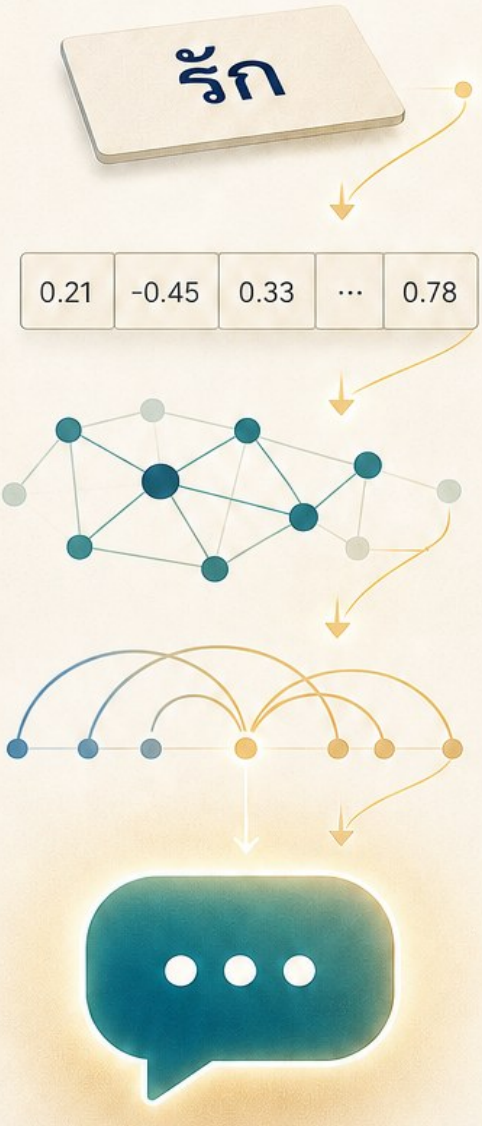
ก่อนจะมาเป็น ChatGPT

ประวัติย่อของ AI ทางด้านภาษา

ตั้งแต่คอมพิวเตอร์อ่านคำไม่เข้าใจ
จนถึงวันที่เราคุยกับมันได้ผ่าน **LLM**



1 คำ
จุดเริ่มต้นของภาษา

2 แปลงเป็นตัวเลข
แทนความหมายของคำ

3 เข้าใจความสัมพันธ์
เชื่อมโยงความหมาย

4 ใส่ใจบริบท
เลือกโฟกัสสิ่งที่เกี่ยวข้อง

5 คุยกับเราได้
เข้าใจ สร้าง และตอบสนอง
ผ่าน **LLM**

AI&IT Pulse

จัดทำโดยเพจ

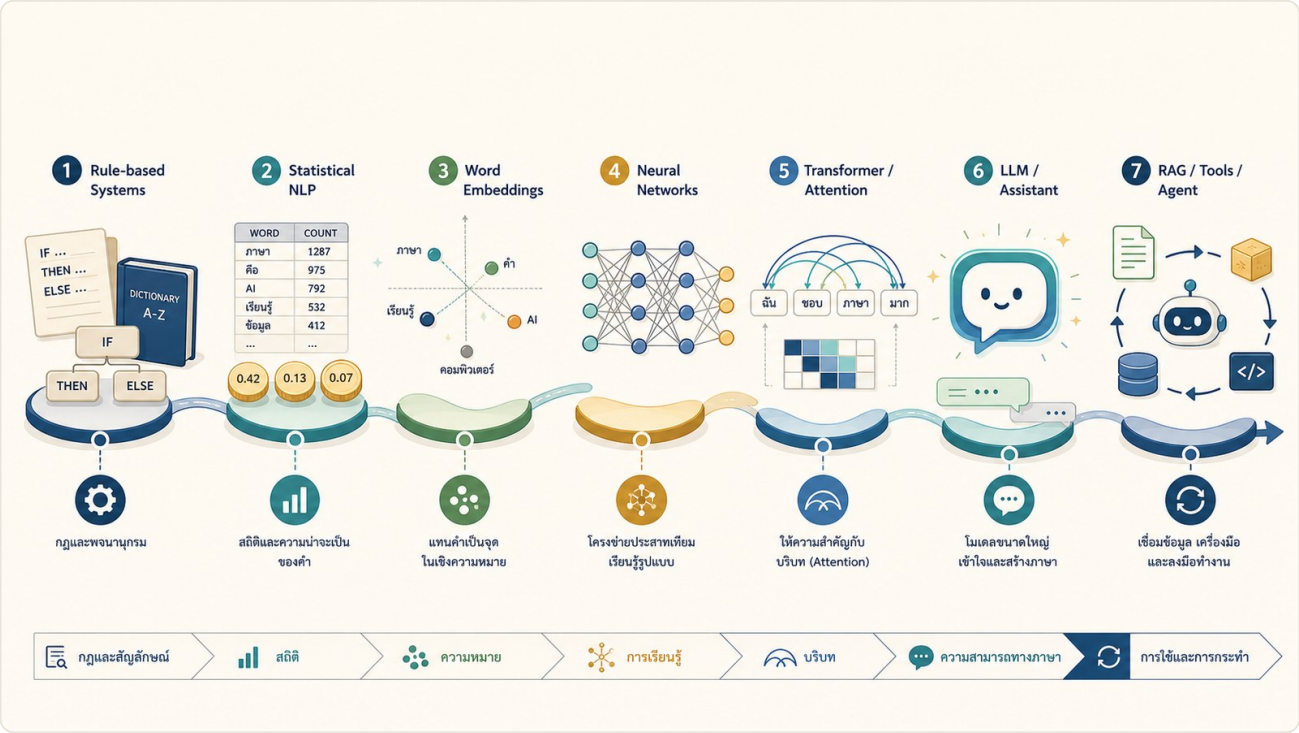
แจกฟรี ห้ามจำหน่าย

ก่อนจะเริ่มอ่าน

หนังสือเล่มนี้เล่าเรื่องเดียว — เรื่องที่ว่า ChatGPT และ AI ภาษาที่เราใช้กันทุกวันนี้ "เกิดขึ้นมาได้ยังไง"

ไม่ใช่ตำราเรียน ไม่มีโค้ด ไม่มีสูตรคณิตศาสตร์ และไม่ต้องมีพื้นฐานสายคอมแม้แต่นิดเดียว ขอแค่คุณเคยคุยกับ AI chatbot สักตัวแล้วเคยสงสัยว่า "ไอ้สิ่งนี้มันคิดยังไง" เท่านั้นก็พอ

เราจะเดินทางไปด้วยกันทีละก้าว ตั้งแต่ยุคที่มนุษย์พยายามเขียนกฎให้เครื่องเข้าใจภาษา ผ่านยุคสถิติ ยุคที่คำกลายเป็นตัวเลข ยุคโครงข่ายประสาทเทียม จนถึง Transformer และ LLM ที่อยู่เบื้องหลัง ChatGPT — และก้าวต่อไปอย่าง RAG, tools และ AI Agent



ภาพรวมการเดินทางทั้งเล่ม: จากยุคเขียนกฎ สู่ LLM และ Agent

ตลอดเล่ม ให้ถือคำถามหนึ่งติดมือไว้เสมอ:

> **"มันเข้าใจจริง หรือแค่เดาคำให้ฟังดูดี?"**

อ่านจบแล้ว คุณจะต่อจิ๊กซอว์ภาพใหญ่ได้เอง

สารบัญ

บทนำ

สำหรับคนส่วนใหญ่ ChatGPT โผล่เข้ามาในชีวิตแบบไม่มีปี่มีขลุ่ย อยู่ดี ๆ ก็มีกล่องข้อความที่พิมพ์คุยด้วยได้เหมือนคุยกับคน ตอบได้แทบทุกเรื่อง และฉลาดจนน่าขนลุก หลายคนเลยเข้าใจว่ามันคือสิ่งประดิษฐ์ที่เพิ่งถือกำเนิดขึ้นเมื่อปลายปี 2022 — ทั้งที่จริงแล้ว มันคือ *ปลายทาง* ของการเดินทางที่ยาวนานหลายสิบปี หนังสือเล่มนี้จะพาคุณย้อนกลับไปเดินบนถนนเส้นนั้นตั้งแต่ต้น

ถ้ามองให้ดี ตลอดประวัติศาสตร์ของ AI ภาษา มีคำถามเดียวที่นักวิจัยถามซ้ำมาตลอด นั่นคือ "จะทำยังไงให้เครื่องจัดการกับภาษามนุษย์ได้" สิ่งที่เปลี่ยนไปในแต่ละยุคคือ *คำตอบ* ของคำถามนี้ และคำตอบนั้นค่อย ๆ ขยับผ่านสามแนวคิดใหญ่ — จากยุคที่เราพยายาม **เขียนกฎ** ให้เครื่องทำตามทุกกรณี มาสู่ยุคที่ปล่อยให้เครื่อง **นับสถิติ** หารูปแบบจากข้อมูลเอง จนถึงยุคที่เครื่อง **เรียนรู้จากตัวอย่าง** ด้วยโครงข่ายประสาทเทียม แต่ละบทในเล่มนี้คือก้าวหนึ่งบนเส้นทางของการเปลี่ยนผ่านนั้น

เพื่อไม่ให้หลงทาง ลองมองภาพรวมของการเดินทางทั้งเล่มเป็นช่วง ๆ ดังนี้ สองบทแรกปูพื้นว่าทำไมภาษาถึงยากสำหรับเครื่อง และทำไม ChatGPT ถึง "รู้สึก" ต่างจาก AI ที่เราเคยเจอ บทที่ 3 และ 4 พาไปดูยุคของกฎและยุคของสถิติ ที่มนุษย์ค่อย ๆ ถอยออกจากการบอกเครื่องทุกอย่าง บทที่ 5 และ 6 เล่าเรื่องวันที่คำกลายเป็นตัวเลข และเครื่องเริ่มเรียนรู้เองจากตัวอย่างผ่านโครงข่ายประสาทเทียม บทที่ 7 และ 8 คือหัวใจของเรื่อง — Transformer กับพลังของ attention และวันที่โมเดลภาษากลายมาเป็นผู้ช่วยที่คุยกับเราได้จริง และบทที่ 9 ปิดท้ายด้วยคำถามว่าโลกหลัง LLM กำลังเดินไปทางไหน ผ่าน RAG, tools และ AI Agent

หนังสือเล่มนี้ไม่มีโค้ด ไม่มีสูตรคณิตศาสตร์ และไม่ต้องการพื้นฐานสายคอมแม้แต่น้อย ขอเพียงคุณถือคำถามหนึ่งติดมือไว้ตลอดการเดินทาง — *"มันเข้าใจจริง หรือแค่เดาคำให้ฟังดูดี?"* เมื่ออ่านจบ คุณจะไม่เพียงรู้ว่า ChatGPT ทำงานยังไง แต่จะมองเห็นภาพใหญ่ว่ามนุษย์ใช้เวลาและความพยายามมากแค่ไหน กว่าจะสอนให้เครื่องคุยกับเราได้อย่างทุกวันนี้

บทที่ 1 — ทำไมจู่ ๆ AI ถึงพูดเหมือนคน

ลองนึกภาพว่าคุณเปิด ChatGPT ขึ้นมาเป็นครั้งแรก ยังไม่แน่ใจด้วยซ้ำว่าจะพิมพ์อะไรดี คุณเลยลองพิมพ์อะไรที่ยากนิดหน่อย เช่น "อธิบายเรื่องดอกเบี้ยทบต้นให้ฟังเหมือนฉันอายุสิบขวบ"

แล้วมันก็ตอบ

ไม่ใช่ตอบกาว ๆ ไม่ใช่โยนนิยามในตำรามาให้ แต่ตอบแบบที่เข้าใจง่าย เป็นธรรมชาติ มีตัวอย่างน่ารัก ๆ ราวกับมีคนใจดีนั่งอยู่ตรงข้ามแล้วค่อย ๆ เล่าให้ฟัง คุณอ่านจบแล้วนั่งนิ่งไปแวบหนึ่ง รู้สึกทึ่ง ปนกับความรู้สึกแปลก ๆ ที่บอกไม่ถูก — มันตอบ *เหมือนคน* เกินไปหน่อย

ถ้าคุณเคยรู้สึกแบบนี้ คุณไม่ได้รู้สึกอยู่คนเดียว คนทั้งโลกรู้สึกแบบเดียวกันในช่วงปลายปี 2022 และความรู้สึกนั้นแหละคือจุดเริ่มต้นของหนังสือเล่มนี้

หนังสือเล่มนี้จะพาคุณไปหาคำตอบของคำถามเดียวที่ซ่อนอยู่ในความรู้สึก "ขนลุก" นั้น — สิ่งที่กำลังคุยกับคุณอยู่ มัน **เข้าใจจริง หรือแค่ทำนายคำเก่งมาก** กันแน่

วันที่โลกทั้งโลกพร้อมใจกันทึ่ง

ChatGPT เปิดตัวเมื่อวันที่ 30 พฤศจิกายน 2022 ในฐานะ "research preview" — พูดง่าย ๆ คือเป็นตัวทดลองที่เปิดให้คนทั่วไปเข้ามาลองใช้ฟรี ๆ ไม่มีงานเปิดตัวอลังการ ไม่มีโฆษณาทางทีวี มีแค่บล็อกโพสต์สั้น ๆ จากบริษัทที่ชื่อ OpenAI

แต่สิ่งที่เกิดขึ้นหลังจากนั้นไม่สั้นเลย

ภายในห้าวัน มีคนสมัครใช้ครบหนึ่งล้านคน ซีอีโอของ OpenAI ทวีตยืนยันตัวเลขนี้ด้วยตัวเอง และมันก็ยังไม่หยุดแค่นั้น ตามการประมาณการของบริษัทวิเคราะห์ UBS ภายในสองเดือน ChatGPT น่าจะมีผู้ใช้งานราว 100 ล้านคนต่อเดือน

เพื่อให้เห็นภาพว่าตัวเลขนี้ผิดปกติแค่ไหน UBS เปรียบเทียบว่า Instagram ใช้เวลาราวสองปีครึ่ง และ TikTok ใช้เวลาราวเก้าเดือน กว่าจะไปถึงหลัก 100 ล้านผู้ใช้เดียวกัน นักวิเคราะห์ถึงกับบอกว่าในรอบยี่สิบปีที่ติดตามวงการอินเทอร์เน็ตมา ไม่เคยเห็นแอปไหนโตเร็วขนาดนี้

(พูดให้แฟร์ ตัวเลข 100 ล้านนี้เป็นการ *ประมาณการ* ไม่ใช่ตัวเลขทางการจาก OpenAI และสถิติ "โตเร็วที่สุด" นี้ก็ถูกแอป Threads ของ Instagram แซงไปในกลางปี 2023 — แต่ไม่ว่าจะวัดยังไง ความรู้สึก "นี่มันไม่ปกติ" ก็เป็นของจริง)

คำถามที่น่าสนใจกว่าตัวเลขก็คือ — *ทำไม* ผู้คนถึงตื่นเต้นขนาดนั้น

ทำไมมันถึง "รู้สึก" ต่างจาก AI ที่เราเคยเจอ

ก่อนปลายปี 2022 พวกเราส่วนใหญ่ก็เคยเจอ "AI" มาแล้วทั้งนั้น แต่เป็น AI คนละแบบ

ลองนึกถึงครั้งสุดท้ายที่คุณโทรเข้าสายด่วนของธนาคารหรือค่ายมือถือ แล้วเจอระบบตอบรับอัตโนมัติ คุณพยายามอธิบายปัญหาของคุณ แต่มันได้แต่พูดเมนูเดิมวนไป "กรุณากด 1 สำหรับยอดเงินคงเหลือ กด 2 สำหรับ..." คุณลองพูดปัญหาใหม่อีกรอบ มันก็พ่นเมนูเดิมซ้ำอีกที สุดท้ายคุณยอมแพ้แล้วกด 0 เพื่อขอคุยกับคนจริง ๆ

ความหงุดหงิดนั้นมีต้นเหตุที่ชัดเจน ระบบแบบเก่าทำงานจาก "สคริปต์" ที่เขียนไว้ล่วงหน้า ถ้าคำถามของคุณตรงกับสคริปต์ มันก็ตอบได้ แต่ถ้าคุณถามนอกกรอบแม้แต่นิดเดียว มันก็จนปัญญา เพราะไม่มีใครเคยเขียนคำตอบสำหรับสิ่งที่คุณเพิ่งพูดเอาไว้

นี่คือ "AI ก่อน ChatGPT" ที่คนส่วนใหญ่คุ้นเคย — มีประโยชน์ในกรอบแคบ ๆ แต่พอเจอภาษาจริงของมนุษย์ที่ไหลไปได้ทุกทิศทาง มันก็พังทันที

ChatGPT รู้สึกต่างออกไปเพราะมันไม่มีกำแพงแบบนั้นให้ชน คุณถามอะไรนอกกรอบแค่ไหนมันก็มีอะไรตอบ มันจำได้ว่าเมื่อกี้คุณคุยอะไรไป ปรับวิธีอธิบายตามที่คุณขอ (เล่าแบบเด็กสิบขวบก็ได้ เล่าแบบนักวิชาการก็ได้) และที่สำคัญ มันไม่เคยพูดประโยค "ขออภัย ไม่เข้าใจคำถาม" วนซ้ำจนคุณอยากปาโทรศัพท์ทิ้ง

> **จำง่าย ๆ ว่า:** AI แบบเก่าตอบได้เฉพาะสิ่งที่มีคน "เขียนคำตอบไว้ให้ล่วงหน้า" ส่วน ChatGPT เหมือนสร้างคำตอบขึ้นมาใหม่สด ๆ ทุกครั้ง — นั่นแหละคือสิ่งที่ทำให้มันรู้สึกเหมือนกำลังคุยกับ "ใครสักคน" ไม่ใช่กับเมนู

"พูดเหมือนคน" กับ "เข้าใจเหมือนคน" ไม่ใช่เรื่องเดียวกัน

ตรงนี้แหละคือจุดที่น่าสนใจที่สุด และเป็นหัวใจของทั้งเล่ม

การที่ ChatGPT ตอบได้ลื่นไหลเหมือนคน ทำให้เรารู้สึกโดยอัตโนมัติว่ามัน "เข้าใจ" สิ่งที่เราพูด แต่ลองคิดเล่น ๆ ดี ๆ ว่า — มันสร้างภาษาที่ *ฟังดูเหมือนเข้าใจ* ได้ แต่นั่นจำเป็นต้องแปลว่ามันเข้าใจในแบบที่เราเข้าใจจริง ๆ หรือเปล่า

คุณอาจคิดว่านี่เป็นคำถามของคนที่ไม่รู้เรื่องเทคโนโลยี แต่จริง ๆ แล้วตรงกันข้าม มันคือคำถามที่แม้แต่คนสร้าง AI เองก็ยังเถียงกันไม่จบ

มีงานวิจัยชิ้นหนึ่งที่ตีพิมพ์ในวารสาร PNAS ปี 2023 รายงานผลสำรวจนักวิจัยด้านภาษาและ AI จำนวน 480 คน ในคำถามว่าโมเดลภาษาที่ฝึกจากข้อความล้วน ๆ สามารถ "เข้าใจ" ภาษาได้ในระดับที่มีความหมายหรือไม่ ผลออกมาแบ่งเกือบครึ่ง — 51% บอกว่าได้ ส่วนอีก 49% บอกว่าไม่ได้

ลองคิดดูว่ามันแปลว่าอะไร แม้แต่ในหมู่ผู้เชี่ยวชาญ คำถามนี้ก็ยังไม่มีคำตอบที่ทุกคนเห็นตรงกัน ฉะนั้นถ้าคุณก็สงสัยเรื่องนี้อยู่ คุณกำลังถามคำถามเดียวกับที่นักวิทยาศาสตร์ระดับโลกถามกันอยู่

มาดูสองมุมที่กำลังเถียงกัน

มุมที่หนึ่ง: "มันแค่ทำนายคำเก่ง ๆ เท่านั้นเอง" นักวิจัยกลุ่มหนึ่งในปี 2021 ตั้งฉายาให้โมเดลภาษาว่าเป็น "นกแก้วเชิงสถิติ" (stochastic parrots) — ระบบที่ต่อคำเป็นประโยคจากรูปแบบที่เคยเห็นมา ตามความน่าจะเป็นว่าคำไหนมักตามคำไหน "โดยไม่ได้อ้างอิงถึงความหมาย" เลย มุมนี้เตือนว่าเรามักหลงคิดไปเองว่าเครื่องเข้าใจ เพียงเพราะมันพูดเก่ง — แนวโน้มแบบนี้มีมานานแล้ว ตั้งแต่ยุคแชตบอตตัวแรก ๆ ในทศวรรษ 1960 ที่คนเริ่มเทใจให้โปรแกรมง่าย ๆ ราวกับมันเข้าใจจริง (เรื่องนี้น่าทึ่งมากจนเราจะเก็บไว้เล่าเต็ม ๆ ในบทถัด ๆ ไป)

มุมที่สอง: "มันมีอะไรมากกว่าการเดาคำธรรมดา" อีกฝ่ายแย้งว่า ใช่ โมเดลถูกฝึกให้ "ทำนายคำถัดไป" ก็จริง แต่การจะทำนายคำถัดไปได้ ดีจริง ๆ ในข้อความสารพัดแบบ มันบังคับให้โมเดลต้องเรียนรู้รูปแบบที่ซับซ้อนกว่าที่ใครคาดไว้มาก — ทั้งไวยากรณ์ ข้อเท็จจริง และร่องรอยของการใช้เหตุผล ผลลัพธ์ที่ได้จึงเกินกว่าจะเรียกว่า "แค่ระบบเดาคำในมือถือ" ได้สนิทใจ

แล้วใครถูก? หนังสือเล่มนี้จะไม่รีบตอบให้คุณในบทแรก เพราะการตั้งคำถามให้ถูกข้อ สำคัญกว่าการรีบหาคำตอบสำเร็จรูป สิ่งที่อยากให้คุณ "ติดไม้ติดมือ" ออกจากบทนี้ไป คือคำถามนั้น ไม่ใช่คำตอบ

> **ลองนึกภาพว่า** วันหนึ่งมีมนุษย์ต่างดาวโผล่มา แล้วพูดภาษาไทยได้คล่องปร๋อตั้งแต่ประโยคแรก เราจะสรุปได้ทันทีเลยไหมว่ามัน "เข้าใจ" โลกแบบเดียวกับเรา หรือมันแค่เลียนเสียงเราได้เนียนจนน่าขนลุก? นักวิจัยบางคนใช้ภาพมนุษย์ต่างดาวแบบนี้แหละอธิบายความรู้สึกที่มีต่อ AI ภาษา

เครื่องมือใช้เป็น ก่อนจะเข้าใจกลไก

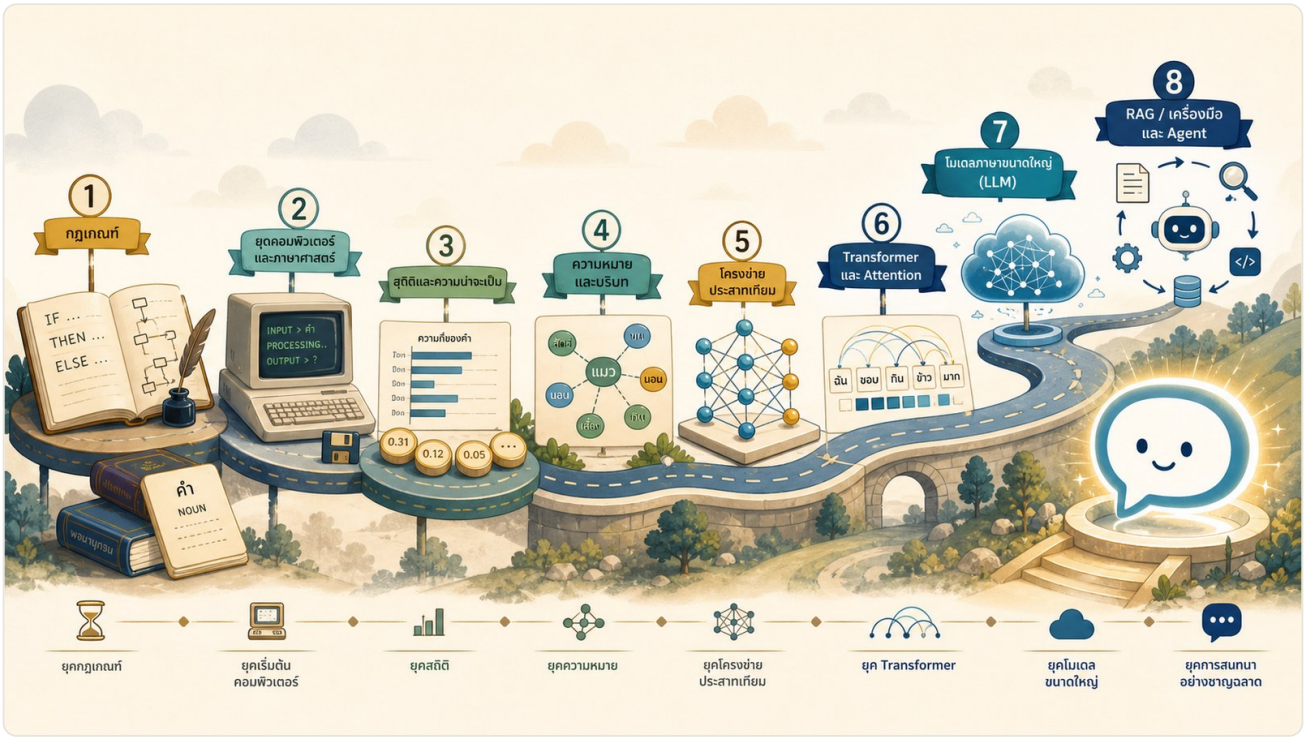
ระหว่างที่เรายังไม่รู้คำตอบสุดท้าย มีนิสัยหนึ่งที่ช่วยให้คุณใช้ AI ได้ฉลาดขึ้นทันที — ทุกครั้งที่ ChatGPT ตอบอะไรมา ให้ถามตัวเองในใจสั้น ๆ ว่า

> *"นี่คือสิ่งที่มันรู้จริง หรือมันกำลังเดาให้ฟังดูดี?"*

คำถามเดียวนี้เปลี่ยนวิธีที่คุณใช้ AI ไปเลย เพราะมันคอยเตือนว่าคำตอบที่ฟังดูมั่นใจ ไม่ได้แปลว่าถูกเสมอไป โดยเฉพาะเรื่องตัวเลข ชื่อคน วันที่ หรือข้อเท็จจริงที่ตรวจสอบได้ ของพวกนี้ควรเช็กซ้ำเสมอ ส่วนงานที่ AI เก่งจริง เช่น ช่วยร่าง ช่วยเรียบเรียง ช่วยอธิบายแนวคิดให้เข้าใจง่าย คุณก็ปล่อยให้มันช่วยได้เต็มที่

ตลอดทั้งเล่ม คุณจะค่อย ๆ เข้าใจว่า *ทำไม* มันถึง "เดาให้ฟังดูดี" ได้เก่งขนาดนั้น และนั่นจะทำให้คำถามข้างบนของคุณคมขึ้นเรื่อย ๆ

ChatGPT ไม่ได้โผล่มาจากที่ว่างเปล่า



แผนที่การเดินทางจาก rule-based AI สู่ ChatGPT

เรื่องที่คนมักเข้าใจผิดที่สุดเกี่ยวกับ ChatGPT คือคิดว่ามันเป็นสิ่งประดิษฐ์ชิ้นเดียวที่จู่ ๆ ก็เกิดขึ้นในปี 2022

ความจริงคือ มันเป็น *ปลายทาง* ของการเดินทางที่ยาวกว่าหกสิบปี เป็นการต่อยอดความคิดทีละชั้น ๆ จากนักวิจัยหลายรุ่น โดยที่แต่ละก้าวก็เกิดขึ้นเพราะก้าวก่อนหน้า "ยังไม่พอ"

ก่อนหน้านั้นไม่นาน ในปี 2020 OpenAI เคยปล่อยโมเดลรุ่นพี่ชื่อ GPT-3 ที่ทำให้นักพัฒนากลุ่มเล็ก ๆ ตื่นเต้นกันมาก มีคนบอกว่าได้ลองเล่นแล้วรู้สึก "เหมือนเห็นอนาคต" ขณะที่นิตยสาร MIT Technology Review พาดหัวบทความเกี่ยวกับมันไว้อย่างเจ็บแสบว่า *"ดีอย่างน่าตกใจ — และไม่มีสติปัญญาเอาเสียเลย"*

สังเกตว่าพาดหัวนั้นก็ถามคำถามเดียวกับเราเป๊ะ ดีจนน่าตกใจ แต่เข้าใจจริงไหม? น่าสนใจตรงที่ GPT-3 ในตอนนั้นยังต้องเข้าถึงผ่านช่องทางสำหรับนักพัฒนา คนทั่วไปอย่างเรา ๆ ยังเล่นไม่ได้ สิ่งที่ ChatGPT ทำในปี 2022 จึงไม่ใช่การฉลาดขึ้นแบบก้าวกระโดด แต่คือการ "เปิดประตู" ให้ทุกคนเดินเข้ามาคุยกับมันได้ง่าย ๆ ผ่านหน้าจอแชต — และนั่นก็เพียงพอจะเปลี่ยนโลก

แล้วก่อนหน้า GPT-3 ล่ะ มีอะไร? นั่นแหละคือแผนที่การเดินทางของเราในเล่มนี้

นี่คือเส้นทางคร่าว ๆ ที่เราจะเดินไปด้วยกัน แต่ละช่วงคือคำตอบของคำถามว่า "แล้วทำไมแค่นั้นมันถึงยังไม่พอ":

- **ยุคเขียนกฎ** — มนุษย์พยายามสอนภาษาให้เครื่องด้วยการเขียนกฎทีละข้อ แต่ภาษาจริงมีข้อยกเว้นมากเกินจะเขียนครบ

- **ยุคสถิติ** — เปลี่ยนวิธีคิดใหม่ ให้เครื่องดูตัวอย่างเยอะ ๆ แล้วจับรูปแบบเอง นี่คือต้นกำเนิดของไอเดีย "ทำนายคำถัดไป"
- **เมื่อคำกลายเป็นตัวเลข** — เครื่องเริ่มจับได้ว่าคำไหนความหมายใกล้กัน ผ่านการแปลงคำเป็นพิกัดบน "แผนที่ความหมาย"
- **โครงข่ายประสาทเทียม** — ระบบที่เรียนรู้จากตัวอย่างด้วยการลองผิดลองถูกแล้วปรับตัวเอง
- **Transformer (ปี 2017)** — จุดเปลี่ยนครั้งใหญ่ที่ทำให้เครื่องมองบริบทได้กว้างขึ้นมาก และเป็นฐานของ AI ภาษาทุกตัวในวันนี้
- **จากโมเดลดิบ สู่ผู้ช่วย** — ทำไม AI ที่อ่านข้อความมาทั้งโลกถึงยังต้องผ่านการ "ฝึกมารยาท" ก่อนจะกลายเป็น ChatGPT
- **ก้าวต่อไป** — RAG, เครื่องมือ และ AI Agent ที่กำลังพา AI ออกไปไกลกว่าการแค่ตอบแชต

ไม่ต้องจำศัพท์พวกนี้ตอนนี้เลย เราจะแกะมันทีละคำ ทีละบท แบบที่ไม่ต้องมีพื้นฐานอะไรมาก่อน ขอแค่คุณถือคำถามหลักติดมือไปด้วย แล้วทุกบทจะค่อย ๆ เติมชิ้นส่วนของคำตอบให้คุณเอง

ศัพท์ที่เพิ่งเจอ: - **LLM (Large Language Model / โมเดลภาษาขนาดใหญ่)** — สมองเบื้องหลัง ChatGPT คือระบบที่เรียนภาษาจากการอ่านข้อความจำนวนมหาศาล แล้วใช้สิ่งที่เรียนมาสร้างประโยคตอบเรา (เราจะค่อย ๆ เจาะว่ามันทำงานยังไงตลอดทั้งเล่ม) - **research preview** — เวอร์ชันทดลองที่บริษัทเปิดให้คนทั่วไปใช้ฟรีเพื่อเก็บข้อมูลและปรับปรุง ไม่ใช่ผลิตภัณฑ์ที่เสร็จสมบูรณ์

สรุปง่าย ๆ ใน 5 ข้อ

1. ChatGPT เปิดตัวปลายปี 2022 แล้วโตเร็วจนน่าตกใจ เพราะมันคุยกับเราได้เป็นธรรมชาติแบบที่ AI รุ่นก่อนทำไม่ได้
2. AI แบบเก่าตอบได้แค่สิ่งที่มีคนเขียนคำตอบไว้ล่วงหน้า ส่วน ChatGPT เหมือนสร้างคำตอบใหม่สด ๆ ทุกครั้ง
3. "พูดเหมือนคน" ไม่จำเป็นต้องแปลว่า "เข้าใจเหมือนคน" — และนี่คือคำถามที่แม้แต่นักวิจัยเองก็ยังเถียงกันไม่จบ
4. ChatGPT ไม่ได้โผล่มาจากที่ว่างเปล่า แต่เป็นปลายทางของการเดินทางกว่าหกสิบปี ที่เราจะไล่ดูทีละก้าว
5. ตลอดเล่มนี้ ให้ถือคำถามหนึ่งติดมือไว้เสมอ — "มันรู้จริง หรือแค่เดาให้ฟังดูดี?"

บทที่ 2 — ทำไมภาษามนุษย์ถึงยากสำหรับเครื่อง

ลองอ่านประโยคนี้ช้า ๆ แล้วตอบในใจก่อนอ่านต่อ:

> **"เขาถ่ายรูปตา"**

ในประโยคนี้ คำว่า "ตา" หมายถึงอะไร?

อย่าเพิ่งเลื่อนสายตาลงไป ลองตอบจริง ๆ ก่อน

...

เอาล่ะ ถ้าคุณตอบว่า "ดวงตา" — เขากำลังถ่ายภาพดวงตาของใครสักคนแบบใกล้ ๆ — ก็ถูก แต่ถ้าคุณตอบว่า "คุณตา" — เขากำลังถ่ายรูปให้ปู่ฝ่ายแม่ — ก็ถูกเหมือนกัน ประโยคสั้น ๆ สามคำนี้ตีความได้สองแบบที่สมเหตุสมผลพอ ๆ กัน และถ้าไม่มีบริบทอื่นมาช่วย คุณก็เลือกไม่ได้จริง ๆ ว่าผู้พูดหมายถึงอันไหน

นี่แหละคือจุดที่น่าสนใจ ถ้าเราเติมว่า "เขาเพิ่งกลับจากร้านตัดแว่น" คุณจะเดาทันทีว่าหมายถึงดวงตา แต่ถ้าเปลี่ยนเป็น "วันนี้วันเกิดคุณตาพอดี" คุณก็เดาอีกแบบทันที สมองของคุณหยิบบริบทมาตัดสินใจให้เองโดยที่คุณไม่ทันรู้ตัวด้วยซ้ำ

และนั่นแหละคือสิ่งที่ยากที่สุดสำหรับเครื่อง

ในบทที่แล้ว เราตั้งคำถามหลักของทั้งเล่มไว้ว่า ChatGPT "รู้จริง หรือแค่เดาให้ฟังดูดี?" ก่อนจะตอบคำถามนั้นได้ เราต้องเข้าใจก่อนว่า "โจทย์" ที่ AI ทุกยุคพยายามแก้ มันคืออะไร — และมันยากแค่ไหน บทนี้คือคำตอบของ "ทำไมแค่นั้นยังไม่พอ" ข้อแรกสุด นั่นคือ ทำไมภาษาที่เราใช้กันทุกวันแบบไม่ต้องคิด ถึงเป็นกำแพงที่นักวิจัยทั้งโลกปีนกันอยู่หลายสิบปี

เรื่องที่เราคิดว่าง่าย แต่จริง ๆ แล้วยากมาก

คนส่วนใหญ่รู้สึกว่าภาษาเป็นเรื่องง่าย เพราะเราใช้มันได้ตั้งแต่ยังพูดไม่ชัด คุณไม่เคยต้อง "ตั้งใจ" แปลความหมายของประโยคที่เพื่อนส่งมา มันเข้าใจของมันเอง

เลยมีความเชื่อหนึ่งที่ฟังดูสมเหตุสมผลมาก — "ก็แค่ป้อนคำศัพท์กับไวยากรณ์ให้คอมพิวเตอร์สิ ในเมื่อมันคำนวณเลขยาก ๆ ได้เร็วกว่าเราตั้งเยอะ เรื่องภาษาจะยากกว่าได้ยังไง"

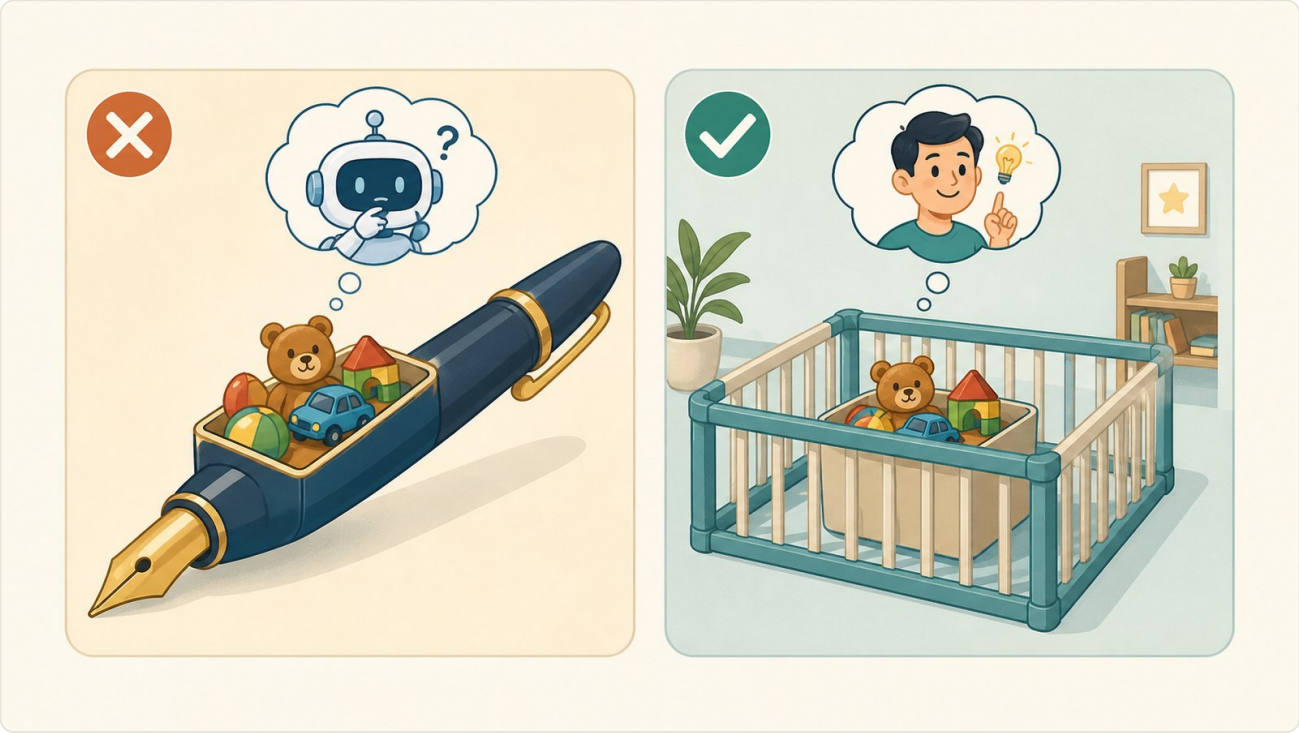
ฟังดูเข้าท่า แต่ลองคิดเล่น ๆ ดู เครื่องคิดเลขเก่งเพราะตัวเลขไม่เคยกำกวม เลข 7 ก็คือ 7 หมากรุกก็มีกฎตายตัว เดินม้าได้แค่แบบเดียว ไม่มีวันที่ม้าจะ "หมายถึงอย่างอื่น" แต่ภาษามนุษย์ตรงข้ามกันคนละขั้ว มันเต็มไปด้วยความหมายที่ไม่ได้อยู่ในตัวอักษร

และเรื่องนี้ก็ไม่ใช่ปัญหาใหม่เลย

ย้อนกลับไปปี 1960 นักวิจัยชื่อ เยโฮชัว บาร์-ฮิลเลล (Yehoshua Bar-Hillel) กำลังทำงานเรื่องการแปลภาษาด้วยเครื่อง ซึ่งเป็นความฝันใหม่ของยุคนั้น รัฐบาลหลายประเทศทุ่มเงินมหาศาล เพราะหวังว่าวันหนึ่งเครื่องจะแปลภาษาได้เองอัตโนมัติ แต่บาร์-ฮิลเลลกลับเขียนประโยคสั้น ๆ ขึ้นมาประโยคหนึ่งเพื่อชี้ให้เห็นว่ามันยากกว่าที่ทุกคนคิด

ประโยคนั้นคือ "จอห์นตัวน้อยกำลังหากล่องของเล่นของเขา ในที่สุดเขาก็เจอมัน กล่องอยู่ใน *pen*"

คำว่า pen ในภาษาอังกฤษแปลได้ทั้ง "ปากกา" และ "คอกกั้นเด็กเล็ก" แล้วเครื่องจะรู้ได้ยังไงว่ากล่องของเล่นอยู่ใน "ปากกา" หรือใน "คอกเด็ก"? คุณกับผมรู้ทันทีว่าต้องเป็นคอกเด็ก เพราะเรารู้โดยไม่ต้องมีใครสอนว่ากล่องของเล่นมันใหญ่กว่าปากกา แต่เล็กพอจะวางในคอกเด็กได้ ความรู้แบบนี้แหละที่เครื่องไม่มีติดตัวมา



commonsense reasoning: มนุษย์รู้ทันทีว่าฉากไหนสมเหตุสมผล แต่เครื่องไม่รู้

บาร์-ฮิลเลลสรุปว่า ถ้าขาดความรู้เรื่องโลกจริงแบบนี้ การแปลภาษาให้ถูกต้องสมบูรณ์ก็เป็นไปไม่ได้ เขาเขียนเรื่องนี้เมื่อกว่าหกสิบปีก่อนที่ ChatGPT จะถือกำเนิด — โจทย์เก่า แต่คำตอบยังต้องรออีกนาน

ปัญหาแรก คำเดียวมีหลายหน้า

ความยากชั้นแรกของภาษามาจากเรื่องง่าย ๆ ที่เรามองข้าม — คำคำเดียวมีได้หลายความหมาย นักภาษาศาสตร์เรียกปรากฏการณ์นี้ว่า **ความกำกวมระดับคำ** และมันเป็นอุปสรรคใหญ่ของทุกระบบที่พยายามจัดการกับภาษามนุษย์

ภาษาไทยเต็มไปด้วยตัวอย่างแบบนี้ คำว่า "ตา" ที่เราเล่นกันตอนต้นบทมีอย่างน้อยสามความหมาย — ดวงตา, คุณตา, และตาในเกม (เช่น "ตาใครตามัน")

ลองดูคำว่า "ฟัน" ในสามประโยคนี้:

- "แปรงฟันให้สะอาดด้วยนะ"
- "เขาฟันต้นไม้จนล้มทั้งต้น"
- "ระวังโดนฟันราคาตอนซื้อของฝาก"

คำเดียวกันเป๊ะ แต่หมายถึงคนละเรื่องเลย — อวัยวะในปาก, การใช้มีดสับ, และการถูกโก่งราคา คุณอ่านปุ๊บเข้าใจปั๊บ ไม่ต้องหยุดคิด แต่เครื่องต้องตัดสินใจทุกครั้งว่า "ฟัน" ในประโยคนี้คืออันไหนในสามอันนี้

และนี่คือกับดักของมัน — เครื่องจะเลือกความหมายถูกได้ ต้องดูบริบท แต่บริบทก็ประกอบด้วยคำอื่น ๆ ที่แต่ละคำก็มีหลายความหมายเหมือนกัน มันเลยกลายเป็นการไล่หาความหมายที่ต้องพึ่งความหมาย ที่ต้องพึ่งความหมาย วนไปเรื่อย ๆ

> **จำง่าย ๆ ว่า:** คำเดียวมีหลายหน้า เครื่องต้องเลือกหน้าที่ถูกให้ได้ ซึ่งต้องอาศัยบริบท แต่บริบทก็สร้างจากคำที่มีหลายหน้าอีกที — งูกินหางกันไปเรื่อย ๆ

ปัญหาที่สอง ทั้งประโยคก็ตีความได้หลายแบบ

สมมติว่าเราแก้ปัญหาคำหลายความหมายได้แล้ว ความยากชั้นต่อมาก็มารออยู่ — บางทีคำแต่ละคำชัดเจนดี แต่พอเอามาเรียงเป็นประโยค โครงสร้างของมันกลับตีความได้หลายแบบ

ตัวอย่างที่โด่งดังที่สุดในวงการมาจากภาษาอังกฤษ ในช่วงต้นยุค 1960 นักวิจัยที่มหาวิทยาลัยฮาร์วาร์ดกำลังพัฒนาโปรแกรมแปลภาษา และใช้ประโยคหนึ่งเพื่อโชว์ว่าเครื่องสับสนได้ยังไง — *"Time flies like an arrow"*

ประโยคห้าคำนี้ คนอังกฤษอ่านแล้วเข้าใจทันทีว่า "เวลาผ่านไปไวเหมือนลูกศร" แต่ในทางไวยากรณ์แล้ว มันยังตีความได้อีกอย่างน้อยสองแบบที่ "ถูกหลักทั้งหมด" — เช่น อ่านเป็นคำสั่งว่า "จงจับเวลาการบินของแมลงวันให้เร็วเหมือนลูกศร" หรืออ่านว่า "แมลงวันพันธุ์เวลาชอบลูกศร" ฟังดูไร้สาระสำหรับเรา แต่เครื่องในยุคนั้นไม่มีทางรู้เลยว่าแบบไหน "สมเหตุสมผล" เพราะทั้งสามแบบถูกไวยากรณ์เท่ากันหมด

(ขอแทรกนิดนึง หลายคนเข้าใจว่าประโยคนี้เป็นมุกของนักแสดงตลก กราวโช มาร์กซ์ แต่ไม่มีหลักฐานยืนยันที่รู้แน่คือนักวิจัยฮาร์วาร์ดใช้ประโยคนี้ในงานวิจัยตั้งแต่ปี 1963)

ภาษาไทยก็มีปัญหาแบบเดียวกัน ลองดูประโยค "แม่ซื้อขนมมาให้เด็กที่น่ารักในห้องครัว" — เด็กน่ารักคนนั้นอยู่ในห้องครัว หรือว่าแม่ซื้อขนมในห้องครัว? คำว่า "ในห้องครัว" จะไปเกาะกับ "เด็ก" หรือกับ "ซื้อ" ก็ได้ทั้งคู่ คุณอาจเลือกแบบที่ฟังเข้าท่ากว่าโดยอัตโนมัติ แต่เครื่องต้องนั่งวิเคราะห์โครงสร้างทั้งประโยคก่อนถึงจะเริ่มเดาได้

ปัญหาที่สาม น้ำเสียงและสิ่งที่ไม่ได้พูดออกมา

จนถึงตรงนี้เรายังพูดถึงแต่ "ความหมายตามตัวอักษร" แต่มนุษย์สื่อสารมากกว่าตัวอักษรเยอะ

ลองนึกถึงเพื่อนที่ส่งข้อความมาว่า **"เยี่ยมมากเลย"** หลังจากคุณเล่าให้ฟังว่าวันนี้ตื่นสาย พลาดประชุม แล้วทำกาแฟหกใส่กางเกงตัวโปรด คุณรู้ทันทีว่าเพื่อนไม่ได้ชมจริง — มันคือการประชด ทั้งที่ตัวอักษร "เยี่ยมมากเลย" ไม่ได้บอกอะไรเลยว่ามันคือคำชมหรือคำเยาะเย้ย

ถ้าเครื่องเห็นแค่ข้อความสามคำนี้ลอย ๆ โดยไม่รู้เรื่องราวก่อนหน้า มันไม่มีทางแยกออกเลยว่าคุณกำลังถูกชมหรือถูกแซว และเรื่องการอ่านน้ำเสียง การเข้าใจประชดเสียดสีแบบนี้ ยังเป็นโจทย์ที่ยากสำหรับ AI มาจนถึงทุกวันนี้

อีกชั้นหนึ่งที่ลึกกว่านั้นคือ "สิ่งที่เราไม่ได้พูดออกมา" คนไทยถามกันว่า "กินข้าวยัง?" ประโยคเดียวนี้เป็นได้ทั้งการห่วงใย (แปลว่ารีบกินซะนะ), การชวนกินด้วยกันแบบอ้อม ๆ, หรือแค่คำทักทายที่ไม่ได้อยากรู้คำตอบจริง ๆ ด้วยซ้ำ เราเลือกตีความถูกจากบริบทสังคม — ใครพูด พูดตอนไหน พูดด้วยน้ำเสียงแบบไหน — สิ่งที่เครื่องไม่มีติดตัว

พูดง่าย ๆ คือ มนุษย์มักพูดน้อยกว่าที่ตั้งใจจะสื่อ แล้วปล่อยให้อีกฝ่ายเติมส่วนที่เหลือเอง แต่เครื่องเห็นแค่คำที่พิมพ์ออกมาจริง ๆ เท่านั้น

ปัญหาที่สี่ ความรู้ที่ทุกคนมี แต่เครื่องไม่มี

มาถึงปัญหาที่อาจลึกที่สุด — ความรู้เรื่องโลกที่มนุษย์ทุกคนมีร่วมกัน นักวิจัยเรียกมันว่า **ความรู้สามัญสำนึก** และมันเป็นหนึ่งในปัญหาที่ยากที่สุดในวงการ AI

กลับไปที่ปัญหาเรื่องสรรพนามที่ชี้ไม่ชัด ในปี 1972 นักวิทยาศาสตร์คอมพิวเตอร์ชื่อ เทอร์รี วินอกราด (Terry Winograd) เขียนประโยคตัวอย่างไว้ว่า "สมาชิกสภาเทศบาลปฏิเสธไม่ออกใบอนุญาตให้กลุ่มผู้ประท้วงเพราะ *พวกเขา* กลัวความรุนแรง"

คำว่า "พวกเขา" หมายถึงใคร — สมาชิกสภา หรือผู้ประท้วง? คุณตอบได้แทบจะทันทีว่า "สมาชิกสภา" แต่ลองสังเกตว่าคุณรู้ได้ยังไง คุณใช้ความรู้เรื่องสังคมจริง ๆ ว่าโดยปกติฝ่ายที่มีอำนาจออกใบอนุญาตและกลัวความวุ่นวายคือเจ้าหน้าที่ ไม่ใช่ผู้ประท้วง

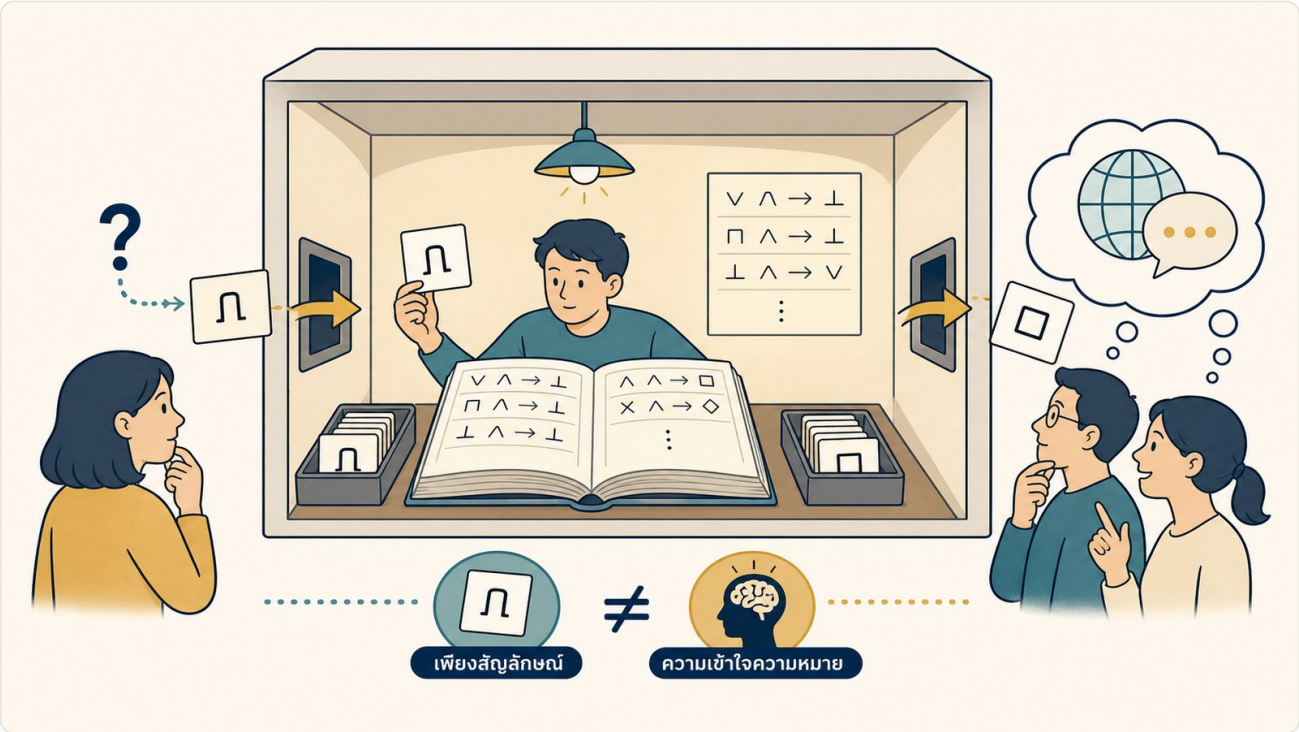
ที่เจ๋งกว่านั้นคือ ถ้าเปลี่ยนแค่คำเดียว จาก "กลัวความรุนแรง" เป็น "ต้องการก่อความรุนแรง" คำตอบของ "พวกเขา" จะพลิกไปเป็นผู้ประท้วงทันที ทั้งที่โครงสร้างประโยคเหมือนเดิมเป๊ะ คุณรู้คำตอบได้เพราะคุณเข้าใจโลก ไม่ใช่เพราะคุณวิเคราะห์ไวยากรณ์

ต่อมาในปี 2012 นักวิจัยชื่อ เฮกเตอร์ เลอเวสก์ (Hector Levesque) เอาประโยคแบบนี้มาทำเป็นบททดสอบความฉลาดของ AI โดยตั้งชื่อตามวินอกราด — และมันเป็นกำแพงที่ AI รุ่นแล้วรุ่นเล่าข้ามไม่ได้อยู่นานมาก

อีกตัวอย่างที่เห็นภาพชัดคือ "ถ้วยรางวัลใส่กระเป๋าเดินทางไม่ได้เพราะ *มัน* ใหญ่เกินไป" — "มัน" คือถ้วยรางวัล (เพราะถ้วยใหญ่เกิน) แต่พอเปลี่ยนเป็น "เพราะมันเล็กเกินไป" — "มัน" กลายเป็นกระเป๋าทันที (เพราะกระเป๋าเล็กเกิน) คุณรู้คำตอบเพราะคุณเข้าใจว่าของชิ้นเดียวจะ "ใหญ่เกิน" และ "เล็กเกิน" พร้อมกันไม่ได้ ความรู้แบบนี้คุณมีมาตั้งแต่เด็ก แต่เครื่องต้องถูกบอกก่อนถึงจะนำมาใช้ได้

จำง่าย ๆ ว่า: มนุษย์รู้ว่ากล่องของเล่นใหญ่กว่าปากกา โดยไม่ต้องมีใครสอน และมีความรู้แบบนี้อีกหลายพันล้านเรื่องที่เราใช้ตีความภาษาทุกวัน เครื่องต้องได้รับความรู้พวกนี้มาก่อน ถึงจะเข้าใจประโยคธรรมดา ๆ ได้

ปมที่ลึกที่สุด เครื่องเห็นสัญลักษณ์ แต่ไม่เห็นความหมาย



การทดลองทางความคิด Chinese Room: จัดการสัญลักษณ์ ไม่เท่ากับเข้าใจ

ทั้งสี่ปัญหาที่ผ่านมา จริง ๆ แล้วมีต้นตอเดียวกัน — สำหรับคอมพิวเตอร์ ตัวอักษรที่เราพิมพ์เข้าไปเป็นแค่ "สัญลักษณ์" ที่ไม่มีความหมายในตัวมันเอง มันเหมือนรูปร่างแปลก ๆ ที่เครื่องจัดเรียงตามกฎ แต่ไม่เคยรู้ว่ารูปร่างเหล่านั้นหมายถึงอะไรในโลกจริง

มีการทดลองความคิดอันหนึ่งที่อธิบายเรื่องนี้ได้ดีมาก ในปี 1980 นักปรัชญาชื่อ จอห์น เซิร์ล (John Searle) ชวนเราจินตนาการแบบนี้

สมมติว่าคุณนั่งอยู่ในห้องปิด มีช่องเล็ก ๆ ด้านนอกคนส่งกระดาษที่เขียนเป็นภาษาจีนเข้ามา คุณอ่านภาษาจีนไม่ออกเลยสักตัว แต่คุณมีคู่มือหนาเตอะที่บอกว่า "ถ้าเห็นสัญลักษณ์ชุดนี้ ให้เขียนสัญลักษณ์ชุดนั้นตอบกลับไป" คุณทำตามกฎได้อย่างแม่นยำ จนคนข้างนอกเชื่อสนิทใจว่าคุณพูดจีนคล่อง — ทั้งที่จริงคุณไม่รู้เลยว่าตัวเองกำลัง "พูด" ว่าอะไร

เซิร์ลใช้เรื่องนี้ตั้งคำถามชวนคิดว่า การทำตามกฎได้อย่างสมบูรณ์แบบ มันเท่ากับ "การเข้าใจ" จริง ๆ หรือเปล่า? คอมพิวเตอร์ก็คล้ายคนในห้องนี้ — มันจัดเรียงสัญลักษณ์ตามกฎได้เก่งมาก แต่ไม่ได้ "รู้จัก" สิ่งที่สัญลักษณ์เหล่านั้นหมายถึงในโลกจริง

ขอย้ำว่าเรื่องห้องจีนนี้เป็น "การทดลองความคิด" ที่นักปรัชญายังเถียงกันไม่จบ ไม่ใช่ข้อพิสูจน์ว่า AI ไม่มีวันเข้าใจอะไรได้เลย เราหยิบมันมาเพื่อให้เห็นภาพว่า "ปัญหา" ที่เครื่องต้องเจอนั้นลึกแค่ไหน ส่วนที่ AI ยุคใหม่ทำได้ดีขึ้นเรื่อย ๆ นั้น เป็นเรื่องของบทต่อ ๆ ไป

ลองนึกภาพมนุษย์ต่างดาวที่อ่านพจนานุกรมไทยจบทั้งเล่ม รู้ทุกคำ แต่ไม่เคยมาเหยียบโลก ถ้าเราบอกว่า "เรื่องนี้ก็กล้วย ๆ" ต่างดาวจะงงทันที เพราะพจนานุกรมบอกว่ากล้วยคือผลไม้สีเหลือง ไม่ได้บอกว่ามันแปลว่า "ง่าย" ได้ด้วย ความรู้จากพจนานุกรมไม่เท่ากับความเข้าใจจากการใช้ชีวิตจริง

ศัพท์ที่เพิ่งเจอ: - **ความกำกวมระดับคำ** — การที่คำคำเดียวมีได้หลายความหมาย เช่น "ตา" "ฟัน" "ขัน" เครื่องต้องเลือกความหมายที่ถูกต้องจากบริบท - **ความรู้สามัญสำนึก** — ความรู้เรื่องโลกที่มนุษย์ทุกคนมีร่วมกันโดยไม่ต้องสอน เช่น กล่องใหญ่กว่าปากกา หรือของชิ้นเดียวจะใหญ่เกินและเล็กเกินพร้อมกันไม่ได้

เกร็ดพิเศษ ภาษาไทยยากกว่านั้นอีกนิด

ที่เล่ามาทั้งหมดคือความยากที่ภาษาทุกภาษาในโลกมีร่วมกัน แต่ภาษาไทยยังแถมความยากพิเศษมาอีกชั้น — เราเขียนติดกันโดยไม่เว้นวรรคระหว่างคำ

ลองดูข้อความที่เขียนติดกันว่า "ตากลม" เครื่องต้องเดาก่อนว่านี่คือ "ตา-กลม" (ดวงตาที่กลม) หรือ "ตาก-ลม" (เอาไปตากลม) สำหรับภาษาอังกฤษที่เว้นวรรคทุกคำ ปัญหานี้ไม่มีเลย แต่ภาษาไทยต้องตัดคำให้ออกก่อน ถึงจะเริ่มตีความความหมายได้แม้แต่คำเดียว — และการตัดคำเองก็กำกวมได้อีก

พูดง่าย ๆ คือ ภาษาอังกฤษเริ่มต้นที่บันไดขั้นแรก ส่วนภาษาไทยต้องหาบันไดขั้นแรกให้เจอก่อนด้วยซ้ำ

ลองมองภาษาแบบที่เครื่องต้องเจอ

ทีนี้มาลองฝึกกัน ด้านล่างคือชุดประโยคกำกวมภาษาไทยให้คุณลองตีความเอง ลองอ่านแต่ละประโยคแล้วถามตัวเองว่า "มันหมายถึงอะไรได้บ้าง" ก่อนดูเฉลย คุณจะเริ่มรู้สึกถึงสิ่งที่เครื่องต้องเผชิญทุกครั้งที่อ่านประโยคหนึ่ง

มีประโยคหนึ่งในชุดนี้ที่ต้องเล่าเกร็ดก่อน — "เขารักกัน" คำว่า "กัน" ในภาษาไทยเป็นได้ทั้งคำที่บอกว่า "ต่างคนต่างทำต่อกัน" (เขาสองคนรักกัน) และยังเป็น *ชื่อเล่นที่คนไทยใช้กันจริง ๆ* ด้วย พอรู้แบบนี้ ประโยคนี้ก็อ่านได้สองแบบทันที — "พวกเขารักกัน" (เป็นคู่รักกัน) หรือ "เขารักกัน" (เขาหลงรักคนที่ชื่อกัน) ทั้งสองแบบถูกหมด

#	ประโยค	ตีความแบบ A	ตีความแบบ B	ความกำกวมมาจาก
1	"แม่ซื้อขนมมาให้เด็กที่น่ารักในห้องครัว"	เด็กน่ารักอยู่ในห้องครัว	แม่ซื้อขนมในห้องครัว	โครงสร้างประโยค
2	"เขาถ่ายรูปตา"	ถ่ายภาพดวงตา	ถ่ายรูปให้คุณตา	คำว่า "ตา"
3	"เขารักกัน"	พวกเขารักซึ่งกันและกัน	เขารักคนชื่อกัน	คำว่า "กัน"
4	"เขาบอกเพื่อนว่าเขาสอบผ่าน"	ตัวเขาเองสอบผ่าน	เพื่อนสอบผ่าน	สรรพนาม "เขา"
5	"เสียงขันดังมาก"	เสียงไก่ขัน (ไก่ร้อง)	เสียงขัน (ภาชนะตักน้ำ) กระทบกันดัง	คำว่า "ขัน"
6	"เพื่อนบอกว่าดีมากเลย"	ชมจริง ๆ	ประชดเสียดสี	น้ำเสียง

การ์ดตัวอย่างประโยคกำกวมในภาษาไทย

สังเกตไหมว่าแต่ละประโยคกำกวมด้วย "คนละสาเหตุ" — บางอันเพราะคำมีหลายความหมาย (2, 3, 5) บางอันเพราะโครงสร้างประโยค (1) บางอันเพราะสรรพนามชี้ไม่ชัด (4) และบางอันเพราะน้ำเสียงที่ตัวอักษรบอกไม่ได้ (6) นี่คือความกำกวมทั้งสี่แบบที่เราคุยกันมาตลอดบท มากองรวมกันอยู่ในประโยคสั้น ๆ ที่คุณเจอได้ในชีวิตประจำวัน

เช็กลิสต์ "อะไรทำให้ประโยคหนึ่งกำกวม"

ครั้งหน้าที่คุณเจอประโยคที่ตีความได้หลายแบบ ลองไล่ถามตัวเองตามนี้ มันจะช่วยให้คุณ "มองภาษาแบบที่เครื่องต้องมอง" ได้ชัดขึ้น:

- ☐ **คำมีหลายความหมายไหม** — มีคำไหนในประโยคที่แปลได้มากกว่าหนึ่งแบบ?
- ☐ **โครงสร้างตีความได้หลายแบบไหม** — ใครทำอะไรกับใคร ส่วนขยายไปเกาะกับคำไหนกันแน่?
- ☐ **สรรพนามชี้ชัดไหม** — คำว่า "เขา / มัน / นี่ / นั่น" หมายถึงสิ่งไหนแน่?
- ☐ **ขาดน้ำเสียงหรือบริบทไหม** — ถ้าอ่านแค่ตัวอักษร เราแยกออกไหมว่าชมหรือประชด?
- ☐ **ต้องใช้ความรู้เรื่องโลกไหม** — ต้องรู้ว่าสิ่งนี้ทำงานยังไงในชีวิตจริง ถึงจะเข้าใจประโยค?

ถ้าประโยคไหนติ๊กถูกหลายข้อ นั่นแหละคือประโยคที่เครื่องในยุคก่อนต้องปวดหัวที่สุด

บทนี้คือกุญแจของทั้งเล่ม

ทั้งหมดที่เราเดินผ่านมาในบทนี้ ไม่ได้มีไว้เพื่อบอกว่า "AI โง่" หรือ "เครื่องไม่มีทางทำได้" ตรงกันข้าม มันมีไว้เพื่อให้คุณรู้สึกถึง "ขนาดของโจทย์" ที่นักวิจัยทั้งโลกพยายามแก้กันมาหกสิบกว่าปี

ลองนึกย้อนไปที่คนในห้องจีน เขาหยิบสัญลักษณ์มาจัดเรียงตอบได้อย่างไม่มีที่ติ แต่ไม่เคยรู้เลยว่ามันหมายถึงอะไร นั่นคือจุดที่ทุกยุคของ AI ภาษาพยายามจะก้าวข้ามไปให้ได้ — ทำยังไงให้เครื่องไม่ใช่แค่ "คนในห้องที่ทำตามกฎ" แต่เริ่มแตะความหมายที่อยู่หลังตัวอักษรได้จริง พอคุณเห็นว่าโจทย์มันใหญ่แค่ไหน คุณจะเข้าใจทันทีว่าทำไมนักวิจัยถึงต้องลองแล้วลองอีก เปลี่ยนวิธีคิดกันครั้งแล้วครั้งเล่าตลอดหกสิบกว่าปี

และคำถามหลักของเราก็ยังอยู่ตรงนั้น เมื่อ ChatGPT ตอบคุณได้อย่างน่าทึ่ง มันก้าวข้ามกำแพงพวกนี้มาได้จริง หรือมันแค่ "เดาให้ฟังดูดี" เก่งขึ้นมาก? เก็บคำถามนี้ติดมือไว้ แล้วในบทถัดไป เราจะไปดูความพยายามแรกสุดของมนุษย์ — ยุคที่เราลองเขียนกฎทุกข้อให้เครื่องทำตาม และได้เรียนรู้ว่าทำไมแค่นั้นถึงยังไม่พอ

สรุปง่าย ๆ ใน 5 ข้อ

1. ภาษาที่เราใช้แบบไม่ต้องคิด เต็มไปด้วยความกำกวม — คำเดียวหลายความหมาย ประโยคเดียวหลายการตีความ
2. มนุษย์สื่อสารมากกว่าตัวอักษร เราอ่านน้ำเสียง อ่านประชด และเข้าใจสิ่งที่อีกฝ่ายไม่ได้พูดออกมา
3. เราตีความถูกเพราะมี "ความรู้เรื่องโลก" มหาศาลที่ใช้โดยไม่รู้ตัว แต่เครื่องไม่มีความรู้นี้ติดตัวมา
4. ลึกที่สุดแล้ว เครื่องเห็นแค่สัญลักษณ์ที่ไม่มีความหมายในตัว มันจัดเรียงตามกฎได้ แต่ไม่ได้ "รู้จัก" สิ่งที่คำหมายถึง
5. ทุกยุคของ AI ภาษาที่เราจะไปดูต่อ คือความพยายามแก้โจทย์ข้อเดียวกันนี้ — และนั่นคือเหตุผลว่าทำไมมันใช้เวลาหลายสิบปี

บทที่ 3 — ยุคเขียนกฎให้เครื่องเข้าใจภาษา

ราวปี 1964 ถึง 1966 ที่ห้องแล็บของ MIT มีนักวิทยาศาสตร์คอมพิวเตอร์คนหนึ่งชื่อ Joseph Weizenbaum เขาใช้เวลาหลายเดือนเขียนโปรแกรมสนทนาตัวหนึ่งขึ้นมา ตั้งชื่อว่า ELIZA

วันหนึ่งเขาเปิดให้เลขานุการของตัวเองลองพิมพ์คุยกับมัน เลขานุการคนนี้นั่งอยู่ในห้องเดียวกับ Weizenbaum มาทั้งปี เห็นเขาสร้างเจ้าโปรแกรมนี้กับมือ รู้ดีว่ามันเป็นแค่โค้ดคอมพิวเตอร์ ไม่ใช่คน

แต่หลังจากพิมพ์คุยไปได้ไม่กี่ประโยค เธอกลับหันมาขอให้ Weizenbaum ออกไปจากห้องก่อน เพราะอยากคุยกับมันเป็นการส่วนตัว

Weizenbaum เล่าเหตุการณ์นี้ไว้ในงานเขียนของเขาเอง และมันทำให้เขาตกใจมาก เขาสร้างโปรแกรมง่าย ๆ ตัวหนึ่งขึ้นมา ไม่ได้ตั้งใจจะหลอกใคร แต่กลับมีคนรู้สึกว่ามัน "เข้าใจ" จนอยากเปิดใจคุยด้วยแบบไม่มีคนอื่นฟัง

(เกร็ดนี้เล่าต่อกันมาจากปากของ Weizenbaum เอง นักประวัติศาสตร์รุ่นหลังตั้งข้อสังเกตว่าตัวตนของเลขานุการคนนี้ไม่เคยถูกยืนยันชัด ๆ และรายละเอียดอาจถูกเล่าให้กลมขึ้นตามกาลเวลา — แต่ปรากฏการณ์ที่มันสะท้อนนั้นจริงและเกิดซ้ำกับคนอื่น ๆ อีกมาก)

คำถามคือ โปรแกรมจากปี 1966 ที่ไม่มีพลังประมวลผลอะไรเทียบกับมือถือในกระเป๋าคุณตอนนี้เลย ทำให้คนรู้สึกแบบนั้นได้อย่างไร และคำตอบของมันก็พาเราเข้าสู่ยุคแรกของเรื่องราวทั้งหมด — ยุคที่มนุษย์พยายามสอนเครื่องให้เข้าใจภาษาด้วยการ "เขียนกฎ"

"ก็แค่เขียนกฎให้คอมสิ จะยากตรงไหน"

ในบทก่อน เราเห็นแล้วว่าภาษามนุษย์เต็มไปด้วยความกำกวมและความหมายที่ซ่อนอยู่หลังตัวอักษร และเครื่องเห็นแค่สัญลักษณ์ที่ไม่มีความหมายในตัว ตอนจบเราทิ้งคำถามไว้ว่า แล้วมนุษย์ลองแก้โจทย์นี้ครั้งแรกด้วยวิธีไหน

วิธีแรกที่นึกออกก็คือวิธีที่ฟังดูสมเหตุสมผลที่สุด — ถ้าเครื่องไม่รู้กฎของภาษา เราก็เขียนกฎให้มันสิ

ฟังดูง่ายใช่ไหม ถ้าเรารู้กฎไวยากรณ์ทุกข้อ รู้ความหมายของทุกคำ เราก็แค่ป้อนทั้งหมดให้คอมพิวเตอร์ แล้วมันก็น่าจะเข้าใจภาษาได้ นี่คือสิ่งที่คนเก่ง ๆ ทั้งวงการคิดเหมือนกันในยุค 1950 ถึง 1980 และพวกเขาก็ลงมือทำจริง

วิธีนี้เรียกว่า **rule-based** — แปลตรงตัวคือ "อิงกฎ" พูดเป็นภาษาคนก็คือ ให้ผู้เชี่ยวชาญด้านภาษานั่งเขียนกฎทีละข้อ ๆ ใส่ลงไปในเครื่อง แล้วเครื่องก็ทำตามกฎนั้นอย่างเคร่งครัด เช่น

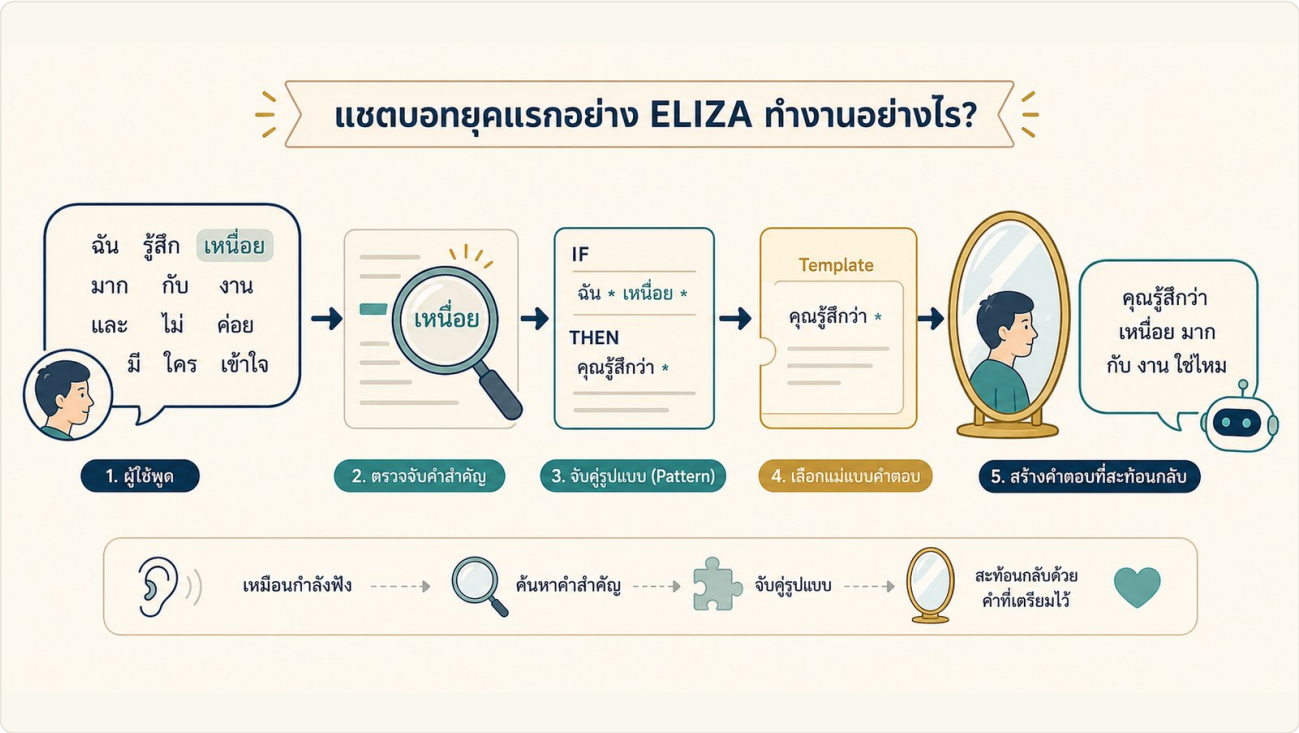
- ถ้าประโยคขึ้นต้นด้วยคำกริยา ให้ถือว่าเป็นประโยคคำสั่ง
- ถ้าเจอคำว่า "I" ตามด้วย "am" ให้รู้ว่า "I" คือประธาน

- ถ้าผู้ใช้พิมพ์คำว่า "แม่" ให้ถามกลับว่า "เล่าเรื่องครอบครัวคุณให้ฟังหน่อยสิ"

หัวใจของมันคือ เครื่องไม่ได้ "เรียนรู้" อะไรเลย มันแค่ทำตามสิ่งที่มนุษย์เขียนมาให้แล้ว ถ้ามีกฎ มันทำตาม ถ้าไม่มีกฎ มันก็ไม่รู้จะทำอะไร

> **ศัพท์ที่เพิ่งเจอ: rule-based (อิงกฎ)** — วิธีสอนเครื่องให้ทำงานกับภาษา โดยมนุษย์เขียนกฎและพจนานุกรมให้ครบ แล้วเครื่องทำตามกฎทีละขั้น ไม่ได้คิดเองหรือเรียนรู้เอง

ELIZA ทำงานยังไง — กระจกที่ฉลาดกว่าปกตินิดหน่อย



การทำงานแบบ pattern matching ของ ELIZA

กลับมาที่ ELIZA เพื่อจะเห็นว่ากฎพวกนี้ทำงานจริงยังไง

Weizenbaum ตั้งใจเลือกให้ ELIZA สวมบทเป็น "นักจิตบำบัด" สไตล์หนึ่งที่ชื่อ Rogerian ซึ่งเป็นสไตล์ที่นักบำบัดมักไม่พูดอะไรมาก แต่สะท้อนคำพูดของคนไข้กลับไปเป็นคำถาม เขาเลือกสไตล์นี้อย่างจงใจ เพราะมันทำให้โปรแกรมไม่จำเป็นต้อง "รู้" อะไรเกี่ยวกับโลกจริงเลย แค่เอาคำของคุณมาเรียงใหม่ก็พอ

ลองดูตัวอย่างจริง ถ้าคุณพิมพ์ว่า "I am very unhappy these days" (ช่วงนี้ฉันเศร้ามาก) ELIZA จะสแกนหาคำสำคัญในประโยค เจอคำว่า "unhappy" แล้วมันมีกฎเขียนไว้ว่า "ถ้าเจอรูปประโยค *I am X* ให้แปลงเป็น *How long have you been X?*" มันจึงตอบกลับว่า "How long have you been very unhappy these days?" (คุณเศร้าแบบนี้มานานแค่ไหนแล้ว)

อ่านแล้วรู้สึกเหมือนมันกำลังฟังคุณอยู่จริง ๆ ใช่ไหม แต่จริง ๆ แล้วมันแค่หยิบคำของคุณมาเรียงใหม่ในรูปคำถาม วิธีนี้มีชื่อเรียกว่า **pattern matching** — การจับคู่รูปแบบ คือเครื่องมองหา "รูปแบบประโยค" ที่ตรงกับกฎที่เขียนไว้ แล้วตอบตามแม่พิมพ์ที่เตรียมมา มันไม่รู้เลยว่า "unhappy" แปลว่าอะไร ไม่รู้ว่าคุณกำลังเศร้าจริงหรือล้อเล่น

> **ศัพท์ที่เพิ่งเจอ: pattern matching (การจับคู่รูปแบบ)** — เครื่องมองหารูปแบบของประโยคที่ตรงกับกฎที่เขียนไว้ล่วงหน้า แล้วตอบตามแม่พิมพ์ ไม่ได้เข้าใจความหมายของคำ

นี่แหละคือจุดที่น่าสนใจ และมันโยงกลับมาที่คำถามหลักของหนังสือเราพอดี ELIZA ไม่ได้ "เข้าใจ" คุณ มันแค่สะท้อนคำของคุณกลับมาเหมือนกระจก แต่กระจกบานนี้สะท้อนกลับมาในรูปคำถามที่ดูใส่ใจ คุณเลยรู้สึกว่ามีคนกำลังรับฟัง ปรากฏการณ์นี้คนเรียกกันว่า **ELIZA effect** — การที่เรารู้สึกว่าเครื่องเข้าใจเรา ทั้งที่จริง ๆ มันแค่ทำตามกฎง่าย ๆ

> **จำง่าย ๆ ว่า:** "เข้าใจจริง" กับ "ทำให้รู้สึกว่าเข้าใจ" เป็นคนละเรื่องกัน — และ ELIZA เมื่อปี 1966 ก็พิสูจน์ให้เห็นแล้วว่าเครื่องทำอย่างหลังได้ดีจนน่าตกใจ โดยไม่ต้องเข้าใจอะไรเลยสักนิด

คำถาม "เข้าใจจริง หรือแค่เดาให้ฟังดูดี" ที่เราถือติดมือมาตั้งแต่บทแรก จึงไม่ใช่เรื่องใหม่ของยุค ChatGPT มันเป็นคำถามที่นั่งรออยู่ในวงการมาตั้งแต่ปีที่ ELIZA ทำให้เลขานุการคนหนึ่งอยากคุยส่วนตัวแล้ว

วันที่ทุกคนคิดว่าปัญหากำลังจะถูกแก้

ก่อน ELIZA สิบกว่าปี วงการเคยมีช่วงที่มั่นใจสุด ๆ ว่ากฎจะพาไปถึงฝั่ง

เช้าวันที่ 7 มกราคม 1954 ที่นิวยอร์ก ทีมนักวิจัยจากมหาวิทยาลัย Georgetown ร่วมกับ IBM เปิดให้นักข่าวและนักวิทยาศาสตร์มาดูการสาธิต พวกเขาป้อนประโยคภาษารัสเซียกว่า 60 ประโยคเข้าเครื่อง แล้วเครื่องก็แปลออกมาเป็นภาษาอังกฤษได้ คนดูทึ่งกันทั้งห้อง

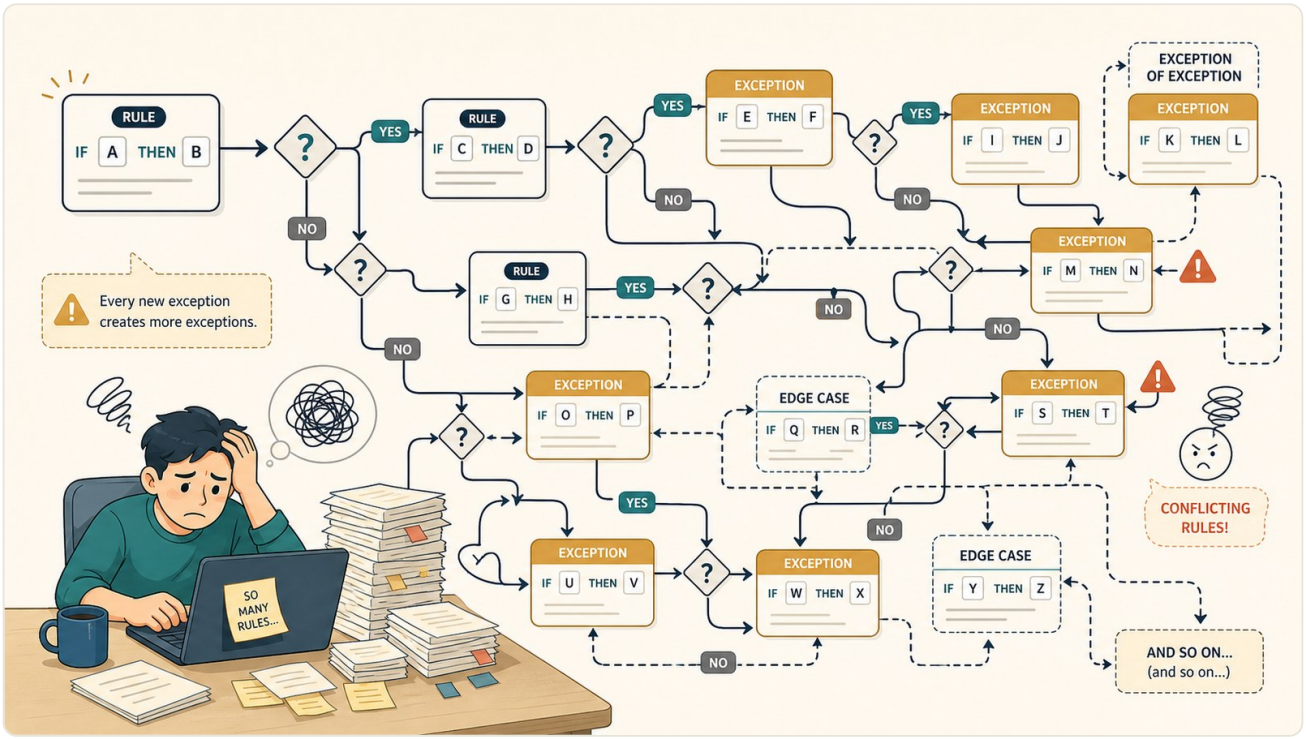
แต่เบื้องหลังความน่าทึ่งนั้น ระบบนี้เล็กกว่าที่คิดมาก มันใช้กฎไวยากรณ์แค่ 6 ข้อ คำศัพท์แค่ 250 คำ และเน้นประโยคในวงแคบ ๆ เรื่องเคมีอินทรีย์เป็นหลัก

ถึงอย่างนั้น บรรยากาศตอนนั้นก็เต็มไปด้วยความหวัง นักวิจัยยุคนั้นคาดการณ์ต่อสาธารณะว่า ภายใน 3 ถึง 5 ปี ปัญหาการแปลภาษาด้วยเครื่องน่าจะแก้ได้สำเร็จ ฟังดูใกล้แค่เอื้อม

แล้วเกิดอะไรขึ้น พอพวกเขาพยายามขยายจาก 250 คำไปสู่ภาษาจริงทั้งภาษา จาก 6 กฎไปสู่กฎที่ครอบคลุมทุกสถานการณ์ — ปัญหาก็เริ่มบานปลายจนคุมไม่อยู่ ถึงปี 1966 รัฐบาลสหรัฐตั้งคณะกรรมการชื่อ ALPAC มาประเมินความคืบหน้า ผลสรุปออกมาเจ็บปวด ว่าสิบปีของการวิจัยไม่ได้ตอบสนองความคาดหวังที่เคยตั้งไว้ และทำให้เงินทุนวิจัยการแปลภาษาด้วยเครื่องในสหรัฐลดลงอย่างมากหลังจากนั้น

สัญญา 3 ถึง 5 ปี กลายเป็นโจทย์ที่ใช้เวลากว่าหกสิบปีกว่าจะเริ่มทำได้ดีจริง คำถามคือ ทำไม

เมื่อกฎเริ่มระเบิดด้วยข้อยกเว้น



กฎและข้อยกเว้นที่บานปลายจนควบคุมยากในระบบ rule-based

ปัญหาไม่ได้อยู่ที่กฎ "ผิด" ปัญหาคือภาษาจริงมีข้อยกเว้นมากกว่ากฎเสมอ

ลองดูตัวอย่างง่าย ๆ แค่เรื่องเดียว — การเติมพหูพจน์ในภาษาอังกฤษ เริ่มจากกฎพื้นฐานที่ทุกคนรู้

- กฎ: เติม -s ท้ายคำ (cat → cats)

ดูเรียบร้อยดี แต่เดี๋ยวก่อน ยังมีข้อยกเว้น

- บางคำเปลี่ยนรูปทั้งคำ: mouse → mice, child → children
- บางคำไม่เปลี่ยนเลย: sheep → sheep, fish → fish
- คำที่ลงท้าย -y ต้องเปลี่ยนเป็น -ies: baby → babies

โอเค งั้นเพิ่มกฎ "-y เปลี่ยนเป็น -ies" เข้าไป แต่เดี๋ยวก่อนอีกที

- monkey ไม่ได้กลายเป็น monkies แต่เป็น monkeys

เห็นไหมว่าเกิดอะไรขึ้น เราต้องเขียน "ข้อยกเว้นของข้อยกเว้น" ซ้อนเข้าไปอีกชั้น และนี่แค่เรื่องพหูพจน์คำเดียวเท่านั้น เราต้องเขียนกฎหลายสิบข้อแล้วยังไม่ครบ ลองนึกภาพว่าถ้าต้องเขียนกฎให้ครอบคลุมทุกแง่มุมของภาษา — ทั้งกาล ทั้งสำนวน ทั้งความกำกวมระดับคำที่เราคุยกันในบทก่อน — กฎจะบานปลายไปขนาดไหน

ที่แย่กว่านั้นคือ พอกฎเยอะขึ้นถึงหลักพันหลักหมื่นข้อ การเพิ่มกฎใหม่หนึ่งข้อมักไปขัดกับกฎเก่าที่มีอยู่ แก้ตรงนี้ก็พังตรงโน้น ระบบที่ใหญ่ขึ้นเรื่อย ๆ จึงกลายเป็นฝันร้ายในการดูแล

นักวิจัยยุคนั้นเจอกำแพงที่ลึกกว่าแค่เรื่องจำนวนกฎ มันคือปัญหาที่ Edward Feigenbaum นักวิทยาศาสตร์คอมพิวเตอร์ที่ Stanford ตั้งชื่อไว้ในปี 1981 ว่า **knowledge acquisition bottleneck** — แปลเป็นภาษาคนคือ "คอขวดของการได้มาซึ่งความรู้"

พูดง่าย ๆ คือ การจะดึงความรู้ออกมาจากหัวผู้เชี่ยวชาญ แล้วแปลงให้เป็นกฎที่เครื่องทำตามได้ มันยากและช้ามาก ผู้เชี่ยวชาญหลายคนทำสิ่งที่ตัวเองเก่งได้ แต่อธิบายเป็นกฎชัด ๆ ไม่ได้ เพราะส่วนใหญ่เขาทำด้วยสัญชาตญาณที่สั่งสมมา ยิ่งงานซับซ้อน การนั่งเค้นความรู้ออกมาเขียนเป็นกฎก็ยิ่งกลืนเวลาและงบประมาณของทั้งโครงการ

> **ศัพท์ที่เพิ่งเจอ: knowledge acquisition bottleneck (คอขวดของการได้มาซึ่งความรู้)** — ความยากในการดึงความรู้จากหัวผู้เชี่ยวชาญมาเขียนเป็นกฎให้เครื่อง เป็นคอขวดที่ทำให้ระบบอิงกฎโตต่อได้ยาก

แต่อย่าเพิ่งดูถูกยุคของกฎ

ถึงตรงนี้อาจฟังดูเหมือนยุคของกฎเป็นความล้มเหลว — ไม่ใช่เลย

ลองดู MYCIN ระบบที่ Edward Shortliffe สร้างขึ้นที่ Stanford ในต้นทศวรรษ 1970 เพื่อช่วยวินิจฉัยโรคติดเชื้อและแนะนำยาปฏิชีวนะ มันใช้กฎประมาณ 600 ข้อที่เก็บความรู้ทางการแพทย์เอาไว้ และเมื่อนำไปทดสอบที่ Stanford Medical School มันได้คะแนนการยอมรับ 65% ซึ่งสูงกว่าแพทย์ผู้เชี่ยวชาญบางคนที่ร่วมประเมิน (ซึ่งอยู่ระหว่าง 42 ถึง 62%)

นี่ไม่ใช่ของเล่น ในโลกแคบ ๆ ที่ควบคุมได้และมีกฎชัดเจน ระบบอิงกฎทำงานได้ดีจริง บางครั้งดีกว่ามนุษย์ด้วยซ้ำ (น่าสนใจว่า MYCIN กลับไม่เคยถูกใช้รักษาคนไข้จริงในโรงพยาบาลเลย ไม่ใช่เพราะมันไม่เก่ง แต่เพราะข้อจำกัดเรื่องเครื่องคอมพิวเตอร์ยุคนั้นและประเด็นความรับผิดชอบทางกฎหมาย)

และที่สำคัญ ระบบอิงกฎไม่ได้หายไปไหน มันยังทำงานอยู่รอบตัวเราทุกวันนี้ — ตัวตรวจคำสะกดผิด ตัวตรวจไวยากรณ์ การกรองคำหยาบตามรายการ การค้นหารูปแบบข้อความด้วยสิ่งที่เรียกว่า regex แม้แต่ระบบ AI สมัยใหม่หลายตัวก็ยังเอากฎมาผสมกับการเรียนรู้แบบใหม่เพื่อให้ผลลัพธ์ดีขึ้น

ระบบอิงกฎเหมือนค้อน — เป็นเครื่องมือที่ยอดเยี่ยมสำหรับงานตอกตะปู ปัญหาไม่ใช่ว่าค้อนไม่ดี ปัญหาคือภาษามนุษย์ทั้งภาษาไม่ใช่ตะปู มันคืองานที่ใหญ่และยืดหยุ่นเกินกว่าจะแก้ด้วยเครื่องมือเดียวที่ต้องเขียนทุกอย่างด้วยมือ

สัญญาณว่าระบบกฎกำลังจะพัง

บทเรียนของยุคนี้มีค่ากับเรามากกว่าแค่ประวัติศาสตร์ เพราะมันใช้ได้กับทุกระบบที่พยายามแก้ปัญหาด้วยการ "เขียนกฎทุกข้อเอง" ไม่ว่าจะเป็น AI หรือแม้แต่กฎระเบียบในที่ทำงานของคุณ ลองใช้เช็กลิสต์นี้สังเกตดู

เช็กลิสต์: ระบบกฎกำลังจะพังหรือยัง

- ☐ **กฎเริ่มขัดกันเอง** — เพิ่มกฎใหม่ทีไร กฎเก่าทำงานเพี้ยน ต้องไล่แก้หลายจุดพร้อมกัน
- ☐ **ข้อยกเว้นท่วม** — แทบทุกกฎใหม่ที่เขียน ต้องตามด้วยกฎข้อยกเว้นอีกหลายข้อ วนไม่จบ
- ☐ **เปราะบาง** — พอเจอสถานการณ์ที่ไม่เคยวางแผนไว้ ระบบล้มทันที ไม่ตอบแบบ "พอใช้ได้"
- ☐ **ไม่ปรับตัวเอง** — มีอะไรเปลี่ยน ต้องนั่งแก้ด้วยมือทุกครั้ง ระบบไม่เรียนรู้จากของใหม่
- ☐ **ออกนอกกรอบไม่ได้** — ใช้ได้ดีเฉพาะในเรื่องที่ตั้งใจออกแบบไว้ พอเจอเรื่องนอกขอบเขตก็จบ

ถ้าติ๊กถูกตั้งแต่สามข้อขึ้นไป นั่นแหละสัญญาณว่าวิธี "เขียนกฎให้ครบ" กำลังจะถึงทางตัน — เหมือนที่วงการ AI ภาษาเจอเมื่อหลายสิบปีก่อน

ลองนึกถึงภาพนี้ดู สมมติคุณต้องเขียนคู่มือให้พนักงานใหม่ โดยคู่มือต้องบอกวิธีตอบสนองทุกสถานการณ์ที่ลูกค้าอาจพูดหรือทำได้ คุณเขียนกรณีทั่ว ๆ ไปได้สบาย แต่พอลูกค้าคนหนึ่งถามอะไรนอกสคริปต์ พนักงานก็จ๋อยทันที แล้วคุณก็ต้องกลับไปเพิ่มหน้าใหม่ในคู่มือ จากนั้นก็มีลูกค้าอีกคนถามนอกกรอบอีก คุณจะเขียนคู่มือเล่มนี้ไปถึงหน้าที่กี่พันก็ไม่มีวันจบ — เพราะสถานการณ์ในโลกจริงมีมากกว่าหน้ากระดาษเสมอ นั่นแหละคือสิ่งที่ระบบอิงกฎเผชิญกับภาษามนุษย์

ถ้าเขียนกฎไม่มีวันครบ แล้วจะทำยังไงต่อ

นี่คือจุดที่วงการทั้งวงการต้องหยุดคิดใหม่

ถ้าปัญหาคือ "มนุษย์เขียนกฎให้ครบไม่ได้" ทางออกอาจไม่ใช่การพยายามเขียนกฎให้หนักขึ้น แต่เป็นการ "เลิกเขียนกฎเองทั้งหมด" แล้วเปลี่ยนคำถามเป็น — จะเป็นไปได้ไหม ถ้าเราไม่บอกกฎให้เครื่อง แต่ให้เครื่องดูตัวอย่างภาษาจริงเยอะ ๆ แล้วปล่อยให้มันจับรูปแบบเอาเองว่าคำไหนมักจะตามคำไหน

ความคิดนี้ฟังดูแปลกในตอนนั้น แต่มันคือจุดเปลี่ยนครั้งใหญ่ที่จะพาเราเข้าสู่บทถัดไป — ยุคที่ "สถิติ" และ "การเรียนจากตัวอย่าง" เริ่มเข้ามาแทนที่การนั่งเขียนกฎทีละข้อ และมันคือก้าวที่เปลี่ยนทิศทางของเรื่องทั้งหมดไปตลอดกาล

สรุปง่าย ๆ ใน 5 ข้อ

1. ยุคแรกของ AI ภาษาใช้วิธี rule-based — มนุษย์เขียนกฎและพจนานุกรมให้ครบ แล้วเครื่องทำตามทีละขั้น โดยไม่ได้เรียนรู้เอง

2. ELIZA (1966) ทำให้คนรู้สึกว่ามันเข้าใจ ทั้งที่จริง ๆ มันแค่จับคู่คำสำคัญแล้วสะท้อนคำของเรากลับเป็นคำถาม — นี่คือ ELIZA effect
3. วิธีนี้ได้ผลดีในโลกแคบ ๆ ที่ควบคุมได้ (อย่าง MYCIN ที่วินิจฉัยโรคได้ดีกว่าหมอบางคน) แต่พังเมื่อเจอภาษาจริงทั้งภาษา
4. เพราะภาษามีข้อยกเว้นมากกว่ากฎเสมอ กฎจึงบานปลาย ขัดกันเอง และดูแลไม่ไหว — รวมถึงคอขวดของการดึงความรู้ออกมาเขียนเป็นกฎ
5. ยุคของกฎไม่ได้โง่ มันเป็นก้าวที่จำเป็นและยังใช้อยู่ทุกวันนี้ แต่มันสอนเราว่าภาษาซับซ้อนเกินกว่าจะเขียนกฎครบ — และนั่นเปิดทางให้แนวคิดใหม่ในบทถัดไป

บทที่ 4 — เมื่อสถิติเข้ามาแทนกฎ

ลองหยิบมือถือขึ้นมาแล้วเปิดช่องพิมพ์ข้อความ พิมพ์คำว่า "วันนี้" ลงไปคำเดียว แล้วหยุดมองเหนือแป้นพิมพ์สักครู่

มันจะมีคำแนะนำเด้งขึ้นมาให้คุณเลือก อาจเป็น "อากาศ" "ฝน" หรือ "ดี" — และบ่อยครั้งคำที่มันเดา ก็ดันเป็นคำที่คุณกำลังจะพิมพ์พอดี

คำถามที่น่าสนใจคือ ไม่มีใครนั่งเขียนกฎใส่มือถือคุณว่า "หลังคำว่าวันนี้ ให้แนะนำคำว่าอากาศ" ไม่มีนักภาษาศาสตร์คนไหนกรอกตารางไวยากรณ์ให้มัน แล้วมันรู้ได้ยังไงว่าควรเดาคำไหน

คำตอบของคำถามเล็ก ๆ นี้ คือจุดเปลี่ยนครั้งใหญ่ที่สุดครั้งหนึ่งในเรื่องราวทั้งหมดของเรา และมันคือบรรพบุรุษโดยตรงของ ChatGPT

ทางตันของยุคเขียนกฎ และทางออกที่ฟังดูบ้า ๆ

จำจุดที่เราค้างกันไว้บทก่อนได้ไหม

ยุคแรกของ AI ภาษาคือยุคที่มนุษย์พยายามเขียนกฎให้เครื่องทำตามทีละข้อ แต่ภาษาจริงมีข้อยกเว้นมากเกินกว่าจะเขียนกฎได้ครบ กฎใหม่ขัดกฎเก่า ระบบบานปลายจนดูแลไม่ไหว เราเลยทิ้งคำถามไว้ว่า — จะเป็นไปได้ไหม ถ้าเราไม่บอกกฎให้เครื่อง แต่ให้เครื่องดูตัวอย่างภาษาจริงเยอะ ๆ แล้วปล่อยให้มันจับรูปแบบเอาเองว่าคำไหนมักจะตามคำไหน

ในยุคนั้นความคิดนี้ฟังดูแปลกมาก เพราะมันเหมือนการยอมแพ้ แทนที่จะสอนเครื่องให้ "เข้าใจ" ไวยากรณ์ เรากลับบอกว่า "ช่างมันเถอะเรื่องเข้าใจ ขอแค่ให้มันนับเอาว่าอะไรมาคู่กับอะไรบ่อย ๆ ก็พอ"

แต่ปรากฏว่าการ "ยอมแพ้" แบบนี้แหละ ที่ได้ผลกว่าที่ทุกคนคิดมาก และที่น่าทึ่งคือ ไอเดียตั้งต้นของมันเกิดขึ้นก่อนที่จะมีมือถือ ก่อนจะมีอินเทอร์เน็ตด้วยซ้ำ — เกิดขึ้นในห้องนั่งเล่นของนักคณิตศาสตร์คนหนึ่ง เมื่อเจ็ดสิบกว่าปีก่อน

เกมเดาตัวอักษรในห้องนั่งเล่น

ปี 1951 นักคณิตศาสตร์ชื่อ Claude Shannon เล่นเกมง่าย ๆ อย่างหนึ่งกับภรรยาของเขา

เขาหยิบข้อความจากนวนิยายเรื่องหนึ่งมาปิดไว้ ไม่ให้เธอเห็น แล้วให้เธอเดาทีละตัวอักษรว่า ตัวถัดไปคืออะไร ถ้าเดาผิดก็เดาใหม่ จนกว่าจะถูก แล้วค่อยขยับไปตัวต่อไป

ลองคิดเล่น ๆ ว่า ถ้าภาษาเป็นเรื่องสุ่มล้วน ๆ คนเราควรจะเดาตัวอักษรถูกได้ยากมาก เพราะตัวอักษรในภาษาอังกฤษมีตั้ง 26 ตัว โอกาสเดาถูกควรอยู่แค่ราว 1 ใน 26

แต่ผลที่ออกมาไม่ใช่แบบนั้นเลย ภรรยาของเขาเดาถูกได้บ่อยอย่างน่าตกใจ — ในการทดลองหนึ่ง เดาถูกราว ๆ เกือบ 7 ครั้งจาก 10 ครั้ง

ทำไมถึงเป็นแบบนั้น ลองนึกถึงตอนคุณเห็นคำว่า "สวัสดีค..." คุณแทบไม่ต้องคิดเลยว่าตัวต่อไปคือ "รับ" หรือ "ะ" เพราะคุณเคยเห็นรูปแบบนี้ซ้ำมาทั้งชีวิต

นี่คือสิ่งที่ Shannon ค้นพบ — ภาษาไม่ได้สุ่ม มันเต็มไปด้วยรูปแบบที่ทำนายได้ และเราทุกคนมี "สถิติของภาษา" ฝังอยู่ในหัวโดยไม่รู้ตัว เราเดาคำต่อไปได้ ไม่ใช่เพราะท่องกฎไวยากรณ์ แต่เพราะเคยได้ยินเคยเห็นรูปแบบนั้นซ้ำ ๆ จนคุ้น

และนี่แหละคือจุดที่น่าสนใจ — ถ้ามนุษย์เดาได้เพราะเคยเห็นรูปแบบซ้ำ ๆ เครื่องก็น่าจะทำได้เหมือนกัน ขอแค่ให้มันได้ "เห็น" ภาษามากพอ Shannon เสนอไว้ตั้งแต่ตอนนั้นว่า สักวันเครื่องคอมพิวเตอร์จะทำเกมเดาแบบนี้ได้เอง

หัวใจของเรื่องทั้งหมด: เครื่องที่เดาคำถัดไป

ตรงนี้คือแนวคิดที่เป็นหัวใจของทั้งหนังสือเล่มนี้ ถ้าจะจำอะไรจากบทนี้แค่อย่างเดียว ขอให้จำสิ่งนี้

แทนที่จะถามว่า "ประโยคนี้ถูกไวยากรณ์ไหม" เราเปลี่ยนมาถามเครื่องว่า "หลังคำนี้ มักจะตามด้วยคำอะไร"

วิธีให้เครื่องตอบคำถามนี้ก็ตรงไปตรงมาอย่างไม่น่าเชื่อ — เอาข้อความจริงจำนวนมหาศาลมาให้มันอ่าน แล้วให้มัน "นับ" ว่าในข้อความเป็นล้าน ๆ ประโยคนั้น หลังคำว่า "กิน" มีคำอะไรตามมาบ้าง และตามมาบ่อยแค่ไหน

พอนับเสร็จ เครื่องก็จะรู้ว่าหลัง "กิน" คำว่า "ข้าว" ตามมาบ่อยกว่าคำว่า "รถ" เป็นล้านเท่า เวลาต้องเดาคำถัดไป มันก็เลือกคำที่เคยตามมาบ่อยที่สุด

> **จำง่าย ๆ ว่า:** language model แบบสถิติ ก็คือ "เครื่องนับว่าคำไหนมักตามคำไหน" แล้วใช้ข้อมูลนั้นเดาคำถัดไป — เหมือนแป้นพิมพ์มือถือที่เดาคำให้คุณ แต่ขยายใหญ่ขึ้นมหาศาล

สังเกตไหมว่าเครื่องไม่จำเป็นต้อง "เข้าใจ" เลยว่า "กินข้าว" หมายความว่าอะไร มันไม่รู้ว่าข้าวคืออาหาร ไม่รู้ว่าคนหิวแล้วต้องกิน มันแค่รู้ว่าหลังคำว่า "กิน" คำว่า "ข้าว" มาบ่อยกว่าคำอื่นมาก ๆ เท่านั้นเอง

นี่คือคำตอบที่ตรงที่สุดต่อคำถามหลักของหนังสือเล่มนี้ ที่ว่า AI "เข้าใจจริง หรือแค่เดาให้ฟังดูดี" — language model แบบสถิติคือตัวอย่างที่บริสุทธิ์ที่สุดของการ "เดาเก่ง โดยไม่ต้องเข้าใจ"

ศัพท์ที่เพิ่งเจอ: - **Language model (โมเดลภาษา):** ระบบที่เดาว่าคำถัดไปน่าจะเป็นคำอะไร จากคำที่มาก่อนหน้า หัวใจของมันคือการทำนายคำถัดไป — และแนวคิดเดียวกันนี้ยังอยู่ใน ChatGPT จนถึงทุกวันนี้ - **n-gram:** วิธีเดาคำถัดไปโดยดูจากคำก่อนหน้าแค่ไม่กี่คำ ถ้าดูคำก่อนหน้า 1 คำเรียกว่า bigram ถ้าดู 2 คำเรียกว่า trigram ยิ่งดูย้อนหลังมากก็ยิ่งเดาแม่นขึ้น แต่ก็ต้องใช้ข้อมูลมากขึ้นด้วย

หลังคำว่า "กินข้าว" มักตามด้วยอะไร

ลองดูภาพง่าย ๆ ว่าเครื่องเก็บข้อมูลแบบนี้ไว้ยังไง ตารางข้างล่างเป็นตัวอย่างสมมติที่ทำขึ้นเพื่อให้เห็นภาพเท่านั้น ไม่ใช่ตัวเลขจริงจากระบบไหน แต่หลักการตรงกับของจริง

คำก่อนหน้า	คำที่มักตามมา (เรียงจากบ่อยสุด)
กิน	ข้าว, อาหาร, ยา
กินข้าว	กับ, เสร็จ, แล้ว
ขอบคุณ	มาก, ครับ, นะ
วันนี้	อากาศ, ฝน, ดี

การทำนายคำถัดไปของโมเดลภาษาเชิงสถิติ

ลองสังเกตสองแถวบนเทียบกัน

ถ้าเครื่องเห็นแค่คำว่า "กิน" คำเดียว มันเดาว่าน่าจะตามด้วย "ข้าว" — แบบนี้คือการดูคำก่อนหน้า 1 คำ

แต่ถ้าเครื่องเห็นทั้ง "กินข้าว" สองคำ มันจะเดาเก่งขึ้น เพราะตอนนี้มันรู้แล้วว่าน่าจะตามด้วย "กับ" หรือ "เสร็จ" ซึ่งเป็นการต่อประโยคที่เป็นธรรมชาติกว่า — แบบนี้คือการดูย้อนหลัง 2 คำ

นี่แหละคือไอเดียของ n-gram ยิ่งให้เครื่องดูคำก่อนหน้าได้มากขึ้น มันก็ยิ่งเดาได้เข้าเป้าขึ้น เพราะมีบริบทให้พิจารณามากขึ้น

จากเกมในห้องนั่งเล่น สู่การแปลภาษารัฐสภาแคนาดา

ไอเดียนี้สวยในทางทฤษฎี แต่มันใช้งานจริงได้ไหม คำตอบมาจากทีมนักวิจัยกลุ่มหนึ่งที่ IBM ในช่วงปลายทศวรรษ 1980

ทีมนี้ตั้งโจทย์ท้าทายมาก — พวกเขาจะลองสร้างระบบแปลภาษาอังกฤษเป็นฝรั่งเศส โดยไม่ใส่กฎไวยากรณ์ฝรั่งเศสเข้าไปเลยแม้แต่ข้อเดียว ระบบไม่รู้ด้วยซ้ำว่าฝรั่งเศสคืออะไร มันจะแค่ "นับ" เอาว่าคำอังกฤษคำไหนมักปรากฏคู่กับคำฝรั่งเศสคำไหน

ปัญหาเดียวคือ วิธีนี้ต้องใช้ข้อความสองภาษาที่แปลตรงกันเป๊ะ ๆ จำนวนมหาศาล ซึ่งหายากมากในยุคนั้น

แล้วโชคก็เข้าข้าง ตามที่มีการเล่าไว้ นักวิจัยคนหนึ่งของ IBM ไปได้ยินบนเครื่องบินว่า บันทึกการประชุมรัฐสภาแคนาดามีอยู่ในรูปแบบดิจิทัล ทั้งภาษาอังกฤษและฝรั่งเศสคู่กัน เพราะแคนาดาใช้สองภาษาราชการ — รวมแล้วหลายล้านคู่ประโยค ม้วนเทปข้อมูลนั้นถูกส่งมาถึงห้องแล็บ และกลายเป็นวัตถุดิบชั้นดีให้เครื่องได้ "เห็นภาษา" มากพอ

ผลคือ ระบบที่ไม่รู้ไวยากรณ์ฝรั่งเศสเลยสักนิด กลับแปลได้ดีพอใช้ เพียงเพราะมันนับสถิติจากตัวอย่างจริงจำนวนมหาศาล นี่คือหลักฐานชิ้นสำคัญที่บอกว่า แนวทางสถิติใช้ได้จริง ไม่ใช่แค่ทฤษฎีสวย ๆ

"ทุกครั้งที่นักภาษาศาสตร์ออกจากทีม ระบบก็ดีขึ้น"

หัวหน้าทีมคนสำคัญในเรื่องนี้คือ Frederick Jelinek เขาเคยพูดประโยคหนึ่งที่กลายเป็นตำนานในวงการ — ทำนองว่า ทุกครั้งที่นักภาษาศาสตร์คนหนึ่งออกจากทีม ระบบรู้จำเสียงของพวกเขาก็ทำงานได้ดีขึ้น

ฟังเผิน ๆ เหมือนเขากำลังดูถูกนักภาษาศาสตร์ แต่ความจริงไม่ใช่แบบนั้นเลย Jelinek พูดประโยคนี้แบบติดตลก และหลายปีต่อมาเขายังออกมาอธิบายเองว่า เพื่อนสนิทหลายคนของเขาก็เป็นนักภาษาศาสตร์ เขาไม่ได้หมายความตามตัวอักษร

สิ่งที่ประโยคนี้สื่อจริง ๆ คือการเปลี่ยนวิธีคิดครั้งใหญ่ของยุคนั้น — ในงานบางอย่าง การให้เครื่อง "เดาจากข้อมูลจำนวนมาก" กลับได้ผลดีกว่าการให้ผู้เชี่ยวชาญ "นั่งเขียนกฎ" ทีละข้อ ไม่ใช่เพราะความรู้ภาษาไร้ค่า แต่เพราะภาษาจริงซับซ้อนเกินกว่าจะเขียนกฎให้ครบ ปล่อยให้เครื่องนับเอาจากตัวอย่างจริงเสียยังตรงกว่า

> **จำง่าย ๆ ว่า:** การเปลี่ยนจากกฎมาสู่สถิติ ไม่ใช่แค่เปลี่ยนวิธีการ แต่คือการเปลี่ยนคำถาม — จาก "กฎข้อนี้ถูกไหม" มาเป็น "รูปแบบนี้เกิดขึ้นบ่อยแค่ไหน"

ทำไมต้องเข้าใจว่ามันแค่ "เดา"

ทีนี้พอรู้แล้วว่าเครื่องทำงานแบบนี้ มันช่วยให้เราใช้ AI อย่างรู้เท่าทันขึ้นมาก

เวลา ChatGPT หรือ AI ตัวไหนตอบอะไรผิด หลายคนงงว่า "มันฉลาดขนาดนี้ ทำไมตอบเรื่องง่าย ๆ ผิดได้" คำตอบอยู่ตรงนี้แหละ — ลึกลงไปแล้ว มันไม่ได้ "เปิดดูคำตอบ" จากคลังความรู้ที่ถูกต้องแน่นอน มันกำลัง "เดา" คำถัดไปจากรูปแบบที่เคยเห็นต่างหาก

เมื่อมันเดาจากรูปแบบ มันก็เดาผิดได้เป็นธรรมดา โดยเฉพาะเวลาเจอเรื่องที่รูปแบบในข้อมูลพาไปผิดทาง บางทีคำตอบที่ "ฟังดูเข้าท่าที่สุด" กับคำตอบที่ "ถูกต้องจริง ๆ" มันคนละอันกัน และเครื่องเลือกอันแรกเสมอ

ลองคิดดู: นี่คือ "รู้" หรือ "เดา"

ครั้งหน้าที่ AI ตอบคุณ ลองถามตัวเองเร็ว ๆ ว่า

1. เรื่องนี้มีรูปแบบให้เดาง่าย ๆ ไหม (เช่น เขียนอีเมลทั่วไป สรุปย่อหน้า) — ถ้าใช่ มันมักทำได้ดี
2. หรือเรื่องนี้ต้องการความถูกต้องเป๊ะ ๆ ที่เดาไม่ได้ (เช่น ตัวเลข วันที่ ชื่อเฉพาะ กฎหมาย) — ถ้าใช่ ให้ระวังและตรวจสอบเสมอ

แค่แยกสองอย่างนี้ออก ก็ช่วยให้คุณเชื่อ AI ถูกจังหวะ และไม่โดนมันหลอกในจังหวะที่มันแค่ "เดาให้ฟังดูดี"

ของดี แต่ยังไม่พอ

ระบบสถิติแบบ n-gram นี้ยืดหยุ่นกว่ายุคกฎมาก ภาษาเปลี่ยน มีคำใหม่ มีสแลงใหม่ ก็แค่เอาข้อความใหม่ ๆ มาให้มันนับเพิ่ม ไม่ต้องนั่งเขียนกฎใหม่ทั้งระบบ ในยุค 1990 ถึง 2000 วิธีนี้จึงกลายเป็นแกนหลักของงานภาษาเกือบทั้งวงการ

แต่มันมีจุดอ่อนใหญ่อยู่อย่างหนึ่ง — มันมองย้อนหลังได้แค่ไม่กี่คำ

ลองนึกถึงประโยคยาว ๆ เช่น "แมวที่ฉันเลี้ยงไว้ตั้งแต่เด็กและพาไปหาหมอตลอด ตอนนี้มัน..." กว่าจะถึงคำว่า "มัน" เครื่องที่ดูย้อนหลังแค่ 2-3 คำ ก็ลืมไปแล้วว่าต้นประโยคพูดถึง "แมว" มันเลยไม่รู้ว่า "มัน" หมายถึงอะไร

ภาษาจริงต้องการการจำบริบทยาว ๆ แบบนี้ตลอดเวลา และนี่คือเพดานที่ระบบนับคำธรรมดา ๆ ข้ามไม่พ้น เป็นเหตุผลที่วงการต้องมองหาวิธีใหม่ในบทถัด ๆ ไป

สรุปง่าย ๆ ใน 5 ข้อ

1. แทนที่จะเขียนกฎให้เครื่อง ยุคสถิติเปลี่ยนมาให้เครื่องดูข้อความจริงเยอะ ๆ แล้วนับเอาเองว่าคำไหนมักตามคำไหน
2. หัวใจของมันคือการ "ทำนายคำถัดไป" จากความน่าจะเป็น — นี่คือต้นกำเนิดของคำว่า language model และยังเป็นแกนของ ChatGPT จนถึงทุกวันนี้

3. Claude Shannon พิสูจน์ตั้งแต่ปี 1951 ว่าภาษาไม่ได้สุ่ม มันมีรูปแบบที่ทำนายได้ และทีม IBM พิสูจน์ภายหลังว่าวิธีนับสถิติแปลภาษาได้จริงโดยไม่ต้องใช้กฎ
4. เครื่องไม่ได้ "เข้าใจ" ความหมาย มันแค่เดาคำที่มักมาด้วยกัน — นี่คือเหตุผลที่ AI เดาผิดได้ และทำไมเราต้องตรวจสอบเรื่องที่ต้องการความเป๊ะ
5. จุดอ่อนใหญ่คือมันมองบริบทย้อนหลังได้แค่ไม่กี่คำ จำเรื่องยาว ๆ ไม่ได้ — และนั่นเปิดทางให้แนวคิดใหม่ในบทถัดไป

ก่อนไปบทต่อไป

ลองสังเกตอะไรอย่างหนึ่ง

ตลอดทั้งบทนี้ เครื่องของเรานับคำเป็นก้อน ๆ มันรู้ว่า "กิน" มาคู่กับ "ข้าว" บ่อย แต่มันไม่รู้เลยว่าคำว่า "แมว" กับ "หมา" เกี่ยวข้องกัน ไม่รู้ว่าทั้งคู่เป็นสัตว์เลี้ยง ไม่รู้ว่า "กรุงเทพ" กับ "เชียงใหม่" ต่างก็เป็นจังหวัด สำหรับมันแล้ว คำแต่ละคำเป็นแค่สัญลักษณ์ที่แยกขาดจากกัน ไม่มีความหมายเชื่อมโยงกันเลย

แล้วถ้าเราหาวิธีทำให้เครื่องรู้ได้ล่ะ ว่า "แมว" กับ "หมา" อยู่ใกล้กันในแง่ความหมาย — เหมือนสองเมืองที่อยู่ใกล้กันบนแผนที่

นั่นคือก้าวต่อไปที่เปลี่ยนเกมอีกครั้ง การเปลี่ยน "คำ" ให้กลายเป็น "ตัวเลข" และเปลี่ยน "ความหมาย" ให้กลายเป็น "ระยะห่าง" บนแผนที่ — เป็นเรื่องของบทถัดไป

บทที่ 5 — เมื่อคำกลายเป็นตัวเลข และความหมายกลายเป็นระยะห่าง

ลองนึกถึงแผนที่รถไฟฟ้าในกรุงเทพดูสักครู่

ถ้ามีคนนัดคุณว่า "เจอกันแถว BTS อโศกนะ" แล้วอีกคนเสริมว่า "หรือจะนานาก็ได้" คุณรู้ทันทีโดยไม่ต้องเปิดแผนที่เลยว่าสองที่นี้อยู่ใกล้กัน เดินถึงกันได้ด้วยซ้ำ แต่ถ้ามีคนบอกว่า "อโศก" กับ "บางหว้า" คุณก็รู้สึกได้เลยว่าสองที่นี้คนละฝั่งเมือง ไกลกันมาก

สังเกตไหมว่าเราตัดสิน "ความใกล้-ไกล" ได้จากแค่ "ตำแหน่ง" บนแผนที่ ไม่ต้องรู้ด้วยซ้ำว่าแต่ละสถานีหน้าตาเป็นยังไง ขอแค่รู้ว่ามันอยู่ตรงไหน เราก็เดาความสัมพันธ์ของมันได้

ทีนี้ลองคิดเล่น ๆ ตามคำถามที่เราค้างไว้ท้ายบทก่อน — ถ้าเราวาดแผนที่แบบนี้ให้กับ "คำ" ได้ล่ะ? แผนที่ที่คำซึ่งความหมายใกล้กันถูกวางไว้ใกล้กัน และคำที่ไม่เกี่ยวกันเลยถูกวางไว้คนละมุม นี่แหละคือก้าวที่เปลี่ยนเกมในบทนี้

ปัญหาที่เราทิ้งค้างไว้

จำจุดที่เราหยุดกันไว้บทก่อนได้ไหม

ตลอดยุคสถิติ เครื่องของเราเก่งขึ้นมากในการ "ทำนายคำถัดไป" มันนับได้ว่า "กิน" มักมาคู่กับ "ข้าว" และเดาประโยคต่อได้ดีในระดับหนึ่ง แต่มันมีจุดบอดใหญ่อยู่อย่างหนึ่งที่เราชี้ไว้ — มันมองคำแต่ละคำเป็นแค่ **สัญลักษณ์ที่แยกขาดจากกัน**

แปลเป็นภาษาคนก็คือ ในสายตาเครื่องสมัยนั้น คำว่า "แมว" เป็นแค่หมายเลขลำดับหนึ่งในพจนานุกรม สมมติว่าเป็นคำที่ 4,521 ส่วน "หมา" เป็นคำที่ 9,832 และ "ตู้เย็น" เป็นคำที่ 17,044 ตัวเลขพวกนี้เป็นแค่ป้ายชื่อ มันไม่ได้บอกอะไรเลยว่าคำไหนเกี่ยวกับคำไหน

ผลคือ ทางคณิตศาสตร์แล้ว ระยะห่างระหว่าง "แมว" กับ "หมา" เท่ากับระยะห่างระหว่าง "แมว" กับ "ตู้เย็น" เป๊ะ ๆ เครื่องไม่มีทางรู้เลยว่าแมวกับหมาต่างก็เป็นสัตว์เลี้ยงสี่ขา ส่วนตู้เย็นเป็นคนละโลกกัน

นี่คือกำแพงที่เราต้องข้าม — จะทำยังไงให้เครื่อง "รู้สึก" ได้ว่าแมวกับหมาอยู่ใกล้กัน

ความคิดเก่าแก่จากนักภาษาศาสตร์สองคน

เรื่องน่าทึ่งคือ กุญแจที่ไขปัญหานี้ ไม่ได้มาจากวิศวกรคอมพิวเตอร์ แต่มาจากนักภาษาศาสตร์ — และเก่าแก่กว่าที่คุณคิดมาก

ปี 1954 นักภาษาศาสตร์ชาวอเมริกันชื่อ Zellig Harris เสนอแนวคิดหนึ่งในบทความชื่อ "Distributional Structure" ใจความว่า คำที่มักปรากฏในบริบทคล้าย ๆ กัน ก็มักจะมีความหมายเกี่ยวข้องกัน อีกสามปีต่อมา

ปี 1957 นักภาษาศาสตร์ชาวอังกฤษชื่อ J.R. Firth สรุปแนวคิดเดียวกันนี้ด้วยประโยคที่ติดหูจนกลายเป็นตำนาน — ในเวอร์ชันที่คนชอบยกมากันว่า:

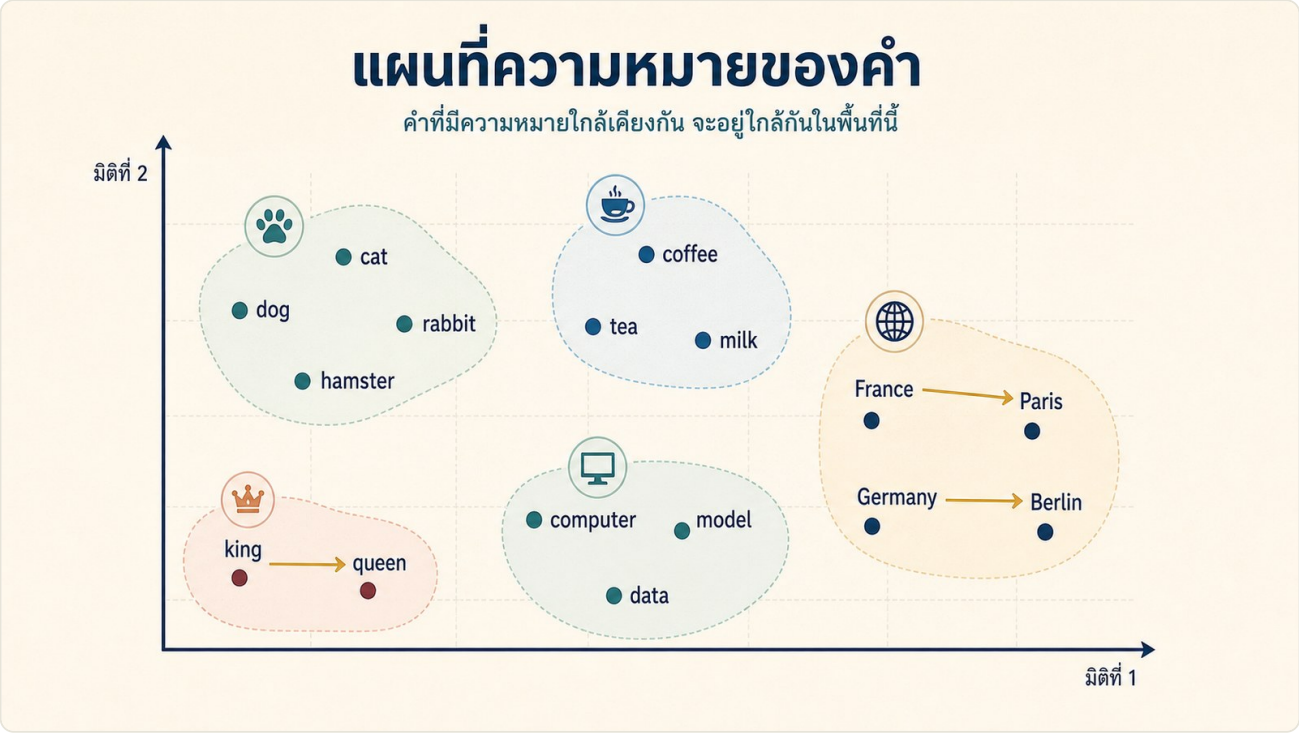
> "You shall know a word by the company it keeps" — คุณจะรู้จักคำหนึ่งได้ ด้วยการดูว่าคำนั้นคบหากับใคร

ฟังดูเป็นนามธรรม แต่จริง ๆ แล้วคุณใช้หลักนี้อยู่ทุกวันโดยไม่รู้ตัว

ลองนึกว่าคุณเจอคำว่า "บาริสต้า" เป็นครั้งแรกในชีวิต แล้วเห็นมันโผล่มาในประโยคพร้อมกับ "กาแฟ" "เอสเพรสโซ่" "ลาเต้" "คาเฟ่" "ชงเครื่อง" — ถึงคุณจะไม่เคยเปิดพจนานุกรมเลย คุณก็เดาออกแทบจะทันทีว่าบาริสต้าน่าจะเป็นคนที่ทำกาแฟ คุณรู้จักคำนี้จาก "เพื่อนที่มันคบ" ล้วน ๆ

นี่แหละคือไอเดียทั้งหมด ในยุคของ Harris กับ Firth ยังไม่มีคอมพิวเตอร์ที่จะเอาแนวคิดนี้ไปคำนวณกับข้อความจำนวนมหาศาลได้ มันเลยเป็นแค่ข้อสังเกตทางภาษาศาสตร์อยู่หลายสิบปี รอวันที่เทคโนโลยีจะตามมาทัน

เปลี่ยนคำให้เป็นพิกัด



แผนที่ความหมายของ word embeddings: คำใกล้กัน = ความหมายใกล้กัน

วันนั้นมาถึงในเดือนมกราคม ปี 2013

นักวิจัยชื่อ Tomáš Mikolov กับทีมที่ Google เผยแพร่เครื่องมือชื่อ **word2vec** ที่ทำสิ่งที่ Firth พูดไว้ให้เป็นจริงในสเกลมหึมา — มันอ่านข้อความจำนวนมหาศาล (ในการทดลองหนึ่งใช้ถึงราว 1,600 ล้านคำ) แล้วจับว่าคำไหนชอบปรากฏใกล้คำไหน จากนั้นก็แปลงคำแต่ละคำให้กลายเป็น "พิกัด" บนแผนที่ความหมาย

ตรงนี้คือหัวใจ ขอหยุดอธิบายคำสำคัญสักนิด

> **ศัพท์ที่เพิ่งเจอ — embedding** *embedding* คือการแปลงคำให้กลายเป็นชุดตัวเลขที่ทำหน้าที่เหมือน "พิกัด" ของคำนั้นบนแผนที่ความหมาย พูดง่าย ๆ คือ แทนที่จะให้ "แมว" เป็นแค่ป้ายชื่อโดด ๆ เราให้มันมี "ที่อยู่" บนแผนที่ และคำที่ความหมายคล้ายกันจะได้ที่อยู่ใกล้กัน

ทำไมต้องเป็นตัวเลข? เพราะลึกลงไป คอมพิวเตอร์ทำได้แค่คำนวณตัวเลข มันไม่เข้าใจตัวอักษร "แ-ม-ว" แต่ถ้าเราเปลี่ยนทุกคำให้เป็นพิกัดที่บวกลบเทียบระยะกันได้ จู่ ๆ เครื่องก็เริ่มทำเรื่องที่เคยทำไม่ได้เลย — มันวัด "ความใกล้ทางความหมาย" ออกมาเป็นตัวเลขได้

และพอวาดผลลัพธ์ออกมา ภาพที่ได้ก็น่าทึ่ง คำว่า "cat" "dog" "rabbit" "hamster" ไปกองรวมกันอยู่เป็นกลุ่ม เพราะในข้อความจริงพวกมันปรากฏในบริบทเดียวกันบ่อยมาก (อาหารสัตว์เลี้ยง โรงพยาบาลสัตว์ พาไปเดินเล่น) โดยที่ไม่มีมนุษย์คนไหนไปบอกเครื่องเลยว่า "พวกนี้คือสัตว์เลี้ยงนะ" เครื่องค้นพบหมวดหมู่นี้เองล้วน ๆ จากการดูว่าคำไหนคบกับใคร

> **จำง่าย ๆ ว่า:** ก่อน word2vec เครื่องรู้จัก "แมว" แค่ในฐานะหมายเลขในพจนานุกรม หลัง word2vec เครื่องรู้ว่า "แมว" อยู่ย่านเดียวกับ "หมา" และไกลจาก "รถบัส" — โดยไม่มีใครสอนกฎข้อนี้ให้มันเลย

เมื่อความสัมพันธ์กลายเป็น "ทิศทาง"

ทีนี้มาถึงส่วนที่ทำให้คนทั้งวงการตื่นเต้นที่สุด

ปลายปีเดียวกัน Mikolov ร่วมกับ Wen-tau Yih และ Geoffrey Zweig นำเสนองานอีกชิ้นที่งานประชุม NAACL พวกเขาพบว่า บนแผนที่ความหมายนี้ "ความสัมพันธ์" ระหว่างคำกลายเป็น "ทิศทาง" ที่จับต้องได้

ลองนึกภาพแบบนี้ บนแผนที่ ทิศทางจากคำว่า "France" ไปยัง "Paris" (คือทิศจากประเทศไปเมืองหลวง) ขนานกันพอดีกับทิศทางจาก "Germany" ไปยัง "Berlin" ราวกับว่าแนวคิดเรื่อง "เมืองหลวงของประเทศนี้" ถูกเก็บไว้เป็น "ทิศทางเดินบนแผนที่" ที่ใช้ซ้ำได้กับทุกคู่ประเทศ

แต่ตัวอย่างที่โด่งดังที่สุด — โด่งดังจนกลายเป็นภาพจำของวงการ — คือเรื่องนี้:

king – man + woman ≈ queen

แปลเป็นภาษาคนคือ ถ้าเอาพิกัดของคำว่า "king" (ราชา) ลบทิศของความเป็น "man" (ผู้ชาย) ออก แล้วบวกทิศของความเป็น "woman" (ผู้หญิง) เข้าไป พิกัดที่ได้จะไปตกใกล้คำว่า "queen" (ราชินี) พอดี ราวกับเครื่องรู้ว่า "ราชาที่เป็นผู้หญิง" คือ "ราชินี" ทั้งที่ไม่มีใครสอนมันเรื่องเพศหรือยศเลยสักนิด

ฟังดูเหมือนเวทมนตร์ และมันก็เป็นเรื่องจริงที่เกิดขึ้นได้ แต่...

ความจริงที่คนมักไม่ค่อยเล่าต่อ

ตัวอย่าง king−man+woman≈queen ถูกเล่าซ้ำจนหลายคนเชื่อว่า word2vec "เข้าใจความหมาย" ของคำแล้วจริง ๆ ความจริงซับซ้อนกว่านั้น และเป็นเรื่องที่ควรรู้ไว้

ปี 2019 ทีมนักวิจัยนำโดย Malvina Nissim ตรวจสอบตัวอย่างพวกนี้อย่างจริงจัง แล้วพบความลับเล็ก ๆ ที่ซ่อนอยู่เบื้องหลังเดโมสวย ๆ — เวลาคำนวณ "king − man + woman" จริง ๆ คำที่อยู่ใกล้ผลลัพธ์ที่สุดมักไม่ใช่ "queen" แต่เป็น "king" ตัวมันเอง!

ทำไมถึงเป็นแบบนั้น? เพราะ "king" ใกล้กับ "king" อยู่แล้วโดยธรรมชาติ การลบ "man" บวก "woman" แค่ขยับมันไปนิดเดียว มันเลยยังวนกลับมาใกล้ตัวเองมากที่สุด เดโมที่เราเห็นถึงผลออกมาสวยงาม เพราะคนทำ **ตัดคำตั้งต้นทั้งสาม (king, man, woman) ออกจากรายการคำตอบที่เป็นไปได้ก่อน** พอตัดสามคำนี้ทิ้ง คำที่เหลือซึ่งใกล้ที่สุดถึงค่อยกลายเป็น "queen" และถ้าไม่ตัด ความแม่นยำของตัวอย่างทำนองนี้ก็ตกลงอย่างรุนแรง

แล้วเราควรสรุปยังไงดี?

ไม่ใช่ว่าตัวอย่างนี้ "หลอก" — ทิศทางความสัมพันธ์บนแผนที่มีอยู่จริง โดยเฉพาะกับคำที่มีความหมายชัดเจนอย่างประเทศกับเมืองหลวง แต่มันไม่ใช่ "กฎ" ที่แม่นยำเป๊ะทุกครั้ง มันเป็น "แนวโน้ม" ที่บางทีก็สวยงาม บางทีก็เพี้ยน

ของจริงซับซ้อนกว่านี้ แต่หลักการเรื่อง "ความสัมพันธ์มีทิศทางบนแผนที่" ยังเป็นจริง และนั่นคือสิ่งที่สำคัญที่ควรจำ ไม่ใช่ตัวเดโมที่ถูกคัดมาอย่างดี

word2vec ไม่ใช่จุดจบของเรื่องนี้ ปีถัดมา (2014) ทีมจาก Stanford นำโดย Jeffrey Pennington, Richard Socher และ Christopher Manning ก็เผยแพร่วิธีทำแผนที่ความหมายอีกแบบชื่อ **GloVe** ที่ดูภาพรวมของทั้งคลังข้อความแทนที่จะดูทีละหน้าต่างแคบ ๆ รายละเอียดวิธีต่างกัน แต่เป้าหมายเดียวกัน — และที่สำคัญ ทั้งสองทีมแจกแผนที่ที่ฝึกเสร็จแล้วให้ใช้ฟรี ทำให้แนวคิด embedding แพร่ไปทั่ววงการอย่างรวดเร็ว

"อยู่ใกล้กัน" ไม่ได้แปลว่า "เหมือนกัน"

คำตรงข้ามอาจอยู่ใกล้กันเพราะใช้ในบริบทคล้ายกัน

ก่อนจะไปต่อ มีความเข้าใจผิดหนึ่งที่ต้องเคลียร์ เพราะมันบอกอะไรลึก ๆ เกี่ยวกับวิธีคิดของเครื่อง

ลองทายดู — บนแผนที่ความหมาย คำว่า "ร้อน" กับ "เย็น" อยู่ใกล้กันหรือไกลกัน?

หลายคนคงตอบว่าไกล เพราะมันเป็นคำตรงข้ามกัน แต่คำตอบคือ มันมัก **อยู่ใกล้กันมาก**

เพราะอะไร? เพราะสองคำนี้ปรากฏในบริบทคล้ายกันแทบจะทุกครั้ง — "น้ำร้อน/น้ำเย็น" "อากาศร้อน/อากาศเย็น" "อาหารร้อน/อาหารเย็น" ในเมื่อ "เพื่อนที่มันคบ" เป็นกลุ่มเดียวกันเกือบหมด เครื่องก็วางมันไว้ย่านเดียวกัน

นี่เตือนเราว่า "อยู่ใกล้กัน" บนแผนที่ความหมาย ไม่ได้แปลว่า "ความหมายเหมือนกัน" แต่แปลว่า "ปรากฏในบริบทคล้ายกัน" — ซึ่งบ่อยครั้งก็ใกล้เคียงกัน แต่ไม่ใช่สิ่งเดียวกัน และนี่คือสัญญาณแรกที่บอกเราว่า ต่อให้แผนที่นี้ทรงพลังแค่ไหน มันก็ยังไม่ใช่ "ความเข้าใจ" แบบที่มนุษย์มี

จุดที่แผนที่นี้ยังไปไม่ถึง

มาถึงข้อจำกัดใหญ่ที่สุด ที่จะเป็นโจทย์ให้บทต่อ ๆ ไป

จำเรื่องคำกำกวมในบทแรก ๆ ได้ไหม? คำว่า "ตา" ที่หมายถึงได้ทั้งอวัยวะที่ใช้มอง คุณตาที่เป็นญาติผู้ใหญ่ และตาข่ายดักปลา หรือคำว่า "ขัน" ที่เป็นได้ทั้งภาชนะตักน้ำ การหัวเราะขำขัน และการขันนอตให้แน่น

ปัญหาคือ ในโลกของ embedding **หนึ่งคำได้พิกัดเดียวเท่านั้น**

คำว่า "ตา" ทุกความหมายถูกยัดให้อยู่ในจุดเดียวบนแผนที่ จุดนั้นเลยกลายเป็นเหมือน "จุดกึ่งกลาง" ที่เฉลี่ยทุกความหมายปนกัน — ไม่ใช่ตำแหน่งของอวัยวะ ไม่ใช่ตำแหน่งของคุณตา ไม่ใช่ตำแหน่งของตาข่าย แต่เป็นที่ไหนสักแห่งตรงกลางที่ไม่ตรงกับความหมายไหนเต็ม ๆ เลย

เหมือนถ้ามีคนถามว่าคุณอยู่ที่ไหน แล้วคุณตอบว่า "อยู่กึ่งกลางระหว่างบ้านกับที่ทำงาน" — มันไม่ผิด แต่มันไม่ใช่ที่ไหนเลยจริง ๆ

ที่สำคัญคือ embedding ตัดสินพิกัดของคำ **ก่อน** ที่จะเห็นประโยค มันจึงแยกไม่ออกว่า "ตา" ในประโยค "ตาของฉันเจ็บ" กับ "ตาของฉันแก่แล้ว" หมายถึงคนละอย่าง เพราะทั้งสองประโยคมันใช้พิกัดของ "ตา" จุดเดียวกันเป๊ะ

พูดอีกแบบคือ แผนที่นี้รู้จักคำแบบ "ลอย ๆ ในสุญญากาศ" — รู้ว่าโดยทั่วไปคำนี้คบกับใคร แต่ยังไม่รู้ว่า "ในประโยคนี้ ตอนนี้ คำนี้หมายถึงอะไร" และนั่นแหละคือกำแพงถัดไปที่รอให้ข้าม

แล้วมันเข้าใจจริงไหม

กลับมาที่คำถามแกนกลางของหนังสือเราสักครู่ — embedding ทำให้เครื่อง "เข้าใจ" ภาษาจริงไหม หรือยังแค่เดาเก่งขึ้นเฉย ๆ

คำตอบที่ซื่อสัตย์คือ มันก้าวมาใกล้กว่าเดิมหนึ่งก้าวสำคัญ แต่ยังไม่ใช่ความเข้าใจ

word2vec ไม่ได้ "รู้" ว่าราชาคืออะไร ไม่ได้รู้ว่าแมวมีขนนุ่ม มันแค่คำนวณว่าคำไหนปรากฏใกล้คำไหนในข้อความนับพันล้านคำ แล้วแปลงสถิตินั้นเป็นพิกัด สิ่งที่เราเรียกว่า "ความหมาย" ในมุมของเครื่อง จริง ๆ แล้วคือ **ตำแหน่งและระยะห่างบนแผนที่** ไม่ใช่คำนิยามแบบในพจนานุกรม และไม่ใช่ความรู้เกี่ยวกับโลกจริงแบบที่เรามี

ถ้าข้อมูลที่ป้อนเข้าไปมีอคติ แผนที่ก็มีอคติตาม ถ้าข้อมูลพูดถึงคำสองคำในบริบทคล้ายกัน แผนที่ก็วางมันไว้ใกล้กัน ไม่ว่ามันจะ "ควร" ใกล้กันจริงไหมก็ตาม นี่คือทั้งพลังและขีดจำกัดของแนวคิดนี้ในประโยคเดียว

> **อีกมุมหนึ่งที่ควรรู้:** แผนที่ที่เราเล่ามาทั้งบทเป็นภาพ 2 มิติเพื่อให้นึกภาพง่าย แต่แผนที่ความหมายจริงมีหลายร้อยมิติ — มากเกินกว่าสมองคนจะวาดออกมาได้ ไม่ต้องไปจินตนาการมันออกหรอก หลักการเหมือนเดิมเป๊ะ คือ "คำใกล้กัน = ความหมายเกี่ยวกัน, ความสัมพันธ์ = ทิศทาง" แค่เกิดในพื้นที่ที่มีมิติมากกว่าแผ่นกระดาษมหาศาล

สรุปง่าย ๆ ใน 5 ข้อ

1. ก่อนหน้านี้เครื่องมองคำเป็นแค่สัญลักษณ์แยกขาด ไม่รู้ว่า "แมว" กับ "หมา" เกี่ยวกัน — embedding แก้ปัญหานี้ด้วยการเปลี่ยนคำให้เป็น "พิกัด" บนแผนที่ความหมาย

2. หลักการมาจากนักภาษาศาสตร์ตั้งแต่ยุค 1950s (Harris 1954, Firth 1957): "คุณจะรู้จักคำหนึ่งได้ ด้วยการดูว่ามันคบกับใคร" คำที่ปรากฏในบริบทคล้ายกันจะถูกวางไว้ใกล้กัน
3. word2vec (2013) ทำแนวคิดนี้ให้เป็นจริงในสเกลมหาศาล จนความสัมพันธ์ของคำกลายเป็น "ทิศทาง" บนแผนที่ (เช่น France→Paris ขนานกับ Germany→Berlin)
4. ตัวอย่าง king−man+woman≈queen เป็นเรื่องจริงที่น่าทึ่ง แต่ใช้ได้ก็ต่อเมื่อตัดคำตั้งต้นออกก่อน — เป็นแนวโน้ม ไม่ใช่กฎตายตัว
5. ข้อจำกัดใหญ่: หนึ่งคำได้พิกัดเดียว เครื่องจึงยังแยกไม่ออกว่า "ตา/ขัน" ในแต่ละประโยคหมายถึงอะไร เพราะมันตัดสินพิกัดก่อนเห็นบริบท

ก่อนไปบทต่อไป

ตลอดบทนี้เราพูดถึง "แผนที่ความหมาย" ราวกับว่ามันมีอยู่แล้ว แต่มีคำถามหนึ่งที่เราจงใจข้ามไป

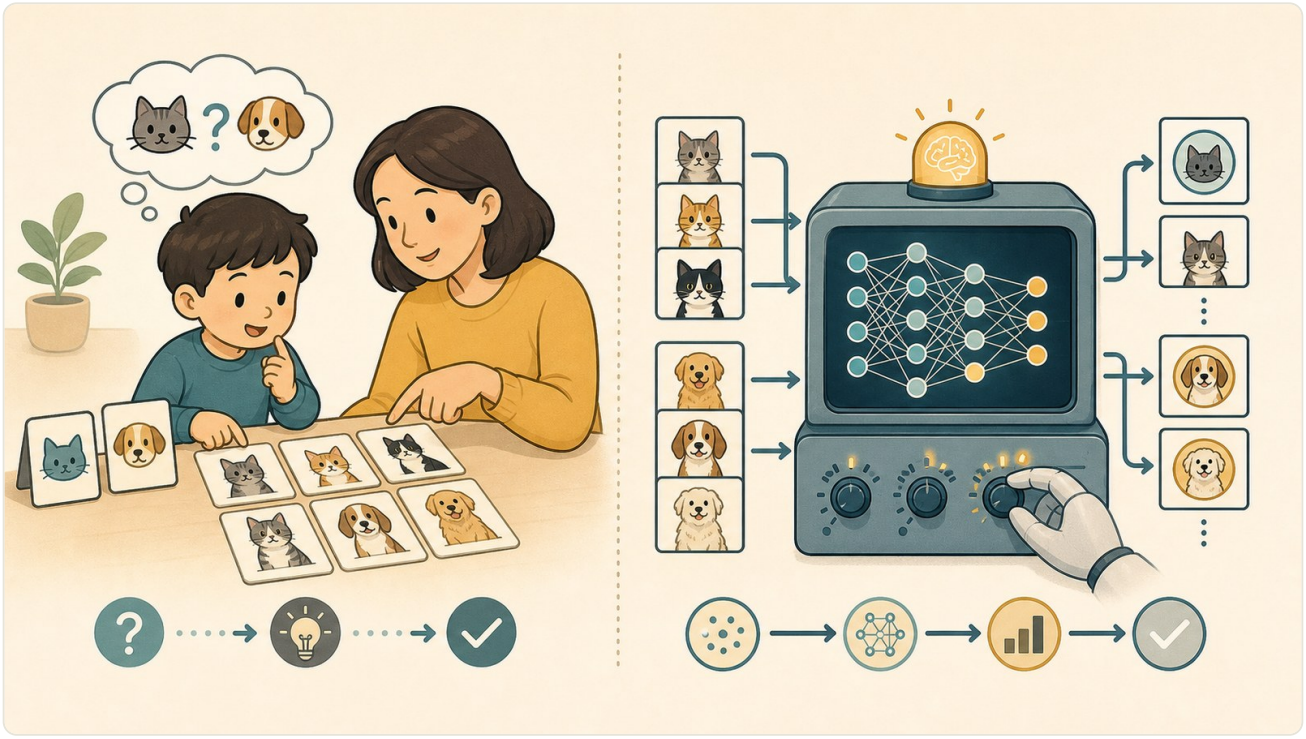
แล้วเครื่อง **หา** พิกัดพวกนี้มาจากไหน? ใครเป็นคนบอกว่า "แมว" ควรอยู่พิกัดนี้ "หมา" ควรอยู่ตรงนั้น? คำตอบคือ — ไม่มีใครบอกเลย ไม่มีมนุษย์คนไหนนั่งกรอกพิกัดให้ทีละคำ เพราะถ้าทำแบบนั้นเราก็ย้อนกลับไปสู่ปัญหา "เขียนกฎไม่มีวันครบ" ของยุคแรกอีก

ความจริงคือ เครื่องเริ่มจากการสุ่มวางคำทุกคำไว้มั่ว ๆ บนแผนที่ แล้วค่อย ๆ **ขยับเอง** ทีละนิด ทุกครั้งที่มันเดาผิด มันก็ปรับตำแหน่งใหม่ ลองผิดลองถูกซ้ำแล้วซ้ำเล่านับล้านครั้ง จนคำต่าง ๆ ไหลไปอยู่ในตำแหน่งที่เข้าท่าขึ้นเรื่อย ๆ ด้วยตัวมันเอง

กลไกการ "เรียนรู้ด้วยการลองผิดลองถูกแล้วปรับตัวเอง" นี้ ไม่ได้ใช้แค่กับการวาดแผนที่คำเท่านั้น มันคือหัวใจของสิ่งที่เรียกว่า **โครงข่ายประสาทเทียม** ซึ่งจะเปลี่ยนทุกอย่างในเรื่องราวของเรา — และเป็นเรื่องของบทถัดไป

บทที่ 6 — โครงข่ายประสาทเทียม กับการเรียนรู้จากตัวอย่าง

ลองนึกถึงตอนที่เด็กเล็ก ๆ คนหนึ่งหัดแยกแมวกับหมา



เรียนจากตัวอย่าง เหมือนเด็กค่อย ๆ หัดแยกแมวกับหมา

เราไม่เคยหยิบคู่มือมาอ่านให้เขาฟังว่า "แมวต้องมีหูสามเหลี่ยมชี้ขึ้น น้ำหนักไม่เกินเท่านั้นกิโล หางต้องงอนได้ เสียงร้องประมาณนี้" จริงไหม? เราแค่ชี้ไปที่รูปแล้วพูดว่า "อันนี้แมวนะ" ชี้อีกอันบอกว่า "อันนั้นหมานะ" ทำซ้ำสิบครั้ง ยี่สิบครั้ง สามสิบครั้ง

แล้ววันหนึ่ง เด็กก็ชี้แมวตัวที่เขาไม่เคยเห็นมาก่อนได้ถูก ทั้งที่เราไม่เคยบอกกฎสักข้อ

เขาไม่ได้ท่องกฎ เขา "จับทาง" ได้เองจากตัวอย่าง

และนี่แหละคือกลไกเดียวกันเป๊ะ กับสิ่งที่เราทิ้งค้างไว้ตอนจบบทที่แล้ว — จำได้ไหมว่าเครื่องไม่ได้มีใครมานั่งกรอกพิกัดของคำให้ มันเริ่มจากสุ่มวางคำมั่ว ๆ บนแผนที่ความหมาย แล้วค่อย ๆ ขยับเองทุกครั้งที่เดาผิด ลองผิดลองถูกซ้ำแล้วซ้ำเล่าจนเข้าที่เข้าทาง

กลไก "เรียนเองจากตัวอย่าง ด้วยการเดา-เช็ก-ปรับ" นี้ มีชื่อเรียกว่า **โครงข่ายประสาทเทียม** และมันคือหัวใจที่อยู่เบื้องหลังทุกอย่างใน ChatGPT — บทนี้เราจะมาทำความรู้จักมันให้ชัด ๆ โดยไม่ต้องแตะคณิตศาสตร์สักตัว

> **ศัพท์ที่เพิ่งเจอ: โครงข่ายประสาทเทียม (neural network)** = ระบบคอมพิวเตอร์ที่เรียนรู้จากตัวอย่างเอง โดยเริ่มจากการเดา แล้วค่อย ๆ ปรับตัวเองทุกครั้งที่เดาผิด จนทำได้ดีขึ้นเรื่อย ๆ — ชื่อมันมาจากการได้แรงบันดาลใจ (อย่างหยาบ ๆ) จากการเชื่อมโยงของเซลล์สมอง

ปัญหาที่ซ่อนอยู่ในคำว่า "ก็แค่สอนกฎให้คอม"

ย้อนกลับไปสองบทก่อน เราเล่าถึงยุคที่มนุษย์พยายามเขียนกฎทุกข้อให้เครื่อง แล้วมันพังเพราะภาษาจริงมีข้อยกเว้นนับไม่ถ้วน เขียนกฎเท่าไหร่ก็ไม่มีวันครบ

ทีนี้ลองคิดเล่น ๆ ว่า ถ้าเราจะเขียนโปรแกรมให้คอมพิวเตอร์ดูรูปแล้วบอกว่า "นี่คือแมว" เราต้องเขียนกฎอะไรบ้าง?

"ถ้ามีหูสองข้าง..." — แต่หมาก็มีหูสองข้าง "ถ้ามีขนสีส้ม..." — แต่แมวมีตั้งหลายสี และหมาบางตัวก็สีส้ม "ถ้ามีหนวด..." — แมวในรูปบางรูปก็มองไม่เห็นหนวด

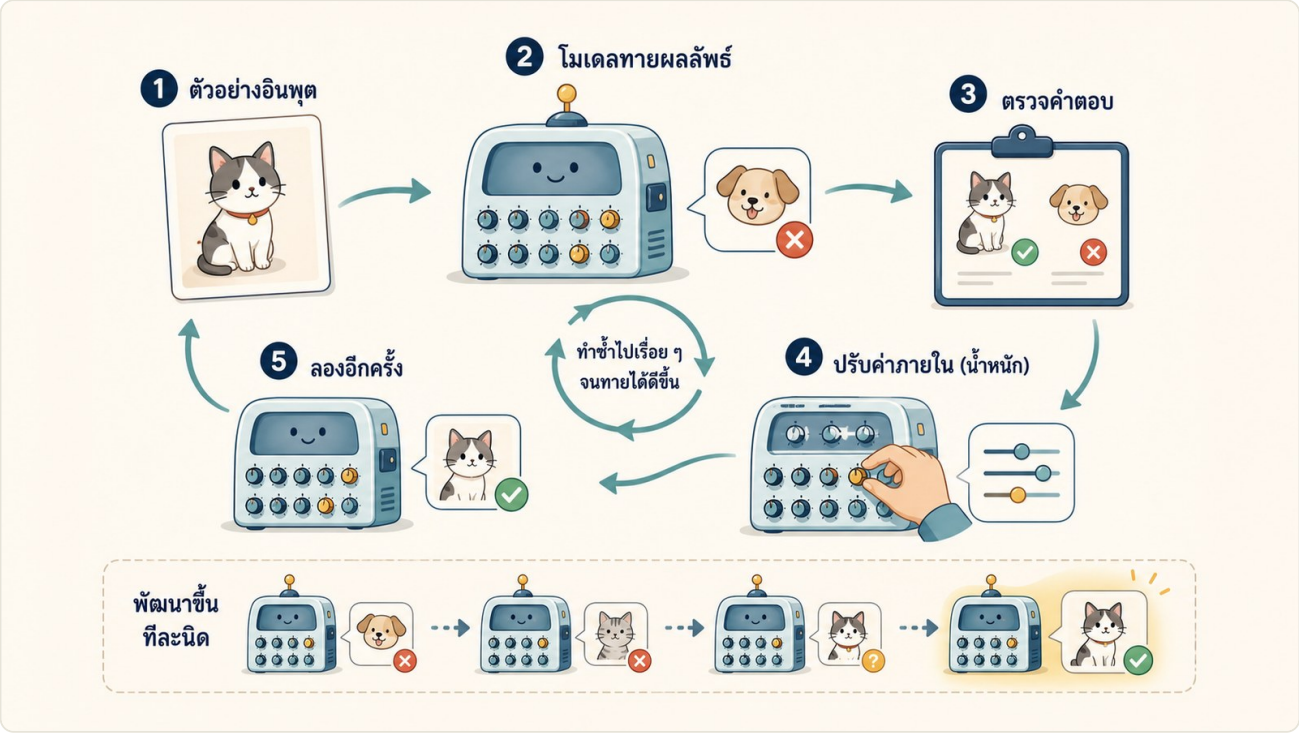
ยิ่งเขียน ยิ่งงอกข้อยกเว้น เหมือนเดิมเป๊ะ และนั่นยังเป็นแค่ "แมวในรูปที่ถ่ายจากด้านหน้า แสงดี ๆ" เท่านั้น ลองนึกถึงแมวที่หันข้าง แมวที่ม้วนตัวนอน แมวที่อยู่ในเงามืดดูสิ กฎที่เราอุตส่าห์เขียนก็ใช้ไม่ได้แล้ว

ปัญหาคือ สิ่งที่ "ทำให้แมวเป็นแมว" มันลื่นไหลเกินกว่าจะมัดด้วยถ้อยคำ — เรารู้ว่ามันคือแมวเมื่อเห็น แต่บอกไม่ถูกว่ารู้ได้ยังไง

แล้วถ้าเราบอกเครื่องเป็นคำพูดไม่ได้ เราจะสอนมันยังไง?

คำตอบก็เหมือนที่เราสอนเด็ก — เลิกบอกกฎ แล้วให้มันดูตัวอย่างเยอะ ๆ แทน

ความคิดที่พลิกทุกอย่าง: เดา เช็ก แล้วปรับ



วงจรการเรียนรู้ของ neural network: เดา เช็ก แล้วปรับ

หัวใจของโครงข่ายประสาทเทียมเรียบง่ายอย่างน่าตกใจ ขอเล่าเป็นขั้น ๆ แบบไม่มีสูตรเลย

ตอนเริ่มต้น เครื่อง "โง่" สนิท มันยังไม่รู้อะไรทั้งนั้น เราโยนรูปแมวให้มันดู มันก็เดามั่ว ๆ ว่า "หมา"

เราบอกว่า "ผิด อันนี้แมว"

เครื่องก็ปรับค่าภายในตัวเองนิดหน่อย — ขยับให้ครั้งหน้ามีโอกาสตอบ "แมว" มากขึ้นอีกนิดเมื่อเจอรูปคล้าย ๆ นี้

แล้วเราก็โยนรูปต่อไป มันเดา เราบอกถูกหรือผิด มันปรับอีกนิด โยนรูปต่อไป เดา เช็ก ปรับ — วนแบบนี้ไม่ใช่สิบครั้งหรือร้อยครั้ง แต่เป็นล้านครั้ง

ทีละนิด ทีละนิด ค่าภายในของมันก็เข้าที่เข้าทางขึ้นเรื่อย ๆ จนวันหนึ่งมันดูรูปแมวที่ไม่เคยเห็นมาก่อนแล้วตอบ "แมว" ได้ถูกเป็นส่วนใหญ่

นี่คือภาพเดียวกับการนุ่งคำบนแผนที่ความหมายในบทที่แล้วเป๊ะ ๆ — สุ่มก่อน ผิดแล้วขยับ ทำซ้ำจนเข้าที่

ปุ่มจูนนับล้านปุ่ม

"ค่าภายใน" ที่เครื่องคอยปรับนี้ มีชื่อทางเทคนิคว่า **น้ำหนัก (weights)** แต่เราไม่ต้องจำคำนี้ก็ได้ ขอให้นึกภาพแบบนี้แทน

ลองนึกถึงวิทยุเครื่องเก่าที่มีปุ่มหมุนปรับคลื่น ปกติมันมีปุ่มเดียวหรือสองปุ่ม เราหมุนจนเสียงชัด

ทีนี้ลองนึกว่า แทนที่จะมีปุ่มเดียว เครื่องนี้มีปุ่มนับล้านปุ่มเรียงกันเต็มไปหมด โครงข่ายประสาทเทียมก็คือระบบที่ค่อย ๆ หมุนปุ่มทุกปุ่มพร้อมกันทีละนิด ทุกครั้งที่ตอบผิด มันจะรู้ว่าควรหมุนปุ่มไหนไปทางไหน เพื่อให้ครั้งหน้าเดาแม่นขึ้น

หมุนปุ่มล้านปุ่มซ้ำเป็นล้านรอบ จนสุดท้ายเสียงที่ออกมา "ชัด" — นั่นคือตอนที่เครื่องเก่งแล้ว

(ปุ่มพวกนี้จริง ๆ เป็นแค่ตัวเลขในหน่วยความจำคอมพิวเตอร์นะ ไม่ใช่ปุ่มจับต้องได้ แต่หลักการเหมือนกัน)

ส่วนเทคนิคที่ทำให้เครื่องรู้ว่า "ตอนตอบผิดครั้งนี้ ควรหมุนปุ่มไหนบ้าง" ทั้ง ๆ ที่มีตั้งล้านปุ่ม มีชื่อเรียกว่า **backpropagation** เราจะไม่ลงรายละเอียดมัน ขอให้รู้แค่ว่ามันคือ "กลเม็ดที่ทำให้เครื่องเรียนจากความผิดพลาดของตัวเองได้ แม้จะมีหลายชั้นซ้อนกัน" ก็พอ — แล้วเดี๋ยวเราจะพูดถึงคำว่า "หลายชั้น" กัน

> **จำง่าย ๆ ว่า:** กฎบอกเครื่องว่า "ต้องทำอะไร" ส่วนตัวอย่างบอกเครื่องแค่ว่า "อันไหนถูก อันไหนผิด" — ที่เหลือเครื่องไปหาเอาเองว่าต้องปรับตัวยังไงถึงจะถูกบ่อยขึ้น

"Deep" แปลว่าลึก และลึกคือมีหลายชั้น

คุณคงเคยได้ยินคำว่า "deep learning" มาบ่อย ทีนี้เรามาดูกันว่า "deep" (ลึก) มันลึกตรงไหน

โครงข่ายประสาทเทียมไม่ได้ทำงานทีเดียวจบ แต่ส่งข้อมูลผ่านเป็น "ชั้น ๆ" (layers) ลองนึกถึงการดูรูปแมวแบบนี้

ชั้นแรกสุด มองเห็นแค่ของง่าย ๆ — เส้น ขอบ จุดสีอ่อนสีเข้ม

ชั้นถัดมารับงานต่อ เอาเส้นกับขอบมาประกอบเป็นรูปทรง — วงกลม สามเหลี่ยม ส่วนโค้ง

ชั้นถัดไปอีก เอารูปทรงมารวมเป็นชิ้นส่วน — นี่ดูเหมือนหู นี่เหมือนตา นี่เหมือนหนวด

ชั้นบน ๆ เอาชิ้นส่วนทั้งหมดมารวมแล้วสรุปว่า "เออ ทั้งหมดนี้รวมกันแล้วคือแมว"

แต่ละชั้นต่อยอดจากชั้นก่อนหน้า จับสิ่งที่ซับซ้อนขึ้นเรื่อย ๆ ยิ่งมีชั้นเยอะ ("ยิ่งลึก") เครื่องก็ยิ่งเรียนรู้สิ่งที่ซับซ้อนได้ — และนั่นคือที่มาของคำว่า "deep learning" มันไม่ใช่คำโฆษณาหรู ๆ แต่หมายถึง "โครงข่ายที่มีหลายชั้นจริง ๆ" ตามตัวอักษร

> **ศัพท์ที่เพิ่งเจอ: deep learning (การเรียนรู้เชิงลึก)** = การใช้โครงข่ายประสาทเทียมที่มี "หลายชั้น" ซ้อนกัน แต่ละชั้นจับรูปแบบที่ซับซ้อนขึ้นทีละขั้น จึงเรียนรู้สิ่งที่ยากและละเอียดได้

ชายที่เดินสวนกระแสอยู่คนเดียว

เรื่องนี้ฟังดูเข้าท่า แล้วทำไมโลกถึงเพิ่งมาตื่นเต้นกับมันไม่กี่ปีมานี้?

คำตอบคือไอเดียนี้ไม่ใช่ของใหม่เลย มันเก่าพอ ๆ กับคอมพิวเตอร์ยุคแรก ๆ และมีคน ๆ หนึ่งที่ยืนเฝ้ามันอยู่เกือบทั้งชีวิต ทั้งที่ทั้งวงการเดินจากไปแล้ว — เรื่องของเขาคือเส้นเลือดใหญ่ของบทนี้

ย้อนไปกรกฎาคม ปี 1958 ที่ห้องทดลองของมหาวิทยาลัย Cornell นักจิตวิทยาชื่อ Frank Rosenblatt เปิดตัวสิ่งที่เขาเรียกว่า "perceptron" — เครื่องที่เรียนรู้จากตัวอย่างได้ ไม่ใช่ทำตามกฎที่มนุษย์เขียน มันยังทำได้แค่งานง่าย ๆ อย่างแยกการ์ดซ้ายขวา แต่ในยุคที่คอมพิวเตอร์ทุกตัวเอาแต่ "รอรับคำสั่ง" เครื่องที่ "เรียนเองได้" คือสิ่งที่ไม่เคยมีมาก่อน

สื่อในยุคนั้นตื่นเต้นกันสุดขีด หนังสือพิมพ์ New York Times พาดหัวว่า "เครื่องมือใหม่ของกองทัพเรือเรียนรู้ด้วยการลงมือทำ" ส่วนนิตยสาร The New Yorker ถึงกับเรียกมันว่า "คู่แข่งตัวแรกที่จริงจังของสมองมนุษย์เท่าที่เคยสร้างกันมา"

ฟังดูคุ้น ๆ ไหม? ความตื่นเต้นเกินจริงแบบนี้ — แต่ความจริงคือ Rosenblatt มาเร็วเกินไปหลายสิบปี ฮาร์ดแวร์ในยุคนั้นช้าเกินไป และข้อมูลก็มีน้อยเกินไป ไอเดียถูกต้อง แต่โลกยังไม่พร้อม

เมื่อความหวังถูกเบรกกะทันหัน

ปี 1969 นักวิจัยจาก MIT สองคน คือ Marvin Minsky และ Seymour Papert ตีพิมพ์หนังสือชื่อ "Perceptrons" ที่พิสูจน์ด้วยคณิตศาสตร์ว่า perceptron **แบบชั้นเดียว** มีงานบางอย่างที่มันแก้ไม่ได้เลย

ขอย้ำคำว่า "แบบชั้นเดียว" ให้ชัด เพราะตรงนี้สำคัญมาก

สิ่งที่หนังสือพิสูจน์จริง ๆ คือข้อจำกัดของ perceptron เวอร์ชันง่ายสุด ที่มีแค่ชั้นเดียว มันไม่ได้พิสูจน์ว่าโครงข่ายประสาทเทียม "ทุกแบบ" ใช้ไม่ได้ — โดยเฉพาะแบบหลายชั้นที่เราเพิ่งเล่าไป กลับมีพลังมากกว่าเยอะ

แต่วงการในตอนนั้นตีความเลยเถิดไป ราวกับว่าประตูทั้งบานถูกปิดตาย ผลก็คือ ความสนใจและทุนวิจัยด้านโครงข่ายประสาทเทียมหดหายไป**ส่วนหนึ่ง**ในช่วงหลายปีต่อมา นักวิจัยส่วนใหญ่หันไปทำอย่างอื่นแทน

จะบอกว่าหนังสือเล่มนี้ "ฆ่า" โครงข่ายประสาทเทียมก็ไม่ถูกนัก นักประวัติศาสตร์ยังถกเถียงกันอยู่ว่ามันเป็นสาเหตุหลักหรือเป็นแค่ปัจจัยหนึ่ง แต่ที่แน่ ๆ คือ ช่วงเวลาแห่งความเงียบงันได้เริ่มขึ้นแล้ว

คนที่ไม่ยอมเดินจากไป

ในช่วงปีเงียบ ๆ นั้นเอง มีนักวิทยาศาสตร์ชาวอังกฤษคนหนึ่งชื่อ Geoffrey Hinton ที่ยังดื้อดึงทำงานกับโครงข่ายประสาทเทียมต่อไป ทั้งที่คนส่วนใหญ่มองว่ามันเป็นทางตันไปแล้ว

ปี 1986 Hinton ร่วมกับเพื่อนอีกสองคน คือ David Rumelhart และ Ronald Williams ตีพิมพ์งานชิ้นหนึ่งในวารสาร Nature ที่อธิบายวิธีทำให้โครงข่าย **หลายชั้น** เรียนรู้ได้จริง — ใช่แล้ว นี่แหละคือ backpropagation ที่เราพูดถึงไปก่อนหน้านี้ มันคือกุญแจที่ปลดล็อกข้อจำกัด "ชั้นเดียว" ที่ Minsky กับ Papert ชี้ไว้

วงการเริ่มกลับมาสนใจอีกครั้ง แต่ก็ยังไม่ระเบิด เพราะของสองอย่างที่ Rosenblatt เคยขาดเมื่อปี 1958 ก็ยังขาดอยู่ — เครื่องที่แรงพอ และข้อมูลที่เยอะพอ

เรื่องของ Hinton ยังไม่จบ เก็บเขาไว้ในใจก่อน เพราะเขาจะกลับมาในวินาทีที่ทุกอย่างเปลี่ยน

ความลับที่ทำงานเงียบ ๆ มาตลอด

ระหว่างที่วงการวิชาการยังลังเล โครงข่ายประสาทเทียมกลับแอบทำงานจริงอยู่เงียบ ๆ ในที่ที่คนทั่วไปไม่ทันสังเกต

ต้นทศวรรษ 1990 นักวิจัยชื่อ Yann LeCun ที่ห้องแล็บของ AT&T Bell Labs สร้างโครงข่ายชื่อ LeNet ขึ้นมาอ่านลายมือ — โดยเฉพาะรหัสไปรษณีย์ที่คนเขียนบนซองจดหมายให้ระบบของไปรษณีย์สหรัฐฯ ใช้คัดแยกจดหมาย

นี่ไม่ใช่การทดลองในห้องแล็บ แต่เป็นของที่ใช้งานจริง อ่านลายมือคนจริง ๆ ทุกวัน และมันแม่นมาก — เวอร์ชันปี 1998 อ่านตัวเลขลายมือผิดพลาดเพียงไม่ถึง 1 ใน 100 ตัวเท่านั้น นั่นคือความแม่นกว่า 99% ตั้งแต่ปี 1998

บทเรียนตรงนี้สำคัญ — โครงข่ายประสาทเทียม "ทำงานได้จริง" มานานแล้ว เพียงแต่ทำในงานเล็ก ๆ เฉพาะทาง เงียบ ๆ ก่อนที่มันจะดังเป็นพาดหัวข่าว สิ่งที่จะเกิดในปี 2012 จึงไม่ได้โผล่มาจากสุญญากาศ

วันที่ทั้งวงการต้องยอมรับ

ปี 2012 มีการแข่งขันชื่อ ImageNet ที่ให้คอมพิวเตอร์ดูรูปภาพแล้วทายว่าในรูปคืออะไร จากคลังรูปกว่าหนึ่งล้านรูป แบ่งเป็นพันหมวดหมู่ ก่อนหน้านี้ทุกทีมใช้วิธีดั้งเดิม และไม่มีใครเชื่อว่า deep learning จะเอาชนะงานใหญ่ขนาดนี้ได้

แล้วทีมหนึ่งก็ส่งโครงข่ายชื่อ AlexNet เข้าแข่ง

หัวหน้าทีมคือ Alex Krizhevsky นักศึกษาที่มีอาจารย์ที่ปรึกษาชื่อ — Geoffrey Hinton คนเดิมจากปี 1986 นั่นเอง AlexNet เป็นโครงข่าย 8 ชั้น ฝึกด้วยการ์ดจอเกม (GPU) สองตัวเป็นเวลาห้าถึงหกวัน ป้อนด้วยข้อมูลรูปภาพมหาศาลจาก ImageNet

ผลออกมาวันที่ 30 กันยายน 2012 และมันทำเอาทั้งวงการอึ้ง

AlexNet ทายผิดเพียง 15.3% ขณะที่ทีมอันดับสองทายผิดถึง 26.2% — ต่างกันเกือบ 11 เปอร์เซ็นต์ ในการแข่งขันที่ปกติคนเบียดกันชนะแค่เศษเสี้ยวเปอร์เซ็นต์ ช่องว่างขนาดนี้แทบจะเป็นไปไม่ได้ Yann LeCun (คนเดียวกับที่สร้าง LeNet) เรียกมันว่า "จุดเปลี่ยนที่ไม่มีข้อกังขา" ของวงการ

ทำไมมันถึงได้ผลในปี 2012 ทั้งที่ไอเดียมีมาตั้งแต่ 1958? เพราะของสองอย่างที่ Rosenblatt และ Hinton เคยรอคอย ในที่สุดก็มาครบ — **ข้อมูลมหาศาล** จาก ImageNet และ **เครื่องที่แรงพอ** อย่าง GPU ไอเดียเดิมที่เคยถูกมองว่าตายแล้ว พอเจอเชื้อเพลิงที่ใช่ ก็ลุกพรึ่บขึ้นมา

AlexNet ไม่ได้แค่ชนะการแข่งขัน มันเปลี่ยนคำถามของทั้งวงการ จาก "deep learning พอจะใช้ได้จริงไหม?" กลายเป็น "แล้วมีอะไรบ้างที่ deep learning ทำไม่ได้?"

ตอนจบของชายที่รอคอย

แล้ว Geoffrey Hinton ล่ะ ชายที่ดื้อดึงอยู่กับโครงข่ายประสาทเทียมตั้งแต่ยุคที่ไม่มีใครเอาด้วย?

ปี 2018 Hinton ได้รับรางวัล Turing Award ซึ่งเปรียบได้กับรางวัลโนเบลของวงการคอมพิวเตอร์ ร่วมกับ Yann LeCun และ Yoshua Bengio สำหรับงานบุกเบิกด้าน deep learning

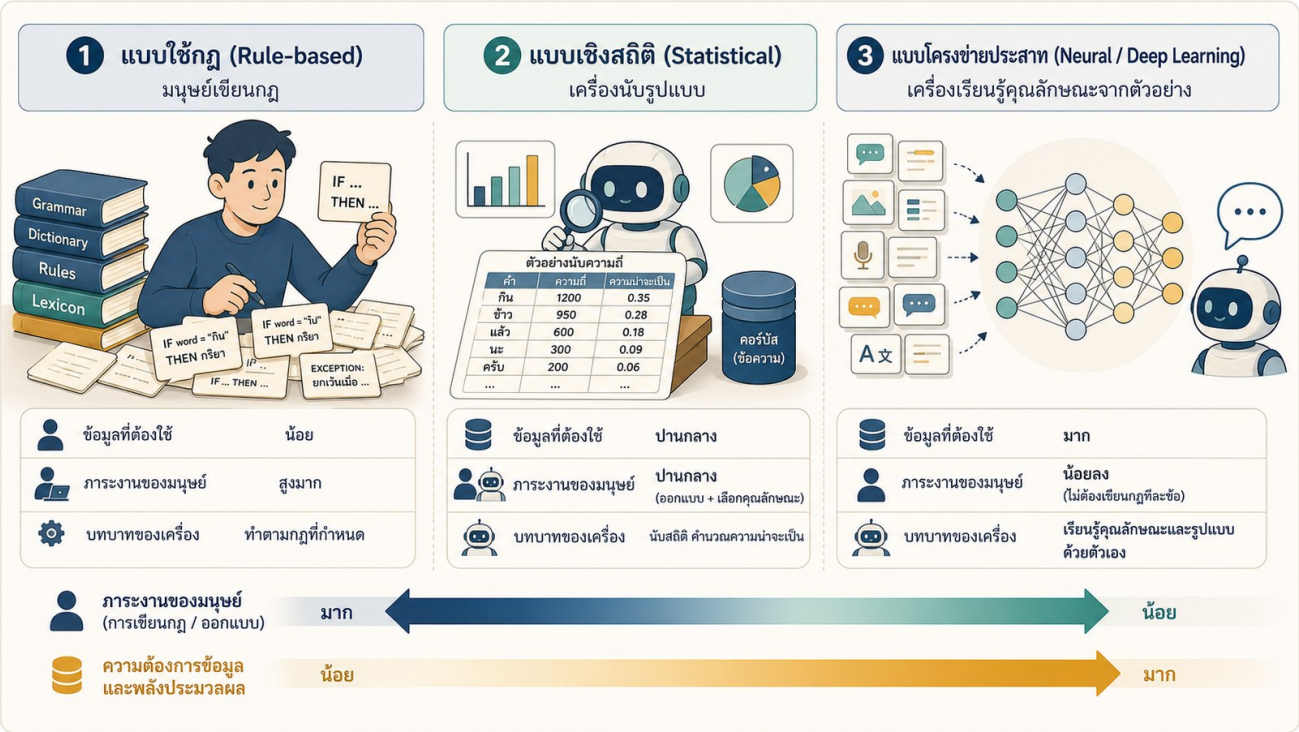
และปี 2024 เขาก็ได้รับรางวัล**โนเบลสาขาฟิสิกส์** ร่วมกับ John Hopfield สำหรับการค้นพบพื้นฐานที่ทำให้โครงข่ายประสาทเทียมเป็นจริงได้

จากไอเดียที่ถูกมองว่าตายแล้วเมื่อปี 1969 สู่รางวัลโนเบลในปี 2024 — เป็นเรื่องราวที่ใช้เวลากว่าครึ่งศตวรรษ และสอนเราว่า บางครั้งไอเดียที่ถูกต้อง แค่ต้องรอจนกว่าโลกจะพร้อมรับมัน

รวบทุกอย่างไว้ในตารางเดียว

มาถึงตรงนี้ เราเดินทางผ่านสามวิธีคิดใหญ่ ๆ ที่เครื่องใช้ทำความเข้าใจภาษาและโลก ตั้งแต่บทแรก ๆ จนถึงบทนี้ ขอรวบทุกอย่างไว้ในตารางเดียวที่จะติดตัวเราไปจนจบเล่ม

แนวทาง	ใครเป็นคนคิดตรรกะให้	จุดแข็ง	จุดอ่อน
เขียนกฎ (Rule-based)	มนุษย์เขียนกฎทุกข้อเอง เครื่องแค่ทำตาม	คาดเดาได้ ตรวจสอบได้ ใช้ข้อมูลน้อย เหมาะกับโลกแคบ ๆ ที่ควบคุมได้	ภาษาจริงมีข้อยกเว้นนับไม่ถ้วน เขียนกฎไม่มีวันครบ กฎเริ่มขัดกันเอง
ใช้สถิติ (Statistical)	มนุษย์ยังต้องบอกว่า "ให้นับอะไร" แล้วเครื่องนับความถี่จากข้อมูลเอง	ยืดหยุ่นกว่ากฎ จับรูปแบบจากตัวอย่างจริงได้ ปรับตามข้อมูลใหม่ได้	มองบริบทได้สั้น มนุษย์ยังต้องออกแบบว่าจะให้ดูอะไร ไม่เข้าใจความหมายลึก
โครงข่ายประสาท (Neural / Deep)	เครื่องหา**ทั้งกฎและสิ่งที่ควรสังเกต**เองจากตัวอย่าง มนุษย์แค่บอกว่าอันไหนถูกผิด	เรียนรูปแบบซับซ้อนมาก ๆ ได้เอง ไม่ต้องให้คนคอยบอกว่าต้องดูอะไร สเกลขึ้นได้	**หิวข้อมูลและพลังประมวลผลมหาศาล** อธิบายว่ามันคิดยังไงได้ยาก เหมือนกล่องดำ



เปรียบเทียบสามแนวทาง: เขียนกฎ ใช้สถิติ และโครงข่ายประสาท

ความต่างที่สำคัญที่สุด อยู่ในคอลัมน์แรก — "ใครเป็นคนคิดตรรกะให้"

ในยุคกฎ มนุษย์ทำงานหนักทั้งหมด เราต้องคิดทุกอย่างให้เครื่อง ในยุคสถิติ เราแบ่งงานกัน เราบอกว่าให้ดูอะไร เครื่องไปนับเอง พอมาถึงโครงข่ายประสาท เราแทบจะปล่อยมือ เราแค่ยื่นตัวอย่างพร้อมเฉลยให้ แล้วเครื่องไปหาเองหมดว่าควรสังเกตอะไรและสรุปยังไง

และนี่คือเหตุผลว่าทำไม "ข้อมูล" ถึงสำคัญมากในยุคนี้ ยิ่งเราขยับภาระจากมนุษย์ไปให้เครื่องเรียนเองมากเท่าไหร่ เครื่องก็ยิ่งต้องการตัวอย่างมากขึ้นเท่านั้น โครงข่ายประสาทที่ทรงพลังจึงต้องกินข้อมูลมหาศาล — และนั่นคือเหตุผลที่ AlexNet ต้องรอจนถึงปี 2012 ที่ ImageNet และ GPU พร้อมแล้ว

กำแพงที่โครงข่ายยุคนั้นยังข้ามไม่ได้

ก่อนจะปิดบท มีปัญหาหนึ่งที่ต้องวางไว้ เพราะมันคือโจทย์ของบทถัดไปทั้งบท

โครงข่ายประสาทเก่งเรื่องรูปภาพมาก แต่พอมาเจอ "ภาษา" มันเจอความยุ่งยากแบบใหม่ เพราะภาษามาเป็นลำดับ — คำต่อคำ และความหมายของคำหลัง ๆ ขึ้นอยู่กับคำที่มาก่อนหน้า

โครงข่ายที่ใช้อ่านภาษาในยุคนั้น (มีชื่อเรียกว่า RNN และรุ่นพัฒนาขึ้นชื่อ LSTM) ทำงานด้วยการอ่านไล่ไปทีละคำ พร้อมพยายาม "จำ" สิ่งที่อ่านมาก่อนหน้า

ปัญหาคือ มันมีความจำสั้น

ลองนึกถึงเวลาอ่านนิยายยาว ๆ ที่ต้องจำชื่อตัวละครจากบทแรก เพื่อเข้าใจฉากในบทสุดท้าย โครงข่ายพวกนี้ทำแบบนั้นได้ไม่ค่อยดี ยิ่งประโยคยาว มันยิ่ง "ลืม" ตอนต้น พออ่านมาถึงท้ายประโยค มันก็ลืมไปแล้วว่าตอนต้นพูดถึงอะไร

LSTM ช่วยให้จำได้ยาวขึ้นบ้าง แต่ก็ยังไม่หายขาด กับข้อความที่ยาวมาก ๆ ปัญหา "ลืมต้นเรื่อง" ก็ยังตามมาหลอกหลอน

และนี่แหละคือกำแพงที่รอให้ใครสักคนมาทลายในบทถัดไป

แล้วมันเข้าใจจริงไหม

กลับมาที่คำถามแกนกลางของเราอีกครั้ง — โครงข่ายประสาทที่แยกแมวกับหมาได้ มัน "เข้าใจ" จริงไหม หรือแค่จับรูปแบบเก่งขึ้น?

คำตอบที่ซื่อสัตย์ก็เหมือนเดิม — มันก้าวมาไกลขึ้นมาก แต่ยังไม่ใช่ความเข้าใจแบบเรา

เครื่องที่ทายว่ารูปนี้คือแมว ไม่ได้ "รู้" ว่าแมวคือสัตว์เลี้ยงที่ขนนุ่ม ชอบนอน ร้องเหมียว มันแค่หมุนปุ่มนับล้านปุ่มจนรูปแบบของพิกเซลในรูปนี้ ไปจุดประกายให้คำตอบ "แมว" โผล่ออกมา นั่นคือการจับรูปแบบที่ทรงพลังอย่างยิ่ง แต่ไม่ใช่ความรู้เกี่ยวกับโลกจริง

และต้องพูดให้ชัดเรื่องชื่อ "โครงข่ายประสาทเทียม" ด้วย มันได้แรงบันดาลใจมาจากเซลล์สมองก็จริง แต่ของจริงในสมองเราซับซ้อนกว่านี้มหาศาล — หน่วยเล็ก ๆ ในโครงข่ายเป็นแค่การคำนวณตัวเลขง่าย ๆ ไม่ใช่เซลล์ชีวภาพ และสมองจริงก็ไม่ได้เรียนรู้ด้วยวิธี backpropagation แบบที่เครื่องใช้

พูดง่าย ๆ คือ มันเหมือนเครื่องบินที่ได้แรงบันดาลใจจากนก — บินได้เหมือนกัน แต่ไม่ได้กระพือปีก และไม่ได้เป็นนก

สรุปง่าย ๆ ใน 5 ข้อ

1. โครงข่ายประสาทเทียมเรียนแบบเดียวกับที่เด็กหัดแยกแมวกับหมา — ไม่ใช่ท่องกฎ แต่ดูตัวอย่างเยอะ ๆ เดา เช็กว่าผิดหรือถูก แล้วปรับตัวเองทีละนิด วนซ้ำเป็นล้านครั้งจนเก่งขึ้น
2. "deep" (ลึก) หมายถึงมีหลายชั้นจริง ๆ ชั้นล่างจับของง่าย (เส้น ขอบ) ชั้นบนเอามารวมเป็นของซับซ้อน (ตา หู ใบหน้า) ยิ่งลึกยิ่งเรียนเรื่องยากได้
3. ไอเดียนี้เก่าตั้งแต่ปี 1958 (Rosenblatt) เคยถูกมองว่าตายไปหลังปี 1969 (Minsky-Papert ชี้ข้อจำกัดของแบบ "ชั้นเดียว") แต่ Geoffrey Hinton ไม่ยอมเลิก จนมันระเบิดในปี 2012 ตอนข้อมูลและ GPU พร้อม (AlexNet) — และพา Hinton ไปถึงรางวัลโนเบลปี 2024
4. ความต่างหลักจากยุคก่อน: ยุคกฎมนุษย์คิดทุกอย่าง ยุคสถิติแบ่งกันทำ ส่วนโครงข่ายประสาทให้เครื่องหาเองเกือบหมด — แลกมาด้วยการต้องป้อนข้อมูลมหาศาล นี่คือเหตุผลว่าทำไม "ข้อมูล" ถึงสำคัญ

5. ข้อจำกัดที่ยังเหลือ: โครงข่ายยุคก่อน (RNN/LSTM) อ่านภาษาแบบไล่ทีละคำ แล้ว "ลืม" ต้นประโยคเมื่อข้อความยาว — บริบทยาวจึงยังเป็นกำแพง

ก่อนไปบทต่อไป

เราเหลือกำแพงหนึ่งบานค้างไว้ — โครงข่ายที่อ่านภาษาไล่ทีละคำแล้วลืมต้นเรื่อง เหมือนอ่านนิยายแล้วลืมชื่อตัวละครกลางเล่ม

ปี 2017 มีงานวิจัยชิ้นหนึ่งที่ชื่อกวน ๆ ว่า "Attention Is All You Need" (แค่ความใส่ใจก็พอแล้ว) เสนอวิธีคิดใหม่หมด แทนที่จะให้เครื่องอ่านไล่ทีละคำแล้วพยายามจำ ทำไมไม่ให้มัน "เหลือบมอง" ทั้งประโยคพร้อมกันไปเลย แล้วดูว่าคำไหนเกี่ยวกับคำไหน?

วิธีคิดนี้แก้ปัญหา "ลืมต้นเรื่อง" ได้อย่างหมดจด และกลายเป็นสถาปัตยกรรมที่อยู่เบื้องหลัง ChatGPT และ AI ภาษาทุกตัวในวันนี้ — มันมีชื่อว่า Transformer และเป็นเรื่องของบทถัดไป

บทที่ 7 — Transformer และพลังของ Attention

ปี 2017 นักวิจัยกลุ่มหนึ่งจาก Google ส่งบทความวิจัยชิ้นหนึ่งขึ้นเว็บ ชื่อของมันคือ **"Attention Is All You Need"** — แปลตรง ๆ ว่า "แค่ความใส่ใจก็เพียงพอแล้ว"

ในแวดวงวิชาการที่ชื่อบทความมักจะยาวเหยียดและสุภาพเรียบร้อย ชื่อนี้ฟังดูเหมือนคนลุกขึ้นพูดกลางที่ประชุมว่า "งานสามสิบปีที่พวกคุณสร้างกันมา? ไม่จำเป็นแล้วครับ"

เพราะสิ่งที่บทความนี้กำลังจะบอกก็คือ เครื่องมือหลักที่วงการใช้สอน AI ให้อ่านภาษามาตลอด — สิ่งที่เราเล่าค้างไว้ตอนท้ายบทก่อนในชื่อ RNN และ LSTM — มันโยนทิ้งได้เลย

ฟังดูบ้าบิ่นมาก และมันก็ควรจะล้มเหลว แต่มันไม่ได้ล้มเหลว มันกลายเป็นจุดเปลี่ยนที่ทำให้เกิด ChatGPT และ AI ภาษาทุกตัวที่เราใช้กันทุกวันนี้

เรื่องนี้เลยน่าเล่า เพราะมันคือชั่วโมงที่กำแพงซึ่งเราค้างไว้เมื่อบทที่แล้ว — กำแพงที่ AI อ่านประโยคยาว ๆ แล้วลืมต้นเรื่อง — ถูกทลายลงในที่สุด

กำแพงเดิมที่เราค้างไว้

ทบทวนสั้น ๆ ก่อน เพื่อให้เห็นว่าโจทย์คืออะไร

โครงข่ายประสาทเทียมที่ใช้อ่านภาษาในยุคก่อนหน้า (RNN และรุ่นพัฒนาขึ้นชื่อ LSTM) ทำงานด้วยวิธีเดียว คือ **อ่านไล่ทีละคำ** — คำแรก คำสอง คำสาม ไปจนจบประโยค พร้อมพยายาม "จำ" สิ่งที่อ่านผ่านมา

ลองนึกภาพคนที่อ่านหนังสือผ่านรูกุญแจ เห็นทีละคำ เลื่อนไปทีละคำ พออ่านมาถึงคำที่ห้าสิบ ความทรงจำเรื่องคำที่หนึ่งก็เลือนไปเกือบหมดแล้ว นี่คือปัญหาแรก — **มันมีความจำสั้น** ยิ่งประโยคยาว ยิ่งลืมตอนต้น

ปัญหาที่สองซ่อนอยู่ลึกกว่านั้น และสำคัญกว่าที่หลายคนคิด เพราะต้องอ่านไล่ทีละคำ การฝึกโมเดลแบบนี้จึง **เร่งความเร็วไม่ได้** ต่อให้มีคอมพิวเตอร์ทรงพลังแค่ไหน มันก็ต้องรอประมวลผลคำที่หนึ่งให้เสร็จก่อน ถึงจะขยับไปคำที่สองได้ ทำงานเป็นคิวตลอด เหมือนต่อให้คุณมีพ่อครัวร้อยคน แต่สั่งให้ทำอาหารทีละจาน คนที่เหลือก็ต้องยืนรอเฉย ๆ

จำปัญหาข้อสองนี้ไว้ให้ดี เพราะเดี๋ยวมันจะกลายเป็นกุญแจของทั้งเรื่อง

ถ้าไม่ต้องอ่านทีละคำล่ะ

คำถามที่ทีม Google ตั้งขึ้นนั้นเรียบง่ายจนน่าตกใจ

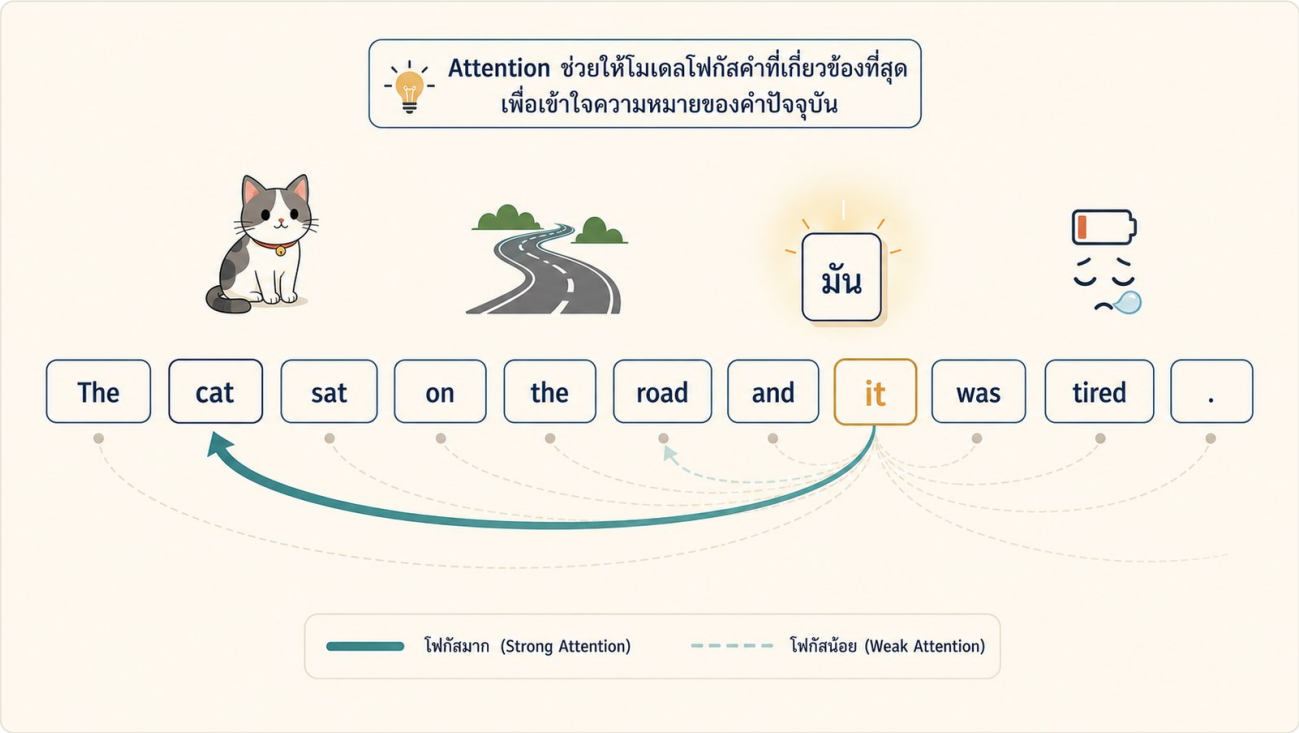
ในเมื่อปัญหาทั้งสองข้อมาจากการ "อ่านไล่ทีละคำ" แล้วทำไมเราไม่เลิกอ่านทีละคำไปเลยล่ะ? ทำไมไม่ให้โมเดล **เหลือบมองทั้งประโยคพร้อมกันทีเดียว** แล้วดูว่าคำไหนเกี่ยวข้องกับคำไหน?

นี่แหละคือหัวใจของกลไกที่ชื่อว่า **attention**

> **ศัพท์ที่เพิ่งเจอ — attention** แปลเป็นภาษาคนคือ "กลไกที่ให้โมเดลเหลือบมองทั้งประโยคพร้อมกัน แล้วชั่งน้ำหนักว่าคำที่กำลังพิจารณาอยู่นั้น เกี่ยวข้องกับคำไหนในประโยคมากที่สุด" บางทีก็แปลกันว่า "กลไกความใส่ใจ" — แต่ขอเตือนไว้ก่อนว่าคำว่า "ใส่ใจ" ในที่นี้ไม่ได้แปลว่ามันมีจิตใจหรือความตั้งใจแบบเรา มันคือการคำนวณตัวเลขล้วน ๆ ว่าคำไหนควรส่งผลต่อคำไหน เราจะกลับมาเรื่องนี้ตอนท้ายบท

แทนที่จะอ่านไล่แล้วพยายามจำ โมเดลที่มี attention จะทำเหมือนคนวางหนังสือกางออกบนโต๊ะ แล้วกวาดสายตามองได้ทั้งหน้าในคราวเดียว เห็นคำต้นประโยคกับคำท้ายประโยคพร้อมกัน ลากเส้นเชื่อมโยงกันได้ทันที โดยไม่ต้องกลัวว่าจะลืม เพราะมันไม่ได้ "จำ" — มันแค่ "มองเห็นทั้งหมดอยู่ตรงนั้น"

"มัน" ในประโยคนี้ หมายถึงอะไร



attention เชื่อมคำว่า "มัน" กลับไปยังคำที่เกี่ยวข้องมากที่สุด

มาดูตัวอย่างที่ทำให้เห็นภาพชัดที่สุด เป็นตัวอย่างคลาสสิกที่คนใช้สอนเรื่อง attention กันมานาน

ลองอ่านประโยคนี้:

> **เจ้าแมวไม่ยอมข้ามถนน เพราะมันเหนื่อยเกินไป**

คำถามง่าย ๆ — คำว่า **"มัน"** ในประโยคนี้ หมายถึงอะไร? แมว หรือถนน?

คุณรู้คำตอบทันทีโดยไม่ต้องคิด มันคือ "แมว" แน่นอน เพราะถนนไม่มีทางเหนื่อยได้ มีแต่สิ่งมีชีวิตเท่านั้นที่เหนื่อย

แต่สังเกตนะว่าคุณรู้ได้ยังไง คุณไม่ได้รู้จากไวยากรณ์ คุณรู้จาก **ความหมาย** — คุณเอาคำว่า "เหนื่อย" ไปจับคู่กับสิ่งที่เหนื่อยได้ แล้วก็เจอว่ามีแค่ "แมว" ตัวเดียวที่เข้าข่าย นี่คือสิ่งที่สมองคุณทำในเสี้ยววินาทีโดยไม่รู้ตัวด้วยซ้ำ

ทีนี้ลองคิดในมุมของ AI ยุคเก่าที่อ่านทีละคำ กว่ามันจะไล่อ่านมาถึงคำว่า "มัน" คำว่า "แมว" ผ่านไปตั้งหลายคำแล้ว ถ้าประโยคยาวกว่านี้ มันอาจจะเลือน ๆ ไปแล้วว่าใครเป็นตัวเอกของประโยค

แต่โมเดลที่มี attention ทำต่างออกไป ตอนที่มันประมวลคำว่า "มัน" มันจะเหลือบมองย้อนกลับไปทั้งประโยคพร้อมกัน แล้วถามตัวเองว่า "คำนี้เกี่ยวกับคำไหนมากที่สุด" จากนั้นมันจะให้ **น้ำหนัก** กับคำว่า "แมว" สูงกว่าคำว่า "ถนน"

พูดง่าย ๆ คือมันลากเส้นหนา ๆ จากคำว่า "มัน" กลับไปหา "แมว" และลากเส้นบาง ๆ ไปหา "ถนน" — แล้วเลือกเส้นที่หนากว่า นั่นแหละคือ attention กำลังทำงาน

คำเดิม แต่ความหมายขยับได้แล้ว

ตรงนี้คือจุดที่อยากให้คุณย้อนกลับไปนึกถึงเรื่องเมื่อสองบทก่อน

ตอนที่เราเล่าเรื่องการแปลงคำเป็นตัวเลขบนแผนที่ความหมาย (embedding) เราทิ้งปัญหาหนึ่งไว้ — **คำหนึ่งคำมีพิกัดเดียวตายตัว** ไม่ว่าจะเจอมันในประโยคไหน คำว่า "ขัน" ก็มีตำแหน่งเดียวบนแผนที่ ทั้งที่จริงมันอาจหมายถึงภาชนะตักน้ำ หรือเสียงไก่ขัน หรืออาการขำขัน ก็ได้ ขึ้นอยู่กับประโยค แต่แผนที่ความหมายแบบเดิมแยกไม่ออก เพราะมันให้พิกัดเดียวมาตั้งแต่ต้น

attention คือสิ่งที่เข้ามาแก้ปัญหานี้

กลับมาที่คำว่า "ขัน" ที่เพิ่งพูดถึง ลองอ่านสองประโยคนี้:

> "เขายก**ขัน**ตักน้ำขึ้นมาล้างหน้า" "พอได้ยินมุกนั้น ทุกคนก็**ขัน**กันทั้งวง"

คำว่า "ขัน" ตัวเดียวกันเป๊ะ แต่คุณรู้ทันทีว่าประโยคแรกหมายถึงภาชนะตักน้ำ ส่วนประโยคหลังหมายถึงอาการขำ คุณรู้ได้เพราะคำรอบข้าง — "ตักน้ำ" กับ "มุก" บอกใบ้คนละทาง

attention ทำให้โมเดลทำแบบเดียวกันได้ — ตอนประมวลคำว่า "ขัน" มันมองไปทั้งประโยค เห็นว่ามี "ตักน้ำ" หรือ "มุก" อยู่ แล้วก็ปรับความหมายของคำว่า "ขัน" ให้เข้ากับเพื่อนบ้านในประโยคนั้น ๆ (Google เองก็อธิบายหลักการเดียวกันนี้ด้วยตัวอย่างภาษาอังกฤษ — คำว่า bank ที่แปลได้ทั้ง "ธนาคาร" และ "ริมตลิ่ง" ขึ้นกับว่ามี "แม่น้ำ" หรือ "ถนน" อยู่ในประโยค)

นี่คือก้าวที่สวยงามมาก จำที่เราบอกไว้ตอนบทแรก ๆ ไหมว่าภาษามนุษย์ยากตรงที่คำเดียวมีหลายความหมาย และความหมายขึ้นกับบริบท? attention คือกลไกแรกที่จัดการเรื่องนี้ได้อย่างเป็นธรรมชาติ จาก "คำหนึ่งคำ พิกัดเดียวตายตัว" กลายเป็น "คำหนึ่งคำ ที่ความหมายขยับตามเพื่อนรอบข้างได้"

> **จำง่าย ๆ ว่า:** attention คือการที่โมเดล "มองทั้งประโยคพร้อมกัน" เพื่อหาว่าคำไหนเกี่ยวกับคำไหน แทนที่จะอ่านไล่ทีละคำแล้วพยายามจำให้ครบ — และนี่คือสิ่งที่ทำให้คำเดียวกันมีความหมายต่างกันได้ตามบริบท

attention ไม่ใช่ของใหม่ — แต่ความกล้าต่างหากที่ใหม่

ถึงตรงนี้ต้องเล่าให้ครบ เพราะมีความเข้าใจผิดที่พบบ่อย

หลายคนคิดว่าบทความปี 2017 เป็นคนคิดค้น attention ขึ้นมา — ไม่ใช่ จริง ๆ แล้ว attention มีมาตั้งแต่ปี 2014 แล้ว นักวิจัยกลุ่มหนึ่ง (นำโดย Bahdanau) เอามันมาใช้เป็น "ของเสริม" ติดเข้ากับ RNN เพื่อช่วยให้โมเดลแปลภาษามองย้อนกลับไปดูคำต้นฉบับได้ดีขึ้น ปีถัดมาก็มีคนปรับปรุงต่อ attention จึงเป็นเครื่องมือที่วงการรู้จักดีอยู่แล้วหลายปี

แต่ตอนนั้น attention เป็นแค่ "อุปกรณ์เสริม" — RNN ยังเป็นพระเอก อ่านไล่ทีละคำเหมือนเดิม แค่มี attention มาช่วยให้จำดีขึ้นนิดหน่อย ทุกคนคิดว่าต้องเป็นแบบนี้ ทิ้ง RNN ไม่ได้หรอก

สิ่งที่ทีม Vaswani และเพื่อนอีกเจ็ดคนทำในปี 2017 คือก้าวที่ใจกล้ากว่านั้น — พวกเขา **โยน RNN ทิ้งทั้งหมด** แล้วใช้ attention เพียงอย่างเดียว ไม่อ่านไล่ทีละคำอีกต่อไป มองทั้งประโยคพร้อมกันล้วน ๆ

นั่นแหละคือที่มาของชื่อ "Attention Is All You Need" — มันไม่ได้บอกว่า "เราคิด attention ได้" มันกำลังประกาศว่า "RNN ที่พวกคุณพึ่งพากันมา ไม่จำเป็นอีกแล้ว แค่ attention อย่างเดียวก็พอ"

> **ศัพท์ที่เพิ่งเจอ — Transformer** ชื่อเรียกสถาปัตยกรรม (โครงสร้างพื้นฐานของโมเดล) แบบใหม่ที่บทความนี้เสนอ — โมเดลที่ทำงานด้วย attention ล้วน ๆ ไม่มีการอ่านไล่ทีละคำแบบ RNN เลย พูดง่าย ๆ คือ Transformer คือ "พิมพ์เขียว" ของโมเดล ส่วน ChatGPT คือบ้านหลังหนึ่งที่สร้างจากพิมพ์เขียวนี้ (ยังต้องผ่านอีกหลายขั้นกว่าจะเป็น ChatGPT — เป็นเรื่องของบทถัดไป)

มีเรื่องเล่ากันว่าชื่อ "Transformer" มาจากการที่หนึ่งในทีมชอบเสียงของคำนี้ และว่ากันว่าชื่อบทความก็แอบล้อชื่อเพลงของวง Beatles ที่ชื่อ "All You Need Is Love" ด้วย — เรื่องพวกนี้สนุกดี แต่ยังไม่มีหลักฐานชั้นต้นยืนยันชัด ๆ เลยขอเล่าไว้เป็นเกร็ดเฉย ๆ ไม่ใช่ข้อเท็จจริงที่ปักธงได้

ทำไมมันถึงเปลี่ยนโลก

ถ้าแค่ "มองทั้งประโยคพร้อมกัน" มันก็ฟังดูเป็นแค่ลูกเล่นเล็ก ๆ ใช่ไหม แล้วทำไมมันถึงสะเทือนวงการขนาดนั้น? เหตุผลมีสามข้อ และข้อสุดท้ายคือข้อที่สำคัญที่สุด

ข้อแรก — หมดปัญหาลืมต้นประโยค เมื่อทุกคำมองเห็นทุกคำในประโยคได้โดยตรง คำว่า "มัน" ที่อยู่ท้ายประโยคก็เชื่อมกลับไปหา "แมว" ที่อยู่ต้นประโยคได้ทันที ไม่ว่าจะห่างกันกี่คำ กำแพง "ความจำสั้น" ที่เราค้างไว้เมื่อบทก่อน ถูกทลายตรงนี้

ข้อสอง — ฝึกแบบขนานได้ จำปัญหาข้อสองที่ให้จำไว้ตอนต้นบทได้ไหม? เพราะ Transformer ไม่ต้องรออ่านคำก่อนหน้าให้เสร็จก่อน มันประมวลทุกคำพร้อมกันได้เลย การฝึกโมเดลจึงทำพร้อม ๆ กันทีเดียวได้ เหมือนพ่อครัวร้อยคนทำอาหารร้อยจานพร้อมกัน แทนที่จะทำทีละจาน นี่ทำให้ฝึกได้เร็วกว่าเดิมหลายเท่า

ข้อสาม — มันสเกลขึ้นได้ และนี่คือข้อที่เปลี่ยนทุกอย่าง เมื่อฝึกได้เร็วและฝึกพร้อมกันได้ เราก็ป้อนข้อมูลปริมาณมหาศาลให้มันเรียนได้ในเวลาที่สมเหตุสมผล — และสร้างโมเดลที่ใหญ่ขึ้นเรื่อย ๆ ได้จริง

ลองคิดดูว่าสามข้อนี้ต่อกันเป็นลูกโซ่ยังไง: มองทั้งประโยคได้ → เลยฝึกพร้อมกันได้ → เลยป้อนข้อมูลมหาศาลได้ → เลยสร้างโมเดลยักษ์ได้ ประตูบานนี้เองที่เปิดทางไปสู่ยุคของโมเดลภาษาขนาดใหญ่ที่เราเรียกกันว่า LLM

ตามบทความต้นฉบับ Transformer แปลภาษาได้ทั้ง **ดีกว่าและเร็วกว่า** ทุกระบบที่มีมาก่อนหน้า Google เองก็เผยแพร่บทอธิบายตามมาในเดือนสิงหาคมปีนั้น ยืนยันว่ามันเร็วกว่าของเดิมได้หลายเท่าตัว วงการเลยตื่นทันที

คลื่นที่ตามมาภายในปีเดียว

สิ่งที่เกิดขึ้นหลังจากนั้นเร็วจนน่าใจหาย

มิถุนายน 2018 — แค่ปีเดียวหลังบทความออก — OpenAI ปล่อยโมเดลชื่อ **GPT-1** ออกมา สร้างบน Transformer ชื่อ "GPT" ที่คุณคุ้นเคยทุกวันนี้ ตัว T ตัวสุดท้ายนั่นแหละย่อมาจาก Transformer

ตุลาคมปีเดียวกัน Google ปล่อยโมเดลชื่อ **BERT** ที่สร้างบน Transformer เช่นกัน และมันทำลายสถิติงานวัดผลด้านภาษาหลายมาตรวัดพร้อมกันในคราวเดียว วงการ NLP สั่นสะเทือน

ภายในเวลาประมาณปีกว่า ๆ สนามที่เคยเต็มไปด้วย RNN ก็เปลี่ยนมาเป็น Transformer แทบทั้งหมด ไม่ค่อยมีการเปลี่ยนผ่านครั้งไหนในประวัติศาสตร์ AI ภาษาที่เกิดเร็วและเด็ดขาดเท่านี้ — และบทความปี 2017 ก็กลายเป็นหนึ่งในงานวิจัยที่ถูกอ้างอิงมากที่สุดในประวัติศาสตร์วิทยาการคอมพิวเตอร์

และนี่คือเรื่องที่อยากให้จำกลับบ้าน — **GPT, Claude, Gemini และ LLM ชั้นนำเกือบทุกตัวที่คุณเคยใช้ล้วนสร้างบน Transformer ทั้งสิ้น** ถ้าคุณเคยพิมพ์คุยกับ ChatGPT แล้วประหลาดใจว่ามันเข้าใจบริบทประโยคยาว ๆ ได้ดีจัง — นั่นแหละคือ attention กำลังทำงานอยู่เบื้องหลัง

ก่อนจะตื่นเต้นเกินไป

ทุกครั้งที่เราเล่าถึงก้าวกระโดด เราจะหยุดถามคำถามเดิมเสมอ — แล้วมันเข้าใจจริงไหม หรือแค่เก่งขึ้น?

ขอวางหลักหมายไว้สามจุดให้ชัด

หนึ่ง คำว่า "attention" หรือ "ความใส่ใจ" ไม่ได้แปลว่าโมเดลมีจิตใจหรือความตั้งใจ มันคือการคำนวณตัวเลขว่าคำไหนควรมีน้ำหนักต่อคำไหนมากที่สุด ไม่มีใครนั่ง "สนใจ" อะไรอยู่ในนั้น มีแต่คณิตศาสตร์ — เราใช้คำว่า "ใส่ใจ" เพราะมันช่วยให้นึกภาพออก ไม่ใช่เพราะเครื่องรู้สึกอะไร

สอง Transformer จับบริบทได้ดีขึ้นมากก็จริง แต่ "จับบริบทเก่งขึ้น" ไม่เท่ากับ "เข้าใจโลกแบบมนุษย์" มันยังคงทำงานด้วยการชั่งน้ำหนักความสัมพันธ์ของคำ ไม่ใช่การรู้ความหมายแบบที่เรารู้ว่าแมวคืออะไร และถึงมันจะมองได้กว้างกว่า RNN มาก แต่ก็ยังมีขีดจำกัดอยู่ว่ามองได้ไกลแค่ไหนในคราวเดียว ไม่ใช่ไร้ขอบเขต

สาม แม้แต่คำว่า "LLM ทุกตัวใช้ Transformer" ก็ยังพูดเต็มปากไม่ได้ ที่ถูกคือ "เกือบทั้งหมด" — มีงานวิจัยสายอื่นอยู่บ้าง เพียงแต่ตอนนี้ยังเป็นส่วนน้อยในสนาม

พูดให้ตรงคือ Transformer ทำให้โมเดลจัดการบริบทยาวและความสัมพันธ์ของคำได้ดีกว่าเดิมมหาศาล — แต่มันก็ยังเป็นนักทำนายคำที่เก่งขึ้น ไม่ใช่ผู้เข้าใจที่ตื่นขึ้นมา คำถามแกนกลางของเรายังไม่จบ

สรุปง่าย ๆ ใน 5 ข้อ

1. ปัญหาเดิมของ AI ยุค RNN/LSTM คือมันอ่านภาษาไล่ทีละคำ เลยลืมต้นประโยคเมื่อข้อความยาว และเร่งความเร็วการฝึกไม่ได้เพราะต้องทำเป็นคิว
2. **attention** คือกลไกที่ให้โมเดล "เหลือบมองทั้งประโยคพร้อมกัน" แล้วชั่งน้ำหนักว่าคำไหนเกี่ยวกับคำไหน เช่น รู้ว่า "มัน" ในประโยคหมายถึง "แมว" ไม่ใช่ "ถนน" โดยดูจากความหมาย
3. นี่ยังแก้ปัญหาเก่าจากบทแผนที่ความหมายด้วย — คำเดียวกันที่เคยมีพิกัดเดียวตายตัว ตอนนี้ความหมายขยับตามบริบทรอบข้างได้แล้ว
4. **Transformer** (บทความ "Attention Is All You Need" ปี 2017) ไม่ได้คิด attention ขึ้นมาใหม่ แต่กล้าโยน RNN ทิ้งทั้งหมดแล้วใช้ attention ล้วน ผลคือมองบริบทได้กว้าง ฝึกแบบขนานได้ จึงสเกลขึ้นเป็นโมเดลยักษ์ได้ — ประตูสู่ยุค LLM
5. GPT, Claude, Gemini และ LLM ชั้นนำเกือบทุกตัวสร้างบน Transformer แต่ "จับบริบทเก่งขึ้น" ยังไม่เท่ากับ "เข้าใจแบบมนุษย์"

ก่อนไปบทต่อไป

ตอนนี้เรามีสถาปัตยกรรมที่ทรงพลังอยู่ในมือแล้ว — เครื่องยนต์ที่มองทั้งประโยคได้พร้อมกัน และสเกลขึ้นได้เท่าที่เรากล้าป้อนข้อมูลให้

แต่เครื่องยนต์เปล่า ๆ ยังขับไปไหนไม่ได้ Transformer ที่เพิ่งสร้างเสร็จก็เหมือนสมองที่ยังว่างเปล่า มันยังไม่รู้ภาษาสักคำ

คำถามถัดไปจึงเป็น — แล้วเราสอนภาษาให้มันยังไง? คำตอบคือเราเอาข้อความปริมาณมหาศาลระดับอ่านหนังสือทั้งห้องสมุดมาป้อนให้มัน แล้วฝึกมันด้วยเกมง่าย ๆ เกมเดียว คือ "ทายคำถัดไป" — กระบวนการที่เรียกว่า pretraining

แต่ที่น่าสนใจกว่านั้นคือ ต่อให้อ่านมามากมายมหาศาลแค่ไหน โมเดลดิบ ๆ ที่ผ่าน pretraining มา ก็ยัง **ไม่ใช่ ChatGPT** อยู่ดี มันเหมือนคนที่อ่านหนังสือมาทั้งโลกแต่ยังไม่รู้จะตอบคำถามให้ตรงประเด็นยังไง ต้องมีอีกหลายขั้นกว่าจะกลายเป็นผู้ช่วยที่เราคุยด้วยได้ — และนั่นคือเรื่องของบทถัดไป

บทที่ 8 — Pretraining, LLM และวันที่ AI กลายเป็นผู้ช่วย

ลองนึกภาพเด็กคนหนึ่งที่โตขึ้นมาในห้องสมุดขนาดยักษ์ ตั้งแต่จำความได้เขาทำอย่างเดียวคืออ่าน — ประวัติศาสตร์ทุกยุค ตำราวิทยาศาสตร์ บทกวี สูตรทำอาหาร คู่มือซ่อมรถ นิยายสืบสวน ข่าวเก่านับล้านฉบับ อ่านจนไม่เหลือหนังสือสักเล่มในห้องที่เขายังไม่เคยพลิก

วันหนึ่งเขาเดินออกมาจากห้องสมุด แล้วมีคนเดินเข้ามาถามว่า "ช่วยสรุปรายงานหน้านี้ให้หน่อยได้ไหม"

คุณคิดว่าเขาจะตอบยังไง?

ถ้าเด็กคนนี้ไม่เคยมีใครคุยด้วยเลย ไม่เคยถูกถามคำถามแล้วถูกสอนว่า "เวลามีคนถาม ให้ตอบสิ่งที่เขาอยากรู้" สิ่งที่เขาทำอาจไม่ใช่การสรุป — เขาอาจพูดต่อเรื่อยไปเกี่ยวกับหัวข้อนั้น หรือยกประโยคถัดไปที่ "น่าจะตามมา" ออกมาเหมือนกำลังเขียนรายงานต่อ เพราะตลอดชีวิตเขารู้จักภาษาในฐานะ "ข้อความที่ไหลต่อกันไป" ไม่ใช่ "บทสนทนาที่มีคนถามและมีคนตอบ"

เด็กคนนี้แหละคือภาพของ LLM ที่เพิ่งอ่านจบทั้งห้องสมุด — รู้เยอะมหาศาล แต่ยังไม่รู้จะเป็น "ผู้ช่วย" ยังไง และเรื่องของบทนี้คือ เราพาเด็กคนนี้เดินทางจากนักอ่านที่เก่งกาจ มาเป็นผู้ช่วยที่คุณคุยด้วยทุกวันได้ยังไง

เกมเดียวที่ใช้สอนทั้งห้องสมุด

ในบทก่อนเราทิ้งท้ายไว้ว่า เรามี Transformer ที่เป็นเหมือนเครื่องยนต์ทรงพลังอยู่ในมือ แต่มันยังว่างเปล่า ไม่รู้ภาษาสักคำ คำถามคือเราสอนภาษาให้มันยังไง

คำตอบเรียบง่ายจนน่าตกใจ เราป้อนข้อความปริมาณมหาศาลให้มัน แล้วฝึกมันด้วย "เกม" เดียว — เกมทายคำถัดไป เกมเดียวกับที่เราเล่ามาตั้งแต่บทเรื่องสถิติ และนี่คือกระดูกสันหลังของหนังสือทั้งเล่ม

วิธีฝึกคือเอาประโยคจริงมา ปิดคำถัดไปไว้ แล้วให้โมเดลเดาว่าคำไหนน่าจะมา ถ้าเดาผิดก็ปรับตัวเองนิดหน่อย ถ้าเดาถูกก็เก็บรูปแบบนั้นไว้ ทำแบบนี้ซ้ำ ไม่ใช่ร้อยครั้งพันครั้ง แต่เป็นพันล้านครั้ง กับข้อความจากอินเทอร์เน็ต หนังสือ และวิกิพีเดียที่กองรวมกันมหาศาล

ขั้นตอนนี้แหละที่เรียกว่า **pretraining**

> **ศัพท์ที่เพิ่งเจอ Pretraining (พรีเทรนนิง)** — ช่วงที่โมเดลอ่านข้อความปริมาณมหาศาลเพื่อ "เรียนภาษา" ก่อนจะมาตอบเรา โดยฝึกด้วยการทายคำถัดไปซ้ำ ๆ เป็นพันล้านครั้ง ผลที่ได้คือโมเดลที่รู้ภาษาและรู้เรื่องโลก แต่ยังไม่ได้ถูกสอนให้เป็นผู้ช่วย

มีจุดที่คนเข้าใจผิดบ่อยตรงนี้ หลายคนคิดว่าโมเดล "จำ" ทุกหน้าที่อ่านไว้เหมือนเครื่องสแกนยักษ์ที่เก็บอินเทอร์เน็ตทั้งก้อนไว้ในเครื่อง ความจริงไม่ใช่แบบนั้น มันไม่ได้เก็บข้อความไว้ตรง ๆ แต่ "อัดสรุป" รูปแบบของภาษาเอาไว้ — ไวยากรณ์ ความหมายของคำ ความเชื่อมโยงระหว่างแนวคิด และข้อเท็จจริงเกี่ยวกับโลก

พูดง่าย ๆ คือ มันเหมือนคนที่อ่านมาเยอะจนพูดได้คล่องและเล่าเรื่องได้ ไม่ใช่คนที่ท่องจำทุกหน้าได้แบบถ่ายเอกสาร

> **จำง่าย ๆ ว่า:** pretraining ไม่ใช่การ "ยัดข้อมูลเข้าเครื่อง" แต่เป็นการเล่นเกมทายคำถัดไปกับข้อความทั้งห้องสมุด จนโมเดลซึมซับรูปแบบของภาษาและโลกเอาไว้ในตัวเอง

ยิ่งโตยิ่งเปลี่ยน — เรื่องของตระกูล GPT

ที่นี้ก็มีคำถามต่อมาว่า ถ้าวิธีฝึกมันง่ายแค่นี้ ทำไมเพิ่งมาเก่งเอาตอนนี้ คำตอบส่วนใหญ่อยู่ที่คำเดียว — ขนาด

ลองดูตระกูลโมเดลที่ชื่อ GPT ของบริษัท OpenAI เป็นตัวอย่าง เพราะมันคือเส้นทางตรงที่นำไปสู่ ChatGPT

GPT-1 ออกมาในปี 2018 มีขนาดราว 117 ล้าน parameters — ตัวเลขนี้คือจำนวน "ปุ่มปรับจูน" ที่โมเดลค่อย ๆ หมุนระหว่างฝึก (เราเคยเล่าภาพปุ่มปรับจูนนี้ไว้ในบทโครงข่ายประสาทเทียม) GPT-1 ยังไม่ได้ทำอะไรหวือหวา แต่มันพิสูจน์ว่าแนวคิด pretraining ใช้ได้จริง

GPT-2 ตามมาในเดือนกุมภาพันธ์ 2019 ใหญ่ขึ้นเป็นราว 1.5 พันล้าน parameters หรือราว 10 เท่าของรุ่นก่อน คราวนี้มันเขียนข้อความต่อได้ลื่นและน่าเชื่อจน OpenAI กังวลว่าจะถูกเอาไปสร้างข่าวปลอมหรือสแปม เลยเลือกปล่อยแค่รุ่นเล็กก่อนในตอนแรก

GPT-3 มาในเดือนพฤษภาคม 2020 กระโดดไปที่ 175 พันล้าน parameters ฝึกด้วยข้อความราว 570 พันล้าน token จากหลายแหล่งปนกัน ทั้ง Common Crawl (ข้อความจากเว็บทั่วอินเทอร์เน็ต) หนังสือ และวิกิพีเดีย

แล้วเรื่องที่น่าสนใจก็คือ พอโมเดลใหญ่ถึงระดับหนึ่ง ความสามารถใหม่ ๆ เริ่ม "โผล่" ขึ้นมาเองโดยไม่มีใครสอนตรง ๆ มันเริ่มแปลภาษาได้ ตอบคำถามได้ เขียนโค้ดได้ ทั้งที่ไม่เคยถูกฝึกเฉพาะสำหรับงานเหล่านั้น นักวิจัยเรียกปรากฏการณ์นี้ว่า emergent abilities หรือ "ความสามารถที่ผุดขึ้น" เหมือนเด็กที่อ่านหนังสือมาถึงจุดหนึ่งแล้วจู่ ๆ ก็เริ่มเชื่อมโยงเรื่องต่าง ๆ เข้าด้วยกันได้เอง

แต่ — และนี่คือจุดพลิกของทั้งบท — GPT-3 ขนาดยักษ์ตัวนี้ ยังไม่ใช่ ChatGPT

รู้เยอะ แต่ไม่รู้มารยาท

GPT-3 ที่ผ่านแค่ pretraining เราเรียกมันว่า **base model** หรือโมเดลดิบ มันเก่งมากในเรื่องเดียว คือ "ต่อข้อความ" ให้ฟังดูเป็นไปได้ เพราะนั่นคือเกมเดียวที่มันถูกฝึกมาทั้งชีวิต

ปัญหาคือ "ต่อข้อความ" กับ "ตอบคำถาม" ไม่ใช่เรื่องเดียวกัน

ลองนึกภาพคุณพิมพ์ถาม base model ว่า "เมืองหลวงของฝรั่งเศสคืออะไร" แทนที่มันจะตอบสั้น ๆ ว่า "ปารีส" มันอาจต่อข้อความออกไปว่า "เมืองหลวงของอิตาลีคืออะไร เมืองหลวงของเยอรมนีคืออะไร" — เพราะในข้อมูลที่มันอ่านมา ประโยคหน้าตาแบบนี้มักเป็นชุดคำถามในแบบฝึกหัด มันเลยเดาว่าสิ่งที่ "น่าจะตามมา" คือคำถามข้อถัดไป ไม่ใช่คำตอบ

มันไม่ได้โง่ มันแค่ไม่เคยถูกสอนว่าเวลาเจอคำถาม คุณควรหยุดแล้วตอบ ไม่ใช่เขียนต่อ นี่คือเด็กห้องสมุดของเราเป๊ะ ๆ — รู้ว่าปารีสคือเมืองหลวงฝรั่งเศสแน่นอน แต่ไม่รู้ "มารยาท" ของการเป็นผู้ช่วย

แถม base model อย่าง GPT-3 ยังมีปัญหาอื่นอีก บางครั้งมันสร้างเนื้อหาที่ไม่จริง บางครั้งสร้างข้อความที่หยาบคายหรือเป็นอันตราย และบ่อยครั้งก็ตอบไม่ตรงกับสิ่งที่ผู้ใช้ต้องการ มันเหมือนคนอัจฉริยะที่อ่านมาทั้งโลก แต่ปล่อยไปคุยกับลูกค้าเองไม่ได้ ต้องฝึกอีกพอสมควร

แล้วเราฝึก "มารยาท" ให้โมเดลยังไง? นี่คือสองขั้นถัดไป

ขั้นที่สอง — สอนให้ฟังคำถามแล้วตอบ

ขั้นแรกของการเปลี่ยนนักอ่านให้เป็นผู้ช่วย เรียกว่า **instruction tuning** หรือการสอนให้ทำตามคำสั่ง

วิธีก็ตรงไปตรงมา ทีมงานเตรียมตัวอย่างจำนวนมากที่มีคู่กันสองอย่าง — "คำสั่งหรือคำถาม" กับ "คำตอบที่ดีที่คนเขียนเอาไว้ให้" เช่น คำสั่งว่า "สรุปย่อหน้านี้" คู่กับตัวอย่างคำสรุปที่ดี แล้วเอาตัวอย่างพวกนี้มาฝึกโมเดลต่อจาก base model ให้มันซึมซับว่า "อ๋อ เวลาเจอคำสั่งหน้าตาแบบนี้ สิ่งที่ควรทำคือตอบแบบนี้"

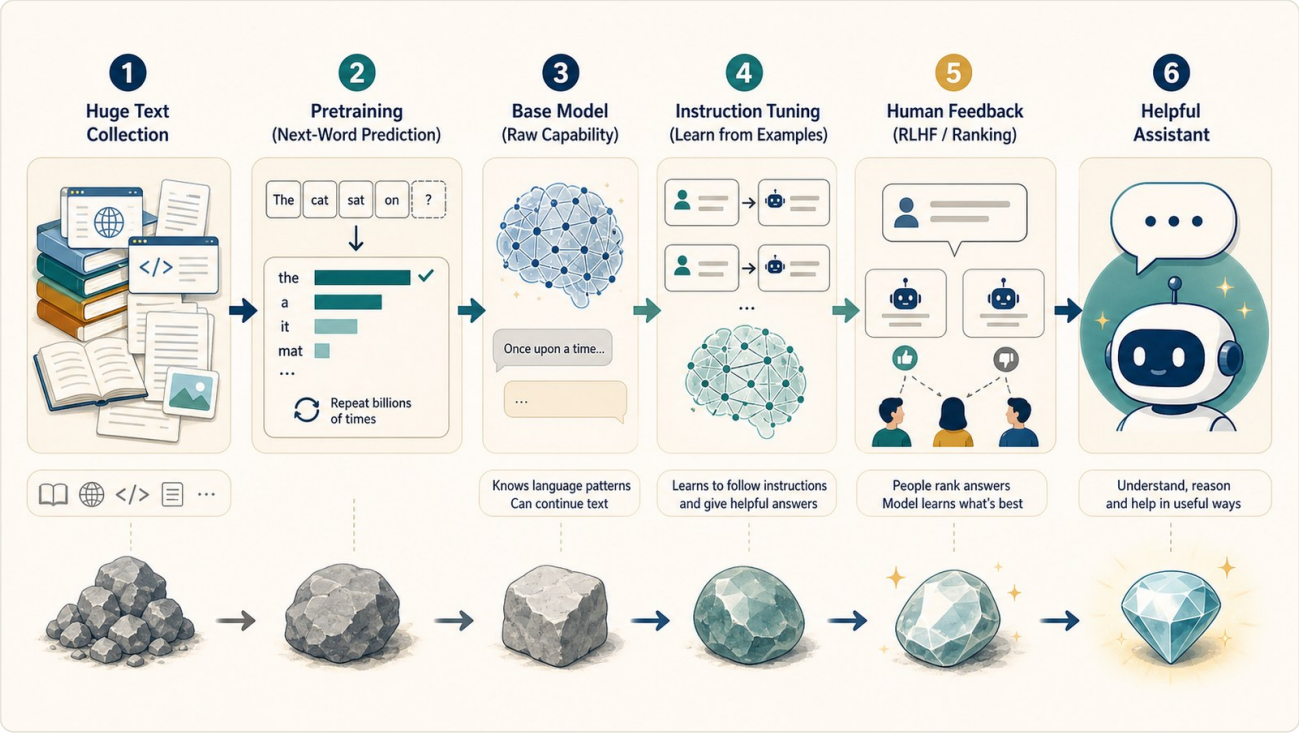
การฝึกต่อจาก base model ด้วยข้อมูลเฉพาะแบบนี้ มีชื่อเรียกรวม ๆ ว่า fine-tuning

> **ศัพท์ที่เพิ่งเจอ Fine-tuning (ไฟน์-ทูนนิง)** — การฝึก "ต่อยอด" จาก base model ด้วยข้อมูลเฉพาะทาง เพื่อปรับพฤติกรรมของโมเดลไปในทางที่เราต้องการ เปรียบเหมือนคนที่จบมหาวิทยาลัยมาแล้ว (pretraining) แล้วไปเข้าคอร์สฝึกงานเฉพาะตำแหน่งต่อ (fine-tuning)

พอผ่าน instruction tuning โมเดลก็เริ่มเปลี่ยนนิสัย จากที่เคยเจอคำถามแล้วเขียนต่อ ตอนนี้มันเริ่ม "ฟัง" แล้วตอบให้ตรงประเด็น เด็กห้องสมุดของเราเริ่มเข้าใจว่าเวลามีคนถาม เขาควรตอบ

แต่การตอบ "ตรงประเด็น" กับการตอบ "ให้ดีในแบบที่คนชอบ" ก็ยังไม่เหมือนกันอีก คำตอบสองแบบอาจตรงคำถามทั้งคู่ แต่แบบหนึ่งอาจชัดเจน สุภาพ และเป็นประโยชน์กว่าอีกแบบมาก เราจะสอนเรื่องนี้ยังไง?

ขั้นที่สาม — เรียนจากความชอบของคน



จาก pretraining สู่ AI assistant ทีละขั้น

ขั้นสุดท้ายคือการให้โมเดลเรียนจาก "ความเห็นของมนุษย์" โดยตรง

ภาพง่าย ๆ คือแบบนี้ ให้โมเดลลองตอบคำถามเดียวกันออกมาหลายแบบ แล้วให้คนจริง ๆ มาช่วยจัดอันดับว่าคำตอบไหนดีกว่าคำตอบไหน — อันไหนมีประโยชน์กว่า อันไหนปลอดภัยกว่า อันไหนตรงใจคนถามกว่า จากนั้นโมเดลก็ค่อย ๆ ปรับตัวเองให้ออกคำตอบในทางที่คนชอบมากขึ้น

วิธีนี้มีชื่อทางเทคนิคว่า RLHF (การเรียนรู้จากผลป้อนกลับของมนุษย์) แต่เราไม่ต้องลงรายละเอียดกลไกเบื้องหลังเลย จำแค่หัวใจของมันพอ — **โมเดลเรียนว่าคำตอบแบบไหนคนชอบมากกว่า โดยมีคนคอยช่วยจัดอันดับให้**

งานวิจัยชิ้นสำคัญที่วางรากของเรื่องนี้คือ InstructGPT ของ OpenAI ในปี 2022 (Ouyang และคณะ) ซึ่งเอา GPT-3 มาผ่านทั้ง instruction tuning และการเรียนจาก feedback ของมนุษย์

และผลที่ได้น่าทึ่งมาก

ในการประเมินของงานวิจัยนั้น มนุษย์ที่มาให้คะแนน "ชอบ" คำตอบจาก InstructGPT รุ่นเล็กขนาด 1.3 พันล้าน parameters มากกว่าคำตอบจาก GPT-3 ดิบ ๆ ขนาด 175 พันล้าน parameters

หยุดคิดตรงนี้สักครู่ โมเดลที่เล็กกว่าเกือบ 100 เท่า กลับมีประโยชน์กับคนมากกว่า เพียงเพราะมันถูกสอนให้ฟังและตอบในแบบที่คนต้องการ

> **จำง่าย ๆ ว่า:** การอ่านมาเยอะไม่ได้แปลว่าจะตอบคำถามได้ดี โมเดลที่ใหญ่กว่า 100 เท่า อาจมีประโยชน์น้อยกว่าโมเดลที่ถูกสอนให้ฟังคำถามเป็น — วิธีฝึกสำคัญไม่แพ้ขนาด

ChatGPT ที่เปิดตัวเมื่อวันที่ 30 พฤศจิกายน 2022 ก็คือผลลัพธ์ของเส้นทางนี้ — โมเดลที่ผ่านครบทั้งสามขั้น (รายละเอียดเล็กน้อย: ตัวที่อยู่เบื้องหลัง ChatGPT รุ่นแรกคือ GPT-3.5 ซึ่งเป็น GPT รุ่นที่ปรับปรุงต่อมา ไม่ใช่ GPT-3 ดิบตัวเดิม) และคนก็แห่กันเข้ามาใช้เร็วอย่างที่ไม่เคยมีมาก่อน ได้ผู้ใช้หนึ่งล้านคนภายในห้าวัน และตามการประมาณการของ UBS ก็แตะ 100 ล้านคนในราวสองเดือน เร็วกว่าแอปดังในอดีตอย่าง TikTok หรือ Instagram มาก

เก่งจริงในเรื่องไหน

ทีนี้พอเรารู้ว่ามันถูกสร้างมายังไง เราก็จะเดาออกว่ามันเก่งอะไรและพลาดตรงไหน

LLM ที่ผ่านสามขั้นนี้มาเก่งในงานที่เกี่ยวกับ "การจัดการภาษา" เป็นพิเศษ เช่น

- **สรุป** ข้อความยาว ๆ ให้สั้นและจับใจความ
- **เขียนและเรียบเรียง** ร่างอีเมล ร่างบทความ ปรับสำนวน
- **แปลภาษา** จากภาษาหนึ่งไปอีกภาษา
- **อธิบาย** แนวคิดยาก ๆ ด้วยภาษาที่ง่ายลง หรืออธิบายในมุมใหม่
- **ช่วยคิดและช่วยร่าง** ไอเดีย โครงเรื่อง หรือโครงโค้ดเบื้องต้น

สังเกตไหมว่างานพวกนี้มีอะไรร่วมกัน — มันคืองานที่ "วัตถุดิบอยู่ในมือเราอยู่แล้ว" หรือเป็นความรู้ทั่วไปที่เปลี่ยนแปลงช้า ตรงนี้แหละที่ LLM ทำได้ดีและเชื่อถือได้มากกว่า

ปัญหาเริ่มตรงที่อีกฝั่งหนึ่ง

เมื่อมันเดาให้ฟังดูดี แทนที่จะบอกว่าไม่รู้

AI hallucination: ตอบคล่องและมั่นใจ แม้ข้อเท็จจริงอาจผิด

จุดอ่อนที่สำคัญที่สุดที่คุณต้องรู้จักมีชื่อว่า hallucination

> **ศัพท์ที่เพิ่งเจอ Hallucination (ฮัลลูซิเนชัน)** — เวลาที่โมเดลสร้างคำตอบที่ฟังดูสมเหตุสมผลและมั่นใจมาก แต่ความจริงแล้วผิดหรือไม่มีอยู่จริง สำคัญตรงที่ — มันไม่ใช่การ "โกหก" เพราะการโกหกต้องตั้งใจ แต่นี่คือผลพลอยได้ตามธรรมชาติของเครื่องที่ทำงานด้วยการ "ทายคำที่น่าจะเป็นไปได้ทางภาษา" โดยไม่มีกลไกตรวจสอบข้อเท็จจริงในตัวเอง

ลองย้อนกลับไปที่เด็กห้องสมุด ถ้ามีคนถามเรื่องที่ไม่มีในหนังสือเล่มไหนที่เขาอ่าน เขาไม่ได้ถูกสอนให้พูดว่า "เรื่องนี้ไม่ทราบ" สิ่งที่กลไกในหัวเขาทำโดยอัตโนมัติคือ ประกอบคำที่ "น่าจะตามกันมา" ออกมาเป็นคำตอบที่ฟังดูเนียนมาก — ด้วยน้ำเสียงมั่นใจเท่ากับตอนที่เขาตอบเรื่องที่รู้จริง

เรื่องนี้ไม่ใช่ทฤษฎีลอย ๆ ปี 2023 ทนายความคนหนึ่งในสหรัฐฯ ใช้ ChatGPT ช่วยร่างเอกสารยื่นศาล ChatGPT สร้างการอ้างอิงคดีความขึ้นมาหลายคดี ครบทั้งชื่อคดี ปี และศาล ดูสมบูรณ์แบบทุกอย่าง — ปัญหาเดียวคือคดีเหล่านั้นไม่มีอยู่จริง ทนายยื่นเอกสารไปโดยไม่ได้ตรวจสอบ สุดท้ายเรื่องแดงในชั้นศาลและถูกลงโทษ คดีนี้รู้จักกันในชื่อ Mata v. Avianca

นี่คือ hallucination ในชีวิตจริงที่มีราคาต้องจ่าย และมันโยงไปสู่จุดอ่อนข้อที่สอง

โมเดลไม่รู้ว่าตัวเองไม่รู้ ตามปกติแล้ว LLM ไม่มีกลไกที่เชื่อถือได้ในการบอกว่า "คำตอบนี้ฉันมั่นใจแค่ไหน" มันตอบทุกอย่างด้วยน้ำเสียงมั่นใจเท่ากันหมด ไม่ว่าจะถูกหรือผิด ดังนั้นความมั่นใจในน้ำเสียงของมัน จึงไม่ใช่สัญญาณว่าข้อมูลถูกต้อง — ข้อนี้เป็นเรื่องที่คนพลาดกันบ่อยที่สุด

จุดอ่อนข้อที่สามคือ **knowledge cutoff** หรือวันตัดข้อมูล โมเดลเรียนจากข้อความที่รวบรวมมาถึงวันหนึ่งเท่านั้น เรื่องที่เกิดขึ้นหลังวันนั้นมันไม่รู้เลย เหมือนสารานุกรมเล่มที่พิมพ์ไว้เมื่อหลายปีก่อน — รู้ทุกอย่างจนถึงวันพิมพ์ แต่ไม่รู้อะไรที่เกิดหลังจากนั้น และที่อันตรายกว่าสารานุกรมคือ ถ้าคุณถามเรื่องใหม่ที่อยู่หลังเส้นวันตัดข้อมูล มันมักไม่บอกว่า "ไม่มีข้อมูล" แต่จะ "เดา" คำตอบที่ฟังดูดีออกมาแทน

RLHF ไม่ได้ทำให้มันซื่อสัตย์ขึ้น 100%

ตรงนี้ขอชวนระวังความเข้าใจผิดที่พบบ่อย หลายคนคิดว่าในเมื่อ ChatGPT ผ่านการเรียนจาก feedback ของมนุษย์มาแล้ว มันก็น่าจะ "เลิก" hallucinate ไปแล้วสิ

ความจริงไม่ใช่แบบนั้น

การเรียนจาก feedback ของมนุษย์ช่วยให้โมเดลมีประโยชน์ขึ้น ตอบตรงขึ้น และปลอดภัยขึ้นในหลายมิติจริง แต่มันไม่ได้แก้ปัญหา hallucination ให้หมดไป ที่น่าแปลกใจคือ ในงานวิจัย InstructGPT เอง พบว่าการเพิ่มขั้นเรียนจาก feedback กลับทำให้โมเดล hallucinate มากขึ้นเล็กน้อยเมื่อเทียบกับการทำแค่ instruction tuning อย่างเดียว — ถึงอย่างนั้น โดยรวมแล้วคนก็ยังชอบผลลัพธ์ของมันมากกว่าอยู่ดี เพราะด้านอื่น ๆ ดีขึ้นชัดเจน

พูดอีกแบบคือ ขั้นตอนนี้ทำให้โมเดล "มารยาทดีขึ้น" แต่ไม่ได้ทำให้มัน "ซื่อสัตย์ขึ้น" โดยอัตโนมัติ และมีงานวิจัยที่เสนอว่า hallucination เป็นข้อจำกัดพื้นฐานของโมเดลแบบทายคำถัดไป ที่ลดได้แต่กำจัดให้หมดเกลี้ยงไม่ได้

นี่คือเหตุผลว่าทำไมเราถึงต้องมีวิธีใช้มันอย่างฉลาด

ใช้ให้เป็น — เมื่อไหร่ควรเชื่อ เมื่อไหร่ต้องตรวจ

ข่าวดีคือ พอเข้าใจว่ามันทำงานยังไง คุณก็เดาออกแล้วว่าควรไว้ใจมันตรงไหน และต้องตรวจสอบตรงไหน ลองใช้เช็กลิสต์นี้เป็นเข็มทิศ

เช็กลิสต์: เชื่อ LLM ได้แค่ไหน

เชื่อได้มากกว่า (แต่งานสำคัญก็ยังควรเหลือบตรวจ):

- ☐ ความรู้ทั่วไปที่เปลี่ยนแปลงช้า (วิทยาศาสตร์พื้นฐาน ประวัติศาสตร์ที่รู้กันทั่วไป)
- ☐ งานจัดการภาษาที่วัตถุดิบอยู่ในมือคุณแล้ว (สรุป แปล เรียบเรียง อธิบาย)
- ☐ การช่วยระดมไอเดียและมุมมองกว้าง ๆ

ต้องตรวจสอบเสมอก่อนนำไปใช้จริง:

- [ ] ข้อเท็จจริงเฉพาะเจาะจง — ตัวเลข วันที่ ชื่อคน ที่อยู่ ราคา
- [ ] เรื่องที่เปลี่ยนเร็ว — ข่าว กฎหมาย ราคา ข้อมูลยา
- [ ] เหตุการณ์ล่าสุด ที่อาจอยู่หลังวันตัดข้อมูล
- [ ] การอ้างอิงและลิงก์ทุกชนิด — เปิดดูเองเสมอ อย่าเชื่อทันที
- [ ] เรื่องการแพทย์ กฎหมาย และการเงินที่ต้องตัดสินใจจริง

แล้วจะ "จับสัญญาณ" ว่าคำตอบนี้อาจเป็น hallucination ได้ยังไง? ลองสังเกตสามอย่างนี้

หนึ่ง มันตอบเรื่องที่คุณตรวจสอบเองได้ยาก ด้วยรายละเอียดที่ "เฉพาะเจาะจงเกินไป" เช่น เลขสถิติเป๊ะ ๆ หรือชื่องานวิจัยพร้อมปี สอง ถ้าคุณถามคำถามเดิมซ้ำในแชตใหม่ แล้วได้คำตอบที่ไม่ตรงกัน นั่นเป็นสัญญาณว่ามันกำลังเดา ไม่ใช่ดึงข้อเท็จจริง สาม เวลามันให้ลิงก์หรือการอ้างอิงมา — ให้ถือว่าต้องเปิดดูจริงทุกครั้ง เพราะนี่คือจุดที่มันสร้างของปลอมขึ้นมาบ่อยที่สุด

ลองทำดู: ทดสอบขอบของโมเดลเอง

1. ถามเรื่องที่เปลี่ยนแปลงช้าและรู้กันทั่วไป เช่น "อธิบายทฤษฎีบทพีทาโกรัส" — สังเกตว่าคำตอบมักดีและเชื่อถือได้
2. ถามเรื่องที่เพิ่งเกิดเมื่อไม่กี่วันก่อน เช่นข่าวล่าสุดในประเทศคุณ — สังเกตว่ามันตอบได้จริง หรือเดาขึ้นมา หรือยอมรับว่าไม่รู้
3. ขอให้มันยกงานวิจัยพร้อมปีที่ตีพิมพ์ในหัวข้อเฉพาะทาง แล้วลองค้นจริงว่ามีงานนั้นอยู่ไหม

ทำสามข้อนี้สักครั้ง แล้วคุณจะ "รู้สึก" ได้เองว่าเส้นแบ่งระหว่างเรื่องที่ไว้ใจได้กับเรื่องที่ต้องตรวจอยู่ตรงไหน

กลับมาที่คำถามที่เดินมาทั้งเล่ม

ตั้งแต่บทแรกเราถือคำถามหนึ่งติดตัวมาตลอด — มันเข้าใจจริง หรือแค่เดาคำให้ฟังดูดี? มาถึงบทนี้ เราเห็นคำตอบชัดขึ้นมากแล้ว

สิ่งที่เรียกว่า ChatGPT คือนักทายคำถัดไปที่เก่งกาจอย่างเหลือเชื่อจากการอ่านทั้งห้องสมุด แล้วถูกขัดเกลาให้กลายเป็นผู้ช่วยที่ฟังคำถามเป็นและตอบในแบบที่คนชอบ มันรู้รูปแบบของภาษาและความเชื่อมโยงของแนวคิดมากมายมหาศาล จนหลายครั้งให้ความรู้สึกเหมือนคุยกับคนที่เข้าใจ

แต่ "ให้ความรู้สึกเหมือนเข้าใจ" กับ "เข้าใจแบบที่คนเราเข้าใจว่าแมวคืออะไร" ยังเป็นคนละเรื่อง สิ่งที่บอกเราชัดที่สุดคือ hallucination — มันเดาคำตอบที่ฟังดูดีออกมาได้โดยไม่รู้ตัวว่าผิด เพราะหัวใจของมันยังคงเป็นการทายคำที่น่าจะเป็นไปได้ ไม่ใช่การค้นหาความจริง

นี่ไม่ใช่การดูถูกมัน มันคือเครื่องมือที่ทรงพลังอย่างที่โลกไม่เคยมีมาก่อน — แค่เราควรรู้ว่ามันเป็นเครื่องมือแบบไหน เพื่อใช้มันได้อย่างที่มันเก่งจริง

สรุปง่าย ๆ ใน 5 ข้อ

1. **Pretraining** คือการเอาข้อความระดับทั้งห้องสมุดมาฝึกโมเดลด้วยเกมเดียว — ทายคำถัดไป — จนมันซึมซับภาษาและความรู้เรื่องโลก ผลที่ได้คือ **LLM** หรือโมเดลภาษาขนาดใหญ่
2. โมเดลดิบ ๆ (base model) รู้เยอะแต่ยังไม่ใช่ผู้ช่วย — มันเอาแต่ "ต่อข้อความ" ไม่ "ตอบคำถาม" เหมือนเด็กที่อ่านมาทั้งโลกแต่ยังไม่รู้มารยาท
3. การเปลี่ยนมันเป็นผู้ช่วยมีอีกสองขั้น — **instruction tuning** (สอนให้ฟังคำสั่งแล้วตอบ) และการเรียนจาก **feedback ของมนุษย์** (คนช่วยจัดอันดับว่าคำตอบไหนดีกว่า) ผลคือ ChatGPT
4. วิธีฝึกสำคัญไม่แพ้ขนาด — ในงานวิจัย InstructGPT โมเดลที่เล็กกว่าเกือบ 100 เท่าแต่ถูกขัดเกลามาดีกลับมีประโยชน์กับคนมากกว่าโมเดลยักษ์ที่ยังดิบ
5. มันเก่งเรื่องสรุป เขียน แปล อธิบาย แต่พลาดเรื่อง **hallucination** (เดาให้ฟังดูดีทั้งที่ผิด), ไม่รู้ว่าตัวเองไม่รู้, และข้อมูลล้าสมัยหลังวันตัดข้อมูล — การเรียนจาก feedback ช่วยให้ดีขึ้น แต่ไม่ได้แก้หมด

ก่อนไปบทต่อไป

ตอนนี้เรามีผู้ช่วยที่คุยรู้เรื่องอยู่ในมือแล้ว แต่บทนี้ก็เผยจุดอ่อนของมันออกมาครบสามข้อ — มันไม่รู้เรื่องใหม่หลังวันตัดข้อมูล มันยัง hallucinate อยู่ และเอาเข้าจริง มันก็ได้แต่ "พูด" ลงไปทำงานในโลกจริงเองไม่ได้ เปิดเว็บค้นข้อมูลใหม่ก็ไม่ได้ กดจองตั๋วก็ไม่ได้

เหมือนสมองที่ฉลาดมากแต่ถูกขังอยู่ในห้องที่ไม่มีหน้าต่าง — รู้เยอะ แต่เอื้อมไปแตะโลกข้างนอกไม่ได้

แล้วโลกแก้ปัญหานี้ยังไง? คำตอบคือ ยื่นหนังสืออ้างอิงให้มันเปิดก่อนตอบ (RAG) ยื่นเครื่องมือให้มันใช้ (tools) และในที่สุดก็ปล่อยให้มันวางแผนแล้วลงมือทำงานหลายขั้นเองได้ (Agent) — นั่นคือก้าวถัดไปที่เรากำลังอยู่กันในตอนนี้ และเป็นเรื่องของบทสุดท้าย

บทที่ 9 — หลังจาก LLM โลกกำลังไปทางไหน

ลองนึกภาพเช้าวันหนึ่ง คุณเปิด ChatGPT แล้วพิมพ์ถามว่า "ข่าวใหญ่เมื่อวานนี้คืออะไรบ้าง" แล้วมันตอบกลับมาว่า "ขออภัย ฉันไม่มีข้อมูลเหตุการณ์ล่าสุด"

หรือบางทีอาจแย่กว่านั้น — มันตอบมาเป็นฉาก ๆ ด้วยน้ำเสียงมั่นใจเต็มร้อย ชื่อคน ตัวเลข วันที่ครบ ฟังดูน่าเชื่อทุกอย่าง พอคุณลองไปเช็กดูจริง ๆ กลับไม่มีข่าวนั้นอยู่เลย มันแต่งขึ้นมาเอง

แล้วคุณอาจจะนึกแย้งในใจว่า "เอ๊ะ แต่ ChatGPT เดี๋ยวนี้มันค้นเว็บได้นี่นา" — ใช่ คุณอาจเคยเห็นมันหยุดคิดแล้วเด้งขึ้นมาว่า "กำลังค้นหา..." ก่อนตอบ จำความรู้สึกนั้นไว้ก่อน เพราะนั่นแหละคือตัวอย่างของสิ่งที่เรากำลังจะเล่ากันทั้งบทนี้พอดี

คำถามที่น่าสนใจกว่าคือ — ทำไมตั้งแต่แรก มันถึงตอบเรื่องข่าวเมื่อวานไม่ได้? และโลกแก้ปัญหานี้ยังไง?

สมองที่ฉลาดแต่ขังอยู่ในห้องไม่มีหน้าต่าง

บทที่แล้วเราจบลงตรงภาพนี้พอดี — ผู้ช่วยที่เก่งกาจที่เราคุยอยู่ทุกวัน เอาเข้าจริงเหมือนสมองที่ฉลาดมากแต่ถูกขังอยู่ในห้องที่ไม่มีหน้าต่าง

ลองนึกถึงนักวิชาการที่ฉลาดที่สุดในโลกคนหนึ่ง เขาอ่านหนังสือมานับล้านเล่ม จำได้แทบทุกบรรทัด ตอบได้แทบทุกเรื่อง แต่เขาถูกขังอยู่ในห้องปิดมาหลายเดือนแล้ว ไม่มีหน้าต่าง ไม่มีโทรศัพท์ ไม่มีอินเทอร์เน็ต

คนแบบนี้จะตอบเรื่องข่าวเมื่อวานได้ยังไง ในเมื่อทุกอย่างที่เขารู้คือสิ่งที่เขาอ่านมา *ก่อน* เข้าห้อง? และต่อให้เขาฉลาดแค่ไหน เขาก็เอื้อมมือออกไปจองโต๊ะร้านอาหารให้คุณไม่ได้ เพราะตัวเขาอยู่ในห้อง

นี่คือข้อจำกัดสามข้อของ LLM ที่เราทิ้งค้างไว้เมื่อบทก่อน สรุปสั้น ๆ อีกที

หนึ่ง — มันไม่รู้เรื่องใหม่หลังวันตัดข้อมูล (knowledge cutoff คือวันสุดท้ายที่ข้อมูลฝึกของมันหยุดอยู่) ข่าวเมื่อวานจึงอยู่นอกความรู้ของมัน

สอง — เมื่อถูกถามเรื่องที่มันไม่รู้จริง มันมักจะเดาคำตอบที่ฟังดูดีออกมาแทนที่จะบอกว่าไม่รู้ นี่คือ hallucination ที่เรารู้จักกันแล้ว

สาม — มันได้แต่ "พูด" ลงมือทำอะไรในโลกจริงเองไม่ได้ ส่งอีเมล กดจอง คำนวณเลขยาก ๆ ค้นข้อมูลสด — ทำเองไม่ได้สักอย่าง

ข่าวดีคือ ตลอดไม่กี่ปีที่ผ่านมา โลกหาวิธี "เปิดประตูห้อง" ออกทีละบาน และมันก็คือสามคำที่คุณน่าจะได้ยินบ่อยขึ้นเรื่อย ๆ — RAG, tools และ Agent

เปิดประตูบานแรก — ยื่นหนังสืออ้างอิงให้มันเปิดก่อนตอบ (RAG)

บานแรกที่ง่ายที่สุดที่จะเข้าใจ คือยื่นหนังสือเข้าไปในห้องให้นักวิชาการของเราเปิดอ่านก่อนตอบ

เทคนิคนี้ชื่อว่า **RAG** ย่อมาจาก Retrieval-Augmented Generation แปลเป็นภาษาคนคือ "การสร้างคำตอบโดยมีการดึงข้อมูลมาช่วย" — แทนที่จะให้โมเดลตอบจากความจำล้วน ๆ เราให้มันไปค้นเอกสารที่เกี่ยวข้องมาก่อน แล้วค่อยตอบโดยอิงเอกสารนั้นเป็นหลัก

วิธีที่เข้าใจง่ายที่สุดคือเทียบกับการสอบ

LLM ปกติเหมือนสอบแบบ *ปิดตำรา* — ห้ามเปิดหนังสือ ต้องตอบจากความจำที่อ่านมาล้วน ๆ ถ้าจำผิดหรือลืม ก็ตอบผิด ส่วน RAG เหมือนเปลี่ยนให้เป็นสอบแบบ *เปิดตำรา* — ก่อนตอบ มันเปิดเอกสารที่เกี่ยวข้องดูก่อน แล้วใช้เนื้อหาในนั้นช่วยเรียบเรียงคำตอบ

แนวคิดนี้ไม่ใช่ของใหม่เอี่ยม มันถูกเสนอตั้งแต่ปี 2020 โดย Patrick Lewis และทีมนักวิจัยกว่าสิบคนจาก Meta AI, UCL และ NYU ในงานวิจัยที่ตีพิมพ์ที่ NeurIPS 2020 ซึ่งเป็นหนึ่งในงานประชุมวิจัย AI ที่ใหญ่ที่สุดในโลก มีเกร็ดน่ารักอยู่นิดหนึ่ง — ภายหลัง Lewis เคยพูดถึงชื่อย่อ "RAG" อย่างอารมณ์ดีว่า ถ้ารู้ว่ามันจะกลายเป็นคำที่คนใช้กันทั้งวงการขนาดนี้ คงตั้งชื่อให้ฟังดูดีกว่านี้สักหน่อย บางครั้งแนวคิดที่ทรงพลังที่สุดก็มาพร้อมชื่อที่ไม่ได้ตั้งใจให้เท่

แล้ว RAG มีประโยชน์จริงตรงไหน?

ลองนึกถึงบริษัทที่มีคู่มือสินค้า นโยบาย และคำถามที่ลูกค้าถามบ่อยรวมกันหลายร้อยหน้า ปกติ LLM ทั่วไปไม่มีทางรู้เรื่องภายในบริษัทคุณเลย เพราะมันไม่เคยอ่านเอกสารพวกนั้น แต่ถ้าเอาเอกสารทั้งหมดใส่เข้าคลังของระบบ RAG แล้ว เวลาลูกค้าถามอะไร ระบบจะไปค้นหน้าที่เกี่ยวข้องที่สุดออกมาก่อน แล้วส่งให้ LLM ตอบ ผลคือ chatbot ที่ตอบได้ราวกับอ่านคู่มือทั้งเล่มมาแล้ว — โดยไม่ต้องไปฝึกโมเดลใหม่ตั้งแต่ต้น แค่เพิ่มเอกสารลงคลังก็พอ และยังบอกได้ด้วยว่าคำตอบนี้อ้างมาจากเอกสารหน้าไหน เหมือนเชิงอรรถในรายงานวิจัย

นี่คือเหตุผลที่ RAG ช่วยลด hallucination ได้มาก — เพราะแทนที่จะเดาจากความจำ มันมีหลักฐานจริงวางอยู่ตรงหน้าให้อ้างอิง

แต่จุดที่สำคัญคือ — RAG ไม่ใช่ยาวิเศษ

ถ้าเอกสารในคลังไม่ครบ หรือใส่ข้อมูลผิดเข้าไป มันก็ตอบผิดตามเอกสารนั่นแหละ และบางครั้งต่อให้ดึงเอกสารที่ถูกมาวางให้แล้ว โมเดลก็ยังเผลอตอบนอกเรื่องไปจากเอกสารได้อยู่ดี เหมือนยื่นหนังสือที่ถูกต้องให้คนที่อ่านไม่ครบ — ช่วยได้เยอะ แต่ไม่ได้แปลว่าจะไม่พลาดเลย พูดง่าย ๆ คือ RAG ลด hallucination ได้มาก แต่ไม่ได้ทำให้มันหายขาด

> **จำง่าย ๆ ว่า:** RAG คือการเปลี่ยน LLM จากนักเรียนที่สอบปิดตำรา ให้กลายเป็นคนที่เปิดเอกสารอ้างอิงดูก่อนตอบ มันตอบเรื่องเฉพาะทางและเรื่องในองค์กรของคุณได้ดีขึ้นมาก แต่คำตอบจะดีแค่ไหนก็ขึ้นกับว่าเอกสารที่มันหยิบมานั้นดีแค่ไหน

เปิดประตูบานที่สอง — ยื่นเครื่องมือให้มันใช้ (tools)

RAG แก้เรื่อง "ไม่รู้ข้อมูลเฉพาะทาง" ได้ก็จริง แต่ยังมีอีกข้อจำกัดที่หนังสือในห้องแก้ไม่ได้ — นั่นคือบางอย่างมันต้อง *ลงมือทำ* ไม่ใช่แค่ *รู้*

จำเรื่องข่าวเมื่อวานตอนต้นบทได้ไหม? นี่แหละคือจุดที่มันถูกแก้

ความสามารถที่ทำให้ LLM สั่งใช้เครื่องมือภายนอกได้ เรียกว่า **tool use** หรือในเชิงเทคนิคเรียก **function calling** (การเรียกใช้ฟังก์ชัน) แปลเป็นภาษาคนคือ การยอมให้โมเดลกดเรียกเครื่องมือมาช่วยทำงานที่มันทำเองไม่ได้ เช่น เครื่องคิดเลข การค้นเว็บ หรือการเรียกโปรแกรมอื่นให้ทำงานแทน

ลองนึกถึงนักบัญชีที่เก่งมาก แต่คุณไม่ยอมให้เขาถือเครื่องคิดเลข ให้ใช้แต่ความจำในหัว — คำนวณเลขยาว ๆ เขาก็มีโอกาสพลาด พอยื่นเครื่องคิดเลขให้ ทุกอย่างก็แม่นยำขึ้นทันที tool use ก็แบบเดียวกัน มันไม่ได้ทำให้ LLM "ฉลาดขึ้น" แต่ทำให้มันทำงานที่ไม่ถนัดได้ดีขึ้น โดยอาศัยเครื่องมือที่เหมาะกับงานนั้น

และนี่คือคำตอบของเรื่องข่าวเมื่อวาน — พอ LLM ต่อกับเครื่องมือ "ค้นเว็บ" ได้ มันก็ออกไปค้นข้อมูลสดจากอินเทอร์เน็ตมาก่อนตอบได้ ข้อจำกัดเรื่องวันตัดข้อมูลจึงทุเลาลงไปมาก

ของจริงที่คุณน่าจะเคยเจอตรง ๆ ก็คือตอนที่ ChatGPT ค้นเว็บให้นั่นแหละ ย้อนไปในปี 2023 OpenAI ค่อย ๆ เปิดให้ ChatGPT ค้นเว็บได้ — เริ่มจาก ChatGPT Plugins ในเดือนมีนาคม แล้วนำการค้นเว็บผ่าน Bing เข้ามาอย่างเป็นทางการในเดือนตุลาคมปีนั้น ส่วนฝั่งนักพัฒนา OpenAI ประกาศ function calling อย่างเป็นทางการเมื่อวันที่ 13 มิถุนายน 2023 ทำให้คนสร้างแอปสามารถบอกโมเดลได้ว่ามีเครื่องมืออะไรให้มันเรียกใช้บ้าง

นี่คือจุดที่ดูเหมือนเส้นแบ่งจะเริ่มเบลอ หลายคนเลยสับสนว่า RAG กับ tool use มันต่างกันยังไง ในเมื่อทั้งคู่ก็ "ไปหาข้อมูลมาก่อนตอบ"

แยกง่าย ๆ แบบนี้ — RAG คือการดึง *เอกสาร* มาวางเป็นข้อมูลประกอบให้โมเดลอ่านก่อนตอบ ส่วน tool use คือการให้โมเดล *สั่งเครื่องมือทำงาน* ไม่ว่าจะเป็นค้นเว็บ คำนวณ หรือเรียกโปรแกรม การค้นเว็บสด ๆ จึงเป็น tool use ส่วนการตอบจากคู่มือบริษัทคือ RAG ทั้งสองอย่างใช้ร่วมกันได้ แต่เป็นคนละแนวคิด

ข้อควรระวังก็ยังมีอยู่ — การค้นเว็บไม่ได้แปลว่าคำตอบจะถูกเสมอ ถ้าเว็บที่มันค้นมาผิด หรือมันสรุปข้อมูลจากเว็บผิด คำตอบสุดท้ายก็ยังพลาดได้ เครื่องมือช่วยให้มันเอื้อมไปแตะโลกได้ก็จริง แต่หัวใจที่ตีความและเรียบเรียงยังคงเป็น LLM ตัวเดิม

เปิดประตูบานสุดท้าย — ปล่อยให้มันออกไปทำงานเอง (Agent)

ทีนี้ลองคิดเล่น ๆ ว่า ถ้าเราไม่ได้แค่ยื่นหนังสือกับเครื่องมือเข้าไปในห้อง แต่เปิดประตูปล่อยให้นักวิชาการของเรา *เดินออกมา* มีขา มีมือ แล้วลงมือทำงานหลายขั้นด้วยตัวเองล่ะ?

นั่นคือสิ่งที่เรียกว่า **AI Agent** (เอไอที่ทำงานได้ด้วยตัวเอง) — ระบบที่ให้ LLM ไม่ใช่แค่ตอบคำถามเดียวจบ แต่รับงานที่ซับซ้อน วางแผนแบ่งเป็นขั้นย่อย ลงมือทำทีละขั้น ตรวจผล แล้วปรับแผนไปเรื่อย ๆ จนงานสำเร็จ

ถ้า LLM คือสมอง RAG คือหนังสืออ้างอิง และ tools คือมือ — Agent ก็คือการรวมทั้งหมดเข้าด้วยกันให้กลายเป็น "คนทำงาน" ที่ครบเครื่อง คือมีสมอง มีความจำว่าทำอะไรไปแล้ว มีเครื่องมือ และมีอิสระพอจะลงมือเองได้โดยไม่ต้องให้เราสั่งทีละก้าว

หัวใจของ Agent ที่ทำให้มันต่างจาก LLM ธรรมดา คือมันสลับไปมาระหว่าง "คิด" กับ "ทำ" เป็นรอบ ๆ — คิดว่าจะทำอะไรต่อ ลงมือทำ ดูผลที่ได้ แล้วคิดใหม่ตามผลนั้น แนวคิดนี้มีรากมาจากงานวิจัยชื่อ ReAct ของ Shunyu Yao และคณะจาก Google Research และ Princeton ในปี 2022 ที่แสดงให้เห็นว่า LLM ทำงานได้ดีขึ้นมากเมื่อมันรู้จักหาข้อมูลเพิ่มระหว่างทาง แทนที่จะคิดอยู่คนเดียวในหัวแล้วเดาผิด เหมือนนักสืบที่ไม่ได้นั่งคิดอยู่เฉย ๆ แต่ออกไปหาหลักฐาน แล้วกลับมาปรับสมมติฐานตามสิ่งที่เจอ

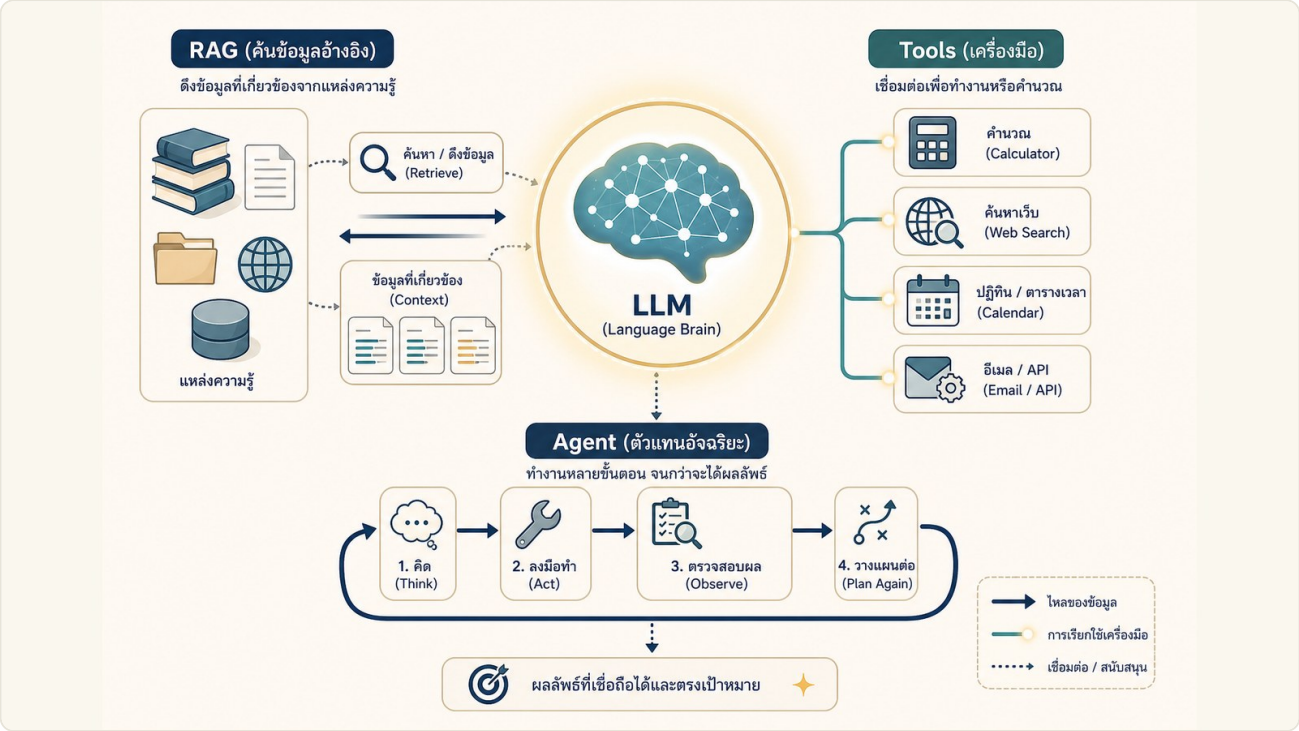
ตัวอย่างจริงที่จับต้องได้ใกล้ตัวคือในวงการประกันภัย หลังเกิดภัยพิบัติทีหนึ่ง บริษัทจะได้รับเรื่องเคลมพร้อมกันเป็นพัน ๆ ราย ซึ่งแต่ละรายต้องผ่านหลายขั้น — ยืนยันกรมธรรม์ ตรวจสอบสภาพอากาศวันเกิดเหตุ ตรวจจับการฉ้อโกง คำนวณค่าชดเชย มีตัวอย่างในวงการที่ใช้ระบบหลาย Agent ทำงานร่วมกัน แต่ละตัวรับผิดชอบขั้นหนึ่ง ช่วยให้ประมวลผลเคลมที่เคยใช้เวลาหลายวันเสร็จเร็วขึ้นอย่างมีนัยสำคัญ แล้วยื่นผลให้มนุษย์ตรวจสอบขั้นสุดท้าย

สังเกตตรงนี้ให้ดี — Agent ที่ใช้งานได้จริงในยุคนี้ ไม่ได้ "แทน" คนทั้งหมด แต่ทำส่วนที่ซ้ำ ๆ และต้องประกอบข้อมูลจากหลายแหล่งให้เร็วขึ้น แล้วส่งให้คนตัดสินใจขั้นสุดท้าย

แล้วของจริงไปถึงไหนแล้ว? ในเดือนตุลาคม 2024 Anthropic เปิดตัวความสามารถที่เรียกว่า computer use ให้ Claude มองหน้าจอ เลื่อนเมาส์ คลิก และพิมพ์ได้เหมือนคนนั่งทำงานหน้าคอม — แต่ตอนเปิดตัว มันยังทำงานบนแบบทดสอบมาตรฐานสำเร็จเพียงราว ๆ หนึ่งในห้าเท่านั้น ตัวเลขนี้บอกอะไรเราชัดเจน — ทิศทางกำลังมุ่งไปทาง "ลงมือทำในโลกจริง" จริง ๆ แต่มันยังอยู่ในช่วงต้น ยังไม่ใช่ของที่ทำได้ทุกอย่างอย่างที่บางพาดหัวข่าวทำให้รู้สึก

และนี่คือจุดที่เราต้องระวังไม่ให้เคลิ้มไปกับกระแส

อย่าเพิ่งเคลิ้มกับคำว่า Agent



RAG, tools และ agent ต่อยอดจาก LLM อย่างไร

ช่วงสองสามปีนี้คำว่า Agent มาแรงมาก จนหลายคนเริ่มจินตนาการว่ามันคือ AI ที่คิดเองได้ทุกเรื่องเหมือนมนุษย์ — ซึ่งยังไม่จริง

วิธีที่ดีที่สุดในการมองเรื่องนี้แบบไม่หลงทาง คือดูทั้งสองด้านที่บริษัทวิเคราะห์อย่าง Gartner คาดการณ์ไว้

ด้านหนึ่ง Gartner จัดให้ Agentic AI เป็นหนึ่งในเทรนด์เทคโนโลยีเชิงกลยุทธ์สำคัญที่สุดของปี 2025 และคาดการณ์ว่าภายในปี 2026 ราว 40% ของแอปพลิเคชันระดับองค์กรจะมี AI agent สำหรับงานเฉพาะด้าน เทียบกับที่มีไม่ถึง 5% ในปี 2025 ฟังดูเหมือนทุกอย่างกำลังพุ่งทะยาน

แต่อีกด้านหนึ่ง — Gartner เองก็เตือนว่า กว่า 40% ของโปรเจกต์ AI agent มีแนวโน้มจะถูกยกเลิกก่อนสิ้นปี 2027 เพราะต้นทุนสูงและยังพิสูจน์ไม่ชัดว่าคุ้มค่าจริงไหม

ตัวเลขทั้งสองนี้เป็น *การคาดการณ์* ของนักวิเคราะห์ ไม่ใช่สิ่งที่เกิดขึ้นแล้ว แต่พอวางคู่กัน มันให้ภาพที่สมดุลและตรงความจริงที่สุด — Agent คือทิศทางที่น่าจับตาจริง แต่ก็ไม่ใช่เวทมนตร์ที่จะทำงานได้ทุกอย่างในชั่วข้ามคืน

และจุดอ่อนเชิงโครงสร้างที่ต้องจำคือ Agent ทำงานหลายขั้นต่อกัน ซึ่งแปลว่าถ้ามันพลาดในขั้นต้น ๆ ความผิดพลาดนั้นจะถูกสะสมและขยายต่อไปในขั้นถัด ๆ ไป ยิ่งปล่อยให้มันทำเองยาว ยิ่งต้องมีจุดที่คนคอยตรวจ — นี่ไม่ใช่ข้อบกพร่องที่จะหายไปง่าย ๆ แต่เป็นธรรมชาติของการทำงานเป็นทอด ๆ

ศัพท์ที่เพิ่งเจอ

RAG (Retrieval-Augmented Generation): วิธีให้ LLM ค้นเอกสารที่เกี่ยวข้องมาอ่านก่อนตอบ เหมือนสอบแบบเปิดตำรา ช่วยให้ตอบเรื่องเฉพาะทาง/ในองค์กรได้ และลด hallucination ได้มาก (แต่ไม่หายขาด)

Tool use / Function calling: การที่ LLM สั่งใช้เครื่องมือภายนอก เช่น ค้นเว็บ คำนวณ หรือเรียกโปรแกรมอื่น เพื่อทำสิ่งที่มันทำเองด้วยการเดาคำไม่ได้ — การค้นเว็บเรื่องข่าวล่าสุดคือตัวอย่างที่ใกล้ตัวที่สุด

AI Agent: ระบบที่ให้ LLM วางแผนและลงมือทำงานหลายขั้นด้วยตัวเอง — คิด ทำ ตรวจ แล้วปรับแผน — ไม่ใช่แค่ตอบคำถามครั้งเดียวจบ

ทั้งเล่มในภาพเดียว

ภาพสรุปเส้นทางทั้งหมดของ AI ภาษา

มาถึงตรงนี้ เราเดินทางมาไกลมากแล้ว ลองถอยออกมาดูแผนที่ทั้งหมดในคราวเดียว

เราเริ่มจากยุคที่มนุษย์ **เขียนกฎ** ให้เครื่องทำตามทีละข้อ (rule-based) — ได้ผลในโลกแคบ ๆ แต่ภาษาจริงมีข้อยกเว้นมากเกินกว่าจะเขียนกฎครบ จากนั้นวงการก็เปลี่ยนมาให้เครื่อง **เรียนจากตัวอย่าง** ด้วยสถิติ (statistical) — นับว่าคำไหนมักตามคำไหน แล้วเดาคำถัดไปจากความน่าจะเป็น นี่คือต้นกำเนิดของแนวคิด "ทำนายคำถัดไป" ที่อยู่ใจกลาง ChatGPT จนทุกวันนี้

ต่อมาเราหาวิธีเปลี่ยนคำให้เป็นตัวเลขที่บอกความหมาย (**embeddings**) คำที่ความหมายใกล้กันก็อยู่ใกล้กันบนแผนที่ความหมาย แล้ว **โครงข่ายประสาทเทียม** (neural networks) ก็เข้ามาช่วยให้เครื่องจับรูปแบบที่ซับซ้อนขึ้นมากจากข้อมูลมหาศาล

จุดพลิกครั้งใหญ่มาถึงในปี 2017 กับ **Transformer** และกลไก attention ที่ให้โมเดลเหลือบมองทั้งประโยคพร้อมกันเพื่อดูว่าคำไหนเกี่ยวกับคำไหน — เปิดทางให้ฝึกโมเดลขนาดมหึมาได้จริง จากนั้นยุคของ **pretraining** ก็มาถึง โมเดลอ่านข้อความระดับทั้งห้องสมุดจนกลายเป็น **LLM** แล้วถูกขัดเกลาด้วย instruction tuning และ feedback ของมนุษย์ จนกลายเป็น **ผู้ช่วย** ที่เราคุยด้วยได้ในชื่อ ChatGPT

และก้าวล่าสุดที่เราอยู่กันตอนนี้ ก็คือการต่อขยายผู้ช่วยตัวนั้นด้วย **RAG, tools และ Agent** — ยื่นหนังสือ ยื่นเครื่องมือ และปล่อยให้มันลงมือทำงานเองได้

กลับมาที่คำถามที่เดินมาตั้งแต่หน้าแรก

ตั้งแต่บทแรกเราถือคำถามหนึ่งติดตัวมาตลอดเล่ม — ChatGPT มันเข้าใจจริง หรือแค่เดาคำเก่งมาก?

ตอนนี้คุณเห็นแผนที่ทั้งหมดแล้ว และคำตอบก็อยู่ในแผนที่นั้นเอง

ChatGPT ไม่ได้เกิดขึ้นชั่วข้ามคืน และไม่ได้เป็นเวทมนตร์ มันคือปลายทาง — อย่างน้อยก็ปลายทาง *ในตอนนี้* — ของการเดินทางหลายสิบปีที่มนุษย์ค่อย ๆ สอนเครื่องให้ทำนายภาษาได้แม่นขึ้นเรื่อย ๆ จากกฎ มาเป็นสถิติ มาเป็นแผนที่ความหมาย มาเป็นโครงข่ายประสาท มาเป็น Transformer แล้วถูกขัดเกลาให้กลายเป็นผู้ช่วยที่มีประโยชน์ และกำลังถูกต่อขยายให้เอื้อมไปแตะโลกจริงได้

มันน่าทึ่งจริง ๆ — แต่มันไม่ใช่ความเข้าใจแบบที่คนเราเข้าใจว่าแมวคืออะไร หัวใจของมันยังคงเป็นการทำนายว่าอะไรน่าจะตามมาต่อ ไม่ใช่การค้นหาความจริง RAG, tools และ Agent ทำให้มันเก่งขึ้น เอื้อมไกลขึ้น และพึ่งพาได้มากขึ้น แต่ไม่ได้เปลี่ยนธรรมชาติข้อนี้

และนั่นแหละคือสิ่งที่ทำให้คุณใช้มันได้ฉลาดกว่าคนทั่วไป — เพราะคุณรู้แล้วว่ามันเป็นเครื่องมือแบบไหน เก่งตรงไหน และต้องระวังตรงไหน

ถ้าอยากเดินต่อจากตรงนี้

หนังสือเล่มนี้ตั้งใจให้คุณ "เห็นภาพทั้งแผนที่" โดยไม่ต้องแตะโค้ดสักบรรทัด และถ้าอ่านมาถึงตรงนี้ คุณก็ทำสำเร็จแล้ว — คุณเล่าให้คนอื่นฟังได้ว่า AI ภาษาเดินทางมายังไง และเลือกใช้มันได้อย่างเข้าใจ

คุณไม่จำเป็นต้องสร้าง RAG หรือ Agent เองเลยก็ใช้ AI ได้ดีขึ้นมากจากความเข้าใจนี้ แต่ถ้าวันหนึ่งคุณเกิดอยากลงมือทำของพวกนี้ขึ้นมาจริง ๆ — อยากสร้าง chatbot ที่รู้เรื่องบริษัทตัวเอง หรืออยากต่อ Agent ให้ทำงานแทน — ข่าวดีคือ มีหนังสือที่ลงลึกกว่านี้รออยู่ ทั้งเรื่อง RAG, AI Agent และการออกแบบระบบ AI ให้ใช้งานได้จริง ไม่ต้องรีบ ค่อยไปเมื่อพร้อม แค่อยากให้รู้ว่าทางข้างหน้ามีอยู่

ลองทำดู: เทสต์ทั้งสามประตูด้วยตัวเอง

1. **ลอง tools:** เปิด ChatGPT (หรือ AI ที่คุณใช้) แล้วถามข่าวที่เพิ่งเกิดเมื่อวาน สังเกตว่ามันค้นเว็บให้ไหม แล้วลองกดดูแหล่งที่มันอ้างว่าจริงหรือเปล่า

2. **ลองนึกถึง RAG:** คิดดูว่างานในชีวิตคุณมีเอกสารกองไหนบ้าง (คู่มือ ระเบียบ โน้ตเก่า) ที่ถ้าให้ AI อ่านก่อนตอบ มันจะช่วยคุณได้จริง — นั่นคือโจทย์ที่ RAG เกิดมาเพื่อแก้
3. **ลองคิดแบบ Agent:** ลองนึกงานหนึ่งที่ต้องทำหลายขั้นต่อกัน แล้วถามตัวเองว่า ขั้นไหนที่ถ้าพลาดแล้วจะลามไปทั้งงาน — ขั้นนั้นแหละคือจุดที่คุณยังอยากให้คนคอยตรวจ ไม่ใช่ปล่อยให้ AI ทำเองทั้งหมด

สรุปง่าย ๆ ใน 5 ข้อ

1. LLM เพียงลำพังมีเพดานสามข้อ — ไม่รู้เรื่องใหม่หลังวันตัดข้อมูล ยัง hallucinate และลงมือทำในโลกจริงเองไม่ได้ เหมือนสมองฉลาดที่ขังอยู่ในห้องไม่มีหน้าต่าง
2. **RAG** = ยื่นหนังสืออ้างอิงให้เปิดก่อนตอบ (สอบเปิดตำรา) ช่วยเรื่องข้อมูลเฉพาะทาง/ในองค์กร และลด hallucination ได้มาก แต่ดีแค่ไหนก็ขึ้นกับเอกสารที่มันหยิบมา
3. **Tools / function calling** = ยื่นเครื่องมือให้ใช้ เช่น ค้นเว็บ คำนวณ เรียกโปรแกรม — นี่คือสิ่งที่ทำให้ ChatGPT ตอบเรื่องข่าวเมื่อวานได้ แต่ถ้าเครื่องมือหาข้อมูลผิดมา คำตอบก็ยังพลาดได้
4. **Agent** = ปล่อยให้มันวางแผนและลงมือทำงานหลายขั้นเอง (คิด → ทำ → ตรวจ → คิดใหม่) น่าจับตามาก แต่ยังอยู่ช่วงต้น และความผิดพลาดสะสมข้ามขั้นได้ — Gartner คาดทั้งว่ามันจะแพร่หลายและว่าหลายโปรเจกต์จะถูกยกเลิก
5. ภาพรวมทั้งเล่ม — กฎ → สถิติ → embeddings → neural nets → Transformer → LLM → ผู้ช่วย → RAG/tools/Agent — บอกเราว่า ChatGPT คือปลายทาง (ในตอนนี้) ของการสอนเครื่องให้ทำนายภาษาเก่งขึ้นเรื่อย ๆ น่าทึ่งมาก แต่ไม่ใช่เวทมนตร์ และไม่ใช่ความเข้าใจแบบมนุษย์

สรุปการเดินทางทั้งเล่ม

จาก “เขียนกฎ” สู่ AI ที่เดาคำได้เหมือนเข้าใจ

ตลอดเล่มนี้ คุณได้เดินผ่านวิวัฒนาการของ AI ภาษา ทีละก้าว:

- **ยุคเขียนกฎ** — มนุษย์พยายามเขียนกฎทุกข้อ แต่ภาษามีข้อยกเว้นไม่รู้จบ
- **ยุคสถิติ** — เครื่องเริ่มเดาคำถัดไปจากความถี่ในข้อมูลจริง
- **คำกลายเป็นตัวเลข** — embedding เปลี่ยนความหมายให้เป็นระยะห่าง
- **โครงข่ายประสาทเทียม** — เครื่องเรียนรู้เองจากตัวอย่าง เดา–เช็ก–ปรับ
- **Transformer & Attention** — กลไกที่ทำให้เครื่องมองทั้งประโยคพร้อมกัน
- **LLM & ผู้ช่วย** — pretraining บวกการสอนให้ฟังคน จนกลายเป็น ChatGPT
- **หลัง LLM** — RAG, tools และ Agent ที่ต่อยอดความสามารถออกไป

“มันเข้าใจจริง หรือแค่เดาคำให้ฟังดูดี?”

คำถามเดียวที่พาคุณเดินมาตลอดทั้งเล่ม
