Dénes Tornyi

About

pinting.github.io
astameteo@posteo.net
github.com/pinting

I thrive in interdisciplinary environments, as my goal is to integrate different perspectives of existence. I believe that human development is driven more by emotion than by technology alone. I wish to believe in a society where people are not just building blocks, but individuals who actively take part in the system they form.

Born in 1996. My earliest exposure to technology was the single computer in our household without an internet connection. I spent my days playing video games and exploring the file system, clicking on every installed executable and daydreaming about a database engine that could link documents and render multimedia content seamlessly. During the summer of 2006, I stumbled upon the digital manual of the video game "StarCraft" written in HTML3 and was astonished by the rendering capabilities of Internet Explorer 6. Despite having no knowledge of English and no formal training, I learned HTML by dissecting the source code of the manual.

Since that summer, my passion for learning has never faded. I have experimented with various technologies, including cross-platform desktop applications, web applications, embedded applications and video-game projects. Recently, I have focused on the advancements in natural language processing (NLP) and conducted research on constraining LLMs.

Besides working, I am a Linux and FOSS enthusiast; in love with literature, poetry, music, theater and cinema.

References

Nokia Solutions and Networks Kft.

2024 March – 2025 December

Returned to Nokia with the goal of continuing my research on LLMs and RAG systems. Established "Project Octopus" as a centralized project space for publishing and inner-sourcing NLP solutions across the company. Collaborated with various departments to streamline and reduce the internal chaos within R&D operations.

Conducted research in collaboration with **Bell Labs**. Architected and implemented a Go-based interpreter state machine that enforced SQL language constraints during SELECT query generation, ensuring field and table consistency throughout the prediction process. The engine translated grammar rules into an inflatable graph of nodes where each node was decomposed to a token trie (utilizing a cache to reduce computational waste). Navigating the graph (by reaching one of the terminating tokens of the token trie of a node) resulted in a state change which influenced the generation of the children nodes (thus creating the foundation for semantic enforcement). The engine was integrated into a custom logits processor implementation of llama.cpp to guide LLMs toward generating grammatically and semantically correct SELECT queries.

Conducted an experiment on an architecture on PostgreSQL, utilizing PGRX to write Rust-based extensions that minimized HTTP 1.1/SQL communication overhead between services. The design integrated chunking, embedding, and retrieval into a single monolithic database process using fastembed and pgvector. Performance benchmarks revealed that pgvector lagged behind Milvus and Qdrant in search speed which halted further development.

Architected and implemented a microservice based retrieval-augmented generation (RAG) pipeline comprising multiple independent (Go-based) services that coordinated through a shared PostgreSQL database. The entire solution was deployed via Docker containers and used HTTP/1.1 and SQL for inter-service communication.

- The *scraper* services synchronized data from external APIs (Jira, GitLab, Confluence, Zendesk and manual uploads) into PostgreSQL. Attachments (PDF / Word / Excel documents and images) were also fetched and stored in an S3-compatible object storage (MinIO).
- The *processor* services were responsible for transforming scraped raw data into Markdown documents. This included converting HTML to Markdown; summarizing Git differences using LLMs; converting PDF / Word / Excel documents to text using IBM's Docling solution.
- The *validation* service was responsible for IDF-based document scoring, which involved summing the top N highest ranking IDF terms in each of the generated Markdown documents. Only documents exceeding a predefined threshold were retained for further processing.
- The *chunking* service that partitioned documents into optimized fragments for embedding, employing an algorithm that tried to reach the minimum chunk size by slicing between paragraphs, sentences, words or characters in descending priority.

- The *embedding* services generating vector representations: a dense embedding service using Hugging Face TEI (with recent Qwen models) and a sparse embedding service powered by a custom-built BM25 ranking model with single-pass wordification algorithm and TF-IDF dictionary construction.
- The *indexer* service that exported embeddings from PostgreSQL to Qdrant or Milvus vector databases for efficient retrieval.
- The *proxy* service that performed hybrid search (through the indexer service) and implemented the de facto OpenAI LLM interface, enabling any third-party UI to interact with it as a standard LLM.

As the RAG pipeline matured, *gonokia*, a repository of reusable Go packages for cross-project code sharing was published. The most notable package was *archive*, which solved the algorithmically complex challenge of orchestrating multiple concurrent streams into a single stream, enabling parallel log collection from K8s nodes and on-the-fly merging the incoming log streams into a single tape archive (TAR or ZIP). Another key innovation was the *httpdump* package, providing a transparent HTTP transport middleware for capturing, dumping and replaying HTTP network communication. Additionally, reusable API clients for Jira, GitLab, Confluence and Zendesk were built.

ScoutinScience B.V.

2023 August – 2024 January

Designed and implemented a matchmaking system to connect Dutch companies with university graduates by processing company websites and student theses using LLMs.

- As the company was a small startup where cost control was a high priority, the data pipeline was designed to be modular and easily deployable on rented machines, with an output that could be manually imported into the production service (which used PostgreSQL). This meant the pipeline was a separate monolithic application written in Python, centered around an SQLite database. 1st, company websites were scraped using official Dutch company records; 2nd, student theses were scraped from official university databases; 3rd, websites and theses were summarized into concise natural language descriptions using LLMs; 4th, each summary was embedded using SBERT; 5th, an NxN similarity matrix was constructed between companies and students; 6th, the SQLite database was migrated to the production PostgreSQL database.
- The production service utilized a .NET backend (with PostgreSQL) and a React-based frontend. The interface enabled students to discover companies relevant to their BSc/MSc thesis topics.

Beyond the main project, conducted code reviews, refined the company's CI/CD pipeline, and refactored the existing user management architecture to support not only roles but also fine-grained per-user permissions.

Nokia Solutions and Networks Kft.

2022 March – 2023 June

Designed and implemented (in Go using the K8s Operator SDK) an update orchestrator and an update receiver service to handle firmware rollouts for customer-owned hardware. Developed a MinIO-based firmware storage service with synchronization between regional data centers along with the implementation of HTTP range headers on S3 objects and client-side tools for managing firmware entries.

Accedo Broadband HU Kft.

2018 March – 2021 February

Contributed to the development OTT client-side applications for WebOS, Tizen, and Web platforms (for OSN, NRK, ProSieben, Deutsche Telekom, and HBO) using JavaScript with an internally developed library and ReactJS.

Designed and implemented a real-time third-party API validator middleware between the application and the network layer; containerized CI/CD toolset for Tizen and WebOS; an internal service operated inside Amazon AWS to synchronize Salesforce time tracking with the BambooHR calendar.

Youwon Hungary Kft.

2014 July – 2015 May

Contributed to the development of an online second-hand marketplace using TypeScript and AngularJS on the client side, C# and ASP.NET Web API 4 on the server side, and Entity Framework 6 for database management with Apache Lucene indexing.

Designed and implemented a Facebook Messenger like service using WebRTC and SignalR over the existing architecture: enabling audio/video calls, text messages along with their message history and the synchronization of ongoing conversation tabs across devices.