

การพัฒนาและเปรียบเทียบประสิทธิภาพของโมเดลภาษาขนาดใหญ่สำหรับการสร้างกรณีทดสอบอัตโนมัติในงานทดสอบซอฟต์แวร์ระบบธนาคาร

Development and comparative evaluation of large language models for automated test case generation in banking software testing

หทัยรัตน์ เจนวิทยา¹ และรัฐศิลป์ รานอกphanuwat^{1*}

Hathairat Janwittaya¹ and Ratthaslip Ranokphanuwat^{1*}

Received: 4 August 2025 : Revised: 25 September 2025 : Accepted: 21 October 2025

บทคัดย่อ

1) การทดสอบซอฟต์แวร์หรือการสร้างกรณีทดสอบ (Test Case) เป็นขั้นตอนสำคัญที่ใช้เวลาและต้องอาศัยความเชี่ยวชาญ โดยเฉพาะในระบบที่ซับซ้อนและมีความเสี่ยงสูง เช่น ระบบธุรกรรมทางการเงินของธนาคาร เนื่องจากการสร้างกรณีทดสอบใช้เวลานานและอาจเกิดข้อผิดพลาด จึงมีความจำเป็นต้องประยุกต์ใช้เทคโนโลยีปัญญาประดิษฐ์เพื่อเพิ่มความรวดเร็วและครอบคลุมในการสร้างกรณีทดสอบ 2) งานวิจัยพัฒนาและเปรียบเทียบประสิทธิภาพของ Large Language Models (LLMs) จำนวน 4 โมเดล ได้แก่ LLaMA3.2-3B, Typhoon2-8b, Gemma-3-4b และ Qwen3-8B สำหรับการสร้างกรณีทดสอบอัตโนมัติในงานทดสอบซอฟต์แวร์ 3) วิธีการดำเนินงานวิจัยประกอบด้วย 4 ขั้นตอนหลัก ได้แก่ 3.1) ชุดข้อมูลกรณีทดสอบ 20,000 รายการ ครอบคลุมกรณี Positive 15,763 รายการ และ Negative 4,237 รายการ ในบริบทธุรกรรมธนาคาร 3.2) ปรับแต่งโมเดลด้วยเทคนิค LoRA และเฟรมเวิร์ก LangChain เชื่อมต่อกับฐานข้อมูลเวกเตอร์ ChromaDB ผ่านสถาปัตยกรรม Retrieval-Augmented Generation (RAG) 3.3) ประเมินผลใช้ตัวชี้วัดเชิงเวกเตอร์ ได้แก่ Cosine Distance, Euclidean Distance และ Manhattan Distance 3.4) ประเมินความครอบคลุมของ Test Case ใช้ตัวชี้วัด Test Coverage, Functional Coverage และ Requirement Coverage ร่วมกับความพึงพอใจจากผู้เชี่ยวชาญด้านการทดสอบซอฟต์แวร์ 4) ผลการวิจัยพบว่า Qwen3-8B มีค่า Loss ต่ำสุด (0.1558) มีความแม่นยำในการเรียนรู้สูงสุด ขณะที่ Gemma-3-4b มีค่า Euclidean Distance (0.5494) และ Manhattan Distance (11.8533) ต่ำสุด แสดงถึงความใกล้เคียงกับข้อมูลจริงมากที่สุด ในด้านความครอบคลุมพบว่า Gemma-3-4b มีค่า Test Coverage สูงสุด (93%) และ Functional Coverage (91%) ส่วน Qwen3-8B มี Requirement Coverage สูงสุด (92%) และจากการประเมินผู้เชี่ยวชาญพบว่า Gemma-3-4b ได้คะแนนเฉลี่ยสูงสุด (43.2) คิดเป็นร้อยละ (86.4%) สรุปผลการศึกษาการใช้ LLM ในการสร้าง Test Case สามารถเพิ่มความครอบคลุม ความแม่นยำ และประสิทธิภาพในงานทดสอบซอฟต์แวร์ได้อย่างมีนัยสำคัญ อีกทั้งยังสามารถประยุกต์ใช้งานผ่านระบบแชทบอทเพื่อสร้างกรณีทดสอบ จากข้อกำหนดความต้องการซอฟต์แวร์ได้โดยอัตโนมัติ ซึ่งมีศักยภาพสูงสำหรับการใช้งานจริงในกระบวนการพัฒนาซอฟต์แวร์ในอุตสาหกรรม

คำสำคัญ: โมเดลภาษาขนาดใหญ่, การทดสอบซอฟต์แวร์, การสร้างกรณีทดสอบ, ตัวชี้วัดความครอบคลุมในการทดสอบ, การฝังเวกเตอร์

Abstract

1) Software testing and test case generation are critical steps that require significant time and expertise, especially in complex and high-risk environments such as banking transaction systems. Since creating test cases is time-consuming and prone to errors, it is essential to apply artificial intelligence technologies to accelerate the process and enhance test coverage. 2) This research develops and compares the performance of four Large Language Models (LLMs):

¹ หลักสูตรวิศวกรรมศาสตรมหาบัณฑิต, สาขาวิชาวิศวกรรมคอมพิวเตอร์, วิทยาลัยวิศวกรรมศาสตร์และเทคโนโลยี, มหาวิทยาลัยบูรพา

¹ Department of Computer Engineering, College of Engineering and Technology, Dhurakij Pundit University

* Corresponding author: Ratthaslip Ranokphanuwat, Email: ratthaslip.ran@dpu.ac.th

LLaMA3.2-3B, Typhoon2-8b, Gemma-3-4b, and Qwen3-8B, for automated test case generation in software testing.

3) The research methodology consists of four main steps: 3.1) Utilizing a dataset of 20,000 test cases, which covers 15,763 positive and 4,237 negative cases within the banking transaction domain. 3.2) Fine-tuning the models using the LoRA technique and the LangChain framework, integrated with the ChromaDB vector database through a Retrieval-Augmented Generation (RAG) architecture. 3.3) Evaluating performance using vector-based metrics, including Cosine Distance, Euclidean Distance, and Manhattan Distance. 3.4) Assessing test case coverage using Test Coverage, Functional Coverage, and Requirement Coverage, combined with evaluations from software testing experts. 4) The results indicate that Qwen3-8B achieved the lowest loss (0.1558), demonstrating the highest learning accuracy, while Gemma-3-4b obtained the lowest Euclidean Distance (0.5494) and Manhattan Distance (11.8533), indicating the closest similarity to the ground-truth data. In terms of coverage, Gemma-3-4b achieved the highest Test Coverage (93%) and Functional Coverage (91%), whereas Qwen3-8B achieved the highest Requirement Coverage (92%). Furthermore, expert evaluations revealed that Gemma-3-4b received the highest average score of 43.2, equivalent to 86.4%. In conclusion, the study demonstrates that employing LLMs for test case generation can significantly improve the coverage, accuracy, and efficiency of software testing. Additionally, these models can be seamlessly integrated into chatbot systems to automatically generate test cases based on software requirements, highlighting their high practical potential for real-world applications in the software development industry.

Keywords: Large language models, software testing, test case generation, test coverage metrics, vector embedding

บทนำ

ที่มาและความสำคัญ

ในกระบวนการพัฒนาซอฟต์แวร์ การสร้างกรณีทดสอบ Test Case สำหรับการทดสอบแบบ Manual Testing ถือเป็นภารกิจสำคัญของ Software Tester และ Software QA เพื่อรับรองคุณภาพ ความถูกต้อง และความครบถ้วนของระบบ โดยเฉพาะในระบบขนาดใหญ่ที่มีข้อกำหนดซับซ้อน การสร้าง Test Case ที่ครอบคลุมและแม่นยำต้องอาศัยเวลา ความชำนาญ และแรงงานคนเป็นจำนวนมาก ส่งผลให้เกิดข้อจำกัดด้านความเร็วและความครอบคลุมของการทดสอบ อาจนำไปสู่ความผิดพลาดที่หลุดรอดหลังการส่งมอบงาน

ปัจจุบันความก้าวหน้าของเทคโนโลยีปัญญาประดิษฐ์ โดยเฉพาะโมเดลภาษาขนาดใหญ่ (LLMs) ได้แสดงศักยภาพในการเข้าใจข้อกำหนดเชิงลึกและสร้างข้อความที่มีโครงสร้างซับซ้อนได้อย่างมีประสิทธิภาพ

งานวิจัยนี้จึงมีวัตถุประสงค์เพื่อพัฒนาและประยุกต์ใช้ LLMs ในการสร้างกรณีทดสอบสำหรับ Functional Testing แบบ Manual Testing เพื่อเพิ่มความครอบคลุม ความแม่นยำ และลดภาระงานของผู้ทดสอบ

วัตถุประสงค์ของการวิจัย

1. พัฒนาเครื่องมือที่ใช้ LLM สำหรับการสร้าง Test Case ในระดับ Functional Testing จาก Software Requirements

2. พัฒนาและปรับแต่ง Fine-tune โมเดล LLaMA3.2-3B, Typhoon2-8b, Gemma-3-4b และ Qwen3-8B เพื่อเพิ่มประสิทธิภาพการสร้าง Test Case

3. ออกแบบระบบ Retrieval-Augmented Generation (RAG) เชื่อมต่อ LLM กับฐานข้อมูลผ่าน Lang Chain Framework เพื่อสร้าง Test Case ที่แม่นยำและสอดคล้องกับข้อกำหนด

4. ประเมินความครอบคลุม Test Case ด้วย Test Coverage, Functional Coverage และ Requirement Coverage พร้อมทั้งประเมินความพึงพอใจของผู้เชี่ยวชาญด้าน Software Testing

ขอบเขตการวิจัย

1. สร้างกรณีทดสอบกรณีปกติ Positive Test Cases และกรณีผิดพลาด Negative Test Cases ข้อกำหนดซอฟต์แวร์

2. พัฒนาโมเดล และปรับแต่ง Fine-tuning โมเดล LLaMA3.2-3B, Typhoon2-8b, Gemma-3-4b และ Qwen 3-8B ร่วมกับ LangChain Framework เพื่อสร้างและปรับปรุง Test Case โดยใช้แนวคิด Retrieval-Augmented Generation (RAG)

3. ชุดข้อมูลสำหรับการสร้าง Test Cases ที่เกี่ยวข้องกับกระบวนการฝากถอนเงินของธนาคาร โดยเน้นธุรกรรมการเงินเป็นระบบที่ต้องการความแม่นยำและความ

ปลอดภัยสูงจึงถูกออกแบบให้ครอบคลุมทั้งกรณี Positive Cases และ Negative Cases รูปแบบ Manual Testing จำนวน 20,000 Test Case เพื่อให้สามารถทดสอบระบบได้อย่างมีประสิทธิภาพ

4. ประเมินประสิทธิภาพของโมเดล ค่า Loss จาก Vector Embedding, ความครอบคลุมของ Test Case Coverage และความพึงพอใจของผู้เชี่ยวชาญด้าน Software Testing

5. เครื่องมือสร้าง Test Case สำหรับ Functional Testing แบบ Manual Testing พร้อมการประเมินประสิทธิภาพของแต่ละโมเดล

6. ผลลัพธ์ที่ได้จากการวิจัยนี้คือเครื่องมือสำหรับสร้างกรณีทดสอบ Test Case สำหรับ Functional Testing แบบ Manual Testing พร้อมการประเมินประสิทธิภาพของแต่ละโมเดล

ประโยชน์ที่คาดว่าจะได้รับ

1. พัฒนาเพื่อช่วยลดภาระงานของนักทดสอบ Tester, QA ในการสร้าง Test Cases แบบ Manual Testing ทำให้ประหยัดเวลาในกระบวนการทดสอบซอฟต์แวร์

2. เพิ่มความครอบคลุมของการทดสอบ Test Coverage, Functional Coverage และ Requirement Coverage โมเดลที่พัฒนาสามารถสร้างกรณีทดสอบที่ครอบคลุมทั้ง Positive และ Negative Cases ช่วยลดโอกาสที่ข้อผิดพลาดจะไปถึงผู้ใช้งาน

3. ระบบรองรับการโต้ตอบแบบ ChatBot ช่วยให้ผู้ใช้สามารถป้อนข้อกำหนดความต้องการและรับผลลัพธ์ได้ทันที โดยไม่จำเป็นต้องมีความรู้ด้านการพัฒนาระบบหรือการเขียน Test Case

แนวคิดทฤษฎี และงานวิจัยที่เกี่ยวข้อง

แนวคิดและทฤษฎีที่เกี่ยวข้อง

1. Large Language Models ใช้โครงสร้างแบบ Transformer ผ่านการฝึกเบื้องต้น Pre-training โดยเรียนรู้จากบริบทของข้อความเพื่อทำนายปรับแต่ง Fine-tuning ด้วยชุดข้อมูลเฉพาะด้าน Loss Function สำหรับ LLMs อาศัย Loss Function เพื่อวัดว่าผลลัพธ์ของโมเดล สูตรคำนวณที่ใช้ใน LLM ค่า Loss Function สำหรับ LLMs เพื่อวัดว่าผลลัพธ์ของโมเดลแตกต่างจากค่าจริงมากหรือน้อย Loss Function ที่นิยมใช้ใน LLMs คือ Cross-Entropy Loss ซึ่งคำนวณ (Radford et al., 2019) ใช้สำหรับวัดความแตกต่างระหว่างผลลัพธ์ที่โมเดลคาดการณ์ และค่าความจริง โดยอาศัยฟังก์ชัน Cross-

Entropy Loss ซึ่งนิยมใช้ในงานประมวลผลภาษาธรรมชาติ โดยแสดงดังสมการ 1

$$L = - \sum_{i=1}^N y_i \log(\hat{y}_i) \quad (1)$$

2. LoRA (Low-Rank Adaptation) เทคนิคการปรับแต่งโมเดล Fine-tuning ที่เน้นลดการใช้ทรัพยากรคำนวณและหน่วยความจำ โดยทำงานผ่านการ freeze พารามิเตอร์หลักของโมเดล และ low-rank matrices เพื่อเรียนรู้การเปลี่ยนแปลงสำหรับงานช่วยให้สามารถปรับโมเดลขนาดใหญ่ได้อย่างมีประสิทธิภาพโดยไม่ต้องปรับพารามิเตอร์ทั้งหมด แนวคิดและหลักการเพื่อจัดการกับปัญหาการปรับแต่งโมเดลขนาดใหญ่ที่ต้องใช้พารามิเตอร์จำนวนมากและใช้ทรัพยากรสูงในการอัปเดตน้ำหนักของโมเดล วิธีนี้ใช้แนวคิด Low-Rank Decomposition ซึ่งช่วยลดจำนวนพารามิเตอร์ที่ต้องปรับแต่งให้น้อยลง โดยใช้เมทริกซ์ที่มีลำดับชั้นต่ำ low-rank matrix แทนการอัปเดตโดยตรงของพารามิเตอร์โมเดลเดิมโดยใช้วิธีที่เป็นรูปแบบหนึ่งของ Matrix Low-Rank Matrix (Hu et al., 2021) โดยแสดงดังสมการ 2

$$\Delta W = A \times B \quad (2)$$

โดยที่ A และ B จะต้องเป็น Low-Rank Matrix ซึ่งจะเป็น matrix ที่ขนาดเล็กกว่าเมื่อเทียบกับ weight ของโมเดล LLMs ที่จะทำการปรับแต่ง โดยจะทำให้จำนวนพารามิเตอร์ที่ใช้ในการฝึกมีขนาดลดลงและ ΔW จะเป็นค่า weight ที่เปลี่ยนแปลงไปจากโมเดลเดิม

3. Embedding เป็นเทคนิคที่ใช้แปลงข้อมูลที่อยู่ในรูปแบบของข้อความให้กลายเป็นเวกเตอร์ตัวเลขในเชิงคณิตศาสตร์ ช่วยให้โมเดลเรียนรู้ความหมายเชิงลึกของข้อมูล การใช้ Embedding ในงานประมวลผลภาษาธรรมชาติ ซึ่งมีจุดมุ่งหมายเพื่อให้เวกเตอร์ที่ได้สะท้อนความหมายของคำหรือประโยคอย่างถูกต้องในเชิงบริบทในแนวทางการประเมินความถูกต้องหรือความเหมาะสมของการจับคู่ข้อมูล เช่น การจับคู่คำถามกับคำตอบ หรือ Test Case กับคำตอบของโมเดล Predicted Answer จากโมเดล Transformer Contextual Embedding ประกอบด้วย Token Embedding ของคำ, บอกลำดับคำ และ แยกกลุ่มประโยค เมื่อนำทั้งหมดมาบวกเข้าด้วยกันแล้ว Transformer จะได้ Contextual Embedding ของโทเคน ซึ่งสะท้อนความหมายตามบริบทในประโยคนั้น ๆ ได้ดียิ่งขึ้น (Devlin et al., 2019) แสดงดังสมการ 3

$$\text{Embedding}(w_i) = \text{Transformer}(E_{token_i} + E_{position_i} + E_{segment_i}) \quad (3)$$

4. Cosine Distance ตัวชี้วัดเชิงเวกเตอร์ที่ใช้ประเมินความแตกต่างความไม่คล้ายคลึงกันระหว่างเวกเตอร์สองตัวโดยพิจารณามุมระหว่างเวกเตอร์ คำนวณจากค่า Cosine Similarity ซึ่งเป็นค่าความคล้ายคลึงเชิงมุมระหว่างเวกเตอร์โดยค่าอยู่ในช่วง 0,1 ค่าที่ใกล้ 0 หมายถึงเวกเตอร์มีทิศทางคล้ายกันมาก ในขณะที่ค่าที่ใกล้ 1 หมายถึงมีทิศทางต่างกันมาก จุดเด่นของ Cosine Distance คือไม่ไวต่อขนาดของเวกเตอร์ length-invariant ทำให้เหมาะกับข้อมูลมิติสูงและเวกเตอร์ที่มีความบาง เช่น ข้อมูลเอกสารแบบ TF-IDF หรือเวกเตอร์ฝังความหมาย embeddings ของประโยคใช้อย่างแพร่หลายในงานด้าน Information Retrieval (IR) และ Natural Language Processing (NLP) เนื่องจากให้ผลลัพธ์ที่สอดคล้องกับความคล้ายเชิงความหมายของข้อมูลในเชิงภาษาธรรมชาติ โดยเฉพาะในกรณีที่ใช้ร่วมกับการทำ L2-normalization ของเวกเตอร์ (Dhillon *et al.*, 2002) โดยคำนวณจากมุมระหว่างเวกเตอร์ และ ซึ่ง หมายถึงคอตโปรดัคต์ และ คือค่า L2 norm ของเวกเตอร์แต่ละตัว ค่าที่ได้จะอยู่ในช่วง 0 – 1 โดยค่าที่ใกล้ศูนย์หมายถึงเวกเตอร์มีทิศทางใกล้เคียงกันมาก ในขณะที่ค่าที่ใกล้หนึ่งสะท้อนถึงความแตกต่างสูง (Schütze *et al.*, 2008) แสดงดังสมการ 4

$$d_{cos}(A, B) = 1 - \frac{A \cdot B}{\|A\| \|B\|} \quad (4)$$

5. Euclidean Distance การวัดระยะทางจริงระหว่างจุดสองจุดในพื้นที่ n มิติเหมือนใช้ไม้บรรทัดวัดระยะทางระหว่างสองจุด ใช้วัดว่าเวกเตอร์ A และ B โดยพิจารณาค่าผลต่างยกกำลังสองในแต่ละมิติแล้วนำมารวมและถอดรากที่สอง ค่า และ แทนค่าของเวกเตอร์ในแต่ละมิติ ผลลัพธ์จึงสะท้อนระยะทางตรง straight-line distance ระหว่างสองจุดอยู่ห่างกันแค่ไหนในเชิงพิกัดนำค่าของแต่ละมิติมาหา “ผลต่างยกกำลังสอง” แล้วถอดราก เหมาะกับกรณีที่ต้องการวัดระยะทางเชิงปริมาณอย่างแท้จริง เช่น การวัดความแตกต่างของคำตอบที่ได้จากโมเดลกับคำตอบจริง การจัดกลุ่มข้อมูล Clustering และงานด้าน Computer Vision เป็นเครื่องมือที่ได้รับความนิยมในหลายสาขาวิชา เช่น การจัดกลุ่มข้อมูล Clustering การวิเคราะห์รูปแบบ Pattern Recognition และการวัดความคล้ายคลึงของข้อมูล Similarity Measurement โดยเฉพาะในอัลกอริทึม K-means ซึ่งใช้ระยะทางนี้ในการวัด

ระยะห่างระหว่างจุดข้อมูลและศูนย์กลางของกลุ่ม Centroid F (Cheng, 2024) โดยแสดงดังสมการ 5

$$\text{Euclidean}(A, B) = \sqrt{\sum_{i=1}^n (A_i - B_i)^2} \quad (5)$$

6. Manhattan Distance การวัดระยะทางระหว่างเวกเตอร์สองตัวโดยรวมระยะทางของแต่ละแกนโดยไม่ยกกำลังหรือถอดราก วัดตามแนวแกน เหมือนการเดินทางตามถนนในลักษณะ Grid เหมาะสำหรับข้อมูลที่อยู่ในตาราง grid-based data ข้อมูลที่มีลักษณะไม่ต่อเนื่อง หรือกรณีที่ต้องการลดผลกระทบจากค่าที่สูงเกินไปในบางมิติ outlier-sensitive (Gao & Li, 2016) โดยการรวมผลของแต่ละมิติ โดยไม่ถอดรากหรือยกกำลัง จึงคล้ายการคำนวณระยะทางที่เดินไปตามเส้นทางกริด (grid-based path) มากกว่าการวัดระยะตรงแสดงดังสมการ 6

$$\text{Manhattan}(A, B) = \sum_{i=1}^n |A_i - B_i| \quad (6)$$

7. LangChain เฟรมเวิร์กโอเพนซอร์สที่ได้รับการออกแบบมาเพื่อรองรับการพัฒนาแอปพลิเคชันที่ใช้โมเดลภาษาขนาดใหญ่ LLMs โดยเฉพาะโดยมีวัตถุประสงค์หลักในการเพิ่มขีดความสามารถของ LLM ให้สามารถโต้ตอบกับแหล่งข้อมูลภายนอกเครื่องมือภายนอกหน่วยความจำ และระบบเวิร์กโฟลว์ได้อย่างมีประสิทธิภาพ LangChain มีโครงสร้างการทำงานแบบองค์รวมที่เชื่อมโยงองค์ประกอบต่าง ๆ เข้าไว้ด้วยกันได้แก่ Prompts, LLM, Chains, Agents, Tools, Memory และ Retrieval System การประมวลผลเริ่มต้นจากการป้อน Requirement เข้าไปในระบบ จากนั้น LangChain จะใช้ PromptTemplate สร้างคำสั่งที่เหมาะสมเพื่อส่งให้ LLM ทำการสรุปข้อมูลเชิงความเข้าใจระบบจะใช้ Retrieval Chain ดึงข้อมูลที่เกี่ยวข้องจากคลังข้อมูลภายนอกผ่านเทคนิคการฝังข้อมูล Embedding ลงใน Vector Store เพื่อให้ได้บริบทที่เกี่ยวข้องมากที่สุดหลังจากนั้น Agent จะวิเคราะห์สถานการณ์และเลือกใช้เครื่องมือ Tools ที่เหมาะสม เช่น การสร้าง Test Case การวิเคราะห์สาเหตุ หรือการถามกลับเพื่อขอข้อมูลเพิ่มเติม Agent สามารถทำงานแบบ Reason Act ได้โดยใช้แนวคิด ReAct Prompting ซึ่งช่วยให้ LLM สามารถตัดสินใจเลือกเครื่องมือได้อย่างอัตโนมัติ (Guan & Fang, 2025)

8. Software Testing Life Cycl กระบวนการกำหนดขั้นตอนสำหรับการทดสอบซอฟต์แวร์ โดยมีวัตถุประสงค์เพื่อให้การทดสอบเป็นไปอย่างมีประสิทธิภาพและช่วยเพิ่มความครอบคลุมของการทดสอบ test coverage ลดความเสี่ยงในการส่งมอบซอฟต์แวร์ที่มีข้อบกพร่อง และเพิ่มความเชื่อมั่นต่อคุณภาพของระบบงานที่พัฒนา STLC ยังการประเมินคุณภาพซอฟต์แวร์โดยสามารถเชื่อมโยงกรณีทดสอบกับข้อกำหนดและฟังก์ชันของระบบเพื่อประเมิน Requirement Coverage ได้อย่างมีประสิทธิภาพ (Masuda *et al.*, 2020)

9. Test Coverage การวัดสัดส่วนของระบบระหว่างการทดสอบโดยมีจุดประสงค์เพื่อประเมินว่ากรณีทดสอบที่สร้างขึ้นสามารถตรวจสอบโค้ดได้ครอบคลุมมากเพียงใด ตัวอย่างเช่น การวัดจำนวนที่ทดสอบแล้วเมื่อเทียบกับทั้งหมดในระบบช่วยยืนยันว่าการทดสอบสามารถครอบคลุมส่วนประกอบของโปรแกรมได้มากที่สุด ลดโอกาสการเกิดบั๊กที่ไม่ได้ตรวจพบ (Cheng *et al.*, 2016) โดยแสดงดังสมการ 7

$$Test\ Coverage(\%) = \frac{TC_{exe}}{TC_{des}} \times 100 \quad (7)$$

10. Functional Coverage การวัดสัดส่วนของฟังก์ชันการทำงานของระบบที่ได้รับการทดสอบเมื่อเทียบกับจำนวนฟังก์ชันทั้งหมดที่ระบุไว้ในเอกสารข้อกำหนดซอฟต์แวร์โดยมุ่งเน้นประเมินว่ากรณีทดสอบครอบคลุมฟังก์ชันทั้งหมดของระบบช่วยตรวจสอบว่าการทดสอบครอบคลุมทุกฟีเจอร์หลักของระบบตามที่ออกแบบไว้ ลดความเสี่ยงในการขาดการทดสอบฟังก์ชันสำคัญ (Nazi *et al.*, 2022) โดยแสดงดังสมการ 8

$$Fuctional\ Coverage(\%) = \frac{\sum_{j=1}^m w_j 1[f_j covered]}{\sum_{j=1}^m w_j} \times 100 \quad (8)$$

11. Requirement Coverage เป็นการวัดสัดส่วนของข้อกำหนดที่มีกรณีทดสอบครอบคลุมเมื่อเทียบกับข้อกำหนดทั้งหมดในระบบ เป็นตัวชี้วัดสำคัญว่ากรณีทดสอบตอบสนองต่อความต้องการของผู้ใช้ครบถ้วนหรือไม่ (Wang *et al.*, 2025; Omar, 2013) แสดงดังสมการ 9

$$Requirement\ Coverage(\%) = \frac{\sum_{i=1}^n w_i 1[r_i covered]}{\sum_{i=1}^n w_i} \times 100 \quad (9)$$

งานวิจัยที่เกี่ยวข้อง

1. การสร้างชุดทดสอบอัตโนมัติสำหรับแอปพลิเคชันมือถือด้วย LLM

(Hoffmann & Frister, 2024) นำเสนอการประยุกต์ใช้โมเดลภาษาขนาดใหญ่ที่ผ่านการปรับแต่งแบบ Supervised Fine-Tuning (SFT) เพื่อสร้างกรณีทดสอบอัตโนมัติสำหรับแอปพลิเคชันมือถือที่พัฒนาด้วยภาษา Dart โดยใช้ชุดข้อมูลจาก GitHub ซึ่งประกอบด้วยคู่มือโค้ดและไฟล์ทดสอบจำนวน 16,252 คู่ แบ่งเป็นข้อมูลฝึก 90% และข้อมูลทดสอบ 10% ผลการทดลองแสดงให้เห็นว่าโมเดล Test Gen – Dart v0.2 ที่ผ่านการปรับแต่งมีประสิทธิภาพเหนือกว่าโมเดลพื้นฐาน Code Llama 13B 4-bit โดยเพิ่ม Syntactic Correctness ได้ 15.38% และ Functional Correctness ได้ 16.67% เมื่อเทียบกับโมเดลดั้งเดิม

2. การสร้างชุดทดสอบสำหรับโปรเซสเซอร์ด้วย LLM Deng

(Wang *et al.*, 2024) เสนอแนวทาง LLM-TG สำหรับสร้างชุดทดสอบอัตโนมัติในกระบวนการตรวจสอบความถูกต้องของการออกแบบโปรเซสเซอร์ (Design Verification) โดยใช้โมเดล GPT-4 ผ่าน 4 ขั้นตอนหลัก ได้แก่ 1) การสร้างชุดทดสอบด้วย Prompt 2) การปรับแต่งชุดทดสอบ 3) การใช้ผลลัพธ์การทดสอบเพื่อสร้างชุดทดสอบเพิ่มเติม และ 4) การสร้าง Prompt Library ผลการทดลองกับโปรเซสเซอร์ RV64GC (Xuantie-C910) พบว่า LLM-TG สามารถเพิ่ม Block Coverage ได้ 8.34% และ Expression Coverage ได้ 5.8% เมื่อเทียบกับวิธีดั้งเดิม เช่น RISC-V-DV และ LLM4DV นอกจากนี้ยังแสดงให้เห็นว่า LLM-TG ช่วยลดภาระงานและเพิ่มประสิทธิภาพในการสร้างชุดทดสอบสำหรับโปรเซสเซอร์

3. การสร้าง Unit Test อัตโนมัติด้วย Generative AI (Bhatia *et al.*, 2024) นำเสนอการเปรียบเทียบประสิทธิภาพของเครื่องมือสร้าง Unit Test อัตโนมัติที่ใช้ ChatGPT กับเครื่องมือ Search-Based Software Testing (SBST) ที่ได้รับความนิยมอย่าง Pynguin โดยมุ่งเน้นการสร้างชุดทดสอบสำหรับโค้ดภาษา Python ทั้งในรูปแบบ Procedural, Function-based และ Class-based ผลการทดลองพบว่า ChatGPT สามารถสร้างชุด Unit Test ที่มีค่า coverage ใกล้เคียงกับเครื่องมือ Pynguin และในบางกรณียังสามารถให้ค่า coverage ที่สูงกว่า โดยเฉพาะอย่างยิ่งเมื่อมีการประยุกต์ใช้เทคนิค prompt engineering แบบ iterative โดยการป้อนข้อมูลของ statement ที่ยังไม่ถูกทดสอบ (missed statements) กลับไปยังโมเดล เพื่อกระตุ้นให้สร้าง Test Case เพิ่มเติม

สำหรับส่วนที่ขาดหาย ส่งผลให้ coverage โดยรวมของโค้ดเพิ่มสูงขึ้น สะท้อนให้เห็นถึง ศักยภาพของการผสาน LLM และ SBST ในการสร้าง Unit Test ที่มีคุณภาพสูง ครอบคลุมมากขึ้น และช่วยลดภาระของผู้พัฒนาในกระบวนการสร้างชุดทดสอบด้วยตนเอง

จากงานวิจัยข้างต้นแสดงให้เห็นถึงแนวโน้มการนำ LLM มาประยุกต์ใช้ในการสร้างชุดทดสอบอัตโนมัติในหลากหลายบริบท ทั้งในระดับ Mobile Application, Processor Verification และ Unit Testing โดยแต่ละงานเน้นเทคนิคการปรับแต่งโมเดลและการออกแบบกระบวนการสร้างชุดทดสอบที่เหมาะสมกับลักษณะของระบบเป้าหมาย โดยงานวิจัยนี้เป็นการทำงานต่อเนื่องต่อดังกล่าว โดยเน้นการพัฒนาและเปรียบเทียบประสิทธิภาพของ LLM สำหรับการสร้างกรณีทดสอบ Functional Testing แบบ Manual Testing ด้วยเทคนิค Retrieval-Augmented Generation เพิ่มความครอบคลุมและความแม่นยำในการทดสอบซอฟต์แวร์ในระบบธนาคารที่มีความซับซ้อนสูง

การทดลอง

ภาพรวมของระบบ

สถาปัตยกรรมของระบบที่นำเสนอเน้นการผสมผสานความสามารถของ LLM กับ RAG framework ผ่านโครงสร้างการสืบค้นข้อมูลแบบ similarity search บนฐานข้อมูลเวกเตอร์ รวมทั้งการจัดการ prompt สำหรับ LLM ด้วย LangChain เพื่อให้การสร้าง Test Case มีบริบทที่ถูกต้อง แม่นยำ และตรงตามความต้องการของซอฟต์แวร์ในหลากหลายมิติ การสร้าง Test Case อัตโนมัติ เริ่มจากผู้ใช้ป้อนข้อกำหนดซอฟต์แวร์ จากนั้น AI Agent (LLMChain GPT-4.1 mini) จะทำการวิเคราะห์และประสานงานกับเครื่องมือค้นหาข้อมูลเชิงบริบทในฐานข้อมูล ChromaDB (ฐานข้อมูลเวกเตอร์) ด้วย similarity search ผ่าน LangChain จากนั้นนำข้อมูลที่ได้มาสร้าง Prompt ส่งให้โมเดล LLM ที่ผ่าน Fine-Tuning ทำการสร้าง Test Case ทั้งแบบ Positive และ Negative จากข้อกำหนดซอฟต์แวร์ที่ให้ และส่งผลลัพธ์กลับให้ผู้ใช้ สำหรับสถาปัตยกรรมการทำงาน แสดงใน Figure 1

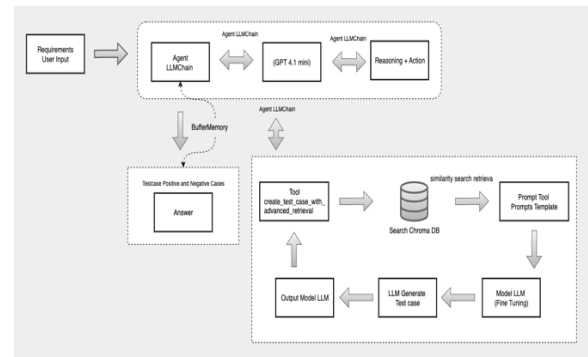


Figure 1 System Architecture

การออกแบบการวิจัย

1. ส่วนคำถาม (Input Stage) ระบบจะรับข้อกำหนดความต้องการซอฟต์แวร์ จากผู้ใช้งานทาง Web Interface ที่พัฒนาด้วย Streamlit
2. ส่วนการวิเคราะห์ถึงบริบท (Context Analysis with RAG) ระบบนำข้อกำหนดความต้องการที่ได้รับมาเข้าสู่กระบวนการวิเคราะห์เชิงบริบท โดยใช้โมเดล LLM และเชื่อมต่อกับระบบ RAG และการดึงบริบทใช้ Vector Search ผ่านฐานข้อมูลเวกเตอร์ ChromaDB ที่ใช้การแปลงข้อความเป็น Vector embedding แล้วค้นหาจากฐานข้อมูลด้วย semantic similarity
3. ส่วนการประมวลผล (Processing & Test Case Generation) จะนำบริบทและข้อกำหนดความต้องการที่สรุปได้เข้าสู่โมเดล GPT-4.1 mini และโมเดล LLM ที่ทำ Fine-Tuned แล้ว เพื่อให้ระบบทำการวิเคราะห์สร้าง Test Case
4. ส่วนการตอบคำถาม (Answer Generation & Memory Storage) การประมวลผลจะถูกส่งกลับให้ผู้ใช้ในรูปแบบคำตอบครอบคลุมทั้ง Positive และ Negative Test Cases และเก็บประวัติการใช้งานเพื่อใช้ต่อยอดการสนทนา (Tiwari et al., 2022)

การเก็บรวบรวมข้อมูล

การดำเนินการวิจัยครั้งนี้ได้จัดเตรียมชุดข้อมูลที่ประกอบด้วย ข้อกำหนดซอฟต์แวร์ และ กรณีทดสอบ (Test Cases) โดยครอบคลุมการทำงานหลักของระบบธนาคารได้แก่ การฝากเงิน, การถอนเงิน, การแสดงรายการเดินบัญชี e-Statement, การปรับวงเงิน การลงทะเบียนด้วยชื่อผู้ใช้และรหัสผ่าน และการยืนยันตัวตนด้วยการสแกนใบหน้า ประกอบด้วยข้อมูลทั้งหมดจำนวน 20,000 รายการ โดยแบ่งเป็น Positive Cases จำนวน 15,763 รายการ (78.82%) และ Negative Cases จำนวน 4,237 รายการ (21.18%) โดยชุดข้อมูลทั้งหมดจัดเก็บในรูปแบบไฟล์ CSV โครงสร้างข้อมูลแสดงใน Figure 2

Test Case ID	Requirement Text	Test Case Description	Test Steps	Expected Result	Test Case Type	Label	Automation Status	Priority
TC-0001	System shall allow user to login with valid email and password.	User can login with valid email and password.	1. Enter valid email in the email field. 2. Enter valid password in the password field. 3. Click the login button.	User is redirected to the dashboard page.	Positive	1	Manual	Medium
TC-0002	System shall allow user to login with valid email and password.	User can login with valid email and password.	1. Enter valid email in the email field. 2. Enter valid password in the password field. 3. Click the login button.	User is redirected to the dashboard page.	Positive	1	Manual	Medium
TC-0003	System shall allow user to login with valid email and password.	User can login with valid email and password.	1. Enter valid email in the email field. 2. Enter valid password in the password field. 3. Click the login button.	User is redirected to the dashboard page.	Positive	1	Manual	Medium
TC-0004	System shall allow user to login with valid email and password.	User can login with valid email and password.	1. Enter valid email in the email field. 2. Enter valid password in the password field. 3. Click the login button.	User is redirected to the dashboard page.	Positive	1	Manual	Medium
TC-0005	System shall allow user to login with valid email and password.	User can login with valid email and password.	1. Enter valid email in the email field. 2. Enter valid password in the password field. 3. Click the login button.	User is redirected to the dashboard page.	Positive	1	Manual	Medium
TC-0006	System shall allow user to login with valid email and password.	User can login with valid email and password.	1. Enter valid email in the email field. 2. Enter valid password in the password field. 3. Click the login button.	User is redirected to the dashboard page.	Positive	1	Manual	Medium
TC-0007	System shall allow user to login with valid email and password.	User can login with valid email and password.	1. Enter valid email in the email field. 2. Enter valid password in the password field. 3. Click the login button.	User is redirected to the dashboard page.	Positive	1	Manual	Medium
TC-0008	System shall allow user to login with valid email and password.	User can login with valid email and password.	1. Enter valid email in the email field. 2. Enter valid password in the password field. 3. Click the login button.	User is redirected to the dashboard page.	Positive	1	Manual	Medium
TC-0009	System shall allow user to login with valid email and password.	User can login with valid email and password.	1. Enter valid email in the email field. 2. Enter valid password in the password field. 3. Click the login button.	User is redirected to the dashboard page.	Positive	1	Manual	Medium
TC-0010	System shall allow user to login with valid email and password.	User can login with valid email and password.	1. Enter valid email in the email field. 2. Enter valid password in the password field. 3. Click the login button.	User is redirected to the dashboard page.	Positive	1	Manual	Medium
TC-0011	System shall allow user to login with valid email and password.	User can login with valid email and password.	1. Enter valid email in the email field. 2. Enter valid password in the password field. 3. Click the login button.	User is redirected to the dashboard page.	Positive	1	Manual	Medium
TC-0012	System shall allow user to login with valid email and password.	User can login with valid email and password.	1. Enter valid email in the email field. 2. Enter valid password in the password field. 3. Click the login button.	User is redirected to the dashboard page.	Positive	1	Manual	Medium
TC-0013	System shall allow user to login with valid email and password.	User can login with valid email and password.	1. Enter valid email in the email field. 2. Enter valid password in the password field. 3. Click the login button.	User is redirected to the dashboard page.	Positive	1	Manual	Medium
TC-0014	System shall allow user to login with valid email and password.	User can login with valid email and password.	1. Enter valid email in the email field. 2. Enter valid password in the password field. 3. Click the login button.	User is redirected to the dashboard page.	Positive	1	Manual	Medium
TC-0015	System shall allow user to login with valid email and password.	User can login with valid email and password.	1. Enter valid email in the email field. 2. Enter valid password in the password field. 3. Click the login button.	User is redirected to the dashboard page.	Positive	1	Manual	Medium
TC-0016	System shall allow user to login with valid email and password.	User can login with valid email and password.	1. Enter valid email in the email field. 2. Enter valid password in the password field. 3. Click the login button.	User is redirected to the dashboard page.	Positive	1	Manual	Medium
TC-0017	System shall allow user to login with valid email and password.	User can login with valid email and password.	1. Enter valid email in the email field. 2. Enter valid password in the password field. 3. Click the login button.	User is redirected to the dashboard page.	Positive	1	Manual	Medium
TC-0018	System shall allow user to login with valid email and password.	User can login with valid email and password.	1. Enter valid email in the email field. 2. Enter valid password in the password field. 3. Click the login button.	User is redirected to the dashboard page.	Positive	1	Manual	Medium
TC-0019	System shall allow user to login with valid email and password.	User can login with valid email and password.	1. Enter valid email in the email field. 2. Enter valid password in the password field. 3. Click the login button.	User is redirected to the dashboard page.	Positive	1	Manual	Medium
TC-0020	System shall allow user to login with valid email and password.	User can login with valid email and password.	1. Enter valid email in the email field. 2. Enter valid password in the password field. 3. Click the login button.	User is redirected to the dashboard page.	Positive	1	Manual	Medium

Figure 2 Example of Test Case Data

จาก Figure 2 โครงสร้างข้อมูล ประกอบด้วย 9 คอลัมน์ ได้แก่

- Test Case ID: รหัสเฉพาะของแต่ละกรณีทดสอบ
- Requirement Text: ข้อความข้อกำหนดซอฟต์แวร์ที่เกี่ยวข้อง
- Test Case Description: คำอธิบายภาพรวมของกรณีทดสอบ
- Test Steps: ขั้นตอนการทดสอบที่ต้องดำเนินการ
- Expected Result: ผลลัพธ์ที่คาดหวังจากการทดสอบ
- Test Case Type: ประเภทของกรณีทดสอบ (Positive หรือ Negative)
- Label: หมวดหมู่ที่เกี่ยวข้อง
- Automation Status: สถานะการทำงาน
- Priority: ระดับความสำคัญของกรณีทดสอบ

ในการจัดลำดับความสำคัญของ Test Cases ประกอบด้วย ระดับ Medium จำนวน 18,432 รายการ (92.16%) ระดับ High จำนวน 975 รายการ (4.88%) ระดับ Critical จำนวน 274 รายการ (1.37%) และระดับ Low จำนวน 319 รายการ (1.60%) เพื่อการพัฒนาและประเมินโมเดล ชุดข้อมูลถูกแบ่งเป็น 2 ส่วน ได้แก่ Training set และ Validation set ซึ่งใช้วิธีการสุ่มแบบ Stratified ตามประเภท Test Case แสดงดัง Table 1

Table 1 Test Case Data Sets

Dataset	Percentage	Test Case
Training Set	80%	16,000
Validation Set	20%	4,000

เครื่องมือและเทคโนโลยีที่ใช้

1. โมเดลภาษาขนาดใหญ่ (LLMs) ใช้ LLaMA 3.2-3B, Typhoon2-8b, Gemma-3-4b และ Qwen3-8B ในกระบวนการปรับปรุงโมเดล

2. การพัฒนา Backend ใช้ภาษา Python และเฟรมเวิร์ก LangChain สำหรับบริหารจัดการการเชื่อมต่อระหว่างโมเดล LLMs กับข้อมูลจากระบบ RAG และฐานข้อมูลเวกเตอร์

3. Agent ใช้แบบ ReAct Agent ที่สามารถตัดสินใจเลือกเครื่องมือให้เหมาะสมกับ Intent ของผู้ใช้ ส่งผลให้ระบบมีความยืดหยุ่นในการตอบสนองและเลือก pipeline การประมวลผลอย่างเหมาะสม

4. Memory Conversation ระบบจัดการเก็บบริบทการสนทนา (BufferMemory) เพื่อเก็บบริบทการสนทนาในแต่ละ session ซึ่งช่วยให้ LLMs สามารถอ้างอิงถึงบทสนทาก่อนหน้า ทำให้การตอบคำถามมีความต่อเนื่องและมีความเข้าใจบริบท

5. ฐานข้อมูล ChromaDB ระบบใช้สำหรับเก็บ embedding vectors ของเอกสารหรือบริบทที่เกี่ยวข้องกับงานทดสอบซอฟต์แวร์เพื่อใช้ในการสร้างคำตอบโดย LLMs

6. LangChain เป็นไลบรารีหลักที่ใช้ในการสร้าง Agent, Chain, Tool, PromptTemplate และ Memory โดยเฉพาะอย่างยิ่งเหมาะกับงานที่ต้องเชื่อมต่อ LLM กับข้อมูลภายนอกเข้าด้วยกันใน Chain เดียว

7. OpenAI API ใช้สำหรับเรียกใช้โมเดล GPT-4.1-mini ซึ่งมีประสิทธิภาพสูงในการสรุป วิเคราะห์ และประเมินผล

8. การพัฒนา Frontend โดยใช้ Streamlit เป็นเครื่องมือสำหรับสร้าง Chatbot ซึ่งเป็นโอเพนซอร์สสร้างแอปพลิเคชันเว็บเชิงโต้ตอบ

การพัฒนาโมเดล

การออกแบบ Prompt เพื่อสร้าง Test Case ของโมเดล LLMs โดยโครงสร้างของ Prompt แบ่งออกเป็น 3 ส่วน ประกอบด้วย

System Prompt กำหนดให้โมเดลมีบทบาทเป็น Software Tester มืออาชีพ ที่มีหน้าที่สร้าง Test Case โดยพิจารณาทั้งด้าน Functional, Positive / Negative, Security และ Performance Testing

User Prompt Requirement ข้อความที่ผู้ใช้ระบุ “ความต้องการของระบบ” เพื่อให้โมเดลสร้าง Test Case ตามบริบท

Expected Output Format เป็นส่วนคำตอบจากโมเดล เพื่อให้การสร้าง Test Case โดยอัตโนมัติ

การปรับแต่งโมเดลเทคนิค LoRA

การปรับแต่งโมเดล เน้นการลดภาระด้านการประมวลผล และหน่วยความจำ ในขณะที่ยังสามารถปรับแต่งโมเดลให้เหมาะสมกับงานเฉพาะด้านได้อย่างมีประสิทธิภาพ โดยการฝึกโมเดลจะผ่านฟังก์ชัน FastLanguageModel.get ซึ่งเป็นการเรียกใช้งานจากไลบรารี Parameter - Efficient Fine-Tuning ที่รองรับ LoRA layer ตามที่กำหนด และการตั้งค่าพารามิเตอร์เทคนิค LoRA ที่ใช้ในงานวิจัยนี้ แสดงดัง Table 2

Table 2 LoRA Technique Parameter Settings

Parameter	Value	Description
R	32	LoRA rank higher=more capacity/better fit, but more VRAM/time
lora_alpha	64	Scaling factor adjusts the updates to match the value of r
target_module	"q_proj", "k_proj", "v_proj", "o_proj", "gate_proj", "up_proj", "down_proj"	Layers where LoRA adapters are applied (Attention + MLP) for broad adaptation. Layers where LoRA adapters are applied Attention + MLP for broad adaptation
lora_dropout	0.05	Dropout on LoRA paths to reduce overfitting disabled at inference
Bias	none	Reduce the number of parameters and model complexity

Parameter	Value	Description
gradient_checkpointing	unsloth	Recomputes activations to save VRAM slight training slowdown
random_state	42	Constant value for reproducible randomization

เทคนิคนี้ช่วยให้สามารถปรับแต่งโมเดลขนาดใหญ่ได้ โดยไม่จำเป็นต้องอัปเดตพารามิเตอร์ทั้งหมด ส่งผลให้โมเดลสามารถเรียนรู้ลักษณะเฉพาะของงานสร้าง Test Case ได้อย่างแม่นยำ สามารถประหยัดเวลาและทรัพยากรในกระบวนการปรับแต่งได้อย่างมีประสิทธิภาพ

การปรับแต่งโมเดล (Fine-tuning)

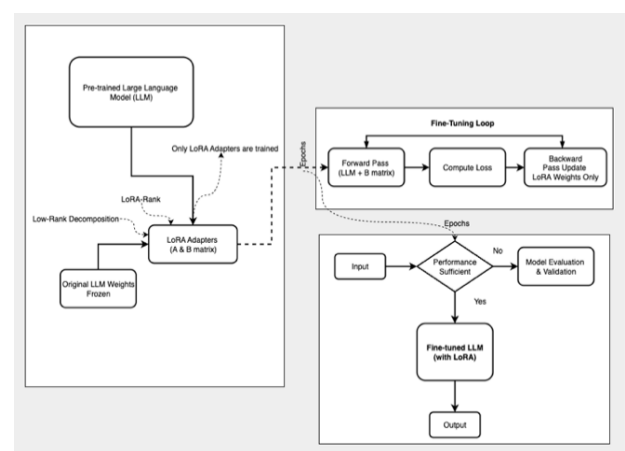


Figure 3 Fine-tuning Data Data Process

กระบวนการของเทคนิค Low-Rank Adaptation (LoRA) แสดงดัง Figure 3 ซึ่งเป็นกระบวนการ Fine-tuning Data สำหรับการปรับแต่งโมเดล LLMs อย่างมีประสิทธิภาพ โดยเริ่มจากโมเดล Pre-trained LLM ซึ่งมี Original LLM Weights ที่ถูก Frozen ระหว่างกระบวนการปรับแต่ง (Low-Rank Decomposition) เพื่อสร้าง LoRA Adapters ที่ประกอบด้วยเมทริกซ์ A และ B โดยกระบวนการปรับแต่งเกิดขึ้นใน Fine-Tuning Loop ที่ประกอบด้วย ขั้นตอน Forward Pass โดยใช้ LLM บวกกับเมทริกซ์ B การคำนวณ Loss และขั้นตอนการส่งผ่านกลับ Backward Pass ระบบจะทำการประเมินประสิทธิภาพ เป็นระยะ ๆ และดำเนินการฝึกจนกว่าจะได้โมเดลที่ปรับแต่งที่สามารถให้ผลลัพธ์ที่ต้องการ ซึ่งการปรับแต่งโมเดลด้วยเทคนิคนี้ สามารถลดจำนวนพารามิเตอร์

ที่ต้องฝึกอย่างมาก ช่วยประหยัดหน่วยความจำและเวลาในการฝึก ในขณะที่ยังคงรักษาประสิทธิภาพของโมเดลไว้ได้ใกล้เคียงกับการปรับแต่งแบบเต็มรูปแบบ

การตั้งค่าพารามิเตอร์ สำหรับการปรับแต่งโมเดล LLaMA3.2-3B, Typhoon2-8b, Gemma-3-4b และ Qwen3-8B ระหว่างการฝึกโมเดลที่เหมาะสม (Hu *et al.*, 2024) แสดงดัง Table 3

Table 3 Model Training Parameter Settings

Parameter	Value	Description
per_device_train_batch_size	1	Number of training samples per GPU
gradient_accumulation_steps	2	Number of gradient accumulation steps suitable for the value of r
warmup_steps	5	Number of steps to increase the learning rate
num_train_epochs	2	Number of epochs the model will train on the data
learning_rate	2e-4	Learning rate used during training
logging_steps	1	Logging frequency
optim	adamw_8bit	High performance with lower memory usage
weight_decay	0.01	Weight decay rate to prevent overfitting
lr_scheduler_type	linear	Linear decay after warmup

กระบวนการวัดความคล้ายคลึงระหว่างเวกเตอร์

การวัดความคล้ายคลึงระหว่างเวกเตอร์ เป็นขั้นตอนสำคัญในการตรวจสอบคุณภาพของคำตอบที่ได้จากโมเดล โดยมีวัตถุประสงค์เพื่อวัดระดับความสอดคล้องหรือความใกล้เคียงระหว่างคำตอบที่โมเดลสร้างขึ้นกับคำตอบจริง โดยงานวิจัยนี้ได้เลือกใช้เมตริกเชิงเวกเตอร์ 3 ประเภทได้แก่

- 1) **Cosine Distance** ใช้วัดความคล้ายคลึงเชิงมุมระหว่างเวกเตอร์ของสองข้อความโดยมีค่าระหว่าง 0 - 1 เมื่อแปลงเป็น Cosine Distance ค่ายิ่งน้อยแสดงถึงความใกล้เคียงสูง
- 2) **Euclidean Distance** ใช้วัดระยะห่างเชิงเรขาคณิตระหว่างเวกเตอร์สองจุดในพื้นที่หลายมิติ ค่ายิ่งน้อยแสดงว่าคำตอบอยู่ใกล้กับคำตอบจริง
- 3) **Manhattan Distance** เมตริกที่ใช้วัดระยะห่างระหว่างเวกเตอร์สองจุดในพื้นที่หลายมิติค่ายิ่งใกล้ 0 เวกเตอร์ทั้งสองใกล้เคียงหรือเหมือนกันมาก ยิ่งค่าสูงมากเวกเตอร์ทั้งสองแตกต่างกันมาก เพื่อประเมินความคล้ายคลึงระหว่างคำตอบที่โมเดลสร้างขึ้นกับคำตอบที่ถูกต้องโดยใช้โมเดล Alibaba-NLP/gte-multilingual-base (Abbad *et al.*, 2016; Srikaewsiew *et al.*, 2022; Zhang *et al.*, 2022)

กระบวนการทำงานของ Lang Chain

- 1. การออกแบบ Agent ด้วยแนวคิด ReAct (Reasoning + Acting) ใน LangChain ช่วยให้ Agent ที่ขับเคลื่อนด้วย LLMs เช่น GPT 4.1mini สามารถวิเคราะห์ปัญหาและเลือกใช้เครื่องมือ ทำให้ระบบมีความยืดหยุ่นและตอบสนองได้อย่างชาญฉลาด (Jacob *et al.*, 2024)
- 2. การสร้าง Chains เพื่อควบคุมลำดับขั้นตอนอัตโนมัติ โดย LangChain จะใช้โครงสร้าง Chain เพื่อเชื่อมโยงการทำงาน
- 3. Memory เก็บบริบทการสนทนา (Conversation Buffer Memory) ซึ่งเป็นรูปแบบ Short Term Memory เป็นหน่วยความจำระยะสั้นของแชทบอทโดยจะบันทึกคำถามและคำตอบก่อนหน้า
- 4. การเชื่อมต่อระบบ RAG โดยกำหนดเป็น Retrieval QChain หรือกำหนด Custom Chain ที่เชื่อมกับฐานข้อมูลเวกเตอร์ Chroma DB ในงานวิจัยนี้ใช้การแปลงข้อกำหนดความต้องการของผู้ใช้งานให้เป็นเวกเตอร์ฝังตัว (Embedding) และนำไปใช้ค้นหาริบทหรือข้อมูลที่เกี่ยวข้องจากฐานข้อมูลเวกเตอร์ดังกล่าว ก่อนจะประมวลผลเพื่อนำเสนอคำตอบที่เหมาะสมต่อไป (Bag *et al.*, 2024)
- 5. การนำ Prompt Template มาใช้ในการสร้าง Prompt ที่สามารถกำหนดตัวแปรสำคัญ เช่น requirement และ context ได้อย่างยืดหยุ่น ช่วยให้โมเดลสามารถรับข้อมูลข้อกำหนดความต้องการและบริบทที่เกี่ยวข้องในแต่ละกรณีได้อย่างถูกต้องแม่นยำยิ่งขึ้น
- 6. การจัดการเครื่องมือ Agent Tool ในระบบ จะดำเนินการโดยการเลือกเครื่องมือ และชุดกระบวนการประมวลผลหรือ Chain ที่เหมาะสมสำหรับแต่ละภารกิจ

ผลการเปรียบเทียบประสิทธิภาพ

การทดสอบจำนวน 4 โมเดล ประกอบด้วย LLaMA 3.2-3B, Typhoon2-8b, Gemma-3-4b และ Qwen3-8B บน Testing Set เพื่อประเมินประสิทธิภาพของเครื่องมือที่พัฒนาขึ้น โดยใช้ค่า Loss เป็นเกณฑ์ในการประเมิน แสดงดัง Table 4

Table 4 Performance Evaluation Using Loss

Model	Average Training Loss
LLaMA3.2-3B	0.1946
Typhoon2-8b	0.1703
Gemma-3-4b	0.2529
Qwen3-8B	0.1558

จาก Table 4 แสดงผลการประเมินประสิทธิภาพของโมเดลภาษาขนาดใหญ่ 4 โมเดล โดยใช้เกณฑ์ Average Training Loss เป็นตัวชี้วัดสำคัญในการประเมินความสามารถในการเรียนรู้ของโมเดล จากผลการทดลองพบว่า Qwen3-8B แสดงประสิทธิภาพสูงสุดด้วยค่า Average Training Loss ที่ต่ำที่สุด (0.1558) แสดงความสามารถในการเรียนรู้จากข้อมูลฝึกอบรมได้อย่างมีประสิทธิภาพ ตามด้วย Typhoon2-8b (0.1703) และ LLaMA3.2-3B (0.1946) ตามลำดับ ในขณะที่ Gemma-3-4b แสดงค่า Average Training Loss สูงสุด (0.2529) ซึ่งอาจสะท้อนให้เห็นถึงกระบวนการเรียนรู้ของโมเดล การที่ Qwen3-8B และ Typhoon2-8b ซึ่งเป็นโมเดลขนาด 8 พันล้านพารามิเตอร์แสดงประสิทธิภาพที่ดีกว่าโมเดลขนาดเล็กกว่าเป็นไปตามคาดการณ์ แต่การที่ Gemma-3-4b แสดง Loss สูงกว่าแม้จะมีขนาดกลางระหว่าง LLaMA3.2-3B และโมเดล 8B อาจเป็นผลมาจากปัจจัยด้านสถาปัตยกรรม โมเดลการฝึกโมเดล หรือลักษณะของข้อมูลที่ใช้ในการฝึก

ผลการประเมินโดยใช้ชุดข้อมูลทดสอบ

การประเมินผลจะใช้เมตริกหลัก 3 รายการ ได้แก่ ค่าเฉลี่ย Cosine Distance, Euclidean Distance และ Manhattan Distance เปรียบเทียบเวกเตอร์ Embedding ระหว่างคำตอบจากโมเดลกับคำตอบจริง โดยใช้โมเดล Alibaba-NLP/gte-multilingual-base แสดงดัง Table 5

Table 5 Evaluation Using Test Dataset

Model	Cosine Distance	Euclidean Distance	Manhattan Distance
LLaMA3.2-3B	0.1738	0.5514	11.9051
Typhoon2-8b	0.1746	0.5524	11.9234
Gemma-3-4b	0.1736	0.5494	11.8533
Qwen3-8B	0.1765	0.5624	12.1406

การประเมินความใกล้เคียงระหว่างคำตอบ

การประเมินความใกล้เคียงระหว่างคำตอบที่ได้โดยพิจารณาจาก 3 เมตริกหลัก ได้แก่ Cosine Distance, Euclidean Distance และ Manhattan Distance การวิเคราะห์เปรียบเทียบลักษณะการแจกแจงระยะห่างโดยใช้ข้อมูลตัวอย่างจำนวน 4,000 รายการวิเคราะห์ความแตกต่างอย่างชัดเจนในลักษณะการแจกแจงของแต่ละวิธี

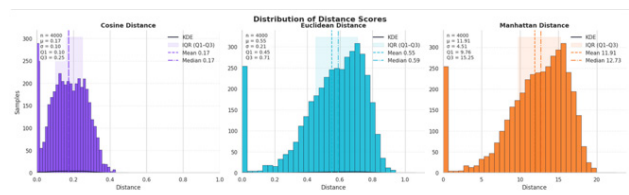


Figure 4 Average Similarity Score of Model LLaMA3.2-3B

Model LLaMA3.2-3B ผลการวิเคราะห์แสดงให้เห็นความแตกต่างลักษณะการแจกแจงโดย Cosine Distance มีการแจกแจงที่มีลักษณะใกล้เคียงกับการแจกแจงปกติแต่เบ้ขวาเล็กน้อย ด้วยค่าเฉลี่ย (μ) และค่ามัธยฐานที่เท่ากับ 0.17 และมี Interquartile Range (IQR) ที่แคบเพียง 0.15 ($Q1 = 0.10$, $Q3 = 0.25$) สะท้อนถึงการกระจุกตัวของข้อมูลในช่วงค่าที่ค่อนข้างต่ำ ในทางตรงกันข้าม Euclidean Distance แสดงการแจกแจงแบบปกติที่สมมาตรมากขึ้น ด้วยค่าเฉลี่ย 0.55 และค่ามัธยฐาน 0.59 และมี IQR เท่ากับ 0.26 ($Q1 = 0.45$, $Q3 = 0.71$) ซึ่งแสดงถึงการกระจายตัวที่สมดุรอบค่ากลาง ส่วน Manhattan Distance มีการแจกแจงที่เบ้ขวาอย่างเด่นชัด ด้วยค่าเฉลี่ย 11.91 ที่ต่ำกว่าค่ามัธยฐาน 12.73 และมี IQR ที่กว้างมาก คือ 8.52 ($Q1 = 9.76$, $Q3 = 15.25$) สะท้อนถึงการกระจายตัวที่หลากหลาย

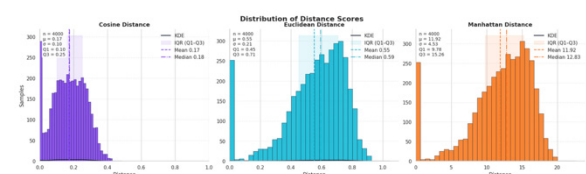


Figure 5 Average Similarity Score of Model Typhoon2-8b

Model Typhoon2-8b ผลการวิเคราะห์แสดงให้เห็นลักษณะการแจกแจงและคุณสมบัติทางสถิติ Cosine Distance มีการแจกแจงที่เบ้ขวาอย่างเด่นชัด ด้วยค่าเฉลี่ย (μ) 0.17 ที่ใกล้เคียงกับค่ามัธยฐาน 0.18 และมี Interquartile Range (IQR) ที่แคบเพียง 0.15 ($Q1 = 0.10$, $Q3 = 0.25$) การกระจุกตัวของข้อมูลในช่วงค่าที่ค่อนข้างต่ำและมีการกระจายตัวที่จำกัด ในทางตรงกันข้าม Euclidean Distance แสดงการแจกแจงแบบปกติที่สมมาตรมากที่สุด ด้วยค่าเฉลี่ย 0.55 และค่ามัธยฐาน 0.59 ที่ใกล้เคียงกันและมี IQR เท่ากับ 0.26 ($Q1 = 0.45$, $Q3 = 0.71$) แสดงการกระจายตัวที่สมดุลและสมมาตรรอบค่ากลาง Manhattan Distance มีการแจกแจงที่เบ้ขวาอย่างเด่นชัดที่สุด ด้วยค่าเฉลี่ย 11.92 ที่ต่ำกว่าค่ามัธยฐาน 12.83 และมี IQR ที่กว้างมาก คือ 5.48 ($Q1 = 9.78$, $Q3 = 15.26$) สะท้อนถึงการกระจายตัวที่หลากหลาย

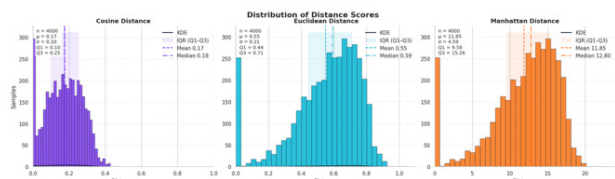


Figure 6 Average Similarity Score of Model Gemma-3-4b

Model Gemma-3-4b ผลการวิเคราะห์แสดงให้เห็นความแตกต่างลักษณะการแจกแจงโดย Cosine Distance มีการแจกแจงที่เบ้ขวาอย่างเด่นชัด ด้วยค่าเฉลี่ย (μ) 0.17 และค่ามัธยฐาน 0.18 ที่ใกล้เคียงกัน พร้อมด้วย Interquartile Range (IQR) ที่แคบเพียง 0.15 ($Q1 = 0.10$, $Q3 = 0.25$) สะท้อนถึงการกระจุกตัวของข้อมูลในช่วงค่าที่ค่อนข้างต่ำและมีการกระจายตัวที่จำกัด ในทางตรงกันข้าม Euclidean Distance แสดงการแจกแจงแบบปกติที่สมมาตรมากที่สุด ด้วยค่าเฉลี่ย 0.55 และค่ามัธยฐาน 0.59 ที่ใกล้เคียงกัน และมี IQR เท่ากับ 0.26 ($Q1 = 0.45$, $Q3 = 0.71$) แสดงถึงการกระจายตัวที่สมดุลและสมมาตรรอบค่ากลาง ส่วน Manhattan Distance มีการแจกแจงที่เบ้ซ้ายอย่างเด่นชัด ด้วยค่าเฉลี่ย 11.85 ที่ต่ำกว่าค่ามัธยฐาน 12.80 อย่างชัดเจน และมี IQR ที่ 5.71 ($Q1 = 9.55$, $Q3 = 15.26$) สะท้อนถึงการกระจายตัวที่หลากหลายและมีความไม่สมมาตรสูง โดยพบว่ามีข้อมูลจำนวนมากที่มีค่าสูงกว่าค่ากลางของการแจกแจง

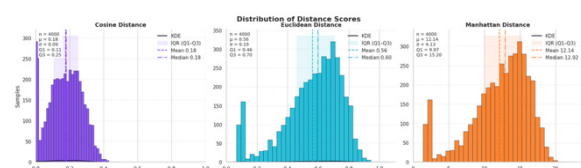


Figure 7 Average Similarity Score of Model Qwen3-8B

Model Qwen3-8B ผลการวิเคราะห์แสดงให้เห็นความแตกต่างลักษณะการแจกแจงโดย Cosine Distance มีการแจกแจงที่เบ้ขวาอย่างเด่นชัด ด้วยค่าเฉลี่ย (μ) และค่ามัธยฐานที่เท่ากับ 0.18 และมีค่าเบี่ยงเบนมาตรฐาน (σ) ที่ต่ำเพียง 0.09 พร้อมด้วย Interquartile Range (IQR) ที่แคบเท่ากับ 0.16 ($Q1 = 0.09$, $Q3 = 0.25$) การกระจุกตัวของข้อมูลในช่วงค่าที่ค่อนข้างต่ำและมีความแปรผันที่จำกัดในทางตรงกันข้าม Euclidean Distance แสดงการแจกแจงที่เข้าใกล้แบบปกติมากที่สุด ด้วยค่าเฉลี่ย 0.56 และค่ามัธยฐาน 0.60 ที่ใกล้เคียงกัน มีค่าเบี่ยงเบนมาตรฐาน 0.19 และ IQR เท่ากับ 0.24 ($Q1 = 0.46$, $Q3 = 0.70$) แสดงถึงการกระจายตัวที่สมดุลและสมมาตร Manhattan Distance มีการแจกแจงลักษณะเบ้ซ้ายเล็กน้อย ด้วยค่าเฉลี่ย 12.14 ที่ต่ำกว่าค่ามัธยฐาน 12.92 อย่างเด่นชัด มีค่าเบี่ยงเบนมาตรฐานสูงที่ 4.13 และ IQR ที่กว้าง 5.23 ($Q1 = 9.97$, $Q3 = 15.20$) สะท้อนถึงการกระจายตัวที่หลากหลายและมีความไม่สมมาตร โดยพบว่ามีข้อมูลจำนวนมากที่มีค่าสูงกว่าค่ากลางของการแจกแจง

ผลการวิเคราะห์ความครอบคลุม Coverage Analysis

โดยการวิเคราะห์ครอบคลุม 3 ด้าน Test Coverage, Functional Coverage, Requirement Coverage โดยกำหนดข้อกำหนดความต้องการมีทั้งหมด 50 ข้อ เนื้อหาเกี่ยวกับระบบในธนาคารและระบบทั่วไป (Non-Banking) ทำการทดสอบทั้ง 4 โมเดล ได้แก่ LLaMA3.2-3B, Typhoon2-8b, Gemma-3-4b และ Qwen3-8B (Laplanche & Kassab, 2022) แสดงดัง Table 6

Table 6 Coverage Analysis

Model	Test Coverage	Functional Coverage	Requirement Coverage
LLaMA3.2-3B	87%	85%	86%
Typhoon2-8b	91%	90%	89%
Gemma-3-4b	93%	91%	90%
Qwen3-8B	90%	89%	92%

จาก Table 6 แสดงผลการวิเคราะห์ความครอบคลุมของโมเดลภาษาขนาดใหญ่ 4 โมเดล จากผลการทดลองพบว่า Gemma-3-4b แสดงประสิทธิภาพสูงสุดในด้าน Test Coverage (93%) Functional Coverage (91%) และ Requirement Coverage สูงสุด (90%) ตามด้วย Qwen3-8B ที่มี Requirement Coverage สูงสุด (92%) แต่มี Test Coverage ที่ (90%) และ Functional Coverage ที่ (89%) ใน

ขณะที่ Typhoon2-8b แสดงประสิทธิภาพในระดับกลางโดย Requirement Coverage สูงสุด (89%) Test Coverage ที่ (91%) และ Functional Coverage ที่ (90%) ตามด้วย LLaMA3.2-3B แสดงประสิทธิภาพต่ำสุดในทุกเกณฑ์การประเมิน ด้วยค่าความครอบคลุมอยู่ในช่วง Test Coverage (87%) Functional Coverage (85%) และ Requirement Coverage สูงสุด (86%) การเปรียบเทียบนี้สะท้อนให้เห็นว่าโมเดลที่มีพารามิเตอร์มากกว่าไม่จำเป็นต้องแสดงประสิทธิภาพที่ดีกว่าในทุกด้าน โดย Gemma-3-4b ที่มีขนาดเล็กกว่ากลับแสดงประสิทธิภาพในการทดสอบและฟังก์ชันการทำงาน ซึ่งอาจเป็นผลมาจากการออกแบบสถาปัตยกรรมที่เหมาะสมและกระบวนการฝึกที่มีประสิทธิภาพ

ผลการประเมินผลโดยผู้เชี่ยวชาญ Expert Review

ประเมินผลความถูกต้องของ Test Case ที่สร้างโดยโมเดลตรงกับข้อกำหนดซอฟต์แวร์ ทั้งในด้าน Input, Process และ Expected Output โดยผู้เชี่ยวชาญด้านการทดสอบซอฟต์แวร์ ได้แก่ Software Tester และ QA จำนวน 5 คน ตำแหน่ง Software Tester, QA ซึ่งมีประสบการณ์ในการทดสอบระบบซอฟต์แวร์โดยหัวข้อประเมินมีทั้งหมด 10 หัวข้อ เกณฑ์ประเมินมาตราส่วน 5 ระดับแสดงดัง Table 7 ผู้เชี่ยวชาญให้

คะแนนประเมินในแต่ละหัวข้อ ตั้งแต่ระดับ 1 ต่ำที่สุด ถึงระดับ 5 สูงที่สุด โดยมีคะแนนรวมเต็ม 50 คะแนน ผลการประเมินนี้แสดงให้เห็นถึงคุณภาพของ Test Case ที่สร้างจากโมเดลในมุมมองของผู้ปฏิบัติงานจริง ซึ่งเป็นปัจจัยสำคัญในการพิจารณาความเหมาะสมของการประยุกต์ใช้โมเดลในการสร้างกรณีทดสอบสำหรับงานทดสอบซอฟต์แวร์ (Ammann & Offutt, 2017) และผลการทดลองแสดง Figure 8

Table 7 Criteria and Operational Definitions for Satisfaction Ratings

Satisfaction	Description
5 - Very satisfied	Fully meets the criteria clear and practically applicable.
4 - Satisfied	Meets most criteria with good detail.
3 - Moderately satisfied	Adequately meets the criteria but needs improvement.
2 - Somewhat	Meets some criteria lacks completeness.
1 - Very dissatisfied	Does not meet the criteria not directly usable.

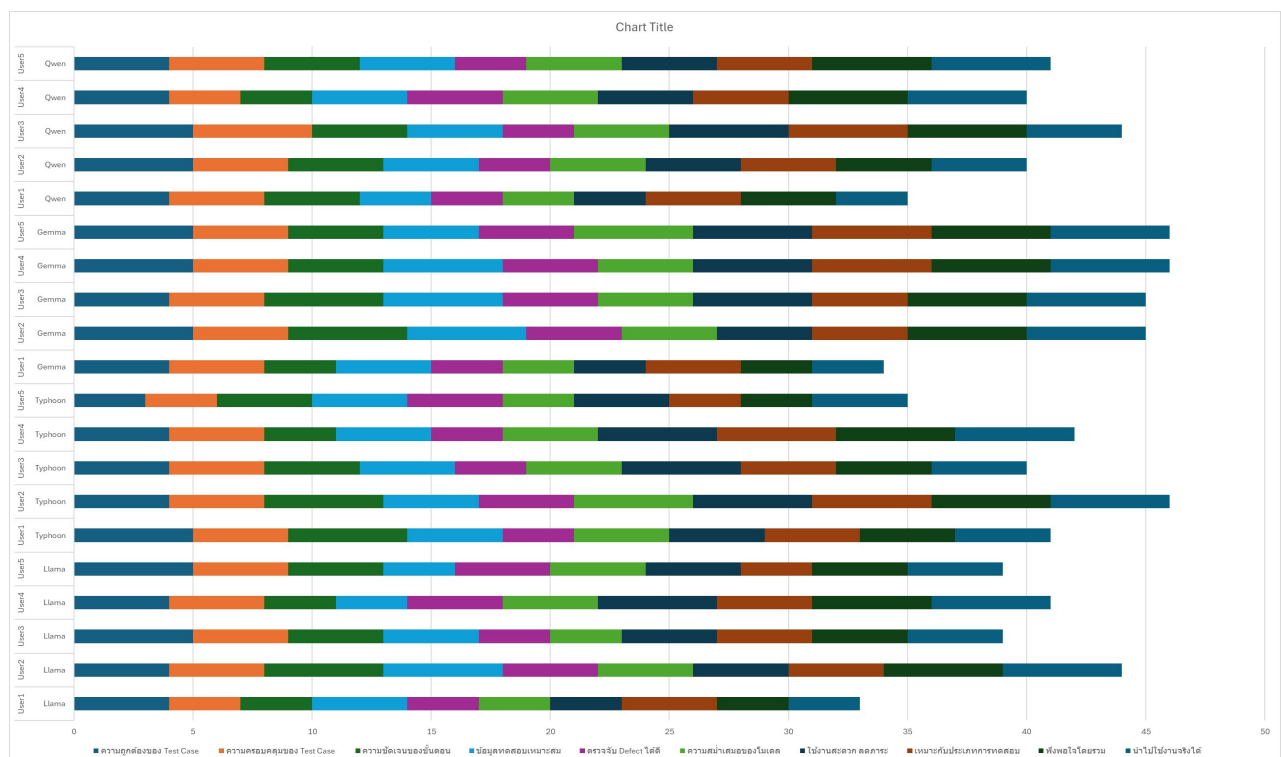


Figure 8 Graph Showing Quality Evaluation Results of Test Cases

จาก Figure 8 จากผลการประเมินความพึงพอใจของผู้เชี่ยวชาญด้านการทดสอบซอฟต์แวร์ต่อกรณีทดสอบ (Test Case) ที่สร้างโดยโมเดล พบว่าโมเดล Gemma-3-4B มีประสิทธิภาพโดดเด่นที่สุด โดยได้รับ คะแนนเฉลี่ย 43.2 คะแนน คิดเป็นร้อยละ 86.4% เป็นคะแนนสูงสุดในแทบทุกหัวข้อการประเมิน สะท้อนให้เห็นถึงความสามารถของโมเดลในการสร้าง Test Case ที่มีความครอบคลุมมีขั้นตอนที่ชัดเจนและสอดคล้องกับข้อกำหนดของระบบ Requirement รองลงมาคือโมเดล Typhoon2-8B คะแนนเฉลี่ย 40.8 คะแนน คิดเป็นร้อยละ 81.6% แสดงให้เห็นถึงประสิทธิภาพที่ดีในหลายด้าน โดยเฉพาะด้านความถูกต้องและความครอบคลุมของกรณีทดสอบซึ่งเป็นปัจจัยสำคัญในการประเมินคุณภาพ Test Case โมเดล Qwen3-8B คะแนนเฉลี่ย 40 คะแนน คิดเป็นร้อยละ 80% และ LLaMA-3.2-3B คะแนนเฉลี่ย 39.2 คะแนน คิดเป็นร้อยละ 78.4% พบว่ามีผลการประเมินใกล้เคียงกันตามลำดับ แม้จะมีคะแนนเฉลี่ยต่ำกว่าโมเดลอื่น แต่ยังคงอยู่ในระดับที่สามารถยอมรับได้สำหรับการสร้าง Test Case เพื่อการทดสอบซอฟต์แวร์ทุกโมเดลมีจุดแข็งและจุดอ่อนที่แตกต่างกันในแต่ละหัวข้อการประเมิน สะท้อนให้เห็นถึงศักยภาพของโมเดลในการสร้าง Test Case ที่มีคุณภาพเพียงพอสำหรับการประยุกต์ใช้ในงานทดสอบซอฟต์แวร์จริง ในการปรับปรุงประสิทธิภาพ เช่น การเพิ่มความครอบคลุมเชิงลึกของกรณีทดสอบ (Depth of Coverage), การสร้างกรณีเชิงลบ (Negative Case) และการเพิ่มความหลากหลายของสถานการณ์ทดสอบ

การใช้งานระบบ Chatbot

ในงานวิจัยนี้ได้พัฒนาระบบสร้างกรณีทดสอบอัตโนมัติโดยใช้โมเดลภาษาขนาดใหญ่เพื่อรองรับการโต้ตอบผ่านระบบ Chatbot ซึ่งช่วยให้ผู้ใช้งานสามารถป้อนข้อกำหนดความต้องการของระบบ (Requirement) และรับกรณีทดสอบที่สอดคล้องกับเงื่อนไขดังกล่าวได้โดยอัตโนมัติผ่านอินเทอร์เฟซ

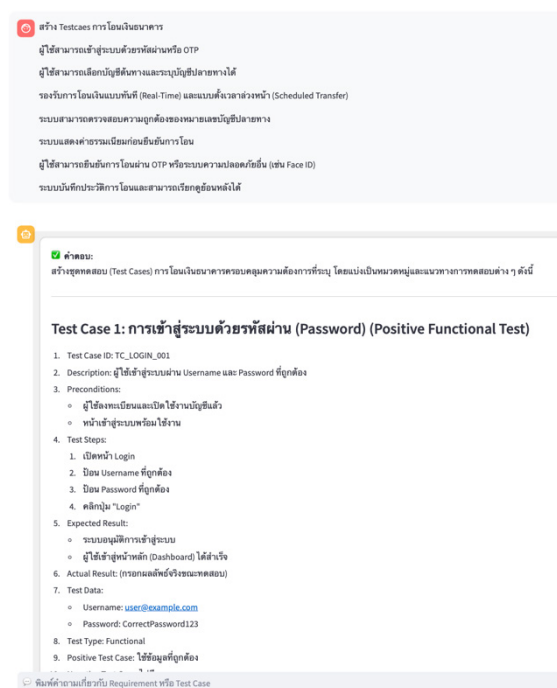


Figure 9 Chatbot User Interface Display

จาก Figure 9 แสดงตัวอย่างการใช้งานระบบสร้าง Test Case อัตโนมัติด้วยโมเดล LLM ในบริบทของระบบธนาคาร ผ่านธนาคาร เพื่อจำลองกรณีการใช้งานจริงของผู้ใช้ที่เข้าสู่ระบบด้วยรหัสผ่านอย่างถูกต้อง แสดงให้เห็นกรณีการเข้าสู่ระบบด้วยรหัสผ่านที่ถูกต้อง ซึ่งเป็นหนึ่งใน Positive Functional Test ของระบบธนาคารผ่านธนาคาร โดยผู้ใช้งานใส่ Requirement ที่ต้องการ เช่น “เข้าสู่ระบบด้วยรหัสผ่าน” ระบบจะสร้าง Test Case ที่ประกอบด้วย Testcase, Test Steps, Expected Result และข้อมูลตัวอย่าง Test Data อย่างครบถ้วน เพื่อให้สามารถนำไปใช้ทดสอบระบบได้จริง

สรุปผลการทดลองและข้อเสนอแนะ

1. การเปรียบเทียบประสิทธิภาพของโมเดล LLM ในการสร้าง Test Case

การประเมินและเปรียบเทียบประสิทธิภาพของโมเดลภาษาขนาดใหญ่ จำนวน 4 โมเดล ได้แก่ LLaMA3.2-3B, Typhoon2-8b, Gemma-3-4b และ Qwen3-8B โดยมุ่งเน้นการประเมินความสามารถของแต่ละโมเดลในการสร้างกรณีทดสอบจากข้อกำหนดความต้องการของซอฟต์แวร์ โดยใช้ตัวชี้วัดหลัก 4 ตัว ได้แก่ ค่า Loss, Cosine Distance, Euclidean Distance และ Manhattan Distance ผลการเปรียบเทียบพบว่า

โมเดล Qwen3-8B มีค่า Loss ต่ำที่สุดที่ (0.1558) ซึ่งสะท้อนถึงความสามารถในการเรียนรู้ของข้อมูล, โมเดล

Typhoon2-8b (0.1703), โมเดล LLaMA3.2-3B (0.1946) และ โมเดล Gemma-3-4b (0.2529) บ่งชี้ว่าโมเดล Qwen3-8B มีความแม่นยำในการสร้าง Test Case มากที่สุด ในขณะที่ Gemma-3-4b มีความคลาดเคลื่อนของผลลัพธ์สูงสุด เมื่อพิจารณาความใกล้เคียงเชิงเวกเตอร์พบว่า Gemma-3-4b ให้ผลลัพธ์ที่ใกล้เคียงกับข้อมูลจริงมากที่สุด โดยมีค่า Euclidean Distance ต่ำสุดที่ (0.5494) และ Manhattan Distance ต่ำสุดที่ (11.8533) สะท้อนข้อมูลเวกเตอร์ที่โมเดลสร้างขึ้นมีการกระจายตัวใกล้เคียงกับข้อมูลจริง ในทางกลับกัน Qwen3-8B แม้จะมีค่า Loss ต่ำที่สุดแต่กลับมีค่า Euclidean (0.5624) และ Manhattan (12.1406) สูงที่สุด ซึ่งชี้ให้เห็นว่าข้อมูลที่สร้างอาจมีการกระจายตัวในเชิงมิติสูงกว่ากลุ่มอื่น ตัวชี้วัด Cosine Distance พบว่า Typhoon2-8b มีค่าสูงสุดที่ (0.1746) และ Qwen3-8B มีค่าใกล้เคียงที่ (0.1765) แสดงถึงความใกล้เคียงของทิศทางเวกเตอร์กับข้อมูลจริง ขณะที่ Cosine Distance ของโมเดล LLaMA3.2-3B (0.1738) และ Gemma-3-4b (0.1736) อยู่ในระดับใกล้เคียงกัน การประเมินในครั้งนี้สะท้อนให้เห็นความแตกต่างด้านจุดแข็งและจุดอ่อนของแต่ละโมเดลอย่างชัดเจน โดย Qwen3-8B มีความโดดเด่นด้านความแม่นยำของการสร้าง Test Case (Loss ต่ำ) แต่มีการกระจายเชิงเวกเตอร์ค่อนข้างสูง ในขณะที่ Gemma-3-4b มีความใกล้เคียงเชิงเวกเตอร์สูงแต่ความแม่นยำต่ำกว่า ส่วน Typhoon2-8b มีจุดเด่นด้านทิศทางเวกเตอร์ที่ใกล้เคียงข้อมูลจริง และ LLaMA3.2-3B มีความสมดุลในทุกตัวชี้วัด การเปรียบเทียบนี้ชี้ให้เห็นว่าการเลือกใช้โมเดล LLM สำหรับการสร้าง Test Case ควรพิจารณาจากหลายตัวชี้วัดร่วมกัน ไม่ควรพิจารณาเพียงค่าใดค่าหนึ่งเพื่อสะท้อนประสิทธิภาพของโมเดลได้อย่างรอบด้าน ผลลัพธ์ยังบ่งชี้ว่าแต่ละโมเดลเหมาะสมกับการประยุกต์ใช้ในบริบทที่ต่างกัน ขึ้นอยู่กับความต้องการด้านความแม่นยำหรือความใกล้เคียงเชิงเวกเตอร์ของระบบทดสอบซอฟต์แวร์ที่ต้องการพัฒนา

2. การประเมินความครอบคลุมของ Test Case

การวัดความครอบคลุมโดยใช้ตัวชี้วัดหลัก 3 ประเภท ได้แก่ Test Coverage, Functional Coverage และ Requirement Coverage ผลการประเมินพบว่า

โมเดล Gemma-3-4b มีค่าความครอบคลุมสูงที่สุดในด้าน Test Coverage (93%) และ Functional Coverage (91%) พร้อมทั้งมีค่า Requirement Coverage อยู่ที่ 90% ซึ่งสะท้อนให้เห็นถึงความสามารถในการสร้างกรณีทดสอบที่ครอบคลุมทั้งในระดับฟังก์ชันและข้อกำหนด

โมเดล Qwen3-8B มีค่า Requirement Coverage สูงสุดที่ 92% และมีค่า Test Coverage 90% กับ Functional Coverage 89% ซึ่งแม้จะต่ำกว่า Gemma-3-4b เล็กน้อยในด้าน Test และ Functional Coverage แต่มีจุดแข็งด้านการครอบคลุมข้อกำหนดซอฟต์แวร์

โมเดล Typhoon2-8b พบว่ามีค่าความครอบคลุม Requirement Coverage 89%, Test Coverage 91% และ Functional Coverage 90% ซึ่งแสดงถึงความสมดุลระหว่างการครอบคลุมข้อกำหนดและฟังก์ชันของระบบ

โมเดล LLaMA3.2-3B มีค่าความครอบคลุมน้อยที่สุด โดยมี Requirement Coverage 86%, Test Coverage 87% และ Functional Coverage 85% ส่งผลให้การสร้าง Test Case ไม่ครอบคลุมทุกรายละเอียดของระบบเงื่อนไขที่ซับซ้อน

ผลการประเมินชี้ให้เห็นว่าโมเดล **Gemma-3-4b** มีศักยภาพโดดเด่นในการสร้างกรณีทดสอบที่ครอบคลุมในทุกมิติของระบบมากที่สุด รองลงมาคือ Qwen3-8B และ Typhoon2-8b ซึ่งมีค่าความครอบคลุมใกล้เคียงกันและอยู่ในระดับสูง ส่วน LLaMA3.2-3B แม้จะมีความสามารถในระดับที่เหมาะสม แต่มีค่าความครอบคลุมต่ำกว่าโมเดลอื่น โดยภาพรวมผลการประเมินนี้สะท้อนว่าโมเดลที่สามารถทำความเข้าใจบริบทของข้อกำหนดและฟังก์ชันได้ดีจะมีแนวโน้มสร้าง Test Case ที่ครอบคลุมได้มากกว่า

3. ผลการประเมินจากผู้เชี่ยวชาญ

จากผลการประเมินความพึงพอใจของผู้เชี่ยวชาญด้านการทดสอบซอฟต์แวร์ต่อกรณีทดสอบ Test Case ที่สร้างโดย 4 โมเดล พบว่า

โมเดล **Gemma-3-4b** มีประสิทธิภาพโดยรวมดีที่สุด ได้คะแนนเฉลี่ย **43.2 คะแนน** คิดเป็นร้อยละ 86.4 ซึ่งเป็นคะแนนสูงสุดทุกหัวข้อ การประเมินแสดงให้เห็นถึงความสามารถของโมเดลในการสร้าง Test Case ที่มีความครอบคลุม มีความชัดเจนในขั้นตอนและมีความสอดคล้องกับข้อกำหนดของระบบ Requirement

โมเดล Typhoon2-8B คะแนนเฉลี่ย 40.8 คะแนน คิดเป็นร้อยละ 81.6 มีจุดเด่นด้านความถูกต้อง และความครอบคลุมของกรณีทดสอบ โดยเฉพาะในการสร้างกรณีทดสอบที่สามารถตรวจจับข้อผิดพลาดได้อย่างเหมาะสม

โมเดล Qwen3-8B คะแนนเฉลี่ย 40 คะแนน คิดเป็นร้อยละ 80 แสดงศักยภาพที่ดีในหลายด้าน โดยเฉพาะความชัดเจนและความสม่ำเสมอของรูปแบบ Test Case

โมเดล LLaMA3.2-3B คะแนนเฉลี่ย 39.2 คะแนน คิดเป็นร้อยละ 78.4 มีคะแนนเฉลี่ยต่ำสุดเมื่อเทียบกับโมเดล

อื่น แต่ยังคงถือว่าอยู่ในระดับที่สามารถสร้าง Test Case เพื่อการทดสอบซอฟต์แวร์จุดเด่นด้าน ความเร็วในการประมวลผลและการใช้ทรัพยากรที่ต่ำจึงเหมาะสำหรับการนำไปใช้ในสภาพแวดล้อมที่มีข้อจำกัดด้านหน่วยประมวลผล

4. การพัฒนาระบบ Chatbot

ระบบที่พัฒนาขึ้นรองรับการใช้งานผ่านเว็บไซต์ โดยผู้ใช้สามารถป้อนข้อกำหนดซอฟต์แวร์และรับ Test Case ที่สร้างโดย LLM ระบบนี้สามารถนำไปใช้จริงในองค์กรเพื่อเพิ่มประสิทธิภาพการทำงานของ Software Tester และ QA ได้อย่างมีนัยสำคัญ

5. สรุป

จากผลการวิจัยสรุปได้ว่า การประยุกต์ใช้โมเดลภาษาขนาดใหญ่ในการสร้าง Test Case สำหรับ Functional Testing แบบ Manual Testing เป็นแนวทางที่มีศักยภาพสูงสามารถเพิ่มคุณภาพ ความครอบคลุม และความยืดหยุ่นในการทดสอบซอฟต์แวร์ อีกทั้งยังรองรับการใช้งานจริงผ่านระบบ Chatbot ที่พัฒนาไว้ เมื่อรวมกับข้อเสนอแนะในการพัฒนาต่อ ระบบนี้มีศักยภาพในการต่อยอดเป็นเครื่องมือสำคัญสำหรับอุตสาหกรรมซอฟต์แวร์ในอนาคต

6. ข้อเสนอแนะ

ควรมีการขยายขอบเขตของชุดข้อมูลที่ใช้ในการฝึกและประเมินโดยเพิ่มความหลากหลายของโดเมนข้อกำหนดความต้องการ ให้ครอบคลุมบริบทที่หลากหลายมากขึ้น เช่น ด้านอุตสาหกรรม เพื่อประเมินศักยภาพของโมเดลในการสร้างกรณีทดสอบในสภาพแวดล้อมที่แตกต่างกัน โมเดลจะประยุกต์ใช้กับระบบที่มีความซับซ้อนหลากหลาย นอกจากนี้พัฒนาระบบให้สามารถแปลงกรณีทดสอบในรูปแบบสคริปต์สำหรับการทดสอบอัตโนมัติได้โดยตรง เพื่อให้สามารถเชื่อมต่อกับเครื่องมือทดสอบซอฟต์แวร์ที่ใช้กันอย่างแพร่หลายในอุตสาหกรรม เช่น Selenium สำหรับการทดสอบส่วนติดต่อผู้ใช้ (UI Testing), pytest สำหรับการทดสอบแบบ (Unit/Integration Testing) และ Postman สำหรับการทดสอบ API โดยสามารถจัดเตรียมชุดข้อมูลแบบคู่ขนาน ที่ประกอบด้วยคำอธิบาย Test Case ควบคู่กับ Test Script ภาษา Python หรือ JavaScript เพื่อสอนให้โมเดลเรียนรู้จากขั้นตอนเชิงข้อความไปเป็นโค้ดอัตโนมัติได้อย่างถูกต้องแม่นยำ ทั้งนี้ การประยุกต์ใช้เทคนิค Instruction Fine-Tuning ช่วยให้โมเดลสามารถสลับโหมดการทำงานระหว่างการสร้าง Test Case และการสร้าง Test Script ซึ่งจะทำให้ระบบสามารถดำเนินการ

ทดสอบได้ครบวงจร ตั้งแต่การสร้าง Test Case การแปลงเป็น Test Script ไปจนถึงการรันทดสอบและประเมินผลโดยอัตโนมัติ ช่วยเพิ่มศักยภาพของระบบให้สามารถประยุกต์ใช้ได้จริง

กิตติกรรมประกาศ

ผู้วิจัยขอขอบพระคุณผู้ที่มีส่วนเกี่ยวข้องทุกท่านที่เอื้อเฟื้อเครื่องมือ อุปกรณ์ ที่จำเป็นต่อการดำเนินงานวิจัยครั้งนี้เป็นอย่างดี ทั้งนี้ ขอขอบพระคุณผู้เชี่ยวชาญด้านการทดสอบซอฟต์แวร์ที่ได้กรุณาสละเวลาให้คำปรึกษา ให้ข้อเสนอแนะ และแนวทางในการพัฒนาระบบอย่างสร้างสรรค์ รวมถึงผู้ที่มีส่วนเกี่ยวข้องทุกท่านที่ให้ความร่วมมือและสนับสนุนอย่างเต็มที่ตลอดระยะเวลาการวิจัย

เอกสารอ้างอิง

- Abbad, A., Abbad, K., & Tairi, H. (2016). Face recognition based on city-block and Mahalanobis cosine distance. In *2016 13th International Conference on Computer Graphics, Imaging and Visualization (CGIV)* (pp. 112–114). IEEE.
- Ammann, P., & Offutt, J. (2017). *Introduction to software testing* (2nd ed.). Cambridge University Press.
- Bag, S., Gupta, A., Kaushik, R., & Jain, C. (2024). RAG beyond text: Enhancing image retrieval in RAG systems. In *Proceedings of the 2024 International Conference on Electrical, Computer and Energy Technologies (ICECET)* (pp. 310–315).
- Bhatia, S., Gandhi, T., Kumar, D., & Jalote, P. (2024). Unit test generation using generative AI: A comparative performance analysis of autogeneration tools. In *Proceedings of the 2024 International Workshop on Large Language Models for Code (LLM4Code '24)* (pp. 1–8). ACM.
- Cheng, F. (2024). A comparative study of the performance of Spark-based k-means algorithm based on Euclidean distance and Manhattan distance. In *Proceedings of the 3rd International Conference on Computer Communication and Artificial Intelligence (CCAI)* (pp. 224–229).
- Cheng, Y., Wang, M., Xiong, Y., Hao, D., & Zhang, L. (2016). Empirical evaluation of test coverage for functional programs. In *2016 IEEE International*

- Conference on Software Testing, Verification and Validation (ICST)* (pp. 255–265). IEEE.
- Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)* (pp. 4171–4186).
- Dhillon, I. S., Guan, Y., & Kogan, J. (2002). Refining clusters in high-dimensional text data. In *Proceedings of the Workshop on Clustering High Dimensional Data and its Applications at the Second SIAM International Conference on Data Mining* (pp. 71–82). SIAM.
- Gao, X., & Li, G. (2016). A KNN model based on Manhattan distance to identify the SNARE proteins. *Computational and Mathematical Methods in Medicine, 2016*, Article 6480195. <https://doi.org/10.1155/2016/6480195>
- Guan, W., & Fang, Y. (2025). *Optimizing web-based AI query retrieval with GPT integration in LangChain: A CoT-enhanced prompt engineering approach*. arXiv. <https://arxiv.org/abs/2503.12345>
- Hoffmann, J., & Frister, D. (2024). Generating software tests for mobile applications using fine-tuned large language models. In *Proceedings of the 45th International Conference on Software Engineering (ICSE)* (pp. 1234–1245).
- Hu, E., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, L., & Chen, W. (2021). LoRA: Low-rank adaptation of large language models. arXiv. <https://arxiv.org/abs/2106.09685>
- Hu, J., Liao, X., Gao, J., Qi, Z., Zheng, H., & Wang, C. (2024). *Optimizing large language models with an enhanced LoRA fine-tuning algorithm for efficiency and robustness in NLP tasks*. arXiv. <https://arxiv.org/abs/2403.10567>
- Jacob, T. P., Bizotto, B. L. S., & Sathiyarayanan, M. (2024). Constructing the ChatGPT for PDF files with LangChain – AI. In *Proceedings of the 2024 International Conference on Inventive Computation Technologies (ICICT)* (pp. 78–85).
- Laplante, P. A., & Kassab, M. (2022). *Requirements engineering for software and systems*. Auerbach Publications.
- Masuda, S., Nishi, Y., & Suzuki, K. (2020). Complex software testing analysis using international standards. In *2020 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)* (pp. 241–246). IEEE.
- Nazi, A., Huang, Q., Shojaei, H., Esfeden, H. A., Mirhosseini, A., & Ho, R. (2022). Adaptive test generation for fast functional coverage closure. In *DVCON USA*.
- Omar, S. F. (2013). *A software traceability approach to support requirement-based test coverage analysis* [Doctoral dissertation, Universiti Teknologi Malaysia].
- Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). *Language models are unsupervised multitask learners*. arXiv. <https://arxiv.org/abs/1905.00546>
- Schütze, H., Manning, C. D., & Raghavan, P. (2008). *Introduction to information retrieval*. Cambridge University Press.
- Srikaewsiew, T., Khianchainat, K., Tharatipyakul, A., Pongnumkul, S., & Kanjanawattana, S. (2022). A comparison of the instructor-trainee dance dataset using cosine similarity, Euclidean distance, and angular difference. In *2022 26th International Computer Science and Engineering Conference (ICSEC)* (pp. 235–240). IEEE.
- Tiwari, D., Zhang, L., Monperrus, M., & Baudry, B. (2022). Production monitoring to improve test suites. *IEEE Transactions on Reliability*, 71(3), 1381–1397. <https://doi.org/10.1109/TR.2022.3149480>
- Wang, Z., Guo, X., & Tsuchiya, T. (2025). Graph-centric approaches for coverage optimization in software requirement testing. In *2025 IEEE 49th Annual Computers, Software, and Applications Conference (COMPSAC)* (pp. 1270–1280). IEEE.
- Wang, Z., Liu, K., Li, G., & Jin, Z. (2024). HITS: High-coverage LLM-based unit test generation via method slicing. In *Proceedings of the 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)* (pp. 45–56).

Zhang, J., Wang, F., Ma, F., & Song, G. (2022). Text similarity calculation method based on optimized cosine distance. In *2022 International Conference on Electronics and Devices, Computational Science (ICEDCS)* (pp. 37–39). IEEE.