Yuxuan (Wayne) Wang

https://waynewangyuxuan.github.io/personal_site/ || waw009@ucsd.edu

EDUCATION SUMMARY:

University of California, San Diego, Jacobs School of Engineering, San Diego, CA — September 2025 - June 2027
Master of Science, Computer Science

New York University, Tandon School of Engineering, Brooklyn, NY — September 2021 - May 2025
Bachelor of Science**,** Computer Science & Math

TECHNICAL SKILLS:

C/C++ || **Java** || **Python** || **Linux** (Kernel, Shell/Bash) || **Multi-Threaded/Process Programming** (Threads, Memory Management, IPC) || **Parallel Computing** (CUDA) || **Agent/Workflow Development** (LangGraph, n8n, Dify) || **Information Retrieval** (query processing, dense/sparse re-ranking) || **Computer Networking** (TCP/IP, OSI Model, Sockets) ||

EXPERIENCES:

Data Engineering Intern - ByteDance — June 2025 – September 2025
Global Monetization Product and Technology - TikTok Ads Diagnosis
(Java, Python, SQL, Kafka, Hive, Spark, Big Data, TikTok Ads)

- Migrated 20+ ads analytics pipelines (Kafka→Hive/HDFS; ClickHouse→Doris), refactoring schemas/HiveSQL and shipping a Java engine adapter; built automated QA (schema/volume diffs, query-parity, p50/p95 latency tracking) to gate multi-DC cutovers.
- Designed and shipped Intent-vs-Actual Budget Delta metric; re-architected 2 primary and upgraded 3 downstream pipelines (~150 TB backfill) to align budget caliber across delivery stages; exposed results via materialized views/APIs for AdDebug.
- Hardened reliability & performance with idempotent backfills, partition-safe replays, and chat-ops alerts; reduced rollout incidents and query drift, and authored runbooks to onboard partner teams.

Research - New York University — June 2024 – May 2025
Evaluation of Graph-Based Vocabulary Mismatch Solution in Information Retrieval
Supervised by Professor Torsten Suel @ New York University, Tandon School of Engineering
(Information Retrieval, Search Engine, Query Processing, Database, Linux, HPC, Python)

- Conducted a systematic study on impact of seed quality graph-based expansion in LADR (Lexically-Accelerated Dense Retrieval); designed a plug-and-play retrieval pipeline supporting multiple sparse retrieval models as initial seed sets.
- Implemented a graph expansion module for candidate documents using KNN/HNSW; incorporated a vector dimension masking mechanism (PRFDIME) that constructs semantic masks via pseudo-relevance feedback to enhance query representation and improve reranking effectiveness.
- Demonstrated that reranking using only the top-3000 graph-expanded passages via Bi-Encoder/Cross-Encoder models can achieve comparable performance to full-corpus reranking, while improving recall and reducing computational cost.

Data Engineering Intern - CITIC Poly Fund (Guangzhou) — June 2023 - August 2023
(NLP-based News ETL, Web Scraping, Python, AWS, NoSQL)

- Built an end-to-end investment-intelligence platform for the chip/EDA sector—ingesting Wind + multi-site crawlers into a Kafka→Lambda/DynamoDB/S3 pipeline with de-dup, structured field extraction (amount/round/investors), and company-profile enrichment—and delivered an insights layer (event/keyword mining, BERT sentiment, T5 summarization, weekly trends) with a Streamlit dashboard that replaced manual spreadsheets with a department-wide weekly brief.

PROJECTS:

TripPlanner — March 2024 - May 2025

- Built an AWS serverless travel assistant—owning route optimization/mapping and real-time notifications: integrated Amazon Location Service + OSRM for multi-stop routing (TSP heuristic, K-d tree), produced GeoJSON with static map renders to S3, added Redis caching to de-dupe/accelerate repeat routes, and delivered user-tunable alerts via Step Functions/EventBridge/SNS with OpenWeatherMap integration; exposed via API Gateway/Lambda with DynamoDB persistence and an Amplify front end.

ShadowDash — September 2024 - Jan 2025

- Built a C++ DSL that bypasses Ninja's .ninja text parsing by constructing the build DAG in-memory (rule/build/pool); validated on zlib and LLVM, cutting parse time by ~40% (zlib, -O3) and ~2× (LLVM, -O1), with macros & user-defined literals for concise manifests and direct integration with Ninja state.