separate test set with bootstrap confidence intervals.

We also evaluate stress scenarios: dense overlapping dialogue, heavy acronym usage, and ambiguous delegation chains. These scenarios approximate practical enterprise set- tings where naive extraction models often fail.

TABLE I

PERFORMANCE METRICS ACROSS CORE COMPONENTS (SUGGESTEDTABLE 2)

Component	Precision	Recall	F1
Action Span Detection	0.92	0.90	0.91
Owner Assignment	0.91	0.89	0.90
Deadline Parsing	0.89	0.87	0.88
Priority Classification	0.88	0.85	0.86
End-to-End Task Object	0.90	0.88	0.89

B. Accuracy of Task Extraction

The action extraction stack achieves strong performance under noisy conversational conditions:

- Span detection F1: 0.91
- Full-field task extraction (all required fields correct): 0.84
- Owner attribution accuracy: 0.90
- Deadline normalization accuracy: 0.88

Errors are mainly caused by implicit commitments and multi-party delegated ownership statements.

A focused error taxonomy shows the following results: 31% of the errors caused by implicit commitments, 24% caused by reference resolution (“can someone from ops handle this?”), 19% caused by temporal ambiguity, 14% caused by jargon, and 12% caused by noisy turn boundaries.

This analysis was used to incorporate role priors and context from the meeting history, achieving measurable improvements in owner attribution.

C. Precision/Recall Metrics

The macro-averaged extraction metrics over the domains are provided in Table I.

In the cross-domain transfer test results on the data sets of software to consulting, consulting to research, the degradation of F1 remains within 0.05 with the help of the lightweight domain adaptation mechanism. Without the mechanism, the degradation is 0.11, which shows the importance of periodic domain calibration.

D. Productivity Score Validation

To validate the productivity model, three senior project managers rated 220 meetings on a five-level effectiveness scale based on outcomes and execution trace quality. Model score P_m correlates strongly with mean expert rating:

$$r(P_m, \text{expert}) = 0.81, p < 0.001. \quad (25)$$

As observed from the ablation study, the execution term Xm has the largest impact on predictive alignment when removed, as seen by a drop of 0.14 correlation.

We also examine the stability of rankings over three weeks. The Spearman rank correlation between automated rankings and expert rankings is 0.77, which implies that score traces reliably reproduce relative patterns of effectiveness across teams. Calibration plots reveal a small overestimation for newly formed teams with few historical completion records; however, hierarchical priors address this by shrinking early estimates to organizational baselines.

TABLE II

COMPARISON WITH EXISTING MEETING AI/PRODUCTIVITY SYSTEMS (SUGGESTED TABLE 1)

Capability	**Typical Trans-script Tools**	**Task As-sistants**	**Proposed System**
Structured action extraction with uncertainity	Partial	Moderate	Strong
Automatic calendar optimization	Rare	Limited	Native
Dynamic productivity trajectory	No	Limited	Yes
Cross-platform schema-preserving export	Limited	Platform Specific	Canonical CTO
Post-meeeting execution feedback loop	No	Partial	Full-loop

E .System Latency Analysis

Latency is measured from utterance arrival to UI surfacing of validated action items:

• Median: 0.94 s

• p90: 1.42 s

• p95: 1.80 s

• p99: 2.67 s

Latency to suggest a calendar suggestion - validated to ranked slot list - has a 620 ms average with warm cache and 1.1 s with cold cache. The export dispatch is asynchronous, taking typically 1.5 s or less to complete for three concurrent target platforms.

Inference accounts for 54% of median la- tency, network transit 21%, queueing 15%, and rendering 10% according to the profiling results.

Optimization opportunities include model distillation to support intent span tagging, as well as regional edge deployment to minimize round-trip times.

F. Comparative Assessment

Table II presents a brief summary of the qualitative limitations of the preceding system classes and the capabilities offered in this paper.

In addition to functional comparison, we also compare the maturity of dimensions such as explainability support, auditability, and rollback safety. Our system performs better since each generated artifact is associated with source evidence and deterministic transformation records, allowing root cause analysis in case of disputes over extracted commitments by the users.

It is important to note that the comparison is done at a system capability class level and not a product level.

VI. DISCUSSION

A. Scalability

The architecture has horizontal scaling through stateless inference gateways and partitioned event streams based on the meeting ID. In load testing, the throughput has a near-linear scale to 10,000 concurrent meetings with independently autoscaled diarization/extraction services. The most critical bottleneck is the external API rate limits, which we mitigate through adaptive backoff, prioritization of retry

queues, and eventual consistency. Shadow deployment is used to manage the model lifecycle, where we can test the new extraction model on live traffic without affecting the production output. Drift monitors measure the changes in the language of commitments across teams, which triggers active learning.

The computational complexity is mainly due to the transformer inference and slot optimization. For a given bounded window length W, the extraction complexity is O (T * W * d), where d is the hidden size. The scheduling complexity is O (N * log M), where N is the number of tasks and M is the slots, with heuristic candidate pruning.

The storage complexity is managed through a multi-tiered retention model, where we have hot event storage for recent meetings, warm aggregates for quarterly analytics, and cold archival storage. This architecture provides cost reduction without compromising trend analysis.

B. Ethical Considerations

The meeting analytics systems handle highly sensitive or- ganizational communications. Thus, we require participant con- sent, role-based access control, and data minimization. We only store task-relevant derived features for long-term analytics. The raw segments can then be controlled by retention windows.

The importance of transparency is also a key factor. We allow users to see why a task was created, which utterances contributed to a task, and confidence levels for all inferred fields. This is a key factor to prevent automation run amok.

Behavioral chilling is also an issue. This is because participants may change how they communicate if they think they are being surveilled. We address behavioral chilling by making process feedback available at a team level, allowing meeting-level opt-out, and controlling individual dashboards via governance policy. The organization should also publish acceptable use policies.

C. AI Bias Mitigation

The risk of bias is associated with owner attribution, interpretation of speaking balance, and inference of priority. The measures to mitigate the risk are:

- Counterfactual fairness tests for owner attribution on demographic proxies.
- Calibration checks to monitor subgroup performance to avoid over/under-estimation of confidence.
- Human-in-the-loop verification for low-confidence, high-impact tasks.
- Exclusion of protected attributes to avoid decision paths and logging.

We also avoid the punitive use of productivity scores at the individual level. The model is intended for team process improvement rather than surveillance or personnel rankings.

The process of continuous auditing of bias includes checking for temporal drifts since the fairness of the system may change over time with the composition of the team and the evolution of language. We suggest the following metrics to report on fairness on a quarterly basis: subgroup calibration,false assignment parity, and override rate disparity. For mitigation, the following methods can be employed: reweighting, annotation, and threshold adjustment based on the context of use.

From a governance perspective, automated actions with high business impact, such as escalating deadlines to the boards, should require confirmation when the confidence level of the model falls below a certain threshold.

D. Threats to Validity

Internal validity may be impacted by the presence of annotation noise in delegated ownership labels and the availability of historical completion logs, which differ based on organizational maturity. External validity may be impacted by the coverage of pilots, which are presently skewed toward knowledge work domains. Meeting norms in manufacturing and healthcare domains may require additional work. Construct validity of productivity remains a work in progress, as there is no universally agreed-upon ground truth. We mitigate this through multi-rater expert panels.

We believe the consistent results on extraction quality, scheduling feasibility, and score alignment suggest that integrated architectures have value beyond the traditional transcript-centric approaches.

VII. CONCLUSION AND FUTURE SCOPE

The paper proposed a holistic AI-based meeting analytics extension that transcends traditional transcript-based solutions to support full-cycle productivity orchestration. The solution bridges the

conversation-execution gap by integrating structured action extraction, schedule-aware calendar synchronization, productivity modeling, insight-driven visualization, and schema-agnostic export adapters.

The experimental results indicate good extraction accuracy, good alignment between productivity modeling and expert intuition, and fast interactive response times. The paper also makes a general argument that meeting intelligence is an operational systems challenge that combines NLP, optimization, human factors, and interoperability engineering. The future direction includes cross-lingual task ontology transfer for adaptation to multiple languages, modeling meeting interventions to predict long-term delivery outcomes, federated privacy-preserving training, and reinforcement learning-based proactive agenda management based on predicted productivity gain.

A further direction is proactive simulation. The system may, before the meeting takes place, estimate the expected productivity of the meeting based on the structure of the agenda, the amount of participants' load, and the historical decision velocity. A further direction is the multimodal integration of slide events, code review artifacts, and whiteboard snapshots to improve the grounding of commitments in technical meetings.

Ultimately, the long-term objective is not automation for its own sake but trustworthy augmentation of collaborative intelligence—ensuring that organizational conversations trans- late into timely, accountable, and measurable execution.

REFERENCES

[1] S. Renals, T. Hain, and H. Bourlard, “Recognition and understanding of meetings: The AMI and AMIDA projects,” in Proc. IEEE ASRU, 2007,pp. 238–247.

[2] A. Janin et al., “The ICSI meeting corpus,” in Proc. IEEE ICASSP, 2003, pp. 364–367.

[3] I. McCowan et al., “The AMI meeting corpus,” in Proc. 5th Int. Conf. Methods and Techniques in Behavioral Research, 2005, pp. 1–4.

[4] M. Zhong et al., “QMSum: A new benchmark for query-based multi- domain meeting summarization,” in Proc. NAACL-HLT, 2021, pp. 5905– 5921.

[5] B. Gliwa, I. Mochol, M. Biesek, and A. Wawer, “SAMSum corpus: A human-annotated dialogue dataset for abstractive summarization,” in Proc. EMNLP-IJCNLP Workshop, 2019, pp. 70–79.

[6] A. Vaswani et al., “Attention is all you need,” in Advances in Neural Information Processing Systems (NeurIPS), 2017, pp. 5998–6008.

[7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of deep bidirectional transformers for language understanding,” in Proc. NAACL-HLT, 2019, pp. 4171–4186.

[8] Y. Liu et al., “RoBERTa: A robustly optimized BERT pretraining approach,” arXiv:1907.11692, 2019.

[9] H. Bredin et al., “pyannote.audio: Neural building blocks for speaker diarization,” in Proc. IEEE ICASSP, 2020, pp. 7124–7128.

[10] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, “BERTScore: Evaluating text generation with BERT,” in Proc. ICLR, 2020.

[11] A. van den Oord, Y. Li, and O. Vinyals, “Representation learning with contrastive predictive coding,” arXiv:1807.03748, 2018.

[12 J. Brooke and G. M. Hirst, “Measuring and predicting discussion productivity in meetings,” IEEE Trans. Affective Computing, vol. 11, no. 4, pp. 718–732, 2020.

[13] M. M. Rahman, S. Cahyawijaya, and P. Fung, “Action item detection in meeting transcripts using transformer-based sequence labeling,” in Proc. COLING, 2022, pp. 4211–4222.

[14] X. Li, Z. Han, and M. Sun, “Graph-enhanced meeting understanding for decision and action extraction,” in Proc. ACL Findings, 2023, pp. 11234–11247.

[15] S. Boyd and L. Vandenberghe, Convex Optimization. Cambridge, U.K.: Cambridge Univ. Press, 2004.