Patient Journey Visualization using NoSQL and Machine Learning Algorithms

S.Bhavya sri
School of Computer Science and Engineering
Vellore Institute of Technology
Chennai-600127, Tamilnadu
saguturubhavya.sri2022@vitstudent.ac.in

T.Sai Keerthi
School of Computer Science and Engineering
Vellore Institute of Technology
Chennai-600127,Tamilnadu
thatikonda.sai2022@vitstudent.ac.in

Dr. Gayathri Devi S
School of Computer Science and Engineering
Vellore Institute of Technology
Chennai-600127, Tamil Nadu
gayathridevi.s@vit.ac.in

Natarajan B
School of Computer Science and Engineering
Vellore Institute of Technology
Chennai-600127, Tamil Nadu
natarajan.b@vit.ac.in

Abstract*—The need for analytical systems that provide efficiency and scalability is growing as healthcare data grows in volume and complexity. This work proposes an integrated framework for patient journey analysis by combining the strengths of NoSQL databases, represented by MongoDB; graph databases, represented by Neo4j; and machine learning-based clustering techniques. The proposed framework models the main aspects of patients' interactions, such as encounters, diagnoses, medications, and procedures, using synthetic Electronic Health Record data. Neo4j allows for the identification of insights that are relationship-based, while MongoDB provides flexible and adaptive document storage. Machine learning clustering methods are then used to find hidden patterns in the behavior of patients. These technologies together will demonstrate how the integration of NoSQL data management with graph analytics contributes to better decision-making in healthcare and allows for more personalized treatment approaches.

Keywords: MongoDB, Neo4j, Machine Learning, Patient Journey Analytics, Healthcare Data, Clustering, NoSQL Databases, Graph Analytics, Data Mining, Predictive Healthcare, Big Data Analytics.

I. Introduction

Over the past years, healthcare organizations have increasingly migrated from relational databases to more flexible and scalable NoSQL architectures capable of handling enormous volumes of diverse data. The patient data is often semi-structured in nature, including clinical observations, prescriptions, and interactions. Interconnected datasets such as this require a hybrid analytical approach that combines the scalability of document stores like MongoDB with the relationship modeling power of graph databases like Neo4j.

- Database Focus: Uses NoSQL databases to handle semi-structured and unstructured healthcare data.
- Objective: The aim is to observe, assess, and document every phase of the patient's experience from the consultative stages to treatment and follow-up.
- ML Integration: uses machine learning algorithms to find trends, predict outcomes and generate insights.
- Visualization: Provides interactive visualizations of the patient pathway for easy comprehension.

II. RELATED WORK

Recent research has explored various aspects of using NoSQL databases and machine learning in healthcare analytics, but with the focus mostly on scalability, flexible integration of data, and patient journey modeling. Early theoretical work by Taylor & Francis [1] compared SQL to NoSQL systems, highlighting how MongoDB and Neo4j might simplify data preparation, although without practical applications in healthcare.

A more applied IEEE study [2], using Neo4j to enhance the machine learning processing in tele-rehabilitation, showed faster graph-based queries but lacks large-scale validation. Also, a thesis [3] modeled EHRs as graphs in order to enhance the interpretability of the predictions. It does not compare against other NoSQL systems and does not use any external datasets.

Clustering and sequence analysis applied to patient journeys were shown in the research [4]; other JMIR research revealed hidden patterns but did not evaluate the performance of NoSQL. Another concept-focused work of Taylor & Francis [5] demonstrated how Neo4j's flexible schema captures complex healthcare interactions; it did not include experimentation.

ProQuest's IoHT diagnostic model [6] used Neo4j in conjunction with wearable data, improving contextual analytics but still lacking real-time ingestion and data privacy. An overview by IEEE [7] discussed the storage of nation-wide EHRs using NoSQL, emphasizing scalability, but again avoided comparisons with ML or graph-based approaches.

Frontiers [8] demonstrated machine learning to predict hospital discharge dates; however, it relied on relational data and not on NoSQL. Springer's case study used process mining in order to map patient journeys but did not use real-time NoSQL pipelines.

The JUIT "Patient Explorer" prototype [10] provided only basic visualization facilities and didn't have ML and NoSQL support. Elsevier's review [11] proposed hybrid SQL-NoSQL models for ML pipelines, but this work was largely conceptual and with limited healthcare validation.

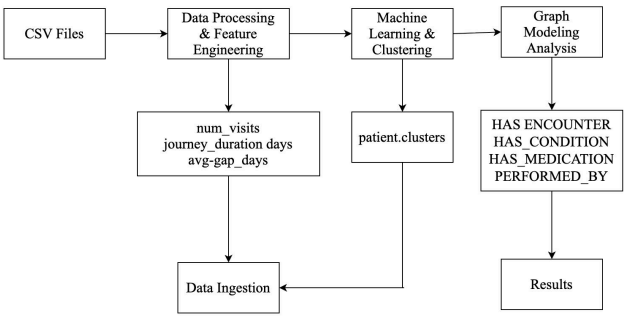
In one dissertation on agile healthcare analytics [12], data-driven infectious-disease tracking was presented, but NoSQL and ML-based decision support were not included. An IEEE study [13]

compared SQL vs. NoSQL for long-term storage of EHRs but did not link systems across hospitals or integrate predictive models.

Clustering and NLP were applied to the patient narratives in the IJDesign study [14], yielding good results, but still without structured NoSQL support. Lastly, the IEEE effort [15] demonstrated once again Neo4j's efficiency gains for tele-rehabilitation ML preprocessing, though at the small-scale testing level.

III. METHODOLOGY

A. System Overview



B. Data Ingestion Layer

Health Care CSV files of patients, encounters, conditions, medications, procedures are imported using mongoimport. The flexible schema of MongoDB efficiently stores diversified medical attributes.

C. Data Processing and Feature Engineering Layer

A Python script, denormalize.py, cleans and unifies information, generating a set of key features around number of visits, journey duration, and average gap between encounters. These are stored in the collection called 'patient_features'.

D. Machine Learning and Clustering Layer

K-Means Clustering Python script performs unsupervised learning, grouping similar patients from their journey characteristics in clusters. The result will be stored into patient_clusters, validated with the Silhouette Score, and plotted via pairplots or cluster diagrams.

E. Graph Modeling and Relationship Analysis Layer

Neo4j represents a patient, encounter, condition, medication, and provider as nodes in a graph, connected via relationships defined by Cypher. This allows multi-hop queries to find patterns in co-occurring conditions and treatment paths.

F. Analytical Visualization and Result Storage

The scripts generate summaries and visualizations that include cluster reports, lists of sample patients, and journey timelines stored in the result directory, marrying statistical clustering with graph-based insights.

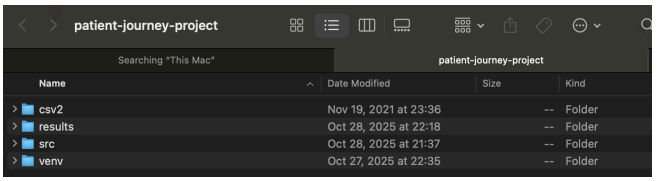
IV. RESULTS AND DISCUSSION

A. Environment Setup

The following tools and libraries were used:

- MongoDB 8.0.2 - document-based data storage with flexible schema handling.
- Neo4j 5.x - for graph-based modeling and the investigation of relationships.
- Python Libraries: pandas, numpy, pymongo, scikit-learn, matplotlib, and seaborn.
- Dataset Source: Synthea-generated synthetic Electronic Health Records (EHR) data.

The project directory structure included:



B. Data Ingestion and Storage

These data are made up of patients, encounters, conditions, medications, and procedures that were imported to MongoDB using mongoimport, each into its collection within the patient_journey database. With MongoDB's flexible schema, it was easy to integrate these diverse records and handle more than 1 million entries, thus supporting fast read/write operations during pre-processing.

C. Data Preprocessing and Feature Engineering

Denormalization was done via denormalize.py, which merged data from a number of collections into a single patient-level dataset using pymongo. Each record included the number of visits, journey start/end dates, journey duration, and average visit gap. The integrated data at the end was stored in MongoDB as the patient_features collection.

D. Patient Clustering Pipeline

Then, the clustering model was applied on the patient feature dataset, which was stored in a collection called patient_features. The model automatically determined four optimal clusters for a Silhouette Score of 0.475, representing a moderate, yet marked, separation between the clusters.

Table I: summarizes the key statistics for each cluster.

Cluster	Avg. No. of Visits	Avg. Journey Duration (days)	Avg. Gap (days)
0	26.87	14,866.53	584.22
1	80.27	19,626.15	276.65
2	34.35	4,854.59	156.55
3	627.40	27,906.00	51.97

Cluster Statistics and Insights

After performing **K-Means clustering (k=4)** on patient journey features, the following distribution was obtained.

Each cluster represents a distinct group of patients with similar engagement patterns.

Table III: Cluster Statistics and Insight

Cluster	**Count**	**Mean Num. Visits**	**Mean Journey Duration (Days)**	**Mean Avg. Gap (Days)**
0	213	26.87	14,866.53	584.22
1	374	80.27	19,626.15	276.65
2	566	34.35	4,854.59	156.55
3	10	627.40	27,906.00	51.97

E. Neo4j Graph Database Implementation

To capture complex interconnections, the same dataset was imported into **Neo4j**, creating over **527,803 nodes** and **831,645 relationships**.

The following node labels and relationship types were established.

Node Labels	**Relationship Types**
Patient, Condition, Medication, Procedure, Encounter, Provider, Organization, Allergy	HAS_CONDITION, HAS_MEDICATION, HAS_PROCEDURE, HAS_ENCOUNTER, PERFORMED_BY, WORKS_FOR, HAS_ALLERGY

The graph structure enabled seamless exploration of multi-entity pathways such as **Patient → Encounter → Provider → Medication → Condition**.

Neo4j Query Insights

To explore clinical patterns and dataset connectivity, a series of analytical Cypher queries were executed.

The following table summarizes their objectives and corresponding findings.

Table IV: Neo4j Query Insights for Patient Journey Analysis

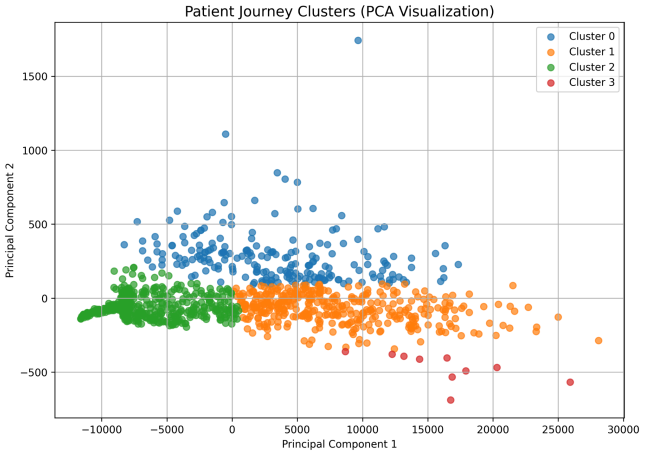
Query No.	**Objective / Purpose**	**Cypher Query Executed**	**Key Outcome**
1	Verify patient dataset size	MATCH (p:Patient) RETURN COUNT(p);	Confirmed total number of patient nodes (~1,163).
2	Count total encounters	MATCH (e:Encounter) RETURN COUNT(e);	Validated total encounters (~61,459 records).
3	Identify most frequent medical conditions	MATCH (p:Patient)-[:HAS_CONDITION]->(c:Condition) RETURN c.description, COUNT(p) ORDER BY COUNT(p) DESC LIMIT 10;	Revealed most prevalent health conditions among patients.
4	Identify commonly prescribed	MATCH (p:Patient)-[:HAS_MEDICATION]->(	Determined frequently used medications

	medications	m:Medication) RETURN m.description, COUNT(p) ORDER BY COUNT(p) DESC LIMIT 10;	and treatment patterns
5	Determine average encounters per patient	MATCH (p:Patient)-[:HAS_ENCOUNTER]->(e:Encounter) RETURN COUNT(e)* 1.0/COUNT(DISTINCT p);	Estimated an average of ~53 encounters per patient, showing care continuity.
6	Find top cities with most patients	MATCH (p:Patient) RETURN p.city AS City, COUNT(p) AS PatientCount ORDER BY PatientCount DESC LIMIT 10;	Found cities with highest patient counts (e.g., Springfield, Boston).
7	Count unique providers and workloads	MATCH (pr:Provider) <-[:PERFORMED_BY]-(e:Encounter) RETURN pr.name, COUNT(e) ORDER BY COUNT(e) DESC LIMIT 10;	Highlighted most active healthcare providers.
8	Verify relationships across graph model	MATCH ()-[r]->() RETURN TYPE(r), COUNT(r);	Displayed all relationships (HAS_CONDITION, HAS_MEDICATION, etc.) to confirm linkage accuracy.

F. Visualization and Export Layer

The processed results were exported as visual and tabular summaries. The framework generated:

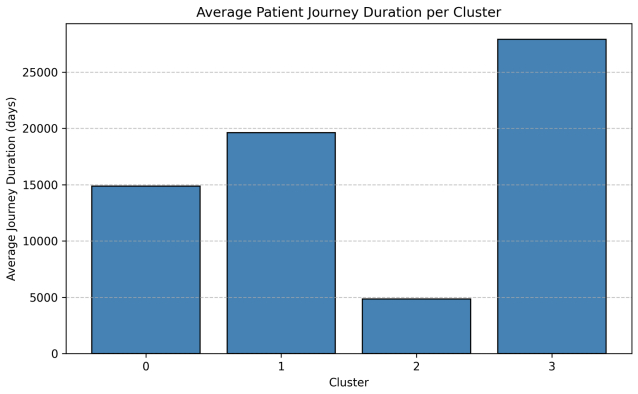
- Cluster Visualization: Color-coded scatter plots showing patient groupings.



- Pair Plot: Relationship among journey metrics across clusters.

- Timeline Graphs: Treatment durations per cluster.

Sample Patient Records: One representative record per cluster for interpretability.



G. System Integration

Finally, MongoDB and Neo4j were integrated conceptually to enable dual-mode querying:

- MongoDB supported the feature-level analytics: quantitative clustering.
- Neo4j provided relationship-level analytics - qualitative graph exploration.

H. Discussion

This combination of MongoDB and Neo4j forms a hybrid NoSQL system that facilitates scalable storage along with rich relational analytics. MongoDB ingests denormalized data with high velocity, while Neo4j uncovers complex relationships that are difficult to represent across traditional tables. Clustering results identify valuable patient behavioral segments that can be utilized for personalized care and resource planning. Neo4j further confirms treatment overlaps, commonly occurring conditions, and care continuity, while the visual graphs enrich clinical insight. Overall, the multimodel approach enhances scalability, lessens query complexity, and strengthens interpretability for patient journey analytics.

V. CONCLUSION AND FUTURE WORK

The proposed hybrid NoSQL framework couples MongoDB and Neo4j for examining aspects of the patient journeys through both a statistical and relational lens. Large EHR data sets are preprocessed and denormalized by MongoDB to create patient-level features like visit frequency, journey duration, and average visit gaps. Utilizing these features as input, a K-Means model discovered four patient clusters with a Silhouette Score of 0.475, thereby highlighting different healthcare utilization patterns.

Neo4j adds relational depth by modeling relationships among the patients, encounters, providers, conditions, and medications. Co-occurring conditions, provider workloads, and other clinically meaningful relationships come into view through Cypher queries. Together, MongoDB and Neo4j create a scalable, multidimensional, and interpretable system to understand the trajectories of patient care.

Future Work

Future enhancements will focus on integrating FHIR/HL7 standards for real-world interoperability, adding temporal models like LSTM or BERT for outcome prediction, enabling real-time analytics through Kafka and Streamlit, and incorporating privacy-preserving methods such as federated learning and differential privacy. Clinical expert validation will also be pursued to confirm the usefulness of the identified clusters and relationship patterns.

References

[1] Celesti, A., Celesti, F., Galletta, A., Fazio, M., & Villari, M. (2020, July). Improving machine learning algorithm processing time in tele-rehabilization through a nosql graph database approach: A preliminary study. In 2020 IEEE Symposium on Computers and Communications (ISCC) (pp. 1-6). IEEE.

[2] Ahmed, N., Heyer, R., & Broneske, D. (2023). Graph Representation of Patient's Data in EHR for Outcome Prediction (Doctoral dissertation, Master Thesis, Otto-von-Guericke University).

[3] Allam, A., Feuerriegel, S., Rebhan, M., & Krauthammer, M. (2021). Analyzing patient trajectories with artificial intelligence. Journal of medical internet research, 23(12), e29812.

[4] Rojas, E., Munoz-Gama, J., Sepúlveda, M., & Capurro, D. (2016). Process mining in healthcare: A literature review. Journal of biomedical informatics, 61, 224-236.

[5] Mahyoub, M. A., Dougherty, K., Yadav, R. R., Berio-Dorta, R., & Shukla, A. (2024). Development and validation of a machine learning model integrated with the clinical workflow for inpatient discharge date prediction. Frontiers in Digital Health, 6, 1455446.

[6] Abo-Hamad, W. (2017, June). Patient pathways discovery and analysis using process mining techniques: An emergency department case study. In International conference on health care systems engineering (pp. 209-219). Cham: Springer International Publishing.

[7] El-Bouri, R., Taylor, T., Youssef, A., Zhu, T., & Clifton, D. A. (2021). Machine learning in patient flow: a review. Progress in Biomedical Engineering, 3(2), 022002.

[8] Reychav, I., McHaney, R., Babbar, S., Weragalaarachchi, K., Azaizah, N., & Nevet, A. (2022). Graph network techniques to model and analyze emergency department patient flow. Mathematics, 10(9), 1526.

[9] Schäfer, J., & Wiese, L. (2023). A graph-based data model for digital health applications. In Graph Databases (pp. 157-177). CRC Press.

[10] Mohammed, S., & Fiaidhi, J. (2021). The road map of building e-diagnostics services using neo4j graph connectivity and analytics for the internet of healthcare things (ioht). International Information Institute (Tokyo). Information, 24(2), 93-106.

[11] Lodha, K., & Kumar, Y. (2022). Software Solution for Augment Curation and Patient Explorer.

[12] Asiri, M. A. A. (2024). An Agile Data Analytics Framework to Improve Healthcare Process Performance in Infectious Disease Propagation
(Doctoral dissertation, State University of New York at Binghamton).
[13] Oufkir, L., & El Hamzaoui, A. (2024, July). EHR Data Persistence for a Nationwide Shared Medical Record: An Overview. In 2024 11th
International Conference on Wireless Networks and Mobile Communications (WINCOM) (pp. 1-9). IEEE.
[14] Meher, D., Kaur, B., Pawar, A. N., & Parekh, S. (2023). Databases for machine learning: A journey from SQL to NOSQL. In Recent Advances
in Material, Manufacturing, and Machine Learning (pp. 248-257). CRC Press.
[15] Celesti, A., Celesti, F., Galletta, A., Fazio, M., & Villari, M. (2020, July). Improving machine learning algorithm processing time in tele-rehabilization through a nosql graph database approach: A preliminary study. In 2020 IEEE Symposium on Computers and Communications
(ISCC) (pp. 1-6). IEEE.