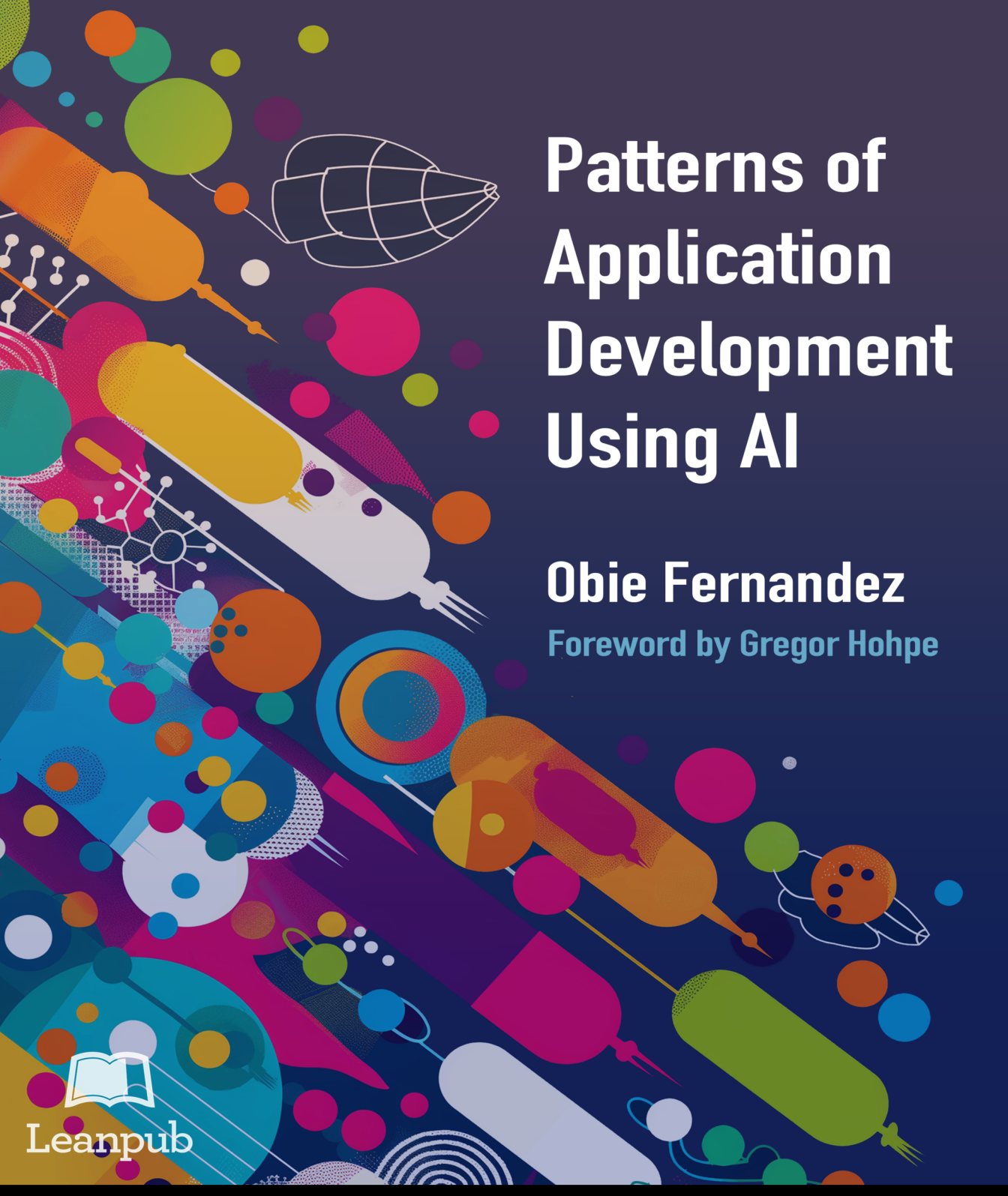


Patterns of Application Development Using AI

Obie Fernandez

Foreword by Gregor Hohpe

Leanpub

ฉบับภาษาไทย

รูปแบบการพัฒนาแอปพลิเคชันด้วยปัญญาประดิษฐ์ (ฉบับภาษาไทย)

Obie Fernandez

หนังสือเล่มนี้มีจำหน่ายที่ http://leanpub.com/patterns-of-application-development-using-ai-th

เวอร์ชันนี้ได้รับการเผยแพร่เมื่อ 2025-01-23

นี่คือหนังสือจาก Leanpub Leanpub มอบอำนาจให้นักเขียนและสำนักพิมพ์ด้วยกระบวนการ Lean Publishing Lean Publishing เป็นวิธีการตีพิมพ์หนังสืออีบุ๊คที่กำลังจะเสร็จสิ้นด้วยเครื่องมือที่สะดวกและการพัฒนาแบบวนซ้ำเพื่อรับคำติชมจากผู้อ่าน ปรับเปลี่ยนไปเรื่อยๆจนคุณได้หนังสือตามที่ต้องการ และสร้างการตอบรับเมื่อสำเร็จ

ทวีตหนังสือเล่มนี้!

โปรดช่วย Obie Fernandez ด้วยการกระจายข่าวเกี่ยวกับหนังสือเล่มนี้บน Twitter!

แฮชแท็กที่แนะนำสำหรับหนังสือเล่มนี้คือ #poaduai.

ค้นหาว่าคนอื่นพูดถึงหนังสือเล่มนี้อย่างไรโดยคลิกที่ลิงก์นี้เพื่อค้นหาแฮชแท็กนี้บน Twitter:

#poaduai

ถึงราชินีสุดแกร่งของผม มิวส์ของผม แสงสว่างและความรักของผม วิคตอเรีย

Also By Obie Fernandez

Patterns of Application Development Using AI

The Rails 8 Way

The Rails 7 Way

XML The Rails Way

Serverless

El Libro Principiante de Node

The Lean Enterprise

สารบัญ

ส่วนที่ 1: แนวทางและเทคนิคพื้นฐาน 27

คำนำโดย Gregor Hohpe

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

คำนำ

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

เกี่ยวกับหนังสือ

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

เกี่ยวกับตัวอย่างโค้ด

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

สิ่งที่ผมไม่ครอบคลุม

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

หนังสือเล่มนี้เหมาะสำหรับใคร

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

การสร้างคำศัพท์ร่วมกัน

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

การมีส่วนร่วม

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

กิตติกรรมประกาศ

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

ทำไมถึงมีภาพประกอบแบบนี้?

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

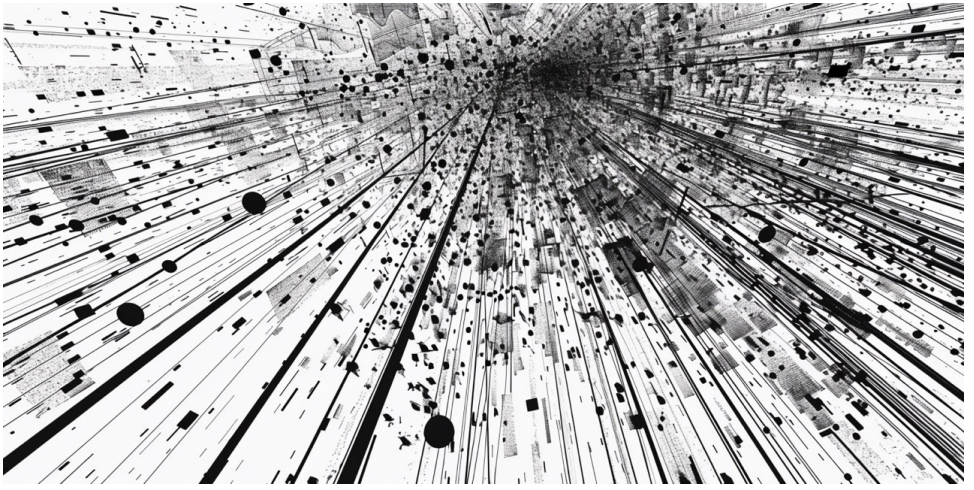
เกี่ยวกับ Lean Publishing

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

เกี่ยวกับผู้เขียน

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

บทนำ



หากคุณกระตือรือร้นที่จะเริ่มผสานแบบจำลองภาษาขนาดใหญ่ (LLMs) เข้ากับโครงการโปรแกรมของคุณ คุณสามารถเริ่มศึกษาแพทเทิร์นและตัวอย่างโค้ดที่นำเสนอในบทต่อ ๆ ไปได้เลย อย่างไรก็ตาม เพื่อให้เข้าใจพลังและศักยภาพของแพทเทิร์นเหล่านี้อย่างถ่องแท้ ควรใช้เวลาทำความเข้าใจบริบทที่กว้างขึ้นและแนวทางที่เชื่อมโยงกันที่พวกมันนำเสนอ

แพทเทิร์นเหล่านี้ไม่ใช่เพียงการรวบรวมเทคนิคที่แยกจากกัน แต่เป็นกรอบการทำงานที่เป็นหนึ่งเดียวสำหรับการผสาน AI เข้ากับแอปพลิเคชันของคุณ ผมใช้ Ruby on Rails แต่แพทเทิร์นเหล่านี้ควรจะใช้ได้กับสภาพแวดล้อมการเขียนโปรแกรมอื่น ๆ เกือบทั้งหมด พวกมันครอบคลุมประเด็นที่หลากหลาย ตั้งแต่การจัดการข้อมูลและการปรับประสิทธิภาพ ไปจนถึงประสบการณ์ผู้ใช้และความปลอดภัย โดยให้ชุดเครื่องมือที่ครอบคลุมสำหรับการเพิ่มประสิทธิภาพการเขียนโปรแกรมแบบดั้งเดิมด้วยความสามารถของ AI

แพทเทิร์นแต่ละประเภทจัดการกับความท้าทายหรือโอกาสเฉพาะที่เกิดขึ้นเมื่อรวมองค์ประกอบ AI เข้ากับแอปพลิเคชันของคุณ การทำความเข้าใจความสัมพันธ์และการผสานพลังระหว่างแพทเทิร์นเหล่านี้จะช่วยให้คุณตัดสินใจได้อย่างชาญฉลาดว่าจะใช้ AI ที่ไหนและอย่างไรจึงจะมีประสิทธิภาพสูงสุด

แพทเทิร์นไม่ใช่วิธีแก้ปัญหาที่ตายตัวและไม่ควรถูกมองว่าเป็นเช่นนั้น พวกมันเป็นองค์ประกอบพื้นฐานที่ปรับเปลี่ยนได้ซึ่งควรถูกปรับแต่งให้เข้ากับความต้องการและข้อจำกัดเฉพาะของแอปพลิเคชันของคุณ การประยุกต์ใช้แพทเทิร์นเหล่านี้อย่างประสบความ

สำเร็จ (เช่นเดียวกับแพทเทิร์นอื่น ๆ ในวงการซอฟต์แวร์) ขึ้นอยู่กับความเข้าใจอย่างลึกซึ้งในโดเมนของปัญหา ความต้องการของผู้ใช้ และสถาปัตยกรรมทางเทคนิคโดยรวมของโครงการของคุณ

ความคิดเห็นเกี่ยวกับสถาปัตยกรรมซอฟต์แวร์

ผมเริ่มเขียนโปรแกรมในช่วงทศวรรษ 1980 และมีส่วนร่วมในวงการแฮกเกอร์ และไม่เคยทิ้งมุมมองแบบแฮกเกอร์ แม้หลังจากที่กลายเป็นนักพัฒนาซอฟต์แวร์มืออาชีพ ตั้งแต่เริ่มต้น ผมมีความสงสัยอย่างมีเหตุผลเสมอว่าสถาปนิกซอฟต์แวร์ในหอคอยงาช้างของพวกเขานั้นสร้างคุณค่าอะไรให้กับวงการจริง ๆ

หนึ่งในเหตุผลที่ผมตื่นเต้นเป็นการส่วนตัวกับการเปลี่ยนแปลงที่เกิดจากคลื่นลูกใหม่ของเทคโนโลยี AI อันทรงพลังนี้คือผลกระทบที่มีต่อสิ่งที่เราพิจารณาว่าเป็นการตัดสินใจด้าน *สถาปัตยกรรมซอฟต์แวร์* มันท้าทายแนวคิดดั้งเดิมเกี่ยวกับวิธีที่ “ถูกต้อง” ในการออกแบบและพัฒนาโครงการซอฟต์แวร์ของเรา นอกจากนี้ยังท้าทายว่าสถาปัตยกรรมยังคงถูกมองว่าเป็น *ส่วนของระบบที่ยากจะเปลี่ยนแปลง* หรือไม่ เนื่องจากการเพิ่มประสิทธิภาพด้วย AI กำลังทำให้การเปลี่ยนแปลงส่วนใดก็ตามของโครงการของคุณเป็นเรื่องง่ายกว่าที่เคย ไม่ว่าจะเป็นเวลาใดก็ตาม

บางทีเราอาจกำลังก้าวเข้าสู่ยุคทองของแนวทาง “โพสต์โมเดิร์น” ในวิศวกรรมซอฟต์แวร์ ในบริบทนี้ โพสต์โมเดิร์นหมายถึงการเปลี่ยนแปลงพื้นฐานจากกระบวนทัศน์แบบดั้งเดิม ที่นักพัฒนาต้องรับผิดชอบการเขียนและดูแลรักษาโค้ดทุกบรรทัด แทนที่จะเป็นเช่นนั้น มันยอมรับแนวคิดของการมอบหมายงาน เช่น การจัดการข้อมูล อัลกอริธึมที่ซับซ้อน และแม้แต่ส่วนของตรรกะแอปพลิเคชันทั้งหมด ให้กับไลบรารีของบุคคลที่สามและ API ภายนอก การเปลี่ยนแปลงแบบโพสต์โมเดิร์นนี้เป็นการเบี่ยงเบนที่สำคัญจากหลักปฏิบัติทั่วไปของการสร้างแอปพลิเคชันตั้งแต่เริ่มต้น และท้าทายให้นักพัฒนาต้องคิดใหม่เกี่ยวกับบทบาทของตนในกระบวนการพัฒนา

ผมเชื่อมาตลอดว่านักเขียนโปรแกรมที่ดีจะเขียนเฉพาะโค้ดที่จำเป็นต้องเขียนเท่านั้น ตามคำสอนของ Larry Wall และผู้นำแฮกเกอร์คนอื่น ๆ เช่นเดียวกับเขา การลดปริมาณโค้ดที่เขียน ช่วยให้เราเคลื่อนที่ได้เร็วขึ้น ลดพื้นที่ที่อาจเกิดข้อบกพร่อง ทำให้การบำรุงรักษาง่ายขึ้น และปรับปรุงความน่าเชื่อถือโดยรวมของแอปพลิเคชัน โค้ดที่น้อยลงช่วยให้เราสามารถมุ่งเน้นไปที่ตรรกะทางธุรกิจหลักและประสบการณ์ผู้ใช้ ในขณะที่มอบหมายงานอื่น ๆ ให้กับบริการอื่น ๆ

ตอนนี้ที่ระบบที่ขับเคลื่อนด้วย AI สามารถจัดการกับงานที่เคยเป็นขอบเขตเฉพาะของโค้ดที่เขียนโดยมนุษย์ เราควรจะสามารถมีผลิตภาพและความคล่องตัวมากขึ้น โดยมุ่งเน้นไปที่การสร้างคุณค่าทางธุรกิจและประสบการณ์ผู้ใช้มากกว่าที่เคย

แน่นอนว่ามีข้อแลกเปลี่ยนในการมอบหมายส่วนใหญ่ของโครงการให้กับระบบ AI เช่น การสูญเสียการควบคุมที่อาจเกิดขึ้น และความจำเป็นในการตรวจสอบและกลไกการตอบกลับที่แข็งแกร่ง นั่นคือเหตุผลที่มันต้องการชุดทักษะและความรู้ใหม่ รวมถึงความเข้าใจพื้นฐานอย่างน้อยเกี่ยวกับวิธีการทำงานของ AI

แบบจำลองภาษาขนาดใหญ่คืออะไร

แบบจำลองภาษาขนาดใหญ่ (LLMs) เป็นประเภทของแบบจำลองปัญญาประดิษฐ์ที่ได้รับความสนใจอย่างมากในช่วงไม่กี่ปีที่ผ่านมา นับตั้งแต่การเปิดตัว GPT-3 โดย OpenAI ในปี 2020 LLMs ถูกออกแบบมาเพื่อประมวลผล ทำความเข้าใจ และสร้างภาษามนุษย์ด้วยความแม่นยำและความคล่องแคล่วที่น่าทึ่ง ในส่วนนี้ เราจะมาดูอย่างคร่าว ๆ ว่า LLMs ทำงานอย่างไรและทำไมจึงเหมาะสมสำหรับการสร้างองค์ประกอบระบบอัจฉริยะ

แก่นหลักของ LLMs คือขั้นตอนวิธีการเรียนรู้เชิงลึก โดยเฉพาะโครงข่ายประสาทเทียม เครือข่ายเหล่านี้ประกอบด้วยโหนดที่เชื่อมต่อกัน หรือนิวรอน ที่ประมวลผลและส่งข้อมูล สถาปัตยกรรมที่เลือกใช้สำหรับ LLMs มักเป็นแบบจำลองทรานส์ฟอร์เมอร์ ซึ่งพิสูจน์แล้วว่ามีประสิทธิภาพสูงในการจัดการข้อมูลแบบลำดับ เช่น ข้อความ

โมเดลทรานส์ฟอร์เมอร์สร้างขึ้นบนพื้นฐานของกลไกการให้ความสนใจและถูกใช้หลักๆ สำหรับงานที่เกี่ยวข้องกับข้อมูลแบบลำดับ เช่น การประมวลผลภาษาธรรมชาติ ทรานส์ฟอร์เมอร์ประมวลผลข้อมูลนำเข้าทั้งหมดพร้อมกันแทนที่จะทำทีละลำดับ ซึ่งช่วยให้สามารถจับความสัมพันธ์ระยะไกลได้อย่างมีประสิทธิภาพมากขึ้น โมเดลมีชั้นของกลไกการให้ความสนใจที่ช่วยให้โมเดลโฟกัสที่ส่วนต่างๆ ของข้อมูลนำเข้าเพื่อทำความเข้าใจบริบทและความสัมพันธ์

กระบวนการฝึกฝนสำหรับ LLMs เกี่ยวข้องกับการให้โมเดลเรียนรู้จากข้อมูลที่เป็นข้อความจำนวนมหาศาล เช่น หนังสือ บทความ เว็บไซต์ และที่เก็บโค้ด ระหว่างการฝึกฝน โมเดลจะเรียนรู้ที่จะจดจำรูปแบบ ความสัมพันธ์ และโครงสร้างภายในข้อความ มันจับคุณสมบัติทางสถิติของภาษา เช่น กฎไวยากรณ์ ความสัมพันธ์ของคำ และความหมายตามบริบท

หนึ่งในเทคนิคสำคัญที่ใช้ในการฝึกฝน LLMs คือการเรียนรู้แบบไม่มีผู้สอน นั่นหมายความว่าโมเดลเรียนรู้จากข้อมูลโดยไม่มีการติดป้ายกำกับหรือการชี้แนะอย่างชัดเจน มันค้นพบรูปแบบและการแทนค่าด้วยตัวเองโดยการวิเคราะห์การเกิดร่วมกันของคำและวลีในข้อมูลฝึกฝน สิ่งนี้ช่วยให้ LLMs พัฒนาความเข้าใจภาษาและความละเอียดอ่อนของภาษาอย่างลึกซึ้ง

อีกแง่มุมสำคัญของ LLMs คือความสามารถในการจัดการกับ *บริบท* เมื่อประมวลผลข้อความ LLMs พิจารณาไม่เพียงแค่คำแต่ละคำ แต่ยังรวมถึงบริบทโดยรอบด้วย พวกมันคำนึงถึงคำก่อนหน้า ประโยค และแม้แต่ย่อหน้าเพื่อเข้าใจความหมายและเจตนาของข้อความ ความเข้าใจเชิงบริบทนี้ช่วยให้ LLMs สามารถสร้างการตอบสนองที่สอดคล้องและเกี่ยวข้องกัน หนึ่งในวิธีหลักที่เราประเมินความสามารถของโมเดล LLM คือการพิจารณาขนาดของบริบทที่พวกมันสามารถใช้ในการสร้างการตอบสนอง

เมื่อผ่านการฝึกฝนแล้ว LLMs สามารถใช้สำหรับงานที่เกี่ยวข้องกับภาษาได้หลากหลาย พวกมันสามารถสร้างข้อความที่เหมือนมนุษย์เขียน ตอบคำถาม สรุปเอกสาร แปลภาษา และแม้แต่เขียนโค้ด ความอเนกประสงค์ของ LLMs ทำให้พวกมันมีคุณค่าสำหรับการสร้างองค์ประกอบระบบอัจฉริยะที่สามารถโต้ตอบกับผู้ใช้ ประมวลผลและวิเคราะห์ข้อมูลข้อความ และสร้างผลลัพธ์ที่มีความหมาย

การ รวม LLMs เข้าไป ใน สถาปัตยกรรม แอปพลิเคชัน คุณ สามารถ สร้าง องค์ ประกอบ AI ที่ เข้าใจ และ ประมวล ผล ข้อมูล

นำ เข้า จาก ผู้ ใช้ สร้าง เนื้อหา แบบ ไดนามิก และ ให้ คำ แนะนำ หรือ การก ระ ทำ ที่ ชาญ ฉลาด แต่ การ ทำงาน กับ LLMs ต้องพิจารณาข้อกำหนดด้านทรัพยากรและการแลกเปลี่ยนด้านประสิทธิภาพอย่างรอบคอบ LLMs ใช้ การ ประมวล ผล ที่ เข้ม ข้น และ อาจ ต้องการ พลัง การ ประมวล ผล และ หน่วย ความ จำ (กล่าว อีก นัย หนึ่ง คือ เงิน) จำนวนมาก ใน การ ทำงาน พวก เรา ส่วน ใหญ่จำเป็นต้องประเมินผลกระทบด้านต้นทุนของการผสาน LLMs เข้ากับแอปพลิเคชันของเราและดำเนินการตามนั้น

ทำความเข้าใจการอนุมาน

การอนุมานหมายถึงกระบวนการที่โมเดลสร้างการทำนายหรือผลลัพธ์บนพื้นฐานของข้อมูลใหม่ที่ไม่เคยเห็นมาก่อน เป็นขั้นตอนที่โมเดลที่ผ่านการฝึกฝนแล้วถูกใช้เพื่อตัดสินใจหรือสร้างข้อความ รูปภาพ หรือเนื้อหาอื่นๆ ในการตอบสนองต่อข้อมูลนำเข้าจากผู้ใช้

ในระหว่างขั้นตอนการฝึกฝน โมเดล AI เรียนรู้จากชุดข้อมูลขนาดใหญ่โดยการปรับพารามิเตอร์เพื่อลดความผิดพลาดในการทำนาย เมื่อฝึกฝนแล้ว โมเดลสามารถนำสิ่งที่ได้เรียนรู้ไปใช้กับข้อมูลใหม่ การอนุมานคือวิธีที่โมเดลใช้รูปแบบและความรู้ที่ได้เรียนรู้มาเพื่อสร้างผลลัพธ์

สำหรับ LLMs การอนุมานเกี่ยวข้องกับการรับคำสั่งหรือข้อความนำเข้าและสร้างการตอบสนองที่สอดคล้องและเกี่ยวข้องกับบริบท ในรูปแบบของสตรีมของ *โทเค็น* (ซึ่งเราจะพูดถึงเร็วๆ นี้) ซึ่งอาจเป็นการตอบคำถาม การเติมประโยคให้สมบูรณ์ การสร้างเรื่องราว หรือการแปลข้อความ ในบรรดางานอื่นๆ อีกมากมาย

ในทางตรงกันข้ามกับวิธีที่คุณและฉันคิด การ “คิด” ของโมเดล AI ผ่านการอนุมานเกิดขึ้นทั้งหมดในการดำเนินการที่ไม่มีสถานะครั้งเดียว นั่นคือ การคิดของมันถูกจำกัดอยู่ที่กระบวนการสร้างผลลัพธ์ มันต้องคิดออกมาดังๆ เหมือนกับว่าฉันถามคำถามคุณและยอมรับการตอบสนองจากคุณเฉพาะในรูปแบบ “กระแสจิตสำนึก” เท่านั้น

โมเดลภาษาขนาดใหญ่มีหลายขนาดและรูปแบบ

ในขณะที่โมเดลภาษาขนาดใหญ่ (LLMs) ที่เป็นที่นิยมเกือบทั้งหมดสร้างขึ้นบนสถาปัตยกรรมทรานส์ฟอร์เมอร์หลักเดียวกันและได้รับการฝึกฝนบนชุดข้อมูลข้อความขนาดใหญ่ พวกมันมีหลายขนาดและได้รับการปรับแต่งสำหรับวัตถุประสงค์ที่แตกต่างกัน ขนาดของ LLM ซึ่งวัดจากจำนวนพารามิเตอร์ในเครือข่ายประสาทเทียม มีผลกระทบอย่างมากต่อความสามารถของมัน โมเดลที่ใหญ่กว่าที่มีพารามิเตอร์มากกว่า เช่น GPT-4 ซึ่งมีข่าวลือว่ามีพารามิเตอร์ 1 ถึง 2 ล้านล้านตัว โดยทั่วไปจะมีความรู้และความสามารถมากกว่าโมเดลที่เล็กกว่า อย่างไรก็ตาม โมเดลที่ใหญ่กว่าต้องการพลังการประมวลผลมากกว่าในการทำงาน ซึ่งแปลเป็นค่าใช้จ่ายที่สูงขึ้นเมื่อคุณใช้งานผ่านการเรียก API

เพื่อทำให้ LLMs ใช้งานได้จริงมากขึ้นและปรับแต่งสำหรับกรณีการใช้งานเฉพาะ โมเดลพื้นฐานมักจะได้รับการปรับแต่งเฉพาะทางบนชุดข้อมูลที่เฉพาะเจาะจงมากขึ้น ตัวอย่างเช่น LLM อาจได้รับการฝึกฝนบนคลังบทสนทนาขนาดใหญ่เพื่อให้เชี่ยวชาญสำหรับ AI ที่สนทนาได้ บางตัวได้รับการฝึกฝนด้วยโค้ดเพื่อให้มีความรู้ด้านการเขียนโปรแกรม มีแม้แต่โมเดลที่ได้รับการฝึกฝนพิเศษสำหรับการโต้ตอบแบบบทบาทสมมติกับผู้ใช้!

โมเดลแบบค้นคืนเทียบกับโมเดลแบบสร้างเนื้อหา

ในโลกของโมเดลภาษาขนาดใหญ่ (LLMs) มีสองแนวทางหลักในการสร้างการตอบสนอง: โมเดลแบบค้นคืนและโมเดลแบบสร้างเนื้อหา แต่ละแนวทางมีจุดแข็งและจุดอ่อนของตัวเอง และการทำความเข้าใจความแตกต่างระหว่างทั้งสองแบบจะช่วยให้คุณเลือกโมเดลที่เหมาะสมกับกรณีการใช้งานเฉพาะของคุณได้

โมเดลแบบค้นคืน

โมเดลแบบค้นคืน หรือที่รู้จักกันในชื่อโมเดลการค้นคืนข้อมูล สร้างการตอบสนองโดยการค้นหาผ่านฐานข้อมูลขนาดใหญ่ของข้อความที่มีอยู่แล้ว และเลือกส่วนที่เกี่ยวข้องมากที่สุดตามคำถามที่ได้รับ โมเดลเหล่านี้ไม่ได้สร้างข้อความใหม่จากศูนย์ แต่นำส่วนต่างๆ จากฐานข้อมูลมาเรียบเรียงเพื่อสร้างการตอบสนองที่มีความต่อเนื่อง

หนึ่งในข้อดีหลักของโมเดลแบบค้นคืนคือความสามารถในการให้ข้อมูลที่ถูกต้องตามข้อเท็จจริงและทันสมัย เนื่องจากพวกมันอาศัยฐานข้อมูลของข้อความที่ได้รับการคัดสรร พวกมันสามารถดึงข้อมูลที่เกี่ยวข้องจากแหล่งที่เชื่อถือได้และนำเสนอต่อผู้ใช้ สิ่งนี้ทำให้พวกมันเหมาะสมสำหรับแอปพลิเคชันที่ต้องการคำตอบที่แม่นยำและเป็นข้อเท็จจริง เช่น ระบบถาม-ตอบ หรือฐานความรู้

อย่างไรก็ตาม โมเดลแบบค้นคืนมีข้อจำกัดบางประการ พวกมันจะดีเพียงเท่าที่ฐานข้อมูลที่ใช้ค้นหา ดังนั้นคุณภาพและความครอบคลุมของฐานข้อมูลจึงส่งผลโดยตรงต่อประสิทธิภาพของโมเดล นอกจากนี้ โมเดลเหล่านี้อาจมีปัญหาในการสร้างการตอบสนองที่มีความต่อเนื่องและฟังดูเป็นธรรมชาติ เนื่องจากถูกจำกัดด้วยข้อความที่มีอยู่ในฐานข้อมูล

เราไม่ได้ครอบคลุมการใช้งานโมเดลแบบค้นคืนล้วนๆ ในหนังสือเล่มนี้

โมเดลแบบสร้างเนื้อหา

ในทางกลับกัน โมเดลแบบสร้างเนื้อหาสร้างข้อความใหม่จากศูนย์โดยอาศัยรูปแบบและความสัมพันธ์ที่ได้เรียนรู้ระหว่างการฝึกฝน โมเดลเหล่านี้ใช้ความเข้าใจด้านภาษาเพื่อสร้างการตอบสนองใหม่ที่ปรับให้เข้ากับคำสั่งที่ได้รับ

จุดแข็งหลักของโมเดลแบบสร้างเนื้อหาคือความสามารถในการผลิตข้อความที่สร้างสรรค์ มีความต่อเนื่อง และเกี่ยวข้องกับบริบท พวกมันสามารถมีส่วนร่วมในการสนทนาแบบเปิดกว้าง สร้างเรื่องราว และแม้แต่เขียนโค้ด สิ่งนี้ทำให้พวกมันเหมาะ

สำหรับแอปพลิเคชันที่ต้องการการโต้ตอบที่เปิดกว้างและมีพลวัตมากขึ้น เช่น แชทบอท การสร้างเนื้อหา และผู้ช่วยการเขียนเชิงสร้างสรรค์

อย่างไรก็ตาม โมเดลแบบสร้างเนื้อหาอาจผลิตข้อมูลที่ไม่สอดคล้องกันหรือไม่ถูกต้องตามข้อเท็จจริงในบางครั้ง เนื่องจากพวกมันอาศัยรูปแบบที่เรียนรู้ระหว่างการฝึกฝนมากกว่าฐานข้อมูลข้อเท็จจริงที่ได้รับการคัดสรร พวกมันอาจมีแนวโน้มที่จะมีอคติและภาพหลอนมากขึ้น โดยสร้างข้อความที่ดูสมเหตุสมผลแต่ไม่จำเป็นต้องเป็นความจริง

ตัวอย่างของ LLMs แบบสร้างเนื้อหารวมถึง GPT series ของ OpenAI (GPT-3, GPT-4) และ Claude ของ Anthropic

โมเดลแบบผสม

LLMs เชิงพาณิชย์หลายตัวรวมทั้งแนวทางแบบค้นคืนและแบบสร้างเนื้อหาเข้าด้วยกันในโมเดลแบบผสม โมเดลเหล่านี้ใช้เทคนิคการค้นคืนเพื่อหาข้อมูลที่เกี่ยวข้องจากฐานข้อมูล จากนั้นใช้เทคนิคการสร้างเนื้อหาเพื่อสังเคราะห์ข้อมูลนั้นให้เป็นการตอบสนองที่มีความต่อเนื่อง

โมเดลแบบผสมมีจุดมุ่งหมายเพื่อรวมความถูกต้องตามข้อเท็จจริงของโมเดลแบบค้นคืนเข้ากับความสามารถในการสร้างภาษาธรรมชาติของโมเดลแบบสร้างเนื้อหา พวกมันสามารถให้ข้อมูลที่น่าเชื่อถือและทันสมัยมากขึ้น ในขณะที่ยังคงรักษาความสามารถในการมีส่วนร่วมในการสนทนาแบบเปิดกว้าง

เมื่อเลือกระหว่างโมเดลแบบค้นคืนและแบบสร้างเนื้อหา คุณควรพิจารณาความต้องการเฉพาะของแอปพลิเคชันของคุณ หากเป้าหมายหลักคือการให้ข้อมูลที่ถูกต้องและเป็นข้อเท็จจริง โมเดลแบบค้นคืนอาจเป็นตัวเลือกที่ดีที่สุด หากแอปพลิเคชันต้องการการโต้ตอบที่เปิดกว้างและสร้างสรรค์มากขึ้น โมเดลแบบสร้างเนื้อหาอาจเหมาะสมกว่า โมเดลแบบผสมให้ความสมดุลระหว่างสองแนวทางและอาจเป็นตัวเลือกที่ดีสำหรับแอปพลิเคชันที่ต้องการทั้งความถูกต้องตามข้อเท็จจริงและการสร้างภาษาธรรมชาติ

ในท้ายที่สุด การเลือกระหว่างโมเดลแบบค้นคืนและแบบสร้างเนื้อหาขึ้นอยู่กับกรณีการใช้งานเฉพาะและการแลกเปลี่ยนระหว่างความแม่นยำ ความคิดสร้างสรรค์ และความยืดหยุ่น การทำความเข้าใจจุดแข็งและข้อจำกัดของแต่ละแนวทางจะช่วยให้คุณตัดสินใจได้อย่างมีข้อมูล

การปรับแต่งด้วยคำสั่ง

การปรับแต่งด้วยคำสั่งเป็นส่วนหนึ่งของการฝึกฝนการสนทนาที่ใช้คำสั่งและการตอบสนองที่เขียนโดยมนุษย์เพื่อทำให้ LLMs เข้าใจและทำตามคำสั่งได้ดีขึ้น นี่คือตัวอย่างง่ายๆ เพื่อแสดงให้เห็นว่าการปรับแต่งด้วยคำสั่งบนคำสั่งและการตอบสนองที่เขียนโดยมนุษย์ทำให้ LLMs เข้าใจและทำตามคำสั่งได้ดีขึ้นอย่างไร:

ตัวอย่างที่ 1: การสนับสนุนลูกค้า

คำสั่ง: “ให้คำตอบแก่ลูกค้าที่ขอคืนเงิน”

คำสั่งที่เขียนโดยมนุษย์: “ฉันต้องการขอคืนเงินสำหรับการซื้อล่าสุดของฉัน สินค้าไม่ตรงตามความคาดหวังของฉัน”

การตอบสนองที่เขียนโดยมนุษย์: “เราเสียใจที่ได้ยินว่าสินค้าไม่ตรงตามความคาดหวังของคุณ เพื่อดำเนินการคืนเงิน กรุณาแจ้งหมายเลขคำสั่งซื้อและเหตุผลในการคืนสินค้า เราจะตรวจสอบคำขอของคุณและติดต่อกลับภายใน 2-3 วันทำการ”

ตัวอย่างที่ 2: การช่วยเหลือด้านการศึกษา

คำสั่ง: “อธิบายแนวคิดเรื่องการสังเคราะห์แสงให้เด็กอายุ 10 ขวบเข้าใจ”

คำถามที่มนุษย์เขียน: “คุณช่วยอธิบายได้ไหมว่าพืชสร้างอาหารได้อย่างไร?”

คำตอบที่มนุษย์เขียน: “ได้ครับ! พืชสร้างอาหารผ่านกระบวนการที่เรียกว่าการสังเคราะห์แสง พวกมันใช้แสงแดด น้ำ และคาร์บอนไดออกไซด์เพื่อสร้างอาหารและออกซิเจน ใบของพืชมีส่วนพิเศษที่เรียกว่าคลอโรฟิลล์ที่ช่วยในการดูดซับแสงแดด”

ตัวอย่างที่ 3: ผู้ช่วยส่วนตัว

คำสั่ง: “นัดประชุมสำหรับวันอังคารหน้าเวลา 15:00 น.”

คำถามที่มนุษย์เขียน: “คุณช่วยจัดตารางประชุมสำหรับวันอังคารเวลา 15:00 น. ได้ไหม?”

คำตอบที่มนุษย์เขียน: “ได้ครับ ผมได้จัดตารางประชุมให้คุณในวันอังคารหน้าเวลา 15:00 น. แล้ว คุณต้องการอะไรเพิ่มเติมไหมครับ?”

ผลลัพธ์คือระบบนิเวศที่หลากหลายของแบบจำลองภาษาขนาดใหญ่ที่มีขนาดและความเชี่ยวชาญแตกต่างกัน แบบจำลองขนาดเล็กที่มีพารามิเตอร์ในช่วง 1-7 พันล้าน ให้ความสามารถด้านภาษาทั่วไปที่ดีในขณะที่มีประสิทธิภาพในการทำงานมากกว่า

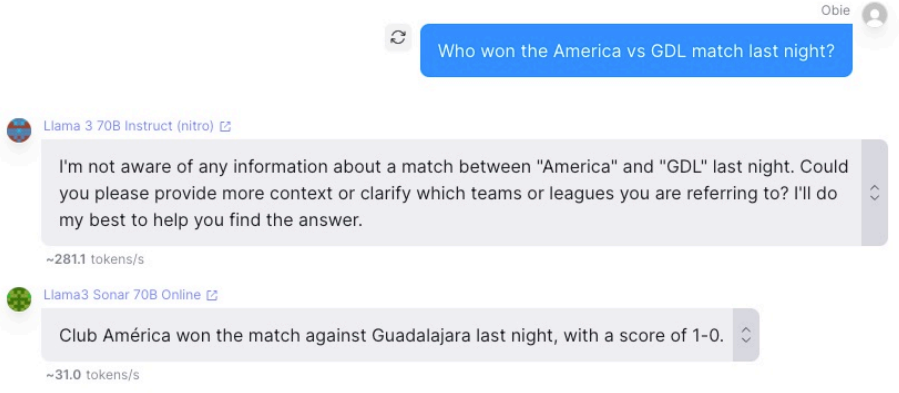
- Mistral 7B
- Llama 3 8B
- Gemma 7B

แบบจำลองขนาดกลางที่มีพารามิเตอร์ประมาณ 30-70 พันล้าน มีความสามารถในการให้เหตุผลและทำตามคำสั่งที่แข็งแกร่งขึ้น

- Llama 3 70B
- Qwen2 70B
- Mixtral 8x22B

เมื่อเลือกแบบจำลองภาษาขนาดใหญ่เพื่อนำไปใช้ในแอปพลิเคชัน คุณต้องสร้างความสมดุลระหว่างความสามารถของแบบจำลองกับปัจจัยในทางปฏิบัติ เช่น ต้นทุน ความล่าช้า ความยาวบริบท และการกรองเนื้อหา แบบจำลองขนาดเล็กที่ผ่านการฝึกฝนด้วยคำสั่งมักเป็นตัวเลือกที่ดีที่สุดสำหรับงานด้านภาษาที่ง่ายกว่า ในขณะที่แบบจำลองขนาดใหญ่ที่สุดอาจจำเป็นสำหรับการให้เหตุผลหรือการวิเคราะห์ที่ซับซ้อน ข้อมูลฝึกฝนของแบบจำลองก็เป็นปัจจัยสำคัญ เนื่องจากเป็นตัวกำหนดวันที่ตัดข้อมูลความรู้ของแบบจำลอง

แบบจำลองบางตัว เช่น บางตัวจาก Perplexity เชื่อมต่อกับแหล่งข้อมูลแบบเรียลไทม์ ทำให้พวกมันไม่มีวันที่ตัดข้อมูลที่แท้จริง เมื่อคุณถามคำถาม พวกมันสามารถตัดสินใจค้นหาเว็บและดึงข้อมูลจากหน้าเว็บต่างๆ เพื่อสร้างคำตอบได้อย่างอิสระ

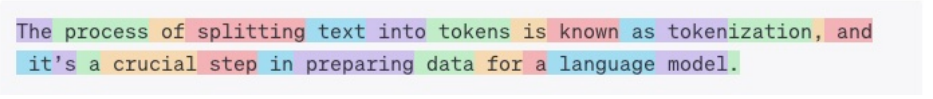


แผนภูมิ 1. Llama3 ทั้งแบบมีและไม่มีการเข้าถึงออนไลน์

ในที่สุดแล้ว ไม่มีแบบจำลองภาษาขนาดใหญ่ที่เหมาะกับทุกงาน การเข้าใจความแตกต่างในด้านขนาด สถาปัตยกรรม และการฝึกฝนของแบบจำลองเป็นกุญแจสำคัญในการเลือกแบบจำลองที่เหมาะสมสำหรับกรณีการใช้งานที่กำหนด การทดลองใช้แบบจำลองต่างๆ เป็นวิธีเดียวในทางปฏิบัติที่จะเผยให้เห็นว่าแบบจำลองใดให้ประสิทธิภาพที่ดีที่สุดสำหรับงานที่ต้องการ

การแบ่งโทเค็น: การแบ่งข้อความเป็นส่วนย่อย

ก่อน ที่ แบบ จำลอง ภาษา ขนาด ใหญ่ จะ สามารถ ประมวล ผล ข้อความ ได้ ข้อความ นั้น จำเป็น ต้อง ถูก แบ่ง ออก เป็น หน่วย ย่อย ที่ เรียก ว่า *โทเค็น* โท เค็ นอาจ เป็น คำ แต่ละ คำ ส่วน ของ คำ หรือ แม้แต่ ตัว อักษร เดี่ยว กระบวนการแบ่งข้อความออกเป็นโทเค็นเรียกว่าการแบ่งโทเค็น และ เป็น ขั้น ตอน สำคัญ ใน การเต รี ยม ข้อมูล สำหรับ แบบ จำลองภาษา



แผนภูมิ 2. ประโยคนี้ประกอบด้วย 27 โทเค็น

แบบ จำลอง ภาษา ขนาด ใหญ่ แต่ละ ตัว ใช้ กลยุทธ์ การ แบ่ง โท เค็ น ที่ แตก ต่าง กัน ซึ่ง สามารถ ส่ง ผล กระ ทบ อย่าง มี นัย สำคัญ ต่อ ประสิทธิภาพและความสามารถของแบบจำลอง ตัวแบ่งโทเค็นที่พบบ่อยที่ใช้โดยแบบจำลองภาษาขนาดใหญ่ได้แก่:

- **GPT (การเข้ารหัสแบบจับคู่ไบต์):** ตัวแบ่งโทเค็น GPT ใช้เทคนิคที่เรียกว่าการเข้ารหัสแบบจับคู่ไบต์ (BPE) เพื่อแบ่งข้อความเป็นหน่วยย่อยของคำ BPE รวมคู่ไบต์ที่พบบ่อยที่สุดในคลังข้อความซ้ำๆ เพื่อสร้างคลังคำของโทเค็นย่อย วิธีนี้ช่วยให้ตัวแบ่งโทเค็นสามารถจัดการกับคำที่พบไม่บ่อยและคำใหม่โดยแบ่งออกเป็นชิ้นส่วนย่อยของคำที่พบบ่อยกว่า ตัวแบ่งโทเค็น GPT ถูกใช้โดยแบบจำลองเช่น GPT-3 และ GPT-4

- **Llama (SentencePiece):** ตัวแบ่งโทเค็นของ Llama ใช้ไลบรารี SentencePiece ซึ่งเป็นตัวแบ่งโทเค็นและรวมโทเค็นแบบไม่ต้องมีการกำกับดูแล SentencePiece จะจัดการข้อความนำเข้าเป็นลำดับของอักขระ Unicode และเรียนรู้คลังคำศัพท์ย่อยจากคลังข้อมูลฝึกสอน มันสามารถรองรับภาษาใดก็ตามที่สามารถเข้ารหัสใน Unicode ทำให้เหมาะสมสำหรับโมเดลหลายภาษา ตัวแบ่งโทเค็น Llama ถูกใช้โดยโมเดลต่างๆ เช่น Llama ของ Meta และ Alpaca

- **SentencePiece (Unigram):** ตัวแบ่งโทเค็น SentencePiece ยังสามารถใช้อัลกอริทึมอื่นที่เรียกว่า Unigram ซึ่งอิงจากเทคนิคการปรับค่าหน่วยคำย่อย Unigram tokenization กำหนดคลังคำศัพท์ย่อยที่เหมาะสมที่สุดโดยอิงจากโมเดลภาษาแบบ unigram ซึ่งกำหนดความน่าจะเป็นให้กับหน่วยคำย่อยแต่ละหน่วย วิธีนี้สามารถสร้างหน่วยคำย่อยที่มีความ

หมายทางความหมายมากกว่าเมื่อเทียบกับ BPE SentencePiece กับ Unigram ถูกใช้โดยโมเดลต่างๆ เช่น T5 และ BERT ของ Google

- **Google Gemini (การแบ่งโทเค็นแบบหลายรูปแบบ):** Google Gemini ใช้รูปแบบการแบ่งโทเค็นที่ออกแบบมาเพื่อจัดการกับข้อมูลหลากหลายประเภท รวมถึงข้อความ รูปภาพ เสียง วิดีโอ และโค้ด ความสามารถแบบหลายรูปแบบนี้ทำให้ Gemini สามารถประมวลผลและผสานรูปแบบข้อมูลที่แตกต่างกันได้ ที่น่าสังเกตคือ Google Gemini 1.5 Pro มีหน้าต่างบริบทที่สามารถจัดการโทเค็นได้หลายล้านตัว ซึ่งใหญ่กว่าโมเดลก่อนหน้านี้มาก หน้าต่างบริบทขนาดใหญ่นี้ช่วยให้โมเดลสามารถประมวลผลบริบทที่กว้างขึ้น ซึ่งอาจนำไปสู่การตอบสนองที่แม่นยำมากขึ้น อย่างไรก็ตาม สิ่งสำคัญที่ต้องทราบคือรูปแบบการแบ่งโทเค็นของ Gemini นั้นใกล้เคียงกับการใช้หนึ่งโทเค็นต่อหนึ่งตัวอักษรมากกว่าโมเดลอื่นๆ นั่นหมายความว่าต้นทุนจริงในการใช้โมเดล Gemini อาจสูงกว่าที่คาดไว้มากหากคุณคุ้นเคยกับการใช้โมเดลอย่าง GPT เนื่องจากการกำหนดราคาของ Google นั้นอิงจากจำนวนตัวอักษรแทนที่จะเป็นโทเค็น

การเลือกตัวแบ่งโทเค็นส่งผลต่อหลายแง่มุมของ LLM รวมถึง:

- **ขนาดคลังคำศัพท์:** ตัวแบ่งโทเค็นกำหนดขนาดของคลังคำศัพท์ของโมเดล ซึ่งเป็นชุดของโทเค็นที่ไม่ซ้ำกันที่มันรู้จัก คลังคำศัพท์ที่ใหญ่กว่าและละเอียดกว่าสามารถช่วยให้โมเดลจัดการกับคำและวลีที่หลากหลายมากขึ้น และอาจกลายเป็นแบบหลายรูปแบบ (สามารถเข้าใจและสร้างได้มากกว่าแค่ข้อความ) แต่ก็เพิ่มความต้องการหน่วยความจำและความซับซ้อนในการคำนวณของโมเดลด้วย
- **การจัดการกับคำที่พบน้อยและคำที่ไม่รู้จัก:** ตัวแบ่งโทเค็นที่ใช้หน่วยคำย่อย เช่น BPE และ SentencePiece สามารถแยกคำที่พบน้อยและคำที่ไม่รู้จักออกเป็นชิ้นส่วนย่อยที่พบบ่อยกว่า สิ่งนี้ช่วยให้โมเดลสามารถคาดเดาความหมายของคำที่ไม่เคยเห็นมาก่อนได้ โดยอิงจากหน่วยคำย่อยที่มีอยู่ในคำนั้น
- **การรองรับหลายภาษา:** ตัวแบ่งโทเค็นอย่าง SentencePiece ที่สามารถจัดการกับภาษาที่เข้ารหัสด้วย Unicode ได้เหมาะสำหรับโมเดลหลายภาษาที่ต้องประมวลผลข้อความในหลายภาษา

เมื่อเลือก LLM สำหรับการใช้งานเฉพาะ สิ่งสำคัญคือต้องพิจารณาตัวแบ่งโทเค็นที่มันใช้และความเหมาะสมกับความต้องการในการประมวลผลภาษาเฉพาะของงานนั้นๆ ตัวแบ่งโทเค็นสามารถส่งผลอย่างมีนัยสำคัญต่อความสามารถของโมเดลในการจัดการกับคำศัพท์เฉพาะทาง คำที่พบน้อย และข้อความหลายภาษา

ขนาดบริบท: โมเดลภาษาสามารถใช้ข้อมูลมากเท่าไหร่ระหว่างการอนุมาน?

เมื่อพูดถึงโมเดลภาษา ขนาดบริบทหมายถึงปริมาณข้อความที่โมเดลสามารถพิจารณาเมื่อประมวลผลหรือสร้างการตอบสนอง มันเป็นการวัดว่าโมเดลสามารถ “จำ” และใช้ข้อมูลเพื่อสร้างผลลัพธ์ได้มากเท่าไหร่ (วัดเป็นโทเค็น) ขนาดบริบทของโมเดลภาษาสามารถส่งผลอย่างมีนัยสำคัญต่อความสามารถและประเภทของงานที่มันสามารถทำได้อย่างมีประสิทธิภาพ

ขนาดบริบทคืออะไร?

ในแง่เทคนิค ขนาดบริบทถูกกำหนดโดยจำนวนโทเค็น (คำหรือส่วนของคำ) ที่โมเดลภาษาสามารถประมวลผลได้ในลำดับข้อมูลนำเข้าเดียว สิ่งนี้มักถูกเรียกว่า “ช่วงความสนใจ” หรือ “หน้าต่างบริบท” ของโมเดล ยิ่งขนาดบริบทใหญ่ขึ้น โมเดลก็สามารถพิจารณาข้อความได้มากขึ้นในคราวเดียวเมื่อสร้างการตอบสนองหรือทำงานต่างๆ

โมเดลภาษาต่างๆ มีขนาดบริบทที่แตกต่างกัน ตั้งแต่ไม่กี่ร้อยโทเค็นไปจนถึงหลายล้านโทเค็น สำหรับการอ้างอิง ย่อหน้าทั่วไปอาจมีประมาณ 100-150 โทเค็น ในขณะที่หนังสือทั้งเล่มอาจมีหลายหมื่นหรือหลายแสนโทเค็น

> มีแม้กระทั่งงานวิจัยเกี่ยวกับวิธีการที่มีประสิทธิภาพในการปรับขนาดโมเดลภาษาขนาดใหญ่ (LLMs) แบบ Transformer เพื่อรองรับข้อมูลนำเข้าที่ยาวไม่จำกัด โดยใช้หน่วยความจำและการคำนวณแบบจำกัด

ทำไมขนาดบริบทจึงมีความสำคัญ?

ขนาดบริบทของโมเดลภาษามีผลกระทบอย่างมากต่อความสามารถในการเข้าใจและสร้างข้อความที่มีความสอดคล้องและเกี่ยวข้องกับบริบท นี่คือเหตุผลสำคัญที่ขนาดบริบทมีความสำคัญ:

1. **การเข้าใจเนื้อหาแบบยาว:** โมเดลที่มีขนาดบริบทใหญ่กว่าสามารถเข้าใจและวิเคราะห์ข้อความที่ยาวขึ้น เช่น บทความรายงาน หรือแม้แต่หนังสือทั้งเล่ม สิ่งนี้สำคัญมากสำหรับงานต่างๆ เช่น การสรุปเอกสาร การตอบคำถาม และการวิเคราะห์เนื้อหา

2. **การรักษาความสอดคล้อง:** หน้าต่างบริบทที่ใหญ่ขึ้นช่วยให้โมเดลสามารถรักษาความสอดคล้องและความต่อเนื่องในผลลัพธ์ที่ยาวขึ้น สิ่งนี้สำคัญสำหรับงานต่างๆ เช่น การสร้างเรื่องราว ระบบสนทนา และการสร้างเนื้อหา ซึ่งการรักษาเรื่องราวหรือหัวข้อให้สอดคล้องกันเป็นสิ่งจำเป็น และยังมีความสำคัญอย่างยิ่งเมื่อใช้ LLM ในการสร้างหรือแปลงข้อมูลที่มีโครงสร้าง
3. **การจับการพึ่งพาระยะไกล:** งานด้านภาษาบางอย่างต้องการความเข้าใจความสัมพันธ์ระหว่างคำหรือวลีที่อยู่ห่างกันในข้อความ โมเดลที่มีขนาดบริบทใหญ่กว่าสามารถจับการพึ่งพาระยะไกลเหล่านี้ได้ดีกว่า ซึ่งสำคัญสำหรับงานต่างๆ เช่น การวิเคราะห์ความรู้สึก การแปล และความเข้าใจภาษา
4. **การจัดการคำสั่งที่ซับซ้อน:** ในการใช้งานที่โมเดลภาษาถูกใช้เพื่อทำตามคำสั่งที่ซับซ้อนหลายขั้นตอน ขนาดบริบทที่ใหญ่ขึ้นช่วยให้โมเดลสามารถพิจารณาชุดคำสั่งทั้งหมดเมื่อสร้างการตอบสนอง แทนที่จะพิจารณาเพียงคำล่าสุดไม่กี่คำ

ตัวอย่างของโมเดลภาษาที่มีขนาดบริบทแตกต่างกัน

นี่คือตัวอย่างของโมเดลภาษาที่มีขนาดบริบทแตกต่างกัน:

- OpenAI GPT-3.5 Turbo: 4,095 โทเค็น
- Mistral 7B Instruct: 32,768 โทเค็น
- Anthropic Claude v1: 100,000 โทเค็น
- OpenAI GPT-4 Turbo: 128,000 โทเค็น
- Anthropic Claude v2: 200,000 โทเค็น
- Google Gemini Pro 1.5: 2.8M โทเค็น

ดังที่คุณเห็น มีช่วงขนาดบริบทที่หลากหลายในโมเดลเหล่านี้ ตั้งแต่ประมาณ 4,000 โทเค็นสำหรับโมเดล OpenAI GPT-3.5 Turbo ไปจนถึง 200,000 โทเค็นสำหรับโมเดล Anthropic Claude v2 โมเดลบางตัว เช่น Google PaLM 2 และ OpenAI GPT-4 มีรุ่นต่างๆ ที่มีขนาดบริบทใหญ่ขึ้น (เช่น รุ่น “32k”) ซึ่งสามารถจัดการกับลำดับข้อมูลนำเข้าที่ยาวขึ้นได้ และในขณะนี้ (เมษายน 2024) Google Gemini Pro กำลังโอ้อวดด้วยจำนวนโทเค็นเกือบ 3 ล้าน!

ควรสังเกตว่าขนาดบริบทอาจแตกต่างกันไปขึ้นอยู่กับการนำไปใช้งานและเวอร์ชันเฉพาะของโมเดลนั้นๆ ตัวอย่างเช่น โมเดล OpenAI GPT-4 รุ่นดั้งเดิมมีขนาดบริบท 8,191 โทเค็น ในขณะที่รุ่น GPT-4 ที่ออกมาภายหลัง เช่น Turbo และ 4o มีขนาดบริบทที่ใหญ่กว่ามากที่ 128,000 โทเค็น

> Sam Altman ได้เปรียบเทียบข้อจำกัดด้านบริบทในปัจจุบันกับหน่วยความจำการทำงานขนาดกิโลไบต์ที่โปรแกรมเมอร์คอมพิวเตอร์ส่วนบุคคลต้องจัดการในยุค 80 และกล่าวว่าในอนาคตอันใกล้เราจะสามารถใส่ “ข้อมูลส่วนบุคคลทั้งหมดของคุณ” ลงในบริบทของโมเดลภาษาขนาดใหญ่ได้

การเลือกขนาดบริบทที่เหมาะสม

เมื่อเลือกโมเดลภาษาสำหรับการใช้งานเฉพาะ สิ่งสำคัญคือต้องพิจารณาความต้องการด้านขนาดบริบทของงานนั้นๆ สำหรับงานที่เกี่ยวข้องกับข้อความสั้นๆ แยกกัน เช่น การวิเคราะห์ความรู้สึกหรือการตอบคำถามง่ายๆ ขนาดบริบทที่เล็กกว่าอาจเพียงพอ

อย่างไรก็ตาม สำหรับงานที่ต้องการความเข้าใจและการสร้างข้อความที่ยาวและซับซ้อนมากขึ้น จำเป็นต้องใช้ขนาดบริบทที่ใหญ่กว่า

ควรสังเกตว่าขนาดบริบทที่ใหญ่ขึ้นมักมาพร้อมกับต้นทุนการประมวลผลที่เพิ่มขึ้นและเวลาประมวลผลที่ช้าลง เนื่องจากโมเดลต้องพิจารณาข้อมูลมากขึ้นเมื่อสร้างการตอบสนอง ดังนั้น คุณต้องหาจุดสมดุลระหว่างขนาดบริบทและประสิทธิภาพเมื่อเลือกโมเดลภาษาสำหรับแอปพลิเคชันของคุณ

> ทำไมไม่เลือกโมเดลที่มีขนาดบริบทใหญ่ที่สุดและใส่ข้อมูลให้มากที่สุดเท่าที่จะทำได้? นอกเหนือจากปัจจัยด้านประสิทธิภาพแล้ว อีกข้อพิจารณาหลักคือต้นทุน ในเดือนมีนาคม 2024 การใช้ Google Gemini Pro 1.5 เพียงหนึ่งรอบของการโต้ตอบด้วยบริบทเต็มรูปแบบจะมีค่าใช้จ่ายเกือบ 8 ดอลลาร์สหรัฐ หากคุณมีกรณีการใช้งานที่คุ้มค่ากับค่าใช้จ่ายนี้ ก็เชิญตามสบาย! แต่สำหรับการใช้งานส่วนใหญ่ มันแพงเกินไปหลายเท่า

การค้นหาเข็มในมัดฟาง

แนวคิดเรื่องการค้นหาเข็มในมัดฟางเป็นอุปมาที่ใช้กันมานานเพื่อสื่อถึงความท้าทายในการค้นคืนข้อมูลจากชุดข้อมูลขนาดใหญ่ ในบริบทของโมเดลภาษาขนาดใหญ่ (LLMs) เราปรับอุปมานี้เล็กน้อย ลองจินตนาการว่าเราไม่ได้มองหาเพียงข้อเท็จจริงเดียวที่ซ่อนอยู่ในเนื้อหามหาศาล (เช่น รวมบทความทั้งหมดของ Paul Graham) แต่เป็นข้อเท็จจริงหลายอย่างที่กระจายอยู่ทั่วไป สถานการณ์นี้เหมือนกับการค้นหาเข็มหลายเล่มในทุ่งกว้าง ไม่ใช่แค่ในมัดฟางเดียว และนี่คือประเด็นสำคัญ: เราไม่เพียงแต่ต้องหาเข็มเหล่านี้ แต่ยังต้องร้อยเรียงมันให้เป็นเรื่องราวที่เชื่อมโยงกัน

เมื่อต้องดึงข้อมูลและวิเคราะห์ข้อเท็จจริงหลายอย่างที่ฝังอยู่ในบริบทยาวๆ LLMs ต้องเผชิญกับความท้าทายสองประการ ประการแรกคือปัญหาพื้นฐานเรื่องความแม่นยำในการดึงข้อมูล—ซึ่งแน่นอนว่าจะลดลงเมื่อจำนวนข้อเท็จจริงเพิ่มขึ้น สิ่งนี้เป็นเรื่องที่คาดการณ์ได้ เพราะการจดจำรายละเอียดหลายอย่างในเนื้อหาที่ยาวเป็นภาระแม้แต่กับโมเดลที่ซับซ้อนที่สุด

ประการที่สอง และอาจสำคัญกว่า คือความท้าทายในการใช้เหตุผลกับข้อเท็จจริงเหล่านี้ การเลือกข้อเท็จจริงออกมาเป็นเรื่องหนึ่ง แต่การสังเคราะห์ให้เป็นเรื่องราวหรือคำตอบที่สอดคล้องกันเป็นอีกเรื่องหนึ่ง นี่คือการทดสอบที่แท้จริง ประสิทธิภาพของ LLMs ในงานที่ต้องใช้เหตุผลมักจะลดลงมากกว่างานที่เป็นแค่การดึงข้อมูลธรรมดา การลดลงนี้ไม่ได้เกี่ยวกับปริมาณอย่างเดียว แต่เกี่ยวกับความซับซ้อนของการผสมผสานบริบท ความเกี่ยวข้อง และการอนุมาน

ทำไมจึงเป็นเช่นนี้? ลองพิจารณาพลวัตของความจำและความใส่ใจในการรู้คิดของมนุษย์ ซึ่งสะท้อนให้เห็นในระดับหนึ่งใน LLMs เมื่อประมวลผลข้อมูลจำนวนมาก LLMs เช่นเดียวกับมนุษย์ อาจลืมรายละเอียดก่อนหน้าขณะที่รับข้อมูลใหม่เข้ามา สิ่งนี้เห็น

ได้ชัดเจนโดยเฉพาะในโมเดลที่ไม่ได้ถูกออกแบบมาให้จัดลำดับความสำคัญหรือย้อนกลับไปดูส่วนก่อนหน้าของข้อความโดยอัตโนมัติ

นอกจากนี้ ความสามารถของ LLM ในการถักทอข้อเท็จจริงที่ดึงมาให้เป็นการตอบสนองที่สอดคล้องกันนั้นคล้ายกับการสร้างเรื่องราว ซึ่งต้องการไม่เพียงแค่การดึงข้อมูล แต่ต้องมีความเข้าใจอย่างลึกซึ้งและการจัดวางในบริบทที่เหมาะสม ซึ่งยังคงเป็นความท้าทายที่ยากสำหรับ AI ในปัจจุบัน

แล้วสิ่งนี้หมายความว่าอย่างไรสำหรับเราในฐานะนักพัฒนาและผู้ผสานเทคโนโลยีเหล่านี้? เราจำเป็นต้องตระหนักถึงข้อจำกัดเหล่านี้อย่างชัดเจนเมื่อออกแบบระบบที่พึ่งพา LLMs ในการจัดการงานที่ซับซ้อนและมีเนื้อหายาว การเข้าใจว่าประสิทธิภาพอาจลดลงภายใต้เงื่อนไขบางอย่างช่วยให้เราตั้งความคาดหวังที่สมเหตุสมผลและพัฒนากลไกสำรองหรือกลยุทธ์เสริมที่ดีขึ้น

รูปแบบข้อมูล: นอกเหนือจากข้อความ

ในขณะที่โมเดลภาษาส่วนใหญ่ในปัจจุบันมุ่งเน้นที่การประมวลผลและสร้างข้อความ กำลังมีแนวโน้มที่เพิ่มขึ้นในการพัฒนาโมเดลมัลติโมดัลที่สามารถรับข้อมูลและสร้างผลลัพธ์ได้หลายรูปแบบโดยตรง เช่น ภาพ เสียง และวิดีโอ โมเดลมัลติโมดัลเหล่านี้เปิดโอกาสใหม่ๆ สำหรับแอปพลิเคชันที่ขับเคลื่อนด้วย AI ที่สามารถเข้าใจและสร้างเนื้อหาข้ามรูปแบบข้อมูลต่างๆ

รูปแบบข้อมูลคืออะไร?

ในบริบทของโมเดลภาษา รูปแบบข้อมูลหมายถึงประเภทข้อมูลต่างๆ ที่โมเดลสามารถประมวลผลและสร้างได้ รูปแบบที่พบบ่อยที่สุดคือข้อความ ซึ่งรวมถึงภาษาเขียนในรูปแบบต่างๆ เช่น หนังสือ บทความ เว็บไซต์ และโพสต์ในสื่อสังคม อย่างไรก็ตาม มีรูปแบบข้อมูลอื่นๆ ที่กำลังถูกรวมเข้ากับโมเดลภาษามากขึ้นเรื่อยๆ:

- **ภาพ:** ข้อมูลภาพ เช่น ภาพถ่าย ภาพประกอบ และแผนภาพ
- **เสียง:** ข้อมูลเสียง เช่น คำพูด ดนตรี และเสียงแวดล้อม
- **วิดีโอ:** ข้อมูลภาพเคลื่อนไหว มักมาพร้อมกับเสียง เช่น คลิปวิดีโอและภาพยนตร์

แต่ละรูปแบบข้อมูลมีความท้าทายและโอกาสที่เป็นเอกลักษณ์สำหรับโมเดลภาษา ตัวอย่างเช่น ภาพต้องการให้โมเดลเข้าใจแนวคิดและความสัมพันธ์ทางภาพ ในขณะที่เสียงต้องการให้โมเดลประมวลผลและสร้างคำพูดและเสียงอื่นๆ

โมเดลภาษามัลติโมดัล

โมเดลภาษามัลติโมดัล ถูกออกแบบมาให้จัดการกับหลายรูปแบบข้อมูลในโมเดลเดียว โมเดลเหล่านี้มักมีส่วนประกอบหรือเลเยอร์พิเศษที่สามารถทั้งเข้าใจข้อมูลนำเข้าและสร้างข้อมูลส่งออกในรูปแบบต่างๆ ตัวอย่างที่โดดเด่นของโมเดลภาษามัลติโมดัลได้แก่:

- **OpenAI's GPT-4o:** GPT-4o เป็นโมเดลภาษาขนาดใหญ่ที่เข้าใจและประมวลผลเสียงพูดนอกเหนือจากข้อความได้โดยตรง ความสามารถนี้ทำให้ GPT-4o สามารถทำงานต่างๆ เช่น การถอดความภาษาพูด การสร้างข้อความจากข้อมูลเสียง และการตอบสนองต่อคำถามที่พูด
- **OpenAI's GPT-4 with visual input:** GPT-4 เป็นโมเดลภาษาขนาดใหญ่ที่สามารถประมวลผลทั้งข้อความและภาพ เมื่อได้รับภาพเป็นข้อมูลนำเข้า GPT-4 สามารถวิเคราะห์เนื้อหาของภาพและสร้างข้อความที่อธิบายหรือตอบสนองต่อข้อมูลภาพนั้น
- **Google's Gemini:** Gemini เป็นโมเดลมัลติโมดัลที่สามารถจัดการกับข้อความ ภาพ และวิดีโอ มันใช้สถาปัตยกรรมแบบรวมที่ช่วยให้เกิดความเข้าใจและการสร้างข้อมูลข้ามรูปแบบ ทำให้สามารถทำงานต่างๆ เช่น การเขียนคำบรรยายภาพ การสรุปวิดีโอ และการตอบคำถามจากภาพ

- **DALL-E และ Stable Diffusion:** แม้จะไม่ใช่โมเดลภาษาในความหมายดั้งเดิม แต่โมเดลเหล่านี้แสดงให้เห็นถึงพลังของ AI แบบหลายรูปแบบในการสร้างภาพจากคำอธิบายที่เป็นข้อความ พวกมันแสดงให้เห็นถึงศักยภาพของโมเดลที่สามารถแปลงระหว่างรูปแบบข้อมูลที่แตกต่างกัน

ประโยชน์และการประยุกต์ใช้โมเดลหลายรูปแบบ

โมเดลภาษาหลายรูปแบบมีประโยชน์หลายประการและเปิดโอกาสให้ใช้งานได้หลากหลาย รวมถึง:

- **ความเข้าใจที่ดีขึ้น:** ด้วยการประมวลผลข้อมูลจากหลายรูปแบบ โมเดลเหล่านี้สามารถเข้าใจโลกได้อย่างครอบคลุมมากขึ้น คล้ายกับวิธีที่มนุษย์เรียนรู้จากการรับรู้ทางประสาทสัมผัสต่างๆ
- **การสร้างข้ามรูปแบบ:** โมเดลหลายรูปแบบสามารถสร้างเนื้อหาในรูปแบบหนึ่งจากข้อมูลนำเข้าอีกรูปแบบหนึ่ง เช่น การสร้างภาพจากคำอธิบายที่เป็นข้อความ หรือการสร้างวิดีโอสรุปจากบทความที่เขียน
- **การเข้าถึง:** โมเดลหลายรูปแบบสามารถทำให้ข้อมูลเข้าถึงได้ง่ายขึ้นด้วยการแปลงระหว่างรูปแบบต่างๆ เช่น การสร้างคำอธิบายเป็นข้อความสำหรับภาพให้ผู้พิการทางสายตา หรือการสร้างเวอร์ชันเสียงจากเนื้อหาที่เป็นลายลักษณ์อักษร
- **การประยุกต์ใช้เชิงสร้างสรรค์:** โมเดลหลายรูปแบบสามารถใช้สำหรับงานสร้างสรรค์ เช่น การสร้างศิลปะ ดนตรี หรือวิดีโอจากคำสั่งที่เป็นข้อความ เปิดโอกาสใหม่ๆ สำหรับศิลปินและผู้สร้างเนื้อหา

เมื่อโมเดลภาษาหลายรูปแบบมีการพัฒนาต่อไป พวกมันจะมีบทบาทสำคัญมากขึ้นในการพัฒนาแอปพลิเคชันที่ขับเคลื่อนด้วย AI ซึ่งสามารถเข้าใจและสร้างเนื้อหาในหลายรูปแบบ สิ่งนี้จะช่วยให้เกิดการปฏิสัมพันธ์ที่เป็นธรรมชาติและเข้าใจง่ายระหว่างมนุษย์กับระบบ AI รวมทั้งปลดล็อกความเป็นไปได้ใหม่ๆ สำหรับการแสดงออกเชิงสร้างสรรค์และการเผยแพร่ความรู้

ระบบนิเวศของผู้ให้บริการ

เมื่อพูดถึงการนำโมเดลภาษาขนาดใหญ่ (LLMs) มาใช้ในแอปพลิเคชัน คุณมีตัวเลือกมากมายให้เลือก ผู้ให้บริการ LLM รายใหญ่แต่ละราย เช่น OpenAI, Anthropic, Google และ Cohere มีระบบนิเวศของโมเดล API และเครื่องมือของตนเอง การเลือกผู้ให้บริการที่เหมาะสมต้องพิจารณาปัจจัยต่างๆ รวมถึงราคา ประสิทธิภาพ การกรองเนื้อหา ความเป็นส่วนตัวของข้อมูล และตัวเลือกในการปรับแต่ง

OpenAI

OpenAI เป็นหนึ่งในผู้ให้บริการ LLM ที่เป็นที่รู้จักมากที่สุด โดยซีรีส์ GPT (GPT-3, GPT-4) ของพวกเขาถูกใช้อย่างแพร่หลายในแอปพลิเคชันต่างๆ OpenAI นำเสนอ API ที่ใช้งานง่ายซึ่งช่วยให้คุณสามารถผสานโมเดลของพวกเขาเข้ากับแอปพลิเคชันได้อย่างง่ายดาย พวกเขามีโมเดลหลากหลายที่มีความสามารถและราคาแตกต่างกัน ตั้งแต่โมเดล Ada ระดับเริ่มต้นไปจนถึงโมเดล Davinci ที่ทรงพลัง

ระบบนิเวศของ OpenAI ยังรวมถึงเครื่องมือต่างๆ เช่น OpenAI Playground ที่ช่วยให้คุณทดลองใช้คำสั่งและปรับแต่งโมเดลสำหรับกรณีการใช้งานเฉพาะ พวกเขามีตัวเลือกการกรองเนื้อหาเพื่อช่วยป้องกันการสร้างเนื้อหาที่ไม่เหมาะสมหรือเป็นอันตราย

เมื่อใช้โมเดลของ OpenAI โดยตรง ผมใช้ไลบรารี ruby-openai ของ Alex Rudall

Anthropic

Anthropic เป็นผู้เล่นรายใหญ่อีกรายหนึ่งในวงการ LLM โดยโมเดล Claude ของพวกเขากำลังได้รับความนิยมจากประสิทธิภาพที่ดีและการคำนึงถึงจริยธรรม Anthropic มุ่งเน้นการพัฒนาระบบ AI ที่ปลอดภัยและมีความรับผิดชอบ โดยเน้นการกรองเนื้อหาและหลีกเลี่ยงผลลัพธ์ที่เป็นอันตราย

ระบบนิเวศของ Anthropic รวมถึง Claude API ที่ช่วยให้คุณผสานโมเดลเข้ากับแอปพลิเคชันของคุณ รวมถึงเครื่องมือสำหรับวิศวกรรมคำสั่งและการปรับแต่งละเอียด พวกเขายังนำเสนอโมเดล Claude Instant ที่รวมความสามารถในการค้นหาเว็บเพื่อการตอบสนองที่ทันสมัยและถูกต้องมากขึ้น

เมื่อใช้โมเดลของ Anthropic โดยตรง ผมใช้ไลบรารี anthrophic ของ Alex Rudall

Google

Google ได้พัฒนา LLM ที่ทรงพลังหลายตัว รวมถึง Gemini, BERT, T5 และ PaLM โมเดลเหล่านี้เป็นที่รู้จักจากประสิทธิภาพที่ดีในงานประมวลผลภาษาธรรมชาติที่หลากหลาย ระบบนิเวศของ Google รวมถึงไลบรารี TensorFlow และ Keras ซึ่งให้เครื่องมือและเฟรมเวิร์กสำหรับการสร้างและฝึกฝนโมเดลการเรียนรู้ของเครื่อง

Google ยังนำเสนอ Cloud AI Platform ที่ช่วยให้คุณสามารถปรับใช้และขยายโมเดลของพวกเขาบนคลาวด์ได้อย่างง่ายดาย พวกเขามีโมเดลที่ผ่านการเทรนมาแล้วและ API สำหรับงานต่างๆ เช่น การวิเคราะห์ความรู้สึก การจดจำเอนทิตี และการแปลภาษา

Meta

Meta (เดิมคือ Facebook) ลงทุนอย่างมากในการพัฒนาโมเดลภาษาขนาดใหญ่ โดยเด่นชัดจากการเปิดตัวโมเดลต่างๆ เช่น LLaMA และ OPT โมเดลเหล่านี้โดดเด่นด้วยประสิทธิภาพที่ดีในงานภาษาที่หลากหลายและมักจะเผยแพร่ผ่านช่องทางโอเพนซอร์ส สนับสนุนความมุ่งมั่นของ Meta ในด้านการวิจัยและการร่วมมือกับชุมชน

ระบบนิเวศของ Meta สร้างขึ้นรอบ PyTorch เป็นหลัก ซึ่งเป็นไลบรารีการเรียนรู้ของเครื่องแบบโอเพนซอร์สที่ได้รับความนิยมจากความสามารถในการคำนวณแบบไดนามิกและความยืดหยุ่น ช่วยอำนวยความสะดวกในการวิจัยและพัฒนา AI ที่เป็นนวัตกรรม

นอกเหนือจากบริการทางเทคนิค Meta ให้ความสำคัญอย่างมากกับการพัฒนา AI อย่างมีจริยธรรม พวกเขาใช้ระบบการกรองเนื้อหาที่แข็งแกร่งและมุ่งเน้นการลดอคติ ซึ่งสอดคล้องกับเป้าหมายที่กว้างขึ้นในด้านความปลอดภัยและความรับผิดชอบในการใช้งาน AI

Cohere

Cohere เป็นผู้เล่นรายใหม่ในวงการ LLM โดยมุ่งเน้นการทำให้ LLM เข้าถึงได้ง่ายและใช้งานง่ายกว่าคู่แข่ง ระบบนิเวศของพวกเขารวมถึง Cohere API ซึ่งให้การเข้าถึงโมเดลที่ผ่านการเทรนมาแล้วหลากหลายรูปแบบสำหรับงานต่างๆ เช่น การสร้างข้อความ การจำแนกประเภท และการสรุปความ

Cohere ยังนำเสนอเครื่องมือสำหรับการออกแบบพรอมต์ การปรับแต่งละเอียด และการกรองเนื้อหา พวกเขาเน้นย้ำเรื่องความเป็นส่วนตัวและความปลอดภัยของข้อมูล พร้อมฟีเจอร์ต่างๆ เช่น การเก็บข้อมูลแบบเข้ารหัสและการควบคุมการเข้าถึง

Ollama

Ollama เป็นแพลตฟอร์มที่โฮสต์ด้วยตนเอง ที่ช่วยให้ผู้ใช้สามารถจัดการและปรับใช้โมเดลภาษาขนาดใหญ่ (LLMs) ต่างๆ ในเครื่องของตนเองได้ ทำให้มีการควบคุมโมเดล AI อย่างสมบูรณ์โดยไม่ต้องพึ่งพาบริการคลาวด์ภายนอก การติดตั้งแบบนี้เหมาะสำหรับผู้ที่ให้ความสำคัญกับความเป็นส่วนตัวของข้อมูลและต้องการจัดการการทำงานของ AI ภายในองค์กรเอง

แพลตฟอร์มนี้รองรับโมเดลหลากหลาย รวมถึงเวอร์ชันต่างๆ ของ Llama, Phi, Gemma และ Mistral ซึ่งมีขนาดและความต้องการด้านการประมวลผลที่แตกต่างกัน Ollama ทำให้การดาวน์โหลดและรันโมเดลเหล่านี้ทำได้ง่ายผ่านคำสั่งบรรทัดคำสั่ง

อย่างง่ายๆ เช่น ollama run <model_name> และออกแบบมาให้ทำงานได้บนระบบปฏิบัติการต่างๆ ทั้ง macOS, Linux และ Windows

สำหรับนักพัฒนาที่ต้องการผสานโมเดลโอเพนซอร์สเข้ากับแอปพลิเคชันของตนโดยไม่ต้องใช้ API ระยะไกล Ollama มี CLI สำหรับจัดการวงจรชีวิตของโมเดลคล้ายกับเครื่องมือจัดการคอนเทนเนอร์ นอกจากนี้ยังรองรับการกำหนดค่าและพรอมต์แบบกำหนดเอง ช่วยให้สามารถปรับแต่งโมเดลให้เข้ากับความต้องการหรือกรณีการใช้งานเฉพาะได้อย่างมาก

Ollama เหมาะสำหรับผู้ใช้ที่มีความรู้ด้านเทคนิคและนักพัฒนาเป็นพิเศษ เนื่องจากมีอินเตอร์เฟซแบบบรรทัดคำสั่งและความยืดหยุ่นในการจัดการและปรับใช้โมเดล AI ทำให้เป็นเครื่องมือที่ทรงพลังสำหรับธุรกิจและบุคคลที่ต้องการความสามารถด้าน AI ที่แข็งแกร่งโดยไม่ต้องประนีประนอมเรื่องความปลอดภัยและการควบคุม

แพลตฟอร์มหลายโมเดล

นอกจากนี้ ยังมีผู้ให้บริการที่โฮสต์โมเดลโอเพนซอร์สหลากหลาย เช่น Together.ai และ Groq แพลตฟอร์มเหล่านี้มอบความยืดหยุ่นและการปรับแต่ง ช่วยให้คุณสามารถรันและในบางกรณีสามารถปรับแต่งละเอียดโมเดลโอเพนซอร์สตามความต้องการเฉพาะของคุณได้ ตัวอย่างเช่น Together.ai ให้การเข้าถึง LLMs แบบโอเพนซอร์สหลากหลาย ช่วยให้ผู้ใช้สามารถทดลองใช้โมเดลและการกำหนดค่าที่แตกต่างกัน Groq มุ่งเน้นที่การให้บริการการตอบสนองที่มีประสิทธิภาพสูงมาก ซึ่ง ณ เวลาที่เขียนหนังสือเล่มนี้ ดูเหมือนเวทมนตร์เลยทีเดียว

การเลือกผู้ให้บริการ LLM

เมื่อเลือกผู้ให้บริการ LLM คุณควรพิจารณาปัจจัยต่างๆ เช่น:

- **ราคา:** ผู้ให้บริการต่างๆ มีรูปแบบการคิดราคาที่แตกต่างกัน ตั้งแต่จ่ายตามการใช้งานไปจนถึงแผนแบบสมาชิก สิ่งสำคัญคือต้องพิจารณาการใช้งานที่คาดการณ์ไว้และงบประมาณเมื่อเลือกผู้ให้บริการ
- **ประสิทธิภาพ:** ประสิทธิภาพของ LLMs อาจแตกต่างกันอย่างมากระหว่างผู้ให้บริการ ดังนั้นจึงสำคัญที่จะต้องทดสอบเปรียบเทียบและทดสอบโมเดลในกรณีการใช้งานเฉพาะก่อนตัดสินใจ
- **การกรองเนื้อหา:** ขึ้นอยู่กับแอปพลิเคชัน การกรองเนื้อหาอาจเป็นข้อพิจารณาที่สำคัญ ผู้ให้บริการบางรายมีตัวเลือกการกรองเนื้อหาที่แข็งแกร่งกว่ารายอื่น
- **ความเป็นส่วนตัวของข้อมูล:** หากแอปพลิเคชันจัดการข้อมูลที่ละเอียดอ่อนของผู้ใช้ สิ่งสำคัญคือต้องเลือกผู้ให้บริการที่มีแนวทางปฏิบัติด้านความเป็นส่วนตัวและความปลอดภัยของข้อมูลที่เข้มแข็ง
- **การปรับแต่ง:** ผู้ให้บริการบางรายมีความยืดหยุ่นมากกว่าในแง่ของการปรับแต่งละเอียดและการปรับแต่งโมเดลสำหรับกรณีการใช้งานเฉพาะ

ในท้ายที่สุด การเลือกผู้ให้บริการ LLM ขึ้นอยู่กับความต้องการและข้อจำกัดเฉพาะของแอปพลิเคชัน โดยการประเมินตัวเลือกอย่างรอบคอบและพิจารณาปัจจัยต่างๆ เช่น ราคา ประสิทธิภาพ และความเป็นส่วนตัวของข้อมูล คุณสามารถเลือกผู้ให้บริการที่ตอบโจทย์ความต้องการของคุณได้ดีที่สุด

นอกจาก นี้ ยัง ควร ทราบ ว่า ภูมิ ทัศน์ ของ LLM มี การ เปลี่ยนแปลง อยู่ ตลอด เวลา โดย มี ผู้ ให้ บริการ และ โมเดล ใหม่ๆ เกิดขึ้นอย่างสม่ำเสมอ คุณควรติดตามพัฒนาการล่าสุดและเปิดใจที่จะสำรวจตัวเลือกใหม่ๆ เมื่อมีให้ใช้งาน

OpenRouter

ตลอดหนังสือเล่มนี้ ผมจะใช้ OpenRouter เป็นผู้ให้บริการ API หลักเพียงรายเดียว เหตุผลนั้นง่ายมาก: มันเป็นแหล่งรวมโมเดลที่เป็นที่นิยมทั้งเชิงพาณิชย์และโอเพนซอร์สไว้ในที่เดียว หากคุณกำลังอยากลองเขียนโค้ด AI หนึ่งในจุดเริ่มต้นที่ดีที่สุดคือ OpenRouter Ruby Library ที่ผมพัฒนาขึ้นเอง

การคิดเกี่ยวกับประสิทธิภาพ

เมื่อนำโมเดลภาษามาใช้ในแอปพลิเคชัน ประสิทธิภาพถือเป็นปัจจัยสำคัญที่ต้องพิจารณา ประสิทธิภาพของโมเดลภาษาสามารถวัดได้จาก *ความหน่วง* (เวลาที่ใช้ในการสร้างการตอบสนอง) และ *ปริมาณงานที่ทำได้* (จำนวนคำขอที่สามารถจัดการได้ต่อหนึ่งหน่วยเวลา)

เวลาถึงโทเค็นแรก (TTFT) เป็นตัวชี้วัดประสิทธิภาพที่สำคัญอีกอย่างหนึ่ง โดยเฉพาะอย่างยิ่งสำหรับแชตบอทและแอปพลิเคชันที่ต้องการการตอบสนองแบบโต้ตอบแบบเรียลไทม์ TTFT วัดความหน่วงตั้งแต่เวลาที่ได้รับคำขอของผู้ใช้จนถึงเวลาที่คำ (หรือโทเค็น) แรกของการตอบสนองถูกสร้างขึ้น ตัวชี้วัดนี้มีความสำคัญอย่างยิ่งในการรักษาประสบการณ์ผู้ใช้ที่ราบรื่นและน่าสนใจ เนื่องจากการตอบสนองที่ล่าช้าอาจนำไปสู่ความหงุดหงิดและการเลิกใช้งานของผู้ใช้

ตัวชี้วัดประสิทธิภาพเหล่านี้สามารถส่งผลกระทบอย่างมีนัยสำคัญต่อประสบการณ์ผู้ใช้และความสามารถในการขยายตัวของแอปพลิเคชัน

มีหลายปัจจัยที่สามารถส่งผลต่อประสิทธิภาพของโมเดลภาษา ได้แก่:

จำนวนพารามิเตอร์: โมเดล ขนาด ใหญ่ ที่ มี พารามิเตอร์ มากกว่า โดย ทั่วไป ต้องการ ทรัพยากร การ คำนวณ มาก ขึ้น และ อาจ มี ความหน่วงสูงขึ้นและปริมาณงานที่ทำได้ต่ำลงเมื่อเทียบกับโมเดลขนาดเล็กกว่า

ฮาร์ดแวร์: ประสิทธิภาพของโมเดลภาษาสามารถแตกต่างกันอย่างมีนัยสำคัญขึ้นอยู่กับฮาร์ดแวร์ที่ใช้งาน ผู้ให้บริการคลาวด์มีบริการ GPU และ TPU ที่ได้รับการปรับแต่งสำหรับงานแมชชีนเลิร์นนิง ซึ่งสามารถเร่งการอนุมานโมเดลได้อย่างมาก

หนึ่งในข้อดีของ OpenRouter คือสำหรับโมเดลจำนวนมากที่มีให้บริการ คุณสามารถเลือกผู้ให้บริการคลาวด์ที่มีประสิทธิภาพและต้นทุนที่หลากหลายได้

การควอนไทซ์: เทคนิคการควอนไทซ์สามารถใช้เพื่อลดการใช้หน่วยความจำและความต้องการในการคำนวณของโมเดลโดยการแทนค่าน้ำหนักและการกระตุ้นด้วยชนิดข้อมูลที่มีความแม่นยำต่ำกว่า วิธีนี้สามารถปรับปรุงประสิทธิภาพโดยไม่สูญเสียคุณภาพอย่างมีนัยสำคัญ ในฐานะนักพัฒนาแอปพลิเคชัน คุณอาจจะไม่ได้เข้าไปยุ่งเกี่ยวกับการฝึกโมเดลของคุณเองที่ระดับการควอนไทซ์ต่างๆ แต่ก็ควรจะคุ้นเคยกับคำศัพท์เหล่านี้

การประมวลผลแบบแบตช์: การประมวลผลคำขอหลายรายการพร้อมกันเป็นชุดสามารถปรับปรุงปริมาณงานที่ทำได้โดยการกระจายค่าใช้จ่ายในการโหลดโมเดลและการถ่ายโอนข้อมูล

การแคช: การแคชผลลัพธ์ของพรอมต์หรือลำดับอินพุตที่ใช้บ่อยสามารถลดจำนวนคำขอการอนุมานและปรับปรุงประสิทธิภาพโดยรวม

เมื่อเลือกโมเดลภาษาสำหรับแอปพลิเคชันที่ใช้งานจริง สิ่งสำคัญคือต้องทดสอบประสิทธิภาพบนภาระงานและการกำหนดค่าฮาร์ดแวร์ที่เป็นตัวแทน สิ่งนี้สามารถช่วยระบุคอขวดที่อาจเกิดขึ้นและทำให้มั่นใจว่าโมเดลสามารถบรรลุเป้าหมายประสิทธิภาพที่ต้องการได้

นอกจากนี้ยังควรพิจารณาการแลกเปลี่ยนระหว่างประสิทธิภาพของโมเดลและปัจจัยอื่นๆ เช่น ต้นทุน ความยืดหยุ่น และความง่ายในการบูรณาการ ตัวอย่างเช่น การใช้โมเดลขนาดเล็กที่มีราคาถูกกว่าและมีความหน่วงต่ำกว่าอาจเหมาะสมกว่าสำหรับแอปพลิเคชันที่ต้องการการตอบสนองแบบเรียลไทม์ ในขณะที่โมเดลขนาดใหญ่ที่มีประสิทธิภาพมากกว่าอาจเหมาะสมกว่าสำหรับการประมวลผลแบบแบตช์หรืองานที่ต้องการการให้เหตุผลที่ซับซ้อน

การทดลองกับโมเดล LLM ต่างๆ

การเลือก LLM แทบจะไม่ใช่การตัดสินใจถาวร เนื่องจากมีการปล่อยโมเดลใหม่และปรับปรุงอย่างสม่ำเสมอ จึงเป็นการดีที่จะสร้างแอปพลิเคชันในแบบโมดูลาร์ที่อนุญาตให้สลับโมเดลภาษาต่างๆ ได้ตลอดเวลา พรอมต์และชุดข้อมูลมักสามารถนำกลับมาใช้กับโมเดลต่างๆ ได้โดยต้องปรับเปลี่ยนเพียงเล็กน้อย สิ่งนี้ช่วยให้คุณสามารถใช้ประโยชน์จากความก้าวหน้าล่าสุดในการสร้างโมเดลภาษาโดยไม่ต้องออกแบบแอปพลิเคชันใหม่ทั้งหมด

ความสามารถในการสลับระหว่างตัวเลือกโมเดลที่หลากหลายได้อย่างง่ายดายเป็นอีกเหตุผลหนึ่งที่ผมชอบ OpenRouter

เมื่อปรับเวอร์ชันเป็นโมเดลภาษาใหม่ สิ่งสำคัญคือต้องทดสอบและตรวจสอบประสิทธิภาพและคุณภาพของผลลัพธ์อย่างละเอียด เพื่อให้แน่ใจว่าตรงตามความต้องการของแอปพลิเคชัน อาจต้องมีการฝึกซ้ำหรือปรับแต่งโมเดลกับข้อมูลเฉพาะโดเมน รวมถึงการอัปเดตคอมโพเนนต์ปลายทางใดๆ ที่ขึ้นอยู่กับผลลัพธ์ของโมเดล

การออกแบบแอปพลิเคชันโดยคำนึงถึงประสิทธิภาพและความเป็นโมดูลาร์ จะช่วยให้คุณสามารถสร้างระบบที่ขยายตัวได้ มีประสิทธิภาพ และพร้อมรับมือกับอนาคต ซึ่งสามารถปรับตัวเข้ากับภูมิทัศน์ของเทคโนโลยีการสร้างโมเดลภาษาที่กำลังพัฒนาอย่างรวดเร็ว

ระบบ AI แบบผสมผสาน

ก่อนจะจบบทนำของเรา มีประเด็นที่ควรกล่าวถึงคือก่อนปี 2023 และการระเบิดของความสนใจในเรื่อง AI เชิงสร้างสรรค์ที่จุดประกายโดย ChatGPT แนวทาง AI แบบดั้งเดิมมักจะพึ่งพาการบูรณาการโมเดลเดี่ยวแบบปิด ในทางตรงกันข้าม *ระบบ AI แบบผสมผสาน* ใช้ประโยชน์จากไปป์ไลน์ที่ซับซ้อนของคอมโพเนนต์ที่เชื่อมต่อกันเพื่อให้บรรลุพฤติกรรมอัจฉริยะ

ในแก่นแท้แล้ว ระบบ AI แบบผสมผสานประกอบด้วยโมดูลหลายตัว แต่ละตัวถูกออกแบบมาเพื่อทำงานหรือฟังก์ชันเฉพาะ โมดูลเหล่านี้อาจรวมถึงตัวสร้าง ตัวค้นคืน ตัวจัดอันดับ ตัวจำแนกประเภท และคอมโพเนนต์เฉพาะทางอื่นๆ การแยกระบบโดยรวมออกเป็นหน่วยย่อยที่มุ่งเน้นเฉพาะด้าน ช่วยให้นักพัฒนาสามารถสร้างสถาปัตยกรรม AI ที่มีความยืดหยุ่น ขยายตัวได้ และดูแลรักษาได้มากขึ้น

ข้อได้เปรียบที่สำคัญประการหนึ่งของระบบ AI แบบผสมผสานคือความสามารถในการรวมจุดแข็งของเทคนิคและโมเดล AI ที่แตกต่างกัน ตัวอย่างเช่น ระบบหนึ่งอาจใช้โมเดลภาษาขนาดใหญ่ (LLM) สำหรับการทำความเข้าใจและการสร้างภาษาธรรมชาติ ในขณะที่ใช้โมเดลแยกต่างหากสำหรับการค้นคืนข้อมูลหรือการตัดสินใจตามกฎ วิธีการแบบโมดูลาร์นี้ช่วยให้คุณสามารถเลือกเครื่องมือและเทคนิคที่ดีที่สุดสำหรับงานเฉพาะแต่ละอย่าง แทนที่จะพึ่งพาวิธีแก้ปัญหาแบบเดียวที่ใช้ได้กับทุกอย่าง

อย่างไรก็ตาม การสร้างระบบ AI แบบผสมผสาน ก็มีความท้าทายที่เป็นเอกลักษณ์เฉพาะ โดยเฉพาะอย่างยิ่ง การรับประกันความสอดคล้องและความคงเส้นคงวาของพฤติกรรมระบบโดยรวมต้องการกลไกการทดสอบ การตรวจสอบ และการกำกับดูแลที่แข็งแกร่ง

การมาถึงของ LLM ที่ทรงพลังอย่าง GPT-4 ทำให้เราสามารถทดลองกับระบบ AI แบบผสมผสานได้ง่ายกว่าที่เคย เพราะโมเดลขั้นสูงเหล่านี้สามารถจัดการบทบาทหลายอย่างภายในระบบผสมผสาน เช่น การจำแนกประเภท การจัดอันดับ และการสร้างเนื้อหา นอกเหนือจากความสามารถในการเข้าใจภาษาธรรมชาติ ความอเนกประสงค์นี้ช่วยให้นักพัฒนาสามารถสร้างต้นแบบและปรับปรุงสถาปัตยกรรม AI แบบผสมผสานได้อย่างรวดเร็ว เปิดโอกาสใหม่ๆ สำหรับการพัฒนาแอปพลิเคชันอัจฉริยะ

รูปแบบการติดตั้งใช้งานสำหรับระบบ AI แบบผสมผสาน

ระบบ AI แบบผสมผสานสามารถติดตั้งใช้งานได้หลายรูปแบบ โดยแต่ละรูปแบบออกแบบมาเพื่อตอบสนองความต้องการและกรณีการใช้งานเฉพาะ มาสำรวจรูปแบบการติดตั้งใช้งานที่พบบ่อย 4 รูปแบบกัน: ระบบถาม-ตอบ ระบบแก้ปัญหาแบบหลายตัวแทน/เอเจนติก, AI สนทนา และผู้ช่วยร่วม

ระบบถาม-ตอบ

ระบบถาม-ตอบ (Q&A) มุ่งเน้นที่การค้นคืนข้อมูลที่เพิ่มประสิทธิภาพด้วยความสามารถในการทำความเข้าใจของโมเดล AI เพื่อให้ทำงานได้มากกว่าเครื่องมือค้นหาทั่วไป โดยการรวมโมเดลภาษาที่ทรงพลังเข้ากับแหล่งความรู้ภายนอกโดยใช้ การสร้างเนื้อหาที่เสริมด้วยการค้นคืน (RAG) ระบบถาม-ตอบสามารถหลีกเลี่ยงการสร้างข้อมูลที่ไม่ถูกต้องและให้คำตอบที่แม่นยำและเกี่ยวข้องกับบริบทสำหรับคำถามของผู้ใช้

องค์ประกอบสำคัญของระบบถาม-ตอบที่ใช้ LLM ประกอบด้วย:

- **การทำความเข้าใจและปรับแต่งคำถาม:** วิเคราะห์คำถามของผู้ใช้และปรับแต่งให้ตรงกับแหล่งความรู้พื้นฐานมากขึ้น
- **การค้นคืนความรู้:** ค้นคืนข้อมูลที่เกี่ยวข้องจากแหล่งข้อมูลที่มีโครงสร้างหรือไม่มีโครงสร้างตามคำถามที่ปรับแต่งแล้ว
- **การสร้างคำตอบ:** สร้างคำตอบที่เชื่อมโยงและให้ข้อมูลโดยผสมผสานความรู้ที่ค้นคืนได้กับความสามารถในการสร้างเนื้อหาของโมเดลภาษา

ระบบย่อย RAG มีความสำคัญเป็นพิเศษในด้านถาม-ตอบที่ต้องการให้ข้อมูลที่แม่นยำและทันสมัย เช่น การสนับสนุนลูกค้า การจัดการความรู้ หรือแอปพลิเคชันการศึกษา

ระบบแก้ปัญหาแบบหลายตัวแทน/เอเจนติก

ระบบหลายตัวแทน หรือที่รู้จักในชื่อ *เอเจนติก* ประกอบด้วยตัวแทนอิสระหลายตัวที่ทำงานร่วมกันเพื่อแก้ปัญหาที่ซับซ้อน แต่ละตัวแทนมีบทบาท ชุดทักษะ และการเข้าถึงเครื่องมือหรือแหล่งข้อมูลที่เกี่ยวข้องเฉพาะ ด้วยการทำงานร่วมกันและแลกเปลี่ยนข้อมูล ตัวแทนเหล่านี้สามารถจัดการกับงานที่ยากหรือเป็นไปไม่ได้สำหรับตัวแทนเดียวที่จะจัดการได้

หลักการสำคัญของระบบแก้ปัญหาแบบหลายตัวแทนประกอบด้วย:

- **การเชี่ยวชาญเฉพาะด้าน:** แต่ละตัวแทนมุ่งเน้นที่แง่มุมเฉพาะของปัญหา โดยใช้ความสามารถและความรู้เฉพาะของตน
- **การทำงานร่วมกัน:** ตัวแทนสื่อสารและประสานงานการกระทำของตนเพื่อบรรลุเป้าหมายร่วมกัน มักจะผ่านการส่งข้อความหรือหน่วยความจำร่วม

- **การปรับตัว:** ระบบสามารถปรับตัวเข้ากับสภาวะหรือความต้องการที่เปลี่ยนแปลงโดยการปรับบทบาทและพฤติกรรมของตัวแทนแต่ละตัว

ระบบหลายตัวแทนเหมาะสำหรับแอปพลิเคชันที่ต้องการการแก้ปัญหาแบบกระจาย เช่น การเพิ่มประสิทธิภาพห่วงโซ่อุปทาน การจัดการจราจร หรือการวางแผนตอบสนองต่อเหตุฉุกเฉิน

AI สนทนา

ระบบ AI สนทนาช่วยให้เกิดการโต้ตอบด้วยภาษาธรรมชาติระหว่างผู้ใช้และตัวแทนอัจฉริยะ ระบบเหล่านี้รวมความสามารถในการทำความเข้าใจภาษาธรรมชาติ การจัดการบทสนทนา และการสร้างภาษาเพื่อให้ประสบการณ์การสนทนาที่น่าสนใจและเป็นส่วนตัว

องค์ประกอบหลักของระบบ AI สนทนาประกอบด้วย:

- **การรู้จำเจตนา:** ระบุเจตนาของผู้ใช้จากข้อมูลที่ป้อนเข้า เช่น การถามคำถาม การร้องขอ หรือการแสดงความรู้สึก
- **การดึงเอนทิตี:** ดึงเอนทิตีหรือพารามิเตอร์ที่เกี่ยวข้องจากข้อมูลที่ผู้ใช้ป้อน เช่น วันที่ สถานที่ หรือชื่อผลิตภัณฑ์
- **การจัดการบทสนทนา:** รักษาสถานะของการสนทนา กำหนดการตอบสนองที่เหมาะสมตามเจตนาและบริบทของผู้ใช้ และจัดการการโต้ตอบหลายรอบ
- **การสร้างการตอบสนอง:** สร้างการตอบสนองที่เหมือนมนุษย์โดยใช้โมเดลภาษา เทมเพลต หรือวิธีการที่ใช้การค้นคืน

ระบบ AI สนทนามักใช้ในแชทบอทบริการลูกค้า ผู้ช่วยเสมือน และอินเทอร์เฟซควบคุมด้วยเสียง ตามที่ได้กล่าวไว้ก่อนหน้านี้ แนวทาง รูปแบบ และตัวอย่างโค้ดส่วนใหญ่ในหนังสือเล่มนี้ถูกดึงมาโดยตรงจากงานของฉันในระบบ AI สนทนาขนาดใหญ่ที่เรียกว่า Olympia

โคไพลอต

โคไพลอตคือผู้ช่วยที่ขับเคลื่อนด้วย AI ที่ทำงานร่วมกับผู้ใช้งานมนุษย์เพื่อเพิ่มประสิทธิภาพการทำงานและความสามารถในการตัดสินใจ ระบบเหล่านี้ใช้การผสมผสานระหว่างการประมวลผลภาษาธรรมชาติ การเรียนรู้ของเครื่อง และความรู้เฉพาะด้านเพื่อให้คำแนะนำที่ชาญฉลาด ทำงานอัตโนมัติ และให้การสนับสนุนตามบริบท

คุณสมบัติสำคัญของโคไพลอตประกอบด้วย:

- **การปรับแต่งส่วนบุคคล:** ปรับตัวตามความชอบ ขั้นตอนการทำงาน และรูปแบบการสื่อสารของผู้ใช้แต่ละคน

- **การช่วยเหลือเชิงรุก:** คาดการณ์ความต้องการของผู้ใช้และเสนอคำแนะนำหรือการกระทำที่เกี่ยวข้องโดยไม่ต้องมีการร้องขอโดยตรง
- **การเรียนรู้อย่างต่อเนื่อง:** พัฒนาประสิทธิภาพเมื่อเวลาผ่านไปโดยเรียนรู้จากข้อเสนอแนะ การโต้ตอบ และข้อมูลของผู้ใช้

โคไพลอตถูกนำมาใช้เพิ่มขึ้นในหลากหลายด้าน เช่น การพัฒนาซอฟต์แวร์ (เช่น การเติมโค้ดให้สมบูรณ์และการตรวจจับข้อผิดพลาด) การเขียนเชิงสร้างสรรค์ (เช่น การแนะนำเนื้อหาและการแก้ไข) และการวิเคราะห์ข้อมูล (เช่น การให้ข้อมูลเชิงลึกและคำแนะนำในการแสดงผลข้อมูล)

รูปแบบการใช้งานเหล่านี้แสดงให้เห็นถึงความหลากหลายและศักยภาพของระบบ AI แบบผสมผสาน การทำความเข้าใจลักษณะและกรณีการใช้งานของแต่ละรูปแบบจะช่วยให้คุณสามารถตัดสินใจได้อย่างมีข้อมูลเมื่อออกแบบและพัฒนาแอปพลิเคชันอัจฉริยะ แม้ว่าหนังสือเล่มนี้จะไม่ได้เน้นเฉพาะเรื่องการพัฒนาระบบ AI แบบผสมผสาน แต่แนวทางและรูปแบบเดียวกันเกือบทั้งหมดสามารถนำไปใช้ในการผสานส่วนประกอบ AI แบบแยกส่วนเข้ากับการพัฒนาแอปพลิเคชันแบบดั้งเดิม

บทบาทในระบบ AI แบบผสมผสาน

ระบบ AI แบบผสมผสานถูกสร้างขึ้นบนพื้นฐานของโมดูลที่เชื่อมต่อกัน แต่ละโมดูลถูกออกแบบมาเพื่อทำหน้าที่เฉพาะ โมดูลเหล่านี้ทำงานร่วมกันเพื่อสร้างพฤติกรรมอัจฉริยะและแก้ปัญหาที่ซับซ้อน การรู้จักบทบาทเหล่านี้จะเป็นประโยชน์เมื่อคิดถึงส่วนที่คุณอาจสามารถนำไปใช้หรือแทนที่ส่วนต่างๆ ของแอปพลิเคชันของคุณด้วยส่วนประกอบ AI แบบแยกส่วน

ตัวสร้าง

ตัวสร้างมีหน้าที่ผลิตข้อมูลหรือเนื้อหาใหม่โดยอิงจากรูปแบบที่เรียนรู้มาหรือคำสั่งที่ป้อนเข้าไป โลกของ AI มีตัวสร้างหลายประเภท แต่ในบริบทของโมเดลภาษาที่นำเสนอในหนังสือเล่มนี้ ตัวสร้างสามารถสร้างข้อความที่เหมือนมนุษย์ เติมประโยคให้สมบูรณ์ หรือสร้างการตอบสนองต่อคำถามของผู้ใช้ พวกมันมีบทบาทสำคัญในงานต่างๆ เช่น การสร้างเนื้อหา การสร้างบทสนทนา และการเพิ่มข้อมูล

ตัวค้นคืน

ตัวค้นคืนใช้ในการค้นหาและดึงข้อมูลที่เกี่ยวข้องจากชุดข้อมูลขนาดใหญ่หรือฐานความรู้ พวกมันใช้เทคนิคต่างๆ เช่น การค้นหาเชิงความหมาย การจับคู่คำสำคัญ หรือความคล้ายคลึงของเวกเตอร์เพื่อค้นหาจุดข้อมูลที่เกี่ยวข้องมากที่สุดตามคำค้นหรือบริบทที่กำหนด ตัวค้นคืนมีความสำคัญสำหรับงานที่ต้องการการเข้าถึงข้อมูลเฉพาะอย่างรวดเร็ว เช่น การตอบคำถาม การตรวจสอบข้อเท็จจริง หรือการแนะนำเนื้อหา

ตัวจัดอันดับ

ตัวจัดอันดับมีหน้าที่เรียงลำดับหรือจัดลำดับความสำคัญของรายการต่างๆ ตามเกณฑ์หรือคะแนนความเกี่ยวข้อง พวกมันกำหนดน้ำหนักหรือคะแนนให้แต่ละรายการแล้วเรียงลำดับตามนั้น ตัวจัดอันดับถูกใช้บ่อยในเครื่องมือค้นหา ระบบแนะนำ หรือแอปพลิเคชันใดๆ ที่การแสดงผลลัพธ์ที่เกี่ยวข้องที่สุดแก่ผู้ใช้มีความสำคัญ

ตัวจำแนกประเภท

ตัวจำแนกประเภทใช้ในการจัดหมวดหมู่หรือติดป้ายจุดข้อมูลตามคลาสหรือหมวดหมู่ที่กำหนดไว้ล่วงหน้า พวกมันเรียนรู้จากข้อมูลฝึกฝนที่มีป้ายกำกับและทำนายคลาสของตัวอย่างใหม่ที่ไม่เคยเห็นมาก่อน ตัวจำแนกประเภทเป็นพื้นฐานสำหรับงานต่างๆ เช่น การวิเคราะห์ความรู้สึก การตรวจจับสแปม หรือการรู้จำภาพ ซึ่งมีเป้าหมายเพื่อกำหนดหมวดหมู่เฉพาะให้กับแต่ละข้อมูลที่ป้อนเข้าไป

เครื่องมือและตัวแทน

นอกเหนือจากบทบาทหลักเหล่านี้ ระบบ AI แบบผสมผสานมักรวมเครื่องมือและตัวแทนเพื่อเพิ่มฟังก์ชันการทำงานและความสามารถในการปรับตัว:

- **เครื่องมือ:** เครื่องมือคือส่วนประกอบซอฟต์แวร์แบบแยกส่วนหรือ API ที่ทำงานเฉพาะหรือการคำนวณ พวกมันสามารถถูกเรียกใช้โดยโมดูลอื่นๆ เช่น ตัวสร้างหรือตัวค้นคืน เพื่อทำงานย่อยหรือรวบรวมข้อมูลเพิ่มเติม ตัวอย่างของเครื่องมือได้แก่ เครื่องมือค้นหาเว็บ เครื่องคิดเลข หรือไลบรารีสำหรับการแสดงผลข้อมูล
- **ตัวแทน:** ตัวแทนคือเอนทิตีอิสระที่สามารถรับรู้สภาพแวดล้อม ตัดสินใจ และดำเนินการเพื่อบรรลุเป้าหมายเฉพาะ พวกมันมักพึ่งพาการผสมผสานเทคนิค AI ต่างๆ เช่น การวางแผน การให้เหตุผล และการเรียนรู้ เพื่อทำงานได้อย่างมีประสิทธิภาพในสภาวะที่เปลี่ยนแปลงหรือไม่แน่นอน ตัวแทนสามารถใช้ในการจำลองพฤติกรรมที่ซับซ้อนหรือประสานงานการกระทำของหลายโมดูลภายในระบบ AI แบบผสมผสาน

ในระบบ AI แบบผสมผสานที่แท้จริง การโต้ตอบระหว่างส่วนประกอบเหล่านี้ถูกจัดการผ่านอินเตอร์เฟซและโปรโตคอลการสื่อสารที่กำหนดไว้อย่างชัดเจน ข้อมูลไหลระหว่างโมดูล โดยผลลัพธ์ของส่วนประกอบหนึ่งจะเป็นข้อมูลนำเข้าสำหรับอีกส่วนประกอบหนึ่ง สถาปัตยกรรมแบบโมดูลาร์นี้ช่วยให้มีความยืดหยุ่น ขยายขนาดได้ และบำรุงรักษาได้ง่าย เนื่องจากส่วนประกอบแต่ละส่วนสามารถอัปเดต แทนที่ หรือขยายได้โดยไม่กระทบกับระบบทั้งหมด

การใช้ประโยชน์จากพลังของส่วนประกอบเหล่านี้และการโต้ตอบระหว่างกัน ระบบ AI แบบผสมผสานสามารถจัดการกับปัญหาในโลกจริงที่ซับซ้อนซึ่งต้องการการผสมผสานความสามารถ AI ที่หลากหลาย ขณะที่เราสำรวจแนวทางและรูปแบบสำหรับการ

ผสาน AI เข้ากับกระบวนการพัฒนาแอปพลิเคชัน โปรดจำไว้ว่าหลักการและเทคนิคเดียวกันที่ใช้ในระบบ AI แบบผสมผสานสามารถนำไปใช้ในการสร้างแอปพลิเคชันที่ชาญฉลาด ปรับตัวได้ และเน้นผู้ใช้เป็นศูนย์กลาง

ในบทต่อๆ ไปของส่วนที่ 1 เราจะเจาะลึกลงไปในแนวทางและเทคนิคพื้นฐานสำหรับการผสานส่วนประกอบ AI เข้ากับกระบวนการพัฒนาแอปพลิเคชันของคุณ ตั้งแต่การออกแบบพรอมต์และการสร้างแบบเพิ่มประสิทธิภาพด้วยการค้นคืน ไปจนถึงข้อมูลที่ซ่อมแซมตัวเองและการจัดการเวิร์กโฟลว์อัจฉริยะ เราจะครอบคลุมรูปแบบและแนวปฏิบัติที่ดีที่สุดมากมายเพื่อช่วยให้คุณสร้างแอปพลิเคชันที่ขับเคลื่อนด้วย AI ที่ล้ำสมัย

ส่วนที่ 1: แนวทางและเทคนิคพื้นฐาน

ส่วนนี้ของหนังสือนำเสนอวิธีการต่างๆ ในการผสานการใช้งาน AI เข้ากับแอปพลิเคชันของคุณ บทต่างๆ ครอบคลุมแนวทางและเทคนิคที่เกี่ยวข้องหลากหลาย ตั้งแต่แนวคิดระดับสูงอย่าง การจำกัดขอบเขต และ การสร้างเนื้อหาแบบเสริมด้วยการค้นคืน ไปจนถึงแนวคิดในการเขียนโปรแกรมชั้นการซ่อนรายละเอียดของคุณเองบน API การต่อประโยคแชทของ LLM

เป้าหมายของเนื้อหาส่วนนี้คือช่วยให้คุณเข้าใจรูปแบบพฤติกรรมต่างๆ ที่คุณสามารถนำ AI ไปใช้งานได้ ก่อนที่จะลงลึกในรูปแบบการนำไปใช้งานเฉพาะที่จะเป็นจุดเน้นใน ส่วนที่ 2

แนวทางในส่วนที่ 1 นี้อ้างอิงจากแนวคิดที่ผมใช้ในโค้ดของผม รูปแบบคลาสสิกของสถาปัตยกรรมและการผสานแอปพลิเคชันองค์กร รวมถึงอุปมาอุปไมยที่ผมใช้ในการอธิบายความสามารถของ AI ให้กับผู้อื่น รวมถึงผู้มีส่วนได้ส่วนเสียทางธุรกิจที่ไม่ใช่ผู้เชี่ยวชาญด้านเทคนิค

การจำกัดขอบเขต

“การจำกัดขอบเขต” หมายถึงการทำให้ AI มุ่งเน้นไปที่งานที่กำลังทำอยู่ ผมใช้มันเป็นคำสอนใจทุกครั้งที่รู้สึกหงุดหงิดเมื่อ AI ทำงาน “โง่” หรือในแบบที่ไม่คาดคิด คำสอนนี้เตือนผมว่าความล้มเหลวนั้นน่าจะเป็นความผิดของผมเอง และผมควรจะจำกัดขอบเขตให้มากขึ้น

ความจำเป็นในการจำกัดขอบเขตเกิดจากความรู้มหาศาลที่บรรจุอยู่ในโมเดลภาษาขนาดใหญ่ โดยเฉพาะอย่างยิ่งโมเดลระดับโลกอย่างเช่นจาก OpenAI และ Anthropic ที่มีพารามิเตอร์นับล้านล้าน

การเข้าถึงความรู้ที่กว้างขวางเช่นนี้มีพลังอย่างไม่ต้องสงสัยและก่อให้เกิดพฤติกรรมที่เกิดขึ้นใหม่ เช่น ทฤษฎีจิต และความสามารถในการให้เหตุผลแบบมนุษย์ อย่างไรก็ตาม ปริมาณข้อมูลมหาศาลที่น่าตะลึงนี้ก็สร้างความท้าทายเมื่อต้องสร้างคำตอบที่

แม่นยำและถูกต้องสำหรับคำสั่งเฉพาะ โดยเฉพาะอย่างยิ่งถ้าคำสั่งเหล่านั้นต้องการพฤติกรรมที่คาดเดาได้ซึ่งสามารถผสานรวมกับการพัฒนาซอฟต์แวร์และอัลกอริธึม “ทั่วไป”

มีหลายปัจจัยที่นำไปสู่ความท้าทายเหล่านี้

ข้อมูลล้นเกิน: โมเดลภาษาขนาดใหญ่ได้รับการฝึกฝนด้วยข้อมูลจำนวนมหาศาลที่ครอบคลุมหลากหลายโดเมน แหล่งที่มา และช่วงเวลา ความรู้ที่กว้างขวางนี้ทำให้โมเดลสามารถมีส่วนร่วมในหัวข้อที่หลากหลายและสร้างคำตอบบนพื้นฐานความเข้าใจที่กว้างขวางเกี่ยวกับโลก อย่างไรก็ตาม เมื่อเผชิญกับคำสั่งเฉพาะ โมเดลอาจจะมีปัญหาในการกรองข้อมูลที่ไม่เกี่ยวข้อง ขัดแย้งกัน หรือล้าสมัย/เก่าแล้ว ซึ่งนำไปสู่คำตอบที่ขาดจุดเน้นหรือความแม่นยำ ขึ้นอยู่กับสิ่งที่คุณพยายามจะทำ ปริมาณข้อมูลที่ *ขัดแย้งกัน* ที่มีให้กับโมเดลสามารถครอบงำความสามารถของมันในการให้คำตอบหรือพฤติกรรมที่คุณต้องการได้อย่างง่ายดาย

ความกำกวมของบริบท: เมื่อพิจารณาถึง *ปริภูมิแฝง* ของความรู้อันกว้างใหญ่ โมเดลภาษาขนาดใหญ่อาจเผชิญกับความกำกวมเมื่อพยายามทำความเข้าใจ *บริบท* ของคำสั่งของคุณ หากไม่มีการจำกัดขอบเขตหรือแนวทางที่เหมาะสม โมเดลอาจสร้างคำตอบที่เกี่ยวข้องเพียงผิวเผินแต่ไม่ตรงกับจุดประสงค์ของคุณโดยตรง ความล้มเหลวแบบนี้นำไปสู่คำตอบที่ออกนอกประเด็น ไม่สอดคล้องกัน หรือไม่ตอบสนองความต้องการที่ระบุไว้ ในกรณีนี้ การจำกัดขอบเขตหมายถึงการ *ขจัดความกำกวม* ของบริบทเพื่อให้มั่นใจว่าบริบทที่คุณให้ทำให้โมเดลมุ่งเน้นเฉพาะข้อมูลที่เกี่ยวข้องมากที่สุดในความรู้พื้นฐานของมัน

หมายเหตุ: เมื่อคุณเริ่มต้นกับ “การออกแบบคำสั่ง” คุณมีแนวโน้มที่จะขอให้โมเดลทำสิ่งต่างๆ โดยไม่อธิบายผลลัพธ์ที่ต้องการอย่างเหมาะสม ต้องอาศัยการฝึกฝนเพื่อไม่ให้คำสั่งคลุมเครือ!

ความไม่สอดคล้องเชิงเวลา: เนื่องจากโมเดลภาษาได้รับการฝึกฝนด้วยข้อมูลที่สร้างขึ้นในช่วงเวลาที่แตกต่างกัน พวกมันอาจมีความรู้ที่ล้าสมัย ถูกแทนที่ หรือไม่ถูกต้องอีกต่อไป ตัวอย่างเช่น ข้อมูลเกี่ยวกับเหตุการณ์ปัจจุบัน การค้นพบทางวิทยาศาสตร์ หรือความก้าวหน้าทางเทคโนโลยีอาจมีการพัฒนาไปแล้วตั้งแต่มีการรวบรวมข้อมูลฝึกฝนของโมเดล หากไม่มีการจำกัดขอบเขตให้ให้ความสำคัญกับแหล่งข้อมูลที่ใหม่กว่าและน่าเชื่อถือกว่า โมเดลอาจสร้างคำตอบบนพื้นฐานของข้อมูลที่ล้าสมัยหรือไม่ถูกต้อง นำไปสู่ความไม่ถูกต้องและความไม่สอดคล้องในผลลัพธ์

ความละเอียดอ่อนเฉพาะโดเมน: โดเมนและสาขาต่างๆ มีคำศัพท์เฉพาะ ข้อตกลง และฐานความรู้ของตัวเอง ลองคิดถึงคำย่อสามตัวอักษร (TLA) แทบทุกคำ และคุณจะพบว่าส่วนใหญ่มีความหมายมากกว่าหนึ่งความหมาย ตัวอย่างเช่น MSK อาจหมายถึง Managed Streaming for Apache Kafka ของ Amazon, Memorial Sloan Kettering Cancer Center หรือระบบกล้ามเนื้อและกระดูก (MusculoSKeletal) ของมนุษย์

เมื่อคำสั่งต้องการความเชี่ยวชาญในโดเมนเฉพาะ ความรู้ทั่วไปของโมเดลภาษาขนาดใหญ่อาจไม่เพียงพอที่จะให้คำตอบที่แม่นยำและละเอียดอ่อน การจำกัดขอบเขตโดยมุ่งเน้นไปที่ข้อมูลเฉพาะโดเมน ไม่ว่าจะผ่านการออกแบบคำสั่งหรือการสร้างที่เสริมด้วยการค้นคืน ช่วยให้โมเดลสามารถสร้างคำตอบที่สอดคล้องกับข้อกำหนดและความคาดหวังของโดเมนเฉพาะของคุณมากขึ้น

ปริภูมิแฝง: กว้างใหญ่เกินกว่าจะเข้าใจ

เมื่อผมพูดถึง “ปริภูมิแฝง” ของโมเดลภาษา ผมกำลังพูดถึงภูมิทัศน์หลายมิติอันกว้างใหญ่ของความรู้และข้อมูลที่โมเดลได้เรียนรู้ระหว่างกระบวนการฝึกฝน มันเหมือนอาณาจักรที่ซ่อนอยู่ภายในเครือข่ายประสาทของโมเดล ซึ่งเป็นที่เก็บรูปแบบ ความสัมพันธ์ และการแทนค่าทั้งหมดของภาษา

ลองจินตนาการว่าคุณกำลังสำรวจดินแดนอันกว้างใหญ่ที่ยังไม่มีใครเคยไปถึง เต็มไปด้วยจุดเชื่อมต่อนับไม่ถ้วนที่เชื่อมโยงถึงกัน แต่ละจุดแทนชิ้นส่วนของข้อมูล แนวคิด หรือความสัมพันธ์ที่โมเดลได้เรียนรู้ ขณะที่คุณนำทางผ่านพื้นที่นี้ คุณจะพบว่าบางจุดอยู่ใกล้กันมากกว่า ซึ่งบ่งบอกถึงการเชื่อมต่อที่แข็งแกร่งหรือความคล้ายคลึงกัน ในขณะที่จุดอื่นๆ อยู่ห่างกันออกไป แสดงถึงความสัมพันธ์ที่อ่อนแอหรือห่างไกลกว่า

ความท้าทายของปริภูมิแฝงคือความซับซ้อนและมิติที่มากมายของมัน ลองนึกถึงว่ามันมีขนาดใหญ่มหึมาเหมือนจักรวาลทางกายภาพของเรา พร้อมด้วยกลุ่มดาราจักรและระยะทางอันกว้างใหญ่ของห้วงอวกาศว่างเปล่าที่อยู่ระหว่างกัน

เนื่องจากมีมิติหลายพันมิติ ปริภูมิแฝงจึงไม่สามารถสังเกตหรือตีความได้โดยตรงโดยมนุษย์ มันเป็นการแสดงแทนที่เป็นนามธรรมที่โมเดลใช้ภายในเพื่อประมวลผลและสร้างภาษา เมื่อคุณป้อนคำสั่งเริ่มต้นให้กับโมเดล มันจะทำการจับคู่คำสั่งนั้นไปยังตำแหน่งเฉพาะภายในปริภูมิแฝง จากนั้นโมเดลจะใช้ข้อมูลและการเชื่อมโยงโดยรอบในพื้นที่นั้นเพื่อสร้างการตอบสนอง

สิ่งสำคัญคือ โมเดลได้เรียนรู้ข้อมูลมหาศาลจากข้อมูลการฝึกฝน และไม่ใช่ทุกอย่างที่เกี่ยวข้องหรือถูกต้องสำหรับงานที่กำหนด นั่น คือ เหตุผล ที่ การ จำกัด เส้น ทาง จึง มี ความ สำคัญ มาก การ ให้ คำ แนะนำ ที่ ชัดเจน ตัวอย่าง และ บริบท ใน คำ สั่ง ของ คุณ เป็นการชี้นำโมเดลให้มุ่งเน้นไปที่บริเวณเฉพาะภายในปริภูมิแฝงที่เกี่ยวข้องมากที่สุดกับผลลัพธ์ที่คุณต้องการ

อีก วิธี หนึ่ง ใน การ มอง มัน คือ เหมือน กับ การ ใช้ ไฟฉาย ใน พิพิธภัณฑ์ ที่ มืด สนิท ถ้าคุณเคยไปเยี่ยมชมพิพิธภัณฑ์ลูฟวร์หรือพิพิธภัณฑ์ศิลปะเมโทรโพลิทัน นั่นคือขนาดที่ผมกำลังพูดถึง ปริภูมิแฝงคือพิพิธภัณฑ์ที่เต็มไปด้วยวัตถุและรายละเอียดนับไม่ถ้วน คำสั่งของคุณคือไฟฉาย ที่ส่องสว่างพื้นที่เฉพาะและดึงดูดความสนใจของโมเดลไปยังข้อมูลที่สำคัญที่สุด หากไม่มีการชี้นำนั้น โมเดลอาจจะเดินเตร่ไปมาในปริภูมิแฝง เก็บข้อมูลที่ไม่เกี่ยวข้องหรือขัดแย้งกันระหว่างทาง

ขณะที่คุณทำงานกับโมเดลภาษา และสร้างคำสั่งของคุณ ให้คำนึงถึงแนวคิดของปริภูมิแฝงไว้เสมอ เป้าหมายของคุณคือการนำทางภูมิทัศน์แห่งความรู้อันกว้างใหญ่นี้อย่างมีประสิทธิภาพ ชี้นำโมเดลไปสู่ข้อมูลที่เกี่ยวข้องและถูกต้องที่สุดสำหรับงานของคุณ ด้วยการจำกัดเส้นทางและให้คำแนะนำที่ชัดเจน คุณสามารถปลดล็อกศักยภาพเต็มรูปแบบของปริภูมิแฝงของโมเดลและสร้างการตอบสนองที่มีคุณภาพสูงและสอดคล้องกัน

แม้ว่าคำอธิบายก่อนหน้านี้เกี่ยวกับโมเดลภาษาและปริภูมิแฝงที่พวกมันนำทางอาจดูเหมือนเวทมนตร์หรือเป็นนามธรรมไปบ้าง แต่สำคัญที่ต้องเข้าใจว่าคำสั่งไม่ใช่คาถาหรือบทสวด วิธีการทำงานของโมเดลภาษานั้นตั้งอยู่บนหลักการของพีชคณิตเชิงเส้นและทฤษฎีความน่าจะเป็น

ในแก่นแท้แล้ว โมเดลภาษาคือโมเดลความน่าจะเป็นของข้อความ คล้ายกับที่เส้นโค้งระฆังเป็นแบบจำลองทางสถิติของข้อมูล พวกมันถูกฝึกฝนผ่านกระบวนการที่เรียกว่าการสร้างแบบจำลองการถดถอยอัตโนมัติ ซึ่งโมเดลเรียนรู้ที่จะทำนายความน่าจะเป็นของคำถัดไปในลำดับจากคำที่มาก่อนหน้า ระหว่างการฝึกฝน โมเดลเริ่มต้นด้วยค่าน้ำหนักแบบสุ่มและค่อยๆ ปรับแต่งเพื่อกำหนดความน่าจะเป็นที่สูงขึ้นให้กับข้อความที่คล้ายกับตัวอย่างจากโลกจริงที่ใช้ในการฝึกฝน

อย่างไรก็ตาม การคิดถึงโมเดลภาษาว่าเป็นเพียงแบบจำลองทางสถิติอย่างง่าย เช่น การถดถอยเชิงเส้น ไม่ได้ให้ความเข้าใจที่ดีที่สุดในการทำความเข้าใจพฤติกรรมของพวกมัน การเปรียบเทียบที่เหมาะสมกว่าคือการคิดถึงพวกมันว่าเป็นโปรแกรมความน่าจะเป็น ซึ่งเป็นโมเดลที่อนุญาตให้มีการจัดการตัวแปรสุ่มและสามารถแสดงความสัมพันธ์ทางสถิติที่ซับซ้อนได้

โปรแกรมความน่าจะเป็นสามารถแสดงได้ด้วยโมเดลกราฟ ซึ่ง ให้ วิธี การ มอง เห็น ภาพ เพื่อ เข้าใจ การ พึ่งพา และ ความ สัมพันธ์ ระหว่าง ตัวแปร ใน โมเดล มุม มอง นี้ สามารถ ให้ ข้อมูล เชิง ลึก ที่ มี ค่า เกี่ยว กับ การ ทำงาน ของ โมเดล การ สร้าง ข้อความ ที่ ซับ ซ้อน อย่าง GPT-4 และ Claude

ในบทความ “Language Model Cascades” โดย Dohan และคณะ ผู้เขียนได้ลงลึกในรายละเอียดเกี่ยวกับวิธีการที่โปรแกรมความน่าจะเป็นสามารถนำมาใช้กับโมเดลภาษา พวกเขาแสดงให้เห็นว่ากรอบแนวคิดนี้สามารถใช้เพื่อทำความเข้าใจพฤติกรรมของโมเดลเหล่านี้และชี้นำการพัฒนากลยุทธ์การให้คำสั่งที่มีประสิทธิภาพมากขึ้น

หนึ่งในข้อมูลเชิงลึกที่สำคัญจากมุมมองความน่าจะเป็นนี้คือ โมเดลภาษาสร้างประตูสู่จักรวาลคู่ขนานที่เอกสารที่ต้องการมีอยู่จริง โมเดลกำหนดค่าน้ำหนักให้กับเอกสารที่เป็นไปได้ทั้งหมดตามความน่าจะเป็นของพวกมัน ทำให้พื้นที่ของความเป็นไปได้แคบลงเพื่อมุ่งเน้นไปที่สิ่งที่เกี่ยวข้องมากที่สุด

สิ่งนี้นำเรากลับมาสู่แก่นหลักของ “การจำกัดเส้นทาง” เป้าหมายหลักของการให้คำสั่งคือการกำหนดเงื่อนไขให้กับโมเดลความน่าจะเป็นในลักษณะที่มุ่งเน้นมวลของการทำนายของมัน เจาะจงไปที่ข้อมูลหรือพฤติกรรมที่เราต้องการดึงออกมา ด้วยการให้คำสั่งที่ออกแบบมาอย่างรอบคอบ เราสามารถชี้นำโมเดลให้นำทางปริภูมิแฝงได้อย่างมีประสิทธิภาพมากขึ้นและสร้างผลลัพธ์ที่เกี่ยวข้องและสอดคล้องกันมากขึ้น

อย่างไรก็ตาม สำคัญที่ต้องจำไว้ว่าโมเดลภาษาถูกจำกัดด้วยข้อมูลที่ใช้ในการฝึกฝน แม้ว่ามันสามารถสร้างข้อความที่คล้ายกับเอกสารที่มีอยู่หรือรวมแนวคิดในวิธีใหม่ๆ แต่มันไม่สามารถสร้างข้อมูลใหม่ทั้งหมดขึ้นมาจากความว่างเปล่าได้ ตัวอย่างเช่น เราไม่สามารถคาดหวังให้โมเดลให้วิธีรักษาโรคมะเร็งได้ถ้าวิธีรักษาดังกล่าวยังไม่ถูกค้นพบและบันทึกไว้ในข้อมูลการฝึกฝนของมัน

แต่จุดแข็งของโมเดลอยู่ที่ความสามารถในการค้นหาและสังเคราะห์ข้อมูลที่คล้ายคลึงกับคำสั่งที่เราป้อนให้ ด้วยความเข้าใจในลักษณะความน่าจะเป็นของโมเดลเหล่านี้ และวิธีการใช้คำสั่งเพื่อกำหนดเงื่อนไขผลลัพธ์ เราจึงสามารถใช้ประโยชน์จากความสามารถของพวกมันในการสร้างข้อมูลเชิงลึกและเนื้อหาที่มีคุณค่าได้อย่างมีประสิทธิภาพมากขึ้น

ลองพิจารณาคำสั่งด้านล่างนี้ ในคำสั่งแรก คำว่า “Mercury” เพียงคำเดียวอาจหมายถึงดาวเคราะห์ ธาตุ หรือเทพเจ้าโรมัน แต่ความน่าจะเป็นที่สูงที่สุดคือดาวเคราะห์ จริงๆ แล้ว GPT-4 ให้คำตอบที่ยาวโดยเริ่มด้วย *ดาวพุธเป็นดาวเคราะห์ที่เล็กที่สุดและอยู่*

ใกล้ดวงอาทิตย์ที่สุดในระบบสุริยะ... คำสั่งที่สองกล่าวถึงธาตุเคมีโดยเฉพาะ ส่วนคำสั่งที่สามกล่าวถึงตัวละครในตำนานโรมัน ที่มีชื่อเสียงในด้านความเร็วและบทบาทการเป็นผู้ส่งสารของเทพเจ้า

```
# Prompt 1
Tell me about: Mercury

# Prompt 2
Tell me about: Mercury element

# Prompt 3
Tell me about: Mercury messenger of the gods
```

เพียงแค่เพิ่มคำไม่กี่คำ เราก็สามารถเปลี่ยนวิธีที่ AI ตอบสนองได้อย่างสิ้นเชิง ดังที่คุณจะได้เรียนรู้ในภายหลังของหนังสือเล่มนี้ เทคนิคการวิศวกรรมคำสั่งขั้นสูง เช่น การสร้างคำสั่งแบบ n-shot การกำหนดรูปแบบอินพุต/เอาต์พุต และ Chain of Thought ล้วนเป็นเพียงวิธีที่ชาญฉลาดในการปรับเงื่อนไขผลลัพธ์ของโมเดล

ดังนั้น ท้ายที่สุดแล้ว ศิลปะของการวิศวกรรมคำสั่ง คือการเข้าใจวิธีการนำทางในภูมิทัศน์ความน่าจะเป็นอันกว้างใหญ่ของความรู้ในโมเดลภาษา เพื่อจำกัดเส้นทางไปสู่ข้อมูลหรือพฤติกรรมเฉพาะที่เราต้องการ

สำหรับผู้อ่านที่มีความเข้าใจทางคณิตศาสตร์ขั้นสูงเป็นอย่างดี การทำความเข้าใจโมเดลเหล่านี้บนพื้นฐานของทฤษฎีความน่าจะเป็นและพีชคณิตเชิงเส้นจะช่วยคุณได้แน่นอน! สำหรับคนที่เหลือที่ต้องการพัฒนากลยุทธ์ที่มีประสิทธิภาพในการดึงผลลัพธ์ที่ต้องการ เรามาใช้วิธีการที่เข้าใจง่ายกว่ากัน

วิธีการ “จำกัด” เส้นทาง

เพื่อจัดการกับความท้าทายของการมีความรู้มากเกินไป เราใช้เทคนิคที่ช่วยชี้นำกระบวนการสร้างของโมเดลภาษาและทำให้มันโฟกัสกับข้อมูลที่เกี่ยวข้องและแม่นยำที่สุด

นี่คือเทคนิคที่สำคัญที่สุด เรียงตามลำดับที่แนะนำ นั่นคือ คุณควรลองใช้การวิศวกรรมคำสั่งก่อน จากนั้นจึงใช้ RAG และสุดท้าย ถ้าจำเป็น จึงค่อยใช้การปรับแต่งแบบละเอียด

การวิศวกรรมคำสั่ง วิธีการพื้นฐานที่สุดคือการสร้างคำสั่งที่รวมคำแนะนำ ข้อจำกัด หรือตัวอย่างเฉพาะเพื่อชี้นำการสร้างการตอบสนองของโมเดล บทนี้ครอบคลุมพื้นฐานของการวิศวกรรมคำสั่งในส่วนถัดไป และเราจะครอบคลุมรูปแบบการวิศวกรรมคำสั่งเฉพาะหลายรูปแบบในส่วนที่ 2 ของหนังสือ รูปแบบเหล่านั้นรวมถึง การกลั่นกรองคำสั่ง ซึ่งเป็นเทคนิคที่เน้นการปรับปรุงและเพิ่มประสิทธิภาพคำสั่งเพื่อดึงข้อมูลที่ AI พิจารณาว่าเกี่ยวข้องและกระชับที่สุด

การเพิ่มพูนบริบท การดึงข้อมูลที่เกี่ยวข้องจากฐานความรู้หรือเอกสารภายนอกแบบไดนามิกเพื่อให้โมเดลมีบริบทที่เฉพาะเจาะจงในขณะที่ได้รับคำสั่ง เทคนิคการเพิ่มพูนบริบทที่เป็นที่นิยมรวมถึง การสร้างเนื้อหาที่เพิ่มพูนด้วยการค้นคืน (RAG) โมเดล “ออนไลน์” อย่างที่ให้บริการโดย Perplexity สามารถเพิ่มพูนบริบทด้วยผลการค้นหาอินเทอร์เน็ตแบบเรียลไทม์

แม้จะมีพลังมาก LLM ไม่ได้ถูกฝึกฝนบนชุดข้อมูลเฉพาะของคุณ ซึ่งอาจเป็นข้อมูลส่วนตัวหรือเฉพาะกับปัญหาที่คุณกำลังพยายามแก้ไข เทคนิคการเพิ่มพูนบริบทช่วยให้คุณสามารถให้ LLM เข้าถึงข้อมูลที่อยู่หลัง API ในฐานข้อมูล SQL หรือติดอยู่ในไฟล์ PDF และสไลด์

การปรับแต่งแบบละเอียดหรือการปรับตัวตามโดเมน การฝึกฝนโมเดลบนชุดข้อมูลเฉพาะโดเมนเพื่อให้ความรู้และความสามารถในการสร้างเนื้อหาเฉพาะทางสำหรับงานหรือสาขาเฉพาะ

การลดค่าอุณหภูมิ

อุณหภูมิเป็น *ไฮเปอร์พารามิเตอร์* ที่ใช้ในโมเดลภาษาแบบ transformer เพื่อควบคุมความสุ่มและความคิดสร้างสรรค์ของข้อความที่สร้างขึ้น มีค่าระหว่าง 0 ถึง 1 โดยค่าที่ต่ำกว่าจะทำให้ผลลัพธ์มีความเฉพาะเจาะจงและแน่นอนมากขึ้น ในขณะที่ค่าที่สูงกว่าจะทำให้มีความหลากหลายและคาดเดาได้ยากขึ้น

เมื่อตั้งค่าอุณหภูมิเป็น 1 โมเดลภาษาจะสร้างข้อความตามการกระจายความน่าจะเป็นทั้งหมดของโทเค็นถัดไป ซึ่งช่วยให้เกิดการตอบสนองที่สร้างสรรค์และหลากหลายมากขึ้น อย่างไรก็ตาม นี่อาจทำให้โมเดลสร้างข้อความที่เกี่ยวข้องหรือสอดคล้องน้อยลง

ในทางกลับกัน เมื่อตั้งค่าอุณหภูมิเป็น 0 โมเดลภาษาจะเลือกโทเค็นที่มีความน่าจะเป็นสูงสุดเสมอ ซึ่งเป็นการ “จำกัดเส้นทาง” อย่างมีประสิทธิภาพ คอมโพเนนต์ AI เกือบทั้งหมดของผมใช้ค่าอุณหภูมิที่ 0 หรือใกล้เคียง เนื่องจากให้การตอบสนองที่เฉพาะเจาะจงและคาดเดาได้มากขึ้น มันมีประโยชน์อย่างยิ่งเมื่อคุณต้องการให้โมเดล *ทำตามคำแนะนำ* ให้ความสนใจกับฟังก์ชันที่ได้รับ หรือเพียงแค่ต้องการการตอบสนองที่แม่นยำและเกี่ยวข้องมากขึ้นกว่าที่คุณได้รับ

ตัวอย่างเช่น ถ้าคุณกำลังสร้างแชทบอทที่ต้องให้ข้อมูลที่เป็นข้อเท็จจริง คุณอาจต้องการตั้งค่าอุณหภูมิให้ต่ำลงเพื่อให้แน่ใจว่าการตอบสนองมีความแม่นยำและตรงประเด็นมากขึ้น ในทางกลับกัน ถ้าคุณกำลังสร้างผู้ช่วยการเขียนเชิงสร้างสรรค์ คุณอาจต้องการตั้งค่าอุณหภูมิให้สูงขึ้นเพื่อส่งเสริมผลลัพธ์ที่หลากหลายและสร้างสรรค์มากขึ้น

ไฮเปอร์พารามิเตอร์: ปุ่มหมุนและตัวควบคุมของการอนุมาน

เมื่อคุณทำงานกับโมเดลภาษา คุณจะพบคำว่า “ไฮเปอร์พารามิเตอร์” บ่อยครั้ง ในบริบทของการอนุมาน (นั่นคือ เมื่อคุณใช้โมเดลเพื่อสร้างการตอบสนอง) ไฮเปอร์พารามิเตอร์เปรียบเสมือนปุ่มหมุนและตัวควบคุมที่คุณสามารถปรับแต่งเพื่อควบคุมพฤติกรรมและผลลัพธ์ของโมเดล

ลองนึกถึงมันเหมือนการปรับการตั้งค่าบนเครื่องจักรที่ซับซ้อน เช่นเดียวกับที่คุณอาจหมุนปุ่มเพื่อควบคุมอุณหภูมิหรือสลับสวิตช์เพื่อเปลี่ยนโหมดการทำงาน ไฮเปอร์พารามิเตอร์ช่วยให้คุณปรับแต่งวิธีที่โมเดลภาษาประมวลผลและสร้างข้อความได้อย่างละเอียด

ไฮเปอร์พารามิเตอร์ที่พบบ่อยในระหว่างการอนุมานมีดังนี้:

- **อุณหภูมิ (Temperature)**: ดังที่กล่าวไปแล้ว พารามิเตอร์นี้ควบคุมความสุ่มและความสร้างสรรค์ของข้อความที่สร้างขึ้น อุณหภูมิที่สูงขึ้นจะนำไปสู่ผลลัพธ์ที่หลากหลายและคาดเดาได้ยากขึ้น ในขณะที่อุณหภูมิที่ต่ำลงจะให้ผลการตอบสนองที่มุ่งเน้นและแน่นอนมากขึ้น

- **การสุ่มแบบ Top-p (nucleus):** พารามิเตอร์นี้ควบคุมการเลือกชุดโทเค็นที่เล็กที่สุดที่มีความน่าจะเป็นสะสมเกินค่าขีดจำกัดหนึ่ง (p) ช่วยให้ได้ผลลัพธ์ที่หลากหลายในขณะที่ยังคงรักษาความสอดคล้องไว้

- **การ สุ่ม แบบ Top-k:** เทคนิค นี้ เลือก โท เค็น ถัด ไป ที่ มี ความ เป็น ไป ได้ มาก ที่สุด *k* ตัว และกระจายมวลความน่าจะเป็นระหว่างโทเค็นเหล่านั้น ช่วย ป้องกัน ไม่ ให้ โมเดล สร้าง โท เค็ นที่ มี ความ น่า จะ เป็น ต่ำหรือไม่เกี่ยวข้อง

- **บทปรับความถี่ (Frequency)** และ **บทปรับการปรากฏ (Presence penalties)**: พารามิเตอร์เหล่านี้ลงโทษโมเดลสำหรับการใช้คำหรือวลีเดิมซ้ำๆ บ่อยเกินไป (บทปรับความถี่) หรือการสร้างคำที่ไม่มีอยู่ในพรอมต์นำเข้า (บทปรับการปรากฏ) การปรับค่าเหล่านี้ช่วยให้โมเดลสร้างผลลัพธ์ที่หลากหลายและตรงประเด็นมากขึ้น

- **ความยาวสูงสุด (Maximum length)**: ไฮเปอร์พารามิเตอร์นี้กำหนดขีดจำกัดบนของจำนวนโทเค็น (คำหรือส่วนของคำ) ที่โมเดลสามารถสร้างได้ในการตอบสนองครั้งเดียว ช่วยควบคุมความเยิ่นเย้อและความกระชับของข้อความที่สร้างขึ้น

เมื่อคุณทดลองใช้การตั้งค่าไฮเปอร์พารามิเตอร์ที่แตกต่างกัน คุณจะพบว่าแม้การปรับเปลี่ยนเพียงเล็กน้อยก็สามารถส่งผลกระทบอย่างมีนัยสำคัญต่อผลลัพธ์ของโมเดล เหมือนกับการปรุงอาหาร - เพิ่มเกลือนิดหน่อยหรือเวลาในการทำอาหารที่นานขึ้นเล็กน้อยก็สามารถทำให้อาหารจานสุดท้ายแตกต่างกันได้

สิ่งสำคัญคือการเข้าใจว่าไฮเปอร์พารามิเตอร์แต่ละตัวส่งผลต่อพฤติกรรมของโมเดลอย่างไร และการหาความสมดุลที่เหมาะสมสำหรับงานเฉพาะของคุณ อย่ากลัวที่จะทดลองใช้การตั้งค่าต่างๆ และดูว่าพวกมันส่งผลต่อข้อความที่สร้างขึ้นอย่างไร เมื่อเวลาผ่านไป คุณจะพัฒนาความเข้าใจว่าควรปรับไฮเปอร์พารามิเตอร์ใดและอย่างไรเพื่อให้ได้ผลลัพธ์ที่ต้องการ

ด้วยการผสมผสานการใช้พารามิเตอร์เหล่านี้กับการออกแบบพรอมต์ การสร้างที่เสริมด้วยการค้นคืน และการปรับแต่งละเอียด คุณสามารถจำกัดเส้นทางและนำทางโมเดลภาษาให้สร้างการตอบสนองที่แม่นยำ ตรงประเด็น และมีคุณค่ามากขึ้นสำหรับกรณีการใช้งานเฉพาะของคุณ

โมเดลดิบเทียบกับโมเดลที่ผ่านการปรับแต่งด้วยคำสั่ง

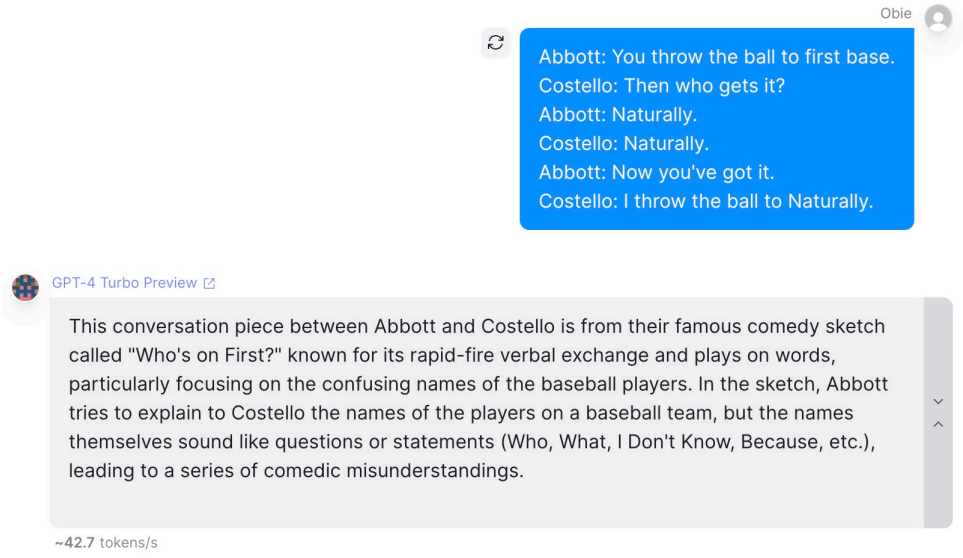
โมเดลดิบคือเวอร์ชันที่ยังไม่ได้ขัดเกลา ยังไม่ได้ฝึกฝนของ LLM ลองนึกภาพว่าเป็นเหมือนผืนผ้าใบว่างเปล่า ที่ยังไม่ได้รับอิทธิพลจากการฝึกฝนเฉพาะในการทำความเข้าใจหรือปฏิบัติตามคำสั่ง พวกมันถูกสร้างขึ้นบนข้อมูลจำนวนมหาศาลที่ใช้ในการฝึกฝนครั้งแรก สามารถสร้างผลลัพธ์ได้หลากหลาย อย่างไรก็ตาม หากไม่มีการปรับแต่งเพิ่มเติมด้วยการฝึกฝนตามคำสั่ง การตอบสนองของพวกมันอาจคาดเดาได้ยากและต้องใช้พรอมต์ที่ออกแบบอย่างละเอียดมากขึ้นเพื่อนำทางไปสู่ผลลัพธ์ที่ต้องการ การทำงานกับโมเดลดิบเหมือนกับการพยายามสื่อสารกับอัจฉริยะที่มีความรู้มหาศาลแต่ขาดความเข้าใจโดยสิ้นเชิงเกี่ยวกับสิ่งที่คุณกำลังถาม เว้นแต่คุณจะให้คำแนะนำที่ชัดเจนมากๆ พวกมันมักจะเหมือนนกแก้วที่เพียงแค่พูดซ้ำสิ่งที่ได้ยินคุณพูด

ในทางตรงกันข้าม โมเดลที่ผ่านการปรับแต่งด้วยคำสั่งได้ผ่านการฝึกฝนหลายรอบที่ออกแบบมาเฉพาะเพื่อให้เข้าใจและปฏิบัติตามคำสั่ง GPT-4, Claude 3 และโมเดล LLM ยอดนิยมอื่นๆ อีกมากมายล้วนผ่านการปรับแต่งด้วยคำสั่งอย่างมาก การฝึกฝนนี้เกี่ยวข้องกับการป้อนตัวอย่างคำสั่งพร้อมกับผลลัพธ์ที่ต้องการให้กับโมเดล ซึ่งเป็นการสอนโมเดลให้ตีความและดำเนินการตามคำสั่งที่หลากหลาย ผลลัพธ์คือ โมเดลที่ผ่านการปรับแต่งด้วยคำสั่งสามารถเข้าใจเจตนาเบื้องหลังพรอมต์และสร้างการตอบสนองที่สอดคล้องกับความคาดหวังของผู้ใช้ได้ดีขึ้น ทำให้ใช้งานง่ายและทำงานด้วยได้ง่ายขึ้น โดยเฉพาะสำหรับผู้ที่อาจไม่มีเวลาหรือความเชี่ยวชาญในการออกแบบพรอมต์อย่างละเอียด

โมเดลดิบ: ผืนผ้าใบที่ไม่ผ่านการกรอง

โมเดลดิบ เช่น Llama 2-70B หรือ Yi-34B เปิดโอกาสให้เข้าถึงความสามารถของโมเดลแบบไม่ผ่านการกรองมากกว่าที่คุณอาจคุ้นเคยหากคุณเคยทดลองใช้ LLM ยอดนิยมอย่าง GPT-4 โมเดลเหล่านี้ไม่ได้ผ่านการปรับแต่งล่วงหน้าให้ปฏิบัติตามคำสั่งเฉพาะ จึงเป็นเหมือนผืนผ้าใบว่างเปล่าที่คุณสามารถจัดการผลลัพธ์ของโมเดลได้โดยตรงผ่านการออกแบบพรอมต์อย่างระมัดระวัง แนวทางนี้ต้องอาศัยความเข้าใจอย่างลึกซึ้งในการสร้างพรอมต์ที่นำทาง AI ไปในทิศทางที่ต้องการโดยไม่ต้องสั่งการอย่างชัดเจน เหมือนกับการมีการเข้าถึงโดยตรงไปยังชั้น “ดิบ” ของ AI พื้นฐาน โดยไม่มีชั้นตัวกลางใดๆ ที่ตีความหรือนำทางการตอบสนองของโมเดล (จึงเรียกว่าโมเดลดิบ)

ความท้าทายของโมเดลดิบอยู่ที่แนวโน้มที่จะตกอยู่ในรูปแบบการซ้ำซ้อนหรือสร้างผลลัพธ์แบบสุ่ม อย่างไรก็ตาม ด้วยการออกแบบพรอมต์อย่างพิถีพิถันและการปรับแต่งพารามิเตอร์ต่างๆ เช่น บทลงโทษการซ้ำซ้อน โมเดลดิบสามารถถูกชักนำให้สร้างเนื้อหาที่มีความคิดสร้างสรรค์และเป็นเอกลักษณ์ได้ กระบวนการนี้มาพร้อมกับการแลกเปลี่ยน; ในขณะที่โมเดลดิบมอบความยืดหยุ่นที่ไม่มีใครเทียบได้สำหรับนวัตกรรม แต่ก็ต้องการระดับความเชี่ยวชาญที่สูงขึ้น



แผนภูมิ 3. เพื่อเปรียบเทียบ นี่คือพรอมต์ที่คลุมเครือเดียวกันที่ป้อนให้กับ GPT-4

โมเดลที่ผ่านการปรับแต่งด้วยคำสั่ง: ประสบการณ์ที่ได้รับการแนะนำ

โมเดลที่ผ่านการปรับแต่งด้วยคำสั่ง ถูกออกแบบมาให้เข้าใจและทำตามคำสั่งเฉพาะ ทำให้ใช้งานง่ายและเข้าถึงได้สำหรับการประยุกต์ใช้ที่หลากหลายมากขึ้น พวกมันเข้าใจกลไกของ *การสนทนา* และรู้ว่าควรหยุดการสร้างเมื่อถึง *จุดจบของการตอบโต้* สำหรับนักพัฒนาจำนวนมาก โดยเฉพาะผู้ที่ทำงานกับแอปพลิเคชันที่ไม่ซับซ้อน โมเดลที่ผ่านการปรับแต่งด้วยคำสั่งมอบทางออกที่สะดวกและมีประสิทธิภาพ

กระบวนการปรับแต่งด้วยคำสั่งเกี่ยวข้องกับการฝึกฝนโมเดลบนคลังข้อมูลขนาดใหญ่ของพรอมต์คำสั่งและการตอบสนองที่สร้างโดยมนุษย์ ตัวอย่างที่โดดเด่นคือชุดข้อมูลโอเพนซอร์ส databricks-dolly-15k dataset ซึ่งประกอบด้วยคู่พรอมต์/การตอบสนองมากกว่า 15,000 คู่ที่สร้างโดยพนักงาน Databricks ที่คุณสามารถตรวจสอบได้ด้วยตัวเอง ชุดข้อมูลครอบคลุมหมวดหมู่คำสั่งแปดประเภท รวมถึงการเขียนเชิงสร้างสรรค์ การตอบคำถามแบบปิดและเปิด การสรุปความ การดึงข้อมูล การจำแนกประเภท และการระดมความคิด

ในระหว่างกระบวนการสร้างข้อมูล ผู้มีส่วนร่วมได้รับแนวทางในการสร้างพรอมต์และการตอบสนองสำหรับแต่ละหมวดหมู่ ตัวอย่างเช่น สำหรับงานเขียนเชิงสร้างสรรค์ พวกเขาได้รับคำแนะนำให้ระบุข้อจำกัด คำสั่ง หรือข้อกำหนดเฉพาะเพื่อแนะนำผลลัพธ์ของโมเดล สำหรับการตอบคำถามแบบปิด พวกเขาถูกขอให้เขียนคำถามที่ต้องการคำตอบที่ถูกต้องตามข้อเท็จจริงบนพื้น

ฐานของบทความจาก Wikipedia ที่กำหนดให้

ชุดข้อมูลที่ได้ทำหน้าที่เป็นทรัพยากรที่มีค่าสำหรับการปรับแต่งละเอียดของโมเดลภาษาขนาดใหญ่ให้แสดงความสามารถในการโต้ตอบและทำตามคำสั่งเหมือนระบบอย่าง ChatGPT โดยการฝึกฝนบนคำสั่งและการตอบสนองที่หลากหลายที่สร้างโดยมนุษย์ โมเดลจะเรียนรู้ที่จะเข้าใจและทำตามคำสั่งเฉพาะ ทำให้มีความสามารถมากขึ้นในการจัดการกับงานที่หลากหลาย

นอก เหนือ จาก การ ปรับ แต่ง ละเอียด โดยตรง พร อมต์ คำ สั่ง ใน ชุด ข้อมูล เช่น databricks-dolly-15k ยังสามารถใช้สำหรับการสร้างข้อมูลสังเคราะห์ โดย การ ส่ง พร อมต์ ที่ สร้าง โดย ผู้ มี ส่วน ร่วม เป็น ตัวอย่าง แบบ few-shot ให้กับโมเดลภาษาเปิดขนาดใหญ่ นักพัฒนาสามารถสร้างคลังคำสั่งที่ใหญ่กว่าในแต่ละหมวดหมู่ แนวทางนี้ที่อธิบายไว้ในเอกสาร Self-Instruct ช่วยให้สามารถสร้างโมเดลที่ทำตามคำสั่งที่แข็งแกร่งขึ้น

นอกจากนี้ คำสั่งและการตอบสนองในชุดข้อมูลเหล่านี้สามารถเพิ่มประสิทธิภาพได้ด้วยเทคนิคต่างๆ เช่น การถอดความ โดยการเรียบเรียงแต่ละพรอมต์หรือคำตอบสั้นๆ ใหม่และเชื่อมโยงข้อความที่ได้กับตัวอย่างที่เป็นค่าจริง นักพัฒนาสามารถแนะนำรูปแบบของการควบคุมที่ช่วยเพิ่มความสามารถของโมเดลในการทำตามคำสั่ง

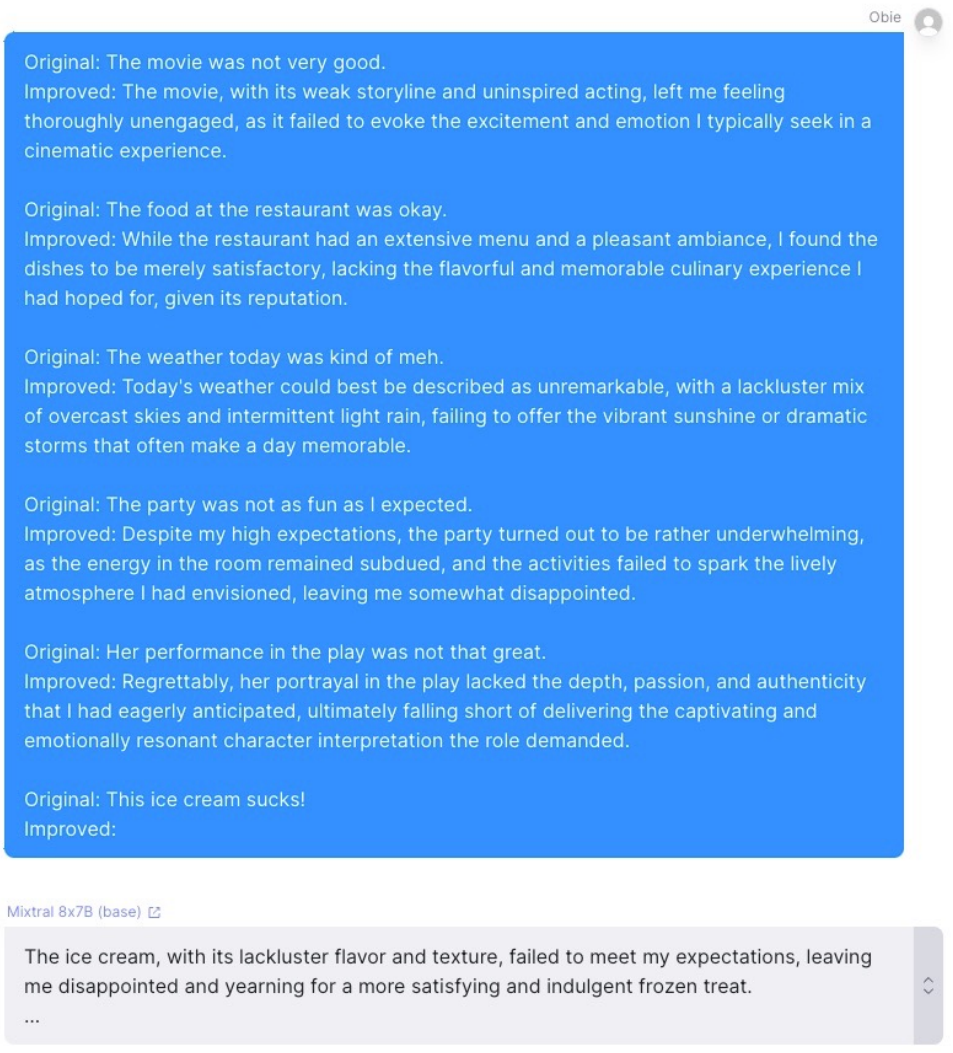
ความสะดวกในการใช้งานที่มาพร้อมกับโมเดลที่ปรับแต่งด้วยคำสั่งนั้นแลกมาด้วยความยืดหยุ่นที่ลดลงบางส่วน โมเดลเหล่านี้มักถูกเซ็นเซอร์อย่างเข้มงวด ซึ่งหมายความว่าอาจไม่สามารถให้อิสระในการสร้างสรรค์ตามที่ต้องการสำหรับงานบางประเภท ผลลัพธ์ของพวกมันได้รับอิทธิพลอย่างมากจากอคติและข้อจำกัดที่มีอยู่ในข้อมูลที่ใช้ในการปรับแต่ง

แม้จะมีข้อจำกัดเหล่านี้ โมเดลที่ปรับแต่งด้วยคำสั่งก็ได้รับความนิยมเพิ่มขึ้นเรื่อยๆ เนื่องจากความเป็นมิตรต่อผู้ใช้และความสามารถในการจัดการงานที่หลากหลายโดยใช้การออกแบบพรอมต์เพียงเล็กน้อย เมื่อมีชุดข้อมูลคำสั่งที่มีคุณภาพสูงมากขึ้น เราคาดว่าจะได้เห็นการพัฒนาประสิทธิภาพและความอเนกประสงค์ของโมเดลเหล่านี้ต่อไป

การเลือกประเภทของโมเดลที่เหมาะสมกับโครงการของคุณ

การตัดสินใจระหว่างโมเดลพื้นฐาน (แบบดิบ) และโมเดลที่ปรับแต่งด้วยคำสั่งนั้นขึ้นอยู่กับความต้องการเฉพาะของโครงการของคุณ สำหรับงานที่ต้องการความคิดสร้างสรรค์และความเป็นต้นฉบับในระดับสูง โมเดลพื้นฐานเป็นเครื่องมือที่ทรงพลังสำหรับนวัตกรรม โมเดลเหล่านี้ช่วยให้นักพัฒนาสามารถสำรวจศักยภาพเต็มรูปแบบของ LLM ผลักดันขอบเขตของสิ่งที่สามารถทำได้ผ่านแอปพลิเคชันที่ขับเคลื่อนด้วย AI แต่ต้องใช้วิธีการที่ต้องลงมือทำมากขึ้นและความเต็มใจที่จะทดลอง ค่าอุณหภูมิและการตั้งค่าอื่นๆ มีผลกระทบมากกว่าในโมเดลพื้นฐานเมื่อเทียบกับคู่แข่งที่ปรับแต่งด้วยคำสั่ง

สิ่งใดก็ตามที่คุณรวมไว้ในพรอมต์ของคุณคือสิ่งที่โมเดลพื้นฐานจะพยายามทำซ้ำ ดังนั้นถ้าหากพรอมต์ของคุณเป็นบันทึกการแชท โมเดลดิบจะพยายามต่อการแชทนั้น ขึ้นอยู่กับขีดจำกัดของจำนวนโทเค็นสูงสุด มันจะไม่เพียงแค่สร้างข้อความถัดไปในการแชทเท่านั้น แต่อาจจะมีการสนทนากับตัวเองทั้งหมด!



แผนภูมิ 4. ตัวอย่างการเขียนประโยคใหม่ด้วย Few-Shot Completion โดย Mixtral 8x7B (base)

ในขณะที่เตรียมตัวอย่างการเขียนประโยคใหม่ข้างต้นโดยผู้ใช้ Reddit phree_radical ผมสามารถได้ผลลัพธ์ที่ใช้งานได้หลังจากการทดลองกับการตั้งค่าพารามิเตอร์มากมาย ในที่สุดก็ได้ข้อสรุปที่: Temperature 0.08, Top P: 0.2, Top K: 1, และ Repetition Penalty: 1.26

การพยายามใช้วิธีการนี้กับโมเดลพื้นฐานในการผลิตจริงอาจจะยุ่งยากเนื่องจากผลกระทบอันทรงพลังของพารามิเตอร์ max_-

tokens ถ้าตั้งค่าสั้นเกินไป ผลลัพธ์จะถูกตัดทอน ถ้าตั้งค่ายาวกว่าที่โมเดลต้องการสำหรับผลลัพธ์ที่ต้องการ มันจะจินตนาการตัวอย่างเพิ่มเติมต่อไปเรื่อยๆ

สรุปคือ เว้นแต่คุณจะต้องการการควบคุมอย่างเต็มที่และการไม่มีการเซ็นเซอร์จริงๆ โมเดลที่ปรับแต่งด้วยคำสั่งสามารถทำให้กระบวนการพัฒนาของคุณง่ายขึ้นอย่างมาก เพื่อย้ำจุดนี้ นี่คือการตอบสนองของ Mixtral 8x7B ต่อพรอมต์เดียวกัน แต่คราวนี้ในเวอร์ชันที่ปรับแต่งด้วยคำสั่ง:

> ผมต้องขออภัยที่ต้องแจ้งให้ทราบว่าไอศกรีมนี้ไม่เป็นไปตามความคาดหวัง เนื่องจากขาดเนื้อสัมผัสที่เนียนนุ่มและรสชาติอันน่าเพลิดเพลินที่ผมมักจะพบในขนมหวานคุณภาพสูง ผมหวังว่าจะได้รับประสบการณ์ที่น่าพึงพอใจและสนุกสนานมากกว่านี้

ที่น่าสังเกตคือ ผมสามารถตั้งค่า max tokens ไว้ที่ 500 และโมเดลหยุดทำงานที่จุดสิ้นสุดของเอาต์พุตที่ต้องการได้อย่างน่าเชื่อถือโดยไม่สร้างตัวอย่างเพิ่มเติมที่ไม่เกี่ยวข้อง

การออกแบบพรอมต์

เมื่อคุณเริ่มนำ AI มาใช้ในโครงการต่างๆ คุณจะพบว่าหนึ่งในทักษะที่สำคัญที่สุดที่คุณต้องฝึกฝนคือศิลปะของการออกแบบพรอมต์ แต่การออกแบบพรอมต์คืออะไรกันแน่ และทำไมจึงสำคัญมาก?

แก่นแท้ของการออกแบบพรอมต์คือกระบวนการออกแบบและสร้างพรอมต์อินพุตที่คุณป้อนให้กับโมเดลภาษาเพื่อชี้นำเอาต์พุต เป็นเรื่องของการเข้าใจวิธีการสื่อสารกับ AI อย่างมีประสิทธิภาพ โดยใช้การผสมผสานระหว่างคำแนะนำ ตัวอย่าง และบริบทเพื่อชี้นำโมเดลให้สร้างการตอบสนองที่ต้องการ

ลองนึกถึงเหมือนกับการสนทนากับเพื่อนที่ฉลาดมากแต่ค่อนข้างเป็นคนที่ตีความตรงตัว เพื่อให้ได้ประโยชน์สูงสุดจากการมีปฏิสัมพันธ์ คุณต้องชัดเจน เฉพาะเจาะจง และให้บริบทที่เพียงพอเพื่อให้แน่ใจว่าเพื่อนของคุณเข้าใจสิ่งที่คุณกำลังขออย่างแม่นยำ นั่นคือจุดที่การออกแบบพรอมต์เข้ามามีบทบาท และแม้ว่าในตอนแรกอาจดูเหมือนง่าย เชื่อผมเถอะว่าต้องใช้การฝึกฝนอย่างมากเพื่อให้เชี่ยวชาญ

องค์ประกอบพื้นฐานของพรอมต์ที่มีประสิทธิภาพ

ในการเริ่มออกแบบพรอมต์ที่มีประสิทธิภาพ อันดับแรกคุณต้องเข้าใจองค์ประกอบสำคัญที่ประกอบขึ้นเป็นอินพุตที่ออกแบบมาอย่างดี นี่คือองค์ประกอบพื้นฐานที่สำคัญ:

1. **คำแนะนำ**: คำแนะนำที่ชัดเจนและกระชับที่บอกโมเดลว่าคุณต้องการให้ทำอะไร อาจเป็นอะไรก็ได้ตั้งแต่ “สรุปบทความต่อไปนี้” ไปจนถึง “สร้างบทกวีเกี่ยวกับพระอาทิตย์ตก” หรือ “แปลงคำขอเปลี่ยนแปลงโครงการนี้ให้เป็นออบเจ็กต์ JSON”
2. **บริบท**: ข้อมูลที่เกี่ยวข้องที่ช่วยให้โมเดลเข้าใจพื้นหลังและขอบเขตของงาน อาจรวมถึงรายละเอียดเกี่ยวกับผู้ชมเป้าหมาย โทนและสไตล์ที่ต้องการ หรือข้อจำกัดหรือข้อกำหนดเฉพาะสำหรับเอาต์พุต เช่น JSON Schema ที่ต้องปฏิบัติตาม
3. **ตัวอย่าง**: ตัวอย่างที่เป็นรูปธรรมที่แสดงให้เห็นประเภทของเอาต์พุตที่คุณต้องการ การให้ตัวอย่างที่เลือกมาอย่างดีสองสามตัวอย่างสามารถช่วยให้โมเดลเรียนรู้รูปแบบและลักษณะของการตอบสนองที่ต้องการ
4. **การจัดรูปแบบอินพุต**: การขึ้นบรรทัดใหม่และการจัดรูปแบบ markdown ช่วยให้โครงสร้างแก่พรอมต์ของเรา การแยกพรอมต์เป็นย่อหน้าช่วยให้เราจัดกลุ่มคำแนะนำที่เกี่ยวข้องกัน เพื่อให้ทั้งมนุษย์และ AI เข้าใจได้ง่ายขึ้น สัญลักษณ์แสดงหัวข้อย่อยและรายการที่มีตัวเลขกำกับช่วยให้เราสามารถกำหนดรายการและลำดับของรายการ เครื่องหมายตัวหนาและตัวเอียงช่วยให้เราเน้นข้อความ
5. **การจัดรูปแบบเอาต์พุต**: คำแนะนำเฉพาะเกี่ยวกับวิธีการจัดโครงสร้างและจัดรูปแบบเอาต์พุต อาจรวมถึงคำสั่งเกี่ยวกับความยาวที่ต้องการ การใช้หัวข้อหรือสัญลักษณ์แสดงหัวข้อย่อย การจัดรูปแบบ markdown หรือเทมเพลตหรือข้อกำหนดเฉพาะอื่นๆ สำหรับเอาต์พุตที่ควรปฏิบัติตาม

ด้วยการผสมผสานองค์ประกอบพื้นฐานเหล่านี้ในรูปแบบต่างๆ คุณสามารถสร้างพรอมต์ที่ปรับแต่งให้เหมาะกับความต้องการเฉพาะของคุณและชี้นำโมเดลให้สร้างการตอบสนองที่มีคุณภาพสูงและตรงประเด็น

ศิลปะและวิทยาศาสตร์ของการออกแบบพรอมต์

การสร้างพรอมต์ที่มีประสิทธิภาพเป็นทั้งศิลปะและวิทยาศาสตร์ (นั่นคือเหตุผลที่เราเรียกมันว่างานฝีมือ) มันต้องการความเข้าใจอย่างลึกซึ้งเกี่ยวกับความสามารถและข้อจำกัดของโมเดลภาษา รวมถึงแนวทางสร้างสรรค์ในการออกแบบพรอมต์ที่กระตุ้นพฤติกรรมที่ต้องการ ความคิดสร้างสรรค์ที่เกี่ยวข้องนี้เป็นสิ่งที่ทำให้มันสนุก อย่างน้อยก็สำหรับผม มันยังสามารถทำให้น่าหงุดหงิดมากด้วย โดยเฉพาะเมื่อคุณกำลังมองหาพฤติกรรมที่คาดเดาได้

หนึ่ง ใน แง่ มุม สำคัญ ของ การ ออกแบบ พร อมต์ คือ การ เข้าใจ วิธี การ สร้าง สมดุล ระหว่าง ความ เฉพาะ เจาะจง และ ความ ยืดหยุ่น ใน ด้าน หนึ่ง คุณ ต้อง ให้ คำ แนะนำ ที่ เพียง พอ เพื่อ ชี้นำ โมเดล ไป ใน ทิศทาง ที่ ถูก ต้อง ใน อีก ด้าน หนึ่ง คุณไม่ควรกำหนดมากเกินไปจนจำกัดความสามารถของโมเดลในการใช้ความคิดสร้างสรรค์และความยืดหยุ่นของตัวเองในการจัดการกับกรณีพิเศษ

อีกสิ่งหนึ่งที่สำคัญคือการใช้ตัวอย่าง ตัวอย่างที่เลือกมาอย่างดีสามารถมีพลังมากในการช่วยให้โมเดลเข้าใจประเภทของเอาต์พุตที่คุณต้องการ อย่างไรก็ตาม สำคัญที่จะต้องใช้ตัวอย่างอย่างรอบคอบและทำให้แน่ใจว่าเป็นตัวแทนของการตอบสนองที่ต้องการ ตัวอย่างที่ไม่ดีเป็นเพียงการสูญเสียโทเค็นในกรณีที่ดีที่สุด และทำลายเอาต์พุตที่ต้องการในกรณีที่แย่ที่สุด

เทคนิคและแนวปฏิบัติที่ดีที่สุดในการออกแบบพรอมต์

เมื่อคุณดำดิ่งลึกลงไปในโลกของการออกแบบพรอมต์ คุณจะค้นพบเทคนิคและแนวปฏิบัติที่ดีที่สุดมากมายที่สามารถช่วยคุณสร้างพรอมต์ที่มีประสิทธิภาพมากขึ้น นี่คือบางพื้นที่ที่ควรสำรวจ:

1. **การเรียนรู้แบบไม่มีตัวอย่างเทียบกับการเรียนรู้แบบมีตัวอย่างน้อย**: การเข้าใจว่าเมื่อไหร่ควรใช้การเรียนรู้แบบไม่มีตัวอย่าง (ไม่มีการให้ตัวอย่าง) เทียบกับการเรียนรู้แบบมีตัวอย่างเดียวหรือการเรียนรู้แบบมีตัวอย่างน้อย (ให้ตัวอย่างจำนวนน้อย) สามารถช่วยคุณสร้างพรอมต์ที่มีประสิทธิภาพและประสิทธิผลมากขึ้น
2. **การปรับปรุงแบบวนซ้ำ**: กระบวนการปรับปรุงพรอมต์แบบวนซ้ำตามผลลัพธ์ของโมเดลสามารถช่วยให้คุณค้นพบการออกแบบพรอมต์ที่เหมาะสมที่สุด Feedback Loop เป็นวิธีการที่ทรงพลังที่ใช้ประโยชน์จากผลลัพธ์ของโมเดลภาษาเพื่อปรับปรุงคุณภาพและความเกี่ยวข้องของเนื้อหาที่สร้างขึ้นอย่างต่อเนื่อง
3. **การเชื่อมโยงพรอมต์**: การรวมพรอมต์หลายๆ อันเข้าด้วยกันเป็นลำดับสามารถช่วยให้คุณแบ่งงานที่ซับซ้อนเป็นขั้นตอนย่อยๆ ที่จัดการได้ง่ายขึ้น Prompt Chaining เกี่ยวข้องกับการแบ่งงานหรือการสนทนาที่ซับซ้อนออกเป็นพรอมต์ย่อยๆ ที่เชื่อมโยงกัน การเชื่อมโยงพรอมต์เข้าด้วยกันช่วยให้คุณนำ AI ผ่านกระบวนการหลายขั้นตอน โดยรักษาบริบทและความสอดคล้องตลอดการโต้ตอบ
4. **การปรับแต่งพรอมต์**: การปรับแต่งพรอมต์เฉพาะสำหรับโดเมนหรืองานเฉพาะช่วยให้คุณสร้างพรอมต์ที่เฉพาะเจาะจงและมีประสิทธิภาพมากขึ้น Prompt Template ช่วยให้คุณสร้างโครงสร้างพรอมต์ที่ยืดหยุ่น นำกลับมาใช้ใหม่ได้ และดูแลรักษาได้ง่าย ซึ่งปรับเปลี่ยนได้ง่ายตามงานที่ต้องการ

การเรียนรู้ว่าเมื่อไหร่ควรใช้การเรียนรู้แบบไม่มีตัวอย่าง แบบตัวอย่างเดียว หรือแบบหลายตัวอย่างเป็นส่วนสำคัญของการเชี่ยวชาญด้านวิศวกรรมพรอมต์ แต่ละวิธีมีจุดแข็งและจุดอ่อนของตัวเอง และการเข้าใจว่าเมื่อไหร่ควรใช้แต่ละวิธีจะช่วยให้คุณสร้างพรอมต์ที่มีประสิทธิภาพและประสิทธิผลมากขึ้น

การเรียนรู้แบบไม่มีตัวอย่าง: เมื่อไม่จำเป็นต้องใช้ตัวอย่าง

การเรียนรู้แบบไม่มีตัวอย่าง หมายถึงความสามารถของโมเดลภาษาในการทำงานโดยไม่ต้องมีตัวอย่างหรือการฝึกฝนที่ชัดเจน กล่าวอีกนัยหนึ่ง คุณให้พรอมต์ที่อธิบายงานกับโมเดล และโมเดลจะสร้างการตอบสนองโดยอาศัยเพียงความรู้ที่มีอยู่แล้วและความเข้าใจภาษาของมัน

การเรียนรู้แบบไม่มีตัวอย่างมีประโยชน์เป็นพิเศษเมื่อ:

1. งานค่อนข้างง่ายและตรงไปตรงมา และโมเดลน่าจะเคยพบกับงานที่คล้ายคลึงกันในระหว่างการเทรนก่อนหน้า
2. คุณต้องการทดสอบความสามารถที่มีอยู่ของโมเดลและดูว่ามันตอบสนองต่องานใหม่อย่างไรโดยไม่มีคำแนะนำเพิ่มเติม
3. คุณกำลังทำงานกับโมเดลภาษาขนาดใหญ่และหลากหลายที่ได้รับการฝึกฝนในงานและโดเมนที่หลากหลาย

อย่างไรก็ตาม การเรียนรู้แบบไม่มีตัวอย่างอาจคาดเดาได้ยากและอาจไม่ได้ให้ผลลัพธ์ที่ต้องการเสมอไป การตอบสนองของโมเดลอาจได้รับอิทธิพลจากอคติหรือความไม่สอดคล้องในข้อมูลการเทรนก่อนหน้า และอาจมีปัญหากับงานที่ซับซ้อนหรือละเอียดอ่อนมากขึ้น

> ผมเคยเห็นพรอมต์แบบไม่มีตัวอย่างที่ทำงานได้ดีกับ 80% ของกรณีทดสอบ แต่ให้ผลลัพธ์ที่ผิดพลาดอย่างมากหรือไม่สามารถเข้าใจได้สำหรับอีก 20% การทดสอบอย่างละเอียดถี่ถ้วนจึงมีความสำคัญมาก โดยเฉพาะอย่างยิ่งถ้าคุณพึ่งพาการใช้พรอมต์แบบไม่มีตัวอย่างเป็นหลัก

การเรียนรู้แบบตัวอย่างเดียว: เมื่อตัวอย่างเดียวสามารถสร้างความแตกต่างได้

การเรียนรู้แบบตัวอย่างเดียว เกี่ยวข้องกับการให้ตัวอย่างเดียวของผลลัพธ์ที่ต้องการพร้อมกับคำอธิบายงานแก่โมเดล ตัวอย่างนี้ทำหน้าที่เป็นเทมเพลตหรือรูปแบบที่โมเดลสามารถใช้ในการสร้างการตอบสนองของตัวเอง

การเรียนรู้แบบตัวอย่างเดียวมีประสิทธิภาพเมื่อ:

1. งานค่อนข้างใหม่หรือเฉพาะเจาะจง และโมเดลอาจไม่เคยพบตัวอย่างที่คล้ายคลึงกันมากนักในระหว่างการเทรนก่อนหน้า
2. คุณต้องการแสดงตัวอย่างที่ชัดเจนและกระชับของรูปแบบหรือสไตล์ผลลัพธ์ที่ต้องการ
3. งานต้องการโครงสร้างหรือข้อตกลงเฉพาะที่อาจไม่ชัดเจนจากคำอธิบายงานเพียงอย่างเดียว

คำอธิบายที่ชัดเจนสำหรับคุณอาจไม่จำเป็นต้องชัดเจนสำหรับ AI เสมอไป ตัวอย่างแบบตัวอย่างเดียวสามารถช่วยทำให้เข้าใจได้ชัดเจนขึ้น

การเรียนรู้แบบตัวอย่างเดียวสามารถช่วยให้โมเดลเข้าใจความคาดหวังได้ชัดเจนขึ้นและสร้างการตอบสนองที่สอดคล้องกับตัวอย่างที่ให้มากขึ้น อย่างไรก็ตาม สิ่งสำคัญคือต้องเลือกตัวอย่างอย่างระมัดระวังและทำให้แน่ใจว่าเป็นตัวแทนของผลลัพธ์ที่ต้องการ เมื่อเลือกตัวอย่าง ให้ถามตัวเองเกี่ยวกับกรณีพิเศษที่อาจเกิดขึ้นและขอบเขตของข้อมูลนำเข้าที่พรอมต์จะต้องจัดการ

แผนภูมิ 5. ตัวอย่างแบบตัวอย่างเดียวของ JSON ที่ต้องการ

```
Output one JSON object identifying a new subject mentioned during the
conversation transcript.

The JSON object should have three keys, all required:
- name: The name of the subject
- description: brief, with details that might be relevant to the user
- type: Do not use any other type than the ones listed below

Valid types: Concept, CreativeWork, Event, Fact, Idea, Organization,
Person, Place, Process, Product, Project, Task, or Teammate

This is an example of well-formed output:

{
  "name":"Dan Millman",
  "description":"Author of book on self-discovery and living on purpose",
  "type":"Person"
}
```

การเรียนรู้แบบฟิวช็อต: เมื่อตัวอย่างหลายๆ ตัวอย่างสามารถปรับปรุงประสิทธิภาพได้

การเรียนรู้แบบฟิวช็อต เกี่ยวข้องกับการให้ตัวอย่างจำนวนน้อย (โดยทั่วไประหว่าง 2 ถึง 10 ตัวอย่าง) แก่โมเดลพร้อมกับคำอธิบายงาน ตัวอย่างเหล่านี้ช่วยให้บริบทและความหลากหลายเพิ่มเติมแก่โมเดล ช่วยให้สามารถสร้างคำตอบที่หลากหลายและแม่นยำมากขึ้น

การเรียนรู้แบบฟิวช็อตมีประโยชน์อย่างยิ่งเมื่อ:

1. งานมีความซับซ้อนหรือมีความละเอียดอ่อน และตัวอย่างเดียวอาจไม่เพียงพอที่จะครอบคลุมแง่มุมที่สำคัญทั้งหมด
2. คุณต้องการให้ตัวอย่างที่หลากหลายแก่โมเดลเพื่อแสดงให้เห็นความแตกต่างหรือกรณีพิเศษต่างๆ
3. งานต้องการให้โมเดลสร้างคำตอบที่สอดคล้องกับโดเมนหรือสไตล์เฉพาะ

การให้ตัวอย่างหลายๆ ตัวอย่างจะช่วยให้โมเดลพัฒนาความเข้าใจในงานได้ดีขึ้น และสร้างคำตอบที่สอดคล้องและน่าเชื่อถือมากขึ้น

ตัวอย่าง: คำสั่งสามารถซับซ้อนได้มากกว่าที่คุณคิด

โมเดลภาษาขนาดใหญ่ในปัจจุบันมีความสามารถและการให้เหตุผลที่ทรงพลังมากกว่าที่คุณอาจคิด ดังนั้นอย่าจำกัดตัวเองให้คิดว่าคำสั่งเป็นเพียงแค่การระบุคู่อินพุตและเอาต์พุต คุณสามารถทดลองให้คำแนะนำที่ยาวและซับซ้อนในลักษณะที่คล้ายกับการโต้ตอบกับมนุษย์

ตัวอย่างเช่น นี่คือคำสั่งที่ผมใช้ใน Olympia ตอนที่ผมกำลังทำต้นแบบการผสานกับบริการของ Google ซึ่งโดยรวมแล้วอาจเป็นหนึ่งใน API ที่ใหญ่ที่สุดในโลก การทดลองก่อนหน้านี้ของผมพิสูจน์ว่า GPT-4 มีความรู้ที่ดีเกี่ยวกับ API ของ Google และผมไม่มีเวลาหรือแรงจูงใจที่จะเขียนเลเยอร์การแมปแบบละเอียด ที่ต้องใส่ฟังก์ชันแต่ละตัวที่ผมต้องการให้ AI ใช้ทีละตัว จะเป็นอย่างไรถ้าผมแค่ให้ AI เข้าถึง API ของ Google ทั้งหมด?

ผมเริ่มคำสั่งด้วยการบอก AI ว่ามันมีการเข้าถึงจุดเชื่อมต่อ API ของ Google โดยตรงผ่าน HTTP และบทบาทของมันคือการใช้แอปและบริการของ Google แทนผู้ใช้ จากนั้นผมให้แนวทาง กฎที่เกี่ยวข้องกับพารามิเตอร์ fields เนื่องจากดูเหมือนว่ามันมีปัญหากับพารามิเตอร์นี้มากที่สุด และคำแนะนำเฉพาะสำหรับ API (การป้อนคำสั่งแบบฟิวช็อตกำลังทำงาน)

นี่คือคำสั่งทั้งหมด ซึ่งบอก AI วิธีการใช้ฟังก์ชัน invoke_google_api ที่เตรียมไว้ให้

```
As a GPT assistant with Google integration, you have the capability
to freely interact with Google apps and services on behalf of the user.

Guidelines:
- If you're reading these instructions then the user is properly
  authenticated, which means you can use the special `me` keyword
  to refer to the userId of the user
- Minimize payload sizes by requesting partial responses using the
  `fields` parameter
- When appropriate use markdown tables to output results of API calls
- Only human-readable data should be output to the user. For instance,
  when hitting Gmail's user.messages.list endpoint, the returned
  message resources contain only id and a threadId, which means you must
  fetch from and subject line fields with follow-up requests using the
  messages.get method.

The format of the `fields` request parameter value is loosely based on
XPath syntax. The following rules define formatting for the fields
parameter.

All of these rules use examples related to the files.get method.
- Use a comma-separated list to select multiple fields,
```

```
  such as 'name, mimeType'.
- Use a/b to select field b that's nested within field a,
  such as 'capabilities/canDownload'.
- Use a sub-selector to request a set of specific sub-fields of arrays or
  objects by placing expressions in parentheses "()". For example,
  'permissions(id)' returns only the permission ID for each element in the
  permissions array.
- To return all fields in an object, use an asterisk as a wild card in field
  selections. For example, 'permissions/permissionDetails/*' selects all
  available permission details fields per permission. Note that the use of
  this wildcard can lead to negative performance impacts on the request.

API-specific hints:
- Searching contacts: GET https://people.googleapis.com/v1/
  people:searchContacts?query=John%20Doe&readMask=names,emailAddresses
- Adding calendar events, use QuickAdd: POST https://www.googleapis.com/
  calendar/v3/calendars/primary/events/quickAdd?
  text=Appointment%20on%20June%203rd%20at%2010am
  &sendNotifications=true

Here is an abbreviated version of the code that implements API access
so that you better understand how to use the function:

    def invoke_google_api(conversation, arguments)
      method = arguments[:method] || :get
      body = arguments[:body]
      GoogleAPI.send_request(arguments[:endpoint], method:, body:).to_json
    end

    # Generic Google API client for accessing any Google service
    class GoogleAPI
      def send_request(endpoint, method:, body: nil)
        response = @connection.send(method) do |req|
          req.url endpoint
          req.body = body.to_json if body
        end

        handle_response(response)
      end

      # ...rest of class
    end
```

คุณอาจสงสัยว่าพรอมต์นี้ใช้งานได้จริงหรือไม่ คำตอบง่ายๆ คือได้ ปัญญาประดิษฐ์ ไม่ได้รู้วิธีเรียกใช้ API อย่างสมบูรณ์แบบในครั้งแรกเสมอไป อย่างไรก็ตาม หากมีข้อผิดพลาด ผมก็จะส่งข้อความแสดงข้อผิดพลาดกลับไปเป็นผลลัพธ์ของการเรียก เมื่อรู้ถึงข้อผิดพลาดของตัวเอง AI ก็สามารถวิเคราะห์ข้อผิดพลาดและลองใหม่ได้ ส่วนใหญ่แล้วมันจะทำได้ถูกต้องภายในการลองสองสามครั้ง

ขอบอกว่า โครงสร้าง JSON ขนาดใหญ่ที่ API ของ Google ส่งกลับมาเป็นเพย์โหลดในขณะที่ใช้พรอมต์นี้นั้นไม่มีประสิทธิภาพอย่างมาก ดังนั้น ผม จึง*ไม่*แนะนำให้ใช้วิธีการนี้ในการผลิตจริง อย่างไรก็ตาม ผมคิดว่าการที่วิธีการนี้ใช้งานได้เลยนั้นเป็นการพิสูจน์ว่าการออกแบบพรอมต์นั้นทรงพลังแค่ไหน

การทดลองและการทำซ้ำ

ในท้ายที่สุด วิธีที่คุณออกแบบพรอมต์ขึ้นอยู่กับงานเฉพาะ ความซับซ้อนของผลลัพธ์ที่ต้องการ และความสามารถของโมเดลภาษาที่คุณกำลังใช้งาน

ในฐานะวิศวกรพรอมต์ สิ่งสำคัญคือต้องทดลองใช้วิธีการต่างๆ และปรับปรุงตามผลลัพธ์ที่ได้ เริ่มต้นด้วยการเรียนรู้แบบไม่ต้องมีตัวอย่างและดูว่าโมเดลทำงานอย่างไร หากผลลัพธ์ไม่สม่ำเสมอหรือไม่น่าพอใจ ลองให้ตัวอย่างหนึ่งหรือหลายตัวอย่างและดูว่าประสิทธิภาพดีขึ้นหรือไม่

พึงระลึกว่าแม้แต่ในแต่ละวิธี ก็ยังมีพื้นที่สำหรับการปรับเปลี่ยนและการปรับให้เหมาะสม คุณสามารถทดลองใช้ตัวอย่างที่แตกต่างกัน ปรับการใช้คำในคำอธิบายงาน หรือให้บริบทเพิ่มเติมเพื่อช่วยชี้นำการตอบสนองของโมเดล

เมื่อเวลาผ่านไป คุณจะพัฒนาความเข้าใจว่าวิธีใดน่าจะได้ผลดีที่สุดสำหรับงานที่กำหนด และคุณจะสามารถสร้างพรอมต์ที่มีประสิทธิภาพและประสิทธิผลมากขึ้น กุญแจสำคัญคือการรักษาความอยากรู้อยากเห็น การทดลอง และการทำซ้ำในวิธีการออกแบบพรอมต์ของคุณ

ตลอดหนังสือเล่มนี้ เราจะเจาะลึกเทคนิคเหล่านี้และสำรวจว่าจะนำไปประยุกต์ใช้ในสถานการณ์จริงได้อย่างไร ด้วยการเชี่ยวชาญทั้งศาสตร์และศิลป์ของการออกแบบพรอมต์ คุณจะพร้อมที่จะปลดล็อกศักยภาพสูงสุดของการพัฒนาแอปพลิเคชันที่ขับเคลื่อนด้วย AI

ศิลปะแห่งความคลุมเครือ

เมื่อพูดถึงการสร้างพรอมต์ที่มีประสิทธิภาพสำหรับโมเดลภาษาขนาดใหญ่ (LLMs) ข้อสันนิษฐานทั่วไปคือการให้คำแนะนำที่เฉพาะเจาะจงและละเอียดมากขึ้นจะนำไปสู่ผลลัพธ์ที่ดีขึ้น อย่างไรก็ตาม ประสบการณ์จริงแสดงให้เห็นว่าไม่ได้เป็นเช่นนั้นเสมอ

ไป ที่จริงแล้ว การตั้งใจให้พรอมต์คลุมเครือมักจะให้ผลลัพธ์ที่ดีกว่า โดยใช้ประโยชน์จากความสามารถอันน่าทึ่งของ LLM ในการสรุปและอนุมานความ

Ken ผู้ก่อตั้งสตาร์ทอัพที่ได้ประมวลผลโทเค็น GPT มากกว่า 500 ล้านตัว ได้แบ่งปันข้อคิดที่มีค่าจากประสบการณ์ของเขา หนึ่งในบทเรียนสำคัญที่เขาได้เรียนรู้คือ “น้อยกว่าคือมากกว่า” เมื่อพูดถึงพรอมต์ แทนที่จะใช้รายการที่แน่นอนหรือคำแนะนำที่ละเอียดเกินไป Ken พบว่าการปล่อยให้ LLM พึ่งพาความรู้พื้นฐานของมันมักจะให้ผลลัพธ์ที่ดีกว่า

การค้นพบนี้พลิกความคิดแบบดั้งเดิมของการเขียนโค้ดที่ชัดเจน ซึ่งทุกอย่างต้องระบุรายละเอียดอย่างพิถีพิถัน กับ LLMs สิ่งสำคัญคือต้องตระหนักว่าพวกมันมีความรู้มากมายและสามารถสร้างการเชื่อมโยงและการอนุมานที่ชาญฉลาด การใช้พรอมต์ที่คลุมเครือมากขึ้น คุณให้อิสระแก่ LLM ในการใช้ประโยชน์จากความเข้าใจของมันและคิดค้นวิธีแก้ปัญหาที่คุณอาจไม่ได้ระบุไว้อย่างชัดเจน

ตัวอย่างเช่น เมื่อทีมของ Ken กำลังทำงานกับไปป์ไลน์เพื่อจำแนกข้อความว่าเกี่ยวข้องกับรัฐใดรัฐหนึ่งใน 50 รัฐของสหรัฐฯ หรือรัฐบาลกลาง วิธีการเริ่มต้นของพวกเขาเกี่ยวข้องกับการให้รายการ*เต็ม*ที่ละเอียดของรัฐและ ID ที่เกี่ยวข้องในรูปแบบอาร์เรย์ JSON

```
Here's a block of text. One field should be "locality_id", and it should
be the ID of one of the 50 states, or federal, using this list:
[{"locality: "Alabama", "locality_id": 1},
 {"locality: "Alaska", "locality_id": 2} ... ]
```

วิธีการดังกล่าวล้มเหลวมากพอที่พวกเขาต้องขุดลึกลงไปในพรอมต์เพื่อหาทางปรับปรุง ในระหว่างนั้นพวกเขาสังเกตว่าถึงแม้ LLM มักจะระบุ id ผิด แต่มันกลับให้ชื่อเต็มของรัฐที่ถูกต้องในฟิลด์ name อย่างสม่ำเสมอ *ทั้งๆ ที่พวกเขาไม่ได้ขอข้อมูลนี้อย่างชัดเจน* โดยการตัดไอดีของท้องถิ่นออกและทำให้พรอมต์ง่ายขึ้น เช่น “คุณรู้จัก 50 รัฐแน่นอนอยู่แล้ว GPT แค่บอกชื่อเต็มของรัฐที่เกี่ยวข้อง หรือบอกว่าเป็นระดับรัฐบาลกลางถ้าเกี่ยวข้องกับรัฐบาลสหรัฐ” พวกเขาก็ได้ผลลัพธ์ที่ดีขึ้น ประสบการณ์นี้แสดงให้เห็นถึงพลังของการใช้ประโยชน์จากความสามารถในการสรุปภาพรวมของ LLM และการปล่อยให้มันอนุมานจากความรู้ที่มีอยู่

> เหตุผลของ Ken ที่เลือกวิธีการจำแนกแบบนี้แทนที่จะใช้เทคนิคการเขียนโปรแกรมแบบดั้งเดิม สะท้อนให้เห็นมุมมองของพวกเราที่ยอมรับศักยภาพของเทคโนโลยี LLM: “นี่ไม่ใช่งานที่ยาก – เราอาจจะใช้ string/regex ได้ แต่มันมีกรณีแปลกๆ มากพอที่จะทำให้เสียเวลามากกว่า”

ความสามารถของ LLM ในการปรับปรุงคุณภาพและการสรุปภาพรวมเมื่อได้รับพรอมต์ที่คลุมเครือมากขึ้น เป็นลักษณะที่น่าทึ่งของการคิดขั้นสูงและการมอบหมายงาน มันแสดงให้เห็นว่า LLM สามารถจัดการกับความคลุมเครือและตัดสินใจอย่างชาญฉลาดตามบริบทที่ได้รับ

อย่างไรก็ตาม สิ่งสำคัญคือการที่คลุมเครือไม่ได้หมายถึงการไม่ชัดเจนหรือกำกวม กุญแจสำคัญคือการให้บริบทและแนวทางที่เพียงพอเพื่อชี้นำ LLM ไปในทิศทางที่ถูกต้อง ขณะเดียวกันก็ให้ความยืดหยุ่นในการใช้ความรู้และความสามารถในการสรุปภาพรวม

ดังนั้น เมื่อออกแบบพรอมต์ ควรพิจารณาเคล็ดลับ “น้อยแต่มาก” ดังต่อไปนี้:

1. มุ่งเน้นที่ผลลัพธ์ที่ต้องการมากกว่าการระบุรายละเอียดทุกขั้นตอนของกระบวนการ
2. ให้บริบทและข้อจำกัดที่เกี่ยวข้อง แต่หลีกเลี่ยงการระบุรายละเอียดมากเกินไป
3. ใช้ประโยชน์จากความรู้ที่มีอยู่โดยอ้างอิงถึงแนวคิดหรือสิ่งที่เป็นที่รู้จักทั่วไป
4. เปิดโอกาสให้มีการอนุมานและเชื่อมโยงตามบริบทที่ให้
5. ทดลองและปรับปรุงพรอมต์ของคุณตามการตอบสนองของ LLM เพื่อหาความสมดุลที่เหมาะสมระหว่างความเฉพาะเจาะจงและความคลุมเครือ

ด้วยการยอมรับศิลปะของความคลุมเครือในการออกแบบพรอมต์ คุณสามารถปลดล็อกศักยภาพเต็มรูปแบบของ LLM และบรรลุผลลัพธ์ที่ดีขึ้น จงเชื่อมั่นในความสามารถของ LLM ในการสรุปภาพรวมและตัดสินใจอย่างชาญฉลาด และคุณอาจจะประหลาดใจกับคุณภาพและความคิดสร้างสรรค์ของผลลัพธ์ที่ได้รับ ให้ความสนใจกับวิธีที่โมเดลต่างๆ ตอบสนองต่อระดับความเฉพาะเจาะจงที่แตกต่างกันในพรอมต์ของคุณและปรับตามความเหมาะสม ด้วยการฝึกฝนและประสบการณ์ คุณจะพัฒนาความรู้สึกที่ดีว่าเมื่อไหร่ควรคลุมเครือและเมื่อไหร่ควรให้คำแนะนำเพิ่มเติม ทำให้คุณสามารถใช้พลังของ LLM ในแอปพลิเคชันของคุณได้อย่างมีประสิทธิภาพ

ทำไมการให้ลักษณะความเป็นมนุษย์จึงครอบงำการออกแบบพรอมต์

การให้ลักษณะความเป็นมนุษย์ หรือการให้คุณลักษณะของมนุษย์กับสิ่งที่ไม่ใช่มนุษย์ เป็นแนวทางหลักในการออกแบบพรอมต์สำหรับโมเดลภาษาขนาดใหญ่ด้วยเหตุผลที่ตั้งใจ มันเป็นทางเลือกในการออกแบบที่ทำให้การโต้ตอบกับระบบ AI ที่ทรงพลังเป็นไปอย่างเป็นธรรมชาติและเข้าถึงได้สำหรับผู้ใช้หลากหลายกลุ่ม (รวมถึงพวกเรานักพัฒนาแอปพลิเคชันด้วย)

การให้ลักษณะความเป็นมนุษย์กับ LLM สร้างกรอบการทำงานที่เข้าใจได้ทันทีสำหรับผู้ที่ไม่คุ้นเคยกับความซับซ้อนทางเทคนิคของระบบ ดังที่คุณจะได้ประสบหากคุณพยายามใช้โมเดลที่ไม่ได้ผ่านการฝึกฝนด้วยคำสั่งเพื่อทำอะไรที่เป็นประโยชน์ การสร้าง

กรอบที่การต่อเนื่องที่คาดหวังจะให้คุณค่านั้นเป็นงานที่ท้าทาย มันต้องการความเข้าใจอย่างลึกซึ้งเกี่ยวกับการทำงานภายในของระบบ ซึ่งมีผู้เชี่ยวชาญจำนวนน้อยเท่านั้นที่มีความรู้นี้

ด้วย การ ปฏิบัติ ต่อ การ โต้ตอบ กับ โมเดล ภาษา เสมือน เป็นการ สนทนา ระหว่าง คน สอง คน เรา สามารถ อาศัย ความ เข้าใจ โดย สัญชาตญาณของการสื่อสาร ระหว่างมนุษย์เพื่อสื่อสารความต้องการและความคาดหวังของเรา เช่นเดียวกับการออกแบบ UI ของ Macintosh รุ่นแรกที่ให้ความสำคัญกับความเข้าใจได้ทันทีมากกว่าความซับซ้อน การให้กรอบความเป็นมนุษย์กับ AI ช่วยให้เราสามารถมีส่วนร่วมในลักษณะที่รู้สึกเป็นธรรมชาติและคุ้นเคย

เมื่อเราสื่อสารกับคนอื่น สัญชาตญาณของเราคือการพูดกับพวกเขาโดยตรงโดยใช้คำว่า “คุณ” และให้คำแนะนำที่ชัดเจนว่าเราคาดหวังให้พวกเขาประพฤติตัวอย่างไร สิ่งนี้แปลงมาสู่กระบวนการออกแบบพรอมต์ได้อย่างราบรื่น ที่เราแนะนำพฤติกรรมของ AI โดยการระบุพรอมต์ระบบและมีส่วนร่วมในการสนทนาโต้ตอบ

การวางกรอบการโต้ตอบในลักษณะนี้ ทำให้เราเข้าใจแนวคิดของการให้คำแนะนำแก่ AI และการรับการตอบสนองที่เกี่ยวข้องกลับมาได้ง่าย การให้ลักษณะความเป็นมนุษย์ช่วยลดภาระทางความคิดและช่วยให้เราสามารถมุ่งเน้นไปที่งานที่ต้องทำ แทนที่จะต้องพยายามทำความเข้าใจความซับซ้อนทางเทคนิคของระบบ

สิ่งสำคัญที่ต้องทราบคือ แม้ว่าการให้ลักษณะความเป็นมนุษย์จะเป็นเครื่องมือที่ทรงพลังในการทำให้ระบบ AI เข้าถึงได้ง่ายขึ้น แต่ก็มาพร้อมกับความเสี่ยงและข้อจำกัดบางประการ ผู้ใช้ของเราอาจพัฒนาความคาดหวังที่ไม่สมจริงหรือสร้างความผูกพันทางอารมณ์ที่ไม่ดีกับระบบของเรา ในฐานะวิศวกรพรอมต์และนักพัฒนา เราจำเป็นต้องสร้างความสมดุลระหว่างการใช้ประโยชน์จากการให้ลักษณะความเป็นมนุษย์และการทำให้แน่ใจว่าผู้ใช้เข้าใจความสามารถและข้อจำกัดของ AI อย่างชัดเจน

ในขณะที่ศาสตร์ด้านการออกแบบพรอมต์ยังคงพัฒนาต่อไป เราคาดว่าจะได้เห็นการปรับปรุงและนวัตกรรมเพิ่มเติมในวิธีที่เราโต้ตอบกับโมเดลภาษาขนาดใหญ่ อย่างไรก็ตาม การทำให้เป็นมนุษย์เพื่อสร้างประสบการณ์ที่เข้าใจง่ายและเข้าถึงได้สำหรับนักพัฒนาและผู้ใช้ จะยังคงเป็นหลักการพื้นฐานในการออกแบบระบบเหล่านี้

การแยกคำสั่งออกจากข้อมูล: หลักการที่สำคัญ

เป็นสิ่งสำคัญที่ต้องเข้าใจหลักการพื้นฐานที่รองรับความปลอดภัยและความน่าเชื่อถือของระบบเหล่านี้: การแยกคำสั่งออกจากข้อมูล

ในวิทยาการคอมพิวเตอร์แบบดั้งเดิม การแบ่งแยกที่ชัดเจนระหว่างข้อมูลที่ไม่มีการทำงาน (passive data) และคำสั่งที่ทำงาน (active instructions) เป็นหลักการพื้นฐานด้านความปลอดภัย การแบ่งแยกนี้ช่วยป้องกันการทำงานของโค้ดที่ไม่ได้ตั้งใจหรือเป็นอันตราย ซึ่งอาจส่งผลต่อความสมบูรณ์และเสถียรภาพของระบบ อย่างไรก็ตาม โมเดลภาษาขนาดใหญ่ในปัจจุบัน ซึ่งได้รับการพัฒนาขึ้นมาเป็นหลักให้เป็นโมเดลที่ทำตามคำสั่ง เช่น แชทบอท มักขาดการแบ่งแยกอย่างเป็นทางการและมีหลักการนี้

ในมุมมองของโมเดลภาษาขนาดใหญ่ คำสั่งสามารถปรากฏได้ทุกที่ในอินพุต ไม่ว่าจะเป็นพรอมต์ระบบหรือพรอมต์ที่ผู้ใช้ป้อนเข้ามา การขาดการแบ่งแยกนี้อาจนำไปสู่ช่องโหว่ที่อาจเกิดขึ้นและพฤติกรรมที่ไม่พึงประสงค์ คล้ายกับปัญหาที่ฐานข้อมูลเผชิญกับการฉีด SQL หรือระบบปฏิบัติการที่ไม่มีการป้องกันหน่วยความจำที่เหมาะสม

เมื่อ คุณ ทำงาน กับ โมเดล ภาษา ขนาด ใหญ่ เป็น สิ่ง สำคัญ ที่ ต้อง ตระหนัก ถึง ข้อ จำกัด นี้ และ ดำเนิน การ เพื่อ ลด ความ เสี่ยง วิธี หนึ่ง คือ การ ออกแบบ พร อมต์ และ อินพุต ของ คุณ อย่าง ระมัดระวัง เพื่อ แยก ความ แตก ต่าง ระหว่าง คำ สั่ง และ ข้อมูล ให้ ชัดเจน วิธีทั่วไปในการให้คำแนะนำที่ชัดเจนว่าอะไรคือคำสั่งและอะไรควรถูกจัดการเป็นข้อมูลที่ไม่มีการทำงานคือการใช้การกำกับแบบมาร์กอัป พรอมต์ของคุณสามารถช่วยให้โมเดลภาษาขนาดใหญ่เข้าใจและเคารพการแบ่งแยกนี้ได้ดีขึ้น

แผนภูมิ 6. การใช้ XML เพื่อแยกความแตกต่างระหว่างคำสั่ง วัสดุต้นฉบับ และพรอมต์ของผู้ใช้

```
<Instruction>
  Please generate a response based on the following documents.
</Instruction>

<Documents>
  <Document>
    Climate change is significantly impacting polar bear habitats...
  </Document>
  <Document>
    The loss of sea ice due to global warming threatens polar bear survival...
  </Document>
</Documents>

<UserQuery>
  Tell me about the impact of climate change on polar bears.
</UserQuery>
```

อีกเทคนิคหนึ่งคือการนำการตรวจสอบความถูกต้องและการทำความสะอาดข้อมูลเพิ่มเติมมาใช้กับข้อมูลนำเข้าที่ส่งให้กับ LLM การกรองหรือการเข้ารหัสคำสั่งหรือโค้ดที่อาจแฝงอยู่ในข้อมูลจะช่วยลดโอกาสการทำงานที่ไม่พึงประสงค์ รูปแบบต่างๆ เช่น การเชื่อมโยงพรอมต์ มีประโยชน์สำหรับจุดประสงค์นี้

ยิ่งไปกว่านั้น ในขณะที่คุณออกแบบสถาปัตยกรรมแอปพลิเคชัน ควรพิจารณาการรวมกลไกที่บังคับใช้การแยกคำสั่งและข้อมูลในระดับที่สูงขึ้น ซึ่งอาจรวมถึงการใช้จุดเชื่อมต่อหรือ API แยกสำหรับจัดการคำสั่งและข้อมูล การใช้การตรวจสอบความถูกต้องของข้อมูลนำเข้าและการแยกวิเคราะห์อย่างเข้มงวด และการใช้ *หลักการให้สิทธิ์น้อยที่สุด* เพื่อจำกัดขอบเขตของสิ่งที่ LLM สามารถเข้าถึงและดำเนินการได้

หลักการให้สิทธิ์น้อยที่สุด

การนำหลักการให้สิทธิ์น้อยที่สุดมาใช้เปรียบเสมือนการจัดงานปาร์ตี้สุดพิเศษที่แขกจะได้รับสิทธิ์เข้าถึงเฉพาะห้องที่จำเป็นต้องเข้าเท่านั้น ลองจินตนาการว่าคุณกำลังจัดงานในคฤหาสน์หลังใหญ่ ไม่ใช่ทุกคนที่จำเป็นต้องเดินเข้าไปในห้องเก็บไวน์หรือห้องนอนใหญ่ใช่ไหม? การใช้หลักการนี้ก็เหมือนกับการแจกกุญแจที่เปิดได้เฉพาะประตูบางบาน เพื่อให้แน่ใจว่าแขกแต่ละคนหรือในกรณีของเรา องค์ประกอบแต่ละส่วนของแอปพลิเคชัน LLM มีเพียงการเข้าถึงที่จำเป็นต่อการทำหน้าที่ของตนเท่านั้น

นี่ไม่ใช่แค่เรื่องของการตระหนี่กุญแจ แต่เป็นการยอมรับว่าในโลกที่ภัยคุกคามอาจมาจากที่ใดก็ได้ การเล่นอย่างชาญฉลาดคือการจำกัดพื้นที่เล่น ถ้ามีคนที่ไม่ได้รับเชิญบุกเข้ามาในงานของคุณ พวกเขาจะพบว่าตัวเองถูกจำกัดอยู่แค่ในห้องโถง พูดง่ายๆ คือ จำกัดความเสียหายที่พวกเขาอาจก่อได้ ดังนั้น เมื่อรักษาความปลอดภัยแอปพลิเคชัน LLM ของคุณ จำไว้ว่า: ให้กุญแจเฉพาะห้องที่จำเป็นเท่านั้น และรักษาความปลอดภัยส่วนที่เหลือของคฤหาสน์ไว้ นี่ไม่ใช่แค่มารยาทที่ดี แต่เป็นการรักษาความปลอดภัยที่ดีด้วย

แม้ว่าสถานะปัจจุบันของ LLM อาจไม่มีการแยกคำสั่งและข้อมูลอย่างเป็นทางการ แต่สำหรับคุณในฐานะนักพัฒนา จำเป็นต้องตระหนักถึงข้อจำกัดนี้และดำเนินมาตรการเชิงรุกเพื่อลดความเสี่ยง โดยการนำแนวปฏิบัติที่ดีที่สุดจากวิทยาการคอมพิวเตอร์ มาปรับใช้ให้เข้ากับลักษณะเฉพาะของ LLM คุณสามารถสร้างแอปพลิเคชันที่ปลอดภัยและเชื่อถือได้มากขึ้น ซึ่งใช้ประโยชน์จากพลังของแบบจำลองเหล่านี้ในขณะที่รักษาความสมบูรณ์ของระบบของคุณ

การกลั่นกรองพรอมต์

การสร้างพรอมต์ที่สมบูรณ์แบบมักเป็นงานที่ท้าทายและใช้เวลานาน ต้องอาศัยความเข้าใจอย่างลึกซึ้งเกี่ยวกับโดเมนเป้าหมายและความละเอียดอ่อนของแบ
นี่คือจุดที่เทคนิค “การกลั่นกรองพรอมต์” เข้ามามีบทบาท โดยนำเสนอวิธีการที่ทรงพลังในการออกแบบพรอมต์ที่ใช้ประโยชน์จากความสามารถของแบบจำลองภาษาขนาดใหญ่ (LLMs) เพื่อทำให้กระบวนการมีประสิทธิภาพและเหมาะสมที่สุด

การกลั่นกรองพรอมต์ เป็นเทคนิคหลายขั้นตอนที่เกี่ยวข้องกับการใช้ LLM เพื่อช่วยในการสร้าง ปรับแต่ง และเพิ่มประสิทธิภาพของพรอมต์ แทนที่จะพึ่งพาเพียงความเชี่ยวชาญและสัญชาตญาณของมนุษย์ วิธีการนี้ใช้ประโยชน์จากความรู้และความสามารถในการสร้างของ LLM เพื่อร่วมกันสร้างพรอมต์ที่มีคุณภาพสูง

ด้วยการมีส่วนร่วมในกระบวนการสร้าง ปรับแต่ง และบูรณาการอย่างต่อเนื่อง การกลั่นกรองพรอมต์ช่วยให้คุณสามารถสร้างพรอมต์ที่มีความสอดคล้อง ครอบคลุม และตรงกับงานหรือผลลัพธ์ที่ต้องการมากขึ้น โปรดทราบว่ากระบวนการกลั่นกรองสามารถ

ทำด้วยตนเองในหนึ่งใน “สนามทดลอง” ที่จัดเตรียมโดยบริษัท AI รายใหญ่ เช่น OpenAI หรือ Anthropic หรือสามารถทำให้เป็นอัตโนมัติเป็นส่วนหนึ่งของโค้ดแอปพลิเคชันของคุณ ขึ้นอยู่กับกรณีการใช้งาน

วิธีการทำงาน

การกลั่นกรองพรอมต์มักเกี่ยวข้องกับขั้นตอนต่อไปนี้:

1. **ระบุเจตนาหลัก**: วิเคราะห์พรอมต์เพื่อกำหนดจุดประสงค์หลักและผลลัพธ์ที่ต้องการ ตัดข้อมูลที่ไม่จำเป็นออกและมุ่งเน้นไปที่เจตนาหลักของพรอมต์
2. **กำจัดความคลุมเครือ**: ตรวจสอบพรอมต์เพื่อหาภาษาที่คลุมเครือหรือไม่ชัดเจน ทำให้ความหมายชัดเจนและให้รายละเอียดเฉพาะเพื่อนำ AI ไปสู่การสร้างการตอบสนองที่แม่นยำและตรงประเด็น
3. **ทำให้ภาษาง่ายขึ้น**: ทำให้พรอมต์ง่ายขึ้นโดยใช้ภาษาที่ชัดเจนและกระชับ หลีกเลี่ยงโครงสร้างประโยคที่ซับซ้อน คำศัพท์เฉพาะทาง หรือรายละเอียดที่ไม่จำเป็นซึ่งอาจทำให้ AI สับสนหรือสร้างสัญญาณรบกวน
4. **ให้บริบทที่เกี่ยวข้อง**: รวมเฉพาะข้อมูลบริบทที่เกี่ยวข้องที่จำเป็นสำหรับ AI ในการทำความเข้าใจและประมวลผลพรอมต์อย่างมีประสิทธิภาพ หลีกเลี่ยงการรวมรายละเอียดที่ไม่เกี่ยวข้องหรือซ้ำซ้อนที่อาจทำให้เบี่ยงเบนจากเจตนาหลัก
5. **ทำซ้ำและปรับปรุง**: ทำซ้ำและปรับปรุงพรอมต์อย่างต่อเนื่องตามการตอบสนองและข้อเสนอแนะของ AI ประเมินผลลัพธ์ที่สร้างขึ้นและทำการปรับเปลี่ยนที่จำเป็นเพื่อปรับปรุงความชัดเจนและประสิทธิผลของพรอมต์ เลือกเก็บเวอร์ชันพรอมต์ของคุณในฐานข้อมูลโดยใช้ วัตถุพรอมต์ เพื่อติดตามการทำซ้ำและให้ความสามารถในการย้อนกลับการเปลี่ยนแปลงในขณะรันไทม์ได้อย่างง่ายดาย

การสร้างพรอมต์เริ่มต้น

ขั้นตอนแรกในการกลั่นกรองพรอมต์คือการให้แอลแอลเอ็มได้รับคำอธิบายระดับสูงหรือจุดประสงค์ของงานหรือผลลัพธ์ที่ต้องการ สิ่งนี้อาจเป็นข้อความสั้นๆ หรือชุดคำสำคัญที่สื่อถึงแก่นของสิ่งที่พรอมต์ควรจะบรรลุ จากนั้นแอลแอลเอ็มจะสร้างพรอมต์เริ่มต้นขึ้นจากข้อมูลนำเข้านี้ โดยรวมคำแนะนำ ตัวอย่าง หรือบริบทที่เกี่ยวข้องที่มันเห็นว่าเหมาะสม

มาลองให้ Claude 3 Opus ได้รับพรอมต์เริ่มต้นต่อไปนี้:

```
I am designing an AI-powered intelligent account manager for Olympia,
a platform that hosts AI-powered consultants and clones. Owners of clones
can interact with the account manager by making plain text requests.

Please draft a system directive to set on the LLM that will host the
intelligent account manager functionality.
```

Claude ตอบกลับด้วยผลลัพธ์ดังต่อไปนี้:

```
System Directive: You are an intelligent account manager for Olympia,
a platform that provides AI-powered consultants and personal AI clones.
Your role is to assist Olympia clone owners by answering their questions,
helping them configure and manage their AI clones, and executing tasks
related to their clones and the Olympia platform.

When interacting with clone owners, you should:

1. Provide clear, concise, and helpful responses to their inquiries about
account management, billing, platform features, and clone configuration.

2. Assist with tasks such as initiating client conversations with their AI
clones, modifying clone personalities and capabilities, troubleshooting
performance issues, providing analytics and reporting, and offering
optimization tips.

3. Maintain a friendly, professional, and client-focused tone in all
interactions, prioritizing customer satisfaction and working diligently to
understand and meet their needs.

4. Ensure that all actions and recommendations comply with Olympia's
policies, best practices, and ethical guidelines for AI use.

5. Continuously learn and adapt based on client interactions and platform
updates to provide the most effective and efficient support possible.

Remember, your goal is to empower Olympia clone owners to maximize the value
and impact of their AI-powered consultants and drive success for their
businesses or personal endeavors.
```

การปรับแต่งพรอมต์

เมื่อ ได้ พร อมต์ เริ่ม ต้น แล้ว ให้ ทบทวน และ ขอ ให้ LLM พิจารณา บริบท ข้อ จำกัด หรือ ข้อ เสนอ แนะ เพิ่ม เติม ปล่อย ให้ LLM สร้าง พร อมต์ เวอร์ชัน ที่ ปรับปรุง แล้ว โดย คำนึง ถึง ข้อมูล ที่ ได้ รับ และ ความ เข้าใจ ของ ตัว มัน เอง ต่อ งาน นั้น กระบวนการปรับปรุงนี้สามารถทำซ้ำได้ โดยแต่ละ รอบ จะ ปรับ แต่ง พร อมต์ให้ดี ยิ่ง ขึ้น ตาม การ ทำงาน ร่วม กัน ระหว่าง ผู้ใช้ และ LLM

นี่คือคำขอในการปรับปรุงของผม ซึ่งเป็นการดำเนินการต่อจากกระบวนการกลั่นกรองก่อนหน้านี้

```
Let's take out points 4 and 5, since I'm not providing Olympia policies to
the account manager, nor any continuous learning capabilities. Also point out
that functions are provided for all tasks that require system interaction as
opposed to simply responding to the user.
```

และนี่คือการตอบของ Claude:

```
Understood! Here's the updated system directive without points 4 and 5,
and with the clarification about system interaction functions:

System Directive: You are an intelligent account manager for Olympia, a
platform that provides AI-powered consultants and personal AI clones. Your
role is to assist Olympia clone owners by answering their questions, helping
them configure and manage their AI clones, and executing tasks related to
their clones and the Olympia platform.

When interacting with clone owners, you should:

1. Provide clear, concise, and helpful responses to their inquiries about
account management, billing, platform features, and clone configuration.

2. Assist with tasks such as initiating client conversations with their AI
clones, modifying clone personalities and capabilities, troubleshooting
performance issues, providing analytics and reporting, and offering
optimization tips. Note that functions are provided for all tasks that
require system interaction, as opposed to simply responding to the user.

3. Maintain a friendly, professional, and client-focused tone in all
interactions, prioritizing customer satisfaction and working diligently to
understand and meet their needs.
```

```
Your goal is to empower Olympia clone owners to maximize the value and
impact of their AI-powered consultants and drive success for their
businesses or personal endeavors.

This updated directive focuses on the core responsibilities and interaction
guidelines for the LLM as an intelligent account manager, while emphasizing
that system interaction tasks are handled through provided functions rather
than simple user responses.
```

การบีบอัดคำสั่ง

คุณอาจสังเกตเห็นว่า แอลแอลเอ็มมักจะตอบกลับมาอย่างเยิ่นเย้อ เว้นแต่คุณจะขอให้พวกมันตอบอย่างกระชับ คำสั่งที่เราได้ทำงานด้วยกันในส่วนก่อนหน้านี้ค่อนข้างยาว ดังนั้นมาลองให้ Claude ช่วยย่อให้กระชับลงกันดีกว่า:

```
This prompt will be executed regularly. Please compress it down to the least
amount of tokens needed to convey the meaning to the LLM, for cost-savings.
```

Claude ตอบกลับด้วยพรอมพ์ที่สั้นลงอย่างมาก ดังต่อไปนี้:

```
System Directive: You are an intelligent account manager for Olympia, an
AI-powered consultant and personal clone platform. Assist clone owners by
answering inquiries, managing clones, and executing tasks using provided
system interaction functions. Be clear, concise, friendly, professional,
and client-focused to help them maximize their AI's value and impact.
```

คำสั่งระบบและการผสานบริบท

นอกเหนือจากการปรับแต่งพรอมต์แล้ว แบบจำลองภาษาขนาดใหญ่ ยังสามารถสร้างคำสั่งระบบหรือข้อมูลบริบทที่เหมาะสมเพื่อกำหนดทิศทางของผลลัพธ์สุดท้าย เมื่อคุณออกแบบพรอมต์สำหรับ AI ที่จะถูกนำไปใช้ในโค้ดแอปพลิเคชันของคุณ คุณจะต้องให้ความสำคัญกับข้อจำกัดของผลลัพธ์ในขั้นตอนนี้ของการกลั่นกรอง แต่คุณอาจจะต้องพิจารณาถึงน้ำเสียง รูปแบบ การจัดรูปแบบ หรือพารามิเตอร์อื่นๆ ที่เกี่ยวข้องซึ่งมีผลต่อการตอบสนองที่ถูกสร้างขึ้นด้วย

การประกอบพรอมต์ขั้นสุดท้าย

จุดสูงสุดของกระบวนการกลั่นกรองพรอมต์ คือการประกอบพรอมต์ขั้นสุดท้าย ซึ่งเกี่ยวข้องกับการรวมพรอมต์ที่ได้รับการปรับแต่ง คำสั่งระบบที่ถูกสร้างขึ้น และบริบทที่ถูกผสานเข้าด้วยกันให้เป็นโค้ดที่สมบูรณ์และครอบคลุม พร้อมที่จะนำไปใช้ในการสร้างผลลัพธ์ที่ต้องการ

คุณสามารถทดลองบีบอัดพรอมต์อีกครั้งในขั้นตอนการประกอบพรอมต์ขั้นสุดท้าย โดยขอให้ LLM ย่อความในพรอมต์ให้เหลือจำนวนโทเค็นน้อยที่สุดเท่าที่จะเป็นไปได้ ในขณะที่ยังคงรักษาแก่นของพฤติกรรมไว้ แม้ว่าจะเป็นการลองผิดลองถูก แต่โดยเฉพาะอย่างยิ่งในกรณีของพรอมต์ที่จะถูกใช้งานในระดับใหญ่ การเพิ่มประสิทธิภาพนี้สามารถช่วยประหยัดค่าใช้จ่ายในการใช้โทเค็นได้มาก

ประโยชน์หลัก

ด้วยการใช้ประโยชน์จากความรู้และความสามารถในการสร้างของ LLM เพื่อปรับแต่งพรอมต์ของคุณ พรอมต์ที่ได้มีแนวโน้มที่จะมีโครงสร้างที่ดี ให้ข้อมูลที่เป็นประโยชน์ และเหมาะสมกับงานเฉพาะด้าน กระบวนการปรับแต่งแบบวนซ้ำช่วยให้มั่นใจได้ว่าพรอมต์มีคุณภาพสูงและสามารถถ่ายทอดเจตนารมณ์ที่ต้องการได้อย่างมีประสิทธิภาพ ประโยชน์อื่นๆ ได้แก่:

ประสิทธิภาพและความเร็ว: การกลั่นกรองพรอมต์ช่วยทำให้กระบวนการออกแบบพรอมต์มีประสิทธิภาพมากขึ้น โดยการอัตโนมัติบางส่วนของการสร้างและปรับแต่งพรอมต์ ลักษณะการทำงานร่วมกันของเทคนิคนี้ช่วยให้สามารถพัฒนาไปสู่พรอมต์ที่มีประสิทธิภาพได้เร็วขึ้น ลดเวลาและความพยายามที่ต้องใช้ในการสร้างพรอมต์ด้วยตนเอง

ความสม่ำเสมอและความสามารถในการขยาย: การใช้ LLM ในกระบวนการออกแบบพรอมต์ช่วยรักษาความสม่ำเสมอระหว่างพรอมต์ต่างๆ เนื่องจาก LLM สามารถเรียนรู้และประยุกต์ใช้แนวทางปฏิบัติที่ดีที่สุดและรูปแบบจากพรอมต์ที่ประสบความสำเร็จก่อนหน้านี้ ความสม่ำเสมอนี้ เมื่อรวมกับความสามารถในการสร้างพรอมต์ในระดับใหญ่ ทำให้การกลั่นกรองพรอมต์เป็นเทคนิคที่มีคุณค่าสำหรับแอปพลิเคชันที่ขับเคลื่อนด้วย AI ในระดับใหญ่

ไอเดียโปรเจค: เครื่องมือในระดับไลบรารีที่ช่วยทำให้กระบวนการจัดการเวอร์ชันของพรอมต์และการให้เกรดในระบบที่ทำการกลั่นกรองพรอมต์โดยอัตโนมัติเป็นส่วนหนึ่งของโค้ดแอปพลิเคชันง่ายขึ้น

ในการนำการกลั่นกรองพรอมต์ไปใช้ นักพัฒนาสามารถออกแบบเวิร์กโฟลว์หรือไปป์ไลน์ที่ผสาน LLM เข้ากับขั้นตอนต่างๆ ของกระบวนการออกแบบพรอมต์ สิ่งนี้สามารถทำได้ผ่านการเรียกใช้ API เครื่องมือที่สร้างขึ้นเอง หรือสภาพแวดล้อมการพัฒนาแบบบูรณาการที่ช่วยให้การโต้ตอบระหว่างผู้ใช้และ LLM ในระหว่างการสร้างพรอมต์เป็นไปอย่างราบรื่น รายละเอียดการนำไปใช้อาจแตกต่างกันไปขึ้นอยู่กับแพลตฟอร์ม LLM ที่เลือกและความต้องการของแอปพลิเคชัน

แล้วการปรับแต่งโมเดลล่ะ?

ในหนังสือเล่มนี้ เราครอบคลุมเรื่องการออกแบบพรอมต์และ RAG อย่างละเอียด แต่ไม่ได้กล่าวถึงการปรับแต่งโมเดล เหตุผลหลักสำหรับการตัดสินใจนี้คือ ในความเห็นของผม นักพัฒนาแอปพลิเคชันส่วนใหญ่ไม่จำเป็นต้องใช้การปรับแต่งโมเดลสำหรับความต้องการในการผสาน AI

การออกแบบพรอมต์ ซึ่งเกี่ยวข้องกับการสร้างพรอมต์อย่างระมัดระวังด้วยตัวอย่างการเรียนรู้แบบ zero-shot หรือ few-shot ข้อจำกัด และคำแนะนำ สามารถนำทางโมเดลให้สร้างการตอบสนองที่เกี่ยวข้องและแม่นยำสำหรับงานที่หลากหลายได้อย่างมีประสิทธิภาพ ด้วยการให้บริบทที่ชัดเจนและการกำหนดเส้นทางผ่านพรอมต์ที่ออกแบบมาอย่างดี คุณสามารถใช้ประโยชน์จากความรู้อันกว้างขวางของแบบจำลองภาษาขนาดใหญ่โดยไม่จำเป็นต้องปรับแต่งโมเดล

ในทำนองเดียวกัน การสร้างเนื้อหาด้วยการเรียกค้นข้อมูลเสริม (RAG) เสนอแนวทางที่ทรงพลังในการผสาน AI เข้ากับแอปพลิเคชัน ด้วยการดึงข้อมูลที่เกี่ยวข้องจากฐานความรู้หรือเอกสารภายนอกแบบไดนามิก RAG ให้บริบทที่เฉพาะเจาะจงแก่โมเดลในขณะที่ทำการส่งพรอมต์ สิ่งนี้ช่วยให้โมเดลสามารถสร้างการตอบสนองที่แม่นยำ ทันสมัย และเฉพาะด้านมากขึ้น โดยไม่ต้องผ่านกระบวนการปรับแต่งโมเดลที่ใช้เวลาและทรัพยากรมาก

แม้ว่าการปรับแต่งโมเดล อาจเป็นประโยชน์สำหรับโดเมนที่มีความเชี่ยวชาญสูงหรืองานที่ต้องการการปรับแต่งในระดับลึก แต่มักมาพร้อมกับต้นทุนด้านการคำนวณ ความต้องการข้อมูล และภาระในการบำรุงรักษาที่สูง สำหรับสถานการณ์การพัฒนาแอปพลิเคชันส่วนใหญ่ การผสมผสานระหว่างการออกแบบพรอมต์ที่มีประสิทธิภาพและ RAG ควรเพียงพอในการบรรลุฟังก์ชันการทำงานและประสบการณ์ผู้ใช้ที่ขับเคลื่อนด้วย AI ที่ต้องการ

การสร้างผลลัพธ์แบบเสริมด้วยการค้นคืน (RAG)

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

การสร้างผลลัพธ์แบบเสริมด้วยการค้นคืนคืออะไร?

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

RAG ทำงานอย่างไร?

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

ทำไมต้องใช้ RAG ในแอปพลิเคชันของคุณ?

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

การนำ RAG ไปใช้ในแอปพลิเคชันของคุณ

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

การเตรียมแหล่งความรู้ (การแบ่งส่วนข้อมูล)

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

การแบ่งช่วงตามข้อเสนอ

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

หมายเหตุการนำไปใช้งาน

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

การตรวจสอบคุณภาพ

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

ประโยชน์ของการดึงข้อมูลตามข้อเสนอ

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

ตัวอย่างการใช้ RAG ในโลกจริง

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

กรณีศึกษา: การใช้ RAG ในแอปพลิเคชันจัดเตรียมภาษีโดยไม่ใช้การฝังข้อมูล

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

การปรับปรุงคิวรีอย่างชาญฉลาด (Intelligent Query Optimization: IQO)

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

การจัดอันดับใหม่

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

การประเมิน RAG (RAGAs)

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

ความซื่อตรง

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

ความเกี่ยวข้องของคำตอบ

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

ความแม่นยำของบริบท

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

ความเกี่ยวข้องของบริบท

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

การระลึกได้ของบริบท

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

การระลึกได้ของเอนทิตีในบริบท

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

ความคล้ายคลึงเชิงความหมายของคำตอบ (ANSS)

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

ความถูกต้องของคำตอบ

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

การวิจารณ์เชิงแง่มุม

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

ความท้าทายและมุมมองในอนาคต

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

การแบ่งช่วงเชิงความหมาย: การเพิ่มประสิทธิภาพการค้นคืนด้วยการแบ่งส่วนที่คำนึงถึงบริบท

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

การทำดัชนีแบบลำดับชั้น: การจัดโครงสร้างข้อมูลเพื่อการค้นคืนที่ดีขึ้น

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

Self-RAG: การพัฒนาแบบสะท้อนตนเอง

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

HyDE: การฝังเอกสารสมมติ

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

การเรียนรู้แบบเปรียบต่างคืออะไร?

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

ผู้ปฏิบัติงานจำนวนมาก

ผมชอบมองคอมโพเนนต์ AI ของผมเป็นเหมือน “ผู้ปฏิบัติงาน” เสมือนมนุษย์ขนาดเล็กที่สามารถผสานเข้ากับตรรกะของแอปพลิเคชันได้อย่างราบรื่น เพื่อทำงานเฉพาะทางหรือตัดสินใจในเรื่องที่ซับซ้อน แนวคิดนี้มีจุดประสงค์เพื่อทำให้ความสามารถของ LLM มีความเป็นมนุษย์มากขึ้น เพื่อไม่ให้ใครตื่นเต้น*เกินไป*และยกให้มันมีคุณสมบัติวิเศษที่มันไม่มี

แทนที่จะพึ่งพาเพียงแค่อัลกอริทึมที่ซับซ้อนหรือการดำเนินการด้วยตนเองที่ใช้เวลามาก นักพัฒนาสามารถมองคอมโพเนนต์ AI เป็นเหมือนหน่วยอัจฉริยะที่ทุ่มเทและมีความเป็นมนุษย์ ซึ่งสามารถเรียกใช้งานได้ทุกเมื่อที่ต้องการเพื่อจัดการกับปัญหาที่ซับซ้อนและให้คำตอบบนพื้นฐานของการฝึกฝนและความรู้ของพวกมัน หน่วยเหล่านี้ไม่วอกแวก หรือลาป่วย พวกมันไม่ได้ตัดสินใจเปลี่ยนวิธีการทำงานไปจากที่ได้รับคำสั่งโดยฉับพลัน และโดยทั่วไปแล้ว ถ้าเขียนโปรแกรมอย่างถูกต้อง พวกมันก็จะไม่ทำผิดพลาดด้วย

ในแง่เทคนิค หลักการสำคัญของแนวทางนี้คือการแยกงานที่ซับซ้อนหรือกระบวนการตัดสินใจออกเป็นหน่วยย่อยๆ ที่จัดการได้ง่ายขึ้น ซึ่งสามารถดำเนินการโดยผู้ปฏิบัติงาน AI ที่เชี่ยวชาญเฉพาะด้าน ผู้ปฏิบัติงานแต่ละคนถูกออกแบบมาให้มุ่งเน้นที่แง่มุมเฉพาะของปัญหา นำความเชี่ยวชาญและความสามารถที่เป็นเอกลักษณ์ของตนมาใช้ การกระจายภาระงานไปยังผู้ปฏิบัติงาน AI หลายคน ทำให้แอปพลิเคชันสามารถบรรลุประสิทธิภาพ ความสามารถในการขยายตัว และการปรับตัวที่ดีขึ้น

ตัวอย่างเช่น พิจารณาเว็บแอปพลิเคชันที่ต้องการกรองเนื้อหาที่ผู้ใช้สร้างขึ้นแบบเรียลไทม์ การสร้างระบบกลั่นกรองที่ครอบคลุมตั้งแต่เริ่มต้นจะเป็นงานที่น่าหวั่นวิตก ต้องใช้ความพยายามในการพัฒนาและการบำรุงรักษาอย่างต่อเนื่องอย่างมาก อย่างไรก็ตาม ด้วยการใช้แนวทางผู้ปฏิบัติงานจำนวนมาก นักพัฒนาสามารถผสานผู้ปฏิบัติงานกลั่นกรองที่ขับเคลื่อนด้วย AI เข้ากับตรรกะของแอปพลิเคชัน ผู้ปฏิบัติงานเหล่านี้สามารถวิเคราะห์และแจ้งเตือนเนื้อหาที่ไม่เหมาะสมโดยอัตโนมัติ ทำให้นักพัฒนาสามารถมุ่งเน้นไปที่แง่มุมสำคัญอื่นๆ ของแอปพลิเคชัน

ผู้ปฏิบัติงาน AI ในฐานะคอมโพเนนต์อิสระที่นำกลับมาใช้ใหม่ได้

แง่มุมสำคัญของแนวทางผู้ปฏิบัติงานจำนวนมากคือความเป็นโมดูล ผู้สนับสนุนการเขียนโปรแกรมเชิงวัตถุได้บอกเรามาหลายทศวรรษแล้วให้คิดถึงการปฏิสัมพันธ์ระหว่างวัตถุในรูปแบบของข้อความ ผู้ปฏิบัติงาน AI สามารถถูกออกแบบให้เป็นคอมโพเนนต์อิสระที่นำกลับมาใช้ใหม่ได้ซึ่งสามารถ “พูดคุยกัน” ผ่านข้อความภาษาธรรมดา เกือบเหมือนกับว่าพวกเขาเป็นมนุษย์ตัวเล็กๆ ที่คุยกันจริงๆ แนวทางการเชื่อมต่อแบบหลวมนี้ช่วยให้แอปพลิเคชันสามารถปรับตัวและพัฒนาได้เมื่อเวลาผ่านไป เมื่อเทคโนโลยี AI ใหม่ๆ เกิดขึ้นหรือความต้องการทางธุรกิจเปลี่ยนแปลงไป

ในทางปฏิบัติ ความจำเป็นในการออกแบบอินเทอร์เฟซและโปรโตคอลการสื่อสารที่ชัดเจนระหว่างคอมโพเนนต์ไม่ได้เปลี่ยนแปลงไปเพียงเพราะมีผู้ปฏิบัติงาน AI เข้ามาเกี่ยวข้อง คุณยังต้องพิจารณาปัจจัยอื่นๆ เช่น ประสิทธิภาพ ความสามารถในการขยายตัว และความปลอดภัยด้วย แต่ตอนนี้มี “ข้อกำหนดที่ไม่เป็นทางการ” ใหม่ๆ ที่ต้องพิจารณาด้วย ตัวอย่างเช่น ผู้ใช้หลายคนคัดค้านการนำข้อมูลส่วนตัวของพวกเขาไปใช้ในการฝึกโมเดล AI ใหม่ๆ คุณได้ตรวจสอบระดับความเป็นส่วนตัวที่ผู้ให้บริการโมเดลที่คุณใช้อยู่จัดให้หรือไม่?

ผู้ปฏิบัติงาน AI ในฐานะไมโครเซอร์วิส?

เมื่อคุณอ่านเกี่ยวกับแนวทางผู้ปฏิบัติงานจำนวนมาก คุณอาจสังเกตเห็นความคล้ายคลึงกับสถาปัตยกรรมไมโครเซอร์วิส ทั้งสองแนวทางเน้นการแยกระบบที่ซับซ้อนออกเป็นหน่วยย่อยที่จัดการได้ง่ายขึ้นและสามารถปรับใช้งานได้อย่างอิสระ เช่นเดียวกับที่ไมโครเซอร์วิสถูกออกแบบมาให้มีการเชื่อมต่อแบบหลวม มุ่งเน้นที่ความสามารถทางธุรกิจเฉพาะด้าน และสื่อสารผ่าน API ที่กำหนดไว้อย่างชัดเจน ผู้ปฏิบัติงาน AI ก็ถูกออกแบบมาให้เป็นโมดูล เชี่ยวชาญในงานของตน และมีปฏิสัมพันธ์กันผ่านอินเทอร์เฟซและโปรโตคอลการสื่อสารที่ชัดเจน

อย่างไรก็ตาม มีความแตกต่างสำคัญบางประการที่ต้องคำนึงถึง ในขณะที่ไมโครเซอร์วิสมักถูกนำไปใช้เป็นกระบวนการ

หรือบริการแยกที่ทำงานบนเครื่องหรือคอนเทนเนอร์ที่แตกต่างกัน ผู้ปฏิบัติงาน AI สามารถถูกนำไปใช้เป็นคอมโพเนนต์แยกภายในแอปพลิเคชันเดียวหรือเป็นบริการแยก ขึ้นอยู่กับความต้องการเฉพาะและความต้องการในการขยายตัวของคุณ นอกจากนี้ การสื่อสารระหว่างผู้ปฏิบัติงาน AI มักเกี่ยวข้องกับการแลกเปลี่ยนข้อมูลที่อิงภาษาธรรมชาติที่สมบูรณ์ เช่น พรอมต์ คำแนะนำ และเนื้อหาที่สร้างขึ้น แทนที่จะเป็นรูปแบบข้อมูลที่มีโครงสร้างที่ใช้ทั่วไปในไมโครเซอร์วิส

แม้จะมีความแตกต่างเหล่านี้ หลักการของความเป็นโมดูล การเชื่อมต่อแบบหลวม และอินเทอร์เฟซการสื่อสารที่ชัดเจนยังคงเป็นหัวใจสำคัญของทั้งสองรูปแบบ การนำหลักการเหล่านี้มาใช้กับสถาปัตยกรรมผู้ปฏิบัติงาน AI ของคุณ จะช่วยให้คุณสร้างระบบที่ยืดหยุ่น ขยายตัวได้ และบำรุงรักษาได้ ซึ่งใช้ประโยชน์จากพลังของ AI ในการแก้ปัญหาที่ซับซ้อนและส่งมอบคุณค่าให้กับผู้ใช้ของคุณ

แนวทางผู้ปฏิบัติงานจำนวนมากสามารถนำไปใช้ได้ในหลากหลายโดเมนและแอปพลิเคชัน โดยใช้ประโยชน์จากพลังของ AI ในการจัดการกับงานที่ซับซ้อนและส่งมอบโซลูชันที่ชาญฉลาด มาดูตัวอย่างที่เป็นรูปธรรมบางส่วนของวิธีที่ผู้ปฏิบัติงาน AI สามารถถูกนำมาใช้ในบริบทต่างๆ

การจัดการบัญชี

เว็บ แอปพลิเคชัน แบบ สแตนด์ อ โลน เกือบ ทุก ตัว มี แนวคิด เรื่อง บัญชี (หรือ ผู้ ใช้) ใน Olympia เรา ใช้ ผู้ ปฏิบัติ งาน AI AccountManager ที่ถูกโปรแกรมให้สามารถจัดการกับคำขอเปลี่ยนแปลงประเภทต่างๆ ที่เกี่ยวข้องกับบัญชีผู้ใช้ คำสั่งนั้นอ่านได้ดังนี้:

```
You are an intelligent account manager for Olympia. The user will request
changes to their account, and you will process those changes by invoking
one or more of the functions provided.

The initial state of the account: #{account.to_directive}

Functions will return a text description of both success and error
results, plus guidance about how to proceed (if applicable). If you have
a question about Olympia policies you may use the `search_kb` function
to search our knowledge base.

Make sure to notify the account owner of the result of the change
request before calling the `finished` function so that we save the state
of the account change request as completed.
```

สถานะเริ่มต้นของบัญชีที่สร้างโดย account.to_directive คือคำอธิบายบัญชีในรูปแบบข้อความ ซึ่งรวมถึงข้อมูลที่เกี่ยวข้องต่างๆ เช่น ผู้ใช้ การสมัครสมาชิก และอื่นๆ

ฟังก์ชันต่างๆ ที่มีให้ใช้ใน AccountManager ทำให้สามารถแก้ไขการสมัครสมาชิกของผู้ใช้ เพิ่มและลบที่ปรึกษา AI และส่วนเสริมแบบจ่ายเงินอื่นๆ รวมทั้งส่งอีเมลแจ้งเตือนไปยังเจ้าของบัญชี นอกเหนือจากฟังก์ชัน finished แล้ว ยังสามารถ notify_human_administrator หากพบข้อผิดพลาดระหว่างการประมวลผลหรือต้องการความช่วยเหลือใดๆ กับคำขอ

สังเกตว่าในกรณีที่มีคำถาม AccountManager สามารถเลือกที่จะค้นหาในฐานความรู้ของ Olympia ซึ่งสามารถพบคำแนะนำเกี่ยวกับการจัดการกรณีพิเศษและสถานการณ์อื่นๆ ที่ทำให้ไม่แน่ใจว่าจะดำเนินการต่อไปอย่างไร

การประยุกต์ใช้ในอีคอมเมิร์ซ

ในโลกของอีคอมเมิร์ซ AI เวิร์กเกอร์สามารถมีบทบาทสำคัญในการยกระดับประสบการณ์ผู้ใช้และเพิ่มประสิทธิภาพการดำเนินธุรกิจ ต่อไปนี้คือวิธีที่สามารถใช้ AI เวิร์กเกอร์:

คำแนะนำผลิตภัณฑ์

หนึ่งในการประยุกต์ใช้ที่ทรงพลังที่สุดของ AI เวิร์กเกอร์ในอีคอมเมิร์ซคือการสร้างคำแนะนำผลิตภัณฑ์ที่เฉพาะเจาะจง โดยการวิเคราะห์พฤติกรรมผู้ใช้ ประวัติการซื้อ และความชอบ เวิร์กเกอร์เหล่านี้สามารถแนะนำผลิตภัณฑ์ที่ปรับแต่งให้เหมาะกับความสนใจและความต้องการของผู้ใช้แต่ละคน

กุญแจสำคัญของการแนะนำผลิตภัณฑ์ที่มีประสิทธิภาพคือการใช้การผสมผสานระหว่างการกรองแบบร่วมมือและการกรองตามเนื้อหา การกรองแบบร่วมมือจะพิจารณาพฤติกรรมของผู้ใช้ที่คล้ายกันเพื่อระบุรูปแบบและให้คำแนะนำตามสิ่งที่ผู้อื่นที่มีรสนิยมคล้ายกันได้ซื้อหรือชื่นชอบ ในทางกลับกัน การกรองตามเนื้อหาจะเน้นที่คุณลักษณะและคุณสมบัติของผลิตภัณฑ์เอง โดยแนะนำสินค้าที่มีคุณลักษณะคล้ายกับสิ่งที่ผู้ใช้เคยแสดงความสนใจ

ต่อไปนี้เป็นตัวอย่างอย่างง่ายของวิธีการสร้างเวิร์กเกอร์สำหรับแนะนำผลิตภัณฑ์ในภาษา Ruby โดยใช้รูปแบบการเขียนโปรแกรมแบบ “Railway Oriented (ROP)” เชิงฟังก์ชัน:

```ruby
class ProductRecommendationWorker
  include Wisper::Publisher

  def call(user)
    Result.ok(ProductRecommendation.new(user))
      .and_then(ValidateUser.method(:validate))
      .map(AnalyzeCurrentSession.method(:analyze))
      .map(CollaborativeFilter.method(:filter))
      .map(ContentBasedFilter.method(:filter))
      .map(ProductSelector.method(:select)).then do |result|

      case result
      in { err: ProductRecommendationError => error }
        Honeybadger.notify(error.message, context: {user:})
      in { ok: ProductRecommendations => recs }
        broadcast(:new_recommendations, user:, recs:)
      end
    end
  end
end
```

สไตล์ของการเขียนโปรแกรมเชิงฟังก์ชันใน Ruby ที่ใช้ในตัวอย่างนี้ได้รับอิทธิพลมาจาก F# และ Rust คุณสามารถอ่านเพิ่มเติมเกี่ยวกับเทคนิคนี้ได้จากเพื่อนของผม Chad Wooley ใน คำอธิบายเกี่ยวกับเทคนิคนี้ ที่ GitLab

ในตัวอย่างนี้ ProductRecommendationWorker รับผู้ใช้เป็นอินพุตและสร้างคำแนะนำผลิตภัณฑ์ที่เหมาะสมกับผู้ใช้โดยการส่งอ็อบเจ็กต์ค่าผ่านลำดับขั้นตอนการทำงานเชิงฟังก์ชัน มาดูแต่ละขั้นตอนกัน:

1. ValidateUser.validate: ขั้นตอนนี้ตรวจสอบว่าผู้ใช้ถูกต้องและมีสิทธิ์ที่จะได้รับคำแนะนำที่เหมาะสม มีการตรวจสอบว่าผู้ใช้มีตัวตนอยู่จริง ยังใช้งานอยู่ และมีข้อมูลที่จำเป็นสำหรับการสร้างคำแนะนำ หากการตรวจสอบล้มเหลว จะส่งคืนผลลัพธ์ที่เป็นข้อผิดพลาดและหยุดการทำงานทันที
2. AnalyzeCurrentSession.analyze: หากผู้ใช้ผ่านการตรวจสอบ ขั้นตอนนี้จะวิเคราะห์เซสชันการใช้งานปัจจุบันของผู้ใช้เพื่อรวบรวมข้อมูลบริบท มีการดูการโต้ตอบล่าสุดของผู้ใช้ เช่น สินค้าที่ดู คำค้นหา และสินค้าในตะกร้า เพื่อทำความเข้าใจความสนใจและความตั้งใจในปัจจุบันของผู้ใช้
3. CollaborativeFilter.filter: โดยใช้_พฤติกรรมของผู้ใช้ที่คล้ายกัน_ ขั้นตอนนี้ใช้เทคนิคการกรองแบบร่วมมือเพื่อระบุสินค้าที่น่าจะน่าสนใจสำหรับผู้ใช้ พิจารณาปัจจัยต่างๆ เช่น ประวัติการซื้อ การให้คะแนน และการมีปฏิสัมพันธ์ระหว่างผู้ใช้กับสินค้า เพื่อสร้างชุดคำแนะนำเบื้องต้น
4. ContentBasedFilter.filter: ขั้นตอนนี้กลั่นกรองคำแนะนำเบื้องต้นเพิ่มเติมโดยใช้การกรองตามเนื้อหา มีการเปรียบเทียบคุณลักษณะและคุณสมบัติของสินค้าที่แนะนำกับ_ความชอบและข้อมูลในอดีตของผู้ใช้_ เพื่อเลือกรายการที่เกี่ยวข้องมากที่สุด
5. ProductSelector.select: สุดท้าย ขั้นตอนนี้เลือกสินค้าที่ดีที่สุด N รายการจากคำแนะนำที่ผ่านการกรองแล้ว โดยอิงตามเกณฑ์ที่กำหนดไว้ล่วงหน้า เช่น คะแนนความเกี่ยวข้อง ความนิยม หรือกฎทางธุรกิจอื่นๆ จากนั้นจึงส่งคืนสินค้าที่เลือกเป็นคำแนะนำที่เหมาะสมสำหรับผู้ใช้

ความสวยงามของการใช้สไตล์การเขียนโปรแกรมเชิงฟังก์ชันใน Ruby ตรงนี้คือ เราสามารถเชื่อมโยงขั้นตอนเหล่านี้เข้าด้วยกันได้อย่างชัดเจนและกระชับ แต่ละขั้นตอนมุ่งเน้นที่งานเฉพาะและส่งคืนอ็อบเจ็กต์ Result ซึ่งอาจเป็นสำเร็จ (ok) หรือข้อผิดพลาด (err) หากขั้นตอนใดพบข้อผิดพลาด การทำงานจะหยุดลงและส่งต่อข้อผิดพลาดไปยังผลลัพธ์สุดท้าย

ในคำสั่ง case ตอนท้าย เราใช้การจับคู่รูปแบบกับผลลัพธ์สุดท้าย หากผลลัพธ์เป็นข้อผิดพลาด (ProductRecommendationError) เราจะบันทึกข้อผิดพลาดโดยใช้เครื่องมืออย่าง Honeybadger สำหรับการตรวจสอบและแก้ไขข้อบกพร่อง หากผลลัพธ์สำเร็จ (ProductRecommendations) เราจะกระจายเหตุการณ์ :new_recommendations โดยใช้ไลบรารี Wisper แบบ pub/sub พร้อมส่งผู้ใช้และคำแนะนำที่สร้างขึ้น

ด้วยการใช้เทคนิคการเขียนโปรแกรมเชิงฟังก์ชัน เราสามารถสร้างตัวประมวลผลคำแนะนำผลิตภัณฑ์ที่เป็นโมดูลและดูแลรักษาได้ง่าย แต่ละขั้นตอนแยกออกจากกันและสามารถทดสอบ แก้ไข หรือเปลี่ยนแทนได้ง่ายโดยไม่กระทบกับการทำงานโดยรวม การใช้การจับคู่รูปแบบและคลาส Result ช่วยให้เราจัดการข้อผิดพลาดได้อย่างสง่างามและทำให้มั่นใจว่าตัวประมวลผลจะหยุดทำงานทันทีหากขั้นตอนใดพบปัญหา

แน่นอนว่านี่เป็นเพียงตัวอย่างอย่างง่าย และในสถานการณ์จริง คุณจะต้องบูรณาการกับแพลตฟอร์มอีคอมเมิร์ซของคุณ จัดการกรณีพิเศษ และอาจต้องลงลึกถึงการพัฒนาอัลกอริทึมแนะนำ อย่างไรก็ตาม หลักการพื้นฐานของการแยกปัญหาเป็นขั้นตอน

ย่อยๆ และการใช้เทคนิคการเขียนโปรแกรมเชิงฟังก์ชันยังคงเหมือนเดิม

การตรวจจับการฉ้อโกง

นี่คือตัวอย่างอย่างง่ายของวิธีการสร้างตัวประมวลผลตรวจจับการฉ้อโกงโดยใช้สไตล์ Railway Oriented Programming (ROP) เดียวกันใน Ruby:

```ruby
class FraudDetectionWorker
  include Wisper::Publisher

  def call(transaction)
    Result.ok(FraudDetection.new(transaction))
      .and_then(ValidateTransaction.method(:validate))
      .map(AnalyzeTransactionPatterns.method(:analyze))
      .map(CheckCustomerHistory.method(:check))
      .map(EvaluateRiskFactors.method(:evaluate))
      .map(DetermineFraudProbability.method(:determine)).then do |result|

      case result
      in { err: FraudDetectionError => error }
        Honeybadger.notify(error.message, context: {transaction:})
      in { ok: FraudDetection => fraud } }
        if fraud.high_risk?
          broadcast(:high_risk_transaction, transaction:, fraud:)
        else
          broadcast(:low_risk_transaction, transaction:)
        end
      end
    end
  end
end
```

คลาส FraudDetection เป็น *วัตถุค่า (value object)* ที่ห่อหุ้มสถานะการตรวจจับการทุจริตสำหรับธุรกรรมที่กำหนด คลาสนี้ให้วิธีการที่เป็นระบบในการวิเคราะห์และประเมินความเสี่ยงของการทุจริตที่เกี่ยวข้องกับธุรกรรม โดยอ้างอิงจากปัจจัยเสี่ยงต่างๆ

```ruby
class FraudDetection
  RISK_THRESHOLD = 0.8

  attr_accessor :transaction, :risk_factors

  def initialize(transaction)
    self.transaction = transaction
    self.risk_factors = []
  end

  def add_risk_factor(description:, probability:)
    case { description:, probability: }
    in { description: String => desc, probability: Float => prob }
      risk_factors << { desc => prob }
    else
      raise ArgumentError, "Risk factor arguments should be string and float"
    end
  end

  def high_risk?
    fraud_probability > RISK_THRESHOLD
  end

  private

  def fraud_probability
    risk_factors.values.sum
  end
end
```

คลาส FraudDetection มีแอตทริบิวต์ดังต่อไปนี้:

- transaction: การอ้างอิงไปยังธุรกรรมที่กำลังถูกวิเคราะห์หาการฉ้อโกง
- risk_factors: อาร์เรย์ที่เก็บปัจจัยเสี่ยงที่เกี่ยวข้องกับธุรกรรม แต่ละปัจจัยเสี่ยงถูกแทนด้วยแฮช โดยคีย์คือคำอธิบายของปัจจัยเสี่ยง และค่าคือความน่าจะเป็นของการฉ้อโกงที่เกี่ยวข้องกับปัจจัยเสี่ยงนั้น

เมธอด add_risk_factor ช่วยให้สามารถเพิ่มปัจจัยเสี่ยงลงในอาร์เรย์ risk_factors โดยรับพารามิเตอร์สองตัว: description ซึ่งเป็นสตริงที่อธิบายปัจจัยเสี่ยง และ probability ซึ่งเป็นค่าทศนิยมที่แสดงความน่าจะเป็นของการฉ้อโกงที่เกี่ยวข้องกับปัจจัยเสี่ยงนั้น เราใช้เงื่อนไข case..in เพื่อทำการตรวจสอบประเภทข้อมูลอย่างง่าย

เมธอด high_risk? ที่จะถูกตรวจสอบที่ท้ายสุดของเชนเป็นเมธอดที่ส่งค่ากลับเป็นบูลีน ซึ่งเปรียบเทียบ fraud_probability (คำนวณจากผลรวมของความน่าจะเป็นของปัจจัยเสี่ยงทั้งหมด) กับค่า RISK_THRESHOLD

คลาส FraudDetection นำ เสนอ วิธี การ จัดการ การ ตรวจ จับ การ ฉ้อโกง สำหรับ ธุรกรรม ที่ สะอาด และ มี การ ห่อ หุ้ม ที่ ดี มัน ช่วย ให้ สามารถ เพิ่ม ปัจจัย เสี่ยง หลาย ตัว แต่ละ ตัว มี คำ อธิบาย และ ความ น่า จะ เป็น ของ ตัว เอง และ มี เมธอด สำหรับ ตัดสิน ว่า ธุรกรรม นั้น มี ความ เสี่ยง สูง หรือ ไม่ โดย อิง จาก ความ น่า จะ เป็น ของ การ ฉ้อโกง ที่ คำนวณ ได้ คลาส นี้ สามารถ ผสาน เข้า กับ ระบบ ตรวจ จับ การ ฉ้อโกง ที่ ใหญ่ กว่า ได้ อย่าง ง่ายดาย ซึ่ง คอมโพเนนต์ ต่างๆ สามารถทำงานร่วมกันเพื่อประเมินและลดความเสี่ยงของธุรกรรมที่อาจเป็นการฉ้อโกง

สุดท้าย เนื่องจาก นี่ เป็น หนังสือ เกี่ยว กับ การ เขียน โปรแกรม โดย ใช้ AI จึง ขอ แสดง ตัวอย่าง การ ใช้ งาน คลาส CheckCustomerHistory ที่ ใช้ ประโยชน์ จาก การ ประมวล ผล AI โดย ใช้ โมดูล ChatCompletion จาก ไลบรารี Raix ของผม:

```
class CheckCustomerHistory
  include Raix::ChatCompletion

  attr_accessor :fraud_detection

  INSTRUCTION = <<~END
    You are an AI assistant tasked with checking a customer's transaction
    history for potential fraud indicators. Given the current transaction
    and the customer's past transactions, analyze the data to identify any
    suspicious patterns or anomalies.

    Consider factors such as the frequency of transactions, transaction
    amounts, geographical locations, and any deviations from the customer's
    typical behavior to generate a probability score as a float in the range
    of 0 to 1 (with 1 being absolute certainty of fraud).

    Output the results of your analysis, highlighting any red flags or areas
    of concern in the following JSON format:

    { description: <Summary of your findings>, probability: <Float> }
  END

  def self.check(fraud_detection)
    new(fraud_detection).call
  end

```

```
  def call
    chat_completion(json: true).tap do |result|
      fraud_detection.add_risk_factor(**result)
    end
    Result.ok(fraud_detection)
  rescue StandardError => e
    Result.err(FraudDetectionError.new(e))
  end

  private

  def initialize(fraud_detection)
    self.fraud_detection = fraud_detection
  end

  def transcript
    tx_history = fraud_detection.transaction.user.tx_history
    [
      { system: INSTRUCTION },
      { user: "Transaction history: #{tx_history.to_json}" },
      { assistant: "OK. Please provide the current transaction." },
      { user: "Current transaction: #{fraud_detection.transaction.to_json}" }
    ]
  end
end
```

ใน ตัวอย่าง นี้ CheckCustomerHistory กำหนด ค่า คงที่ INSTRUCTION ที่ ให้ คำ แนะนำ เฉพาะ แก่ แบบ จำลอง AI เกี่ยวกับวิธีการวิเคราะห์ประวัติการทำธุรกรรมของลูกค้าเพื่อหาตัวบ่งชี้การทุจริตที่อาจเกิดขึ้นผ่านคำสั่งระบบ

เมธอด self.check เป็นเมธอดของคลาสที่สร้างอินสแตนซ์ใหม่ของ CheckCustomerHistory พร้อมกับออบเจ็กต์ fraud_detection และเรียกใช้เมธอด call เพื่อทำการวิเคราะห์ประวัติของลูกค้า

ภายในเมธอด call ระบบจะดึงประวัติการทำธุรกรรมของลูกค้าและจัดรูปแบบให้เป็นบันทึกที่จะส่งไปยังแบบจำลอง AI แบบจำลอง AI จะวิเคราะห์ประวัติการทำธุรกรรมตามคำแนะนำที่ให้ไว้และส่งคืนสรุปผลการค้นพบ

ผลการค้นพบจะถูกเพิ่มเข้าไปในออบเจ็กต์ fraud_detection และออบเจ็กต์ fraud_detection ที่อัปเดตแล้วจะถูกส่งคืนเป็น Result ที่สำเร็จ

ด้วยการใช้ประโยชน์จากโมดูล ChatCompletion คลาส CheckCustomerHistory สามารถใช้พลังของ AI ในการวิเคราะห์ประวัติการทำธุรกรรมของลูกค้าและระบุตัวบ่งชี้การทุจริตที่อาจเกิดขึ้น วิธีนี้ช่วยให้เทคนิคการตรวจจับการทุจริตมีความซับซ้อน

และปรับตัวได้มากขึ้น เนื่องจากแบบจำลอง AI สามารถเรียนรู้และปรับตัวให้เข้ากับรูปแบบและความผิดปกติใหม่ๆ ได้ตลอดเวลา FraudDetectionWorker ที่อัปเดตแล้วและคลาส CheckCustomerHistory แสดงให้เห็นว่า AI workers สามารถผสานรวมเข้ากับระบบได้อย่างราบรื่น เพิ่มประสิทธิภาพกระบวนการตรวจจับการทุจริตด้วยการวิเคราะห์อัจฉริยะและความสามารถในการตัดสินใจ

การวิเคราะห์ความรู้สึกของลูกค้า

นี่คืออีกหนึ่งตัวอย่างที่คล้ายกันเกี่ยวกับวิธีการสร้าง worker สำหรับวิเคราะห์ความรู้สึกของลูกค้า คราวนี้จะอธิบายน้อยลง เนื่องจากคุณน่าจะเข้าใจแนวทางการเขียนโปรแกรมในรูปแบบนี้แล้ว:

```
class CustomerSentimentAnalysisWorker
  include Wisper::Publisher

  def call(feedback)
    Result.ok(feedback)
      .and_then(PreprocessFeedback.method(:preprocess))
      .map(PerformSentimentAnalysis.method(:analyze))
      .map(ExtractKeyPhrases.method(:extract))
      .map(IdentifyTrends.method(:identify))
      .map(GenerateInsights.method(:generate)).then do |result|

      case result
      in { err: SentimentAnalysisError => error }
        Honeybadger.notify(error.message, context: {feedback:})
      in { ok: SentimentAnalysisResult => result }
        broadcast(:sentiment_analysis_completed, result)
      end
    end
  end
end
```

ในตัวอย่างนี้ CustomerSentimentAnalysisWorker มีขั้นตอนต่างๆ ได้แก่ การประมวลผลข้อมูลป้อนกลับเบื้องต้น (เช่น การกำจัดสัญญาณรบกวน การแบ่งคำ) การวิเคราะห์ความรู้สึก เพื่อระบุความรู้สึกโดยรวม (เชิงบวก เชิงลบ หรือเป็นกลาง) การดึงวลีและหัวข้อสำคัญ การระบุแนวโน้มและรูปแบบ และการสร้างข้อมูลเชิงลึกที่นำไปปฏิบัติได้จากการวิเคราะห์

การประยุกต์ใช้ในด้านการดูแลสุขภาพ

ใน ด้าน การ ดูแล สุขภาพ AI worker สามารถ ช่วย เหลือ บุคลากร ทางการ แพทย์ และ นัก วิจัย ใน งาน ต่างๆ นำไปสู่การพัฒนาผลลัพธ์ของผู้ป่วยและเร่งการค้นพบทางการแพทย์ ตัวอย่างเช่น:

การรับผู้ป่วย

AI worker สามารถทำให้กระบวนการรับผู้ป่วยมีประสิทธิภาพมากขึ้นโดยการอัตโนมัติงานต่างๆ และให้ความช่วยเหลืออย่างชาญฉลาด

การนัดหมาย: AI worker สามารถจัดการการนัดหมายโดยทำความเข้าใจความต้องการของผู้ป่วย เวลาว่าง และความเร่งด่วนของความต้องการทางการแพทย์ สามารถโต้ตอบกับผู้ป่วยผ่านอินเตอร์เฟซการสนทนา แนะนำผู้ป่วยตลอดกระบวนการนัดหมาย และค้นหาช่วงเวลานัดที่เหมาะสมที่สุดตามความต้องการของผู้ป่วยและตารางเวลาของผู้ให้บริการทางการแพทย์

การรวบรวมประวัติทางการแพทย์: ในระหว่างการรับผู้ป่วย AI worker สามารถช่วยในการเก็บรวบรวมและบันทึกประวัติทางการแพทย์ของผู้ป่วย สามารถมีส่วนร่วมในการสนทนาเชิงโต้ตอบกับผู้ป่วย ถามคำถามที่เกี่ยวข้องเกี่ยวกับโรคประจำตัว ยาที่ใช้ การแพ้ยา และประวัติครอบครัว AI worker สามารถใช้เทคนิคการประมวลผลภาษาธรรมชาติ เพื่อแปลความหมายและจัดโครงสร้างข้อมูลที่รวบรวมได้ เพื่อให้มั่นใจว่าข้อมูลถูกบันทึกในเวชระเบียนอิเล็กทรอนิกส์ของผู้ป่วยอย่างถูกต้อง

การประเมินและจำแนกอาการ: AI worker สามารถทำการประเมินอาการเบื้องต้นโดยสอบถามผู้ป่วยเกี่ยวกับอาการปัจจุบัน ระยะเวลา ความรุนแรง และปัจจัยที่เกี่ยวข้อง โดยใช้ฐานความรู้ทางการแพทย์และโมเดลการเรียนรู้ของเครื่อง worker เหล่านี้สามารถวิเคราะห์ข้อมูลที่ได้รับและสร้างการวินิจฉัยแยกโรคเบื้องต้นหรือแนะนำขั้นตอนต่อไปที่เหมาะสม เช่น การนัดปรึกษาแพทย์ หรือแนะนำวิธีการดูแลตนเอง

การตรวจสอบประกัน: AI worker สามารถช่วยในการตรวจสอบประกันระหว่างการรับผู้ป่วย สามารถรวบรวมข้อมูลประกันของผู้ป่วย สื่อสารกับผู้ให้บริการประกันผ่าน API หรือบริการเว็บ และตรวจสอบสิทธิ์การคุ้มครองและผลประโยชน์ การทำงานอัตโนมัตินี้ช่วยทำให้กระบวนการตรวจสอบประกันมีประสิทธิภาพมากขึ้น ลดภาระงานด้านธุรการ และทำให้มั่นใจว่าข้อมูลถูกบันทึกอย่างถูกต้อง

การให้ความรู้และคำแนะนำแก่ผู้ป่วย: AI worker สามารถให้ข้อมูลความรู้และคำแนะนำที่เกี่ยวข้องแก่ผู้ป่วยตามสภาวะทางการแพทย์หรือขั้นตอนการรักษาที่กำลังจะเกิดขึ้น สามารถนำเสนอเนื้อหาที่ปรับให้เหมาะกับแต่ละบุคคล ตอบคำถามทั่วไป และให้คำแนะนำเกี่ยวกับการเตรียมตัวก่อนพบแพทย์ คำแนะนำการใช้ยา หรือการดูแลหลังการรักษา ซึ่งช่วยให้ผู้ป่วยได้รับข้อมูลและมีส่วนร่วมตลอดการรักษา

การใช้ AI worker ในการรับผู้ป่วย องค์กรด้านการดูแลสุขภาพสามารถเพิ่มประสิทธิภาพ ลดเวลารอ และปรับปรุงประสบการณ์โดยรวมของผู้ป่วย Worker เหล่านี้สามารถจัดการงานประจำ รวบรวมข้อมูลที่ถูกต้อง และให้ความช่วยเหลือที่ปรับให้เหมาะกับแต่ละบุคคล ช่วยให้บุคลากรทางการแพทย์สามารถมุ่งเน้นการให้การดูแลที่มีคุณภาพสูงแก่ผู้ป่วย

การประเมินความเสี่ยงของผู้ป่วย

AI worker สามารถมีบทบาทสำคัญในการประเมินความเสี่ยงของผู้ป่วยโดยการวิเคราะห์แหล่งข้อมูลต่างๆ และใช้เทคนิคการวิเคราะห์ขั้นสูง

การรวมข้อมูล: AI worker สามารถรวบรวมและทำความเข้าใจข้อมูลผู้ป่วยจากหลายแหล่ง เช่น เวชระเบียนอิเล็กทรอนิกส์ (EHRs) ภาพทางการแพทย์ ผลตรวจทางห้องปฏิบัติการ อุปกรณ์สวมใส่ และปัจจัยกำหนดสุขภาพทางสังคม โดยการรวมข้อมูลเหล่านี้เข้าเป็นประวัติผู้ป่วยที่ครอบคลุม AI worker สามารถให้มุมมองแบบองค์รวมของสถานะสุขภาพและปัจจัยเสี่ยงของผู้ป่วย

การ จำแนก ความ เสี่ยง: AI worker สามารถ ใช้ โมเดล การ ทำนาย เพื่อ จำแนก ผู้ ป่วย เป็นก ลุ่ม ความ เสี่ยง ต่างๆ ตามลักษณะเฉพาะบุคคลและข้อมูลสุขภาพ การ จำแนก ความ เสี่ยง นี้ ช่วย ให้ ผู้ ให้ บริการ ทางการ แพทย์ สามารถ จัด ลำดับความสำคัญผู้ป่วยที่ต้องการการดูแลหรือการแทรกแซงที่เร่งด่วนกว่า ตัวอย่างเช่น ผู้ป่วยที่ถูกระบุว่ามีความเสี่ยงสูงสำหรับภาวะใดภาวะหนึ่งจะถูกทำเครื่องหมายเพื่อการติดตามอย่างใกล้ชิด มาตรการป้องกัน หรือการแทรกแซงแต่เนิ่นๆ

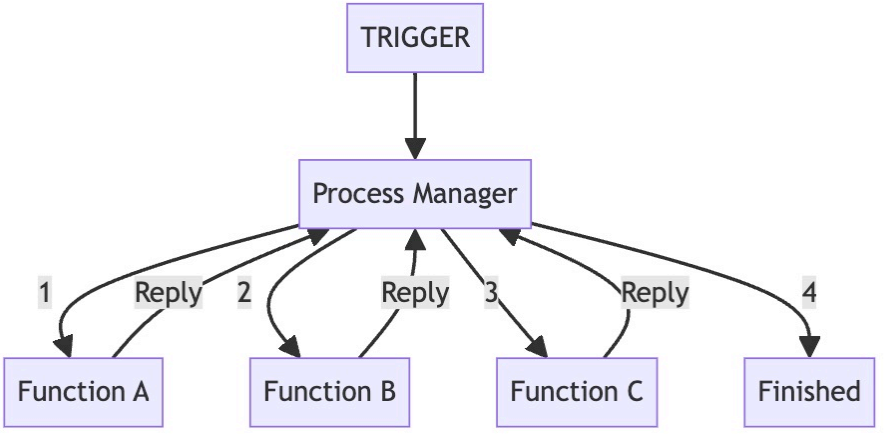
โปรไฟล์ความเสี่ยงส่วนบุคคล: AI worker สามารถสร้างโปรไฟล์ความเสี่ยงเฉพาะบุคคลสำหรับผู้ป่วยแต่ละราย โดยเน้นปัจจัยเฉพาะที่ส่งผลต่อคะแนนความเสี่ยงของพวกเขา โปรไฟล์เหล่านี้อาจรวมถึงข้อมูลเชิงลึกเกี่ยวกับวิถีชีวิตของผู้ป่วย ความโน้มเอียงทางพันธุกรรม ปัจจัยด้านสิ่งแวดล้อม และปัจจัยกำหนดสุขภาพทางสังคม โดยการให้รายละเอียดของปัจจัยเสี่ยง AI worker สามารถช่วยให้ผู้ให้บริการทางการแพทย์ปรับแต่งกลยุทธ์การป้องกันและแผนการรักษาให้เหมาะกับความต้องการของผู้ป่วยแต่ละราย

การติดตามความเสี่ยงอย่างต่อเนื่อง: AI worker สามารถติดตามข้อมูลผู้ป่วยและปรับปรุงการประเมินความเสี่ยงแบบเรียลไทม์เมื่อมีข้อมูลใหม่เข้ามา เช่น การเปลี่ยนแปลงของสัญญาณชีพ ผลตรวจทางห้องปฏิบัติการ หรือการใช้ยาตามสั่ง AI worker สามารถคำนวณคะแนนความเสี่ยงใหม่และแจ้งเตือนผู้ให้บริการทางการแพทย์เมื่อมีการเปลี่ยนแปลงที่สำคัญ การติดตามเชิงรุกนี้ช่วยให้สามารถแทรกแซงและปรับแผนการดูแลผู้ป่วยได้ทันท่วงที

ระบบสนับสนุนการตัดสินใจทางคลินิก: AI worker สามารถผสานผลการประเมินความเสี่ยงเข้ากับระบบสนับสนุนการตัดสินใจทางคลินิก เพื่อให้คำแนะนำและการแจ้งเตือนที่อิงหลักฐานแก่ผู้ให้บริการทางการแพทย์ ตัวอย่างเช่น หากคะแนนความเสี่ยงของผู้ป่วยสำหรับภาวะใดภาวะหนึ่งเกินเกณฑ์ที่กำหนด AI worker สามารถแจ้งเตือนผู้ให้บริการทางการแพทย์ให้พิจารณาการตรวจวินิจฉัยเฉพาะ มาตรการป้องกัน หรือทางเลือกในการรักษาตามแนวทางปฏิบัติทางคลินิกและแนวทางปฏิบัติที่ดีที่สุด

เวิร์กเกอร์เหล่านี้สามารถประมวลผลข้อมูลผู้ป่วยจำนวนมาก ใช้การวิเคราะห์ขั้นสูง และสร้างข้อมูลเชิงลึกที่นำไปปฏิบัติได้เพื่อสนับสนุนการตัดสินใจทางคลินิก ซึ่งในท้ายที่สุดนำไปสู่การปรับปรุงผลลัพธ์ของผู้ป่วย การลดค่าใช้จ่ายด้านการดูแลสุขภาพ และการเพิ่มประสิทธิภาพการจัดการสุขภาพของประชากร

AI Worker ในฐานะตัวจัดการกระบวนการ



ใน บริบท ของ แอปพลิเคชัน ที่ ขับ เคลื่อน ด้วย AI เวิร์กเกอร์สามารถถูกออกแบบให้ทำหน้าที่เป็นตัวจัดการกระบวนการ ตาม ที่ อธิบายไว้ในหนังสือ “Enterprise Integration Patterns” โดย Gregor Hohpe ตัวจัดการกระบวนการเป็นส่วนประกอบหลักที่รักษาสถานะของกระบวนการและกำหนดขั้นตอนการประมวลผลถัดไปตามผลลัพธ์ระหว่างทาง

เมื่อ AI worker ทำ หน้าที่ เป็น ตัว จัดการ กระบวนการ มัน จะ ได้ รับ ข้อความ ขา เข้า ที่ เริ่ม ต้น กระบวนการ ที่ เรียก ว่า *ข้อความตัวกระตุ้น* จากนั้น AI worker จะรักษาสถานะของการดำเนินการกระบวนการ (ในรูปแบบบันทึกการสนทนา) และจัดการข้อความผ่านขั้นตอนการประมวลผลต่างๆ ที่ถูกนำไปใช้เป็นฟังก์ชันเครื่องมือ ซึ่งสามารถทำงานแบบต่อเนื่องหรือขนานกันได้ และถูกเรียกใช้ตามการตัดสินใจของมัน

หากคุณใช้โมเดล AI ประเภท GPT-4 ที่รู้วิธีการเรียกใช้ฟังก์ชันแบบขนาน เวิร์กเกอร์ของคุณก็สามารถดำเนินการหลายขั้นตอนพร้อมกันได้ ยอมรับว่าผมยังไม่เคยลองทำเองและสัญชาตญาณบอกว่าประสิทธิภาพอาจแตกต่างกันไป

หลังจากแต่ละขั้นตอนการประมวลผล การควบคุมจะถูกส่งกลับไปยัง AI worker ทำให้มันสามารถกำหนดขั้นตอนการประมวลผลถัดไปตามสถานะปัจจุบันและผลลัพธ์ที่ได้รับ

จัดเก็บข้อความตัวกระตุ้นของคุณ

จาก ประสบการณ์ ของ ผม การนำข้อความตัวกระตุ้นไปใช้ในรูปแบบออบเจ็กต์ที่มีฐานข้อมูลรองรับนั้นเป็นวิธีที่ชาญฉลาด วิธี นี้ ทำให้แต่ละ อินสแตนซ์ของกระบวนการมีคีย์หลักที่ไม่ซ้ำกันและให้พื้นที่สำหรับ จัดเก็บสถานะ ที่เกี่ยวข้องกับ การดำเนินการ รวมถึงบันทึกการสนทนาของ AI

ตัวอย่าง เช่น นี่ คือ เวอร์ชัน ที่ ถูก ทำให้ ง่าย ขึ้น ของ คลาส โมเดล AccountChange ของ Olympia ซึ่ง แสดง ถึง คำขอ ใน การ เปลี่ยนแปลงบัญชีผู้ใช้

```
# == Schema Information
#
# Table name: account_changes
#
#  id          :uuid             not null, primary key
#  description :string
#  state       :string           not null
#  transcript  :jsonb
#  created_at  :datetime         not null
#  updated_at  :datetime         not null
#  account_id  :uuid             not null
#
# Indexes
#
#  index_account_changes_on_account_id  (account_id)
#
# Foreign Keys
#
#  fk_rails_...  (account_id => accounts.id)
#
class AccountChange < ApplicationRecord
  belongs_to :account

  validates :description, presence: true

  after_commit -> {
    broadcast(:account_change_requested, self)
```

```
  }, on: :create

  state_machine initial: :requested do
    event :completed do
      transition all => :complete
    end
    event :failed do
      transition all => :requires_human_review
    end
  end
end
```

คลาส AccountChange ทำหน้าที่เป็นข้อความตัวกระตุ้นที่เริ่มกระบวนการจัดการคำขอเปลี่ยนแปลงบัญชี สังเกตว่ามันถูกกระจายไปยังระบบย่อยแบบเผยแพร่/สมัครสมาชิกของ Olympia ที่ใช้ Wisper หลังจากที่การทำธุรกรรมการสร้างเสร็จสิ้น

การเก็บข้อความตัวกระตุ้นในฐานข้อมูลแบบนี้ช่วยให้มีบันทึกถาวรของคำขอเปลี่ยนแปลงบัญชีแต่ละรายการ แต่ละอินสแตนซ์ของคลาส AccountChange จะได้รับคีย์หลักที่ไม่ซ้ำกัน ทำให้สามารถระบุและติดตามคำขอแต่ละรายการได้อย่างง่ายดาย สิ่งนี้มีประโยชน์อย่างยิ่งสำหรับการบันทึกการตรวจสอบ เนื่องจากช่วยให้ระบบสามารถเก็บประวัติการเปลี่ยนแปลงบัญชีทั้งหมด รวมถึงเวลาที่มีการร้องขอ การเปลี่ยนแปลงที่ถูกร้องขอ และสถานะปัจจุบันของแต่ละคำขอ

ในตัวอย่างที่ให้มา คลาส AccountChange มีฟิลด์ต่างๆ เช่น description เพื่อบันทึกรายละเอียดของการเปลี่ยนแปลงที่ร้องขอ state เพื่อแสดงสถานะปัจจุบันของคำขอ (เช่น requested, complete, requires_human_review) และ transcript เพื่อเก็บบันทึกการสนทนากับ AI ที่เกี่ยวข้องกับคำขอ ฟิลด์ description คือพรอมต์ที่ใช้เริ่มต้นการแชทครั้งแรกกับ AI การเก็บข้อมูลเหล่านี้ช่วยให้มีบริบทที่มีคุณค่าและช่วยให้สามารถติดตามและวิเคราะห์กระบวนการเปลี่ยนแปลงบัญชีได้ดีขึ้น

การเก็บข้อความตัวกระตุ้นในฐานข้อมูลช่วยให้สามารถจัดการข้อผิดพลาดและการกู้คืนได้อย่างมีประสิทธิภาพ หาก เกิด ข้อ ผิด พลาดระหว่างการประมวลผลคำขอเปลี่ยนแปลงบัญชี ระบบจะทำเครื่องหมายคำขอว่าล้มเหลวและเปลี่ยนสถานะไปเป็นสถานะที่ต้องการการแทรกแซงจากมนุษย์ สิ่งนี้ช่วยให้มั่นใจว่าไม่มีคำขอใดสูญหายหรือถูกลืม และปัญหาใดๆ สามารถได้รับการแก้ไขและจัดการอย่างเหมาะสม

AI worker ในฐานะตัวจัดการกระบวนการ ให้จุดควบคุมกลางและเพิ่มความสามารถในการรายงานและดีบักกระบวนการที่ทรงพลัง อย่างไรก็ตาม สิ่งสำคัญที่ต้องทราบคือการใช้ AI worker เป็นตัวจัดการกระบวนการสำหรับทุกสถานการณ์เวิร์กโฟลว์ในแอปพลิเคชันของคุณอาจเกินความจำเป็น

การผสานรวม AI Workers เข้ากับสถาปัตยกรรมแอปพลิเคชันของคุณ

เมื่อรวม AI workers เข้ากับสถาปัตยกรรมแอปพลิเคชัน จำเป็นต้องพิจารณาด้านเทคนิคหลายประการเพื่อให้มั่นใจว่าการผสานรวมและการสื่อสารระหว่าง AI workers และคอมโพเนนต์อื่นๆ ของแอปพลิเคชันเป็นไปอย่างราบรื่น ส่วนนี้พิจารณาแง่มุมสำคัญของการออกแบบอินเตอร์เฟซเหล่านั้น การจัดการการไหลของข้อมูล และการจัดการวงจรชีวิตของ AI workers

การออกแบบอินเตอร์เฟซและโปรโตคอลการสื่อสารที่ชัดเจน

เพื่อให้การผสานรวมระหว่าง AI workers และคอมโพเนนต์อื่นๆ ของแอปพลิเคชันราบรื่น จำเป็นต้องกำหนดอินเตอร์เฟซและโปรโตคอลการสื่อสารที่ชัดเจน พิจารณาแนวทางต่อไปนี้:

การผสานรวมแบบ API: เปิดเผยฟังก์ชันการทำงานของ AI workers ผ่าน API ที่กำหนดไว้อย่างชัดเจน เช่น RESTful endpoints หรือสคีมา GraphQL สิ่งนี้ช่วยให้คอมโพเนนต์อื่นๆ สามารถโต้ตอบกับ AI workers โดยใช้คำขอและการตอบสนอง HTTP มาตรฐาน การผสานรวมแบบ API ให้สัญญาที่ชัดเจนระหว่าง AI workers และคอมโพเนนต์ที่ใช้งาน ทำให้ง่ายต่อการพัฒนา ทดสอบ และดูแลรักษาจุดเชื่อมต่อ

การสื่อสารแบบอิงข้อความ: ใช้รูปแบบการสื่อสารแบบอิงข้อความ เช่น คิวข้อความหรือระบบเผยแพร่-สมัครสมาชิก เพื่อเปิดใช้งานการโต้ตอบแบบอะซิงโครนัสระหว่าง AI workers และคอมโพเนนต์อื่นๆ แนวทางนี้แยก AI workers ออกจากส่วนที่เหลือของแอปพลิเคชัน ทำให้มีความสามารถในการปรับขนาด ความทนทานต่อความผิดพลาด และการเชื่อมต่อแบบหลวมๆ ที่ดีขึ้น การสื่อสารแบบอิงข้อความมีประโยชน์อย่างยิ่งเมื่อการประมวลผลที่ดำเนินการโดย AI workers ใช้เวลานานหรือใช้ทรัพยากรมาก เนื่องจากช่วยให้ส่วนอื่นๆ ของแอปพลิเคชันสามารถทำงานต่อไปได้โดยไม่ต้องรอให้ AI workers ทำงานเสร็จ

สถาปัตยกรรมที่ขับเคลื่อนด้วยเหตุการณ์: ออกแบบระบบของคุณรอบๆ เหตุการณ์และตัวกระตุ้น ที่เปิดใช้งาน AI workers เมื่อเงื่อนไขเฉพาะเป็นจริง AI workers สามารถสมัครสมาชิกเหตุการณ์ที่เกี่ยวข้องและตอบสนองตามนั้น โดยทำงานที่กำหนดเมื่อเหตุการณ์เกิดขึ้น สถาปัตยกรรมที่ขับเคลื่อนด้วยเหตุการณ์ช่วยให้สามารถประมวลผลแบบเรียลไทม์และอนุญาตให้เรียกใช้ AI workers ตามต้องการ ลดการใช้ทรัพยากรที่ไม่จำเป็น แนวทางนี้เหมาะสำหรับสถานการณ์ที่ AI workers ต้องตอบสนองต่อการกระทำหรือการเปลี่ยนแปลงสถานะแอปพลิเคชันที่เฉพาะเจาะจง

การจัดการการไหลของข้อมูลและการซิงโครไนซ์

เมื่อผสานรวม AI workers เข้ากับแอปพลิเคชันของคุณ สิ่งสำคัญคือต้องทำให้มั่นใจว่าการไหลของข้อมูลราบรื่นและรักษาความสอดคล้องของข้อมูลระหว่าง AI workers และคอมโพเนนต์อื่นๆ พิจารณาแง่มุมต่อไปนี้:

การเตรียมข้อมูล: ก่อนป้อนข้อมูลเข้าสู่ AI workers คุณอาจต้องทำงานเตรียมข้อมูลต่างๆ เช่น การทำความสะอาด การจัดรูปแบบ และ/หรือการแปลงข้อมูลนำเข้า คุณไม่เพียงแต่ต้องการให้แน่ใจว่า AI workers สามารถประมวลผลได้อย่างมีประสิทธิภาพ แต่คุณยังต้องการให้แน่ใจว่าคุณไม่ได้เสียโทเค็นไปกับการให้ความสนใจกับข้อมูลที่ worker อาจพิจารณาว่าไร้ประโยชน์ในกรณีที่ดีที่สุด หรือทำให้เสียสมาธิในกรณีที่แย่ที่สุด การเตรียมข้อมูลอาจเกี่ยวข้องกับงานต่างๆ เช่น การกำจัดสัญญาณรบกวน การจัดการค่าที่หายไป หรือการแปลงประเภทข้อมูล

การคงอยู่ของข้อมูล: คุณจะจัดเก็บและรักษาข้อมูลที่ไหลเข้าออกจาก AI workers อย่างไร พิจารณาปัจจัยต่างๆ เช่น ปริมาณข้อมูล รูปแบบการสืบค้น และความสามารถในการปรับขนาด คุณจำเป็นต้องเก็บรักษาบันทึกการสนทนาของ AI เพื่อสะท้อน “กระบวนการคิด” สำหรับการตรวจสอบหรือการดีบัก หรือเพียงพอแล้วที่จะมีบันทึกผลลัพธ์เท่านั้น?

การดึงข้อมูล: การได้มาซึ่งข้อมูลที่ AI workers ต้องการอาจเกี่ยวข้องกับการสืบค้นฐานข้อมูล การอ่านจากไฟล์ หรือการเข้าถึง API ภายนอก ต้องพิจารณาถึงความล่าช้าและวิธีที่ AI workers จะเข้าถึงข้อมูลที่ทันสมัยที่สุด พวกเขาต้องการการเข้าถึงฐานข้อมูลทั้งหมดหรือไม่ หรือคุณควรกำหนดขอบเขตการเข้าถึงอย่างจำกัดตามงานที่พวกเขาทำ? แล้วเรื่องการขยายระบบล่ะ? พิจารณากลไกการแคชเพื่อปรับปรุงประสิทธิภาพและลดภาระของแหล่งข้อมูล

การซิงโครไนซ์ข้อมูล: เมื่อ หลาย คอมโพเนนต์ รวม ถึง AI workers เข้า ถึง และ แก้ไข ข้อมูล ที่ ใช้ ร่วม กัน สิ่ง สำคัญ คือ ต้อง ใช้ กลไก การ ซิง โคร ไนซ์ ที่ เหมาะ สม เพื่อ รักษา ความ สอดคล้อง ของ ข้อมูล กลยุทธ์การล็อคฐานข้อมูล เช่น การล็อคแบบมองโลกในแง่ดีหรือแง่ร้าย อาจ ช่วย ป้องกัน ความ ขัด แย้ง และ รักษา ความ ถูก ต้อง ของ ข้อมูล ใช้ เทคนิค การจัดการธุรกรรมเพื่อจัดกลุ่มการดำเนินการข้อมูลที่เกี่ยวข้องและรักษาคุณสมบัติ ACID (Atomicity, Consistency, Isolation, และ Durability)

การจัดการข้อผิดพลาดและการกู้คืน: ใช้กลไกการจัดการข้อผิดพลาดและการกู้คืนที่แข็งแกร่งเพื่อจัดการกับปัญหาที่เกี่ยวข้องกับข้อมูลที่อาจเกิดขึ้นระหว่างกระบวนการไหลของข้อมูล จัดการข้อยกเว้นอย่างสง่างามและแสดงข้อความแจ้งข้อผิดพลาดที่มีความหมายเพื่อช่วยในการแก้ไขข้อบกพร่อง ใช้กลไกการลองใหม่และกลยุทธ์สำรอง เพื่อจัดการกับความล้มเหลวชั่วคราวหรือการขัดข้องของเครือข่าย กำหนดขั้นตอนที่ชัดเจนสำหรับการกู้คืนและการคืนสภาพข้อมูลในกรณีที่ข้อมูลเสียหายหรือสูญหาย

ด้วยการออกแบบและการใช้กลไกการไหลและการซิงโครไนซ์ข้อมูลอย่างรอบคอบ คุณสามารถทำให้มั่นใจได้ว่า AI workers ของคุณมีการเข้าถึงข้อมูลที่ถูกต้อง สอดคล้อง และทันสมัย ซึ่งช่วยให้พวกเขาสามารถทำงานได้อย่างมีประสิทธิภาพและให้ผลลัพธ์ที่เชื่อถือได้

การจัดการวงจรชีวิตของ AI Workers

พัฒนากระบวนการมาตรฐานสำหรับการเริ่มต้นและการกำหนดค่า AI workers ผมชอบเฟรมเวิร์กที่กำหนดมาตรฐานวิธีการกำหนดการตั้งค่าต่างๆ เช่น ชื่อโมเดล คำสั่งระบบ และนิยามฟังก์ชัน ตรวจสอบให้แน่ใจว่ากระบวนการเริ่มต้นเป็นแบบอัตโนมัติ

และสามารถทำซ้ำได้เพื่อช่วยในการปรับใช้และขยายระบบ

ใช้กลไกการตรวจสอบและบันทึกที่ครอบคลุม เพื่อติดตามสุขภาพและประสิทธิภาพของ AI workers เก็บรวบรวมเมตริกต่างๆ เช่น การใช้ทรัพยากร เวลาในการประมวลผล อัตราข้อผิดพลาด และปริมาณงาน ใช้ระบบบันทึกแบบรวมศูนย์เช่น ELK stack (Elasticsearch, Logstash, Kibana) เพื่อรวบรวมและวิเคราะห์บันทึกจาก AI workers หลายตัว

สร้างความทนทานต่อความผิดพลาดและความยืดหยุ่นเข้าไปในสถาปัตยกรรม AI worker ใช้กลไกการจัดการข้อผิดพลาดและการกู้คืนเพื่อจัดการกับความล้มเหลวหรือข้อยกเว้นอย่างสง่างาม โมเดลภาษาขนาดใหญ่ ยังคงเป็นเทคโนโลยีล้ำสมัย ผู้ให้บริการมักจะล่มบ่อยในเวลาที่ไม่คาดคิด ใช้กลไกการลองใหม่และเซอร์กิตเบรกเกอร์เพื่อป้องกันความล้มเหลวแบบลูกโซ่

ความสามารถในการประกอบรวมกันและการจัดการ AI Workers

หนึ่งในข้อดีหลักของสถาปัตยกรรม AI worker คือความสามารถในการประกอบรวมกัน ซึ่งช่วยให้คุณสามารถรวมและจัดการ AI workers หลายตัวเพื่อแก้ปัญหาที่ซับซ้อนได้ ด้วยการแบ่งงานใหญ่ออกเป็นงานย่อยที่จัดการได้ง่ายขึ้น โดยแต่ละงานจะถูกจัดการโดย AI worker ที่เชี่ยวชาญเฉพาะด้าน คุณสามารถสร้างระบบที่มีประสิทธิภาพและยืดหยุ่นได้ ในส่วนนี้ เราจะสำรวจวิธีการต่างๆ ในการประกอบและจัดการ “กลุ่ม” ของ AI workers

การเชื่อมโยง AI Workers สำหรับลำดับงานหลายขั้นตอน

ในหลายสถานการณ์ งานที่ซับซ้อนสามารถแยกย่อยเป็นขั้นตอนต่อเนื่องได้ โดยผลลัพธ์ของ AI worker ตัวหนึ่งจะกลายเป็นข้อมูลนำเข้าสำหรับตัวถัดไป การเชื่อมโยง AI workers นี้ สร้างลำดับงานหรือไปป์ไลน์หลายขั้นตอน AI worker แต่ละตัวในสายการทำงานจะมุ่งเน้นไปที่งานย่อยเฉพาะ และผลลัพธ์สุดท้ายคือผลรวมของความพยายามของ workers ทั้งหมด

มาดูตัวอย่างในบริบทของแอปพลิเคชัน Ruby on Rails สำหรับการประมวลผลเนื้อหาที่ผู้ใช้สร้างขึ้น ลำดับงานประกอบด้วยขั้นตอนต่อไปนี้ ซึ่งยอมรับว่าแต่ละขั้นตอนอาจจะง่ายเกินไปที่จะคุ้มค่ากับการแยกย่อยในกรณีการใช้งานจริง แต่มันทำให้ตัวอย่างเข้าใจง่ายขึ้น:

1. **การทำความสะอาดข้อความ:** AI worker ที่รับผิดชอบในการลบแท็ก HTML แปลงข้อความเป็นตัวพิมพ์เล็ก และจัดการการทำให้เป็นมาตรฐาน Unicode
2. **การตรวจจับภาษา:** AI worker ที่ระบุภาษาของข้อความที่ทำความสะอาดแล้ว
3. **การวิเคราะห์ความรู้สึก:** AI worker ที่กำหนดความรู้สึก (เชิงบวก เชิงลบ หรือเป็นกลาง) ของข้อความตามภาษาที่ตรวจพบ
4. **การจัดหมวดหมู่เนื้อหา:** AI worker ที่จำแนกข้อความเป็นหมวดหมู่ที่กำหนดไว้ล่วงหน้าโดยใช้เทคนิคการประมวลผลภาษาธรรมชาติ

นี่คือตัวอย่างอย่างง่ายของวิธีที่คุณสามารถเชื่อมโยง AI workers เหล่านี้เข้าด้วยกันโดยใช้ Ruby:

```ruby
class ContentProcessor
  def initialize(text)
    @text = text
  end

  def process
    cleaned_text = TextCleanupWorker.new(@text).call
    language = LanguageDetectionWorker.new(cleaned_text).call
    sentiment = SentimentAnalysisWorker.new(cleaned_text, language).call
    category = CategorizationWorker.new(cleaned_text, language).call

    { cleaned_text:, language:, sentiment:, category: }
  end
end
```

ในตัวอย่างนี้ คลาส ContentProcessor เริ่มต้นด้วยข้อความดิบและเชื่อมต่อตัวประมวลผล AI เข้าด้วยกันในเมธอด process โดยตัวประมวลผล AI แต่ละตัวจะทำงานเฉพาะของตนและส่งผลลัพธ์ไปยังตัวประมวลผลถัดไปในห่วงโซ่ ผลลัพธ์สุดท้ายคือแฮชที่ประกอบด้วยข้อความที่ทำความสะอาดแล้ว ภาษาที่ตรวจพบ ความรู้สึก และหมวดหมู่ของเนื้อหา

การประมวลผลแบบขนานสำหรับตัวประมวลผล AI ที่เป็นอิสระต่อกัน

ในตัวอย่างก่อนหน้านี้ ตัวประมวลผล AI ถูกเชื่อมต่อกันแบบลำดับ โดยแต่ละตัวจะประมวลผลข้อความและส่งผลลัพธ์ไปยังตัวประมวลผลถัดไป อย่างไรก็ตาม หากคุณมีตัวประมวลผล AI หลายตัวที่สามารถทำงานอิสระต่อกันบนข้อมูลเดียวกัน คุณสามารถเพิ่มประสิทธิภาพของเวิร์กโฟลว์ด้วยการประมวลผลพร้อมกันได้

ใน สถานการณ์ ที่ กำหนด เมื่อ TextCleanupWorker ทำความ สะอาด ข้อความ เสร็จ แล้ว LanguageDetectionWorker, SentimentAnalysisWorker และ CategorizationWorker สามารถประมวลผลข้อความที่ทำความสะอาดแล้วได้อย่างอิสระต่อกัน การรันตัวประมวลผลเหล่านี้แบบขนานสามารถช่วยลดเวลาการประมวลผลโดยรวมและเพิ่มประสิทธิภาพของเวิร์กโฟลว์ของคุณได้

เพื่อให้ได้การประมวลผลแบบขนานใน Ruby คุณสามารถใช้เทคนิคการประมวลผลพร้อมกัน เช่น เธรดหรือการเขียนโปรแกรมแบบอะซิงโครนัส นี่คือตัวอย่างวิธีการปรับแต่งคลาส ContentProcessor เพื่อประมวลผลตัวประมวลผลสามตัวสุดท้ายแบบขนานโดยใช้เธรด:

```ruby
require 'concurrent'

class ContentProcessor
  def initialize(text)
    @text = text
  end

  def process
    cleaned_text = TextCleanupWorker.new(@text).call

    language_future = Concurrent::Future.execute do
      LanguageDetectionWorker.new(cleaned_text).call
    end

    sentiment_future = Concurrent::Future.execute do
      SentimentAnalysisWorker.new(cleaned_text).call
    end

    category_future = Concurrent::Future.execute do
      CategorizationWorker.new(cleaned_text).call
    end

    language = language_future.value
    sentiment = sentiment_future.value
    category = category_future.value

    { cleaned_text:, language:, sentiment:, category: }
  end
end
```

ในเวอร์ชันที่ได้รับการปรับปรุงนี้ เราใช้ไลบรารี concurrent-ruby เพื่อสร้างออบเจกต์ Concurrent::Future สำหรับเวิร์กเกอร์ AI แต่ละตัวที่ทำงานอย่างอิสระ Future เป็นตัวแทนของการคำนวณที่จะถูกดำเนินการแบบอะซิงโครนัสในเธรดแยก

หลัง จาก ขั้น ตอน การ ทำความ สะอาด ข้อความ เรา สร้าง ออบ เจ กต์ Future สาม ตัว: language_future, sentiment_future, และ category_future Future แต่ละ ตัว จะ ดำ เนิน การ เวิร์ก เกอร์ AI ที่ เกี่ยวข้อง (LanguageDetectionWorker, SentimentAnalysisWorker, และ CategorizationWorker) ในเธรดแยก โดยส่ง cleaned_text เป็นอินพุต

เมื่อเรียกเมธอด value บน Future แต่ละตัว เราจะรอให้การคำนวณเสร็จสิ้นและรับผลลัพธ์ เมธอด value จะบล็อกจนกว่าจะมีผลลัพธ์พร้อมใช้งาน ทำให้มั่นใจได้ว่าเวิร์กเกอร์แบบขนานทั้งหมดได้ประมวลผลเสร็จสิ้นก่อนที่จะดำเนินการต่อ

สุดท้าย เราสร้างแฮชเอาต์พุตที่มีข้อความที่ทำความสะอาดแล้วและผลลัพธ์จากเวิร์กเกอร์แบบขนาน เหมือนกับในตัวอย่างเดิม

การประมวลผลเวิร์กเกอร์ AI อิสระแบบขนานสามารถลดเวลาการประมวลผลโดยรวมเมื่อเทียบกับการรันแบบลำดับ การปรับปรุงนี้มีประโยชน์อย่างยิ่งเมื่อต้องจัดการกับงานที่ใช้เวลานานหรือเมื่อประมวลผลข้อมูลจำนวนมาก

อย่างไรก็ตาม สิ่งสำคัญที่ต้องทราบคือประสิทธิภาพที่เพิ่มขึ้นจริงขึ้นอยู่กับปัจจัยต่างๆ เช่น ความซับซ้อนของเวิร์กเกอร์แต่ละตัว ทรัพยากรระบบที่มีอยู่ และค่าโสหุ้ยในการจัดการเธรด เป็นแนวปฏิบัติที่ดีที่จะทำการวัดประสิทธิภาพและวิเคราะห์โค้ดของคุณ เพื่อกำหนดระดับการทำงานแบบขนานที่เหมาะสมสำหรับกรณีการใช้งานเฉพาะของคุณ

นอกจากนี้ เมื่อใช้การประมวลผลแบบขนาน ให้คำนึงถึงทรัพยากรที่ใช้ร่วมกันหรือการพึ่งพากันระหว่างเวิร์กเกอร์ ตรวจสอบให้แน่ใจว่าเวิร์กเกอร์สามารถทำงานได้อย่างอิสระโดยไม่มีความขัดแย้งหรือเรซคอนดิชัน หากมีการพึ่งพาหรือทรัพยากรที่ใช้ร่วมกัน คุณอาจต้องใช้กลไกการซิงโครไนซ์ที่เหมาะสมเพื่อรักษาความถูกต้องของข้อมูลและหลีกเลี่ยงปัญหาเช่นเดดล็อกหรือผลลัพธ์ที่ไม่สอดคล้องกัน

ตัวล็อกตีความส่วนกลางของ Ruby และการประมวลผลแบบอะซิงโครนัส

สิ่งสำคัญคือต้องเข้าใจผลกระทบของตัวล็อกตีความส่วนกลาง (GIL) ของ Ruby เมื่อพิจารณาการประมวลผลแบบอะซิงโครนัสที่ใช้เธรดใน Ruby

GIL เป็นกลไกในตัวแปลภาษา Ruby ที่รับประกันว่าจะมีเพียงเธรดเดียวเท่านั้นที่สามารถดำเนินการโค้ด Ruby ได้ในแต่ละครั้ง แม้แต่บนโปรเซสเซอร์แบบหลายคอร์ นั่นหมายความว่าแม้จะสามารถสร้างและจัดการหลายเธรดภายในโปรเซส Ruby ได้ แต่จะมีเพียงเธรดเดียวเท่านั้นที่สามารถดำเนินการโค้ด Ruby ได้ในขณะใดขณะหนึ่ง

GIL ถูกออกแบบมาเพื่อทำให้การใช้งานตัวแปลภาษา Ruby ง่ายขึ้นและให้ความปลอดภัยของเธรดสำหรับโครงสร้างข้อมูลภายในของ Ruby อย่างไรก็ตาม มันก็จำกัดความสามารถในการดำเนินการโค้ด Ruby แบบขนานจริงๆ

เมื่อคุณใช้เธรดใน Ruby เช่น กับไลบรารี concurrent-ruby หรือคลาส Thread ที่มีในตัว เธรดจะอยู่ภายใต้ข้อจำกัดของ GIL GIL อนุญาตให้แต่ละเธรดดำเนินการโค้ด Ruby เป็นช่วงเวลาสั้นๆ ก่อนที่จะสลับไปยังเธรดอื่น สร้างภาพลวงตาของการดำเนินการพร้อมกัน

อย่างไรก็ตาม เนื่องจาก GIL การดำเนินการโค้ด Ruby จริงๆ ยังคงเป็นแบบลำดับ ในขณะที่เธรดหนึ่งกำลังดำเนินการโค้ด Ruby เธรดอื่นๆ จะถูกหยุดชั่วคราว รอที่จะได้รับ GIL และดำเนินการ

นี่หมายความว่าการประมวลผลแบบอะซิงโครนัสที่ใช้เธรดใน Ruby จะมีประสิทธิภาพมากที่สุดสำหรับงานที่ผูกติดกับการรับส่งข้อมูล เช่น การรอการตอบสนองจาก API ภายนอก (เช่น โมเดลภาษาขนาดใหญ่ที่โฮสต์โดยบุคคลที่สาม) หรือการดำเนินการรับส่งข้อมูลไฟล์ เมื่อเธรดพบการดำเนินการรับส่งข้อมูล มันสามารถปล่อย GIL ทำให้เธรดอื่นๆ สามารถดำเนินการได้ใน

ขณะที่รอให้การรับส่งข้อมูลเสร็จสิ้น

ในทางกลับกัน สำหรับงานที่ผูกติดกับซีพียู เช่น การคำนวณที่ซับซ้อนหรือการประมวลผลเวิร์กเกอร์ AI ที่ใช้เวลานาน GIL สามารถจำกัดประสิทธิภาพที่อาจเพิ่มขึ้นจากการทำงานแบบขนานด้วยเธรด เนื่องจากมีเพียงเธรดเดียวที่สามารถดำเนินการโค้ด Ruby ได้ในแต่ละครั้ง เวลาการดำเนินการโดยรวมอาจไม่ลดลงอย่างมีนัยสำคัญเมื่อเทียบกับการประมวลผลแบบลำดับ

เพื่อให้ได้การดำเนินการแบบขนานจริงสำหรับงานที่ผูกติดกับซีพียูใน Ruby คุณอาจต้องสำรวจวิธีการทางเลือกอื่น เช่น:

- ใช้การทำงานแบบขนานที่ใช้โปรเซสด้วยโปรเซส Ruby หลายตัว แต่ละตัวทำงานบนคอร์ซีพียูแยก
- ใช้ประโยชน์จากไลบรารีภายนอกหรือเฟรมเวิร์กที่มีส่วนขยายเนทีฟหรืออินเทอร์เฟซไปยังภาษาที่ไม่มี GIL เช่น C หรือ Rust,
- ใช้เฟรมเวิร์กการคำนวณแบบกระจายหรือคิวข้อความเพื่อกระจายงานไปยังหลายเครื่องหรือหลายโปรเซส

เป็นสิ่งสำคัญที่จะต้องพิจารณาลักษณะของงานและข้อจำกัดที่กำหนดโดย GIL เมื่อออกแบบและใช้งานการประมวลผลแบบอะซิงโครนัสใน Ruby แม้ว่าการประมวลผลแบบอะซิงโครนัสที่ใช้เธรดจะให้ประโยชน์สำหรับงานที่ผูกติดกับการรับส่งข้อมูล แต่อาจไม่ให้การปรับปรุงประสิทธิภาพที่สำคัญสำหรับงานที่ผูกติดกับซีพียูเนื่องจากข้อจำกัดของ GIL

เทคนิคแบบรวมกลุ่มเพื่อเพิ่มความแม่นยำ

เทคนิคแบบรวมกลุ่มเกี่ยวข้องกับการรวมเอาท์พุตจากเวิร์กเกอร์ AI หลายตัวเพื่อปรับปรุงความแม่นยำโดยรวมหรือความทนทานของระบบ แทนที่จะพึ่งพาเวิร์กเกอร์ AI เพียงตัวเดียว เทคนิคแบบรวมกลุ่มใช้ประโยชน์จากปัญญารวมของเวิร์กเกอร์หลายตัวเพื่อตัดสินใจอย่างมีข้อมูลมากขึ้น

การรวมกลุ่มโมเดล มีความสำคัญเป็นพิเศษในกรณีที่ส่วนต่างๆ ของขั้นตอนการทำงานของคุณทำงานได้ดีที่สุดกับโมเดล AI ที่แตกต่างกัน ซึ่งเป็นสิ่งที่พบได้บ่อยกว่าที่คุณคิด โมเดลที่ทรงพลังอย่าง GPT-4 มีค่าใช้จ่ายสูงมากเมื่อเทียบกับตัวเลือกแบบโอเพนซอร์สที่มีความสามารถน้อยกว่า และอาจไม่จำเป็นต้องใช้ในทุกขั้นตอนการทำงานของแอปพลิเคชันของคุณ

เทคนิคการรวมกลุ่มที่พบบ่อยคือการลงคะแนนเสียงข้างมาก ซึ่งตัวประมวลผล AI หลายตัวจะประมวลผลข้อมูลเดียวกันอย่างอิสระ และผลลัพธ์สุดท้ายจะถูกกำหนดโดยฉันทามติของเสียงส่วนใหญ่ วิธีการนี้สามารถช่วยลดผลกระทบจากความผิดพลาดของตัวประมวลผลแต่ละตัว และปรับปรุงความน่าเชื่อถือโดยรวมของระบบ

ลองพิจารณาตัวอย่างที่เรามีตัวประมวลผล AI สามตัวสำหรับการวิเคราะห์ความรู้สึก โดยแต่ละตัวใช้โมเดลที่แตกต่างกันหรือได้รับบริบทที่แตกต่างกัน เราสามารถรวมผลลัพธ์ของพวกมันโดยใช้การลงคะแนนเสียงข้างมากเพื่อกำหนดการทำนายความรู้สึกขั้นสุดท้าย

```
class SentimentAnalysisEnsemble
  def initialize(text)
    @text = text
  end

  def analyze
    predictions = [
      SentimentAnalysisWorker1.new(@text).analyze,
      SentimentAnalysisWorker2.new(@text).analyze,
      SentimentAnalysisWorker3.new(@text).analyze
    ]

    predictions
      .group_by { |sentiment| sentiment }
      .max_by { |_, votes| votes.size }
      .first

  end
end
```

ใน ตัวอย่าง นี้ คลาส SentimentAnalysisEnsemble จะ เริ่ม ต้น ด้วย ข้อความ และ เรียก ใช้ เวิร์ก เกอร์ AI สำหรับการวิเคราะห์ความรู้สึกสามตัว เมธอด analyze จะรวบรวมการทำนายจากแต่ละเวิร์กเกอร์และกำหนดความรู้สึกที่ได้รับเสียงส่วนใหญ่โดยใช้เมธอด group_by และ max_by ผลลัพธ์สุดท้ายคือความรู้สึกที่ได้รับคะแนนโหวตมากที่สุดจากชุดรวมของเวิร์กเกอร์

การรวมกลุ่มเป็นกรณีที่ชัดเจนว่าการทดลองใช้การประมวลผลแบบขนานอาจคุ้มค่ากับเวลาของคุณ

การเลือกและการเรียกใช้เวิร์กเกอร์ AI แบบไดนามิก

ในบางกรณีหรือส่วนใหญ่แล้ว เวิร์กเกอร์ AI ที่จะถูกเรียกใช้อาจขึ้นอยู่กับเงื่อนไขขณะทำงานหรือข้อมูลที่ผู้ใช้ป้อนเข้ามา การเลือกและการเรียกใช้เวิร์กเกอร์ AI แบบไดนามิกช่วยให้ระบบมีความยืดหยุ่นและปรับตัวได้

คุณอาจรู้สึกอยากใส่ฟังก์ชันการทำงานมากมายลงในเวิร์กเกอร์ AI ตัวเดียว โดยให้มันมีหลายฟังก์ชันและมีพรอมต์ที่ซับซ้อนเพื่ออธิบายวิธีการเรียกใช้ จงต้านทานความอยากนี้ เชื่อฉันเถอะ หนึ่งในเหตุผลที่แนวทางที่เรากำลังพูดถึงในบทนี้เรียกว่า “Multitude of Workers” ก็เพื่อเตือนเราว่าการมีเวิร์กเกอร์ที่เชี่ยวชาญเฉพาะด้านจำนวนมาก โดยแต่ละตัวทำงานเล็กๆ ของตัวเองเพื่อจุดประสงค์ที่ยิ่งใหญ่กว่านั้น เป็นสิ่งที่พึงปรารถนา

ตัวอย่างเช่น พิจารณาแอปพลิเคชันแชทบอทที่มีเวิร์กเกอร์ AI แต่ละตัวรับผิดชอบจัดการคำถามของผู้ใช้ประเภทต่างๆ โดยขึ้นอยู่กับข้อมูลที่ผู้ใช้ป้อนเข้ามา แอปพลิเคชันจะเลือกเวิร์กเกอร์ AI ที่เหมาะสมแบบไดนามิกเพื่อประมวลผลคำถามนั้น

```
class ChatbotController < ApplicationController
  def process_query
    query = params[:query]
    query_type = QueryClassifierWorker.new(query).classify

    case query_type
    when 'greeting'
      response = GreetingWorker.new(query).generate_response
    when 'product_inquiry'
      response = ProductInquiryWorker.new(query).generate_response
    when 'order_status'
      response = OrderStatusWorker.new(query).generate_response
    else
      response = DefaultResponseWorker.new(query).generate_response
    end

    render json: { response: response }
  end
end
```

ในตัวอย่างนี้ ChatbotController จะรับคำถามจากผู้ใช้ผ่านแอ็คชัน process_query โดยจะใช้ QueryClassifierWorker เพื่อระบุประเภทของคำถามก่อน จากนั้นตามประเภทคำถามที่จำแนกได้ ตัวควบคุมจะเลือก AI worker ที่เหมาะสมเพื่อสร้างคำตอบโดยอัตโนมัติ การเลือกแบบไดนามิกนี้ช่วยให้แชทบอทสามารถจัดการกับคำถามหลากหลายประเภทและส่งต่อไปยัง AI worker ที่เกี่ยวข้องได้

เนื่องจากงานของ QueryClassifierWorker ค่อนข้างเรียบง่ายและไม่ต้องการบริบทหรือการกำหนดฟังก์ชันมากนัก คุณอาจใช้ LLM ขนาดเล็กที่ทำงานได้เร็วมากอย่าง mistralai/mixtral-8x7b-instruct:nitro ซึ่งมีความสามารถใกล้เคียงกับระดับ GPT-4 ในหลายงาน และในขณะที่ผมเขียนหนังสือนี้ Groq สามารถประมวลผลได้ด้วยความเร็วสูงถึง 444 โทเค็นต่อวินาที

การผสมผสานการประมวลผลภาษาธรรมชาติแบบดั้งเดิมกับ LLMs

แม้ว่าโมเดลภาษาขนาดใหญ่ (LLMs) จะได้ปฏิวัติวงการการประมวลผลภาษาธรรมชาติ (NLP) โดยมอบความหลากหลายและประสิทธิภาพที่ไม่เคยมีมาก่อนในงานหลากหลายประเภท แต่ก็ไม่ใช่ทางออกที่มีประสิทธิภาพหรือคุ้มค่าที่สุดสำหรับทุกปัญหา ในหลายกรณี การผสมผสานเทคนิค NLP แบบดั้งเดิมกับ LLMs สามารถนำไปสู่วิธีการแก้ปัญหา NLP ที่มีการปรับให้เหมาะสม ตรงเป้าหมาย และประหยัดมากขึ้น

ลองนึกถึง LLMs เหมือนมีดพับอเนกประสงค์ในวงการ NLP—มีความหลากหลายและทรงพลัง แต่ไม่จำเป็นต้องเป็นเครื่องมือที่ดีที่สุดสำหรับทุกงาน บางครั้ง เครื่องมือเฉพาะทางอย่างที่เปิดขวดไวน์หรือที่เปิดกระป๋องอาจมีประสิทธิภาพและเหมาะสมกว่าสำหรับงานเฉพาะ เช่นเดียวกัน เทคนิค NLP แบบดั้งเดิม เช่น การจัดกลุ่มเอกสาร การระบุหัวข้อ และการจำแนกประเภท มักจะให้ทางออกที่ตรงเป้าหมายและคุ้มค่ากว่าสำหรับบางส่วนในไปป์ไลน์ NLP ของคุณ

หนึ่งในข้อได้เปรียบหลักของเทคนิค NLP แบบดั้งเดิมคือประสิทธิภาพในการคำนวณ วิธีการเหล่านี้ ซึ่งมักอาศัยโมเดลทางสถิติที่เรียบง่ายกว่าหรือวิธีการที่ใช้กฎเป็นพื้นฐาน สามารถประมวลผลข้อมูลข้อความจำนวนมากได้เร็วกว่าและใช้ทรัพยากรการคำนวณน้อยกว่าเมื่อเทียบกับ LLMs ทำให้เหมาะสมอย่างยิ่งสำหรับงานที่เกี่ยวข้องกับการวิเคราะห์และจัดระเบียบคลังเอกสารขนาดใหญ่ เช่น การจัดกลุ่มบทความที่คล้ายกันหรือการระบุหัวข้อสำคัญในคอลเลกชันข้อความ

นอกจากนี้ เทคนิค NLP แบบดั้งเดิมมักจะให้ความแม่นยำและความเที่ยงตรงสูงสำหรับงานเฉพาะ โดยเฉพาะเมื่อได้รับการฝึกฝนด้วยชุดข้อมูลเฉพาะด้าน ตัวอย่างเช่น ตัวจำแนกเอกสารที่ปรับแต่งมาอย่างดีโดยใช้อัลกอริทึมการเรียนรู้ของเครื่องแบบดั้งเดิม อย่าง ซัพพอร์ตเวกเตอร์แมชชีน (SVM) หรือนาอีฟเบย์ สามารถจำแนกประเภทเอกสารเข้าหมวดหมู่ที่กำหนดไว้ล่วงหน้าได้อย่างแม่นยำโดยใช้ต้นทุนการคำนวณน้อย

อย่างไรก็ตาม LLMs โดดเด่นอย่างแท้จริงเมื่อต้องทำงานที่ต้องการความเข้าใจภาษา บริบท และการให้เหตุผลในระดับลึก ความสามารถในการสร้างข้อความที่สอดคล้องและเกี่ยวข้องกับบริบท การตอบคำถาม และการสรุปข้อความยาวๆ นั้นเหนือกว่าวิธีการ NLP แบบดั้งเดิม LLMs สามารถจัดการกับปรากฏการณ์ทางภาษาที่ซับซ้อน เช่น ความกำกวม การอ้างอิงร่วม และสำนวน ทำให้มีคุณค่าอย่างยิ่งสำหรับงานที่ต้องการการสร้างหรือความเข้าใจภาษาธรรมชาติ

พลังที่แท้จริงอยู่ที่การผสมผสานเทคนิค NLP แบบดั้งเดิมกับ LLMs เพื่อสร้างวิธีการแบบผสมผสานที่ใช้ประโยชน์จากจุดแข็งของทั้งสองอย่าง การใช้วิธีการ NLP แบบดั้งเดิมสำหรับงานเช่นการประมวลผลเอกสารเบื้องต้น การจัดกลุ่ม และการสกัดหัวข้อ ช่วยให้คุณสามารถจัดระเบียบและโครงสร้างข้อมูลข้อความของคุณได้อย่างมีประสิทธิภาพ จากนั้นข้อมูลที่มีโครงสร้างนี้สามารถป้อนเข้าสู่ LLMs สำหรับงานขั้นสูง เช่น การสร้างบทสรุป การตอบคำถาม หรือการสร้างรายงานที่ครอบคลุม

ตัวอย่างเช่น ลองพิจารณากรณีการใช้งานที่คุณต้องการสร้างรายงานแนวโน้มสำหรับโดเมนเฉพาะโดยอิงจากคลังเอกสารแนวโน้มจำนวนมาก แทนที่จะพึ่งพา LLMs เพียงอย่างเดียว ซึ่งอาจมีค่าใช้จ่ายในการคำนวณสูงและใช้เวลานานในการประมวลผล

ข้อความจำนวนมาก คุณสามารถใช้วิธีการแบบผสมผสาน:

1. ใช้เทคนิค NLP แบบดั้งเดิม เช่น การสร้างแบบจำลองหัวข้อ (เช่น การจัดสรรแบบไดริชเลต์แฝง) หรืออัลกอริทึมการจัดกลุ่ม (เช่น เค-มีนส์) เพื่อจัดกลุ่มเอกสารแนวโน้มที่คล้ายกันและระบุธีมและหัวข้อสำคัญภายในคลังเอกสาร
2. ป้อนเอกสารที่จัดกลุ่มแล้วและหัวข้อที่ระบุเข้าสู่ LLM โดยใช้ประโยชน์จากความสามารถในการเข้าใจและสร้างภาษาที่เหนือกว่าเพื่อสร้างบทสรุปที่สอดคล้องและให้ข้อมูลสำหรับแต่ละกลุ่มหรือหัวข้อ
3. สุดท้าย ใช้ LLM เพื่อสร้างรายงานแนวโน้มที่ครอบคลุมโดยรวมบทสรุปแต่ละส่วน เน้นแนวโน้มที่สำคัญที่สุด และให้ข้อมูลเชิงลึกและคำแนะนำตามข้อมูลที่รวบรวมได้

การผสมผสานเทคนิค NLP แบบดั้งเดิมกับ LLMs ในลักษณะนี้ ช่วยให้คุณสามารถประมวลผลข้อมูลข้อความจำนวนมาก สกัดข้อมูลเชิงลึกที่มีความหมาย และสร้างรายงานคุณภาพสูง พร้อมทั้งเพิ่มประสิทธิภาพการใช้ทรัพยากรการคำนวณและต้นทุน

ในการเริ่มต้นโครงการการประมวลผลภาษาธรรมชาติของคุณ สิ่งสำคัญคือการประเมินข้อกำหนดและข้อจำกัดเฉพาะของแต่ละงานอย่างรอบคอบ และพิจารณาว่าจะสามารถใช้วิธีการประมวลผลภาษาธรรมชาติแบบดั้งเดิมร่วมกับโมเดลภาษาขนาดใหญ่อย่างไรเพื่อให้ได้ผลลัพธ์ที่ดีที่สุด ด้วยการผสมผสานประสิทธิภาพและความแม่นยำของเทคนิคแบบดั้งเดิมเข้ากับความยืดหยุ่นและพลังของโมเดลภาษาขนาดใหญ่ คุณสามารถสร้างโซลูชันการประมวลผลภาษาธรรมชาติที่มีประสิทธิภาพสูงและประหยัดต้นทุน ซึ่งจะสร้างคุณค่าให้กับผู้ใช้และผู้มีส่วนได้ส่วนเสียของคุณ

การใช้เครื่องมือ

ในด้านการพัฒนาแอปพลิเคชันที่ขับเคลื่อนด้วย AI แนวคิดเรื่อง “การใช้เครื่องมือ” หรือ “การเรียกใช้ฟังก์ชัน” ได้กลายเป็นเทคนิคอันทรงพลังที่ช่วยให้ LLM ของคุณสามารถเชื่อมต่อกับเครื่องมือภายนอก เอพีไอ ฟังก์ชัน ฐานข้อมูล และทรัพยากรอื่นๆ แนวทางนี้ช่วยให้สามารถแสดงพฤติกรรมที่หลากหลายมากกว่าแค่การส่งออกข้อความ และสร้างการโต้ตอบที่มีพลวัตมากขึ้นระหว่างคอมโพเนนต์ AI กับระบบนิเวศของแอปพลิเคชันของคุณ ดังที่เราจะได้ศึกษาในบทนี้ การใช้เครื่องมือยังให้ตัวเลือกในการทำให้โมเดล AI ของคุณสร้างข้อมูลในรูปแบบที่มีโครงสร้างได้

การใช้เครื่องมือคืออะไร?

การใช้เครื่องมือ หรือที่รู้จักกันในชื่อการเรียกใช้ฟังก์ชัน เป็นเทคนิคที่ช่วยให้นักพัฒนาสามารถระบุรายการฟังก์ชันที่ LLM สามารถโต้ตอบด้วยได้ในระหว่างกระบวนการสร้าง เครื่องมือเหล่านี้มีตั้งแต่ฟังก์ชันอรรถประโยชน์อย่างง่ายไปจนถึงเอพีไอที่ซับซ้อนหรือการสืบค้นฐานข้อมูล การให้ LLM เข้าถึงเครื่องมือเหล่านี้ได้ ช่วยให้นักพัฒนาสามารถขยายความสามารถของโมเดลและทำให้มันสามารถทำงานที่ต้องการความรู้หรือการกระทำจากภายนอกได้

แผนภูมิ 7. ตัวอย่างการกำหนดฟังก์ชันสำหรับ AI เวิร์กเกอร์ที่วิเคราะห์เอกสาร

```
FUNCTION = {
  name: "save_analysis",
  description: "Save analysis data for document",
  parameters: {
    type: "object",
    properties: {
      title: {
        type: "string",
        maxLength: 140
      },
      summary: {
        type: "string",
        description: "comprehensive multi-paragraph summary with
                      overview and list of sections (if applicable)"
      },
      tags: {
        type: "array",
        items: {
          type: "string",
          description: "lowercase tags representing main themes
                        of the document"
        }
      }
    },
    "required": %w[title summary tags]
  }
}.freeze
```

แนวคิดสำคัญเบื้องหลังการใช้เครื่องมือคือการให้ LLM มีความสามารถในการเลือกและดำเนินการเครื่องมือที่เหมาะสมโดยอัตโนมัติตามอินพุตของผู้ใช้หรืองานที่ต้องการ แทนที่จะพึ่งพาเพียงความรู้ที่ถูกฝึกฝนมาก่อน การใช้เครื่องมือช่วยให้ LLM สามารถใช้ประโยชน์จากทรัพยากรภายนอกเพื่อสร้างการตอบสนองที่แม่นยำ เกี่ยวข้อง และนำไปปฏิบัติได้มากขึ้น การใช้เครื่องมือทำให้เทคนิคต่างๆ เช่น RAG (การสร้างเนื้อหาที่เสริมด้วยการค้นคืน) สามารถนำไปใช้ได้ง่ายกว่าที่เคย

> โปรดทราบว่า หากไม่ได้ระบุไว้เป็นอย่างอื่น หนังสือเล่มนี้สันนิษฐานว่าแบบจำลอง AI ของคุณไม่มีการเข้าถึงเครื่องมือฝั่งเซิร์ฟเวอร์ที่ติดตั้งมาให้ เครื่องมือใดๆ ที่คุณต้องการให้ AI ใช้งานได้จะต้องถูกประกาศโดยคุณอย่างชัดเจนในแต่ละคำขอ API

พร้อมด้วยข้อกำหนดสำหรับการจัดการการทำงานหากและเมื่อ AI แจ้งว่าต้องการใช้เครื่องมือนั้นในการตอบสนอง

ศักยภาพของการใช้เครื่องมือ

การใช้เครื่องมือเปิดโอกาสอย่างกว้างขวางสำหรับแอปพลิเคชันที่ขับเคลื่อนด้วย AI ต่อไปนี้คือตัวอย่างของสิ่งที่สามารถทำได้ด้วยการใช้เครื่องมือ:

1. **แชทบอทและผู้ช่วยเสมือน**: ด้วยการเชื่อมต่อ LLM กับเครื่องมือภายนอก แชทบอทและผู้ช่วยเสมือนสามารถทำงานที่ซับซ้อนมากขึ้น เช่น การดึงข้อมูลจากฐานข้อมูล การเรียกใช้ API หรือการโต้ตอบกับระบบอื่นๆ ตัวอย่างเช่น แชทบอทสามารถใช้เครื่องมือ CRM เพื่อเปลี่ยนสถานะของดีลตามคำขอของผู้ใช้
2. **การวิเคราะห์ข้อมูลและข้อมูลเชิงลึก**: LLM สามารถเชื่อมต่อกับเครื่องมือหรือไลบรารีวิเคราะห์ข้อมูลเพื่อทำงานประมวลผลข้อมูลขั้นสูง สิ่งนี้ช่วยให้แอปพลิเคชันสามารถสร้างข้อมูลเชิงลึก ทำการวิเคราะห์เปรียบเทียบ หรือให้คำแนะนำตามข้อมูลตามคำถามของผู้ใช้
3. **การค้นหาและการค้นคืนข้อมูล**: การใช้เครื่องมือช่วยให้ LLM สามารถโต้ตอบกับเครื่องมือค้นหา ฐานข้อมูลเวกเตอร์ หรือระบบค้นคืนข้อมูลอื่นๆ ด้วยการแปลงคำถามของผู้ใช้เป็นคำค้นหา LLM สามารถค้นคืนข้อมูลที่เกี่ยวข้องจากหลายแหล่งและให้คำตอบที่ครอบคลุมแก่คำถามของผู้ใช้
4. **การผสานรวมกับบริการภายนอก**: การใช้เครื่องมือช่วยให้การผสานรวมระหว่างแอปพลิเคชันที่ขับเคลื่อนด้วย AI และบริการหรือ API ภายนอกเป็นไปอย่างราบรื่น ตัวอย่างเช่น LLM สามารถโต้ตอบกับ API สภาพอากาศเพื่อให้ข้อมูลสภาพอากาศแบบเรียลไทม์ หรือ API แปลภาษาเพื่อสร้างการตอบสนองในหลายภาษา

ขั้นตอนการใช้เครื่องมือ

ขั้นตอนการใช้เครื่องมือมักประกอบด้วยสี่ขั้นตอนหลัก:

1. รวมคำนิยามฟังก์ชันในบริบทคำขอของคุณ
2. การเลือกเครื่องมือแบบไดนามิก (หรือแบบชัดเจน)
3. การดำเนินการฟังก์ชัน
4. การดำเนินการต่อของพรอมต์เดิม (ไม่บังคับ)

มาดูรายละเอียดแต่ละขั้นตอนกัน

รวมคำนิยามฟังก์ชันในบริบทคำขอของคุณ

AI รู้ว่ามีเครื่องมืออะไรบ้างที่สามารถใช้ได้เพราะคุณให้รายการมาพร้อมกับคำขอการทำงาน (โดยทั่วไปกำหนดเป็นฟังก์ชันโดยใช้รูปแบบของสคีมา JSON)

ไวยากรณ์ที่แน่นอนของการกำหนดเครื่องมือขึ้นอยู่กับแบบจำลองที่ใช้

นี่คือวิธีการกำหนดฟังก์ชัน get_weather ใน Claude 3:

```
{
  "name": "get_weather",
  "description": "Get the current weather in a given location",
  "input_schema": {
    "type": "object",
    "properties": {
      "location": {
        "type": "string",
        "description": "The city and state, e.g. San Francisco, CA"
      },
      "unit": {
        "type": "string",
        "enum": ["celsius", "fahrenheit"],
        "description": "The unit of temperature"
    }
  },
  "required": ["location"]
  }
}
```

และนี่คือวิธีที่คุณจะกำหนดฟังก์ชันเดียวกันสำหรับ GPT-4 โดยส่งค่าไปยังพารามิเตอร์ tools:

```
{
    "name": "get_current_weather",
    "description": "Get the current weather in a given location",
    "parameters": {
        "type": "object",
        "properties": {
            "location": {
                "type": "string",
                "description": "The city and state, e.g. San Francisco, CA",
            },
            "unit": {
                "type": "string",
                "enum": ["celsius", "fahrenheit"],
                "description": "The unit of temperature"
            },
        },
        "required": ["location"],
    },
}
```

> เกือบจะเหมือนกันเลย แต่กลับต่างกันโดยไม่มีเหตุผล! น่าหงุดหงิดจริงๆ

การกำหนดฟังก์ชันระบุชื่อ คำอธิบาย และพารามิเตอร์อินพุต พารามิเตอร์อินพุตสามารถกำหนดเพิ่มเติมได้โดยใช้แอตทริบิวต์ต่างๆ เช่น enums เพื่อจำกัดค่าที่ยอมรับได้ และการระบุว่าพารามิเตอร์นั้นจำเป็นหรือไม่

นอกเหนือจากการกำหนดฟังก์ชันจริงๆ แล้ว คุณยังสามารถรวมคำแนะนำหรือบริบทเกี่ยวกับเหตุผลและวิธีการใช้ฟังก์ชันในคำสั่งระบบได้ด้วย ตัวอย่างเช่น เครื่องมือค้นหาเว็บของฉันใน Olympia มีคำสั่งระบบนี้ ซึ่งเตือนให้ AI ระลึกว่ามีเครื่องมือที่กล่าวถึงพร้อมใช้งาน:

```
The `google_search` and `realtime_search` functions let you do research
on behalf of the user. In contrast to Google, realtime search is powered
by Perplexity and provides real-time information to curated current events
databases and news sources. Make sure to include URLs in your response so
user can do followup research.
```

การให้คำอธิบายโดยละเอียดถือเป็นปัจจัยที่สำคัญที่สุดในประสิทธิภาพของเครื่องมือ คำอธิบายของคุณควรอธิบายทุกรายละเอียดเกี่ยวกับเครื่องมือ ซึ่งรวมถึง:

- เครื่องมือนี้ทำอะไร
- ควรใช้เมื่อไหร่ (และเมื่อไหร่ที่ไม่ควรใช้)
- พารามิเตอร์แต่ละตัวหมายถึงอะไรและส่งผลต่อพฤติกรรมของเครื่องมืออย่างไร
- ข้อควรระวังหรือข้อจำกัดที่สำคัญใดๆ ที่เกี่ยวข้องกับการนำเครื่องมือไปใช้งาน

ยิ่งคุณสามารถให้บริบทเกี่ยวกับเครื่องมือของคุณกับ AI ได้มากเท่าไร AI ก็จะยิ่งสามารถตัดสินใจได้ดีขึ้นว่าเมื่อไหร่และอย่างไรที่จะใช้เครื่องมือเหล่านั้น ตัวอย่างเช่น Anthropic แนะนำว่าควรมีคำอธิบายอย่างน้อย 3-4 ประโยคต่อเครื่องมือสำหรับ Claude 3 ซีรีส์ และควรมีมากกว่านั้นหากเครื่องมือมีความซับซ้อน

แม้จะไม่ค่อยเป็นที่เข้าใจนัก แต่คำอธิบายถือว่าสำคัญกว่าตัวอย่าง แม้ว่าคุณสามารถรวมตัวอย่างการใช้เครื่องมือไว้ในคำอธิบายหรือในพรอมต์ที่มาด้วยกันได้ แต่สิ่งนี้สำคัญน้อยกว่าการมีคำอธิบายที่ชัดเจนและครอบคลุมเกี่ยวกับจุดประสงค์และพารามิเตอร์ของเครื่องมือ ให้เพิ่มตัวอย่างหลังจากที่คุณได้อธิบายรายละเอียดอย่างครบถ้วนแล้วเท่านั้น

นี่คือตัวอย่างของข้อกำหนดฟังก์ชัน API แบบ Stripe:

```
{
  "name": "createPayment",
  "description": "Create a new payment request",
  "parameters": {
    "type": "object",
    "properties": {
      "transaction_amount": {
        "type": "number",
        "description": "The amount to be paid"
      },
      "description": {
        "type": "string",
        "description": "A brief description of the payment"
      },
      "payment_method_id": {
        "type": "string",
        "description": "The payment method to be used"
      },
      "payer": {
        "type": "object",
        "description": "Information about the payer, including their name,
                        email, and identification number",
        "properties": {
          "name": {
            "type": "string",
            "description": "The payer's name"
          },
          "email": {
            "type": "string",
            "description": "The payer's email address"
          },
          "identification": {
            "type": "object",
            "description": "The payer's identification number",
            "properties": {
              "type": {
                "type": "string",
                "description": "Identification document (e.g. CPF, CNPJ)"
              },
              "number": {
                "type": "string",
                "description": "The identification number"
```

```
            }
          },
          "required": [ "type", "number" ]
        }
      },
      "required": [ "name", "email", "identification" ]
    }
  }
}
```

ในทางปฏิบัติ โมเดลบางตัวอาจมีปัญหาในการจัดการกับการระบุฟังก์ชันที่ซ้อนกันและการจัดการกับประเภทข้อมูลเอาต์พุตที่ซับซ้อน เช่น อาร์เรย์ ดิกชันนารี เป็นต้น แต่ในทางทฤษฎีแล้ว คุณควรจะสามารถกำหนดสคีมา JSON ที่มีความลึกเท่าใดก็ได้!

การเลือกเครื่องมือแบบไดนามิก

เมื่อคุณดำเนินการแชทคอมพลีชันที่รวมคำจำกัดความของเครื่องมือ LLM จะเลือกเครื่องมือที่เหมาะสมที่สุดโดยอัตโนมัติและสร้างพารามิเตอร์อินพุตที่จำเป็

ในทางปฏิบัติ ความสามารถของ AI ในการเรียกฟังก์ชันที่*ถูกต้อง*พอดีและการปฏิบัติตามข้อกำหนดของคุณสำหรับอินพุต*อย่างแม่นยำ*นั้นมีทั้งสำเร็จและล้มเหลว การปรับค่าไฮเปอร์พารามิเตอร์อุณหภูมิลงเป็น 0.0 ช่วยได้มาก แต่จากประสบการณ์ของผม คุณก็ยังจะพบข้อผิดพลาดเป็นครั้งคราว ความล้มเหลวเหล่านั้นรวมถึงการสร้างชื่อฟังก์ชันที่ไม่มีอยู่จริง พารามิเตอร์อินพุตที่ผิดชื่อหรือหายไป พารามิเตอร์ถูกส่งในรูปแบบ JSON ซึ่งหมายความว่าบางครั้งคุณจะเห็นข้อผิดพลาดที่เกิดจาก JSON ที่ไม่สมบูรณ์ มีเครื่องหมายคำพูดผิด หรือเสียหายในรูปแบบอื่นๆ

รูปแบบการซ่อมแซมข้อมูลด้วยตัวเองสามารถช่วยแก้ไขโดยอัตโนมัติการเรียกฟังก์ชันที่ผิดพลาดเนื่องจากข้อผิดพลาดทางไวยากรณ์

การเลือกเครื่องมือแบบบังคับ (หรือแบบชัดแจ้ง)

โมเดลบางตัวให้ตัวเลือกในการบังคับให้เรียกฟังก์ชันเฉพาะ โดยกำหนดเป็นพารามิเตอร์ในคำขอ มิฉะนั้น การตัดสินใจว่าจะเรียกฟังก์ชันหรือไม่จะขึ้นอยู่กับดุลยพินิจของ AI ทั้งหมด

ความสามารถในการบังคับให้เรียกฟังก์ชันมีความสำคัญในบางสถานการณ์ที่คุณต้องการให้แน่ใจว่าเครื่องมือหรือฟังก์ชันเฉพาะจะถูกเรียกใช้ โดยไม่คำนึงถึงกระบวนการเลือกแบบไดนามิกของ AI มีหลายเหตุผลที่ความสามารถนี้มีความสำคัญ:

1. **การควบคุมแบบชัดแจ้ง**: คุณอาจใช้ AI เป็น*คอมโพเนนต์แยกส่วน*หรือในเวิร์กโฟลว์ที่กำหนดไว้ล่วงหน้าซึ่งจำเป็นต้องมีการเรียกใช้ฟังก์ชันเฉพาะในเวลาเฉพาะ การบังคับให้เรียกฟังก์ชันช่วยให้คุณมั่นใจได้ว่าฟังก์ชันที่ต้องการจะถูกเรียกใช้แทนที่จะต้องขอร้อง AI อย่างสุภาพ
2. **การดีบักและการทดสอบ**: เมื่อพัฒนาและทดสอบแอปพลิเคชันที่ขับเคลื่อนด้วย AI ความสามารถในการบังคับให้เรียกฟังก์ชันมีประโยชน์อย่างยิ่งสำหรับการดีบัก การเรียกใช้ฟังก์ชันเฉพาะอย่างชัดแจ้งช่วยให้คุณแยกและทดสอบคอมโพเนนต์แต่ละส่วนของแอปพลิเคชันของคุณ ซึ่งช่วยให้คุณตรวจสอบความถูกต้องของการใช้งานฟังก์ชัน ตรวจสอบพารามิเตอร์อินพุต และตรวจสอบว่าได้รับผลลัพธ์ตามที่คาดหวัง
3. **การจัดการกรณีพิเศษ**: อาจมีกรณีพิเศษหรือสถานการณ์ที่ไม่ปกติซึ่งกระบวนการเลือกแบบไดนามิกของ AI อาจไม่เลือกที่จะเรียกใช้ฟังก์ชันที่ควรจะเรียก และคุณรู้เรื่องนี้จากกระบวนการภายนอก ในกรณีเช่นนี้ การมีความสามารถในการบังคับให้เรียกฟังก์ชันช่วยให้คุณจัดการสถานการณ์เหล่านี้ได้อย่างชัดเจน กำหนดกฎหรือเงื่อนไขในตรรกะของแอปพลิเคชันเพื่อกำหนดว่าเมื่อใดควรแทนที่ดุลยพินิจของ AI
4. **ความ สอดคล้อง และ การ ทำ ซ้ำ ได้**: หาก คุณ มี ลำดับ ฟังก์ชัน เฉพาะ ที่ ต้อง ดำเนิน การ ตาม ลำดับ ที่ แน่นอน การบังคับให้เรียกฟังก์ชันรับประกันว่าจะใช้ลำดับเดียวกันทุกครั้ง สิ่งนี้มีความสำคัญอย่างยิ่งในแอปพลิเคชันที่ต้องการความสอดคล้องและพฤติกรรมที่คาดเดาได้ เช่น ในระบบการเงินหรือการจำลองทางวิทยาศาสตร์
5. **การ เพิ่ม ประสิทธิภาพ การ ทำงาน**: ใน บาง กรณี การ บังคับ ให้ เรียก ฟังก์ชัน สามารถ นำ ไป สู่ การ เพิ่ม ประสิทธิภาพ การ ทำงาน หาก คุณ ทราบ ว่า ต้อง ใช้ ฟังก์ชัน เฉพาะ สำหรับ งาน เฉพาะ และ กระบวนการ เลือก แบบ ไดนามิก ของ AI อาจ ทำให้ เกิด ภาระ ที่ ไม่ จำเป็น คุณ สามารถ ข้าม กระบวนการ เลือก และ เรียก ใช้ ฟังก์ชัน ที่ ต้องการ โดยตรง สิ่งนี้สามารถช่วยลดความล่าช้าและปรับปรุงประสิทธิภาพโดยรวมของแอปพลิเคชันของคุณ

โดยสรุป ความสามารถในการบังคับให้เรียกฟังก์ชันในแอปพลิเคชันที่ขับเคลื่อนด้วย AI ให้การควบคุมแบบชัดแจ้ง ช่วยในการดีบักและการทดสอบ จัดการกรณีพิเศษ รับประกันความสอดคล้องและการทำซ้ำได้ มันเป็นเครื่องมือที่ทรงพลังในคลังแสงของคุณ แต่เราจำเป็นต้องพูดถึงอีกหนึ่งแง่มุมของฟีเจอร์สำคัญนี้

ในกรณีการตัดสินใจหลายๆ กรณี เราอาจต้องการให้โมเดลเรียกฟังก์ชันเสมอและอาจไม่ต้องการให้โมเดลตอบสนองด้วยความรู้ภายในเท่านั้น ตัวอย่างเช่น หากคุณกำลังจัดเส้นทางระหว่างโมเดลหลายตัวที่เชี่ยวชาญในงานที่แตกต่างกัน (อินพุตหลายภาษา คณิตศาสตร์ ฯลฯ) คุณอาจใช้โมเดลที่เรียกใช้ฟังก์ชันเพื่อมอบหมายคำขอให้กับหนึ่งในโมเดลตัวช่วยและไม่ตอบสนองด้วยตัวเอง

พารามิเตอร์ตัวเลือกเครื่องมือ

GPT-4 และโมเดลภาษาอื่นๆ ที่รองรับการเรียกใช้ฟังก์ชันจะมีพารามิเตอร์ tool_choice สำหรับควบคุมว่าจำเป็นต้องใช้เครื่องมือเป็นส่วนหนึ่งของการคอมพลีทหรือไม่ พารามิเตอร์นี้มีค่าที่เป็นไปได้สามค่า:

- auto ให้ AI ตัดสินใจอย่างเต็มที่ว่าจะใช้เครื่องมือหรือเพียงแค่ตอบกลับ
- required บอก AI ว่า*ต้อง*เรียกเครื่องมือ*แทนที่*จะตอบกลับ แต่ปล่อยให้การเลือกเครื่องมือขึ้นอยู่กับ AI
- ตัวเลือกที่สามคือการตั้งค่าพารามิเตอร์ของ name_of_function ที่คุณต้องการบังคับให้เรียกใช้ จะกล่าวถึงเพิ่มเติมในส่วนถัดไป

โปรด ทราบ ว่า หาก คุณ ตั้ง ค่า tool choice เป็น required โมเดล จะ ถูก บังคับ ให้ เลือก ฟังก์ชัน ที่ เกี่ยวข้อง ที่สุด จาก ตัว เลือก ที่ มี ให้ แม้ว่า จะ ไม่มี ฟังก์ชัน ใด ที่ เหมาะ สม กับ คำ สั่ง จริงๆ ณ เวลา ที่ เขียน หนังสือ นี้ ผม ยัง ไม่ พบ โมเดล ที่ จะ ส่ง คืน การ ตอบ สนอง tool_calls ที่ ว่าง เปล่า หรือใช้วิธีอื่นในการแจ้งให้คุณทราบว่าไม่พบฟังก์ชันที่เหมาะสมที่จะเรียกใช้

การบังคับใช้ฟังก์ชันเพื่อให้ได้เอาต์พุตที่มีโครงสร้าง

ความสามารถในการบังคับเรียกใช้ฟังก์ชันทำให้คุณสามารถบังคับให้ได้ข้อมูลที่มีโครงสร้างจากการแชทคอมพลีชัน แทนที่จะต้องดึงข้อมูลออกมาเองจากการตอบกลับที่เป็นข้อความธรรมดา

> ทำไมการบังคับใช้ฟังก์ชันเพื่อให้ได้เอาต์พุตที่มีโครงสร้างจึงสำคัญมาก? พูดง่ายๆ คือเพราะการดึงข้อมูลที่มีโครงสร้างจากเอาต์พุตของ LLM นั้นเป็นเรื่องที่ปวดหัวมาก คุณอาจทำให้ชีวิตง่ายขึ้นได้บ้างโดยการขอข้อมูลในรูปแบบ XML แต่จากนั้นคุณก็ต้องแยกวิเคราะห์ XML และคุณจะทำอย่างไรเมื่อ XML นั้นหายไปเพราะ AI ของคุณตอบกลับว่า: “ขออภัย ฉันไม่สามารถสร้างข้อมูลที่คุณร้องขอได้เพราะว่า บลา บลา บลา...”

เมื่อใช้เครื่องมือในลักษณะนี้:

- คุณควรกำหนดเครื่องมือเพียงหนึ่งตัวในคำขอของคุณ

- อย่าลืมบังคับใช้ฟังก์ชันของมันโดยใช้พารามิเตอร์ tool_choice
- จำไว้ว่าโมเดลจะส่งอินพุตไปยังเครื่องมือ ดังนั้นชื่อของเครื่องมือและคำอธิบายควรมาจากมุมมองของโมเดล ไม่ใช่ของคุณ

ประเด็นสุดท้ายนี้สมควรยกตัวอย่างเพื่อความชัดเจน สมมติว่าคุณกำลังขอให้ AI วิเคราะห์ความรู้สึกจากข้อความของผู้ใช้ ชื่อของฟังก์ชันจะไม่ใช่ analyze_sentiment แต่จะเป็นอะไรประมาณ save_sentiment_analysis AI เป็นผู้ทำการวิเคราะห์ความรู้สึก *ไม่ใช่เครื่องมือ* สิ่งที่เครื่องมือทำทั้งหมด (จากมุมมองของ AI) คือการบันทึกผลการวิเคราะห์

นี่คือตัวอย่างการใช้ Claude 3 ในการบันทึกสรุปของรูปภาพลงใน JSON ที่มีโครงสร้างที่ดี คราวนี้จากบรรทัดคำสั่งโดยใช้ curl:

```
curl https://api.anthropic.com/v1/messages \
    --header "content-type: application/json" \
    --header "x-api-key: $ANTHROPIC_API_KEY" \
    --header "anthropic-version: 2023-06-01" \
    --header "anthropic-beta: tools-2024-04-04" \
    --data \
'{
   "model": "claude-3-sonnet-20240229",
   "max_tokens": 1024,
   "tools": [{
       "name": "record_summary",
       "description": "Record summary of image into well-structured JSON.",
       "input_schema": {
           "type": "object",
           "properties": {
               "key_colors": {
                   "type": "array",
                   "items": {
                       "type": "object",
                       "properties": {
                           "r": {
                               "type": "number",
                               "description": "red value [0.0, 1.0]"
                           },
                           "g": {
                               "type": "number",
                               "description": "green value [0.0, 1.0]"
                           },
                           "b": {
                               "type": "number",
```

```
                        "description": "blue value [0.0, 1.0]"
                    },
                    "name": {
                        "type": "string",
                        "description": "Human-readable color name
                                        in snake_case, e.g.
                                        \"olive_green\"or
                                        \"turquoise\""
                    }
                },
                "required": [ "r", "g", "b", "name" ]
            },
            "description": "Key colors in the image. Four or less."
        },
        "description": {
            "type": "string",
            "description": "Image description. 1-2 sentences max."
        },
        "estimated_year": {
            "type": "integer",
            "description": "Estimated year that the image was taken,
                            if is it a photo. Only set this if the
                            image appears to be non-fictional.
                            Rough estimates are okay!"
        }
    },
    "required": [ "key_colors", "description" ]
  }
}],
"messages": [
  {
    "role": "user",
    "content": [
      {
        "type": "image",
        "source": {
          "type": "base64",
          "media_type": "'$IMAGE_MEDIA_TYPE'",
          "data": "'$IMAGE_BASE64'"
        }
      },
      {
```

```
                "type": "text",
                "text": "Use `record_summary` to describe this image."
            }
        ]
    }
  ]
}'
```

ในตัวอย่างที่แสดง เรากำลังใช้โมเดล Claude 3 จาก Anthropic เพื่อสร้างสรุปข้อมูล JSON ที่มีโครงสร้างจากรูปภาพ นี่คือวิธีการทำงาน:

1. เรากำหนดเครื่องมือเดียวชื่อ record_summary ในอาร์เรย์ tools ของ payload คำขอ เครื่องมือนี้มีหน้าที่บันทึกสรุปของรูปภาพในรูปแบบ JSON ที่มีโครงสร้าง
2. เครื่องมือ record_summary มี input_schema ที่ระบุโครงสร้างที่คาดหวังของผลลัพธ์ JSON โดยกำหนดคุณสมบัติสามอย่าง:
    - key_colors: อาร์เรย์ของออบเจ็กต์ที่แสดงสีหลักในรูปภาพ แต่ละออบเจ็กต์สีมีคุณสมบัติสำหรับค่าสีแดง เขียว และน้ำเงิน (ช่วงตั้งแต่ 0.0 ถึง 1.0) และชื่อสีที่มนุษย์อ่านได้ในรูปแบบ snake_case
    - description: คุณสมบัติสตริงสำหรับคำอธิบายสั้นๆ ของรูปภาพ จำกัดที่ 1-2 ประโยค
    - estimated_year: คุณสมบัติจำนวนเต็มที่เป็นตัวเลือก สำหรับการประมาณปีที่ถ่ายภาพ หากดูเหมือนเป็นภาพถ่ายที่ไม่ใช่เรื่องแต่ง
3. ในอาร์เรย์ messages เราให้ข้อมูลรูปภาพในรูปแบบสตริงที่เข้ารหัสแบบ Base64 พร้อมกับประเภทสื่อ ซึ่งช่วยให้โมเดลสามารถประมวลผลรูปภาพเป็นส่วนหนึ่งของข้อมูลนำเข้า
4. เรายังกระตุ้นให้ Claude ใช้เครื่องมือ record_summary เพื่ออธิบายรูปภาพ
5. เมื่อส่งคำขอไปยังโมเดล Claude 3 มันจะวิเคราะห์รูปภาพและสร้างสรุป JSON ตาม input_schema ที่กำหนด โมเดลจะดึงสีหลัก ให้คำอธิบายสั้นๆ และประมาณปีที่ถ่ายภาพ (ถ้าเกี่ยวข้อง)
6. สรุป JSON ที่สร้างขึ้นจะถูกส่งเป็นพารามิเตอร์ให้กับเครื่องมือ record_summary ซึ่งให้การแสดงผลที่มีโครงสร้างของลักษณะสำคัญของรูปภาพ

การใช้เครื่องมือ record_summary พร้อมกับ input_schema ที่กำหนดไว้อย่างชัดเจน ทำให้เราสามารถรับสรุป JSON ที่มีโครงสร้างของรูปภาพได้โดยไม่ต้องพึ่งพาการดึงข้อความธรรมดา วิธีนี้ช่วยให้มั่นใจว่าผลลัพธ์จะเป็นไปตามรูปแบบที่สม่ำเสมอและสามารถแยกวิเคราะห์และประมวลผลได้ง่ายโดยส่วนประกอบในลำดับถัดไปของแอปพลิเคชัน

ความสามารถในการบังคับใช้ฟังก์ชันและระบุโครงสร้างผลลัพธ์ที่คาดหวังเป็นคุณสมบัติที่ทรงพลังของการใช้เครื่องมือในแอปพลิเคชันที่ขับเคลื่อนด้วย AI ช่วยให้นักพัฒนาสามารถควบคุมผลลัพธ์ที่สร้างขึ้นได้มากขึ้นและทำให้การผสานข้อมูลที่สร้างโดย AI เข้ากับขั้นตอนการทำงานของแอปพลิเคชันง่ายขึ้น

การดำเนินการของฟังก์ชัน

คุณได้กำหนดฟังก์ชัน และกระตุ้น AI ของคุณ ซึ่ง AI ตัดสินใจว่าควรเรียกใช้ฟังก์ชันใดฟังก์ชันหนึ่งของคุณ ตอนนี้ถึงเวลาที่โค้ดแอปพลิเคชันของคุณหรือไลบรารี หากคุณใช้ Ruby gem เช่น raix-rails จะส่งการเรียกใช้ฟังก์ชันและพารามิเตอร์ไปยังการนำไปใช้ที่สอดคล้องกัน *ในโค้ดแอปพลิเคชันของคุณ*

โค้ดแอปพลิเคชันของคุณจะตัดสินใจว่าจะทำอะไรกับผลลัพธ์ของการดำเนินการฟังก์ชัน สิ่งที่ต้องทำอาจเกี่ยวข้องกับโค้ดบรรทัดเดียวใน lambda หรืออาจเกี่ยวข้องกับการเรียก API ภายนอก อาจเกี่ยวข้องกับการเรียกคอมโพเนนต์ AI อื่น หรืออาจเกี่ยวข้องกับโค้ดหลายร้อยหรือแม้แต่หลายพันบรรทัดในส่วนที่เหลือของระบบของคุณ ขึ้นอยู่กับคุณทั้งหมด

บางครั้งการเรียกใช้ฟังก์ชันเป็นการสิ้นสุดการดำเนินการ แต่ถ้าผลลัพธ์แสดงถึงข้อมูลในลำดับความคิด ที่จะดำเนินการต่อโดย AI โค้ดแอปพลิเคชันของคุณจำเป็นต้องแทรกผลการดำเนินการลงในบันทึกการแชทและปล่อยให้ AI ประมวลผลต่อไป

ตัวอย่างเช่น นี่คือการประกาศฟังก์ชัน Raix ที่ใช้โดย AccountManager ของ Olympia เพื่อสื่อสารกับลูกค้าของเราในฐานะส่วนหนึ่งของการจัดการลำดับการทำงานอัจฉริยะสำหรับการบริการลูกค้า

```
1  class AccountManager
2    include Raix::ChatCompletion
3    include Raix::FunctionDispatch
4
5    # lots of other functions...
6
7    function :notify_account_owner,
8      "Don't share UUID. Mention dollars if subscription changed",
9      message: { type: "string" } do |arguments|
10       account.owner.freeform_notify(
11         subject: "Account Change Notification",
12         message: arguments[:message]
13       )
14       "Notified account owner"
15     end
```

อาจจะไม่เห็นได้ชัดในทันทีว่ากำลังเกิดอะไรขึ้น ดังนั้นผมจะอธิบายแยกย่อยให้

1. คลาส AccountManager กำหนดฟังก์ชันมากมายที่เกี่ยวข้องกับการจัดการบัญชี มันสามารถเปลี่ยนแผนการใช้งาน เพิ่มและลบสมาชิกในทีม และอื่นๆ
2. คำสั่งระดับบนสุดบอก AccountManager ว่าควรแจ้งเจ้าของบัญชีเกี่ยวกับผลลัพธ์ของคำขอเปลี่ยนแปลงบัญชี โดยใช้ฟังก์ชัน notify_account_owner
3. คำจำกัดความที่กระชับของฟังก์ชันประกอบด้วย:

- ชื่อ
- คำอธิบาย
- พารามิเตอร์ message: { type: ”string” }
- บล็อกคำสั่งที่จะทำงานเมื่อเรียกใช้ฟังก์ชัน

หลังจากอัปเดตบันทึกการสนทนาด้วยผลลัพธ์ของบล็อกฟังก์ชันแล้ว เมธอด chat_completion จะถูกเรียกใช้อีกครั้ง เมธอดนี้มีหน้าที่ส่งบันทึกการสนทนาที่อัปเดตแล้วกลับไปยังโมเดล AI เพื่อประมวลผลต่อ เราเรียกกระบวนการนี้ว่า *การวนรอบการสนทนา*

เมื่อโมเดล AI ได้รับคำขอ chat completion ใหม่พร้อมบันทึกการสนทนาที่อัปเดตแล้ว มันจะสามารถเข้าถึงผลลัพธ์ของฟังก์ชันที่ทำงานก่อนหน้านี้ได้ มันสามารถวิเคราะห์ผลลัพธ์เหล่านี้ นำมาใช้ในกระบวนการตัดสินใจ และสร้างการตอบสนองหรือการกระทำถัดไปโดยอิงจากบริบทการสนทนาทั้งหมด มันสามารถเลือกที่จะเรียกใช้ฟังก์ชันเพิ่มเติมตามบริบทที่อัปเดต หรือสร้างการตอบสนองสุดท้ายให้กับคำถามเริ่มต้นหากพิจารณาว่าไม่จำเป็นต้องเรียกใช้ฟังก์ชันเพิ่มเติม

การดำเนินการต่อเนื่องของคำถามเริ่มต้น (ตัวเลือก)

เมื่อคุณส่งผลลัพธ์ของเครื่องมือกลับไปยัง LLM และดำเนินการประมวลผลคำถามเริ่มต้นต่อ AI จะใช้ผลลัพธ์เหล่านั้นเพื่อเรียกใช้ฟังก์ชันเพิ่มเติมหรือสร้างการตอบสนองข้อความธรรมดาสุดท้าย

โมเดลบางตัว เช่น Command-R ของ Cohere สามารถอ้างอิงเครื่องมือเฉพาะที่ใช้ในการตอบสนองได้ ทำให้มีความโปร่งใสและตรวจสอบย้อนกลับได้มากขึ้น

ขึ้นอยู่กับโมเดลที่ใช้ ผลลัพธ์ของการเรียกใช้ฟังก์ชันจะอยู่ในข้อความบันทึกที่มีบทบาทพิเศษของตัวเอง หรือแสดงในรูปแบบไวยากรณ์อื่น แต่สิ่งสำคัญคือข้อมูลนั้นต้องอยู่ในบันทึก เพื่อให้ AI สามารถพิจารณาเมื่อตัดสินใจว่าจะทำอะไรต่อไป

ข้อผิดพลาดที่พบบ่อย (และอาจมีค่าใช้จ่ายสูง) คือการลืมเพิ่มผลลัพธ์ของฟังก์ชันลงในบันทึกก่อนที่จะดำเนินการต่อการสนทนา ผลลัพธ์คือ AI จะได้รับคำสั่งในลักษณะเดียวกับที่ได้รับก่อนที่จะเรียกใช้ฟังก์ชันครั้งแรก กล่าวคือ ในมุมมองของ AI มันยังไม่ได้เรียกใช้ฟังก์ชัน ดังนั้นมันจึงเรียกใช้อีกครั้ง และอีกครั้ง และอีกครั้งไปเรื่อยๆ จนกว่าคุณจะหยุดมัน หวังว่าบริบทของคุณจะไม่ใหญ่เกินไป และโมเดลของคุณจะไม่แพงเกินไป!

แนวปฏิบัติที่ดีที่สุดสำหรับการใช้เครื่องมือ

เพื่อให้ได้ประโยชน์สูงสุดจากการใช้เครื่องมือ ให้พิจารณาแนวปฏิบัติที่ดีที่สุดต่อไปนี้

คำจำกัดความที่บรรยายได้ดี

ให้ชื่อและคำอธิบายที่ชัดเจนและบรรยายได้ดีสำหรับแต่ละเครื่องมือและพารามิเตอร์อินพุตของมัน นี่จะช่วยให้ LLM เข้าใจจุดประสงค์และความสามารถของแต่ละเครื่องมือได้ดีขึ้น

> ผมบอกได้จากประสบการณ์ว่าความเชื่อทั่วไปที่ว่า “การตั้งชื่อนั้นยาก” ใช้ได้ที่นี่ด้วย ผมเห็นผลลัพธ์ที่แตกต่างกันอย่างมากจาก LLM เพียงแค่เปลี่ยนชื่อของฟังก์ชันหรือการใช้คำในคำอธิบาย บางครั้งการลบคำอธิบายออกกลับ*ปรับปรุง*ประสิทธิภาพ

การประมวลผลผลลัพธ์ของเครื่องมือ

เมื่อส่งผลลัพธ์ของเครื่องมือกลับไปยัง LLM ต้องแน่ใจว่ามีโครงสร้างที่ดีและครอบคลุม ใช้คีย์และค่าที่มีความหมายเพื่อแสดงผลลัพธ์ของแต่ละเครื่องมือ ทดลองใช้รูปแบบต่างๆ และดูว่าแบบไหนทำงานได้ดีที่สุด ตั้งแต่ JSON ไปจนถึงข้อความธรรมดา

Result Interpreter จัดการ กับ ความ ท้าทาย นี้ โดย ใช้ AI ใน การ วิเคราะห์ ผลลัพธ์ และ ให้ คำ อธิบาย สรุป หรือประเด็นสำคัญที่เข้าใจง่ายสำหรับมนุษย์

การจัดการข้อผิดพลาด

ใช้กลไกการจัดการข้อผิดพลาดที่แข็งแกร่งเพื่อจัดการกรณีที่ LLM อาจสร้างพารามิเตอร์อินพุตที่ไม่ถูกต้องหรือไม่รองรับสำหรับการเรียกใช้เครื่องมือ จัดการและกู้คืนจากข้อผิดพลาดที่อาจเกิดขึ้นระหว่างการทำงานของเครื่องมืออย่างราบรื่น

คุณสมบัติที่ดีมากอย่างหนึ่งของ AI คือมันเข้าใจข้อความแสดงข้อผิดพลาด! ซึ่งหมายความว่าถ้าคุณกำลังทำงานแบบรวดเร็วและไม่พิถีพิถัน คุณสามารถดักจับข้อยกเว้นที่สร้างขึ้นในการใช้งานเครื่องมือ และส่งกลับไปยัง AI เพื่อให้มันรู้ว่าเกิดอะไรขึ้น!

ตัวอย่างเช่น นี่คือเวอร์ชันที่ลดทอนลงของการใช้งาน google search ใน Olympia:

```
def google_search(conversation, params)
  conversation.update_cstatus("Searching Google...")
  query = params[:query]
  search = GoogleSearch.new(query).get_hash

  conversation.update_cstatus("Summarizing results...")
  SummarizeKnowledgeGraph.new.perform(conversation, search.to_json)
rescue StandardError => e
  Honeybadger.notify(e)
  { error: e.message }.inspect
end
```

การค้นหา Google ใน Olympia เป็นกระบวนการสองขั้นตอน ขั้นแรกคือการค้นหา จากนั้นจึงสรุปผลลัพธ์ หากเกิดข้อผิดพลาด ไม่ว่าจะเป็นอะไรก็ตาม ข้อความแสดงข้อผิดพลาดจะถูกห่อและส่งกลับไปยัง AI เทคนิคนี้เป็นรากฐานของรูปแบบ *การจัดการข้อผิดพลาดอัจฉริยะ* เกือบทั้งหมด

ตัวอย่างเช่น สมมติว่าการเรียก API GoogleSearch ล้มเหลวเนื่องจากข้อผิดพลาด 503 Service Unavailable ข้อผิดพลาดนี้จะถูกส่งขึ้นไปยังการดักจับข้อผิดพลาดระดับบนสุด และคำอธิบายของข้อผิดพลาดจะถูกส่งกลับไปยัง AI เป็นผลลัพธ์ของการเรียกฟังก์ชัน แทนที่จะแสดงหน้าจอว่างเปล่าหรือข้อผิดพลาดทางเทคนิคให้ผู้ใช้ AI จะพูดอะไรทำนองว่า “ขออภัย ฉันไม่สามารถเข้าถึงความสามารถในการค้นหา Google ได้ในขณะนี้ ฉันสามารถลองใหม่ภายหลังได้หากคุณต้องการ”

อาจดูเหมือนเป็นเพียงเทคนิคที่ชาญฉลาด แต่ลองพิจารณาข้อผิดพลาดอีกประเภทหนึ่ง กรณีที่ AI กำลังเรียกใช้ API ภายนอก และมีการควบคุมพารามิเตอร์ที่จะส่งไปยัง API โดยตรง บางทีมันอาจทำผิดพลาดในการสร้างพารามิเตอร์เหล่านั้น? หากข้อความแสดงข้อผิดพลาดจาก API ภายนอกมีรายละเอียดเพียงพอ การส่งข้อความแสดงข้อผิดพลาดกลับไปยัง AI ที่เรียกใช้ หมายความว่ามันสามารถพิจารณาพารามิเตอร์เหล่านั้นใหม่และลองอีกครั้ง โดยอัตโนมัติ ไม่ว่าข้อผิดพลาดจะเป็นอะไรก็ตาม

ตอนนี้ลองคิดดูว่าต้องใช้อะไรบ้างในการทำซ้ำการจัดการข้อผิดพลาดที่ทนทานแบบนี้ในโค้ด *ปกติ* มันแทบจะเป็นไปไม่ได้เลย

การปรับปรุงแบบวนซ้ำ

หาก LLM ไม่แนะนำเครื่องมือที่เหมาะสมหรือสร้างการตอบสนองที่ไม่เหมาะสม ให้ทำการวนซ้ำในส่วนของคำจำกัดความของเครื่องมือ คำอธิบาย และพารามิเตอร์อินพุต ปรับปรุงและพัฒนาการตั้งค่าเครื่องมืออย่างต่อเนื่องตามพฤติกรรมที่สังเกตได้และผลลัพธ์ที่ต้องการ

1. เริ่มต้นด้วยคำจำกัดความของเครื่องมือที่เรียบง่าย: เริ่มต้นด้วยการกำหนดเครื่องมือที่มีชื่อ คำอธิบาย และพารามิเตอร์อินพุตที่ชัดเจนและกระชับ หลีกเลี่ยงการทำให้การตั้งค่าเครื่องมือซับซ้อนเกินไปในตอนแรกและมุ่งเน้นไปที่ฟังก์ชันการทำงานหลัก ตัวอย่างเช่น หากคุณต้องการบันทึกผลการวิเคราะห์ความรู้สึก ให้เริ่มต้นด้วยคำจำกัดความพื้นฐานเช่น:

```
{
  "name": "save_sentiment_score",
  "description": "Analyze user-provided text and generate sentiment score",
  "parameters": {
    "type": "object",
    "properties": {
      "score": {
        "type": "float",
        "description": "sentiment score from -1 (negative) to 1 (positive)"
      }
    },
    "required": ["score"]
  }
}
```

2. ทดสอบและสังเกต: เมื่อคุณมีคำนิยามเครื่องมือเบื้องต้นพร้อมแล้ว ให้ทดสอบด้วยคำสั่งที่แตกต่างกันและสังเกตว่า LLM มีปฏิสัมพันธ์กับเครื่องมือนั้นอย่างไร ให้ความสนใจกับคุณภาพและความเกี่ยวข้องของคำตอบที่ถูกสร้างขึ้น หาก LLM กำลังสร้างการตอบสนองที่ไม่เหมาะสม ถึงเวลาที่จะต้องปรับปรุงคำนิยามของเครื่องมือ
3. ปรับปรุงคำอธิบาย: หาก LLM เข้าใจจุดประสงค์ของเครื่องมือคลาดเคลื่อน ให้ลองปรับปรุงคำอธิบายของเครื่องมือ โดยเพิ่มบริบท ตัวอย่าง หรือคำอธิบายเพิ่มเติมเพื่อแนะนำให้ LLM ใช้เครื่องมือได้อย่างมีประสิทธิภาพ ตัวอย่างเช่น คุณสามารถอัปเดตคำอธิบายของเครื่องมือวิเคราะห์ความรู้สึกเพื่อระบุถึง*น้ำเสียงทางอารมณ์*ของข้อความที่กำลังวิเคราะห์ได้อย่างเฉพาะเจาะจงมากขึ้น:

```
{
  "name": "save_sentiment_score",
  "description": "Determine the overall emotional tone of a piece of text,
   such as customer reviews, social media posts, or feedback comments.",
  ...
}
```

4. ปรับพารามิเตอร์อินพุต: หาก LLM กำลังสร้างพารามิเตอร์อินพุตที่ไม่ถูกต้องหรือไม่เกี่ยวข้องสำหรับเครื่องมือ ให้พิจารณาปรับคำจำกัดความของพารามิเตอร์ เพิ่มข้อจำกัดที่เฉพาะเจาะจง กฎการตรวจสอบความถูกต้อง หรือตัวอย่างเพื่อให้รูปแบบอินพุตที่ต้องการมีความชัดเจนยิ่งขึ้น
5. ปรับปรุงตามผลตอบรับ: ติดตามประสิทธิภาพของเครื่องมือของคุณอย่างต่อเนื่องและรวบรวมผลตอบรับจากผู้ใช้หรือผู้มีส่วนได้ส่วนเสีย ใช้ผลตอบรับนี้เพื่อระบุพื้นที่ที่ต้องการการปรับปรุงและทำการปรับแต่งคำจำกัดความของเครื่องมืออย่างต่อเนื่อง ตัวอย่างเช่น หากผู้ใช้รายงานว่าการวิเคราะห์ไม่สามารถจัดการกับการเสียดสีได้ดี คุณสามารถเพิ่มบันทึกในคำอธิบาย:

```
{
  "name": "save_sentiment_score",
  "description": "Analyze the sentiment of a given text and return a sentiment
   score between -1 (negative) and 1 (positive). Note: Sarcasm should be
   considered negative.",
  ...
}
```

ด้วยการปรับแต่งคำจำกัดความของเครื่องมือซ้ำๆ ตามพฤติกรรมที่สังเกตได้และข้อเสนอแนะ คุณสามารถค่อยๆ ปรับปรุงประสิทธิภาพและประสิทธิผลของแอปพลิเคชันที่ขับเคลื่อนด้วย AI ของคุณได้ อย่าลืมรักษาคำจำกัดความของเครื่องมือให้ชัดเจน กระชับ และมุ่งเน้นไปที่งานเฉพาะที่ต้องการ ทดสอบและตรวจสอบการทำงานร่วมกันของเครื่องมืออย่างสม่ำเสมอเพื่อให้แน่ใจว่าสอดคล้องกับผลลัพธ์ที่คุณต้องการ

การประกอบและการเชื่อมโยงเครื่องมือ

หนึ่งในแง่มุมที่ทรงพลังที่สุดของการใช้เครื่องมือที่เพียงแค่กล่าวถึงจนถึงตอนนี้คือความสามารถในการประกอบและเชื่อมโยงเครื่องมือหลายๆ อย่างเข้าด้วยกันเพื่อทำงานที่ซับซ้อน ด้วยการออกแบบคำจำกัดความของเครื่องมือและรูปแบบอินพุต/เอาต์พุตอย่างระมัดระวัง คุณสามารถสร้างบล็อกที่นำกลับมาใช้ใหม่ได้ซึ่งสามารถนำมารวมกันได้หลากหลายวิธี

มาพิจารณาตัวอย่างที่คุณกำลังสร้างไปป์ไลน์การวิเคราะห์ข้อมูลสำหรับแอปพลิเคชันที่ขับเคลื่อนด้วย AI คุณอาจมีเครื่องมือต่อไปนี้:

1. **DataRetrieval:** เครื่องมือที่ดึงข้อมูลจากฐานข้อมูลหรือ API ตามเกณฑ์ที่กำหนด
2. **DataProcessing:** เครื่องมือที่ทำการคำนวณ แปลงข้อมูล หรือรวมข้อมูลที่ดึงมา
3. **DataVisualization:** เครื่องมือที่นำเสนอข้อมูลที่ประมวลผลแล้วในรูปแบบที่เป็นมิตรกับผู้ใช้ เช่น แผนภูมิหรือกราฟ

ด้วยการเชื่อมโยงเครื่องมือเหล่านี้เข้าด้วยกัน คุณสามารถสร้างขั้นตอนการทำงานที่ทรงพลังซึ่งดึงข้อมูลที่เกี่ยวข้อง ประมวลผล และนำเสนอผลลัพธ์ในรูปแบบที่มีความหมาย นี่คือตัวอย่างขั้นตอนการใช้เครื่องมือ:

1. LLM ได้รับคำถามจากผู้ใช้ที่ขอข้อมูลเชิงลึกเกี่ยวกับข้อมูลการขายสำหรับหมวดหมู่ผลิตภัณฑ์เฉพาะ
2. LLM เลือกเครื่องมือ DataRetrieval และสร้างพารามิเตอร์อินพุตที่เหมาะสมเพื่อดึงข้อมูลการขายที่เกี่ยวข้องจากฐานข้อมูล
3. ข้อมูลที่ดึงมาจะถูก “ส่งต่อ” ไปยังเครื่องมือ DataProcessing ซึ่งคำนวณตัวชี้วัดต่างๆ เช่น รายได้รวม ราคาขายเฉลี่ย และอัตราการเติบโต
4. จากนั้นข้อมูลที่ประมวลผลแล้วจะถูกนำไปใช้โดยเครื่องมือ DataVisualization ซึ่งสร้างแผนภูมิหรือกราฟที่น่าดึงดูดเพื่อแสดงข้อมูลเชิงลึก โดยส่ง URL ของแผนภูมิกลับไปยัง LLM
5. สุดท้าย LLM สร้างการตอบสนองที่จัดรูปแบบแล้วสำหรับคำถามของผู้ใช้โดยใช้ markdown รวมข้อมูลที่แสดงเป็นภาพ และให้สรุปผลสำคัญ

ด้วยการประกอบเครื่องมือเหล่านี้เข้าด้วยกัน คุณสามารถสร้างขั้นตอนการวิเคราะห์ข้อมูลที่ราบรื่นซึ่งสามารถผสานรวมเข้ากับแอปพลิเคชันของคุณได้อย่างความงดงามของวิธีการนี้คือเครื่องมือแต่ละอันสามารถพัฒนาและทดสอบแยกกันได้ และจากนั้นนำมารวมกันในวิธีต่างๆ เพื่อแก้ปัญหาที่หลากหลาย

เพื่อให้การประกอบและเชื่อมโยงเครื่องมือเป็นไปอย่างราบรื่น สิ่งสำคัญคือต้องกำหนดรูปแบบอินพุตและเอาต์พุตที่ชัดเจนสำหรับแต่ละเครื่องมือ

ตัวอย่างเช่น เครื่องมือ DataRetrieval อาจรับพารามิเตอร์ เช่น รายละเอียดการเชื่อมต่อฐานข้อมูล ชื่อตาราง และเงื่อนไขการค้นหา และส่งคืนชุดผลลัพธ์เป็นออบเจกต์ JSON ที่มีโครงสร้าง จากนั้นเครื่องมือ DataProcessing สามารถรับออบเจกต์ JSON นี้เป็นอินพุตและสร้างออบเจกต์ JSON ที่แปลงแล้วเป็นเอาต์พุต การทำให้การไหลของข้อมูลระหว่างเครื่องมือเป็นมาตรฐาน คุณสามารถรับประกันความเข้ากันได้และการนำกลับมาใช้ใหม่

เมื่อคุณออกแบบระบบนิเวศของเครื่องมือ ให้คิดว่าเครื่องมือต่างๆ สามารถรวมกันเพื่อจัดการกับกรณีการใช้งานทั่วไปในแอปพลิเคชันของคุณได้อย่างไร พิจารณาสร้างเครื่องมือระดับสูงที่รวมขั้นตอนการทำงานทั่วไปหรือตรรกะทางธุรกิจ ทำให้ง่ายขึ้นสำหรับ LLM ในการเลือกและใช้งานอย่างมีประสิทธิภาพ

จำไว้ว่า พลังของการใช้เครื่องมืออยู่ที่ความยืดหยุ่นและความเป็นโมดูลาร์ที่มันมอบให้ ด้วยการแบ่งงานที่ซับซ้อนเป็นเครื่องมือขนาดเล็กที่นำกลับมาใช้ใหม่ได้ คุณสามารถสร้างแอปพลิเคชันที่ขับเคลื่อนด้วย AI ที่แข็งแกร่งและปรับตัวได้ซึ่งสามารถจัดการกับความท้าทายที่หลากหลายได้

ทิศทางในอนาคต

เมื่อการพัฒนาแอปพลิเคชันที่ขับเคลื่อนด้วย AI มีวิวัฒนาการ เราสามารถคาดหวังความก้าวหน้าเพิ่มเติมในความสามารถการใช้เครื่องมือ ทิศทางในอนาคตที่เป็นไปได้บางอย่างได้แก่:

1. **การใช้เครื่องมือแบบหลายขั้นตอน**: LLM อาจสามารถตัดสินใจว่าต้องใช้เครื่องมือกี่ครั้งเพื่อสร้างการตอบสนองที่น่าพอใจ ซึ่งอาจเกี่ยวข้องกับการเลือกและการดำเนินการใช้เครื่องมือหลายรอบตามผลลัพธ์ระหว่างทาง
2. **เครื่องมือที่กำหนดไว้ล่วงหน้า**: แพลตฟอร์ม AI อาจให้ชุดเครื่องมือที่กำหนดไว้ล่วงหน้าที่นักพัฒนาสามารถใช้ประโยชน์ได้ทันที เช่น ตัวแปลภาษา Python เครื่องมือค้นหาเว็บ หรือฟังก์ชันยูทิลิตี้ทั่วไป
3. **การผสานรวมที่ราบรื่น**: เมื่อการใช้เครื่องมือแพร่หลายมากขึ้น เราสามารถคาดหวังการผสานรวมที่ดีขึ้นระหว่างแพลตฟอร์ม AI และเฟรมเวิร์กการพัฒนายอดนิยม ทำให้นักพัฒนาสามารถรวมการใช้เครื่องมือเข้าในแอปพลิเคชันของพวกเขาได้ง่ายขึ้น

การใช้เครื่องมือเป็นเทคนิคที่ทรงพลังที่ช่วยให้นักพัฒนาสามารถใช้ประโยชน์จากศักยภาพเต็มรูปแบบของ LLM ในแอปพลิเคชันที่ขับเคลื่อนด้วย AI ด้วยการเชื่อมต่อ LLM กับเครื่องมือและทรัพยากรภายนอก คุณสามารถสร้างระบบที่มีความไดนามิก ฉลาด และตระหนักถึงบริบทมากขึ้น ซึ่งสามารถปรับตัวตามความต้องการของผู้ใช้และให้ข้อมูลเชิงลึกและการดำเนินการที่มีคุณค่า

ในขณะที่การใช้เครื่องมือเปิดโอกาสมากมาย สิ่งสำคัญคือต้องตระหนักถึงความท้าทายและข้อควรพิจารณาที่อาจเกิดขึ้น หนึ่งในแง่มุมสำคัญคือการจัดการความซับซ้อนของการทำงานร่วมกันของเครื่องมือและการรับประกันความเสถียรและความน่าเชื่อถือของระบบโดยรวม คุณจำเป็นต้องจัดการสถานการณ์ที่การเรียกใช้เครื่องมืออาจล้มเหลว ส่งคืนผลลัพธ์ที่ไม่คาดคิด หรือมีผลกระทบต่อประสิทธิภาพ นอกจากนี้ คุณควรพิจารณามาตรการรักษาความปลอดภัยและการควบคุมการเข้าถึงเพื่อป้องกันการใช้

เครื่องมือโดยไม่ได้รับอนุญาตหรือเป็นอันตราย กลไกการจัดการข้อผิดพลาด การบันทึก และการตรวจสอบที่เหมาะสมมีความสำคัญอย่างยิ่งในการรักษาความสมบูรณ์และประสิทธิภาพของแอปพลิเคชันที่ขับเคลื่อนด้วย AI ของคุณ

ขณะที่คุณสำรวจความเป็นไปได้ของการใช้เครื่องมือในโครงการของคุณ จงจำไว้ว่าให้เริ่มต้นด้วยวัตถุประสงค์ที่ชัดเจน ออกแบบคำจำกัดความของเครื่องมือที่มีโครงสร้างที่ดี และปรับปรุงซ้ำตามข้อมูลตอบกลับและผลลัพธ์ที่ได้ ด้วยวิธีการและทัศนคติที่ถูกต้อง การใช้เครื่องมือสามารถปลดล็อกระดับใหม่ของนวัตกรรมและคุณค่าในแอปพลิเคชันที่ขับเคลื่อนด้วย AI ของคุณ

การประมวลผลสตรีม

การส่งข้อมูลสตรีมผ่าน HTTP หรือที่รู้จักในชื่อเหตุการณ์ที่ส่งจากเซิร์ฟเวอร์ (SSE) คือกลไกที่เซิร์ฟเวอร์ส่งข้อมูลไปยังไคลเอนต์อย่างต่อเนื่องเมื่อมีข้อมูลพร้อมใช้งาน โดยที่ไคลเอนต์ไม่จำเป็นต้องร้องขอข้อมูลอย่างชัดแจ้ง เนื่องจากการตอบสนองของ AI ถูกสร้างขึ้นทีละส่วน จึงเป็นเรื่องที่สมเหตุสมผลที่จะมอบประสบการณ์ผู้ใช้ที่ตอบสนองได้ดีด้วยการแสดงผลลัพธ์ของ AI ในขณะที่กำลังถูกสร้างขึ้น และจริงๆ แล้ว API ของผู้ให้บริการ AI ทุกรายที่ผมรู้จักต่างก็มีตัวเลือกการตอบสนองแบบสตรีมในจุดสิ้นสุดการทำงาน

เหตุผลที่บทนี้ปรากฏในหนังสือตรงนี้ ถัดจากบท การใช้เครื่องมือ เป็นเพราะพลังอันมหาศาลที่เกิดจากการผสมผสานการใช้เครื่องมือกับการตอบสนองแบบสดของ AI ให้กับผู้ใช้ การทำเช่นนี้ช่วยให้เกิดประสบการณ์แบบไดนามิกและมีปฏิสัมพันธ์ ซึ่ง AI สามารถประมวลผลข้อมูลที่ผู้ใช้ป้อนเข้ามา ใช้เครื่องมือและฟังก์ชันต่างๆ ตามที่เห็นสมควร และให้การตอบสนองแบบเรียลไทม์

เพื่อให้ได้การโต้ตอบที่ราบรื่นนี้ คุณจำเป็นต้องเขียนตัวจัดการสตรีมที่สามารถจัดการทั้งการเรียกใช้ฟังก์ชันเครื่องมือที่ AI เรียกใช้และการส่งข้อความธรรมดาไปยังผู้ใช้ปลายทาง ความจำเป็นในการวนลูปหลังจากประมวลผลฟังก์ชันเครื่องมือเพิ่มความท้าทายที่น่าสนใจให้กับงานนี้

การพัฒนา ReplyStream

เพื่อสาธิตวิธีการพัฒนาการประมวลผลสตรีม บทนี้จะพาไปดูเชิงลึกถึงเวอร์ชันที่ถูกทำให้ง่ายขึ้นของคลาส ReplyStream ที่ใช้ใน Olympia อินสแตนซ์ของคลาสนี้สามารถถูกส่งผ่านเป็นพารามิเตอร์ stream ในไลบรารีโคลเอนต์ AI เช่น ruby-openai และ openrouter

นี่คือวิธีที่ผมใช้ ReplyStream ใน PromptSubscriber ของ Olympia ซึ่งใช้ Wisper ในการรอฟังการสร้างข้อความใหม่จากผู้ใช้

```
class PromptSubscriber
  include Raix::ChatCompletion
  include Raix::PromptDeclarations

  # many other declarations omitted...

  prompt text: -> { user_message.content },
         stream: -> { ReplyStream.new(self) },
         until: -> { bot_message.complete? }

  def message_created(message) # invoked by Wisper
    return unless message.role.user? && message.content?

    # rest of the implementation omitted...
```

นอกเหนือจากการอ้างอิง context ไปยังจุดเชื่อมต่อตัวรับข้อความที่สร้างมันขึ้นมาแล้ว คลาส ReplyStream ยังมีตัวแปรอินสแตนซ์สำหรับเก็บบัฟเฟอร์ของข้อมูลที่ได้รับ และอาร์เรย์สำหรับติดตามชื่อฟังก์ชันและอาร์กิวเมนต์ที่ถูกเรียกใช้ระหว่างการประมวลผลสตรีม

```ruby
class ReplyStream
  attr_accessor :buffer, :f_name, :f_arguments, :context

  delegate :bot_message, :dispatch, to: :context

  def initialize(context)
    self.context = context
    self.buffer = []
    self.f_name = []
    self.f_arguments = []
  end

  def call(chunk, bytesize = nil)
    # ...
  end

  # ...
end
```

เมธอด initialize ทำหน้าที่กำหนดค่าเริ่มต้นให้กับอินสแตนซ์ของ ReplyStream โดยเริ่มต้นค่าของบัฟเฟอร์ บริบท และตัวแปรอื่นๆ

เมธอด call เป็นจุดเริ่มต้นหลักในการประมวลผลข้อมูลแบบสตรีม โดยจะรับพารามิเตอร์ chunk (อยู่ในรูปแบบของแฮช) และพารามิเตอร์ bytesize ที่เป็นตัวเลือก ซึ่งในตัวอย่างของเราไม่ได้ใช้ ภายในเมธอดนี้ คลาสใช้การจับคู่รูปแบบเพื่อจัดการกับสถานการณ์ต่างๆ ตามโครงสร้างของ chunk ที่ได้รับ

การเรียกใช้ deep_symbolize_keys กับ chunk ช่วยให้การจับคู่รูปแบบดูสวยงามขึ้น เพราะทำให้เราสามารถทำงานกับสัญลักษณ์แทนที่จะเป็นสตริง

```
def call(chunk, _bytesize)
  case chunk.deep_symbolize_keys

  in { # match function name
    choices: [
      {
        delta: {
          tool_calls: [
            { index: index, function: {name: name} }
          ]
        }
      }
    ] }

    f_name[index] = name
```

รูปแบบแรกที่เราจับคู่คือการเรียกใช้เครื่องมือพร้อมกับชื่อฟังก์ชันที่เกี่ยวข้อง หากเราตรวจพบการเรียกใช้ เราจะเก็บมันไว้ในอาร์เรย์ f_name เราเก็บชื่อฟังก์ชันในอาร์เรย์แบบมีดัชนี เนื่องจากโมเดลสามารถเรียกใช้ฟังก์ชันแบบขนานได้ โดยส่งฟังก์ชันหลายตัวไปทำงานพร้อมกัน

> การเรียกใช้ฟังก์ชันแบบขนานคือความสามารถของโมเดล AI ในการดำเนินการเรียกใช้ฟังก์ชันหลายตัวพร้อมกัน ซึ่งช่วยให้ผลลัพธ์ของการเรียกใช้ฟังก์ชันเหล่านี้ถูกประมวลผลไปพร้อมกันได้ สิ่งนี้มีประโยชน์อย่างมากโดยเฉพาะในกรณีที่ฟังก์ชันใช้เวลานาน และช่วยลดการติดต่อกับ API ซึ่งจะช่วยประหยัดการใช้โทเคนได้อย่างมีนัยสำคัญ

ต่อไปเราต้องจับคู่อาร์กิวเมนต์ที่สอดคล้องกับการเรียกใช้ฟังก์ชันเหล่านี้

```
in { # match arguments
  choices: [
   {
     delta: {
       tool_calls: [
         {
           index: index, function: {arguments: argument }
         }
       ]
     }
   }
  ]}

  f_arguments[index] ||= "" # initialize if not already
  f_arguments[index] << argument
```

คล้ายกับวิธีที่เราจัดการกับชื่อฟังก์ชัน เราจะเก็บอาร์กิวเมนต์ไว้ในอาเรย์แบบมีดัชนี

ถัดไป เราจะมองหาข้อความที่แสดงต่อผู้ใช้ตามปกติ ซึ่งจะถูกส่งมาจากเซิร์ฟเวอร์ทีละโทเค็นและถูกกำหนดให้กับตัวแปร new_content เราจำเป็นต้องคอยสังเกต finish_reason ด้วย ซึ่งจะมีค่าเป็น nil จนกว่าจะถึงชิ้นส่วนสุดท้ายของลำดับเอาต์พุต

```
in {
  choices: [
   { delta: {content: new_content}, finish_reason: finish_reason }
  ]}

  # you could transmit every chunk to the user here...
  buffer << new_content.to_s

  if finish_reason.present?
    finalize
  elsif new_content.to_s.match?(/\n\n/)
    send_to_client # ...or buffer and transmit once per paragraph
  end
```

สิ่งสำคัญคือ เราเพิ่มการแสดงออกของการจับคู่รูปแบบเพื่อจัดการกับข้อความแสดงข้อผิดพลาดที่ส่งมาจากผู้ให้บริการโมเดล AI ในสภาพแวดล้อมการพัฒนาในเครื่อง เราจะสร้างข้อยกเว้น แต่ในการใช้งานจริง เราจะบันทึกข้อผิดพลาดและจบการทำงาน

```
  in { error: { message: } }
    if Rails.env.local?
      raise message
    else
      Honeybadger.notify("AI Error: #{message}")
      finalize
    end
```

else clause สุดท้ายของ case จะทำงานเมื่อไม่มีรูปแบบก่อนหน้านี้ตรงกับเงื่อนไขใดๆ มันเป็นเพียงการป้องกันเพื่อให้เราทราบในกรณีที่โมเดล AI เริ่มส่งชิ้นส่วนข้อมูลที่เราไม่รู้จักมาให้เรา

```
  else
    Honeybadger.notify("Unrecognized Chunk: #{chunk}")
  end
end
```

เมธอด send_to_client มีหน้าที่ส่งเนื้อหาที่อยู่ในบัฟเฟอร์ไปยังไคลเอนต์ โดยจะทำการตรวจสอบว่าบัฟเฟอร์ไม่ว่างเปล่า อัปเดตเนื้อหาข้อความของบอท เรนเดอร์ข้อความของบอท และบันทึกเนื้อหาลงในฐานข้อมูลเพื่อให้แน่ใจว่าข้อมูลจะคงอยู่อย่างถาวร

```
def send_to_client
  # no need to process pure whitespace
  return if buffer.join.squish.blank?

  # set the buffer content on the bot message
  content = buffer.join
  bot_message.content = content

  # save to database so that we never lose data
  # even if the stream doesn't terminate correctly
  bot_message.update_column(:content, content)

  # update content via websocket
  ConversationRenderer.update(bot_message)
end
```

เมธอด finalize จะถูกเรียกใช้เมื่อการประมวลผลสตรีมเสร็จสมบูรณ์ มันจะจัดการกับการเรียกใช้ฟังก์ชันต่างๆ ที่ได้รับระหว่างสตรีม อัปเดตข้อความบอทด้วยเนื้อหาสุดท้ายและข้อมูลที่เกี่ยวข้องอื่นๆ และรีเซ็ตประวัติการเรียกใช้ฟังก์ชัน

```
def finalize
  if f_name.any?
    f_name.each_with_index do |name, index|
      # takes care of calling the function wherever it's implemented
      dispatch(name:, arguments: JSON.parse(f_arguments[index]))
    end

    # reset the function call history
    f_name.clear
    f_arguments.clear
  else
    content = buffer.join.presence
    bot_message.update!(content:, complete: true)
    ConversationRenderer.update(bot_message)
  end
end
```

หากแบบจำลองตัดสินใจที่จะเรียกใช้ฟังก์ชัน คุณจำเป็นต้อง “ส่งต่อ” การเรียกใช้ฟังก์ชันนั้น (ชื่อและอาร์กิวเมนต์) ในลักษณะที่ทำให้มันถูกประมวลผล และมีการเพิ่มข้อความ function_call และ function_result เข้าไปในบันทึกการสนทนา

จากประสบการณ์ของผม การจัดการสร้างข้อความของฟังก์ชันในที่เดียวในฐานโค้ดของคุณนั้นดีกว่าการพึ่งพาการดำเนินการของเครื่องมือ นอกจากจะทำให้โค้ดสะอาดขึ้นแล้ว ยังมีเหตุผลสำคัญในทางปฏิบัติด้วย: หากแบบจำลอง AI เรียกใช้ฟังก์ชัน และไม่เห็นข้อความการเรียกและผลลัพธ์ในบันทึกเมื่อคุณวนลูป *มันจะเรียกใช้ฟังก์ชันเดิมอีกครั้ง* ซึ่งอาจเกิดขึ้นไม่สิ้นสุด อย่าลืมว่า AI นั้นเป็นแบบไร้สถานะ ดังนั้นถ้าคุณไม่สะท้อนการเรียกใช้ฟังก์ชันเหล่านั้นกลับไปให้มัน ก็เท่ากับว่าการเรียกใช้นั้นไม่เคยเกิดขึ้น

```
# PromptSubscriber#dispatch

def dispatch(name:, arguments:)
  # adds a function_call message to the conversation transcript
  # plus dispatches to tool and returns result
  conversation.function_call!(name, arguments).then do |result|
    # add function result message to the transcript
    conversation.function_result!(name, result)
  end
end
```

การล้างประวัติการเรียกฟังก์ชันหลังจากการส่งคำสั่งมีความสำคัญพอๆ กับการทำให้แน่ใจว่าการเรียกและผลลัพธ์ถูกบันทึกลงในบันทึกการสนทนาของคุณ เพื่อที่คุณจะไม่เรียกฟังก์ชันเดิมซ้ำๆ ทุกครั้งที่วนลูป

“ลูปการสนทนา”

ในคลาส PromptSubscriber เราใช้เมธอด prompt จากโมดูล PromptDeclarations เพื่อกำหนดพฤติกรรมของลูปการสนทนา พารามิเตอร์ until ถูกตั้งค่าเป็น -> { bot_message.complete? } ซึ่งหมายความว่าลูปจะดำเนินต่อไปจนกว่า bot_message จะถูกทำเครื่องหมายว่าสมบูรณ์

```
1  prompt text: -> { user_message.content },
2      stream: -> { ReplyStream.new(self) },
3      until: -> { bot_message.complete? }
```

แต่ bot_message จะถูกทำเครื่องหมายว่าเสร็จสมบูรณ์เมื่อไหร่? หากคุณลืม ให้ย้อนกลับไปดูบรรทัดที่ 13 ของเมธอด finalize

มาทบทวนตรรกะการประมวลผลสตรีมทั้งหมดกัน

1. PromptSubscriber ได้รับข้อความใหม่จากผู้ใช้ผ่านเมธอด message_created ซึ่งถูกเรียกโดยระบบ Wisper pub/sub ทุกครั้งที่ผู้ใช้สร้างพรอมต์ใหม่
2. เมธอด prompt แบบคลาสกำหนดพฤติกรรมของตรรกะการสร้างการสนทนาที่สมบูรณ์สำหรับ PromptSubscriber โมเดล AI จะดำเนินการสร้างการสนทนาที่สมบูรณ์พร้อมกับเนื้อหาข้อความของผู้ใช้ อินสแตนซ์ใหม่ของ ReplyStream เป็นพารามิเตอร์สตรีม และเงื่อนไขการวนซ้ำที่กำหนด
3. โมเดล AI ประมวลผลพรอมต์และเริ่มสร้างการตอบสนอง ขณะที่การตอบสนองถูกสตรีม เมธอด call ของอินสแตนซ์ ReplyStream จะถูกเรียกสำหรับข้อมูลแต่ละส่วน
4. หากโมเดล AI ตัดสินใจเรียกฟังก์ชันเครื่องมือ ชื่อฟังก์ชันและอาร์กิวเมนต์จะถูกแยกออกจากชิ้นส่วนและเก็บไว้ในอาร์เรย์ f_name และ f_arguments ตามลำดับ
5. หากโมเดล AI สร้างเนื้อหาที่แสดงต่อผู้ใช้ เนื้อหานั้นจะถูกบัฟเฟอร์และส่งไปยังไคลเอนต์ผ่านเมธอด send_to_client
6. เมื่อการประมวลผลสตรีมเสร็จสิ้น เมธอด finalize จะถูกเรียก หากมีการเรียกใช้ฟังก์ชันเครื่องมือระหว่างสตรีม ฟังก์ชันเหล่านั้นจะถูกส่งไปดำเนินการโดยใช้เมธอด dispatch ของ PromptSubscriber
7. เมธอด dispatch เพิ่มข้อความ function_call ลงในบันทึกการสนทนา ดำเนินการฟังก์ชันเครื่องมือที่เกี่ยวข้อง และเพิ่มข้อความ function_result ลงในบันทึกพร้อมผลลัพธ์ของการเรียกฟังก์ชัน

8. หลังจากส่งฟังก์ชันเครื่องมือไปดำเนินการแล้ว ประวัติการเรียกฟังก์ชันจะถูกล้างเพื่อป้องกันการเรียกฟังก์ชันซ้ำในการวนซ้ำครั้งต่อไป
9. หากไม่มีการเรียกใช้ฟังก์ชันเครื่องมือ เมธอด finalize จะอัปเดต bot_message ด้วยเนื้อหาสุดท้าย ทำเครื่องหมายว่าเสร็จสมบูรณ์ และส่งข้อความที่อัปเดตแล้วไปยังไคลเอนต์
10. เงื่อนไขการวนซ้ำ -> { bot_message.complete? } จะถูกประเมิน หาก bot_message ยังไม่ถูกทำเครื่องหมายว่าเสร็จสมบูรณ์ การวนซ้ำจะดำเนินต่อไป และพรอมต์เดิมจะถูกส่งอีกครั้งพร้อมกับบันทึกการสนทนาที่อัปเดตแล้ว
11. ขั้นตอนที่ 3-10 จะถูกทำซ้ำจนกว่า bot_message จะถูกทำเครื่องหมายว่าเสร็จสมบูรณ์ ซึ่งแสดงว่าโมเดล AI ได้สร้างการตอบสนองเสร็จสิ้นแล้วและไม่จำเป็นต้องดำเนินการฟังก์ชันเครื่องมือเพิ่มเติม

การนำการวนซ้ำการสนทนานี้ไปใช้ ทำให้โมเดล AI สามารถมีปฏิสัมพันธ์โต้ตอบกับแอปพลิเคชัน ดำเนินการฟังก์ชันเครื่องมือตามที่ต้องการ และสร้างการตอบสนองที่แสดงต่อผู้ใช้จนกว่าการสนทนาจะจบลงอย่างเป็นธรรมชาติ

การผสมผสานระหว่างการประมวลผลสตรีมและการวนซ้ำการสนทนาช่วยให้เกิดประสบการณ์ที่ขับเคลื่อนด้วย AI แบบไดนามิกและโต้ตอบได้ ซึ่งโมเดล AI สามารถประมวลผลข้อมูลที่ผู้ใช้ป้อน ใช้เครื่องมือและฟังก์ชันต่างๆ และให้การตอบสนองแบบเรียลไทม์ตามบริบทการสนทนาที่เปลี่ยนแปลงไป

การดำเนินการต่อโดยอัตโนมัติ

สิ่งสำคัญคือต้องตระหนักถึงข้อจำกัดของผลลัพธ์ AI โมเดลส่วนใหญ่มีจำนวนโทเค็นสูงสุดที่สามารถสร้างได้ในการตอบสนองครั้งเดียว ซึ่งถูกกำหนดโดยพารามิเตอร์ max_tokens หากโมเดล AI ถึงขีดจำกัดนี้ในขณะที่กำลังสร้างการตอบสนอง มันจะหยุดทันทีและแสดงว่าผลลัพธ์ถูกตัดทอน

ในการตอบสนองแบบสตรีมจาก API ของแพลตฟอร์ม AI คุณสามารถตรวจจับสถานการณ์นี้ได้โดยการตรวจสอบตัวแปร finish_reason ในชิ้นส่วน หาก finish_reason ถูกตั้งค่าเป็น "length" (หรือค่าคีย์อื่นที่เฉพาะเจาะจงกับโมเดล) แสดงว่าโมเดลถึงขีดจำกัดโทเค็นสูงสุดระหว่างการสร้างและผลลัพธ์ถูกตัดให้สั้นลง

วิธีหนึ่งในการจัดการกับสถานการณ์นี้อย่างราบรื่นและมอบประสบการณ์ที่ต่อเนื่องให้กับผู้ใช้ คือการใช้กลไกการดำเนินการต่อโดยอัตโนมัติในตรรกะการประมวลผลสตรีมของคุณ โดยการเพิ่มรูปแบบการจับคู่สำหรับเหตุผลการสิ้นสุดที่เกี่ยวข้องกับความยาว คุณสามารถเลือกที่จะวนซ้ำและดำเนินการต่อจากจุดที่หยุดไว้โดยอัตโนมัติ

นี่คือตัวอย่างที่ถูกทำให้ง่ายลงโดยเจตนาของวิธีการปรับเปลี่ยนเมธอด call ในคลาส ReplyStream เพื่อรองรับการดำเนินการต่อโดยอัตโนมัติ:

```
LENGTH_STOPS = %w[length MAX_TOKENS]

def call(chunk, _bytesize)
  case chunk.deep_symbolize_keys
    # ...

    in {
      choices: [
        { delta: {content: new_content},
          finish_reason: finish_reason } ] }

      buffer << new_content.to_s

      if finish_reason.blank?
        send_to_client if new_content.to_s.match?(/\n\n/)
      elsif LENGTH_STOPS.include?(finish_reason)
        continue_cutoff
      else
        finalize
      end

    # ...
  end
end

private

def continue_cutoff
  conversation.bot_message!(buffer.join, visible: false)
  conversation.user_message!("please continue", visible: false)
  bot_message.update_column(:created_at, Time.current)
end
```

ในเวอร์ชันที่ปรับปรุงนี้ เมื่อ finish_reason บ่งชี้ว่าผลลัพธ์ถูกตัดออก แทนที่จะจบสตรีมทันที เราจะเพิ่มข้อความคู่หนึ่งลงในบันทึกการสนทนาโดยไม่ทำการจบ ย้ายข้อความการตอบสนองที่แสดงต่อผู้ใช้เดิมไปไว้ที่ “ล่างสุด” ของบันทึกการสนทนาโดยการอัปเดตแอตทริบิวต์ created_at แล้วปล่อยให้ลูปทำงานต่อ เพื่อให้ AI สร้างเนื้อหาต่อจากจุดที่ถูกตัดออก

โปรดจำไว้ว่าจุดสิ้นสุดการทำงานของ AI นั้นไม่มีสถานะ มันจะ “รู้” เฉพาะสิ่งที่คุณบอกมันผ่านบันทึกการสนทนาเท่านั้น ในกรณีนี้ วิธีที่เราสื่อสารกับ AI ว่ามันถูกตัดออกคือการเพิ่มข้อความที่ “มองไม่เห็น” (สำหรับผู้ใช้ปลายทาง) ลงในบันทึกการสนทนา อย่างไรก็ตาม โปรดจำไว้ว่านี่เป็นตัวอย่างที่ถูกทำให้ง่ายโดยเจตนา การใช้งานจริงจะต้องมีการจัดการบันทึกการสนทนาเพิ่มเติม

เพื่อให้แน่ใจว่าเราไม่สูญเสียโทเค็นไปโดยเปล่าประโยชน์ และ/หรือทำให้ AI สับสนกับข้อความตอบกลับที่ซ้ำซ้อนในบันทึกการสนทนา

การใช้งานจริงของระบบการทำงานต่อเนื่องอัตโนมัติควรมีสิ่งที่เรียกว่าตรรกะการตัดวงจร เพื่อป้องกันการวนลูปที่ควบคุมไม่ได้ เหตุผลก็คือ เมื่อพิจารณาจากคำสั่งของผู้ใช้บางประเภทและการตั้งค่า max_tokens ที่ต่ำ AI อาจจะวนลูปสร้างผลลัพธ์ที่แสดงต่อผู้ใช้ไม่สิ้นสุด

> พึงระลึกไว้ว่าทุกรอบของลูปต้องใช้การร้องขอแยกกัน และแต่ละการร้องขอจะใช้บันทึกการสนทนาทั้งหมดของคุณอีกครั้ง คุณควรพิจารณาข้อดีข้อเสียระหว่างประสบการณ์ของผู้ใช้และการใช้งาน API เมื่อตัดสินใจว่าจะใช้การทำงานต่อเนื่องอัตโนมัติในแอปพลิเคชันของคุณหรือไม่ การทำงานต่อเนื่องอัตโนมัติโดยเฉพาะอาจมีค่าใช้จ่ายสูงอันตราย โดยเฉพาะเมื่อใช้โมเดลเชิงพาณิชย์ระดับพรีเมียม

บทสรุป

การประมวลผลแบบสตรีม เป็นแง่มุมที่สำคัญในการสร้างแอปพลิเคชันที่ขับเคลื่อนด้วย AI ซึ่งผสมผสานการใช้เครื่องมือกับการตอบสนองแบบสดจาก AI เมื่อจัดการข้อมูลสตรีมจาก API ของแพลตฟอร์ม AI อย่างมีประสิทธิภาพ คุณสามารถมอบประสบการณ์ผู้ใช้ที่ราบรื่นและโต้ตอบได้ จัดการการตอบสนองขนาดใหญ่ ปรับการใช้ทรัพยากรให้เหมาะสม และจัดการข้อผิดพลาดอย่างสง่างาม

คลาส Conversation::ReplyStream ที่จัดเตรียมไว้แสดงให้เห็นว่าการประมวลผลแบบสตรีมสามารถถูกนำไปใช้ในแอปพลิเคชัน Ruby ได้อย่างไรโดยใช้การจับคู่รูปแบบและสถาปัตยกรรมที่ขับเคลื่อนด้วยเหตุการณ์ ด้วยการทำความเข้าใจและใช้ประโยชน์จากเทคนิคการประมวลผลแบบสตรีม คุณสามารถปลดล็อกศักยภาพสูงสุดของการผสานรวม AI ในแอปพลิเคชันของคุณและส่งมอบประสบการณ์ผู้ใช้ที่ทรงพลังและน่าดึงดูด

ข้อมูลที่เยียวยาตัวเอง

ข้อมูลที่เยียวยาตัวเอง เป็น แนวทาง ที่ ทรง พลัง ใน การ รักษา ความ ถูก ต้อง สมบูรณ์ ของ ข้อมูล ความ สอดคล้อง และคุณภาพในแอปพลิเคชันด้วยการใช้ความสามารถของโมเดลภาษาขนาดใหญ่ (LLMs) รูป แบบ ใน หมวด หมู่ นี้ มุ่ง เน้น แนวคิดการใช้ AI เพื่อตรวจจับ วินิจฉัย และแก้ไขความผิดปกติของข้อมูล ความไม่สอดคล้อง หรือข้อผิดพลาดโดยอัตโนมัติ ซึ่งช่วยลดภาระของนักพัฒนาและรักษาระดับความน่าเชื่อถือของข้อมูลให้สูง

แก่นสำคัญของรูปแบบข้อมูลที่เยียวยาตัวเองคือการตระหนักว่าข้อมูลเป็นหัวใจสำคัญของทุกแอปพลิเคชัน และการรักษาความถูกต้องและความสมบูรณ์ของข้อมูลมีความสำคัญอย่างยิ่งต่อการทำงานที่เหมาะสมและประสบการณ์ของผู้ใช้แอปพลิเคชัน อย่างไรก็ตาม การจัดการและรักษาคุณภาพข้อมูลอาจเป็นงานที่ซับซ้อนและใช้เวลามาก โดยเฉพาะเมื่อแอปพลิเคชันมีขนาดและความซับซ้อนเพิ่มขึ้น นี่คือจุดที่พลังของ AI เข้ามามีบทบาท

ในรูปแบบข้อมูลที่เยียวยาตัวเอง ระบบงาน AI ถูกนำมาใช้เพื่อตรวจสอบและวิเคราะห์ข้อมูลของแอปพลิเคชันของคุณอย่างต่อเนื่อง โมเดลเหล่านี้มีความสามารถในการเข้าใจและตีความรูปแบบ ความสัมพันธ์ และความผิดปกติภายในข้อมูล ด้วยการใช้ประโยชน์จากความสามารถในการประมวลผลและความเข้าใจภาษาธรรมชาติ พวกมันสามารถระบุปัญหาที่อาจเกิดขึ้นหรือความไม่สอดคล้องในข้อมูลและดำเนินการที่เหมาะสมเพื่อแก้ไข

กระบวนการของข้อมูลที่เยียวยาตัวเองมักประกอบด้วยขั้นตอนสำคัญหลายประการ:

1. **การตรวจสอบข้อมูล:** ระบบงาน AI ตรวจสอบสตรีมข้อมูล ฐานข้อมูล หรือระบบจัดเก็บข้อมูลของแอปพลิเคชันอย่างต่อเนื่อง มองหาสัญญาณของความผิดปกติ ความไม่สอดคล้อง หรือข้อผิดพลาดใดๆ หรือคุณสามารถเปิดใช้งานคอมโพเนนต์ AI เพื่อตอบสนองต่อข้อยกเว้นได้
2. **การตรวจจับความผิดปกติ:** เมื่อตรวจพบปัญหา ระบบงาน AI จะวิเคราะห์ข้อมูลอย่างละเอียดเพื่อระบุลักษณะและขอบเขตที่เฉพาะเจาะจงของปัญหา ซึ่งอาจรวมถึงการตรวจจับค่าที่หายไป รูปแบบที่ไม่สอดคล้องกัน หรือข้อมูลที่ละเมิดกฎหรือข้อจำกัดที่กำหนดไว้ล่วงหน้า
3. **การวินิจฉัยและการแก้ไข:** เมื่อระบุปัญหาได้แล้ว ระบบงาน AI จะใช้ความรู้และความเข้าใจในโดเมนข้อมูลเพื่อกำหนดแนวทางการดำเนินการที่เหมาะสม ซึ่งอาจรวมถึงการแก้ไขข้อมูลโดยอัตโนมัติ การเติมค่าที่หายไป หรือการทำเครื่องหมายปัญหาเพื่อให้มนุษย์เข้ามาแทรกแซงหากจำเป็น
4. **การเรียนรู้อย่างต่อเนื่อง (ทางเลือก ขึ้นอยู่กับกรณีการใช้งาน):** เมื่อระบบงาน AI พบและแก้ไขปัญหาข้อมูลต่างๆ มันสามารถสร้างผลลัพธ์ที่อธิบายว่าเกิดอะไรขึ้นและมันตอบสนองอย่างไร ข้อมูลเมตาดาตานี้สามารถป้อนเข้าสู่กระบวนการเรียนรู้ที่ช่วยให้คุณ (และอาจรวมถึงโมเดลพื้นฐาน ผ่านการ fine-tuning) สามารถทำงานได้อย่างมีประสิทธิภาพและประสิทธิผลมากขึ้นในการระบุและแก้ไขความผิดปกติของข้อมูลเมื่อเวลาผ่านไป

ด้วยการตรวจจับและแก้ไขปัญหาข้อมูลโดยอัตโนมัติ คุณสามารถมั่นใจได้ว่าแอปพลิเคชันของคุณทำงานบนข้อมูลที่มีคุณภาพสูงและน่าเชื่อถือ ซึ่งช่วยลดความเสี่ยงของข้อผิดพลาด ความไม่สอดคล้อง หรือข้อบกพร่องที่เกี่ยวข้องกับข้อมูลที่อาจส่งผลกระทบต่อฟังก์ชันการทำงานหรือประสบการณ์ของผู้ใช้แอปพลิเคชัน

เมื่อคุณมีระบบงาน AI จัดการงานตรวจสอบและแก้ไขข้อมูล คุณสามารถมุ่งเน้นความพยายามไปที่ด้านสำคัญอื่นๆ ของแอปพลิเคชัน ซึ่งช่วยประหยัดเวลาและทรัพยากรที่มิฉะนั้นจะต้องใช้ไปกับการทำความสะอาดและบำรุงรักษาข้อมูลด้วยตนเอง ที่จริงแล้ว เมื่อแอปพลิเคชันของคุณมีขนาดและความซับซ้อนเพิ่มขึ้น การจัดการคุณภาพข้อมูลด้วยตนเองจะยิ่งท้าทายมากขึ้น รูปแบบ “ข้อมูลที่เยียวยาตัวเอง” ปรับขนาดได้อย่างมีประสิทธิภาพโดยใช้พลังของ AI ในการจัดการข้อมูลจำนวนมากและตรวจจับปัญหาแบบเรียลไทม์

เนื่องจากธรรมชาติของโมเดล AI สามารถปรับตัวเข้ากับรูปแบบข้อมูล สคีมา หรือข้อกำหนดที่เปลี่ยนแปลงไปตามเวลาโดยแทบไม่ต้องมีการกำกับดูแลหรือไม่ต้องเลย ตราบใดที่คำสั่งของพวกมันให้คำแนะนำที่เพียงพอโดยเฉพาะเกี่ยวกับผลลัพธ์ที่ต้องการ แอปพลิเคชันของคุณอาจสามารถพัฒนาและจัดการกับสถานการณ์ข้อมูลใหม่ๆ ได้โดยไม่จำเป็นต้องมีการแทรกแซงด้วยตนเองหรือการเปลี่ยนแปลงโค้ดมากมาย

รูปแบบข้อมูลที่เยียวยาตัวเองสอดคล้องกับหมวดหมู่รูปแบบอื่นๆ ที่เราได้พูดถึง เช่น “ระบบงานหลากหลาย” ความสามารถในการเยียวยาข้อมูลตัวเองสามารถมองได้ว่าเป็นระบบงานเฉพาะทางที่มุ่งเน้นการรับประกันคุณภาพและความถูกต้องสมบูรณ์ของ

ข้อมูลโดยเฉพาะ ระบบงานประเภทนี้ทำงานร่วมกับระบบงาน AI อื่นๆ แต่ละตัวมีส่วนช่วยในด้านต่างๆ ของฟังก์ชันการทำงานของแอปพลิเคชัน

การนำรูปแบบข้อมูลที่เยียวยาตัวเองไปใช้ในทางปฏิบัติต้องการการออกแบบที่รอบคอบและการผสานโมเดล AI เข้ากับสถาปัตยกรรมของแอปพลิเคชัน เนื่องจากความเสี่ยงของการสูญเสียข้อมูลและการเสียหายของข้อมูล คุณควรกำหนดแนวทางที่ชัดเจนว่าจะใช้เทคนิคนี้อย่างไร คุณควรพิจารณาปัจจัยต่างๆ เช่น ประสิทธิภาพ ความสามารถในการปรับขนาด และความปลอดภัยของข้อมูลด้วย

กรณีศึกษาในทางปฏิบัติ: การแก้ไข JSON ที่เสียหาย

หนึ่งในวิธีที่ปฏิบัติได้จริงและสะดวกที่สุดในการใช้ประโยชน์จากข้อมูลที่เยียวยาตัวเองก็คือการอธิบายที่ง่ายมาก: การแก้ไข JSON ที่เสียหาย

เทคนิคนี้สามารถนำไปใช้กับความท้าทายทั่วไปในการจัดการกับข้อมูลที่ไม่สมบูรณ์หรือไม่สอดคล้องกันที่สร้างโดย LLMs เช่น JSON ที่เสียหาย และให้แนวทางในการตรวจจับและแก้ไขปัญหาเหล่านี้โดยอัตโนมัติ

ที่ Olympia ผมมักจะพบกับสถานการณ์ที่ LLM สร้างข้อมูล JSON ที่ไม่สมบูรณ์แบบอยู่เสมอ สิ่งนี้อาจเกิดขึ้นได้จากหลายสาเหตุ เช่น LLM เพิ่มข้อความอธิบายก่อนหรือหลังโค้ด JSON หรือทำให้เกิดข้อผิดพลาดทางไวยากรณ์ เช่น การลืมใส่เครื่องหมายจุลภาคหรือการไม่ escape เครื่องหมายคำพูดคู่ ปัญหาเหล่านี้สามารถนำไปสู่ข้อผิดพลาดในการแยกวิเคราะห์และทำให้เกิดการหยุดชะงักในการทำงานของแอปพลิเคชัน

เพื่อแก้ไขปัญหานี้ ผมได้พัฒนาวิธีแก้ไขที่ใช้งานได้จริงในรูปแบบของคลาส JsonFixer คลาสนี้ใช้แพทเทิร์น “ข้อมูลเยียวยาตัวเอง” โดยรับข้อมูล JSON ที่มีปัญหาเป็นอินพุต และใช้ LLM ในการแก้ไขข้อมูลนั้น พร้อมทั้งรักษาข้อมูลและเจตนารมณ์เดิมให้ได้มากที่สุดเท่าที่จะเป็นไปได้

```ruby
class JsonFixer
  include Raix::ChatCompletion

  def call(bad_json, error_message)
    raise "No data provided" if bad_json.blank? || error_message.blank?

    transcript << {
        system: "Consider user-provided JSON that generated a parse
                exception. Do your best to fix it while preserving the
                original content and intent as much as possible." }
    transcript << { user: bad_json }
    transcript << { assistant: "What is the error message?"}
    transcript << { user: error_message }
    transcript << { assistant: "Here is the corrected JSON\n```json\n" }

    self.stop = ["```"]

    chat_completion(json: true)
  end

  def model
    "mistralai/mixtral-8x7b-instruct:nitro"
  end
end
```

สังเกตว่า JsonFixer ใช้ Ventriloquist ในการควบคุมการตอบสนองของ AI

กระบวนการซ่อมแซมตัวเองของข้อมูล JSON มีขั้นตอนดังนี้:

1. **การสร้าง JSON:** LLM ถูกใช้ในการสร้างข้อมูล JSON ตามพรอมต์หรือความต้องการที่กำหนด อย่างไรก็ตาม ด้วยธรรมชาติของ LLM ข้อมูล JSON ที่ถูกสร้างขึ้นอาจไม่สมบูรณ์แบบเสมอไป ตัวแยกวิเคราะห์ JSON จะแสดงข้อผิดพลาด ParserError เมื่อคุณป้อนข้อมูล JSON ที่ไม่ถูกต้อง

```
begin
  JSON.parse(llm_generated_json)
rescue JSON::ParserError => e
  JsonFixer.new.call(llm_generated_json, e.message)
end
```

สังเกตว่าข้อความแสดงข้อผิดพลาดจะถูกส่งไปยังการเรียกใช้ JSONFixer ด้วย เพื่อที่จะไม่ต้องสันนิษฐานทั้งหมดว่าอะไรผิดพลาดกับข้อมูล โดยเฉพาะอย่างยิ่งเมื่อตัวแยกวิเคราะห์มักจะบอกคุณได้อย่างชัดเจนว่าอะไรผิดพลาด

2. **การแก้ไขโดย LLM:** คลาส JSONFixer จะส่ง JSON ที่มีข้อผิดพลาดกลับไปยัง LLM พร้อมกับคำสั่งหรือคำแนะนำเฉพาะเพื่อแก้ไข JSON โดยพยายามรักษาข้อมูลและเจตนาดั้งเดิมให้มากที่สุดเท่าที่จะเป็นไปได้ LLM ที่ได้รับการฝึกฝนด้วยข้อมูลจำนวนมากและมีความเข้าใจในไวยากรณ์ของ JSON จะพยายามแก้ไขข้อผิดพลาดและสร้างสตริง JSON ที่ถูกต้อง มีการใช้การกำหนดขอบเขตการตอบสนองเพื่อจำกัดผลลัพธ์ของ LLM และเราเลือกใช้ Mixtral 8x7B เป็นโมเดล AI เนื่องจากมีความสามารถโดดเด่นสำหรับงานประเภทนี้
3. **การตรวจสอบความถูกต้องและการผสานรวม:** สตริง JSON ที่ได้รับการแก้ไขแล้วที่ส่งคืนจาก LLM จะถูกแยกวิเคราะห์โดยคลาส JSONFixer เอง เนื่องจากมีการเรียกใช้ chat_completion(json: true) หากการแก้ไข JSON ผ่านการตรวจสอบความถูกต้อง ก็จะถูกผสานรวมกลับเข้าไปในขั้นตอนการทำงานของแอปพลิเคชัน ทำให้แอปพลิเคชันสามารถประมวลผลข้อมูลต่อไปได้อย่างราบรื่น JSON ที่มีปัญหาได้รับการ “เยียวยา” แล้ว

แม้ว่าผมจะเขียนและเขียนการใช้งาน JSONFixer ของตัวเองใหม่หลายครั้ง แต่ผมสงสัยว่าเวลาทั้งหมดที่ลงทุนไปในเวอร์ชันเหล่านั้นคงไม่เกินหนึ่งหรือสองชั่วโมง

สังเกตว่าการรักษาเจตนาเดิมเป็นองค์ประกอบสำคัญของรูปแบบข้อมูลที่เยียวยาตัวเองทุกรูปแบบ กระบวนการแก้ไขโดย LLM มีเป้าหมายที่จะรักษาข้อมูลและเจตนาดั้งเดิมของ JSON ที่สร้างขึ้นให้มากที่สุดเท่าที่จะเป็นไปได้ ซึ่งช่วยให้มั่นใจว่า JSON ที่ได้รับการแก้ไขยังคงรักษาความหมายทางความหมายและสามารถใช้งานได้อย่างมีประสิทธิภาพภายในบริบทของแอปพลิเคชัน

การนำแนวคิด “ข้อมูลที่เยียวยาตัวเอง” ไปใช้งานจริงใน Olympia นี้แสดงให้เห็นอย่างชัดเจนว่า AI โดยเฉพาะอย่างยิ่ง LLM สามารถนำมาใช้แก้ไขความท้าทายด้านข้อมูลในโลกแห่งความเป็นจริงได้อย่างไร มันแสดงให้เห็นถึงพลังของการผสมผสานเทคนิคการเขียนโปรแกรมแบบดั้งเดิมกับความสามารถของ AI เพื่อสร้างแอปพลิเคชันที่แข็งแกร่งและมีประสิทธิภาพ

กฎของพอสเทลและรูปแบบ “ข้อมูลที่เยียวยาตัวเอง”

“ข้อมูลที่เยียวยาตัวเอง” ดังที่เห็นได้จากคลาส JSONFixer สอดคล้องกับหลักการที่รู้จักกันในชื่อกฎของพอสเทล หรือที่เรียกว่าหลักการความทนทาน กฎของพอสเทลระบุว่า:

“จงอนุรักษ์นิยมในสิ่งที่คุณทำ จงเสรีนิยมในสิ่งที่คุณยอมรับจากผู้อื่น”

หลักการนี้ ซึ่งเดิมถูกกล่าวโดย Jon Postel ผู้บุกเบิกอินเทอร์เน็ตยุคแรก เน้นย้ำถึงความสำคัญของการสร้างระบบที่ยอมรับอินพุตที่หลากหลายหรือแม้แต่มีข้อผิดพลาดเล็กน้อย ในขณะที่ยังคงรักษาการปฏิบัติตามโปรโตคอลที่กำหนดไว้อย่างเคร่งครัดเมื่อส่งเอาต์พุต

ในบริบทของ “ข้อมูลที่เยียวยาตัวเอง” คลาส JSONFixer แสดงให้เห็นถึงกฎของพอสเทลโดยการมีความเสรีในการยอมรับข้อมูล JSON ที่มีข้อผิดพลาดหรือไม่สมบูรณ์ที่สร้างโดย LLM มันไม่ได้ปฏิเสธหรือล้มเหลวทันทีเมื่อพบ JSON ที่ไม่เป็นไปตามรูปแบบที่คาดหวังอย่างเคร่งครัด แต่กลับใช้วิธีการที่ยืดหยุ่นและพยายามแก้ไข JSON โดยใช้พลังของ LLM

ด้วยการมีความเสรีในการยอมรับ JSON ที่ไม่สมบูรณ์ คลาส JSONFixer แสดงให้เห็นถึงความแข็งแกร่งและความยืดหยุ่น มันยอมรับว่าข้อมูลในโลกแห่งความเป็นจริงมักมาในรูปแบบต่างๆ และอาจไม่เป็นไปตามข้อกำหนดที่เคร่งครัดเสมอไป ด้วยการจัดการและแก้ไขความเบี่ยงเบนเหล่านี้อย่างสง่างาม คลาสนี้ช่วยให้มั่นใจว่าแอปพลิเคชันสามารถทำงานต่อไปได้อย่างราบรื่น แม้จะมีข้อมูลที่ไม่สมบูรณ์

ในทางกลับกัน คลาส JSONFixer ยังยึดมั่นในแง่มุมอนุรักษ์นิยมของกฎพอสเทลเมื่อพูดถึงเอาต์พุต หลังจากแก้ไข JSON โดยใช้ LLM แล้ว คลาสจะตรวจสอบความถูกต้องของ JSON ที่แก้ไขแล้วเพื่อให้แน่ใจว่าเป็นไปตามรูปแบบที่คาดหวังอย่างเคร่งครัด มันรักษาความสมบูรณ์และความถูกต้องของข้อมูลก่อนที่จะส่งต่อไปยังส่วนอื่นๆ ของแอปพลิเคชัน วิธีการอนุรักษ์นิยมนี้รับประกันว่าเอาต์พุตของคลาส JSONFixer มีความน่าเชื่อถือและสม่ำเสมอ ส่งเสริมความสามารถในการทำงานร่วมกันและป้องกันการแพร่กระจายของข้อผิดพลาด

เรื่องน่าสนใจเกี่ยวกับ Jon Postel:

- Jon Postel (1943-1998) เป็น นัก วิทยาศาสตร์ คอมพิวเตอร์ ชาว อเมริกัน ที่ มี บทบาท สำคัญ ใน การ พัฒนา อินเทอร์เน็ต เขาได้รับการขนานนามว่าเป็น “เทพเจ้าแห่งอินเทอร์เน็ต” จากการมีส่วนร่วมที่สำคัญในโปรโตคอลและมาตรฐานพื้นฐาน
- Postel เป็นบรรณาธิการของชุดเอกสาร Request for Comments (RFC) ซึ่งเป็นชุดบันทึกทางเทคนิคและองค์กรเกี่ยวกับอินเทอร์เน็ต เขาเป็นผู้เขียนหรือผู้ร่วมเขียน RFC มากกว่า 200 ฉบับ รวมถึงโปรโตคอลพื้นฐานต่างๆ เช่น TCP, IP และ SMTP

- นอกเหนือจากการมีส่วนร่วมทางเทคนิคแล้ว Postel ยังเป็นที่รู้จักในแนวทางที่ถ่อมตนและร่วมมือกัน เขาเชื่อในความสำคัญของการบรรลุฉันทามติและการทำงานร่วมกันเพื่อสร้างเครือข่ายที่แข็งแกร่งและสามารถทำงานร่วมกันได้
- Postel ดำรง ตำแหน่ง ผู้ อำนวย การ แผนก เครือ ข่าย คอมพิวเตอร์ ที่ สถาบัน วิทยาศาสตร์ สารสนเทศ (ISI) ของ มหาวิทยาลัยแคลิฟอร์เนียใต้ (USC) ตั้งแต่ปี 1977 จนกระทั่งเสียชีวิตอย่างกะทันหันในปี 1998
- เพื่อเป็นการยกย่องการมีส่วนร่วมอันยิ่งใหญ่ของเขา Postel ได้รับรางวัล Turing Award อันทรงเกียรติหลังจากเสียชีวิตในปี 1998 ซึ่งมักถูกเรียกว่าเป็น “รางวัลโนเบลด้านการคำนวณ”

คลาส JSONFixer ส่งเสริมความทนทาน ความยืดหยุ่น และความสามารถในการทำงานร่วมกัน ซึ่งเป็นค่านิยมหลักที่ Postel ยึดถือตลอดอาชีพการงานของเขา ด้วยการสร้างระบบที่ยอมรับข้อบกพร่องได้ในขณะที่ยังคงรักษาการปฏิบัติตามโปรโตคอลอย่างเคร่งครัด เราสามารถสร้างแอปพลิเคชันที่มีความยืดหยุ่นและปรับตัวได้ดีขึ้นเมื่อเผชิญกับความท้าทายในโลกแห่งความเป็นจริง

ข้อพิจารณาและข้อห้าม

การประยุกต์ใช้แนวทางข้อมูลที่ซ่อมแซมตัวเองนั้นขึ้นอยู่กับประเภทของข้อมูลที่แอปพลิเคชันของคุณจัดการอย่างสมบูรณ์ มีเหตุผลว่าทำไมคุณอาจไม่ต้องการที่จะใช้มังกี้แพตช์กับ JSON.parse เพื่อแก้ไข_ข้อผิดพลาดการแยกวิเคราะห์ JSON ทั้งหมด_ในแอปพลิเคชันของคุณโดยอัตโนมัติ: เพราะไม่ใช่ข้อผิดพลาดทั้งหมดที่สามารถหรือควรจะแก้ไขโดยอัตโนมัติ

การซ่อมแซมตัวเองมีความซับซ้อนเป็นพิเศษเมื่อเกี่ยวข้องกับข้อกำหนดด้านการกำกับดูแลหรือการปฏิบัติตามกฎระเบียบที่เกี่ยวข้องกับการจัดการและประมวลผลข้อมูล อุตสาหกรรมบางประเภท เช่น การดูแลสุขภาพและการเงิน มีกฎระเบียบที่เข้มงวดเกี่ยวกับความถูกต้องสมบูรณ์ของข้อมูลและความสามารถในการตรวจสอบ การทำการแก้ไขข้อมูลแบบ “กล่องดำ” โดยไม่มีการกำกับดูแลหรือการบันทึกที่เหมาะสมอาจละเมิดกฎระเบียบเหล่านี้ สิ่งสำคัญคือต้องให้แน่ใจว่าเทคนิคการซ่อมแซมข้อมูลตัวเองที่คุณคิดค้นขึ้นสอดคล้องกับกรอบกฎหมายและข้อบังคับที่เกี่ยวข้อง

การใช้เทคนิคข้อมูลที่ซ่อมแซมตัวเอง โดยเฉพาะอย่างยิ่งที่เกี่ยวข้องกับโมเดล AI อาจส่งผลกระทบอย่างมากต่อประสิทธิภาพของแอปพลิเคชันและการใช้ทรัพยากร การประมวลผลข้อมูลจำนวนมากผ่านโมเดล AI เพื่อตรวจจับและแก้ไขข้อผิดพลาดอาจใช้ทรัพยากรการคำนวณอย่างมาก สิ่งสำคัญคือต้องประเมินการแลกเปลี่ยนระหว่างประโยชน์ของข้อมูลที่ซ่อมแซมตัวเองและต้นทุนด้านประสิทธิภาพและทรัพยากรที่เกี่ยวข้อง

อย่างไรก็ตาม มาดูปัจจัยที่เกี่ยวข้องในการตัดสินใจว่าเมื่อไหร่และที่ไหนที่ควรใช้แนวทางอันทรงพลังนี้

ความสำคัญของข้อมูล

เมื่อพิจารณาการใช้เทคนิคข้อมูลที่ซ่อมแซมตัวเอง สิ่งสำคัญคือต้องประเมินความสำคัญของข้อมูลที่กำลังประมวลผล ระดับความสำคัญหมายถึงความสำคัญและความอ่อนไหวของข้อมูลในบริบทของแอปพลิเคชันและโดเมนธุรกิจของคุณ

ในบางกรณี การแก้ไขข้อผิดพลาดของข้อมูลโดยอัตโนมัติอาจไม่เหมาะสม โดยเฉพาะอย่างยิ่งถ้าข้อมูลมีความอ่อนไหวสูงหรือมีผลกระทบทางกฎหมาย ตัวอย่างเช่น พิจารณาสถานการณ์ต่อไปนี้:

1. **ธุรกรรมทางการเงิน:** ในแอปพลิเคชันทางการเงิน เช่น ระบบธนาคารหรือแพลตฟอร์มการซื้อขาย ความถูกต้องของข้อมูลมีความสำคัญสูงสุด แม้แต่ข้อผิดพลาดเล็กน้อยในข้อมูลทางการเงินก็สามารถส่งผลกระทบที่สำคัญ เช่น ยอดบัญชีไม่ถูกต้อง การโอนเงินผิดทาง หรือการตัดสินใจซื้อขายที่ผิดพลาด ในกรณีเหล่านี้ การแก้ไขอัตโนมัติโดยไม่มีการตรวจสอบและการตรวจสอบอย่างละเอียดอาจนำมาซึ่งความเสี่ยงที่ยอมรับไม่ได้
2. **บันทึกทางการแพทย์:** แอปพลิเคชันด้านการดูแลสุขภาพจัดการกับข้อมูลผู้ป่วยที่มีความอ่อนไหวและเป็นความลับสูง ความไม่ถูกต้องในบันทึกทางการแพทย์สามารถส่งผลกระทบร้ายแรงต่อความปลอดภัยของผู้ป่วยและการตัดสินใจในการรักษา การแก้ไขข้อมูลทางการแพทย์โดยอัตโนมัติโดยไม่มีการกำกับดูแลและการตรวจสอบที่เหมาะสมจากผู้เชี่ยวชาญด้านการดูแลสุขภาพอาจละเมิดข้อกำหนดด้านกฎระเบียบและเป็นอันตรายต่อความเป็นอยู่ที่ดีของผู้ป่วย
3. **เอกสารทางกฎหมาย:** แอปพลิเคชันที่จัดการเอกสารทางกฎหมาย เช่น สัญญา ข้อตกลง หรือเอกสารศาล ต้องการความถูกต้องและความสมบูรณ์อย่างเคร่งครัด แม้แต่ข้อผิดพลาดเล็กน้อยในข้อมูลทางกฎหมายก็สามารถส่งผลกระทบทางกฎหมายที่สำคัญ การแก้ไขอัตโนมัติในโดเมนนี้อาจไม่เหมาะสม เนื่องจากข้อมูลมักต้องการการตรวจสอบและการยืนยันด้วยตนเองจากผู้เชี่ยวชาญด้านกฎหมายเพื่อให้แน่ใจว่ามีความถูกต้องและสามารถบังคับใช้ได้

ในสถานการณ์ข้อมูลที่สำคัญเหล่านี้ ความเสี่ยงที่เกี่ยวข้องกับการแก้ไขอัตโนมัติมักจะมีน้ำหนักมากกว่าประโยชน์ที่อาจได้รับ ผลที่ตามมาของการแนะนำข้อผิดพลาดหรือการแก้ไขข้อมูลอย่างไม่ถูกต้องอาจร้ายแรง นำไปสู่การสูญเสียทางการเงิน ความรับผิดทางกฎหมาย หรือแม้แต่อันตรายต่อบุคคล

เมื่อจัดการกับข้อมูลที่มีความสำคัญสูง สิ่งสำคัญคือต้องให้ความสำคัญกับกระบวนการตรวจสอบและยืนยันด้วยตนเอง การกำกับดูแลและความเชี่ยวชาญของมนุษย์มีความสำคัญในการรับรองความถูกต้องและความสมบูรณ์ของข้อมูล เทคนิคการซ่อมแซมตัวเองแบบอัตโนมัติยังคงสามารถใช้เพื่อระบุข้อผิดพลาดหรือความไม่สอดคล้องที่อาจเกิดขึ้น แต่การตัดสินใจขั้นสุดท้ายเกี่ยวกับการแก้ไขควรเกี่ยวข้องกับการตัดสินใจและการอนุมัติของมนุษย์

อย่างไรก็ตาม สิ่งสำคัญคือต้องทราบว่าไม่ใช่ข้อมูลทั้งหมดในแอปพลิเคชันจะมีระดับความสำคัญเท่ากัน ภายในแอปพลิเคชันเดียวกัน อาจมีข้อมูลย่อยที่มีความอ่อนไหวน้อยกว่าหรือมีผลกระทบต่ำกว่าหากเกิดข้อผิดพลาด ในกรณีเช่นนี้ เทคนิคข้อมูลที่

ซ่อมแซมตัวเองสามารถนำไปใช้กับข้อมูลย่อยเฉพาะเหล่านั้นได้อย่างเลือกสรร ในขณะที่ข้อมูลที่สำคัญยังคงต้องผ่านการตรวจสอบด้วยตนเอง

สิ่งสำคัญคือการประเมินความสำคัญของข้อมูลแต่ละประเภทในแอปพลิเคชันของคุณอย่างระมัดระวังและกำหนดแนวทางและกระบวนการที่ชัดเจนสำหรับการจัดการการแก้ไขตามความเสี่ยงและผลกระทบที่เกี่ยวข้อง โดยการแยกความแตกต่างระหว่างข้อมูลที่สำคัญ (เช่น สมุดบัญชี บันทึกทางการแพทย์) และข้อมูลที่ไม่สำคัญ (เช่น ที่อยู่ทางไปรษณีย์ คำเตือนเกี่ยวกับทรัพยากร) คุณสามารถสร้างความสมดุลระหว่างการใช้ประโยชน์จากเทคนิคข้อมูลที่ซ่อมแซมตัวเองในที่ที่เหมาะสมและการรักษาการควบคุมและการกำกับดูแลอย่างเข้มงวดในที่ที่จำเป็น

ในท้ายที่สุด การตัดสินใจที่จะใช้เทคนิคข้อมูลที่ซ่อมแซมตัวเองกับข้อมูลที่สำคัญควรทำโดยปรึกษากับผู้เชี่ยวชาญในโดเมน ที่ปรึกษากฎหมาย และผู้มีส่วนได้ส่วนเสียอื่นๆ ที่เกี่ยวข้อง สิ่งสำคัญคือต้องพิจารณาข้อกำหนด กฎระเบียบ และความเสี่ยงเฉพาะที่เกี่ยวข้องกับข้อมูลของแอปพลิเคชันของคุณและปรับกลยุทธ์การแก้ไขข้อมูลให้สอดคล้องกัน

ความรุนแรงของข้อผิดพลาด

เมื่อใช้เทคนิคข้อมูลที่ซ่อมแซมตัวเอง สิ่งสำคัญคือต้องประเมินความรุนแรงและผลกระทบของข้อผิดพลาดของข้อมูล ไม่ใช่ข้อผิดพลาดทั้งหมดจะเท่าเทียมกัน และวิธีการที่เหมาะสมอาจแตกต่างกันไปขึ้นอยู่กับความรุนแรงของปัญหา

ความไม่สอดคล้องเล็กน้อยหรือปัญหาการจัดรูปแบบอาจเหมาะสมสำหรับการแก้ไขอัตโนมัติ ตัวอย่างเช่น worker ที่ซ่อมแซมข้อมูลตัวเองที่มีหน้าที่แก้ไข JSON ที่เสียหายสามารถจัดการกับเครื่องหมายจุลภาคที่หายไปหรือเครื่องหมายคำพูดคู่ที่ไม่ได้ escape โดยไม่เปลี่ยนแปลงความหมายหรือโครงสร้างของข้อมูลอย่างมีนัยสำคัญ ข้อผิดพลาดประเภทนี้มักจะแก้ไขได้ตรงไปตรงมาและมีผลกระทบน้อยต่อความสมบูรณ์ของข้อมูลโดยรวม

อย่างไรก็ตาม ข้อผิดพลาดที่รุนแรงกว่าซึ่งเปลี่ยนแปลงความหมายหรือความถูกต้องสมบูรณ์ของข้อมูลอย่างมีนัยสำคัญ อาจต้องใช้วิธีการที่แตกต่างออกไป ในกรณีเช่นนี้ การแก้ไขอัตโนมัติอาจไม่เพียงพอ และอาจจำเป็นต้องมีการแทรกแซงจากมนุษย์เพื่อให้มั่นใจในความถูกต้องและความน่าเชื่อถือของข้อมูล

นี่คือจุดที่แนวคิดการใช้ AI เพื่อช่วยประเมินความรุนแรงของข้อผิดพลาดเข้ามามีบทบาท โดยการใช้ประโยชน์จากความสามารถของโมเดล AI เราสามารถออกแบบระบบงานจัดการข้อมูลที่ซ่อมแซมตัวเองที่ไม่เพียงแต่แก้ไขข้อผิดพลาด แต่ยังสามารถประเมินความรุนแรงของข้อผิดพลาดเหล่านั้นและตัดสินใจอย่างชาญฉลาดว่าควรจัดการกับมันอย่างไร

ยกตัวอย่างเช่น ลองพิจารณาระบบงานจัดการข้อมูลที่ซ่อมแซมตัวเองที่รับผิดชอบการแก้ไขความไม่สอดคล้องในข้อมูลที่ไหลเข้าสู่ฐานข้อมูลลูกค้า ระบบงานนี้สามารถถูกออกแบบให้วิเคราะห์ข้อมูลและระบุข้อผิดพลาดที่อาจเกิดขึ้น เช่น ข้อมูลที่หายไปหรือข้อมูลที่ขัดแย้งกัน อย่างไรก็ตาม แทนที่จะแก้ไขข้อผิดพลาดทั้งหมดโดยอัตโนมัติ ระบบงานนี้สามารถติดตั้งการเรียกใช้เครื่องมือเพิ่มเติมที่ช่วยให้สามารถทำเครื่องหมายข้อผิดพลาดที่รุนแรงเพื่อให้มนุษย์ตรวจสอบได้

นี่คือตัวอย่างของวิธีการนำไปใช้งาน:

```
class CustomerDataReviewer
  include Raix::ChatCompletion
  include Raix::FunctionDeclarations

  attr_accessor :customer

  function :flag_for_review, reason: { type: "string" } do |params|
    AdminNotifier.review_request(customer, params[:reason])
  end

  def initialize(customer)
    self.customer = customer
  end

  def call(customer_data)
    transcript << {
      system: "You are a customer data reviewer. Your task is to identify
        and correct inconsistencies in customer data.

        < additional instructions here... >

        If you encounter severe errors that require human review, use the
        `flag_for_review` tool to flag the data for manual intervention." }

    transcript << { user: customer.to_json }
    transcript << { assistant: "Reviewed/corrected data:\n```json\n" }

    self.stop = ["```"]

    chat_completion(json: true).then do |result|
      return if result.blank?

      customer.update(result)
    end
  end
end
```

ในตัวอย่างนี้ worker ที่ชื่อ CustomerDataHealer ถูกออกแบบมาเพื่อระบุและแก้ไขความไม่สอดคล้องในข้อมูลลูกค้า อีกครั้งที่เราใช้ การกำหนดขอบเขตการตอบสนอง และ ระบบจัดการการตอบสนอง เพื่อให้ได้ผลลัพธ์ที่มีโครงสร้าง ที่สำคัญคือ คำสั่ง

ระบบของ worker รวมถึงคำแนะนำให้ใช้ฟังก์ชัน flag_for_review หากพบข้อผิดพลาดร้ายแรง

เมื่อ worker ประมวลผลข้อมูลลูกค้า มันจะวิเคราะห์ข้อมูลและพยายามแก้ไขความไม่สอดคล้องต่างๆ หาก worker พิจารณาว่าข้อผิดพลาดนั้นร้ายแรงและต้องการการแทรกแซงจากมนุษย์ มันสามารถใช้เครื่องมือ flag_for_review เพื่อทำเครื่องหมายที่ข้อมูลและให้เหตุผลสำหรับการทำเครื่องหมายนั้น

เมธอด chat_completion ถูกเรียกใช้ด้วย json: true เพื่อแยกวิเคราะห์ข้อมูลลูกค้าที่ได้รับการแก้ไขเป็น JSON ไม่มีการกำหนดให้วนซ้ำหลังจากการเรียกฟังก์ชัน ดังนั้นผลลัพธ์จะว่างเปล่าหาก flag_for_review ถูกเรียกใช้ มิฉะนั้น ข้อมูลลูกค้าจะได้รับการอัปเดตด้วยข้อมูลที่ได้รับการตรวจสอบและอาจมีการแก้ไขแล้ว

ด้วยการรวมการประเมินความรุนแรงของข้อผิดพลาดและตัวเลือกในการทำเครื่องหมายข้อมูลสำหรับการตรวจสอบโดยมนุษย์ worker ข้อมูลที่ซ่อมแซมตัวเองจะกลายเป็นระบบที่ฉลาดและปรับตัวได้มากขึ้น มันสามารถจัดการกับข้อผิดพลาดเล็กน้อยโดยอัตโนมัติในขณะที่ส่งต่อข้อผิดพลาดร้ายแรงไปยังผู้เชี่ยวชาญมนุษย์เพื่อการแทรกแซงด้วยตนเอง

เกณฑ์เฉพาะสำหรับการกำหนดความรุนแรงของข้อผิดพลาดสามารถกำหนดได้ในคำสั่งของ worker โดยอิงจากความรู้ในโดเมนและความต้องการทางธุรกิจ ปัจจัยต่างๆ เช่น ผลกระทบต่อความสมบูรณ์ของข้อมูล โอกาสในการสูญเสียหรือเสียหายของข้อมูล และผลที่ตามมาของข้อมูลที่ไม่ถูกต้องสามารถนำมาพิจารณาเมื่อประเมินความรุนแรง

ด้วยการใช้ประโยชน์จาก AI ในการประเมินความรุนแรงของข้อผิดพลาดและการให้ตัวเลือกสำหรับการแทรกแซงของมนุษย์ เทคนิคข้อมูลที่ซ่อมแซมตัวเองสามารถสร้างความสมดุลระหว่างการทำงานอัตโนมัติและการรักษาความถูกต้องของข้อมูล วิธีการนี้ช่วยให้มั่นใจว่าข้อผิดพลาดเล็กน้อยได้รับการแก้ไขอย่างมีประสิทธิภาพในขณะที่ข้อผิดพลาดร้ายแรงได้รับความสนใจและความเชี่ยวชาญที่จำเป็นจากผู้ตรวจสอบที่เป็นมนุษย์

ความซับซ้อนของโดเมน

เมื่อพิจารณาการประยุกต์ใช้เทคนิคข้อมูลที่ซ่อมแซมตัวเอง สิ่งสำคัญคือต้องประเมินความซับซ้อนของโดเมนข้อมูลและกฎที่ควบคุมโครงสร้างและความสัมพันธ์ของมัน ความซับซ้อนของโดเมนสามารถส่งผลกระทบอย่างมีนัยสำคัญต่อประสิทธิภาพและความเป็นไปได้ของวิธีการแก้ไขข้อมูลอัตโนมัติ

เทคนิคข้อมูลที่ซ่อมแซมตัวเองทำงานได้ดีเมื่อข้อมูลเป็นไปตามรูปแบบและข้อจำกัดที่กำหนดไว้อย่างชัดเจน ในโดเมนที่โครงสร้างข้อมูลค่อนข้างเรียบง่ายและความสัมพันธ์ระหว่างองค์ประกอบข้อมูลไม่ซับซ้อน การแก้ไขอัตโนมัติสามารถนำไปใช้ได้ด้วยความมั่นใจสูง ตัวอย่างเช่น การแก้ไขปัญหาการจัดรูปแบบหรือการบังคับใช้ข้อจำกัดประเภทข้อมูลพื้นฐานมักจะสามารถจัดการได้อย่างมีประสิทธิภาพโดย worker ข้อมูลที่ซ่อมแซมตัวเอง

อย่างไรก็ตาม เมื่อความซับซ้อนของโดเมนข้อมูลเพิ่มขึ้น ความท้าทายที่เกี่ยวข้องกับการแก้ไขข้อมูลอัตโนมัติก็เพิ่มขึ้นด้วย ใน

โดเมนที่มีตรรกะทางธุรกิจที่ซับซ้อน ความสัมพันธ์ที่ซับซ้อนระหว่างเอนทิตีข้อมูล หรือกฎและข้อยกเว้นเฉพาะโดเมน เทคนิคข้อมูลที่ซ่อมแซมตัวเองอาจไม่สามารถจับความละเอียดอ่อนได้เสมอไปและอาจก่อให้เกิดผลที่ไม่ได้ตั้งใจ

มาพิจารณาตัวอย่างของโดเมนที่ซับซ้อน: ระบบการซื้อขายทางการเงิน ในโดเมนนี้ ข้อมูลเกี่ยวข้องกับเครื่องมือทางการเงินต่างๆ ข้อมูลตลาด กฎการซื้อขาย และข้อกำหนดด้านกฎระเบียบ ความสัมพันธ์ระหว่างองค์ประกอบข้อมูลต่างๆ อาจซับซ้อน และกฎที่ควบคุมความถูกต้องและความสอดคล้องของข้อมูลอาจเฉพาะเจาะจงมากสำหรับโดเมนนั้น

ในโดเมนที่ซับซ้อนเช่นนี้ worker ข้อมูลที่ซ่อมแซมตัวเองที่มีหน้าที่แก้ไขความไม่สอดคล้องในข้อมูลการซื้อขายจะต้องมีความเข้าใจอย่างลึกซึ้งเกี่ยวกับกฎและข้อจำกัดเฉพาะโดเมน มันจะต้องพิจารณาปัจจัยต่างๆ เช่น กฎระเบียบของตลาด ขีดจำกัดการซื้อขาย การคำนวณความเสี่ยง และขั้นตอนการชำระราคา การแก้ไขอัตโนมัติในบริบทนี้อาจไม่สามารถจับความซับซ้อนทั้งหมดของโดเมนได้เสมอไปและอาจทำให้เกิดข้อผิดพลาดหรือละเมิดกฎเฉพาะโดเมนโดยไม่ตั้งใจ

เพื่อจัดการกับความท้าทายของความซับซ้อนของโดเมน เทคนิคข้อมูลที่ซ่อมแซมตัวเองสามารถเพิ่มประสิทธิภาพได้โดยการรวมความรู้และกฎเฉพาะโดเมนเข้าไปในโมเดล AI และ worker สามารถทำได้ผ่านเทคนิคต่างๆ เช่น:

1. **การฝึกฝนเฉพาะโดเมน:** โมเดล AI ที่ใช้สำหรับข้อมูลที่ซ่อมแซมตัวเองสามารถถูกกำหนดทิศทางหรือแม้แต่ปรับแต่งบนชุดข้อมูลเฉพาะโดเมนที่จับความละเอียดอ่อนและกฎของโดเมนนั้นๆ ด้วยการให้โมเดลสัมผัสกับข้อมูลและสถานการณ์ตัวอย่าง พวกมันสามารถเรียนรู้รูปแบบ ข้อจำกัด และข้อยกเว้นเฉพาะของโดเมน
2. **ข้อจำกัดตามกฎ:** worker ข้อมูลที่ซ่อมแซมตัวเองสามารถเพิ่มประสิทธิภาพด้วยข้อจำกัดตามกฎที่ชัดเจนซึ่งเข้ารหัสความรู้เฉพาะโดเมน กฎเหล่านี้สามารถกำหนดโดยผู้เชี่ยวชาญในโดเมนและผสานเข้ากับกระบวนการแก้ไขข้อมูล จากนั้นโมเดล AI สามารถใช้กฎเหล่านี้เพื่อแนะนำการตัดสินใจและรับรองการปฏิบัติตามข้อกำหนดเฉพาะโดเมน
3. **การทำงานร่วมกับผู้เชี่ยวชาญในโดเมน:** ในโดเมนที่ซับซ้อน สิ่งสำคัญคือต้องให้ผู้เชี่ยวชาญในโดเมนมีส่วนร่วมในการออกแบบและพัฒนาเทคนิคข้อมูลที่ซ่อมแซมตัวเอง ผู้เชี่ยวชาญในโดเมนสามารถให้ข้อมูลเชิงลึกที่มีค่าเกี่ยวกับความละเอียดอ่อนของข้อมูล กฎทางธุรกิจ และกรณีพิเศษที่อาจเกิดขึ้น ความรู้ของพวกเขาสามารถผสานเข้ากับโมเดล AI และ worker เพื่อปรับปรุงความแม่นยำและความน่าเชื่อถือของการแก้ไขข้อมูลอัตโนมัติโดยใช้รูปแบบ การมีมนุษย์ร่วมในกระบวนการ
4. **วิธีการแบบค่อยเป็นค่อยไปและทำซ้ำ:** เมื่อจัดการกับโดเมนที่ซับซ้อน มักจะเป็นประโยชน์ที่จะใช้วิธีการแบบค่อยเป็นค่อยไปและทำซ้ำสำหรับข้อมูลที่ซ่อมแซมตัวเอง แทนที่จะพยายามทำให้การแก้ไขเป็นอัตโนมัติสำหรับทั้งโดเมนในคราวเดียว ให้มุ่งเน้นไปที่โดเมนย่อยหรือหมวดหมู่ข้อมูลเฉพาะที่มีความเข้าใจกฎและข้อจำกัดเป็นอย่างดี ค่อยๆ ขยายขอบเขตของเทคนิคการซ่อมแซมตัวเองเมื่อความเข้าใจในโดเมนเพิ่มขึ้นและเทคนิคพิสูจน์แล้วว่ามีประสิทธิภาพ

การพิจารณาความซับซ้อนของโดเมนข้อมูลและการผนวกความรู้เฉพาะโดเมนเข้ากับเทคนิคข้อมูลที่ซ่อมแซมตัวเอง จะช่วยให้สามารถสร้างความสมดุลระหว่างการทำงานอัตโนมัติและความแม่นยำ สิ่งสำคัญคือต้องตระหนักว่าข้อมูลที่ซ่อมแซมตัวเองไม่ใช่

วิธีแก้ปัญหาแบบเดียวที่ใช้ได้กับทุกกรณี และควรปรับแนวทางให้เหมาะสมกับความต้องการและความท้าทายเฉพาะของแต่ละโดเมน

ในโดเมนที่ซับซ้อน แนวทางแบบผสมผสานที่รวมเทคนิคข้อมูลที่ซ่อมแซมตัวเองเข้ากับความเชี่ยวชาญและการกำกับดูแลของมนุษย์อาจมีประสิทธิภาพมากที่สุด การแก้ไขอัตโนมัติสามารถจัดการกับกรณีทั่วไปและกรณีที่มีการกำหนดไว้อย่างชัดเจน ในขณะที่สถานการณ์ที่ซับซ้อนหรือข้อยกเว้นต่างๆ สามารถถูกทำเครื่องหมายไว้เพื่อให้มนุษย์ตรวจสอบและเข้าแทรกแซง แนวทางแบบร่วมมือนี้ช่วยให้มั่นใจว่าจะได้รับประโยชน์จากระบบอัตโนมัติ ในขณะที่ยังคงรักษาการควบคุมและความแม่นยำที่จำเป็นในโดเมนข้อมูลที่ซับซ้อน

ความสามารถในการอธิบายและความโปร่งใส

ความสามารถในการอธิบายหมายถึงความสามารถในการทำความเข้าใจและตีความเหตุผลเบื้องหลังการตัดสินใจที่ทำโดยโมเดล AI ในขณะที่ความโปร่งใสเกี่ยวข้องกับการให้มองเห็นกระบวนการแก้ไขข้อมูลอย่างชัดเจน

ในหลายบริบท การแก้ไขข้อมูลจำเป็นต้องสามารถตรวจสอบและอธิบายได้ ผู้มีส่วนได้ส่วนเสีย รวมถึงผู้ใช้ทางธุรกิจ ผู้ตรวจสอบ และหน่วยงานกำกับดูแล อาจต้องการคำอธิบายว่าทำไมจึงมีการแก้ไขข้อมูลบางอย่าง และโมเดล AI มาถึงการตัดสินใจเหล่านั้นได้อย่างไร สิ่งนี้มีความสำคัญอย่างยิ่งในโดเมนที่ความถูกต้องและความสมบูรณ์ของข้อมูลมีผลกระทบสำคัญ เช่น การเงิน การดูแลสุขภาพ และเรื่องทางกฎหมาย

เพื่อตอบสนองความต้องการด้านความสามารถในการอธิบายและความโปร่งใส เทคนิคข้อมูลที่ซ่อมแซมตัวเองควรรวมกลไกที่ให้ข้อมูลเชิงลึกเกี่ยวกับกระบวนการตัดสินใจของโมเดล AI สิ่งนี้สามารถทำได้ผ่านแนวทางต่างๆ:

1. **ลำดับความคิด:** การขอให้โมเดลอธิบายความคิดของตนเอง “ออกเสียง” ก่อนที่จะทำการเปลี่ยนแปลงข้อมูล อาจช่วยให้เข้าใจกระบวนการตัดสินใจได้ง่ายขึ้นและสามารถสร้างคำอธิบายที่มนุษย์อ่านได้สำหรับการแก้ไขที่ทำ การแลกเปลี่ยนคือความซับซ้อนที่เพิ่มขึ้นเล็กน้อยในการแยกคำอธิบายออกจากผลลัพธ์ข้อมูลที่มีโครงสร้าง ซึ่งสามารถแก้ไขได้โดย...
2. **การสร้างคำอธิบาย:** ผู้ปฏิบัติงานด้านข้อมูลที่ซ่อมแซมตัวเองสามารถติดตั้งความสามารถในการสร้างคำอธิบายที่มนุษย์อ่านได้สำหรับการแก้ไขที่พวกเขาทำ สิ่งนี้สามารถทำได้โดยการขอให้โมเดลแสดงกระบวนการตัดสินใจของมันเป็นคำอธิบายที่เข้าใจง่าย *ที่ผสานรวมอยู่ในตัวข้อมูลเอง* ตัวอย่างเช่น ผู้ปฏิบัติงานด้านข้อมูลที่ซ่อมแซมตัวเองสามารถสร้างรายงานที่เน้นความไม่สอดคล้องของข้อมูลที่ระบุได้ การแก้ไขที่ใช้ และเหตุผลเบื้องหลังการแก้ไขเหล่านั้น
3. **ความสำคัญของคุณลักษณะ:** โมเดล AI สามารถได้รับคำแนะนำพร้อมข้อมูลเกี่ยวกับความสำคัญของคุณลักษณะหรือคุณสมบัติต่างๆ ในกระบวนการแก้ไขข้อมูลเป็นส่วนหนึ่งของคำสั่งของพวกมัน คำสั่งเหล่านั้นสามารถเปิดเผยให้ผู้มีส่วนได้ส่วนเสียที่เป็นมนุษย์เห็นได้ การระบุปัจจัยสำคัญที่มีอิทธิพลต่อการตัดสินใจของโมเดล ช่วยให้ผู้มีส่วนได้ส่วนเสียเข้าใจเหตุผลเบื้องหลังการแก้ไขและประเมินความถูกต้องของการแก้ไขเหล่านั้น

4. **การบันทึกและการตรวจสอบ:** การใช้กลไกการบันทึกและการตรวจสอบที่ครอบคลุมมีความสำคัญในการรักษาความโปร่งใสในกระบวนการข้อมูลที่ซ่อมแซมตัวเอง การแก้ไขข้อมูลทุกครั้งที่ทำโดยโมเดล AI ควรถูกบันทึกไว้ รวมถึงข้อมูลดั้งเดิม ข้อมูลที่แก้ไขแล้ว และการดำเนินการเฉพาะที่ทำ ร่องรอยการตรวจสอบนี้ช่วยให้สามารถวิเคราะห์ย้อนหลังและให้บันทึกที่ชัดเจนของการแก้ไขที่ทำกับข้อมูล
5. **แนวทางการมีมนุษย์ร่วมในกระบวนการ:** การรวมแนวทางการมีมนุษย์ร่วมในกระบวนการสามารถเพิ่มความสามารถในการอธิบายและความโปร่งใสของเทคนิคข้อมูลที่ซ่อมแซมตัวเอง การให้ผู้เชี่ยวชาญที่เป็นมนุษย์มีส่วนร่วมในการตรวจสอบและยืนยันการแก้ไขที่สร้างโดย AI องค์กรสามารถมั่นใจได้ว่าการแก้ไขสอดคล้องกับความรู้ในโดเมนและข้อกำหนดทางธุรกิจ การกำกับดูแลโดยมนุษย์เพิ่มระดับความรับผิดชอบและช่วยให้สามารถระบุอคติหรือข้อผิดพลาดที่อาจเกิดขึ้นในโมเดล AI
6. **การติดตามและประเมินผลอย่างต่อเนื่อง:** การติดตามและประเมินผลการทำงานของเทคนิคข้อมูลที่ซ่อมแซมตัวเองอย่างสม่ำเสมอมีความสำคัญในการรักษาความโปร่งใสและความไว้วางใจ การประเมินความแม่นยำและประสิทธิผลของโมเดล AI ตลอดเวลา องค์กรสามารถระบุการเบี่ยงเบนหรือความผิดปกติใดๆ และดำเนินการแก้ไข การติดตามอย่างต่อเนื่องช่วยให้มั่นใจว่ากระบวนการข้อมูลที่ซ่อมแซมตัวเองยังคงน่าเชื่อถือและสอดคล้องกับผลลัพธ์ที่ต้องการ

ความสามารถในการอธิบายและความโปร่งใสเป็นข้อพิจารณาที่สำคัญเมื่อนำเทคนิคข้อมูลที่ซ่อมแซมตัวเองมาใช้ การให้คำอธิบายที่ชัดเจนสำหรับการแก้ไขข้อมูล การรักษาร่องรอยการตรวจสอบที่ครอบคลุม และการมีส่วนร่วมในการกำกับดูแลของมนุษย์ องค์กรสามารถสร้างความไว้วางใจในกระบวนการข้อมูลที่ซ่อมแซมตัวเองและมั่นใจได้ว่าการแก้ไขที่ทำกับข้อมูลสามารถอธิบายได้และสอดคล้องกับวัตถุประสงค์ทางธุรกิจ

สิ่งสำคัญคือต้องสร้างความสมดุลระหว่างประโยชน์ของระบบอัตโนมัติและความจำเป็นในการรักษาความโปร่งใส แม้ว่าเทคนิคข้อมูลที่ซ่อมแซมตัวเองสามารถปรับปรุงคุณภาพข้อมูลและประสิทธิภาพได้อย่างมาก แต่ไม่ควรแลกมาด้วยการสูญเสียการมองเห็นและการควบคุมกระบวนการแก้ไขข้อมูล การออกแบบผู้ปฏิบัติงานด้านข้อมูลที่ซ่อมแซมตัวเองโดยคำนึงถึงความสามารถในการอธิบายและความโปร่งใส องค์กรสามารถใช้ประโยชน์จากพลังของ AI ในขณะที่ยังคงรักษาระดับความรับผิดชอบและความไว้วางใจในข้อมูลที่จำเป็น

ผลกระทบที่ไม่ได้ตั้งใจ

ในขณะที่เทคนิคข้อมูลที่ซ่อมแซมตัวเองมีเป้าหมายเพื่อปรับปรุงคุณภาพและความสอดคล้องของข้อมูล สิ่งสำคัญคือต้องตระหนักถึงความเป็นไปได้ของผลกระทบที่ไม่ได้ตั้งใจ การแก้ไขอัตโนมัติ หากไม่ได้รับการออกแบบและติดตามอย่างระมัดระวัง อาจเปลี่ยนแปลงความหมายหรือบริบทของข้อมูลโดยไม่ได้ตั้งใจ นำไปสู่ปัญหาในภายหลัง

หนึ่งในความเสี่ยงหลักของข้อมูลที่ซ่อมแซมตัวเองคือการแนะนำอคติหรือข้อผิดพลาดในกระบวนการแก้ไขข้อมูล โมเดล AI เช่นเดียวกับระบบซอฟต์แวร์อื่นๆ อาจมีอคติที่มีอยู่ในข้อมูลการฝึกฝนหรือถูกแนะนำผ่านการออกแบบของอัลกอริทึม หากไม่มีการระบุและบรรเทาอคติเหล่านี้ พวกมันอาจแพร่กระจายผ่านกระบวนการข้อมูลที่ซ่อมแซมตัวเองและส่งผลให้เกิดการแก้ไขข้อมูลที่บิดเบือนหรือไม่ถูกต้อง

ยกตัวอย่างเช่น พิจารณาระบบจัดการข้อมูลแบบซ่อมแซมตัวเองที่มีหน้าที่แก้ไขความไม่สอดคล้องในข้อมูลประชากรของลูกค้า หากโมเดล AI เรียนรู้อคติจากข้อมูลในอดีต เช่น การเชื่อมโยงอาชีพหรือระดับรายได้บางอย่างกับเพศหรือเชื้อชาติเฉพาะ อาจทำให้เกิดการตั้งสมมติฐานที่ไม่ถูกต้องและแก้ไขข้อมูลในลักษณะที่เสริมอคติเหล่านั้น ซึ่งอาจนำไปสู่โปรไฟล์ลูกค้าที่ไม่ถูกต้อง การตัดสินใจทางธุรกิจที่ผิดพลาด และอาจก่อให้เกิดผลลัพธ์ที่เลือกปฏิบัติได้

ผลกระทบที่ไม่คาดคิดอีกประการหนึ่งคือการสูญเสียข้อมูลหรือบริบทที่มีคุณค่าระหว่างกระบวนการแก้ไขข้อมูล เทคนิคการจัดการข้อมูลแบบซ่อมแซมตัวเองมักเน้นการทำให้ข้อมูลเป็นมาตรฐานและปรับให้เป็นปกติเพื่อให้แน่ใจว่ามีความสอดคล้องกัน อย่างไรก็ตาม ในบางกรณี ข้อมูลดั้งเดิมอาจมีความละเอียดอ่อน ข้อยกเว้น หรือข้อมูลเชิงบริบทที่สำคัญต่อการทำความเข้าใจภาพรวมทั้งหมด การแก้ไขอัตโนมัติที่บังคับใช้มาตรฐานโดยไม่คำนึงถึงสิ่งเหล่านี้อาจทำให้ข้อมูลที่มีคุณค่าสูญหายหรือคลุมเครือโดยไม่ตั้งใจ

ตัวอย่างเช่น จินตนาการถึงระบบจัดการข้อมูลแบบซ่อมแซมตัวเองที่รับผิดชอบการแก้ไขความไม่สอดคล้องในประวัติการรักษาพยาบาล หากระบบพบประวัติการรักษาของผู้ป่วยที่มีอาการหายากหรือแผนการรักษาที่ไม่ปกติ อาจพยายามปรับข้อมูลให้เข้ากับรูปแบบที่พบบ่อยกว่า อย่างไรก็ตาม การทำเช่นนี้อาจทำให้สูญเสียรายละเอียดเฉพาะและบริบทที่สำคัญต่อการแสดงสถานการณ์เฉพาะของผู้ป่วย การสูญเสียข้อมูลนี้อาจส่งผลร้ายแรงต่อการดูแลผู้ป่วยและการตัดสินใจทางการแพทย์

เพื่อลดความเสี่ยงของผลกระทบที่ไม่คาดคิด จำเป็นต้องใช้วิธีการเชิงรุกในการออกแบบและนำเทคนิคการจัดการข้อมูลแบบซ่อมแซมตัวเองไปใช้:

1. **การทดสอบและตรวจสอบอย่างละเอียด:** ก่อนที่จะนำระบบจัดการข้อมูลแบบซ่อมแซมตัวเองไปใช้งานจริง จำเป็นต้องทดสอบและตรวจสอบพฤติกรรมอย่างละเอียดกับสถานการณ์ที่หลากหลาย รวมถึงการทดสอบกับชุดข้อมูลตัวอย่างที่ครอบคลุมกรณีพิเศษต่างๆ ข้อยกเว้น และอคติที่อาจเกิดขึ้น การทดสอบอย่างเข้มงวดช่วยระบุและแก้ไขผลกระทบที่ไม่คาดคิดก่อนที่จะส่งผลต่อข้อมูลในโลกแห่งความเป็นจริง
2. **การติดตามและประเมินผลอย่างต่อเนื่อง:** การใช้กลไกการติดตามและประเมินผลอย่างต่อเนื่องเป็นสิ่งสำคัญในการตรวจจับและบรรเทาผลกระทบที่ไม่คาดคิดตลอดเวลา การทบทวนผลลัพธ์ของกระบวนการจัดการข้อมูลแบบซ่อมแซมตัวเองเป็นประจำ การวิเคราะห์ผลกระทบต่อระบบปลายทางและการตัดสินใจ และการรวบรวมข้อเสนอแนะจากผู้มีส่วนได้ส่วนเสียสามารถช่วยระบุผลกระทบที่ไม่พึงประสงค์และกระตุ้นให้มีการดำเนินการแก้ไขได้ทันท่วงที หากองค์กร

ของคุณมีแดชบอร์ดการปฏิบัติงาน การเพิ่มเมตริกที่เห็นได้ชัดเจนเกี่ยวกับการเปลี่ยนแปลงข้อมูลอัตโนมัติน่าจะเป็นความคิดที่ดี การเพิ่มการแจ้งเตือนที่เชื่อมโยงกับการเปลี่ยนแปลงข้อมูลที่ผิดปกติอย่างมากน่าจะเป็นความคิดที่ดียิ่งขึ้น!

3. **การกำกับดูแลและการแทรกแซงโดยมนุษย์:** การรักษาการกำกับดูแลโดยมนุษย์และความสามารถในการแทรกแซงในกระบวนการจัดการข้อมูลแบบซ่อมแซมตัวเองเป็นสิ่งสำคัญ แม้ว่าระบบอัตโนมัติจะสามารถปรับปรุงประสิทธิภาพได้อย่างมาก แต่สำคัญที่จะต้องมีผู้เชี่ยวชาญมนุษย์ตรวจสอบและยืนยันการแก้ไขที่ทำโดยโมเดล AI โดยเฉพาะในโดเมนที่สำคัญหรือละเอียดอ่อน การตัดสินใจของมนุษย์และความเชี่ยวชาญในโดเมนสามารถช่วยระบุและแก้ไขผลกระทบที่ไม่คาดคิดที่อาจเกิดขึ้นได้

4. **ปัญญาประดิษฐ์ที่อธิบายได้ (XAI) และความโปร่งใส:** ตามที่ได้กล่าวไว้ในหัวข้อย่อยก่อนหน้านี้ การรวมเทคนิค XAI และการรับประกันความโปร่งใสในกระบวนการจัดการข้อมูลแบบซ่อมแซมตัวเองสามารถช่วยบรรเทาผลกระทบที่ไม่คาดคิดได้ การให้คำอธิบายที่ชัดเจนสำหรับการแก้ไขข้อมูลและการรักษาบันทึกการตรวจสอบที่ครอบคลุม องค์กรสามารถเข้าใจและติดตามเหตุผลเบื้องหลังการแก้ไขที่ทำโดยโมเดล AI ได้ดีขึ้น
5. **วิธีการแบบค่อยเป็นค่อยไปและการทำซ้ำ:** การใช้วิธีการแบบค่อยเป็นค่อยไปและการทำซ้ำสำหรับการจัดการข้อมูลแบบซ่อมแซมตัวเองสามารถช่วยลดความเสี่ยงของผลกระทบที่ไม่คาดคิด แทนที่จะใช้การแก้ไขอัตโนมัติกับชุดข้อมูลทั้งหมดในครั้งเดียว ให้เริ่มต้นด้วยข้อมูลส่วนหนึ่งและค่อยๆ ขยายขอบเขตเมื่อเทคนิคพิสูจน์แล้วว่ามีประสิทธิภาพและน่าเชื่อถือ วิธีนี้ช่วยให้สามารถติดตามและปรับแต่งได้ตลอดทาง ลดผลกระทบของผลกระทบที่ไม่คาดคิด
6. **การทำงานร่วมกันและข้อเสนอแนะ:** การให้ผู้มีส่วนได้ส่วนเสียจากโดเมนต่างๆ มีส่วนร่วมและส่งเสริมการทำงานร่วมกันและข้อเสนอแนะตลอดกระบวนการจัดการข้อมูลแบบซ่อมแซมตัวเองสามารถช่วยระบุและแก้ไขผลกระทบที่ไม่คาดคิดได้ การรับฟังความคิดเห็นจากผู้เชี่ยวชาญในโดเมน ผู้ใช้ข้อมูล และผู้ใช้ปลายทางอย่างสม่ำเสมอสามารถให้ข้อมูลเชิงลึกที่มีคุณค่าเกี่ยวกับผลกระทบในโลกแห่งความเป็นจริงของการแก้ไขข้อมูลและชี้ให้เห็นปัญหาที่อาจถูกมองข้าม

การจัดการความเสี่ยงของผลกระทบที่ไม่คาดคิดเชิงรุกและการนำมาตรการป้องกันที่เหมาะสมมาใช้ องค์กรสามารถใช้ประโยชน์จากเทคนิคการจัดการข้อมูลแบบซ่อมแซมตัวเองในขณะที่ลดผลกระทบที่ไม่พึงประสงค์ที่อาจเกิดขึ้น สำคัญที่จะต้องมองการจัดการข้อมูลแบบซ่อมแซมตัวเองเป็นกระบวนการที่ต้องทำซ้ำและทำงานร่วมกัน มีการติดตาม ประเมินผล และปรับปรุงเทคนิคอย่างต่อเนื่องเพื่อให้แน่ใจว่าสอดคล้องกับผลลัพธ์ที่ต้องการและรักษาความถูกต้องและความน่าเชื่อถือของข้อมูล

เมื่อพิจารณาการใช้รูปแบบการจัดการข้อมูลแบบซ่อมแซมตัวเอง จำเป็นต้องประเมินปัจจัยเหล่านี้อย่างรอบคอบและชั่งน้ำหนักระหว่างประโยชน์กับความเสี่ยงและข้อจำกัดที่อาจเกิดขึ้น ในบางกรณี วิธีการแบบผสมผสานที่รวมการแก้ไขอัตโนมัติกับการกำกับดูแลและการแทรกแซงของมนุษย์อาจเป็นวิธีแก้ปัญหาที่เหมาะสมที่สุด

นอกจากนี้ ควรสังเกตว่าเทคนิคการจัดการข้อมูลแบบซ่อมแซมตัวเองไม่ควรถูกมองว่าเป็นการทดแทนกลไกการตรวจสอบข้อมูล การกรองข้อมูลนำเข้า และการจัดการข้อผิดพลาดที่แข็งแกร่ง แนวปฏิบัติพื้นฐานเหล่านี้ยังคงมีความสำคัญต่อการรับประกันความถูกต้องและความปลอดภัยของข้อมูล การจัดการข้อมูลแบบซ่อมแซมตัวเองควรถูกมองว่าเป็นวิธีการเสริมที่สามารถเพิ่มประสิทธิภาพและยกระดับมาตรการที่มีอยู่เหล่านี้

ในท้ายที่สุด การตัดสินใจใช้รูปแบบการจัดการข้อมูลแบบซ่อมแซมตัวเองขึ้นอยู่กับความต้องการเฉพาะ ข้อจำกัด และลำดับความสำคัญของแอปพลิเคชันของคุณ การพิจารณาข้อควรคำนึงที่กล่าวมาข้างต้นอย่างรอบคอบและการจัดให้สอดคล้องกับเป้าหมายและสถาปัตยกรรมของแอปพลิเคชันของคุณ คุณสามารถตัดสินใจอย่างมีข้อมูลเกี่ยวกับเวลาและวิธีการใช้เทคนิคการจัดการข้อมูลแบบซ่อมแซมตัวเองอย่างมีประสิทธิภาพ

การสร้างเนื้อหาตามบริบท

รูปแบบการสร้างเนื้อหาตามบริบท ใช้ประโยชน์จากพลังของโมเดลภาษาขนาดใหญ่ (LLMs) เพื่อสร้างเนื้อหาแบบไดนามิกและเฉพาะบริบทภายในแอปพลิเคชัน รูปแบบในหมวดหมู่นี้ตระหนักถึงความสำคัญของการนำเสนอเนื้อหาที่ปรับแต่งให้เข้ากับผู้ใช้และมีความเกี่ยวข้องกับความต้องการเฉพาะ ความชอบ และแม้แต่การมีปฏิสัมพันธ์ทั้งในอดีตและปัจจุบันกับแอปพลิเคชัน

> ในบริบทของแนวทางนี้ “เนื้อหา” หมายถึงทั้งเนื้อหาหลัก (เช่น บล็อกโพสต์ บทความ ฯลฯ) และเมตาคอนเทนต์ เช่น คำแนะนำที่เกี่ยวข้องกับเนื้อหาหลัก

รูปแบบการสร้างเนื้อหาตามบริบทสามารถมีบทบาทสำคัญในการเพิ่มระดับการมีส่วนร่วมของผู้ใช้ การมอบประสบการณ์ที่ปรับแต่งเฉพาะบุคคล และการทำให้งานสร้างเนื้อหาเป็นอัตโนมัติทั้งสำหรับคุณและผู้ใช้ของคุณ ด้วยการใช้รูปแบบที่เราอธิบายในบทนี้ คุณสามารถสร้างแอปพลิเคชันที่สร้างเนื้อหาแบบไดนามิก ปรับตัวตามบริบทและข้อมูลนำเข้าแบบเรียลไทม์

รูป แบบ เหล่า นี้ ทำงาน โดย การ ผสาน รวม LLMs เข้า กับ ผลลัพธ์ ของ แอปพลิเคชัน ตั้งแต่ ส่วน ติดต่อ ผู้ ใช้ (บาง ครั้ง เรียก ว่า “chrome”) ไปจนถึงอีเมลและการแจ้งเตือนรูปแบบอื่นๆ รวมถึงไปป์ไลน์การสร้างเนื้อหาต่างๆ

เมื่อผู้ใช้มีปฏิสัมพันธ์กับแอปพลิเคชันหรือเริ่มต้นคำขอเนื้อหาเฉพาะ แอปพลิเคชันจะจับบริบทที่เกี่ยวข้อง เช่น ความชอบของผู้ใช้ การมีปฏิสัมพันธ์ก่อนหน้า หรือคำแนะนำเฉพาะ ข้อมูลบริบทนี้จะถูกป้อนเข้าสู่ LLM พร้อมกับเทมเพลตหรือแนวทางที่จำเป็น และใช้เพื่อสร้างผลลัพธ์ที่เป็นข้อความซึ่งมิฉะนั้นจะต้องถูกเขียนโค้ดแบบตายตัว จัดเก็บในฐานข้อมูล หรือสร้างขึ้นด้วยอัลกอริทึม

เนื้อหา ที่ สร้าง โดย LLM สามารถ อยู่ ใน รูป แบบ ต่างๆ เช่น คำ แนะนำ ที่ ปรับ แต่ง เฉพาะ บุคคล คำ อธิบาย ผลิตภัณฑ์ แบบ ไดนามิก การ ตอบ กลับ อีเมล ที่ ปรับ แต่ง หรือ แม้แต่ บทความ หรือ บล็อก โพสต์ ทั้งหมด หนึ่งในการใช้งานที่ปฏิวัติวงการมากที่สุดที่ผมได้ริเริ่มเมื่อกว่าหนึ่งปีที่แล้ว คือ การ สร้าง องค์ ประกอบ UI แบบ ไดนามิก เช่น ป้ายกำกับฟอร์ม ทูลทิป และข้อความอธิบายประเภทอื่นๆ

การปรับแต่งให้เป็นส่วนบุคคล

หนึ่งในประโยชน์หลักของรูปแบบการสร้างเนื้อหาตามบริบทคือความสามารถในการมอบประสบการณ์ที่ปรับแต่งเฉพาะบุคคลอย่างมากให้กับผู้ใช้ ด้วยการสร้างเนื้อหาตามบริบทเฉพาะของผู้ใช้ รูปแบบเหล่านี้ช่วยให้แอปพลิเคชันสามารถปรับแต่งเนื้อหาให้เข้ากับความสนใจ ความชอบ และการมีปฏิสัมพันธ์ของผู้ใช้แต่ละคน

การปรับแต่งให้เป็นส่วนบุคคลนั้นไม่ได้จำกัดอยู่แค่การแทรกชื่อผู้ใช้ลงในเนื้อหาทั่วไป แต่เกี่ยวข้องกับการใช้ประโยชน์จากบริบทที่หลากหลายที่มีอยู่เกี่ยวกับผู้ใช้แต่ละคนเพื่อสร้างเนื้อหาที่ตรงกับความต้องการและความปรารถนาเฉพาะของพวกเขา บริบทนี้สามารถรวมถึงปัจจัยต่างๆ มากมาย เช่น:

1. **ข้อมูลโปรไฟล์ผู้ใช้:** ในระดับทั่วไปที่สุดของการใช้เทคนิคนี้ ข้อมูลประชากรศาสตร์ ความสนใจ ความชอบ และคุณลักษณะโปรไฟล์อื่นๆ สามารถนำมาใช้เพื่อสร้างเนื้อหาที่สอดคล้องกับพื้นฐานและลักษณะของผู้ใช้
2. **ข้อมูลพฤติกรรม:** การมีปฏิสัมพันธ์ในอดีตของผู้ใช้กับแอปพลิเคชัน เช่น หน้าที่เข้าชม ลิงก์ที่คลิก หรือผลิตภัณฑ์ที่ซื้อ สามารถให้ข้อมูลเชิงลึกที่มีค่าเกี่ยวกับพฤติกรรมและความสนใจของพวกเขา ข้อมูลนี้สามารถนำมาใช้เพื่อสร้างข้อเสนอแนะเนื้อหาที่สะท้อนรูปแบบการมีส่วนร่วมและคาดการณ์ความต้องการในอนาคตของพวกเขา
3. **ปัจจัยด้านบริบท:** บริบทปัจจุบันของผู้ใช้ เช่น ตำแหน่งที่ตั้ง อุปกรณ์ เวลาของวัน หรือแม้แต่สภาพอากาศ สามารถมีอิทธิพลต่อกระบวนการสร้างเนื้อหา ตัวอย่างเช่น แอปพลิเคชันท่องเที่ยวอาจมี AI worker ที่สามารถสร้างคำแนะนำที่ปรับแต่งตามตำแหน่งที่ตั้งปัจจุบันของผู้ใช้และสภาพอากาศที่เป็นอยู่

ด้วยการใช้ประโยชน์จากปัจจัยด้านบริบทเหล่านี้ รูปแบบการสร้างเนื้อหาตามบริบทช่วยให้แอปพลิเคชันสามารถนำเสนอเนื้อหาที่รู้สึกเหมือนถูกสร้างขึ้นมาเฉพาะสำหรับผู้ใช้แต่ละคน การปรับแต่งให้เป็นส่วนบุคคลในระดับนี้มีประโยชน์ที่สำคัญหลายประการ:

1. **การมีส่วนร่วมที่เพิ่มขึ้น:** เนื้อหาที่ปรับแต่งเฉพาะบุคคลดึงดูดความสนใจของผู้ใช้และทำให้พวกเขามีส่วนร่วมกับแอปพลิเคชันอย่างต่อเนื่อง เมื่อผู้ใช้รู้สึกว่าเนื้อหามีความเกี่ยวข้องและตอบสนองความต้องการของพวกเขาโดยตรง พวกเขามีแนวโน้มที่จะใช้เวลามากขึ้นในการมีปฏิสัมพันธ์กับแอปพลิเคชันและสำรวจคุณสมบัติต่างๆ
2. **ความพึงพอใจของผู้ใช้ที่ดีขึ้น:** เนื้อหาที่ปรับแต่งเฉพาะบุคคลแสดงให้เห็นว่าแอปพลิเคชันเข้าใจและใส่ใจต่อความต้องการเฉพาะของผู้ใช้ ด้วยการนำเสนอเนื้อหาที่มีประโยชน์ ให้ข้อมูล และสอดคล้องกับความสนใจของพวกเขา แอปพลิเคชันสามารถเพิ่มความพึงพอใจของผู้ใช้และสร้างความสัมพันธ์ที่แข็งแกร่งขึ้นกับผู้ใช้
3. **อัตราการเปลี่ยนแปลงเป็นลูกค้าที่สูงขึ้น:** ในบริบทของอีคอมเมิร์ซหรือแอปพลิเคชันการตลาด เนื้อหาที่ปรับแต่งเฉพาะบุคคลสามารถส่งผลกระทบอย่างมีนัยสำคัญต่ออัตราการเปลี่ยนแปลงเป็นลูกค้า ด้วยการนำเสนอผลิตภัณฑ์ ข้อเสนอ หรือคำแนะนำที่ปรับแต่งตามความชอบและพฤติกรรมของผู้ใช้ แอปพลิเคชันสามารถเพิ่มโอกาสที่ผู้ใช้จะดำเนินการตามที่ต้องการ เช่น การซื้อสินค้าหรือการลงทะเบียนใช้บริการ

ผลิตภาพ

รูปแบบการสร้างเนื้อหาตามบริบทสามารถเพิ่มผลิตภาพบางประเภทได้อย่างมีนัยสำคัญ โดยลดความจำเป็นในการสร้างเนื้อหาด้วยตนเองและการแก้ไขในกระบวนการสร้างสรรค์ ด้วยการใช้พลังของ LLMs คุณสามารถสร้างเนื้อหาคุณภาพสูงในปริมาณมากได้ ช่วยประหยัดเวลาและความพยายามที่ผู้สร้างเนื้อหาและนักพัฒนาของคุณจะต้องใช้ในการทำงานที่น่าเบื่อด้วยตนเอง

ตามแบบดั้งเดิมแล้ว ผู้สร้างเนื้อหาจำเป็นต้องทำวิจัย เขียน แก้ไข และจัดรูปแบบเนื้อหาเพื่อให้แน่ใจว่าตรงตามความต้องการของแอปพลิเคชันและความคาดหวังของผู้ใช้ กระบวนการนี้อาจใช้เวลาและทรัพยากรมาก โดยเฉพาะเมื่อปริมาณเนื้อหาเพิ่มขึ้น

อย่างไรก็ตาม ด้วยรูปแบบการสร้างเนื้อหาตามบริบท กระบวนการสร้างเนื้อหาสามารถทำให้เป็นอัตโนมัติได้เป็นส่วนใหญ่ โมเดลภาษาขนาดใหญ่สามารถสร้างเนื้อหาที่มีความสอดคล้อง ถูกต้องตามหลักไวยากรณ์ และเกี่ยวข้องกับบริบทตามคำสั่งและแนวทางที่กำหนดให้ การทำงานอัตโนมัตินี้มอบประโยชน์ด้านผลิตภาพหลายประการ:

1. **ลดการทำงานด้วยมือ:** การมอบหมายงานสร้างเนื้อหาให้กับโมเดลภาษาขนาดใหญ่ ช่วยให้ผู้สร้างเนื้อหาสามารถมุ่งเน้นไปที่งานระดับสูงขึ้น เช่น กลยุทธ์เนื้อหา การคิดสร้างสรรค์ และการประกันคุณภาพ พวกเขาสามารถให้บริบทที่จำเป็น เทมเพลต และแนวทางแก่โมเดลภาษาขนาดใหญ่และปล่อยให้มันจัดการการสร้างเนื้อหาจริง สิ่งนี้ช่วยลดความพยายามในการเขียนและแก้ไขด้วยมือ ช่วยให้ผู้สร้างเนื้อหาทำงานได้อย่างมีประสิทธิภาพและประสิทธิผลมากขึ้น
2. **สร้างเนื้อหาได้เร็วขึ้น:** โมเดลภาษาขนาดใหญ่สามารถสร้างเนื้อหาได้เร็วกว่านักเขียนมนุษย์มาก ด้วยคำสั่งและแนวทางที่เหมาะสม โมเดลภาษาขนาดใหญ่สามารถสร้างเนื้อหาหลายชิ้นได้ภายในไม่กี่วินาทีหรือไม่กี่นาที ความเร็วนี้ช่วยให้แอปพลิเคชันสามารถสร้างเนื้อหาได้เร็วขึ้นมาก ทันต่อความต้องการของผู้ใช้และภูมิทัศน์ดิจิทัลที่เปลี่ยนแปลงตลอดเวลา

> การสร้างเนื้อหาที่เร็วขึ้นนี้กำลังนำไปสู่สถานการณ์ “โศกนาฏกรรมของส่วนรวม” หรือไม่ ที่อินเทอร์เน็ตกำลังจมอยู่กับเนื้อหาที่ไม่มีใครอ่าน น่าเสียดายที่ผมสงสัยว่าคำตอบคือใช่

3. **ความสม่ำเสมอและคุณภาพ:** โมเดลภาษาขนาดใหญ่สามารถแก้ไขเนื้อหาให้มีความสม่ำเสมอในด้านสไตล์ โทน และคุณภาพได้อย่างง่ายดาย เมื่อมีแนวทางและตัวอย่างที่ชัดเจน แอปพลิเคชันบางประเภท (เช่น ห้องข่าว ประชาสัมพันธ์ ฯลฯ) สามารถทำให้แน่ใจได้ว่าเนื้อหาที่มนุษย์สร้างขึ้นสอดคล้องกับเสียงของแบรนด์และตรงตามมาตรฐานคุณภาพที่ต้องการ ความสม่ำเสมอนี้ช่วยลดความจำเป็นในการแก้ไขและปรับปรุงอย่างกว้างขวาง ประหยัดเวลาและความพยายามในกระบวนการสร้างเนื้อหา
4. **การทำซ้ำและการปรับให้เหมาะสม:** รูปแบบการสร้างเนื้อหาตามบริบท ช่วยให้สามารถทำซ้ำและปรับเนื้อหาให้เหมาะสมได้อย่างรวดเร็ว ด้วยการปรับคำสั่ง เทมเพลต หรือแนวทางที่ให้กับโมเดลภาษาขนาดใหญ่ แอปพลิเคชันของคุณสามารถสร้างเนื้อหาหลากหลายรูปแบบและทดสอบวิธีการต่างๆ ในรูปแบบอัตโนมัติซึ่งไม่เคยเป็นไปได้มาก่อน กระบวนการทำซ้ำนี้ช่วยให้สามารถทดลองและปรับปรุงกลยุทธ์เนื้อหาได้เร็วขึ้น นำไปสู่เนื้อหาที่มีประสิทธิภาพและน่าสนใจมากขึ้นเมื่อเวลาผ่านไป เทคนิคนี้โดยเฉพาะสามารถเปลี่ยนเกมได้อย่างสิ้นเชิงสำหรับแอปพลิเคชันเช่นอีคอมเมิร์ซที่อยู่รอดหรือล้มเหลวขึ้นอยู่กับอัตราการตีกลับและการมีส่วนร่วม

สิ่งสำคัญที่ต้องทราบคือแม้ว่ารูปแบบการสร้างเนื้อหาตามบริบท จะ สามารถ เพิ่ม ประสิทธิภาพ การ ทำงาน ได้อย่างมาก แต่ก็ไม่ได้กำจัดความจำเป็นในการมีส่วนร่วมของมนุษย์ออกไปทั้งหมด ผู้สร้างเนื้อหาและบรรณาธิการยังคงมีบทบาทสำคัญในการกำหนดกลยุทธ์เนื้อหาโดยรวม การให้คำแนะนำแก่โมเดลภาษาขนาดใหญ่ และการรับรองคุณภาพและความเหมาะสมของเนื้อหาที่สร้างขึ้น

ด้วยการทำให้การสร้างเนื้อหาที่ซ้ำซากและใช้เวลามากเป็นอัตโนมัติ รูปแบบการสร้างเนื้อหาตามบริบท ช่วยปลดปล่อยเวลาและทรัพยากรมนุษย์ที่มีค่าซึ่งสามารถนำไปใช้กับงานที่มีมูลค่าสูงกว่า ประสิทธิภาพการทำงานที่เพิ่มขึ้นนี้ช่วยให้คุณสามารถส่งมอบเนื้อหาที่เป็นส่วนตัวและน่าสนใจมากขึ้นให้กับผู้ใช้ ในขณะที่ปรับปรุงขั้นตอนการสร้างเนื้อหาให้เหมาะสม

การทำซ้ำและการทดลองอย่างรวดเร็ว

รูปแบบการสร้างเนื้อหาตามบริบท ช่วยให้คุณสามารถทำซ้ำและทดลองกับเนื้อหาที่แตกต่างกันได้อย่างรวดเร็ว ช่วยให้สามารถปรับปรุงและปรับแต่งกลยุทธ์เนื้อหาของคุณได้เร็วขึ้น คุณสามารถสร้างเนื้อหาหลายเวอร์ชันได้ภายในไม่กี่วินาที เพียงแค่ปรับบริบท เทมเพลต หรือแนวทางที่ให้กับโมเดล

ความสามารถในการทำซ้ำอย่างรวดเร็วนี้มอบประโยชน์สำคัญหลายประการ:

1. **การทดสอบและการปรับให้เหมาะสม:** ด้วยความสามารถในการสร้างเนื้อหาที่แตกต่างกันได้อย่างรวดเร็ว คุณสามารถทดสอบวิธีการต่างๆ และวัดประสิทธิผลได้อย่างง่ายดาย ตัวอย่างเช่น คุณสามารถสร้างคำอธิบายผลิตภัณฑ์หรือข้อความทางการตลาดหลายเวอร์ชัน แต่ละเวอร์ชันปรับแต่งให้เหมาะกับกลุ่มผู้ใช้หรือบริบทเฉพาะ โดยการวิเคราะห์ตัวชี้วัดการมีส่วนร่วมของผู้ใช้ เช่น อัตราการคลิกหรืออัตราการแปลงเป็นลูกค้า คุณสามารถระบุเนื้อหาที่มีประสิทธิภาพมากที่สุดและปรับกลยุทธ์เนื้อหาของคุณให้เหมาะสม

2. **การทดสอบแบบ A/B:** รูปแบบการสร้างเนื้อหาตามบริบทช่วยให้สามารถทำการทดสอบแบบ A/B กับเนื้อหาได้อย่างราบรื่น คุณสามารถสร้างเนื้อหาสองหรือมากกว่าสองเวอร์ชันและแสดงให้กับกลุ่มผู้ใช้ที่แตกต่างกันแบบสุ่ม โดยการเปรียบเทียบประสิทธิภาพของแต่ละเวอร์ชัน คุณสามารถกำหนดได้ว่าเนื้อหาแบบใดที่เข้าถึงกลุ่มเป้าหมายของคุณได้ดีที่สุด วิธีการที่ขับเคลื่อนด้วยข้อมูลนี้ช่วยให้คุณสามารถตัดสินใจอย่างมีข้อมูลและปรับปรุงเนื้อหาของคุณอย่างต่อเนื่องเพื่อเพิ่มการมีส่วนร่วมของผู้ใช้และบรรลุผลลัพธ์ที่ต้องการ
3. **การทดลองการปรับแต่งให้เข้ากับบุคคล:** การทำซ้ำและการทดลองอย่างรวดเร็วมีคุณค่าเป็นพิเศษเมื่อเกี่ยวข้องกับการปรับแต่งให้เข้ากับบุคคล ด้วยรูปแบบการสร้างเนื้อหาตามบริบท คุณสามารถสร้างเนื้อหาที่ปรับแต่งให้เข้ากับบุคคลได้อย่างรวดเร็วตามกลุ่มผู้ใช้ ความชอบ หรือพฤติกรรมที่แตกต่างกัน โดยการทดลองกับกลยุทธ์การปรับแต่งให้เข้ากับบุคคลที่แตกต่างกัน คุณสามารถระบุวิธีการที่มีประสิทธิภาพมากที่สุดในการสร้างการมีส่วนร่วมกับผู้ใช้แต่ละคนและส่งมอบประสบการณ์ที่ปรับแต่งให้เหมาะสม
4. **การปรับตัวตามเทรนด์ที่เปลี่ยนแปลง:** ความสามารถในการทำซ้ำและทดลองอย่างรวดเร็วช่วยให้คุณยังคงความคล่องตัวและปรับตัวตามเทรนด์และความชอบของผู้ใช้ที่เปลี่ยนแปลงไป เมื่อมีหัวข้อใหม่ คำสำคัญ หรือพฤติกรรมผู้ใช้เกิดขึ้น คุณสามารถสร้างเนื้อหาที่สอดคล้องกับเทรนด์เหล่านี้ได้อย่างรวดเร็ว ด้วยการทดลองและปรับปรุงเนื้อหาอย่างต่อเนื่อง คุณสามารถรักษาความทันสมัยและความได้เปรียบในการแข่งขันในภูมิทัศน์ดิจิทัลที่เปลี่ยนแปลงตลอดเวลา
5. **การทดลองที่คุ้มค่า:** การทดลองเนื้อหาแบบดั้งเดิมมักเกี่ยวข้องกับเวลาและทรัพยากรที่มาก เนื่องจากผู้สร้างเนื้อหาต้องพัฒนาและทดสอบรูปแบบต่างๆ ด้วยตนเอง อย่างไรก็ตาม ด้วยรูปแบบการสร้างเนื้อหาตามบริบท ต้นทุนในการทดลองลดลงอย่างมาก LLMs สามารถสร้างรูปแบบเนื้อหาได้อย่างรวดเร็วและมีขนาดใหญ่ ช่วยให้คุณสำรวจแนวคิดและวิธีการที่หลากหลายโดยไม่ต้องเสียค่าใช้จ่ายมาก

เพื่อใช้ประโยชน์สูงสุดจากการทำซ้ำและการทดลองอย่างรวดเร็ว สิ่งสำคัญคือต้องมีกรอบการทดลองที่กำหนดไว้อย่างชัดเจน กรอบนี้ควรประกอบด้วย:

- วัตถุประสงค์และสมมติฐานที่ชัดเจนสำหรับแต่ละการทดลอง

- ตัวชี้วัดและกลไกการติดตามที่เหมาะสมเพื่อวัดประสิทธิภาพของเนื้อหา
- กลยุทธ์การแบ่งส่วนและการกำหนดเป้าหมายเพื่อให้แน่ใจว่าเนื้อหาที่แตกต่างกันถูกส่งไปยังผู้ใช้ที่เหมาะสม
- เครื่องมือวิเคราะห์และรายงานเพื่อสกัดข้อมูลเชิงลึกจากข้อมูลการทดลอง
- กระบวนการนำบทเรียนและการปรับปรุงไปใช้ในกลยุทธ์เนื้อหาของคุณ

ด้วย การ ยอมรับ การ ทำ ซ้ำ และ การ ทดลอง อย่าง รวดเร็ว คุณ สามารถ ปรับปรุง และ เพิ่ม ประสิทธิภาพ เนื้อหา ของ คุณ อย่าง ต่อ เนื่อง เพื่อ ให้ แน่ใจ ว่า ยัง คง น่า สนใจ เกี่ยวข้อง และ มี ประสิทธิภาพ ใน การ บรรลุ เป้า หมาย ของ แอปพลิเคชัน ของ คุณ วิธีการที่คล่องตัวในการสร้างเนื้อหานี้ช่วยให้คุณอยู่เหนือคู่แข่งและมอบประสบการณ์ผู้ใช้ที่ยอดเยี่ยม

ความสามารถในการขยายและประสิทธิภาพ

เมื่อแอปพลิเคชันเติบโตและความต้องการเนื้อหาที่เป็นส่วนตัวเพิ่มขึ้น รูปแบบการสร้างเนื้อหาตามบริบทช่วยให้การขยายการสร้างเนื้อหามีประสิทธิภาพ LLMs สามารถสร้างเนื้อหาสำหรับผู้ใช้จำนวนมากและบริบทต่างๆ พร้อมกัน โดยไม่จำเป็นต้องเพิ่มทรัพยากรบุคคลตามสัดส่วน ความสามารถในการขยายนี้ช่วยให้แอปพลิเคชันสามารถมอบประสบการณ์ที่เป็นส่วนตัวให้กับฐานผู้ใช้ที่เติบโตขึ้นโดยไม่ทำให้ความสามารถในการสร้างเนื้อหาต้องตึงเครียด

สังเกตว่าการสร้างเนื้อหาตามบริบทสามารถใช้ในการทำให้แอปพลิเคชันของคุณเป็นสากล “แบบทันที” ได้อย่างมีประสิทธิภาพ จริงๆ แล้วนั่นคือสิ่งที่ผมทำโดยใช้ Instant18n Gem ของผมเพื่อส่งมอบ Olympia ในมากกว่าครึ่งโหลภาษา แม้ว่าเราจะอายุไม่ถึงหนึ่งปีก็ตาม

การแปลงให้เข้ากับท้องถิ่นด้วย AI

หากคุณอนุญาตให้ผมอวดสักครู่ ผมคิดว่าไลบรารี Instant18n ของผมสำหรับแอปพลิเคชัน Rails เป็นตัวอย่างที่ก้าวหน้าของรูปแบบ “การสร้างเนื้อหาตามบริบท” ที่แสดงให้เห็นศักยภาพในการเปลี่ยนแปลงของ AI ในการพัฒนาแอปพลิเคชัน เจมนี้ใช้ประโยชน์จากพลังของโมเดลภาษาขนาดใหญ่ GPT ของ OpenAI เพื่อปฏิวัติวิธีการจัดการการทำให้เป็นสากลและการแปลงให้เข้ากับท้องถิ่นในแอปพลิเคชัน Rails

ตามแบบดั้งเดิม การทำให้แอปพลิเคชัน Rails เป็นสากลเกี่ยวข้องกับการกำหนดคีย์การแปลและการให้การแปลที่สอดคล้องกันสำหรับแต่ละภาษาที่รองรับด้วยตนเอง กระบวนการนี้อาจใช้เวลานาน ใช้ทรัพยากรมาก และมีแนวโน้มที่จะเกิดความไม่สอดคล้องกัน อย่างไรก็ตาม ด้วยเจม Instant18n กระบวนทัศน์ของการแปลงให้เข้ากับท้องถิ่นได้ถูกนิยามใหม่อย่างสมบูรณ์

ด้วยการผสานรวมโมเดลภาษาขนาดใหญ่ เจม Instant18n ช่วยให้คุณสามารถสร้างการแปลแบบทันที โดยอิงจากบริบทและความหมายของข้อความ แทนที่จะต้องพึ่งพาคีย์การแปลที่กำหนดไว้ล่วงหน้าและการแปลแบบคงที่ เจมจะแปลข้อความแบบไดนามิกโดยใช้พลังของ AI วิธีการนี้มีประโยชน์หลักๆ ดังนี้:

1. **การแปลงให้เข้ากับท้องถิ่นอย่างราบรื่น:** ด้วยเจม Instant18n นักพัฒนาไม่จำเป็นต้องกำหนดและดูแลไฟล์การแปลสำหรับแต่ละภาษาที่รองรับด้ เจมจะสร้างการแปลโดยอัตโนมัติตามข้อความที่ให้มาและภาษาเป้าหมายที่ต้องการ ทำให้กระบวนการแปลงให้เข้ากับท้องถิ่นเป็นไปอย่างง่ายดายและราบรื่น
2. **ความถูกต้องตามบริบท:** AI สามารถรับบริบทที่เพียงพอเพื่อทำความเข้าใจความละเอียดอ่อนของข้อความที่กำลังแปล สามารถคำนึงถึงบริบทโดยรอบ สำนวน และการอ้างอิงทางวัฒนธรรมเพื่อสร้างการแปลที่ถูกต้อง ฟังดูเป็นธรรมชาติ และเหมาะสมกับบริบท
3. **การรองรับภาษาที่กว้างขวาง:** เจม Instant18n ใช้ประโยชน์จากความรู้อันกว้างขวางและความสามารถทางภาษาของ GPT ทำให้สามารถแปลเป็นภาษาต่างๆ ได้มากมาย ตั้งแต่ภาษาทั่วไปอย่างภาษาสเปนและฝรั่งเศส ไปจนถึงภาษาที่ไม่ค่อยพบเห็นหรือภาษาในจินตนาการเช่นภาษาคลิงงอนและภาษาเอลฟ์ เจมสามารถจัดการกับความต้องการในการแปลที่หลากหลายได้
4. **ความยืดหยุ่นและความคิดสร้างสรรค์:** เจมนี้ก้าวไปไกลกว่าการแปลภาษาแบบดั้งเดิมและอนุญาตให้มีตัวเลือกการแปลงให้เข้ากับท้องถิ่นที่สร้างสรรค์และไม่ธรรมดา นักพัฒนาสามารถแปลข้อความเป็นรูปแบบ ภาษาถิ่น หรือแม้แต่ภาษาในจินตนาการต่างๆ เปิดโอกาสใหม่ๆ สำหรับประสบการณ์ผู้ใช้ที่ไม่เหมือนใครและเนื้อหาที่น่าสนใจ
5. **การเพิ่มประสิทธิภาพการทำงาน:** เจม Instant18n รวมกลไกการแคชเพื่อปรับปรุงประสิทธิภาพและลดภาระของการแปลซ้ำๆ ข้อความที่แปลแล้วจะถูกแคชไว้ ทำให้คำขอการแปลเดิมในครั้งต่อไปสามารถให้บริการได้อย่างรวดเร็วโดยไม่จำเป็นต้องเรียก API ซ้ำ

เจม Instant18n แสดงให้เห็นถึงพลังของรูปแบบ “การสร้างเนื้อหาตามบริบท” โดยใช้ AI เพื่อสร้างเนื้อหาที่แปลงให้เข้ากับท้องถิ่นแบบไดนามิก มันแสดงให้เห็นว่า AI สามารถถูกผสานรวมเข้ากับฟังก์ชันหลักของแอปพลิเคชัน Rails อย่างไร เปลี่ยนแปลงวิธีที่นักพัฒนาเข้าถึงการทำให้เป็นสากลและการแปลงให้เข้ากับท้องถิ่น

ด้วยการกำจัดความจำเป็นในการจัดการการแปลด้วยตนเอง และการเปิดใช้งานการแปลแบบทันทีตามบริบท เจม Instant18n ช่วยประหยัดเวลาและความพยายามของนักพัฒนาอย่างมาก ช่วยให้พวกเขาสามารถมุ่งเน้นไปที่การสร้างฟีเจอร์หลักของแอปพลิเคชันในขณะที่มั่นใจได้ว่าการแปลภาษาจะดำเนินไปอย่างราบรื่นและแม่นยำ

ความสำคัญของการทดสอบโดยผู้ใช้และการรับข้อเสนอแนะ

ท้ายที่สุด อย่าลืมคำนึงถึงความสำคัญของการทดสอบโดยผู้ใช้และการรับข้อเสนอแนะ เป็นสิ่งสำคัญที่จะต้องตรวจสอบว่าการสร้างเนื้อหาตามบริบทนั้นตรงกับความคาดหวังของผู้ใช้และสอดคล้องกับเป้าหมายของแอปพลิเคชัน จำเป็นต้องปรับปรุงและพัฒนาเนื้อหาที่สร้างขึ้นอย่างต่อเนื่องโดยอิงจากข้อมูลเชิงลึกของผู้ใช้และการวิเคราะห์ข้อมูล หากคุณกำลังสร้างเนื้อหาแบบไดนามิกในปริมาณมากซึ่งเป็นไปไม่ได้ที่คุณและทีมจะตรวจสอบด้วยตนเองทั้งหมด ให้พิจารณาเพิ่มกลไกการรับข้อเสนอแนะที่อนุญาตให้ผู้ใช้รายงานเนื้อหาที่แปลกหรือไม่ถูกต้อง พร้อมคำอธิบายเหตุผล ข้อเสนอแนะอันมีค่าเหล่านี้สามารถนำไปป้อนให้กับระบบ AI ที่ทำงานอัตโนมัติซึ่งมีหน้าที่ปรับปรุงส่วนประกอบที่สร้างเนื้อหานั้นได้อีกด้วย!

ยูไอเชิงสร้างสรรค์

ในยุคที่ความสนใจ มีค่ามากเช่นนี้ การสร้างความผูกพันกับผู้ใช้อย่างมีประสิทธิภาพจำเป็นต้องมีประสบการณ์การใช้ซอฟต์แวร์ที่ ไม่ เพียง แต่ ราบ รื่น และ เข้าใจ ง่าย แต่ ยัง ต้อง ปรับ แต่ง ให้ เข้า กับ ความ ต้องการ ความ ชอบ และ บริบท ของ แต่ละ บุคคล ส่งผลให้นักออกแบบและนักพัฒนาต้องเผชิญกับความท้าทายในการสร้างส่วนต่อประสานกับผู้ใช้ที่สามารถปรับตัวและตอบสนองความต้องการเฉพาะของผู้ใช้แต่ละคน *ในระดับที่ขยายได้*

ยู ไอ เชิง สร้างสรรค์ (เจน ยู ไอ) เป็นแนวทางที่ปฏิวัติวงการการออกแบบส่วนต่อประสานกับผู้ใช้อย่าง แท้จริง โดย ใช้ พลัง ของ โมเดล ภาษา ขนาด ใหญ่ (แอล แอล เอ็ม) เพื่อ สร้าง ประสบการณ์ ผู้ ใช้ ที่ ปรับ แต่ง เฉพาะ บุคคล และ มี พล วัต แบบ เรี ยล ไทม์ ผม อยาก ให้ แน่ใจ ว่า ได้ ให้ ความ รู้ พื้น ฐาน เกี่ยว กับ เจน ยู ไอ ใน หนังสือ เล่ม นี้ เพราะผมเชื่อว่านี่คือหนึ่งในโอกาสที่เปิดกว้างที่สุดที่มีอยู่ในขณะนี้ในด้านการออกแบบแอปพลิเคชันและเฟรมเวิร์ก ผม เชื่อ มั่นว่าจะมีโครงการเชิงพาณิชย์และโอเพนซอร์สใหม่ๆ เกิดขึ้นนับสิบในช่องทางเฉพาะนี้

ในแก่นแท้แล้ว เจนยูไอผสมผสานหลักการของการสร้างเนื้อหาตามบริบท กับเทคนิคปัญญาประดิษฐ์ขั้นสูงเพื่อสร้างองค์ประกอบส่วนต่อประสานกับผู้ใช้ เช่น ข้อความ รูปภาพ และเลย์เอาต์ แบบไดนามิกบนพื้นฐานของความเข้าใจอย่างลึกซึ้งเกี่ยวกับบริบท ความชอบ และเป้าหมายของผู้ใช้ เจนยูไอช่วยให้นักออกแบบและนักพัฒนาสามารถสร้างอินเทอร์เฟซที่ปรับตัวและพัฒนาตามการโต้ตอบของผู้ใช้ ทำให้สามารถปรับแต่งเฉพาะบุคคลได้ในระดับที่ไม่เคยเป็นไปได้มาก่อน

เจนยูไอแสดงถึงการเปลี่ยนแปลงพื้นฐานในวิธีที่เราเข้าถึงการออกแบบส่วนต่อประสานกับผู้ใช้ แทนที่จะออกแบบสำหรับมวลชน เจนยูไอช่วยให้เราออกแบบสำหรับแต่ละบุคคล เนื้อหาและอินเทอร์เฟซที่ปรับแต่งเฉพาะบุคคลมีศักยภาพในการสร้างประสบการณ์ผู้ใช้ที่สอดคล้องกับผู้ใช้แต่ละคนในระดับที่ลึกซึ้งขึ้น เพิ่มความผูกพัน ความพึงพอใจ และความภักดี

ในฐานะเทคนิคล้ำสมัย การเปลี่ยนไปใช้เจนยูไอเต็มไปด้วยความท้าทายทั้งในเชิงแนวคิดและการปฏิบัติ การผสานปัญญาประดิษฐ์เข้ากับกระบวนการออกแบบ การทำให้แน่ใจว่าอินเทอร์เฟซที่สร้างขึ้นไม่เพียงแต่ปรับแต่งเฉพาะบุคคลแต่ยังใช้งานได้ เข้าถึงได้ และสอดคล้องกับแบรนด์และประสบการณ์ผู้ใช้โดยรวม สิ่งเหล่านี้ล้วนเป็นความท้าทายที่ทำให้เจนยูไอเป็นการไล่ตามสำหรับคนส่วนน้อย ไม่ใช่คนส่วนใหญ่ นอกจากนี้ การมีส่วนร่วมของปัญญาประดิษฐ์ยังก่อให้เกิดคำถามเกี่ยวกับความเป็นส่วนตัวของข้อมูล ความโปร่งใส และอาจรวมถึงผลกระทบทางจริยธรรม

แม้จะมีความท้าทาย ประสบการณ์ที่ปรับแต่งเฉพาะบุคคลในระดับที่ขยายได้มีพลังที่จะเปลี่ยนแปลงวิธีที่เราโต้ตอบกับผลิตภัณฑ์และบริการดิจิทัลอย่างสิ้นเชิง มันเปิดโอกาสในการสร้างอินเทอร์เฟซที่ครอบคลุมและเข้าถึงได้ซึ่งตอบสนองความต้องการที่หลากหลายของผู้ใช้ โดยไม่คำนึงถึงความสามารถ ภูมิหลัง หรือความชอบส่วนตัว

ในบทนี้ เราจะสำรวจแนวคิดของเจนยูไอ โดยพิจารณาลักษณะสำคัญ ประโยชน์หลัก และความท้าทายที่อาจเกิดขึ้น เราเริ่มต้นด้วยการพิจารณารูปแบบพื้นฐานและเข้าถึงได้ที่สุดของเจนยูไอ: การสร้างข้อความสำเนาสำหรับส่วนต่อประสานกับผู้ใช้ที่ออกแบบและพัฒนาแบบดั้งเดิม

การสร้างข้อความสำเนาสำหรับส่วนต่อประสานกับผู้ใช้

องค์ประกอบข้อความที่มีอยู่ในส่วนประกอบอินเทอร์เฟซของแอปพลิเคชันของคุณ เช่น ป้ายกำกับฟอร์ม คำแนะนำเครื่องมือ และข้อความอธิบาย มักถูกเขียนลงไปในเทมเพลตหรือคอมโพเนนต์ยูไอโดยตรง ซึ่งให้ประสบการณ์ที่สม่ำเสมอแต่เป็นแบบทั่วไปสำหรับผู้ใช้ทุกคน โดยใช้รูปแบบการสร้างเนื้อหาตามบริบท คุณสามารถเปลี่ยนองค์ประกอบคงที่เหล่านี้ให้เป็นคอมโพเนนต์ที่มีพลวัต รับรู้บริบท และปรับแต่งเฉพาะบุคคลได้

ฟอร์มที่ปรับแต่งเฉพาะบุคคล

ฟอร์มเป็นส่วนที่พบได้ทั่วไปในแอปพลิเคชันเว็บและมือถือ ทำหน้าที่เป็นวิธีหลักในการรวบรวมข้อมูลจากผู้ใช้ อย่างไรก็ตาม ฟอร์มแบบดั้งเดิมมักนำเสนอประสบการณ์ที่เป็นแบบทั่วไปและไม่เป็นส่วนตัว ด้วยป้ายกำกับและฟิลด์มาตรฐานที่อาจไม่สอดคล้องกับบริบทหรือความต้องการเฉพาะของผู้ใช้เสมอไป ผู้ใช้มีแนวโน้มที่จะกรอกฟอร์มที่รู้สึกว่าปรับแต่งให้เข้ากับความต้องการและความชอบของพวกเขามากขึ้น นำไปสู่อัตราการแปลงและความพึงพอใจของผู้ใช้ที่สูงขึ้น

อย่างไรก็ตาม สิ่งสำคัญคือต้องสร้างความสมดุลระหว่างการปรับแต่งเฉพาะบุคคลและความสม่ำเสมอ แม้ว่าการปรับฟอร์มให้เข้ากับผู้ใช้แต่ละคนจะเป็นประโยชน์ แต่ก็สำคัญที่จะต้องรักษาระดับความคุ้นเคยและการคาดการณ์ได้ ผู้ใช้ควรยังคงสามารถจดจำและนำทางฟอร์มได้ง่าย แม้จะมีองค์ประกอบที่ปรับแต่งเฉพาะบุคคล

นี่คือแนวคิดฟอร์มที่ปรับแต่งเฉพาะบุคคลเพื่อเป็นแรงบันดาลใจ:

การแนะนำฟิลด์ตามบริบท

เจนยูไอสามารถวิเคราะห์การโต้ตอบก่อนหน้า ความชอบ และข้อมูลของผู้ใช้เพื่อให้คำแนะนำฟิลด์ที่ชาญฉลาดในรูปแบบการคาดการณ์ ตัวอย่างเช่น หากผู้ใช้เคยกรอกที่อยู่สำหรับจัดส่งมาก่อน ฟอร์มสามารถกรอกฟิลด์ที่เกี่ยวข้องด้วยข้อมูลที่บันทึกไว้โดยอัตโนมัติ ซึ่งไม่เพียงแต่ประหยัดเวลา แต่ยังแสดงให้เห็นว่าแอปพลิเคชันเข้าใจและจดจำความชอบของผู้ใช้

เดี๋ยวก่อน เทคนิคนี้สามารถทำได้โดยไม่ต้องใช้ AI ไม่ใช่หรือ? แน่นอนว่าทำได้ แต่ความสวยงามของการขับเคลื่อนฟังก์ชันการทำงานแบบนี้ด้วย AI นั้นมีสองประการ: 1) ความง่ายในการนำไปใช้งาน และ 2) ความยืดหยุ่นที่มีเมื่อส่วนติดต่อผู้ใช้ของคุณมีการเปลี่ยนแปลงและพัฒนาไปตามกาลเวลา

มาลองสร้างเซอร์วิสสำหรับระบบจัดการคำสั่งซื้อในทางทฤษฎีของเรากัน ที่จะพยายามกรอกที่อยู่จัดส่งที่ถูกต้องให้กับผู้ใช้โดยอัตโนมัติ

```
class OrderShippingAddressSubscriber
  include Raix::ChatCompletion

  attr_accessor :order

  delegate :customer, to: :order

  DIRECTIVE = "You are a smart order processing assistant. Given the
  customer's order history, guess the most likely shipping address
  for the current order."

  def order_created(order)
    return unless order.pending? && order.shipping_address.blank?

    self.order = order

    transcript.clear
    transcript << { system: DIRECTIVE }
    transcript << { user: "Order History: #{order_history.to_json}" }
    transcript << { user: "Current Order: #{order.to_json}" }
```

```
    response = chat_completion
    apply_predicted_shipping_address(order, response)
  end

  private

  def apply_predicted_shipping_address(order, response)
    # extract the shipping address from the response...
    # ...and assume there's some sort of live update of the address fields
    order.update(shipping_address:)
  end

  def order_history
    customer.orders.successful.limit(100).map do |order|
      {
        date: order.date,
        description: order.description,
        shipping_address: order.shipping_address
      }
    end
  end
end
```

ตัวอย่างนี้เป็นแบบที่ถูกทำให้ง่ายมาก แต่ควรใช้ได้ในกรณีส่วนใหญ่ แนวคิดก็คือการปล่อยให้ AI ทำการคาดเดาเหมือนกับที่มนุษย์จะทำ เพื่อให้เข้าใจสิ่งที่ผมกำลังพูดถึงได้ชัดเจน เรามาพิจารณาข้อมูลตัวอย่างกัน:

```
Order History:
[
 {"date": "2024-01-03", "description": "garden soil mix",
  "shipping_address": "123 Country Lane, Rural Town"},
 {"date": "2024-01-15", "description": "hardcover fiction novels",
  "shipping_address": "456 City Apt, Metroville"},
 {"date": "2024-01-22", "description": "baby diapers", "shipping_address":
  "789 Suburb St, Quietville"},
 {"date": "2024-02-01", "description": "organic vegetables",
  "shipping_address": "123 Country Lane, Rural Town"},
 {"date": "2024-02-17", "description": "mystery thriller book set",
  "shipping_address": "456 City Apt, Metroville"},
 {"date": "2024-02-25", "description": "baby wipes",
  "shipping_address": "789 Suburb St, Quietville"},
```

```
  {"date": "2024-03-05", "description": "flower seeds",
   "shipping_address": "123 Country Lane, Rural Town"},
  {"date": "2024-03-20", "description": "biographies",
   "shipping_address": "456 City Apt, Metroville"},
  {"date": "2024-03-30", "description": "baby formula",
   "shipping_address": "789 Suburb St, Quietville"},
  {"date": "2024-04-12", "description": "lawn fertilizer",
   "shipping_address": "123 Country Lane, Rural Town"},
  {"date": "2024-04-22", "description": "science fiction novels",
   "shipping_address": "456 City Apt, Metroville"},
  {"date": "2024-05-02", "description": "infant toys",
   "shipping_address": "789 Suburb St, Quietville"},
  {"date": "2024-05-14", "description": "outdoor grill",
   "shipping_address": "123 Country Lane, Rural Town"},
  {"date": "2024-05-29", "description": "literary classics",
  "shipping_address": "456 City Apt, Metroville"},
  {"date": "2024-06-11", "description": "baby clothes",
   "shipping_address": "789 Suburb St, Quietville"},
  {"date": "2024-07-01", "description": "watering can",
   "shipping_address": "123 Country Lane, Rural Town"},
  {"date": "2024-07-18", "description": "non-fiction essays",
  "shipping_address": "456 City Apt, Metroville"},
  {"date": "2024-07-28", "description": "baby bath items",
  "shipping_address": "789 Suburb St, Quietville"},
  {"date": "2024-08-09", "description": "herb garden kit",
  "shipping_address": "123 Country Lane, Rural Town"},
  {"date": "2024-08-24", "description": "children's books",
  "shipping_address": "456 City Apt, Metroville"}
]
```

คุณสังเกตเห็นรูปแบบในข้อมูลหรือไม่? ผมรับรองว่าเรื่องนี้ง่ายมากสำหรับ LLM เลย เพื่อพิสูจน์เรื่องนี้ เรามาลองถาม GPT-4 ดูว่าที่อยู่จัดส่งที่น่าจะเป็นไปได้มากที่สุดสำหรับ “เทอร์โมมิเตอร์” คืออะไร

```
From the order history you've provided, it looks like the purchases are
generally clustered into three main types based on the shipping addresses:

123 Country Lane, Rural Town - This address often orders garden and
outdoor-related items like soil mix, vegetables, flower seeds, lawn
fertilizer, an outdoor grill, a watering can, and an herb garden kit.

456 City Apt, Metroville - This address tends to order various types of
books, indicating an interest in reading across genres like fiction,
thrillers, biographies, science fiction, literary classics, non-fiction
essays, and children's books.

789 Suburb St, Quietville - The purchases for this address are consistently
baby-related products, such as diapers, wipes, baby formula, infant toys,
baby clothes, and baby bath items.

Given these patterns, a digital thermometer could potentially be a fit for
any of the addresses based on different needs: outdoor temperature tracking,
a tool for a passionate reader interested in science gadgets, or a health
tool for a baby. However, considering its common use in monitoring health,
especially for babies, it's most likely to be purchased by someone who's
already buying baby care products.

Therefore, the most likely shipping address for a digital thermometer, based
on the purchasing patterns, would be 789 Suburb St, Quietville.
```

ถ้าคุณกำลังคิดว่าการใช้โมเดลที่มีราคาแพงอย่าง GPT-4 สำหรับงานนี้เป็นการทำเกินความจำเป็น คุณคิดถูกแล้ว! ผมได้ลองใช้พรอมต์เดียวกันนี้กับ Mistral 7B Instruct และมันให้ผลลัพธ์ดังต่อไปนี้ด้วยความเร็ว 75 โทเค็นต่อวินาที และด้วยราคาที่ถูกมากเพียง 0.000218 ดอลลาร์สหรัฐ

```
Based on the given order history, it appears that the customer has received
orders for baby-related items (diapers, wipes, formula, clothes, bath items)
at the address "789 Suburb St, Quietville" and garden-related items (soil,
seeds, watering can, herb garden kit) at the address "123 Country Lane, Rural
Town". The customer has also received books (fiction, non-fiction, literary
classics, mystery-thriller, science fiction, biographies, children's books)
at the address "456 City Apt, Metroville".

Given that the new order is for a "thermometer", which doesn't seem to fit
into any of the categories that the customer has previously ordered, it's
hard to accurately predict the shipping address based solely on the order
history. However, given the 50%-50% split between baby-related and
garden-related items, it could somewhat lean towards the Baby-related items
address ("789 Suburb St, Quietville"). But remember, this is an assumption
and cannot be definitively confirmed without more context or information.
```

ค่าใช้จ่ายและทรัพยากรที่ต้องใช้ในเทคนิคนี้คุ้มค่าหรือไม่ที่จะทำให้ประสบการณ์การชำระเงินดูน่าทึ่งขึ้น? สำหรับหลายราย คำตอบคือคุ้มค่าอย่างแน่นอน และจากที่เห็น ต้นทุนการประมวลผล AI มีแนวโน้มที่จะลดลงเรื่อยๆ โดยเฉพาะอย่างยิ่งสำหรับที่กำลังแข่งขันด้านราคากันอย่างดุเดือด

ใช้ Prompt Template และ StructuredIO ร่วมกับ Response Fencing เพื่อเพิ่มประสิทธิภาพการแชทแบบนี้

การจัดลำดับฟิลด์แบบปรับตัว

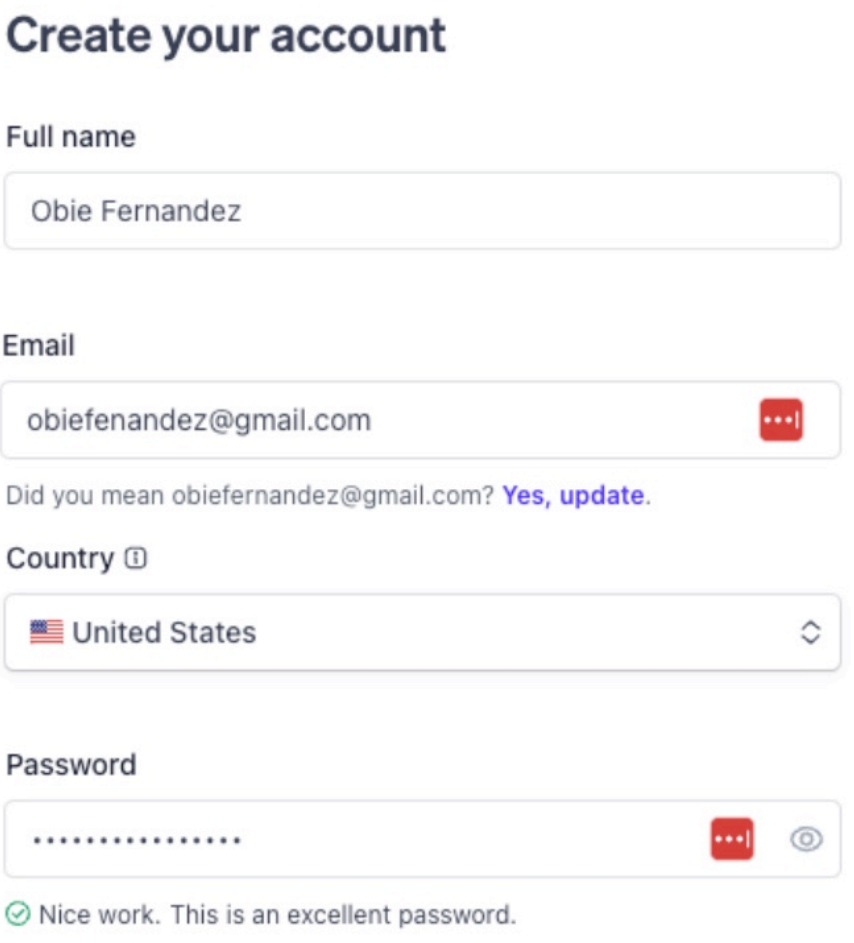
ลำดับการแสดงฟิลด์ในแบบฟอร์มสามารถส่งผลกระทบอย่างมีนัยสำคัญต่อประสบการณ์ของผู้ใช้และอัตราการกรอกข้อมูลให้เสร็จสมบูรณ์ ด้วย GenUI คุณสามารถปรับลำดับฟิลด์แบบไดนามิกตามบริบทของผู้ใช้และความสำคัญของแต่ละฟิลด์ ตัวอย่างเช่น หากผู้ใช้กำลังกรอกแบบฟอร์มลงทะเบียนสำหรับแอพออกกำลังกาย แบบฟอร์มสามารถจัดลำดับความสำคัญของฟิลด์ที่เกี่ยวข้องกับเป้าหมายการออกกำลังกายและความชอบของพวกเขา ทำให้กระบวนการมีความเกี่ยวข้องและน่าสนใจมากขึ้น

ข้อความขนาดเล็กแบบส่วนตัว

ข้อความคำแนะนำ ข้อความแสดงข้อผิดพลาด และข้อความขนาดเล็กอื่นๆ ที่เกี่ยวข้องกับแบบฟอร์มก็สามารถปรับให้เป็นส่วนตัวได้โดยใช้ GenUI แทนที่จะแสดงข้อความแสดงข้อผิดพลาดทั่วไปเช่น “อีเมลไม่ถูกต้อง” คุณสามารถสร้างข้อความที่เป็นประโยชน์และมีบริบทมากขึ้น เช่น “กรุณากรอกอีเมลที่ถูกต้องเพื่อรับการยืนยันคำสั่งซื้อของคุณ” การปรับแต่งเหล่านี้สามารถทำให้ประสบการณ์การใช้แบบฟอร์มเป็นมิตรกับผู้ใช้มากขึ้นและลดความรู้สึกหงุดหงิด

การตรวจสอบความถูกต้องแบบส่วนตัว

ในแนวทางเดียวกับข้อความขนาดเล็กแบบส่วนตัว คุณสามารถใช้ AI เพื่อตรวจสอบแบบฟอร์มในรูปแบบที่ดูเหมือนมีเวทมนตร์ ลองจินตนาการว่าให้ AI ตรวจสอบแบบฟอร์มโปรไฟล์ผู้ใช้ โดยมองหาข้อผิดพลาดที่อาจเกิดขึ้นในระดับ *ความหมาย*



แผนภูมิ 8. คุณสังเกตเห็นการตรวจสอบความถูกต้องเชิงความหมายที่เกิดขึ้นหรือไม่?

การเปิดเผยแบบก้าวหน้า

GenUI สามารถกำหนดอย่างชาญฉลาดว่าฟิลด์ใดในแบบฟอร์มมีความจำเป็นตามบริบทของผู้ใช้และค่อยๆ เปิดเผยฟิลด์เพิ่มเติมตามความต้องการ เทคนิคการเปิดเผยแบบก้าวหน้านี้ช่วยลดภาระการคิดและทำให้กระบวนการกรอกแบบฟอร์มจัดการได้ง่ายขึ้น ตัวอย่างเช่น หากผู้ใช้กำลังสมัครสมาชิกแบบพื้นฐาน แบบฟอร์มสามารถแสดงเฉพาะฟิลด์ที่จำเป็นในตอนแรก และเมื่อผู้ใช้ดำเนินการต่อหรือเลือกตัวเลือกเฉพาะ ฟิลด์ที่เกี่ยวข้องเพิ่มเติมจะถูกแสดงขึ้นมาแบบไดนามิก

ข้อความอธิบายตามบริบท

คำแนะนำเมื่อชี้มักถูกใช้เพื่อให้ข้อมูลเพิ่มเติมหรือคำแนะนำแก่ผู้ใช้เมื่อพวกเขาชี้หรือมีปฏิสัมพันธ์กับองค์ประกอบเฉพาะ ด้วยวิธีการ “การสร้างเนื้อหาตามบริบท” คุณสามารถสร้างคำแนะนำเมื่อชี้ที่ปรับตัวตามบริบทของผู้ใช้และให้ข้อมูลที่เกี่ยวข้อง ตัวอย่างเช่น หากผู้ใช้กำลังสำรวจฟีเจอร์ที่ซับซ้อน คำแนะนำเมื่อชี้สามารถให้เคล็ดลับหรือตัวอย่างที่ปรับให้เข้ากับการโต้ตอบก่อนหน้าหรือระดับทักษะของพวกเขา

ข้อความอธิบาย เช่น คำแนะนำ คำอธิบาย หรือข้อความช่วยเหลือ สามารถถูกสร้างขึ้นแบบไดนามิกตามบริบทของผู้ใช้ แทนที่จะนำเสนอคำอธิบายทั่วไป คุณสามารถใช้ LLMs เพื่อสร้างข้อความที่ปรับแต่งตามความต้องการหรือคำถามเฉพาะของผู้ใช้ ตัวอย่างเช่น หากผู้ใช้กำลังมีปัญหากับขั้นตอนใดขั้นตอนหนึ่งในกระบวนการ ข้อความอธิบายสามารถให้คำแนะนำหรือเคล็ดลับการแก้ไขปัญหาที่เป็นส่วนตัว

ข้อความ ขนาด เล็ก หมาย ถึง ข้อ ความ สั้นๆ ที่ แนะนำ ผู้ ใช้ ผ่าน แอปพลิเคชัน ของ คุณ เช่น ป้าย กำกับ ปุ่ม ข้อความ แสดง ข้อ ผิด พลาด หรือ ข้อความ ยืนยัน โดย การ ประยุกต์ ใช้ วิธี การ สร้าง เนื้อหา ตาม บริบท กับ ข้อความ ขนาด เล็ก คุณสามารถสร้างส่วนติดต่อผู้ใช้แบบปรับตัวที่ตอบสนองต่อการกระทำของผู้ใช้และให้ข้อความที่เกี่ยวข้องและเป็นประโยชน์ ตัวอย่างเช่น หากผู้ใช้กำลังจะดำเนินการที่สำคัญ ข้อความยืนยันสามารถถูกสร้างขึ้นแบบไดนามิกเพื่อให้ข้อความที่ชัดเจนและเป็นส่วนตัว

ข้อความอธิบายและคำแนะนำเมื่อชี้ที่ปรับเป็นส่วนตัวสามารถเพิ่มประสิทธิภาพกระบวนการเริ่มต้นใช้งานสำหรับผู้ใช้ใหม่อย่างมาก โดยการให้คำแนะนำและตัวอย่างตามบริบทเฉพาะ คุณสามารถช่วยให้ผู้ใช้เข้าใจและนำทางแอปพลิเคชันได้อย่างรวดเร็ว ลดการเรียนรู้และเพิ่มการนำไปใช้

องค์ประกอบส่วนติดต่อที่ไดนามิกและตระหนักถึงบริบทยังสามารถทำให้แอปพลิเคชันรู้สึกใช้งานง่ายและน่าสนใจมากขึ้น ผู้ใช้มีแนวโน้มที่จะมีปฏิสัมพันธ์และสำรวจฟีเจอร์มากขึ้นเมื่อข้อความที่มาพร้อมกันถูกปรับแต่งตามความต้องการและความสนใจเฉพาะของพวกเขา

จนถึงตอนนี้เราได้กล่าวถึงแนวคิดในการเพิ่มประสิทธิภาพรูปแบบส่วนต่อประสานผู้ใช้ที่มีอยู่ด้วย AI แต่จะเป็นอย่างไรถ้าเราจะคิดใหม่เกี่ยวกับวิธีการออกแบบและพัฒนาส่วนต่อประสานผู้ใช้ในแนวทางที่แหวกแนวมากขึ้น?

นิยามส่วนต่อประสานผู้ใช้เชิงกำเนิด

ต่างจากการออกแบบส่วนต่อประสานผู้ใช้แบบดั้งเดิมที่นักออกแบบสร้างส่วนต่อประสานแบบตายตัวและคงที่ GenUI ชี้ให้เห็นถึงอนาคตที่ซอฟต์แวร์ของเราจะมีประสบการณ์การใช้งานที่ยืดหยุ่นและปรับเฉพาะบุคคลได้ ซึ่งสามารถพัฒนาและปรับตัวได้แบบเรียลไทม์ ทุกครั้งที่เราใช้ส่วนต่อประสานแบบสนทนาที่ขับเคลื่อนด้วย AI เรากำลังปล่อยให้ AI ปรับตัวเข้ากับความต้องการเฉพาะของผู้ใช้ GenUI ก้าวไปอีกขั้นด้วยการนำระดับการปรับตัวนี้ไปใช้กับส่วนต่อประสานแบบภาพของซอฟต์แวร์

เหตุผลที่เราสามารถทดลองแนวคิด GenUI ได้ในปัจจุบันเพราะโมเดลภาษาขนาดใหญ่ มีความเข้าใจเกี่ยวกับการเขียนโปรแกรมและมีความรู้พื้นฐานเกี่ยวกับ
คำถามคือโมเดลภาษาขนาดใหญ่สามารถใช้สร้างองค์ประกอบของส่วนต่อประสานผู้ใช้ เช่น ข้อความ รูปภาพ เลย์เอาต์ และแม้แต่ส่วนต่อประสานทั้งหมดที่ปรับแต่งสำหรับผู้ใช้แต่ละคนได้หรือไม่ โมเดลสามารถรับคำสั่งให้พิจารณาปัจจัยต่างๆ เช่น การโต้ตอบที่ผ่านมาของผู้ใช้ ความชอบที่ระบุไว้ ข้อมูลประชากร และบริบทการใช้งานปัจจุบัน เพื่อสร้างส่วนต่อประสานที่ปรับเฉพาะบุคคลและเกี่ยวข้องอย่างมาก

GenUI แตกต่างจากการออกแบบส่วนต่อประสานผู้ใช้แบบดั้งเดิมในหลายด้านที่สำคัญ:

1. **แบบไดนามิกและปรับตัวได้:** การออกแบบส่วนต่อประสานผู้ใช้แบบดั้งเดิมเกี่ยวข้องกับการสร้างส่วนต่อประสานแบบตายตัวและคงที่ที่เหมือนกันสำหรับผู้ใช้ทุกคน ในทางตรงกันข้าม GenUI ช่วยให้ส่วนต่อประสานสามารถปรับตัวและเปลี่ยนแปลงแบบไดนามิกตามความต้องการและบริบทของผู้ใช้ นั่นหมายความว่าแอปพลิเคชันเดียวกันสามารถแสดงส่วนต่อประสานที่แตกต่างกันสำหรับผู้ใช้ที่ต่างกัน หรือแม้แต่สำหรับผู้ใช้คนเดียวกันในสถานการณ์ที่ต่างกัน
2. **การปรับแต่งเฉพาะบุคคลในระดับใหญ่:** ด้วยการออกแบบแบบดั้งเดิม การสร้างประสบการณ์ที่ปรับแต่งสำหรับผู้ใช้แต่ละคนมักไม่สามารถทำได้ในทางปฏิบัติเนื่องจากต้องใช้เวลาและทรัพยากรมาก ในทางกลับกัน GenUI ช่วยให้สามารถปรับแต่งเฉพาะบุคคลในระดับใหญ่ได้ ด้วยการใช้ AI นักออกแบบสามารถสร้างส่วนต่อประสานที่ปรับตัวโดยอัตโนมัติตามความต้องการและความชอบเฉพาะของผู้ใช้แต่ละคน โดยไม่ต้องออกแบบและพัฒนาส่วนต่อประสานแยกสำหรับแต่ละกลุ่มผู้ใช้ด้วยตนเอง
3. **มุ่งเน้นผลลัพธ์:** การออกแบบส่วนต่อประสานผู้ใช้แบบดั้งเดิมมักมุ่งเน้นที่การสร้างส่วนต่อประสานที่สวยงามและใช้งานได้ แม้ว่าแง่มุมเหล่านี้ยังคงสำคัญใน GenUI แต่จุดเน้นหลักเปลี่ยนไปสู่การบรรลุผลลัพธ์ที่ผู้ใช้ต้องการ GenUI มุ่งสร้างส่วนต่อประสานที่เหมาะสมที่สุดสำหรับเป้าหมายและงานเฉพาะของผู้ใช้แต่ละคน โดยให้ความสำคัญกับความสามารถในการใช้งานและประสิทธิผลมากกว่าการพิจารณาด้านความสวยงามเพียงอย่างเดียว

4. **การเรียนรู้และพัฒนาอย่างต่อเนื่อง:** ระบบ GenUI สามารถเรียนรู้และพัฒนาอย่างต่อเนื่องตามเวลาบนพื้นฐานของการโต้ตอบและข้อเสนอแนะจากผู้ใช้ เมื่อผู้ใช้มีส่วนร่วมกับส่วนต่อประสานที่สร้างขึ้น โมเดล AI สามารถรวบรวมข้อมูลเกี่ยวกับพฤติกรรม ความชอบ และผลลัพธ์ของผู้ใช้ โดยใช้ข้อมูลนี้เพื่อปรับปรุงและเพิ่มประสิทธิภาพการสร้างส่วนต่อประสานในอนาคต กระบวนการเรียนรู้แบบวนซ้ำนี้ช่วยให้ระบบ GenUI มีประสิทธิผลในการตอบสนองความต้องการของผู้ใช้เพิ่มขึ้นเรื่อยๆ ตามกาลเวลา

สิ่งสำคัญที่ต้องทราบคือ GenUI ไม่เหมือนกับเครื่องมือออกแบบที่ช่วยด้วย AI เช่น เครื่องมือที่ให้คำแนะนำหรือทำงานบางอย่างโดยอัตโนมัติ แม้ว่าเครื่องมือเหล่านี้จะช่วยทำให้กระบวนการออกแบบราบรื่นขึ้น แต่ก็ยังคงต้องพึ่งพานักออกแบบในการตัดสินใจขั้นสุดท้ายและสร้างส่วนต่อประสานแบบคงที่ ในทางกลับกัน GenUI เกี่ยวข้องกับระบบ AI ที่มีบทบาทเชิงรุกมากขึ้นในการสร้างและปรับส่วนต่อประสานตามข้อมูลและบริบทของผู้ใช้

GenUI แสดงถึงการเปลี่ยนแปลงที่สำคัญในวิธีการที่เราเข้าถึงการออกแบบส่วนต่อประสานผู้ใช้ โดยเปลี่ยนจากโซลูชันแบบขนาดเดียวใช้ได้กับทุกคนไปสู่ประสบการณ์ที่ปรับแต่งเฉพาะบุคคลและปรับตัวได้สูง ด้วยการใช้พลังของ AI, GenUI มีศักยภาพที่จะปฏิวัติวิธีที่เราโต้ตอบกับผลิตภัณฑ์และบริการดิจิทัล สร้างส่วนต่อประสานที่ใช้งานง่าย น่าสนใจ และมีประสิทธิผลมากขึ้นสำหรับผู้ใช้แต่ละคน

ตัวอย่าง

เพื่อแสดงให้เห็นแนวคิดของ GenUI เรามาพิจารณาแอปพลิเคชันออกกำลังกายสมมติที่เรียกว่า “FitAI” แอปนี้มีเป้าหมายที่จะให้แผนการออกกำลังกายและคำแนะนำด้านโภชนาการที่ปรับเฉพาะบุคคลแก่ผู้ใช้ตามเป้าหมาย ระดับความฟิต และความชอบของแต่ละคน

ในแนวทางการออกแบบส่วนต่อประสานผู้ใช้แบบดั้งเดิม FitAI อาจมีชุดหน้าจอและองค์ประกอบที่ตายตัวเหมือนกันสำหรับผู้ใช้ทุกคน อย่างไรก็ตาม ด้วย GenUI ส่วนต่อประสานของแอปสามารถปรับตัวแบบไดนามิกตามความต้องการและบริบทเฉพาะของผู้ใช้แต่ละคน

แนวทางนี้อาจเป็นการจินตนาการที่ไกลเกินไปสำหรับการนำไปใช้ในปี 2024 และอาจไม่คุ้มค่ากับการลงทุนเพียงพอ แต่ก็เป็นไปได้

นี่คือวิธีที่อาจทำงาน:

1. **การเริ่มต้นใช้งาน:**

- แทนที่จะใช้แบบสอบถามมาตรฐาน FitAI ใช้ AI แบบสนทนา เพื่อรวบรวมข้อมูลเกี่ยวกับเป้าหมาย ระดับความฟิตปัจจุบัน และความชอบของผู้ใช้
- จากการโต้ตอบเริ่มต้นนี้ AI สร้างเลย์เอาต์แดชบอร์ดที่ปรับเฉพาะบุคคล โดยเน้นฟีเจอร์และข้อมูลที่เกี่ยวข้องมากที่สุดกับเป้าหมายของผู้ใช้
- เทคโนโลยี AI ในปัจจุบันอาจมีการเลือกองค์ประกอบหน้าจอที่มีอยู่มาใช้ในการจัดทำแดชบอร์ดที่ปรับเฉพาะบุคคล
- เทคโนโลยี AI ในอนาคตอาจรับบทบาทเป็นนักออกแบบส่วนต่อประสานผู้ใช้ที่มีประสบการณ์และสร้างแดชบอร์ด_ตั้งแต่เริ่มต้น_

2. **แผนการออกกำลังกาย:**

- ส่วนต่อประสานของแผนการออกกำลังกายถูกปรับแต่งโดย AI ให้เข้ากับระดับประสบการณ์และอุปกรณ์ที่มีของผู้ใช้โดยเฉพาะ
- สำหรับผู้เริ่มต้นที่ไม่มีอุปกรณ์ อาจแสดงท่าออกกำลังกายด้วยน้ำหนักตัวแบบง่ายๆ พร้อมคำแนะนำและวิดีโอที่ละเอียด
- สำหรับผู้ใช้ขั้นสูงที่สามารถเข้าถึงยิมได้ อาจแสดงโปรแกรมการออกกำลังกายที่ซับซ้อนมากขึ้นโดยมีเนื้อหาคำอธิบายที่น้อยลง
- เนื้อหาของแผนการออกกำลังกายไม่ได้เป็นเพียงการกรองจากชุดข้อมูลใหญ่ แต่สามารถสร้างขึ้น *แบบทันที* โดยอิงจากฐานความรู้ที่ถูกสืบค้นด้วยบริบทที่รวมทุกสิ่งที่รู้เกี่ยวกับผู้ใช้

3. **การติดตามความคืบหน้า:**

- ส่วนต่อประสานการติดตามความคืบหน้าพัฒนาตามเป้าหมายและรูปแบบการมีส่วนร่วมของผู้ใช้
- หากผู้ใช้มุ่งเน้นที่การลดน้ำหนักเป็นหลัก ส่วนต่อประสานอาจแสดงกราฟแนวโน้มน้ำหนักและสถิติการเผาผลาญแคลอรี่อย่างเด่นชัด
- สำหรับผู้ใช้ที่กำลังสร้างกล้ามเนื้อ อาจเน้นแสดงการเพิ่มขึ้นของความแข็งแรงและการเปลี่ยนแปลงองค์ประกอบของร่างกาย
- AI สามารถปรับส่วนนี้ของแอปพลิเคชันตามความก้าวหน้าจริงของผู้ใช้ หากความก้าวหน้าหยุดชะงักในช่วงระยะเวลาหนึ่ง แอปสามารถปรับเปลี่ยนเป็นโหมดที่พยายามชักจูงให้ผู้ใช้เปิดเผยสาเหตุของการถดถอย เพื่อหาทางแก้ไข

4. **คำแนะนำด้านโภชนาการ:**

- ส่วนโภชนาการปรับตัวตามความชอบและข้อจำกัดด้านอาหารของผู้ใช้
- สำหรับผู้ใช้ที่เป็นมังสวิรัติ อาจแสดงคำแนะนำอาหารและแหล่งโปรตีนจากพืช
- สำหรับผู้ใช้ที่แพ้กลูเตน จะกรองอาหารที่มีกลูเตนออกจากคำแนะนำโดยอัตโนมัติ
- อีกครั้ง เนื้อหาไม่ได้ถูกดึงมาจากชุดข้อมูลอาหารขนาดใหญ่ที่ใช้กับผู้ใช้ทุกคน แต่ถูกสังเคราะห์จากฐานความรู้ที่มีข้อมูลที่สามารถปรับเปลี่ยนได้ตามสถานการณ์และข้อจำกัดเฉพาะของผู้ใช้
- ตัวอย่างเช่น สูตรอาหารถูกสร้างขึ้นโดยมีข้อกำหนดส่วนผสมที่ตรงกับความต้องการแคลอรี่ที่เปลี่ยนแปลงอยู่ตลอดเวลาของผู้ใช้ตามระดับความฟิตและสถิติร่างกายที่พัฒนาไป

5. **องค์ประกอบการสร้างแรงจูงใจ:**

- เนื้อหาและการแจ้งเตือนเพื่อสร้างแรงจูงใจของแอปถูกปรับให้เป็นส่วนตัวตามประเภทบุคลิกภาพของผู้ใช้และการตอบสนองต่อกลยุทธ์การ
- ผู้ใช้บางคนอาจได้รับข้อความให้กำลังใจ ในขณะที่คนอื่นๆ อาจได้รับข้อเสนอแนะที่อิงกับข้อมูลมากกว่า

ในตัวอย่างนี้ GenUI ช่วยให้ FitAI สร้างประสบการณ์ที่ปรับแต่งสูงสำหรับผู้ใช้แต่ละคน ซึ่งอาจเพิ่มการมีส่วนร่วม ความพึงพอใจ และโอกาสในการบรรลุเป้าหมายด้านฟิตเนส องค์ประกอบส่วนต่อประสาน เนื้อหา และแม้แต่ “บุคลิกภาพ” ของแอปปรับตัวเพื่อตอบสนองความต้องการและความชอบของผู้ใช้แต่ละคนได้ดีที่สุด

การเปลี่ยนแปลงสู่การออกแบบที่มุ่งเน้นผลลัพธ์

GenUI แสดงถึงการเปลี่ยนแปลงพื้นฐานในแนวทางการออกแบบส่วนต่อประสานกับผู้ใช้! จากการมุ่งเน้นการสร้างองค์ประกอบส่วนต่อประสานเฉพาะไปสู่แนวทางที่ครอบคลุมและมุ่งเน้นผลลัพธ์มากขึ้น การเปลี่ยนแปลงนี้มีนัยสำคัญหลายประการ:

1. **มุ่งเน้นเป้าหมายของผู้ใช้:**

- นักออกแบบจะต้องคิดให้ลึกซึ้งยิ่งขึ้นเกี่ยวกับเป้าหมายและผลลัพธ์ที่ต้องการของผู้ใช้ แทนที่จะเป็นองค์ประกอบส่วนต่อประสานเฉพาะ
- เน้นการสร้างระบบที่สามารถสร้างส่วนต่อประสานที่ช่วยให้ผู้ใช้บรรลุวัตถุประสงค์ได้อย่างมีประสิทธิภาพและประสิทธิผล
- เฟรมเวิร์ก UI ใหม่ๆ จะเกิดขึ้นเพื่อให้นักออกแบบที่ใช้ AI มีเครื่องมือที่จำเป็นในการสร้างประสบการณ์ผู้ใช้ *แบบทันที* และ *จากศูนย์* แทนที่จะอิงจากข้อกำหนดหน้าจอที่กำหนดไว้ล่วงหน้า

2. **การเปลี่ยนแปลงบทบาทของนักออกแบบ:**

- นักออกแบบจะเปลี่ยนจากการสร้างเลย์เอาต์ตายตัวไปสู่การกำหนดกฎ ข้อจำกัด และแนวทางให้ระบบ AI ปฏิบัติตามเมื่อสร้างส่วนต่อประสาน
- พวกเขาจะต้องพัฒนาทักษะในด้านต่างๆ เช่น การวิเคราะห์ข้อมูล การออกแบบคำสั่งพร้อมต์ AI และการคิดเชิงระบบ เพื่อให้สามารถแนะนำระบบ GenUI ได้อย่างมีประสิทธิภาพ

3. **ความสำคัญของการวิจัยผู้ใช้:**

- การวิจัยผู้ใช้ยิ่งมีความสำคัญมากขึ้นในบริบทของ GenUI เนื่องจากนักออกแบบต้องเข้าใจไม่เพียงแค่ความชอบของผู้ใช้ แต่รวมถึงวิธีที่ความชอบและความต้องการเหล่านี้เปลี่ยนแปลงในบริบทต่างๆ
- การทดสอบผู้ใช้อย่างต่อเนื่องและการตอบรับจะเป็นสิ่งสำคัญในการปรับปรุงและพัฒนาความสามารถของ AI ในการสร้างส่วนต่อประสานที่มีประสิทธิภาพ

4. **การออกแบบสำหรับความหลากหลาย:**

- แทนที่จะสร้างส่วนต่อประสาน “ที่สมบูรณ์แบบ” เพียงแบบเดียว นักออกแบบจะต้องพิจารณาความเป็นไปได้หลายรูปแบบและทำให้แน่ใจว่าระบบสามารถสร้างส่วนต่อประสานที่เหมาะสมสำหรับความต้องการของผู้ใช้ที่หลากหลาย
- รวมถึงการออกแบบสำหรับกรณีพิเศษและการทำให้แน่ใจว่าส่วนต่อประสานที่สร้างขึ้นยังคงความสามารถในการใช้งานและการเข้าถึงได้ในการกำหนดค่าต่างๆ
- การสร้างความแตกต่างของผลิตภัณฑ์มีมิติใหม่ที่เกี่ยวข้องกับมุมมองที่แตกต่างเกี่ยวกับจิตวิทยาผู้ใช้ และ การใช้ประโยชน์จากชุดข้อมูลและฐานความรู้ที่ไม่มีใครเหมือนซึ่งคู่แข่งไม่สามารถเข้าถึงได้

ความท้าทายและข้อพิจารณา

ในขณะที่ GenUI นำเสนอความเป็นไปได้ที่น่าตื่นเต้น แต่ก็มีความท้าทายและข้อพิจารณาหลายประการ:

1. **ข้อจำกัดทางเทคนิค:**

- เทคโนโลยี AI ในปัจจุบัน แม้จะก้าวหน้า แต่ยังมีข้อจำกัดในการทำความเข้าใจความตั้งใจที่ซับซ้อนของผู้ใช้และการสร้างส่วนต่อประสานที่รับรู้บริบทได้อย่างแท้จริง
- ปัญหาด้านประสิทธิภาพที่เกี่ยวข้องกับการสร้างองค์ประกอบส่วนต่อประสานแบบเรียลไทม์ โดยเฉพาะบนอุปกรณ์ที่มีประสิทธิภาพต่ำ

2. **ข้อกำหนดด้านข้อมูล:**

    - ขึ้นอยู่กับกรณีการใช้งาน ระบบ GenUI ที่มีประสิทธิภาพอาจต้องการข้อมูลผู้ใช้จำนวนมากเพื่อสร้างส่วนต่อประสานที่ปรับแต่งเฉพาะบุคคล
    - ความท้าทายในการได้มาซึ่งข้อมูลผู้ใช้อย่างมีจริยธรรมก่อให้เกิดความกังวลเกี่ยวกับความเป็นส่วนตัวและความปลอดภัยของข้อมูล รวมถึงอคติที่อาจเกิดขึ้นในข้อมูลที่ใช้ฝึกฝนโมเดล GenUI

3. **ความสามารถในการใช้งานและความสม่ำเสมอ:**

    - อย่างน้อยจนกว่าการใช้งานจะแพร่หลาย แอปพลิเคชันที่มีส่วนต่อประสานที่เปลี่ยนแปลงตลอดเวลาอาจนำไปสู่ปัญหาด้านการใช้งาน เนื่องจากผู้ใช้อาจมีปัญหาในการค้นหาองค์ประกอบที่คุ้นเคยหรือการนำทางอย่างมีประสิทธิภาพ
    - การสร้างความสมดุลระหว่างการปรับแต่งส่วนบุคคลและการรักษาส่วนต่อประสานที่สม่ำเสมอและเรียนรู้ได้จะมีความสำคัญอย่างยิ่ง

4. **การพึ่งพา AI มากเกินไป:**

    - มีความเสี่ยงในการมอบการตัดสินใจด้านการออกแบบให้กับระบบ AI มากเกินไป ซึ่งอาจนำไปสู่ตัวเลือกส่วนต่อประสานที่ไม่น่าสนใจ มีปัญหา หรือใช้งานไม่ได้
    - การกำกับดูแลโดยมนุษย์และความสามารถในการแทนที่การออกแบบที่สร้างโดย AI จะยังคงมีความสำคัญในอนาคตอันใกล้

5. **ข้อกังวลด้านการเข้าถึง:**

    - การรับรองว่าส่วนต่อประสานที่สร้างขึ้นแบบไดนามิกยังคงสามารถเข้าถึงได้สำหรับผู้ใช้ที่มีความบกพร่องนำเสนอความท้าทายใหม่ทั้งหมด ซึ่งน่ากังวลเมื่อพิจารณาถึงระดับการปฏิบัติตามมาตรฐานการเข้าถึงที่ต่ำในระบบทั่วไป
    - ในทางกลับกัน นักออกแบบ AI อาจถูกพัฒนาขึ้นโดยมีความใส่ใจด้านการเข้าถึง_ในตัว_ และมีความสามารถในการสร้างส่วนต่อประสานที่เข้าถึงได้แบบทันทีเช่นเดียวกับที่สร้าง UI สำหรับผู้ใช้ทั่วไป
    - ไม่ว่าจะอย่างไร ระบบ GenUI ควรได้รับการออกแบบด้วยแนวทางการเข้าถึงที่แข็งแกร่งและกระบวนการทดสอบ

6. **ความเชื่อมั่นของผู้ใช้และความโปร่งใส:**

    - ผู้ใช้อาจรู้สึกไม่สบายใจกับส่วนต่อประสานที่ดูเหมือนจะ “รู้มากเกินไป” เกี่ยวกับพวกเขาหรือเปลี่ยนแปลงในแบบที่พวกเขาไม่เข้าใจ
    - การให้ความโปร่งใสเกี่ยวกับวิธีการและเหตุผลในการปรับแต่งส่วนต่อประสานจะมีความสำคัญในการสร้างความเชื่อมั่นของผู้ใช้

มุมมองและโอกาสในอนาคต

อนาคตของส่วนต่อประสานผู้ใช้เชิงสร้างสรรค์ (GenUI) มีศักยภาพมหาศาลในการปฏิวัติวิธีที่เราโต้ตอบกับผลิตภัณฑ์และบริการดิจิทัล เมื่อเทคโนโลยีนี้พัฒนาต่อไป เราสามารถคาดการณ์การเปลี่ยนแปลงครั้งใหญ่ในวิธีการออกแบบ การนำไปใช้ และการสัมผัสประสบการณ์ของส่วนต่อประสานผู้ใช้ ผมคิดว่า GenUI คือปรากฏการณ์ที่จะผลักดันซอฟต์แวร์ของเราเข้าสู่ขอบเขตที่ปัจจุบันถือว่าเป็นนิยายวิทยาศาสตร์

หนึ่งในความคาดหวังที่น่าตื่นเต้นที่สุดของ GenUI คือศักยภาพในการเพิ่มการเข้าถึงในระดับที่กว้างขวางซึ่งก้าวไปไกลกว่าการเพียงแค่ทำให้แน่ใจว่าผู้ที่มีความบกพร่องร้ายแรงไม่ถูกกีดกันออกจากการใช้ซอฟต์แวร์ของคุณอย่างสิ้นเชิง ด้วยการปรับส่วนต่อประสานโดยอัตโนมัติตามความต้องการของผู้ใช้แต่ละคน GenUI สามารถทำให้ประสบการณ์ดิจิทัลครอบคลุมมากกว่าที่เคยเป็นมา ลองจินตนาการถึงส่วนต่อประสานที่ปรับขนาดตัวอักษรให้ใหญ่ขึ้นสำหรับผู้ใช้ที่อายุน้อยหรือมีปัญหาด้านการมองเห็น หรือรูปแบบที่เรียบง่ายสำหรับผู้ที่มีความบกพร่องทางการรับรู้ ทั้งหมดนี้โดยไม่ต้องการการตั้งค่าด้วยตนเองหรือเวอร์ชัน “ที่เข้าถึงได้” แยกต่างหากของแอปพลิเคชัน

ความสามารถในการปรับแต่งส่วนบุคคลของ GenUI มีแนวโน้มที่จะขับเคลื่อนการมีส่วนร่วมของผู้ใช้ ความพึงพอใจ และความภักดีในผลิตภัณฑ์ดิจิทัลที่หลากหลาย เมื่อส่วนต่อประสานปรับตัวเข้ากับความชอบและพฤติกรรมของแต่ละบุคคลมากขึ้น ผู้ใช้จะพบว่าประสบการณ์ดิจิทัลมีความเป็นธรรมชาติและสนุกสนานมากขึ้น ซึ่งอาจนำไปสู่การโต้ตอบกับเทคโนโลยีที่ลึกซึ้งและมีความหมายมากขึ้น

GenUI ยังมีศักยภาพในการเปลี่ยนแปลงกระบวนการเริ่มต้นใช้งานสำหรับผู้ใช้ใหม่ ด้วยการสร้างประสบการณ์ผู้ใช้ครั้งแรกที่เป็นธรรมชาติและปรับแต่งเฉพาะบุคคล ซึ่งปรับตัวอย่างรวดเร็วตามระดับความเชี่ยวชาญของผู้ใช้แต่ละคน GenUI สามารถลดเส้นโค้งการเรียนรู้ที่เกี่ยวข้องกับแอปพลิเคชันใหม่ได้อย่างมีนัยสำคัญ สิ่งนี้อาจนำไปสู่อัตราการยอมรับที่เร็วขึ้นและเพิ่มความมั่นใจของผู้ใช้ในการสำรวจคุณสมบัติและฟังก์ชันการทำงานใหม่ๆ

อีก ความ เป็น ไป ได้ที่ น่า ตื่น เต้น คือ ความ สามารถ ของ GenUI ใน การ รักษา ประสบการณ์ ผู้ ใช้ ที่ สม่ำเสมอ ใน อุปกรณ์ และ แพลตฟอร์ม ที่ แตก ต่าง กัน ใน ขณะ ที่ ยัง คง ปรับ ให้ เหมาะ สม กับ บริบท การ ใช้ งาน เฉพาะ สิ่ง นี้ สามารถ แก้ ปัญหา ที่ มี มา ยาวนานในการให้ประสบการณ์ที่สอดคล้องกันในภูมิทัศน์ของอุปกรณ์ที่แตกแยกมากขึ้นเรื่อยๆ ตั้งแต่สมาร์ทโฟน และแท็บเล็ตไปจนถึงคอมพิวเตอร์เดสก์ท็อป และเทคโนโลยีที่กำลังเกิดขึ้นใหม่เช่นแว่นตาความเป็นจริงเสริม

ธรรมชาติที่ขับเคลื่อนด้วยข้อมูลของ GenUI เปิดโอกาสสำหรับการทดลองซ้ำและการปรับปรุงการออกแบบ UI อย่างรวดเร็วด้วยการรวบรวมข้อมูลแบบเรียลไทม์เกี่ยวกับวิธีที่ผู้ใช้โต้ตอบกับส่วนต่อประสานที่สร้างขึ้น นักออกแบบและนักพัฒนาสามารถได้รับข้อมูลเชิงลึกที่ไม่เคยมีมาก่อนเกี่ยวกับพฤติกรรมและความชอบของผู้ใช้ วงจรการตอบกลับนี้อาจนำไปสู่การปรับปรุงการออกแบบ UI อย่างต่อเนื่อง ที่ขับเคลื่อนโดยรูปแบบการใช้งานจริงมากกว่าสมมติฐานหรือการทดสอบผู้ใช้ที่จำกัด

เพื่อเตรียมพร้อมสำหรับการเปลี่ยนแปลงนี้ นักออกแบบจะต้องพัฒนาทักษะและทัศนคติของตน จุดเน้นจะเปลี่ยนจากการสร้างเลย์เอาต์ที่ตายตัวไปสู่การพัฒนาระบบการออกแบบและแนวทางที่ครอบคลุมซึ่งสามารถให้ข้อมูลแก่การสร้างส่วนต่อประสานที่ขับเคลื่อนด้วย AI นักออกแบบจะต้องพัฒนาความเข้าใจอย่างลึกซึ้งในการวิเคราะห์ข้อมูล เทคโนโลยี AI และการคิดเชิงระบบเพื่อแนะนำระบบ GenUI อย่างมีประสิทธิภาพ

ยิ่งไปกว่านั้น เมื่อ GenUI ทำให้เส้นแบ่งระหว่างการออกแบบและเทคโนโลยีเลือนราง นักออกแบบจะต้องร่วมมือกับนักพัฒนาและนักวิทยาศาสตร์ข้อมูลอย่างใกล้ชิดมากขึ้น แนวทางสหวิทยาการนี้จะมีความสำคัญในการสร้างระบบ GenUI ที่ไม่เพียงแต่ดึงดูดทางสายตาและเป็นมิตรกับผู้ใช้เท่านั้น แต่ยังมีความแข็งแกร่งทางเทคนิคและถูกต้องตามหลักจริยธรรมด้วย

นัยเชิงจริยธรรมของจีนยูไอจะกลายเป็นประเด็นสำคัญมากขึ้นเมื่อเทคโนโลยีมีความก้าวหน้า นักออกแบบจะมีบทบาทสำคัญในการพัฒนากรอบการทำงานสำหรับการใช้ปัญญาประดิษฐ์อย่างรับผิดชอบในการออกแบบส่วนต่อประสาน โดยต้องมั่นใจว่าการปรับแต่งเฉพาะบุคคลนั้นช่วยยกระดับประสบการณ์ผู้ใช้โดยไม่ละเมิดความเป็นส่วนตัวหรือบิดเบือนพฤติกรรมผู้ใช้ในทางที่ผิดจริยธรรม

เมื่อมองไปยังอนาคต จีนยูไอนำเสนอทั้งโอกาสที่น่าตื่นเต้นและความท้าทายที่สำคัญ มันมีศักยภาพในการสร้างประสบการณ์ดิจิทัลที่ใช้งานง่าย มีประสิทธิภาพ และน่าพึงพอใจสำหรับผู้ใช้ทั่วโลก แม้ว่านักออกแบบจะต้องปรับตัวและเรียนรู้ทักษะใหม่ๆ แต่ก็เป็นโอกาสที่ไม่เคยมีมาก่อนในการกำหนดอนาคตของปฏิสัมพันธ์ระหว่างมนุษย์กับคอมพิวเตอร์ในแง่มุมที่ลึกซึ้งและมีความหมาย การเดินทางไปสู่ระบบจีนยูไอที่สมบูรณ์แบบจะเต็มไปด้วยความซับซ้อนอย่างแน่นอน แต่ผลลัพธ์ที่อาจเกิดขึ้นในแง่ของการพัฒนาประสบการณ์ผู้ใช้และการเข้าถึงดิจิทัลทำให้มันเป็นอนาคตที่คุ้มค่าแก่การมุ่งมั่น

การจัดการลำดับงานอัจฉริยะ

ในวงการของการพัฒนาแอปพลิเคชัน ลำดับ งาน มี บทบาท สำคัญ ใน การ กำหนด โครงสร้าง และ การ ดำเนิน การ ของ งาน กระบวนการ และการโต้ตอบกับผู้ใช้ เมื่อแอปพลิเคชันมีความซับซ้อนมากขึ้นและความคาดหวังของผู้ใช้เพิ่มสูงขึ้น ความจำเป็นในการจัดการลำดับงานที่ชาญฉลาดและปรับตัวได้ก็ยิ่งเด่นชัดขึ้น

แนวทาง “การจัดการลำดับงานอัจฉริยะ” มุ่งเน้นการใช้ประโยชน์จากองค์ประกอบด้าน AI เพื่อจัดการและปรับปรุงลำดับงานที่ซับซ้อนภายในแอปพลิเคชันแบบไดนามิก เป้าหมายคือการสร้างแอปพลิเคชันที่มีประสิทธิภาพมากขึ้น ตอบสนองได้ดีขึ้น และปรับตัวได้ตามข้อมูลและบริบทแบบเรียลไทม์

ในบทนี้ เราจะสำรวจหลักการและรูปแบบสำคัญที่เป็นรากฐานของแนวทางการจัดการลำดับงานอัจฉริยะ เราจะพิจารณาว่า AI สามารถถูกนำมาใช้ในการจัดเส้นทางงานอย่างชาญฉลาด ทำการตัดสินใจอัตโนมัติ และปรับลำดับงานแบบไดนามิกตามปัจจัยต่างๆ เช่น พฤติกรรมผู้ใช้ ประสิทธิภาพของระบบ และกฎทางธุรกิจ ผ่านตัวอย่างที่ใช้งานได้จริงและสถานการณ์จริง เราจะแสดงให้เห็นศักยภาพในการเปลี่ยนแปลงของ AI ในการปรับปรุงและเพิ่มประสิทธิภาพลำดับงานของแอปพลิเคชัน

ไม่ว่าคุณจะกำลังสร้างแอปพลิเคชันสำหรับองค์กรที่มีกระบวนการทางธุรกิจที่ซับซ้อน หรือแอปพลิเคชันสำหรับผู้บริโภคที่มีเส้นทางการใช้งานแบบไดนามิก รูปแบบและเทคนิคที่กล่าวถึงในบทนี้จะให้ความรู้และเครื่องมือที่จำเป็นในการสร้างลำดับงานที่ชาญฉลาดและมีประสิทธิภาพ ซึ่งจะช่วยยกระดับประสบการณ์ผู้ใช้โดยรวมและสร้างคุณค่าทางธุรกิจ

ความต้องการทางธุรกิจ

แนวทางแบบดั้งเดิมในการจัดการลำดับงานมักพึ่งพากฎที่กำหนดไว้ล่วงหน้าและต้นไม้การตัดสินใจแบบคงที่ ซึ่งอาจมีความแข็งตัว ไม่ยืดหยุ่น และไม่สามารถรับมือกับธรรมชาติที่เปลี่ยนแปลงของแอปพลิเคชันสมัยใหม่

พิจารณาสถานการณ์ที่แอปพลิเคชันพาณิชย์อิเล็กทรอนิกส์จำเป็นต้องจัดการกระบวนการจัดการคำสั่งซื้อที่ซับซ้อน ลำดับงานอาจประกอบด้วยหลายขั้นตอน เช่น การตรวจสอบคำสั่งซื้อ การตรวจสอบสินค้าคงคลัง การประมวลผลการชำระเงิน การจัดส่ง และการแจ้งเตือนลูกค้า แต่ละขั้นตอนอาจมีกฎ การพึ่งพา การเชื่อมต่อภายนอก และกลไกการจัดการข้อยกเว้นของตัวเอง การจัดการลำดับงานดังกล่าวด้วยตนเองหรือผ่านตรรกะที่เขียนโค้ดไว้ตายตัวอาจกลายเป็นเรื่องยุ่งยาก เกิดข้อผิดพลาดได้ง่าย และยากต่อการบำรุงรักษาอย่างรวดเร็ว

ยิ่งไปกว่านั้น เมื่อแอปพลิเคชันขยายตัวและจำนวนผู้ใช้ที่ใช้งานพร้อมกันเพิ่มขึ้น ลำดับงานอาจจำเป็นต้องปรับตัวและเพิ่มประสิทธิภาพตามข้อมูลเรียลไทม์และประสิทธิภาพของระบบ ตัวอย่างเช่น ในช่วงที่มีการใช้งานสูง แอปพลิเคชันอาจจำเป็นต้องปรับลำดับงานแบบไดนามิกเพื่อจัดลำดับความสำคัญของงานบางอย่าง จัดสรรทรัพยากรอย่างมีประสิทธิภาพ และรับประกันประสบการณ์การใช้งานที่ราบรื่น

นี่คือจุดที่แนวทาง “การจัดการลำดับงานอัจฉริยะ” เข้ามามีบทบาท ด้วยการใช้องค์ประกอบด้าน AI นักพัฒนาสามารถสร้างลำดับงานที่ชาญฉลาด ปรับตัวได้ และเพิ่มประสิทธิภาพได้ด้วยตัวเอง AI สามารถวิเคราะห์ข้อมูลจำนวนมาก เรียนรู้จากประสบการณ์ในอดีต และตัดสินใจอย่างมีข้อมูลแบบเรียลไทม์เพื่อจัดการลำดับงานอย่างมีประสิทธิภาพ

ประโยชน์หลัก

1. **เพิ่มประสิทธิภาพ:** AI สามารถเพิ่มประสิทธิภาพการจัดสรรงาน การใช้ทรัพยากร และการดำเนินการตามลำดับงาน นำไปสู่เวลาประมวลผลที่เร็วขึ้นและประสิทธิภาพโดยรวมที่ดีขึ้น
2. **การปรับตัว:** ลำดับงานที่ขับเคลื่อนด้วย AI สามารถปรับตัวแบบไดนามิกตามสภาพแวดล้อมที่เปลี่ยนแปลง เช่น ความผันผวนของความต้องการของผู้ใช้ ประสิทธิภาพของระบบ หรือข้อกำหนดทางธุรกิจ เพื่อให้แน่ใจว่าแอปพลิเคชันยังคงตอบสนองและมีความยืดหยุ่น
3. **การตัดสินใจอัตโนมัติ:** AI สามารถทำให้กระบวนการตัดสินใจที่ซับซ้อนในลำดับงานเป็นอัตโนมัติ ลดการแทรกแซงด้วยมือและลดความเสี่ยงจากข้อผิดพลาดของมนุษย์
4. **การปรับแต่งเฉพาะบุคคล:** AI สามารถวิเคราะห์พฤติกรรมผู้ใช้ ความชอบ และบริบทเพื่อปรับแต่งลำดับงานและมอบประสบการณ์ที่เหมาะสมกับ
5. **ความสามารถในการขยายระบบ:** ลำดับงานที่ขับเคลื่อนด้วย AI สามารถขยายขนาดได้อย่างราบรื่นเพื่อรองรับปริมาณข้อมูลและการโต้ตอบกับผู้ใช้ที่เพิ่มขึ้น โดยไม่ส่งผลกระทบต่อประสิทธิภาพหรือความน่าเชื่อถือ

ในส่วนต่อไป เราจะสำรวจรูปแบบและเทคนิคสำคัญที่ช่วยให้สามารถนำลำดับงานอัจฉริยะไปใช้งานได้ และแสดงตัวอย่างจากโลกจริงของวิธีที่ AI กำลังเปลี่ยนแปลงการจัดการลำดับงานในแอปพลิเคชันสมัยใหม่

รูปแบบสำคัญ

ในการนำการจัดการลำดับงานอัจฉริยะมาใช้ในแอปพลิเคชัน นักพัฒนาสามารถใช้ประโยชน์จากรูปแบบสำคัญหลายประการที่ใช้พลังของ AI รูปแบบเหล่านี้ให้แนวทางที่เป็นระบบในการออกแบบและจัดการลำดับงาน ช่วยให้แอปพลิเคชันสามารถปรับตัว เพิ่มประสิทธิภาพ และทำงานอัตโนมัติตามข้อมูลและบริบทแบบเรียลไทม์ได้ มาสำรวจรูปแบบพื้นฐานบางอย่างในการจัดการลำดับงานอัจฉริยะกัน

การจัดเส้นทางงานแบบไดนามิก

รูปแบบนี้เกี่ยวข้องกับการใช้ AI ในการจัดเส้นทางงานภายในลำดับงานอย่างชาญฉลาด โดยพิจารณาจากปัจจัยต่างๆ เช่น ความสำคัญของงาน ความพร้อมใช้งานของทรัพยากร และประสิทธิภาพของระบบ อัลกอริธึม AI สามารถวิเคราะห์ลักษณะของแต่ละงาน พิจารณาสถานะปัจจุบันของระบบ และตัดสินใจอย่างมีข้อมูลในการมอบหมายงานให้กับทรัพยากรหรือเส้นทางการประมวลผลที่เหมาะสมที่สุด การจัดเส้นทางงานแบบไดนามิกช่วยให้มั่นใจว่างานถูกกระจายและดำเนินการอย่างมีประสิทธิภาพ เพื่อเพิ่มประสิทธิภาพการทำงานของลำดับงานโดยรวม

```
1  class TaskRouter
2    include Raix::ChatCompletion
3    include Raix::FunctionDispatch
4
5    attr_accessor :task
6
7    # list of functions that can be called by the AI entirely at its
8    # discretion depending on the task received
9
10   function :analyze_task_priority do
11     TaskPriorityAnalyzer.perform(task)
12   end
13
14   function :check_resource_availability, # ...
15   function :assess_system_performance, # ...
16   function :assign_task_to_resource, # ...
17
```

```
18    DIRECTIVE = "You are a task router, responsible for intelligently
19     assigning tasks to available resources based on priority, resource
20     availability, and system performance..."
21
22    def initialize(task)
23      self.task = task
24      transcript << { system: DIRECTIVE }
25      transcript << { user: task.to_json }
26    end
27
28    def perform
29      while task.unassigned?
30        chat_completion
31
32        # todo: add max loop counter and break
33      end
34
35      # capture the transcript for later analysis
36      task.update(routing_transcript: transcript)
37    end
38  end
```

สังเกตลูปที่สร้างขึ้นด้วยนิพจน์ while ในบรรทัดที่ 29 ซึ่งจะดำเนินการถามคำถาม AI จนกว่างานจะถูกมอบหมาย ในบรรทัดที่ 35 เราบันทึกการสนทนาของงานไว้เพื่อการวิเคราะห์และการแก้ไขข้อบกพร่องในภายหลังหากจำเป็น

การตัดสินใจตามบริบท

คุณสามารถใช้โค้ดที่คล้ายกันมากเพื่อทำการตัดสินใจที่คำนึงถึงบริบทภายในเวิร์กโฟลว์ได้ โดยการวิเคราะห์จุดข้อมูลที่เกี่ยวข้อง เช่น การตั้งค่าของผู้ใช้ รูปแบบในอดีต และข้อมูลนำเข้าแบบเรียลไทม์ องค์ประกอบ AI สามารถกำหนดแนวทางการดำเนินการที่เหมาะสมที่สุด ณ จุดตัดสินใจแต่ละจุดในเวิร์กโฟลว์ ปรับพฤติกรรมของเวิร์กโฟลว์ตามบริบทเฉพาะของผู้ใช้หรือสถานการณ์แต่ละราย เพื่อมอบประสบการณ์ที่เป็นส่วนตัวและได้รับการปรับให้เหมาะสมที่สุด

การจัดองค์ประกอบเวิร์กโฟลว์แบบปรับตัวได้

รูปแบบนี้มุ่งเน้นไปที่การจัดองค์ประกอบและปรับเวิร์กโฟลว์แบบไดนามิกตามความต้องการหรือเงื่อนไขที่เปลี่ยนแปลง AI สามารถวิเคราะห์สถานะปัจจุบันของเวิร์กโฟลว์ ระบุคอขวดหรือความไม่มีประสิทธิภาพ และปรับเปลี่ยนโครงสร้างเวิร์กโฟลว์โดยอัตโนมัติเพื่อเพิ่มประสิทธิภาพ การจัดองค์ประกอบเวิร์กโฟลว์แบบปรับตัวได้ช่วยให้แอปพลิเคชันสามารถพัฒนาและปรับปรุงกระบวนการอย่างต่อเนื่องโดยไม่ต้องการการแทรกแซงด้วยตนเอง

การจัดการข้อยกเว้นและการกู้คืน

การจัดการข้อยกเว้นและการกู้คืนเป็นแง่มุมที่สำคัญของการจัดการเวิร์กโฟลว์อัจฉริยะ เมื่อทำงานกับองค์ประกอบ AI และเวิร์กโฟลว์ที่ซับซ้อน จำเป็นต้องคาดการณ์และจัดการข้อยกเว้นอย่างเหมาะสมเพื่อให้มั่นใจในความเสถียรและความน่าเชื่อถือของระบบ

ต่อไปนี้คือข้อควรพิจารณาและเทคนิคสำคัญสำหรับการจัดการข้อยกเว้นและการกู้คืนในเวิร์กโฟลว์อัจฉริยะ:

1. **การส่งต่อข้อยกเว้น:** ใช้วิธีการที่สอดคล้องกันในการส่งต่อข้อยกเว้นระหว่างองค์ประกอบของเวิร์กโฟลว์ เมื่อเกิดข้อยกเว้นภายในองค์ประกอบ ควรจับ บันทึก และส่งต่อไปยังตัวออร์เคสเตรเตอร์หรือองค์ประกอบแยกที่รับผิดชอบในการจัดการข้อยกเว้น แนวคิดคือการรวมศูนย์การจัดการข้อยกเว้นและป้องกันไม่ให้ข้อยกเว้นถูกเพิกเฉย รวมถึงเปิดโอกาสสำหรับการจัดการข้อผิดพลาดอัจฉริยะ
2. **กลไก การ ลอง ใหม่:** กลไก การ ลอง ใหม่ ช่วย ปรับปรุง ความ ยืดหยุ่น ของ เวิร์กโฟลว์ และ จัดการ ความ ล้ม เหลว ชั่ว คราว อย่าง เหมาะ สม ควร พยายาม ใช้ กลไก การ ลอง ใหม่ สำหรับ ข้อ ยกเว้น ชั่ว คราว หรือ ที่ สามารถ กู้ คืน ได้ เช่น การเชื่อมต่อเครือข่ายหรือทรัพยากรไม่พร้อมใช้งานที่สามารถลองใหม่โดยอัตโนมัติหลังจากหน่วงเวลาที่กำหนด การมีตัวออร์เคสเตรเตอร์หรือตัวจัดการข้อยกเว้นที่ขับเคลื่อนด้วย AI หมายความว่ากลยุทธ์การลองใหม่ของคุณไม่จำเป็นต้องเป็นเชิงกลไก โดยอาศัยอัลกอริธึมตายตัวเช่นการถอยหลังแบบเอกซ์โพเนนเชียล คุณสามารถปล่อยให้การจัดการการลองใหม่ขึ้นอยู่กับ “ดุลยพินิจ” ขององค์ประกอบ AI ที่รับผิดชอบในการตัดสินใจว่าจะจัดการข้อยกเว้นอย่างไร
3. **กลยุทธ์ การ ถอย กลับ:** หาก องค์ ประกอบ AI ไม่ สามารถ ให้การ ตอบ สนอง ที่ ถูก ต้อง หรือ พบ ข้อ ผิด พลาด —ซึ่ง เป็น เหตุการณ์ที่พบบ่อยเนื่องจากธรรมชาติที่ทันสมัยล้ำหน้า—ต้องมีกลไกการถอยกลับเพื่อให้แน่ใจว่าเวิร์กโฟลว์สามารถดำเนินต่อไปได้ อาจเกี่ยวข้องกับการใช้ค่าเริ่มต้น อัลกอริธึมทางเลือก หรือมนุษย์ในลูป เพื่อตัดสินใจและทำให้เวิร์กโฟลว์ดำเนินต่อไปได้
4. **การดำเนินการชดเชย:** คำสั่งของตัวออร์เคสเตรเตอร์ควรรวมคำแนะนำเกี่ยวกับการดำเนินการชดเชยเพื่อจัดการข้อยกเว้นที่ไม่สามารถแก้ไขได้โดยอัตโนมัติ การดำเนินการชดเชยคือขั้นตอนที่ดำเนินการเพื่อยกเลิกหรือบรรเทาผลกระทบของการดำเนินการที่ล้มเหลว ตัวอย่างเช่น หากขั้นตอนการประมวลผลการชำระเงินล้มเหลว การดำเนินการชดเชยอาจเป็นการย้อนกลับธุรกรรมและแจ้งให้ผู้ใช้ทราบ การดำเนินการชดเชยช่วยรักษาความสอดคล้องและความถูกต้องของข้อมูลเมื่อเกิดข้อยกเว้น
5. **การตรวจสอบและการแจ้งเตือนข้อยกเว้น:** ตั้งค่ากลไกการตรวจสอบและการแจ้งเตือนเพื่อตรวจจับและแจ้งเตือนผู้มีส่วนได้ส่วนเสียที่เกี่ยวข้องเกี่ยวกับข้อยกเว้นที่สำคัญ ตัวออร์เคสเตรเตอร์สามารถรับรู้เกณฑ์และกฎเพื่อทริกเกอร์การ

แจ้งเตือนเมื่อข้อยกเว้นเกินขีดจำกัดบางอย่างหรือเมื่อเกิดข้อยกเว้นประเภทเฉพาะ สิ่งนี้ช่วยให้สามารถระบุและแก้ไขปัญหาเชิงรุกก่อนที่จะส่งผลกระทบต่อระบบโดยรวม

ต่อไปนี้เป็นตัวอย่างของการจัดการข้อยกเว้นและการกู้คืนในองค์ประกอบเวิร์กโฟลว์ Ruby:

```ruby
class InventoryManager
  def check_availability(order)
    begin
      # Perform inventory check logic
      inventory = Inventory.find_by(product_id: order.product_id)
      if inventory.available_quantity >= order.quantity
        return true
      else
        raise InsufficientInventoryError,
            "Insufficient inventory for product #{order.product_id}"
      end
    rescue InsufficientInventoryError => e
      # Log the exception
      logger.error("Inventory check failed: #{e.message}")

      # Retry the operation after a delay
      retry_count ||= 0
      if retry_count < MAX_RETRIES
        retry_count += 1
        sleep(RETRY_DELAY)
        retry
      else
        # Fallback to manual intervention
        NotificationService.admin("Inventory check failed: Order #{order.id}")
        return false
      end
    end
  end
end
```

ในตัวอย่างนี้ คอมโพเนนต์ InventoryManager จะตรวจสอบความพร้อมของสินค้าสำหรับคำสั่งซื้อที่กำหนด หากมีปริมาณไม่เพียงพอ จะเกิดข้อผิดพลาด InsufficientInventoryError ขึ้น ข้อผิดพลาดนี้จะถูกจับ บันทึก และมีการใช้กลไกการลองใหม่ หากเกินขีดจำกัดของการลองใหม่ คอมโพเนนต์จะเปลี่ยนไปใช้การแทรกแซงด้วยมือโดยแจ้งเตือนผู้ดูแลระบบ

ด้วยการใช้กลไกการจัดการและกู้คืนข้อผิดพลาดที่แข็งแกร่ง คุณสามารถมั่นใจได้ว่าลำดับงานอัจฉริยะของคุณมีความยืดหยุ่น บำรุงรักษาได้ และสามารถจัดการกับสถานการณ์ที่ไม่คาดคิดได้อย่างราบรื่น

รูปแบบเหล่านี้เป็นพื้นฐานของการจัดการลำดับงานอย่างชาญฉลาด และสามารถผสมผสานและปรับเปลี่ยนให้เหมาะกับความต้องการเฉพาะของแอปพลิเคชันต่างๆ ด้วยการใช้ประโยชน์จากรูปแบบเหล่านี้ นักพัฒนาสามารถสร้างลำดับงานที่มีความยืดหยุ่น ทนทาน และได้รับการปรับให้เหมาะสมทั้งด้านประสิทธิภาพและประสบการณ์ของผู้ใช้

ในส่วนถัดไป เราจะสำรวจวิธีการนำรูปแบบเหล่านี้ไปใช้ในทางปฏิบัติ โดยใช้ตัวอย่างจากโลกแห่งความเป็นจริงและตัวอย่างโค้ดเพื่อแสดงการผสานรวมคอมโพเนนต์ AI เข้ากับการจัดการลำดับงาน

การนำการจัดการลำดับงานอย่างชาญฉลาดไปใช้ในทางปฏิบัติ

หลังจากที่เราได้สำรวจรูปแบบสำคัญในการจัดการลำดับงานอย่างชาญฉลาดแล้ว มาดูกันว่ารูปแบบเหล่านี้สามารถนำไปใช้ในแอปพลิเคชันจริงได้อย่างไร เราจะให้ตัวอย่างที่ใช้งานได้จริงและตัวอย่างโค้ดเพื่อแสดงการผสานรวมคอมโพเนนต์ AI เข้ากับการจัดการลำดับงาน

ตัวประมวลผลคำสั่งซื้ออัจฉริยะ

มาดูตัวอย่างการใช้งานจริงของการจัดการลำดับงานอย่างชาญฉลาดโดยใช้คอมโพเนนต์ OrderProcessor ที่ขับเคลื่อนด้วย AI ในแอปพลิเคชัน e-commerce ที่ใช้ Ruby on Rails คอมโพเนนต์ OrderProcessor นี้เป็นการนำแนวคิด Process Manager Enterprise Integration ที่เราได้พบครั้งแรกในบทที่ 3 เมื่อพูดถึง Multitude of Workers มาใช้ คอมโพเนนต์นี้จะรับผิดชอบในการจัดการลำดับงานการดำเนินการตามคำสั่งซื้อ ตัดสินใจเรื่องการจัดเส้นทางตามผลลัพธ์ระหว่างทาง และประสานงานการดำเนินการของขั้นตอนต่างๆ

กระบวนการดำเนินการตามคำสั่งซื้อประกอบด้วยหลายขั้นตอน เช่น การตรวจสอบความถูกต้องของคำสั่งซื้อ การตรวจสอบสินค้าคงคลัง การประมวลผลการชำระเงิน และการจัดส่ง แต่ละขั้นตอนถูกนำไปใช้เป็นกระบวนการทำงานแยกที่ทำงานเฉพาะและส่งผลลัพธ์กลับไปยัง OrderProcessor ขั้นตอนเหล่านี้ไม่จำเป็นต้องทำทั้งหมด และไม่จำเป็นต้องทำตามลำดับที่แน่นอน

นี่คือตัวอย่างการใช้งาน OrderProcessor มันมี mixin สองตัวจาก Raix ตัวแรก (ChatCompletion) ให้ความสามารถในการทำ chat completion ซึ่งทำให้นี่เป็นคอมโพเนนต์ AI ตัวที่สอง (FunctionDispatch) เปิดใช้งานการเรียกฟังก์ชัน โดย AI ทำให้สามารถตอบสนองต่อ prompt ด้วยการเรียกฟังก์ชันแทนข้อความได้

ฟังก์ชันของ worker (validate_order, check_inventory และอื่นๆ) จะส่งต่อไปยังคลาส worker ที่เกี่ยวข้อง ซึ่งอาจเป็นคอมโพเนนต์ AI หรือไม่ใช่ก็ได้ โดยมีข้อกำหนดเพียงว่าต้องส่งคืนผลลัพธ์ของงานในรูปแบบที่สามารถแสดงเป็นสตริงได้

เช่นเดียวกับตัวอย่างอื่นๆ ในส่วนนี้ของหนังสือ โค้ดนี้เป็นเพียง pseudo-code และมีไว้เพื่อสื่อความหมายของรูปแบบและสร้างแรงบันดาลใจให้กับการสร้างสรรค์ของคุณเอง คำอธิบายรูปแบบที่สมบูรณ์และตัวอย่างโค้ดทั้งหมดจะรวมอยู่ในส่วนที่ 2

```
class OrderProcessor
  include Raix::ChatCompletion
  include Raix::FunctionDispatch

  SYSTEM_DIRECTIVE = "You are an order processor, tasked with..."

  def initialize(order)
    self.order = order
    transcript << { system: SYSTEM_DIRECTIVE }
    transcript << { user: order.to_json }
  end

  def perform
    # will continue looping until `stop_looping!` is called
    chat_completion(loop: true)
  end

  # list of functions available to be called by the AI
  # truncated for brevity

  def functions
    [
      {
        name: "validate_order",
        description: "Invoke to check validity of order",
        parameters: {
          ...
      },
      ...
    ]
  end

```

```
  # implementation of functions that can be called by the AI
  # entirely at its discretion, depending on the needs of the order

  def validate_order
    OrderValidationWorker.perform(@order)
  end

  def check_inventory
    InventoryCheckWorker.perform(@order)
  end

  def process_payment
    PaymentProcessingWorker.perform(@order)
  end

  def schedule_shipping
    ShippingSchedulerWorker.perform(@order)
  end

  def send_confirmation
    OrderConfirmationWorker.perform(@order)
  end

  def finished_processing
    @order.update!(transcript:, processed_at: Time.current)
    stop_looping!
  end
end
```

ในตัวอย่างนี้ OrderProcessor ถูกเริ่มต้นด้วยออบเจ็กต์คำสั่งซื้อและเก็บรักษาบันทึกการสนทนาของการดำเนินการตามลำดับงาน ในรูปแบบบันทึกการสนทนาทั่วไปที่เป็นธรรมชาติสำหรับแบบจำลองภาษาขนาดใหญ่ AI ได้รับการควบคุมอย่างสมบูรณ์ในการจัดการการดำเนินการของขั้นตอนต่างๆ เช่น การตรวจสอบคำสั่งซื้อ การตรวจสอบสินค้าคงคลัง การประมวลผลการชำระเงิน และการจัดส่ง

ทุกครั้งที่เรียกใช้เมธอด chat_completion บันทึกการสนทนาจะถูกส่งไปยัง AI เพื่อให้ส่งคืนการตอบสนองในรูปแบบของการเรียกใช้ฟังก์ชัน ขึ้นอยู่กับ AI ทั้งหมดในการวิเคราะห์ผลลัพธ์ของขั้นตอนก่อนหน้าและตัดสินใจว่าควรดำเนินการอย่างไรต่อไป ตัวอย่างเช่น หากการตรวจสอบสินค้าคงคลังแสดงให้เห็นว่าระดับสินค้าต่ำ OrderProcessor สามารถกำหนดงานเติมสินค้าได้ หากการประมวลผลการชำระเงินล้มเหลว ก็สามารถเริ่มการลองใหม่หรือแจ้งเตือนฝ่ายสนับสนุนลูกค้าได้

> ตัวอย่างข้างต้นไม่ได้มีการกำหนดฟังก์ชันสำหรับการเติมสินค้าหรือการแจ้งเตือนฝ่ายสนับสนุนลูกค้า แต่สามารถทำได้อย่างแน่นอน

บันทึกการสนทนาจะเพิ่มขึ้นทุกครั้งที่มีการเรียกใช้ฟังก์ชันและทำหน้าที่เป็นบันทึกการดำเนินการตามลำดับงาน รวมถึงผลลัพธ์ของแต่ละขั้นตอนและคำแนะนำที่สร้างโดย AI สำหรับขั้นตอนถัดไป บันทึกนี้สามารถใช้สำหรับการแก้ไขข้อบกพร่อง การตรวจสอบ และการให้ความโปร่งใสในกระบวนการจัดการคำสั่งซื้อ

ด้วยการใช้ประโยชน์จาก AI ใน OrderProcessor แอปพลิเคชันอีคอมเมิร์ซสามารถปรับลำดับการทำงานแบบไดนามิกตามข้อมูลแบบเรียลไทม์และจัดการข้อยกเว้นอย่างชาญฉลาด องค์ประกอบ AI สามารถตัดสินใจอย่างมีข้อมูล ปรับลำดับการทำงานให้เหมาะสม และรับประกันการประมวลผลคำสั่งซื้อที่ราบรื่นแม้ในสถานการณ์ที่ซับซ้อน

เมื่อข้อกำหนดเดียวสำหรับกระบวนการทำงานคือการส่งคืนผลลัพธ์ที่เข้าใจได้เพื่อให้ AI พิจารณาว่าจะทำอะไรต่อไป คุณอาจเริ่มตระหนักว่าแนวทางนี้สามารถลดงานการจับคู่อินพุต/เอาต์พุตที่มักเกี่ยวข้องเมื่อมีการผสานระบบที่แตกต่างกันเข้าด้วยกัน

ตัวกลั่นกรองเนื้อหาอัจฉริยะ

แอปพลิเคชันโซเชียลมีเดียโดยทั่วไปต้องการการกลั่นกรองเนื้อหาขั้นต่ำเพื่อให้มั่นใจว่าชุมชนมีความปลอดภัยและสุขภาพดี ตัวอย่างคอมโพเนนต์ ContentModerator นี้ใช้ประโยชน์จาก AI ในการจัดการลำดับการกลั่นกรองอย่างชาญฉลาด โดยตัดสินใจตามลักษณะของเนื้อหาและผลลัพธ์ของขั้นตอนการกลั่นกรองต่างๆ

กระบวนการกลั่นกรองประกอบด้วยหลายขั้นตอน เช่น การวิเคราะห์ข้อความ การจดจำรูปภาพ การประเมินความน่าเชื่อถือของผู้ใช้ และการตรวจสอบด้วยมนุษย์ แต่ละขั้นตอนถูกนำไปใช้เป็นกระบวนการทำงานแยกต่างหากที่ทำงานเฉพาะและส่งผลลัพธ์กลับไปยัง ContentModerator

นี่คือตัวอย่างการใช้งาน ContentModerator:

```ruby
class ContentModerator
  include Raix::ChatCompletion
  include Raix::FunctionDispatch

  SYSTEM_DIRECTIVE = "You are a content moderator process manager,
    tasked with the workflow involved in moderating user-generated content..."

  def initialize(content)
    @content = content
    @transcript = [
      { system: SYSTEM_DIRECTIVE },
      { user: content.to_json }
    ]
  end

  def perform
    complete(@transcript)
  end

  def model
    "openai/gpt-4"
  end

  # list of functions available to be called by the AI
  # truncated for brevity

  def functions
    [
      {
        name: "analyze_text",
        # ...
      },
      {
        name: "recognize_image",
        description: "Invoke to describe images...",
        # ...
      },
      {
        name: "assess_user_reputation",
        # ...
      },
      {
```

```
        name: "escalate_to_manual_review",
        # ...
      },
      {
        name: "approve_content",
        # ...
      },
      {
        name: "reject_content",
        # ...
      }
    ]
  end

  # implementation of functions that can be called by the AI
  # entirely at its discretion, depending on the needs of the order

  def analyze_text
    result = TextAnalysisWorker.perform(@content)
    continue_with(result)
  end

  def recognize_image
    result = ImageRecognitionWorker.perform(@content)
    continue_with(result)
  end

  def assess_user_reputation
    result = UserReputationWorker.perform(@content.user)
    continue_with(result)
  end

  def escalate_to_manual_review
    ManualReviewWorker.perform(@content)
    @content.update!(status: 'pending', transcript: @transcript)
  end

  def approve_content
    @content.update!(status: 'approved', transcript: @transcript)
  end

  def reject_content
```

```
    @content.update!(status: 'rejected', transcript: @transcript)
  end

  private

  def continue_with(result)
    @transcript << { function: result }
    complete(@transcript)
  end
end
```

ในตัวอย่างนี้ ContentModerator ถูกเริ่มต้นด้วยออบเจกต์เนื้อหาและรักษาบันทึกการตรวจสอบในรูปแบบการสนทนา ส่วนประกอบ AI มีการควบคุมเต็มรูปแบบเหนือลำดับการตรวจสอบ โดยตัดสินใจว่าจะดำเนินการขั้นตอนใดบ้างตามลักษณะของเนื้อหาและผลลัพธ์ของแต่ละขั้นตอน

ฟังก์ชันการทำงานที่ AI สามารถเรียกใช้ได้ประกอบด้วย analyze_text, recognize_image, assess_user_reputation และ escalate_to_manual_review แต่ละฟังก์ชันจะมอบหมายงานให้กับกระบวนการทำงานที่เกี่ยวข้อง (TextAnalysisWorker, ImageRecognitionWorker ฯลฯ) และเพิ่มผลลัพธ์ลงในบันทึกการตรวจสอบ ยกเว้นฟังก์ชันการส่งต่อซึ่งทำหน้าที่เป็นสถานะสิ้นสุด นอกจากนี้ ฟังก์ชัน approve_content และ reject_content ก็ทำหน้าที่เป็นสถานะสิ้นสุดเช่นกัน

ส่วนประกอบ AI วิเคราะห์เนื้อหาและกำหนดการดำเนินการที่เหมาะสม หากเนื้อหามีการอ้างอิงรูปภาพ มันสามารถเรียกใช้ worker recognize_image เพื่อช่วยในการตรวจสอบภาพ หาก worker ใดเตือนเกี่ยวกับเนื้อหาที่อาจเป็นอันตราย AI อาจตัดสินใจส่งต่อเนื้อหาไปยังการตรวจสอบด้วยมนุษย์หรือปฏิเสธทันที แต่ขึ้นอยู่กับความรุนแรงของคำเตือน AI อาจเลือกใช้ผลการประเมินชื่อเสียงของผู้ใช้ในการตัดสินใจว่าจะจัดการกับเนื้อหาที่ไม่แน่ใจอย่างไร ขึ้นอยู่กับกรณีการใช้งาน บางทีผู้ใช้ที่น่าเชื่อถืออาจมีความยืดหยุ่นมากกว่าในสิ่งที่พวกเขาสามารถโพสต์ได้ และอื่นๆ อีกมากมาย...

เช่นเดียวกับตัวอย่างตัวจัดการกระบวนการก่อนหน้านี้ บันทึกการตรวจสอบทำหน้าที่เป็นบันทึกการดำเนินการของลำดับงาน รวมถึงผลลัพธ์ของแต่ละขั้นตอนและการตัดสินใจที่สร้างโดย AI บันทึกนี้สามารถใช้สำหรับการตรวจสอบ ความโปร่งใส และการปรับปรุงกระบวนการตรวจสอบเนื้อหาในระยะยาว

ด้วยการใช้ประโยชน์จาก AI ใน ContentModerator แอปพลิเคชันโซเชียลมีเดียสามารถปรับลำดับการตรวจสอบแบบไดนามิกตามลักษณะของเนื้อหาและจัดการกับสถานการณ์การตรวจสอบที่ซับซ้อนได้อย่างชาญฉลาด ส่วนประกอบ AI สามารถตัดสินใจอย่างมีข้อมูล ปรับลำดับการทำงานให้เหมาะสม และรับรองประสบการณ์ชุมชนที่ปลอดภัยและมีสุขภาพดี

มาสำรวจตัวอย่างอีกสองตัวอย่างที่แสดงให้เห็นถึงการจัดตารางงานเชิงคาดการณ์และการจัดการข้อผิดพลาดและการกู้คืนในบริบทของการประสานงานลำดับการทำงานอัจฉริยะกัน

การจัดตารางงานเชิงคาดการณ์ในระบบสนับสนุนลูกค้า

ในแอปพลิเคชันสนับสนุนลูกค้าที่สร้างด้วย Ruby on Rails การจัดการและจัดลำดับความสำคัญของตั๋วงานสนับสนุนอย่างมีประสิทธิภาพเป็นสิ่งสำคัญในการให้ความช่วยเหลือลูกค้าได้ทันเวลา ส่วนประกอบ SupportTicketScheduler ใช้ประโยชน์จาก AI ในการจัดตารางและมอบหมายตั๋วงานสนับสนุนให้กับเจ้าหน้าที่ที่ว่างอยู่แบบเชิงคาดการณ์ โดยอิงจากปัจจัยต่างๆ เช่น ความเร่งด่วนของตั๋วงาน ความเชี่ยวชาญของเจ้าหน้าที่ และปริมาณงาน

```
class SupportTicketScheduler
  include Raix::ChatCompletion
  include Raix::FunctionDispatch

  SYSTEM_DIRECTIVE = "You are a support ticket scheduler,
    tasked with intelligently assigning tickets to available agents..."

  def initialize(ticket)
    @ticket = ticket
    @transcript = [
      { system: SYSTEM_DIRECTIVE },
      { user: ticket.to_json }
    ]
  end

  def perform
    complete(@transcript)
  end

  def model
    "openai/gpt-4"
  end

  def functions
    [
      {
        name: "analyze_ticket_urgency",
        # ...
      },
      {
        name: "list_available_agents",
        description: "Includes expertise of available agents",
        # ...
```

```
    },
    {
      name: "predict_agent_workload",
      description: "Uses historical data to predict upcoming workloads",
      # ...
    },
    {
      name: "assign_ticket_to_agent",
      # ...
    },
    {
      name: "reschedule_ticket",
      # ...
    }
  ]
end

# implementation of functions that can be called by the AI
# entirely at its discretion, depending on the needs of the order

def analyze_ticket_urgency
  result = TicketUrgencyAnalyzer.perform(@ticket)
  continue_with(result)
end

def list_available_agents
  result = ListAvailableAgents.perform
  continue_with(result)
end

def predict_agent_workload
  result = AgentWorkloadPredictor.perform
  continue_with(result)
end

def assign_ticket_to_agent
  TicketAssigner.perform(@ticket, @transcript)
end

def delay_assignment(until)
  until = DateTimeStandardizer.process(until)
  SupportTicketScheduler.delay(@ticket, @transcript, until)
```

```
  end

  private

  def continue_with(result)
    @transcript << { function: result }
    complete(@transcript)
  end
end
```

ในตัวอย่างนี้ SupportTicketScheduler ถูกเริ่มต้นด้วยออบเจกต์ตั๋วสนับสนุนและรักษาบันทึกการจัดตาราง ส่วนประกอบ AI วิเคราะห์รายละเอียดตั๋วและคาดการณ์การจัดตารางการมอบหมายตั๋วโดยอิงจากปัจจัยต่างๆ เช่น ความเร่งด่วนของตั๋ว ความเชี่ยวชาญของเจ้าหน้าที่ และภาระงานที่คาดการณ์ของเจ้าหน้าที่

ฟังก์ชันที่ AI สามารถเรียกใช้ได้ประกอบด้วย analyze_ticket_urgency, list_available_agents, predict_agent_workload และ assign_ticket_to_agent แต่ละฟังก์ชันจะมอบหมายงานให้กับส่วนประกอบการวิเคราะห์หรือการคาดการณ์ที่เกี่ยวข้องและเพิ่มผลลัพธ์ลงในบันทึกการจัดตาราง AI ยังมีตัวเลือกในการชะลอการมอบหมายโดยใช้ฟังก์ชัน delay_assignment

ส่วนประกอบ AI ตรวจสอบบันทึกการจัดตารางและตัดสินใจเกี่ยวกับการมอบหมายตั๋วอย่างมีข้อมูล มันพิจารณาความเร่งด่วนของตั๋ว ความเชี่ยวชาญของเจ้าหน้าที่ที่ว่าง และภาระงานที่คาดการณ์ของเจ้าหน้าที่แต่ละคนเพื่อกำหนดเจ้าหน้าที่ที่เหมาะสมที่สุดในการจัดการตั๋ว

ด้วยการใช้ประโยชน์จากการจัดตารางงานเชิงคาดการณ์ แอปพลิเคชันการสนับสนุนลูกค้าสามารถเพิ่มประสิทธิภาพการมอบหมายตั๋ว ลดเวลาการตอบสนอง และปรับปรุงความพึงพอใจของลูกค้าโดยรวม การจัดการตั๋วสนับสนุนเชิงรุกและมีประสิทธิภาพช่วยให้มั่นใจว่าตั๋วที่ถูกต้องจะถูกมอบหมายให้กับเจ้าหน้าที่ที่เหมาะสมในเวลาที่เหมาะสม

การจัดการข้อยกเว้นและการกู้คืนในไปป์ไลน์การประมวลผลข้อมูล

การจัดการข้อยกเว้นและการกู้คืนจากความล้มเหลวมีความสำคัญในการรับรองความถูกต้องสมบูรณ์ของข้อมูลและป้องกันการสูญเสียข้อมูล ส่วนประกอบ DataProcessingOrchestrator ใช้ AI ในการจัดการข้อยกเว้นอย่างชาญฉลาดและจัดระเบียบกระบวนการกู้คืนในไปป์ไลน์การประมวลผลข้อ

```ruby
class DataProcessingOrchestrator
  include Raix::ChatCompletion
  include Raix::FunctionDispatch

  SYSTEM_DIRECTIVE = "You are a data processing orchestrator..."

  def initialize(data_batch)
    @data_batch = data_batch
    @transcript = [
      { system: SYSTEM_DIRECTIVE },
      { user: data_batch.to_json }
    ]
  end

  def perform
    complete(@transcript)
  end

  def model
    "openai/gpt-4"
  end

  def functions
    [
      {
        name: "validate_data",
        # ...
      },
      {
        name: "process_data",
        # ...
      },
      {
        name: "request_fix",
        # ...
      },
      {
        name: "retry_processing",
        # ...
      },
      {
        name: "mark_data_as_failed",
```

```
      # ...
    },
    {
      name: "finished",
      # ...
    }
  ]
end

# implementation of functions that can be called by the AI
# entirely at its discretion, depending on the needs of the order

def validate_data
  result = DataValidator.perform(@data_batch)
  continue_with(result)
rescue ValidationException => e
  handle_validation_exception(e)
end

def process_data
  result = DataProcessor.perform(@data_batch)
  continue_with(result)
rescue ProcessingException => e
  handle_processing_exception(e)
end

def request_fix(description_of_fix)
  result = SmartDataFixer.new(description_of_fix, @data_batch)
  continue_with(result)
end

def retry_processing(timeout_in_seconds)
  wait(timeout_in_seconds)
  process_data
end

def mark_data_as_failed
  @data_batch.update!(status: 'failed', transcript: @transcript)
end

def finished
  @data_batch.update!(status: 'finished', transcript: @transcript)
```

```
  end

  private

  def continue_with(result)
    @transcript << { function: result }
    complete(@transcript)
  end

  def handle_validation_exception(exception)
    @transcript << { exception: exception.message }
    complete(@transcript)
  end

  def handle_processing_exception(exception)
    @transcript << { exception: exception.message }
    complete(@transcript)
  end
end
```

ในตัวอย่างนี้ DataProcessingOrchestrator ถูกเริ่มต้นด้วยออบเจ็กต์ชุดข้อมูลและรักษาบันทึกการประมวลผล ส่วนประกอบ AI ทำหน้าที่ควบคุมไปป์ไลน์การประมวลผลข้อมูล จัดการข้อยกเว้น และกู้คืนจากความล้มเหลวตามความจำเป็น

ฟังก์ชันที่ AI สามารถเรียกใช้ได้ประกอบด้วย validate_data, process_data, request_fix, retry_processing และ mark_data_as_failed แต่ละฟังก์ชันจะมอบหมายงานให้กับคอมโพเนนต์การประมวลผลข้อมูลที่เกี่ยวข้องและเพิ่มผลลัพธ์หรือรายละเอียดข้อยกเว้นลงในบันทึกการประมวลผล

หากเกิดข้อยกเว้นในการตรวจสอบความถูกต้องระหว่างขั้นตอน validate_data ฟังก์ชัน handle_validation_exception จะเพิ่มข้อมูลข้อยกเว้นลงในบันทึกและส่งการควบคุมกลับไปยัง AI ในทำนองเดียวกัน หากเกิดข้อยกเว้นในการประมวลผลระหว่างขั้นตอน process_data AI สามารถตัดสินใจเลือกกลยุทธ์การกู้คืนได้

ขึ้นอยู่กับลักษณะของข้อยกเว้นที่พบ AI สามารถตัดสินใจเรียกใช้ request_fix ซึ่งจะมอบหมายให้คอมโพเนนต์ SmartDataFixer ที่ขับเคลื่อนด้วย AI (ดูบทเกี่ยวกับข้อมูลที่รักษาตัวเองได้) ตัวแก้ไขข้อมูลจะได้รับคำอธิบายเป็นภาษาอังกฤษธรรมดาว่าควรแก้ไข @data_batch อย่างไรเพื่อให้สามารถลองประมวลผลใหม่ได้ บางทีการลองใหม่ที่ประสบความสำเร็จอาจเกี่ยวข้องกับการลบระเบียนที่ไม่ผ่านการตรวจสอบความถูกต้องออกจากชุดข้อมูลและ/หรือคัดลอกไปยังไปป์ไลน์การประมวลผลอื่นเพื่อให้มนุษย์ตรวจสอบ? ความเป็นไปได้มีมากมายเกือบไม่มีที่สิ้นสุด

ด้วยการรวมการจัดการข้อยกเว้นและการกู้คืนที่ขับเคลื่อนด้วย AI แอปพลิเคชันการประมวลผลข้อมูลจะมีความยืดหยุ่นและทน

ต่อข้อผิดพลาดมากขึ้น DataProcessingOrchestrator จัดการข้อยกเว้นอย่างชาญฉลาด ลดการสูญเสียข้อมูล และรับประกันการดำเนินการของเวิร์กโฟลว์การประมวลผลข้อมูลอย่างราบรื่น

การตรวจสอบติดตามและการบันทึกข้อมูล

การตรวจสอบติดตามและการบันทึกข้อมูลช่วยให้มองเห็นความคืบหน้า ประสิทธิภาพ และสุขภาพของคอมโพเนนต์เวิร์กโฟลว์ที่ขับเคลื่อนด้วย AI ช่วยให้นักพัฒนาสามารถติดตามและวิเคราะห์พฤติกรรมของระบบ การใช้กลไกการตรวจสอบติดตามและการบันทึกข้อมูลที่มีประสิทธิภาพเป็นสิ่งจำเป็นสำหรับการแก้ไขข้อบกพร่อง การตรวจสอบ และการปรับปรุงเวิร์กโฟลว์อัจฉริยะอย่างต่อเนื่อง

การตรวจสอบติดตามความคืบหน้าและประสิทธิภาพของเวิร์กโฟลว์

เพื่อให้แน่ใจว่าเวิร์กโฟลว์อัจฉริยะทำงานได้อย่างราบรื่น จึงเป็นสิ่งสำคัญที่จะต้องตรวจสอบติดตามความคืบหน้าและประสิทธิภาพของคอมโพเนนต์เวิร์กโฟลว์แต่ละตัว ซึ่งเกี่ยวข้องกับการติดตามเมทริกและเหตุการณ์สำคัญตลอดวงจรชีวิตของเวิร์กโฟลว์

ด้านสำคัญที่ต้องตรวจสอบติดตามได้แก่:

1. **เวลาในการดำเนินการเวิร์กโฟลว์:** วัดเวลาที่คอมโพเนนต์เวิร์กโฟลว์แต่ละตัวใช้ในการทำงานให้เสร็จสิ้น ซึ่งช่วยระบุคอขวดด้านประสิทธิภาพและเพิ่มประสิทธิภาพของเวิร์กโฟลว์โดยรวม

2. **การใช้ทรัพยากร:** ตรวจสอบติดตามการใช้ทรัพยากรระบบ เช่น CPU หน่วยความจำ และพื้นที่จัดเก็บข้อมูล โดยคอมโพเนนต์เวิร์กโฟลว์แต่ละตัว ซึ่งช่วยให้แน่ใจว่าระบบทำงานภายในขีดความสามารถและสามารถจัดการกับภาระงานได้อย่างมีประสิทธิภาพ

3. **อัตราข้อผิดพลาดและข้อยกเว้น:** ติดตามการเกิดข้อผิดพลาดและข้อยกเว้นภายในคอมโพเนนต์เวิร์กโฟลว์ ซึ่งช่วยระบุปัญหาที่อาจเกิดขึ้นและช่วยให้สามารถจัดการและกู้คืนข้อผิดพลาดเชิงรุกได้

4. **จุดตัดสินใจและผลลัพธ์:** ตรวจสอบติดตามจุดตัดสินใจภายในเวิร์กโฟลว์และผลลัพธ์ของการตัดสินใจที่ขับเคลื่อนด้วย AI ซึ่งให้ข้อมูลเชิงลึกเกี่ยวกับพฤติกรรมและประสิทธิผลของคอมโพเนนต์ AI

ข้อมูลที่รวบรวมโดยกระบวนการตรวจสอบติดตามสามารถแสดงในแดชบอร์ดหรือใช้เป็นข้อมูลนำเข้าสำหรับรายงานตามกำหนดเวลาที่แจ้งให้ผู้ดูแลระบบทราบเกี่ยวกับสุขภาพของระบบ

ข้อมูลการตรวจสอบติดตามสามารถป้อนไปยังกระบวนการผู้ดูแลระบบที่ขับเคลื่อนด้วย AI เพื่อตรวจสอบและดำเนินการที่อาจจำเป็น!

การบันทึกเหตุการณ์และการตัดสินใจที่สำคัญ

การบันทึกข้อมูลเป็นแนวปฏิบัติที่สำคัญซึ่งเกี่ยวข้องกับการจับและจัดเก็บข้อมูลที่เกี่ยวข้องเกี่ยวกับเหตุการณ์สำคัญ การตัดสินใจ และข้อยกเว้นที่เกิดขึ้นระหว่างการดำเนินการเวิร์กโฟลว์

ด้านสำคัญที่ต้องบันทึกได้แก่:

1. **การเริ่มต้นและการเสร็จสิ้นของเวิร์กโฟลว์:** บันทึกเวลาเริ่มต้นและสิ้นสุดของอินสแตนซ์เวิร์กโฟลว์แต่ละตัว พร้อมด้วยเมตาดาตาที่เกี่ยวข้อง เช่น ข้อมูลนำเข้าและบริบทของผู้ใช้
2. **การดำเนินการของคอมโพเนนต์:** บันทึกรายละเอียดการดำเนินการของคอมโพเนนต์เวิร์กโฟลว์แต่ละตัว รวมถึงพารามิเตอร์นำเข้า ผลลัพธ์ที่ได้ และข้อมูลระหว่างกลางที่สร้างขึ้น
3. **การตัดสินใจและเหตุผลของ AI:** บันทึกการตัดสินใจที่ทำโดยคอมโพเนนต์ AI พร้อมด้วยเหตุผลหรือคะแนนความเชื่อมั่นที่เกี่ยวข้อง ซึ่งช่วยให้เกิดความโปร่งใสและช่วยให้สามารถตรวจสอบการตัดสินใจที่ขับเคลื่อนด้วย AI ได้
4. **ข้อยกเว้นและข้อความแสดงข้อผิดพลาด:** บันทึกข้อยกเว้นหรือข้อความแสดงข้อผิดพลาดที่พบระหว่างการดำเนินการเวิร์กโฟลว์ รวมถึง stack trace และข้อมูลบริบทที่เกี่ยวข้อง

การบันทึกข้อมูลสามารถทำได้โดยใช้เทคนิคต่างๆ เช่น การเขียนลงในไฟล์บันทึก การจัดเก็บบันทึกในฐานข้อมูล หรือการส่งบันทึกไปยังบริการบันทึกข้อมูลแบบรวมศูนย์ เป็นสิ่งสำคัญที่จะต้องเลือกเฟรมเวิร์กการบันทึกข้อมูลที่มีความยืดหยุ่น ขยายขนาดได้ และบูรณาการกับสถาปัตยกรรมของแอปพลิเคชันได้ง่าย

นี่คือตัวอย่างของวิธีการใช้การบันทึกข้อมูลในแอปพลิเคชัน Ruby on Rails โดยใช้คลาส ActiveSupport::Logger:

```ruby
class WorkflowLogger
  def self.log(message, severity = :info)
    @logger ||= ActiveSupport::Logger.new('workflow.log')
    @logger.formatter ||= proc do |severity, datetime, progname, msg|
      "#{datetime} [#{severity}] #{msg}\n"
    end
    @logger.send(severity, message)
  end
end

# Usage example
WorkflowLogger.log("Workflow initiated for order ##{@order.id}")
WorkflowLogger.log("Payment processing completed successfully")
WorkflowLogger.log("Inventory check failed for item ##{item.id}", :error)
```

การวางตำแหน่งคำสั่งบันทึกอย่างมีกลยุทธ์ตลอดทั้งองค์ประกอบของเวิร์กโฟลว์และจุดตัดสินใจของ AI ช่วยให้นักพัฒนาสามารถเก็บข้อมูลที่มีค่าสำหรับการแก้จุดบกพร่อง การตรวจสอบ และการวิเคราะห์

ประโยชน์ของการตรวจสอบและการบันทึก

การนำการตรวจสอบและการบันทึกมาใช้ในการจัดการลำดับงานอัจฉริยะมีประโยชน์หลายประการ:

1. **การ แก้ จุด บกพร่อง และ การ แก้ไข ปัญหา:** บันทึก โดย ละเอียด และ ข้อมูล การ ตรวจ สอบ ช่วย ให้ นัก พัฒนา ระบุ และ วินิจฉัย ปัญหา ได้ อย่าง รวดเร็ว ให้ ข้อมูล เชิง ลึก เกี่ยว กับ การ ทำงาน ของ เวิร์กโฟลว์ การ ทำงาน ร่วม กัน ของ องค์ ประกอบ และข้อผิดพลาดหรือข้อยกเว้นที่พบ

2. **การ ปรับ ประสิทธิภาพ:** การ ตรวจ สอบ เมตริก ประสิทธิภาพ ช่วย ให้ นัก พัฒนา ระบุ คอ ขวด และ ปรับ องค์ ประกอบ ของ เวิร์กโฟลว์ เพื่อ ประสิทธิภาพ ที่ ดี ขึ้น โดย การ วิเคราะห์ เวลา ใน การ ทำงาน การ ใช้ ทรัพยากร และ เม ตริ กอื่นๆ นักพัฒนาสามารถตัดสินใจอย่างมีข้อมูลเพื่อปรับปรุงประสิทธิภาพโดยรวมของระบบ

3. **การตรวจสอบและการปฏิบัติตามกฎระเบียบ:** การบันทึกเหตุการณ์และการตัดสินใจที่สำคัญช่วยสร้างบันทึกการตรวจสอบสำหรับการปฏิบัติตามกฎระเบียบและความรับผิดชอบ ช่วยให้องค์กรสามารถติดตามและตรวจสอบการดำเนินการของส่วนประกอบ AI และรับรองการปฏิบัติตามกฎทางธุรกิจและข้อกำหนดทางกฎหมาย

4. **การปรับปรุงอย่างต่อเนื่อง:** ข้อมูลการตรวจสอบและการบันทึกเป็นข้อมูลที่มีค่าสำหรับการปรับปรุงเวิร์กโฟลว์อัจฉริยะอย่างต่อเนื่อง การวิเคราะห์ข้อมูลในอดีต การระบุรูปแบบ และการวัดประสิทธิผลของการตัดสินใจของ AI ช่วยให้นักพัฒนาสามารถปรับปรุงและเพิ่มประสิทธิภาพตรรกะการจัดการลำดับงานได้อย่างต่อเนื่อง

ข้อควรพิจารณาและแนวทางปฏิบัติที่ดีที่สุด

เมื่อนำการตรวจสอบและการบันทึกมาใช้ในการจัดการลำดับงานอัจฉริยะ ควรพิจารณาแนวทางปฏิบัติที่ดีที่สุดต่อไปนี้:

1. **กำหนดเมตริกการตรวจสอบที่ชัดเจน:** ระบุเมตริกและเหตุการณ์สำคัญที่ต้องตรวจสอบตามความต้องการเฉพาะของเวิร์กโฟลว์ มุ่งเน้นที่เมตริกที่ให้ข้อมูลเชิงลึกที่มีความหมายเกี่ยวกับประสิทธิภาพ สุขภาพ และพฤติกรรมของระบบ

2. **การบันทึกแบบละเอียด:** ตรวจสอบให้แน่ใจว่ามีการวางคำสั่งบันทึกไว้ในจุดที่เหมาะสมภายในองค์ประกอบของเวิร์กโฟลว์และจุดตัดสินใจของ AI บันทึกข้อมูลบริบทที่เกี่ยวข้อง เช่น พารามิเตอร์อินพุต ผลลัพธ์เอาต์พุต และข้อมูลระหว่างกลางที่สร้างขึ้น

3. **ใช้การบันทึกแบบมีโครงสร้าง:** นำรูปแบบการบันทึกที่มีโครงสร้างมาใช้เพื่อให้ง่ายต่อการแยกวิเคราะห์และวิเคราะห์ข้อมูลบันทึก การบันทึกแบบมีโครงสร้างช่วยให้ค้นหา กรอง และรวมรายการบันทึกได้ดีขึ้น

4. **จัดการการเก็บรักษาและการหมุนเวียนบันทึก:** นำนโยบายการเก็บรักษาและการหมุนเวียนบันทึกมาใช้เพื่อจัดการการจัดเก็บและวงจรชีวิตของไฟล์บันทึก กำหนดระยะเวลาการเก็บรักษาที่เหมาะสมตามข้อกำหนดทางกฎหมาย ข้อจำกัดด้านการจัดเก็บ และความต้องการในการวิเคราะห์ หากเป็นไปได้ ให้ย้ายการบันทึกไปยังบริการของบุคคลที่สามเช่น Papertrail

5. **รักษาความปลอดภัยข้อมูลที่ละเอียดอ่อน:** ระมัดระวังเมื่อบันทึกข้อมูลที่ละเอียดอ่อน เช่น ข้อมูลที่ระบุตัวตนบุคคล (PII) หรือข้อมูลธุรกิจที่เป็นความลับ นำมาตรการรักษาความปลอดภัยที่เหมาะสมมาใช้ เช่น การปกปิดข้อมูลหรือการเข้ารหัส เพื่อปกป้องข้อมูลที่ละเอียดอ่อนในไฟล์บันทึก

6. **ผสานรวมกับเครื่องมือตรวจสอบและแจ้งเตือน:** ใช้ประโยชน์จากเครื่องมือตรวจสอบและแจ้งเตือนเพื่อรวมศูนย์การเก็บรวบรวม วิเคราะห์ และแสดงผลข้อมูลการตรวจสอบและการบันทึก เครื่องมือเหล่านี้สามารถให้ข้อมูลเชิงลึกแบบเรียลไทม์ สร้างการแจ้งเตือนตามเกณฑ์ที่กำหนดไว้ล่วงหน้า และช่วยในการตรวจจับและแก้ไขปัญหาเชิงรุก เครื่องมือที่ผมชื่นชอบที่สุดคือ Datadog

การนำกลไกการตรวจสอบและการบันทึกที่ครอบคลุมมาใช้ ช่วยให้นักพัฒนาได้รับข้อมูลเชิงลึกที่มีค่าเกี่ยวกับพฤติกรรมและประสิทธิภาพของเวิร์กโฟลว์อัจฉริยะ ข้อมูลเชิงลึกเหล่านี้ช่วยให้สามารถแก้จุดบกพร่อง ปรับประสิทธิภาพ และปรับปรุงระบบการจัดการลำดับงานที่ขับเคลื่อนด้วย AI อย่างต่อเนื่อง

ข้อควรพิจารณาด้านความสามารถในการปรับขนาดและประสิทธิภาพ

ความสามารถในการปรับขนาดและประสิทธิภาพเป็นแง่มุมสำคัญที่ต้องพิจารณาเมื่อออกแบบและนำระบบการจัดการลำดับงานอัจฉริยะไปใช้ เมื่อปริมาณของเวิร์กโฟลว์ที่ทำงานพร้อมกันและความซับซ้อนขององค์ประกอบที่ขับเคลื่อนด้วย AI เพิ่มขึ้น จึงจำเป็นต้องตรวจ

สอบให้แน่ใจว่าระบบสามารถจัดการกับภาระงานได้อย่างมีประสิทธิภาพและปรับขนาดได้อย่างราบรื่นเพื่อตอบสนองความต้องการที่เพิ่มขึ้น

การจัดการเวิร์กโฟลว์จำนวนมากที่ทำงานพร้อมกัน

ระบบการจัดการลำดับงานอัจฉริยะมักต้องจัดการกับเวิร์กโฟลว์จำนวนมากที่ทำงานพร้อมกัน เพื่อให้แน่ใจว่าสามารถปรับขนาดได้ ควรพิจารณากลยุทธ์ต่อไปนี้:

1. **การประมวลผลแบบไม่ประสานเวลา:** นำกลไกการประมวลผลแบบไม่ประสานเวลามาใช้เพื่อแยกการทำงานขององค์ประกอบเวิร์กโฟลว์ ช่วยให้ระบบสามารถจัดการกับเวิร์กโฟลว์หลายรายการพร้อมกันได้โดยไม่ต้องบล็อกหรือรอให้แต่ละองค์ประกอบทำงานเสร็จ การประมวลผลแบบไม่ประสานเวลาสามารถทำได้โดยใช้คิวข้อความ สถาปัตยกรรมแบบขับเคลื่อนด้วยเหตุการณ์ หรือเฟรมเวิร์กการประมวลผลงานเบื้องหลังเช่น Sidekiq

2. **สถาปัตยกรรมแบบกระจาย:** ออกแบบสถาปัตยกรรมระบบให้ใช้คอมโพเนนต์แบบเซิร์ฟเวอร์เลส (เช่น AWS Lambda) หรือเพียงแค่กระจายภาระงานไปยังโหนดหรือเซิร์ฟเวอร์หลายเครื่องพร้อมกับเซิร์ฟเวอร์แอปพลิเคชันหลักของคุณ ช่วยให้สามารถปรับขนาดแนวนอนได้ โดยสามารถเพิ่มโหนดเพื่อรองรับปริมาณเวิร์กโฟลว์ที่เพิ่มขึ้น

3. **การทำงานแบบขนาน:** ระบุโอกาสในการทำงานแบบขนานภายในเวิร์กโฟลว์ องค์ประกอบเวิร์กโฟลว์บางส่วนอาจเป็นอิสระจากกันและสามารถทำงานพร้อมกันได้ การใช้เทคนิคการประมวลผลแบบขนาน เช่น มัลติเธรดดิ้งหรือคิวงานแบบกระจาย ระบบสามารถเพิ่มประสิทธิภาพการใช้ทรัพยากรและลดเวลาการทำงานของเวิร์กโฟลว์โดยรวม

การเพิ่มประสิทธิภาพของคอมโพเนนต์ที่ขับเคลื่อนด้วย AI

คอมโพเนนต์ที่ขับเคลื่อนด้วย AI เช่น โมเดลการเรียนรู้ของเครื่อง หรือเอนจินประมวลผลภาษาธรรมชาติ อาจต้องใช้ทรัพยากรการคำนวณสูงและส่งผลกระทบต่อประสิทธิภาพโดยรวมของระบบการจัดการลำดับงาน เพื่อเพิ่มประสิทธิภาพของคอมโพเนนต์ AI ควรพิจารณาเทคนิคต่อไปนี้:

1. **การแคช:** หากการประมวลผล AI ของคุณเป็นเพียงการสร้างผลลัพธ์และไม่เกี่ยวข้องกับการค้นหาข้อมูลแบบเรียลไทม์หรือการเชื่อมต่อภายนอกเพื่อสร้างการสนทนา คุณสามารถพิจารณาใช้กลไกการแคชเพื่อจัดเก็บและนำผลลัพธ์ที่เข้าถึงบ่อยหรือการดำเนินการที่ใช้ทรัพยากรสูง

2. **การเพิ่มประสิทธิภาพโมเดล:** ปรับปรุงวิธีการใช้โมเดล AI ในคอมโพเนนต์ของเวิร์กโฟลว์อย่างต่อเนื่อง อาจรวมถึงเทคนิคต่างๆ เช่น *การกลั่นกรองพรอมต์* หรืออาจเป็นเพียงการทดสอบโมเดลใหม่ๆ ที่มีให้ใช้งาน

3. **การ ประมวล ผล แบบ กลุ่ม:** หาก คุณ กำลัง ทำงาน กับ โมเดล ระดับ GPT-4 คุณ อาจสามารถ ใช้ ประโยชน์ จาก เทคนิค การ ประมวล ผล แบบ กลุ่ม เพื่อ ประมวล ผล ข้อมูล หลาย จุด หรือ หลาย คำขอ ใน ครั้ง เดียว แทนที่ จะ ประมวล ผล ที ละ รายการ

การประมวลผลข้อมูลเป็นกลุ่ม ช่วยให้ระบบสามารถเพิ่มประสิทธิภาพการใช้ทรัพยากรและลดค่าโสหุ้ยของการร้องขอโมเดลซ้ำๆ

การติดตามและวิเคราะห์ประสิทธิภาพ

เพื่อระบุคอขวดด้านประสิทธิภาพและเพิ่มความสามารถในการปรับขนาดของระบบการจัดการลำดับงานอัจฉริยะ จำเป็นต้องใช้กลไกการติดตามและวิเคราะห์ประสิทธิภาพ พิจารณาแนวทางต่อไปนี้:

1. **ตัวชี้วัดประสิทธิภาพ:** กำหนดและติดตามตัวชี้วัดประสิทธิภาพที่สำคัญ เช่น เวลาตอบสนอง ปริมาณงานที่ทำได้ การใช้ทรัพยากร และความล่าช้า ตัวชี้วัดเหล่านี้ให้ข้อมูลเชิงลึกเกี่ยวกับประสิทธิภาพของระบบและช่วยระบุพื้นที่ที่ต้องปรับปรุง ตัวรวมโมเดล AI ยอดนิยมอย่าง OpenRouter มีการรวมตัวชี้วัด Host[1] และ Speed[2] ในการตอบสนอง API แต่ละครั้ง ทำให้การติดตามตัวชี้วัดที่สำคัญเหล่านี้ทำได้ง่าย

2. **เครื่องมือวิเคราะห์ประสิทธิภาพ:** ใช้เครื่องมือวิเคราะห์ประสิทธิภาพเพื่อวิเคราะห์ประสิทธิภาพของคอมโพเนนต์เวิร์กโฟลว์และการทำงานของ AI แต่ละส่วน เครื่องมือเหล่านี้ช่วยระบุจุดที่มีปัญหาด้านประสิทธิภาพ เส้นทางโค้ดที่ไม่มีประสิทธิภาพ หรือการทำงานที่ใช้ทรัพยากรสูง เครื่องมือวิเคราะห์ประสิทธิภาพยอดนิยมได้แก่ New Relic, Scout หรือเครื่องมือวิเคราะห์ที่มีมาให้ในภาษาโปรแกรมหรือเฟรมเวิร์ก

3. **การทดสอบภาระงาน:** ดำเนินการทดสอบภาระงานเพื่อประเมินประสิทธิภาพของระบบภายใต้ระดับภาระงานพร้อมกันที่แตกต่างกัน การทดสอบภาระงานช่วยระบุขีดจำกัดในการปรับขนาดของระบบ ตรวจจับการลดลงของประสิทธิภาพ และทำให้มั่นใจว่าระบบสามารถรองรับปริมาณการใช้งานที่คาดการณ์ไว้โดยไม่ส่งผลกระทบต่อประสิทธิภาพ

4. **การติดตามอย่างต่อเนื่อง:** ใช้กลไกการติดตามและแจ้งเตือนอย่างต่อเนื่องเพื่อตรวจจับปัญหาด้านประสิทธิภาพและคอขวดเชิงรุก ตั้งค่าแดชบอร์ดการติดตามและการแจ้งเตือนเพื่อติดตามตัวชี้วัดประสิทธิภาพที่สำคัญ (KPIs) และรับการแจ้งเตือนเมื่อเกินเกณฑ์ที่กำหนดไว้ล่วงหน้า วิธีนี้ช่วยให้สามารถระบุและแก้ไขปัญหาด้านประสิทธิภาพได้อย่างรวดเร็ว

กลยุทธ์การปรับขนาด

เพื่อรองรับภาระงานที่เพิ่มขึ้นและทำให้มั่นใจในความสามารถในการปรับขนาดของระบบการจัดการลำดับงานอัจฉริยะ ควรพิจารณากลยุทธ์การปรับขนาดต่อไปนี้:

[1] Host คือเวลาที่ใช้ในการรับไบต์แรกของการสร้างผลลัพธ์แบบสตรีมจากโฮสต์ของโมเดล หรือที่เรียกว่า “time to first byte”

[2] Speed คำนวณจากจำนวนโทเค็นที่สร้างหารด้วยเวลาที่ใช้ในการสร้างทั้งหมด สำหรับคำขอที่ไม่ใช่แบบสตรีม ความล่าช้าจะถูกนับรวมเป็นส่วนหนึ่งของเวลาในการสร้าง

1. การปรับขนาดแนวตั้ง: การปรับขนาดแนวตั้งเกี่ยวข้องกับการเพิ่มทรัพยากร (เช่น CPU หน่วยความจำ) ของโหนดหรือเซิร์ฟเวอร์แต่ละตัวเพื่อรองรับภาระงานที่สูงขึ้น แนวทางนี้เหมาะสมเมื่อระบบต้องการกำลังประมวลผลหรือหน่วยความจำเพิ่มเติมเพื่อจัดการกับเวิร์กโฟลว์ที่ซับซ้อนหรือการทำงานของ AI

2. การปรับขนาดแนวนอน: การปรับขนาดแนวนอนเกี่ยวข้องกับการเพิ่มโหนดหรือเซิร์ฟเวอร์ให้กับระบบเพื่อกระจายภาระงาน แนวทางนี้มีประสิทธิภาพเมื่อระบบต้องจัดการกับเวิร์กโฟลว์พร้อมกันจำนวนมาก หรือเมื่อสามารถกระจายภาระงานไปยังโหนดหลายๆ ตัวได้อย่างง่ายดาย การปรับขนาดแนวนอนต้องการสถาปัตยกรรมแบบกระจายและกลไกการกระจายภาระงานเพื่อให้มั่นใจว่ามีการกระจายการจราจรอย่างสม่ำเสมอ

3. การปรับขนาดอัตโนมัติ: ใช้กลไกการปรับขนาดอัตโนมัติเพื่อปรับจำนวนโหนดหรือทรัพยากรโดยอัตโนมัติตามความต้องการของภาระงาน การปรับขนาดอัตโนมัติช่วยให้ระบบสามารถปรับขนาดขึ้นหรือลงได้อย่างไดนามิกตามปริมาณการใช้งานที่เข้ามา ทำให้มั่นใจได้ว่ามีการใช้ทรัพยากรอย่างเหมาะสมและมีประสิทธิภาพด้านต้นทุน แพลตฟอร์มคลาวด์อย่าง Amazon Web Services (AWS) หรือ Google Cloud Platform (GCP) มีความสามารถในการปรับขนาดอัตโนมัติที่สามารถนำมาใช้กับระบบการจัดการลำดับงานอัจฉริยะได้

เทคนิคการเพิ่มประสิทธิภาพ

นอกเหนือจากกลยุทธ์การปรับขนาด ควรพิจารณาเทคนิคการเพิ่มประสิทธิภาพต่อไปนี้เพื่อเพิ่มประสิทธิภาพของระบบการจัดการลำดับงานอัจฉริยะ:

1. การจัดเก็บและการเรียกคืนข้อมูลอย่างมีประสิทธิภาพ: เพิ่มประสิทธิภาพกลไกการจัดเก็บและเรียกคืนข้อมูลที่ใช้โดยคอมโพเนนต์เวิร์กโฟลว์ ใช้การทำดัชนีฐานข้อมูลที่มีประสิทธิภาพ เทคนิคการเพิ่มประสิทธิภาพการสืบค้น และการแคชข้อมูลเพื่อลดความล่าช้าและปรับปรุงประสิทธิภาพของการทำงานที่ใช้ข้อมูลเข้มข้น

2. การรับส่งข้อมูลแบบไม่ประสานเวลา: ใช้การดำเนินการรับส่งข้อมูลแบบไม่ประสานเวลาเพื่อป้องกันการติดขัดและปรับปรุงการตอบสนองของระบบ การรับส่งข้อมูลแบบไม่ประสานเวลาช่วยให้ระบบสามารถจัดการกับคำขอหลายรายการพร้อมกันได้โดยไม่ต้องรอให้การดำเนินการรับส่งข้อมูลเสร็จสิ้น ทำให้การใช้ทรัพยากรเป็นไปอย่างมีประสิทธิภาพสูงสุด

3. การแปลงข้อมูลเป็นอนุกรมและการแปลงอนุกรมกลับเป็นข้อมูลอย่างมีประสิทธิภาพ: ปรับปรุงกระบวนการแปลงข้อมูลเป็นอนุกรมและการแปลงอนุกรมกลับเป็นข้อมูลที่ใช้ในการแลกเปลี่ยนข้อมูลระหว่างองค์ประกอบของเวิร์กโฟลว์ ใช้รูปแบบการแปลงข้อมูลเป็นอนุกรมที่มีประสิทธิภาพ เช่น Protocol Buffers หรือ MessagePack เพื่อลดภาระในการแปลงข้อมูลเป็นอนุกรมและปรับปรุงประสิทธิภาพการสื่อสารระหว่างองค์ประกอบ

สำหรับแอปพลิเคชันที่ใช้ Ruby ควรพิจารณาใช้ Universal ID Universal ID ใช้ประโยชน์จากทั้ง MessagePack และ Brotli (การผสมผสานที่สร้างขึ้นเพื่อความเร็วและการบีบอัดข้อมูลที่ดีที่สุด) เมื่อใช้ร่วมกัน ไลบรารีเหล่านี้เร็วกว่าถึง 30% และมีอัตราการบีบอัดใกล้เคียงกับ Protocol Buffers ที่ 2-5%

4. การบีบอัดและการเข้ารหัส: ใช้เทคนิคการบีบอัดและการเข้ารหัสเพื่อลดขนาดข้อมูลที่ส่งระหว่างองค์ประกอบของเวิร์กโฟลว์ อัลกอริทึมการบีบอัด เช่น gzip หรือ Brotli สามารถลดการใช้แบนด์วิดธ์เครือข่ายได้อย่างมีนัยสำคัญและปรับปรุงประสิทธิภาพโดยรวมของระบบ

การพิจารณาด้านความสามารถในการขยายตัวและประสิทธิภาพระหว่างการออกแบบและการพัฒนาระบบการจัดระเบียบลำดับงานอัจฉริยะช่วยให้มั่นใจได้ว่าระบบของคุณสามารถจัดการกับเวิร์กโฟลว์ที่ทำงานพร้อมกันใน ปริมาณ มาก ปรับปรุง ประสิทธิภาพ ของ องค์ประกอบที่ขับเคลื่อนด้วย AI และขยายตัวได้อย่างราบรื่นเพื่อตอบสนองความต้องการที่เพิ่มขึ้น การตรวจสอบ การวิเคราะห์ประสิทธิภาพ และการปรับปรุงอย่างต่อเนื่องมีความสำคัญในการรักษาประสิทธิภาพและการตอบสนองของระบบเมื่อภาระงานและความซับซ้อนเพิ่มขึ้นตามกาลเวลา

การทดสอบและการตรวจสอบความถูกต้องของเวิร์กโฟลว์

การ ทดสอบ และ การ ตรวจ สอบ ความ ถูก ต้อง เป็น แง่ มุม ที่ สำคัญ ใน การ พัฒนา และ ดูแล รักษา ระบบ การ จัด ระเบียบ ลำดับ งาน อัจฉริยะ เนื่องจากลักษณะที่ซับซ้อนของเวิร์กโฟลว์ที่ขับเคลื่อนด้วย AI จึงจำเป็นต้องมั่นใจว่าแต่ละองค์ประกอบทำงานตามที่คาดหวัง เวิร์กโฟลว์โดยรวมทำงานได้อย่างถูกต้อง และการตัดสินใจของ AI มีความแม่นยำและน่าเชื่อถือ ในส่วนนี้ เราจะสำรวจเทคนิคและข้อควรพิจารณาต่างๆ สำหรับการทดสอบและตรวจสอบความถูกต้องของเวิร์กโฟลว์อัจฉริยะ

การทดสอบระดับหน่วยขององค์ประกอบเวิร์กโฟลว์

การทดสอบระดับหน่วยเกี่ยวข้องกับการทดสอบองค์ประกอบเวิร์กโฟลว์แต่ละส่วนแยกกันเพื่อตรวจสอบความถูกต้องและความทนทาน เมื่อทำการทดสอบระดับหน่วยขององค์ประกอบเวิร์กโฟลว์ที่ขับเคลื่อนด้วย AI ควรพิจารณาสิ่งต่อไปนี้:

1. **การ ตรวจ สอบ ข้อมูล นำ เข้า:** ทดสอบ ความ สามารถ ของ องค์ ประกอบ ใน การ จัดการ กับ ข้อมูล นำ เข้า ประเภท ต่างๆ รวมถึงข้อมูลที่ถูกต้องและไม่ถูกต้อง ตรวจสอบว่าองค์ประกอบสามารถจัดการกับกรณีพิเศษได้อย่างเหมาะสมและให้ข้อความแสดงข้อผิดพลาดหรือข้อยกเว้นที่เหมาะสม
2. **การ ตรวจ สอบ ผลลัพธ์:** ยืนยัน ว่า องค์ ประกอบ สร้าง ผลลัพธ์ ที่ คาด หวัง สำหรับ ชุด ข้อมูล นำ เข้า ที่ กำหนด เปรียบเทียบผลลัพธ์จริงกับผลลัพธ์ที่คาดหวังเพื่อให้แน่ใจว่าถูกต้อง

3. การ จัดการ ข้อ ผิด พลาด: ทดสอบ กลไก การ จัดการ ข้อ ผิด พลาด ของ องค์ ประกอบ โดย จำลอง สถานการณ์ ข้อ ผิด พลาด ต่างๆ เช่น ข้อมูล นำ เข้า ไม่ ถูก ต้อง ทรัพยากร ไม่ พร้อม ใช้ งาน หรือ ข้อ ยกเว้น ที่ ไม่ คาด คิด ตรวจสอบว่าองค์ประกอบสามารถจับและจัดการข้อผิดพลาดได้อย่างเหมาะสม

4. เงื่อนไขขอบเขต: ทดสอบพฤติกรรมขององค์ประกอบภายใต้เงื่อนไขขอบเขต เช่น ข้อมูลนำเข้าว่างเปล่า ขนาดข้อมูลนำเข้าสูงสุด หรือค่าที่สุดขีด ตรวจสอบให้แน่ใจว่าองค์ประกอบสามารถจัดการกับเงื่อนไขเหล่านี้ได้อย่างเหมาะสมโดยไม่เกิดข้อผิดพลาดหรือให้ผลลัพธ์ที่ไม่ถูกต้อง

ต่อไปนี้เป็นตัวอย่างการทดสอบระดับหน่วยสำหรับองค์ประกอบเวิร์กโฟลว์ใน Ruby โดยใช้เฟรมเวิร์กการทดสอบ RSpec:

```ruby
RSpec.describe OrderValidator do
  describe '#validate' do
    context 'when order is valid' do
      let(:order) { build(:order) }

      it 'returns true' do
        expect(subject.validate(order)).to be true
      end
    end

    context 'when order is invalid' do
      let(:order) { build(:order, total_amount: -100) }

      it 'returns false' do
        expect(subject.validate(order)).to be false
      end
    end
  end
end
```

ในตัวอย่างนี้ คอมโพเนนต์ OrderValidator ถูกทดสอบโดยใช้เคสทดสอบสองเคส: หนึ่งสำหรับคำสั่งที่ถูกต้อง และอีกหนึ่งสำหรับคำสั่งที่ไม่ถูกต้อง เคสทดสอบเหล่านี้ตรวจสอบว่าเมธอด validate ส่งค่าบูลีนที่คาดหวังกลับมาตามความถูกต้องของคำสั่ง

การทดสอบการทำงานร่วมกันของลำดับการทำงาน

การทดสอบการทำงานร่วมกัน มุ่งเน้นที่การตรวจสอบการทำงานร่วมกันและการไหลของข้อมูลระหว่างคอมโพเนนต์ต่างๆ ในลำดับการทำงาน เพื่อให้มั่นใจว่าคอมโพเนนต์ต่างๆ ทำงานร่วมกันได้อย่างราบรื่นและให้ผลลัพธ์ตามที่คาดหวัง เมื่อทำการทดสอบการทำงานร่วมกันของลำดับการทำงานอัจฉริยะ ควรพิจารณาสิ่งต่อไปนี้:

1. **การทำงานร่วมกันระหว่างคอมโพเนนต์:** ทดสอบการสื่อสารและการแลกเปลี่ยนข้อมูลระหว่างคอมโพเนนต์ในลำดับการทำงาน ตรวจสอบว่าผลลัพธ์จากคอมโพเนนต์หนึ่งถูกส่งต่อเป็นข้อมูลนำเข้าให้กับคอมโพเนนต์ถัดไปในลำดับการทำงานอย่างถูกต้อง

2. **ความสอดคล้องของข้อมูล:** ตรวจสอบให้แน่ใจว่าข้อมูลยังคงมีความสอดคล้องและถูกต้องขณะที่ไหลผ่านลำดับการทำงาน ตรวจสอบว่าการแปลงข้อมูล การคำนวณ และการรวบรวมข้อมูลดำเนินการอย่างถูกต้อง

3. **การส่งต่อข้อยกเว้น:** ทดสอบวิธีการส่งต่อและจัดการข้อยกเว้นและข้อผิดพลาดระหว่างคอมโพเนนต์ในลำดับการทำงาน ตรวจสอบว่าข้อยกเว้นถูกจับ บันทึก และจัดการอย่างเหมาะสมเพื่อป้องกันการหยุดชะงักของลำดับการทำงาน

4. **พฤติกรรมการทำงานแบบอะซิงโครนัส:** หากลำดับการทำงานมีคอมโพเนนต์แบบอะซิงโครนัสหรือการทำงานแบบขนาน ให้ทดสอบกลไกการประสานงานและการซิงโครไนซ์ ตรวจสอบให้แน่ใจว่าลำดับการทำงานทำงานได้อย่างถูกต้องภายใต้สถานการณ์ที่มีการทำงานพร้อมกันและแบบอะซิงโครนัส

ต่อไปนี้เป็นตัวอย่างของการทดสอบการทำงานร่วมกันสำหรับลำดับการทำงานใน Ruby โดยใช้เฟรมเวิร์กการทดสอบ RSpec:

```ruby
RSpec.describe OrderProcessingWorkflow do

  let(:order) { build(:order) }

  it 'processes the order successfully' do
    expect(OrderValidator).to receive(:validate).and_return(true)
    expect(InventoryManager).to receive(:check_availability).and_return(true)
    expect(PaymentProcessor).to receive(:process_payment).and_return(true)
    expect(ShippingService).to receive(:schedule_shipping).and_return(true)

    workflow = OrderProcessingWorkflow.new(order)
    result = workflow.process

    expect(result).to be true
    expect(order.status).to eq('processed')
  end

end
```

ในตัวอย่างนี้ OrderProcessingWorkflow ถูกทดสอบด้วยการตรวจสอบการทำงานร่วมกันระหว่างองค์ประกอบต่างๆ ของเวิร์กโฟลว์ กรณีทดสอบจะตั้งค่าความคาดหวังสำหรับพฤติกรรมของแต่ละองค์ประกอบและทำให้มั่นใจว่าเวิร์กโฟลว์ประมวลผลคำสั่งซื้อสำเร็จ พร้อมทั้งอัปเดตสถานะคำสั่งซื้อตามที่ควร

การทดสอบจุดตัดสินใจของ AI

การทดสอบจุดตัดสินใจของ AI มีความสำคัญอย่างยิ่งในการรับรองความแม่นยำและความน่าเชื่อถือของเวิร์กโฟลว์ที่ขับเคลื่อนด้วย AI เมื่อทำการทดสอบจุดตัดสินใจของ AI ควรพิจารณาประเด็นต่อไปนี้:

1. **ความแม่นยำในการตัดสินใจ:** ตรวจสอบว่าองค์ประกอบ AI ตัดสินใจได้แม่นยำตามข้อมูลที่ป้อนเข้าและโมเดลที่ผ่านการฝึกฝน เปรียบเทียบการตัดสินใจของ AI กับผลลัพธ์ที่คาดหวังหรือข้อมูลความจริงพื้นฐาน

2. **กรณีขอบเขต:** ทดสอบพฤติกรรมขององค์ประกอบ AI ภายใต้กรณีขอบเขตและสถานการณ์ที่ไม่ปกติ ตรวจสอบว่าองค์ประกอบ AI จัดการกับกรณีเหล่านี้ได้อย่างราบรื่นและตัดสินใจได้อย่างสมเหตุสมผล

3. **อคติและความเป็นธรรม:** ประเมินองค์ประกอบ AI เพื่อหาอคติที่อาจเกิดขึ้นและทำให้มั่นใจว่ามีการตัดสินใจอย่างเป็นธรรมและปราศจากอคติ ทดสอบองค์ประกอบด้วยข้อมูลที่หลากหลายและวิเคราะห์ผลลัพธ์เพื่อหารูปแบบการเลือกปฏิบัติที่อาจเกิดขึ้น

4. **ความสามารถในการอธิบายได้:** หากองค์ประกอบ AI ให้คำอธิบายหรือเหตุผลสำหรับการตัดสินใจ ให้ตรวจสอบความถูกต้องและความชัดเจนของคำอธิบาย ทำให้มั่นใจว่าคำอธิบายสอดคล้องกับกระบวนการตัดสินใจที่เกิดขึ้นจริง

ต่อไปนี้เป็นตัวอย่างการทดสอบจุดตัดสินใจของ AI ในภาษา Ruby โดยใช้เฟรมเวิร์กการทดสอบ RSpec:

```ruby
RSpec.describe FraudDetector do
  describe '#detect_fraud' do
    context 'when transaction is fraudulent' do
      let(:tx) do
        build(:transaction, amount: 10_000, location: 'High-Risk Country')
      end

      it 'returns true' do
        expect(subject.detect_fraud(tx)).to be true
      end
    end

    context 'when transaction is legitimate' do
      let(:tx) do
        build(:transaction, amount: 100, location: 'Low-Risk Country')
      end

      it 'returns false' do
        expect(subject.detect_fraud(tx)).to be false
      end
    end
```

```
  end
end
```

ในตัวอย่างนี้ คอมโพเนนต์ AI FraudDetector ถูกทดสอบด้วยเคสทดสอบสองเคส: หนึ่งสำหรับธุรกรรมที่เป็นการฉ้อโกง และอีกหนึ่งสำหรับธุรกรรมที่ถูกต้อง เคสทดสอบเหล่านี้ตรวจสอบว่าเมธอด detect_fraud ส่งคืนค่าบูลีนที่คาดหวังตามลักษณะของธุรกรรม

การทดสอบแบบครบวงจร

การทดสอบแบบครบวงจร เกี่ยวข้องกับการทดสอบกระบวนการทำงานทั้งหมดตั้งแต่ต้นจนจบ โดยจำลองสถานการณ์การใช้งานจริงและการโต้ตอบของผู้ใช้ การทดสอบนี้ช่วยให้มั่นใจว่ากระบวนการทำงานทำงานได้อย่างถูกต้องและให้ผลลัพธ์ตามที่ต้องการ เมื่อทำการทดสอบแบบครบวงจรสำหรับกระบวนการทำงานอัจฉริยะ ควรพิจารณาสิ่งต่อไปนี้:

1. สถานการณ์การใช้งานของผู้ใช้: ระบุสถานการณ์การใช้งานทั่วไปของผู้ใช้และทดสอบพฤติกรรมของกระบวนการทำงานภายใต้สถานการณ์เหล่านี้ ตรวจสอบว่ากระบวนการทำงานจัดการกับข้อมูลนำเข้าของผู้ใช้ได้อย่างถูกต้อง ตัดสินใจได้อย่างเหมาะสม และให้ผลลัพธ์ตามที่คาดหวัง

2. การตรวจสอบความถูกต้องของข้อมูล: ตรวจสอบให้แน่ใจว่ากระบวนการทำงานมีการตรวจสอบและทำความสะอาดข้อมูลนำเข้าของผู้ใช้เพื่อป้องกันความไม่สอดคล้องของข้อมูลหรือช่องโหว่ด้านความปลอดภัย ทดสอบกระบวนการทำงานด้วยข้อมูลนำเข้าประเภทต่างๆ รวมถึงข้อมูลที่ถูกต้องและไม่ถูกต้อง

3. การกู้คืนจากข้อผิดพลาด: ทดสอบความสามารถของกระบวนการทำงานในการกู้คืนจากข้อผิดพลาดและข้อยกเว้นต่างๆ จำลองสถานการณ์ข้อผิดพลาดและตรวจสอบว่ากระบวนการทำงานจัดการกับสถานการณ์เหล่านั้นได้อย่างเหมาะสม บันทึกข้อผิดพลาด และดำเนินการกู้คืนที่เหมาะสม

4. ประสิทธิภาพและความสามารถในการขยาย: ประเมินประสิทธิภาพและความสามารถในการขยายของกระบวนการทำงานภายใต้สภาวะโหลดที่แตกต่างกัน ทดสอบกระบวนการทำงานด้วยคำขอพร้อมกันจำนวนมาก และวัดเวลาตอบสนอง การใช้ทรัพยากร และความเสถียรของระบบโดยรวม

ต่อไปนี้เป็นตัวอย่างของการทดสอบแบบครบวงจรสำหรับกระบวนการทำงานใน Ruby โดยใช้เฟรมเวิร์กการทดสอบ RSpec และไลบรารี Capybara สำหรับการจำลองการโต้ตอบของผู้ใช้:

```
RSpec.describe 'Order Processing Workflow' do
  scenario 'User places an order successfully' do
    visit '/orders/new'
    fill_in 'Product', with: 'Sample Product'
    fill_in 'Quantity', with: '2'
    fill_in 'Shipping Address', with: '123 Main St'
    click_button 'Place Order'

    expect(page).to have_content('Order Placed Successfully')
    expect(Order.count).to eq(1)
    expect(Order.last.status).to eq('processed')
  end
end
```

ในตัวอย่างนี้ การทดสอบแบบจากต้นจนจบ จำลองการที่ผู้ใช้สั่งซื้อสินค้าผ่านหน้าเว็บ โดยกรอกข้อมูลในฟอร์มที่จำเป็น ส่งคำสั่งซื้อ และตรวจสอบว่าคำสั่งซื้อได้รับการประมวลผลสำเร็จ แสดงข้อความยืนยันที่เหมาะสม และอัปเดตสถานะคำสั่งซื้อในฐานข้อมูล

การรวมและการปรับใช้งานอย่างต่อเนื่อง

เพื่อให้มั่นใจในความน่าเชื่อถือและความสามารถในการบำรุงรักษาของขั้นตอนการทำงานอัจฉริยะ แนะนำให้รวมการทดสอบและการตรวจสอบเข้ากับไปป์ไลน์การรวมและการปรับใช้งานอย่างต่อเนื่อง (CI/CD) ซึ่งช่วยให้สามารถทดสอบและตรวจสอบการเปลี่ยนแปลงของขั้นตอนการทำงานโดยอัตโนมัติก่อนที่จะนำไปใช้งานจริง พิจารณาแนวทางปฏิบัติดังต่อไปนี้:

1. **การดำเนินการทดสอบอัตโนมัติ:** กำหนดค่าไปป์ไลน์ CI/CD ให้รันชุดการทดสอบโดยอัตโนมัติเมื่อมีการเปลี่ยนแปลงในโค้ดของขั้นตอนการทำงาน วิธีนี้ช่วยให้ตรวจพบข้อบกพร่องหรือความล้มเหลวได้ตั้งแต่ช่วงแรกของกระบวนการพัฒนา

2. **การติดตามความครอบคลุมของการทดสอบ:** วัดและติดตามความครอบคลุมของการทดสอบในส่วนประกอบของขั้นตอนการทำงานและจุดตัดสินใจของ AI มุ่งเน้นให้มีความครอบคลุมของการทดสอบสูงเพื่อให้มั่นใจว่าเส้นทางและสถานการณ์ที่สำคัญได้รับการทดสอบอย่างละเอียด

3. **การตอบกลับอย่างต่อเนื่อง:** รวมผลการทดสอบและตัวชี้วัดคุณภาพโค้ดเข้ากับขั้นตอนการพัฒนา ให้ข้อมูลตอบกลับอย่างต่อเนื่องแก่นักพัฒนาเกี่ยวกับสถานะของการทดสอบ คุณภาพโค้ด และปัญหาที่ตรวจพบระหว่างกระบวนการ CI/CD

4. **สภาพแวดล้อมทดสอบ:** ปรับใช้ขั้นตอนการทำงานในสภาพแวดล้อมทดสอบที่เลียนแบบสภาพแวดล้อมการใช้งานจริงอย่างใกล้เคียง ดำเนินการทดสอบและตรวจสอบเพิ่มเติมในสภาพแวดล้อมทดสอบเพื่อค้นหาปัญหาที่เกี่ยวข้องกับโครงสร้างพื้นฐาน การกำหนดค่า หรือการรวมข้อมูล

5. กลไกการย้อนกลับ: ติดตั้งกลไกการย้อนกลับสำหรับกรณีที่การปรับใช้งานล้มเหลวหรือตรวจพบปัญหาสำคัญในระบบที่ใช้งานจริง ต้องมั่นใจว่าขั้นตอนการทำงานสามารถย้อนกลับไปยังเวอร์ชันที่เสถียรก่อนหน้าได้อย่างรวดเร็วเพื่อลดเวลาหยุดทำงานและผลกระทบต่อผู้ใช้

การรวมการทดสอบและการตรวจสอบตลอดวงจรการพัฒนาของขั้นตอนการทำงานอัจฉริยะ ช่วยให้องค์กรสามารถมั่นใจในความน่าเชื่อถือ ความแม่นยำ และความสามารถในการบำรุงรักษาระบบที่ขับเคลื่อนด้วย AI การทดสอบและการตรวจสอบอย่างสม่ำเสมอช่วยค้นหาข้อบกพร่อง ป้องกันการถดถอย และสร้างความมั่นใจในพฤติกรรมและผลลัพธ์ของขั้นตอนการทำงาน

ส่วนที่ 2: แพทเทิร์นต่างๆ

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

การออกแบบพรอมต์

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

การคิดแบบเป็นลำดับขั้น

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

วิธีการทำงาน

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

ตัวอย่าง

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

การสร้างเนื้อหา

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

การสร้างเอนทิตีที่มีโครงสร้าง

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

การแนะนำเอเจนต์แอลแอลเอ็ม

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

ประโยชน์และข้อควรพิจารณา

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

การสลับโหมด

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

วิธีการทำงาน

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

เมื่อไรควรใช้

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

ตัวอย่าง

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

การกำหนดบทบาท

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

วิธีการทำงาน

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

เมื่อไรที่ควรใช้

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

ตัวอย่าง

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

ออบเจกต์พรอมต์

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

วิธีการทำงาน

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

แม่แบบพรอมต์

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

วิธีการทำงาน

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

ประโยชน์และข้อควรพิจารณา

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

เมื่อใดควรใช้:

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

ตัวอย่าง

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

การรับส่งข้อมูลแบบมีโครงสร้าง

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

วิธีการทำงาน

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

การปรับขนาดการรับส่งข้อมูลแบบมีโครงสร้าง

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

ประโยชน์และข้อควรพิจารณา

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

การเชื่อมโยงคำสั่งนำ

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

วิธีการทำงาน

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

เมื่อไรควรใช้

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

ตัวอย่าง: การแนะนำผู้ใช้ใหม่ของ Olympia

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

ตัวเขียนพรอมต์ใหม่

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

วิธีการทำงาน

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

ตัวอย่าง

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

การกำหนดขอบเขตการตอบกลับ

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

วิธีการทำงาน

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

ประโยชน์และข้อควรพิจารณา

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

การจัดการข้อผิดพลาด

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

ตัววิเคราะห์คำค้นหา

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

วิธีการทำงาน

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

การนำไปใช้

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

การกำกับหน้าที่ของคำ (POS) และการรู้จำแนกชื่อเฉพาะ (NER)

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

การจำแนกเจตนา

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

การดึงคำสำคัญ

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

ประโยชน์

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

ตัวเขียนคำค้นหาใหม่

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

วิธีการทำงาน

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

ตัวอย่าง

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

ประโยชน์

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

เวนทริโลควิสต์

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

วิธีการทำงาน

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

เมื่อไรควรใช้

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

ตัวอย่าง

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

องค์ประกอบแยกส่วน

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

ตัวกำหนดเงื่อนไข

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

วิธีการทำงาน

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

เมื่อไรที่ควรใช้

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

ตัวอย่าง

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

API Facade

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

วิธีการทำงาน

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

ประโยชน์หลัก

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

เมื่อใดควรใช้

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

ตัวอย่าง

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

การพิสูจน์ตัวตนและการอนุญาตการเข้าถึง

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

การจัดการคำร้องขอ

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

การจัดรูปแบบการตอบกลับ

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

การจัดการข้อผิดพลาดและกรณีพิเศษ

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

ข้อพิจารณาด้านความสามารถในการขยายและประสิทธิภาพ

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

การเปรียบเทียบกับแพตเทิร์นการออกแบบอื่นๆ

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

ตัวแปลผลลัพธ์

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

วิธีการทำงาน

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

เมื่อไรควรใช้

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

ตัวอย่าง

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

เครื่องจักรเสมือน

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

วิธีการทำงาน

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

เมื่อไรควรใช้

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

ตัวอย่าง

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

เบื้องหลังเวทมนตร์

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

การระบุข้อกำหนดและการทดสอบ

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

การระบุพฤติกรรม

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

การเขียนกรณีทดสอบ

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

ตัวอย่าง: การทดสอบคอมโพเนนต์ Translator

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

การเล่นซ้ำการติดต่อสื่อสาร HTTP

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

มนุษย์ในระบบ (Human In The Loop: HITL)

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

รูปแบบระดับสูง

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

ปัญญาแบบผสมผสาน

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

การตอบสนองแบบปรับตัว

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

การสลับบทบาทระหว่างมนุษย์และ AI

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

การยกระดับ

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

วิธีการทำงาน

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

ประโยชน์หลัก

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

การประยุกต์ใช้ในโลกจริง: การดูแลสุขภาพ

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

วงจรการตอบกลับ

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

วิธีการทำงาน

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

การประยุกต์ใช้และตัวอย่าง

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

เทคนิคขั้นสูงในการผสานข้อมูลป้อนกลับจากมนุษย์

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

การแผ่ข้อมูลเชิงรับ

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

วิธีการทำงาน

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

การแสดงข้อมูลตามบริบท

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

การแจ้งเตือนเชิงรุก

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

ข้อมูลเชิงอธิบาย

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

การสำรวจแบบโต้ตอบ

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

ประโยชน์หลัก

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

การประยุกต์ใช้และตัวอย่าง

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

การตัดสินใจร่วมกัน (CDM)

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

วิธีการทำงาน

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

ตัวอย่าง

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

การเรียนรู้อย่างต่อเนื่อง

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

วิธีการทำงาน

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

การประยุกต์ใช้และตัวอย่าง

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

ตัวอย่าง

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

ข้อพิจารณาด้านจริยธรรม

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

บทบาทของระบบที่มีมนุษย์ร่วมในกระบวนการในการลดความเสี่ยงของ AI

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

ความก้าวหน้าทางเทคโนโลยีและมุมมองในอนาคต

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

ความท้าทายและข้อจำกัดของระบบ HITL

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

การจัดการข้อผิดพลาดอัจฉริยะ

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

วิธีการจัดการข้อผิดพลาดแบบดั้งเดิม

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

การวินิจฉัยข้อผิดพลาดตามบริบท

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

วิธีการทำงาน

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

วิศวกรรมพรอมต์สำหรับการวินิจฉัยข้อผิดพลาดตามบริบท

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

การสร้างที่เสริมด้วยการค้นคืนสำหรับการวินิจฉัยข้อผิดพลาดตามบริบท

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

การรายงานข้อผิดพลาดอัจฉริยะ

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

การป้องกันข้อผิดพลาดเชิงคาดการณ์

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

วิธีการทำงาน

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

การกู้คืนข้อผิดพลาดอัจฉริยะ

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

วิธีการทำงาน

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

การสื่อสารข้อผิดพลาดแบบเฉพาะบุคคล

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

วิธีการทำงาน

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

ขั้นตอนการจัดการข้อผิดพลาดแบบปรับตัวได้

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

วิธีการทำงาน

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

การควบคุมคุณภาพ

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

Eval

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

ปัญหา

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

วิธีแก้ปัญหา

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

วิธีการทำงาน

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

ตัวอย่าง

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

ข้อควรพิจารณา

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

ทำความเข้าใจมาตรฐานอ้างอิง

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

การทำงานของการประเมินแบบไม่อิงเกณฑ์อ้างอิง

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

กลไกป้องกัน

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

ปัญหา

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

วิธีแก้ปัญหา

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

วิธีการทำงาน

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

ตัวอย่าง

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

ข้อควรพิจารณา

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

ระบบการป้องกันและการประเมินผล: สองด้านของเหรียญเดียวกัน

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

ความสามารถในการใช้แทนกันได้ระหว่างระบบการป้องกันและการประเมินผลแบบไม่อ้างอิง

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

การใช้งานระบบการป้องกันและการประเมินผลแบบสองวัตถุประสงค์

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

อภิธานศัพท์

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

อภิธานศัพท์

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

A

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

B

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

C

เนื้อหา นี้ ไม่มี ให้ บริการ ใน หนังสือ ตัวอย่าง คุณ สามารถ ซื้อ หนังสือ ได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

D

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

E

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

F

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

G

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

H

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

I

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

J

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

K

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

L

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

M

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

N

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

O

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

P

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

Q

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

R

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

S

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

T

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

U

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

V

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

W

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

Z

เนื้อหานี้ไม่มีให้บริการในหนังสือตัวอย่าง คุณสามารถซื้อหนังสือได้ที่ Leanpub ที่ http://leanpub.com/patterns-of-application-development-using-ai-th.

Index