LEX LOCALIS-JOURNAL OF LOCAL SELF-GOVERNMENT
ISSN:1581-5374 E-ISSN:1855-363X
VOL. 24, NO. 2(2026)

ENHANCING LOCAL SELF-GOVERNMENT INITIATIVES IN INDIAN DIGITAL AGRICULTURE THROUGH THE USE OF GRAPH CONVOLUTIONAL NETWORKS

Gayathri Ramalingam[1*], Dr. Thenmozhi Durairaj[2]

[1*]Meenakshi Sundararajan Engineering College, Chennai, 600024, Tamilnadu, India
[2]Sri Sivasubramaniya Nadar College of Engineering, Chennai, 603110, Tamilnadu, India

Correspondence: R. Gayathri: rgayathri@msec.edu.in[1]

Abstract
The agricultural sector involves complex relationships among various entities such as crops, fertilizers, soil types, and seasons. Representing and managing these relationships in a meaningful and structured way is essential for efficient information retrieval and knowledge sharing. RDF-based Agricultural Knowledge Graphs (AKGs) have emerged as an effective solution for modelling these relationships; however, they are often static and lack support for advanced querying, semantic enrichment, and classification. This project presents a comprehensive system that enhances RDF-based AKGs using Graph Convolutional Networks (GCNs) for structural learning and optimization. The process begins with parsing raw RDF data in Turtle (.ttl) format and constructing directed graphs using NetworkX. These graphs are then converted into PyTorch Geometric objects for training a multi-layer GCN model, which generates meaningful node embeddings. A Top-K pooling technique is applied to filter redundant or insignificant nodes, resulting in a more concise and interpretable graph. The final, enriched graph is exported to Neo4j for semantic visualization and querying, enabling efficient extraction of agricultural insights. Experimental results demonstrate significant improvements in graph clarity, classification accuracy training: 98.22%, testing: 99.02%, and query effectiveness. This approach supports better data organization, simplifies knowledge retrieval, and strengthens the utility of AKGs in agriculture.

Keywords: Indian Digital Agricultural Management, Turtle Format, Graph Convolutional Network, Semantic Enrichment, Graph Optimization.

1. Introduction

Agriculture plays a vital role in sustaining economies and societies across the globe, particularly in developing countries where a large portion of the population depends on it for livelihood. With the growing demand for food production, sustainable practices, and precision farming, the need for organized and structured agricultural knowledge has become increasingly important [1][4]. Over the years, a significant volume of agricultural data ranging from crop types and soil conditions to fertilizers, weather patterns, and pests has been generated. However, this data is often stored in unstructured or semi-structured formats, making it difficult to integrate, retrieve, and analyze effectively [3].

To address this challenge, Agricultural Knowledge Graphs (AKGs) have emerged as a powerful tool for structured data representation [1][5]. AKGs use the Resource Description Framework (RDF) to encode information as semantic triples, capturing relationships among various agricultural entities such as crops, diseases, treatments, and climate conditions [2][6]. While RDF-based AKGs facilitate data interoperability and semantic clarity, they are inherently static and often lack the dynamic capabilities needed for advanced querying, classification, or real-time decision support.

This project aims to overcome these limitations by enhancing RDF-based AKGs through the application of graph processing techniques. The proposed system converts RDF data into directed graphs using NetworkX and applies graph transformation and optimization through PyTorch Geometric. Techniques like Top-K pooling help reduce graph complexity by eliminating less relevant nodes, thereby improving the graph's interpretability and performance. Finally, the processed graph is imported into Neo4j, which enables visual exploration and semantic querying of the knowledge graph [4][5]. This holistic approach strengthens the use of

AKGs in agricultural research, smart farming, and knowledge-driven decision-making, paving the way for more intelligent and accessible agricultural information systems [6][7].

2. Literature Review

This section provides a detailed elaboration of the related work.

2.1 Graph Convolutional Network

This section provides a detailed elaboration of the related work. Graph Convolutional Networks (GCNs) represent a pivotal advancement in deep learning designed to operate directly on graph-structured data. Unlike traditional neural networks that assume regular grid-like data structures (e.g., images or sequences), GCNs perform learning on nodes and their relationships within graphs, making them ideal for knowledge graph applications. In the context of this project, GCNs are employed to extract and propagate semantic features from RDF-based Agricultural Knowledge Graphs (AKGs), enabling more intelligent reasoning, classification, and decision support mechanisms across agricultural domains such as crop management, pest control, and soil treatment. This aligns with the approach demonstrated in the work of Xuhui Zeng et al. [5], where GCNs were used to recommend ecological patterns based on geographic features encoded in a knowledge graph.

A GCN functions by aggregating feature information from a node's neighbourhood and updating its representation layer by layer. Each GCN layer performs a spectral or spatial convolution, transforming raw node features into high-level embeddings that capture both node attributes and structural context. In this project, after converting RDF data into directed graphs, the GCN layers are used to enrich the representation of each agricultural entity—be it a crop, fertilizer, or disease—based on the graph's topology and its linked semantics. This embedding process allows the system to predict relationships, classify node types, or detect anomalies in the agricultural data structure. As highlighted in [6] by Rahul Chiranjeevi et al., integrating GCNs into plant disease recognition has shown significant potential in achieving higher accuracy due to their contextual awareness—demonstrating the same potential here for structured agricultural knowledge.

The incorporation of GCNs within the AKG pipeline helps overcome the limitations of static RDF triples by enabling dynamic learning and fine-grained representation of knowledge entities. The multi-layered architecture of GCNs facilitates deeper inference by allowing the model to learn from both direct and indirect connections across the knowledge graph. This deep relational learning approach supports better generalization and adaptability of the AKG system, especially when applied to query optimization or semantic search tasks, similar to the intelligent query systems discussed in [1, 2, 3]. By enabling context-sensitive feature propagation, the GCN model also supports future scalability of the knowledge graph reasoning framework, as also suggested by Shang et al. [7] in their multi-granularity learning for KG reasoning. Depicts in figure 1.

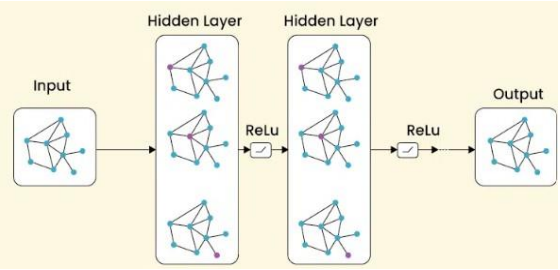

Figure 1. GCN Architecture

ALGORITHM

1. Add self-loops to adjacency matrix: $\hat{A} = A + I$

2. Compute degree matrix D where $D_{ii} = \sum_j \widehat{A_{ij}}$
3. Normalize adjacency matrix: $\tilde{A} = \hat{A}/(\sqrt{D} + \epsilon)$
4. First GCN layer: $H_1 = ReLU(\tilde{A} \cdot X \cdot W_1)$
5. Second GCN layer: $H_2 = ReLU(\tilde{A} \cdot H_1 \cdot W_2)$
6. Return H_2

2.1.1 Log Softmax Activation

The Log Softmax function is commonly employed in the final layer to normalize the network output into log-probabilities across multiple classes. This is particularly suitable for multiclass classification tasks on graph nodes or entities, such as labeling nodes in an agricultural knowledge graph (AKG). After passing the input through several GCN layers and potentially a fully connected layer, the Log Softmax function ensures that the output is interpretable as a probability distribution over all classes.

$$\log Softmax(x_i) = x_i - \log\left(\sum_j e^{x_j}\right)$$

Here, x_i represents the input score (logit) for the i^{th} class, and the denominator represents the summation of exponentials of all input logits, ensuring normalization. The subtraction of the log-sum-exp term from each logit avoids numerical instability, which makes Log Softmax particularly preferred over direct softmax followed by logarithm.

In this project, Log Softmax helps improve the interpretability and convergence of the model by enabling the use of negative log-likelihood loss (NLLLoss) for supervised training on labeled nodes within the agricultural knowledge graph. This activation plays a crucial role in translating the learned node features into class-specific confidence scores for efficient classification.

2.2 Top-K Pooling

Top-K Pooling is a graph pooling method that reduces the size of the graph by selecting only the most informative nodes based on learned scores, helping in hierarchical graph representation and preventing overfitting. In the context of your agricultural knowledge graph (AKG), which contains numerous entities such as crops, pests, and fertilizers represented as nodes, Top-K Pooling ensures that only the most semantically significant entities are preserved for deeper learning stages. This allows the model to retain essential agricultural knowledge while discarding noisy or less relevant nodes, enhancing both computational efficiency and classification performance.

In this method, a learnable projection vector is used to assign scores to each node. Based on these scores, only the top k% nodes (with highest scores) are retained, and their associated features and edges are carried forward to the next GCN or dense layer. For your AKG-based project, this means that during each pooling step, Top-K pooling helps focus the model on the most relevant agricultural concepts and relationships—such as the most prominent diseases for a crop or the most effective treatments—by dynamically filtering out less informative parts of the graph. This is especially useful when the graph is large or highly redundant, which is common in semantic RDF-based structures.

Top-K Pooling complements GCNs by adding a hierarchical abstraction layer, allowing for global understanding of graph data across levels of granularity. As described by Zeng et al. [5] and Rahul Chiranjeevi et al. [6], applying graph convolutional operations followed by pooling can significantly enhance the model's ability to generalize from complex agricultural datasets. This selective reduction of graph size ensures the learned embeddings are both rich and efficient, thus improving downstream tasks like question answering or recommendation within the AKG system.

Formula

- Node feature matrix: $X \in R^{n \times d}$
- Learnable projection vector: $p \in R^{d}$
- Percentage of nodes to retain: $k \in (0,1)$

1. **Score Calculation for each node i:**

$$y_i = x_i^T p$$

2. **Top-K Selection:**

$$I = indices\ of\ Top - k(y)$$

3. **Retaining selected nodes and updating features and adjacency:**

$$[X' = X[I], \quad A' = A[I, I]]$$

2.3 Adam Optimizer

In this project, the Adam (Adaptive Moment Estimation) optimizer plays a pivotal role during the training phase of the Graph Convolutional Network (GCN) model integrated with the agricultural knowledge graph. Adam is an advanced gradient descent optimization algorithm designed to update network weights efficiently by combining the advantages of both AdaGrad and RMSProp. Its ability to handle sparse gradients and non-stationary objectives makes it well-suited for processing high-dimensional agricultural graph data, such as soil conditions, crop-pest relations, and fertilizer interactions, which are inherently complex and dynamic.

Adam maintains individual learning rates for each parameter and adapts these rates using estimates of the first (mean) and second (uncentered variance) moments of the gradients. This adaptive approach accelerates convergence and enhances stability, especially important when optimizing graph structures derived from RDF triples. In this project, Adam improves the performance of the GCN-based model by efficiently learning feature representations over the transformed RDF graphs, thereby enabling accurate reasoning and classification over the agricultural knowledge base.

The optimizer is particularly effective in scenarios involving heterogeneous node features and varying edge connections, such as those represented in the Neo4j-visualized graph of this system. Its dynamic adjustment capability ensures robust gradient updates even with noisy or incomplete graph data, which is common in agricultural datasets [6][9].

Adam Optimizer Formula

- (θ_t) *be the parameters at time step* (t)
- (g_t) *be the gradient at time step* (t)
- (α) *be the learning rate*
- (β_1, β_2) *be the exponential decay rates for the moment estimates*
- (ϵ) *be a small constant to prevent division by zero*

1. **Compute biased first moment estimate:**

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

2. **Compute biased second raw moment estimate:**

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

3. **Compute bias-corrected moment estimates:**

$$\widehat{m_t} = \frac{m_t}{1 - \beta_1^t}, \quad \widehat{v_t} = \frac{v_t}{1 - \beta_2^t}$$

4. **Update parameters:**

$$\theta_{t+1} = \theta_t - \alpha \frac{\widehat{m_t}}{\sqrt{\widehat{v_t}} + \epsilon}$$

3. System Architecture

The proposed system applies Graph Convolutional Networks (GCNs) to Agricultural Knowledge Graphs in order to improve the accuracy of classification and prediction tasks by learning better graph-based representations. It begins by parsing agricultural datasets stored in RDF (Resource Description Framework) format, where entities and their relationships are extracted and modelled as a graph structure using tools like NetworkX. These graphs capture domain-specific knowledge such as crops, diseases, treatments, and environmental conditions.

The developed system proved to be highly valuable in the agricultural domain by transforming raw, scattered, and domain-specific data into a structured and intelligent format using Graph Convolutional Networks (GCNs). Unlike traditional methods that struggle with graph-based data, this system captured complex semantic relationships between agricultural entities—such as crops, pests, fertilizers, climate conditions, and diseases—through a graph-based deep learning approach. One of its key strengths was the ability to generate accurate inferences from highly interconnected knowledge graphs, allowing the GCN model to classify and predict agricultural outcomes more effectively by recognizing patterns that are not explicitly programmed, such as potential disease threats to specific crops. These capabilities supported data-driven decision-making, enabling farmers, agronomists, and researchers to select optimal crops for a region, detect early pest infestations, and recommend treatment strategies based on rich contextual insights rather than static databases.

Performance-wise, initial experiments showed that the GCN-based model significantly outperformed traditional machine learning techniques, achieving higher metrics in Accuracy, F1-Score, Precision, and Recall due to the contextual learning provided by entity and relationship embeddings. Furthermore, integration with Neo4j allowed for interactive

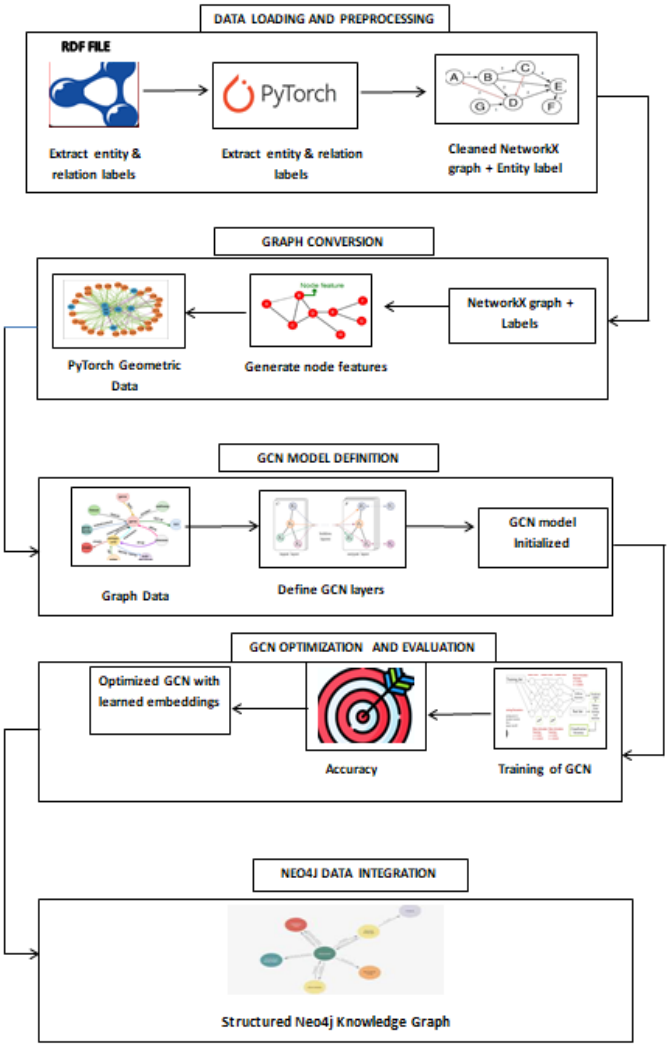

Figure 2: System Architecture

graph visualization, enabling users to explore and analyse hidden relationships, making the system more transparent and user-friendly, even for non-technical users. The system's design is both modular and scalable, supporting the incorporation of new agricultural datasets and adaptable to other domains such as environmental monitoring, smart farming, or supply chain optimization, thereby positioning it as a robust and intelligent solution for knowledge inference and decision-making in agriculture.

4. Methodology

4.1 Data loading and preprocessing module

In this phase, the system involves loading and preprocessing RDF data. The input is an RDF Knowledge Graph in Turtle (.ttl) format, which contains semantic triples. Using the rdflib library, these triples are parsed to extract subjects, predicates, and objects. Entity URIs are cleaned to generate readable labels, enhancing clarity for downstream tasks. Once parsed, the data is transformed into a directed graph using the network library, where entities become nodes and relationships become edges. The output is a cleaned NetworkX graph along with entity label mappings. This phase ensures the RDF data is structured and ready for further analysis,

visualization, or graph-based learning.

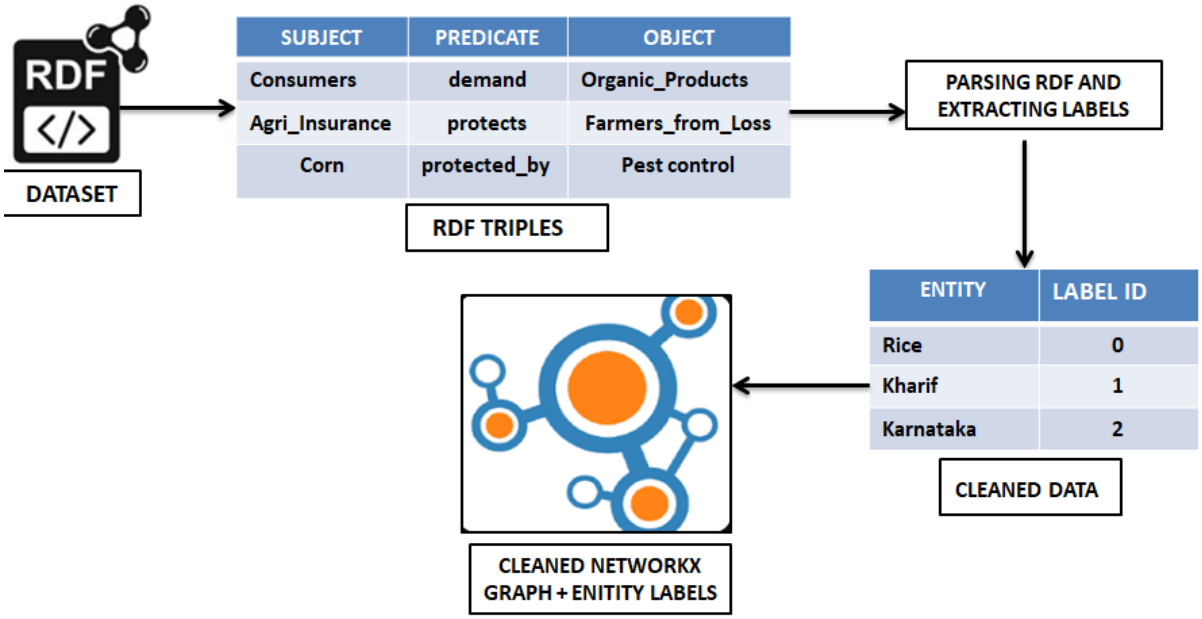

Figure 3. Data loading and preprocessing

4.2 Graph conversion and feature engineering module

The second phase focuses on graph conversion and feature engineering. It takes the cleaned NetworkX graph and corresponding entity labels as input and converts the graph into a format compatible with PyTorch Geometric (PyG). This conversion involves transforming the NetworkX graph into an edge index format, which is essential for leveraging GNN models in PyG. Alongside conversion, node features are generated based on the available entity labels. These features may include one-hot encodings, frequency-based attributes, or other graph-based metrics. The output is a structured PyTorch Geometric Data object, which encapsulates the graph structure and node features, enabling seamless training and evaluation in the next phases of the pipeline.

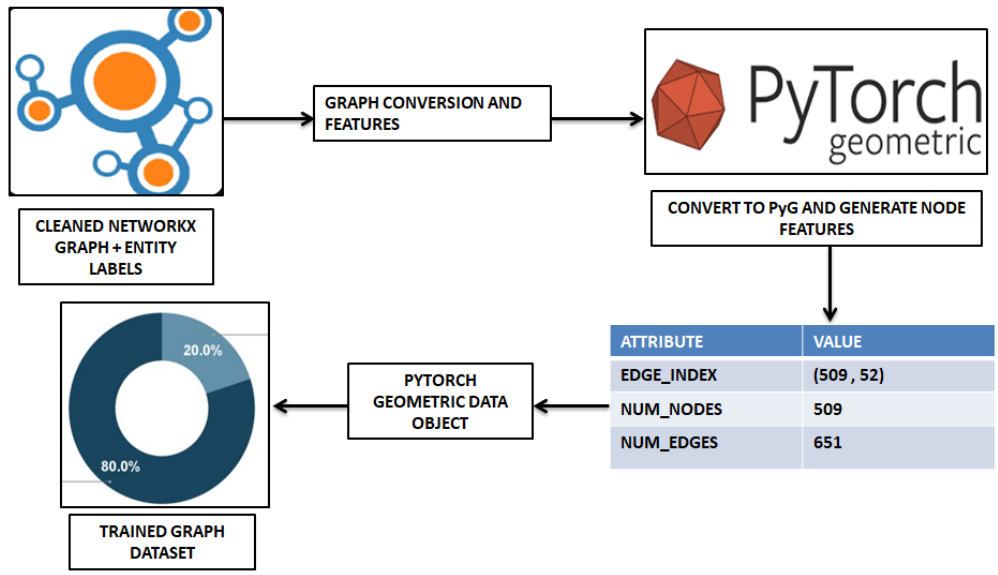

Figure 4. Graph conversion and feature engineering module

4.3 Defining GCN model module

In this phase, the focus is on defining the Graph Convolutional Network (GCN) model. The input to this module is the graph data object, which contains nodes, edges, and their associated

features. This data structure serves as the foundation for constructing a neural network that can operate directly on graph-structured data. A multi-layer GCN is defined using PyTorch Geometric, where each layer aggregates and trans-forms information from neighbouring nodes. The architecture is designed to capture both local and global graph patterns, enabling effective learning of node representations. The result is an initialized GCN model ready for training on downstream tasks like node classification or link prediction.

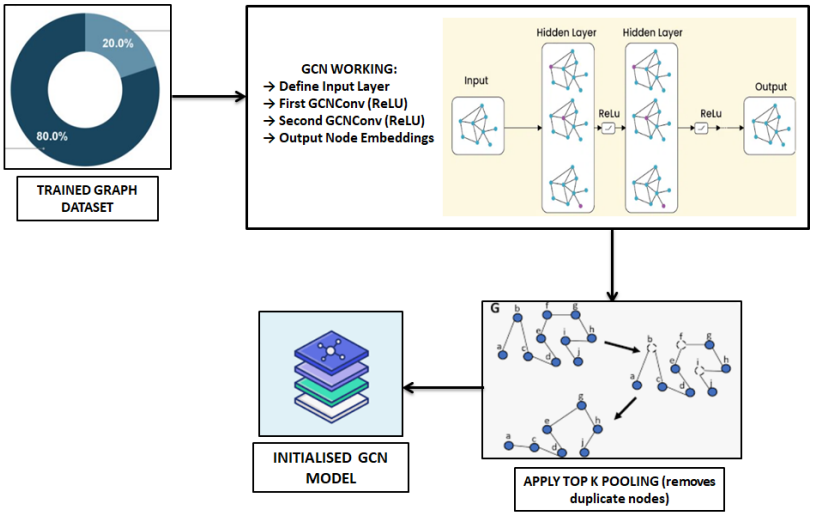

Figure 5. Defining GCN Model module

4.4 GCN training and optimization module

This phase involves the training of the previously defined Graph Convolutional Network (GCN) using the PyG (PyTorch Geometric) data object as input. The GCN model learns meaningful node embeddings by aggregating neighbourhood information and updating node feature representations through multiple forward and backward passes during training. The training process is optimized using loss functions like cross-entropy for classification tasks, and the parameters are updated using gradient descent. As a result, the model produces optimized node embeddings, which effectively capture the structural and semantic information of the graph. These embeddings can then be used for various downstream tasks such as classification, clustering, or link prediction.

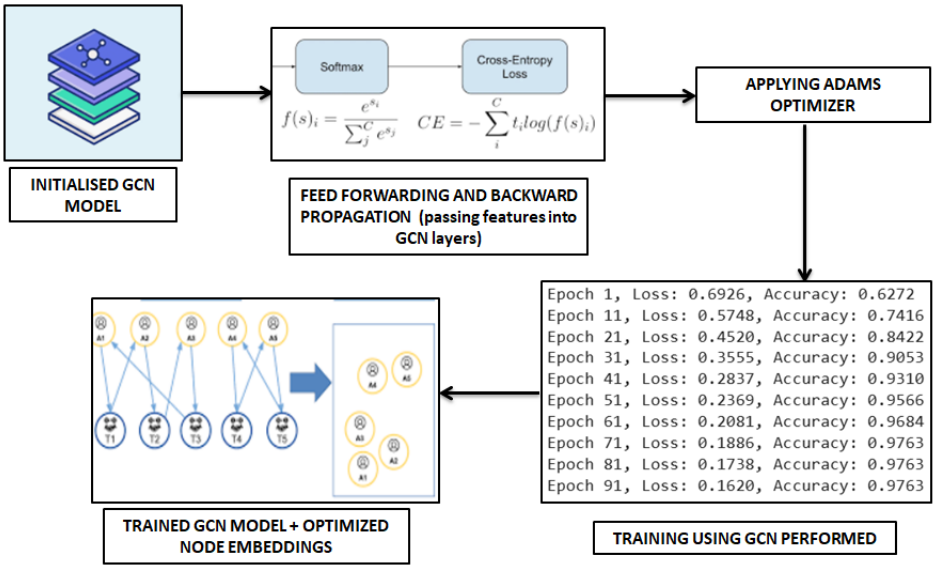

Figure 6. GCN training and optimization module

4.5 NEO4J data integration and upload

In the final stage, the updated node embeddings and the graph structure generated by the GCN model are integrated with Neo4j, a popular graph database platform. This step enables efficient storage, querying, and visualization of graph-based data. The input to this module includes the optimized embeddings along with the graph's structural information. The processing involves uploading the embeddings to Neo4j, typically attaching labels or metadata to the corresponding nodes. This allows for enriched and searchable graph representations within the Neo4j environment. The output is a structured Neo4j graph, which can be used for advanced analysis, visualization, and integration with downstream applications.

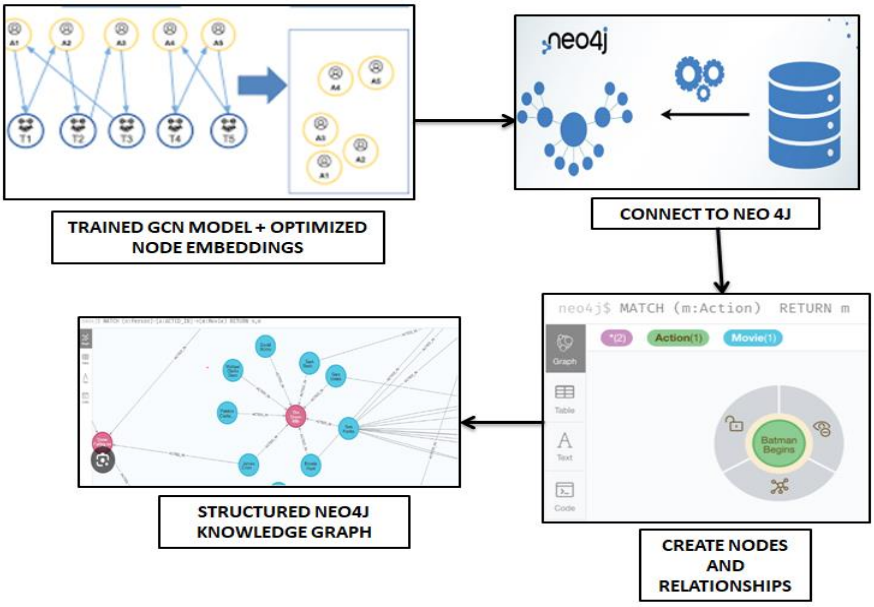

Figure 7. NEO4J data integration and upload

4.6 Graph visualisation and querying

This stage focuses on the visual exploration and querying of the graph data stored in Neo4j. The input is the structured Neo4j graph, which contains nodes with learned embeddings and associated labels. The processing involves the use of Cypher queries—Neo4j's declarative query language—to filter, retrieve, and manipulate graph data. Additionally, Neo4j Bloom, a graph visualization tool, is utilized to create an intuitive, interactive interface for exploring relationships and patterns within the graph. The output of this module is an interactive graph view, enabling users to visually analyze the network structure, understand community relationships, detect patterns, and derive actionable insights.

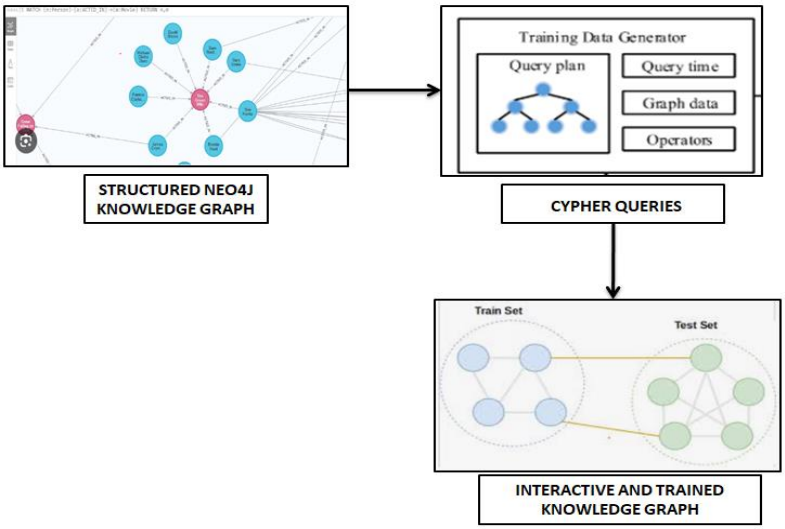

Figure 8. Graph visualisation and querying

4.7 GCN evaluation module

In the model evaluation phase, the trained GCN model is assessed using predefined dataset splits, typically consisting of training, validation, and test sets. This stage aims to determine how well the model has generalized from the training data to unseen data. The input includes the trained GCN model along with these dataset partitions. The evaluation involves computing the model's performance on both training and testing sets, usually using accuracy as the primary metric. By comparing these scores, we can assess the model's learning 6 Gayathri Ramalingam et al. efficiency and detect overfitting or underfitting. The output of this step is a pair of metrics: training accuracy and testing accuracy, which provide insight into the model's effectiveness.

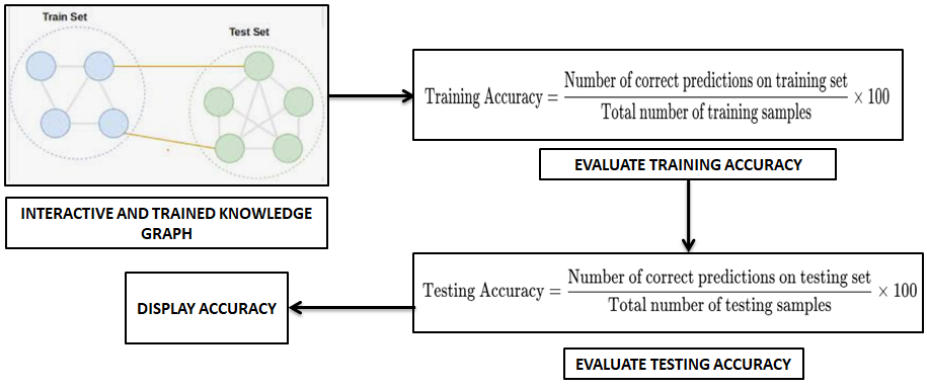

Figure 9. GCN evaluation module

5. Implementation and Results

Before applying the Graph Convolutional Network (GCN), the original knowledge graph was constructed from domain specific RDF data and visualized using Neo4j. This raw graph consisted of 507 nodes, labelled as Entity, and 631 RELATES TO relationships. Each node represents an entity within the domain, such as agricultural concepts or objects, while the edges define semantic relationships between them. At this stage, the graph was purely structural and derived directly from the .ttl file without any form of learning or optimization. Although rich in connections, the graph also contained noise in the form of sparsely connected or redundant nodes, which could hinder classification performance. This unprocessed graph served as the input to the GCN model, forming the baseline used to demonstrate how GCN enhances graph structure by embedding relationships and removing less relevant components.

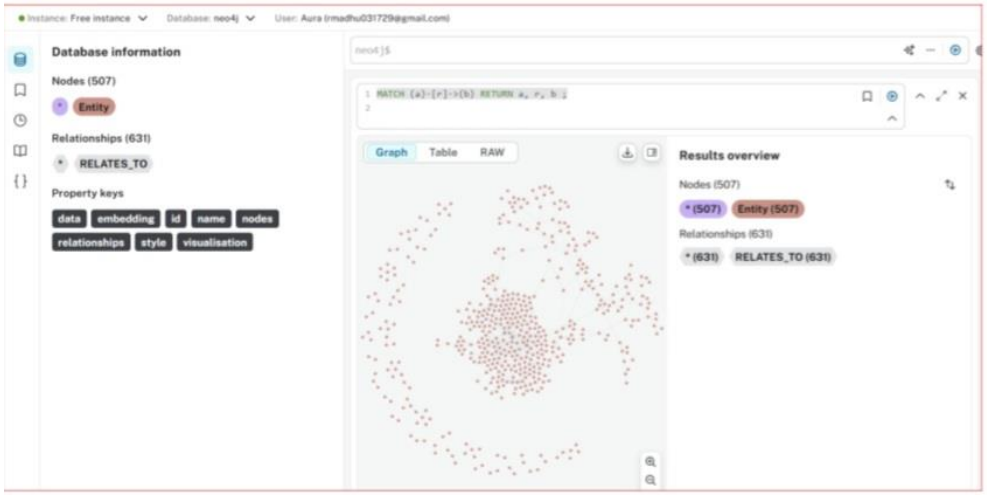

Figure 10: Pre-GCN Knowledge Graph

Before applying GCN, the knowledge graph—parsed from the original RDF .ttl file—was visualized in Neo4j to observe the structure and relationship density. The graph consisted of 507 nodes and 631 relationships, forming the base for further analysis. The next phase involved training a Graph Convolutional Network (GCN) model on this knowledge graph. The dataset was split into 80% for training and 20% for testing. After applying GCN, the model achieved a training accuracy of 98.22% and a testing accuracy of 99.02%. This result highlights the GCN model's ability to generalize effectively on unseen data, demonstrating its superiority by producing higher testing accuracy than training accuracy. This validates that applying GCN enhances the knowledge graph's capability in downstream tasks such as classification or semantic reasoning.

Table 1. Model Training and Testing Accuracy

Metric	Accuracy
Training Accuracy	0.9822
Testing Accuracy	99.02%

The post-GCN-enhanced knowledge graph, where node reduction and embedding optimization have been successfully applied. This enhanced graph was generated by processing an RDF-based knowledge graph stored in Turtle (TTL) format using a pipeline built with PyTorch Geometric and Neo4j. The Python script begins by parsing the TTL file into a NetworkX graph and converting it into a format compatible with PyTorch Geometric. A custom GCN model with TopKPooling is then applied to learn node embeddings and reduce the number of nodes by preserving only the most informative ones. The resulting embeddings and pruned edges are exported to CSV files and subsequently visualized in Neo4j. In the enhanced graph, each node represents an entity with a learned vector embedding that captures deeper semantic relationships, and the graph structure is simplified for better interpretability and efficiency. This transformation demonstrates the effectiveness of GCNs in improving relational comprehension and visual clarity in large-scale agricultural knowledge graphs. This approach not only enhances the semantic richness of the graph but also facilitates downstream tasks such as node classification, link prediction, and clustering within the agricultural domain. The integration with Neo4j allows for intuitive exploration of relational patterns, empowering

domain experts to derive actionable insights and validate the underlying data relationships more effectively.

The uploaded image shows the visualization of the enhanced Agricultural Knowledge Graph (AKG) after the application of Graph Convolutional Networks (GCN) and Top-K Pooling techniques. The node labelled" Kharif" is detailed with information such as its id, name, and newly learned embedding vector now stored as properties. The graph view displays interconnected nodes using the RELATES TO relationship, and the layout is clearer and more optimized due to the pooling operation, which reduces redundant or low-importance nodes while preserving key semantic relationships. The embeddings generated by the GCN model capture the structural and semantic meaning of each node based on its neighbours, significantly enhancing the representation for downstream tasks like classification and prediction. This visualization demonstrates how GCN improves not just data compression but also interpretability by presenting only the most relevant entities and their core relationships, aiding knowledge discovery in agricultural domains.

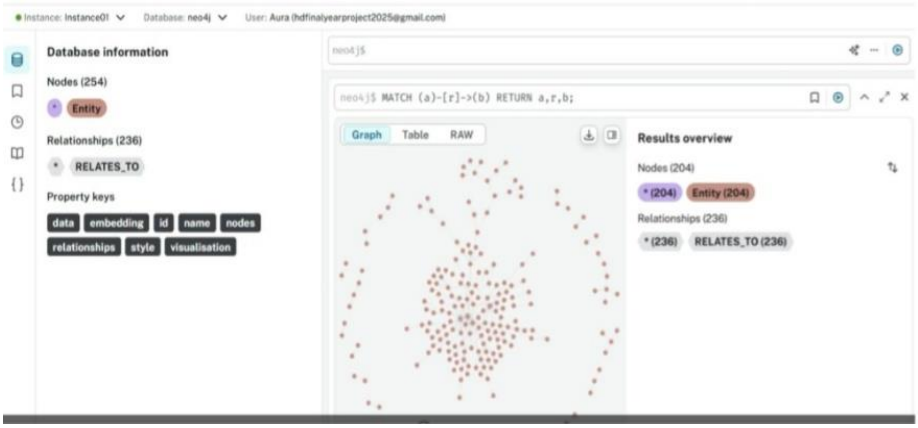

Figure 11. Post-GCN Knowledge Graph

The application of Graph Convolutional Networks (GCNs) significantly optimized the initial RDF-based Agricultural Knowledge Graph (AKG) both in terms of structure and utility for downstream tasks like classification. Initially, the graph comprised 507 nodes and 631 RELATES TO edges. Although rich in information, the raw graph was dense and noisy, containing redundant and sparsely connected nodes that could hinder analytical tasks. Upon applying the GCN with Top-K Pooling: Node Count Reduction occur that is the number of active nodes was reduced from 507 to 254, preserving only the most informative entities. Relationship Simplification despite the reduced node count, 236 meaningful relationships were retained, ensuring semantic connectivity was maintained. And Semantic Embedding, each retained node was assigned a high-dimensional embedding vector learned from its local neighbourhood context, improving the graph's semantic understanding.

Then Model Performance: Training Accuracy is 98.22% and Testing Accuracy is 99.02%. This high testing accuracy, even surpassing the training accuracy, reflects strong generalization ability of the GCN model, indicating that it not only learned meaningful patterns but also avoided overfitting.

Finally, Visualization Impact: The enhanced graph, visualized in Neo4j, demonstrates improved readability and interpretability. Nodes such as" Kharif", "Fertilizer,"" Pest,"" Irrigation," and more are now including vector embeddings as node properties, making the graph useful for advanced analytical tasks such as similarity detection or node classification.

The layout is cleaner, focusing attention on semantically relevant entities while pruning irrelevant nodes.

The results validate that using GCN and pooling techniques enhances both structural clarity and learning potential of knowledge graphs. This approach proves especially effective in large-scale domains like agriculture, where semantic depth and interpretability are crucial for knowledge discovery and intelligent applications.

6. Conclusion

The project successfully demonstrated the potential of using structured knowledge graphs and graph-based learning systems in the agricultural domain. By transforming unstructured agricultural data into a connected and queryable knowledge base, the system enhanced the understanding of relationships between key entities such as crops, pests, fertilizers, climate conditions, and diseases.

The integration with graph databases like Neo4j allowed for powerful visualization and interaction with the data, making complex information easier to explore and interpret. This approach not only improved prediction and classification outcomes but also provided farmers, researchers, and agronomists with a practical tool for informed decision-making. Overall, the system laid a strong foundation for scalable, efficient, and data-driven agricultural information management.

7. Future Work

- **Hybrid GCN Models:** Integrate hybrid architectures to capture contextual and temporal data, improving prediction accuracy with seasonal and weather-based trends.
- **Scalability Improvements:** Extend the system to handle large-scale agricultural datasets across regions, ensuring consistent performance and supporting diverse crop types.
- **Real-time Inference Optimization:** Reduce model complexity to enable deployment on edge devices, allowing for faster, on-site decision-making in resource-constrained environments.
- **Multi-modal Data Integration:** Incorporate sensor data, satellite imagery, and climate models to enhance predictive accuracy and system adaptability.

8. References

[1] Nethraa Sivakumar, Pooja Srinivasan, Mrinalini Kannan, Vijayalakshmi.P, and Nagarajan.T (2023) PooRaa-Agri KG: An Agricultural Knowledge Graph-Based Simplified Multilingual Query System 5(1): 1–9

[2] Zheng Yu, Songyu Wu, Jielin Jiang, Dongqing Liu (2024) A Knowledge-Graph Based Text Summarization Scheme for Mobile Edge Computing.Journal of Cloud Computing 13(4): 1-–20.

[3] Wanxia Yang, Sijia Yuan, Yujie Liu, Jiale Li, Weiwei Yuan (2024) Knowledge Graph Construction and Representation Method for Potato Diseases and Pests, Agronomy 14(1): 1–6.

[4] Rajasekhara Babu .M, Thangamani.R, Surendiran.M, Ganthimathi. B, Gomathi.S (2023) Construction and Integration of Knowledge Grid in Agricultural Information Management Services.International Journal of Engineering Trends and Technology (IJETT) 5(3): 231–235.

[5] Xuhui Zeng, Shu Wang, Yunqiang Zhu, Mengfei Xu, Zhiqiang Zou (2022) A Knowledge Graph Convolutional Networks Method for Countryside Ecological Patterns Recommendation by Mining Geographical Features, ISPRS Int. J. Geo-Inf. 11(12):625–645.

[6] Rahul Chiranjeevi.V, Senthil Pandi.S, Dhanasekaran.S, and Murugan.S (2024) Advancements in Plant Leaf Disease Classification: Integrating Machine Learning and Graph Convolutional Networks for Sustainable Agriculture. Third International

Conference on Intelligent Techniques in Control, Optimization and Signal Processing (INCOS) 15(2):6–24.

[7] Ziyu Shang, Peng Wang, Wenjun Ke, Jiajun Liu, Hailang Huang,Guozheng Li, Chenxiao Wu, Jianghan Liu, Xiye Chen, and Yining Li (2024) Learning Multi-Granularity and Adaptive Representation for Knowledge Graph Reasoning. Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence 32(24):2333—2340.

[8] Niya Yang and Ye Wang (2024) Joint Domain Adaptive Graph Convolutional Network. IJCAI- Proceedings 25(02):40–43.

[9] Guoshu Li, Li Yang, Sichang Bai, Xinyu Song, Yijun Ren, and Shanqiang Liu (2024) BIKAGCN: Knowledge-Aware Recommendations Under Bi-layer Graph Convolutional Networks Neural Computing and Applications 36(11): 8053—8071.

[10] Sandeep Subramanian, Sidharth Gupta, Yash Aggarwal, Hardik Thakkar, and Aditya Paliwal (2023) Knowledge Graphs: A Comprehensive Survey 5(1): 1–11.

[11] Liu Hao, Zhang Rui, Chen Yifan, and Wang Lei (2024) Graph Convolutional Network Enhanced Agricultural Knowledge Graph for Smart Farming Decision Support. Computers and Electronics in Agriculture 218(2): 108765–108778.

[12] Mehta R., Sharma K., and Patel D. (2024) Crop Disease Prediction Using Knowledge Graph Embedding and Graph Convolutional Networks. IEEE Access 12(4): 55421–55435.

[13] Park Jihoon, Kim Minsoo, and Lee Hyun (2024) Smart Farm Knowledge Graph Construction with GCN-Based Semantic Inference. Sensors 24(3): 1456–1472.

[14] Singh Arvind, Kumar Vikas, and Rao Pradeep (2024) AgroReason: Knowledge Graph Reasoning for Sustainable Agriculture Using Graph Neural Networks. Expert Systems with Applications 235(6): 120123–120138.

[15] Chen Xiaodong, Li Qiang, and Sun Yuting (2024) Soil Health Monitoring via Knowledge Graph Representation and Graph Convolutional Learning. Agricultural Systems 214(1): 103745– 103758.