คำนำ

หากจะใช้คำพูดที่ได้ยินกันบ่อยๆ ก็อาจกล่าวได้ว่า หนังสือเล่มนี้คือหนังสือที่ผมอยากให้มีตั้งแต่ตอนที่เริ่มศึกษาเกี่ยวกับฐานข้อมูลเวกเตอร์ (vector database) ในช่วงนั้นผมมองหาแหล่งข้อมูลที่รวบรวมทั้งทฤษฎีพื้นฐาน และการประยุกต์ใช้งานฐานข้อมูลเวกเตอร์เข้าไว้ด้วยกัน แต่กลับไม่พบหนังสือหรือแหล่งข้อมูลที่ต้องการเลย

แหล่งข้อมูลเกี่ยวกับฐานข้อมูลเวกเตอร์ในปัจจุบันกระจัดกระจายอยู่ตามเอกสารของผู้ให้บริการต่างๆ, ในบล็อก, วิดีโอบน YouTube และตัวอย่างโค้ดบน GitHub ซึ่งสิ่งที่ผมต้องการคือ ตัวอย่างการใช้งานที่นำไปใช้ได้จริง มากกว่าที่จะเป็นการอธิบายเชิงทฤษฎีเพียงอย่างเดียว นอกจากนี้ ผมยังชอบแนวทางการเรียนรู้ที่ค่อยๆ เริ่มจากพื้นฐาน ไปสู่ระบบง่ายๆ ที่ใช้งานได้จริง และสามารถนำมาใช้จัดการข้อมูลบนเครื่องของตนเองได้ โดยเฉพาะกับข้อมูลส่วนตัว

ด้วยเหตุนี้ ผมจึงเริ่มรวบรวมข้อมูลจากแหล่งต่างๆ และจากหลายมุมมอง เพื่อนำมารวมให้เป็นองค์ความรู้ที่เป็นระบบ และมีความต่อเนื่อง ซึ่งผลลัพธ์ที่เกิดขึ้นก็คือหนังสือเล่มนี้ โดยผมตั้งใจเขียนขึ้นเพื่อให้เป็นประโยชน์สำหรับนักพัฒนาระดับกลาง ไม่ว่าจะเป็นผู้ที่สนใจทฤษฎีอย่างจริงจัง หรือผู้ที่มีแนวคิดแบบ “เอาโค้ดมาลองเลย” ก็สามารถใช้ประโยชน์ได้

หนังสือเล่มนี้เรียบเรียงเนื้อหาตามลำดับการเรียนรู้ ได้แก่ แนวคิด ⟶ เครื่องมือ ⟶ การประยุกต์ใช้งาน ⟶ ทิศทางในอนาคต โดยแต่ละบทจะต่อยอดจากบทก่อนหน้า พร้อมนำเสนอผ่านตัวอย่างโค้ดที่สามารถใช้งานได้จริง ลำดับเนื้อหาจะค่อยๆ พาผู้อ่านจากการทำความเข้าใจฐานข้อมูลเวกเตอร์ ไปสู่การสร้างแอพพลิเคชันที่ออกแบบมาสำหรับการใช้งานส่วนบุคคล, ทดลองปฏิบัติ และต่อยอด ปิดท้ายด้วยข้อเสนอทางวิสัยทัศน์เกี่ยวกับการกำหนดมาตรฐานอุตสาหกรรมสำหรับภาษา Vector Query Language (VQL) ซึ่งได้รับแรงบันดาลใจจาก SQL

อีกประเด็นหนึ่งที่อาจถือได้ว่าเป็นมุมมองของผู้เขียนก็คือ การให้ความสำคัญกับฐานข้อมูลแบบไฮบริด ที่ผสานฐานข้อมูลเชิงสัมพันธ์ (relational database) เข้ากับความสามารถในการจัดเก็บเวกเตอร์ โดยอาศัยการติดตั้งส่วนขยาย (vector

extension plug-in) ให้กับฐานข้อมูลโอเพนซอร์สที่เป็นที่รู้จักอย่าง SQLite3 และ PostgreSQL ด้วยเหตุผลว่า ฐานข้อมูลทั้งสองระบบนี้เป็นเครื่องมือที่นักพัฒนาฐานข้อมูลส่วนใหญ่คุ้นเคยอยู่แล้ว เพียงเพิ่มชนิดข้อมูล (data type) ใหม่ที่เรียกว่า “vector” ก็สามารถเปิดประตูสู่โลกของการพัฒนาแอพพลิเคชันด้าน AI ได้ทันที ทำให้ผู้พัฒนาเริ่มใช้งานได้ง่าย ขณะเดียวกันก็เรียนรู้เพิ่มเติมน้อยที่สุด



สิ่งสำคัญที่ต้องย้ำคือ แอพพลิเคชันในหนังสือเล่มนี้มีไว้สำหรับการเรียนรู้ภาคปฏิบัติ ปรับแต่งได้ตามต้องการของคุณ แต่ไม่เหมาะกับการใช้ให้บริการบนเว็บที่มีคนใช้จำนวนมาก

สำรวจเนื้อหา

• **บทที่ 1:** แนะนำฐานข้อมูลเวกเตอร์ในฐานะเทคโนโลยีพื้นฐานสำหรับแอพพลิเคชัน AI โดยอธิบายว่าเหตุใดข้อมูลที่ไม่มีโครงสร้าง (ข้อความ, รูปภาพ, เสียง, วิดีโอ) จึงต้องการการค้นหาเชิงความหมายมากกว่าการค้นหาด้วยคีย์เวิร์ด และเปรียบเทียบฐานข้อมูลเวกเตอร์กับฐานข้อมูล SQL และ NoSQL พร้อมกับสร้างความเข้าใจว่า การเอ็มเบดดิ้ง (embedding) สามารถรักษาความหมายได้อย่างไร

• **บทที่ 2:** อธิบายถึงวิวัฒนาการและการใช้งานในปัจจุบันของเวกเตอร์เอ็มเบดดิ้ง เพื่อเชื่อมช่องว่างระหว่างข้อมูลที่ไม่มีโครงสร้าง ไปยังรูปแบบข้อมูลที่เครื่องสามารถประมวลผลได้ โดยกล่าวถึงพื้นฐานทางคณิตศาสตร์ และวิธีที่ความใกล้เคียงของเวกเตอร์สะท้อนถึงความคล้ายคลึงทางความหมาย

• **บทที่ 3:** รู้จักกับ FAISS (Facebook AI Similarity Search) ในฐานะของเครื่องมือค้นหาความคล้ายคลึง พร้อมด้วยชุดเครื่องมือที่ยืดหยุ่นสำหรับสร้างฐานข้อมูลเวกเตอร์

• **บทที่ 4:** สร้างระบบจัดการความรู้ส่วนบุคคล ด้วยการใช้ sqlite-vss ซึ่งเป็นส่วนขยายของ SQLite ที่ใช้ FAISS เพื่อรวมการค้นหาเวกเตอร์เข้ากับโลกของฐานข้อมูลเชิงสัมพันธ์ (relational database)

• **บทที่ 5:** สร้างระบบค้นหาเอกสารทางวิชาการ ซึ่งแสดงให้เห็นว่า เวกเตอร์เอ็มเบดดิ้งสามารถจับความสัมพันธ์เชิงความหมาย และได้ผลลัพธ์ที่การค้นหาด้วยคีย์เวิร์ดหาไม่พบได้อย่างไร

• **บทที่ 6:** พัฒนาระบบ RAG ที่สมบูรณ์ ซึ่งรวมการค้นหาเวกเตอร์เข้ากับความสามารถของโมเดลภาษาขนาดใหญ่ (large language model – LLM) ด้วยการสร้างระบบส่วนตัวแบบโลคอลที่ทำงานบนเดสก์ท็อปเครื่องเดียว โดยใช้ SQLite Vector Similarity Search (SQLite VSS) และ Ollama

• **บทที่ 7:** สร้างระบบ RAG สำหรับเอกสารทางวิทยาศาสตร์ ซึ่งมีความท้าทายเฉพาะตัว ได้แก่ ศัพท์เฉพาะทางเทคนิค, เนื้อหาแบบมีโครงสร้าง, การอ้างอิง และคุณภาพของสิ่งที่นำมาใช้

• **บทที่ 8:** ก้าวไปสู่การพัฒนาระบบค้นหาและ RAG ที่สามารถเข้าใจลักษณะเฉพาะของข้อมูล และการสนทนา ซึ่งสามารถจัดการกับความยุ่งเหยิงของข้อมูลการสนทนาจริงที่มีการโต้ตอบไปมาหลายครั้ง

• **บทที่ 9:** นำเสนอต้นแบบของภาษาสำหรับฐานข้อมูลเวกเตอร์ (Vector Query Language - VQL) ที่ได้รับแรงบันดาลใจจาก SQL เพื่อตอบสนองความต้องการของ 3 กลุ่มคือ นักพัฒนาแอพพลิเคชัน, ผู้ดูแลฐานข้อมูลที่ต้องการเครื่องมือที่ไม่ผูกติดกับผลิตภัณฑ์ และนักวิจัย AI/แมชชีนเลิร์นนิงที่ต้องการการจัดการข้อมูลอย่างมีประสิทธิภาพ

หนังสือเล่มนี้เหมาะกับใคร

กลุ่มผู้อ่านเป้าหมายของหนังสือเล่มนี้ ได้แก่

- นักพัฒนาซอฟต์แวร์ระดับองค์กรที่ต้องการสร้างแอพพลิเคชัน AI ตัวแรกของตนเอง
- นักวิทยาศาสตร์ข้อมูลที่ต้องการพัฒนาการค้นหาเชิงความหมาย
- วิศวกรแมชชีนเลิร์นนิงที่ทำงานกับระบบ RAG
- นักพัฒนาระบบแบ็คเอนด์ที่ต้องการเพิ่มความสามารถด้านเวกเตอร์ให้กับระบบที่มีอยู่

นอกจากนี้ หนังสือเล่มนี้ยังเหมาะสำหรับผู้ต้องการเรียนรู้จากการลงมือปฏิบัติ เนื่องจากมีแอพพลิเคชันเชิงปฏิบัติที่เน้นการใช้งานส่วนบุคคลด้วย

ข้อกำหนดเบื้องต้นสำหรับผู้อ่านได้แก่ ความเชี่ยวชาญในการเขียนโปรแกรม Python, ความเข้าใจพื้นฐานเกี่ยวกับฐานข้อมูลและ SQL และความคุ้นเคยกับการใช้งาน LLM ในเครื่องโลคอล (สำหรับบทที่เกี่ยวกับ RAG) นอกจากนี้ ผู้อ่านควรมีความคุ้นเคยกับแนวคิดทางคณิตศาสตร์เบื้องต้นด้วย

หนังสือเล่มนี้เหมาะกับผู้ต้องการเรียนรู้วิธีสร้างระบบค้นหาเชิงความหมายขนาดเล็กถึงขนาดกลาง, การใช้งานแอพพลิเคชัน RAG ด้วย LLM ในเครื่องของผู้ใช้, การเพิ่มประสิทธิภาพการค้นหาความคล้ายคลึงของเวกเตอร์, การเลือกอินเด็กซ์ที่เหมาะสม และการรวมการค้นหาเวกเตอร์เข้ากับฐานข้อมูลเชิงสัมพันธ์ (relational database)

หนังสือเล่มนี้เน้นการใช้งานภาษา Python และไม่ได้กล่าวถึงบริการฐานข้อมูลเวกเตอร์บนระบบคลาวด์ แต่แอพพลิเคชันและทฤษฎีที่กล่าวถึงก็เพียงพอสำหรับการเริ่มต้นและใช้งานข้อมูลส่วนบุคคล นอกจากนี้ ยังไม่ใช่หนังสือที่สอนวิธีสร้างแอพพลิเคชันฐานข้อมูลเวกเตอร์สำหรับให้บริการบนเว็บ อย่างไรก็ตาม บทเรียนในแต่ละบทสามารถใช้เป็นพื้นฐานและจุดเริ่มต้นในการก้าวไปสู่งานขนาดใหญ่, หลากหลาย และซับซ้อนขึ้นได้

ซอฟต์แวร์, สภาพแวดล้อม และทรัพยากรที่ต้องใช้

แอพพลิเคชันตัวอย่างจะทำงานบนคอมพิวเตอร์ทั่วไปที่ใช้พัฒนาซอฟต์แวร์ ซึ่งมีองค์ประกอบดังนี้

- Python เวอร์ชัน 3.11+
- PostgreSQL, SQLite3, FAISS, Ollama
- ระบบปฏิบัติการ: Linux, macOS หรือ WSL บน Windows
- หน่วยความจำอย่างน้อย 16 GB แต่ควรมี 24 GB
- สำหรับการรัน PostgreSQL และโมเดล 8B LLM ต้องมีพื้นที่ในฮาร์ดดิสก์ 512 GB ถึง 1 TB

สารบัญ

บทที่ 1 บทนำสู่ฐานข้อมูลเวกเตอร์ 23

ทำไมเราต้องใช้ฐานข้อมูลเวกเตอร์? 23

ข้อมูลชนิดใหม่: เวกเตอร์ 25

การค้นหาความคล้ายคลึง 26

ข้อมูลแบบเวกเตอร์มีความแตกต่างอย่างไร? 30

เราใช้ฐานข้อมูลเวกเตอร์กับอะไรบ้าง? 32

เปรียบเทียบฐานข้อมูล SQL กับฐานข้อมูลเวกเตอร์ 34

พื้นฐานของคณิตศาสตร์ทางธุรกิจ: คณิตศาสตร์ทางการบัญชี 34

การเก็บเวกเตอร์ในระบบจัดการฐานข้อมูลเชิงสัมพันธ์ 35

ความจำเป็นในการมีความสามารถด้านเวกเตอร์โดยเฉพาะ 36

เปรียบเทียบฐานข้อมูล NoSQL กับฐานข้อมูลเวกเตอร์ 37

ฐานข้อมูล NoSQL กับการเป็นที่เก็บเวกเตอร์ 37

ข้อจำกัดของส่วนขยายด้านเวกเตอร์ในฐานข้อมูล NoSQL 38

เมื่อใดควรเลือกใช้ NoSQL ที่มีส่วนขยายด้านเวกเตอร์ 38

แนวทางแบบไฮบริด: มีทั้งข้อมูลที่มีโครงสร้างและข้อมูลเวกเตอร์ 39

ความต้องการข้อมูลเวกเตอร์และข้อมูลเมตา 39

ข้อจำกัดของการจัดเก็บในรูปแบบเวกเตอร์อย่างเดียว 40

สถาปัตยกรรมฐานข้อมูลแบบไฮบริด 40

ตัวอย่างของคิวรีแบบไฮบริด 42

ประโยชน์ของแนวทางแบบไฮบริด 42

สรุป 43

บทที่ 2 เอ็มเบดดิ้ง 45
ทำความเข้าใจกับเวกเตอร์เอ็มเบดดิ้ง 45
Word2Vec: จุดเปลี่ยนที่สำคัญ 47
Doc2Vec: จากคำสู่เอกสาร 50
จากเอ็มเบดดิ้งสู่โมเดลภาษาสมัยใหม่ 54
Transformer ที่ทำหน้าที่เข้ารหัสเพียงอย่างเดียว (โมเดลในกลุ่ม BERT) 54
Transformer ที่ทำหน้าที่ถอดรหัสเพียงอย่างเดียว (โมเดลตระกูล GPT) 55
Transformer ที่ทำหน้าที่เข้ารหัสและถอดรหัส (โมเดล T5 และ BART) 56
โมเดลสำหรับเอ็มเบดดิ้ง: ผู้มีหน้าที่สร้างเวกเตอร์ 58
ความแตกต่างจากโมเดลแบบดั้งเดิม 58
บทบาทต่อแอพพลิเคชัน LLM สมัยใหม่ 59
การประยุกต์ใช้และกรณีการใช้งาน 60
ตัวอย่างไปป์ไลน์สำหรับ RAG 61
sentence-transformers: ไลบรารีอเนกประสงค์สำหรับเอ็มเบดดิ้งข้อความ 65
รายละเอียดสำหรับแนวทางในการใช้ SentenceTransformers 70
เอ็มเบดดิ้งเลเยอร์: ประตูสู่ Zero-Shot Learning 76
องค์ประกอบในเอ็มเบดดิ้งของ Transformer 77
ความเชื่อมโยงกับ Zero-Shot Learning 79
คุณลักษณะสำคัญที่ทำให้เกิด Zero-Shot Learning 81
ข้อจำกัดและข้อควรพิจารณา 82
พัฒนาการและแนวโน้มล่าสุด 84
สัมผัสกับคณิตศาสตร์ของเวกเตอร์ด้วย Word2Vec 84
ขั้นตอนที่ 1: การเตรียมและติดตั้ง 84

ขั้นตอนที่ 2: โหลดโมเดล Word2Vec ที่ผ่านการเทรนแล้ว 85
ขั้นตอนที่ 3: สร้างฟังก์ชันทางคณิตศาสตร์ของเวกเตอร์ 86
ขั้นตอนที่ 4: การคำนวณ “King–Queen” ที่คลาสสิก 87
ขั้นตอนที่ 5: การคำนวณอื่นๆ ที่น่าสนใจ 88
ขั้นตอนที่ 6: เครื่องมือทดลองแบบอินเตอร์แอ็คทีฟ 89
คำปิดท้ายเกี่ยวกับคณิตศาสตร์เวกเตอร์ 90
สรุป 90

บทที่ 3 การค้นหาความคล้ายคลึงกันด้วย FAISS 93
เริ่มจากรากฐาน 94
การแทนข้อมูลด้วยเวกเตอร์ 96
มาตรวัดระยะทาง 97
แนวทางในการเลือกใช้ 100
อินเด็กซ์ของ FAISS 100
อินเด็กซ์แบบแฟลต 100
อินเด็กซ์แบบ IVF-based 101
อินเด็กซ์แบบ LSH-based 104
อินเด็กซ์แบบ HNSW-based 104
อินเด็กซ์แบบพิเศษ 105
อินเด็กซ์แบบผสมและแปลงรูปแบบ 106
การเลือกอินเด็กซ์ที่เหมาะสม 106
การควอนไทซ์ 110
SQ 111
PQ 114
ปัญหา ANN 118
ปัญหา 120
ลดต้นทุนในการคำนวณ 121
เทคนิค ANN ที่สำคัญใน FAISS 121

การเลือกใช้อินเด็กซ์ของ FAISS 124
ตัวอย่างการใช้งาน 125
ทำความเข้าใจกับอินเด็กซ์ HNSW 126
HNSW คืออะไร? 127
HNSW ทำงานอย่างไร? 128
คำอธิบายถึงพารามิเตอร์ที่สำคัญ 130
ตัวอย่างการใช้งาน: สร้างระบบค้นหาความคล้ายคลึง 131
วัดประสิทธิภาพด้านการทำงาน 133
แนวทางปฏิบัติที่ดีที่สุด 134
สถาปัตยกรรมและองค์ประกอบของ FAISS 134
รากฐาน 134
แนวคิดหลัก 137
องค์ประกอบที่สำคัญ 137
ขั้นตอนการทำงานทั่วไป 140
ตัวอย่างโค้ดแสดงขั้นตอนการใช้งาน 140
ประเด็นสำคัญต่างๆ 141
การค้นคว้าเพิ่มเติม 142
สรุป 142

บทที่ 4 การค้นหาเชิงความหมายด้วย SQLite3 145
ทำความเข้าใจกับส่วนขยาย SQLite Vector Similarity Search 146
ความสามารถหลัก 146
ภาพรวมของสถาปัตยกรรม 148
ข้อจำกัด 149
การเตรียมสภาพแวดล้อมสำหรับการพัฒนา 150
การติดตั้งองค์ประกอบที่จำเป็น 150
การตรวจสอบการติดตั้ง 150
ตัวกำหนดการทำงาน 152

การออกแบบสคีมาของฐานข้อมูล 152
ความต้องการของสคีมา 152
การกำหนดตาราง 153
การตัดสินใจในการออกแบบสคีมา 155
เชื่อมต่อกับ Reddit ด้วย Python Reddit API Wrapper 156
การสร้างข้อมูลรับรองสำหรับ Reddit API 156
การสร้างไคลเอนต์ PRAW 157
ตัวอย่างการใช้งาน 160
การแยกเนื้อหาและประมวลผลเบื้องต้น 160
ไปป์ไลน์ของการคลีนข้อความ 160
การคัดกรองคุณภาพ 164
การสร้างและจัดเก็บเอ็มเบดดิ้ง 164
ตัวสร้างเอ็มเบดดิ้ง 165
ที่เก็บฐานข้อมูล 167
ไปป์ไลน์การประมวลผลแบบแบตช์ 171
การสร้างอินเด็กซ์ของเวกเตอร์ 173
ทำความเข้าใจกับการทำอินเด็กซ์ของ VSS 173
การจัดการอินเด็กซ์ 174
สร้างการค้นหาเชิงความหมาย 177
ที่เก็บผลลัพธ์จากการค้นหา 178
ตัวค้นหาหรือเสิร์ชเอนจิน 179
รวมทั้งหมดเข้าด้วยกัน 186
ตัวอย่างของโฟลว์การทำงาน 186
ตัวอย่างของเอาต์พุต 189
ส่วนขยาย: การทำอินเด็กซ์แบบเพิ่มเติม 190
สรุป 192

บทที่ 5 สร้างระบบค้นหาบทความวิชาการด้วย PostgreSQL pgvector 195
ความท้าทายในการค้นหาบทความวิชาการ 195
ทำไม ArXiv จึงเป็นแหล่งข้อมูลที่เหมาะสม 196
กรณีการใช้งานจริง 196
เหตุผลในการเลือกเทคโนโลยี 197
ภาพรวมของสถาปัตยกรรม 198
องค์ประกอบของระบบ 198
ดาต้าโฟลว์ 200
ปรัชญาในการออกแบบ 201
การตั้งค่าสภาพแวดล้อม และส่วนประกอบต่างๆ ที่จำเป็น 202
การติดตั้ง PostgreSQL และ pgvector 202
การกำหนดค่าสภาพแวดล้อม Python 204
โครงสร้างไดเรกทอรีและการกำหนดค่า 206
การตรวจสอบและการทดสอบ 207
การออกแบบฐานข้อมูลสำหรับเปเปอร์ทางวิทยาศาสตร์ 208
หลักในการออกแบบสคีมา 208
โครงสร้างของตารางหลัก 210
กลยุทธ์ในการจัดเก็บเวกเตอร์ 214
กลยุทธ์ในการอินเด็กซ์ 215
การทำงานกับ ArXiv และการจัดการไฟล์ PDF 216
สร้างไคลเอนต์สำหรับ API ของ ArXiv 216
ไปป์ไลน์สำหรับการดาวน์โหลดไฟล์ PDF 218
ระบบประมวลผลแบบแบตช์ 220
การสกัดและประมวลผลข้อความจากไฟล์ PDF 222
ความท้าทายในการสกัดไฟล์ PDF 222
การแบ่งก้อนข้อความอย่างชาญฉลาด 225

การสร้างเอ็มเบดดิ้งและการจัดเก็บ 227
กลยุทธ์การเลือกโมเดลสำหรับเอ็มเบดดิ้ง 227
ไปป์ไลน์การประมวลผลแบบแบตช์ 228
สร้างการค้นหาด้วยความคล้ายคลึงกัน 230
แอพพลิเคชันแบบโต้ตอบและอินเทอร์เฟซกับผู้ใช้ (UI) 233
แพ็กเกจของ Docker สำหรับการใช้งานแบบโลคอล 235
สถาปัตยกรรมคอนเทนเนอร์ 235
การกำหนดค่าของ Docker Compose 236
สคริปต์สำหรับเริ่มต้นฐานข้อมูล 238
เวิร์คโฟลว์ในการพัฒนา 239
การออกแบบที่พร้อมสำหรับคลาวด์ 239
การปรับแต่งประสิทธิภาพเบื้องต้น 240
การกำหนดค่าอินเด็กซ์ 240
ประสิทธิภาพของคิวรี 240
การจัดการทรัพยากร 241
ขั้นตอนต่อไป 241
ข้อจำกัดในปัจจุบัน 241
แนวคิดในการเพิ่มความสามารถให้กับแอพพลิเคชัน 241
เราได้ลงมือสร้างอะไรไปบ้าง 242
สิ่งที่ระบบทำสำเร็จ 243
ทักษะทางเทคนิคที่ได้รับ 243
เครื่องมือวิจัยที่ใช้งานได้จริง 243
รากฐานสำหรับระบบที่ก้าวหน้าขึ้น 244
ศักยภาพในอนาคต 244
สรุป 245

บทที่ 6 การสร้างระบบ RAG ด้วย SQLite VSS และ Ollama 247
ภาพรวมสถาปัตยกรรมของระบบ 248
ฐานข้อมูลที่รองรับเวกเตอร์ 249
จัดเตรียมฐานข้อมูลที่ใช้งานเวกเตอร์ได้ 250
การออกแบบสคีมาสำหรับ RAG 251
การสร้างอินเด็กซ์สำหรับค้นหา 252
การประมวลผลข้อความและสร้างเอ็มเบดดิ้ง 254
การจัดการโมเดลเอ็มเบดดิ้ง 254
การแบ่งข้อความออกเป็นก้อน 255
จัดเก็บเนื้อหาด้วยเอ็มเบดดิ้ง 256
สร้างการค้นหาแบบไฮบริด 258
อัลกอริทึมของการค้นหาแบบไฮบริด 259
ส่วนประกอบของการค้นหาเชิงความหมาย 260
ส่วนประกอบของการค้นหาด้วยคีย์เวิร์ด 261
การรวมคะแนนและจัดอันดับ 262
การรวมเข้ากับ LLM 263
Ollama API 263
ฟังก์ชันตรวจสอบสถานะการทำงาน 264
ไปป์ไลน์ของ RAG 265
การจัดรูปแบบของบริบท 266
ไปป์ไลน์สำหรับการตอบคำถาม 267
ลองใช้งานและทดสอบ 269
การโหลดข้อมูลตัวอย่าง 270
ฟังก์ชันหลักสำหรับสาธิตการทำงาน 271
ส่วนอินเทอร์เฟซกับผู้ใช้ 272

ฟังก์ชันช่วยทดสอบอย่างรวดเร็ว 273
ขั้นตอนต่อไป: การขยายระบบ 273
คุณสมบัติอื่นๆ ที่ขาดหายไป 274
การเพิ่มประสิทธิภาพการทำงาน 276
ข้อควรพิจารณาสำหรับระบบที่ใช้งานจริง 277
รูปแบบของ RAG ระดับสูง 277
สรุป 278

บทที่ 7 สร้างระบบ RAG สำหรับเอกสารทางวิทยาศาสตร์ด้วย PostgreSQL และ pgvector 281
เป้าหมายและความสามารถของระบบ 282
ภาพรวมของสถาปัตยกรรม 282
สร้างรากฐานของฐานข้อมูลด้วย pgvector 285
การกำหนดค่าและตั้งค่าฐานข้อมูล 285
การออกแบบสคีมาสำหรับเอกสารทางวิทยาศาสตร์ 286
อินเด็กซ์เวกเตอร์ประสิทธิภาพสูง 288
กลยุทธ์ในการสร้างเอ็มเบดดิ้ง 289
การใช้ ArXiv และการประมวลผลไฟล์ PDF 290
การค้นงานวิจัยด้วย ArXiv API 290
การสกัดข้อความจาก PDF 291
การแบ่งข้อความเป็นก้อนๆ ระดับสูง 295
ไปป์ไลน์การจัดเก็บเอ็มเบดดิ้ง 295
การค้นหาเชิงความหมายหลายระดับ 298
การค้นหาระดับบทคัดย่อ 299
การค้นหาในระดับส่วนของเอกสาร 300
เจาะรายละเอียดของไปป์ไลน์ RAG 302
ผสานเข้ากับโลคอล LLM 302
การตรวจสอบสถานะและค้นหาโมเดลที่พร้อมใช้งาน 303

การดึงบริบทอย่างชาญฉลาด 304
วิศวกรรมพรอมต์สำหรับงานทางวิทยาศาสตร์ 305
ไปป์ไลน์การทำงานของ RAG ที่สมบูรณ์ 306
การสาธิตและส่วนติดต่อกับผู้ใช้แบบโต้ตอบ 308
ลำดับขั้นตอนในการสาธิต 309
การสาธิตการค้นหา 310
การสาธิต RAG 311
ส่วนติดต่อกับผู้ใช้สำหรับการค้นหาแบบโต้ตอบ 312
จุดเริ่มต้นที่สามารถเลือกโหมดได้ 313
หมายเหตุทางเทคนิคเกี่ยวกับ HNSW 314
วิธีประเมินผลลัพธ์ 315
ขั้นต่อไป: การขยายระบบ 315
สรุป 319

บทที่ 8 สร้างระบบค้นหาบทสนทนาและ RAG 321
เป้าหมายและความสามารถของระบบ 322
ภาพรวมสถาปัตยกรรมของระบบ 323
สิ่งที่เราจะลงมือสร้าง 326
ฐานข้อมูลสำหรับจัดเก็บข้อมูลการสนทนา 326
การออกแบบสคีมาสำหรับบทสนทนา 327
สถาปัตยกรรมแบบ 3 ตารางเพื่อประสิทธิภาพสูงสุด 328
การทำอินเด็กซ์เวกเตอร์อย่างมีประสิทธิภาพ 329
การอิมพอร์ตบทสนทนา และไปป์ไลน์การประมวลผลข้อมูล 330
การนำเข้าข้อมูล JSON ที่จัดการข้อผิดพลาดได้ 331
การประมวลผลทรานแซ็กชัน 332
การประทับเวลาและตรวจสอบความถูกต้องของข้อมูล 332
การกู้คืนข้อผิดพลาดและการบันทึกล็อก 333

การสร้างเอ็มเบดดิ้งอย่างมีประสิทธิภาพ และการประมวลผลแบบแบตช์ 335
แพตเทิร์น singleton สำหรับการจัดการโมเดล 335
กลยุทธ์การค่อยๆ ประมวลผลเพิ่มเติม 336
การประมวลผลแบบแบตช์ 336
การเพิ่มไปยังฐานข้อมูลพร้อมการจัดการความขัดแย้ง 337
การค้นหาเชิงบริบทที่เข้าใจบทสนทนา 339
การค้นหาความคล้ายคลึงเชิงความหมาย 339
การรวมหลายตารางเพื่อให้ได้บริบทที่สมบูรณ์ 339
โครงสร้างและการฟอร์แมตผลลัพธ์ 340
การดึงบริบทของบทสนทนา 341
การคำนวณหน้าต่างบริบท 342
การผสานเข้ากับ RAG 343
การจัดการบริบทที่มีโครงสร้าง 343
รวมการทำงานกับโลคอล LLM 343
การเฝ้าติดตามสถานะของระบบและโมเดล 344
การดึงบริบท และนำมาประกอบกัน 345
วิศวกรรมพรอมต์เชิงการสนทนา 345
ไปป์ไลน์ RAG พร้อมการติดตามประสิทธิภาพ 346
สร้าง Web API ด้วย FastAPI 348
โครงสร้างแอพพลิเคชันของ FastAPI 349
โมเดลคำขอพร้อมการตรวจสอบความถูกต้อง 349
สร้างการค้นหาที่เอนด์พอยต์ 350
เอนด์พอยต์สำหรับการถาม-ตอบด้วย RAG 351
ค่าทางสถิติของระบบ และการติดตามตรวจสอบ 351
การเริ่มเซิร์ฟเวอร์และกำหนดค่า 352
การทดลองใช้งานและข้อมูลตัวอย่าง 353
การสร้างข้อมูลตัวอย่างที่สมจริง 353

ครอบคลุมตัวอย่างในหลายหัวข้อ 354
ไปป์ไลน์การประมวลผลข้อมูลตัวอย่าง 356
การสาธิตระบบ 356
การสาธิตความสามารถทีละขั้น 357
การสาธิต RAG ที่มีการตรวจสอบเงื่อนไข 358
ฟังชันการนำเข้าข้อมูลสำหรับการใช้งานจริง 359
จุดเริ่มทำงานของแอพพลิเคชัน 360
สรุป: ระบบความรู้ส่วนบุคคลที่สมบูรณ์ 360

บทที่ 9 Vector Query Language 363
แนวคิดหลัก 365
โมเดลข้อมูล 365
โครงสร้างพื้นฐานของไวยากรณ์ 366
การดำเนินการกับเวกเตอร์ 367
การค้นหาความคล้ายคลึง 367
ค้นหาแบบไฮบริด 368
ค้นหาแบบกำหนดระยะห่าง 369
การทำงานแบบแบตช์ 369
ฟังก์ชันและการรวมของเวกเตอร์ 370
ฟังก์ชันของเวกเตอร์ 370
การรวมเวกเตอร์ 371