# Intelligent Log Analysis Using a Hybrid ELK, GenAI, and Incident Learning Approach

# Abstract

This document presents an intelligent log analysis framework that synergistically integrates Large Language Models (LLMs) with dynamically updated Knowledge Graphs (KGs) for real-time log understanding and IT incident management. The approach addresses current limitations in log analysis by enabling automated incident summarization, root cause analysis, and semantic querying over log data using a unified LLM-driven agent. The LLM translates natural language queries into graph queries and interacts with a continually enriched knowledge graph built from live logs and IT service data. A secure reasoning loop with validation layers ensures robust operation. The novelty lies in combining the contextual reasoning power of LLMs with the structured insight of knowledge graphs, yielding a system that can interpret complex log streams in real-time, automatically correlate events with historical incidents, and provide human-friendly summaries of system issues. This paper outlines the problem context, the proposed solution’s architecture, its novel features, and a comparative analysis against prior art in intelligent log analytics.

# Introduction

Modern IT systems generate massive volumes of log data that record events, errors, and performance metrics. These logs are a valuable source of information for diagnosing incidents and understanding system behaviour. However, current log analysis practices struggle to keep pace with this data deluge. Engineers often rely on static dashboards and manual queries to sift through unstructured log messages, a process that is time-consuming and prone to missing subtle correlations. Traditional tools lack semantic understanding – they treat logs as text strings rather than as carriers of meaning – and hence cannot summarize incidents or explain root causes in human terms. Furthermore, existing monitoring solutions are usually siloed from historical incident knowledge; context from past outages or fixes is not automatically brought to bear on new issues. This disconnect means troubleshooting often starts from scratch, even if similar problems have occurred before.

Recent advances in AI offer an opportunity to revolutionize log analysis. Large Language Models (LLMs) have demonstrated an ability to understand and generate human-like text, including the domain-specific jargon found in system logs. Meanwhile, knowledge graphs excel at capturing relationships between entities (servers, services, errors, configuration changes, etc.) in a structured form. By integrating an LLM with a live-updating knowledge graph, we propose a system that **understands** log data in context. The LLM can interpret logs, translate free-form questions into structured queries, and reason about the system state, while the knowledge graph provides a memory of entities, events, and their relations drawn both from streaming logs and IT Service Management (ITSM) records. In essence, the introduction of an LLMdriven agent supervising a knowledge graph enables a real-time, explainable, and context-aware analysis of logs. The following sections detail the problem space, the novel approach we introduce, and how it advances the state of the art in intelligent log analytics.

# Problem Statement

Despite numerous tools for log management, current practices exhibit several shortcomings that impede efficient incident response:

* **Over-Reliance on Dashboards and Keywords:** Engineers often monitor systems via predefined dashboards or search for error keywords manually. This approach fails when issues manifest as complex patterns across disparate logs rather than obvious error strings. Static dashboards cannot adapt to novel failure modes, and important clues may be overlooked if one does not use the exact right search terms.
* **Lack of Semantic Understanding:** Traditional log analysis treats log entries as raw text or simple key-value pairs. There is little understanding of the *meaning* behind a log message. For example, a message “Node X lost heartbeat with Node Y” might indicate a network partition – a human can infer this, but a typical tool does not. The absence of natural language comprehension means logs aren’t summarized or contextualized; the onus is on the operator to interpret cryptic messages and connect the dots.
* **Disconnection from Historical Context:** When an incident occurs, engineers must manually recall or lookup past incidents that resemble the current one. Present tools do not automatically link ongoing events to historical incident reports or root causes. Valuable lessons learned in the past (e.g. a certain error preceded a database crash last month) remain locked in archives. This lack of memory leads to repetitive investigative effort and sometimes reinvention of fixes.
* **Siloed Data Sources:** Logs often exist separately from other relevant data like configuration changes, user reports, or monitoring alerts. With data spread across systems, engineers struggle to piece together a coherent timeline. Current log analytics platforms rarely integrate well with ITSM ticket data or network topology information, missing higher-level correlations (for instance, linking a spike in error logs to a recently deployed software update).
* **Manual and Reactive Workflow:** Because tools lack automation in understanding logs, incident response is largely reactive. Alerts might fire on simple thresholds, but writing these rules requires foresight and maintenance. There is no automated “reasoner” that looks at all incoming data and proactively identifies likely root causes or synthesizes a summary for human consumption. The burden of analysis remains on humans operating under stress, which is error-prone and slow.

In summary, the problem is that **log analysis today is low-level and disconnected** – it surfaces raw data without insight, demands expertise to interpret, and fails to leverage past knowledge. This limits an organization’s ability to respond quickly to incidents and learn from them. The following section introduces our approach that addresses these issues by blending LLMs and knowledge graphs for a more intelligent solution.

# Novelty of the Proposed Approach

Our proposed system introduces a novel end-to-end pipeline that combines LLM capabilities with knowledge graph reasoning to overcome the above limitations. The key innovative aspects are:

* **Natural Language to Graph Query Translation:** The system employs an LLM as an interpreter between human language and machine queries. Users can ask questions about the logs in natural language (e.g. “Why did the payment service crash yesterday around 3 PM?”), and the LLM component translates this into precise graph queries or search instructions. The LLM understands intent and context, then formulates queries (such as Cypher or SQL for the knowledge graph or time-bounded searches in log indices) that retrieve relevant data. This translation allows **semantic querying** over log data – users are no longer limited by rigid query languages or exact keyword matches.
* **Dynamic Knowledge Graph Enrichment via Plugins:** At the core of the system is a continually updated knowledge graph that represents the IT environment and its state. What’s novel is the way this graph is **enriched in real-time from multiple sources** through specialized plugins. One plugin processes streaming logs to extract structured information (for example, recognizing entities like server names, application components, error codes, and relationships like “causes” or temporal sequences from log messages). Another plugin interfaces with ITSM systems to integrate incident tickets, change records, and alerts into the graph. The knowledge graph thus grows to encapsulate both the live operational data and historical incident knowledge. Unlike static CMDBs (Configuration Management Databases), this graph is *automatically* built and updated, providing a rich context for analysis.
* **Unified LLM Agent for Reasoning Loop:** The LLM is not a passive translator; it acts as an **agent** that orchestrates the reasoning process end-to-end. When a query or an alert comes in, the LLM agent can perform a loop of actions: it may query the knowledge graph for relevant nodes (e.g. recent errors on Service A), analyze the results, possibly ask a follow-up question or refine the query, and even update the knowledge graph with new findings. This resembles an autonomous troubleshooting assistant. The novelty here is in using a single LLM-driven agent to tie together all steps – from interpretation of the problem, retrieval of data, correlation of events, to generating a final explanation. It leverages chain-of-thought reasoning, guiding itself through sub-tasks (much like how a human expert would drill down into logs, check metrics, recall similar past incidents, etc.). By embedding this logic in an LLM, we gain flexibility: the agent can adapt its strategy to different scenarios without explicit reprogramming.
* **Security and Validation Layers:** Integrating an LLM into operational infrastructure raises concerns – for example, ensuring that the LLM’s generated queries are safe and that it cannot be prompted to perform unauthorized actions. Our system is novel in building a **safety harness** around the LLM agent. All inputs (queries, log contents) are sanitized to remove any potentially malicious content (such as injection attacks in log text). The LLM’s actions are constrained via a policy engine – certain high-risk operations (like deleting data) are simply not allowed, and all graph queries run through a validator that checks for efficiency and safety (no full graph scans that could slow the system, for instance). Additionally, the responses of the LLM agent can be audited: the system keeps a trace of what data was retrieved and how the conclusion was formed, providing an explanation for its answers. These layers ensure that the powerful reasoning of the LLM is harnessed in a controlled and robust manner suitable for enterprise environments.

Collectively, these novel features produce an intelligent log analysis system that is **context-aware, conversational, and proactive**. It understands logs not just as text but as events in a connected story, enriched by wider knowledge. It can answer complex questions and justify answers by referencing structured data. And it continuously learns: every incident handled makes the knowledge graph richer, enabling faster root cause identification for future issues. The next section will detail representative prior art to highlight how our approach differs from and advances beyond existing solutions.

# Prior Art

To contextualize the advancement of our proposed system, we surveyed existing patents related to intelligent log analysis, semantic parsing of logs, knowledge graph applications in IT operations, and AIassisted root cause analysis. Ten highly relevant prior art examples are summarized below, ranked in descending order of relevance to our approach (1 = most relevant). Each prior art is listed with its patent number, title, core approach, and noted limitations relative to our proposed system.

1. **US20220027331A1 – Cross-Environment Event Correlation Using Domain-Space Exploration and Machine Learning Techniques:** This patent application discloses an AI-driven system for correlating events across different IT domains (e.g., network, application, container) to diagnose issues. It builds a knowledge graph of events by extracting a “knowledge base” through dependency parsing and other information extraction, then performs clustering to find patterns or commonalities among incidents [1](https://patents.google.com/patent/US20220027331A1/en#:~:text=nodes%20that%20connect%20across%20the,correlate%20common%20issues%20and%2For%20issues) [2](https://patents.google.com/patent/US20220027331A1/en#:~:text=%2A%20the%20domain,sets%2C%20by%20using%20an%20information) . The system can link related events (such as network flaps and application drops) into a unified representation and uses iterative learning with SRE (Site Reliability Engineer) feedback to improve correlation logic. **Limitation:** While it constructs a knowledge graph for correlation, it does not leverage an LLM for natural language queries or summarization. The correlation and diagnosis are based on predefined algorithms and domain-specific exploration, lacking a conversational interface or the dynamic reasoning loop of an LLM agent.
2. **CN117453501A – Log Analysis Method, Device, Equipment and Storage Medium:** This Chinese patent publication proposes an automated log parsing method that heavily uses knowledge graph techniques for log analysis. It constructs a **log knowledge graph** by performing entity recognition and relationship extraction on raw log texts [3](https://patents.google.com/patent/CN117453501A/en#:~:text=In%20this%20embodiment%2C%20the%20log,speech%20tagging%2C%20named) . When an IT work order (incident ticket) is closed, the system automatically retrieves related log data and classifies the logs by type, then applies a corresponding log analysis template to parse them into structured results . The knowledge graph is continuously adjusted with new structured log data, and the parsed results are fed back to retrain a log analysis model and update an entity dictionary [5](https://patents.google.com/patent/CN117453501A/en#:~:text=analysis%20result%20of%20the%20sample%3B,and%20updating%20the%20entity%20dictionary) . This approach essentially creates a pipeline where unstructured logs structured representation knowledge graph improved log classification. **Limitation:** The system relies on pre-defined log analysis templates and a text classification model for each log type [7](https://patents.google.com/patent/CN117453501A/en#:~:text=In%20this%20embodiment%2C%20the%20log,log%20type%20is%20one%20of) . It does not incorporate any LLM or advanced natural language understanding beyond domain-specific NLP. Queries or analysis are not handled in natural language, and the insights are limited by the fixed templates and rules defined for each log category. In contrast, our approach uses an LLM to dynamically interpret logs and formulate queries, allowing more flexible, ad-hoc analysis without needing a template for every log format.

[4](https://patents.google.com/patent/CN117453501A/en#:~:text=In%20the%20embodiment%20of%20the,the%20target%20log%20original%20text)

[6](https://patents.google.com/patent/CN117453501A/en#:~:text=graph%2C%20the%20log%20knowledge%20graph,concentrated%2C%20the%20training%20effect%20of)

1. **US20230050889A1 – Method and System to Generate Knowledge Graph and Sub-Graph Clusters to Perform Root Cause Analysis:** This patent application (published in 2023) describes a system for root cause analysis (RCA) that automatically builds a knowledge graph from various data and then learns graph patterns to identify causes. The method involves **extracting entities and relationships from input content** (which could include logs, metrics, etc.), constructing a knowledge graph, and then using unsupervised machine learning to generate sub-graphs and clusters [8](https://patents.justia.com/patent/20230050889#:~:text=sub,Generated%20Knowledge%20graph) . From these clusters, a probabilistic graphical model (a “root cause model”) is derived to determine the root cause of an issue based on new input (issue content). Essentially, it creates an automated RCA engine that doesn’t rely on manually defined rules by mining relationships in the data. **Limitation:** The approach, while using knowledge graphs, appears to focus on offline model building – generating a static root cause model from historical data. It doesn’t mention real-time log streaming or interactive querying. There is no use of an LLM or natural language interface; the system’s insights are delivered via the probabilistic model. Additionally, the method’s effectiveness depends on the quality of unsupervised clustering; it may require substantial historical data and could struggle with entirely new patterns. Our proposed system differs by operating in real-time with an interactive LLM agent and continuously updating its knowledge graph, rather than relying purely on upfront model training.
2. **CN113282764B – Method and Device for Constructing Network Security Data Knowledge Graph:** This patent (granted in China) targets the construction of a knowledge graph for network security analysis, which includes processing of log data. The method aggregates multiple sources: it fuses dictionary-based knowledge triples, **log-derived knowledge triples, and text-derived knowledge triples** into a layered knowledge graph [9](https://patents.google.com/patent/CN113282764B/en#:~:text=match%20at%20L275%20%E5%9C%A8%E6%89%80%E8%BF%B0%E5%9F%BA%E7%A1%80%E7%9F%A5%E8%AF%86%E5%9B%BE%E8%B0%B1%E4%B8%8A%E8%9E%8D%E5%90%88%E6%89%80%E8%BF%B0%E6%97%A5%E5%BF%97%E7%9F%A5%E8%AF%86%E4%B8%89%E5%85%83%E7%BB%84%EF%BC%8C%E5%BE%97%E5%88%B0%E6%93%8D%E4%BD%9C%E5%85%B3%E7%B3%BB%E5%9B%BE%E8%B0%B1%EF%BC%9BFusing%20the,obtain%20an%20operational%20relationship%20graph) . For example, it might take structured security alerts (dictionary knowledge), combine them with parsed information from security logs (log triples), and integrate unstructured text like incident reports (text triples) to build a comprehensive graph of security events. The graph is then mapped to a vector space for downstream analysis like detecting threats. **Limitation:** This approach is domain-specific to cybersecurity and focuses on knowledge graph construction rather than user interaction. It does not utilize an LLM; the analysis is oriented toward detecting patterns or anomalies in the fused graph data using techniques like vector embeddings. While it automates knowledge extraction from logs and text (similar in spirit to our log enrichment plugin), it doesn’t describe an interactive agent that can reason about the graph in natural language. Also, its primary goal is to improve threat detection, whereas our system is aimed at a broader IT operations context (incidents, performance issues, etc.) with interactive summarization and querying.
3. **US11275791B2 – Automatic Construction and Organization of Knowledge Graphs for Problem Diagnoses:** This U.S. patent (granted 2022) by IBM describes a system for helping root cause analysis by automatically building navigable knowledge graphs from technical reports. In one embodiment, the system takes a collection of troubleshooting reports or data plots and **constructs a knowledge graph where each node represents an individual report/plot, and edges are added between nodes that share common variables or metrics** [10](https://patents.google.com/patent/US11275791/en#:~:text=extracting%20a%20plurality%20of%20variables,an%20edge%20weight%20to%20each) . The edges can be weighted by the strength of commonality. The graph is then organized (e.g., via hierarchical clustering or connected components) and displayed to the user as a “navigation graph” for the reports [11](https://patents.google.com/patent/US11275791/en#:~:text=knowledge%20graph%20103%20from%20the,if%20their%20corresponding%20reports%20have) . This helps analysts see relationships between different problem reports and potentially identify underlying factors. **Limitation:** The system focuses on organizing documents/plots for user navigation and doesn’t extend to parsing raw log streams. It does not employ any LLM; the interface is graphical visualization rather than a Q&A or summarization. Essentially, it’s a static analysis tool to cluster related diagnostic reports. Our approach would complement this with active reasoning: instead of just visual navigation, our LLM agent could answer questions using the knowledge graph and logs. Also, this prior art deals with structured data (plots with variables), whereas our system tackles unstructured logs directly and converts them into a knowledge graph.
4. **US11336507B2 – Anomaly Detection and Filtering Based on System Logs:** This patent (granted 2022) presents a machine learning approach to identify significant anomalies in log data and filter out noise. The method uses a sequential model (specifically LSTM with attention, as described in the documentation) to learn normal log patterns and predict future log sequences [12](https://patents.google.com/patent/US11336507B2/en#:~:text=FIG,embodiments%20of%20the%20present%20disclosure) . It works by **comparing real-time logs to predicted logs**: a first sequence of log entries is used to predict what the next sequence should look like, and if the actual next sequence deviates, it’s flagged as anomalous [13](https://patents.google.com/patent/US11336507B2/en#:~:text=One%20embodiment%20presented%20in%20this,is%20noteworthy%20based%20on%20a) . To reduce false positives, it calculates a “function entropy” of the prior log sequence and a sentiment polarity of the new sequence – basically measuring how unusual and how negative the new logs appear. Only if the sequence is both anomalous and “noteworthy” (e.g., containing error sentiments) does it generate an **anomaly report**, which includes the offending log sequence and an inferred root cause [14](https://patents.google.com/patent/US11336507B2/en#:~:text=sequence%20of%20log%20entries%20from,sequence%20and%20a%20root%20cause) . **Limitation:** This approach is a specialized pipeline for anomaly detection, primarily leveraging deep learning on log sequences plus a bit of sentiment analysis on log text. It does not incorporate a knowledge graph or cross-reference other data sources; each anomaly is detected in isolation based on statistical deviation. While it can output a root cause, that is likely limited to what the model learned (for example, pointing to the log message that didn’t match the prediction). There is no interactive query capability or use of an LLM – the logic is fixed and not easily extensible to answering arbitrary questions about the logs. Our system, on the other hand, aims for a broader understanding of system state and can handle questions or summarizations beyond just anomaly reports.
5. **US8495429B2 – Log Message Anomaly Detection:** This is an earlier (2013) Microsoft patent that introduced a more explainable way to detect anomalies in logs by mining invariants. The technique involves **parsing unstructured log messages into a structured form** consisting of a message template (static text signature) and parameters [15](https://patents.google.com/patent/US8495429B2/en#:~:text=One%20or%20more%20techniques%20and%2For,sequences%20of%20respective%20log%20types) . Logs of the same type (same template) are grouped, and the system automatically infers invariants – consistent relationships or values – within each group. For example, for a log template “Server *X* connected to Database *Y* in *T* seconds,” an invariant might be that *T* is usually below 5. If an invariant is violated (say *T* = 30 for a particular entry), that log is flagged as an anomaly. By applying these learned invariants to new log sequences, the system can catch unusual behaviors. **Limitation:** The method provides more interpretable results than black-box ML (since invariants have clear meanings) [16](https://patents.google.com/patent/US8495429B2/en#:~:text=Current%20and%20previous%20automatic%20detection,may%20need%20extra%20efforts%20to) [17](https://patents.google.com/patent/US8495429B2/en#:~:text=Some%20current%20statistic%20learning%20based,not%20provide%20intuitive%20and%20meaningful) , but it still lacks integration of external knowledge or context. It operates on each log type independently and doesn’t perform higher-level correlation between different log streams or past incidents. There is no knowledge graph or global reasoning – it’s essentially pattern learning per log template. Additionally, being an older approach, it doesn’t use modern NLP or LLMs; it can’t, for example, summarize what a cluster of anomalies mean in human language. Our approach would use some similar parsing (extracting structure from logs), but then connect those pieces in a knowledge graph and use an LLM to reason across templates and systems, providing a more comprehensive incident narrative.
6. **US20150094959A1 – Heterogeneous Log Analysis:** This patent application (2015) addresses the challenge of analyzing logs from diverse sources without prior knowledge of their formats. It proposes a pipeline with multiple components: a **hierarchical log clusterer** to group similar log messages, a **log pattern recognizer** to identify common patterns across clusters, a **log field analyzer** to infer the semantic meaning of parts of log messages, and a **log indexer** to index logs based on these patterns and fields [18](https://patents.google.com/patent/US20150094959A1/en#:~:text=,for%20heterogeneous%20log%20clustering%2C%20in) [19](https://patents.google.com/patent/US20150094959A1/en#:~:text=200%20as%20an%20input%20to,This) . The system automatically tokenizes log lines (even without predefined delimiters) and uses a custom similarity measure that takes into account log message layout and content to perform clustering. The outcome is a structured representation of logs and their variants (often referred to as “log templates” in research literature) for easier search and analysis. **Limitation:** This approach is unsupervised and focuses on structure discovery in logs. While useful for parsing and indexing, it doesn’t incorporate any domain knowledge or historical incident context – it treats the log corpus itself as the sole source of truth. There’s no notion of a knowledge graph linking logs to higher-level concepts like servers or transactions, and no summarization capability. Any root cause analysis still has to be done by a human interpreting the clusters and patterns. Moreover, it doesn’t use an LLM or any kind of advanced reasoning over the content; it’s essentially clustering and regex pattern generation. Our proposed system could actually leverage such a parsing technique as a preprocessing step, but we move beyond by integrating the parsed data into a knowledge graph and using an LLM to derive insights from it.
7. **WO2022155376A1 – Root Cause Analysis Tool for Alarms:** This international patent application (publication year 2022) focuses on handling “alarm floods” in industrial or IT monitoring systems. It introduces a method to automatically perform root cause analysis on a sequence of alarm events. The core idea is to build a **correlation graph** where each alarm is a node and edges connect alarms that frequently occur together or in sequence, with weights representing correlation scores [20](https://patents.google.com/patent/WO2022155376A1/en#:~:text=,alarm%20information%2C%20information%20relating%20to) . Using historical alarm occurrence data, the system identifies clusters of highly correlated alarms by filtering this graph (essentially finding communities of alarms that co-occur). Once clusters are identified, the tool can simplify the alarm storm by pinpointing which alarms in the cluster are likely root causes and which are consequential. For example, if Alarm A often triggers and is followed by a cascade of Alarms B, C, and D, the system would cluster them and possibly identify A as the root cause alarm for that cluster. **Limitation:** The approach is specific to patterned alarm sequences and doesn’t generalize to arbitrary log data or free-form events. It also doesn’t employ a knowledge graph in the semantic sense – the graph here is more of a statistical correlation graph, not a knowledge representation of the environment. There’s no involvement of LLMs or natural language processing; alarms are treated as IDs with timestamps for correlation math. While effective for reducing noise during an alarm flood, it won’t provide explanations in human language or integrate with, say, an incident ticket describing the alarm in context. Our system can be seen as tackling a superset of this problem: it could ingest alarms as one of the log streams, and the LLM could explain an alarm flood by correlating it with known issues (drawing from the knowledge graph that might link Alarm A to a specific system fault, for instance).
8. **US10530796B2 – Graph Database Analysis for Network Anomaly Detection Systems:** This patent (granted 2020) illustrates a security-focused use of graph databases on log data. It teaches a method where streaming log events (particularly from network devices or security systems) are used to construct a real-time **network event graph**. The system extracts key parameters from log entries (like IP addresses, user IDs, connection attempts) and uses them to form graph entities and edges on-the-fly [21](https://patents.google.com/patent/US10530796B2/en#:~:text=Graph%20database%20analysis%20for%20network,malicious%20event%20associated%20with%20the) . Graph metrics (like the number of distinct connections or unusual access patterns) are computed, and various graph queries are executed to detect anomalies or malicious events [22](https://patents.google.com/patent/US10530796B2/en#:~:text=Systems%20and%20methods%20are%20described,of%20the%20teachings%20described%20herein) [23](https://patents.google.com/patent/US10530796B2/en#:~:text=in%20a%20computer,on%20queries%20performed%20on%20the) . For example, a query might look for a subgraph indicating a host making connections to many new endpoints (potential scanning) or multiple failed logins followed by a success (potential brute force attack). **Limitation:** While this approach shares the idea of using a graph data structure over logderived events, it is narrowly aimed at cybersecurity threats and uses fixed queries/metrics for anomaly detection. There is no interactive natural language layer – the queries are predefined by developers or administrators, not dynamically generated by an LLM. The knowledge graph here is transient (focused on recent network events) and doesn’t include broader contextual knowledge like historical incidents or configurations. In contrast, our system’s knowledge graph spans a wider scope (full-stack entities, incident history) and our LLM agent can formulate or adjust queries based on a user’s natural language questions, covering more than just anomaly detection.

# System Architecture Flow

*Figure: High-level system architecture for the proposed LLM+KG log analysis system. The flow of data and queries is shown from log ingestion on the left to user interaction on the right. Key components include (A) ELK stack for collecting and indexing raw logs, (B) a triplet extraction module feeding structured events into the Knowledge Graph store, (C) integration of ITSM and other context data into the graph, (D) the LLM agent which interprets user queries and interacts with the graph, and (E) a user interface for analysts to query and receive summaries.*

The architecture of the proposed system is composed of modular components working in a pipeline to enable end-to-end intelligent analysis of logs:

* **Log Ingestion Layer (ELK Stack):** Raw log data from various sources (application logs, system logs, network logs, etc.) is ingested through an ELK stack – typically using Logstash to parse and ingest, and Elasticsearch to store and index the logs. This layer handles high-volume streaming data and provides quick text-based searching as a fallback. Kibana or a similar dashboard could be used for basic visualization, but importantly, the logs are also forwarded to the next layer for semantic processing. The ingestion pipeline tags each log with metadata (source, timestamp, severity) and retains it in a time-series index.
* **Triplet Extraction and Knowledge Graph Update:** As logs flow in, an AI-driven parsing module analyzes each entry to extract structured information. This module may use NLP techniques (such as a smaller language model or regex patterns augmented with dictionaries) to find entities and relations in the log text. For example, from a log line “WEB SERVER *srv123* TIMEOUT connecting to DATABASE *db45*,” it might extract a triplet like ( srv123 , timeout connecting to , db45 ). These triples represent edges in the **Knowledge Graph (KG)**. The KG is stored in a graph database (e.g., Neo4j or a similar graph DB) which can efficiently store nodes (entities like servers, services, IPs, error codes) and edges (relationships or events connecting those entities). In our KG, we also have node types for higher-level concepts: an “Incident” node (which might come from an ITSM ticket), or a “Change” node (from a deployment/change management system), etc. Plugins connect to external systems like the ITSM database to pull in these records and link them to log-derived nodes (e.g., linking an Incident node to a Server node if the incident is reported on that server). The result is a living knowledge graph that grows with each new log or relevant event, capturing the state of the environment and its history of interactions.
* **LLM Query Processing and Reasoning Agent:** The centerpiece is the LLM-based agent that interfaces with the user and the knowledge graph. This component includes a large language model (fine-tuned for IT operations, if necessary) and a controller that manages its interactions. When a user poses a question or when an automated alert query is triggered, the agent formulates a strategy to answer: it can translate the natural language query into one or multiple graph queries. For instance, if asked “Summarize what happened to the payment service yesterday,” the LLM might generate Cypher queries to retrieve all nodes and edges (log events, alerts, incidents) related to the payment service in the given time range. The results from the graph (and potentially from raw log index searches, if needed) are then fed back into the LLM. The LLM can interpret these results, perhaps ask for more information (iteratively refining the query), and ultimately compose a summary or explanation. This reasoning loop continues until the LLM agent is satisfied that it has enough information to answer the question. The agent employs chain-of-thought prompting internally, meaning it can break complex questions into sub-queries (e.g., “find all errors preceding the crash, find config changes in that period, compare to last known similar incident”). By unifying the process in one agent, the system avoids brittle, hardcoded logic – the LLM can flexibly decide how to search the graph or logs.
* **User Interface and Feedback Loop:** On the front-end, users interact with the system via a conversational interface (web or chat-based). They can ask questions in plain English (or other languages as supported by the LLM) and receive answers with cited evidence (e.g., references to specific log lines or graph entities). The interface visualizes relevant parts of the knowledge graph on demand – for example, showing a subgraph of services and error events when explaining a root cause. Importantly, the user can provide feedback. If the summary is not accurate, the user can correct it (“Actually, that server was not the cause, it was the database – the logs show a DB failover.”). The system records this feedback and can update the knowledge graph (marking certain nodes as root cause, or adding a link between an incident and the true culprit). Over time, this feedback is used to fine-tune the LLM’s behavior (either via continual learning or adjusting its prompt/hint phrases). This **learning loop** means the system becomes smarter with each resolved incident, gradually accumulating a knowledge base of what solutions or explanations were valid.
* **Security and Governance Layer:** Underpinning the agent’s operation is a safety layer (as mentioned in the novelty section). All queries the LLM agent generates to the knowledge graph run through a sandbox. This layer ensures, for example, that the LLM doesn’t accidentally expose sensitive information to a user without permission (it checks user access levels against data labels in the graph). It also prevents the agent from making destructive changes: the agent by default has readonly access to logs and knowledge graph, except through a controlled interface when updating the graph with confirmed information. Additionally, the system can rate-limit and monitor the LLM’s queries to avoid any infinite loops or runaway costs (since large models can be resource-intensive). If the agent’s reasoning seems to go awry (e.g., stuck in a loop or producing irrelevant results), a watchdog process can intervene or reset the session. This governance layer ensures the solution can be trusted in a production environment.

In summary, the architecture marries **data pipeline components** (for ingestion and knowledge storage) with **AI components** (LLM agent for reasoning, NLP for extraction) and wraps them in a user-centric interface with proper controls. This design allows free-flowing analysis: a user’s natural question propagates through the system, pulls together the right evidence from a variety of sources, and comes back as a coherent answer – much like asking a knowledgeable colleague who has read all the logs and remembers all past incidents.

# Comparison Table

To clearly distinguish our proposed system from the prior art, the table below compares key aspects of all 10 cited examples, including their core approaches and limitations:

**Patent No Title Approach Limitation**

|  |  |  |  |
| --- | --- | --- | --- |
| US20220027331A1 | Cross-Environment  Event Correlation  Using DomainSpace ML | Builds a knowledge graph of events across domains; clusters and correlates  multi-domain events for  RCA [1](https://patents.google.com/patent/US20220027331A1/en#:~:text=nodes%20that%20connect%20across%20the,correlate%20common%20issues%20and%2For%20issues) .  [2](https://patents.google.com/patent/US20220027331A1/en#:~:text=,depicts%20a%20relationship%20between%20container)  [4](https://patents.google.com/patent/US20220027331A1/en#:~:text=,depicts%20a%20relationship%20between%20container) | No LLM or interactive Q&A; relies on predefined correlation logic and domain-specific rules. |
| CN117453501A | Log Analysis  Method, Device,  Equipment and  Storage Medium | Uses NLP to construct a log  knowledge graph and applies log-specific templates for automated  log parsing and incident analysis [3](https://patents.google.com/patent/CN117453501A/en#:~:text=In%20this%20embodiment%2C%20the%20log,speech%20tagging%2C%20named) .  [4](https://patents.google.com/patent/CN117453501A/en#:~:text=In%20the%20embodiment%20of%20the,the%20target%20log%20original%20text) | Template-driven and domain-specific; lacks dynamic natural language querying or generative summaries (no LLM integration) [7](https://patents.google.com/patent/CN117453501A/en#:~:text=In%20this%20embodiment%2C%20the%20log,log%20type%20is%20one%20of) . |
| US20230050889A1 | Method and  System to  Generate KG and  Sub-Graph  Clusters for RCA | Extracts entities/ relationships to build a knowledge graph, then uses unsupervised ML to create sub-graphs and probabilistic models for root cause identification  [8](https://patents.justia.com/patent/20230050889#:~:text=sub,Generated%20Knowledge%20graph) . | Offline model training; not real-time interactive analysis. No use of LLM or natural language interface; limited to patterns the model learns. |
| CN113282764B | Constructing  Network Security  Data Knowledge  Graph | Fuses dictionary knowledge, log-derived triples, and text-derived triples into a multi-layer security knowledge graph for threat analysis [9](https://patents.google.com/patent/CN113282764B/en#:~:text=match%20at%20L275%20%E5%9C%A8%E6%89%80%E8%BF%B0%E5%9F%BA%E7%A1%80%E7%9F%A5%E8%AF%86%E5%9B%BE%E8%B0%B1%E4%B8%8A%E8%9E%8D%E5%90%88%E6%89%80%E8%BF%B0%E6%97%A5%E5%BF%97%E7%9F%A5%E8%AF%86%E4%B8%89%E5%85%83%E7%BB%84%EF%BC%8C%E5%BE%97%E5%88%B0%E6%93%8D%E4%BD%9C%E5%85%B3%E7%B3%BB%E5%9B%BE%E8%B0%B1%EF%BC%9BFusing%20the,obtain%20an%20operational%20relationship%20graph) . | Security-focused and static in scope. Does not support conversational queries; no on-the-fly reasoning with an LLM; requires domain expertise to interpret results. |
| US11275791B2 | Automatic  Construction and  Organization of  KGs for Diagnoses | Automatically links related reports/plots by shared variables, constructing a knowledge graph to navigate diagnostic reports  [10](https://patents.google.com/patent/US11275791/en#:~:text=extracting%20a%20plurality%20of%20variables,an%20edge%20weight%20to%20each) . | Not designed for log streams; provides visualization, not automated reasoning. Lacks natural language outputs or questions – user must manually explore the graph. |
| US11336507B2 | Anomaly Detection and Filtering  Based on System  Logs | Uses LSTM-based sequence modeling to predict log patterns and detect anomalies; filters anomalies via entropy and sentiment analysis, then outputs an anomaly report with root cause [14](https://patents.google.com/patent/US11336507B2/en#:~:text=sequence%20of%20log%20entries%20from,sequence%20and%20a%20root%20cause) . | Focused on anomaly detection only; static pipeline with no user interaction. No knowledge graph or memory of past incidents; no semantic understanding beyond sentiment polarity. |

**Patent No Title Approach Limitation**

|  |  |  |  |
| --- | --- | --- | --- |
| US8495429B2 | Log Message  Anomaly Detection | Parses logs into templates and parameters, infers invariants for each log type, and flags deviations as anomalies [15](https://patents.google.com/patent/US8495429B2/en#:~:text=One%20or%20more%20techniques%20and%2For,sequences%20of%20respective%20log%20types) . | Limited to detecting anomalies per log template. No integration of cross-log relationships or external knowledge; lacks summary or explanation beyond the invariant violation. |
| US20150094959A1 | Heterogeneous  Log Analysis | Clusters and tokenizes heterogeneous logs to discover structures/ patterns automatically; indexes logs based on learned templates [19](https://patents.google.com/patent/US20150094959A1/en#:~:text=200%20as%20an%20input%20to,This) . | Provides structure but no higher insight; no knowledge graph or historical context. User must interpret clusters; no interactive querying or LLM to explain patterns. |
| WO2022155376A1 | Root Cause  Analysis Tool for  Alarms | Creates a correlation graph of alarms with weighted edges; identifies alarm clusters and deduces likely root-cause alarms in a flood [20](https://patents.google.com/patent/WO2022155376A1/en#:~:text=,alarm%20information%2C%20information%20relating%20to) . | Narrowly targeted to alarm co-occurrence. Lacks semantic information about why alarms correlate. Not generalizable to arbitrary logs or queries; no use of LLM or KG. |
| US10530796B2 | Graph DB Analysis for Network Anomaly Detection | Converts streaming network log events into a real-time graph; uses graph metrics and queries to find anomalous or malicious patterns [21](https://patents.google.com/patent/US10530796B2/en#:~:text=Graph%20database%20analysis%20for%20network,malicious%20event%20associated%20with%20the) [22](https://patents.google.com/patent/US10530796B2/en#:~:text=Systems%20and%20methods%20are%20described,of%20the%20teachings%20described%20herein) . | Network-security specific; uses fixed query rules for detection. No natural language support or explanatory interface. Graph is transient and limited to low-level events, without broader incident context. |

**Table:** Comparison of prior art approaches in intelligent log/incident analysis. Our proposed system distinguishes itself by combining elements of these approaches – semantic parsing and graph-based knowledge integration – with the adaptive reasoning and natural language interactivity of an LLM, thereby addressing many of the limitations noted above. Each prior system contributes ideas (e.g., graph-based correlation, invariant detection, log clustering), but none offers the holistic, human-in-the-loop solution enabled by the LLM+Knowledge Graph architecture we propose.

[1](https://patents.google.com/patent/US20220027331A1/en#:~:text=nodes%20that%20connect%20across%20the,correlate%20common%20issues%20and%2For%20issues) [2](https://patents.google.com/patent/US20220027331A1/en#:~:text=%2A%20the%20domain,sets%2C%20by%20using%20an%20information) [24](https://patents.google.com/patent/US20220027331A1/en#:~:text=,depicts%20a%20relationship%20between%20container) US20220027331A1 - Cross-Environment Event Correlation Using Domain-Space Exploration and

Machine Learning Techniques - Google Patents <https://patents.google.com/patent/US20220027331A1/en>

CN117453501A - Log analysis method, device, equipment and storage medium - Google

[3](https://patents.google.com/patent/CN117453501A/en#:~:text=In%20this%20embodiment%2C%20the%20log,speech%20tagging%2C%20named)

[4](https://patents.google.com/patent/CN117453501A/en#:~:text=In%20the%20embodiment%20of%20the,the%20target%20log%20original%20text)

[5](https://patents.google.com/patent/CN117453501A/en#:~:text=analysis%20result%20of%20the%20sample%3B,and%20updating%20the%20entity%20dictionary)

[6](https://patents.google.com/patent/CN117453501A/en#:~:text=graph%2C%20the%20log%20knowledge%20graph,concentrated%2C%20the%20training%20effect%20of)

[7](https://patents.google.com/patent/CN117453501A/en#:~:text=In%20this%20embodiment%2C%20the%20log,log%20type%20is%20one%20of)

Patents <https://patents.google.com/patent/CN117453501A/en>

U.S. Patent Application for METHOD AND SYSTEM TO GENERATE KNOWLEDGE GRAPH AND SUB-GRAPH

[8](https://patents.justia.com/patent/20230050889#:~:text=sub,Generated%20Knowledge%20graph)

CLUSTERS TO PERFORM ROOT CAUSE ANALYSIS Patent Application (Application #20230050889 issued

February 16, 2023) - Justia Patents Search <https://patents.justia.com/patent/20230050889>

CN113282764B - Method and device for constructing network security data knowledge graph - Google Patents <https://patents.google.com/patent/CN113282764B/en>

[9](https://patents.google.com/patent/CN113282764B/en#:~:text=match%20at%20L275%20%E5%9C%A8%E6%89%80%E8%BF%B0%E5%9F%BA%E7%A1%80%E7%9F%A5%E8%AF%86%E5%9B%BE%E8%B0%B1%E4%B8%8A%E8%9E%8D%E5%90%88%E6%89%80%E8%BF%B0%E6%97%A5%E5%BF%97%E7%9F%A5%E8%AF%86%E4%B8%89%E5%85%83%E7%BB%84%EF%BC%8C%E5%BE%97%E5%88%B0%E6%93%8D%E4%BD%9C%E5%85%B3%E7%B3%BB%E5%9B%BE%E8%B0%B1%EF%BC%9BFusing%20the,obtain%20an%20operational%20relationship%20graph)

[10](https://patents.google.com/patent/US11275791/en#:~:text=extracting%20a%20plurality%20of%20variables,an%20edge%20weight%20to%20each) [11](https://patents.google.com/patent/US11275791/en#:~:text=knowledge%20graph%20103%20from%20the,if%20their%20corresponding%20reports%20have) US11275791B2 - Automatic construction and organization of knowledge graphs for problem diagnoses - Google Patents <https://patents.google.com/patent/US11275791/en>

[12](https://patents.google.com/patent/US11336507B2/en#:~:text=FIG,embodiments%20of%20the%20present%20disclosure) [13](https://patents.google.com/patent/US11336507B2/en#:~:text=One%20embodiment%20presented%20in%20this,is%20noteworthy%20based%20on%20a) [14](https://patents.google.com/patent/US11336507B2/en#:~:text=sequence%20of%20log%20entries%20from,sequence%20and%20a%20root%20cause) US11336507B2 - Anomaly detection and filtering based on system logs - Google Patents <https://patents.google.com/patent/US11336507B2/en>

[15](https://patents.google.com/patent/US8495429B2/en#:~:text=One%20or%20more%20techniques%20and%2For,sequences%20of%20respective%20log%20types) [16](https://patents.google.com/patent/US8495429B2/en#:~:text=Current%20and%20previous%20automatic%20detection,may%20need%20extra%20efforts%20to) [17](https://patents.google.com/patent/US8495429B2/en#:~:text=Some%20current%20statistic%20learning%20based,not%20provide%20intuitive%20and%20meaningful) US8495429B2 - Log message anomaly detection - Google Patents

<https://patents.google.com/patent/US8495429B2/en>

[18](https://patents.google.com/patent/US20150094959A1/en#:~:text=,for%20heterogeneous%20log%20clustering%2C%20in) [19](https://patents.google.com/patent/US20150094959A1/en#:~:text=200%20as%20an%20input%20to,This) US20150094959A1 - Heterogeneous log analysis - Google Patents <https://patents.google.com/patent/US20150094959A1/en>

1. WO2022155376A1 - Root cause analysis tool for alarms - Google Patents <https://patents.google.com/patent/WO2022155376A1/en>
2. [22](https://patents.google.com/patent/US10530796B2/en#:~:text=Systems%20and%20methods%20are%20described,of%20the%20teachings%20described%20herein) [23](https://patents.google.com/patent/US10530796B2/en#:~:text=in%20a%20computer,on%20queries%20performed%20on%20the) US10530796B2 - Graph database analysis for network anomaly detection systems - Google

Patents

<https://patents.google.com/patent/US10530796B2/en>