

## Yuxuan (Wayne) Wang

[https://waynewangyuxuan.github.io/personal\\_site/](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 - May 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:

**Java** || **C++/C** || **Python** || **Linux** || **Agent/Workflow Development** (LangGraph, n8n, Dify) || **Search Engine Optimization** (query processing, dense/sparse re-ranking) || **Cloud** (AWS, Google Cloud) || **Web Development** (Flask, SwiftUI, Reac, RESTful) || **Cloud** (AWS, Google Cloud) || **DevOps Tools** (Docker, Circle-CI, Kubernetes, Prometheus)

### EXPERIENCES:

**Software Engineering Intern - ByteDance** June 2025 – Present  
Global Monetization Product and Technology - TikTok Ads Diagnosis  
(Java, Python, Agentic Workflow, Big Data, TikTok Ads)

- Built and launched an agentic data Copilot to make complex delivery charts self-serve: GA'd chart analysis across in-delivery modules, lifted analysis accuracy from ~60–70% to 80%+ by adding time-series anomaly detection (Kats) and pattern-aware rules.
- Designed and delivered the Prompt + RAG + Agent LLM stack with knowledge-base chunking & semantic retrieval (doubao embeddings), session memory, and Lark Bot/Web Chat entry points; shipped streaming chat APIs and stood up automated evaluation hooks for accuracy.
- Drove adoption on an internal platform with 1,000+ MAU, embedding Copilot into user flows and an on-call bot for SE/TPS; achieved 72% Q&A and 83% summary satisfaction, and began integrating Auto Diagnoser signals to make insights actionable.

**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, 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** September 2024 - Present

- 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.

