Graph RAG from Scratch

November 7, 2024

1 How to Query a Knowledge Graph with LLMs using RAG

By Cristian Leo

Article on Towards Data Science

In this notebook, we will build a knowledge graph using a dataset of Amazon toy products. We will populate the knowledge graph with embeddings to enable semantic search capabilities. By the end of this tutorial, you will understand how to create a knowledge graph, add embeddings, and perform semantic searches to find products that match natural language queries.

1.1 Table of Contents

---

1. Introduction

Knowledge graphs are powerful tools for modeling complex, interconnected data. They allow us to represent entities (nodes) and the relationships (edges) between them in a flexible and semantically rich way. In contrast to traditional relational databases, knowledge graphs explicitly store relationships, making querying more efficient and intuitive for complex data structures.

In this workshop, we'll explore how to build a knowledge graph from a real-world dataset and enhance it with embeddings to perform semantic searches. We'll use Neo4j, a graph database platform, to store and query our graph.

1.2 2. Setting Up the Environment

Before we begin, ensure you have the necessary tools installed:

- **Clone the Repository**: The dataset and code are available in the GitHub repository rag-knowledge-graph. Clone this repository to your local machine:

  ```
  git clone https://github.com/cristianleoo/rag-knowledge-graph.git
  ```

- **Install Neo4j**: Download and install Neo4j from the official website. Follow the installation instructions specific to your operating system.

- **Start the Neo4j Server**: Once installed, start the Neo4j server. You can do this via the Neo4j Desktop application or by running the server from the command line:

  ```
  neo4j start
  ```

- **Install Required Python Libraries**: Navigate to the cloned repository directory and set up a virtual environment. Install the required libraries using `pip`:

  ```
  cd rag-knowledge-graph
  python -m venv venv
  source venv/bin/activate  # On Windows, use venv\Scripts\activate
  pip install -r requirements.txt
  ```

1.3 3. Loading and Preprocessing the Dataset

First, we'll import the necessary libraries and load the dataset.

[1]:
```
import pandas as pd
pd.set_option('display.max_columns', None)

from IPython.display import display
import pandas as pd
import matplotlib.pyplot as plt
import networkx as nx

from py2neo import Graph, Node, Relationship

import google.generativeai as genai
import time
from tqdm import tqdm
from ratelimit import limits, sleep_and_retry

import os
```

```
/Users/cristianleo/Documents/Documents - Cristian's Laptop/GitHub/rag-knowledge-
graph/venv/lib/python3.12/site-packages/tqdm/auto.py:21: TqdmWarning: IProgress
not found. Please update jupyter and ipywidgets. See
https://ipywidgets.readthedocs.io/en/stable/user_install.html
  from .autonotebook import tqdm as notebook_tqdm
```

1.3.1 Loading the Dataset

[2]:
```
df = pd.read_csv('dataset/products.csv')
df.head()
```

[2]:

	uniq_id	product_name	manufacturer	price	number_available_in_stock	number_of_reviews	number_of_answered_questions	average_review_rating	amazon_category_and_sub_category	customers_who_bought_this_item_also_bought	description
0	eac7efa5dbd3d667f26eb3d3ab504464	Hornby 2014 Catalogue	Hornby	£3.42	5 new	15	1.0	4.9 out of 5 stars	Hobbies > Model Trains & Railway Sets > Rail V…	http://www.amazon.co.uk/Hornby-R8150-Catalogue…	Product Description Hornby 2014 Catalogue Box …
1	b17540ef7e86e461d37f3ae58b7b72ac	FunkyBuys® Large Christmas Holiday Express Fes…	FunkyBuys	£16.99	NaN	2	1.0	4.5 out of 5 stars	Hobbies > Model Trains & Railway Sets > Rail V…	http://www.amazon.co.uk/Christmas-Holiday-Expr…	Size Name:Large FunkyBuys® Large Christmas Hol…
2	348f344247b0c1a935b1223072ef9d8a	CLASSIC TOY TRAIN SET TRACK CARRIAGES LIGHT EN…	ccf	£9.99	2 new	17	2.0	3.9 out of 5 stars	Hobbies > Model Trains & Railway Sets > Rail V…	http://www.amazon.co.uk/Classic-Train-Lights-B…	BIG CLASSIC TOY TRAIN SET TRACK CARRIAGE LIGHT…
3	e12b92dbb8eaee78b22965d2a9bbbd9f	HORNBY Coach R4410A BR Hawksworth Corridor 3rd	Hornby	£39.99	NaN	1	2.0	5.0 out of 5 stars	Hobbies > Model Trains & Railway Sets > Rail V…	NaN	Hornby 00 Gauge BR Hawksworth 3rd Class W 2107…
4	e33a9adeed5f36840ccc227db4682a36	Hornby 00 Gauge 0-4-0 Gildenlow Salt Co. Steam…	Hornby	£32.19	NaN	3	2.0	4.7 out of 5 stars	Hobbies > Model Trains & Railway Sets > Rail V…	http://www.amazon.co.uk/Hornby-R6367-RailRoad-…	Product Description Hornby RailRoad 0-4-0 Gild…

```
                                product_information  \
0  Technical Details Item Weight640 g Product Dim…
1  Technical Details Manufacturer recommended age…
2  Technical Details Manufacturer recommended age…
3  Technical Details Item Weight259 g Product Dim…
4  Technical Details Item Weight159 g Product Dim…

                                product_description  \
0  Product Description Hornby 2014 Catalogue Box …
1  Size Name:Large FunkyBuys® Large Christmas Hol…
2  BIG CLASSIC TOY TRAIN SET TRACK CARRIAGE LIGHT…
3  Hornby 00 Gauge BR Hawksworth 3rd Class W 2107…
4  Product Description Hornby RailRoad 0-4-0 Gild…

      items_customers_buy_after_viewing_this_item  \
0  http://www.amazon.co.uk/Hornby-R8150-Catalogue…
1  http://www.amazon.co.uk/Christmas-Holiday-Expr…
2  http://www.amazon.co.uk/Train-With-Tracks-Batt…
3                                              NaN
4  http://www.amazon.co.uk/Hornby-R2672-RailRoad-…

                    customer_questions_and_answers  \
0  Does this catalogue detail all the previous Ho…
1  can you turn off sounds // hi no you cant turn…
2  What is the gauge of the track // Hi Paul.Trut…
3                                              NaN
4                                              NaN

                                  customer_reviews  \
0  Worth Buying For The Pictures Alone (As Ever) …
1  Four Stars // 4.0 // 18 Dec. 2015 // By\n    \…
2  **Highly Recommended!** // 5.0 // 26 May 2015 …
3  I love it // 5.0 // 22 July 2013 // By\n    \n…
4  Birthday present // 5.0 // 14 April 2014 // By…

                                             sellers
0  {"seller"=>[{"Seller_name_1"=>"Amazon.co.uk", …
1  {"seller"=>{"Seller_name_1"=>"UHD WHOLESALE", …
2  {"seller"=>[{"Seller_name_1"=>"DEAL-BOX", "Sel…
3                                              NaN
4                                              NaN
```

1.3.2 Data Overview

Let's examine the dataset to understand its structure and identify missing values.

[3]:
```
for col in df.columns:
    print(f"{col:<50} | {df[col].isna().sum() / len(df):>6.2%} missing |␣
 ↪{df[col].nunique():>6} unique values | {df[col].dtype}")
```

```
uniq_id                                            |  0.00% missing |  10000
unique values | object
product_name                                       |  0.00% missing |   9964
unique values | object
manufacturer                                       |  0.07% missing |   2651
unique values | object
price                                              | 14.35% missing |   2625
unique values | object
number_available_in_stock                          | 25.00% missing |     89
unique values | object
number_of_reviews                                  |  0.18% missing |    194
unique values | object
number_of_answered_questions                       |  7.65% missing |     19
unique values | float64
average_review_rating                              |  0.18% missing |     19
unique values | object
amazon_category_and_sub_category                   |  6.90% missing |    255
unique values | object
customers_who_bought_this_item_also_bought         | 10.62% missing |   8755
unique values | object
description                                        |  6.51% missing |   8514
unique values | object
product_information                                |  0.58% missing |   9939
unique values | object
product_description                                |  6.51% missing |   8514
unique values | object
items_customers_buy_after_viewing_this_item        | 30.65% missing |   6749
unique values | object
customer_questions_and_answers                     | 90.86% missing |    910
unique values | object
customer_reviews                                   |  0.21% missing |   9901
unique values | object
sellers                                            | 30.82% missing |   6581
unique values | object
```

1.3.3 Data Cleaning and Preprocessing

We perform data cleaning by extracting useful information and handling missing values.

[4]:
```
# Extract currency symbol and price into separate columns
df['currency'] = df['price'].str.extract(r'([^0-9]+)')
df['price_value'] = df['price'].str.extract(r'(\d+\.?\d*)').astype(float)

df['stock_type'] = df['number_available_in_stock'].str.extract(r'([^0-9]+)')
```

```
df['stock_availability'] = df['number_available_in_stock'].str.extract(r'(\d+\.?
 ↪\d*)')

df['average_review_rating'] = df['average_review_rating'].str.replace(' out of␣
 ↪5 stars', '').astype(float)
df['number_of_reviews'] = df['number_of_reviews'].str.replace(',', '').
 ↪fillna(0).astype(int)

df.head()
```

[4]:

```
                           uniq_id  \
0  eac7efa5dbd3d667f26eb3d3ab504464
1  b17540ef7e86e461d37f3ae58b7b72ac
2  348f344247b0c1a935b1223072ef9d8a
3  e12b92dbb8eaee78b22965d2a9bbbd9f
4  e33a9adeed5f36840ccc227db4682a36

                                     product_name manufacturer   price  \
0                            Hornby 2014 Catalogue       Hornby   £3.42
1  FunkyBuys® Large Christmas Holiday Express Fes…    FunkyBuys  £16.99
2  CLASSIC TOY TRAIN SET TRACK CARRIAGES LIGHT EN…          ccf   £9.99
3     HORNBY Coach R4410A BR Hawksworth Corridor 3rd       Hornby  £39.99
4  Hornby 00 Gauge 0-4-0 Gildenlow Salt Co. Steam…       Hornby  £32.19

  number_available_in_stock  number_of_reviews  number_of_answered_questions  \
0                     5 new                 15                           1.0
1                       NaN                  2                           1.0
2                     2 new                 17                           2.0
3                       NaN                  1                           2.0
4                       NaN                  3                           2.0

   average_review_rating                   amazon_category_and_sub_category  \
0                    4.9  Hobbies > Model Trains & Railway Sets > Rail V…
1                    4.5  Hobbies > Model Trains & Railway Sets > Rail V…
2                    3.9  Hobbies > Model Trains & Railway Sets > Rail V…
3                    5.0  Hobbies > Model Trains & Railway Sets > Rail V…
4                    4.7  Hobbies > Model Trains & Railway Sets > Rail V…

         customers_who_bought_this_item_also_bought  \
0  http://www.amazon.co.uk/Hornby-R8150-Catalogue…
1  http://www.amazon.co.uk/Christmas-Holiday-Expr…
2  http://www.amazon.co.uk/Classic-Train-Lights-B…
3                                               NaN
4  http://www.amazon.co.uk/Hornby-R6367-RailRoad-…

                                        description  \
0  Product Description Hornby 2014 Catalogue Box …
```

```
1  Size Name:Large FunkyBuys® Large Christmas Hol…
2  BIG CLASSIC TOY TRAIN SET TRACK CARRIAGE LIGHT…
3  Hornby 00 Gauge BR Hawksworth 3rd Class W 2107…
4  Product Description Hornby RailRoad 0-4-0 Gild…

                                product_information  \
0  Technical Details Item Weight640 g Product Dim…
1  Technical Details Manufacturer recommended age…
2  Technical Details Manufacturer recommended age…
3  Technical Details Item Weight259 g Product Dim…
4  Technical Details Item Weight159 g Product Dim…

                                product_description  \
0  Product Description Hornby 2014 Catalogue Box …
1  Size Name:Large FunkyBuys® Large Christmas Hol…
2  BIG CLASSIC TOY TRAIN SET TRACK CARRIAGE LIGHT…
3  Hornby 00 Gauge BR Hawksworth 3rd Class W 2107…
4  Product Description Hornby RailRoad 0-4-0 Gild…

      items_customers_buy_after_viewing_this_item  \
0  http://www.amazon.co.uk/Hornby-R8150-Catalogue…
1  http://www.amazon.co.uk/Christmas-Holiday-Expr…
2  http://www.amazon.co.uk/Train-With-Tracks-Batt…
3                                              NaN
4  http://www.amazon.co.uk/Hornby-R2672-RailRoad-…

                    customer_questions_and_answers  \
0  Does this catalogue detail all the previous Ho…
1  can you turn off sounds // hi no you cant turn…
2  What is the gauge of the track // Hi Paul.Trut…
3                                              NaN
4                                              NaN

                                  customer_reviews  \
0  Worth Buying For The Pictures Alone (As Ever) …
1  Four Stars // 4.0 // 18 Dec. 2015 // By\n    \…
2  **Highly Recommended!** // 5.0 // 26 May 2015 …
3  I love it // 5.0 // 22 July 2013 // By\n    \n…
4  Birthday present // 5.0 // 14 April 2014 // By…

                                           sellers currency  price_value  \
0  {"seller"=>[{"Seller_name_1"=>"Amazon.co.uk", …        £         3.42
1  {"seller"=>{"Seller_name_1"=>"UHD WHOLESALE", …        £        16.99
2  {"seller"=>[{"Seller_name_1"=>"DEAL-BOX", "Sel…        £         9.99
3                                              NaN        £        39.99
4                                              NaN        £        32.19
```

```
  stock_type stock_availability
0        new                 5
1        NaN               NaN
2        new                 2
3        NaN               NaN
4        NaN               NaN
```

Dropping Unnecessary Columns

[5]:
```
# Drop the original price column if you want
df = df.drop(['price', 'number_available_in_stock',␣
 ↪'customers_who_bought_this_item_also_bought',␣
 ↪'items_customers_buy_after_viewing_this_item',␣
 ↪'customer_questions_and_answers', 'sellers'], axis=1)
df.dropna(subset=['product_information', 'price_value', 'description',␣
 ↪'amazon_category_and_sub_category'], how='any', inplace=True)
df.head()
```

[5]:
```
                           uniq_id  \
0  eac7efa5dbd3d667f26eb3d3ab504464
1  b17540ef7e86e461d37f3ae58b7b72ac
2  348f344247b0c1a935b1223072ef9d8a
3  e12b92dbb8eaee78b22965d2a9bbbd9f
4  e33a9adeed5f36840ccc227db4682a36

                                        product_name manufacturer  \
0                             Hornby 2014 Catalogue       Hornby
1  FunkyBuys® Large Christmas Holiday Express Fes…    FunkyBuys
2  CLASSIC TOY TRAIN SET TRACK CARRIAGES LIGHT EN…          ccf
3     HORNBY Coach R4410A BR Hawksworth Corridor 3rd       Hornby
4  Hornby 00 Gauge 0-4-0 Gildenlow Salt Co. Steam…       Hornby

   number_of_reviews  number_of_answered_questions  average_review_rating  \
0                 15                          1.0                    4.9
1                  2                          1.0                    4.5
2                 17                          2.0                    3.9
3                  1                          2.0                    5.0
4                  3                          2.0                    4.7

                    amazon_category_and_sub_category  \
0  Hobbies > Model Trains & Railway Sets > Rail V…
1  Hobbies > Model Trains & Railway Sets > Rail V…
2  Hobbies > Model Trains & Railway Sets > Rail V…
3  Hobbies > Model Trains & Railway Sets > Rail V…
4  Hobbies > Model Trains & Railway Sets > Rail V…

                                        description  \
0  Product Description Hornby 2014 Catalogue Box …
```

```
1  Size Name:Large FunkyBuys® Large Christmas Hol…
2  BIG CLASSIC TOY TRAIN SET TRACK CARRIAGE LIGHT…
3  Hornby 00 Gauge BR Hawksworth 3rd Class W 2107…
4  Product Description Hornby RailRoad 0-4-0 Gild…

                                 product_information  \
0  Technical Details Item Weight640 g Product Dim…
1  Technical Details Manufacturer recommended age…
2  Technical Details Manufacturer recommended age…
3  Technical Details Item Weight259 g Product Dim…
4  Technical Details Item Weight159 g Product Dim…

                                 product_description  \
0  Product Description Hornby 2014 Catalogue Box …
1  Size Name:Large FunkyBuys® Large Christmas Hol…
2  BIG CLASSIC TOY TRAIN SET TRACK CARRIAGE LIGHT…
3  Hornby 00 Gauge BR Hawksworth 3rd Class W 2107…
4  Product Description Hornby RailRoad 0-4-0 Gild…

                                   customer_reviews currency  price_value  \
0  Worth Buying For The Pictures Alone (As Ever) …        £         3.42
1  Four Stars // 4.0 // 18 Dec. 2015 // By\n    \…        £        16.99
2  **Highly Recommended!** // 5.0 // 26 May 2015 …        £         9.99
3  I love it // 5.0 // 22 July 2013 // By\n    \n…        £        39.99
4  Birthday present // 5.0 // 14 April 2014 // By…        £        32.19

  stock_type stock_availability
0        new                 5
1        NaN               NaN
2        new                 2
3        NaN               NaN
4        NaN               NaN
```

[6]:
```
for col in df.columns:
    print(f"{col:<50} | {df[col].isna().sum() / len(df):>6.2%} missing |␣
↪{df[col].nunique():>6} unique values | {df[col].dtype}")
```

```
uniq_id                                            |  0.00% missing |   7434
unique values | object
product_name                                       |  0.00% missing |   7409
unique values | object
manufacturer                                       |  0.00% missing |   2025
unique values | object
number_of_reviews                                  |  0.00% missing |    180
unique values | int64
number_of_answered_questions                       |  7.68% missing |     19
unique values | float64
average_review_rating                              |  0.09% missing |     17
```

```
unique values | float64
amazon_category_and_sub_category                    |  0.00% missing |    230
unique values | object
description                                         |  0.00% missing |   6690
unique values | object
product_information                                 |  0.00% missing |   7432
unique values | object
product_description                                 |  0.00% missing |   6690
unique values | object
customer_reviews                                    |  0.12% missing |   7364
unique values | object
currency                                            |  0.00% missing |      1
unique values | object
price_value                                         |  0.00% missing |   2404
unique values | float64
stock_type                                          | 24.85% missing |      4
unique values | object
stock_availability                                  | 24.85% missing |     61
unique values | object
```

Handling Remaining Missing Values

[7]:
```
df['amazon_category_and_sub_category'] = df['amazon_category_and_sub_category'].
 ↪fillna('')
df['manufacturer'] = df['manufacturer'].fillna('Unknown')
df['number_of_answered_questions'] = df['number_of_answered_questions'].
 ↪fillna(0.0)
df['average_review_rating'] = df['average_review_rating'].fillna(0.0)
df['description'] = df['description'].fillna('')
df['product_description'] = df['product_description'].fillna('')
df['product_information'] = df['product_information'].fillna('')
df['customer_reviews'] = df['customer_reviews'].fillna('')
df['stock_availability'] = df['stock_availability'].astype(float).fillna(0.0)
df['stock_type'] = df['stock_type'].fillna('Out of stock')
```

Verifying the Cleaned Data

[8]:
```
for col in df.columns:
    print(f"{col:<50} | {df[col].isna().sum() / len(df):>6.2%} missing |␣
 ↪{df[col].nunique():>6} unique values | {df[col].dtype}")
```

```
uniq_id                                             |  0.00% missing |   7434
unique values | object
product_name                                        |  0.00% missing |   7409
unique values | object
manufacturer                                        |  0.00% missing |   2025
unique values | object
number_of_reviews                                   |  0.00% missing |    180
unique values | int64
```

```
number_of_answered_questions                    |  0.00% missing |     20
unique values | float64
average_review_rating                           |  0.00% missing |     18
unique values | float64
amazon_category_and_sub_category                |  0.00% missing |    230
unique values | object
description                                     |  0.00% missing |   6690
unique values | object
product_information                             |  0.00% missing |   7432
unique values | object
product_description                             |  0.00% missing |   6690
unique values | object
customer_reviews                                |  0.00% missing |   7365
unique values | object
currency                                        |  0.00% missing |      1
unique values | object
price_value                                     |  0.00% missing |   2404
unique values | float64
stock_type                                      |  0.00% missing |      5
unique values | object
stock_availability                              |  0.00% missing |     62
unique values | float64
```

1.3.4 Combining Product Information

We create a complete product description by combining several fields. This combined text will be useful for generating embeddings later.

[9]:
```
def complete_product_description(row):
    description = "Product Title: " + row['product_name'] + "\n"
    description += "Product Description: " + row['product_description'] + "\n"
    description += "Product Information: " + row['product_information'] + "\n"
    return description

df['description_complete'] = df.apply(complete_product_description, axis=1)
df.head()
```

[9]:

	uniq_id	product_name	manufacturer
0	eac7efa5dbd3d667f26eb3d3ab504464	Hornby 2014 Catalogue	Hornby
1	b17540ef7e86e461d37f3ae58b7b72ac	FunkyBuys® Large Christmas Holiday Express Fes…	FunkyBuys
2	348f344247b0c1a935b1223072ef9d8a	CLASSIC TOY TRAIN SET TRACK CARRIAGES LIGHT EN…	ccf
3	e12b92dbb8eaee78b22965d2a9bbbd9f		
4	e33a9adeed5f36840ccc227db4682a36		

```
3     HORNBY Coach R4410A BR Hawksworth Corridor 3rd       Hornby
4  Hornby 00 Gauge 0-4-0 Gildenlow Salt Co. Steam…       Hornby

   number_of_reviews  number_of_answered_questions  average_review_rating  \
0                 15                          1.0                    4.9
1                  2                          1.0                    4.5
2                 17                          2.0                    3.9
3                  1                          2.0                    5.0
4                  3                          2.0                    4.7

                    amazon_category_and_sub_category  \
0  Hobbies > Model Trains & Railway Sets > Rail V…
1  Hobbies > Model Trains & Railway Sets > Rail V…
2  Hobbies > Model Trains & Railway Sets > Rail V…
3  Hobbies > Model Trains & Railway Sets > Rail V…
4  Hobbies > Model Trains & Railway Sets > Rail V…

                                        description  \
0  Product Description Hornby 2014 Catalogue Box …
1  Size Name:Large FunkyBuys® Large Christmas Hol…
2  BIG CLASSIC TOY TRAIN SET TRACK CARRIAGE LIGHT…
3  Hornby 00 Gauge BR Hawksworth 3rd Class W 2107…
4  Product Description Hornby RailRoad 0-4-0 Gild…

                                 product_information  \
0  Technical Details Item Weight640 g Product Dim…
1  Technical Details Manufacturer recommended age…
2  Technical Details Manufacturer recommended age…
3  Technical Details Item Weight259 g Product Dim…
4  Technical Details Item Weight159 g Product Dim…

                                 product_description  \
0  Product Description Hornby 2014 Catalogue Box …
1  Size Name:Large FunkyBuys® Large Christmas Hol…
2  BIG CLASSIC TOY TRAIN SET TRACK CARRIAGE LIGHT…
3  Hornby 00 Gauge BR Hawksworth 3rd Class W 2107…
4  Product Description Hornby RailRoad 0-4-0 Gild…

                                    customer_reviews currency  price_value  \
0  Worth Buying For The Pictures Alone (As Ever) …        £         3.42
1  Four Stars // 4.0 // 18 Dec. 2015 // By\n    \…        £        16.99
2  **Highly Recommended!** // 5.0 // 26 May 2015 …        £         9.99
3  I love it // 5.0 // 22 July 2013 // By\n    \n…        £        39.99
4  Birthday present // 5.0 // 14 April 2014 // By…        £        32.19

     stock_type  stock_availability  \
0           new                5.0
```

```
1  Out of stock                 0.0
2           new                 2.0
3  Out of stock                 0.0
4  Out of stock                 0.0

                                description_complete
0  Product Title: Hornby 2014 Catalogue\nProduct …
1  Product Title: FunkyBuys® Large Christmas Holi…
2  Product Title: CLASSIC TOY TRAIN SET TRACK CAR…
3  Product Title: HORNBY Coach R4410A BR Hawkswor…
4  Product Title: Hornby 00 Gauge 0-4-0 Gildenlow…
```

1.4 4. Connecting to Neo4j and Preparing the Database

We establish a connection to the Neo4j graph database.

Note: Replace `"YOUR_PASSWORD"` with your actual Neo4j password

[10]:
```python
# Connect to Neo4j (adjust credentials as needed)
graph = Graph("bolt://localhost:7687", auth=("neo4j", "YOUR_PASSWORD"))

# Clear existing data (optional)
graph.run("MATCH (n) DETACH DELETE n")

def create_knowledge_graph(df):
    # Create unique constraints
    # For Neo4j 5.x and later
    try:
        graph.run("CREATE CONSTRAINT product_id IF NOT EXISTS FOR (p:Product)␣
↪REQUIRE p.uniq_id IS UNIQUE")
        graph.run("CREATE CONSTRAINT manufacturer_name IF NOT EXISTS FOR (m:
↪Manufacturer) REQUIRE m.name IS UNIQUE")
        graph.run("CREATE CONSTRAINT category_name IF NOT EXISTS FOR (c:
↪Category) REQUIRE c.name IS UNIQUE")
    except Exception as e:
        # For Neo4j 4.x
        try:
            graph.run("CREATE CONSTRAINT ON (p:Product) ASSERT p.uniq_id IS␣
↪UNIQUE")
            graph.run("CREATE CONSTRAINT ON (m:Manufacturer) ASSERT m.name IS␣
↪UNIQUE")
            graph.run("CREATE CONSTRAINT ON (c:Category) ASSERT c.name IS␣
↪UNIQUE")
        except Exception as e:
            print(f"Warning: Could not create constraints: {e}")
```

```
    for _, row in df.iterrows():
        # Create Product node
        product = Node("Product",
            uniq_id=row['uniq_id'],
            name=row['product_name'],
            description=row['product_description'],
            price=float(row['price_value']),
            currency=row['currency'],
            review_rating=float(row['average_review_rating']),
            review_count=int(row['number_of_reviews']),
            stock_type=row['stock_type'] if pd.notna(row['stock_type']) else␣
 ↪None,
            description_complete=row['description_complete']
        )

        # Create Manufacturer node
        manufacturer = Node("Manufacturer", name=row['manufacturer'])

        # Create Category nodes from hierarchy
        categories = row['amazon_category_and_sub_category'].split(' > ')
        previous_category = None
        for cat in categories:
            category = Node("Category", name=cat)
            graph.merge(category, "Category", "name")

            if previous_category:
                # Create hierarchical relationship between categories
                rel = Relationship(previous_category, "HAS_SUBCATEGORY",␣
 ↪category)
                graph.merge(rel)
            previous_category = category

        # Create relationships
        graph.merge(product, "Product", "uniq_id")
        graph.merge(manufacturer, "Manufacturer", "name")

        # Connect product to manufacturer
        graph.merge(Relationship(product, "MANUFACTURED_BY", manufacturer))

        # Connect product to lowest-level category
        graph.merge(Relationship(product, "BELONGS_TO", previous_category))

# Create the knowledge graph
create_knowledge_graph(df)
```

1.5 5. Querying and Visualizing the Knowledge Graph

We can now query the graph to retrieve information and visualize the relationships.

1.5.1 Defining a Function for Querying and Visualization

[11]:
```python
def run_query_with_viz(query, title, viz_query=None):
    print(f"\n=== {title} ===")

    # Run and display query results as a DataFrame
    results = graph.run(query).data()
    df = pd.DataFrame(results)
    display(df)

    # Debug: Print the visualization query results
    viz_results = graph.run(viz_query or query).data()
    print(f"\nNumber of visualization records: {len(viz_results)}")

    # Create visualization
    plt.figure(figsize=(12, 8))
    G = nx.Graph()

    # Add nodes and edges
    for record in viz_results:
        product = record.get('p', {})
        manufacturer = record.get('m', {})

        if product and manufacturer:
            # Add nodes
            product_name = product.get('product_name', 'Unknown Product')
            manufacturer_name = manufacturer.get('name', 'Unknown Manufacturer')

            G.add_node(product_name,
                       label=product_name[:30] + "...",
                       type='Product')
            G.add_node(manufacturer_name,
                       label=manufacturer_name,
                       type='Manufacturer')

            # Add edge
            G.add_edge(product_name, manufacturer_name)

    if len(G.nodes()) > 0:  # Only draw if we have nodes
        # Draw graph
        pos = nx.spring_layout(G)

        # Draw nodes by type
```

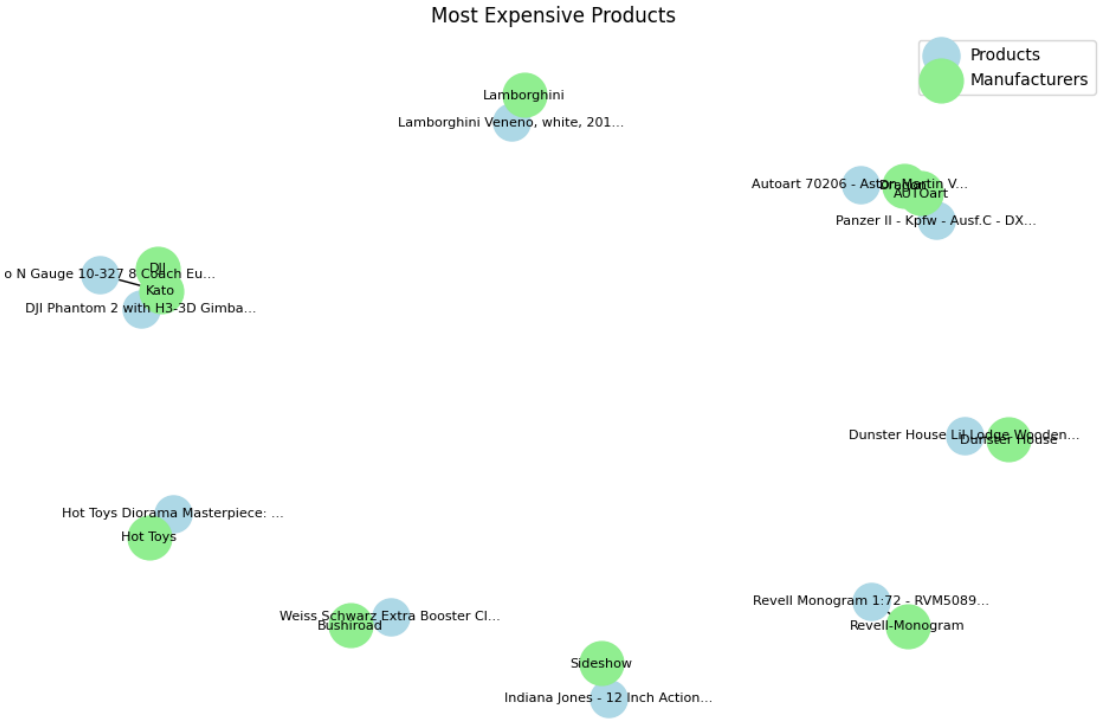
```
        products = [n for n, attr in G.nodes(data=True) if attr['type'] ==␣
 ↪'Product']
        manufacturers = [n for n, attr in G.nodes(data=True) if attr['type'] ==␣
 ↪'Manufacturer']

        if products:
            nx.draw_networkx_nodes(G, pos, nodelist=products,
                                   node_color='lightblue', node_size=500,
                                   label='Products')
        if manufacturers:
            nx.draw_networkx_nodes(G, pos, nodelist=manufacturers,
                                   node_color='lightgreen', node_size=700,
                                   label='Manufacturers')

        # Draw edges and labels
        nx.draw_networkx_edges(G, pos)
        labels = nx.get_node_attributes(G, 'label')
        nx.draw_networkx_labels(G, pos, labels, font_size=8)

        plt.title(title)
        plt.legend()
        plt.axis('off')
    else:
        plt.title(f"{title}\n(No data to visualize)")

    plt.show()

# Test query with simpler visualization
query1 = """
MATCH (p:Product)-[:MANUFACTURED_BY]->(m:Manufacturer)
RETURN m.name as Manufacturer, p.name as Product, p.price as Price
ORDER BY p.price DESC
LIMIT 5
"""

viz_query1 = """
MATCH (p:Product)-[r:MANUFACTURED_BY]->(m:Manufacturer)
RETURN {
    product_name: p.name,
    price: p.price
} as p,
{
    name: m.name
} as m
ORDER BY p.price DESC
LIMIT 10
"""
```

```
run_query_with_viz(query1, "Most Expensive Products", viz_query1)
```

```
=== Most Expensive Products ===
```

	Manufacturer	Product	Price
0	DJI	DJI Phantom 2 with H3-3D Gimbal	995.11
1	Sideshow	Indiana Jones - 12 Inch Action Figures: Indian…	719.95
2	AUTOart	Autoart 70206 - Aston Martin V12 Vantage - 201…	648.95
3	Bushiroad	Weiss Schwarz Extra Booster Clannad Vol.3	629.95
4	Dragon	Panzer II - Kpfw - Ausf.C - DX'10 - 1:6th Scale	592.95

```
Number of visualization records: 10
```



1.6 6. Generating and Storing Embeddings

We enhance our knowledge graph by generating embeddings for product descriptions, enabling semantic search capabilities.

1.6.1 Configuring the Embedding API

Note: Replace `"YOUR_API_KEY"` with your actual API key.

[12]:
```
os.environ["GOOGLE_API_KEY"] = "YOUR_API_KEY"

genai.configure(api_key=os.getenv("GOOGLE_API_KEY"))

result = genai.embed_content(
    model="models/text-embedding-004",
    content="What is the meaning of life?",
    task_type="retrieval_document",
    title="Embedding of single string")

# 1 input > 1 vector output
print(str(result['embedding'])[:50], '... TRIMMED]')
```

```
[-0.02854543, 0.044588115, -0.034197364, -0.004266 … TRIMMED]
```

1.6.2 Defining Functions to Generate and Store Embeddings

[14]:
```
# Rate limiter decorator
@sleep_and_retry
@limits(calls=1500, period=60)
def get_embedding(text):
    try:
        result = genai.embed_content(
            model="models/text-embedding-004",
            content=text,
            task_type="retrieval_document",
        )
        return result['embedding']
    except Exception as e:
        print(f"Error getting embedding: {e}")
        return None

def add_embeddings_to_products(batch_size=50):
    # Get the total number of products to process
    total_query = """
    MATCH (p:Product)
    WHERE p.description_embedding IS NULL
      AND p.description IS NOT NULL
    RETURN count(p) AS total
    """
    total_result = graph.run(total_query).data()
    total_to_process = total_result[0]['total'] if total_result else 0

    print(f"Total products to process: {total_to_process}\n")

    total_processed = 0

    # Initialize tqdm progress bar
```

```
    with tqdm(total=total_to_process, desc='Processing products',␣
↪unit='product') as pbar:
        while True:
            # Get batch of products
            query = """
            MATCH (p:Product)
            WHERE p.description_embedding IS NULL
              AND p.description IS NOT NULL
            RETURN p.uniq_id AS id, p.description AS description
            LIMIT $batch_size
            """
            products = graph.run(query, parameters={'batch_size': batch_size}).
↪data()
            if not products:
                break

            # Process each product in the batch
            for product in products:
                try:
                    if product['description']:
                        embedding = get_embedding(product['description'])
                        if embedding:
                            # Update product with embedding
                            graph.run("""
                            MATCH (p:Product {uniq_id: $id})
                            SET p.description_embedding = $embedding
                            """, parameters={
                                'id': product['id'],
                                'embedding': embedding
                            })
                        total_processed += 1
                        pbar.update(1)  # Update the progress bar
                except Exception as e:
                    print(f"Error processing product {product['id']}: {e}")

            # Add a small delay between batches
            time.sleep(1)

    print(f"\nTotal products processed: {total_processed}")
    return total_processed

# Add embeddings to products
print("Adding embeddings to products...\n")
total_processed = add_embeddings_to_products()
print(f"\nProcess completed. Total products processed: {total_processed}")
```

```
Adding embeddings to products…
```

```
Total products to process: 7434

Processing products:   0%|          | 0/7434 [00:00<?, ?product/s]
Processing products: 100%|      | 7434/7434 [27:20<00:00,  4.53product/s]

Total products processed: 7434

Process completed. Total products processed: 7434
```

1.6.3 Verifying Embeddings

[14]:
```
# Verify embeddings
print("\nVerifying embeddings:")
result = graph.run("""
MATCH (p:Product)
WHERE p.description_embedding IS NOT NULL
RETURN count(p) as count
""").data()
print(f"Products with embeddings: {result[0]['count']}")
```

```
Verifying embeddings:
Products with embeddings: 7434
```

1.7 7. Performing Semantic Search

We use the embeddings to perform a semantic search based on a user's natural language query.

[19]:
```
def semantic_search(query_text, n=5):
    # Get query embedding
    query_embedding = get_embedding(query_text)
    if not query_embedding:
        print("Failed to get query embedding")
        return []

    # Debug: Print embedding info
    print(f"Query embedding length: {len(query_embedding)}")

    # Search for similar products using dot product and magnitude for cosine␣
 ↪similarity
    results = graph.run("""
    MATCH (p:Product)
    WHERE p.description_embedding IS NOT NULL
    WITH p,
```

```
        reduce(dot = 0.0, i in range(0, size(p.description_embedding)-1) |
          dot + p.description_embedding[i] * $embedding[i]) /
        (sqrt(reduce(a = 0.0, i in range(0, size(p.description_embedding)-1) |
          a + p.description_embedding[i] * p.description_embedding[i])) *
         sqrt(reduce(b = 0.0, i in range(0, size($embedding)-1) |
          b + $embedding[i] * $embedding[i])))
        AS similarity
    WHERE similarity > 0
    RETURN
        p.name as name,
        p.description as description,
        p.price as price,
        p.description_embedding as embedding,
        similarity as score
    ORDER BY similarity DESC
    LIMIT $n
    """, parameters={'embedding': query_embedding, 'n': n}).data()

    return results

# Test the search with debug info
print("\nTesting semantic search:")
results = semantic_search("Give me a set of card", n=2)
print(f"\nNumber of results: {len(results)}")

for r in results:
    print(f"\nProduct: {r.get('name', 'No name')}")
    print(f"Price: ${r.get('price', 'N/A')}")
    print(f"Score: {r.get('score', 'N/A'):.3f}")
    desc = r.get('description', 'No description')
    print(f"Description: {desc}")
    print(f"Embedding: {r.get('embedding', 'N/A')[:10]}")
```

```
Testing semantic search:
Query embedding length: 768

Number of results: 2

Product: Yu-Gi-Oh Metal Raiders Booster
Price: $9.76
Score: 0.852
Description: 9 Cards Per Pack.
Embedding: [-0.03146668, 0.013115278, -0.010723814, -0.03084892, 0.034354843,
0.015702378, 0.009507097, 0.010504648, -0.016177453, 0.030757932]

Product: AKB48 Trading Card Game & Collection vol.1 Booster (15packs)
Price: $12.25
```

Score: 0.827
Description: 15 packs, 6 cards per pack.
Embedding: [-0.036870565, 0.0056306464, -0.020418527, -0.0186566, 0.036592394,
0.0010830638, 0.008574845, 0.010452268, -0.005662437, 0.026855843]