

# Fine-tuning OpenAI API model for specific task

นาย กิตติพัฒน์ เรืองอมรวัฒน์ 63070501006

นาย สันหนัฐ พรหมจรรย์ 63070501069

Final Project – Coding in AI 2023

King Mongkut's University of Technology Thonburi

---

## Abstract

OpenAI ย่อมาจาก Open Artificial Intelligence ซึ่งหมายถึง การพัฒนาและให้บริการเทคโนโลยี ปัญญาประดิษฐ์และปัญญาประดิษฐ์ทางธุรกิจที่เปิดเผย (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 จะใช้ทำนายคำหรือตัวหนังสือถัดไปโดยพิจารณาจากที่เป็นบ่งบอกทั้งหมดที่อยู่ข้างหน้า (context) ของมัน ซึ่งทำให้มีการสื่อสารที่มีความเป็นธรรมชาติมากขึ้น และเชื่อมโยงกับประสบการณ์ของผู้ใช้ [1] โดยสำหรับ โมเดล GPT-3 ได้มีการใช้วีรลเน็ตเวิร์ก 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 แนวทางทั่วไปสำหรับประสิทธิภาพเฉพาะงานคือการใช้ 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 ได้ปฏิบัติแอปพลิเคชัน 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 ตรงที่โมเดลที่ทำการ fine-tuning สามารถสร้างขึ้นเพื่อให้ได้รับการตอบสนองที่แม่นยำยิ่งขึ้นและเกี่ยวข้องกับโดเมนหรือ dataset ที่เฉพาะเจาะจง ซึ่งช่วยให้สามารถควบคุมศักยภาพสูงสุดของโมเดล ซึ่งในโปรเจกต์นี้จะทำเกี่ยวกับการ Fine-tuning OpenAI API model สำหรับงานเฉพาะเจาะจง โดยจะมีโดเมนเฉพาะเจาะจง คือการให้ข้อมูลเกี่ยวกับสถานที่ท่องเที่ยวในจังหวัดเชียงใหม่ จากนั้นจะนำ dataset ของสถานที่ท่องเที่ยวในจังหวัดเชียงใหม่มาทำการ fine-tuning ให้กับแต่ละโมเดลของ OpenAI

---

## 2. Theory and Techniques

### 2.1. Unsupervised Fine-tuning

Unsupervised Fine-tuning เป็นส่วนสำคัญในการปรับโมเดล OpenAI API ให้เข้ากับงานเฉพาะ โมเดลที่ pre-trained เบื้องต้นจะช่วยให้เข้าใจภาษาได้กว้าง แต่อาจไม่ได้รับการปรับให้เหมาะสมกับการใช้งานเฉพาะเจาะจง โดยไม่มีการระบุข้อมูลที่ชัดเจน ซึ่ง Unsupervised Fine-tuning จะช่วยให้โมเดลสามารถปรับให้เข้ากับโดเมนหรือข้อกำหนดเฉพาะโดยไม่มี label เฉพาะงานที่ชัดเจน ซึ่งในระหว่างการ Unsupervised Fine-tuning โมเดลจะแสดงกับข้อมูลเฉพาะงาน หากไม่มี keyword ที่ชัดเจนเกี่ยวกับเอาต์พุตที่ต้องการ โมเดลจะปรับพารามิเตอร์ตามรูปแบบและคุณลักษณะที่มีอยู่ในข้อมูลอินพุต โดยปรับข้อมูลให้เข้ากับ ความแตกต่างของงานที่กำหนดได้อย่างมีประสิทธิภาพ

### 2.2. Pre-training Model

Pre-training เป็นรากฐานของความสามารถของโมเดล OpenAI API ช่วยให้เข้าใจภาษาและบริบทในวงกว้างจาก dataset ที่หลากหลาย สำหรับประสิทธิภาพของการ 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

#### 2.2.1. GPT-3.5-Turbo-1106

GPT-3.5 Turbo เป็นโมเดลที่ถูกเผยแพร่โดย Anthropic ในเดือนพฤศจิกายน 2022 เป็นโมเดลที่ใหญ่และมีประสิทธิภาพมากที่สุดในปัจจุบัน ด้วยพารามิเตอร์มากกว่า 175 พันล้านพารามิเตอร์ ต้องใช้ทรัพยากรการ

ประมวลผลในระดับมหาศาลเพื่อการ train และทำนาย มีขนาดใหญ่มากจนองค์กรส่วนมากไม่สามารถ train โมเดลนี้ได้ พารามิเตอร์จำนวนมหาศาลทำให้รองรับข้อมูลในการ train ได้ในระดับหลายล้านคำ แม้ว่าจะมีความสามารถในการสนทนาเหมือนมนุษย์มากที่สุดเท่าที่เคยพบ แต่การนำไปใช้งานจริงก็ต้องการทรัพยากรและงบประมาณจำนวนมหาศาลเช่นกัน สำหรับองค์กรส่วนใหญ่แล้ว GPT-3.5 Turbo ถือเป็น overkill สำหรับการใช้งานทั่วไป ในส่วนโมเดลขนาดเล็กกว่าอย่าง Babbage และ Davinci จึงคุ้มค่ามากกว่า

### 2.2.2. Babbage-002

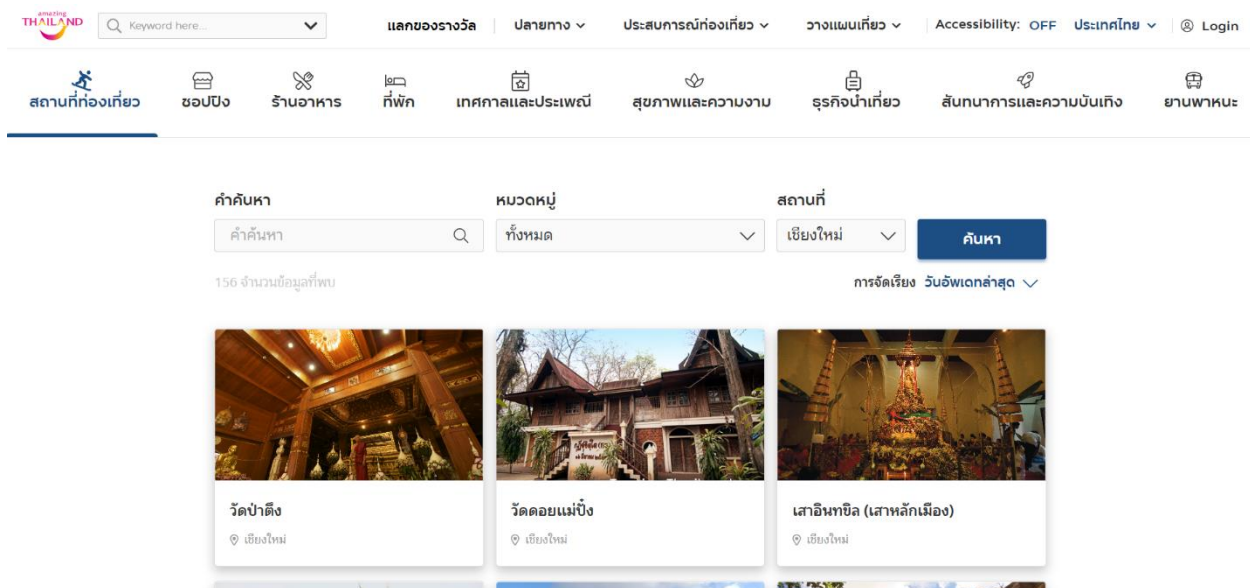
Babbage-002 เป็นโมเดลปัญญาประดิษฐ์สำหรับการสนทนาที่พัฒนาโดย Anthropic เพื่อให้มีประโยชน์และมีความปลอดภัย มีพารามิเตอร์จำนวน 1.3 พันล้านตัว ทำให้เป็นโมเดลขนาดกลางที่ถูกออกแบบมาเพื่อสมดุลระหว่างประสิทธิภาพและความคุ้มค่า Babbage ถูกออกแบบเฉพาะสำหรับการใช้งานจริง เช่น chatbot และ assistant ใช้ข้อมูลและการประมวลผลน้อยกว่าโมเดลอื่นหลายเท่าในการฝึก ทำให้ Babbage มีค่าใช้จ่ายต่ำในการนำมาใช้งาน สามารถฝึกให้สนทนาอย่างปลอดภัยได้หลากหลายบริบทโดยไม่ต้องใช้ข้อมูลจำนวนมาก Babbage ให้คุณภาพการสนทนาที่ดีและความสามารถในการให้เหตุผลโดยไม่ต้องเสียค่าใช้จ่ายสูงหรือมีความเสี่ยงมาก ขนาดที่เหมาะสมช่วยให้หลีกเลี่ยงปัญหาต่าง ๆ เช่น การสร้างข้อมูลเท็จหรือข้อความ toxic ที่เกิดกับโมเดลขนาดใหญ่ Anthropic ออกแบบ Babbage ให้สามารถใช้งานได้จริงและเข้าถึงได้สำหรับองค์กรส่วนใหญ่แทนที่จะเน้นประสิทธิภาพสูงสุด Babbage ตอบโจทย์การประยุกต์ใช้ AI สนทนาได้หลากหลายมากขึ้น แทนที่จะเน้นเพียงประสิทธิภาพเหมือนมนุษย์

## 2.3. Data Collection

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

### 2.3.1. Data Sources

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



รูปที่ 1 แสดงหน้าเว็บไซต์ที่ใช้เป็นแหล่งข้อมูลซึ่งข้อมูลจะเกี่ยวกับสถานที่ท่องเที่ยวในจังหวัดเชียงใหม่

### 2.3.2. Selenium Library

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

### 2.3.3. Dataset Format

Dataset สำคัญในการกำหนดประสิทธิภาพของโมเดล และการจัดเตรียมประกอบด้วยกระบวนการสำคัญหลายประการเพื่อให้แน่ใจว่ามีความเกี่ยวข้อง ความหลากหลาย และประสิทธิภาพ 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

```

1 {"messages": [{"role": "system", "content": "Marv is a factual chatbot that is a"}
2 {"messages": [{"role": "system", "content": "Marv is a factual chatbot that is a"}
3 {"messages": [{"role": "system", "content": "Marv is a factual chatbot that is a"}

```

รูปที่ 2 แสดงสามตัวอย่างรูปแบบ dataset ที่ใช้สำหรับโมเดลเวอร์ชันใหม่

```

1 {"prompt": "<prompt text>", "completion": "<ideal generated text>"}
2 {"prompt": "<prompt text>", "completion": "<ideal generated text>"}
3 {"prompt": "<prompt text>", "completion": "<ideal generated text>"}

```

รูปที่ 3 แสดงสามตัวอย่างรูปแบบ dataset ที่ใช้สำหรับโมเดลเวอร์ชันเก่า

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

## 2.4. Fine-tuning pre-trained model

ในการ Fine-tuning เรามีจุดมุ่งเน้นเพื่อจะให้โมเดลนั้นสามารถทำงานตามที่เราต้องการได้ โดยงานนั้น คือการให้ข้อมูลสถานที่ท่องเที่ยวในเชียงใหม่ โดยข้อมูลนั้นจะต้องมาจาก dataset ของเราโดยโมเดลที่เรานำมา fine-tune นั้นมี 2 โมเดลนั้น 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 มีจำนวนข้อมูลที่ย่อยกว่าประสิทธิภาพดีกว่าแต่ทำงานได้ช้ากว่า โดยจุดมุ่งหมายที่เลือกทั้ง 2 โมเดลนี้เพราะเราต้องการอยากทราบว่าข้อมูลของเรานั้นมีความเหมาะสมกับโมเดลแบบไหนมากที่สุด โดยขั้นตอนในการ fine-tuning มีดังนี้

### 2.4.1. Prepare data

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

```
# Convert the list of dictionaries to JSON format
json_table = json.dumps(table_list, ensure_ascii=False, indent=2)

# Save the JSON data to a file
with open('json_table.json', 'w', encoding='utf-8') as json_file:
    json_file.write(json_table)

# Print the JSON data
print(json_table)

[
  {
    "name": "วัดป่าตึง",
    "type": "ประเภท สถานที่ท่องเที่ยววัด",
    "contact": "ข้อมูลติดต่อเบอร์โทร :+66 5388 0061",
    "section": "ข้อมูลที่อยู่ หมู่ 7 สันกำแพง, เชียงใหม่",
    "facility": "เครื่องอำนวยความสะดวกที่จอดรถ",
    "general": "ข้อมูลทั่วไปหลวงปู่หล้า ดาตัพย์ วัดป่าตึง ตั้งอยู่ในเขตตำบลออนใต้ อำเภอสันกำแพง",
    "important": "ข้อมูลสำคัญมาตรการด้านสุขอนามัยและความปลอดภัย มีการทำความสะอาดสถานที่"
  },
]
```

รูปที่ 4 แสดง Code เกี่ยวกับการจัดเก็บ dataset ในขั้นตอน Web-Scraping

```
# Extract and combine data from key="name", "type", "general" for each entry
assistant_content = [
    f'{entry["general"]}'
    for entry in json_table
]

# Remove "ข้อมูลทั่วไป" from the beginning of each entry in assistant_content
assistant_content = [assis.replace("ข้อมูลทั่วไป", "") for assis in assistant_content]

# Print the resulting list
print(assistant_content)

# Add the desired prefix to each element in result list and store in user_content list
user_content = [f"ชื่อสถานที่: {entry['name']}\n\n{entry['type']}\n\nข้อมูลรายละเอียดสถานที่: "
                for entry in json_table]

# Print the resulting list
print(user_content)

# Create system_content by adding the specified word to each element in user_content
system_content = ["You are a friendly and polite guide who give the travel place information in Thailand." for content in user_content]

# Print the resulting list
print(system_content)
```

รูปที่ 5 แสดง Code เกี่ยวกับการสร้างตัวแปรเพื่อนำไปแปลง Data format

```

# Format the data into the desired format
data_role = [
    {
        "messages": [
            {"role": "system", "content": system_content[i]},
            {"role": "user", "content": user_content[i]},
            {"role": "assistant", "content": assistant_content[i]}
        ]
    }
    for i in range(len(system_content))
]

# Convert the list of dictionaries to JSON format with all values on the same line
data_role_json = "\n".join(json.dumps(entry, ensure_ascii=False, separators=(',', ': ')) for entry in data_role)

# Save the formatted data to a JSON file
with open('data_role.jsonl', 'w', encoding='utf-8') as json_file:
    json_file.write(data_role_json)

# Create a list of dictionaries
data_prompt = [
    {
        "prompt": user_content[i], "completion": assistant_content[i]
    }
    for i in range(len(user_content))
]

# Convert the list of dictionaries to JSON format with all values on the same line
data_prompt_json = "\n".join(json.dumps(entry, ensure_ascii=False, separators=(',', ': ')) for entry in data_prompt)

# Save the formatted data to a JSON file
with open('data_prompt.jsonl', 'w', encoding='utf-8') as json_file:
    json_file.write(data_prompt_json)

```

รูปที่ 6 แสดง Code เกี่ยวกับการแปลง Data format

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

#### 2.4.2. Upload training and validation data

```

import random
import json

# Load your JSON data from a file
with open("fine_tune_data1.jsonl", "r") as file:
    dataset = file.readlines()

# Set the seed for reproducibility
random.seed(42)

# Shuffle your dataset
random.shuffle(dataset)

# Define the split ratio (e.g., 80% training, 20% validation)
split_ratio = 0.8
split_index = int(len(dataset) * split_ratio)

# Split the dataset
training_data = dataset[:split_index]
validation_data = dataset[split_index:]

# Write the training and validation data to separate JSONL files
with open("training_data.jsonl", "w") as training_file:
    training_file.writelines(training_data)

with open("validation_data.jsonl", "w") as validation_file:
    validation_file.writelines(validation_data)

```

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

```

training_data = openai.files.create(
    file=open("training_data.jsonl", "rb"),
    purpose="fine-tune"
)

validation_data = openai.files.create(
    file=open("validation_data.jsonl", "rb"),
    purpose="fine-tune"
)

print(training_data)
print(validation_data)

```

FileObject(id='file-co6ZMnWyGygwbWNFCedz3hi8',  
FileObject(id='file-EsII02TMtG7ZQhTB3AtmCpct',

รูปที่ 8 แสดง Code เกี่ยวกับ file id ที่ใช้ในการ fine-tuning

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

<https://platform.openai.com/files/file-co6ZMnWyGygwbWNFCedz3hi8>

### Files

Date	Name	Desc	Asc
	babbage_training_dataen.jsonl		11/20/2023, 4:19 PM
	step_metrics.csv		11/20/2023, 4:01 AM
	babbage_validation_data.jsonl		11/20/2023, 3:58 AM
	babbage_training_data.jsonl		11/20/2023, 3:58 AM
	step_metrics.csv		11/20/2023, 1:13 AM
	step_metrics.csv		11/19/2023, 8:57 PM
	step_metrics.csv		11/19/2023, 8:50 PM
	step_metrics.csv		11/19/2023, 8:41 PM
	validation_data.jsonl		11/19/2023, 8:34 PM
	training_data.jsonl		11/19/2023, 8:34 PM

### FILE

#### training\_data.jsonl

- File ID: file-co6ZMnWyGygwbWNFCedz3hi8
- Status: Ready
- Purpose: fine-tune
- Size: 126,956 bytes
- Created at: Nov 19, 2023, 8:34 PM

รูปที่ 9 แสดงไฟล์ที่อัปโหลดมาแล้วถูกนำไปเก็บไว้ที่ Base URL ของ API



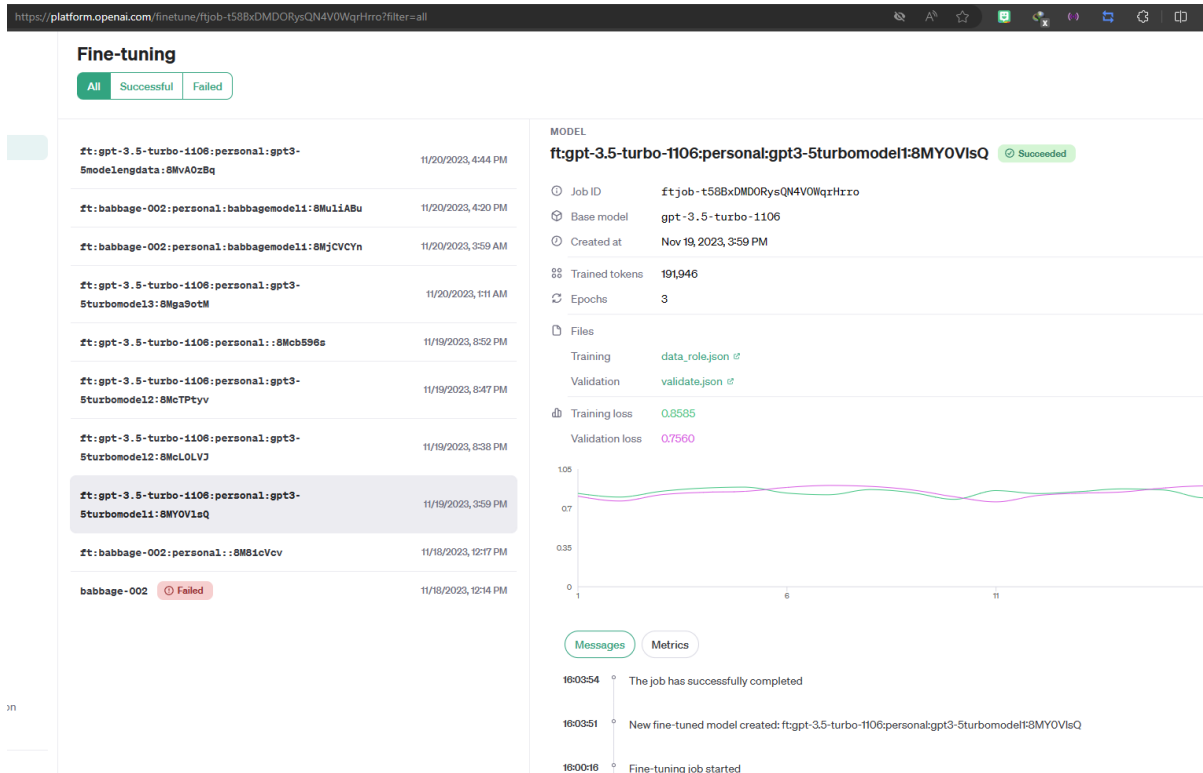
### 2.4.3. Fine-tuning the model

เมื่อ data พร้อมแล้วในขั้นตอนต่อไปก็จะเริ่มการ fine-tuning

```
model1 = openai.fine_tuning.jobs.create(  
    training_file="file-lUVKIZYrqECiJs23499L5a7T",  
    validation_file="file-LM6xSQnLCerJ846jS3KlY6VM",  
    model="gpt-3.5-turbo-1106",  
    suffix="gpt3.5turbomodel1",  
    hyperparameters={  
        "n_epochs": 3,  
        "batch_size": 8,  
        "learning_rate_multiplier": 0.01  
    }  
)
```

รูปที่ 10 แสดง Code เกี่ยวกับการใช้ API เพื่อทำการ fine-tuning

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



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

```
fine_tuning_job_id = "ftjob-t58BxDMDORysQN4V0WqrHrro"

# Retrieve fine-tuning job events
events = openai.fine_tuning.jobs.list_events(fine_tuning_job_id=fine_tuning_job_id,limit=50).data
events.reverse()

# Print each event vertically
for event in events:
    # print(f"Event ID: {event.id}")
    # print(f"Created At: {event.created_at}")
    # print(f"Level: {event.level}")
    # print(f"Message: {event.message}")
    # print(f"Object: {event.object}")
    print(f>Data: {event.data}")
    # print(f"Type: {event.type}\n")
```

รูปที่ 12 แสดง Code เกี่ยวกับการให้ monitor แสดงค่า loss ในแต่ละ step

```
Data: {'step': 1, 'train_loss': 0.8333396911621094, 'valid_loss': 0.888706806716307, 'train_mean_token_accuracy': 0.7819428039955139, 'valid_mean_token_accuracy': 0.35288335517693314}
Data: {'step': 2, 'train_loss': 0.8895911787878113, 'valid_loss': 0.7653575043827893, 'train_mean_token_accuracy': 0.7866817818458928, 'valid_mean_token_accuracy': 0.3551568379918589}
Data: {'step': 3, 'train_loss': 0.8524137139328374, 'valid_loss': 0.8217417136469294, 'train_mean_token_accuracy': 0.770646952758789, 'valid_mean_token_accuracy': 0.3496378482228627}
Data: {'step': 4, 'train_loss': 0.8804341554641724, 'valid_loss': 0.8425188359069671, 'train_mean_token_accuracy': 0.761811375617981, 'valid_mean_token_accuracy': 0.351460221558856}
Data: {'step': 5, 'train_loss': 0.8893145322799683, 'valid_loss': 0.8516218289258786, 'train_mean_token_accuracy': 0.7632192969322285, 'valid_mean_token_accuracy': 0.3468220784164446}
Data: {'step': 6, 'train_loss': 0.8360458016395569, 'valid_loss': 0.886445629821803, 'train_mean_token_accuracy': 0.7769880424575806, 'valid_mean_token_accuracy': 0.36523828594313174}
Data: {'step': 7, 'train_loss': 0.8211231827735901, 'valid_loss': 0.9043076817451525, 'train_mean_token_accuracy': 0.7773622274398804, 'valid_mean_token_accuracy': 0.3489187519325369}
Data: {'step': 8, 'train_loss': 0.8678542971611823, 'valid_loss': 0.8955924964613384, 'train_mean_token_accuracy': 0.7670606978787048, 'valid_mean_token_accuracy': 0.3517361111111114}
Data: {'step': 9, 'train_loss': 0.839046835899353, 'valid_loss': 0.863871925294736, 'train_mean_token_accuracy': 0.7769293785095215, 'valid_mean_token_accuracy': 0.3541666666666667}
Data: {'step': 10, 'train_loss': 0.7803656458854675, 'valid_loss': 0.8832443668899671, 'train_mean_token_accuracy': 0.7922841310501099, 'valid_mean_token_accuracy': 0.35255570117955437}
Data: {'step': 11, 'train_loss': 0.8583548665846692, 'valid_loss': 0.7577265214725944, 'train_mean_token_accuracy': 0.7710498571395874, 'valid_mean_token_accuracy': 0.35549525101763907}
Data: {'step': 12, 'train_loss': 0.8315212138546657, 'valid_loss': 0.8148157398868271, 'train_mean_token_accuracy': 0.7777368426322937, 'valid_mean_token_accuracy': 0.3496378482228627}
Data: {'step': 13, 'train_loss': 0.8492827415466389, 'valid_loss': 0.8349338984693759, 'train_mean_token_accuracy': 0.7781692581176758, 'valid_mean_token_accuracy': 0.35112453843571667}
Data: {'step': 14, 'train_loss': 0.8735841512688054, 'valid_loss': 0.8454584144211832, 'train_mean_token_accuracy': 0.7629884587097168, 'valid_mean_token_accuracy': 0.34648274076893796}
Data: {'step': 15, 'train_loss': 0.8641346096992493, 'valid_loss': 0.8886316599542253, 'train_mean_token_accuracy': 0.773374617099762, 'valid_mean_token_accuracy': 0.36523828594313174}
Data: {'step': 16, 'train_loss': 0.7934806942939758, 'valid_loss': 0.9004088161867937, 'train_mean_token_accuracy': 0.7838431596755981, 'valid_mean_token_accuracy': 0.3489187519325369}
Data: {'step': 17, 'train_loss': 0.8631436824798584, 'valid_loss': 0.892247677842776, 'train_mean_token_accuracy': 0.7780554728507996, 'valid_mean_token_accuracy': 0.3517361111111114}
Data: {'step': 18, 'train_loss': 0.8458332419395447, 'valid_loss': 0.8604461227433157, 'train_mean_token_accuracy': 0.7753578424453735, 'valid_mean_token_accuracy': 0.35288335517693314}
Data: {'step': 19, 'train_loss': 0.8363702893257141, 'valid_loss': 0.7992794129357894, 'train_mean_token_accuracy': 0.7747975587844849, 'valid_mean_token_accuracy': 0.35288335517693314}
Data: {'step': 20, 'train_loss': 0.85851335255127, 'valid_loss': 0.7559931666427261, 'train_mean_token_accuracy': 0.767275869846344, 'valid_mean_token_accuracy': 0.35549525101763907}
```

รูปที่ 13 แสดงค่า loss ในแต่ละ step

จากรูปที่ 12 และ 13 เมื่อ fine-tuning เสร็จแล้วเราสามารถ monitor ค่า loss ในแต่ละ step ออกมาได้ด้วยตัวเองได้ด้วยการใส่ 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 นั้นจะคิดคำนวณจาก epoch, batch size และขนาด dataset ที่ใช้โดยอัตโนมัติ

#### 2.4.4. Use fine-tuned model

วิธีการนำ fine-tuning model ไปใช้

```
1 import openai

1 # Set your OpenAI API key
2 api_key = "sk-jDrQuXc1VMdQxbsL7V8T3B1bkFJAt4rxXVc7mW8HkcFDFCj"
3
4 # Initialize the OpenAI client with your API key
5 openai.api_key = api_key
6
```

รูปที่ 14 แสดง Code เกี่ยวกับการเชื่อม OpenAI API ด้วย Key

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

```
1 response = openai.chat.completions.create(
2     model="ft:gpt-3.5-turbo-1106:personal:gpt3-5-turbomodel1:8MY0V1sQ",
3     temperature=0,
4     max_tokens=1000,
5     messages=[
6         {"role": "system", "content": "You are a friendly and polite guide who give the travel place information in Thailand."},
7         {"role": "user", "content": "ชื่อสถานที่: ม่อนแจ่ม\n\nประเภท สถานที่ท่องเที่ยวและภูเขา\n\nข้อมูลรายละเอียดสถานที่: "}
8     ]
9 )
10 print(response.choices[0].message)
```

ChatCompletionMessage(content='สวัสดีครับ/ค่ะ! สถานที่ท่องเที่ยว "ม่อนแจ่ม" เป็นสถานที่ท่องเที่ยวที่สวยงามและน่าเที่ยวมากในจังหวัดเชียงใหม่ ม่อนแจ่มเป็นจุดชมวิวที่สวยงามที่สุดของม่อนแจ่ม')

รูปที่ 15 แสดง Code เกี่ยวกับการใช้ generate text เพื่อดู output

จากรูปที่ 15 ในการนำมาใช้ generate text เพื่อดู output เราจะใช้คำสั่ง `openai.chat.completions.create()` ในการ request โดยใช้ชื่อของโมเดลจาก Base URL มาใส่ในพารามิเตอร์ `model` เพื่อเลือกจะใช้โมเดลไหนในการ 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

---

### 3. 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

#### 3.1. 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)

```
model1 = openai.fine_tuning.jobs.create(  
    training_file="file-LUVKIZYrqECiJs23499L5a7T",  
    validation_file="file-LM6xSQnLCerJ846jS3KlY6VM",  
    model="gpt-3.5-turbo-1106",  
    suffix="gpt3.5turbomodel1",  
    hyperparameters={  
        "n_epochs": 3,  
        "batch_size": 8,  
        "learning_rate_multiplier": 0.01  
    }  
)
```

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

epoch: 3

batch size: 8

learning rate: 0.01

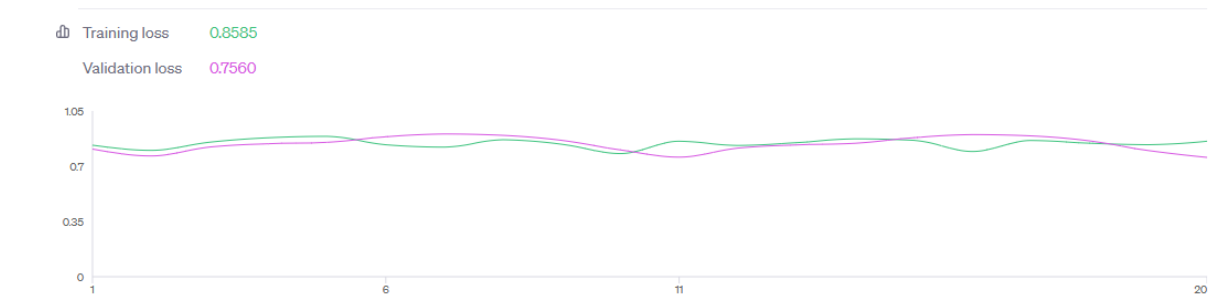
Dataset แบบที่ 1

```
{  
  "messages": [  
    {  
      "role": "system",  
      "content": "คุณเป็นไกด์นำทางที่เป็นมิตรและสุภาพ",  
    },  
    {  
      "role": "user",  
      "content": "ฉันต้องการข้อมูลรายละเอียดเกี่ยวกับม่อนแจ่ม",  
    },  
    {  
      "role": "assistant",  
      "content": "ม่อนแจ่ม เป็นประเภท สถานที่ท่องเที่ยวสวยและภูเขา - ข้อมูล",  
    },  
  ],  
}
```

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



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

```
1 response = openai.chat.completions.create(  
2     model="ft:gpt-3.5-turbo-1106:personal:gpt3-5turbomodel1:8MY0VlsQ",  
3     temperature=0,  
4     max_tokens=1000,  
5     messages=[  
6         {"role": "system", "content": "คุณเป็นไกด์นำทางที่เป็นมิตรและสุภาพ"},  
7         {"role": "user", "content": "ฉันต้องการข้อมูลรายละเอียดเกี่ยวกับม่อนแจ่ม"}  
8     ]  
9 )  
10 print(response.choices[0].message)
```

รูปที่ 19 แสดง Code เกี่ยวกับการใช้ generate text เพื่อดู output

จากรูปที่ 19 คำตอบที่ได้: “ม่อนแจ่ม (Mon Cham) เป็นสถานที่ท่องเที่ยวที่ตั้งอยู่ในอำเภอแมริม จังหวัด เชียงใหม่ ประเทศไทย ม่อนแจ่มเป็นที่รู้จักด้วยทิวทัศน์ที่สวยงามของภูเขาและทุ่งนาที่ล้อมรอบ นอกจากนี้ยังมีสวนผลไม้และสวนดอกไม้ที่สวยงามอีกด้วย\n\nที่ม่อนแจ่มนี้นักท่องเที่ยวสามารถเดินเล่นรอบเขาหรือทำกิจกรรมทางการเกษตรเช่นการเก็บผลไม้หรือดูดอกไม้ได้ นอกจากนี้ยังมีร้านอาหารและรีสอร์ทที่สามารถพักผ่อนและพักค้างคืนได้\n\nการเดินทางไปยังม่อนแจ่มสามารถใช้รถยนต์ส่วนตัวหรือจักรยานจากเชียงใหม่ โดยมีระยะทางประมาณ 40 กิโลเมตร และเส้นทางที่สวยงามและสะดวกสบาย”

## Dataset แบบที่ 2

```

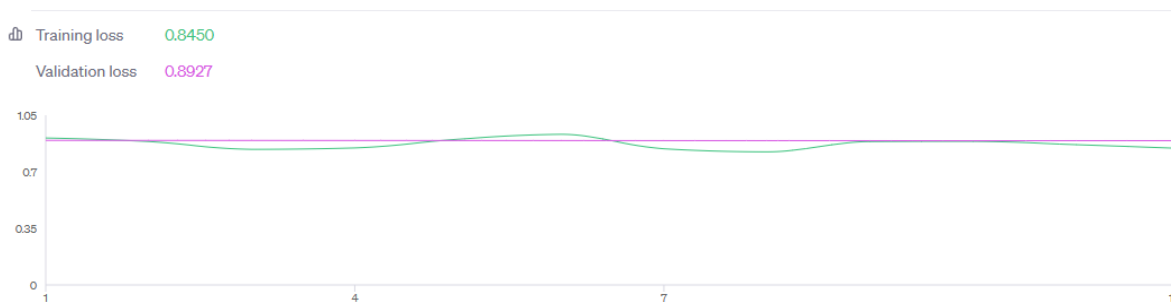
"messages": [
    {
        "role": "system", "content": "You are a friendly and polite guide who give the travel place information in Thailand.",
        "role": "user", "content": "มีสถานที่ท่องเที่ยวที่ไหนที่น่าสนใจในกรุงเทพฯ?",
        "role": "assistant", "content": "สวัสดีครับ! กรุงเทพฯมีสถานที่ท่องเที่ยวที่น่าสนใจมากมาย ไม่ว่าจะเป็นวัดพระแก้ว, พระราชวังสนามจันทร์, หรือสวนสัตว์ดุสิต. หากต้องการข้อมูลเพิ่มเติม กรุณาติดต่อศูนย์บริการนักท่องเที่ยว โทร. 02-123-4567."
    },
    {
        "role": "system", "content": "You are a friendly and polite guide who give the travel place information in Thailand.",
        "role": "user", "content": "มีร้านอาหารอร่อยๆ ในกรุงเทพฯ ไหม?",
        "role": "assistant", "content": "当然มีครับ! กรุงเทพฯมีร้านอาหารอร่อยๆ มากมาย ไม่ว่าจะเป็นร้านอาหารไทย, ร้านอาหารจีน, หรือร้านอาหารฝรั่ง. หากต้องการข้อมูลเพิ่มเติม กรุณาติดต่อศูนย์บริการนักท่องเที่ยว โทร. 02-123-4567."
    },
    {
        "role": "system", "content": "You are a friendly and polite guide who give the travel place information in Thailand.",
        "role": "user", "content": "มีที่พักดีๆ ในกรุงเทพฯ ไหม?",
        "role": "assistant", "content": "当然มีครับ! กรุงเทพฯมีที่พักดีๆ มากมาย ไม่ว่าจะเป็นโรงแรมหรู, โรงแรมราคาประหยัด, หรือที่พักพร้อมอาหารเช้า. หากต้องการข้อมูลเพิ่มเติม กรุณาติดต่อศูนย์บริการนักท่องเที่ยว โทร. 02-123-4567."
    },
    {
        "role": "system", "content": "You are a friendly and polite guide who give the travel place information in Thailand.",
        "role": "user", "content": "มีกิจกรรมดีๆ ในกรุงเทพฯ ไหม?",
        "role": "assistant", "content": "当然มีครับ! กรุงเทพฯมีกิจกรรมดีๆ มากมาย ไม่ว่าจะเป็นการช้อปปิ้ง, การรับประทานอาหาร, หรือการเยี่ยมชมสถานที่ท่องเที่ยว. หากต้องการข้อมูลเพิ่มเติม กรุณาติดต่อศูนย์บริการนักท่องเที่ยว โทร. 02-123-4567."
    },
    {
        "role": "system", "content": "You are a friendly and polite guide who give the travel place information in Thailand.",
        "role": "user", "content": "มีรถสาธารณะในกรุงเทพฯ ไหม?",
        "role": "assistant", "content": "当然มีครับ! กรุงเทพฯมีรถสาธารณะมากมาย ไม่ว่าจะเป็นรถเมล์, รถไฟฟ้า, หรือรถแท็กซี่. หากต้องการข้อมูลเพิ่มเติม กรุณาติดต่อศูนย์บริการนักท่องเที่ยว โทร. 02-123-4567."
    },
    {
        "role": "system", "content": "You are a friendly and polite guide who give the travel place information in Thailand.",
        "role": "user", "content": "มีสถานที่ท่องเที่ยวที่น่าสนใจในเชียงใหม่ไหม?",
        "role": "assistant", "content": "当然มีครับ! เชียงใหม่มีสถานที่ท่องเที่ยวที่น่าสนใจมากมาย ไม่ว่าจะเป็นวัดพระธาตุดอยสุเทพ, พระตำหนักเป๊ตวัน, หรือสวนพฤกษศาสตร์สมเด็จพระนางเจ้าสิริกิติ์. หากต้องการข้อมูลเพิ่มเติม กรุณาติดต่อศูนย์บริการนักท่องเที่ยว โทร. 053-251-111."
    },
    {
        "role": "system", "content": "You are a friendly and polite guide who give the travel place information in Thailand.",
        "role": "user", "content": "มีร้านอาหารอร่อยๆ ในเชียงใหม่ไหม?",
        "role": "assistant", "content": "当然มีครับ! เชียงใหม่มีร้านอาหารอร่อยๆ มากมาย ไม่ว่าจะเป็นร้านอาหารไทย, ร้านอาหารจีน, หรือร้านอาหารฝรั่ง. หากต้องการข้อมูลเพิ่มเติม กรุณาติดต่อศูนย์บริการนักท่องเที่ยว โทร. 053-251-111."
    },
    {
        "role": "system", "content": "You are a friendly and polite guide who give the travel place information in Thailand.",
        "role": "user", "content": "มีที่พักดีๆ ในเชียงใหม่ไหม?",
        "role": "assistant", "content": "当然มีครับ! เชียงใหม่มีที่พักดีๆ มากมาย ไม่ว่าจะเป็นโรงแรมหรู, โรงแรมราคาประหยัด, หรือที่พักพร้อมอาหารเช้า. หากต้องการข้อมูลเพิ่มเติม กรุณาติดต่อศูนย์บริการนักท่องเที่ยว โทร. 053-251-111."
    },
    {
        "role": "system", "content": "You are a friendly and polite guide who give the travel place information in Thailand.",
        "role": "user", "content": "มีกิจกรรมดีๆ ในเชียงใหม่ไหม?",
        "role": "assistant", "content": "当然มีครับ! เชียงใหม่มีกิจกรรมดีๆ มากมาย ไม่ว่าจะเป็นการช้อปปิ้ง, การรับประทานอาหาร, หรือการเยี่ยมชมสถานที่ท่องเที่ยว. หากต้องการข้อมูลเพิ่มเติม กรุณาติดต่อศูนย์บริการนักท่องเที่ยว โทร. 053-251-111."
    },
    {
        "role": "system", "content": "You are a friendly and polite guide who give the travel place information in Thailand.",
        "role": "user", "content": "มีรถสาธารณะในเชียงใหม่ไหม?",
        "role": "assistant", "content": "当然มีครับ! เชียงใหม่มีรถสาธารณะมากมาย ไม่ว่าจะเป็นรถเมล์, รถไฟฟ้า, หรือรถแท็กซี่. หากต้องการข้อมูลเพิ่มเติม กรุณาติดต่อศูนย์บริการนักท่องเที่ยว โทร. 053-251-111."
    }
]

```

รูปที่ 20 แสดงเกี่ยวกับ format Dataset แบบที่ 2

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

ผลลัพธ์



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 มีค่าสูงขึ้นมากและได้ลองไปใช้ถามตอบแบบเดิม

```

1 response = openai.chat.completions.create(
2     model="ft:gpt-3.5-turbo-1106:personal:gpt3-5-turbomodel12:8McL0LVJ",
3     temperature=0,
4     max_tokens=1000,
5     messages=[
6         {"role": "system", "content": "You are a friendly and polite guide who give the travel place information in Thailand."},
7         {"role": "user", "content": "ชื่อสถานที่: ม่อนแจ่ม\nประเภท สถานที่ท่องเที่ยวและภูเขา\nข้อมูลรายละเอียดสถานที่: "}
8     ]
9 )
10 print(response.choices[0].message)

```

รูปที่ 22 แสดง Code เกี่ยวกับการใช้ generate text เพื่อดู output

จากรูปที่ 22 คำตอบที่ได้: “สวัสดีครับ/ค่ะ! สถานที่ท่องเที่ยว "ม่อนแจ่ม" เป็นสถานที่ท่องเที่ยวที่ตั้งอยู่บนเทือกเขาตอย่างงาในจังหวัดเชียงใหม่ ม่อนแจ่มเป็นจุดชมวิวที่สวยงามที่สุดของตอย่างงา ที่มีทิวทัศน์ที่สวยงามของทะเลหมอกและภูเขาที่สลับไส 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 ประโยคและการให้ข้อมูลที่ดีกว่าซึ่งจากผลลัพธ์คาดว่าเพราะ system content ที่เปลี่ยนเป็นภาษาอังกฤษและกำหนดได้ละเอียดกว่า dataset แบบแรกเราจึงตัดสินใจที่จะใช้ dataset แบบที่ 2 ในการทดลองถัดๆไป

### 3.2. 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

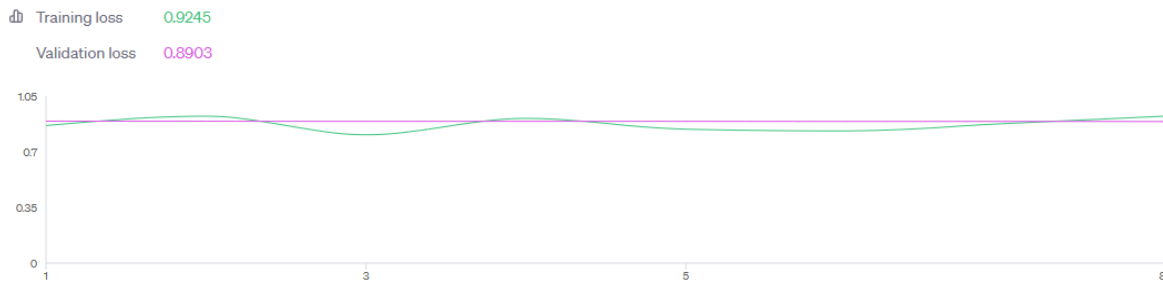
## Hyperparameters แบบที่ 1

```
model3 = openai.fine_tuning.jobs.create(  
    training_file="file-co6ZMnWyGygwbWNFCedz3hi8",  
    validation_file="file-EsII02TMtG7ZQhTB3AtmCpct",  
    model="gpt-3.5-turbo-1106",  
    suffix="gpt3.5turbomodel3",  
    hyperparameters={  
        "n_epochs": 5,  
        "batch_size": 8,  
        "learning_rate_multiplier": 0.01  
    }  
)
```

รูปที่ 23 แสดง Code เกี่ยวกับการเปลี่ยน Hyperparameters แบบที่ 1

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

### ผลลัพธ์



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

จากรูปที่ 24 ผลลัพธ์ที่ได้หลังจากเพิ่ม epoch ไปก็ยังเป็นเหมือนเดิมโมเดลยังไม่สามารถเรียนรู้ dataset ได้จึงได้ลองปรับ hyperparameters เพิ่มเติม



## Hyperparameter แบบที่ 2

```
model4 = openai.fine_tuning.jobs.create(  
    training_file="file-co6ZMnWyGygwbWNFCedz3hi8",  
    validation_file="file-EsII02TmtG7ZQhTB3AtmCpct",  
    model="gpt-3.5-turbo-1106",  
    suffix="gpt3.5turbomodel3",  
    hyperparameters={  
        "n_epochs": 3,  
        "batch_size": 16,  
        "learning_rate_multiplier": 0.001  
    }  
)
```

รูปที่ 25 แสดง Code เกี่ยวกับการเปลี่ยน Hyperparameters แบบที่ 2

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

### ผลลัพธ์



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

### 3.3. ทดสอบ fine-tuning pre-trained model ที่แตกต่างกัน

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

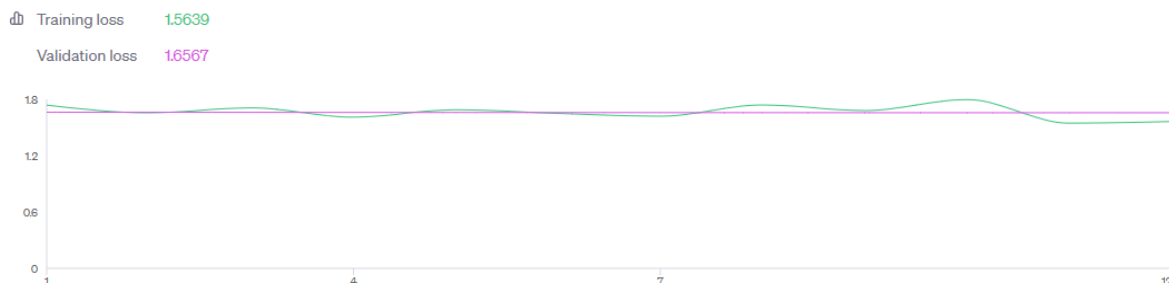
โมเดลที่ 2 (Pre-trained model: babbage-002)

```
model5 = openai.fine_tuning.jobs.create(  
    training_file="file-9NXUyvYYObtIdZDQWDSi6INX",  
    validation_file="file-hq51dEn6Owebj6sJTXCfHZTz",  
    model="babbage-002",  
    suffix="babbagemodel1",  
    hyperparameters={  
        "n_epochs": 3,  
        "batch_size": 8,  
        "learning_rate_multiplier": 0.01  
    }  
)
```

รูปที่ 27 แสดง Pre-trained model: babbage-002

จากรูป 27 มี hyperparameters แบบเดิมกับการทดสอบแรก ดังรูปที่ 16

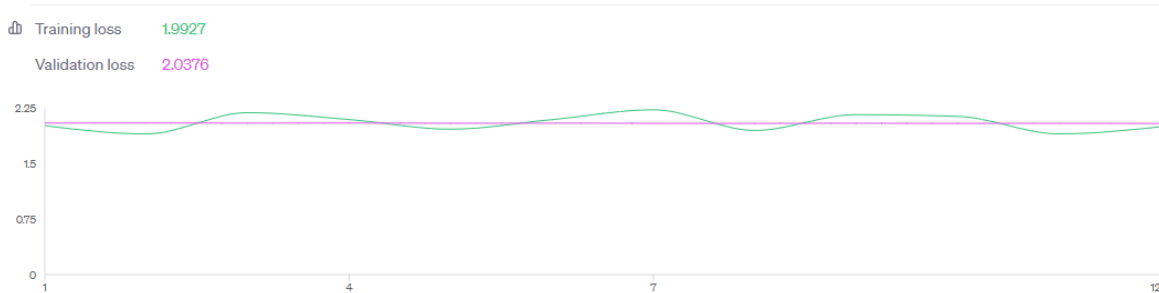
#### ผลลัพธ์



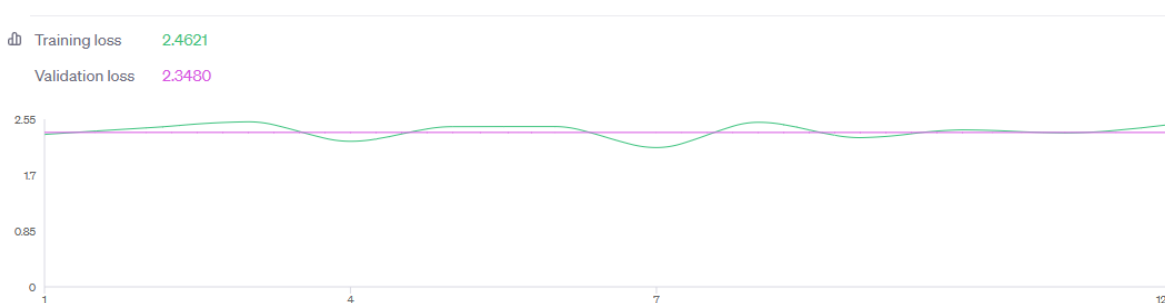
รูปที่ 28 แสดงกราฟ Training loss กับ Validation loss ของโมเดล babbage-002

จากรูป 28 ผลลัพธ์ที่ได้พบว่าโมเดลก็ยังไม่สามารถเรียนรู้ได้เหมือนเดิมเราจึงเริ่มที่จะไปสนใจในการปรับปรุง dataset แทนที่จะปรับ hyperparameters

### 3.4. ทดสอบ fine-tuning data set คนละภาษา



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



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

จากรูปที่ 29 และ 30 พบว่าต่อให้เปลี่ยน dataset เป็นภาษาอังกฤษโมเดลก็ยังไม่เรียนรู้เช่นเดียวกันกับการทดสอบอื่นๆ

## 4. Conclusion

จากการทดสอบทั้ง 4 อย่างพบว่าผลลัพธ์ไม่ได้เป็นไปอย่างที่คาดหวังไว้เนื่องจากคุณภาพของ dataset ยังดีไม่มากพอรวมไปถึงขนาดของข้อมูลที่มีขนาดน้อยเกินไป ดูได้จากผลการทดสอบที่ 1 ที่ลองเปลี่ยนรูปแบบของ dataset พบว่า output ที่ได้มันดีขึ้นในเชิงของการเรียงเรียงประโยคแต่ก็ยังไม่ใช่คำตอบที่มาจาก dataset รวมไปถึงการทดลองที่ 2 และ 3 ที่ลองเปลี่ยน hyperparameters หรือ pre-trained model ก็ยังไม่สามารถเรียนรู้ dataset ได้เช่นเดียวกันจนมาถึงการทดสอบที่ 4 ที่ลองเปลี่ยนเป็นภาษาอังกฤษซึ่งทำให้มี token น้อยลงแต่ค่า loss ก็เพิ่มขึ้นและไม่ได้เรียนรู้ 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](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](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].