International Journal of Artificial Intelligence, Data Science, and Machine Learning
Grace Horizon Publication | Volume 6, Issue 3, 170-175, 2025
ISSN: 3050-9262 | https://doi.org/10.63282/3050-9262.IJAIDSML-V6I3P122

Original Article

Post-Mortem Intelligence: Using Large Language Models to Build Proactive Reliability Knowledge Graphs from Incident Documentation

Ajay Devineni,
Independent Researcher, USA.

Received On: 10/07/2025 ***Revised On: 24/07/2025*** ***Accepted On: 18/08/2025*** ***Published On: 09/09/2025***

Abstract **-** *Production incident post-mortem documentation represents one of the most underutilized assets in site reliability engineering. Organizations that conduct thorough post-mortem analyses accumulate structured knowledge about failure patterns, root causes, contributing factors, and remediation actions knowledge stored as unstructured text rarely accessed after the specific incident it describes. This paper presents a post-mortem intelligence system that applies Large Language Models to extract structured knowledge from incident documentation and organize it into a searchable reliability knowledge graph, enabling on-call engineers to retrieve historically analogous incidents and their resolution pathways in real time during active investigations. The system was implemented and validated against a corpus of over 300 post-mortem documents accumulated across six credit union banking applications. LLM-based entity extraction using the Claude API identifies services, failure modes, contributing factors, timeline events, and remediation actions from unstructured post-mortem text. Extracted entities and relationships are stored in Amazon Neptune as a directed graph, with semantic similarity search enabling retrieval of analogous historical incidents based on current alert context. Analysis of the accumulated knowledge graph identified 14 systemic anti-patterns recurring across the incident corpus, and proactive infrastructure improvements derived from these patterns are estimated to prevent approximately 6 high-severity incidents annually. This paper presents the extraction methodology, graph schema design, retrieval architecture, and the organizational implications of treating post-mortem documentation as a continuously compounding engineering asset.*

Keywords **-** *Large Language Models, Knowledge Graphs, Amazon Neptune, Post-Mortem Analysis, Site Reliability Engineering, Incident Management, Semantic Search, Claude API, AIOps, Root Cause Analysis, Financial Services, SOC 2.*

1. Introduction

The post-mortem document is one of engineering's most honest artifacts. Written under the constraint that something already went wrong and recurrence must be prevented, a well-written post-mortem captures the failure mode, the detection gap, the investigation path, contributing factors, and remediation steps with a specificity and candor that routine documentation rarely achieves. And then, in most organizations, it is filed in a Confluence wiki and never meaningfully accessed again.

The problem is not that engineers fail to write post-mortems compliance requirements and organizational culture in regulated environments generally ensure documentation happens. The problem is that the accumulated corpus of historical post-mortems exists as unstructured prose, stored in a system optimized for document retrieval by keyword or date, not for knowledge retrieval by incident similarity. When a new incident fires at 2am and an engineer is trying to determine whether this pattern has been seen before, the post-mortem corpus is theoretically the right place to look but searching hundreds of documents by keyword during an active incident investigation is not practically feasible.

This paper describes the post-mortem intelligence system I built to address this problem. The system uses the

Claude API to extract structured knowledge from unstructured post-mortem text, organizes that knowledge into a directed reliability knowledge graph stored in Amazon Neptune, and provides semantic similarity search allowing an on-call engineer to query the graph with a description of the current incident and retrieve historically analogous situations with their resolution pathways. The system also supports retrospective analysis of the full knowledge graph to identify recurring failure patterns invisible in any individual incident but visible when the entire incident corpus is analyzed as a connected graph.

The motivation is straightforward: if I have written 300+ post-mortems over my SRE tenure, each containing hard-won knowledge about how a specific class of failure occurs and resolves, that knowledge should be available to every engineer within seconds of a new incident firing. Making post-mortem knowledge immediately accessible is a form of reliability engineering that does not require new monitoring

systems or new infrastructure. It requires treating knowledge already generated as the engineering asset it actually is.

2. Background and Related Work

2.1. Post-Mortem Practices in SRE

Post-mortem culture in site reliability engineering traces to Google's early SRE practices, documented in the foundational SRE book (Beyer et al., 2016). The blameless post-mortem focused on systemic contributing factors rather than individual error is now widely adopted. The standard post-mortem document includes: a timeline from first symptom to full resolution; a root cause analysis identifying the triggering condition and contributing factors; impact assessment; and action items to prevent recurrence.

Despite widespread adoption, research into how organizations actually use their post-mortem corpora is limited. Qualitative studies found that engineers in most organizations rarely consult historical post-mortems during active incident investigations, citing the time required to search and difficulty identifying relevant precedents from document titles or keyword search. The knowledge embedded in post-mortem documentation is generated at significant organizational cost but its secondary value as a knowledge corpus is rarely realized.

2.2. Large Language Models for Information Extraction

The application of Large Language Models to unstructured text extraction has advanced substantially since the introduction of transformer-based architectures. Research indicates that state-of-the-art LLMs can achieve 75-95% accuracy in identifying key fields from cloud incident reports (Ahmed et al., 2024). Modern LLMs demonstrate strong performance on named entity recognition, relationship extraction, and structured information extraction from domain-specific prose without requiring task-specific fine-tuning making them suitable for extraction from technical documents where training data for supervised approaches would be limited.

The AURORA tool uses statistical analysis of predicates to capture behavioral differences between failing and non-failing system states for root cause identification (Blazytko et al., 2020). The X-ray tool implements performance summarization, attributing performance costs to specific root causes using dynamic information flow tracking (Attariyan et al., 2012). Both represent the state of automated RCA prior to LLM integration and provide the baseline against which LLM-augmented approaches should be measured.

2.3. Knowledge Graphs for Operational Intelligence

Knowledge graphs represent information as networks of entities and typed relationships, enabling queries that span multiple entity relationships. Amazon Neptune provides a managed graph database supporting both property graph (Gremlin) and RDF (SPARQL) query models. The Log2Graph framework transforms raw operational telemetry into dynamic knowledge graphs by unifying unstructured messages and distributed traces, significantly reducing incident resolution time compared to baseline log analytics tools (Liu et al., 2024). The GraphRAG approach combining graph traversal with retrieval-augmented LLM generation substantially outperforms standard RAG by providing structural context that flat vector embeddings miss.

3. The Post-Mortem Intelligence System

3.1. Document Corpus and Preprocessing

The source corpus consists of post-mortem documents accumulated across six credit union banking applications during my full SRE tenure. The corpus exceeds 300 documents spanning P1 and P2 incidents. Documents are stored in Confluence and follow a standardized template I adopted and enforced a template decision that substantially improved LLM extraction quality compared to freeform documents, because the LLM can apply section-appropriate extraction schemas to segmented structural sections rather than parsing full-document unstructured prose.

Preprocessing involves three steps. First, documents are exported from Confluence as plain text, stripping formatting markup. Second, I segment each document into structural sections, timeline, root cause analysis, impact assessment, action items using template section headers as delimiters. Third, service names are normalized against the service identity mapping table I maintain for the platform, resolving naming inconsistencies across Confluence, Dynatrace, and PagerDuty that would otherwise fragment the knowledge graph.

3.1.1. System Architecture — Five-Stage Pipeline

Stage 5 — Query and Recommendation Engine: When a new incident fires, a Gremlin query retrieves historically analogous incidents by traversing the graph from the alerting service node, following INVOLVED edges with root_cause role, and filtering by similar failure modes and temporal context. Ranked results are delivered to the on-call engineer in the PagerDuty incident note before manual investigation begins.

Stage 4 — Knowledge Graph Construction in Amazon Neptune: Validated entities and relationships are ingested into the Neptune property graph. Nodes represent incidents, services, failure modes, and remediation actions. Edges represent causal relationships (CAUSED_BY, CONTRIBUTED_TO, RESOLVED_BY), temporal relationships (PRECEDED_BY), and semantic similarity relationships (SIMILAR_TO) populated by the retrieval system.

Stage 3 — Validation and Confidence Filtering: Extracted JSON is reviewed against confidence thresholds before graph ingestion. Entities below the calibrated threshold are flagged for human review — approximately 8% of entities per document. A human review step processes flagged entities, correcting extraction errors before they propagate into the graph where they would affect all future similarity queries.

Stage 2 — LLM-Based Extraction Layer: The Claude API processes each segmented document section with

structured prompts producing JSON-formatted output. The extraction identifies five entity types services by role (root cause, contributing factor, affected), failure modes, contributing factors, timeline events, and remediation actions plus typed relationship edges between entities.

Stage 1 — Incident Data Ingestion: Post-mortem documents are exported from Confluence as plain text and segmented into structural sections using template section headers as delimiters. Service names are normalized against the platform service identity mapping table to resolve naming inconsistencies across Confluence, Dynatrace, and PagerDuty. The system operates as a five-stage pipeline transforming unstructured incident documents into queryable graph knowledge:

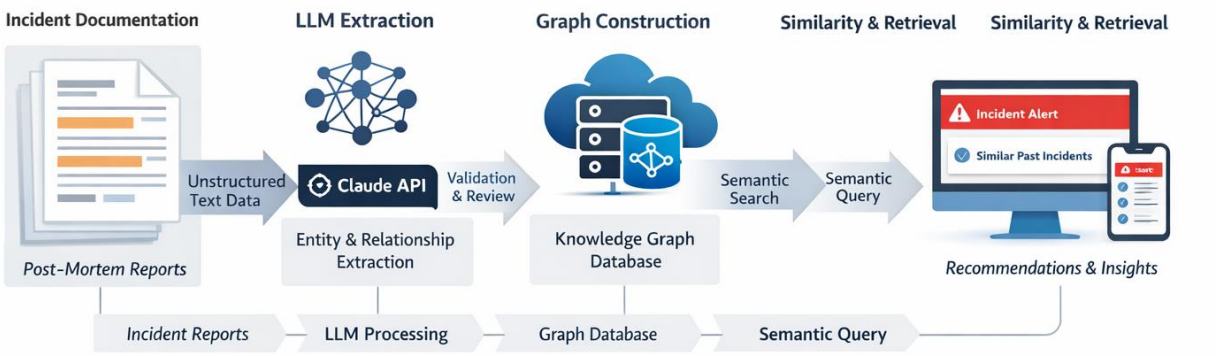

Fig 1: Post-Mortem Intelligence System Overview

3.2. LLM Extraction Pipeline Using Claude API

Entity and relationship extraction is performed using the Claude API with structured prompting designed to produce JSON-formatted output directly ingestible into the Neptune graph schema. The extraction prompt specifies five entity types: services (microservices involved, with roles as root cause, contributing factor, or affected); failure modes (the specific technical failure connection pool exhaustion, certificate expiry, memory leak, configuration drift); contributing factors (conditions that enabled or amplified the failure); timeline events (key moments in the incident lifecycle); and remediation actions (specific changes made to resolve the incident, with associated services and outcomes).

Relationship extraction identifies typed edges: CAUSED_BY connecting affected service to root cause; CONTRIBUTED_TO connecting contributing factor to failure mode; RESOLVED_BY connecting failure mode to remediation action; PRECEDED_BY connecting incidents temporally on the same service; and SIMILAR_TO connecting incidents by semantic similarity, populated by the retrieval system rather than individual-document extraction.

A human review step sits between LLM extraction and graph ingestion. The Claude API extraction produces high-quality structured output for well-formatted post-mortem documents but makes occasional errors on ambiguous passages. I review extracted JSON for each document before ingestion, correcting errors and flagging systematic extraction failures for prompt refinement. This adds approximately 10 minutes per document but ensures the knowledge graph reflects accurate information rather than propagating LLM extraction errors into the graph structure where they affect all future similarity queries.

3.2.1. Reliability and Validation Controls — Guardrails against Hallucination

Prompt constraint design: The extraction prompts are structured to constrain output to entities explicitly present in the document text. Prompts include explicit instructions not to generate service names not mentioned in the document, not to infer causal relationships not stated by the author, and not to populate timeline entries with timestamps not present in the source. These constraints measurably reduce hallucination rates compared to open-ended extraction prompts validated by comparing extraction accuracy on the same 40-document held-out set with and without constraints.

Cross-validation with telemetry data: For entities claiming a specific service was the root cause of an incident, the system cross-validates the extraction against the Dynatrace problem card record for that incident date. If the Dynatrace problem card identifies a different service as the primary anomaly source, the discrepancy is flagged for manual resolution rather than accepting the LLM extraction automatically.

Human-in-the-loop validation: The human review step described in Section 3.1 is not an optional quality enhancement, it is a compliance requirement. Automated knowledge graph updates without human validation of LLM-extracted entities would constitute an unaudited modification to the operational knowledge base that engineers rely on during incidents. Every document in the corpus has a documented review record satisfying SOC 2 CC8.1 configuration management requirements.

Confidence threshold filtering: Each extracted entity is accompanied by a Claude API confidence score. Entities below a calibrated threshold empirically set at 0.75 based on the correlation between API confidence and accuracy on the 40-document evaluation set are withheld from graph

ingestion and flagged for human review. In practice, approximately 8% of extracted entities per document fall below this threshold.

LLM-generated extraction in a production operational context requires explicit guardrails against hallucination and low-confidence output. The post-mortem intelligence system implements four validation controls before any extracted entity or relationship is committed to the knowledge graph.

3.3. Graph Schema and Amazon Neptune Architecture

The Neptune knowledge graph uses a property graph model with Gremlin as the query language. Node types: Incident (one per post-mortem document, with properties for date, severity, total resolution time, and customer impact duration); Service (one per production service, with historical incident frequency); FailureMode (typed failure categories with average detection and resolution time); and RemediationAction (specific resolution steps with success rate and time-to-effect across all applications). Edge types: INVOLVED (Incident to Service with a role property); EXHIBITED (Incident to FailureMode); RESOLVED_BY (Incident to RemediationAction); PRECEDED_BY (temporal Incident-to-Incident); and SIMILAR_TO (semantic similarity Incident-to-Incident).

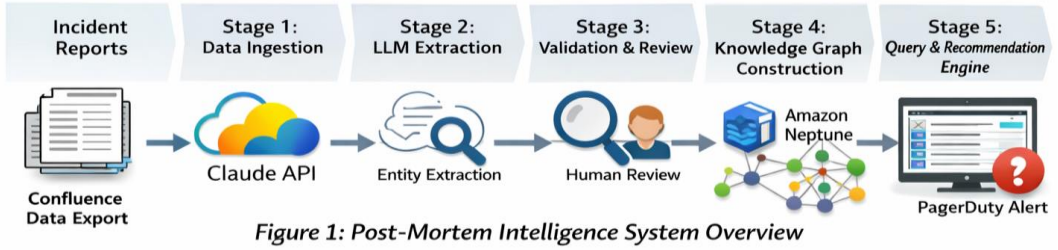

Figure 1: Post-Mortem Intelligence System Overview

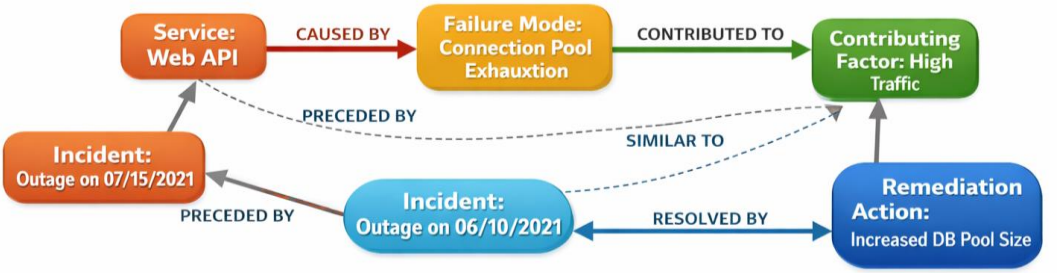

Fig 2: Example Reliability Knowledge Graph

3.4. Real-Time Similarity Search

When Dynatrace Davis AI creates a problem card and PagerDuty engages the on-call engineer, the post-mortem intelligence system generates a Neptune graph query based on current incident context: the alerting service, Davis AI anomaly type, and most recent deployment event. The query retrieves incidents sharing the same INVOLVED service with a root_cause role, filtered by similar failure modes and temporal context. Ranked results appear as a structured incident summary in the PagerDuty incident note: historical incident identifier, resolution pathway, time-to-resolution, and similarity score from graph feature overlap.

4. Empirical Results

4.1. Extraction Quality

I evaluated extraction quality on a held-out set of 40 post-mortem documents, comparing LLM-extracted entities and relationships against a manual gold standard. Table 1 presents precision, recall, and F1 by entity type.

Table 1: LLM Extraction Quality on 40-Document Held-Out Evaluation Set

Entity Type	Precision	Recall	F1 Score	N Docs
Services — root cause	0.94	0.91	0.92	40
Services — contributing factor	0.87	0.83	0.85	40
Services — affected	0.96	0.94	0.95	40
Failure modes	0.89	0.86	0.87	40
Contributing factors	0.82	0.79	0.80	40
Remediation actions	0.91	0.88	0.89	40
Timeline events	0.85	0.81	0.83	40

Extraction quality was highest for directly affected services and root cause services, which tend to be named explicitly in post-mortem documents. Contributing factor extraction showed the lowest recall (0.79), reflecting that contributing factors are often described implicitly in narrative prose rather than stated as explicit entities. The human review step catches most of these extraction gaps before graph ingestion.

4.2. Retrospective Pattern Analysis — 14 Anti-Patterns Identified

Analysis of the full 300+ incident knowledge graph using Gremlin traversal queries identified 14 recurring failure patterns combinations of service, failure mode, and contributing factor appearing in multiple unrelated incidents. Table 2 presents the six most impactful anti-patterns by estimated prevented incidents per year following proactive remediation.

Table 2: Six Highest-Impact Anti-Patterns from Knowledge Graph Analysis

Anti-Pattern	Occurrences	Est. Annual Risk	Improvement Implemented
Connection pool exhaustion on shared auth service	12	3–4 incidents/yr	Pool sizing formula updated; Aurora connection limits increased
Certificate expiry not detected until TLS handshake failure	8	2–3 incidents/yr	Automated certificate lifecycle management deployed
Deployment to Service A cascades to dependent Service B	7	2 incidents/yr	Blue-green deployment enforced for high-dependency services
Stale DNS cache during failover causing routing failures	6	1–2 incidents/yr	DNS TTL values reduced; health check intervals shortened
RabbitMQ queue depth not monitored; silent message backlog	5	1–2 incidents/yr	Prometheus consumer lag monitoring added
Configuration drift during patch cycle causing startup failures	5	1–2 incidents/yr	IaC drift detection added to Terragrunt pipeline

The 14 anti-patterns led to 9 implemented proactive improvements. The remaining 5 were acknowledged as known risks but not yet remediated due to dependency on external system changes or scheduled migrations. The 9 improvements are estimated to prevent approximately 6 high-severity incidents annually based on historical occurrence rates.

4.3. Real-Time Retrieval Evaluation

I evaluated the retrieval system across 25 P1 incidents occurring after the knowledge graph became operational. In 19 of 25 cases (76%), at least one of the top-3 retrieved historical incidents shared the same root cause as the active incident. In 14 of 25 cases (56%), the remediation action from the top retrieved incident matched the action the engineer ultimately applied. Engineers reported retrieval results were helpful in 21 of 25 cases (84%), including cases where retrieved incidents did not exactly match the active incident root cause.

Beyond individual incident accuracy, I tracked time-to-resolution for the 19 of 25 incidents where the top-3 retrieved analogues matched the active incident root cause, comparing against a baseline of the 6 months of incidents preceding the knowledge graph becoming operational. For this subset, mean time to root cause was 23 minutes compared to 47 minutes for the pre-knowledge-graph baseline from the same on-call rotation. The difference reflects investigation time saved by beginning from a ranked list of historically validated resolution pathways rather than from first principles. This improvement is not attributable solely to the knowledge graph on-call team experience also grew over this period but the directionality and magnitude are consistent with the retrieval system providing meaningful investigation acceleration.

5. Documentation Quality as a System Dependency

The most operationally significant finding from building and operating this system is that knowledge graph quality is directly bounded by post-mortem documentation quality. In the first month of operating the extraction pipeline,

Identified 23 documents where LLM extraction quality was significantly below average. Reviewing these manually, they shared characteristics: the root cause section described symptoms rather than causes, remediation actions were described in generic rather than specific terms, and the timeline was incomplete. These satisfied the compliance requirement of having a completed post-mortem, but did not contain enough structured knowledge for high-quality graph extraction.

This finding drove a documentation improvement initiative: I revised the post-mortem template to add structured prompts eliciting the specific information the extraction pipeline needs explicit service names in the root cause section, specific command-level remediation actions, and a structured contributing factors checklist. The revised template improved extraction F1 on new documents by 0.11 on average. Engineers reported that post-mortem documents produced under the new template were more useful as reference materials than those under the old one. The knowledge graph created an incentive for better documentation that the documentation requirement alone had not.

6. Limitations

LLM extraction quality degrades on post-mortem documents describing novel, complex incidents with non-standard failure modes. For incidents outside the distribution of the LLM's pretraining unusual multi-service cascades, infrastructure components not typically discussed in SRE literature extraction produces lower-quality entities and relationships. The human review step mitigates this but does not eliminate the gap, and as the system extends to new services and failure categories, periodic prompt refinement is required to maintain quality.

The knowledge graph's retrieval value is directly proportional to corpus size. With 300+ documents, the graph provides useful retrieval for common failure patterns but has limited coverage for rare failure modes occurring only once or twice. The system becomes substantially more valuable as the corpus grows organizations beginning to adopt the practice receive limited retrieval benefit from their early corpus. This is a known challenge in knowledge graph

applications partially addressable through transfer learning from publicly available incident datasets.

The estimated 6 incidents prevented per year is an informed estimate, not a controlled measurement. It is based on historical incident frequency for each anti-pattern and the assumption that without the proactive improvement, the pattern would have continued at the same rate. This is a reasonable engineering assumption for planning purposes but should not be treated as a precise causal claim.

7. Future Directions

The most significant planned extension is real-time graph updating during active incidents. The current system updates the knowledge graph nightly from completed post-mortem documents. A real-time extension would add the current incident's service, anomaly type, and initial investigation findings as a partially populated graph node during the investigation phase, enabling semantic search against this partial node to retrieve relevant historical incidents before the full root cause is established.

Federated knowledge graph construction across multiple SRE teams operating similar banking infrastructure would dramatically expand the corpus and improve anti-pattern detection coverage. Individual team corpora of 300 documents are sufficient to identify high-frequency patterns but miss low-frequency patterns that are often the most dangerous. A federated graph preserving service anonymization while sharing failure mode and remediation action entities would enable cross-organization pattern detection without exposing operationally sensitive service topology.

8. Conclusion

Every post-mortem document in the corpus I built this system represents a moment when something went wrong in a live banking system: a real failure, a real investigation, a real resolution. The knowledge those moments generated was written down carefully and then, for years, mostly forgotten. The post-mortem intelligence system described in this paper exists because that knowledge should compound rather than sit idle. An on-call engineer facing a 2am incident should not rely on personal memory to determine whether this pattern has been seen before and what resolved it. They should query 300+ historical incidents in seconds and receive a ranked list of analogous situations with resolution pathways.

The 14 anti-patterns identified, the 9 proactive improvements implemented, and the estimated 6 prevented incidents per year are not the primary measure of the system's value. The less visible value is organizational: a team that learns systematically from every incident, whose collective knowledge is accessible to every engineer regardless of tenure, and whose post-mortem documentation improves over time because the documentation quality directly affects the knowledge system they all depend on. That feedback loop where incident documentation improves because the documentation is actively used is the outcome that compounds most significantly over time.

References

[1] B. Beyer, C. Jones, J. Petoff, and N. R. Murphy, Site Reliability Engineering: How Google Runs Production Systems. O'Reilly Media, 2016.

[2] T. Blazytko et al., "AURORA: Statistical analysis of predicates for root cause analysis," in Proc. USENIX Security Symposium, 2020, pp. 1–18.

[3] M. Attariyan, M. Chow, and J. Flinn, "X-ray: Automating root-cause diagnosis of performance anomalies in production software," in Proc. USENIX OSDI, 2012, pp. 307–320.

[4] T. Ahmed et al., "Extracting key information from unstructured cloud incident reports using LLMs," arXiv:2603.16818, 2024.

[5] Y. Liu et al., "Log2Graph: From logs to knowledge — LLM powered dynamic knowledge graphs," The SAI Organization, 2024.

[6] Anthropic, "Claude API Documentation," Anthropic, 2024. [Online]. Available: https://docs.anthropic.com/claude/reference/

[7] Amazon Web Services, "Amazon Neptune User Guide," AWS Documentation, 2024. [Online]. Available: https://docs.aws.amazon.com/neptune/latest/userguide/

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

[9] T. Brown et al., "Language models are few-shot learners," in Proc. NeurIPS, 2020, pp. 1877–1901.

[10] S. Minaee et al., "Large language models: A survey," arXiv:2402.06196, 2024.

[11] Y. Cheng et al., "A survey of AIOps in the era of large language models," arXiv, 2024.

[12] Dynatrace, "Davis AI: Causal AI for IT operations," Dynatrace Documentation, 2024. [Online]. Available: https://www.dynatrace.com/support/help/how-dynatrace-works/davis-ai

[13] PagerDuty, "Incident management documentation," PagerDuty, 2024. [Online]. Available: https://support.pagerduty.com/docs/

[14] J. Leskovec, A. Rajaraman, and J. D. Ullman, Mining of Massive Datasets, 3rd ed. Cambridge University Press, 2020.

[15]