

การพัฒนาระบบถามตอบข้อมูลทางธุรกิจด้วยภาษาธรรมชาติ บนพื้นฐานปัญญาประดิษฐ์เชิงกำเนิด

นพคุณ นันทเสนีย์* และ สุดาสุวรรณค์ งามมงคลวงศ์

คณะเทคโนโลยีดิจิทัลและนวัตกรรมดิจิทัล มหาวิทยาลัยเซาธ์อีสท์บางกอก

298 ซอยสรพาวุธ 27 แขวงบางนาเหนือ เขตบางนา กรุงเทพมหานคร 10260

รับบทความ 9 มีนาคม 2569 แก้ไขบทความ 21 พฤษภาคม 2569 ตอรับบทความ 11 มิถุนายน 2569

บทคัดย่อ

การเข้าถึงข้อมูลทางธุรกิจเพื่อการตัดสินใจอย่างรวดเร็วในปัจจุบันยังคงมีอุปสรรคสำคัญจากทักษะเฉพาะทางด้านภาษา SQL และปัญหาการสร้างข้อมูลเท็จ (Hallucination) ของแบบจำลองภาษาขนาดใหญ่เมื่อต้องทำงานกับโครงสร้างฐานข้อมูลที่ซับซ้อน งานวิจัยนี้จึงมีวัตถุประสงค์เพื่อพัฒนาระบบถามตอบข้อมูลทางธุรกิจด้วยภาษาธรรมชาติ (NL2SQL) และประเมินประสิทธิภาพเชิงเทคนิค โดยประยุกต์ใช้เทคนิคการดึงข้อมูลเสริม (Retrieval-Augmented Generation: RAG) ร่วมกับการให้บริบทโครงสร้างฐานข้อมูล (Schema Injection) ผ่านโมเดลปัญญาประดิษฐ์เชิงกำเนิด gpt-oss-120b บนหน่วยประมวลผลความเร็วสูง Groq LPU กระบวนการวิจัยดำเนินการทดสอบเชิงนำร่องผ่านชุดข้อมูลมาตรฐาน Spider 1.0 ในโดเมนห้างสรรพสินค้า จำนวน 25 คำถาม ครอบคลุมระดับความยากตั้งแต่ง่ายถึงยาก ผลการวิจัยพบว่าระบบที่พัฒนาขึ้นมีประสิทธิภาพสูงกว่าวิธีการป้อนคำถามแบบพื้นฐาน (Zero-shot Baseline) อย่างมีนัยสำคัญ โดยมีความถูกต้องของผลลัพธ์ข้อมูล (Execution Accuracy: EX) สูงถึง 96.00% ขณะที่ความถูกต้องของโครงสร้างคำสั่ง (Exact Matching: EM) อยู่ที่ 32.00% ด้านประสิทธิภาพเวลาประมวลผล ระบบมีระยะเวลาตอบสนองเฉลี่ยเพียง 0.73 วินาที ซึ่งผ่านเกณฑ์มาตรฐานปฏิสัมพันธ์ระหว่างมนุษย์และคอมพิวเตอร์ (HCI) ที่กำหนดไว้ไม่เกิน 3 วินาที ผลลัพธ์ของงานวิจัยนี้ชี้ให้เห็นว่าการบูรณาการสถาปัตยกรรมฐานข้อมูลร่วมกับเทคโนโลยีปัญญาประดิษฐ์เชิงกำเนิด สามารถแก้ปัญหาการหวนข้อมูลและลดความหน่วงในการสื่อสารได้อย่างมีประสิทธิภาพ ช่วยให้ผู้ใช้ทั่วไปและผู้บริหารสามารถเข้าถึงข้อมูลเชิงลึกเพื่อสนับสนุนการตัดสินใจทางธุรกิจได้อย่างแม่นยำและรวดเร็วในระดับการใช้งานจริง

คำสำคัญ: การแปลงภาษาธรรมชาติเป็นคำสั่งฐานข้อมูล; ปัญญาประดิษฐ์เชิงกำเนิด; โมเดลภาษาขนาดใหญ่; ความถูกต้องของผลลัพธ์ข้อมูล

Development of a Business Data Question-Answering System Using Natural Language Based on Generative Artificial Intelligence

Noppakun Nantasane^{*} and Sudasawan Ngammongkolwong

Faculty of Digital Technology and Digital Innovation, Southeast Bangkok University
298 Soi Sanphawut 27, Bang Na Nuea Subdistrict, Bang Na District, Bangkok, 10260

Received 9 March 2026; Revised 21 May 2026; Accepted 11 June 2026

Abstract

Accessing business information for rapid decision-making today remains significantly hampered by specialized SQL language skills and the problem of hallucination in large-scale language models when working with complex database structures. This research aims to develop a natural language-to-database (NL2SQL) business information querying system and evaluate its technical performance by applying Retrieval-Augmented Generation (RAG) techniques combined with schema injection through the gpt-oss-120b generative AI model on a Groq LPU. The research process involved pilot testing using a standard Spider 1.0 dataset in the department store domain, consisting of 25 questions covering difficulty levels from easy to difficult. The results showed that the developed system performed significantly better than the basic query input method (zero-shot baseline), with an execution accuracy of up to 96.00% and an exact matching (EM) accuracy of 32.00%. Processing time efficiency was also assessed. The system has an average response time of only 0.73 seconds, which meets the human-computer interaction (HCI) benchmark of no more than 3 seconds. The results of this research confirm that integrating database architecture with generative artificial intelligence technology can effectively solve data hallucination problems and reduce communication latency, enabling general users and executives to access insightful data to support business decision-making accurately and quickly at a real-world level.

Keywords: NL2SQL; Generative AI; Large Language Models; Execution Accuracy

** Corresponding Author. Tel.: +669 6456 5993, E-mail Address: noppakun@gmail.com*

1. บทนำ

ในยุคดิจิทัล องค์การธุรกิจมีการเก็บรวบรวมข้อมูลในปริมาณมหาศาลจากหลายแหล่ง ไม่ว่าจะเป็นข้อมูลการขาย การผลิต ลูกค้า หรือการเงิน ข้อมูลเหล่านี้ถูกจัดเก็บในระบบฐานข้อมูลเชิงสัมพันธ์ (Relational Database) ซึ่งมักต้องใช้ภาษาคำสั่งเชิงโครงสร้าง (Structured Query Language: SQL) ในการเข้าถึงและวิเคราะห์ อย่างไรก็ตาม การใช้ SQL จำเป็นต้องมีความรู้เฉพาะทาง ส่งผลให้ผู้บริหารและพนักงานทั่วไปที่ไม่มีความเชี่ยวชาญด้านการเขียนโปรแกรม ประสบอุปสรรคในการเข้าถึงข้อมูลเพื่อสนับสนุนการตัดสินใจเชิงกลยุทธ์ [1]

งานวิจัยก่อนหน้านี้ได้พยายามแก้ไขข้อจำกัดดังกล่าว โดยการพัฒนา ระบบ Natural Language Interface to Database (NLIDB) เพื่ออำนวยความสะดวกให้พนักงานส่งคำสั่งเป็นภาษาธรรมชาติแทน SQL [2] อย่างไรก็ตาม วิธีการแบบดั้งเดิมมักใช้กฎ (Rule-based) หรือการจับคู่พจนานุกรม (Schema-based Mapping) ซึ่งมีข้อจำกัดด้านความยืดหยุ่น และมีความแม่นยำต่ำเมื่อต้องนำไปใช้งานกับโครงสร้างข้อมูลจริงในองค์กรที่มีความซับซ้อน

ความก้าวหน้าของแบบจำลองภาษาขนาดใหญ่ (Large Language Models: LLMs) [3], [4] ได้เปิดโอกาสใหม่ในการแก้ปัญหา โดย Liu et al. [5] ได้สำรวจสถานะปัจจุบันของงานวิจัยการแปลงภาษาธรรมชาติเป็นคำสั่งฐานข้อมูล ที่ใช้ LLMs และพบว่าแนวทางดังกล่าวมีศักยภาพสูงในการขับเคลื่อนการพัฒนา ระบบถามตอบฐานข้อมูลรุ่นต่อไป ในทำนองเดียวกัน Hong et al. [6] ได้ทบทวนความท้าทายและแนวโน้มล่าสุดของระบบ Text-to-SQL บนพื้นฐาน LLM โดยชี้ให้เห็นว่าการจัดการโครงสร้างฐานข้อมูล (Schema Linking) และการลดปรากฏการณ์ Hallucination ยังคงเป็นโจทย์สำคัญที่ต้องแก้ไข

งานวิจัยเชิงประยุกต์ได้แสดงให้เห็นถึงความเป็นไปได้ของแนวทางนี้อย่างชัดเจน Gladkykh และ Kirykov [7] ได้นำเสนอกรอบการทำงาน Daticrs Text2SQL ที่ใช้เทคนิค Retrieval-Augmented Generation (RAG) ร่วมกับการให้บริบทโครงสร้างฐานข้อมูล และพบว่าแนวทางดังกล่าวสามารถเพิ่มความแม่นยำในการสร้างคำสั่ง SQL ได้อย่างมีนัยสำคัญ ขณะที่ Chen et al. [8] ได้ศึกษาการนำ LLM มาประยุกต์ใช้สำหรับการวิเคราะห์ข้อมูลเชิงธุรกิจระดับองค์กร และพบว่าการให้บริบทฐานข้อมูลที่เหมาะสมเป็นปัจจัยสำคัญต่อคุณภาพของผลลัพธ์ในบริบทของประเทศไทย เพ็ญพิชชา คำพิระแบ่ง [9] ได้พัฒนาระบบสร้างชุดคำสั่งสืบค้นฐานข้อมูลด้วยภาษา SQL ผ่าน ChatGPT API โดยอัตโนมัติ เพื่อสนับสนุนการร้องขอข้อมูลจากผู้ที่เกี่ยวข้องในองค์กร งานวิจัยดังกล่าวแสดงให้เห็นถึงความเป็นไปได้ของการนำโมเดลภาษาขนาดใหญ่มาใช้ในบริบทข้อมูลภาษาไทย และเป็นแรงบันดาลใจสำคัญสำหรับงานวิจัยชิ้นนี้

ตารางที่ 1 เปรียบเทียบงานวิจัยด้านการแปลงภาษาธรรมชาติเป็นคำสั่งฐานข้อมูลที่เกี่ยวข้อง

งานวิจัย	วิธีการหลัก	EX (%)
Rajkumar et al. [1]	LLM + Few-shot Prompting	~65
Liu และคณะ (2024) [5]	Schema Linking	85.0
งานวิจัยนี้	RAG + Schema Injection + LPU	96.0

จากตารางที่ 1 จะเห็นได้ว่างานวิจัยที่ใช้ LLM ด้วยวิธีการพื้นฐาน (Few-shot Prompting โดยไม่มีบริบทโครงสร้างฐานข้อมูล) มีค่า EX อยู่ในระดับประมาณร้อยละ 65 [1] ในขณะที่งานวิจัยนี้ซึ่งนำ

เทคนิค Retrieval-Augmented Generation (RAG) มาประยุกต์ใช้ร่วมกับการฝังโครงสร้างฐานข้อมูล (Schema Injection) อย่างเป็นระบบ และใช้หน่วยประมวลผลเฉพาะทาง LPU สามารถยกระดับค่า EX ได้สูงถึงร้อยละ 96.00 บนชุดข้อมูลมาตรฐาน Spider 1.0 ซึ่งแสดงให้เห็นถึงศักยภาพของสถาปัตยกรรมที่พัฒนาขึ้นในการนำไปประยุกต์ใช้งานจริงในระดับองค์กร

งานวิจัยนี้จึงนำเสนอแนวทางใหม่ผ่านการบูรณาการเทคนิคการดึงข้อมูลเสริม (Retrieval-Augmented Generation) เพื่อแก้ปัญหาความยืดหยุ่น และผลิตผลอย่างเป็นรูปธรรม ควบคู่ไปกับการเลือกใช้สถาปัตยกรรมหน่วยประมวลผลภาษา (Language Processing Unit: LPU) ของ Groq ซึ่งมีความเร็วในการประมวลผลสูงกว่าระบบทั่วไป เพื่อให้ได้ระบบถามตอบที่มีทั้งความแม่นยำ และรวดเร็วในระดับที่ใช้งานจริงได้

2. ระเบียบวิธีวิจัย

การวิจัยครั้งนี้เป็นการวิจัยเชิงทดลอง (Experimental Research) เพื่อพัฒนาระบบต้นแบบ (Prototype) ในการแปลงภาษาธรรมชาติเป็นคำสั่งฐานข้อมูล และประเมินประสิทธิภาพเปรียบเทียบกับวิธีการพื้นฐาน โดยมีขั้นตอนและระเบียบวิธีวิจัยดังต่อไปนี้

2.1 เครื่องมือและเทคโนโลยีที่ใช้ในการพัฒนา

ระบบต้นแบบถูกพัฒนาขึ้นในรูปแบบเว็บแอปพลิเคชัน (Web Application) โดยอาศัยสถาปัตยกรรมและเทคโนโลยีที่สำคัญ ได้แก่

1. ส่วนเชื่อมต่อผู้ใช้งาน (Frontend): พัฒนาด้วยเฟรมเวิร์ก Vue.js เพื่อรับคำถามภาษาธรรมชาติจากผู้ใช้
2. ส่วนจัดการตรรกะของระบบ (Backend): พัฒนาด้วย PHP (Yii2 Framework) ทำหน้าที่เป็นตัวกลางในการจัดการ Prompt และเชื่อมต่อ API

3. ระบบฐานข้อมูล (Database): ใช้ MySQL ในการจัดเก็บโครงสร้างข้อมูล (Schema) และข้อมูลจำลอง

4. ปัญญาประดิษฐ์ (AI Model): ใช้ Groq API ร่วมกับโมเดลภาษาขนาดใหญ่รุ่น gpt-oss-120b (สถาปัตยกรรม Mixture-of-Experts) ซึ่งประมวลผลผ่านชิป LPU (Language Processing Unit) เพื่อให้เกิดความรวดเร็วในการวิเคราะห์ความหมายและสร้างคำสั่ง SQL

5. สเปกคอมพิวเตอร์แม่ข่าย

- Apple M3 RAM 8GB

- OS Tahoe 26.3.1

6. เครือข่าย

- Fiber Optic 1000/500 Mbps

7. สภาพแวดล้อมซอฟต์แวร์

- Yii2 2.0.53

- MySQL 8.0

- PHP 8.3.29

- Vue.js 3.5.13

- DBeaver Community Edition 26.04

2.2 การรวบรวมและเตรียมชุดข้อมูลทดสอบ (Dataset Preparation)

การวิจัยนี้ใช้ชุดข้อมูลนำร่อง (Pilot Study) จำนวน 25 คำถาม จากฐานข้อมูลมาตรฐาน Spider 1.0 [10] โดเมนห้างสรรพสินค้าโดยผ่านกระบวนการแปลเป็นภาษาไทยและรักษาความหมายเดิมอย่างเคร่งครัด โดยผ่านการตรวจสอบเพื่อให้มั่นใจว่ายังคงรักษาความหมายและบริบทของคำถามเดิมไว้อย่างครบถ้วน การใช้ชุดคำถามจำนวน 25 ข้อ มีขนาดกลุ่มตัวอย่างที่สอดคล้องกับเกณฑ์การประเมินขั้นนำร่อง (Pilot Study) สำหรับการทดสอบระบบซอฟต์แวร์ต้นแบบ [11] โดยมีวัตถุประสงค์หลักเพื่อทดสอบการทำงานของระบบต้นแบบ (Prototype Testing) และพิสูจน์แนวคิด (Proof of Concept: PoC) เบื้องต้นในการนำ

Generative AI มาประยุกต์ใช้แปลงภาษาธรรมชาติเป็นคำสั่ง SQL แม้จำนวนข้อคำถามดังกล่าวจะยังไม่ครอบคลุมและเพียงพอต่อการนำไปใช้อ้างอิงสำหรับการประยุกต์ใช้งานจริงในระดับการผลิต (Production Level) แต่ได้ถูกออกแบบมาให้มีการกระจายระดับความซับซ้อนอย่างรอบด้าน เพื่อให้เห็นแนวโน้มความเป็นไปได้ในการพัฒนาต่อยอด โดยมุ่งเน้นการประเมินจากความถูกต้องของผลลัพธ์ข้อมูล (Execution Accuracy: EX) ซึ่งถือเป็นตัวชี้วัดที่สำคัญที่สุดสำหรับระบบ Text-to-SQL สำหรับการใช้งานจริง ตามแนวทางของ Li et al. [12] เพื่อให้ครอบคลุมการทดสอบอย่างรอบด้านโดยแบ่งเป็น 3 ระดับดังนี้

1. ระดับง่าย (Simple) จำนวน 10 ข้อ: เป็นการสืบค้นข้อมูลจากตารางเดียว (Single Table) ไม่มีการเชื่อมโยงตาราง (No JOIN) อาศัยเพียงเงื่อนไขพื้นฐาน เช่น WHERE, ORDER BY หรือ LIMIT

2. ระดับปานกลาง (Medium) จำนวน 10 ข้อ: เป็นการสืบค้นข้อมูลที่มีการเชื่อมโยง 1-2 ตาราง หรือใช้

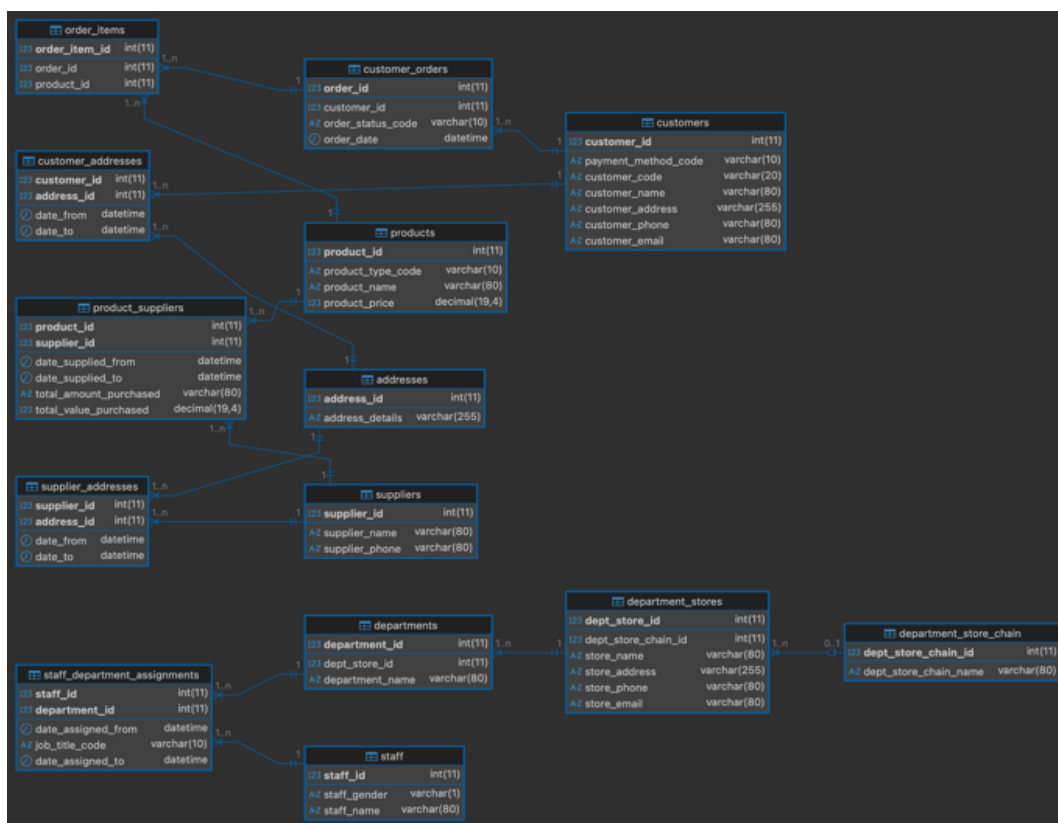
ฟังก์ชันจัดกลุ่มข้อมูล เช่น GROUP BY, HAVING และ Aggregate Functions (COUNT, MAX, MIN)

3. ระดับยาก (Hard) จำนวน 5 ข้อ: เป็นการสืบค้นที่มีความซับซ้อนสูง ต้องเชื่อมโยงมากกว่า 2 ตารางขึ้นไป มีการใช้คำสั่งสืบค้นย่อยซ้อนทับ (Subqueries) หรือใช้ตัวดำเนินการทางเซต (Set Operations) เช่น INTERSECT, EXCEPT

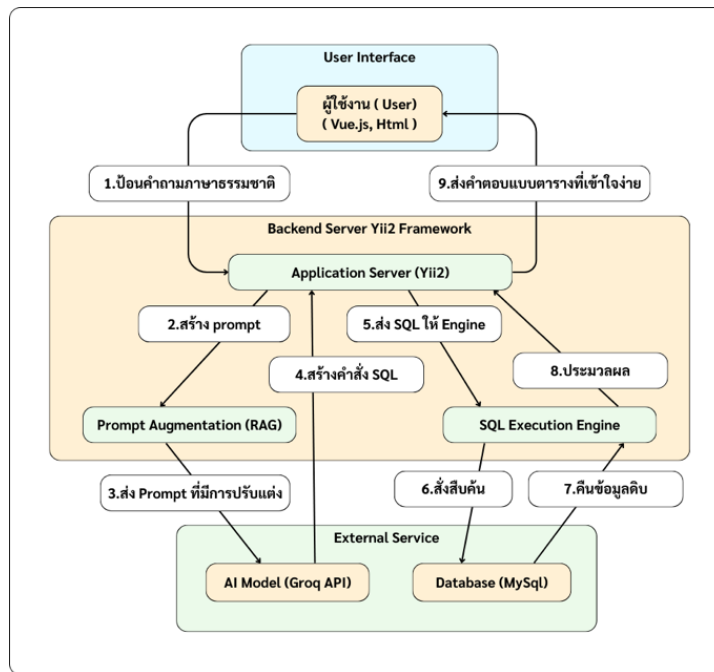
โดยที่โดเมนทางสรรพสินค้าประกอบด้วยข้อมูล 14 ตารางและมีข้อมูลตัวอย่างตารางละ 15 รายการโดยมีรายละเอียดตามรูปที่ 1

2.3 สถาปัตยกรรมและขั้นตอนการทำงานของระบบ (System Process Flow)

กระบวนการทำงานของระบบที่พัฒนาขึ้น (Proposed System) อาศัยเทคนิคการปรับแต่งคำสั่ง (Prompt Engineering) เพื่อลดปัญหาการหลอนข้อมูล (Hallucination) ของ AI โดยมีขั้นตอนดังรูปที่ 2



รูปที่ 1 ER-Diagram อธิบายโครงสร้างฐานข้อมูล Department Store



รูปที่ 2 แสดงขั้นตอนการทำงานของระบบ

1. ผู้ใช้งานป้อนคำถามด้วยภาษาธรรมชาติ: ผู้ใช้งาน (User) ได้ตอบกับระบบผ่าน User Interface ซึ่งพัฒนาด้วย Vue.js และ HTML โดยสามารถพิมพ์คำถามทางธุรกิจในรูปแบบภาษาธรรมชาติ

2. ส่งคำถามไปยัง Backend Server: คำถามจากผู้ใช้งานจะถูกส่งไปยัง Application Server (Yii2 Framework) ซึ่งทำหน้าที่เป็นศูนย์กลางในการควบคุมกระบวนการทั้งหมดของระบบ

3. การสร้าง Prompt และทำ Prompt Augmentation (RAG): Backend จะนำคำถามของผู้ใช้ไปเข้าสู่กระบวนการ Prompt Augmentation (RAG) โดยดึงข้อมูลโครงสร้างฐานข้อมูล (Schema), ความหมายของตาราง (Table), คอลัมน์ (Column) และกฎทางธุรกิจ (Business Rules) จากนั้นจะนำข้อมูลเหล่านี้มาประกอบกับคำถามของผู้ใช้ เพื่อสร้าง Prompt ที่มีบริบทครบถ้วนและแม่นยำมากขึ้น เพื่อให้เข้าใจกระบวนการสร้างบริบท (Context-Aware Prompt) อย่างเป็นรูปธรรม ตัวอย่างต่อไปนี้แสดง

โครงสร้าง Prompt ที่ระบบสร้างขึ้นจริงก่อนส่งไปยังโมเดล gpt-oss-120b ซึ่งประกอบด้วย 3 ส่วนหลัก ดังนี้

ส่วนที่ 1 - System Prompt (คำสั่งบทบาท): กำหนดบทบาทและข้อจำกัดของโมเดล เช่น "You are an expert SQL query generator. Your task is to convert Thai natural language questions into valid MySQL SQL queries based on the provided database schema. Return ONLY the SQL query without any explanation or markdown formatting."

ส่วนที่ 2 - Context Injection (การฝังบริบทฐานข้อมูล): ระบบดึงโครงสร้างฐานข้อมูล (Database Schema) ชื่อตาราง ชื่อคอลัมน์ ประเภทข้อมูล และความสัมพันธ์ (Foreign Keys) มาประกอบเป็นบริบทตัวอย่างเช่น

Database Schema:

Table: products

(product_id INT PK,
product_name VARCHAR,

product_type_code VARCHAR,
product_price DECIMAL)

Table: customers

(customer_id INT PK,
customer_name VARCHAR,
customer_address VARCHAR,
payment_method_code VARCHAR)

Table: customer_orders

(order_id INT PK, customer_id INT
FK->customers, order_status_code
VARCHAR, order_date DATE)

ส่วนที่ 3 - User Question (คำถามของผู้ใช้):
คำถามภาษาไทยที่ผู้ใช้งานป้อนเข้ามา เช่น คำถาม:
"ค้นหาชื่อของสินค้าประเภท Hardware ที่มีราคาสูง
ที่สุด" การออกแบบ Prompt ในลักษณะนี้ อาศัย
หลักการของ Prompt Engineering [13] ที่เน้นการให้
บริบทที่ครบถ้วนและชัดเจนแก่โมเดล ส่งผลให้โมเดล
สามารถอ้างอิงชื่อตารางและคอลัมน์ที่ถูกต้องได้โดยตรง
แทนที่จะต้องคาดเดาขึ้นมาเอง ซึ่งเป็นกลไกหลักที่ช่วย
ขจัดปัญหา Hallucination [14]

4. ส่ง Prompt ที่ปรับปรุงแล้วไปยัง AI Model:
Prompt ที่ผ่านการปรับแต่งแล้วจะถูกส่งไปยัง AI
Model (Groq API) ซึ่งทำงานด้วยหน่วยประมวลผล
ความเร็วสูง (LPU) ทำหน้าที่วิเคราะห์ความหมายของ
คำถาม และแปลงเป็นคำสั่ง SQL ที่เหมาะสมกับ
โครงสร้างฐานข้อมูล

5. การสร้างคำสั่ง SQL: AI Model จะส่งผลลัพธ์
กลับมาเป็น SQL จากนั้น Backend (Yii2) จะตรวจสอบ
ความถูกต้องและความปลอดภัยของ SQL ก่อนนำไปใช้
งานจริง

6. ส่งคำสั่ง SQL ไปยัง SQL Execution Engine:
คำสั่ง SQL ที่ผ่านการตรวจสอบแล้วจะถูกส่งไปยัง SQL
Execution Engine เพื่อใช้ประมวลผลกับฐานข้อมูลจริง

7. ดึงข้อมูลจากฐานข้อมูล: SQL Execution
Engine ติดต่อกับ Database (MySQL) เพื่อดึงข้อมูล
ดิบ (Raw Data) ตามคำสั่ง SQL ที่ได้รับ

8. ประมวลผลและจัดรูปแบบผลลัพธ์: ข้อมูลที่
ได้จากฐานข้อมูลจะถูกส่งกลับมายัง Backend แปลง
ผลลัพธ์ให้อยู่ในรูปแบบของตาราง Html

9. ส่งคำตอบกลับไปแสดงผลแก่ผู้ใช้งาน

2.4 การกำหนดตัวเปรียบเทียบฐานและ การประเมินผล (Baseline Setup and Evaluation)

เพื่อให้สามารถประเมินประสิทธิภาพของระบบ
ที่พัฒนาขึ้นได้อย่างเป็นรูปธรรม ผู้วิจัยได้กำหนด ตัว
เปรียบเทียบฐาน (Baseline) โดยใช้วิธีการ Zero-shot
Prompting ซึ่งเป็นการป้อนคำถามภาษาธรรมชาติ
ให้แก่แบบจำลองภาษาขนาดใหญ่ ประมวลผลและสร้าง
คำสั่ง SQL ในทันที โดยปราศจากการให้ข้อมูลตัวอย่าง
(Examples) และไม่มีการแนบข้อมูลบริบทโครงสร้าง
ฐานข้อมูล (Database Schema) ล่วงหน้า ไม่ว่าจะเป็น
ชื่อตาราง (Tables) ชื่อคอลัมน์ (Columns) หรือ
ความสัมพันธ์ระหว่างข้อมูล (Relationships) วิธีการนี้
เป็นการพึ่งพาความรู้พื้นฐานที่โมเดลเคยถูกฝึกสอนมา
เท่านั้น ซึ่งมักบีบบังคับให้โมเดลต้อง "คาดเดา"
โครงสร้างข้อมูลเป้าหมายเอาเอง และเป็นสาเหตุหลักที่
นำไปสู่ปรากฏการณ์หลอนข้อมูล (Hallucination) ทำให้
คำสั่ง SQL ที่สร้างขึ้นไม่สามารถนำไปใช้งานจริงได้
เพื่อนำมาเปรียบเทียบกับระบบที่พัฒนาขึ้น โดยมีเกณฑ์
การประเมินประสิทธิภาพเชิงเทคนิค 3 ด้าน ดังนี้

1. ความถูกต้องของผลลัพธ์ข้อมูล (Execution
Accuracy: EX): ตรวจสอบว่าผลลัพธ์ (Data Table) ที่
ได้จากการรัน Generated SQL ตรงกับผลลัพธ์ที่ได้จาก
การรันเฉลยมาตรฐาน (Gold Standard SQL) หรือไม่
ถือเป็นตัวชี้วัดหลักของการวิจัยนี้

2. ความถูกต้องของโครงสร้างคำสั่ง (Exact Matching: EM): ตรวจสอบความเหมือนของโครงสร้างตัวอักษรและไวยากรณ์ระหว่างคำสั่งที่ AI สร้างขึ้นกับคำสั่งเฉลย

3. ประสิทธิภาพด้านเวลา (Average Response Time): วัดระยะเวลาเฉลี่ยตั้งแต่เริ่มส่งคำถามจนถึงได้รับคำตอบกลับมาจาก API (Inference Time)

ทั้งนี้ ผู้วิจัยได้กำหนดเกณฑ์มาตรฐานเบื้องต้น (Heuristic Criteria) สำหรับชี้วัดความสำเร็จของระบบต้นแบบ คือ ค่าความถูกต้องของผลลัพธ์ข้อมูล (EX) ต้องไม่ต่ำกว่าร้อยละ 80.00 และใช้เวลาตอบสนองเฉลี่ย (Response Time) ไม่เกิน 3 วินาที

2.5 เกณฑ์การประเมินและการวิเคราะห์ข้อมูล

การวิเคราะห์ข้อมูลในงานวิจัยนี้ดำเนินการโดยการคำนวณค่าสถิติเชิงพรรณนาเพื่อประเมินประสิทธิภาพเชิงเทคนิคของระบบที่พัฒนาขึ้น โดยแบ่งการประเมินออกเป็น 2 ด้าน ได้แก่ ด้านความถูกต้องของผลลัพธ์ และด้านประสิทธิภาพด้านเวลา ทั้งนี้ ได้กำหนดเกณฑ์ชี้วัดประสิทธิภาพ (Evaluation Metrics) เพื่อใช้เป็นแนวทางในการพิจารณาความเหมาะสมของระบบสำหรับการใช้งานจริง ดังรายละเอียดต่อไปนี้

2.5.1 ด้านความถูกต้อง (Accuracy)

วิเคราะห์โดยใช้ค่าร้อยละ (Percentage) การประเมินด้านความถูกต้องวิเคราะห์โดยใช้ค่าร้อยละ (Percentage) ของความถูกต้องโดยพิจารณาใน 2 มิติ ได้แก่ Exact Matching Accuracy (EM) และ Execution Accuracy (EX)

Exact Matching Accuracy (EM) หมายถึง การเปรียบเทียบคำสั่ง SQL ที่ระบบสร้างขึ้นกับชุดคำสั่งเฉลย (Gold SQL) แบบตัวอักษรต่อตัวอักษร หรือเทียบเชิงโครงสร้างไวยากรณ์ (Syntax)

โดยมีเกณฑ์การให้คะแนนดังนี้

1 = SQL ทั้งสองชุดมีโครงสร้างส่วนประกอบเหมือนกันทุกประการ (ไม่นับรวมการเว้นวรรค)

0 = มีการใช้ฟังก์ชันต่างกันหรือเขียนสลับตำแหน่งในลักษณะที่ส่งผลต่อรูปประโยค

Execution Accuracy (EX) หมายถึง ความถูกต้องของผลลัพธ์ (Execution Result) ที่ได้จากการรันคำสั่ง SQL โดยเปรียบเทียบกับผลลัพธ์จากคำสั่ง SQL มาตรฐาน (Gold Standard SQL) โดยไม่สนใจว่าวิธีการเขียน SQL จะแตกต่างกันอย่างไร โดยมีเกณฑ์การให้คะแนนดังนี้

1 = ผลลัพธ์ตรงกับ Gold Standard ทุกประการ

0 = ผลลัพธ์ไม่ตรงกับ Gold Standard

ค่าความถูกต้องโดยรวมของระบบคำนวณจากสูตรดังนี้

$$\text{Accuracy (\%)} = \frac{\text{จำนวนคำตอบที่ถูกต้อง}}{25} \times 100$$

Evaluation Metrics ด้าน Accuracy (เกณฑ์อ้างอิง)

- Accuracy $\geq$ 80% ถือว่าระบบมีความถูกต้อง อยู่ในระดับดีและเหมาะสมต่อการใช้งานเชิงธุรกิจ

- Accuracy $<$ 80% ถือว่ายังต้องมีการปรับปรุง ประสิทธิภาพของระบบ

2.5.2 ด้านประสิทธิภาพเวลา (Time Efficiency)

การประเมินประสิทธิภาพด้านเวลาใช้ค่าเฉลี่ยของระยะเวลาในการตอบสนองของระบบ (Average Response Time) โดยวัดตั้งแต่ผู้ใช้งานส่งคำถามเข้าสู่ระบบจนกระทั่งระบบแสดงผลลัพธ์สุดท้ายจากชุดคำถามทดสอบจำนวน 25 ข้อ

สูตรการคำนวณ

$$\text{Average Response Time} = \frac{\sum_{i=1}^{25} (T_i)}{25}$$

Evaluation Metrics ด้าน Response Time (เกณฑ์อ้างอิง)

Average Response Time ≤ 3 วินาที ถือว่าระบบมีประสิทธิภาพด้านการตอบสนองในระดับที่เหมาะสมสำหรับระบบถามตอบเชิงโต้ตอบ

Average Response Time > 3 วินาที ถือว่าระบบมีความล่าช้าและควรได้รับการปรับปรุง

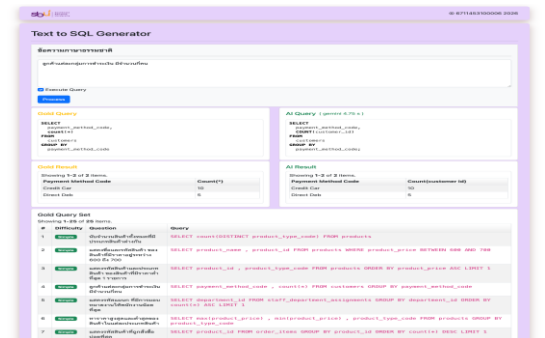
3. ผลการวิจัยและอภิปรายผล

3.1 ผลการวิจัย

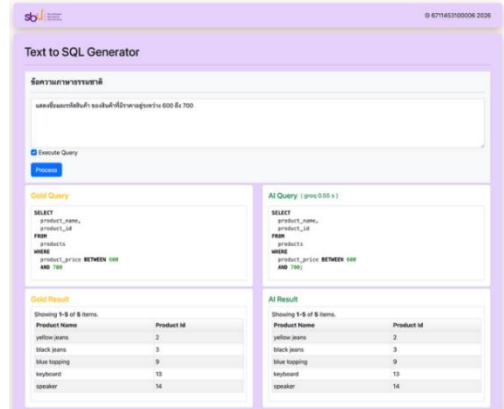
จากการทดสอบประสิทธิภาพของระบบที่พัฒนาขึ้น เทียบกับระบบพื้นฐาน (Zero-shot Baseline) และจำแนกตามระดับความยากของคำถาม ได้ผลการวิจัยดังนี้

3.1.1 ระบบต้นแบบถูกออกแบบส่วนติดต่อผู้ใช้งาน (User Interface) ให้รวบรวมฟังก์ชันหลักไว้ในหน้าจอเดียว

ระบบประกอบด้วยส่วนการทำงาน การแสดงผล และส่วนของ Query Set อ้างอิงที่ใช้ในการทดสอบ



รูปที่ 3 หน้าจอหลักการทำงานและการแสดงผล



รูปที่ 4 หน้าจอแสดงตารางข้อมูลที่ดึงจาก Database โดยใช้ SQL จาก Gold Query เพื่อใช้เปรียบเทียบความถูกต้องของผลลัพธ์ระหว่าง Gold Query กับ AI Query

3.1.2 ผลการเปรียบเทียบประสิทธิภาพกับระบบพื้นฐาน (Baseline Comparison)

ตารางที่ 2 ผลการทดสอบประสิทธิภาพด้านเวลาเปรียบเทียบ Zero-shot

ระดับความยาก (Difficulty Level)	จำนวนคำถาม (ข้อ)	เวลาเฉลี่ยการประมวลผล (วินาที/ข้อ)	
		Proposed	Zero-shot
Simple	10	0.51	0.81
Medium	10	0.75	1.05
Hard	5	1.15	1.24
Overall	25	0.73	2.48



รูปที่ 5 กราฟเวลาเฉลี่ยในการประมวลผลแยกตามระดับความยาก (Average Response Time by Difficulty)

3.1.3 ผลการทดสอบประสิทธิภาพด้านเวลาและความถูกต้องของระบบ

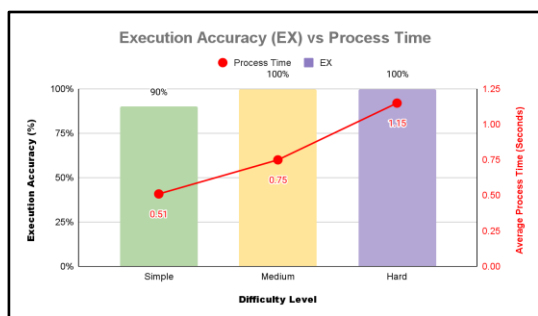
ตารางที่ 3 ตารางเปรียบเทียบประสิทธิภาพด้านเวลาและความถูกต้องของผลลัพธ์ระหว่างระบบที่มีการให้ข้อมูล Schema และไม่มี Schema

ระบบ ที่ทดสอบ	Schema	ความถูกต้องของผลลัพธ์ (EX) %	เวลาเฉลี่ย (วินาที)
Zero-shot	NO	0%	2.48
Proposed	YES	96%	0.73

ผลการทดสอบจากตารางที่ 3 พบประเด็นสำคัญดังนี้

1. ด้านเวลาประมวลผล (Time Efficiency) ระบบใช้เวลาเฉลี่ย 0.73 วินาที (ระดับง่าย 0.51 วินาที, ปานกลาง 0.75 วินาที, และยาก 1.15 วินาที) ซึ่งสอดคล้องกับเกณฑ์ประสิทธิภาพที่กำหนด (≤ 3 วินาที)

2. ด้านความถูกต้อง (Accuracy): ระบบบรรลุค่าความถูกต้องของผลลัพธ์ (EX) เฉลี่ยสูงถึง 96% (ระดับ Simple 90%, Medium 100%, Hard 100%) ในขณะที่ค่า Exact Matching (EM) อยู่ที่ 32.00%



รูปที่ 6 ความถูกต้องของผลลัพธ์ข้อมูลแยกตามระดับความยาก (Execution Accuracy (EX) vs Process Time)

3.2 การอภิปรายผล

จากการทดสอบประสิทธิภาพของระบบถามตอบข้อมูลทางธุรกิจที่พัฒนาขึ้น สามารถอภิปรายประเด็นสำคัญที่ค้นพบเชื่อมโยงกับทฤษฎีและงานวิจัยที่เกี่ยวข้องได้ดังนี้

3.2.1 ประสิทธิภาพของเทคนิค RAG และ Schema Injection ในการลดปัญหาการหลอนข้อมูล

ผลการทดลองชี้ให้เห็นว่าระบบที่พัฒนาขึ้นมีค่า Execution Accuracy (EX) สูงถึง 96.00% เมื่อเปรียบเทียบกับวิธีการประมวลผลแบบพื้นฐาน (Zero-shot Baseline) ที่มีค่า EX อยู่ที่ 0.00% อย่างสิ้นเชิง ความสำเร็จนี้เป็นผลจากการประยุกต์ใช้เทคนิคการดึงข้อมูลเสริม (Retrieval-Augmented Generation: RAG) ร่วมกับการฝังบริบทโครงสร้างฐานข้อมูล (Schema Injection) [14] ที่ระบุว่าการเสริมบริบทข้อมูลเฉพาะทาง (Knowledge-intensive tasks) ช่วยให้โมเดลภาษาขนาดใหญ่สามารถดึงข้อมูลและประมวลผลคำตอบได้แม่นยำขึ้น โดยการทำ Schema Injection ทำหน้าที่เป็นกรอบกำหนดตรรกะ (Logical Constraints) ให้แก่ AI ทำให้สามารถระบุข้อผิดพลาดและคัดค้านได้ถูกต้องตามความเป็นจริง ลดปัญหาการหลอนข้อมูล (Hallucination) ที่มักเกิดขึ้นในโมเดลแบบ Zero-shot ซึ่งขาดการรับรู้ถึงโครงสร้างฐานข้อมูลจริง ยกตัวอย่างเช่น เมื่อป้อนคำถามภาษาธรรมชาติว่า “หาชื่อและรหัสลูกค้า ที่มีที่อยู่ประกอบด้วยคำว่า ‘TN’ ” ในกรณีที่ระบบทำงานร่วมกับเทคนิคการฝังบริบทโครงสร้างฐานข้อมูล (Schema Injection) จะสามารถสร้างคำสั่ง SQL ได้อย่างถูกต้องตรงตามโครงสร้างจริง คือ

```
SELECT customer_name,  
customer_code  
FROM customers  
WHERE customer_address LIKE '%TN%';
```

แต่ในทางตรงกันข้าม เมื่อระบบทำงานแบบพื้นฐาน (Zero-shot Baseline) ผลลัพธ์คำสั่ง SQL ที่ได้กลับเป็น

```
SELECT c.name,
c.id
FROM customers c
JOIN customer_addresses a
ON a.customer_id = c.id
WHERE a.address LIKE '%TN%';
```

ซึ่งแสดงให้เห็นอย่างชัดเจนว่า หากปราศจากการให้ข้อมูลบริบทโครงสร้างฐานข้อมูลล่วงหน้า ระบบจะเกิดปรากฏการณ์หลอนข้อมูล (Hallucination) โดยการคาดเดาชื่อตาราง (Table) และชื่อคอลัมน์ (Column) ขึ้นมาเอง ซึ่งไม่มีอยู่จริงในระบบฐานข้อมูล ส่งผลให้คำสั่งดังกล่าวไม่สามารถนำไปประมวลผลผลลัพธ์ (Execution Failure) ได้

3.2.2 ความแตกต่างระหว่างความถูกต้องของผลลัพธ์ (EX) และไวยากรณ์คำสั่ง (EM)

ประเด็นที่น่าสนใจคือค่าความถูกต้องของไวยากรณ์คำสั่ง (Exact Matching: EM) ที่มีเพียง 32.00% แม้ค่า EX จะสูงถึง 96.00% ก็ตาม ปรากฏการณ์นี้สามารถอธิบายได้ด้วยทฤษฎีการประเมินเชิงความหมาย (Semantic Evaluation) ตามแนวทางของ Li et al. [12] และ Zhong et al. [15] เนื่องจากภาษา SQL มีคุณสมบัติเป็นภาษาเชิงประกาศ (Declarative Language) ที่สามารถเขียนได้หลายรูปแบบเพื่อให้ได้ผลลัพธ์ข้อมูลเดียวกัน (Semantic Equivalence) เช่น การเลือกใช้คำสั่ง JOIN ที่แตกต่างกันหรือการจัดลำดับเงื่อนไขใน WHERE Clause การที่ระบบให้ค่า EX สูงชี้ให้เห็นว่าปัญญาประดิษฐ์เชิงกำเนิดมีความสามารถในการตีความความหมาย (Intent Parsing) ได้อย่างถูกต้องเพื่อตอบสนองต่อความต้องการ

ทางธุรกิจ แม้รูปแบบการเขียนคำสั่งจะไม่ตรงกับชุดข้อมูลเฉลยทุกตัวอักษรก็ตาม

3.2.3 การตอบสนองที่ฉับไวภายใต้ทฤษฎีปฏิสัมพันธ์ระหว่างมนุษย์และคอมพิวเตอร์ (HCI)

ระยะเวลาการตอบสนองเฉลี่ยที่ 0.73 วินาทีต่อคำถาม ถือเป็นประสิทธิภาพที่โดดเด่นอย่างยิ่งเมื่อเทียบกับเกณฑ์มาตรฐานที่กำหนดไว้ไม่เกิน 3 วินาที ผลลัพธ์นี้มีความสำคัญในเชิงการนำไปใช้งานจริงตามทฤษฎีขีดจำกัดของเวลาในการตอบสนองของ Nielsen [16] ซึ่งระบุว่า การตอบสนองที่อยู่ในช่วง 1.0 วินาที จะช่วยรักษา "กระบวนการคิด (Flow of thought)" ของผู้ใช้งานให้มีความต่อเนื่องและไม่รู้สึกถูกรบกวนจากความเร็วระดับนี้เกิดขึ้นจากการใช้สถาปัตยกรรมแบบ Mixture-of-Experts (MoE) ของโมเดล gpt-oss-120b ร่วมกับการประมวลผลผ่านหน่วยประมวลผล Groq LPU ที่ถูกออกแบบมาเพื่อลดความหน่วง (Latency) ในการสร้างโทเค็นโดยเฉพาะ ซึ่งถือเป็นปัจจัยสำคัญที่ทำให้ระบบมีความพร้อมสำหรับการใช้งานในระดับองค์กรที่ต้องการผลลัพธ์แบบเรียลไทม์

3.2.4 การวิเคราะห์ข้อผิดพลาดและแนวทางปรับปรุงในบริบทภาษาไทย

จากการวิเคราะห์ข้อผิดพลาดพบว่าความคลุมเครือของคำศัพท์ภาษาไทยเป็นอุปสรรคสำคัญ เช่น คำว่า "ปริมาณ" ซึ่งระบบอาจตีความได้ทั้ง จำนวน ระเบียบ หรือ ผลรวมยอดเงินการค้นพบนี้ชี้ให้เห็นว่า ลำพังเพียงเทคนิค NL2SQL อาจยังไม่เพียงพอสำหรับการจัดการความกำกวมของภาษามนุษย์ในบางกรณี สอดคล้องกับข้อเสนอแนะที่ว่าในทางปฏิบัติควรมีการกำหนดกฎทางธุรกิจเพิ่มเติมในขั้นตอนการทำ Prompt Engineering เพื่อให้ระบบสอบถามความต้องการที่ชัดเจนจากผู้ใช้งานก่อนประมวลผลคำสั่ง เพื่อลดโอกาสการ

แสดงผลข้อมูลที่ผิดพลาดจากความเข้าใจภาษาที่คลาดเคลื่อน

4. สรุป

การวิจัยนี้บรรลุวัตถุประสงค์ในการพัฒนาระบบแปลงภาษาธรรมชาติเป็นคำสั่งฐานข้อมูลที่มีประสิทธิภาพสูงบนพื้นฐานโมเดลปัญญาประดิษฐ์เชิงกำเนิด gpt-oss-120b ผ่านหน่วยประมวลผล Groq LPU โดยระบบที่พัฒนาขึ้นสามารถทำคะแนนความถูกต้องของผลลัพธ์ข้อมูล (Execution Accuracy: EX) เฉลี่ยสูงถึงร้อยละ 96.00 และใช้เวลาประมวลผลเฉลี่ยเพียง 0.73 วินาที ซึ่งผ่านเกณฑ์ที่กำหนดไว้ (≤ 3 วินาที) ในทุกระดับความยาก

ค่า Exact Matching (EM) ที่อยู่ในระดับต่ำ (ร้อยละ 32.00) นั้น ไม่ได้สะท้อนถึงความบกพร่องของระบบ แต่เป็นผลจากคุณสมบัติของภาษา SQL ที่เป็นภาษาเชิงประกาศ (Declarative Language) ซึ่งสามารถเขียนโครงสร้างคำสั่งได้หลากหลายรูปแบบเพื่อให้ได้ผลลัพธ์เดียวกัน (Semantic Equivalence) ซึ่งสอดคล้องกับแนวทางของ BIRD Benchmark [12] และงานวิจัยด้านการประเมินเชิงความหมาย [15] ที่ชี้ว่าการใช้งาน NL2SQL จริงควรให้ความสำคัญกับ EX เป็นหลัก

ผลการทดสอบแสดงให้เห็นตัวเลขที่น่าสนใจคือระบบทำคะแนน EX ได้สมบูรณ์ 100% ในระดับ Medium และ Hard แต่ได้เพียง 90% ในระดับ Simple ซึ่งขัดกับสัญชาตญาณที่คาดว่าคำถามง่ายน่าจะให้ผลแม่นยำกว่า เมื่อวิเคราะห์กรณีผิดพลาดพบว่าต้นเหตุคือความคลุมเครือเชิงความหมายของภาษาไทย กล่าวคือ คำถามที่ระบุว่า "สินค้าที่ถูกสั่งซื้อในปริมาณมากที่สุด" นั้น มีคำสำคัญ "ปริมาณ" ที่โมเดลต้องตีความว่าหมายถึงปริมาณสะสม (Cumulative Quantity) ในตาราง product_suppliers หรือจำนวนครั้งที่ปรากฏในตาราง order_items ผลการวิเคราะห์ชี้ให้เห็นว่าการปรับปรุงในอนาคตควรเน้นที่การเพิ่ม Business

Glossary หรือคำอธิบายความหมายของคอลัมน์ (Column Description) เข้าไปใน Prompt เพื่อช่วยให้โมเดลแก้ไขความคลุมเครือทางความหมายได้อย่างถูกต้องยิ่งขึ้น

งานวิจัยนี้มีข้อจำกัดที่ควรพิจารณาในการนำผลไปใช้อ้างอิง ประการแรก ชุดคำถามทดสอบจำนวน 25 ข้อในโดเมนห้างสรรพสินค้าเพียงโดเมนเดียว เป็นการทดสอบในระดับนำร่อง (Pilot Study) เพื่อพิสูจน์แนวคิด (Proof of Concept) เท่านั้น ยังไม่สามารถอ้างอิงแทนการประเมินในระดับการผลิต (Production Level) ที่ต้องการชุดข้อมูลขนาดใหญ่และหลากหลายโดเมนได้ เช่น ชุดข้อมูล BIRD Benchmark [12] ที่ครอบคลุมกว่า 11,000 คำถามใน 37 โดเมน ประการที่สอง ระบบได้รับการทดสอบกับคำถามภาษาไทยเป็นหลัก ซึ่งอาจมีความท้าทายเพิ่มเติมในการรองรับภาษาอื่น ๆ หรือสำเนียงภาษาไทยถิ่น ประการที่สาม เกณฑ์ KPI ที่กำหนด ($EX \geq 80\%$ และเวลา ≤ 3 วินาที) เป็นเกณฑ์ที่ผู้วิจัยกำหนดขึ้นสำหรับระบบต้นแบบ มิใช่มาตรฐานอุตสาหกรรมระดับสากล ดังนั้นในการขยายผลสู่การใช้งานจริงในองค์กร ควรมีการกำหนดเกณฑ์ประเมินที่สอดคล้องกับบริบทและความต้องการเฉพาะขององค์กรนั้น ๆ

นอกจากนี้ การใช้สถาปัตยกรรม Mixture-of-Experts ร่วมกับชิป LPU ของ Groq ประกอบกับเทคนิค Prompt Engineering [13] และการให้บริบท Database Schema อย่างเป็นระบบ เป็นปัจจัยสำคัญที่ทำให้ระบบสามารถสร้างคำสั่ง SQL ที่ซับซ้อนได้อย่างถูกต้องแม่นยำและรวดเร็ว [17]

สำหรับการนำไปใช้ประโยชน์ในอนาคต องค์กรสามารถต่อยอดระบบนี้สู่การวิเคราะห์เชิงแนะนำ (Prescriptive Analytics) โดยให้ AI สืบหาต้นตอของปัญหาจากฐานข้อมูลและให้คำแนะนำเชิงธุรกิจ (Actionable Recommendations) รวมถึงพัฒนาให้ระบบสร้างแดชบอร์ดสรุปผลอัตโนมัติ (Automated

Dashboard Generation) ผ่านการทำงานร่วมกับไลบรารี Data Visualization เพื่อแปลงผลลัพธ์ SQL เป็นภาพที่ผู้บริหารเข้าใจได้รวดเร็วขึ้น ในด้านงานวิจัยควรศึกษาการนำโมเดลภาษาขนาดใหญ่มาประมวลผลบนเซิร์ฟเวอร์ขององค์กรเอง (On-Premise LLMs) เพื่อป้องกันการรั่วไหลของข้อมูลธุรกิจ และออกแบบระบบให้รองรับการเปรียบเทียบประสิทธิภาพข้ามโมเดล (Multi-Model Benchmarking) เช่น Groq, Gemini หรือ GPT-4 เพื่อหาแนวทางที่มีความคุ้มค่าสูงสุดในการนำไปใช้งานจริงในระดับองค์กร

5. กิตติกรรมประกาศ

ผู้วิจัยขอกราบขอบพระคุณ ผู้ช่วยศาสตราจารย์ ดร.สุดาสวรรค์ งามมงคลวงศ์ อาจารย์ที่ปรึกษา ที่ให้คำแนะนำทางวิชาการอย่างดียิ่ง ขอขอบคุณคณาจารย์หลักสูตร MSIT คณะเทคโนโลยีดิจิทัลและนวัตกรรมมหาวิทยาลัยเซาท์อีสต์บางกอก ที่ถ่ายทอดองค์ความรู้ตลอดจนผู้เชี่ยวชาญทุกท่านที่สละเวลาประเมินและให้ข้อเสนอแนะอันเป็นประโยชน์ งานวิจัยนี้สำเร็จลุล่วงได้อย่างสมบูรณ์

6. เอกสารอ้างอิง

- [1] N. Rajkumar, R. Li, and D. Bahdanau, "Evaluating the text-to-SQL capabilities of large language models," *arXiv Computation and Language*, arXiv:2204.00498, 2022.
- [2] F. Li and H. V. Jagadish, "Constructing an interactive natural language interface for relational databases," in *Proceedings of the VLDB Endowment*, vol. 8, no. 1, pp. 73-84, 2014.
- [3] T. B. Brown et al., "Language models are few-shot learners," *arXiv Computation and Language*, arXiv:2005.14165, 2020.
- [4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in Neural Information Processing Systems*, vol. 30, 2017.
- [5] X. Liu et al., "A survey of NL2SQL with large language models: Where are we, and where are we going?," *arXiv preprint arXiv:2408.05109*, 2024.
- [6] Z. Hong, Z. Yuan, Q. Zhang, H. Chen, J. Dong, F. Huang, and H. Huang, "Next-generation database interfaces: A survey of LLM-based text-to-SQL," *IEEE Transactions on Knowledge and Data Engineering*, vol. 37, no. 12, pp. 7328-7345, 2025.
- [7] T. Gladkykh and K. Kirykov, "Datrics Text2SQL: A framework for natural language to SQL query generation," *arXiv Databases*, arXiv:2506.12234, 2025.
- [8] A. Chen et al., "Text-to-SQL for enterprise data analytics," *arXiv preprint arXiv:2507.14372*, 2025.
- [9] P. Khampirapang, "Automatic SQL Database Query Generation Using ChatGPT API," M.S. thesis, Dept. Comput. Eng., Dhurakij Pundit Univ., Bangkok, Thailand, 2024.
- [10] T. Yu et al., "Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL

- task,” *arXiv Computation and Language*, arXiv:1809.08887, 2018.
- [11] M. A. Hertzog, “Considerations in determining sample size for pilot studies,” *Research in Nursing & Health*, vol. 31, no. 2, pp. 180–191, 2008.
- [12] J. Li et al., “Can LLM already serve as a database interface? a big bench for large-scale database grounded text-to-sqls,” *arXiv Computation and Language*, arXiv:2305.03111, 2023.
- [13] V. Liu and L. B. Chilton, “Design guidelines for prompt engineering text-to-image generative models,” in *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, 2022, pp. 1–14.
- [14] P. Lewis and E. Perez, “Retrieval-augmented generation for knowledge-intensive NLP tasks,” in *Proceedings of 34th Conference on Neural Information Processing Systems (NeurIPS 2020)*, Vancouver, Canada, 2020.
- [15] R. Zhong, T. Yu, and D. Klein, “Semantic evaluation for text-to-SQL with distilled test suites,” in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020, pp. 396–411.
- [16] J. Nielsen, *Usability Engineering*, San Francisco, CA: Morgan Kaufmann, 1994.
- [17] K. T. Pham, T. H. Nguyen, J. Jo, Q. V. H. Nguyen, and T. T. Nguyen, “Multilingual text-to-SQL: Benchmarking the limits of language models with collaborative language agents,” *arXiv Computation and Language*, arXiv:2509.24405, 2025.