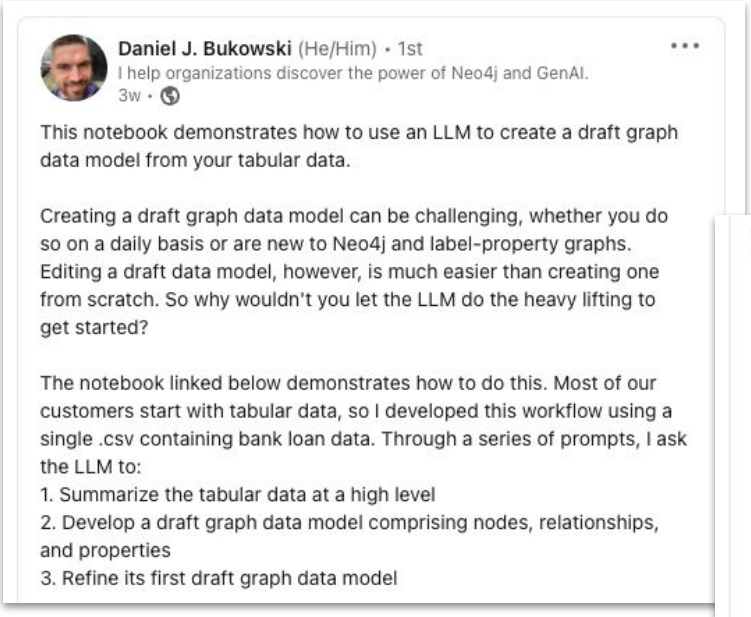
Going Meta #25
LLMs for automated KG construction

(GM#5 revisited)

Where it all started...



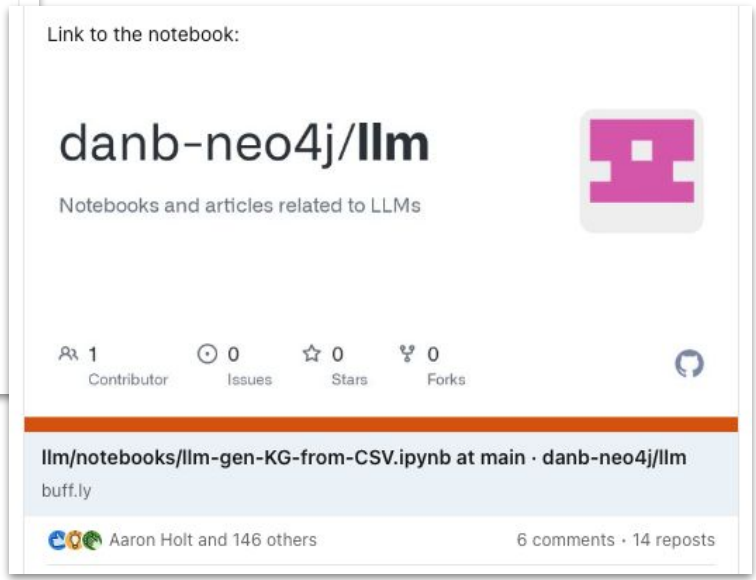
Daniel J. Bukowski (He/Him) • 1st
I help organizations discover the power of Neo4j and GenAI.
3w •

This notebook demonstrates how to use an LLM to create a draft graph data model from your tabular data.

Creating a draft graph data model can be challenging, whether you do so on a daily basis or are new to Neo4j and label-property graphs. Editing a draft data model, however, is much easier than creating one from scratch. So why wouldn't you let the LLM do the heavy lifting to get started?

The notebook linked below demonstrates how to do this. Most of our customers start with tabular data, so I developed this workflow using a single .csv containing bank loan data. Through a series of prompts, I ask the LLM to:
1. Summarize the tabular data at a high level
2. Develop a draft graph data model comprising nodes, relationships, and properties
3. Refine its first draft graph data model

https://github.com/danb-neo4j/llm



Link to the notebook:

danb-neo4j/**llm**

Notebooks and articles related to LLMs

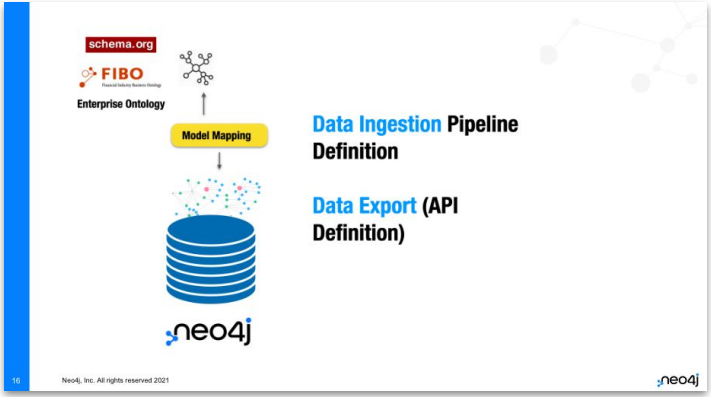
1 Contributor | 0 Issues | 0 Stars | 0 Forks

llm/notebooks/llm-gen-KG-from-CSV.ipynb at main · danb-neo4j/llm
buff.ly

Aaron Holt and 146 others — 6 comments • 14 reposts

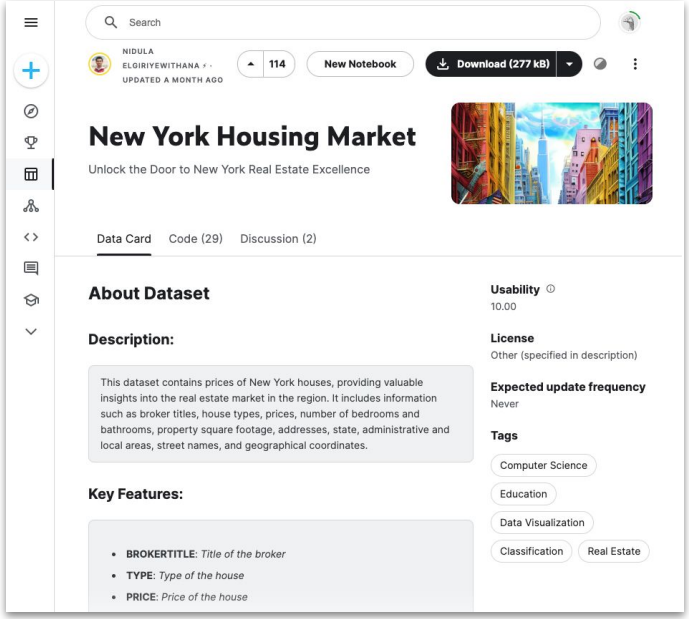
https://www.linkedin.com/in/danieljbukowski/

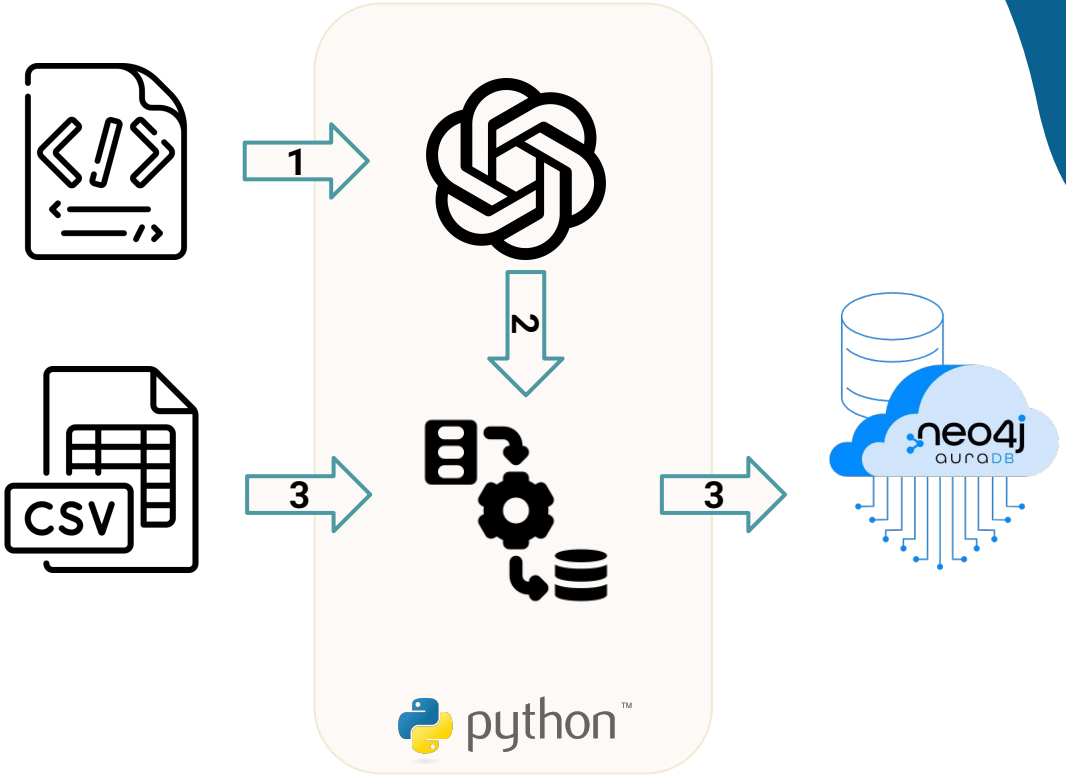
We also explored the topic of automating DB construction back in #5



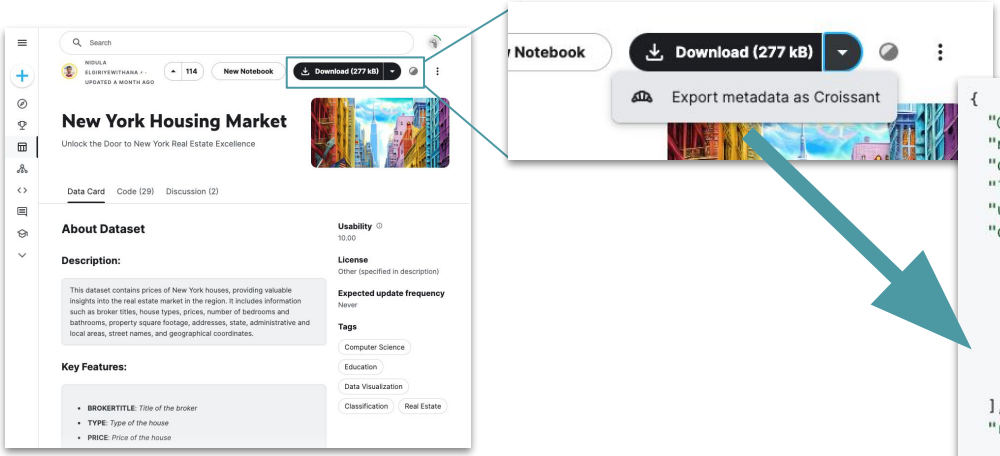
Q: *Modelling best practices, please. How do I create my declarative schema definition?*

The automated KG construction process





The Metadata (is a JSON-LD graph)




```
{
  "@type": "sc:Dataset",
  "name": "minimal_example_with_recommended_fields",
  "description": "This is a minimal example, including the required and the recommended fields
  "license": "https://creativecommons.org/licenses/by/4.0/",
  "url": "https://example.com/dataset/recipes/minimal-recommended",
  "distribution": [
    {
      "@type": "sc:FileObject",
      "name": "minimal.csv",
      "contentUrl": "data/minimal.csv",
      "encodingFormat": "text/csv",
      "sha256": "48a7c257f3c90b2a3e529ddd2cca8f4f1bd8e49ed244ef53927649504ac55354"
    }
  ],
  "recordSet": [
    {
      "@type": "ml:RecordSet",
      "name": "examples",
      "description": "Records extracted from the example table, with their schema.",
      "field": [
        {
          "@type": "ml:Field",
          "name": "name",
          "description": "The first column contains the name.",
          "dataType": "sc:Text",
          "references": {
            "distribution": "minimal.csv",
            "extract": {
              "column": "name"
            }
          }
        },
        {
          "@type": "ml:Field",
          "name": "age",
```


https://github.com/mlcommons/croissant

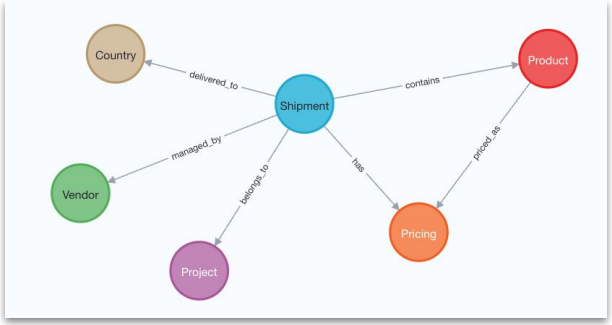
The LLM carries out the modelling


```
{
  "@type": "sc:Dataset",
  "name": "minimal_example_with_recommended_fields",
  "description": "This is a minimal example, including the required and the recommended fields
  "license": "https://creativecommons.org/licenses/by/4.0/",
  "url": "https://example.com/dataset/recipes/minimal-recommended",
  "distribution": [
    {
      "@type": "sc:FileObject",
      "name": "minimal.csv",
      "contentUrl": "data/minimal.csv",
      "encodingFormat": "text/csv",
      "sha256": "48a7c257f3c90b2a3e529ddd2cca8f4f1bd8e49ed244ef53927649504ac55354"
    }
  ],
  "recordSet": [
    {
      "@type": "ml:RecordSet",
      "name": "examples",
      "description": "Records extracted from the example table, with their schema.",
      "field": [
        {
          "@type": "ml:Field",
          "name": "name",
          "description": "The first column contains the name.",
          "dataType": "sc:Text",
          "references": {
            "distribution": "minimal.csv",
            "extract": {
              "column": "name"
            }
          }
        },
        {
          "@type": "ml:Field",
          "name": "age",
```



```
{
  "entities": [
    {
      "name": "Shipment",
      "attributes": [
        {"name": "ID", "type": "Text", "mappedTo": "ID"},
        {"name": "FulfillmentMethod", "type": "Text", "mappedTo": "Fulfill Via"},
        {"name": "ShipmentMode", "type": "Text", "mappedTo": "Shipment Mode"},
        {"name": "ManagedBy", "type": "Text", "mappedTo": "Managed By"},
        {"name": "Weight", "type": "Text", "mappedTo": "Weight (Kilograms)"},
```

```
  ],
  "relationships": [
    {"from": "Shipment", "to": "Product", "type": "contains"},
    {"from": "Shipment", "to": "Pricing", "type": "has"},
    {"from": "Shipment", "to": "Vendor", "type": "managed_by"},
    {"from": "Shipment", "to": "Project", "type": "belongs_to"},
```

...with a simple call to the completions API and some code adapted from episode #5

```
system = "You are a data modelling expert capable of creating high quality entity-relationship models from flat datasets"

prompt=f"""

From the list of features in the following dataset create a list of entities and relationships with their
attributes in a simple json format and map them to the features in the dataset.

The attributes don't need to be named after the features in the dataset, but they should be mapped to the corresponding feature name.
No extra text or comments, only the json as output.

DATASET NAME: {ds_name}

DATASET DESCRIPTION: {ds_description}

DATASET FEATURES: {ds_features}
"""

print(prompt)
```

```
from openai import OpenAI

client = OpenAI()
completion = client.chat.completions.create(
     model="gpt-4",
     temperature=0,
      messages=[
           {"role": "system", "content": system},
           {"role": "user", "content": prompt},
       ])
```


Let’s see it in action!

What next? Use in combination with Dataflow

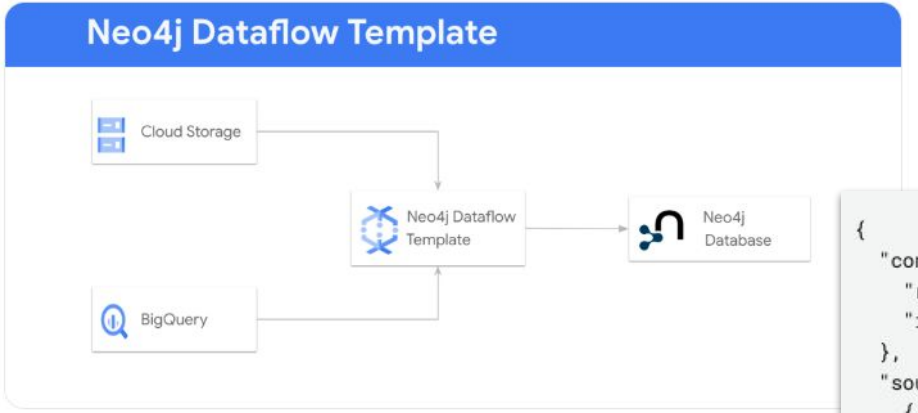


Fig 1: Architecture Diagram of Dataflow from Google Cloud to Neo4j

```
{
  "config": {
    "reset_db": true,
    "index_all_properties": fa
  },
  "sources": [
    {
      "type": "bigquery",
      "name": "persons",
      "query": "SELECT person_
    },
    {
      "type": "bigquery",
      "name": "movies",
      "query": "SELECT movieId, title, imdbRating, yea
    },
    {
      "type": "bigquery",
      "name": "directed"
```

```
  ],
  "targets": [
    {
      "node": {
        "source": "persons",
        "name": "Person",
        "mode": "merge",
        "transform": {
          "group": true
        },
        "mappings": {
          "labels": [
            "\"Person\""
          ],
          "properties": {
            "keys": [
              {"person_tmdbId": "id"}
            ],
            "unique": [],
            "indexed": [
              {"name": "name"}
            ],
```

https://neo4j.com/docs/dataflow-bigquery/job-specification/