DRS

C011007 강지호 C144018 조명진

Agenda

0. Simple Information

1. Current Progress

2. Challenges

3. End-of-Semester Plan

Direction

Pattern-Based Stock Search Model:
Draw a Chart → Find Matching Stocks

❌ Not a numbers-based investing tool
✓ A new stock exploration method centered on chart shapes

❌ Not a service only for experienced investors
✓ A new investing experience that anyone can use intuitively and visually

Goal

Final Goal: Build and Deploy a Real-Time Graph-Based Stock Recommendation Engine

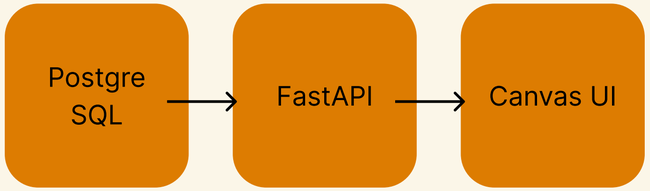
- Store and maintain segment-based vectors for 3,000–4,000 stocks
- Compare user-drawn sketches with precomputed vectors using FastDTW in real time
- Return Top-K similar stocks within 3 seconds
- Provide a web UI where users can draw patterns and view matching stocks on charts

Approach

- Large-scale automated stock price ingestion
  - NASDAQ API
    : real-time automatic ticker collection
  - Yfinance
    : automated OHLCV price data ingestion
- Graph-segment–based pattern comparison
- Shape similarity computed using FastDTW
- Data normalization, missing-value handling, and incremental updates
- FastAPI-based backend + Canvas UI integration

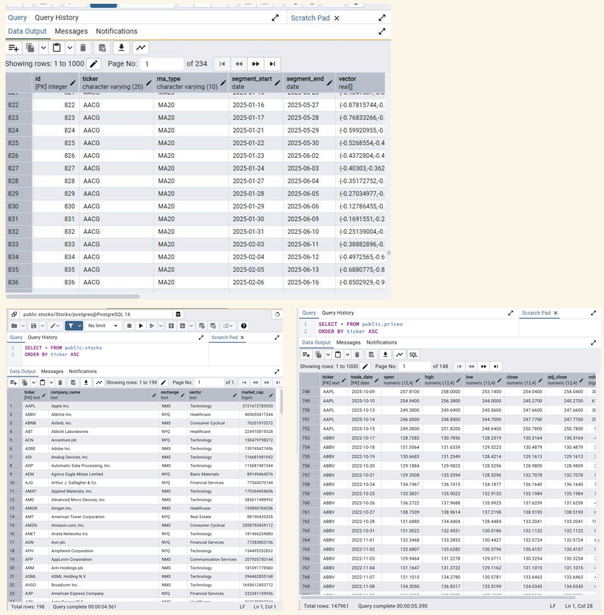
System Design

Store price and vector data for 1,000 tickers in PostgreSQL (pgvector)
➡️ FastAPI server generates a sketch vector from user input
➡️ Query pre-stored ticker vectors from the database
➡️ Perform Top-K similarity search using FastDTW / Cosine similarity
➡️ Return results to the Canvas UI in real time



Evaluation

1. Accuracy Evaluation
   - Input partial segments of the same stock
     → Measure the ratio of cases where the correct stock appears in the recommendations
   - Evaluate performance using traditional chart pattern examples as a comparison baseline
2. Performance Evaluation
   - Measure FastDTW search speed (target: under 3 seconds)
   - Run scaling tests to observe how response time changes as the dataset grows
3. Service Evaluation
   - Conduct UI testing for usability and clarity
   - Collect feedback from real user sketches and interaction patterns



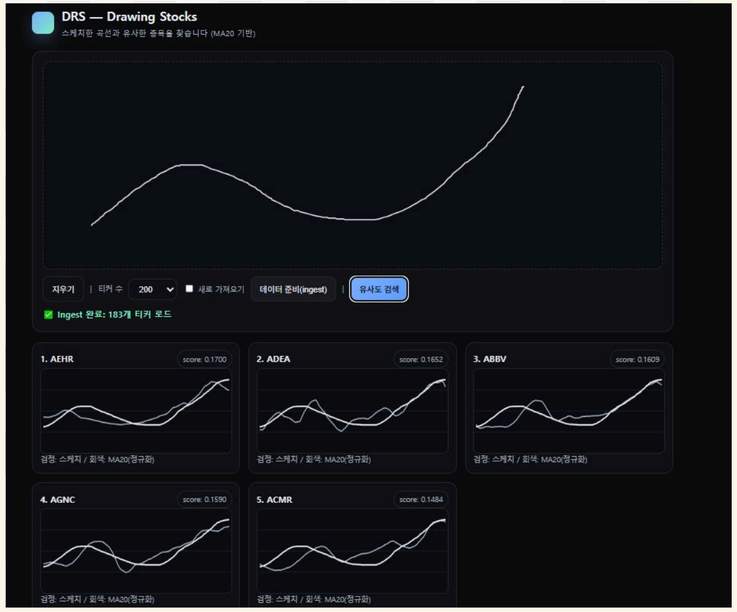
DataBase

Store 1,000 stock profiles, daily price data, and graph vector representations

```python
def dtw_distance(a: np.ndarray, b: np.ndarray) -> float:
    try:
        d, _ = fastdtw(a, b)
        result = float(d)
        return result if np.isfinite(result) else 0.0
    except Exception as e:
        logger.warning(f"DTW calculation failed: {e}")
        return 0.0
def cosine_sim(a: np.ndarray, b: np.ndarray) -> float:
    try:
        # NaN 체크
        if np.any(np.isnan(a)) or np.any(np.isnan(b)):
            logger.warning("NaN detected in cosine_sim input")
            return 0.0

        norm_a = norm(a)
        norm_b = norm(b)

        # Zero vector 체크
        if norm_a < 1e-10 or norm_b < 1e-10:
            return 0.0

        result = float(np.dot(a, b) / (norm_a * norm_b))
        return result if np.isfinite(result) else 0.0
    except Exception as e:
        logger.warning(f"Cosine similarity calculation failed: {e}")
        return 0.0
```

Server

Compute similarity using FastDTW and Cosine functions



Connect the database with the local server

Data-related

1. API Rate Limits & Request Failures
   → Adjust batch size
   → Add retry logic with backoff and sleep intervals
   → Reprocess failed tickers later

2. Data Quality Issues
   → Remove 0 / NaN values and filter out invalid tickers
   → Automatically exclude stocks with no trading activity for the past N days
   → Run data validation queries/scripts regularly

System-related

1. Automation / Scheduling Issues
   → Use a test mode like python ingest_prices.py --dry-run
   → Automatically print summary logs after each run

2. Performance Bottlenecks
   → Apply index tuning for faster queries
   → Cache frequently used segments and preprocessed results

Similarity-related

1. Ambiguity in Defining Ground Truth
   → Use traditional chart patterns + a manually labeled stock set as the evaluation baseline

2. User Sketches Being Too Rough or Inconsistent
   → Normalize vertical scale and horizontal length
   → Anchor the starting point at 0 and resample the sketch to a fixed length

End-of-Semester Plan

1. Implement a similarity matching system between approximately 3,300 NASDAQ stocks and the user’s input data (based on a 20-day window).

2. Automated stock data updating

3. Target accuracy:
   90%+ using traditional chart-pattern benchmarks

- Ability to adjust the search window beyond the default 20-day period
- Search functionality based on traditional chart patterns