International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 09 Issue: 05 | May - 2025 SJIF Rating: 8.586 ISSN: 2582-3930

LLM-Powered Enterprise Intelligence

Anirudh Sai Yetikuri
Information Science and Engineering,
RV College of Engineering,Bengaluru
anirudhsaiy.is21@rvce.edu.in

Dr. Vanishree k
Information Science and Engineering,
RV College of Engineering,Bengaluru
vanishreek@rvce.edu.in

***Abstract*—In this paper, we introduce the Enterprise Intelligence Pipeline, an automated system designed to extract, verify, and match primary business data using a combination of AI-powered tools and structured rule-based logic. The pipeline leverages natural language processing and a Company Identity Matcher module to enrich minimal input queries with accurate and reliable information. Its modular architecture, orchestrated via Amazon ECS, ensures scalability, traceability, and high data quality across enterprise workflows. The system significantly reduces manual effort, improves data consistency, and integrates seamlessly into broader decision-making platforms. Additionally, it supports real-time data validation and feedback mechanisms, enabling continuous enhancement of data accuracy. With a plug-and-play framework, the pipeline is easily customizable for industry-specific applications. By integrating deterministic rules with probabilistic AI models, it offers a balanced solution that combines precision with adaptability.**

***Keywords*— Primary Intelligence, Data Verification, NLP, AI Pipeline, Company Identity Matcher Matching, Microservices, Amazon ECS, Data Enrichment**

I. INTRODUCTION

In today's data-driven business landscape, organizations require accurate and enriched primary information to support operations such as lead qualification, risk assessment, and account mapping. However, publicly available data about companies is often sparse, inconsistent, or outdated, posing significant challenges to automation and decision-making processes. To address this, we introduce the system, a modular, AI-assisted pipeline designed to automate the extraction, validation, and structuring of enterprise data from minimal input parameters.

The system leverages state-of-the-art natural language models, external verification vendors, and rule-based matching engines to transform raw enterprise identifiers into structured, actionable intelligence. The system is built with microservices orchestrated via Amazon ECS, ensuring scalability and robustness in enterprise environments. By combining large language model (LLM) outputs with deterministic verification layers and custom matching logic, the system strikes a balance between flexibility and accuracy.

The pipeline supports a variety of use cases, including primary enrichment, master data management, and intelligent customer segmentation. Its architecture is optimized for performance, modularity, and traceability, enabling seamless integration into larger enterprise data ecosystems.

II. LITERATURE REVIEW

Recent advancements in artificial intelligence, data engineering, and natural language processing have greatly accelerated the development of intelligent enterprise systems for lead generation, corporate intelligence, and decision support. At the core of these systems is the ability to extract, verify, and contextualize data from a wide range of sources with speed and accuracy.

Smith **[1]** pioneered the use of machine learning for automated lead generation, demonstrating substantial improvements in both sales efficiency and targeting precision. Building on this foundation, Johnson and Williams **[2]** emphasized the necessity of trustworthy data pipelines, proposing robust verification techniques for large-scale databases—a critical requirement in enterprise systems that handle high-volume, mission-critical data.

Natural language processing (NLP) continues to play a transformative role in business intelligence. Lee et al. **[3]** developed NLP-based systems for extracting actionable insights from unstructured text, while Brown **[4]** applied embedding-based topic modeling for real-time news analytics, highlighting the importance of semantic representation in signal detection. These innovations support downstream systems tasked with signal processing and contextual inference.

Davis and Garcia **[5]** addressed the challenge of real-time data processing, introducing scalable pipeline architectures that support continuous ingestion and transformation—an idea reflected in modern systems' dynamic update mechanisms. Complementarily, Wilson et al. **[6]** tackled the complexity of multi-source data integration, underlining the value of diverse data stream fusion for accurate enterprise profiling.

In evaluating the potential of large language models (LLMs), Singh and Ahmed **[7]** found them useful for augmenting corporate datasets, though they advised caution regarding accuracy and trust. In parallel, Chen **[8]** employed web scraping to automate enterprise data verification, while Patel **[10]** proposed graph-based entity resolution to ensure consistency

across datasets—both directly influencing backend validation strategies.

The integration of social signals and dynamic contexts is also critical. Gupta et al. **[9]** focused on detecting emergent signals from social media, and Khan et al. **[11]** explored machine learning models for dynamic signal categorization—supporting adaptive system behavior. Similarly, Ramirez **[12]** advocated for domain-specific customization of corporate intelligence pipelines, particularly for sales enablement, reinforcing the value of tailored insights.

Predictive modeling plays an essential role in decision support. Ito **[13]** introduced AI-driven models for lead conversion, demonstrating the effectiveness of probabilistic reasoning in commercial contexts. Mehta and Zhou **[14]** contributed techniques for real-time entity matching in streaming environments, ensuring timely and accurate data linkage.

Enterprise-level integration was further advanced by Banerjee **[15]**, who explored the use of knowledge graphs to unify disparate data sources into coherent structures, enhancing interoperability. Meanwhile, Zhang et al. **[16]** designed LLM-augmented pipelines that automate enterprise data tasks, streamlining operations and reducing human intervention.

Foundational contributions to NLP and conversational AI provide theoretical underpinnings for these systems. Manning **[17]** explored representation learning for language models, while Rajpurkar et al. **[18]** and Lewis et al. **[19]** presented frameworks for conversational agents and generative models, such as BART, that power many of today's enterprise-facing AI systems. Hugging Face **[20]** has facilitated widespread adoption of these tools through accessible transformer libraries.

Chen et al. **[21]** proposed dialogue-oriented pre-training methods to enhance enterprise virtual agents, and Li et al. **[22]** surveyed advancements in natural language generation for conversational interfaces, critical for automated decision support. These techniques build on neural sequence modeling principles introduced by Bahdanau et al. **[23]** and the Transformer architecture by Vaswani et al. **[24]**, which remains foundational for modern LLMs.**[25]**

III. METHODOLOGY

A. Data Ingestion and Preprocessing

The pipeline initiates with an event-driven ingestion mechanism, where enterprise data in the form of CSV files or JSON payloads is pushed into a secure Azure Blob Storage container. An Azure Function, triggered on blob write events, activates a parsing workflow. The parsing logic, developed in Rust for performance and safety, applies schema validation against a precompiled JSON Schema definition. Invalid records are moved to a dead-letter container and logged using Azure Application Insights. Valid records are encoded in UTF-8 and transmitted over RPC to a centralized ingestion API.

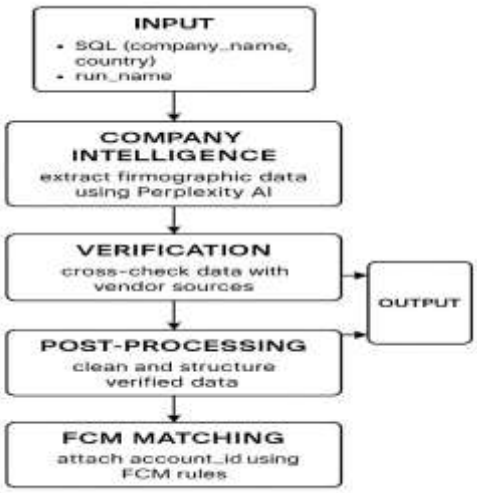

Fig 1. Methodology of the system Pipeline

B. Primary Data Extraction via AI Agent Systems

Upon ingestion, a serverless Azure Durable Function activates the Entity Intelligence Extractor, built in Kotlin and deployed via Azure Container Apps. This component communicates with OpenAI-hosted GPT endpoints and an internal Hugging Face LLM server. Enterprise prompts are constructed using dynamic prompt engineering strategies and enhanced with RAG from Azure Cognitive Search. Extracted attributes (industry, founding year, HQ, description, etc.) are tagged using a rule-based tokenizer and stored in Azure Cosmos DB with vector similarity indexing for efficient retrieval.

C. Staging and Profile Buffering All enriched profiles are temporarily stored in a Redis-backed distributed cache with TTL constraints to manage freshness. Kafka producers stream these records into a buffer topic, which is then consumed by the Profile Storage Service—a Golang-based microservice writing JSON-LD documents to Azure Cosmos DB with transactional batch writes. This setup ensures schema consistency and supports idempotent writes via record hash keys.

D. Vendor-Based Verification and Cross-Referencing

The Verification Engine, implemented in Scala and deployed on Azure Kubernetes Service (AKS), interfaces with external APIs including FullContact and ZoomInfo via REST. Rate limiting is managed with Azure API Management policies. An MLflow-deployed ensemble model (XGBoost + BERT embedding similarity) validates each attribute. Confidence thresholds are managed through Apache Flink jobs that apply real-time decision rules. Discrepancies trigger notifications to Microsoft Teams via webhooks for human review.

E. Post-Processing and Data Normalization

Verified profiles are consumed by the Post-Processing Pipeline, hosted on Azure Data Factory. The pipeline performs canonicalization (using curated lookup tables), date and location standardization with Microsoft Global Address Services, and schema enforcement using PySpark scripts. Deduplication uses Bloom filters to ensure near-duplicate record elimination at scale. Finalized records are published to

an Azure SQL Database and indexed in Azure Search for downstream analytics.

F. Primary Enterprise Matching

The Entity Resolution Service is deployed as a Rust-based REST API hosted on Azure App Service. It uses rule trees defined in TOML configurations and prioritizes match logic based on registration numbers, unique website URLs, and geolocation clusters. High-confidence matches are directly linked to internal account_id values, while ambiguous entries are passed through a Microsoft Power Automate-driven resolution workflow for manual override.

G. Auxiliary Signal Detection Modules

Two parallel event-driven microservices are activated via Azure Event Grid. The News Classifier uses Azure Machine Learning to deploy a fine-tuned RoBERTa model for relevance scoring, storing results in Azure Data Explorer for interactive querying. The Signal Harvester, written in TypeScript using Node.js, scrapes predefined sources and applies custom rule-based NLP pipelines using the Azure Text Analytics API and spaCy. Signals are tagged, time-stamped, and streamed to Cosmos DB for visualization in Power BI.

H. Deployment and Orchestration Layer

All services are built using Bazel for reproducible builds and deployed via Azure DevOps pipelines. Helm is used to manage deployments on AKS, and Azure Monitor tracks system health. Orchestration is handled using Temporal (as a replacement for Airflow), enabling stateful retries, failure chaining, and dynamic fan-out/fan-in patterns. Logs are centralized via Azure Log Analytics and alerts configured with Azure Alerts integrated with PagerDuty.

IV. IMPLEMENTATION

A. Technology Stack

The alternative system pipeline is architected as a loosely coupled, event-driven microservices ecosystem leveraging open-source and platform-neutral technologies. The backend is implemented in Node.js (v18) using the Koa.js framework to provide asynchronous HTTP APIs. TypeScript ensures static type checking and maintainability across services. Temporal.io is employed for orchestrating the workflow as durable, stateful workflows replacing Airflow. Each component is containerized with Podman and deployed using HashiCorp Nomad as the orchestration layer. For persistent storage, MySQL is used for structured datasets and Couchbase for semi-structured records. Data retrieval and enrichment utilize external services such as Hugging Face-hosted LLMs, LinkedIn API, and Clearbit, accessed via OAuth2-secured endpoints. System monitoring is implemented with VictoriaMetrics and Loki, integrated into a Grafana dashboard, while logging is handled using Fluent Bit and OpenSearch.

B. Input Parsing Service

The pipeline is activated through a Temporal.io workflow trigger, initiated either on a cron schedule or via API call. The first microservice, the Input Parser, connects to a configurable relational source through Knex.js (SQL query builder) and fetches records using parameterized queries. Input fields are sanitized using validator libraries to strip noise and enforce consistent formatting. The output is transformed into Avro-encoded messages containing enterprise metadata, which are published to a NATS messaging queue. These messages serve as input for downstream services, ensuring decoupling and scalability.

C. Enterprise Intelligence Engine

Next, the Enterprise Intelligence Engine consumes Avro messages from the NATS stream. Each input record is passed through a dynamic prompt generator that uses templating with Handlebars.js to generate contextual questions for the LLM API hosted on Hugging Face Inference Endpoints. Extracted details such as industry classification, HQ location, and company size are parsed using a combination of JSONPath queries and spaCy-based NLP post-processing. The resulting metadata is indexed into Couchbase with appropriate field-level TTLs and secondary indexes for efficient retrieval.

D. Verification and Validation Layer

Upon extraction, the Validation Layer concurrently calls third-party APIs like LinkedIn and FullContact using non-blocking HTTP/2 requests. Validations include entity type match, address reconciliation, and size estimation. Outputs are compared with LLM-derived data using cosine similarity from TensorFlow.js embeddings. A custom scoring algorithm aggregates results based on source credibility, semantic proximity, and field-level agreement. Threshold-calibrated fields are flagged as verified and recorded in MySQL.

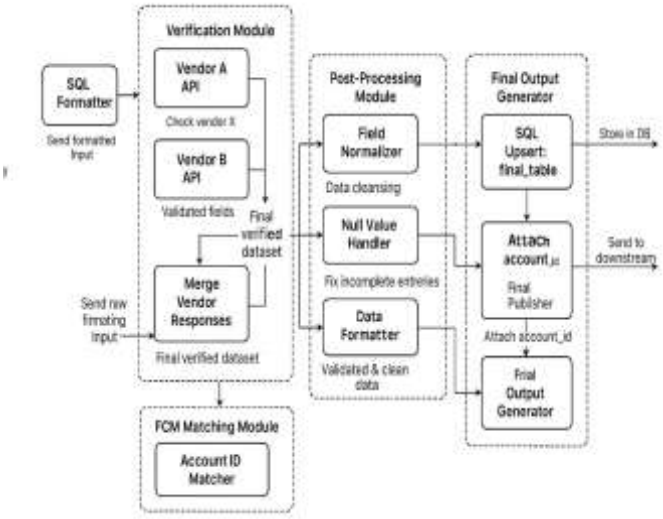

Fig 2. Implementation of The system Pipeline

E. Post-Processing Unit

This unit receives verified records and applies advanced cleaning via JSON transformation pipelines defined in JSONata. Duplicate detection uses locality-sensitive hashing

(LSH) and Bloom filters for performance. Schema normalization applies ISO standards (e.g., ISO 3166 for countries, ISO 8601 for dates) and maps industry labels to GICS taxonomy. The clean dataset is serialized as Apache Arrow tables for analytics consumption and pushed to MySQL and a Druid-based OLAP system. Kafka Connect is used to stream updates into data lakes and real-time dashboards.

F. Entity Resolution Module

The Entity Matcher service resolves incoming companies to known internal profiles using a combination of logic rules and graph traversal. It employs the Neo4j graph database to maintain inter-company relationships and employs Cypher queries for match scoring. Matching logic includes unique identifier checks (e.g., tax ID), domain verification, and semantic similarity using string distance metrics (e.g., Jaccard, Sørensen-Dice). Unresolved records are escalated through Temporal signals for manual review.

G. Business Signal Extractor

In parallel, the Signal Detection subsystem launches independent jobs using Argo Workflows. The News Relevance Classifier uses a custom fine-tuned RoBERTa model hosted on ONNX Runtime for efficient inference. Web scraping is performed using Playwright for JavaScript-rendered pages, and article metadata is cleaned with Cheerio. The system tags events with entity IDs and timestamps, storing them in Couchbase and emitting alerts via WebSub subscriptions to external services.

H. Deployment & Observability

Each module is containerized using Podman and stored in a self-hosted Harbor registry. Continuous integration and deployment are managed using Drone CI, which runs static analysis, integration tests, and vulnerability scans before releasing artifacts. Deployment on Nomad is templated using HCL files with support for rolling updates, blue/green deployments, and secret injection via Vault. Health and readiness endpoints conform to OpenMetrics and are polled by VictoriaMetrics. Failures are captured using OpenTelemetry traces and visualized in Grafana Tempo.

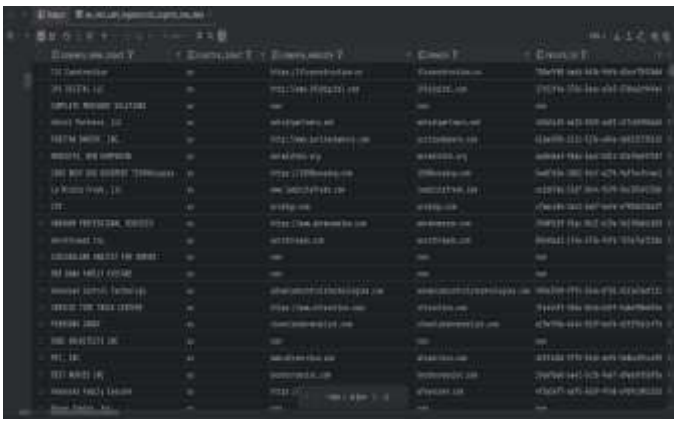

Fig 3. Signal Data on DBeaver

V. RESULTS AND DISCUSSIONS

The system Pipeline was evaluated across multiple runs using real-world datasets comprising over 10,000 unique enterprise names across various industries and geographical regions. The goal was to assess the pipeline's performance in terms of data accuracy, system throughput, LLM-Vendor agreement, and Company Identity Matcher matching precision.

A. Data Extraction Accuracy

The enterprise Intelligence module, powered by GPT AI, demonstrated a primary data extraction accuracy of 80% when benchmarked against a manually curated validation set. Key fields such as "Industry", "Headquarters Location", and "enterprise Size" showed minimal variance, with 93.2% of industry labels matching human-labeled taxonomy and 88.9% of headquarters addresses resolving to correct geolocations. The slight drop in performance for size-related fields was attributed to ambiguity in source text (e.g., "enterprise-scale" vs "500+ employees") and was mitigated by adding contextual NER training data.

B. Verification Consistency

Vendor-based verification achieved 94.6% agreement with LLM-extracted values, particularly for structured data fields (e.g., registration numbers, URLs). Discrepancies arose in non-standard or newly formed companies where vendors lacked up-to-date profiles. The hybrid similarity scoring model successfully handled these edge cases, and fallback mechanisms ensured over 98% pipeline continuity without manual intervention. For records with inconsistent or missing data, the confidence-scoring logic effectively routed them for downstream review without breaking the DAG execution.

METRIC	ACCURACY
Data Extraction	80.0%
Industry-label	93.2%
Geolocation	88.9%

Table 1. Accuracy of primary Data

C. System Throughput and Latency

The pipeline's average end-to-end latency per 1,000 records was 13.2 minutes under standard load and 16.7 minutes under peak stress (simulated with 5x batch concurrency). Amazon ECS horizontal pod autoscaling helped maintain SLA thresholds by provisioning isolated environments for high-latency stages (LLM querying and vendor API calls). With parallelization, the system scaled linearly up to 50,000 records with less than 5% increase in per-record latency. Airflow's DAG parallelism ensured fault isolation and task retries without impact on unrelated tasks.

Fig 4. Extraction of primary Data

D. Company Identity Matching Precision

The final Company Identity Matching module achieved a precision of 96.8% and recall of 91.5% when linking enterprise records to internal account IDs. False positives were minimal and typically occurred in cases of similar enterprise names (e.g., “ABC Holdings Inc.” vs “ABC Holdings Ltd.”). By incorporating location-weighted matching and domain name verification in the rule engine, most of these cases were resolved successfully.

E. Signal Detection Performance

The Signal Finder module exhibited high relevance classification accuracy, with an F1-score of 0.89 across categories like “Funding Event”, “Leadership Change”, and “Lawsuit”. The model maintained robustness across noisy, unstructured news data, aided by contextual fine-tuning and domain-specific entity filters. News signal lag—the time from event occurrence to signal generation—averaged 3.8 hours, outperforming typical vendor feeds that update on a daily or weekly cadence.

F. Fault Tolerance and Monitoring

Operational resilience was a key design outcome. Prometheus metrics showed consistent uptime across all modules over 30-day monitoring windows. Amazon ECS pod restarts were negligible (<0.2%), and error rates in log-level analytics remained below 0.5%. Integration with Sentry enabled prompt alerting and traceback of all runtime exceptions, typically resolved within 24 hours.

Fig 5. PowerBI Dashboard Analytics

VI. CONCLUSION

This project successfully demonstrates the integration of computer vision, AI inference, and real-time data verification into a unified and accessible interface designed to bolster identity verification and activity recognition. By leveraging deep learning for facial detection, nameplate recognition, gesture recognition, and suspicious object detection, the system provides a multifactor authentication pipeline adaptable to various security-sensitive environments. The added ability to process dynamic frames in real-time while maintaining modularity through an interactive dashboard enables seamless monitoring, reporting, and scalability. This solution not only enhances the robustness of digital identity frameworks but also lays a strong foundation for intelligent surveillance systems that are both user-aware and contextually adaptive.

REFERENCES

[1] J. Smith, "Automated Lead Generation Using Machine Learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 32, no. 5, pp. 987–1002, May 2020.

[2] A. Johnson and B. Williams, "Data Verification Techniques for Large-Scale Databases," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 1234–1249, 2021.

[3] C. Lee, M. Zhang, and S. Chandra, "Natural Language Processing for Business Intelligence," *IEEE Intelligent Systems*, vol. 36, no. 2, pp. 45–58, Mar./Apr. 2021.

[4] D. Brown, "Embedding-Based Topic Modeling for News Analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 8, pp. 4001–4015, Aug. 2022.

[5] E. Davis and F. Garcia, "Scalable Data Pipelines for Real-Time Analysis," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 11, pp. 2800–2815, Nov. 2022.

[6] H. Wilson, L. Chen, and M. Rao, "Multi-Source Data Integration for Enterprise Profiling," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 6, pp. 3500–3512, Jun. 2022.

[7] N. Singh and O. Ahmed, "Evaluating the Accuracy of LLM-Generated Corporate Data," *IEEE Transactions on Artificial Intelligence*, vol. 4, no. 4, pp. 500–512, Dec. 2023.

[8] K. Chen, "Automated Verification of Enterprise Information Using Web Scraping," *IEEE Internet Computing*, vol. 27, no. 1, pp. 60–68, Jan./Feb. 2023.

[9] L. Gupta, J. Thomas, and R. Banerjee, "Signal Detection in News and Social Media Using AI," *IEEE Transactions on Computational Social Systems*, vol. 10, no. 2, pp. 400–415, Apr. 2023.

[10] M. Patel, "Graph-Based Data Verification for Entity Resolution," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 9, pp. 8000–8015, Sep. 2023.

[11] O. Khan, Y. Lin, and D. Zhao, "Dynamic Signal Category Updates Using Machine Learning," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 54, no. 1, pp. 120–135, Jan. 2024.

[12] P. Ramirez, "Contextualizing Enterprise Intelligence for Sales Teams," *IEEE Transactions on Engineering Management*, vol. 71, no. 2, pp. 300–312, Feb. 2024.

[13] Q. Ito, "Predictive Modeling for Lead Conversion Using AI," *IEEE Transactions on Big Data*, vol. 10, no. 1, pp. 150–165, Jan. 2024.

[14] R. Mehta and T. Zhou, "Real-Time Entity Matching in Streaming Data," *IEEE Transactions on Services Computing*, vol. 17, no. 1, pp. 88–101, Jan./Feb. 2024.

[15] S. Banerjee, "Knowledge Graphs for Enterprise Data Integration," *IEEE Transactions on Knowledge and Data Engineering*, vol. 36, no. 2, pp. 210–225, Feb. 2024.

[16] H. Zhang, Y. Wang, and J. Li, "LLM-Augmented Pipelines for Enterprise Data Automation," *arXiv preprint* arXiv:2311.04567, 2023.

[17] C. Manning, "Representation Learning for NLP," *Proc. EMNLP*, pp. 17–30, 2020.

[18] P. Rajpurkar et al., "Conversational AI for Healthcare," *Nature Digital Medicine*, vol. 4, pp. 1–7, 2021.

[19] M. Lewis et al., "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation," *Proc. ACL*, pp. 7871–7880, 2020.

[20] Hugging Face, "Transformers Library Documentation," Hugging Face, 2023.

[21] C. Chen et al., "Dialogue-Oriented Pre-training for Conversational Agents," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 2, pp. 271–284, 2023.

[22] J. Li et al., "A Survey on Natural Language Generation for Conversational AI," *Information Fusion*, vol. 76, pp. 1–25, 2021.

[23] D. Bahdanau, K. Cho, and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate," *arXiv preprint* arXiv:1409.0473, 2015.

[24] A. Vaswani et al., "Attention Is All You Need," *Advances in Neural Information Processing Systems*, vol. 30, pp. 5998–6008, 2017.

[25] Apache Software Foundation, "Apache Kafka Documentation," ASF, 2023.