Sigmoid 2025 Hackathon

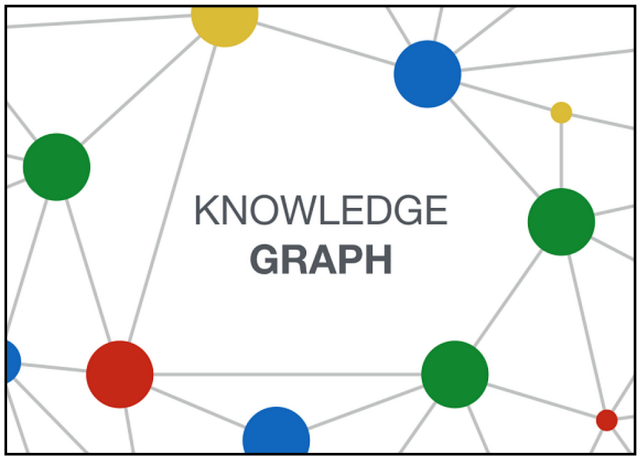
KNOWLEDGE GRAPHS FOR PRODUCT CLASSIFICATION

Team: RandomSeed42

Shatanshu Bodkhe
Mohit Doraiburu
Ayush Pathak
V.V. Vinay Kumar
Panshul Singhai

Index :

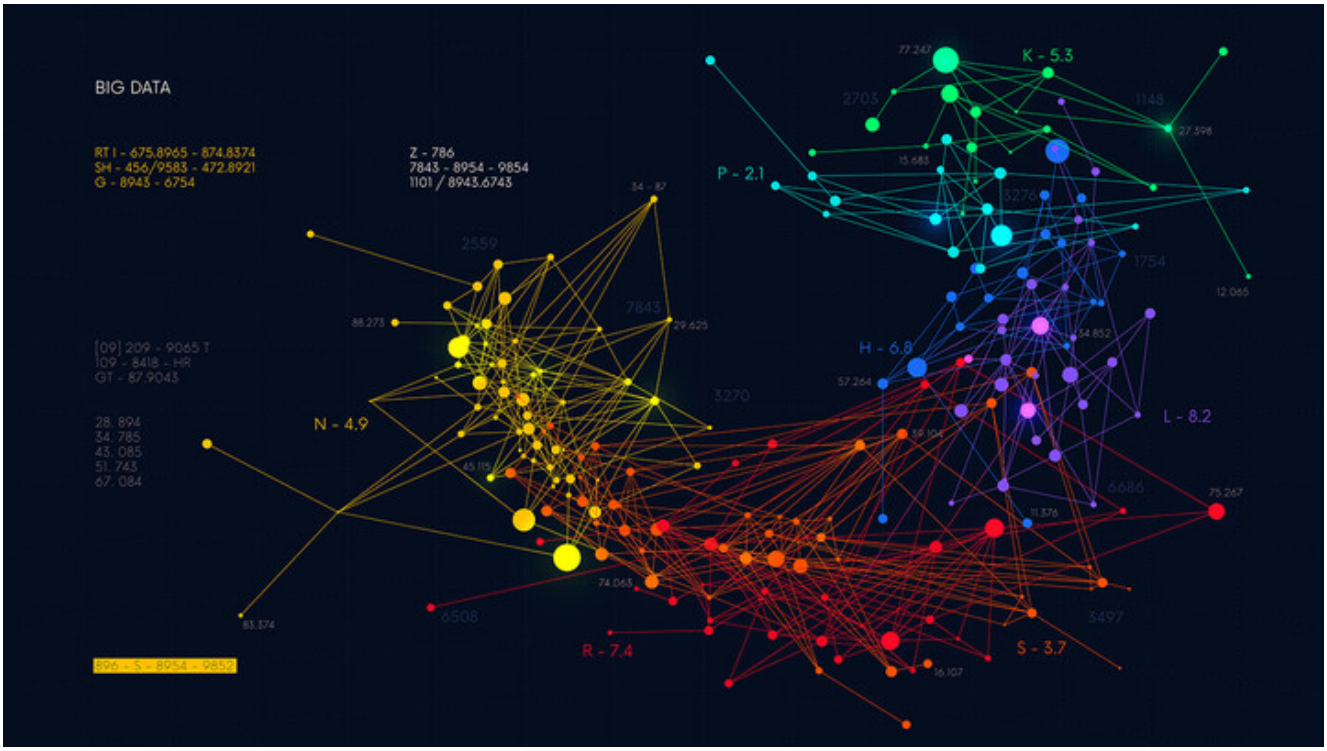
PROBLEM STATEMENT



"Elevating Product Classification game using Knowledge Graphs"

Product classification is a crucial task for e-commerce platforms, retailers, and supply chain management. Traditional classification methods rely on rigid taxonomies and keyword matching, leading to inefficiencies and inconsistencies. A knowledge graph-based approach enhances classification by capturing intricate relationships between products, attributes, and categories, enabling more intelligent and flexible organisation.

IMPORTANCE OF PRODUCT CLASSIFICATION



Accurate product classification improves search relevance, recommendation systems, and inventory management. It helps businesses streamline operations, enhance customer experience, and optimise marketing strategies. Knowledge graphs enable a more dynamic and context-aware classification, ensuring products are correctly categorised based on attributes like material, occasion, and style.

TOOLS AND TECHNOLOGIES

Programming Languages & Libraries

- **Python**: A versatile programming language widely used in data science, web development, and automation.
- **Pandas**: A Python library for data manipulation and analysis, offering powerful data structures like DataFrames.
- **tqdm**: A library for creating progress bars in Python, useful for tracking execution time in loops.

Databases & Graph Technology

- **Neo4j**: A graph database management system optimised for handling highly connected data with Cypher query language.
- **REE (Rust Execution Environment)**: A runtime for executing Rust-based applications efficiently with low-level performance benefits.

AI & Machine Learning

- **Gemini LLM**: A generative AI model by Google that excels in reasoning, text generation, and multimodal understanding.
- **google.generativeai**: A Python library for interacting with Google's Generative AI models, enabling advanced NLP and AI-driven applications.

Web Development & Deployment

- **Flask**: A lightweight Python web framework for building APIs and web applications quickly.
- **ReactJS**: A JavaScript library for creating interactive UIs with a component-based architecture.

DATA UNDERSTANDING AND DATASET PREPARATION

Detailed Data Cleaning Steps for the Flipkart Dataset

1. Understanding the Dataset

- **File Name:** Exhaustive cleaned flipkart_data.csv
- Total Rows: **20,000**
- Total Columns: **78**
- Key Observations:
  - Some columns contain a high percentage of missing values.
  - Certain columns have **mixed data types** (numeric, text, special characters).
  - **Redundant or irrelevant columns** may exist (e.g., unique identifiers).

2. Handling Missing Values

Missing values were found in multiple columns, handled as follows:

Column Name	Missing Values	Handling Strategy
brand	5,864	Filled with NaN
retail_price	78	Filled with NaN
fabric	13,752	Filled with NaN
pattern	6,942	Filled with NaN
sleeve	2,317	Filled with NaN
color	46	Filled with NaN
occasion	9,320	Filled with NaN

- **Numerical Columns:** Missing values replaced with **NaN or Attribute Deletion**.
- **Categorical Columns:** Missing values replaced with **"Unknown"** or the **Not a Number (NaN)**.

3. Data Type Standardisation

- Converted Numeric Columns to Floats:
  - `retail_price`, `discounted_price`, `ratings`, `reviews_count` converted from string to float.
- Converted Categorical Columns to Strings:
  - brand, colour, fabric, pattern, occasion, sleeve.
- Boolean Columns Standardised:
  - Converted "Yes"/"No" values into `True`/`False`.

4. Standardising Text Data

- **Trimmed Extra Spaces** in text-based columns.
- Converted all text to lowercase for consistency.
- **Removed special characters** from names and descriptions.
- **Corrected brand names** (e.g., "Nike Inc." → "Nike").

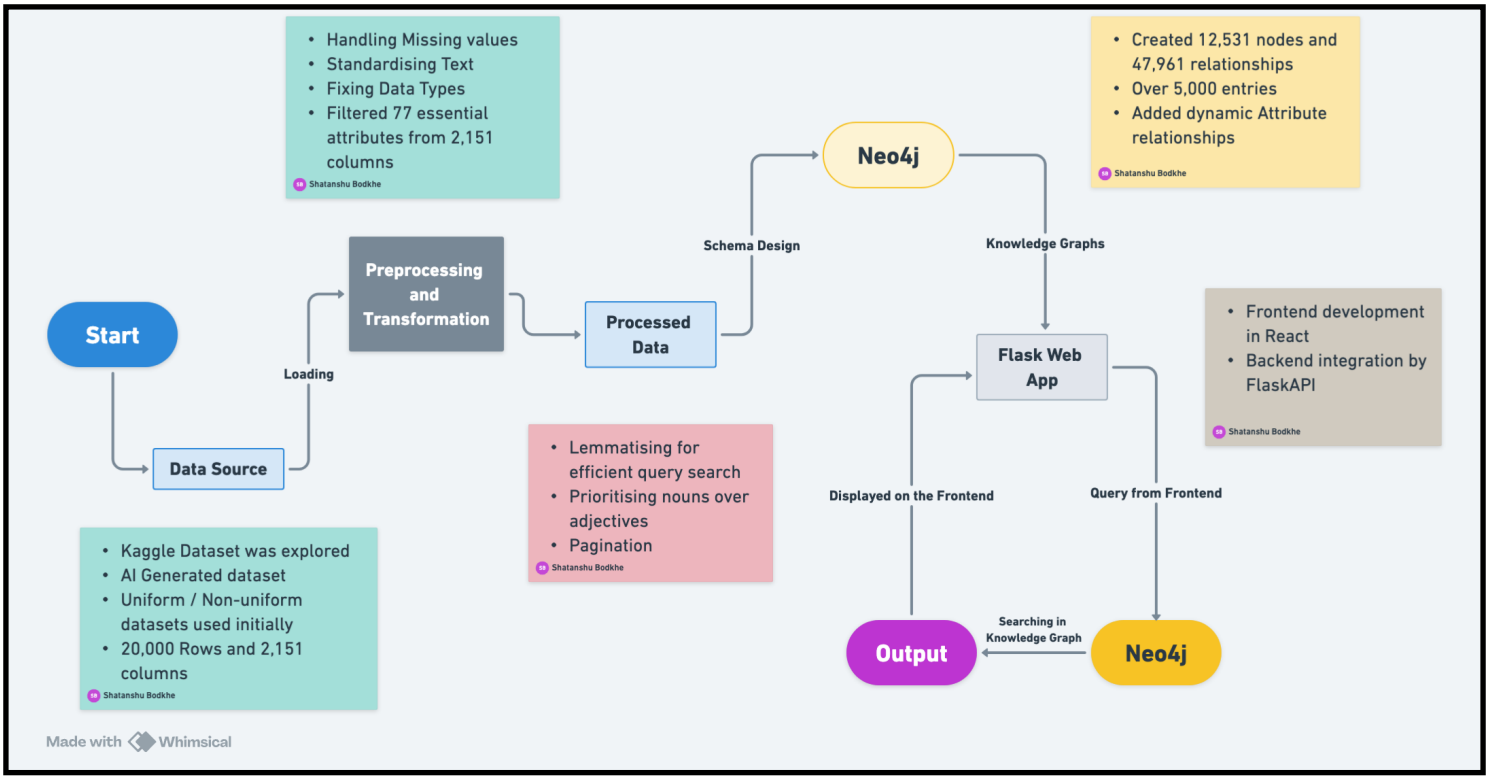
5. Handling Duplicates

- Checked for duplicate product entries based on `product_id`.
- Removed exact duplicates from the dataset.
- Merged similar product variations under one entry where possible.

WORKFLOW DESIGN

A well-defined workflow ensures seamless data ingestion, processing, and querying within the knowledge graph. The process begins with data collection from various sources, followed by cleaning and normalisation to maintain consistency. Next, relationship extraction links products to relevant attributes like material, fit, and occasion.

The structured data is then stored in a graph database, enabling efficient querying and updates. Finally, AI-powered classification and recommendations enhance the graph's utility, ensuring continuous learning and refinement. A modular workflow design ensures scalability, adaptability, and real-time product classification improvements.



Workflow Design for the Project

RELATIONSHIP EXTRACTION AND LINKING

Created 12,531 nodes and 47,961 relationships

The knowledge graph was constructed by identifying key entities such as products, brands, colours, and materials, ensuring each entity was represented as a distinct node.
Relationships were established between these nodes based on product attributes, enabling efficient traversal and retrieval of interconnected data.

Over 5,000 entries

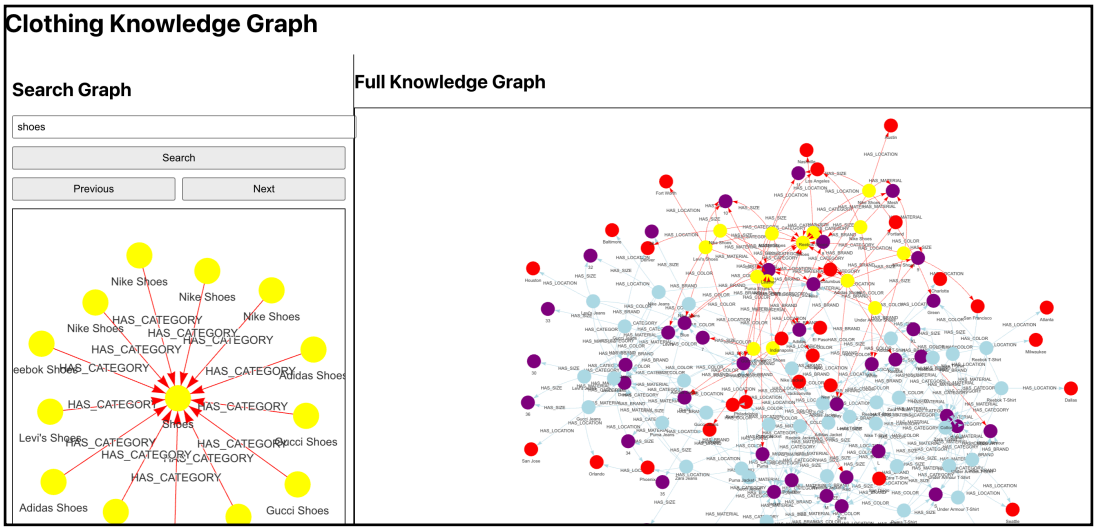
A comprehensive dataset was processed, resulting in a diverse collection of over 5,000 unique products linked to their respective categories, brands, and characteristics.
This extensive mapping facilitates precise search queries and enhances the accuracy of relationship-based recommendations.

Added dynamic attribute relationships

Instead of static predefined relationships, the graph was designed to accommodate dynamic attribute linking, allowing for more flexible and context-aware data exploration.
This enables adaptive querying, where new relationships can be inferred or updated based on evolving product attributes and user interactions.

Reduced the number of attributes

To enhance efficiency and relevance, the dataset was carefully refined by reducing the number of attributes from 2,151 to 77 essential features.
This selection process prioritised meaningful attributes such as product category, brand, colour, material, and occasion, ensuring a more structured and query-friendly knowledge graph while eliminating redundant or less significant data points.



A Knowledge Graph generated on a sample of 1,000 entries of Cloth Data

Initial tests :

On the left side, there is a **search panel** where users can input a query, such as "shoes," and retrieve relevant relationships. Below the search bar, a **focused knowledge graph** appears, displaying the category "Shoes" connected to various subcategories like **Nike Shoes, Adidas Shoes, Reebok Shoes, and Gucci Shoes**, represented as **yellow circular nodes**.

These relationships are labeled with "HAS_CATEGORY," illustrating how different brands fall under the broader "Shoes" category. Navigation buttons such as "Previous" and "Next" allow users to browse additional results.

On the right side, the **full knowledge graph** provides a **comprehensive visualisation** of the entire dataset. It consists of **various interconnected nodes** representing attributes like **colour, brand, size, material, and location**.

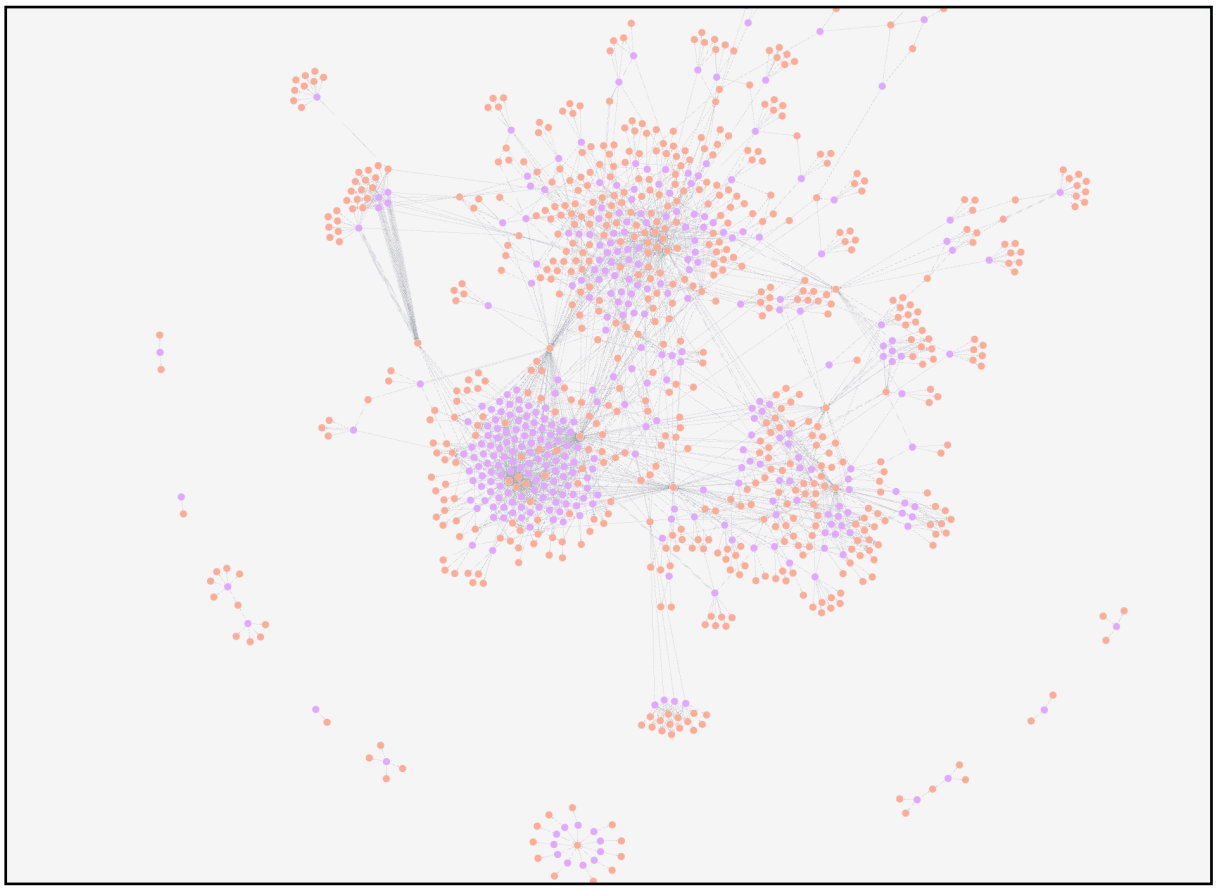
The **red, yellow, purple, and blue nodes** represent different categories, brands, and attributes, connected by edges labeled with relationships such as **HAS_COLOR, HAS_MATERIAL, HAS_SIZE, and HAS_CATEGORY**.

This extensive graph structure enables users to analyse patterns and relationships within the clothing dataset, making it easier to identify **similar products, related brands, and attribute-based classifications**.

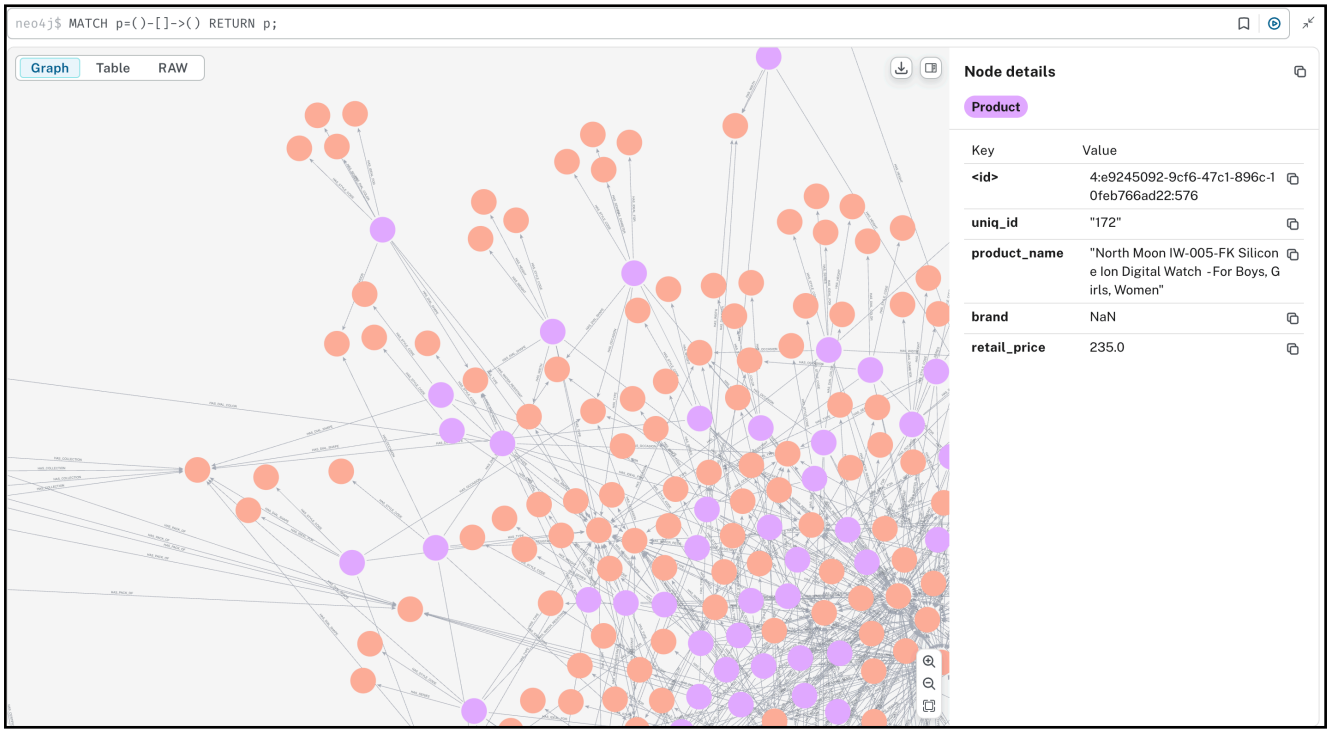
KNOWLEDGE GRAPHS AND QUERYING

We utilised **Neo4j** to generate the knowledge graph, enabling efficient storage and retrieval of product relationships. The querying system is integrated with a **Flask API**, which acts as an intermediary between the frontend and Neo4j.

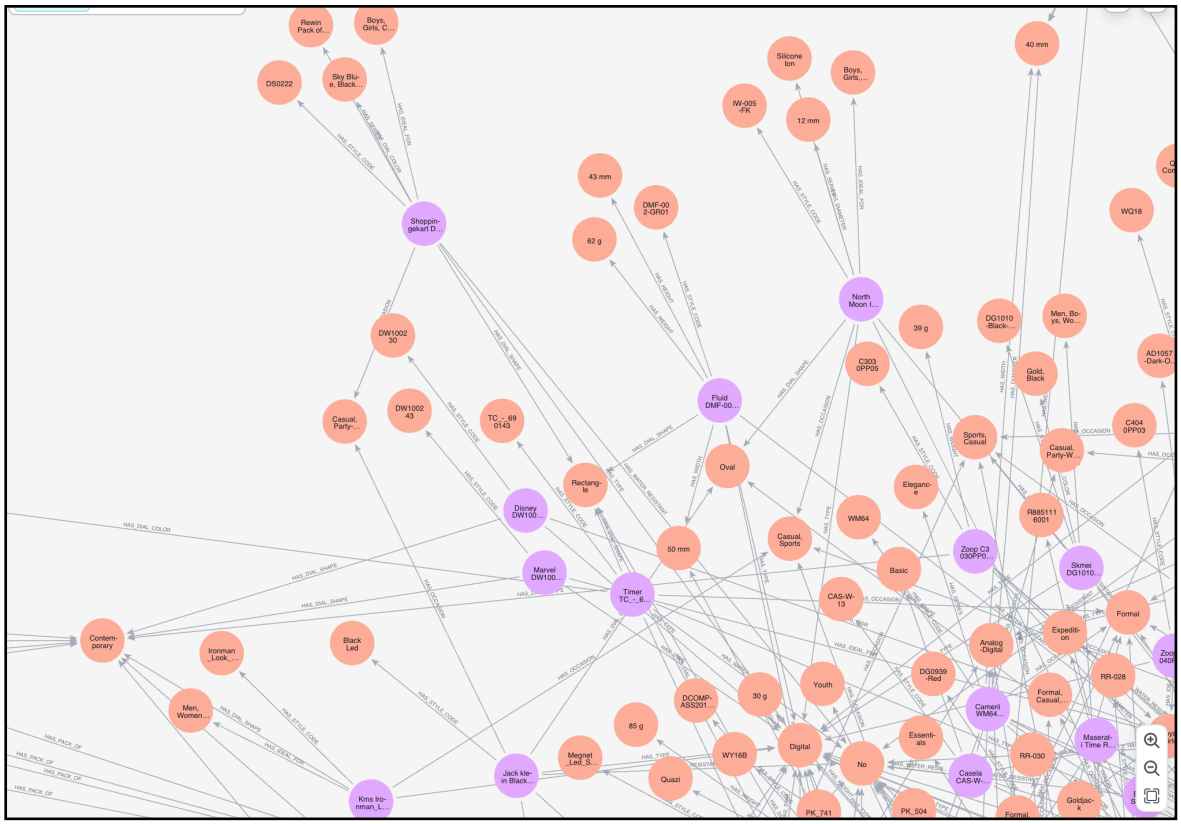
This setup ensures seamless communication and real-time responses to user queries. By leveraging a graph database, we can navigate complex relationships between products, attributes, and categories, allowing for precise and context-aware classification.



Overview of the Knowledge Graph



"Visualising a Knowledge Graph for Product Classification in Neo4j – Showcasing Relationships Between Products, Attributes, and Categories."



- **Query Processing** – Search inputs are broken down into nouns and adjectives, with nouns given higher priority for more relevant results.

- **Scalability** – The dataset includes 77 different attributes, allowing queries to range from highly specific (e.g., “lightweight cotton summer shirt”) to general (e.g., “casual wear”).

- **Comprehensive Filtering** – Attributes like warranty, pack size, model number, weight, warranty period, occasion, and design are incorporated for more refined searches.

- **Precision and Flexibility** – The system adapts to both broad and detailed queries, ensuring users get accurate and meaningful results.

- **Enhanced User Experience** – The approach creates a dynamic and intuitive search mechanism, improving product discovery and classification.

SUMMARY

Our query mechanism enhances product classification by breaking down search inputs into nouns and adjectives, prioritising nouns for relevance.

With **77 attributes**, queries can range from specific (e.g., “lightweight cotton summer shirt”) to general (e.g., “casual wear”), incorporating key factors like **warranty**, **pack size**, **model number**, **weight**, and **occasion** for comprehensive filtering.

This approach balances precision and flexibility, ensuring an intuitive and dynamic search experience.

Future Scope

As knowledge graphs evolve, they can further enhance AI-driven product classification by integrating machine learning models for context-aware recommendations.

Future improvements could include real-time updates, multi-modal search (text + image queries), and semantic understanding of customer preferences.

By leveraging graph embeddings and advanced **NLP** techniques, businesses can achieve more accurate classifications, improving personalisation and search efficiency in e-commerce and retail.

Source Code : https://github.com/AyushPathak2610/knowledge-graph-product-classification