FREE

vibecoding THAILAND

POCKET BOOK · 2026

Local LLM

ฉบับเข้าใจง่าย

รัน AI ไว้ในเครื่องตัวเอง ติดตั้ง อัปเดต จัดการ ลบ ครบวงจร ด้วย Ollama เรียบเรียงจากเอกสารทางการ

- ✔ ข้อมูลไม่ออกนอกเครื่อง ไม่มีค่ารายเดือน ไม่มีลิมิต
- ✔ สมการข้อเดียว รู้ว่าเครื่องไหนรันโมเดลไหนไหว
- ✔ สูตรตามเครื่อง 5 ระดับ ตั้งแต่มือถือถึง Mac

15 บท · 5 ภาค · 11 ภาพ

vibecodingthailand.com

สารบัญ

ภาค 1: ก่อนติดตั้ง AI ลงเครื่อง

ภาค 2: Ollama ครบวงจร

ภาค 3: สูตรตามเครื่อง

ภาค 4: ติดตั้งแล้วใช้ทำอะไร

ภาคผนวก

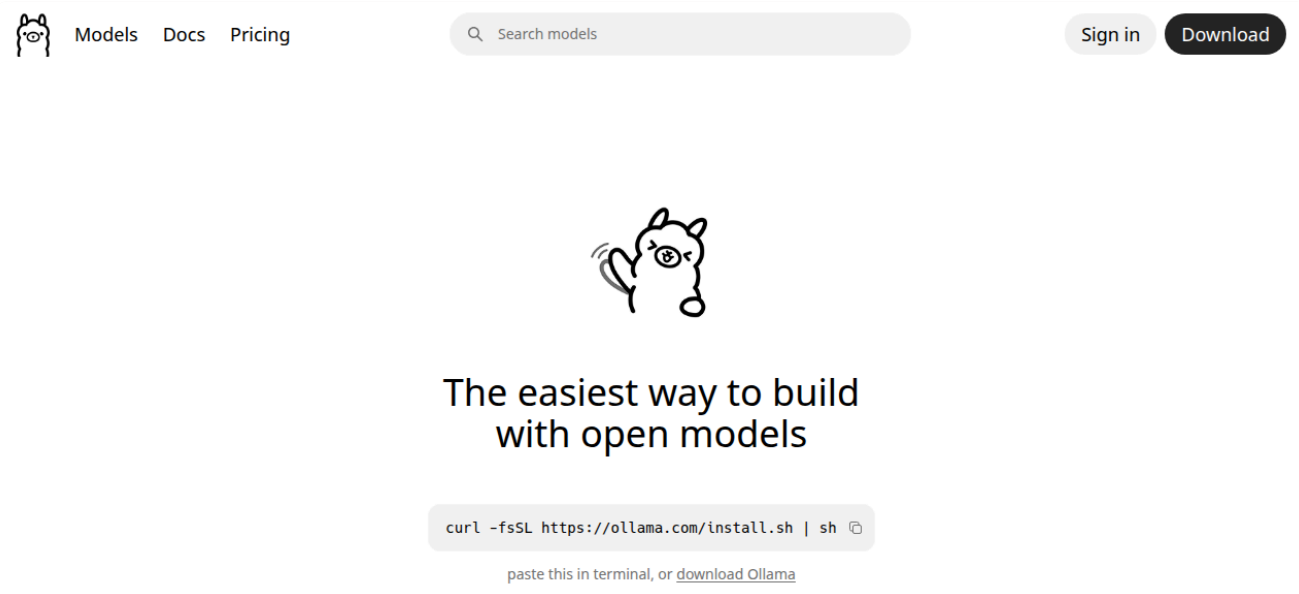
บทที่ 1

คำนำ

ทุกวันนี้การคุยกับ AI แปลว่าเปิดเว็บ พิมพ์ถาม แล้วรอคำตอบจากเซิร์ฟเวอร์ของใครสักคนที่อยู่อีกซีกโลก ทุกข้อความที่พิมพ์ต้องส่งออกไปนอกเครื่องเสมอ และวันไหนเน็ตล่ม โควตาเต็ม หรือค่าสมาชิกหมดอายุ ก็ใช้งาน AI ตัวนั้นไม่ได้

มีอีกทางหนึ่งที่คนพูดถึงกันมากขึ้นเรื่อยๆ คือ **local LLM** หรือการติดตั้งโมเดล AI ให้ทำงานบนเครื่องของเราเอง ไม่ต้องต่อเน็ต ไม่มีค่ารายเดือน และข้อมูลไม่ออกไปไหน

หนังสือเล่มนี้สอนเรื่องนั้นตั้งแต่ศูนย์ เริ่มจากเช็คว่าเครื่องที่มีอยู่รันโมเดลขนาดไหนไหว เลือกตัวที่พอดี ติดตั้ง ใช้งาน อัปเดต และจัดการพื้นที่ ไปจนถึงวิธีลบทุกอย่างที่ติดตั้งไว้เมื่อเลิกใช้ เครื่องมือหลักของทั้งเล่มคือ **Ollama** หนึ่งในโปรแกรมจัดการ local LLM ที่ได้รับความนิยมสูงสุด เนื้อหาเรียบเรียงจากเอกสารทางการของ Ollama และหน้าข้อมูลของโมเดลแต่ละตัวโดยตรง



หน้าแรกของเว็บ ollama.com มีมาสคอตลามะ สโลแกนว่าใช้งานโมเดล open-source ได้ง่ายที่สุด และกล่องคำสั่งติดตั้งเพียงบรรทัดเดียว

หน้าแรกของ ollama.com เครื่องมือหลักของหนังสือเล่มนี้ local LLM อาจฟังดูเป็นเรื่องทางเทคนิค แต่ตัวเครื่องมือจริงมีหน้าตาเป็นมิตรประมาณนี้ (ภาพหน้าเว็บ ollama.com)

หนังสือเล่มนี้เหมาะกับใคร

- คนทำงานที่มีเอกสารลับในมือ อยากให้ AI ช่วยโดยไม่ต้องส่งข้อมูลขึ้นเว็บ

- ครีเอเตอร์และฟรีแลนซ์ที่อยากใช้งาน AI ได้ไม่จำกัดและไม่มีบิลรายเดือน
- เจ้าของธุรกิจที่อยากรู้ว่าเครื่องที่มีอยู่ทำอะไรได้บ้าง ก่อนตัดสินใจลงทุนเพิ่ม
- มือใหม่ที่ใช้ ChatGPT เป็นแล้ว และอยากรู้ว่าโลกของ AI ในเครื่องตัวเองหน้าตาเป็นยังไง

> **ไม่เคยเปิด terminal มาก่อน ไม่เป็นไร**
> เครื่องมือหลักของเล่มนี้สั่งงานผ่าน terminal (หน้าต่างพิมพ์คำสั่ง) ฟังดูน่ากลัว แต่จริงๆ แล้วทั้งเล่มใช้คำสั่งแค่หยิบมือเดียว และทุกคำสั่งมีตัวอย่างให้พิมพ์ตามพร้อมบอกว่าควรเห็นอะไรบนจอ ถ้าพิมพ์ตามได้ ก็ใช้ local LLM เป็น

อ่านยังไง

เล่มนี้แบ่งเป็น 4 ภาค อ่านเรียงตั้งแต่ต้นได้เลย หรือถ้าใจร้อนก็มีทางลัด

อยาก	เปิดไป
รู้ก่อนว่าเครื่องตัวเองรันโมเดลขนาดไหนไหว	บท 2
ลงมือติดตั้งให้เร็วที่สุด	บท 4 แล้วต่อบท 5
หาสูตรให้ตรงกับเครื่องที่ตัวเองใช้	ภาค 3 (บท 8-12)
รู้ว่าติดตั้งแล้วเอาไปทำอะไรได้	ภาค 4 (บท 13-15)

ทุกบทจบด้วยกล่อง **ลองเลย** เป็นการบ้านชิ้นเล็กๆ ที่ทำได้ในไม่กี่นาที ถ้าทำตามครบ พออ่านจบเล่มก็จะมี AI ของตัวเองรันอยู่ในเครื่องจริงๆ และดูแลมันเป็น ไม่ใช่แค่รู้จัก

> **ข้อมูลในเล่มนี้**
> เรียบเรียงจากเอกสารทางการ ณ กรกฎาคม 2026 ข้อมูลอย่างขนาดไฟล์โมเดลและรายชื่อรุ่นเปลี่ยนได้เสมอเมื่อมีเวอร์ชันใหม่ ถ้าสิ่งที่เห็นบนหน้าจอไม่ตรงกับในเล่ม ให้ยึด docs.ollama.com และ ollama.com เป็นหลัก ส่วนวิธีคิดทั้งหมดในเล่มใช้ได้เหมือนเดิม

หนังสือเล่มนี้แจกฟรี จัดทำโดยเพจ vibe coding thailand

บทที่ 2

local LLM คืออะไร ทำไมยังน่ามีไว้ใช้ทั้งที่มี ChatGPT แล้ว

ลองนึกถึงงานที่อยากให้ AI ช่วยที่สุด แต่ไม่เคยกล้าส่งเข้าไปในช่องแชท ไม่ว่าจะเป็นสัญญาลูกค้าที่มีชื่อและตัวเลขจริง ไฟล์เงินเดือนพนักงาน หรือร่างข้อตกลงที่ยังเจรจาไม่จบ งานพวกนี้จึงค้างอยู่ในเครื่อง เพราะเรารู้ดีว่า AI บนเว็บต้องส่งข้อมูลของเราไปประมวลผลที่เซิร์ฟเวอร์ของผู้ให้บริการเสมอ

local LLM เกิดมาเพื่องานแบบนี้

AI ที่ย้ายเข้ามาอยู่ในเครื่องเรา

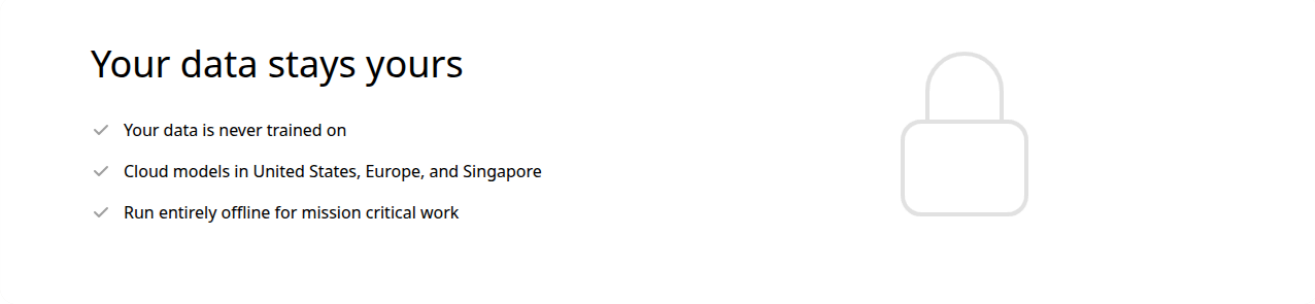
local LLM คือ AI ที่รันในเครื่องของเรา เราจะโหลดโมเดล (AI หนึ่งตัว ซึ่งก็คือสมองแบบเดียวกับที่อยู่เบื้องหลัง ChatGPT) มาเก็บไว้เป็นไฟล์ แล้วใช้ CPU หรือการ์ดจอของเครื่องประมวลผลคำตอบเอง โดยไม่ต้องพึ่งเซิร์ฟเวอร์ของผู้ให้บริการรายใด

เรื่องนี้เป็นไปได้เพราะในช่วงไม่กี่ปีที่ผ่านมา บริษัท AI รายใหญ่หันมาแจกโมเดลแบบ open weights (โมเดลที่เปิดให้ใครก็โหลดไฟล์ไปใช้ได้) กันอย่างจริงจัง Google แจกตระกูล Gemma · OpenAI แจก gpt-oss · Alibaba แจกตระกูล Qwen โมเดลหลายตัวมาพร้อมไลเซนส์แบบ Apache 2.0 จึงนำไปใช้ได้หลากหลายรูปแบบ รวมถึงในเชิงพาณิชย์

ในการใช้งาน local LLM หัวใจสำคัญคือการเลือกขนาดโมเดลให้สอดคล้องกับสเปกเครื่อง คอมพิวเตอร์ที่เรามีอยู่ เครื่องที่มีทรัพยากรจำกัดจำเป็นต้องรันโมเดลขนาดเล็ก ส่วนเครื่องที่มีทรัพยากรสูงสามารถรันโมเดลขนาดใหญ่ที่มีความฉลาดมากขึ้นได้ หน้าที่ของเราคือการติดตั้งโมเดลเข้ามาในเครื่อง จัดการดูแลให้ทำงานได้อย่างเต็มประสิทธิภาพ และรู้วิธีลบออกเมื่อไม่ได้ใช้งานแล้ว

สี่เหตุผลที่ควรติดตั้งโมเดล AI ไว้ในเครื่อง

ข้อมูลไม่ออกจากเครื่อง เมื่อรันแบบ local ทุกอย่างเกิดขึ้นและจบลงในเครื่องของเรา เอกสารทางการของ Ollama ระบุชัดว่าเมื่อรันแบบ local บริษัทจะไม่เห็นพรอมต์ (ข้อความสั่งงาน) หรือข้อมูลของเราเลย งานลับที่ไม่เคยกล้าให้ AI ช่วยจึงกลายเป็นงานที่เราใช้ AI ช่วยได้อย่างเต็มที่



ภาพจากเว็บ ollama.com ในหัวข้อ Your data stays yours มีรูปแม่กุญแจและข้อความว่าจะไม่นำข้อมูลไปเทรนโมเดล พร้อมระบุว่ารันได้โดยไม่ต้องต่ออินเทอร์เน็ตเลย

หน้าแรกของ ollama.com ประกาศว่า "ข้อมูลของคุณเป็นของคุณ" ส่วนข้อสุดท้ายคือหัวใจของเล่มนี้: งานสำคัญรันได้โดยไม่ต้องต่ออินเทอร์เน็ตเลย (ภาพหน้าเว็บ ollama.com)

ทำงานได้โดยไม่ต้องต่อเน็ต โหลดโมเดลเพียงครั้งแรก หลังจากนั้นตัวโมเดลจะถูกบันทึกอยู่ในเครื่องคอมพิวเตอร์อย่างถาวร ต่อให้เน็ตหลุด อยู่บนเครื่องบิน หรือทำงานในพื้นที่อับสัญญาณ ก็ยังสามารถเรียกใช้งานได้ตามปกติ

ไม่มีค่ารายเดือน โมเดลเปิดแจกฟรี โปรแกรมที่ใช้รันก็ฟรี ต้นทุนเดียวคือเครื่องที่มีอยู่แล้วกับค่าไฟ

ไม่มีลิมิตการใช้งาน ไม่มีโควตาข้อความต่อวัน ไม่มีชั่วโมงเร่งด่วนที่ต้องต่อคิว เพราะทุกอย่างประมวลผลในเครื่องของเราเอง จะใช้หนักแค่ไหนก็เรื่องของเรา

ความจริงสามข้อที่ต้องรู้ก่อน

เล่มนี้ไม่ขายฝันว่า local LLM แทน ChatGPT ได้ทุกงาน ความจริงมีสามข้อ

ข้อแรก โมเดลเปิดที่เก่งที่สุดในโลกมีขนาดใหญ่เกินกว่าสเปกคอมพิวเตอร์ทั่วไปจะรันได้ไหว อย่าง Qwen3.5 รุ่นใหญ่สุดที่เปิดให้โหลดก็มีไฟล์ขนาด 81GB ซึ่งเกินขนาดหน่วยความจำ (Memory/RAM) ของคอมพิวเตอร์ทั่วไปเกือบทั้งหมด ส่วนโมเดลที่ใหญ่กว่านั้นบางรุ่นไม่มีให้ดาวน์โหลด เพราะมีให้ใช้งานผ่านระบบคลาวด์ (Cloud) เท่านั้น

ข้อสอง โมเดลที่รันบนคอมพิวเตอร์ทั่วไปได้คือโมเดลขนาดเล็กถึงกลาง โมเดลเหล่านี้มีความสามารถจำกัดกว่าตัวท็อปที่อยู่เบื้องหลัง ChatGPT หรือ Gemini พอสมควร งานที่ต้องการความถูกต้องระดับสูงสุดหรืองานที่ผิดพลาดไม่ได้ยังต้องใช้ AI บนคลาวด์

ข้อสาม เครื่องเป็นตัวกำหนดขีดจำกัด ความสามารถของโมเดลที่รันได้ขึ้นอยู่กับหน่วยความจำ (Memory/RAM) ของเครื่องโดยตรง หากต้องการรันโมเดลที่ฉลาดขึ้นและมีขนาดใหญ่ขึ้น ก็จำเป็นต้องอัปเกรดหน่วยความจำของเครื่องเพิ่มขึ้น ไม่มีทางลัด

เส้นแบ่งที่ใช้ตลอดทั้งเล่มจึงเป็นแบบนี้: **งานที่ต้องการความเป็นส่วนตัว งานที่ทำแบบออฟไลน์ หรือการทำงานปริมาณมากโดยไม่มีข้อจำกัดเรื่องโควตา ควรเลือกใช้ local LLM ส่วนงานซับซ้อนที่ต้องการความถูกต้องสูงสุด ควรยกให้ AI บนระบบคลาวด์** ทั้งสองระบบจึงไม่ได้แข่งกัน แต่เป็นการทำงานร่วมกันสำหรับงานคนละประเภท

คำว่า local ต้องดูให้ขาด

Ollama เองก็มีโมเดลแบบ cloud ให้เรียกใช้ (ชื่อรุ่นลงท้ายว่า cloud) ซึ่งประมวลผลบนเซิร์ฟเวอร์ ไม่ใช่ในเครื่องของเรา ถ้าคุณมาอ่านเล่มนี้เพราะต้องการความเป็นส่วนตัว ต้องแยกโมเดลสองแบบนี้ให้ออกก่อนใช้งาน บท 5 จะบอกวิธีแยกให้ชัดๆ

ลองเลย

ยังไม่ต้องติดตั้งอะไรทั้งนั้น แค่ลองนึกถึงงานของคุณเอง 1 งานที่เข้าเงื่อนไขว่า "อยากให้ AI ช่วย แต่ข้อมูลห้ามออกนอกเครื่อง" แล้วจดไว้สั้นๆ คุณจะได้นำงานชิ้นนี้มาใช้จริงในภาค 4

บทถัดไปจะตอบคำถามที่ทุกคนถามก่อนเริ่ม: เครื่องที่มีอยู่ตอนนี้สามารถรันโมเดลขนาดใหญ่ที่สุดได้เท่าใด

บทที่ 3

เช็คสเปกเครื่องก่อนเลือกโมเดล

คำถามแรกของทุกคนที่อยากลอง local LLM คือ "เครื่องที่มีอยู่ไหวไหม" ข่าวดีคือใช้เวลาเพียง 5 นาทีก็รู้คำตอบ และไม่ต้องเป็นสายฮาร์ดแวร์ก็เข้าใจได้ เพราะทั้งหมดขึ้นอยู่กับตัวเลขเพียงตัวเดียว

memory คือทุกอย่าง

ตอนทำงาน โมเดลทั้งตัวต้องโหลดเข้าไปอยู่ในหน่วยความจำ (Memory) ของเครื่อง โมเดลขนาดใหญ่จะมีขนาดไฟล์ใหญ่ตามไปด้วย ส่วนความจุที่เครื่องจะรองรับได้นั้น ขึ้นอยู่กับหน่วยความจำที่มีอยู่ในเครื่อง ซึ่งแบ่งเป็นสามแบบหลัก

- **RAM** หน่วยความจำหลักของเครื่อง ใช้เมื่อรันด้วย CPU ล้วน
- **VRAM** หน่วยความจำบนการ์ดจอ เร็วกว่ามาก โมเดลที่เข้าไปนั่งใน VRAM ได้ทั้งตัวจะตอบไวสุด
- **unified memory** ในเครื่อง Mac เป็น memory ก้อนเดียวที่ CPU กับ GPU ใช้ร่วมกัน จุดเด่นคือมีให้เลือกความจุสูงมาก (บท 11 ว่ากันเต็มๆ)

สำหรับการ์ดจอ Ollama รองรับ NVIDIA ตั้งแต่ยุค GTX 900 เรื่อยมาจนถึง RTX 50 ส่วน GPU ของ Apple ใช้ผ่าน Metal และ AMD Radeon ใช้ผ่าน ROCm กับ Vulkan การ์ดที่คนทั่วไปใช้เล่นเกมกันในช่วงไม่กี่ปีนี้จึงใช้ได้แทบทั้งหมด

สมการคำนวณหน่วยความจำ

กติกาที่ใช้ตัดสินทุกอย่างในเล่มนี้มีบรรทัดเดียว

> **ขนาดไฟล์โมเดล + memory ที่ต้องเผื่อ ต้องไม่เกิน memory ของเครื่อง**

memory ที่ต้องเผื่อคือส่วนที่โมเดลใช้จำบทสนทนา เรียกว่า context (ความจำระยะสั้นของบทสนทนา) ยิ่งตั้งให้จำบทสนทนาได้นานเท่าไร ก็ยิ่งใช้ memory มากขึ้น เอกสารของ Ollama ระบุชัดว่าการเพิ่ม context length ทำให้โมเดลใช้ memory มากขึ้น และ Ollama ก็กำหนดค่าเริ่มต้นตามขนาด VRAM ของเครื่อง โดยการ์ดที่มี VRAM ต่ำกว่า 24GB จะตั้ง context ไว้ที่ 4K ช่วง 24-48GB ตั้งไว้ที่ 32K และตั้งแต่ 48GB ขึ้นไปตั้งไว้ที่ 256K

ถ้าโมเดลใหญ่เกินหน่วยความจำที่ว่างอยู่ หน่วยความจำจะเต็ม โมเดลจะไม่หยุดทำงานทันที แต่ระบบจะสลับการประมวลผลไปมาระหว่างการ์ดจอกับ RAM ซึ่งทำให้ความเร็วในการตอบช้าลงอย่างเห็นได้ชัด สามารถเช็คสถานะการประมวลผลนี้ได้ผ่านคำสั่ง `ollama ps` (ดูรายละเอียดในบท 5)

quantization · เคล็ดลับที่ช่วยให้รันโมเดลใหญ่บนเครื่องเล็กได้

ตัวเลขขนาดโมเดลที่เห็นตามข่าวคือจำนวนพารามิเตอร์ หน่วยเป็น B (พันล้านพารามิเตอร์ ยิ่งมากยิ่งฉลาดและยิ่งกินที่) แต่ไฟล์ที่เราโหลดจริงมักเล็กกว่าที่คิด เพราะผ่าน quantization (การย่อโมเดลโดยเก็บค่าตัวเลขให้ละเอียดน้อยลง) มาแล้ว

ตัวอย่างที่เห็นภาพชัดที่สุดคือ gpt-oss ของ OpenAI รุ่น 20B ซึ่งเก็บน้ำหนักส่วนใหญ่ด้วยความละเอียดราว 4.25 bits ต่อพารามิเตอร์หลัง quantization ทำให้ไฟล์เหลือ 14GB และ OpenAI ระบุว่าออกแบบมาให้รันบนเครื่องที่มี memory เพียง 16GB ได้ ส่วนรุ่น 120B มีขนาดไฟล์ 65GB

ข้อควรรู้อีกอย่างคือ quantization ต้องแลกกับความแม่นยำ ยิ่งบีบมาก โมเดลยิ่งแม่นยำน้อยลง Ollama เตรียมโมเดลใน library ด้วยระดับ quantization ที่สมดุลมาให้แล้ว มือใหม่ใช้รุ่นที่มีให้ได้เลยโดยไม่ต้องตั้งค่าเพิ่ม (ส่วนสายลึกที่อยากเลือกระดับเอง ดูตัวเลือกในแท็กของแต่ละโมเดลได้ในบท 5)

ตารางเปรียบเทียบสเปกเครื่อง 5 ระดับ

ก่อนดูตาราง ลองดูหน้าเว็บจริงกันก่อน นี่คือตารางรุ่นย่อยของโมเดลตระกูลเดียว (gemma4) บน ollama.com

Models View all →

Name	Size / Usage	Context	Input
gemma4:latest	9.6GB	128K	Text, Image
gemma4:e2b	7.2GB	128K	Text, Image
gemma4:e4b latest	9.6GB	128K	Text, Image
gemma4:12b	7.6GB	256K	Text, Image
gemma4:26b	18GB	256K	Text, Image
gemma4:31b	20GB	256K	Text, Image
gemma4:e2b-mlx MLX	6.5GB	128K	Text, Image
gemma4:e4b-mlx MLX	8.8GB	128K	Text, Image
gemma4:12b-mlx MLX	7.7GB	256K	Text, Image
gemma4:26b-mlx MLX	18GB	256K	Text, Image
gemma4:31b-mlx MLX	19GB	256K	Text, Image
gemma4:cloud	—	256K	Text, Image
gemma4:31b-cloud	—	256K	Text, Image

ตารางรุ่นย่อยของ gemma4 บน ollama.com แสดงขนาดไฟล์ตั้งแต่ 7.2GB ถึง 20GB พร้อมความยาว context และชนิดข้อมูลของแต่ละรุ่น

โมเดลตระกูลเดียวมีขนาดให้เลือกตั้งแต่ 7.2GB ถึง 20GB ตามความจุของหน่วยความจำ คอลัมน์ Size คือขนาดไฟล์โมเดลที่ต้องนำไปคำนวณตามสมการข้างต้น ส่วนวิธีอ่านหน้านี้แบบละเอียดอยู่ในบท 5 (ภาพหน้าเว็บ ollama.com)

ตัวเลขในตารางเป็นขนาดไฟล์จริงจากหน้าโมเดลบน ollama.com ตอนที่เขียนบทนี้

ระดับสเปกเครื่อง	memory	ตัวอย่างโมเดลที่เหมาะสม	ขนาดไฟล์
มือถือ	RAM 4-8GB	qwen3.5:0.8b · qwen3.5:2b	1.0GB · 2.7GB
โน้ตบุ๊กทำงานทั่วไป	RAM 8-16GB	qwen3.5:4b · gemma4:12b · gemma4:e4b	3.4GB · 7.6GB · 9.6GB
เครื่องเกม / การ์ดจอแยก	VRAM 8-24GB	gpt-oss:20b · qwen3.6:27b · gemma4:26b · gemma4:31b	14GB · 17GB · 18GB · 20GB
Mac ตัวแรง	unified memory 32GB ขึ้นไป	ตัวเดียวกับเครื่องเกม + qwen3.6:35b	ถึง 24GB
ระดับสูง/เซิร์ฟเวอร์	64GB ขึ้นไปจนถึงระดับเซิร์ฟเวอร์	gpt-oss:120b · qwen3.5:122b	65GB · 81GB

ตารางนี้เป็นภาพรวมของภาค 3 โดยบท 8 ถึง 12 จะอธิบายเครื่องแต่ละระดับอย่างละเอียด ตอนนี้แค่ดูให้รู้ว่าตัวเองอยู่แถวไหนก็พอ

ถ้าจะดูตัวเลขของเครื่องตัวเอง ให้เช็คสเปค RAM จากหน้าข้อมูลเครื่องของระบบปฏิบัติการ ถ้ามีการ์ดจอแยก ให้เช็ครุ่นการ์ดแล้วดูว่า VRAM ของรุ่นนั้นมีเท่าไร จุดที่พลาดกันบ่อยคือเอา RAM กับ VRAM มาปนกัน สำหรับเครื่องที่มีการ์ดจอแยก ตัวเลขที่ชี้ขาดความเร็วคือ VRAM

> **เผื่อที่ไว้เสมอ**
> อย่าเลือกโมเดลที่มีขนาดไฟล์เท่ากับ memory แบบพอดีเป๊ะ เพราะระบบปฏิบัติการกับโปรแกรมอื่นก็ใช้ memory เช่นกัน และยังต้องเผื่อ memory ให้ context อีก เลือกโมเดลที่มีขนาดเล็กกว่าความจุของเครื่องพอสมควร แล้วชีวิตจะง่ายขึ้น

> **ลองเลย**
> เช็คเครื่องของคุณว่า RAM เท่าไหร่ มีการ์ดจอแยกไหม และถ้ามี VRAM เท่าไหร่ แล้วเอาตัวเลขไปเทียบกับตารางข้างบนเพื่อดูว่าเครื่องของคุณอยู่ระดับไหน จดชื่อโมเดลในแถวนั้นไว้ เพราะได้ใช้แน่ในบท 4

เมื่อทราบขีดจำกัดหน่วยความจำของเครื่องแล้ว บทถัดไปเราจะมาดูประเภทของโมเดลต่าง ๆ ว่ามีโมเดลใดให้เลือกบ้าง และแต่ละค่ายมีความโดดเด่นในด้านใด

บทที่ 4

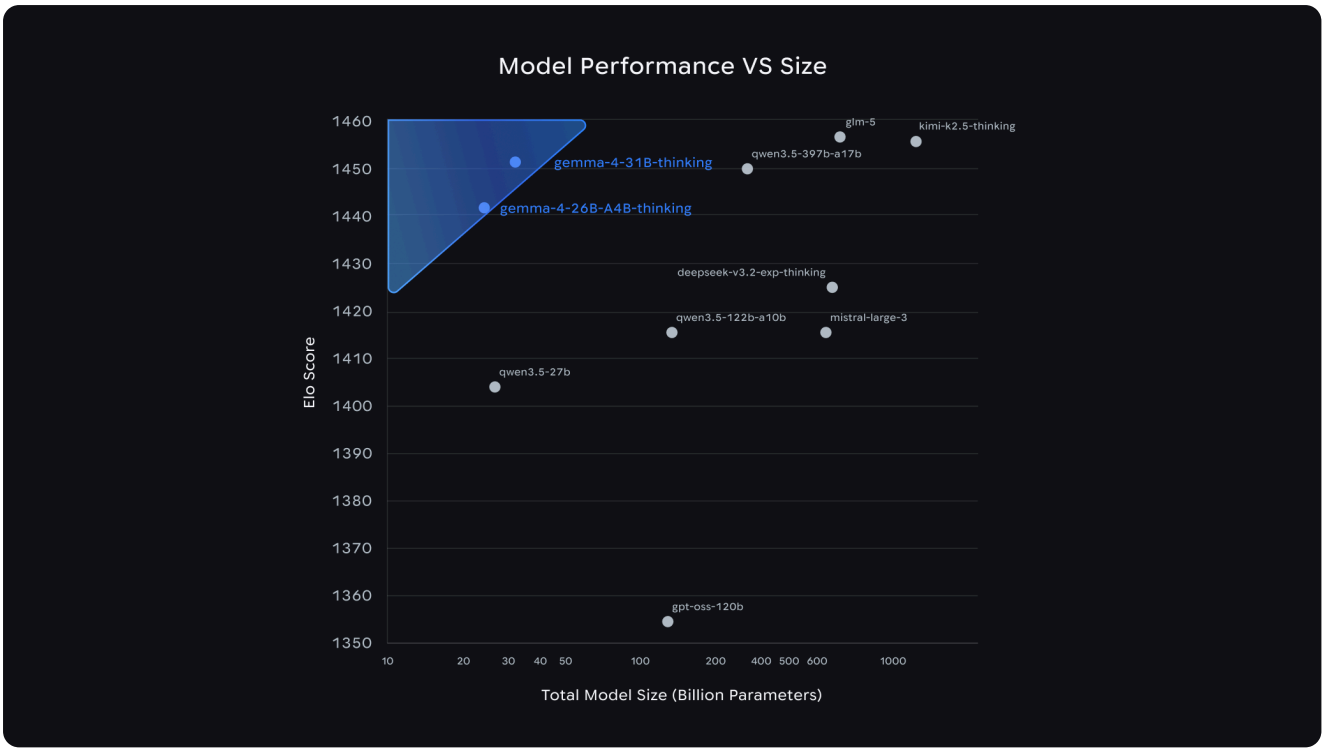
แผนที่วงการ: ใครเป็นใครในโลกโมเดลเปิด

พอเปิดหน้ารวมโมเดลของ ollama.com ครั้งแรก จะเจอรายชื่อเป็นร้อย ชื่ออย่าง qwen3.5 · gemma4 · gpt-oss และ deepseek-r1 ล้วนอ่านยากพอกัน มองแล้วเหมือนเดินเข้าตลาดที่ไม่รู้จักสักร้าน บทนี้คือแผนที่ฉบับย่อ อ่านจบแล้วจะคุ้นกับชื่อเหล่านี้ และรู้ว่าแต่ละตัวมาจากค่ายไหน ถนัดอะไร

หกค่ายที่ต้องรู้จัก

Qwen · ค่าย Alibaba เป็นหนึ่งในตระกูลที่มีขนาดให้เลือกหลากหลายที่สุด Qwen3.5 มีตั้งแต่ 0.8B ไปจนถึง 122B รองรับทั้งข้อความและรูปภาพ รวมถึงภาษามากถึง 201 ภาษา ส่วน Qwen3.6 คือรุ่นถัดมาที่เน้นงานเขียนโค้ดแบบ agent (ให้ AI ลงมือทำงานหลายขั้นตอนเอง) มีขนาด 27B กับ 35B และรองรับ context ได้ยาว 256K

Gemma · ค่าย Google DeepMind รุ่นล่าสุดคือ Gemma 4 ซึ่ง Google ชูให้เป็นโมเดลเปิดเรือธง รุ่นใหญ่สุด 31B ขึ้นเป็นอันดับ 3 ในบรรดาโมเดลเปิดทั่วโลกบนกระดานจัดอันดับ Arena ณ วันเปิดตัว จุดเด่นคือมีหลายรุ่นให้เลือกตามกำลังของอุปกรณ์ ตั้งแต่ E2B กับ E4B ที่ออกแบบมาเพื่อมือถือและอุปกรณ์เล็กโดยเฉพาะ (รับเสียงพูดได้ด้วย) ไปจนถึง 12B สำหรับโน้ตบุ๊ก และ 26B กับ 31B สำหรับเครื่องแรง ทุกตัวมองรูปภาพได้ รองรับกว่า 140 ภาษา และเผยแพร่ภายใต้สัญญาอนุญาต Apache 2.0



กราฟกระจายเทียบคะแนน Elo กับขนาดของโมเดลเปิดหลายตัว จุดของ Gemma 4 31B และ 26B อยู่โซนซ้ายบน คือเล็กกว่าแต่คะแนนสูงกว่าหลายตัวที่ใหญ่กว่ามาก

คะแนนจาก Arena เทียบกับขนาดโมเดลเปิด หลายชื่อในภาพคือโมเดลที่จะเจอตลอดทั้งเล่ม ตั้งแต่ qwen3.5-27b ไปจนถึง gpt-oss-120b แกนนอนทำให้เห็นเรื่องสำคัญของยุคนี้: โมเดลเล็กทำคะแนนไล่ทันโมเดลใหญ่แล้ว (ภาพจาก blog ทางการของ Google)

gpt-oss · ค่าย OpenAI โมเดลเปิดจากเจ้าของ ChatGPT มีสองขนาดคือ 20B กับ 120B จุดขายอยู่ที่ reasoning (การคิดเป็นขั้นเป็นตอนก่อนตอบ) โดยเปิดให้เห็นกระบวนการคิดเต็มๆ และปรับได้ว่าจะให้คิดละเอียดแค่ไหน รุ่น 20B ออกแบบมาให้รันบนเครื่องที่มี memory 16GB ได้ และเผยแพร่ภายใต้สัญญาอนุญาต Apache 2.0 เช่นกัน

DeepSeek · ค่ายจีนที่ดังข้ามคืน เจ้าของ DeepSeek-R1 โมเดล reasoning ที่ทำผลงานเทียบชั้นโมเดลปิดราคาแพงจนเป็นข่าวใหญ่ เผยแพร่ภายใต้สัญญาอนุญาต MIT และมีเวอร์ชัน distill (ถ่ายทอดความรู้จากโมเดลใหญ่สู่โมเดลเล็ก) หลายขนาด จึงใช้โมเดลที่มีความสามารถแบบเดียวกันบนเครื่องเล็กได้

Llama · ค่าย Meta ผู้จุดกระแสโมเดลเปิดยุคแรกและยังเป็นตระกูลที่คนโหลดมากที่สุดบน Ollama โดย llama3.1 มียอดโหลดกว่าร้อยล้านครั้ง มีตั้งแต่รุ่นจิ๋ว 1B ไปจนถึงรุ่นยักษ์ 405B

Mistral · ค่ายฝรั่งเศส ม้ามืดจากยุโรป mistral 7B เป็นหนึ่งในโมเดลเล็กที่ได้รับความนิยมมายาวนาน รุ่นหลังๆ อย่าง mistral-small ก็ยังเป็นโมเดลที่คนนิยมเลือกใช้ในชีวิตประจำวัน

> **แล้วภาษาไทยล่ะ**
> ค่ายที่ประกาศจำนวนภาษาที่รองรับไว้อย่างชัดเจนคือ Qwen (201 ภาษา) กับ Gemma (140+ ภาษา) ทั้งสองค่ายรองรับภาษาไทย แต่ตัวเลขนี้บอกแค่ว่า "พูดได้" ไม่ได้บอกว่า "พูดเก่ง" ยิ่งโมเดลมีขนาดเล็ก ความสามารถด้านภาษารองก็ยิ่งลดลงเร็ว ทางที่ดีควรทดสอบกับงานไทยของตัวเองก่อนใช้จริง ในบท 4 เราจะได้ลองทันทีที่ติดตั้งเสร็จ

ศัพท์สองคำที่ช่วยอ่านสเปคโมเดลออก

MoE (Mixture of Experts) คือสถาปัตยกรรมของโมเดลที่แบ่งเครือข่ายประมวลผลย่อยออกเป็นหลายส่วน (Experts) แต่จะเรียกใช้งานจริงเพียงไม่กี่ส่วนในระหว่างการประมวลผลคำตอบ อย่าง Gemma 4 ขนาด 26B ตอนทำงานจริงจะคำนวณโดยใช้พารามิเตอร์เพียงประมาณ 4B เท่านั้น ผลลัพธ์คือมีประสิทธิภาพความฉลาดเทียบเท่าโมเดลขนาดใหญ่ แต่ประมวลผลได้รวดเร็วใกล้เคียงกับโมเดลขนาดเล็ก สถาปัตยกรรมนี้กำลังเป็นที่นิยมอย่างมากในโมเดลยุคปัจจุบัน

thinking / reasoning คือโมเดลที่คิดในใจก่อนตอบ ต้องรอนานขึ้น แต่ได้คำตอบที่ผ่านการไตร่ตรอง เหมาะกับโจทย์คณิต โค้ด และงานวางแผน ตัวแทนสายนี้คือ deepseek-r1 และ gpt-oss ส่วน Gemma 4 กับ Qwen รุ่นใหม่เลือกเปิดหรือปิดโหมดคิดได้

Ollama อยู่ตรงไหนของภาพนี้

บริษัทผู้พัฒนาทั้งหกคือผู้สร้างโมเดล ส่วน Ollama คือโปรแกรมที่ทำหน้าที่จัดการระบบในเครื่องคอมพิวเตอร์ของเรา โดยทำหน้าที่ดาวน์โหลดโมเดล ประมวลผลโมเดลเพื่อให้เราใช้งานผ่านหน้าแชท และควบคุมจัดการทุกอย่างด้วยคำสั่งเพียงไม่กี่คำ เบื้องหลัง Ollama ใช้ llama.cpp ซึ่งเป็นโปรเจกต์โอเพนซอร์สที่เปิดให้เราสามารถรันโมเดลขนาดใหญ่บนเครื่องคอมพิวเตอร์ทั่วไปได้อย่างมีประสิทธิภาพ

โมเดลที่พร้อมโหลดทั้งหมดอยู่ที่ **ollama.com/library** ที่นี่เปรียบเหมือนคลังเก็บโมเดลที่มีการคัดกรองไว้ให้เลือกใช้งาน แต่ละโมเดลจะมีหน้าเว็บของตัวเอง ซึ่งระบุขนาดไฟล์ความจุของทุกรุ่น ความยาว context และคำสั่งในการดาวน์โหลดที่สามารถคัดลอกไปใช้ได้ทันที การอ่านและเลือกใช้โมเดลในหน้านี้เป็นหัวใจหลักของบท 5

โลกของโมเดลเปิดยังมีคลังเก็บไฟล์ขนาดใหญ่ระดับโลกที่ชื่อ **Hugging Face** ซึ่งผู้พัฒนาทุกค่ายจะนำไฟล์โมเดลฉบับเต็มไปอัปโหลดไว้ที่นั่น ทั้ง Gemma 4 · Qwen · gpt-oss ต่างมีไฟล์ต้นฉบับอยู่ที่นั่น สำหรับผู้ที่ต้องการศึกษาเชิงลึกและมองหาโมเดลประเภทอื่น ๆ ที่ไม่มีใน Ollama ก็สามารถเข้าไปค้นหาเพิ่มเติมได้ แต่หากศึกษาตามหนังสือเล่มนี้ การใช้งานผ่านเว็บ ollama.com เว็บเดียวก็เพียงพอแล้ว

> **ลองเลย**
>
> เปิด ollama.com/library บนมือถือหรือคอมก็ได้ เลื่อนดูสัก 1 นาที ลองหาชื่อทั้งหกค่ายจากบทนี้ให้เจอ เลือกโมเดลที่คุณจดไว้จากบท 2 มาหนึ่งตัว แล้วกดเข้าไปดูหน้าโมเดลนั้น ดูว่ามีรุ่นขนาดไหนให้เลือกบ้าง

เมื่อรู้จักประเภทของโมเดลและขีดจำกัดหน่วยความจำของเครื่องแล้ว ภาคถัดไปจะเป็นขั้นตอนปฏิบัติจริง: เริ่มจากการติดตั้ง Ollama และทดลองรันโมเดลตัวแรกบนเครื่องคอมพิวเตอร์ของเรา

บทที่ 5

ติดตั้ง Ollama และเริ่มใช้งานโมเดลแรก

ถึงเวลาทดลองรันโมเดลตัวแรกบนเครื่อง บทนี้ใช้เวลาราว 10 นาที ไม่นับเวลาดาวน์โหลดไฟล์ขนาดไม่กี่ GB จบบทแล้วเครื่องคอมพิวเตอร์ของคุณจะสามารถประมวลผล AI ที่คุยภาษาไทยได้โดยไม่ต้องเชื่อมต่ออินเทอร์เน็ตอีกเลย

เช็คก่อนติดตั้ง

Ollama รองรับทั้ง Windows · macOS · Linux โดยเอกสารทางการระบุเงื่อนไขขั้นต่ำไว้ดังนี้

- **Windows**: Windows 10 22H2 ขึ้นไป (Home หรือ Pro ก็ได้) ถ้ามีการ์ด NVIDIA ต้องใช้ driver เวอร์ชัน 551.61 ขึ้นไป
- **macOS**: macOS Sonoma (เวอร์ชัน 14) ขึ้นไป เครื่องที่ใช้ชิป Apple M series ใช้ได้ทั้ง CPU และ GPU อย่างเต็มประสิทธิภาพ ส่วนเครื่อง Intel รุ่นเก่าใช้ CPU ได้อย่างเดียว
- **พื้นที่ดิสก์**: ตัวโปรแกรมต้องการราว 4GB และต้องเผื่อพื้นที่สำหรับไฟล์โมเดลอีกต่างหาก (ตัวเล็กไม่กี่ GB ตัวใหญ่หลายสิบ GB ตามตารางบท 2)

ติดตั้ง

ทุกระบบเริ่มที่เว็บเดียวกัน: **ollama.com/download**

Windows โหลดไฟล์ `OllamaSetup.exe` แล้วดับเบิลคลิกติดตั้ง ตัวติดตั้งไม่ต้องใช้สิทธิ์ Administrator เพราะลงในโฟลเดอร์ผู้ใช้ของเราเอง เสร็จแล้ว Ollama จะทำงานเบื้องหลัง และคำสั่ง `ollama` พร้อมใช้ทั้งใน PowerShell และ cmd

macOS โหลดไฟล์ `Ollama.dmg` เปิดขึ้นมาแล้วลากไอคอน Ollama ไปวางในโฟลเดอร์ Applications เปิดแอปครั้งแรก ถ้าเครื่องยังไม่มีคำสั่ง `ollama` ใน terminal แอปจะขออนุญาตสร้างให้เอง

> **สาย Linux**
> ติดตั้งด้วยคำสั่งเดียว: `curl -fsSL https://ollama.com/install.sh | sh` พร้อมตั้งค่าให้ Ollama ทำงานเป็น service อยู่เบื้องหลัง รายละเอียดแบบ manual อยู่ที่ docs.ollama.com/linux

เปิด terminal แล้วทักทาย

เปิดหน้าต่างสำหรับพิมพ์คำสั่ง: บน Windows กดปุ่ม Start แล้วพิมพ์หา Terminal หรือ PowerShell ส่วนบน macOS เปิดแอป Terminal (ค้นจาก Spotlight ได้เลย)

โมเดลตัวแรกที่เล่มนี้ใช้สาธิตคือ **qwen3.5:2b** เหตุผลคือไฟล์มีขนาดเพียง 2.7GB ซึ่งเครื่องคอมพิวเตอร์ทั่วไปสามารถรันได้ง่าย และมาจากตระกูลที่รองรับ 201 ภาษา จึงสามารถคุยไทยได้ (หากเครื่องคอมพิวเตอร์ของคุณมีสเปกสูงกว่านั้น สามารถใช้ชื่อโมเดลอื่นตามที่จดไว้จากบท 2 แทนได้ทันที โดยใช้วิธีรันแบบเดียวกัน)

```
ollama run qwen3.5:2b
```

ครั้งแรกที่รัน Ollama จะโหลดโมเดลจาก library ก่อน จะเห็นแถบความคืบหน้าวิ่งจนครบ เสร็จแล้วหน้าจอจะเปลี่ยนเป็นช่องแชทที่ขึ้นต้นด้วยเครื่องหมาย

```
>>>
```

นั่นคือแถบรับคำสั่งที่พร้อมรับข้อความ ลองพิมพ์เป็นภาษาไทยได้เลย

```
>>> ช่วยอธิบายหน่อยว่า local LLM คืออะไร ตอบสั้นๆ
```

คำตอบจากโมเดลจะปรากฏทีละคำบนหน้าจอ ทั้งหมดนี้เกิดขึ้นในเครื่องล้วนๆ ลองปิด Wi-Fi แล้วถามต่อก็ยังตอบได้เหมือนเดิม

พอคุยเสร็จ ออกจากห้องแชทด้วย

```
/bye
```

เกิดอะไรขึ้นหลังกด Enter

สามเรื่องที่ควรเข้าใจจากคำสั่งแรกนี้

โหลดครั้งเดียว ใช้ได้ตลอด ไฟล์โมเดลเก็บอยู่ในเครื่องถาวร สั่ง `ollama run qwen3.5:2b` ครั้งต่อไปจะเข้าห้องแชททันทีโดยไม่โหลดซ้ำ

คำสั่งเดียวทำสองหน้าที่ `ollama run` แปลว่า "ถ้ายังไม่มีก็โหลดมาก่อน แล้วเปิดแชท" ส่วนการโหลดอย่างเดียวโดยไม่เปิดแชทมีคำสั่งแยก ซึ่งจะเจอในบทถัดไป

Ollama พร้อมทำงานอยู่เบื้องหลังเสมอ แม้ปิด terminal แล้ว Ollama ก็ยังทำงานอยู่ (สังเกตได้จากไอคอนบน taskbar หรือ menubar) เพื่อรอรับงานครั้งถัดไป รวมถึงให้โปรแกรมอื่นในเครื่องเรียกใช้ ซึ่งเป็นเรื่องของบท 7

> **พิมพ์ ollama เฉยๆ ก็ได้**
> รันคำสั่ง `ollama` โดยไม่ต่อท้ายอะไร จะเปิดเมนูให้เลือกว่าจะเริ่มแชทกับโมเดลที่มี หรือเชื่อม Ollama เข้ากับเครื่องมืออื่น เหมาะกับวันที่ไม่อยากจำคำสั่ง

> **ลองเลย**
> ทดลองคุยกับโมเดล qwen3.5:2b เพื่อทดสอบการทำงาน 3 รูปแบบ: สรุปเนื้อหาข่าวสารสั้น ๆ 1 เรื่อง · แปลอีเมลภาษาอังกฤษเป็นภาษาไทย 1 ฉบับ · และร่างข้อความอวยพรวันเกิดเพื่อนร่วมงาน 1 ข้อความ เพื่อสังเกตคุณภาพการใช้ภาษาไทยของโมเดลขนาดเล็กนี้ และนำไปเปรียบเทียบกับผลลัพธ์ของโมเดลขนาดใหญ่ในภาค 3

เมื่อสามารถติดตั้งและรันโมเดลแรกได้แล้ว บทถัดไปจะเป็นคู่มือการจัดการโมเดลแบบครบวงจร ตั้งแต่การค้นหาโมเดลใหม่ การอัปเดต ตลอดจนถึงขั้นตอนการลบไฟล์โมเดลออกจากเครื่องคอมพิวเตอร์อย่างสะอาด

บทที่ 6

คำสั่งดูแลโมเดลครบวงจร: หา โหลด ดู อัปเดต ลบ

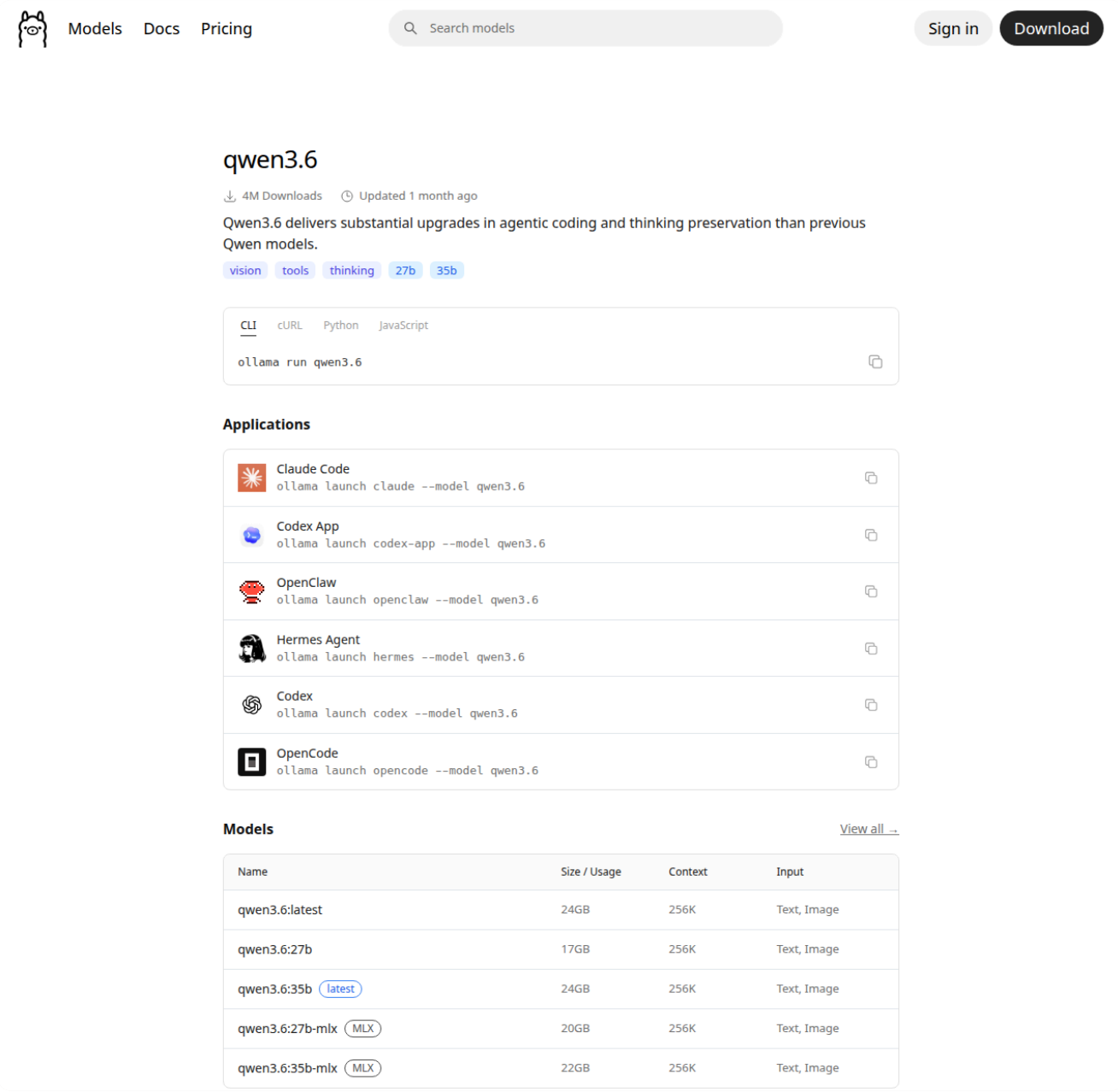
ติดตั้งโมเดลครั้งเดียวแล้วปล่อยทิ้งไว้ไม่ได้ เพราะมีโมเดลรุ่นใหม่ที่เก่งกว่าออกมาเรื่อย ๆ ตัวที่ใช้งานอยู่ก็อาจมีอัปเดตออกมาเพิ่มเติม และสักพักพื้นที่จัดเก็บข้อมูล (ดิสก์) ก็จะเริ่มเต็ม บทนี้จะสอนวิธีจัดการและดูแลโมเดลในเครื่องคอมพิวเตอร์ตั้งแต่ต้นจนจบ สรุปเป็น 5 คำสำคัญ: **หา** → **โหลด** → **ดู** → **อัปเดต** → **ลบ** และเกือบทั้งหมดเป็นคำสั่งสั้น ๆ เพียงคำละบรรทัด

หา · อ่านหน้าโมเดลให้เป็น

แหล่งรวมโมเดลอยู่ที่ **ollama.com/library** เปิดหน้าโมเดลตัวไหนก็จะเจอรายการ "แท็ก" หรือรุ่นย่อยทั้งหมดของโมเดลนั้น หน้าตาแบบนี้ (ตัวอย่างจากหน้า qwen3.6)

```
qwen3.6:27b     17GB · 256K context window · Text, Image
qwen3.6:35b     24GB · 256K context window · Text, Image
```

อ่านแบบนี้: ชื่อโมเดล ตามด้วย `:` และชื่อแท็ก ส่วนข้อมูลท้ายบรรทัดมีสามอย่างที่ต้องเช็คทุกครั้ง: **ขนาดไฟล์** (เอาไปคำนวณตามสูตรจากบท 2) · **ความยาว context** · **รับข้อมูลแบบไหน** (Text อย่างเดียว หรือดูรูปได้ด้วย)



หน้าโมเดล qwen3.6 บน ollama.com แสดงคำสั่งติดตั้ง รายการเครื่องมือที่รองรับ และตารางแท็กพร้อมขนาดไฟล์กับความยาว context

หน้าโมเดลจริงบน ollama.com มีครบทุกอย่างที่บทนี้สอน: กล่องคำสั่งที่ก๊อปได้เลย และตาราง Models ที่บอกขนาดไฟล์ · context · ชนิดข้อมูลของทุกแท็ก (ภาพหน้าเว็บ ollama.com)

แท็กพิเศษที่ควรรู้จักมีสามแบบ

- **latest** คือแท็กที่ได้เมื่อพิมพ์ชื่อโมเดลเฉยๆ โดยไม่ระบุแท็ก เช่น `ollama run qwen3.6` จะได้ตัวที่ทีมงานตั้งเป็นค่าเริ่มต้น ซึ่งไม่จำเป็นต้องเป็นตัวเล็กสุด เช็คขนาดก่อนเสมอ
- **-mlx** คือรุ่นสำหรับชิป Apple โดยเฉพาะ (เรื่องของบท 11)
- **-cloud** คือรุ่นที่ประมวลผลบนเซิร์ฟเวอร์ของ Ollama **ไม่ใช่ local** ตรงนี้ต้องนึกถึงคำเตือนในบท 1: ถ้างานนี้มีข้อกำหนดว่าห้ามส่งข้อมูลออกนอกเครื่อง อย่าใช้แท็กนี้

โหลด · pull

หากต้องการดาวน์โหลดไฟล์โมเดลมาเก็บไว้ในเครื่องโดยยังไม่ต้องเปิดใช้งานหน้าแชท ให้ใช้คำสั่ง `pull`

```
ollama pull qwen3.5:4b
```

ต่างจาก `run` ตรงที่ `pull` โหลดอย่างเดียวแล้วจบ เหมาะกับตอนเตรียมเครื่องล่วงหน้า เช่น โหลดโมเดลไว้ก่อนขึ้นเครื่องบิน

ดู · ls กับ ps

สองคำสั่งนี้ใช้สำหรับตรวจสอบสถานะโมเดลในเครื่อง คำแรกใช้เพื่อดูว่า**มีโมเดลใดถูกดาวน์โหลดมาไว้ในเครื่องแล้วบ้าง**

```
ollama ls
```

ได้รายชื่อโมเดลทุกตัวในเครื่อง พร้อมขนาดไฟล์ของแต่ละตัว นี่คือคำตอบของคำถาม "อะไรกินดิสก์เราอยู่"

คำที่สองดูว่า**ตอนนี้ใครกำลังทำงาน**

```
ollama ps
```

ตัวอย่างผลลัพธ์จากเอกสารทางการ

```
NAME             ID              SIZE      PROCESSOR    CONTEXT    UNTIL
gemma4:latest    c6eb396dbd59    9.6 GB    100% GPU     131072     2 minutes from
```

คอลัมน์สำคัญที่สุดคือ **PROCESSOR**: ค่า `100% GPU` หมายความว่าโมเดลทำงานอยู่บนหน่วยความจำการ์ดจอทั้งหมด จึงประมวลผลคำตอบได้เร็วที่สุด ส่วน `100% CPU` หมายความว่าประมวลผลอยู่บน RAM หลักของเครื่องเพียงอย่างเดียว หากเห็นสัดส่วนแบ่งระหว่าง CPU กับ GPU เช่น `48%/52% CPU/GPU` แปลว่าหน่วยความจำการ์ดจอ (VRAM) เต็ม ตัวโมเดลจึงต้องแบ่งการทำงานไปใช้ RAM ร่วมด้วย ส่งผลให้ประมวลผลได้ช้าลงอย่างเห็นได้ชัด ในบท 6 เราจะใช้คำสั่งนี้เพื่อวิเคราะห์ปัญหารูปแบบดังกล่าว

ส่วน **UNTIL** บอกว่าโมเดลจะค้างอยู่ใน memory ถึงเมื่อไหร่ ปกติจะค้างไว้ 5 นาทีหลังใช้งานเสร็จ เพื่อให้โมเดลตอบคำถามถัดไปได้ทันที ถ้าอยากเอาโมเดลออกจาก memory ทันที ให้ใช้ `ollama stop qwen3.5:2b` ส่วนถ้าอยากดูรายละเอียดของโมเดลตัวไหน ให้ใช้ `ollama show qwen3.5:2b`

อัปเดต · ทั้งโมเดลและตัวโปรแกรม

อัปเดตโมเดล: หน้าโมเดลใน library มีวันที่อัปเดตกำกับทุกแท็ก เมื่อค่ายปล่อยรุ่นใหม่ในแท็กเดิม ให้สั่ง `ollama pull` แท็กนั้นอีกครั้งเพื่อดึงตัวล่าสุดจาก library มาแทน

อัปเดตตัว Ollama บน macOS กับ Windows โปรแกรมจะดาวน์โหลดอัปเดตอัตโนมัติ แค่คลิกไอคอนบน taskbar หรือ menubar แล้วกด "Restart to update" (หรือจะโหลดตัวติดตั้งใหม่จาก ollama.com/download มาทับก็ได้) ส่วน Linux ให้รันสคริปต์ติดตั้งเดิมอีกครั้ง

ลบ · rm และการถอนทั้งโปรแกรม

เลิกใช้โมเดลตัวไหน ลบทิ้งได้เลย พื้นที่กลับคืนมาทันที

```
ollama rm qwen3.5:4b
```

ลบแล้วอยากได้กลับมาก็แค่ `pull` ใหม่ ไม่มีอะไรเสียหาย จึงลองได้เต็มที่: โหลดมาลอง ถ้าไม่ถูกใจก็ลบ แล้วเริ่มใหม่ได้เรื่อยๆ

ถ้าอยากเช็คด้วยตัวเอง ไฟล์โมเดลของแต่ละระบบอยู่ในตำแหน่งต่อไปนี้

ระบบ	ที่เก็บโมเดล
macOS	`~/.ollama/models`
Windows	`C:\Users\<ชื่อผู้ใช้>\.ollama\models`
Linux	`/usr/share/ollama/.ollama/models`

ถ้าดิสก์หลักเต็ม ก็ย้ายที่เก็บได้โดยตั้งตัวแปรสภาพแวดล้อม `OLLAMA_MODELS` ให้ชี้ไปที่โฟลเดอร์ใหม่ (เช่นไดรฟ์ D บน Windows) วิธีตั้งค่าอยู่ใน docs.ollama.com/faq

ถ้าจะถอน Ollama ทั้งโปรแกรม แต่ละระบบมีวิธีต่างกัน: **Windows** ถอนจากหน้า Add or remove programs ตามปกติ ข้อควรระวังจากเอกสารคือ ถ้าเคยย้ายที่เก็บด้วย `OLLAMA_MODELS` ตัวถอนการติดตั้งจะไม่ลบไฟล์โมเดลให้ จึงต้องลบโฟลเดอร์นั้นเอง ส่วน **macOS** ให้ลบแอปกับข้อมูลหลักด้วย

`sudo rm -rf /Applications/Ollama.app` และ `rm -rf ~/.ollama` (รายการไฟล์ย่อยครบชุดอยู่ที่ docs.ollama.com/macos) ขณะที่ **Linux** มีวิธีถอน service อยู่ที่ docs.ollama.com/linux

> **ลองเลย**
> ทดลองใช้งานให้ครบทั้งวงจรในรอบเดียว: ใช้คำสั่ง `ollama pull` เพื่อดาวน์โหลดโมเดลที่เลือกไว้จากบท 2 เข้ามาเพิ่ม ➡️ ใช้ `ollama ls` เพื่อตรวจสอบรายชื่อโมเดลทั้งหมดในเครื่องและขนาดพื้นที่จัดเก็บรวม ➡️ ทดลองคุยกับโมเดลตัวใหม่สักสองสามประโยค แล้วรัน `ollama ps` ตรวจสอบคอลัมน์ PROCESSOR ว่าทำงานบน GPU เต็มร้อยเปอร์เซ็นต์ (`100% GPU`) หรือไม่ ➡️ หากต้องการลบโมเดลตัวใด ให้ใช้คำสั่ง `ollama rm` และตรวจสอบด้วย `ollama ls` อีกครั้งเพื่อยืนยันว่าไฟล์ถูกลบออกไปเรียบร้อยแล้ว

เมื่อรู้วิธีจัดการและดูแลโมเดลแล้ว บทถัดไปจะเป็นการตั้งค่าและปรับแต่งโมเดลให้สอดคล้องกับการใช้งานและขีดจำกัดหน่วยความจำของเครื่อง ตั้งแต่นิสัยการตอบไปจนถึงขนาดความจำบทสนทนา (Context)

บทที่ 7

ปรับแต่งโมเดลให้พอดีกับหน่วยความจำ

ขณะนี้โมเดลยังคงใช้ค่าตั้งต้นจากระบบ มีพฤติกรรมการตอบคำถามที่หลากหลายเกินไป มีข้อจำกัดด้านความยาวของบทสนทนา และอาจทำงานช้าลงในบางกรณี บทนี้จะสอนวิธีปรับแต่ง 3 ส่วนหลัก เพื่อให้สอดคล้องกับความต้องการของเรา ได้แก่ นิสัยและพฤติกรรมการตอบคำถาม ขนาดความยาวหน่วยความจำบทสนทนา (Context) และวิธีตรวจสอบเมื่อระบบตอบช้าหรือมีข้อผิดพลาด

ปั้นนิสัยด้วย system prompt และ Modelfile

system prompt คือข้อกำหนดหรือคำสั่งที่ตัวโมเดลจะยึดถือในทุกบทสนทนา เช่น สั่งให้ตอบเป็นภาษาไทยเสมอ ตอบอย่างกระชับ หรือเน้นวิเคราะห์งานเอกสาร หากต้องการกำหนดค่านี้เป็นการถาวรใน Ollama สามารถทำได้โดยเขียนไฟล์ **Modelfile** ซึ่งทำหน้าที่เป็นเทมเพลตสำหรับสร้างโมเดลที่เราปรับแต่งเอง

มาทดลองสร้างโมเดลที่กำหนดระบบภาษาไทยจากโมเดลสาธิตของเล่มนี้ เริ่มจากเปิดโปรแกรมแก้ไขข้อความ แล้วสร้างไฟล์ชื่อ `Modelfile` (ไม่ต้องระบุนามสกุลไฟล์) โดยใส่เนื้อหาดังนี้

```
FROM qwen3.5:2b
PARAMETER temperature 0.7
SYSTEM """คุณคือโมเดลสนับสนุนงานสำนักงาน ตอบเป็นภาษาไทยเสมอ
ตอบตรงคำถาม กระชับ ถ้าไม่แน่ใจให้บอกว่าไม่แน่ใจ"""
```

สามบรรทัดนี้มีความหมายตรงไปตรงมา: `FROM` เลือกโมเดลตั้งต้น · `PARAMETER temperature` กำหนดความหลากหลายของคำตอบ (ค่าเริ่มต้น 0.8 ยิ่งสูง คำตอบยิ่งสร้างสรรค์ ยิ่งต่ำ คำตอบยิ่งนิ่งและคงเส้นคงวา) · `SYSTEM` คือคำสั่งประจำตัวที่ใส่ไว้ระหว่างเครื่องหมายคำพูดสามตัว

จากนั้นสร้างและเรียกใช้

```
ollama create thai-assistant -f Modelfile
ollama run thai-assistant
```

เท่านี้ก็จะมีโมเดลชื่อ `thai-assistant` เพิ่มขึ้นในระบบอีกหนึ่งตัว ซึ่งโมเดลนี้จะอยู่ในรายการที่แสดงเมื่อรันคำสั่ง `ollama ls` เช่นเดียวกับโมเดลปกติ และสามารถลบได้ด้วยคำสั่ง `ollama rm` เช่นกัน

อยากรู้ว่าโมเดลตัวไหนตั้งค่าจากโรงงานมายังไง แอบดูพิมพ์เขียวของมันได้

```
ollama show --modelfile qwen3.5:2b
```

ขยายความจำด้วย context length

บท 2 อธิบายไว้ว่า context คือความจำระยะสั้นของบทสนทนา และยิ่งยาวก็ยิ่งใช้ memory มาก ค่า context เริ่มต้นของ Ollama ขึ้นกับ VRAM ของเครื่อง (ต่ำกว่า 24GB ได้ 4K · 24-48GB ได้ 32K · 48GB ขึ้นไปได้ 256K) เครื่องส่วนใหญ่จึงเริ่มที่ 4K โทเคน ซึ่งพอสำหรับถามตอบทั่วไป แต่สั้นเกินไปเมื่อต้องอ่านเอกสารยาวๆ เอกสารทางการแนะนำให้ตั้ง context อย่างน้อย 64K สำหรับงานเอกสารและ agent

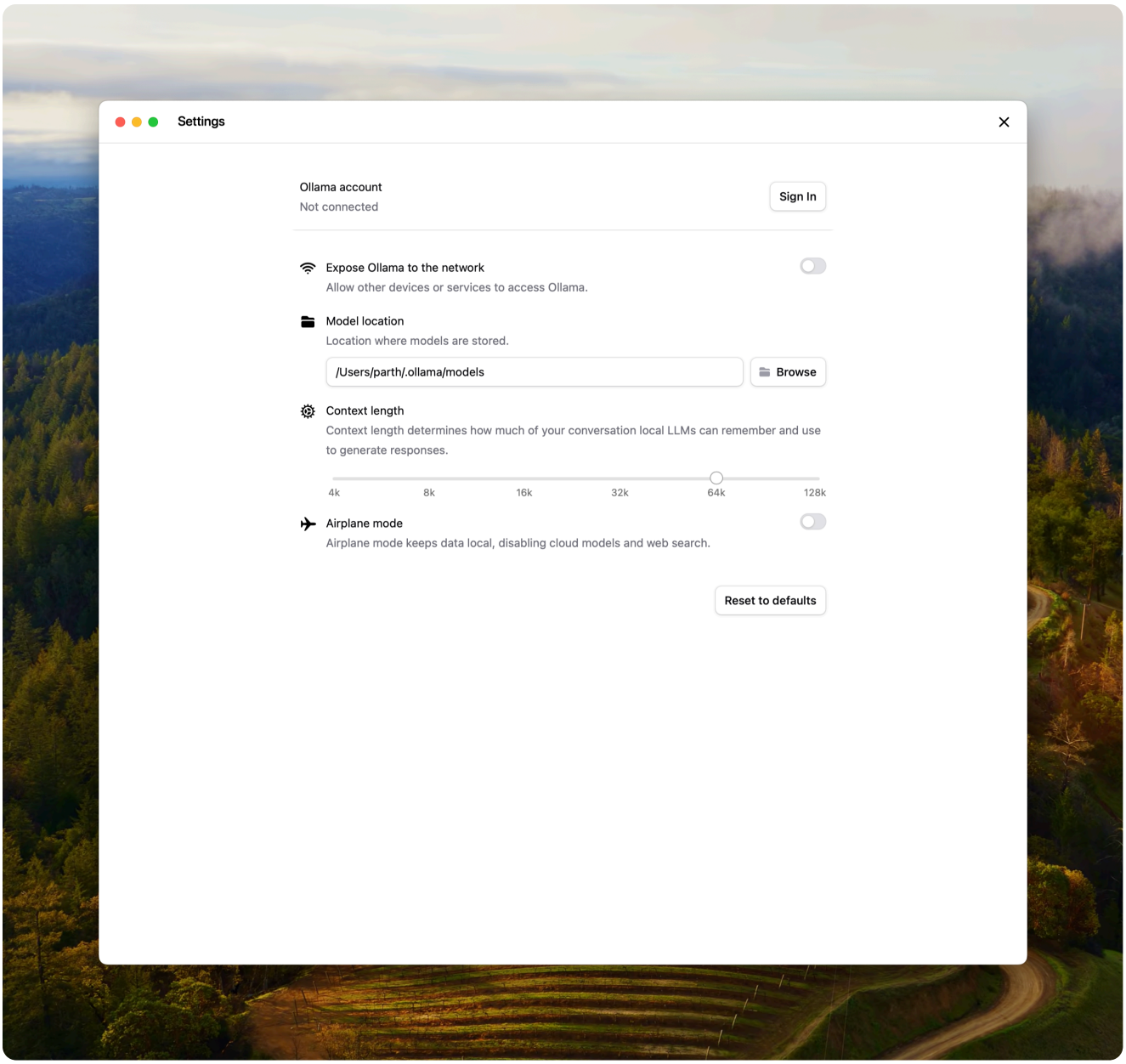
วิธีปรับมีสามระดับ

ชั่วคราวในห้องแชท ระหว่าง `ollama run` พิมพ์

```
/set parameter num_ctx 32768
```

ตั้งถาวรเฉพาะโมเดล เพิ่มบรรทัด `PARAMETER num_ctx 32768` ใน Modelfile ของเราแล้ว `create` ใหม่

ทั้งระบบ ตั้งตัวแปรสภาพแวดล้อม `OLLAMA_CONTEXT_LENGTH` หรือถ้าใช้แอป Ollama ก็เลื่อนแถบ context ในหน้า settings ได้เลย



หน้า settings ของแอป Ollama มีช่องกำหนดที่เก็บโมเดล แถบเลื่อน context length ตั้งแต่ 4k ถึง 128k และสวิตช์ Airplane mode

หน้า settings ของแอป Ollama: แถบเลื่อน Context length ช่วยให้ปรับค่าได้โดยไม่ต้องพิมพ์คำสั่ง ในหน้าเดียวกันยังมีช่องย้ายที่เก็บโมเดล (อธิบายไว้ในบท 5) และสวิตช์ Airplane mode (จะใช้ในบท 13) (ภาพจากเอกสารทางการ)

สิ่งที่ต้องคำนึงถึงเป็นไปตามหลักการคำนวณหน่วยความจำ: context ยิ่งยาวก็ยิ่งใช้หน่วยความจำมาก หากตั้งไว้สูงเกินกว่าความจุของสเปกเครื่อง ระบบการประมวลผลจะช้าลง สามารถตรวจสอบผลลัพธ์ได้ด้วยคำสั่ง `ollama ps` โดยสังเกตว่าคอลัมน์ PROCESSOR ยังแสดงสถานะเป็น `100% GPU` และคอลัมน์ CONTEXT แสดงค่าที่กำหนดใหม่หรือไม่

วิธีวิเคราะห์การทำงานของระบบ

สามอาการยอดฮิต พร้อมวิธีวินิจฉัยจากข้อมูลจริงโดยไม่ต้องเดา

ประมวลผลคำตอบช้าผิดปกติ ให้สันนิษฐานว่าหน่วยความจำการ์ดจอ (VRAM) เต็ม จากนั้นรันคำสั่ง `ollama ps` เพื่อตรวจสอบคอลัมน์ PROCESSOR หากหน้าจอแสดงการแบ่งสัดส่วนการประมวลผลระหว่าง CPU กับ GPU (เช่น `48%/52% CPU/GPU`) หมายความว่าขนาดโมเดลหรือความยาว context ที่ตั้งไว้มีขนาดใหญ่เกินกว่าความจุ VRAM วิธีการแก้ไขเรียงตามลำดับคือ: ลดขนาด context ลง ➡️ เปลี่ยนไปใช้แท็กโมเดลที่มีขนาดไฟล์เล็กลง ➡️ หรือยอมรับความเร็วที่ลดลงหากงานไม่รีบด่วน

ลืมต้นบทสนทนา หรือตอบเหมือนไม่เห็นเอกสารที่เพิ่งวาง นี่เป็นอาการคลาสสิกเมื่อ context เต็ม ส่วนที่เกินความจำจะไม่ถูกใช้ ทางแก้คือเพิ่ม num_ctx ตามหัวข้อที่แล้ว หรือเริ่มบทสนทนาใหม่ให้ความจำว่าง

เครื่องอืดทั้งที่เลิกคุยแล้ว ตามค่าปกติ โมเดลจะยังค้างอยู่ใน memory เป็นเวลา 5 นาทีหลังใช้เสร็จ ถ้าต้องการคืนพื้นที่ใน memory ทันที ให้ใช้ `ollama stop` ตามด้วยชื่อโมเดลนั้น

ถ้าเจออาการแปลกกว่านี้ ให้ดูบันทึกการทำงานของ Ollama ได้ตรงๆ โดยบน macOS ให้รัน `cat ~/.ollama/logs/server.log` ส่วน Windows ให้กด `Win+R` พิมพ์ `explorer %LOCALAPPDATA%\Ollama` แล้วเปิดไฟล์ `server.log` (คู่มือแก้ปัญหาฉบับเต็มอยู่ที่ docs.ollama.com/troubleshooting และภาคผนวกของเล่มนี้รวมอาการยอดฮิตไว้ให้แล้ว)

> **ลองเลย**
>
> ทดลองสร้างโมเดลภาษาไทยเฉพาะทางของตนเองตามแนวทางข้างต้น โดยแก้ไขส่วน `SYSTEM` ให้ตรงกับความต้องการและประเภทงานของคุณจริง ๆ เช่น กำหนดให้สรุปรายงาน หรือให้จำลองเป็นฝ่ายตอบคำถามลูกค้า จากนั้นใช้คำสั่ง `ollama create` ตามด้วย `ollama run` เพื่อเปิดใช้งานและทดสอบการตอบคำถาม หากต้องการแก้ไขเพิ่มเติม สามารถกลับไปแก้ไขเนื้อหา SYSTEM ใน Modelfile แล้วสั่ง create ทับชื่อเดิมได้ทันที

เมื่อปรับแต่งค่าการทำงานของโมเดลได้ตามความเหมาะสมแล้ว แต่การสั่งงานยังทำผ่าน Terminal เป็นหลัก บทถัดไปเราจะติดตั้งระบบหน้าแชทส่วนหน้า (Frontend Interface) เพื่อให้คุยผ่านเว็บเบราว์เซอร์ได้สะดวกขึ้น โดยเชื่อมต่อกับระบบ Ollama หลังบ้านตัวเดิม

บทที่ 8

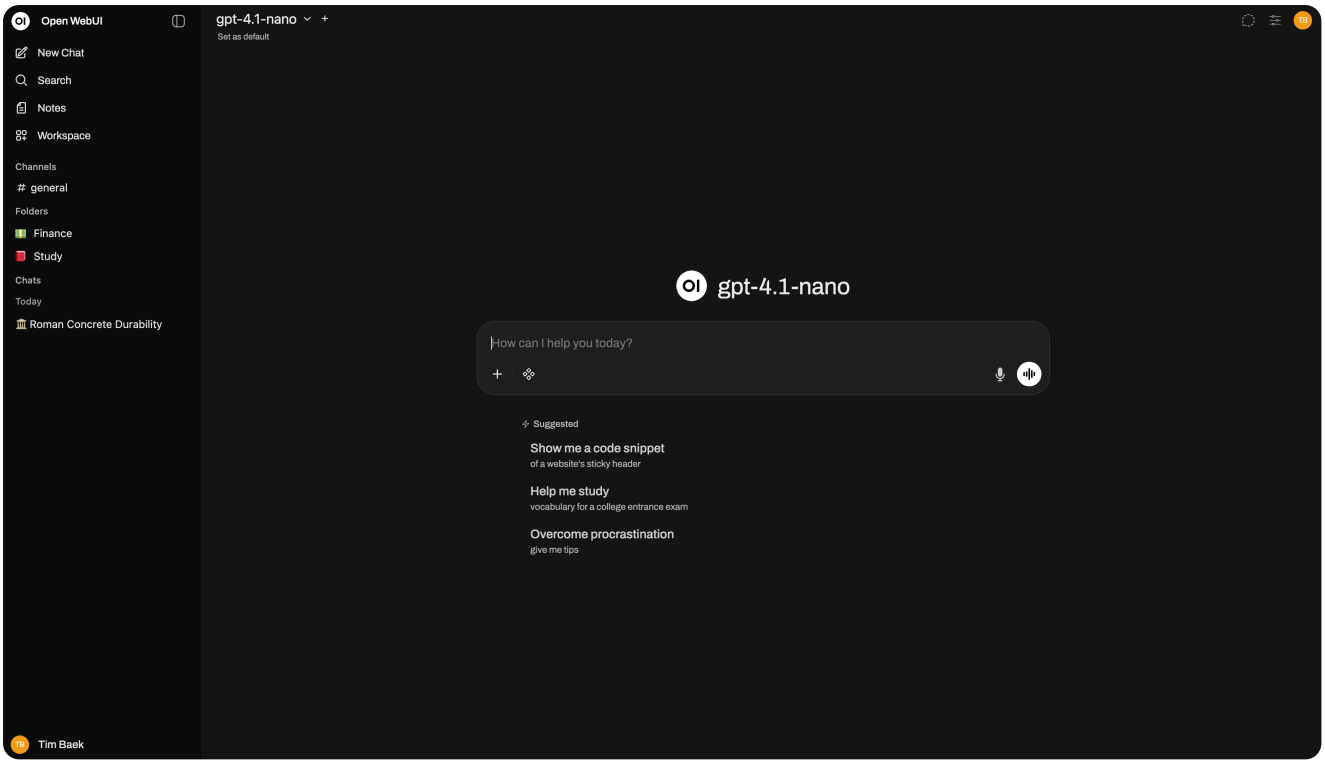
เพิ่มหน้าแชทด้วย Open WebUI

Terminal เหมาะสำหรับการสั่งการและจัดการระบบ แต่สำหรับการพิมพ์พูดคุยระยะยาว อ่านข้อมูลตาราง โค้ดคอมพิวเตอร์ หรือย้อนดูประวัติบทสนทนาเก่า หน้าต่าง Terminal อาจไม่สะดวกนัก ข่าวดีคือระบบ Ollama หลังบ้านได้จัดเตรียมช่องทางเชื่อมต่อรอไว้เรียบร้อยแล้ว เหลือเพียงขั้นตอนการติดตั้งโปรแกรมแสดงหน้าจออินเตอร์เฟสสำหรับพิมพ์แชทเข้ามาเชื่อมต่อเพิ่มเติมเท่านั้น

ประตูที่ชื่อ API

เราไม่จำเป็นต้องสั่งงาน Ollama ผ่าน Terminal เท่านั้น เนื่องจาก Ollama มีการเปิดใช้งาน API (ช่องทางสำหรับเชื่อมต่อและสื่อสารระหว่างโปรแกรม) ไว้ที่ `http://localhost:11434` ตลอดเวลาการทำงาน ส่งผลให้โปรแกรมอื่น ๆ ภายในเครื่องคอมพิวเตอร์ของเราสามารถส่งคำสั่งมาประมวลผลกับโมเดลที่ติดตั้งไว้ในเครื่องได้ทันที นี่คือกลไกการเชื่อมต่อรูปแบบเดียวกับที่จะใช้ในบท 15 สำหรับเครื่องมือช่วยเขียนโค้ด

เล่มนี้เลือก **Open WebUI** มาเป็นหน้าแชทสำหรับ API นี้ Open WebUI เป็นโปรเจกต์โอเพนซอร์สยอดนิยมที่มีหน้าตาเหมือนเว็บแชท AI ที่เราคุ้นเคย แต่ทุกอย่างรันในเครื่องเราเอง ใช้ได้ทั้ง Windows · macOS · Linux



หน้าแชทของ Open WebUI มีช่องพิมพ์ข้อความ ตัวเลือกโมเดลด้านบน และแถบด้านข้างที่แสดงประวัติแชทพร้อมโฟลเดอร์สำหรับจัดหมวดหมู่

หน้าแชทของ Open WebUI: เลือกโมเดลได้จากมุมซ้ายบน มีทั้งประวัติการคุย โฟลเดอร์จัดหมวด และช่องค้นหา ใช้งานได้ครบเหมือนเว็บแชทที่คุ้นเคย แต่ทุกอย่างอยู่ในเครื่องเรา (ภาพจากโปรเจกต์ทางการ)

ติดตั้งผ่าน Python

เอกสารของ Open WebUI ระบุว่าวิธีที่ง่ายที่สุดคือติดตั้งผ่าน pip (pip เป็นตัวติดตั้งแพ็กเกจของภาษา Python ถ้าเครื่องยังไม่มี Python ให้ดาวน์โหลดและติดตั้งจาก python.org ก่อน)

```
pip install open-webui
```

แล้วเปิดเซิร์ฟเวอร์

```
open-webui serve
```

เสร็จแล้วเปิดเบราว์เซอร์ไปที่ `http://localhost:8080` ก็จะเจอหน้าแชทของเราเอง Open WebUI มีระบบล็อกอินของตัวเอง (ทุกอย่างเก็บไว้ในเครื่อง และใน docs มีโหมด single-user สำหรับผู้ใช้คนเดียว) จากนั้นเลือกโมเดลที่มีในเครื่องแล้วเริ่มคุยได้เลย

สาย Docker
ถ้าใช้ Docker อยู่แล้ว เอกสารทางการแนะนำวิธีนี้เป็นหลัก: `docker run -d -p 3000:8080 -v open-webui:/app/backend/data --name open-webui ghcr.io/open-webui/open-webui:main` แล้วเปิด `http://localhost:3000` ส่วนกรณีพิเศษ เช่น ใช้ GPU ช่วยประมวลผล หรือต่อ Ollama ที่อยู่คนละเครื่องด้วย `OLLAMA_BASE_URL` ดูที่ docs.openwebui.com

ใช้วงจรเดิมดูแลหน้าแชทได้

ถ้าติดตั้งผ่าน pip เราก็ดูแล Open WebUI ด้วยวงจร 5 คำเดียวกับโมเดลเป๊ะ

- **อัปเดต**: `pip install -U open-webui` แล้วรัน `open-webui serve` ใหม่
- **ลบ**: `pip uninstall open-webui` และถ้าอยากล้างข้อมูลแชทด้วย ให้ลบโฟลเดอร์ `~/.open-webui`

จุดที่คนงงบ่อยคือความสัมพันธ์ระหว่างสองโปรแกรม: **Ollama ดูแลโมเดล ส่วน Open WebUI เป็นหน้าแชท** ต่อให้ลบ Open WebUI ทิ้ง โมเดลทุกตัวก็ยังอยู่ครบ เราก็ยังกลับไปคุยทาง terminal ได้เหมือนเดิม แต่ถ้า Ollama ไม่ได้ทำงาน หน้าเว็บก็จะไม่มีโมเดลให้เลือก

ทำไมบทนี้สั้น

เนื่องจากโครงสร้างการเชื่อมต่อถูกเตรียมไว้ตั้งแต่การติดตั้ง Ollama ในบท 4 แล้ว บทนี้จึงเป็นเพียงการเปิดใช้งานโปรแกรมแสดงผลหน้าจอเว็บเพื่อเชื่อมต่อกับ API ดังกล่าวเท่านั้น หน้าจอนี้ยังมีฟังก์ชันการทำงานอีกหลายส่วนที่จะนำมาใช้ต่อในบทถัด ๆ ไป โดยเฉพาะการอัปโหลดและส่งไฟล์เอกสารเข้าคลังข้อมูลเพื่อให้โมเดลวิเคราะห์ข้อมูล ซึ่งจะกล่าวถึงอย่างละเอียดในบท 14

ลองเลย
ติดตั้ง Open WebUI ตามวิธีที่ถนัด เปิดหน้าเว็บ เลือกโมเดลตัวเดิม แล้วถามคำถามเดียวกับที่เคยถามผ่าน terminal ในบท 4 ลองสังเกตว่าการจัดหน้าช่วยให้อ่านคำตอบง่ายขึ้นแค่ไหน โดยเฉพาะตารางกับโค้ด

จบภาค 2 แล้ว: ติดตั้งเป็น ดูแลเป็น จูนเป็น และมีหน้าแชทพร้อมใช้ ภาคถัดไปพาไปดูสูตรเฉพาะของเครื่องแต่ละแบบ เริ่มจากเครื่องที่อยู่ในกระเป๋าทุกคน: มือถือ

บทที่ 9

การใช้งาน AI ออฟไลน์บนสมาร์ตโฟน

เครื่องคอมพิวเตอร์ที่พกพาติดตัวตลอดเวลาไม่ใช่โน้ตบุ๊ก แต่คือสมาร์ตโฟน และประสิทธิภาพของมือถือในปัจจุบันสามารถใช้รันโมเดลขนาดเล็ก (Tiny Models) ได้จริง บทนี้เป็นข้อยกเว้นเดียวของเล่มเนื่องจาก **Ollama ไม่มีเวอร์ชันสำหรับมือถือ** เราจึงจำเป็นต้องเปลี่ยนแอปพลิเคชันที่ใช้งาน แต่วิธีการคิดคำนวณขนาดและขั้นตอนการจัดการโมเดลจากบท 2 และบท 5 ยังคงนำมาปรับใช้ได้ครบถ้วน

เครื่องแบบนี้เหมาะกับใคร

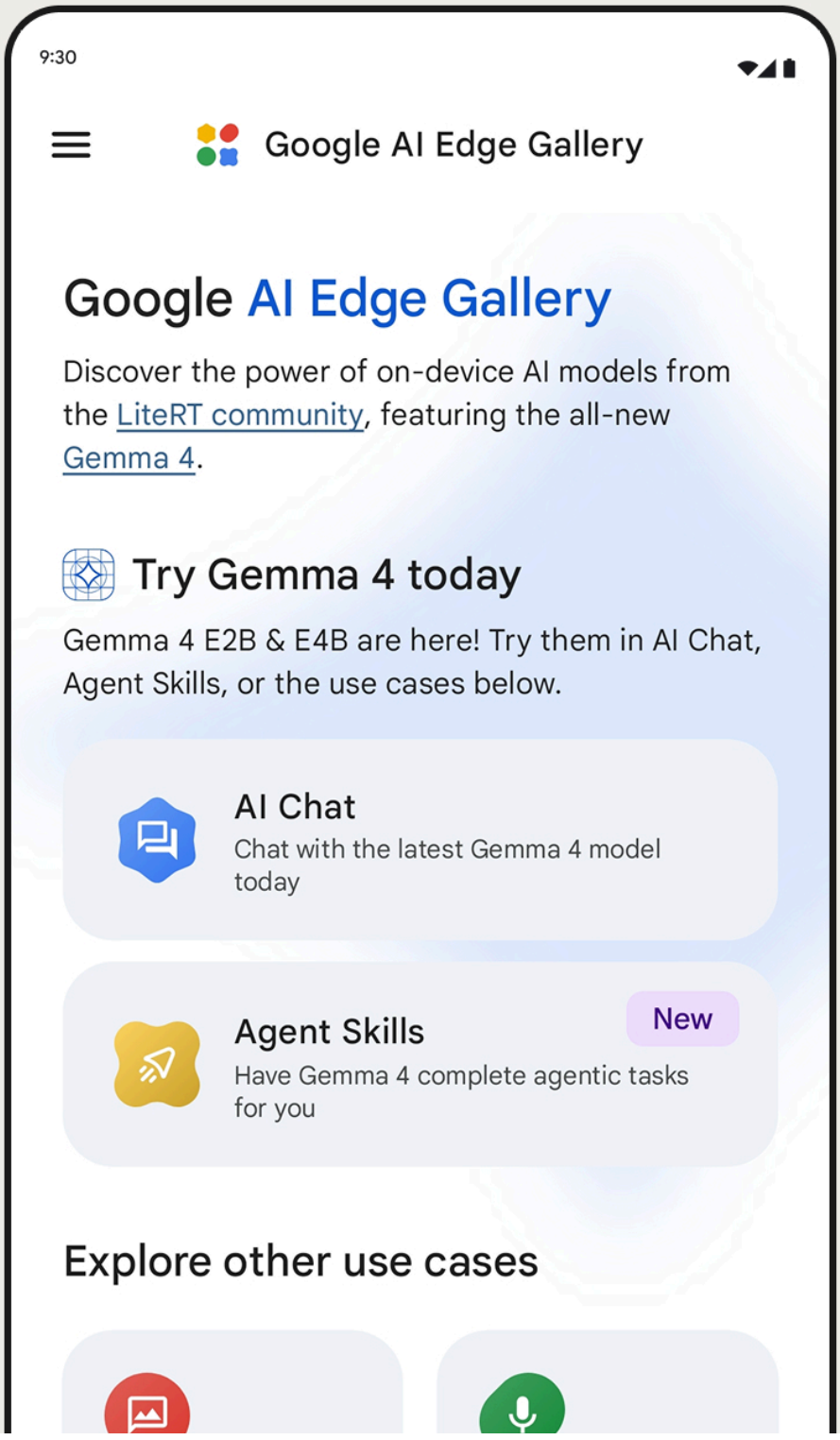
เหมาะสำหรับผู้ที่ต้องการใช้งาน AI แบบออฟไลน์ขณะพกพา เช่น การสรุปเนื้อหาข้อความหรือจดบันทึกไอเดียในพื้นที่ที่ไม่มีสัญญาณอินเทอร์เน็ต และต้องการความมั่นใจว่าข้อมูลบทสนทนาทั้งหมดจะถูกเก็บรักษาไว้ภายในเครื่องอย่างปลอดภัย สมาร์ตโฟนถือเป็นอุปกรณ์ที่มีทรัพยากรหน่วยความจำต่ำที่สุดในตารางเปรียบเทียบจากบท 2 โดยมี RAM ประมาณ 4-8GB ซึ่งต้องแบ่งการทำงานร่วมกับระบบปฏิบัติการและแอปพลิเคชันอื่น ๆ โมเดลที่เหมาะสมกับการรันจึงมีขนาดประมาณ 1-4B

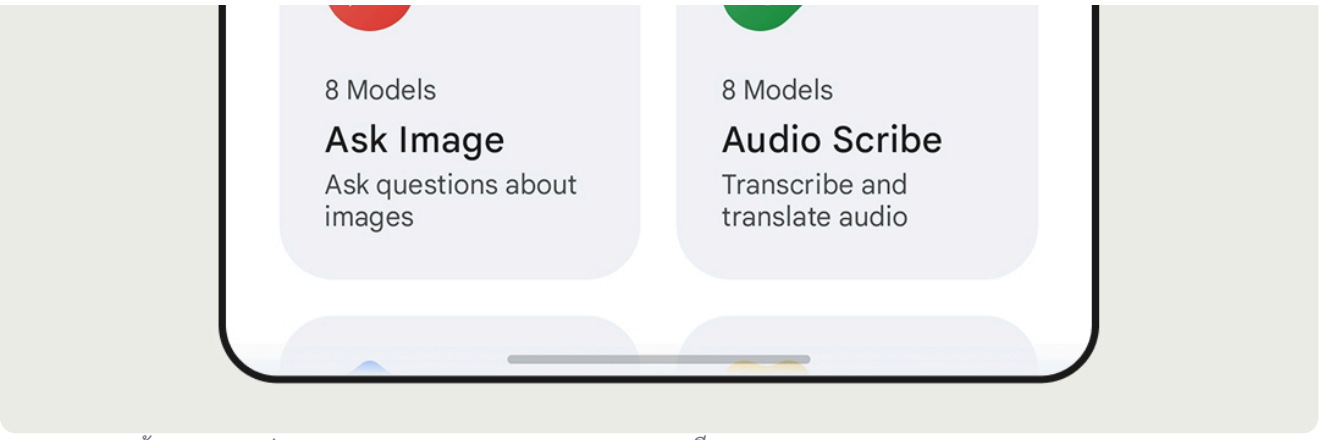
สองแอปที่แนะนำ

PocketPal AI เป็นแอปโอเพนซอร์ส (MIT) ที่มีทั้ง iOS และ Android จุดขายตรงกับหนังสือเล่มนี้เป๊ะ เพราะทุกอย่างอยู่ในเครื่อง ไม่ต้องสมัครบัญชี และไม่มีค่าสมาชิก แอปใช้โมเดล GGUF ตระกูลเดียวกับที่ Ollama ใช้ (Gemma · Qwen · Phi · Llama) โดยค้นหาโมเดลจาก Hugging Face แล้วโหลดผ่านแอปได้เลย การดูแลโมเดลตามบท 5 ก็ทำได้ในเมนูเดียว เพียงเปิดเมนู Models เพื่อค้นหา โหลด ดูขนาด และลบโมเดล

Google AI Edge Gallery เป็นแอปทางการจาก Google (Android 12 ขึ้นไป · iOS 17 ขึ้นไป) ชูตระกูล Gemma 4 เป็นตัวเด่น มีโหมดดูกระบวนการคิดของโมเดล ถามคำถามจากรูปที่ถ่ายด้วยกล้องและถอดเสียงเป็นข้อความ ทุกอย่างทำงานแบบออฟไลน์บนเครื่อง จึงเหมาะกับคนที่อยากลองโมเดลฝั่ง Google โดยเฉพาะ

Explore various on-device LLM use cases





หน้าแรกของแอป Google AI Edge Gallery ชวนลอง Gemma 4 มีเมนู AI Chat · Agent Skills · Ask Image · Audio Scribe

หน้าแรกของ Google AI Edge Gallery: แชทกับ Gemma 4 ถามจากรูป และถอดเสียง ทั้งหมดทำได้บนมือถือเครื่องเดียว (ภาพจากหน้าโปรเจกต์ทางการ)

โมเดลแนะนำ

โมเดล	เหมาะกับ	หมายเหตุ
Gemma 4 E2B / E4B	มือถือเป็นหลัก	Google ออกแบบตระกูล E สำหรับอุปกรณ์ขนาดเล็กโดยตรง ดูรูปและฟังเสียงได้
Qwen3.5 รุ่น 0.8B / 2B	เครื่องสเปคต่ำถึงกลาง	รองรับ 201 ภาษา เป็นตัวเลือกแรกสำหรับงานภาษาไทย
Phi-4 mini	เครื่องรุ่นแรงขึ้นมาหน่อย	ตระกูลจิ๋วของ Microsoft (MIT) รายชื่อภาษาที่รองรับอย่างเป็นทางการมีภาษาไทยอยู่ด้วย

หวังอะไรได้ และหวังอะไรไม่ได้

สิ่งที่สามารถคาดหวังได้คือ การสรุปข้อความ ร่างอีเมลหรือข้อความสั้น ๆ แปลภาษารอบสั้น ๆ และถามตอบความรู้ทั่วไป รวมถึงการเปิดใช้งานแบบออฟไลน์อย่างเต็มรูปแบบ ส่วนสิ่งที่ไม่สามารถคาดหวังได้คือ การประมวลผลเอกสารขนาดยาว งานวิเคราะห์เชิงลึกหลายขั้นตอน และการใช้ภาษาไทยที่สม่ำเสมอเท่าโมเดลขนาดใหญ่ เนื่องจากโมเดลขนาดเล็ก (Tiny Models) ถูกออกแบบมาเพื่อการใช้งานพื้นฐานขณะพกพาเท่านั้น นอกจากนี้การรันโมเดลบนมือถือยังเป็นการดึงทรัพยากรเครื่องอย่างหนัก หากประมวลผลอย่างต่อเนื่องอาจทำให้เครื่องร้อนและแบตเตอรี่ลดลงอย่างรวดเร็วซึ่งเป็นเรื่องปกติ

Bonsai 27B · ข่าวใหญ่ในวงการนี้

กรกฎาคม 2026 ทีม Prism ML ปล่อย **1-bit Bonsai 27B** โดยนำ Qwen3.6-27B (โมเดลระดับเครื่องเกมของบท 10) มาลดความละเอียดของค่าน้ำหนักเหลือ 1.125 bits ต่อพารามิเตอร์ ไฟล์จึงเหลือ ~3.9GB และรันบน iPhone 17 Pro Max ได้จริงที่ ~11 โทเคนต่อวินาที ขณะเดียวกันยังรักษาความสามารถไว้ราว 90% ของโมเดลเต็ม นี่เป็นโมเดลระดับ 27B ตัวแรกที่รันบนมือถือได้ (Apache 2.0 · มีรุ่น ternary ~7.2GB ที่รักษาความสามารถได้ 95% สำหรับโน้ตบุ๊ก) ตอนนี้ยังเหมาะกับสายทดลอง เพราะต้องใช้ตัวรันฉบับพิเศษของทีมผู้สร้าง จึงยังไม่ใช่แบบกดปุ่มเดียวแล้วใช้ได้ในแอปทั่วไป ในแนวทางเดียวกัน Microsoft มี **BitNet b1.58** โมเดล 2B ที่เทรนแบบ 1-bit มาแต่กำเนิด และรันด้วย CPU ล้วนผ่านตัวรันเฉพาะชื่อ bitnet.cpp (ยังเน้นภาษาอังกฤษ และ context สั้น) ความเคลื่อนไหวนี้เป็นสัญญาณว่าขีดความสามารถการรันโมเดลบนอุปกรณ์พกพาขนาดเล็กกำลังพัฒนาขึ้นอย่างรวดเร็ว

ลองเลย

ติดตั้ง PocketPal AI หรือ AI Edge Gallery แล้วเลือกโมเดลจิ๋วจากตารางข้างบนมาหนึ่งตัว จากนั้นเปิดโหมดเครื่องบินก่อนคุย ลองถามคำถามเดียวกับที่เคยถามในบท 4 แล้วนำคำตอบมาเทียบกัน จะเห็นทั้งความน่าทึ่ง (มันตอบได้จริงโดยไม่มีเน็ต) และข้อจำกัด (ภาษาไทยของตัวจิ๋วยังสะดุดกว่าตัวบนคอมชัดเจน)

จากอุปกรณ์พกพาขนาดเล็ก ถัดไปเราจะมาดูแนวทางสำหรับเครื่องคอมพิวเตอร์ที่คนส่วนใหญ่ใช้งานจริง นั่นคือโน้ตบุ๊กทำงานทั่วไปที่ไม่มีหน่วยประมวลผลกราฟิก (GPU) แยก

บทที่ 10

แนวทางสำหรับโน้ตบุ๊กใช้งานทั่วไป (RAM 8–16GB)

ผู้อ่านส่วนใหญ่ใช้โน้ตบุ๊กสำหรับทำงานเอกสาร ท่องเว็บ และประชุมออนไลน์ โดยไม่มีหน่วยประมวลผลกราฟิก (GPU) แยก ข่าวดีคือเครื่องคอมพิวเตอร์สเปกระดับนี้ก็สามารถใช้รันโมเดลสำหรับการใช้งานจริงได้แล้ว ไม่ได้มีจำกัดไว้แค่ทดลองรันเท่านั้น

เครื่องแบบนี้มีลักษณะอย่างไร

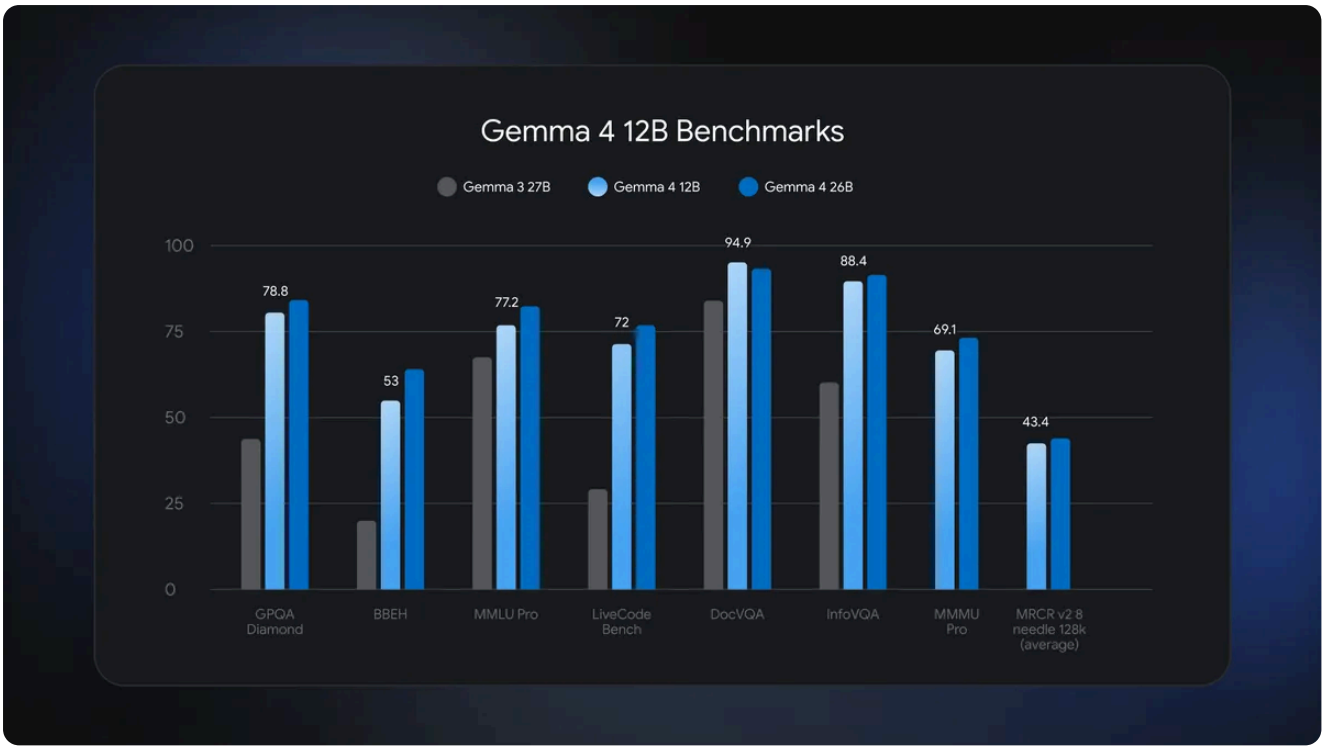
เครื่องระดับนี้ไม่มี VRAM แยก โมเดลจึงอยู่ใน RAM และประมวลผลด้วย CPU ทั้งหมด เมื่อดูใน `ollama ps` จะเห็น `100% CPU` ซึ่งเป็นเรื่องปกติ ไม่ใช่สัญญาณว่าเครื่องมีปัญหา แม้จะช้ากว่าเครื่องที่มีการ์ดจอ แต่ถ้าเลือกโมเดลให้พอดีกับสเปกก็ยังใช้งานได้สบาย สิ่งที่ต้องระวังคือ RAM ต้องใช้ร่วมกับโปรแกรมอื่น ถ้าเปิดเบราว์เซอร์หลายสิบแท็บพร้อมกับรันโมเดล RAM เต็มแน่นอน

โมเดลแนะนำ

ขนาดไฟล์จริงจาก ollama.com ณ เวลาเขียน

โมเดล	ไฟล์	เหมาะกับ
`qwen3.5:4b`	3.4GB	รุ่นเริ่มต้นสำหรับ RAM 8GB · ขยับขึ้นจากตัวสาธิต 2b
`qwen3.5:9b`	6.6GB	ตัวหลักสำหรับเครื่อง RAM 16GB ที่เน้นงานภาษาไทย
`gemma4:12b`	7.6GB	ดาวเด่นของระดับนี้ ดูรายละเอียดด้านล่าง
`gemma4:e4b`	9.6GB	สายมัลติโมดัล รับได้ทั้งรูปและเสียง

ดาวเด่นของเครื่องระดับนี้คือ **Gemma 4 12B** โดย Google ประกาศเป้าหมายตอนเปิดตัวไว้อย่างชัดเจนว่า "นำความสามารถแบบ agent มาไว้บนโน้ตบุ๊ก" ตัวเลขจาก blog ทางการก็น่าสนใจ ผลการทดสอบของโมเดลนี้เข้าใกล้รุ่น 26B ของค่ายเดียวกัน แต่ใช้ความจำไม่ถึงครึ่ง และยังออกแบบให้รันบนโน้ตบุ๊กทั่วไปที่มีความจำ 16GB ซึ่งตรงกับสเปกของเครื่องกลุ่มนี้พอดี



กราฟแท่งเปรียบเทียบคะแนน Gemma 4 12B กับ Gemma 4 26B และ Gemma 3 27B รุ่นก่อนจากข้อสอบ 8 ชุด โดย 12B ทำคะแนนตามหลัง 26B เพียงเล็กน้อยเกือบทุกชุด

กราฟจากประกาศเปิดตัว: Gemma 4 12B (ฟ้าอ่อน) ทำคะแนนตามหลังรุ่น 26B (ฟ้าเข้ม) เพียงเล็กน้อย และทิ้ง Gemma 3 27B รุ่นก่อน (เทา) ไปแล้ว ทั้งที่โมเดลรุ่นก่อนมีขนาดใหญ่กว่า เห็นได้ชัดว่าโมเดลรุ่นเล็กยุคใหม่ทำได้ดีแค่ไหน (ภาพจาก blog ทางการของ Google)

ค่าตั้งต้นสำหรับเครื่องระดับนี้

- ก่อนเปิดโมเดล ให้ปิดโปรแกรมที่กินความจำแต่ไม่ได้ใช้งาน เพื่อเหลือ RAM ไว้ให้โมเดล
- ตั้งค่า context เริ่มต้นของเครื่องระดับนี้ไว้ที่ 4K ถ้าจะให้โมเดลอ่านเอกสารยาว ค่อยเพิ่มทีละขั้นตามวิธีในบท 6 แล้วดูว่าเครื่องยังไหวไหม
- คำตอบจะเร็วแค่ไหนขึ้นอยู่กับ CPU โดยตรง ถ้าเสียบปลั๊ก CPU จะทำงานได้เต็มกำลังกว่าตอนใช้โหมดประหยัดแบต

หวังอะไรได้ · หวังอะไรไม่ได้

สิ่งที่สามารถคาดหวังได้: โมเดลสามารถเขียนภาษาไทยได้คล่อง สามารถสรุปเอกสาร ร่างอีเมล แปลภาษา และตอบคำถามทั่วไปได้ โดยโมเดลตระกูล gemma4 ยังรองรับการประมวลผลข้อมูลรูปภาพอีกด้วย สิ่งที่ไม่สามารถคาดหวังได้: ความเร็วในการประมวลผลคำตอบที่ยาวมาก ๆ และความสามารถ

ในการรันโมเดลขนาดใหญ่ตั้งแต่ระดับ 20B ขึ้นไป ซึ่งจะกล่าวถึงในสองบทถัดไป หากทดลองใช้งานแล้วพบข้อจำกัดและต้องการสเปกที่สูงขึ้น นั่นคือสัญญาณว่าถึงเวลาศึกษาคู่มือในบท 10 และ 11 ก่อนตัดสินใจอัปเกรดฮาร์ดแวร์

> **ลองเลย**
> ทดลองเปลี่ยนไปใช้โมเดลที่มีขนาดใหญ่ขึ้นด้วยคำสั่ง `ollama pull qwen3.5:4b` (หรือ `qwen3.5:9b` สำหรับเครื่องที่มี RAM 16GB) แล้วส่งคำถามภาษาไทยชุดเดียวกับที่เคยใช้ถามโมเดลขนาด 2b ในบท 4 จากนั้นลองเปรียบเทียบคำตอบของทั้งสองตัว ความแตกต่างในความรอบรู้คือผลลัพธ์จากขนาดโมเดลที่ใหญ่ขึ้น และเป็นสาเหตุที่ตารางเปรียบเทียบในบท 2 ต้องจัดระดับโมเดลตามหน่วยความจำ เมื่อทดสอบเสร็จเรียบร้อยแล้ว อย่าลืมลบไฟล์โมเดลที่ไม่ใช้งานด้วยคำสั่ง `ollama rm` เพื่อคืนพื้นที่จัดเก็บข้อมูลตามขั้นตอนในบท 5

บทถัดไปจะขยับขึ้นอีกระดับด้วยเครื่องที่มีการ์ดจอ NVIDIA ซึ่งต่างออกไปทั้งด้านความเร็วและขนาดโมเดลที่รันได้

บทที่ 11

สูตรจัดสเปกคอมเล่นเกม NVIDIA (VRAM 8–24GB)

เครื่องที่ประกอบไว้เล่นเกมคือม้ามืดของวงการนี้ การ์ดจอที่ซื้อมาเพื่อเพิ่ม frame rate กลายเป็นอุปกรณ์รันโมเดลชั้นดี เพราะการ์ดจอเล่นเกมมีสิ่งที่โมเดลต้องการที่สุดอยู่แล้ว: VRAM เร็วๆ กับพลังประมวลผลแบบขนาน

เครื่องแบบนี้เหมาะกับใคร

เครื่องแบบนี้เหมาะกับเกมเมอร์ ครีเอเตอร์สายตัดต่อ หรือใครก็ตามที่มีการ์ด NVIDIA แยกในเครื่องตัวเลขเดียวที่ชี้ชะตาคือ **VRAM ของการ์ด** (ย้ำจากบท 2: ไม่ใช่ RAM ของเครื่อง) Ollama รองรับการ์ด NVIDIA ย้อนไปถึงยุคเก่าพอสมควร โดยต้องใช้ driver เวอร์ชัน 550 ขึ้นไป การ์ด RTX ที่ใช้เล่นเกมกันในช่วงไม่กี่ปีมานี้รองรับทั้งหมด (ฝั่ง AMD Radeon ก็มีทางของตัวเองผ่าน ROCm กับ Vulkan รายชื่อรุ่นอยู่ใน docs.ollama.com/gpu)

เป้าหมายสำคัญของการตั้งค่าเครื่องระดับนี้มีข้อเดียวคือ: เลือกโมเดลที่รันแล้วคำสั่ง `ollama ps` แสดงสถานะเป็น `100% GPU` เนื่องจากหากโมเดลมีขนาดใหญ่เกินไปจนต้องแบ่งการประมวลผลไปที่หน่วยความจำระบบ (RAM) ความเร็วของระบบจะลดลงอย่างเห็นได้ชัดจนเสียจุดเด่นของการ์ดจอแยกไป

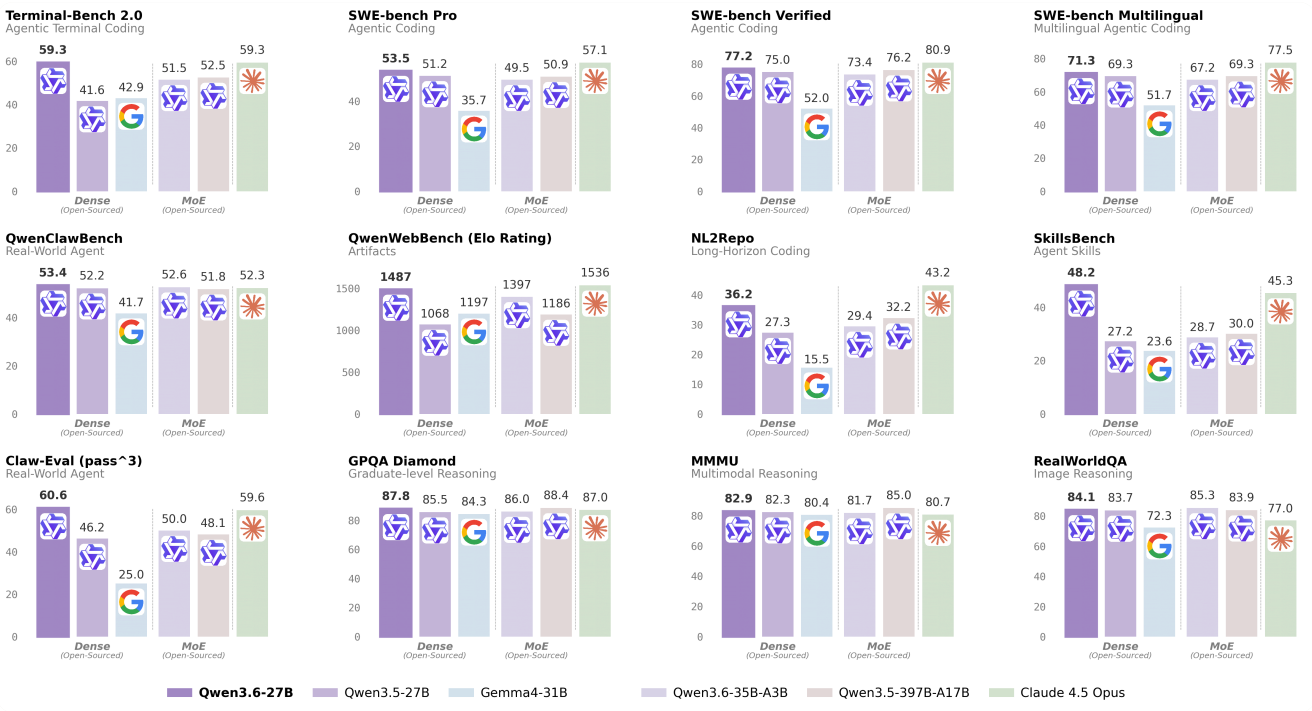
โมเดลแนะนำ · เลือกตาม VRAM

ขนาดไฟล์จริงจาก ollama.com ในวันที่เขียนบทนี้

VRAM	โมเดลที่พอดี	ไฟล์
8GB	`qwen3.5:9b` · `gemma4:12b`	6.6GB · 7.6GB
12-16GB	`gpt-oss:20b` · `gemma4:e4b`	14GB · 9.6GB
24GB (RTX 3090 / 4090)	`qwen3.6:27b` · `gemma4:26b` · `gemma4:31b` · `qwen3.5:27b`	17GB · 18GB · 20GB · 17GB

สามตัวท็อปในกลุ่ม 24GB ต่างกันชัดเจน เลือกตามงานหลักได้เลย

- `qwen3.6:27b` เหมาะกับสายเขียนโค้ดและงาน agent โดย Qwen ระบุว่ารุ่นนี้เป็น "โมเดลเขียนโค้ดระดับเรือธงขนาด 27B" พร้อม context 256K
- `gemma4:26b` โมเดลที่ใช้สถาปัตยกรรม MoE ซึ่งเรียกใช้พารามิเตอร์จริงเพียงประมาณ 4B ในระหว่างทำงาน ทำให้ประมวลผลคำตอบได้รวดเร็วเป็นพิเศษ เหมาะสำหรับการพิมพ์พูดคุยใช้งานทั่วไปในชีวิตประจำวัน
- `gpt-oss:20b` โมเดล reasoning จาก OpenAI ที่เปิดให้เห็นกระบวนการคิดและปรับระดับการใช้เหตุผลให้เหมาะกับงานได้



กราฟแท่งเปรียบเทียบคะแนน Qwen3.6-27B กับโมเดลแบบเปิดรุ่นอื่นและ Claude 4.5 Opus จากข้อสอบสายโค้ดและงาน agent 12 ชุด

ผลทดสอบจากหน้าโมเดลทางการของ Qwen3.6-27B (แท่งม่วงเข้ม) แสดงว่าโมเดลนี้ทำคะแนนนำโมเดลแบบเปิดขนาดเดียวกันในข้อสอบสายโค้ดหลายชุด และบางชุดก็ทำคะแนนเข้าใกล้โมเดลปิดระดับท็อป บทนี้จึงยกให้เป็นตัวหลักสายโค้ดสำหรับกลุ่ม 24GB (ภาพจาก model card ทางการ)

การตั้งค่าสำหรับเครื่องนี้

- การ์ด 24GB ปลดล็อกโบนัสจากบท 2 อัตโนมัติ: Ollama ตั้งค่า context เริ่มต้นเป็น 32K (จากปกติ 4K) จึงใส่เอกสารยาวได้ทันทีโดยไม่ต้องตั้งค่าเพิ่ม
- หลังเปิดโมเดลใหม่เป็นครั้งแรก ให้รัน `ollama ps` ทุกครั้ง ถ้า PROCESSOR ไม่ใช่ `100% GPU` ให้ถอยไปแท็กที่เล็กลงหนึ่งขั้น หรือลด context ตามวิธีในบท 6
- อย่าลืมว่าเกมกับโมเดลแย่ง VRAM ก้อนเดียวกัน ถ้าจะเล่นเกมหนักๆ ให้ใช้ `ollama stop` คืนพื้นที่ก่อน

หวังอะไรได้ · หวังอะไรไม่ได้

สิ่งที่คาดหวังได้: สามารถรันโมเดลสำหรับการทำงานจริงจังที่ตอบสนองรวดเร็ว ใช้ภาษาไทยได้อย่างลื่นไหลและชัดเจน เขียนโค้ดได้มีประสิทธิภาพ และประมวลผลเอกสารยาวได้ดี โดยไม่ต้องเชื่อมต่ออินเทอร์เน็ต สิ่งที่ไม่สามารถคาดหวังได้: การรันโมเดลที่มีขนาดไฟล์ 65GB ขึ้นไป เนื่องจากข้อจำกัดของพื้นที่ VRAM ที่การ์ดจออย่าง RTX 4090 ก็ไม่สามารถบรรจุได้ทั้งหมด โมเดลระดับนั้นจำเป็นต้องศึกษาแนวทางในบท 11 และ 12 เพิ่มเติม

> **ลองเลย**
>
> หาขีดจำกัดสูงสุดของการ์ดจอของคุณ: ตรวจสอบตารางด้านบนและเลือกโมเดลขนาดใหญ่ที่สุดที่ไฟล์มีขนาดเล็กกว่าความจุ VRAM ของเครื่องอย่างชัดเจน จากนั้นใช้คำสั่ง `ollama pull` เพื่อติดตั้งและรันใช้งาน และเช็คด้วย `ollama ps` ว่าสามารถประมวลผลบน GPU เต็ม 100% (`100% GPU`) จริงหรือไม่ หากทำได้สำเร็จ ลองทดสอบให้โมเดลประมวลผลโจทย์ที่มีความซับซ้อนมากกว่าปกติ แล้วเปรียบเทียบคุณภาพของผลลัพธ์กับโมเดลขนาดเล็กก่อนหน้า

ฝั่ง NVIDIA วัดกันที่ VRAM แต่ Mac ใช้หลักอีกแบบ: GPU ใช้ memory ทั้งก้อนได้ด้วย

บทที่ 12

สูตร Mac พลังของ unified memory

ในฝั่งของการ์ดจอแยก ยิ่งต้องการขนาด VRAM สูง ราคาการ์ดจอก็ยิ่งเพิ่มขึ้นเป็นลำดับขั้น แต่สำหรับ Mac ชิป Apple M series จะแตกต่างออกไปเนื่องจากใช้หน่วยความจำร่วมกัน (Unified Memory) ระหว่าง CPU และ GPU การเลือกซื้อ Mac ที่มีหน่วยความจำสูง ๆ จึงช่วยเพิ่มขีดความสามารถในการรองรับโมเดลขนาดใหญ่ได้อย่างมาก นี่คือเหตุผลที่ผู้ใช้งานที่ต้องการรันโมเดลขนาดใหญ่นิยมเลือกใช้ Mac

เครื่องแบบนี้เหมาะกับใคร

แนวทางนี้เหมาะสำหรับผู้ใช้งาน Mac ชิป Apple M series (Ollama รองรับการประมวลผลผ่าน Metal ของ GPU Apple โดยตรง และต้องทำงานบน macOS Sonoma ขึ้นไป สำหรับ Mac Intel รุ่นเก่าจะประมวลผลผ่าน CPU เท่านั้น) จุดแตกต่างสำคัญจากการ์ดจอ NVIDIA คือไม่ต้องนำสเปก RAM และ VRAM มาเปรียบเทียบกัน สามารถใช้ตัวเลข Unified Memory ของเครื่องนำไปคำนวณในสมการวิเคราะห์หน่วยความจำจากบท 2 ได้ทันที แต่เนื่องจากระบบปฏิบัติการและแอปพลิเคชันอื่น ๆ จะใช้พื้นที่หน่วยความจำร่วมกันในก้อนนี้ด้วย จึงจำเป็นต้องเผื่อพื้นที่หน่วยความจำว่างไว้มากกว่าปกติ

แท็ก –mlx · รุ่นเฉพาะสำหรับ Apple Silicon

ในหน้าโมเดลบน ollama.com หลายโมเดลมีแท็กลงท้าย `-mlx` แท็กนี้หมายถึงรุ่นที่ทำมาสำหรับ MLX ซึ่งเป็นเครื่องมือรันโมเดลบน Apple Silicon โดยเฉพาะ วิธีใช้ไม่ต่างจากแท็กปกติ แค่เลือกแท็กนี้แทน

โมเดลแนะนำ · แบ่งตามขนาด memory

ขนาดไฟล์จริงจาก ollama.com ณ เวลาเขียน

memory ของเครื่อง	โมเดลที่พอดี	ไฟล์
16GB (MacBook Air ทั่วไป)	`gemma4:12b-mlx` · `qwen3.5:9b-mlx`	7.7GB · 8.9GB

memory ของเครื่อง	โมเดลที่พอดี	ไฟล์
32GB	`gemma4:26b-mlx` · `qwen3.6:27b-mlx`	18GB · 20GB
48-64GB	`gemma4:31b-mlx` · `qwen3.6:35b-mlx`	19GB · 22GB (เปิด context ยาวๆ ได้สบาย)
96GB ขึ้นไป (Mac Studio)	`gpt-oss:120b`	65GB

แถวสุดท้ายแสดงจุดต่างสำคัญของ Mac อย่างชัดเจน: โมเดล `gpt-oss:120b` มีขนาดไฟล์ถึง 65GB ซึ่งการ์ดจอฝั่งคอมพิวเตอร์เล่นเกมทั่วไปไม่สามารถประมวลผลได้เนื่องจากข้อจำกัดของ VRAM แต่สำหรับเครื่องอย่าง Mac Studio ที่มี Unified Memory สูงเพียงพอ จะสามารถรันได้ทันทีตามการคำนวณสมการหน่วยความจำปกติ ซึ่ง OpenAI ออกแบบโมเดลขนาดนี้มาเพื่องานวิเคราะห์เชิงเหตุผล (Reasoning) ขั้นสูงโดยเทียบเท่าการทำงานบนเครื่องเซิร์ฟเวอร์ GPU ขนาด 80GB

ข้อควรทราบในการกำหนดค่า

- หลักการคำนวณหน่วยความจำร่วมกับ Context จากบท 2 สามารถนำมาปรับใช้กับกรณีนี้ได้โดยตรง: Ollama จะกำหนดค่าเริ่มต้นตามความจุหน่วยความจำที่แชร์ให้ GPU เครื่องที่มีความจุสูงจะได้รับการตั้งค่า Context เริ่มต้นยาวขึ้น โดยเฉพาะเครื่องที่มีความจุหน่วยความจำตั้งแต่ 48GB ขึ้นไปจะได้ค่า Context ยาวสูงสุดถึง 256K
- `ollama ps` ยังเป็นเพื่อนรัก: ดูให้โมเดลขึ้น `100% GPU` เหมือนเดิม
- ปิดแอปหนักๆ ก่อนรันโมเดลใหญ่ เพราะทุกแอปแย่งใช้ memory ก้อนเดียวกัน

หวังอะไรได้ · หวังอะไรไม่ได้

ข้อดี: จ่ายเพิ่มครั้งเดียวก็ได้ memory ก้อนใหญ่ เครื่องเงียบ กินไฟต่ำ และถ้าสั่ง memory มาพอก็รันโมเดลที่เครื่องเกมรันไม่ได้ ข้อจำกัด: อัปเกรดทีหลังไม่ได้ เพราะ memory ของ Mac เพิ่มภายหลังไม่ได้ จึงต้องตัดสินใจตั้งแต่วันซื้อ ใครกำลังเล็ง Mac เครื่องใหม่เพื่องานสายนี้ควรทุ่มงบให้ memory มากที่สุด

> **ลองเลย**
> ถ้าใช้ Mac อยู่ ให้ pull โมเดลจากตารางในแถวที่ตรงกับเครื่อง โดยเลือกแท็ก `-mlx` มาเทียบกับแท็กปกติของโมเดลเดียวกัน ดูทั้งความเร็วในการตอบและความลื่นของเครื่องระหว่างรัน แล้วเก็บแบบที่ชอบ ส่วนอีกแบบใช้ `ollama rm` ตามวิชาบท 5

เหลือปลายทางสุดท้ายบนแผนที่ คือโลกของโมเดลระดับร้อย B ขึ้นไป ซึ่งเครื่องทั่วไปเอื้อมไม่ถึง แต่ควรรู้จักไว้เวลาอ่านข่าว

บทที่ 13

โมเดลรุ่นใหญ่ที่เครื่องทั่วไปเอาไม่อยู่

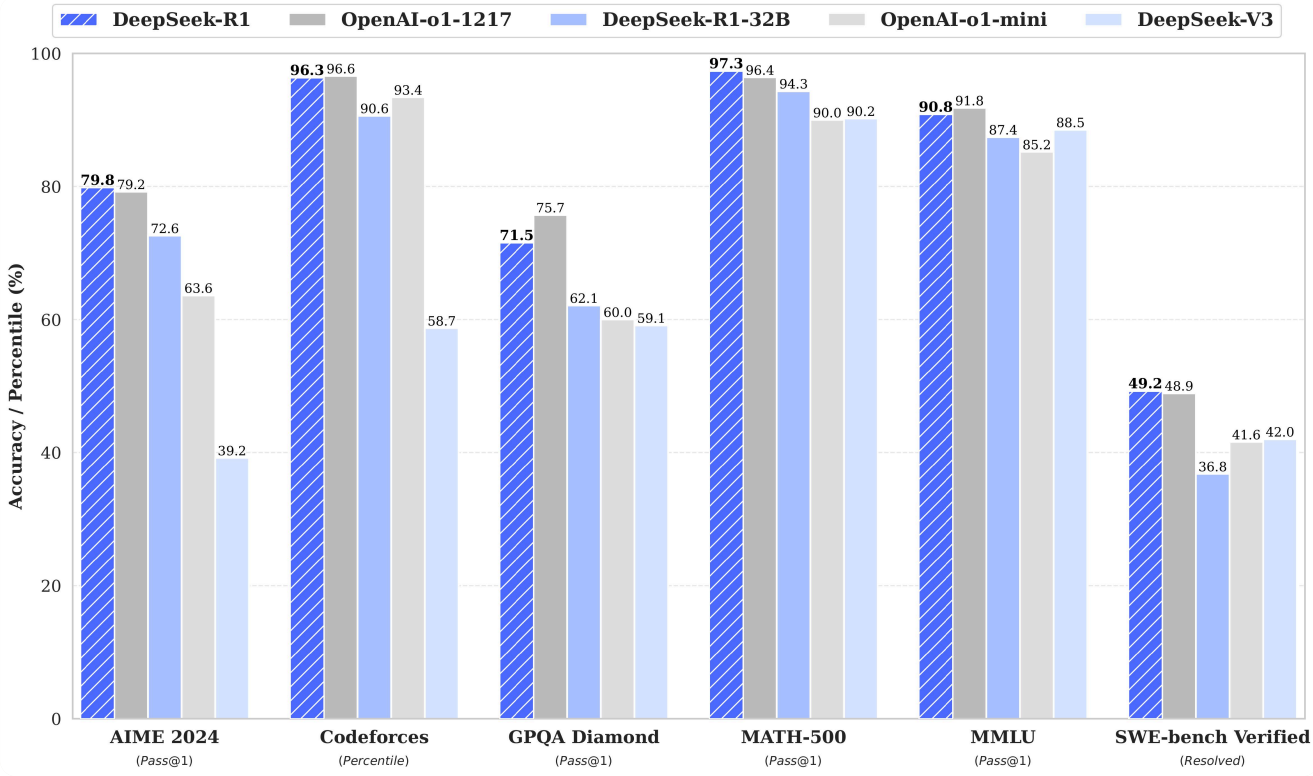
ข่าว AI ชอบหยิบตัวเลขมหึมาของโมเดลมาพาดหัว: 122B · 397B · 671B บทนี้ไม่ใช่ how-to เพราะโมเดลพวกนี้เกินกำลังเครื่องแทบทุกระดับที่พูดถึงในเล่ม แต่อ่านจบแล้วคุณจะอ่านข่าวพวกนี้รู้เรื่อง รู้ว่าแต่ละตัวต้องใช้เครื่องระดับไหน และเครื่องของตัวเองยังต้องอัปเกรดอีกแค่ไหนจึงจะรันได้

สามระดับที่เครื่องทั่วไปเอาไม่อยู่

ขั้นที่หนึ่ง · เวิร์กสเตชัน (memory ราว 70-100GB) ตัวแทนคือ `gpt-oss:120b` (ไฟล์ 65GB) กับ `qwen3.5:122b` (ไฟล์ 81GB) วิธีคำนวณสเปกเครื่องยังเหมือนเดิมและไม่ซับซ้อน: เครื่องต้องมี memory มากกว่าขนาดไฟล์พอสมควร ในทางปฏิบัติจึงต้องใช้ GPU ระดับดาต้าเซ็นเตอร์ 80GB อย่าง H100 (ตามที่ OpenAI ระบุไว้เอง) หรือ Mac Studio ที่สั่ง memory ความจุสูงอย่างที่กล่าวไว้ในบท 11 นี่คือขั้นเดียวในบทนี้ที่คนทั่วไปเอื้อมถึงได้ด้วยเงินก้อนเดียว

ขั้นที่สอง · เซิร์ฟเวอร์หลายการ์ด (หลายร้อย GB) ตัวแทนคือ **Qwen3.5-397B-A17B** โมเดลเปิดรุ่นเรือธงของ Qwen3.5 มีพารามิเตอร์รวม 397B แต่ทำงานจริงครั้งละ 17B ตัวเต็มเปิดให้โหลดได้จริงบน Hugging Face เอกสารของ Qwen ยกตัวอย่างการรันด้วยเซิร์ฟเวอร์ที่แบ่งงานข้าม GPU ถึง 8 ใบ บน Ollama โมเดลนี้จึงมีให้ใช้เป็นแท็ก cloud เท่านั้น อย่างที่เตือนไว้ในบท 1 ว่าสะดวกจริง แต่ไม่ใช่ local แล้ว

ขั้นที่สาม · ระดับสูงสุดของโมเดลเปิด ตัวแทนคือ **DeepSeek-R1** ฉบับเต็ม: 671B ทำงานจริงครั้งละ 37B โมเดลนี้สร้างกระแสฮือฮาไปทั่วโลก เพราะ open ไล่ทัน closed ได้ในงาน reasoning ตัวเต็มมีขนาดใหญ่จนแม้แต่บริษัทยังต้องคิดให้รอบคอบก่อนตัดสินใจลงทุนรัน



กราฟแท่งจากหน้าโมเดล DeepSeek-R1 เปรียบเทียบคะแนนกับ OpenAI o1 ในข้อสอบคณิตศาสตร์และโค้ด 6 ชุด ทั้งสองโมเดลได้คะแนนสูสีกันเกือบทุกชุด และยังมีผลของโมเดล distill ขนาด 32B มาเปรียบเทียบด้วย

กราฟจากหน้าโมเดลทางการของ DeepSeek-R1: ตัวเต็ม (แท่งลายน้ำเงิน) ทำคะแนนสูสีกับ OpenAI o1 (เทา) แทบทุกข้อสอบ ส่วน DeepSeek-R1-32B คือรุ่น distill ที่เครื่องซึ่งมี memory 24GB รันได้จริงตามข้อ 1 ด้านล่าง (ภาพจาก model card ทางการ)

กับดักตัวเลข MoE

สังเกตว่ารุ่นใหญ่ยุคนี้เป็น MoE แทบทั้งหมด (397B ทำงานจริง 17B · 671B ทำงานจริง 37B) ตัวเลข "ทำงานจริง" เล็กจนน่าใจชื้น แต่อย่าหลงกล: **ต้องคำนวณสเปกเครื่องจากตัวเลขรวม ไม่ใช่ตัวเลข active** เพราะพารามิเตอร์ทั้งหมดต้องอยู่ใน memory พร้อมกัน แม้ระบบจะเรียกใช้ทีละส่วน MoE ช่วยให้ประมวลผลเร็วขึ้น แต่ไม่ได้ลด memory ที่ต้องใช้

แล้วคนทั่วไปเอายังไงกับรุ่นใหญ่

มีทางเลือกที่ทำได้จริงสามทาง เรียงจากใกล้ตัวที่สุด

1. **ใช้โมเดล distill** เช่นตระกูล DeepSeek-R1 ที่ถ่ายทอดความสามารถจากตัวเต็มมายัง Qwen/Llama ขนาด 1.5B ถึง 70B เพื่อให้เครื่องทั่วไปได้ลองใช้ความสามารถด้าน reasoning ของตัวเต็ม
2. **ใช้แท็ก cloud** เมื่องานนั้นไม่ใช่งานลับ แต่ต้องรู้ไว้เสมอว่านี่ไม่ใช่การใช้งานแบบ local แล้ว

3. **รอให้เทคนิคการบีบอัดทำให้รุ่นใหญ่รันบนเครื่องทั่วไปได้** บทเรียนจากกล่อง Bonsai ในบท 8 ชัดเจน: โมเดลระดับ 27B เพิ่งรันบนมือถือได้ด้วยการบีบให้เหลือ 1-2 bits ขีดจำกัดของโมเดลที่เครื่องทั่วไปรันได้จึงขยับขึ้นทุกปี

> **ลองเลย**
>
> เปิดหน้า deepseek-r1 บน ollama.com/library แล้วไล่ดูรายการแท็กตั้งแต่ 1.5b ถึง 671b สังเกตว่าขนาดไฟล์เพิ่มขึ้นตามตัวเลข ภาพเดียวนี้สรุปทั้งบทได้: โมเดลตระกูลเดียวกัน ความสามารถเดียวกัน ต่างกันแค่สเปกเครื่องที่ต้องใช้ แล้วลองตอบตัวเองว่าเครื่องของคุณเหมาะกับแท็กไหน

จบภาค 3 แล้ว คุณจะเห็นภาพรวมสเปกเครื่องครบทุกระดับ ภาคสุดท้ายจะกลับมาที่ประเด็นสำคัญ: เมื่อติดตั้งโมเดลแล้ว จะนำไปประยุกต์ใช้งานจริงในชีวิตประจำวันอย่างไรให้เกิดประโยชน์สูงสุด โดยจะเริ่มจากงานที่เหมาะสมกับระบบ local มากที่สุด นั่นคือการจัดการเอกสารลับ

บทที่ 14

เวิร์กโฟลว์การวิเคราะห์เอกสารลับแบบออฟไลน์

ย้อนกลับไปดูงานที่ให้จดไว้ตั้งแต่บท 1: งานที่อยากให้ AI ช่วยที่สุด แต่ข้อมูลห้ามออกนอกเครื่อง ถึงเวลาเอามาทำจริง เพราะเราได้ติดตั้งและจูนทุกอย่างที่ต้องใช้ไว้ครบแล้วตั้งแต่ภาค 2

เหมาะกับใคร สถานการณ์ไหน

เหมาะสำหรับผู้ที่ต้องประมวลผลข้อมูลเอกสารที่มีรายละเอียดส่วนบุคคล ตัวเลขทางการเงิน หรือข้อมูลความลับทางธุรกิจ เช่น การสรุปสัญญา การร่างจดหมายเกี่ยวกับข้อมูลเงินเดือนพนักงาน การวิเคราะห์ความคิดเห็นของลูกค้าที่มีข้อมูลเบอร์โทรศัพท์ติดต่อปะปนอยู่ หรือการปรับปรุงแก้ไขเนื้อหาข้อความประชาสัมพันธ์ที่ยังไม่มีการประกาศอย่างเป็นทางการ งานประเภทนี้ไม่สามารถส่งต่อให้ AI บนระบบคลาวด์ภายนอกได้ แต่สามารถใช้โมเดลในเครื่องประมวลผลแทนได้อย่างปลอดภัย เนื่องจากเอกสารทางการของ Ollama ระบุชัดเจนว่าเมื่อเปิดรันแบบ local แล้ว ข้อมูลการแชทและไฟล์ทั้งหมดจะไม่ถูกส่งออกภายนอกระบบและไม่มีใครสามารถเข้าถึงได้

ของที่ต้องมีก่อน

ของ	จากบท
Ollama + โมเดลขนาดพอดีกับหน่วยความจำ	บท 4 · ตารางบท 2
โมเดลที่ปรับแต่ง system prompt แล้ว (แนะนำ)	บท 6
ความเข้าใจเรื่องแท็ก -cloud	บท 1 และ 5

ขั้นตอนทำจริง

หนึ่ง: จำกัดขอบเขตการเข้าถึงเครือข่ายภายนอก (ทางเลือกเพื่อความปลอดภัยขั้นสูง) Ollama มีโหมดจำกัดการเชื่อมต่อภายนอก (local only) เพื่อปิดกั้นการเรียกใช้งานโมเดลผ่านคลาวด์ทั้งหมด สามารถกำหนดค่านี้ได้ในไฟล์ `~/.ollama/server.json`

```
{
  "disable_ollama_cloud": true
}
```

รีสตาร์ท Ollama หนึ่งครั้ง เท่านี้ต่อให้เผลอพิมพ์แท็ก cloud ก็ใช้ไม่ได้ เหมาะกับเครื่องที่ตั้งใจใช้ทำงานลับโดยเฉพาะ (สิ่งที่ต้องแลกคือจะใช้โมเดล cloud และ web search ของ Ollama ไม่ได้เลย แต่ workflow นี้ต้องการแบบนั้นพอดี) ส่วนคนที่ใช้แอป Ollama บน desktop มีทางลัดกว่านั้น: สวิตช์ **Airplane mode** ในหน้า settings (เห็นได้ในภาพประกอบบท 6) ทำหน้าที่เดียวกันด้วยการกดครั้งเดียว

สอง: สร้างโมเดลปรับแต่งเฉพาะทางด้วย Modelfile ตามรายละเอียดบท 6 ตัวอย่างการกำหนดคุณสมบัติสำหรับงานสรุปสัญญา:

```
FROM qwen3.5:9b
PARAMETER temperature 0.3
SYSTEM """คุณคือโมเดลตรวจสอบเอกสารภาษาไทย ตอบเป็นภาษาไทยเสมอ
สรุปให้สั้น ชี้จุดเสี่ยงหรือตัวเลขผิดปกติเป็นข้อๆ
ห้ามแต่งข้อมูลที่ไม่มีในเอกสาร ถ้าไม่พบให้บอกว่าไม่พบ"""
```

สังเกตว่าเราตั้ง temperature ไว้ต่ำกว่าปกติ เพราะงานเอกสารต้องการคำตอบที่สม่ำเสมอ ไม่ได้ต้องการความคิดสร้างสรรค์

สาม: แบ่งช่วงเวลาประมวลผล เปิดใช้งานโมเดลเฉพาะทางที่สร้างขึ้นด้วยคำสั่ง `ollama run` จากนั้นคัดลอกเนื้อหาเอกสารส่งเข้าไปประมวลผล หากเอกสารมีความยาวเกินไป แนะนำให้แบ่งข้อความออกเป็นส่วนย่อยแทนการส่งข้อมูลทั้งหมดในรอบเดียวเพื่อป้องกันปัญหาข้อมูลล้นขีดจำกัดหน่วยความจำบทสนทนา (หากมีความจำเป็นต้องประมวลผลเอกสารยาวเป็นประจำ ควรปรับตั้งค่าขยายขนาด `num_ctx` ตามขั้นตอนในบท 6 ไว้ล่วงหน้า)

พรอมต์ตั้งต้น

```
สรุปสัญญาฉบับนี้เป็นภาษาไทยไม่เกิน 10 ข้อ
แยกเป็น: คู่สัญญา · มูลค่าและกำหนดจ่าย · ข้อผูกพันหลักของแต่ละฝ่าย ·
เงื่อนไขยกเลิก · จุดที่ควรให้ฝ่ายกฎหมายดูเพิ่ม

[วางเนื้อหาสัญญาตรงนี้]
```

เคล็ดลับและต่อยอด

- ลองทดสอบความเป็นส่วนตัวให้เห็นกับตาสักครั้ง โดยปิด Wi-Fi แล้วรัน workflow ทั้งหมด ถ้าทุกอย่างยังทำงานได้ แปลว่าข้อมูลไม่มีทางออกไปไหนจริงๆ
- คำตอบจากโมเดลเป็นเพียงร่าง ไม่ใช่คำตัดสิน โดยเฉพาะงานกฎหมายและการเงิน คนยังต้องตรวจทานรอบสุดท้ายเสมอ
- หากจำเป็นต้องทำงานประเภทนี้เป็นประจำทุกสัปดาห์ แนะนำให้เขียน Modelfile แยกสร้างโมเดลเฉพาะทางตามประเภทงาน (เช่น โมเดลสำหรับวิเคราะห์สัญญา · โมเดลสรุปความคิดเห็นลูกค้า · หรือโมเดลสำหรับประมวลผลทางการเงิน) เนื่องจากเราสามารถสร้างและตั้งชื่อโมเดลเฉพาะทางได้ไม่จำกัดจำนวน
- ถ้าอยากส่งไฟล์เข้าไปโดยตรงแทนการก๊อปวาง บทถัดไปมีคำตอบ

บทที่ 15

workflow คุยกับไฟล์

การคัดลอกเนื้อหามาวางทีละท่อนแบบบท 13 เหมาะกับงานชิ้นเดียวที่จบเร็ว แต่ถ้าต้องการถามอะไรก็ได้จากเอกสารทั้งกอง วิธีนี้ยังไม่พอ ควรอัปโหลดไฟล์เข้าคลังครั้งเดียว แล้วถามกี่รอบก็ได้ นี่คือฟีเจอร์เด่นของ Open WebUI ที่ติดตั้งไว้ตั้งแต่บท 7

เหมาะกับใคร และใช้ในสถานการณ์ไหน

วิธีนี้เหมาะกับคนที่มีเอกสารชุดใหญ่และต้องกลับมาหาคำตอบซ้ำๆ เช่น คู่มือพนักงาน ระเบียบบริษัท สเปคสินค้า รายงานย้อนหลัง หรืองานวิจัยที่สะสมไว้ คำถามอย่าง "ลาป่วยต้องแจ้งล่วงหน้ากี่วัน" ควรตอบได้ภายใน 10 วินาทีโดยไม่ต้องเปิดไฟล์หาเอง และเพราะทุกอย่างรันในเครื่อง จึงใช้วิธีนี้กับเอกสารภายในองค์กรได้ด้วย

ของที่ต้องมีก่อน

ของ	จากบท
Open WebUI ที่ต่อกับ Ollama แล้ว	บท 7
โมเดลหลักประจำเครื่อง	ตารางบท 2
วิธีเพิ่ม context	บท 6 · จำเป็นจริงในบทนี้

ขั้นตอนทำจริง

หนึ่ง ขยาย context ของโมเดลก่อน ซึ่งเป็นขั้นตอนสำคัญที่สุดในบทนี้ ระบบคุยกับไฟล์จะค้นหาเนื้อหาส่วนที่เกี่ยวข้องในเอกสาร แล้วแนบไปกับคำถามก่อนส่งให้โมเดล ถ้า context สั้น เนื้อหาที่ค้นได้จะถูกตัดทิ้งโดยไม่แจ้งให้ทราบ ผลคือโมเดลตอบเหมือนไม่เคยเห็นเอกสาร เอกสารของ Open WebUI เตือนผู้ใช้ Ollama เรื่องนี้ไว้ตรงๆ โดยแนะนำให้ตั้ง context length ของโมเดลที่ Admin Panel > Models > Advanced Parameters เป็น 8192 อย่างน้อย หรือถ้าเครื่องไหวก็ขยับให้มากกว่า 16000 ไปเลย (นี่คือเรื่อง num_ctx จากบท 6 ที่นำมาใช้ในอีกบริบทหนึ่ง)

สอง อัปโหลดไฟล์เข้าคลัง หน้า Workspace ของ Open WebUI มีพื้นที่สำหรับอัปโหลดเอกสาร รองรับไฟล์ที่มีข้อความหลายประเภท ตั้งแต่ PDF และสเปรดชีตไปจนถึงโค้ด อีกทั้งยังจัดกลุ่มตามโปรเจกต์หรือทีมได้ (ฟีเจอร์ Knowledge)

สาม ถามโดยอ้างอิงเอกสาร พิมพ์ `#` ในช่องแชท แล้วเลือกเอกสารหรือชุดเอกสารที่ต้องการ จากนั้นถามตามปกติ โมเดลจะค้นคำตอบจากเอกสารนั้น พร้อมระบุได้ว่าคำตอบมาจากตรงไหน ถ้าอยากให้โมเดลอ่านหน้าเว็บ ก็วางลิงก์เว็บหลัง `#` ได้เช่นกัน

พรอมต์ตั้งต้น

จากคู่มือพนักงานที่แนบ ช่วยตอบ:
พนักงานทดลองงานลากิจได้ไหม ถ้าได้ กี่วัน และต้องแจ้งใคร
ตอบพร้อมอ้างว่าอยู่ข้อไหนของเอกสาร ถ้าคู่มือไม่ได้ระบุ ให้บอกว่าไม่ระบุ

เคล็ดลับและต่อยอด

- ประโยคกันมั่วท้ายพรอมต์ ("ถ้าไม่ได้ระบุ ให้บอกว่าไม่ระบุ") สำคัญมากกับงานเอกสาร เพราะโมเดลเล็กมักเดาคำตอบเอง
- ถ้าคำตอบแปลกๆ ทั้งที่เอกสารมีข้อมูล ให้สงสัยเรื่อง context เป็นอันดับแรก แล้วกลับไปทำขั้นตอนที่หนึ่ง
- ถ้าเอกสารเปลี่ยนเวอร์ชัน ให้อัปโหลดไฟล์ใหม่แทนแล้วลบไฟล์เดิมออกจากคลัง โดยจัดการไฟล์ทั้งหมดได้ที่ Settings > Data Controls > Manage Files
- ถามภาษาไทยกับเอกสารภาษาอังกฤษได้ โมเดลตระกูลที่รองรับหลายภาษา (บท 3) จะแปลข้อความให้เองขณะตอบ

บทที่ 16

เวิร์กโฟลว์การใช้งาน AI ช่วยเขียนโค้ด

ในบรรดางานทั้งหมด โมเดลเปิดยุคปัจจุบันมีพัฒนาการในด้านการเขียนโค้ดที่รวดเร็วที่สุด และการเชื่อมต่อ API จากบท 7 ก็ช่วยเพิ่มประสิทธิภาพของกระบวนการนี้ได้อย่างเต็มที่: ด้วยการเชื่อมต่อเครื่องมือเขียนโค้ดเข้ากับ Ollama เพื่อใช้งาน AI ช่วยเขียนโค้ดภายในเครื่อง โดยไม่จำเป็นต้องส่งโค้ดคอมพิวเตอร์ออกนอกบริษัทและไม่มีค่าบริการรายเดือน

เหมาะกับใคร ใช้ในสถานการณ์ไหน

เหมาะสำหรับผู้เขียนโปรแกรมทุกระดับ ตั้งแต่มือใหม่ที่เริ่มต้นหัดเขียนสคริปต์ ไปจนถึงโปรแกรมเมอร์ที่ทำงานในองค์กรที่มีข้อกำหนดชัดเจนว่าห้ามส่งโค้ดขึ้นสู่ระบบคลาวด์ภายนอก ตลอดจนผู้ที่ต้องการหลีกเลี่ยงข้อจำกัดของบริการช่วยเขียนโค้ดแบบเสียเงินรายเดือน

ของที่ต้องมีก่อน

ของ	จากบท
Ollama ที่ทำงานอยู่	บท 4
โมเดลสายโค้ดขนาดที่เครื่องรองรับไหว	ตารางในบทนี้
เครื่องมือเขียนโค้ด	สองทางในบทนี้

โมเดลสายโค้ดที่แนะนำ เรียงตามขนาดเครื่องจากตารางในบท 2

ระดับสเปกเครื่อง	โมเดล	หมายเหตุ
โน้ตบุ๊กทั่วไป	`qwen2.5-coder` แท็กเล็ก (0.5b-7b)	โมเดลโค้ดรุ่นเล็กยอดนิยม มีแท็กหลายขนาดให้เลือก
เครื่องเกม 24GB	`qwen3.6:27b`	ตัวเด่นจากบท 10 ที่ Qwen เน้นความสามารถด้านโค้ดเป็นจุดขายหลัก
เครื่องเกม 24GB (ทางเลือก MoE)	`qwen3-coder:30b`	ขนาด 30B แต่ทำงานจริง 3.3B เทรนมาเพื่องาน software engineering โดยเฉพาะ · context 256K

ขั้นตอนทำจริง · สองทางให้เลือก

ทางที่หนึ่ง · ใช้ `ollama launch` ที่มีมาให้ Ollama มีระบบเชื่อมกับเครื่องมือเขียนโค้ดมาให้แล้ว รองรับทั้ง VS Code · Claude Code · Codex · OpenCode และเริ่มใช้งานได้ด้วยคำสั่งเดียว

```
ollama launch
```

ระบบจะแสดงเมนูให้เลือกเครื่องมือที่จะเชื่อม หรือจะระบุชื่อเครื่องมือกับโมเดลที่ต้องการในคำสั่งเลยก็ได้

```
ollama launch claude --model qwen3.6:27b
```

ทางที่สอง · Continue ปรับแต่งได้ละเอียดกว่า Continue คือส่วนขยายโอเพนซอร์สของ IDE (โปรแกรมเขียนโค้ด) และมีคู่มือเชื่อมต่อกับ Ollama อย่างเป็นทางการ ตามเอกสารของ Continue ให้โหลดแท็กโมเดลที่จะใช้ด้วย `ollama pull` ให้ครบก่อน จากนั้นตั้งค่าในไฟล์ config ของ Continue หรือใช้โหมด AUTODETECT เพื่อให้ระบบตรวจพบโมเดลทุกตัวในเครื่องเอง

พรอมต์ตั้งต้น

```
เขียนสคริปต์ Python อ่านไฟล์ Excel รายชื่อลูกค้า
แล้วแยกแถวที่อีเมลผิดรูปแบบออกมาเป็นไฟล์ใหม่
ใส่คอมเมนต์อธิบายเป็นภาษาไทยทีละส่วน
```

เคล็ดลับและต่อยอด

- กับดักอันดับหนึ่งตามเอกสารของ Continue คือใส่ชื่อโมเดลไว้ใน config แต่ลืม `pull` โมเดลลงเครื่อง จึงเจอ error `404 model not found` วิธีแก้คือ pull แท็กเดียวกับที่ระบุใน config
- งานเขียนโค้ดมักต้องใช้ context ยาว (ไฟล์โค้ดหลายไฟล์ + คำสั่ง) เอกสารของ Ollama เองแนะนำอย่างน้อย 64K สำหรับ coding tools วิธีจากบท 6 จึงใช้กับกรณีนี้ได้เลย
- โมเดลสายโค้ดขนาดเล็กมีความเหมาะสมกับการเติมโค้ดสั้น ๆ (Autocomplete) เนื่องจากประมวลผลได้รวดเร็ว ส่วนงานออกแบบโครงสร้างระบบขนาดใหญ่ ควรพิจารณาเลือกใช้โมเดลขนาดใหญ่ที่สุดเท่าที่หน่วยความจำของเครื่องจะรองรับได้ หรือเลือกใช้โมเดลบนคลาวด์หากงานนั้นมีความซับซ้อนสูงมากเกินขีดความสามารถของโมเดลแบบ local (ตามเกณฑ์การแบ่งประเภทงานที่ระบุในบท 1)

จบสาม workflow แล้ว เหลือด่านสุดท้ายของเล่มคือภาคผนวก ซึ่งรวมวิธีแก้ปัญหาที่พบบ่อย อภิธานศัพท์ และวิธีตัดสินใจเองเมื่อมีโมเดลรุ่นใหม่ออกหลังหนังสือเล่มนี้ตีพิมพ์

บทที่ 17

แก้อาการยอดฮิต อภิธานศัพท์ และหลักตัดสินใจด้วยตัวเอง

แก้อาการยอดฮิต

อาการ	สาเหตุที่พบบ่อย	ทางแก้
ตอบช้าผิดปกติ	memory เครื่องไม่พอ โมเดลจึงกระจายงานไปทั้ง CPU และ GPU	`ollama ps` ดูคอลัมน์ PROCESSOR แล้วลด context หรือเปลี่ยนเป็นแท็กที่เล็กลง (บท 6)
ตอบเหมือนไม่เห็นเอกสารที่ให้	context สั้นเกิน ส่วนเกินถูกตัดทิ้ง	เพิ่ม num_ctx (บท 6) · ใน Open WebUI เพิ่มที่ Advanced Parameters (บท 14)
ตอบมั่ว แต่งข้อมูลเอง	ธรรมชาติของโมเดล ยิ่งเล็กยิ่งกล้าเดา	เติมประโยคกันมั่วในพรอมต์ (บท 14) · ลด temperature (บท 6) · ขยับไปใช้โมเดลใหญ่ขึ้น
ภาษาไทยเพี้ยน สลับภาษา	โมเดลเล็กเกินไปจึงรองรับภาษาไทยได้ไม่ดี	เปลี่ยนไปใช้โมเดลที่ใหญ่ขึ้นตามตารางบท 2 · เลือกตระกูลที่ประกาศรองรับหลายภาษา (บท 3) · สั่ง "ตอบเป็นภาษาไทยเสมอ" ใน system prompt (บท 6)
มีการ์ดจอแต่ ps ขึ้น 100% CPU	driver เก่าหรือระบบไม่เห็นการ์ด	อัปเดต driver (NVIDIA ต้อง 550 ขึ้นไป) แล้วดูบันทึกการทำงานประกอบ
เครื่องอืดหลังเลิกคุย	โมเดลค้างใน memory ตามค่าเริ่มต้น 5 นาที	`ollama stop` ชื่อโมเดล (บท 5)
`404 model not found` จากเครื่องมือที่ต่อ Ollama	ระบุชื่อโมเดลไว้ในเครื่องมือ แต่ยังไม่ได้ดาวน์โหลดโมเดลนั้น	`ollama pull` แท็กนั้นให้ตรงเป๊ะ (บท 15)
Open WebUI ไม่มีโมเดลให้เลือก	Ollama ไม่ได้ทำงานอยู่	เปิด Ollama ก่อนแล้วรีเฟรช (บท 7)
terminal บน Windows แสดง	Windows 10 รุ่นเก่าหรือฟอนต์ไม่รองรับ	อัปเดตเป็น Windows 10 22H2 ขึ้นไป หรือเปลี่ยนฟอนต์ terminal

อาการ	สาเหตุที่พบบ่อย	ทางแก้
อักขระประหลาด		
ดิสก์เต็มไม่รู้เพราะอะไร	โมเดลสะสมจากการลองหลายตัว	`ollama ls` ดูขนาดรวม แล้ว `ollama rm` ตัวที่เลิกใช้ (บท 5)

ถ้าเจอปัญหาที่ไม่มีในตารางนี้ ให้ดูบันทึกการทำงานของ Ollama ตามระบบปฏิบัติการที่ใช้: macOS ให้รัน `cat ~/.ollama/logs/server.log` , Linux ให้รัน `journalctl -e -u ollama` ส่วน Windows ให้กด `Win+R` พิมพ์ `explorer %LOCALAPPDATA%\Ollama` แล้วเปิด `server.log` คู่มือฉบับเต็มดูได้ที่ docs.ollama.com/troubleshooting

อภิธานศัพท์ฉบับย่อ · เรียงตามลำดับการใช้งาน

คำ	ความหมายสั้นๆ	บทหลัก
local LLM	AI ที่รันในเครื่องตัวเอง	1
โมเดล	AI หนึ่งตัว เก็บเป็นไฟล์ในเครื่อง	1
open weights	โมเดลเปิดที่ใครก็โหลดไฟล์ไปใช้ได้	1
B	พันล้านพารามิเตอร์ หน่วยขนาดโมเดล	2
quantization	การย่อโมเดลด้วยการลดความละเอียดของตัวเลขที่ใช้เก็บค่า	2
GGUF	ฟอร์แมตไฟล์โมเดลที่ใช้กับ llama.cpp	2
RAM / VRAM / unified memory	memory สามแบบสำหรับเครื่องแต่ละประเภท	2
context	ความจำระยะสั้นของบทสนทนา กิน memory	2 · 6
โทเคน	หน่วยคำย่อยที่โมเดลอ่านและตอบ	2
MoE	โมเดลที่เลือกใช้เพียงบางส่วนในแต่ละครั้ง จึงทำงานเร็วเมื่อเทียบกับขนาดรวม	3 · 12
thinking / reasoning	โมเดลที่คิดในใจก่อนตอบ	3
แท็ก	รุ่นย่อยของโมเดล ระบุหลังเครื่องหมาย `:`	5

คำ	ความหมายสั้นๆ	บทหลัก
-cloud	แท็กที่ประมวลผลบนเซิร์ฟเวอร์ ไม่ใช่ local	1 · 5
-mlx	แท็กรุ่นสำหรับชิป Apple	11
system prompt	คำสั่งประจำตัวที่โมเดลยึดทุกบทสนทนา	6
Modelfile	พิมพ์เขียวสร้างโมเดลฉบับปรับแต่งเอง	6
API	ช่องทางให้โปรแกรมอื่นคุยกับ Ollama	7
Open WebUI	หน้าแชทในเครื่องที่ต่อกับ Ollama	7
distill	ถ่ายทอดความสามารถจากโมเดลใหญ่ไปยังโมเดลเล็ก	3 · 12
1-bit / ternary	วิธีบีบอัดโมเดลแบบสุดขั้ว ตัวแทนคือ Bonsai กับ BitNet	8

ตารางโมเดลแนะนำ ณ กรกฎาคม 2026

สรุปเนื้อหาจากทั้งเล่มไว้ในตารางเดียว โดยอ้างอิงขนาดไฟล์จาก ollama.com ณ เวลาเขียน

ระดับสเปกเครื่อง	ตัวหลัก	ทางเลือก
มือถือ	Gemma 4 E2B/E4B (ผ่านแอป บท 8)	Qwen3.5 0.8B/2B · Phi-4 mini
โน้ตบุ๊ก 8-16GB	`gemma4:12b` (7.6GB)	`qwen3.5:4b` (3.4GB) · `qwen3.5:9b` (6.6GB)
เครื่องเกม 8-24GB	`qwen3.6:27b` (17GB)	`gemma4:26b` (18GB) · `gpt-oss:20b` (14GB)
Mac	แท็ก `-mlx` ของตัวเดียวกับเครื่องเกม	`gpt-oss:120b` (65GB) เมื่อ memory 96GB ขึ้นไป
สายโค้ด	`qwen3.6:27b`	`qwen3-coder:30b` · `qwen2.5-coder` แท็กเล็ก

หลักตัดสินใจด้วยตัวเอง · เมื่อมีโมเดลใหม่ออกหลังหนังสือเล่มนี้ตีพิมพ์

ข้อมูลในตารางข้างบนจะล้าสมัยลงทุกเดือน แต่วิธีคิดยังใช้ได้ ถ้าเห็นข่าวโมเดลใหม่ ให้เปิดหน้าข้อมูลของโมเดลนั้นบน ollama.com/library แล้วถามสี่ข้อ

1. **ขนาดไฟล์ของแท็กที่เล็งไว้เหมาะกับเครื่องไหม** ขนาดไฟล์และพื้นที่ที่ต้องเผื่อไว้รวมกันต้องไม่เกิน memory ของเครื่อง (บท 2)
2. **context ยาวพอกับงานไหม** งานเอกสารและโค้ดต้องการมากกว่างานแชท (บท 6)
3. **รับข้อมูลแบบที่ต้องใช้ไหม** Text อย่างเดียว หรือดูรูปได้ด้วย
4. **อัปเดตล่าสุดเมื่อไหร่ และเป็นแท็ก local จริงหรือ cloud** วันที่อัปเดตบอกได้ว่ารุ่นนั้นใหม่แค่ไหน ส่วนการแยกว่าเป็น local หรือ cloud คือประเด็นที่เล่มนี้ย้ำมาตลอด

เมื่อพิจารณาครบทั้งสี่ประเด็นแล้ว ให้ทดลองใช้งานจริงด้วยคำสั่ง `ollama pull` เพื่อติดตั้งและทดลองส่งพรอมต์ภาษาไทยประมวลผลดูสัก 10 นาที หากผลลัพธ์ไม่ตรงตามความต้องการก็สามารถลบไฟล์โมเดลได้ทันทีด้วยคำสั่ง `ollama rm` เนื้อหาสำคัญทั้งหมดของหนังสือเล่มนี้สามารถสรุปได้เป็นขั้นตอนง่าย ๆ คือ: อ่านสเปกเครื่องและโมเดลให้เข้าใจ ประเมินขีดความสามารถการรองรับของฮาร์ดแวร์ ทดลองสั่งงาน และจัดการลบไฟล์โมเดลอย่างถูกวิธี

ที่มาของข้อมูลทั้งเล่ม

- เอกสารทางการ Ollama: docs.ollama.com (quickstart · CLI · FAQ · GPU · Modelfile · context length · troubleshooting · คู่มือราย OS) และ ollama.com/library (ขนาดไฟล์และแท็กทุกตัวเลข)
- เอกสารทางการ Open WebUI: docs.openwebui.com (quick start · RAG · Knowledge)
- หน้าโมเดลและประกาศทางการของค่าย: Google (blog Gemma 4 · Gemma 4 12B) · OpenAI (gpt-oss) · Qwen (Qwen3.5 · Qwen3.6) · DeepSeek (R1) · Microsoft (Phi-4 mini · BitNet) · Prism ML (Bonsai 27B) บน Hugging Face
- แอปมือถือ: GitHub ของ PocketPal AI และ Google AI Edge Gallery
- เครื่องมือเขียนโค้ด: docs.continue.dev

ทุกบทเขียนจากแหล่งข้างบนเท่านั้น และตัวเลขทุกตัวตรวจสอบกับต้นทางได้ ข้อมูลชุดนี้เป็นข้อมูล ณ กรกฎาคม 2026 รายชื่อโมเดลกับตัวเลขเปลี่ยนเร็ว แต่วิธีคิดยังใช้ได้อีกนาน