+ Code + Text

## Network san install hiih

```
from langchain_community.document_loaders import PyMuPDFLoader
from langchain_text_splitters import RecursiveCharacterTextSplitter

from qdrant_client import QdrantClient, models
from qdrant_client.models import PointStruct
from sentence_transformers import SentenceTransformer

import uuid
import os
from dotenv import load_dotenv

load_dotenv()


loader = PyMuPDFLoader("/content/japanese.pdf")
documents = loader.load()

text_splitter = RecursiveCharacterTextSplitter(
    chunk_size=500,
    chunk_overlap=100,
    separators=["  ", "\n", " "],
)
chunks = text_splitter.split_documents(documents)
chunks
# model = SentenceTransformer("paraphrase-multilingual-mpnet-base-v2")

# QDRANT_API = os.getenv("QDRANT_API")
# QDRANT_URL = os.getenv("QDRANT_URL")
# url = "https://e71963b0-0da3-459d-9d04-ac5e5c9a97a0.europe-west3-0.gcp.cloud.qdrant.io"
# api = "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhY2Nlc3MiOiJtIiwiZXhwIjoxNzQ2NjkzMDQ0fQ.swNPX-9NXDminPRA_V0D_zbfE-EU1wBJ1Rqy
# client = QdrantClient(url=url, api_key=api, timeout=120)

# collection_name = "japanese_book"
# client.create_collection(
#     collection_name=collection_name,
#     vectors_config=models.VectorParams(size=768, distance=models.Distance.COSINE),
# )

# points = [
#     PointStruct(
#         id=str(uuid.uuid4()),
#         vector=model.encode(chunk.page_content.strip()).tolist(),
#         payload={"text": chunk.page_content.strip()}
#     )
#     for chunk in chunks
# ]

# client.upsert(
#     collection_name=collection_name,
#     points=points
# )

# print(f"Inserted {len(points)} semantic chunks into Qdrant!")
```

Show hidden output

```
import os

# Set the correct path to your Documents folder
documents_folder = "/Users/batman./Documents"

all_paths = []
for root, dirs, files in os.walk(documents_folder):
    for file in files:
        full_path = os.path.join(root, file)
        all_paths.append(full_path)

# Print all paths
for path in all_paths:
    print(path)
```

## powerlaw san tatah code

```
!pip install langchain-community pymupdf langchain_community==0.0.29 python-dotenv groq qdrant-client sentence-transformers
```

Show hidden output

## Heregtseet sanguudiig import hiih heseg

```
import networkx as nx
import warnings
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import powerlaw
warnings.filterwarnings('ignore')
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

## read_edgelist ni irmeguudiig graphaas unshina.

```
df = "/content/drive/MyDrive/Colab Notebooks/network.csv"
g = nx.read_edgelist(df, delimiter=",", nodetype=int)
```

## node-uudiing medeelliig iim maygaar harj bolno

```
for n in g.nodes():
    print(n)
```

Show hidden output

## Niit heden node bga ve gedgiig harah heseg

```
g.number_of_nodes()
```

```
2566
```

## Chigleletei graph uu esvel chiglelgui graph gedgiig harah

```
g.is_directed()
```

```
False
```

## Heden irmeg tuhain node-tei holbogdson eseh

```
g.degree(1)
```

```
24
```

## g graph-iin dundaj degree utgiig oloh heseg

```
sum(dict(g.degree()).values())/float(len(g))
```

```
6.697583787996883
```

## g graph iin degree-iin tarhaltiig harah heseg

```
hist = nx.degree_histogram(g)
plt.plot(range(0, len(hist)), hist, ".")
plt.title("Degree Distribution")
plt.xlabel("Degree")
plt.ylabel("#Nodes")
plt.loglog()
plt.show()
```

## 0 node-tei heden graph chigleltei holbogdson eseh

```
list(g.neighbors(0))
```

```
[306, 830, 1599, 273, 1988]
```

## Ego network gedeg ni todorhoi neg node-iin holboltuud deer tovloroh heseg ym

```
ego = nx.ego_graph(g, 0)
nx.draw(ego, with_labels=True)
```

## Niit holbogdson. graph-uudiin too

```
nx.number_connected_components(g)
```

```
85
```

1. Holbogdson hesguudiig haih
2. Graph-uudiig avah

### 3. Graph-iig zurah **bold text**

```
comps = list(nx.connected_components(g))
comp_1 = nx.subgraph(g, comps[1])
nx.draw(comp_1)
```

## 2 node-iin hooron dahi hamgiin bogino zamiig oloh

```
nx.shortest_path(g, source=0, target=30)
```

```
[0, 306, 30]
```

## 2 node-iin hooron dahi zamiin urtiig oloh

```
nx.shortest_path_length(g, source=0, target=30)
```

```
2
```

## Graph-iin diameter-iig butsaana

```
nx.diameter(g.subgraph(comps[0]))
```

```
17
```

## Nyagtraliig butsaaana

```
nx.density(g)
```

```
0.002611143776996835
```

## Dictionary torol butsaana. Key ni graph-iin node baina

```
nx.triangles(g)[0]
```

```
4
```

## Clusteriin utgiig butsaana

```
nx.clustering(g)[0]
```

```
0.4
```

## Dundaj cluster utgiig butsaana.

```
nx.average_clustering(g)
```

```
0.20063633264589634
```

# Minii olson data deerh code heseg

## Random maygaar 100 node nemeh heseg

```
import networkx as nx
import random

g = nx.Graph()
g.add_nodes_from(range(100))

num_edges = 200
for _ in range(num_edges):
    u, v = random.sample(list(g.nodes), 2)
    weight = random.randint(1, 10)
    g.add_edge(u, v, weight=weight)

print("Edges:", list(g.edges(data=True))[:10])
```

```
Edges: [(0, 2, {'weight': 4}), (0, 50, {'weight': 10}), (0, 58, {'weight': 2}), (1, 63, {'weight': 7}), (1, 62, {'weight
```

```
g.number_of_nodes()
```

```
100
```

```
g.number_of_edges()
```

```
195
```

```
g.is_directed()
```

```
False
```

```
g.degree(1)
```

```
2
```

```
sum(dict(g.degree()).values())/float(len(g))
```

```
3.9
```

```
hist = nx.degree_histogram(g)
plt.plot(range(0, len(hist)), hist, ".")
plt.title("Degree Distribution")
plt.xlabel("Degree")
plt.ylabel("#Nodes")
plt.loglog()
plt.show()
```

```
list(g.neighbors(0))
```

```
[2, 50, 58]
```

```
ego = nx.ego_graph(g, 0)
nx.draw(ego, with_labels=True)
```

```
nx.number_connected_components(g)
```

```
3
```

```
comps = list(nx.connected_components(g))
comp_1 = nx.subgraph(g, comps[1])
nx.draw(comp_1)
nx.shortest_path(g, source=0, target=30)
```

```
[0, 2, 31, 90, 30]
```

```
nx.shortest_path_length(g, source=0, target=30)
```

```
4
```

```
nx.diameter(g.subgraph(comps[0]))
```

```
8
```

```
nx.density(g)
```

```
0.03939393939393939
```

```
nx.triangles(g)[0]
```

```
0
```

```
nx.clustering(g)[0]
```

```
0
```

```
nx.average_clustering(g)
```

```
0.02550000000000001
```

Start coding or generate with AI.