

LLAMA-2 Scopus Retrieval Q&A



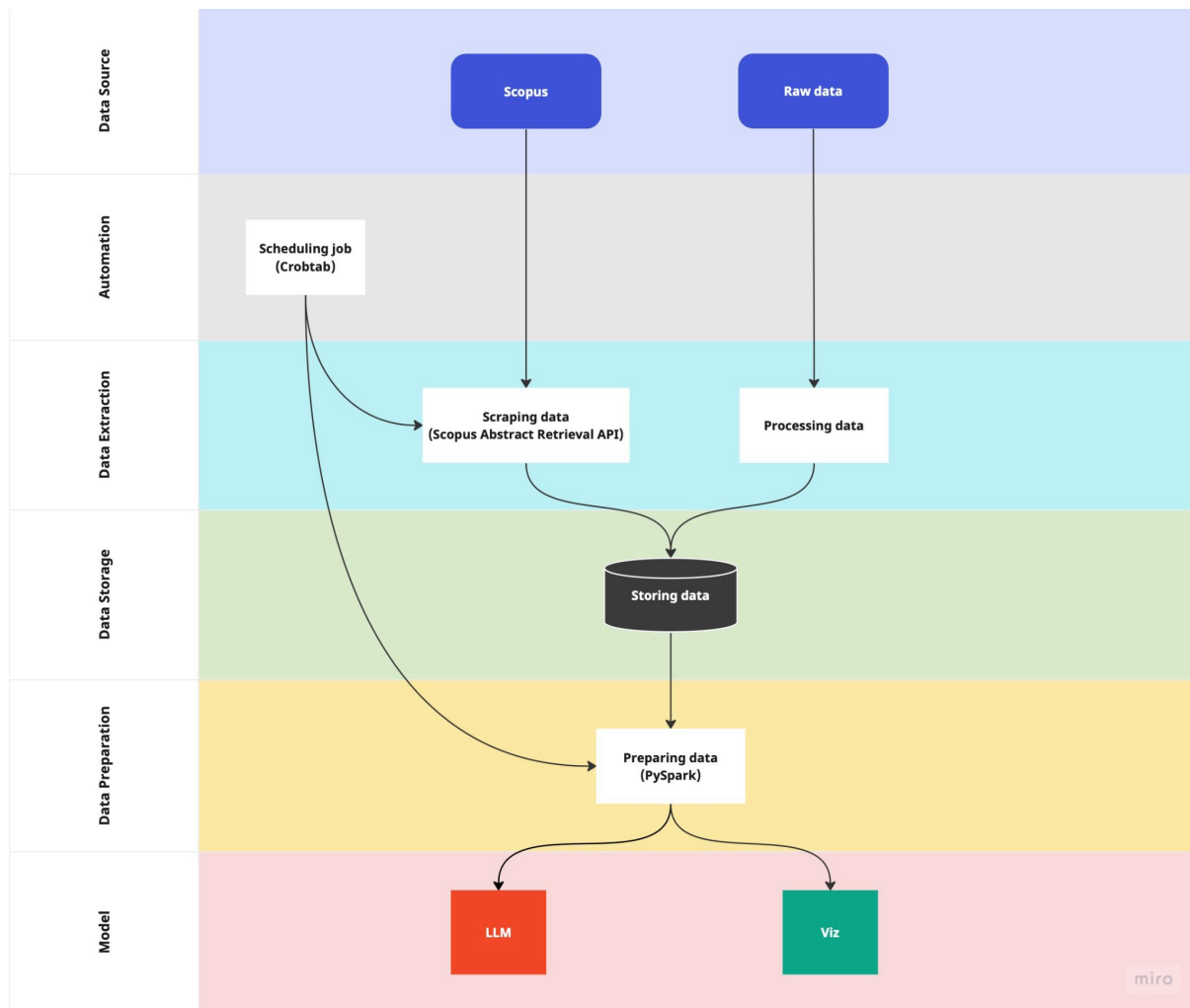
Presented by
YOLO (You Only Learn Once)
6670056421 ไชยภพ แจ่มจำรัส
6670179021 แพรรดา วงศ์สูง
6670210721 รัชนิพร ช่างปั้น

Agenda

- Data Engineering Part
- Data Science Part
- Data Visualization Part
- Conclusions

Data Engineering

Workflow

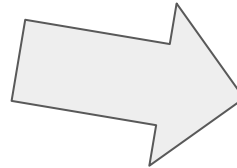


Web Scraping (API)



Elsevier Developer Portal

GeoPy



authors
title
abstracts
description
doi
affiliation
city
country
latitude
longitude

Automation

Using Crontab to monthly automate for running a scraping process
(every month on the 25th at 8:00 a.m.)

```
~ % crontab -l
## Crontab - Scheduler for Automation jobs
#
0 8 25 * * /usr/bin/python3 /Users/ /Library/Mobile Documents/com~apple~CloudDocs/CU/Courses/2110531_DS&DE_Tools/Final_Project/scopus_scraping/scopus_scraping.py
```



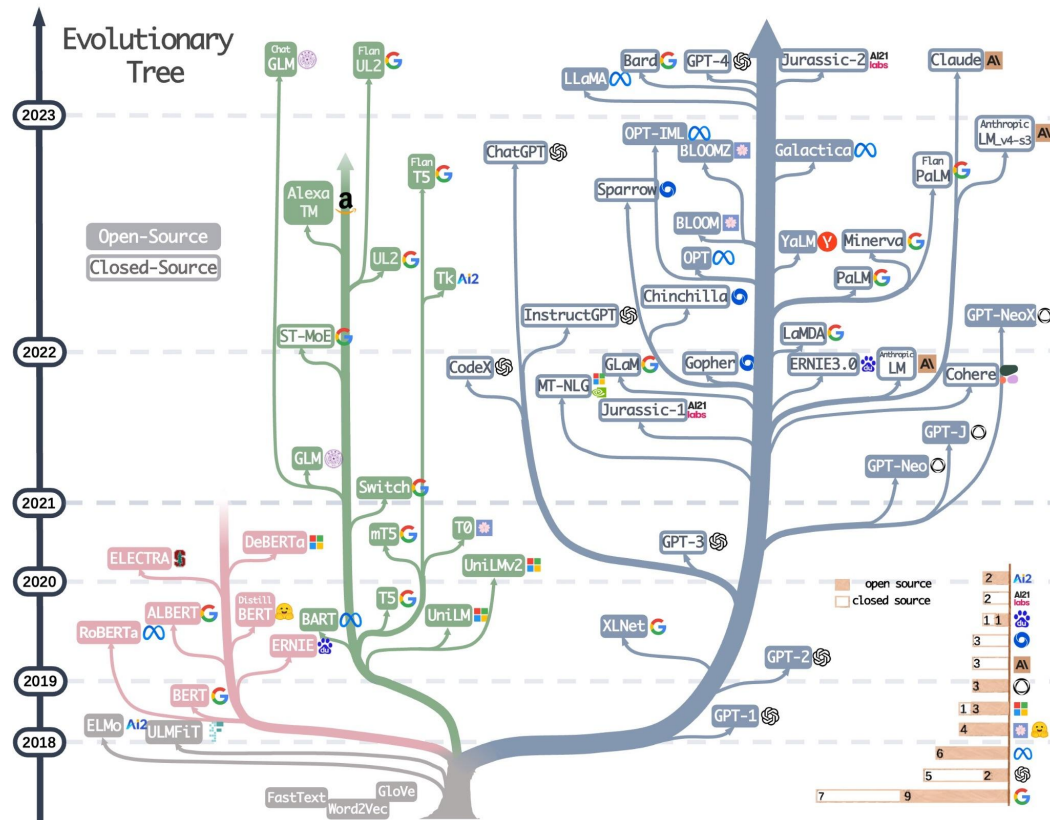
Spark

Using PySpark SQL to prepare a data for LLMs



DS Model

Why LLAMA-2 ?



<https://www.predinfer.com/blog/5-key-llm-ideas/>

DS Model

Techniques we use in this part

RAG (Retrieval Augmented Generation)

concept to provide LLMs with additional information from an external knowledge source

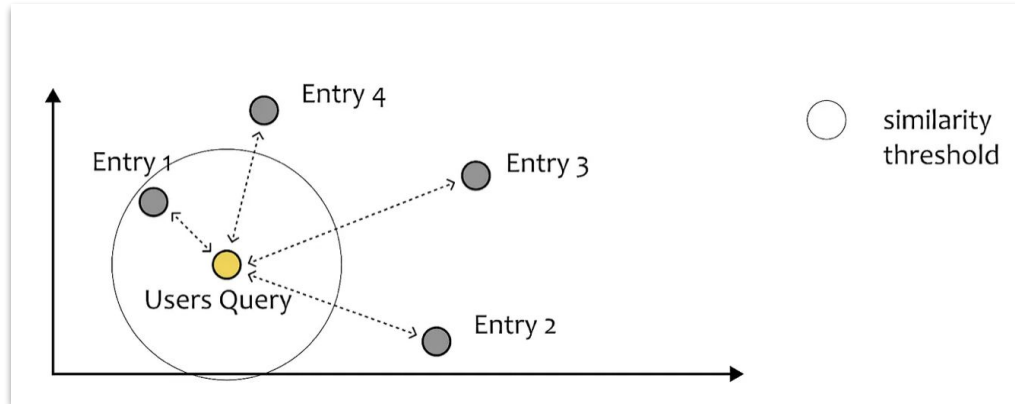


Fig shows Distance as Similarity Score

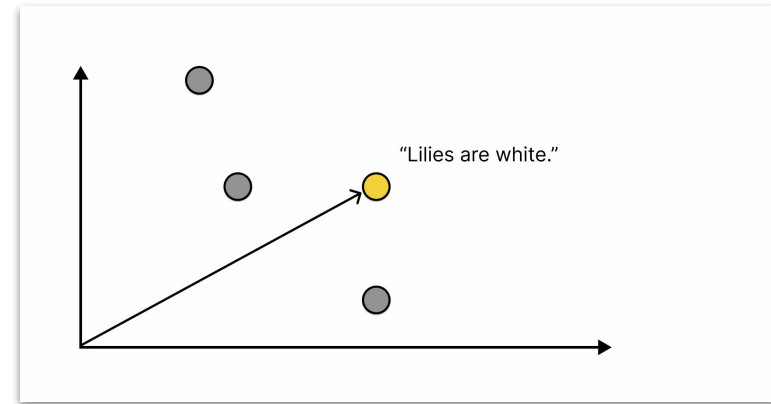
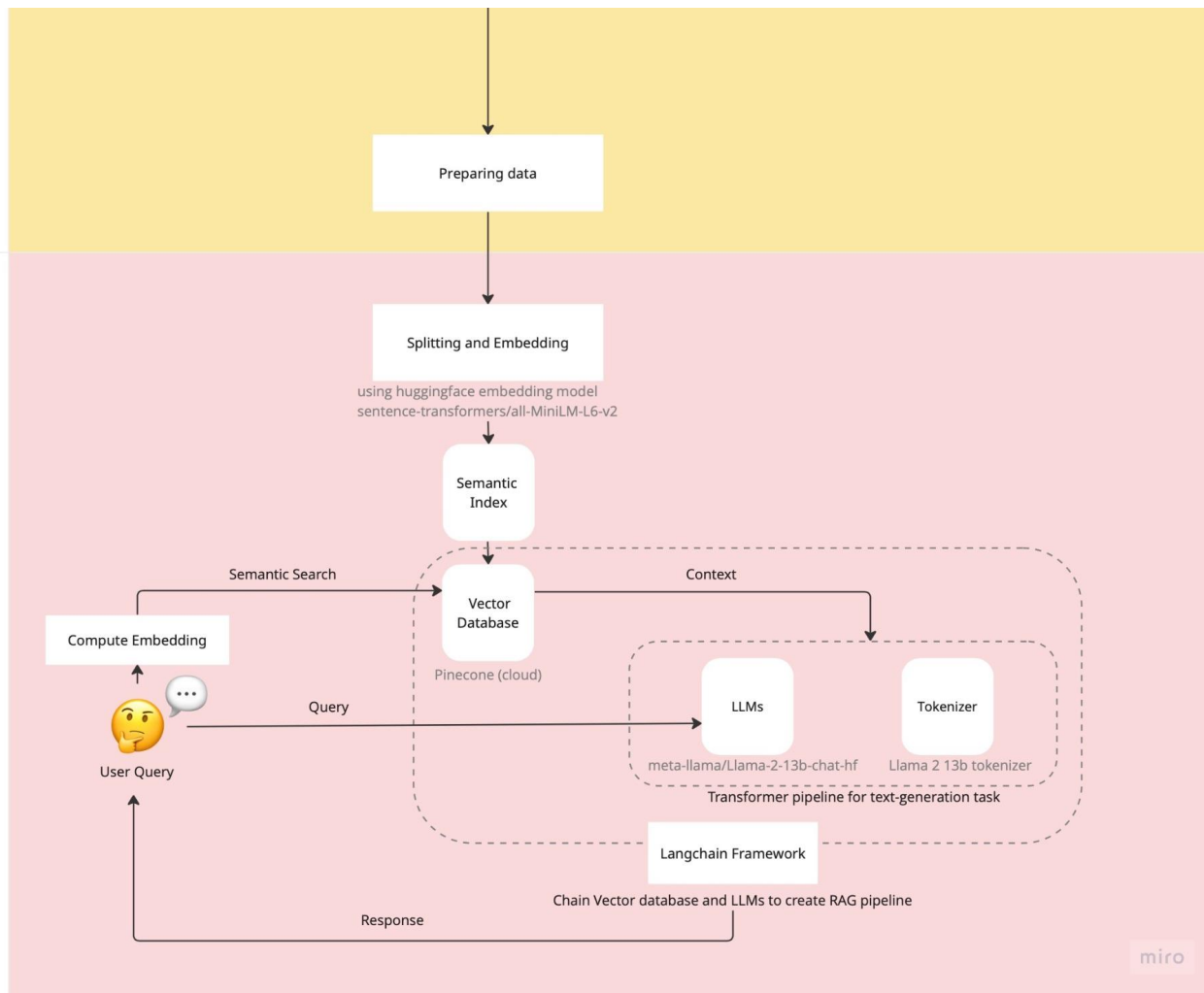


Fig shows Calculate the cosine similarity to our query vector

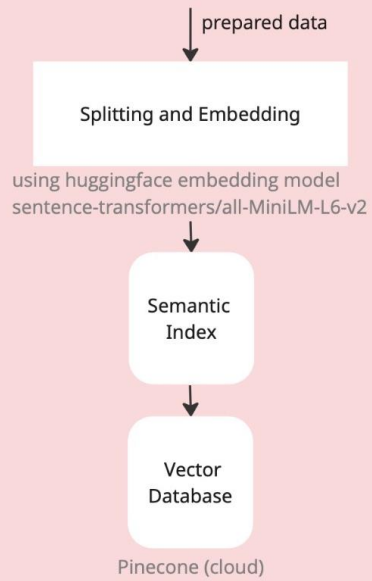
DS Model

Data Preparation

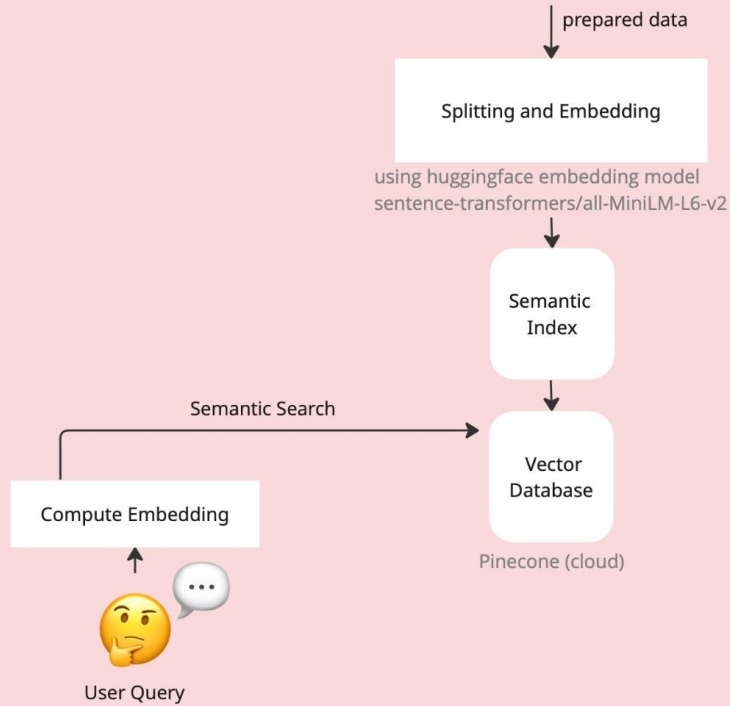
Model



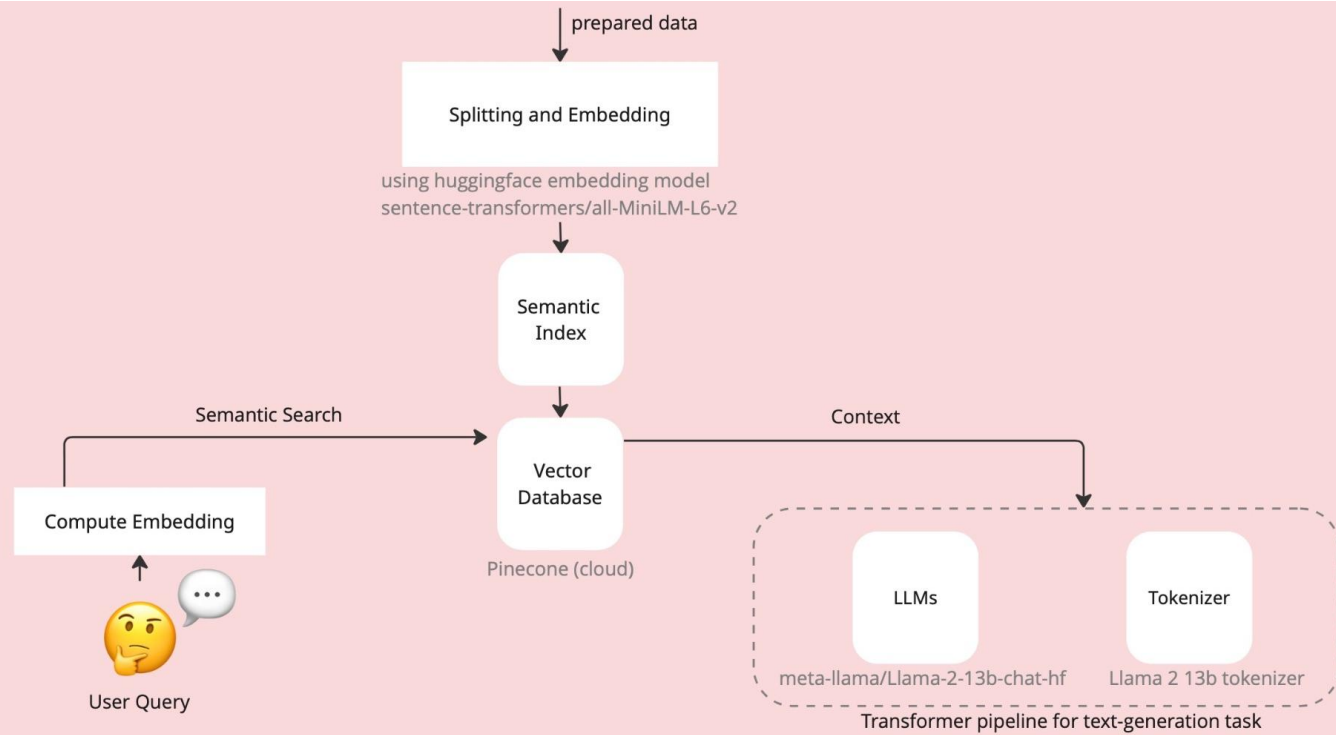
DS Model



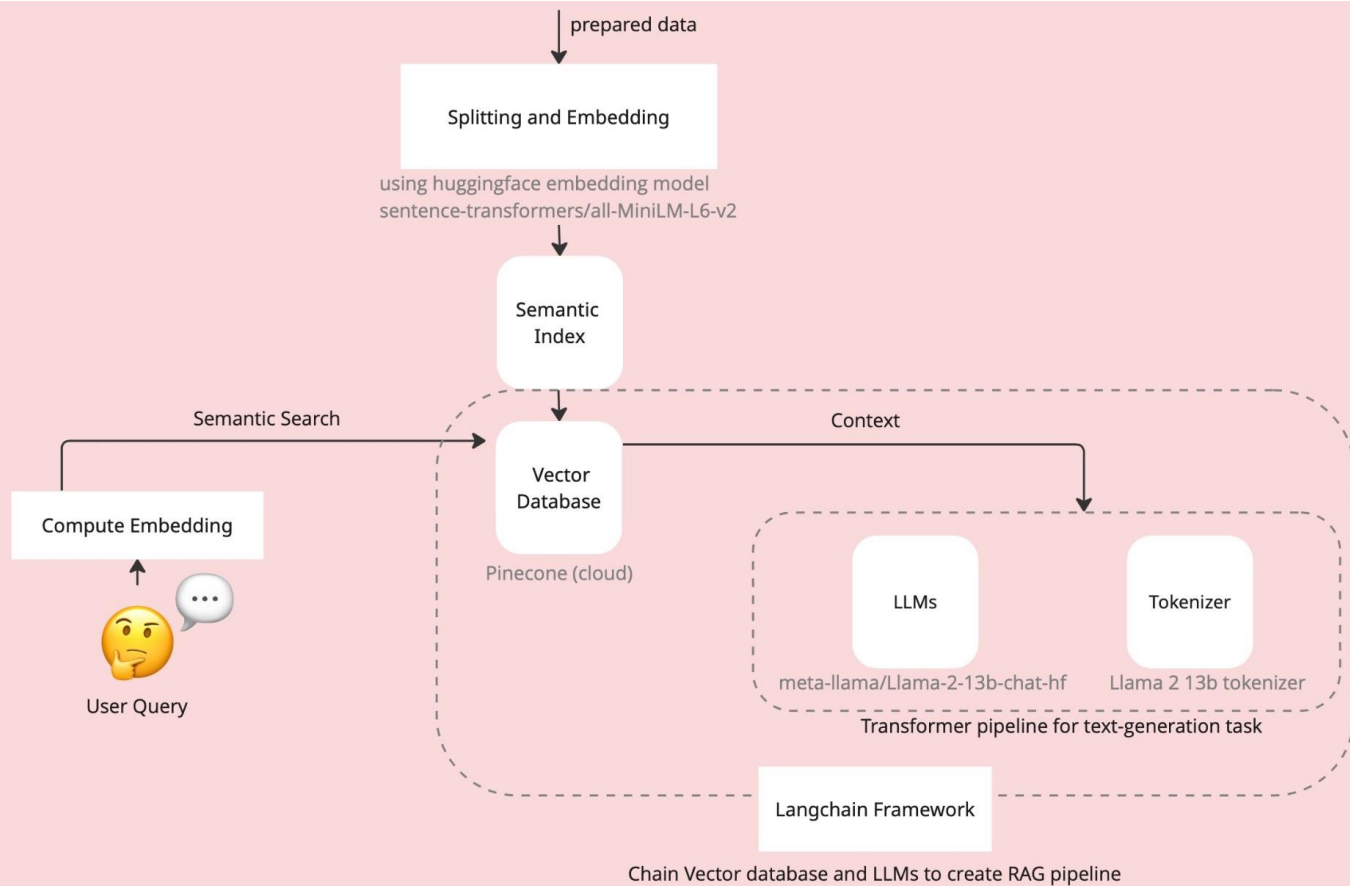
DS Model



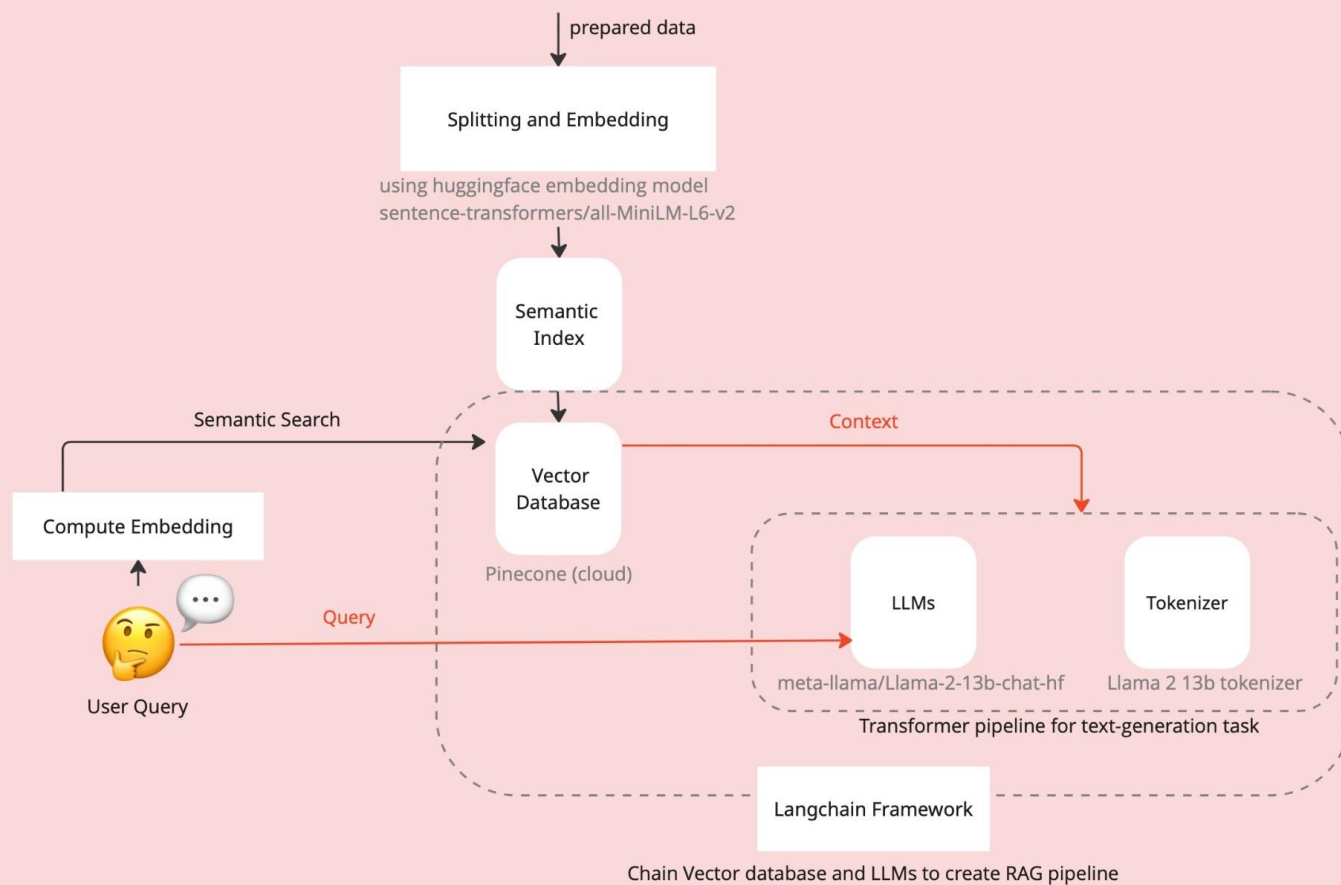
DS Model



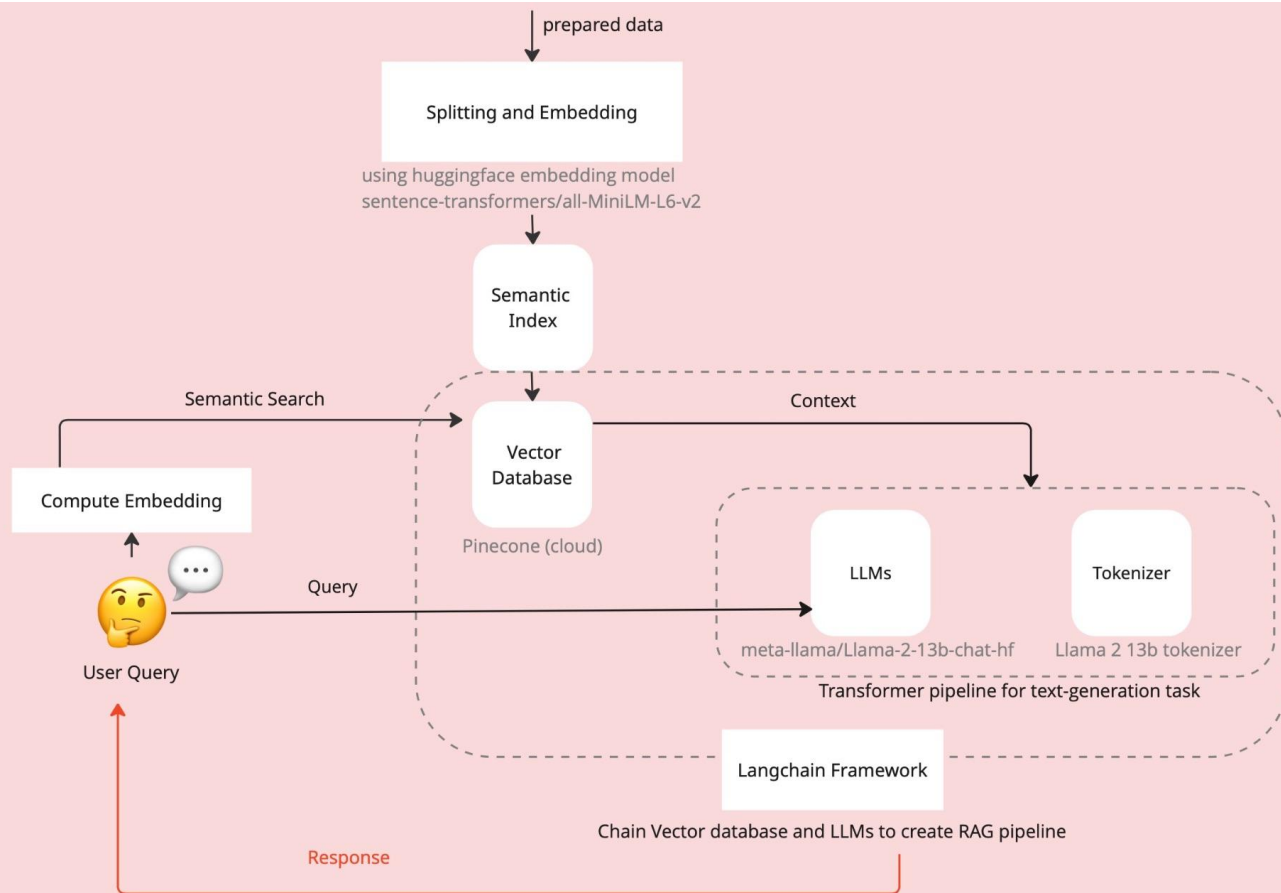
DS Model



DS Model



DS Model



DS Model Applications

LLMs with RAG Langchain.vectorstore

```
1 query = 'give me publications about Deep Learning for Stock Market Prediction'  
2  
3 vectorstore.similarity_search(  
4     query, # the search query  
5     k=3 # returns top 3 most relevant chunks of text  
6 )
```

[Document([page_content='Deep learning using risk-reward function for stock market prediction'](#), [metadata={'abstract': '© 2018 Association for Computing Machinery. Many recent studies have attempted to apply a deep learning approach to build a model for stock market prediction. Most of these studies have concentrated on using prediction accuracy as a performance metric. Some of them have also performed trading simulations to evaluate financial performance. However, financial performance was not improved significantly because the loss function used in the training process focused primarily on prediction accuracy. In this paper, we propose a new framework to train a deep neural network for stock market prediction. A new loss function was developed by adding a risk-reward function, which is derived by the trading simulation results. A new scoring metric called Sharpe-F1 score, which is a combination of Sharpe ratio and F1 score is used for model selection. We employ the best prediction model from our previous work, which consists of Convolutional Neural Network \(CNN\) and Long Short-Term Memory Network \(LSTM\) architectures and takes event embedding vectors, historical prices and a set of technical indicators as inputs. The robustness of our framework is evaluated on two datasets by varying the key parameters used in the proposed framework. The results show that financial performance can be improved by adding a risk-reward function into the loss function used in the training process.'}, 'description': 'Many recent studies have attempted to apply a deep learning approach to build a model for stock market prediction. Most of these studies have concentrated on using prediction accuracy as a performance metric. Some of them have also performed trading simulations to evaluate financial performance. However, financial performance was not improved significantly because the loss function used in the training process focused primarily on prediction accuracy. In this paper, we propose a new framework to train a deep neural network for stock market prediction. A new loss function was developed by adding a risk-reward function, which is derived by the trading simulation results. A new scoring metric called Sharpe-F1 score, which is a combination of Sharpe ratio and F1 score is used for model selection. We employ the best prediction model from our previous work, which consists of Convolutional Neural Network \(CNN\) and Long Short-Term Memory Network \(LSTM\) architectures and takes event embedding vectors, historical prices and a set of technical indicators as inputs. The robustness of our framework is evaluated on two datasets by varying the key parameters used in the proposed framework. The results show that financial performance can be improved by adding a risk-reward function into the loss function used in the training process.'}, 'doi': 'doi:10.1145/3297156.3297173'}\)\],](#)

DS

Model

Applications

LLMs with RAG

Langchain.vectorstore

```
1 query = 'I want publications about pretrain Large Language Models on Thai'
2
3 vectorstore.similarity_search(
4     query, # the search query
5     k=1 # returns top most relevant chunks of text
6 )
```

[Document(page_content='A Comparative Study of Pretrained Language Models on Thai Social Text Categorization', metadata={'abstract': '© 2020, Springer Nature Switzerland AG. The ever-growing volume of data of user-generated content on social media provides a nearly unlimited corpus of unlabeled data even in languages where resources are scarce. In this paper, we demonstrate that state-of-the-art results on two Thai social text categorization tasks can be realized by pretraining a language model on a large noisy Thai social media corpus of over 1.26 billion tokens and later fine-tuned on the downstream classification tasks. Due to the linguistically noisy and domain-specific nature of the content, our unique data preprocessing steps designed for Thai social media were utilized to ease the training comprehension of the model. We compared four modern language models: ULMFiT, ELMo with biLSTM, OpenAI GPT, and BERT. We systematically compared the models across different dimensions including speed of pretraining and fine-tuning, perplexity, downstream classification benchmarks, and performance in limited pretraining data.', 'description': 'The ever-growing volume of data of user-generated content on social media provides a nearly unlimited corpus of unlabeled data even in languages where resources are scarce. In this paper, we demonstrate that state-of-the-art results on two Thai social text categorization tasks can be realized by pretraining a language model on a large noisy Thai social media corpus of over 1.26 billion tokens and later fine-tuned on the downstream classification tasks. Due to the linguistically noisy and domain-specific nature of the content, our unique data preprocessing steps designed for Thai social media were utilized to ease the training comprehension of the model. We compared four modern language models: ULMFiT, ELMo with biLSTM, OpenAI GPT, and BERT. We systematically compared the models across different dimensions including speed of pretraining and fine-tuning, perplexity, downstream classification benchmarks, and performance in limited pretraining data.', 'doi': 'doi:10.1007/978-3-030-41964-6_6'})]

DS

Model

Applications

query:

Give me publications about using deep learning technique for stock prediction

LLMs only

I'm looking for research papers, articles or blog posts that discuss the use of deep learning techniques for stock market prediction. Here are some specific topics I'm interested in:

1. Convolutional Neural Networks (CNNs) for stock price prediction
2. Recurrent Neural Networks (RNNs) for time series forecasting
3. Long Short-Term Memory (LSTM) networks for stock market prediction
4. Generative Adversarial Networks (GANs) for generating synthetic data to augment real data
5. Transfer learning for stock market prediction
6. Multi-modal fusion of different types of data (e.g., financial statements, news articles, social media sentiment) for improved predictions
7. Explainable AI for understanding the reasoning behind the predictions
8. Stock market sentiment analysis using deep learning techniques
9. Portfolio optimization using deep learning models
10. Comparative studies of different deep learning architectures for stock market prediction.

Please provide me with a list of relevant publications, including research papers, articles, and blog posts. Thank you!

LLMs with RAG

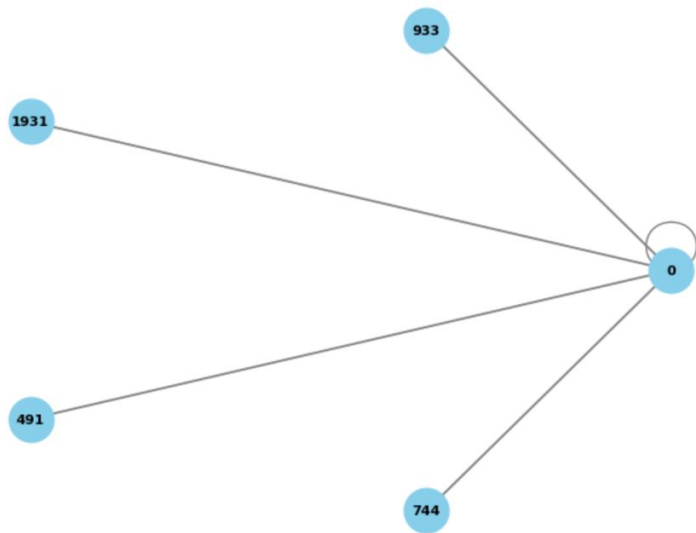
Here are some publications related to using deep learning techniques for stock prediction based on the context you provided:

1. "Stock Trend Prediction using Deep Learning Approach on Technical Indicator and Industrial Specific Information" by J. Liu et al. (2019)
2. "Deep Learning for Stock Market Prediction Using Event Embedding and Technical Indicators" by Y. Zhang et al. (2020)
3. "Explainable Deep Learning for Thai Stock Market Prediction Using Textual Representation and Technical Indicators" by P. Tongkunak et al. (2020)

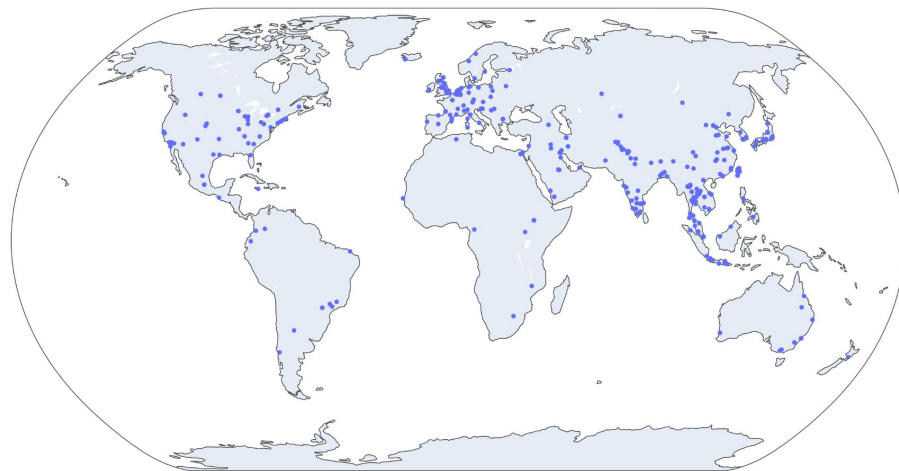
Please note that these are just a few examples of relevant publications, and there may be many more out there. Additionally, it's important to keep in mind that while deep learning techniques have shown promise in stock market prediction, no single method can guarantee accurate predictions, and it's always important to do your own research and consult multiple sources before making investment decisions.

Visualization

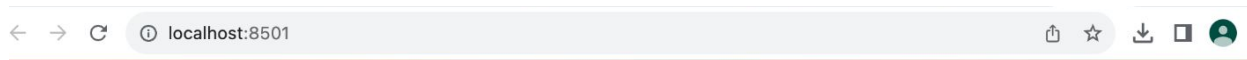
Graph Network



Geospatial Dashboard



Graph Network - Cosine Similarity

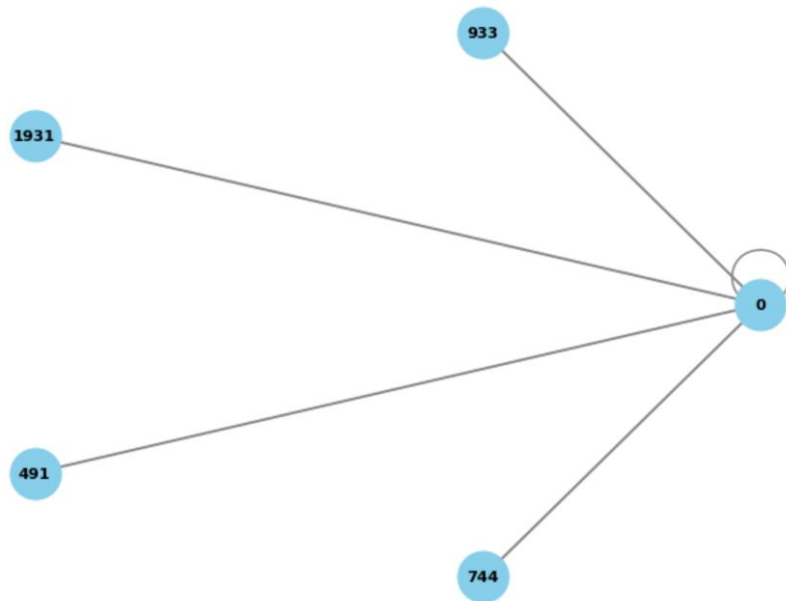


Place the abstract of the research that captures your interest here



```
1 # get text input from Streamlit interface
2 import streamlit as st
3
4 prompt2 = st.text_area("Place the abstract of the research that captures your interest here")
5 if prompt2:
6     st.write(prompt2)
7     # embed
8     from langchain.embeddings import HuggingFaceEmbeddings
9     inference_api_key = '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'
10    from langchain.embeddings import HuggingFaceInferenceAPIEmbeddings
11    embeddings = HuggingFaceInferenceAPIEmbeddings(
12        api_key=inference_api_key, model_name="morenolq/text-cs-scibert"
13    )
```

Graph Network



```
[ ] from transformers import AutoTokenizer, DebertaV2Model, DebertaV2ForSequenceClassification
import torch

tokenizer = AutoTokenizer.from_pretrained('morenolq/thext-cs-sciBERT')
model = AutoModel.from_pretrained('morenolq/thext-cs-sciBERT').cuda()
```

```
[ ] from tqdm import tqdm
vectors = list()
for i in tqdm(train_df.values[:3000]):
    text = i[3]
    # print(text)
    # break
    input_ids = tokenizer.encode(text, max_length=512, truncation=True, padding="max_length", return_tensors="pt").cuda()
    with torch.no_grad():
        embeddings = model(input_ids)
        numvec = embeddings[0][0].detach().cpu().numpy()
        # vectors.append([i[0], numvec])
        vectors.append(numvec)
```

```
[ ] import numpy as np
from sklearn.metrics.pairwise import cosine_similarity

def find_most_similar_vectors(new_vector, all_vectors, top_n=5):
    # Reshape all_vectors to 2D
    all_vectors_flat = all_vectors.reshape(all_vectors.shape[0], -1)

    # Flatten the new_vector
    new_vector_flat = new_vector.flatten()

    # Calculate cosine similarity between the new vector and all vectors
    similarities = cosine_similarity([new_vector_flat], all_vectors_flat)[0]

    # Get indices of top N most similar vectors
    top_indices = similarities.argsort()[::-1][:top_n]

    # Get the actual vectors corresponding to the top indices
    most_similar_vectors = all_vectors[top_indices]

    return most_similar_vectors, top_indices, similarities[top_indices]

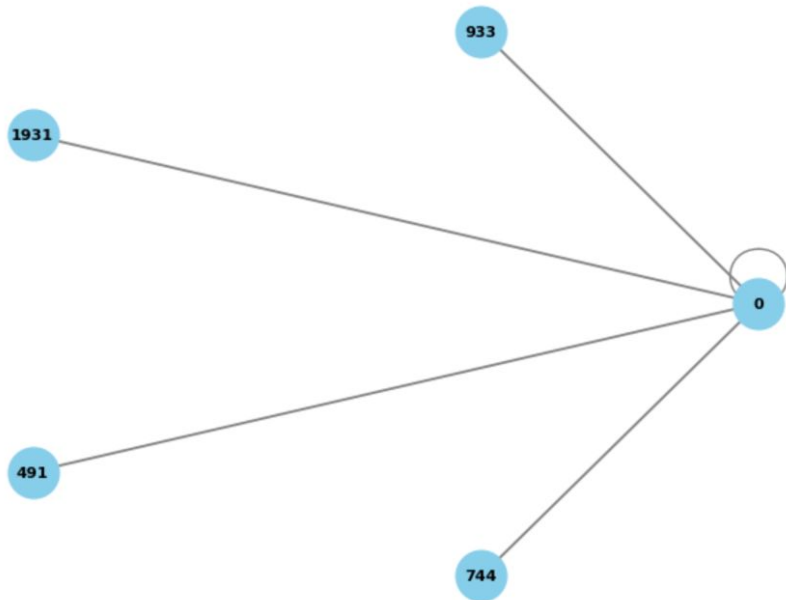
# Load the numpy array from 'test0.npy'
d = np.stack(vectors)

# Assuming the shape of 'all_vectors' is [10, 512, 768]
all_vectors = d

# Extract a new vector (you may want to modify this based on your actual use case)
new_vector = d[0][::]

# Find the 5 most similar vectors
most_similar_vectors, top_indices, top_similarities = find_most_similar_vectors(new_vector, all_vectors, top_n=5)
```

Graph Network - Cosine Similarity



```
import numpy as np
import networkx as nx
import matplotlib.pyplot as plt

# Calculate distances as 1/x
distances = 1 / numpy_array1

# Create a graph
G = nx.Graph()

# Add nodes to the graph with labels from numpy_array2
for label in numpy_array2:
    G.add_node(label)

# Add edges with weights based on the distances
for i, node_label in enumerate(numpy_array2):
    G.add_edge(0, node_label, weight=distances[i])

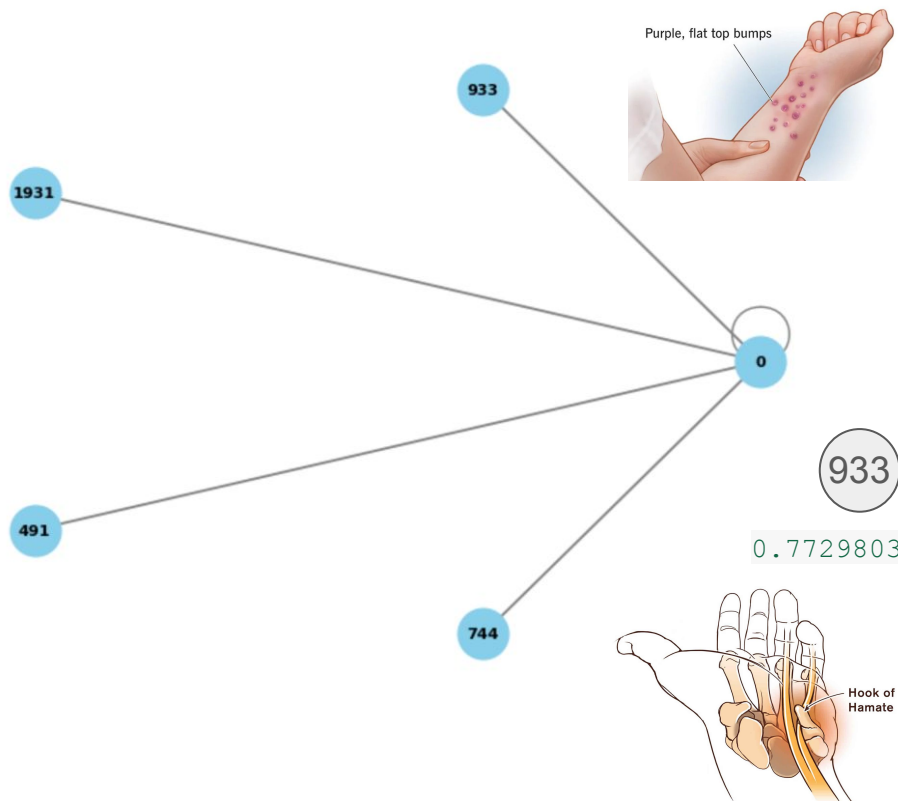
# Define positions for the nodes (circular layout for simplicity)
pos = nx.circular_layout(G)

# Extract edge weights for plotting
edge_weights = [G[u][v]['weight'] for u, v in G.edges]

# Draw the graph
nx.draw(G, pos, with_labels=True, font_weight='bold', node_size=700, node_color='skyblue', font_size=8, width=edge_weights, edge_color='gray', cmap=plt.cm.)

# Show the plot
plt.show()
```

Graph Network



0

© 2021 British Association of Dermatologists. This retrospective study performed to investigate direct immunofluorescence (DIF) findings in oral and cutaneous lichen planus (LP), and to identify any differences between them. There were 147 patients with a definite diagnosis of LP by clinical and histological criteria, with 87 (59.2%) oral and 60 (40.8%) cutaneous specimens. Of these, 125 (85%) had positive DIF findings, with positive yields being significantly lower in oral (79.3%) than in cutaneous (93.3%) LP. Deposition of immunoreactants at the dermoepidermal junction (DEJ) was significantly greater in oral than in cutaneous LP, and fibrin deposition with shaggy pattern at the DEJ was also significantly greater in oral than in cutaneous LP. Deposition of immunoreactants at colloid bodies (CBs) with or without DEJ was significantly greater in cutaneous than in oral LP. IgM deposition at CBs was commonly detected in both groups. We propose that fibrin deposition with shaggy pattern at the DEJ is the best diagnostic indicator of oral LP.

933

© 2022 The Authors. Fractures of the hamate hook are common among professional athletes. The recommended treatment for this is hamate hook excision. The purpose of this study is to evaluate the hand grip strength after hamate hook resection at different levels. Six cadaver forearm flexor digitorum profundus tendons were loaded with 5- to 15-kilograms force and grip strength was subsequently measured. The same measurements were performed in five hamate hook conditions: normal, one-third, two-thirds, total hamate excision, and after perihamate ligament reconstruction. Multilevel mixed-effect models were used to calculate the scaling ratios after each surgical intervention and compared them to a normal hamate hook. A 25%, 36%, 47% reduction, and 7% increase (107% of baseline) in grip strength was found after one-third, two-thirds, total bone was resected, and after perihamate ligament reconstruction, respectively. The study shows an association between grip strength reduction and the level of hamate hook resection. Perihamate ligament reconstruction is recommended as it restores grip strength to normal.

0.7729803

744

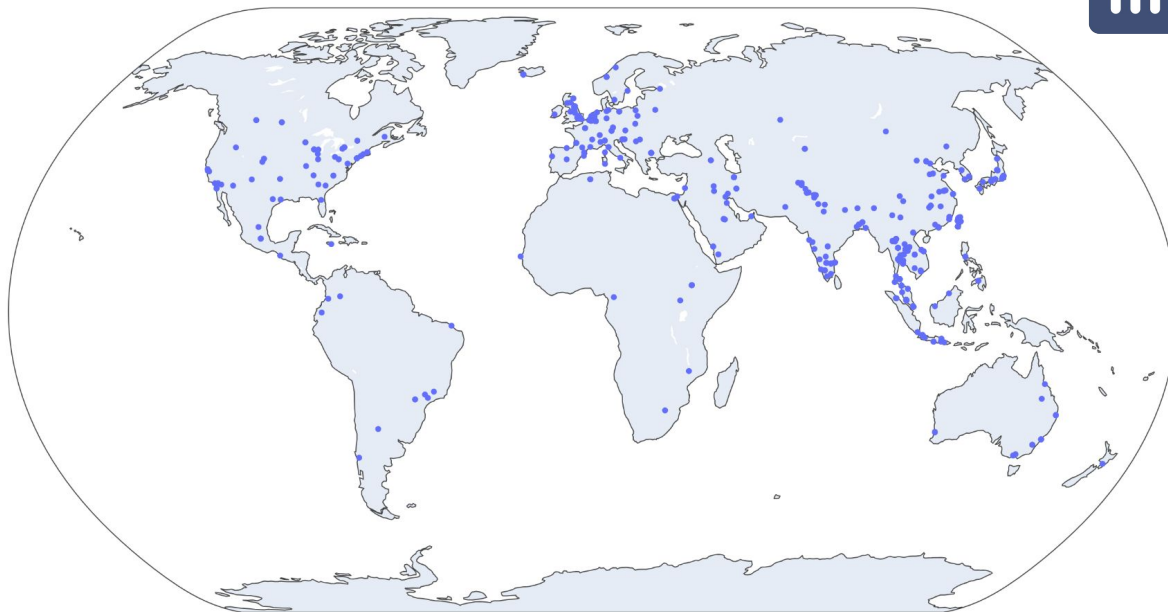
<https://healthjade.net/hamate-fracture/>

<https://my.clevelandclinic.org/health/diseases/17723-lichen-planus>

Global Research Insights Hub



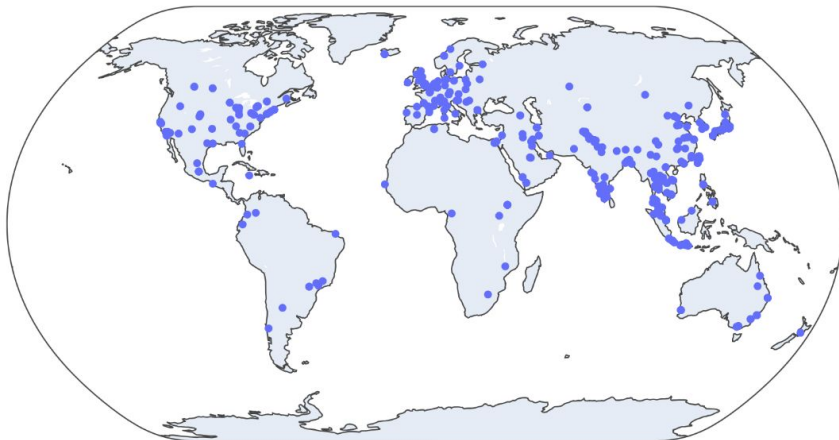
Geographic Scatter Plot



Global Research Insights Hub



Geographic Scatter Plot



```
1 import dash
2 from dash import html, dcc, callback, Output, Input
3 import plotly.express as px
4 import pandas as pd
5 from langchain.embeddings import HuggingFaceInferenceAPIEmbeddings
6
7 # Load CSV data
8 df = pd.read_csv('cleaned_data.csv', sep='|')
9
10 # Create a scatter_geo plot
11 fig = px.scatter_geo(
12     df,
13     lat='latitude',
14     lon='longitude',
15     projection='natural earth', # Choose a map projection
16     title='Geographic Scatter Plot',
17     template='plotly',
18 )
19
20 # Increase the size of the figure
21 fig.update_layout(height=600, width=800) # Adjust the height and width as needed
22
23 # Initialize the Dash app
24 app = dash.Dash(__name__)
25
26 # Define the layout of the app
27 app.layout = html.Div([
28     html.H1("Global Research Insights Hub"), # Add the header here
29     dcc.Graph(figure=fig)
30 ])
31
32 # Run the app
33 if __name__ == '__main__':
34     app.run_server(debug=True)
35
```