**Fine-tuning OpenAI API model for specific task**

นาย กิติพัฒน์ เรืองอมรวัฒน์ 63070501006

นาย สัณหณัฐ พรมจรรย์ 63070501069

Final Project – Coding in AI 2023

King Mongkut's University of Technology Thonburi

**Abstract**

OpenAI ย่อมาจาก Open Artificial Intelligence ซึ่งหมายถึง การพัฒนาและให้บริการเทคโนโลยีปัญญาประดิษฐ์และปัญญาประดิษฐ์ทางธุรกิจที่เปิดเผย (open) สำหรับนักพัฒนาและสาธารณชนทั่วไป โดยมี OpenAI API เป็นเครื่องมือที่สามารถให้บริการ Natural Language Processing (NLP) ที่มีความสามารถสูงและ OpenAI API ได้ใช้แบบจำลอง Generative Pre-trained Transformer (GPT) เป็นพื้นฐาน ซึ่งเป็นแบบจำลองที่ถูกฝึกสอนด้วยข้อมูลมหาศาล (Big Data) จากอินเทอร์เน็ต โดยไม่ต้องมีข้อมูลเฉพาะงาน (unsupervised learning) ก่อนที่จะนำไปใช้งานในงานที่ต้องการความเข้าใจและสร้างเนื้อหาทางด้านภาษา โดย GPT ใช้โครงสร้าง Transformer ที่ทำให้มีความสามารถในการจัดการกับปัญหา Natural Language Processing (NLP) อย่างมีประสิทธิภาพ จะใช้ทำนายคำหรือตัวหนังสือถัดไปโดยพิจารณาคำที่เป็นบ่งบอกทั้งหมดที่อยู่ข้างหน้า (context) ของมัน ซึ่งทำให้มีการสื่อสารที่มีความเป็นธรรมชาติมากขึ้น และเชื่อมโยงกับประสบการณ์ของผู้ใช้ [1] โดยสำหรับโมเดล GPT-3 ได้มีการใช้นิวรัลเน็ตเวิร์ก 96 เลเยอร์ พารามิเตอร์ 175 พันล้านพารามิเตอร์ และ pre-trained โดยใช้ Dataset Common Crawl ที่มีความยาว 500 พันล้านคำ และมีความสามารถในการทำงานกับตัวอย่างงานที่หลากหลาย เช่น การสร้างข้อความ, การตอบคำถาม, และแม้กระทั่งการทำนายโค้ด

1. **Introduction**

[2] เดิมที โมเดล GPT จะ pre-trained เกี่ยวกับข้อความที่หลากหลาย แนวทางทั่วไปสำหรับประสิทธิผลเฉพาะงานคือการใช้ few-shot learning โดยให้คำแนะนำและตัวอย่างผ่าน prompt ซึ่ง fine-tuning จะช่วยปรับปรุงสิ่งนี้โดยใช้ประโยชน์จาก dataset ที่กว้างขวางมากขึ้น ซึ่งนำไปสู่การปรับปรุงประสิทธิภาพในงานต่าง ๆ มากมาย สิ่งสำคัญคือต้องเข้าใจความแตกต่างของ fine-tuning แม้ว่า fine-tuning จะช่วยปรับแต่งรูปแบบการสื่อสารของโมเดลได้อย่างเชี่ยวชาญ แต่ก็อาจไม่เปลี่ยนแปลงเนื้อหาหลักหรือสาระสำคัญอย่างมีนัยสำคัญ เนื่องจากพื้นที่เก็บข้อมูลที่กว้างขวางของข้อมูลภาษาทั่วไปที่มีอยู่ก่อนภายในโมเดล ดังนั้น fine-tuning จะเป็นประโยชน์อย่างยิ่งเมื่อความรู้พื้นฐานของโมเดลสอดคล้องกับงานที่ทำอยู่ แต่โมเดลนั้นจำเป็นต้องได้รับการ fine-tune เพื่อให้ได้ระดับความแม่นยำหรือความจำเพาะที่สูงขึ้น

การพัฒนาโมเดลภาษาขนาดใหญ่ เช่น GPT-3 ของ OpenAI ได้ปฏิวัติแอปพลิเคชัน Natural Language Processing (NLP) โมเดลเหล่านี้แสดงให้เห็นถึงประสิทธิภาพที่โดดเด่นในงานที่เกี่ยวข้องกับภาษาต่าง ๆ เนื่องมาจากการ pre-train ใน dataset ที่กว้างขวาง อย่างไรก็ตาม การบรรลุประสิทธิภาพสูงสุดสำหรับงานเฉพาะมักต้องมีการปรับแต่งโมเดลเหล่านี้กับข้อมูลเฉพาะงานอย่างละเอียด

โดยโปรเจกต์นี้มีจุดมุ่งหมายเพื่อสำรวจความแตกต่างของโมเดล OpenAI API ที่ทำการ fine-tuning ของโมเดลต่าง ๆ และขยายขอบเขตของ fine-tuning โดยเฉพาะโมเดล GPT-3.5 ด้วยใช้ dataset ที่มีอยู่ ซึ่งจะมีความแตกต่างจากฟังก์ชันการทำงานทั่วไปของ ChatGPT-3.5 ตรงที่โมเดลที่การ fine-tuning สามารถสร้างขึ้นเพื่อให้ได้รับการตอบสนองที่แม่นยำยิ่งขึ้นและเกี่ยวข้องกับโดเมนหรือ dataset ที่เฉพาะเจาะจง ซึ่งช่วยให้สามารถควบคุมศักยภาพสูงสุดของโมเดล ซึ่งในโปรเจกต์นี้จะทำเกี่ยวกับการ Fine-tuning OpenAI API model สำหรับงานเฉพาะเจาะจง โดยจะมีโดเมนเฉพาะเจาะจง คือการให้ข้อมูลเกี่ยวกับสถานที่ท่องเที่ยวในจังหวัดเชียงใหม่ จากนั้นจะนำ dataset ของสถานที่ท่องเที่ยวในจังหวัดเชียงใหม่มาทำการ fine-tuning ให้กับแต่ละโมเดลของ OpenAI

1. **Theory and Techniques**
   1. **Unsupervised Fine-tuning**

Unsupervised Fine-tuning เป็นส่วนสำคัญในการปรับโมเดล OpenAI API ให้เข้ากับงานเฉพาะ โมเดลที่ pre-trained เบื้องต้นจะช่วยให้เข้าใจภาษาได้กว้าง แต่อาจไม่ได้รับการปรับให้เหมาะกับการใช้งานเฉพาะเจาะจงโดยไม่มีการระบุข้อมูลที่ชัดเจน ซึ่ง Unsupervised Fine-tuning จะช่วยให้โมเดลสามารถปรับให้เข้ากับโดเมนหรือข้อกำหนดเฉพาะโดยไม่มี label เฉพาะงานที่ชัดเจน ซึ่งในระหว่างการ Unsupervised Fine-tuning โมเดลจะแสดงกับข้อมูลเฉพาะงาน หากไม่มี keyword ที่ชัดเจนเกี่ยวกับเอาต์พุตที่ต้องการ โมเดลจะปรับพารามิเตอร์ตามรูปแบบและคุณลักษณะที่มีอยู่ในข้อมูลอินพุต โดยปรับข้อมูลให้เข้ากับความแตกต่างของงานที่กำหนดได้อย่างมีประสิทธิภาพ

* 1. **Pre-training Model**

Pre-training เป็นรากฐานของความสามารถของโมเดล OpenAI API ช่วยให้เข้าใจภาษาและบริบทในวงกว้างจาก dataset ที่หลากหลาย สำหรับประสิทธิผลของการ fine-tune โดยโมเดลจะเปิดเผยให้รู้จักกับรูปแบบและโครงสร้างภาษาที่หลากหลาย โมเดล OpenAI API ผ่านการ pre-training เกี่ยวกับ dataset ขนาดใหญ่ ซึ่งโดยทั่วไปจะมาจากอินเทอร์เน็ต ขั้นตอนนี้จะทำให้โมเดลมีความเข้าใจภาษาโดยทั่วไป ทำให้สามารถปรับให้เข้ากับ downstream tasks ได้ ซึ่งในส่วนนี้จะเจาะลึก Pre-training โมเดลจำนวนสองโมเดลที่ใช้ใน OpenAI API:   
GPT-3.5-Turbo-1106 และ Babbage-002

* + 1. **GPT-3.5-Turbo-1106**

GPT-3.5 Turbo เป็นโมเดลที่ถูกเผยแพร่โดย Anthropic ในเดือนพฤศจิกายน 2022 เป็นโมเดลที่ใหญ่และมีประสิทธิภาพมากที่สุดในปัจจุบัน ด้วยพารามิเตอร์มากกว่า 175 พันล้านพารามิเตอร์ ต้องใช้ทรัพยากรการประมวลผลในระดับมหาศาลเพื่อการ train และทํานาย มีขนาดใหญ่มากจนองค์กรส่วนมากไม่สามารถ train โมเดลนี้ได้ พารามิเตอร์จำนวนมหาศาลทำให้รองรับข้อมูลในการ train ได้ในระดับหลายล้านล้านคํา แม้ว่าจะมีความสามารถในการสนทนาเสมือนมนุษย์มากที่สุดเท่าที่เคยพบ แต่การนําไปใช้งานจริงก็ต้องการทรัพยากรและงบประมาณจำนวนมหาศาลเช่นกัน สำหรับองค์กรส่วนใหญ่แล้ว GPT-3.5 Turbo ถือเป็น overkill สำหรับการใช้งานทั่วไป ในส่วนโมเดลขนาดเล็กกว่าอย่าง Babbage และ Davinci จึงคุ้มค่ามากกว่า

* + 1. **Babbage-002**

Babbage-002 เป็นโมเดลปัญญาประดิษฐ์สำหรับการสนทนาที่พัฒนาโดย Anthropic เพื่อให้มีประโยชน์ และมีความปลอดภัย มีพารามิเตอร์จำนวน 1.3 พันล้านตัว ทำให้เป็นโมเดลขนาดกลางที่ถูกออกแบบมาเพื่อสมดุลระหว่างประสิทธิภาพและความคุ้มค่า Babbage ถูกออกแบบเฉพาะสำหรับการใช้งานจริง เช่น chatbot และ assistant ใช้ข้อมูลและการประมวลผลน้อยกว่าโมเดลอื่นหลายเท่าในการฝึก ทำให้ Babbage มีค่าใช้จ่ายตํ่าในการนำมาใช้งาน สามารถฝึกให้สนทนาอย่างปลอดภัยได้หลากหลายบริบทโดยไม่ต้องใช้ข้อมูลจำนวนมาก Babbage ให้คุณภาพการสนทนาที่ดีและความสามารถในการให้เหตุผลโดยไม่ต้องเสียค่าใช้จ่ายสูงหรือมีความเสี่ยงมาก ขนาดที่เหมาะสมช่วยให้หลีกเลี่ยงปัญหาต่าง ๆ เช่น การสร้างข้อมูลเท็จหรือข้อความ toxic ที่เกิดกับโมเดลขนาดใหญ่ Anthropic ออกแบบ Babbage ให้สามารถใช้งานได้จริงและเข้าถึงได้สำหรับองค์กรส่วนใหญ่แทนที่จะเน้นประสิทธิภาพสูงสุด Babbage ตอบโจทย์การประยุกต์ใช้ AI สนทนาได้หลากหลายมากขึ้นแทนที่จะเน้นเพียงประสิทธิภาพเสมือนมนุษย์

* 1. **Data Collection**

เทคนิค Web-Scraping เป็นวิธีการที่ใช้ในการดึงข้อมูลจากเว็บไซต์เพื่อรวบรวมข้อมูลเฉพาะงานเพื่อปรับแต่งโมเดล OpenAI API อย่างละเอียด ทำให้สามารถรับ dataset ที่หลากหลายและเฉพาะโดเมนได้ ในส่วนนี้จะสรุปองค์ประกอบสำคัญของการทำ Web-Scraping โดยเน้นไปที่กระบวนการ Web-Scraping

* + 1. **Data Sources**

การเลือกแหล่งข้อมูลที่เหมาะสมถือเป็นส่วนสำคัญของ Web-Scraping โดย target แหล่งข้อมูลต้องได้รับการคัดเลือกอย่างรอบคอบเพื่อให้สอดคล้องกับงานเฉพาะที่มีการปรับแต่งโมเดล OpenAI API อย่างละเอียด ข้อพิจารณารวมถึงความเกี่ยวข้องของเนื้อหา ความหลากหลายของข้อมูล และแง่มุมทางจริยธรรมของการรวบรวมข้อมูล โดยข้อมูลที่เราเลือกมาใช้จะมาจากเว็บไซต์ท่องเที่ยวของการท่องเที่ยวแห่งประเทศไทย [4]   
จากรูปที่ 1 Tourism Authority of Thailand (TAT) ซึ่งเป็นหน่วยงานที่รับผิดชอบในการสนับสนุนและส่งเสริมกิจกรรมท่องเที่ยวในประเทศไทย ในเว็บไซต์นี้สามารถหาข้อมูลเกี่ยวกับสถานที่ท่องเที่ยวที่น่าสนใจ, กิจกรรมท่องเที่ยว, แนะนำที่พัก, สิ่งที่น่าทดลอง, และข้อมูลท่องเที่ยวอื่น ๆ ที่เกี่ยวข้องกับการท่องเที่ยวในประเทศไทย

รูปภาพประกอบด้วย ข้อความ, ภาพหน้าจอ, หน้าเว็บ, เว็บไซต์

คำอธิบายที่สร้างโดยอัตโนมัติ

รูปที่ 1 แสดงหน้าเว็บไซต์ที่ใช้เป็นแหล่งข้อมูลซึ่งข้อมูลจะเกี่ยวกับสถานที่ท่องเที่ยวในจังหวัดเชียงใหม่

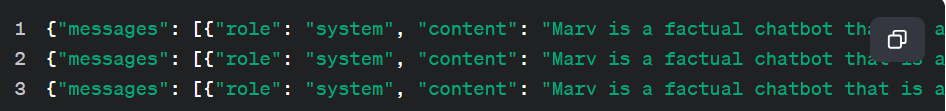
* + 1. **Selenium Library**

เนื่องจากแหล่งข้อมูลที่ใช้งานเป็นเว็บไซต์แบบไดนามิกเว็บ กระบวนการนี้เราจึงเลือกใช้ไลบรารี Selenium ซึ่งเป็นเครื่องมืออันทรงพลังสำหรับการโต้ตอบกับเนื้อหาเว็บแบบไดนามิก โดยให้รายละเอียดเกี่ยวกับการกำหนดค่า การนำทาง การระบุองค์ประกอบ และกระบวนการแยกข้อมูล จัดการกับความท้าทายและการพิจารณาด้านจริยธรรมที่เกี่ยวข้องกับ Web-Scraping โดยไลบรารี Selenium ได้รับการกำหนดค่าให้จำลองเว็บเบราว์เซอร์ ทำให้สามารถโต้ตอบอัตโนมัติกับเนื้อหาไดนามิกของเว็บไซต์เป้าหมายได้ ในส่วนนี้จะให้รายละเอียดเกี่ยวกับกระบวนการตั้งค่า รวมถึงการติดตั้งไลบรารี Selenium และการกำหนดค่าไดรเวอร์เว็บเพื่อให้เข้ากันได้กับเว็บเบราว์เซอร์ที่เลือก จากนั้นเราจะไปสร้าง dataset จากข้อมูลที่ดึงมาจากเว็บเพื่อการปรับแต่งโมเดล OpenAI API และการวิเคราะห์ผลลัพธ์ในภายหลัง

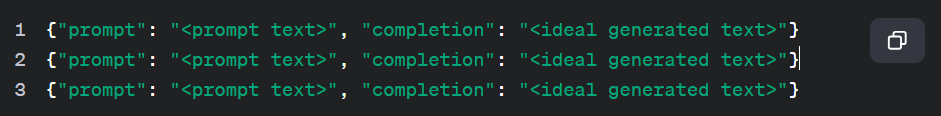
* + 1. **Dataset Format**

Dataset สำคัญในการกำหนดประสิทธิภาพของโมเดล และการจัดเตรียมประกอบด้วยกระบวนการสำคัญหลายประการเพื่อให้แน่ใจว่ามีความเกี่ยวข้อง ความหลากหลาย และประสิทธิผล ส่วนนี้จะแสดงภาพรวมของ dataset รวมถึงรูปแบบและกระบวนการต่าง ๆ ที่ใช้ในการดูแลจัดการเพื่อ fine-tuning

Dataset Format ส่งผลโดยตรงต่อวิธีการประมวลผลของโมเดลและการเรียนรู้จากข้อมูล ในบริบทของ Fine-tuning OpenAI API รูปแบบ dataset ควรสอดคล้องกับข้อกำหนดอินพุตของโมเดล โดย dataset format รวมถึงโครงสร้างจะอยู่ในรูปแบบ JSON Lines ดังรูปที่ 2 และ 3



รูปที่ 2 แสดงสามตัวอย่างรูปแบบ dataset ที่ใช้สำหรับโมเดลเวอร์ชั่นใหม่



รูปที่ 3 แสดงสามตัวอย่างรูปแบบ dataset ที่ใช้สำหรับโมเดลเวอร์ชั่นเก่า

จากรูปที่ 2 รูปแบบนี้ช่วยให้สามารถนำเสนอการแลกเปลี่ยนการสนทนาที่มี role ที่แตกต่างกัน (system, user, assistant) และ content ที่เกี่ยวข้อง โดยทั่วไป role "system" จะกำหนดบริบทหรือคุณลักษณะของ assistant ส่วน role "user" ก่อให้เกิดคำถามหรือ prompts และ role "assistant" จะตอบสนอง โดยผสมผสานข้อมูลที่เป็นข้อเท็จจริงเข้ากับข้อมูลที่แตกต่างกันเล็กน้อย ตามคำอธิบายของข้อความระบบ

* "messages": Key นี้จะแสดง list message ที่มีการแลกเปลี่ยนกันในการสนทนา แต่ละ message จะมี "role" และ "content"
* "role": Key นี้จะระบุ role ของผู้ส่งข้อความ ไม่ว่าจะเป็น "system", "user" หรือ "assistant"
* "content": Key นี้ประกอบด้วย content หรือ text ที่เป็นข้อเท็จจริงของ message

จากรูปที่ 3 "prompt": Key นี้แสดงถึงอินพุตหรือสิ่งกระตุ้นที่มอบให้กับโมเดลภาษา เป็นข้อความหรือบริบทที่ต้องการให้โมเดลสร้างการตอบกลับ โดย prompt เป็นจุดเริ่มต้นสำหรับโมเดลในการสร้างหรือเติมข้อความ และ "completion": Key นี้แสดงถึงเอาต์พุตที่คาดหวังหรืออุดมคติที่ต้องการให้โมเดลสร้างตาม prompt ที่กำหนด โดยทำหน้าที่เป็นข้อมูลอ้างอิงหรือเป้าหมายสำหรับกระบวนการ fine-tuning เป้าหมายคือการฝึกโมเดลในลักษณะที่เมื่อได้รับ prompt ที่คล้ายกันในอนาคต โมเดลจะสร้างการตอบสนองที่ใกล้เคียงกับความสำเร็จที่ให้ไว้มากขึ้น

* 1. **Fine-tuning pre-trained model**

ในการ Fine-tuning เรามีจุดมุงเน้นเพื่อจะให้โมเดลนั้นสามารถทำงานตามที่เราต้องการได้ โดยงานนั้น คือ การให้ข้อมูลสถานที่ท่องเที่ยวในเชียงใหม่ โดยข้อมูลนั้นจะต้องมากจาก dataset ของเราโดยโมเดลที่เรานำมา fine-tune นั้นมี 2 โมเดลนั่น คือ gpt-3.5-turbo และ babbage-002 ความแตกต่างของ 2 โมเดลนี้ คือ จำนวนของข้อมูลที่ได้ใช้ train 2 pre-trained โมเดลนี้ โดย babbage-002 จะมีจำนวนข้อมูลที่น้อยกว่าแต่มีความเร็วที่มากกว่า ในขณะที่ gpt-3.5-turbo มีจำนวนข้อมูลที่เยอะกว่าประสิทธิภาพดีกว่าแต่ทำงานได้ช้ากว่า โดยจุดมุ่งหมายที่เลือกทั้ง 2 โมเดลนี้เพราะว่าเราต้องการอยากทราบว่าข้อมูลของเรานั้นมีความเหมาะสมกับโมเดลแบบไหนมากที่สุด โดยขั้นตอนในการ fine-tuning มีดังนี้

* + 1. **Prepare data**

ในขั้นตอนนี้ คือการเตรียมข้อมูลโดยเหตุผลที่เราต้องเตรียมข้อมูลนั้นเพราะว่าในการ fine-tuning OpenAI model นั้นต้องใช้ข้อมูลที่มี format ที่เป็นแบบดังรูปที่ 2 และ 3

รูปภาพประกอบด้วย ข้อความ, ภาพหน้าจอ, ตัวอักษร

คำอธิบายที่สร้างโดยอัตโนมัติ

รูปที่ 4 แสดง Code เกี่ยวกับการจัดเก็บ dataset ในขั้นตอน Web-Scraping

รูปภาพประกอบด้วย ข้อความ, ภาพหน้าจอ, ซอฟต์แวร์, ซอฟต์แวร์มัลติมีเดีย

คำอธิบายที่สร้างโดยอัตโนมัติ

รูปที่ 5 แสดง Code เกี่ยวกับการสร้างตัวแปรเพื่อนำไปแปลง Data format

รูปภาพประกอบด้วย ข้อความ, ภาพหน้าจอ, ซอฟต์แวร์

คำอธิบายที่สร้างโดยอัตโนมัติ

รูปที่ 6 แสดง Code เกี่ยวกับการแปลง Data format

ในส่วนของก่อนการทำ Data format เราจะนำข้อมูลที่ได้จากการ Scraping มาแปลง format ให้ตรงกับที่ API ต้องการ จากรูปที่ 4 ที่เราเก็บข้อมูลมาจากการทำ Web-Scraping แล้วนำข้อมูลที่ได้ไปแปลง Data format ดังรูปที่ 5 และ 6

* + 1. **Upload training and validation data**

รูปภาพประกอบด้วย ข้อความ, อิเล็กทรอนิกส์, ภาพหน้าจอ, แสดง

คำอธิบายที่สร้างโดยอัตโนมัติ

รูปที่ 7 แสดง Code เกี่ยวกับการแบ่ง training data กับ validation data

จากรูปที่ 7 เราจะทำการอัปโหลดไฟล์ที่เป็น format ที่ตรงกับ API เข้าไปเก็บเพื่อใช้ในการนำไป   
fine-tuning ต่อโดยเราจะทำการแบ่ง training data กับ validation data ก่อนในกรณีที่ข้อมูลยังไม่ถูกแบ่ง

รูปภาพประกอบด้วย ข้อความ, ภาพหน้าจอ, ตัวอักษร, ซอฟต์แวร์

คำอธิบายที่สร้างโดยอัตโนมัติ

รูปที่ 8 แสดง Code เกี่ยวกับ file id ที่ใช้ในการ fine-tuning

จากรูปที่ 8 เมื่อแบ่ง data เสร็จแล้วหลังจากนั้นจะอัปโหลดเข้า API เพื่อ return ดู file id ที่จำเป็นจะต้องนำไปใช้ในการ fine-tuning

รูปภาพประกอบด้วย ข้อความ, ภาพหน้าจอ, ซอฟต์แวร์, ไอคอนคอมพิวเตอร์

คำอธิบายที่สร้างโดยอัตโนมัติ

รูปที่ 9 แสดงไฟล์ที่อัปโหลดมาแล้วถูกนำไปเก็บไว้ที่ Base URL ของ API

* + 1. **Fine-tuning the model**

เมื่อ data พร้อมแล้วในขั้นตอนต่อมาก็จะเริ่มการ fine-tuning

รูปภาพประกอบด้วย ข้อความ, ภาพหน้าจอ, แสดง, ซอฟต์แวร์

คำอธิบายที่สร้างโดยอัตโนมัติ

รูปที่ 10 แสดง Code เกี่ยวกับการใช้ API เพื่อทำการ fine-tuning

จากรูปที่ 10 ในการ fine-tuning ด้วยการใช้ API นั้น เราก็ต้องทำการนำไฟล์ไอดีมาใส่ในพารามิเตอร์ รวมไปถึงการเลือกโมเดลการกำหนดชื่อโมเดลและการเลือก hyperparameters โดยการ fine-tuning OpenAI API model นั้น hyperparameters ที่ทำการปรับได้มีเพียง epoch, batch size และ learning rate เท่านั้น

รูปภาพประกอบด้วย ข้อความ, ภาพหน้าจอ, ซอฟต์แวร์, จำนวน

คำอธิบายที่สร้างโดยอัตโนมัติ

รูปที่ 11 แสดงการส่ง request กลับไปยัง Base URL พร้อมมีค่า monitor

จากรูปที่ 11 หลังจากรันโปรแกรมนี้โมเดลก็จะเริ่มทำการ fine-tuning โดยจะส่ง request กลับไปยัง Base URL และ monitor ค่าต่าง ๆ ให้ดูในระหว่างการ fine-tune

รูปภาพประกอบด้วย ข้อความ, ภาพหน้าจอ, ซอฟต์แวร์, ตัวอักษร

คำอธิบายที่สร้างโดยอัตโนมัติ

รูปที่ 12 แสดง Code เกี่ยวกับการให้ monitor แสดงค่า loss ในแต่ละ step

รูปภาพประกอบด้วย ข้อความ, ภาพหน้าจอ, แบบแผน, สีดำ

คำอธิบายที่สร้างโดยอัตโนมัติ

รูปที่ 13 แสดงค่า loss ในแต่ละ step

จากรูปที่ 12 และ 13 เมื่อ fine-tuning เสร็จแล้วเราสามารถ monitor ค่า loss ในแต่ละ step ออกมาดูด้วยตัวเองได้ด้วยการใส่ fine-tuning job id ของโมเดลนั้น ๆ ซึ่งค่าที่ถูก response ออกมาจะอยู่ในพารามิเตอร์ data ซึ่งบอกค่า loss และค่า accuracy ของทั้ง training และ validation โดยในการ fine-tuning นั้นจะไม่ได้ monitor ค่าในการ train ออกมาเป็น epoch แต่จะแสดงออกมาในรูปแบบของ step โดย step นั้นจะคิดคำนวนจาก epoch, batch size และขนาด dataset ที่ใช้โดยอัตโนมัติ

* + 1. **Use fine-tuned model**

วิธีการนำ fine-tuning model ไปใช้

รูปภาพประกอบด้วย ข้อความ, ภาพหน้าจอ, แสดง, ตัวอักษร

คำอธิบายที่สร้างโดยอัตโนมัติ

รูปที่ 14 แสดง Code เกี่ยวกับการเชื่อม OpenAI API ด้วย Key

จากรูปที่ 14 เราจำเป็นจะต้องนำ API key ของเรามาใส่เพื่อให้ทาง Base URL สามารถรับ request และส่ง response อย่างถูกต้อง

รูปภาพประกอบด้วย ข้อความ, ตัวอักษร, ซอฟต์แวร์, ซอฟต์แวร์มัลติมีเดีย

คำอธิบายที่สร้างโดยอัตโนมัติ

รูปที่ 15 แสดง Code เกี่ยวกับการใช้ generate text เพื่อดู output

จากรูปที่ 15 ในการนำมาใช้ generate text เพื่อดู output เราจะใช้คำสั่ง openai.chat.completions.create() ในการ request โดยจะใช้ชื่อของโมเดลจาก Base URL มาใส่ในพารามิเตอร์ model เพื่อเลือกว่าจะใช้โมเดลไหนในการ generate text โดยพารามิเตอร์อื่นๆที่ได้ใช้ก็มี

* temperature ที่ใช้กำหนดการ random ของคำตอบว่าอยากให้คำตอบมีค่าความ random มากแค่ไหนในการ response แต่ละครั้ง
* max\_tokens ใช้ในการกำหนดความยาวของคำตอบว่าอยากให้จบในกี่ token
* messages ใช้ในการถามตอบกับโมเดลโดยจะเป็นในรูปแบบ JSON เดียวกันกับ data format ที่ใช้ในการ train โดยจะมี system content ที่สามารถกำหนด tone และ style การตอบของโมเดลและมี user content ที่เป็นคำถามใช้ถามโมเดล

ในการดูคำตอบเราจะใช้ reponse.choices[0].message เพื่อดูคำตอบแรกของโมเดล

1. **Experimental Design and Results**

ในส่วนนี้เราจะทำการทดสอบกับโมเดลที่เราได้ Fine-tune ออกมาว่าสามารถ response output ที่เราต้องการได้หรือไม่โดย output ที่เราต้องการ คือ การนำข้อความในส่วนของ dataset มา response ให้เราเมื่อเรา input prompt ในส่วนของสถานที่นั้น ๆ เช่น input : ขอข้อมูลรายละเอียดเกี่ยวกับม่อนแจ่ม  
output : <รายละเอียดข้อมูลที่ใส่ไปใน dataset> โดยเราได้ทำการทดสอบ ดังนี้

* 1. **ทดสอบ fine-tuning dataset ที่แตกต่างกัน**

ในส่วนของการทดสอบนี้นั้นเราจะทำการ fine-tune ด้วย data ที่แตกต่างกันโดยมีขนาดของข้อมูล, รายละเอียดของข้อมูลข้างในที่แตกต่างกันเพื่อหา dataset ที่เหมาะสมที่สุดในการนำมา fine-tuning โดยเราจะนำไปใช้กับ pre-trained โมเดลนี้

**โมเดลที่ 1 (Pre-trained model: gpt-3.5-turbo-1106)**

รูปภาพประกอบด้วย ข้อความ, ภาพหน้าจอ, แสดง, ซอฟต์แวร์

คำอธิบายที่สร้างโดยอัตโนมัติ

รูปที่ 16 แสดง Pre-trained model: gpt-3.5-turbo-1106

epoch: 3

batch size: 8

learning rate: 0.01

**Dataset แบบที่ 1**

รูปภาพประกอบด้วย ภาพหน้าจอ, ข้อความ, ตัวอักษร, ไลน์

คำอธิบายที่สร้างโดยอัตโนมัติ

รูปที่ 17 แสดงเกี่ยวกับ format Dataset แบบที่ 1

จากรูปที่ 16 ใน dataset แบบแรกที่เรานำมา fine-tuning จะเป็นดังนี้ซึ่งมีจำนวน data อยู่ที่ 60 sample ในส่วนของ system content นั้นจะกำหนด style การตอบด้วยคำว่า ”คุณเป็นไกด์นำทางที่เป็นมิตรและสุภาพ” ในทุกๆ messages เพื่อกำหนด tone และ style แบบเดียวกัน ต่อมาในส่วนของ user content นั้นจะมีการกำหนดรูปแบบคำถามที่เหมือนกันโดยจะเป็นในรูปแบบ “ฉันต้องการข้อมูลรายละเอียดเกี่ยวกับ<ชื่อสถานที่>” และ assistant content หรือคำตอบที่ต้องการให้โมเดลตอบจะใส่รายละเอียดข้อมูลของสถานที่นั้นไปเป็น long text ยาว ๆ และนำ dataset ดังกล่าวไป train กับ gpt-3.5-turbo-1106

**ผลลัพธ์**

รูปภาพประกอบด้วย ไลน์, ข้อความ, พล็อต, ภาพหน้าจอ

คำอธิบายที่สร้างโดยอัตโนมัติ

รูปที่ 18 แสดงกราฟ Training loss กับ Validation loss ของ dataset แบบที่ 1

จากรูปที่ 18 หลังจาก fine-tune ด้วย dataset แบบที่ 1 ได้ผลลัพธ์ ซึ่งมีค่า loss ที่ค่อนข้างสูงหลังจากนั้นเราได้นำไปลองใช้ในการถามข้อมูลสถานที่ท่องเที่ยวอย่างม่อนแจ่มได้ผลลัพธ์ดังนี้

**รูปภาพประกอบด้วย ข้อความ, ภาพหน้าจอ, ซอฟต์แวร์, ซอฟต์แวร์มัลติมีเดีย

คำอธิบายที่สร้างโดยอัตโนมัติ**

รูปที่ 19 แสดง Code เกี่ยวกับการใช้ generate text เพื่อดู output

จากรูปที่ 19 คำตอบที่ได้: “ม่อนแจ่ม (Mon Cham) เป็นสถานที่ท่องเที่ยวที่ตั้งอยู่ในอำเภอแม่ริม จังหวัดเชียงใหม่ ประเทศไทย ม่อนแจ่มเป็นที่รู้จักด้วยทิวทัศน์ที่สวยงามของภูเขาและทุ่งนาที่ล้อมรอบ นอกจากนี้ยังมีสวนผลไม้และสวนดอกไม้ที่สวยงามอีกด้วย\n\nที่ม่อนแจ่มนั้นนักท่องเที่ยวสามารถเดินเล่นรอบเขาหรือทำกิจกรรมทางการเกษตรเช่นการเก็บผลไม้หรือดูดอกไม้ได้ นอกจากนี้ยังมีร้านอาหารและรีสอร์ทที่สามารถพักผ่อนและพักค้างคืนได้\n\nการเดินทางไปยังม่อนแจ่มสามารถใช้รถยนต์ส่วนตัวหรือจักรยานจากเชียงใหม่ โดยมีระยะทางประมาณ 40 กิโลเมตร และเส้นทางที่สวยงามและสะดวกสบาย”

**Dataset แบบที่ 2**

**รูปภาพประกอบด้วย ภาพหน้าจอ, ข้อความ, ตัวอักษร

คำอธิบายที่สร้างโดยอัตโนมัติ**

รูปที่ 20 แสดงเกี่ยวกับ format Dataset แบบที่ 2

จากรูปที่ 20 หลังจากที่ลองกับ dataset ในแบบแรกแล้วพบว่าค่า loss ค่อนข้างสูงเราก็ได้ไปค้นหาข้อมูลเพิ่มเติมว่าเป็นเพราะอะไรและก็ได้ไปพบกับเว็บบล็อกที่แนะนำการ fine-tuning พอนำ dataset ของเราไปเปรียบเทียบกับเขาพบว่า dataset ของเรานั้นมีรูปแบบที่ไม่เหมือนกัน dataset ของเขานั้นมีรูปแบบที่ชัดเจนกว่าและมีขนาดจำนวน sample ที่มากกว่าและข้อความที่สั้นกว่าเราจึงได้นำมาปรับใช้กับ dataset เก่าของเราโดยปรับเปลี่ยนรูปแบบคำถามและ style ของโมเดลโดยเราได้เพิ่มความละเอียดใน style การตอบของโมเดลลงไปในส่วนของ system content และปรับเปลี่ยนรูปแบบของ user content ตามตัวอย่างที่เราได้ไปพบมาจากนั้นก็ลดจำนวน sample เพราะต้องการลด token ที่มีมากเนื่องจากการ fine-tune แต่ละครั้งมีค่าใช้จ่ายเลยลองลบสถานที่ที่ไม่ต้องการออกให้เหลือ 40 sample และลดความยาวของข้อมูลในบาง sample ลงจากนั้นนำไป fine-tune กับ gpt-3.5-turbo ใหม่อีกครั้ง

**ผลลัพธ์**

**รูปภาพประกอบด้วย ไลน์, ข้อความ, พล็อต, ภาพหน้าจอ

คำอธิบายที่สร้างโดยอัตโนมัติ**

รูปที่ 21 แสดงกราฟ Training loss กับ Validation loss ของ dataset แบบที่ 2

จากรูปที่ 21 หลังจาก fine-tune ด้วย dataset แบบที่ 2 พบว่า training loss มีค่าที่ต่ำลงไม่มากในขณะที่ validation loss มีค่าสูงขึ้นมากๆและได้ลองไปใช้ถามตอบแบบเดิม

รูปภาพประกอบด้วย ข้อความ, ตัวอักษร, ภาพหน้าจอ

คำอธิบายที่สร้างโดยอัตโนมัติ

รูปที่ 22 แสดง Code เกี่ยวกับการใช้ generate text เพื่อดู output

จากรูปที่ 22 คำตอบที่ได้: “สวัสดีครับ/ค่ะ! สถานที่ท่องเที่ยว "ม่อนแจ่ม" เป็นสถานที่ท่องเที่ยวที่ตั้งอยู่บนเทือกเขาดอยอ่างขางในจังหวัดเชียงใหม่ ม่อนแจ่มเป็นจุดชมวิวที่สวยงามที่สุดของดอยอ่างขาง ที่นี่มีทิวทัศน์ที่สวยงามของทะเลหมอกและภูเขาที่สลัดใส นอกจากนี้ยังมีสวนดอกไม้ที่สวยงามและเป็นที่รู้จักอีกด้วย นอกจากนี้ยังมีกิจกรรมต่างๆ เช่น การปีนเขา การเดินทางด้วยจักรยาน และการเดินเล่นในสวนดอกไม้ ที่นี่ยังมีร้านอาหารและร้านขายของที่ทำจากผลผลิตภูเขาให้บริการด้วยครับ/ค่ะ ถ้าคุณชอบการผจญภัยและการสัมผัสกับธรรมชาติ ม่อนแจ่มเป็นสถานที่ท่องเที่ยวที่น่าสนใจมากครับ/ค่ะ หวังว่าข้อมูลนี้จะมีประโยชน์สำหรับคุณครับ/ค่ะ ถ้ามีข้อสงสัยเพิ่มเติม กรุณาสอบถามเพิ่มเติมครับ/ค่ะ”

จากทั้ง 2 dataset พบว่าไม่ว่าจะเป็น dataset ไหนโมเดลก็ไม่ได้ตอบตรงกับข้อมูลที่อยู่ใน dataset เลยมีเพียงรูปแบบการตอบที่เปลี่ยนไปและการจัดเรียงรายละเอียดที่แตกต่างกันจะเห็นได้ว่าโมเดลจากการ fine-tune ด้วย dataset ทั้ง 2 แบบนั้นล้วนมีค่า loss ที่สูงทั้ง training และ validation ซึ่งแสดงให้เห็นว่าการเรียนรู้ของ data ของโมเดลนั้นยังไม่ดีมากนักแต่ใน dataset ที่ 2 มีการจัดเรียงคำ ประโยคและการให้ข้อมูลที่ดีกว่าซึ่งจากผลลัพธ์คาดว่าเป็นเพราะ system content ที่เปลี่ยนเป็นภาษาอังกฤษและกำหนดได้ละเอียดกว่า dataset แบบแรกเราจึงตัดสินใจที่จะใช้ dataset แบบที่ 2 ในการทดลองถัดๆไป

* 1. **ทดสอบ fine-tuning โมเดลที่ Hyperparameters แตกต่างกัน**

จากการทดสอบแบบที่ 1 เราได้ตั้งสมมุติฐานว่า เป็นเพราะเราปรับ hyperparameters ไม่ดีหรือเปล่าจึงทำให้โมเดลเรียนรู้ได้ไม่ดี ในส่วนของการทดสอบนี้เราได้ทำการเปลี่ยน hyperparameters ให้สอดคล้องกับจำนวน sample และขนาดของ dataset ของเราที่มีขนาดเล็ก โดยใช้ gpt-3.5-turbo เช่นเดิมเราได้ลองปรับทั้งหมด 2 แบบดังนี้

**Hyperparameters แบบที่ 1**

รูปภาพประกอบด้วย ข้อความ, ภาพหน้าจอ, แสดง, ซอฟต์แวร์

คำอธิบายที่สร้างโดยอัตโนมัติ

รูปที่ 23 แสดง Code เกี่ยวกับการเปลี่ยน Hyperparameters แบบที่ 1

จากรูปที่ 23 ใช้ค่า batch size : 8 และ learning rate : 0.01 เหมือนเดิมแต่เพิ่ม epoch ไปเป็น 5

**ผลลัพธ์**

รูปภาพประกอบด้วย ไลน์, พล็อต, ข้อความ, ภาพหน้าจอ

คำอธิบายที่สร้างโดยอัตโนมัติ

รูปที่ 24 แสดงกราฟ Training loss กับ Validation loss ของ Hyperparameters แบบที่ 1

จากรูปที่ 24 ผลลัพธ์ที่ได้หลังจากเพิ่ม epoch ไปก็ยังเป็นเหมือนเดิมโมเดลยังไม่สามารถเรียนรู้ dataset ได้จึงได้ลองปรับ hyperparameters เพิ่มเติม

**Hyperparameter แบบที่ 2**

รูปภาพประกอบด้วย ข้อความ, ภาพหน้าจอ, แสดง, ซอฟต์แวร์

คำอธิบายที่สร้างโดยอัตโนมัติ

รูปที่ 25 แสดง Code เกี่ยวกับการเปลี่ยน Hyperparameters แบบที่ 2

จากรูปที่ 25 โดยในครั้งนี้เราได้ปรับ batch size ขึ้นไปที่ 16 และลด learning rate 0.001 เพื่อให้โมเดลนั้นค่อยๆเรียนรู้ในจำนวน epoch ที่น้อย

**ผลลัพธ์**

รูปภาพประกอบด้วย ไลน์, ข้อความ, ภาพหน้าจอ

คำอธิบายที่สร้างโดยอัตโนมัติ

รูปที่ 26 แสดงกราฟ Training loss กับ Validation loss ของ Hyperparameters แบบที่ 2

จากรูปที่ 26 ผลลัพธ์ที่ได้พบว่า validation loss น้อยกว่าเดิมแต่ดูจากกราฟโมเดลก็ยังคงไม่เรียนรู้ dataset เช่นเดิม

จากผลลัพธ์ทั้ง 2 แบบพบว่าโมเดลนั้นก็ยังไม่ได้เรียนรู้ dataset เช่นเดียวกับการทดสอบที่ 1 และ 2 เราจึงสันนิษฐานว่าเป็นที่ dataset ของเราหรือเปล่าที่เป็นภาษาไทย

* 1. **ทดสอบ fine-tuning pre-trained model ที่แตกต่างกัน**

ต่อมาในการทดสอบที่ 3 เราได้ลองเปลี่ยน pre-trained model ของเราที่จะนำมา fine-tune เป็น babbage-002 ที่เป็น pre-trained model ที่มีขนาดเล็กลงมา

**โมเดลที่ 2 (Pre-trained model: babbage-002)**

รูปภาพประกอบด้วย ข้อความ, ภาพหน้าจอ, แสดง, ซอฟต์แวร์

คำอธิบายที่สร้างโดยอัตโนมัติ

รูปที่ 27 แสดง Pre-trained model: babbage-002

จากรูป 27 มี hyperparameters แบบเดิมกับการทดสอบแรก ดังรูปที่ 16

**ผลลัพธ์**

รูปภาพประกอบด้วย ไลน์, พล็อต, ข้อความ, ภาพหน้าจอ

คำอธิบายที่สร้างโดยอัตโนมัติ

รูปที่ 28 แสดงกราฟ Training loss กับ Validation loss ของโมเดล babbage-002

จากรูป 28 ผลลัพธ์ที่ได้พบว่าโมเดลก็ยังไม่สามารถเรียนรู้ได้เหมือนเดิมเราจึงเริ่มที่จะไปสนใจในการปรับปรุง dataset แทนที่จะปรับ hyperparameters

* 1. **ทดสอบ fine-tuning data set คนละภาษา**

รูปภาพประกอบด้วย ไลน์, พล็อต, ข้อความ, ภาพหน้าจอ

คำอธิบายที่สร้างโดยอัตโนมัติ

รูปที่ 29 แสดงกราฟ Training loss กับ Validation loss ซึ่งเป็นผลลัพธ์ของ gpt-3.5-turbo model

รูปภาพประกอบด้วย ไลน์, พล็อต, ภาพหน้าจอ, แผนภาพ

คำอธิบายที่สร้างโดยอัตโนมัติ

รูปที่ 30 แสดงกราฟ Training loss กับ Validation loss ซึ่งเป็นผลลัพธ์ babbage-002

จากรูปที่ 29 และ 30 พบว่าต่อให้เปลี่ยน dataset เป็นภาษาอังกฤษโมเดลก็ยังไม่เรียนรู้เช่นเดียวกันกับการทดสอบอื่นๆ

1. **Conclusion**

จากการทดสอบทั้ง 4 อย่างพบว่าผลลัพธ์ไม่ได้เป็นไปอย่างที่คาดหวังไว้เนื่องจากคุณภาพของ dataset ยังดีไม่มากพอรวมไปถึงขนาดของข้อมูลที่มีขนาดน้อยเกินไป ดูได้จากผลการทดสอบที่ 1 ที่ลองเปลี่ยนรูปแบบของ dataset พบว่า output ที่ได้นั้นดีขึ้นในเชิงของการเรียบเรียงประโยคแต่ก็ยังไม่ใช่คำตอบที่มาจาก dataset รวมไปถึงการทดลองที่ 2 และ 3 ที่ลองเปลี่ยน hyperparameters หรือ pre-trained model ก็ยังไม่สามารถเรียนรู้ dataset ได้เช่นเดียวกันจนมาถึงการทดสอบที่ 4 ที่ลองเปลี่ยนเป็นภาษาอังกฤษซึ่งทำให้มี token น้อยลงแต่ค่า loss กับเพิ่มขึ้นและไม่ได้เรียนรู้ dataset

**References**

1. Amazon (n.d.), “ตัวอย่างของแบบจำลองพื้นฐานมีอะไรบ้าง", เข้าถึงได้จาก: <https://aws.amazon.com/th/what-is/foundation-models/>, [ออนไลน์; วันที่สืบค้น 15 พฤศจิกายน 2566].
2. Fazmin Nizam, 2566, "Fine-tuning OpenAI GPT 3.5: OpenAI’s latest fine-tuning API", เข้าถึงได้จาก: <https://medium.com/@fazmin/fine-tuning-new-openai-gpt-3-5-practical-example-with-python-b2653f2a5b65>, [ออนไลน์; วันที่สืบค้น 15 พฤศจิกายน 2566].
3. Simón Fishman, 2566, "How to fine-tune chat models", เข้าถึงได้จาก: <https://cookbook.openai.com/examples/how_to_finetune_chat_models>; วันที่สืบค้น 17 พฤศจิกายน 2566].
4. Amazing Thailand (n.d.), "สถานที่ท่องเที่ยว", เข้าถึงได้จาก:<https://thai.tourismthailand.org/Search-result/attraction?destination_id=101&sort_by=datetime_updated_desc&page=1&perpage=15&menu=attraction>; วันที่สืบค้น 8 พฤศจิกายน 2566].