แบบเสนอโครงงานพิเศษ (ปริญญานิพนธ์)

สาขาวิชาวิศวกรรมสารสนเทศและเครือข่าย
ภาควิชาเทคโนโลยีสารสนเทศ
คณะเทคโนโลยีและการจัดการอุตสาหกรรม

1. ข้อมูลขั้นต้นของโครงงาน

1.1 ชื่อโครงงาน

(ภาษาไทย) **แพลตฟอร์มวิศวกรรมความรู้เชิงสัมพันธ์การปรากฏร่วม**
(ภาษาอังกฤษ) **Co-Occurrence Knowledge Engineering Platform**

1.2 ชื่อนักศึกษาผู้ทำโครงงาน

ชื่อ-นามสกุล **นายยงยุทธ ชวนขุนทด**
สาขาวิชา **วิศวกรรมสารสนเทศและเครือข่าย**
ภาควิชา **เทคโนโลยีสารสนเทศ**
ภาคเรียนที่ 1
ปีการศึกษา 2568

1.3 ชื่ออาจารย์ที่ปรึกษา

ชื่อ-นามสกุล **รองศาสตราจารย์ ดร. อนิราช มิ่งขวัญ**

2. รายละเอียดของโครงงาน

2.1 ความเป็นมาและความสำคัญของปัญหา

ในยุคดิจิทัลปัจจุบัน ข้อมูลและความรู้ที่เกิดขึ้นในแต่ละวันมีปริมาณที่เพิ่มขึ้นอย่างรวดเร็ว โดยเฉพาะอย่างยิ่งเอกสารทางวิชาการ หนังสือ และงานวิจัยต่าง ๆ ที่มีการเผยแพร่ในรูปแบบดิจิทัล เช่น ไฟล์ PDF ซึ่งเป็นแหล่งความรู้ที่มีคุณค่าสูง อย่างไรก็ตาม การจัดการ การวิเคราะห์ และการค้นหาความเชื่อมโยงระหว่างความรู้จากเอกสารเหล่านี้ยังคงเป็นปัญหาที่ท้าทาย

ปัญหาหลักที่พบในปัจจุบันคือ การที่ผู้ใช้งานไม่สามารถมองเห็นภาพรวมของความสัมพันธ์และความเชื่อมโยงระหว่างแนวคิดต่าง ๆ ที่ปรากฏในเอกสารหลายฉบับได้อย่างชัดเจน การอ่านและทำความเข้าใจเอกสารแต่ละฉบับแยกกันทำให้เกิดการสูญเสียโอกาสในการค้นพบความรู้ใหม่ที่อาจเกิดขึ้นจากการเชื่อมโยงข้อมูลจากแหล่งที่แตกต่างกัน

นอกจากนี้ การวิเคราะห์ความถี่ของการใช้คำและการปรากฏร่วมของความรู้ **(Co-occurrence)** ในเอกสารยังเป็นกระบวนการที่ซับซ้อนและใช้เวลามาก หากต้องทำด้วยมือหรือเครื่องมือพื้นฐาน ทำให้การสกัดความรู้และการสร้างความเข้าใจเชิงลึกจากเอกสารเป็นไปได้ยาก

ด้วยเหตุนี้ จึงจำเป็นต้องมีระบบที่สามารถแปลงเอกสารจากแหล่งต่าง ๆ ให้กลายเป็นกราฟเครือข่าย **(Network Graph)** ที่แสดงความสัมพันธ์และความเชื่อมโยงระหว่างแนวคิดได้อย่างชัดเจน รวมถึงสามารถสกัดส่วนของกราฟเพื่อนำไปผสมผสานกับข้อมูลจากแหล่งอื่น ๆ เพื่อสร้างความรู้ใหม่และค้นพบความเป็นไปได้ที่ไม่เคยมีมาก่อน ซึ่งจะช่วยเพิ่มประสิทธิภาพในการจัดการความรู้และส่งเสริมการเกิดนวัตกรรมใหม่ ๆ ในอนาคต

2.2 วัตถุประสงค์ของโครงงาน

2.2.1 เพื่อพัฒนาแพลตฟอร์มวิศวกรรมความรู้เชิงสัมพันธ์การปรากฏร่วมที่สามารถแปลงเอกสารให้กลายเป็นกราฟเครือข่ายความรู้ได้อย่างมีประสิทธิภาพ

2.2.2 เพื่อพัฒนาระบบวิเคราะห์การปรากฏร่วมของคำและแนวคิด (Co-occurrence Analysis) ที่สามารถระบุความถี่และความสัมพันธ์ระหว่างคำศัพท์ในเอกสารได้อย่างแม่นยำ

2.2.3 เพื่อสร้างส่วนติดต่อผู้ใช้ (User Interface) ที่ช่วยให้ผู้ใช้สามารถแสดงผลและโต้ตอบกับกราฟเครือข่ายความรู้ได้อย่างง่ายดายและเข้าใจง่าย

2.2.4 เพื่อพัฒนาฟีเจอร์การจัดการส่วนของกราฟ (Graph Management) เพื่อนำไปใช้ในการสร้างกราฟเครือข่ายใหม่หรือผสมผสานกับข้อมูลจากแหล่งอื่น

2.2.5 เพื่อพัฒนาระบบการผสมผสานความรู้จากหลายแหล่งข้อมูลเพื่อค้นหาความเชื่อมโยงและสร้างความรู้ใหม่ที่มีความเชื่อมโยงกัน

2.2.6 เพื่อพัฒนาระบบบูรณาการกับ Large Language Models (LLM) ที่สามารถใช้ข้อมูลจากกราฟเครือข่ายในการปรับปรุงความแม่นยำและประสิทธิภาพของการค้นหาและสกัดความรู้

2.3 ขอบเขตของการทำโครงงานพิเศษ (Scope of Special Project)

2.3.1 การพัฒนาระบบประมวลผลและวิเคราะห์เอกสาร เช่น PDF ที่สามารถดึงข้อความ วิเคราะห์โครงสร้างและแยกแยะเนื้อหาสำคัญจากเอกสารได้อย่างมีประสิทธิภาพ รวมถึงการจัดการกับรูปแบบการจัดวางข้อความและภาษาที่หลากหลาย

2.3.2 การพัฒนาระบบวิเคราะห์การปรากฏร่วม (Co-occurrence Analysis) ที่สามารถ

- 2.3.2.1 วิเคราะห์ความถี่ของคำและวลีในเอกสาร
- 2.3.2.2 ระบุความสัมพันธ์และการปรากฏร่วมของแนวคิดต่าง ๆ
- 2.3.2.3 คำนวณค่าความแข็งแกร่งของความเชื่อมโยงระหว่างคำหรือแนวคิด
- 2.3.2.4 สร้างเมทริกซ์ความสัมพันธ์สำหรับการสร้างกราฟเครือข่าย

2.3.3 การพัฒนาระบบสร้างและจัดการกราฟเครือข่ายความรู้ (Knowledge Network Graph) ที่สามารถ

- 2.3.3.1 แปลงข้อมูลการวิเคราะห์ให้เป็นโครงสร้างกราฟ
- 2.3.3.2 จัดกลุ่มและจัดระเบียบโหนดและขอบเชื่อมตามความสัมพันธ์
- 2.3.3.3 คำนวณคุณสมบัติของกราฟ เช่น ความหนาแน่น ความเป็นศูนย์กลาง
- 2.3.3.4 สนับสนุนการแสดงผลแบบ Interactive และ Dynamic

2.3.4 การพัฒนาส่วนติดต่อผู้ใช้ (User Interface) ที่มีคุณสมบัติ

- 2.3.4.1 อัปโหลดและจัดการไฟล์ PDF
- 2.3.4.2 แสดงผลกราฟเครือข่ายแบบโต้ตอบได้
- 2.3.4.3 เครื่องมือการค้นหาและกรองข้อมูล

2.3.4.4 ส่งออกผลลัพธ์ในรูปแบบต่าง ๆ (รูปภาพ, JSON, CSV)

2.3.5 การพัฒนาฟีเจอร์การสกัดและจัดการส่วนของกราฟ (Graph Management) ที่สามารถ

2.3.5.1 เลือกและสกัดส่วนที่สนใจจากกราฟใหญ่

2.3.5.2 บันทึกและจัดเก็บส่วนกราฟที่สกัดไว้

2.3.5.3 ผสมผสานกราฟจากหลายแหล่งข้อมูล

2.3.5.4 สร้างกราฟใหม่จากการรวมข้อมูลหลายเอกสาร

2.3.6 การพัฒนาระบบฐานข้อมูลสำหรับจัดเก็บข้อมูล

2.3.6.1 ข้อมูลเอกสารและเมตาดาต้า

2.3.6.2 ผลการวิเคราะห์และกราฟเครือข่าย

2.3.6.3 ประวัติการทำงานและการแก้ไข

2.3.6.4 การตั้งค่าและ Preferences ของผู้ใช้

2.3.7 การพัฒนาระบบบูรณาการกับ Large Language Models (LLM) ที่สามารถ

2.3.7.1 แปลงข้อมูล Network Graph ให้เป็นรูปแบบที่ LLM สามารถเข้าใจและประมวลผลได้

2.3.7.2 สร้างระบบ Query Interface ที่ช่วยให้ผู้ใช้สามารถสอบถามข้อมูลจากกราฟผ่าน LLM ได้

2.3.7.3 พัฒนา Context-aware Search ที่ใช้ความรู้จากกราฟเครือข่ายในการปรับปรุงความแม่นยำของการค้นหา

2.3.7.4 สร้างระบบ Knowledge Discovery ที่ใช้ LLM ในการวิเคราะห์และสกัดความรู้ใหม่จากความสัมพันธ์ในกราฟ

2.4 รายละเอียดของทฤษฎีที่ใช้ในการจัดทำปริญญานิพนธ์

2.4.1 สมมติฐาน หรือ ข้อตกลงเบื้องต้นในการจัดทำโครงงานพิเศษ (Assumption of the Study)

2.4.1.1 เอกสารที่นำเข้าสู่ระบบจะอยู่ในรูปแบบ PDF เป็นต้น ฯลฯ ที่มีข้อความที่สามารถสกัดได้ (Text-extractable) และ มี คุณภาพ เพียง พอ สำหรับ การ ประมวล ผล ด้วย เทคนิค Optical Character Recognition (OCR) ในกรณีที่จำเป็น

2.4.1.2 เอกสารที่ใช้ในการวิเคราะห์จะเป็นเอกสารทางวิชาการ งานวิจัย หรือเอกสารที่มีโครงสร้างและเนื้อหาที่ชัดเจน โดยมีการใช้คำศัพท์และแนวคิดที่สามารถระบุและวิเคราะห์ความสัมพันธ์ได้

2.4.1.3 ระบบจะทำงานภายใต้สมมติฐานที่ว่าการปรากฏร่วมของคำหรือแนวคิดในระยะทางที่ใกล้กันภายในเอกสารแสดงถึงความสัมพันธ์หรือความเชื่อมโยงระหว่างแนวคิดเหล่านั้น

2.4.1.4 ผู้ใช้งานระบบจะมีความรู้พื้นฐานในการตีความและวิเคราะห์กราฟเครือข่าย รวมถึงสามารถระบุความสำคัญและความหมายของความสัมพันธ์ที่แสดงในกราฟได้

2.4.1.5 ระบบจะมีประสิทธิภาพในการประมวลผลเอกสารที่มีขนาดปานกลางถึงใหญ่ โดยสมมติว่าทรัพยากรคอมพิวเตอร์ที่ใช้งานมีความสามารถเพียงพอสำหรับการประมวลผลและการแสดงผลกราฟเครือข่ายแบบโต้ตอบได้

2.4.1.6 การบูรณาการกับ Large Language Models (LLM) จะช่วยปรับปรุงความแม่นยำในการระบุและจัดกลุ่มแนวคิด โดยสมมติว่า LLM จะสามารถเข้าใจบริบทและความหมายของข้อความในเอกสารได้อย่างถูกต้อง

2.4.1.7 ผลลัพธ์ที่ได้จากระบบจะมีความเชื่อถือได้และสามารถนำไปใช้ในการตัดสินใจหรือการวิจัยเพิ่มเติมได้ โดยระบบจะมีกลไกในการตรวจสอบและปรับปรุงความแม่นยำของผลการวิเคราะห์

2.4.1.8 การผสมผสานข้อมูลจากหลายแหล่งจะช่วยสร้างความรู้ใหม่ที่มีคุณค่า โดยสมมติว่าข้อมูลจากแหล่งต่าง ๆ จะมีความเข้ากันได้และสามารถรวมเข้าด้วยกันได้อย่างมีความหมาย

2.4.2 คำจำกัดความ (Key Word)

2.4.2.1 **การปรากฏร่วม (Co-occurrence)** หมายถึง การที่คำหรือแนวคิดสองคำหรือมากกว่าปรากฏในตำแหน่งที่ใกล้เคียงกันภายในเอกสารหรือข้อความ ซึ่งแสดงถึงความสัมพันธ์หรือความเชื่อมโยงระหว่างแนวคิดเหล่านั้น

2.4.2.2 **วิศวกรรมความรู้ (Knowledge Engineering)** หมายถึง กระบวนการในการสกัด จัดระเบียบ จัดเก็บ และจัดการความรู้จากแหล่งข้อมูลต่าง ๆ เพื่อให้สามารถนำไปใช้งานได้อย่างมีประสิทธิภาพ

2.4.2.3 **แพลตฟอร์ม (Platform)** หมายถึง ระบบซอฟต์แวร์ที่ให้บริการและเครื่องมือครบครันสำหรับการดำเนินงานเฉพาะด้าน ในที่นี้คือการวิเคราะห์และจัดการความรู้เชิงสัมพันธ์

2.4.2.4 **กราฟเครือข่าย (Network Graph)** หมายถึง โครงสร้างข้อมูลที่ประกอบด้วยโหนด (Nodes) และขอบเชื่อม (Edges) ที่แสดงความสัมพันธ์ระหว่างแนวคิดหรือเอนทิตีต่าง ๆ ในรูปแบบที่เข้าใจได้ง่าย

2.4.2.5 **การวิเคราะห์การปรากฏร่วม (Co-occurrence Analysis)** หมายถึง เทคนิคการวิเคราะห์ข้อมูลที่ใช้ในการระบุและวัดความถี่ของการปรากฏร่วมของคำหรือแนวคิดในข้อความ เพื่อค้นหาความสัมพันธ์และรูปแบบต่าง ๆ

2.4.2.6 **โหนด (Node)** หมายถึง จุดหรือองค์ประกอบพื้นฐานในกราฟเครือข่ายที่แทนคำ แนวคิด หรือเอนทิตีต่าง ๆ ที่ได้จากการวิเคราะห์เอกสาร

2.4.2.7 **ขอบเชื่อม (Edge)** หมายถึง เส้นเชื่อมระหว่างโหนดในกราฟเครือข่ายที่แสดงความสัมพันธ์หรือความเชื่อมโยงระหว่างแนวคิดหรือเอนทิตีต่าง ๆ รวมถึงค่าน้ำหนักที่บ่งบอกถึงความแข็งแกร่งของความสัมพันธ์

2.4.2.8 **การสกัดข้อความ (Text Extraction)** หมายถึง กระบวนการในการดึงข้อความจากเอกสารดิจิทัล เช่น ไฟล์ PDF โดยใช้เทคนิคการประมวลผลเอกสาร

2.4.2.9 **เมทริกซ์ความสัมพันธ์ (Relationship Matrix)** หมายถึง ตารางสองมิติที่แสดงค่าความแข็งแกร่งของความสัมพันธ์ระหว่างคำหรือแนวคิดต่าง ๆ ที่ได้จากการวิเคราะห์การปรากฏร่วม

2.4.2.10 **Large Language Models (LLM)** หมายถึง โมเดลปัญญาประดิษฐ์ที่ได้รับการฝึกฝนด้วยข้อมูลข้อความขนาดใหญ่ เพื่อให้สามารถเข้าใจและประมวลผลภาษาธรรมชาติได้อย่างมีประสิทธิภาพ

2.4.2.11 **การแสดงผลแบบโต้ตอบ (Interactive Visualization)** หมายถึง การแสดงผลข้อมูลในรูปแบบที่ผู้ใช้สามารถโต้ตอบและปรับเปลี่ยนมุมมองหรือรายละเอียดได้ตามต้องการ

2.4.2.12 **การจัดการส่วนของกราฟ (Graph Management)** หมายถึง ระบบที่ช่วยในการเลือก สกัด บันทึก และจัดการส่วนต่าง ๆ ของกราฟเครือข่ายเพื่อนำไปใช้งานหรือวิเคราะห์เพิ่มเติม

2.4.2.13 **การค้นพบความรู้ (Knowledge Discovery)** หมายถึง กระบวนการในการค้นหาและระบุรูปแบบความสัมพันธ์ หรือความรู้ใหม่ที่ซ่อนอยู่ในข้อมูลขนาดใหญ่ผ่านเทคนิคการวิเคราะห์ข้อมูลต่าง ๆ

2.4.3 รายงานการค้นคว้า การศึกษา หรือการวิจัยที่เกี่ยวข้อง

2.4.3.1 **Centroid Terms as Text Representatives**

งาน วิจัย ของ Mario M. Kubek, Herwig Unger (2016) เรื่อง "Centroid Terms as Text Representatives" ได้ศึกษาเทคนิคการสร้างกราฟความรู้จากข้อความ โดยเฉพาะการใช้เทคนิค Co-occurrence Analysis ในการระบุความสัมพันธ์ระหว่างเอนทิตีต่าง ๆ การศึกษานี้ได้ชี้ให้เห็นว่าอัลกอริทึมสำหรับการจัดกลุ่มและการจำแนกข้อความตามหัวข้อนั้นอาศัยข้อมูลเกี่ยวกับระยะทางและความคล้ายคลึงเชิงความหมาย วิธีการมานตรฐานที่อิงตามแบบจำลอง bag-of-words ในการกำหนดข้อมูลนี้จะให้เพียงการประมาณแบบคร่าว ๆ เกี่ยวกับความเกี่ยวข้องของข้อความ นอกจากนี้วิธีการเหล่านี้ยังไม่สามารถค้นหาคำศัพท์ที่เป็นนามธรรมหรือคำที่สามารถอธิบายเนื้อหาของข้อความได้อย่างครอบคลุม งานวิจัยนี้จึงได้นำเสนอวิธีการใหม่ในการกำหนดคำศูนย์กลาง (Centroid Terms) ในข้อความและการประเมินความคล้ายคลึงโดยใช้คำที่เป็นตัวแทนเหล่านั้น ซึ่งแสดงให้เห็นว่าการวิเคราะห์การปรากฏร่วมสามารถช่วยในการค้นพบรูปแบบความสัมพันธ์ที่ซ่อนอยู่ในข้อมูลขนาดใหญ่ได้อย่างมีประสิทธิภาพ และสามารถนำไปประยุกต์ใช้ในการพัฒนาระบบวิศวกรรมความรู้เชิงสัมพันธ์ได้

2.4.3.2 **Spreading activation: a fast calculation method for text centroids**

งาน วิจัย ของ Mario M. Kubek, Thomas Böhme, Herwig Unger (2017) เรื่อง "Spreading activation: a fast calculation method for text centroids" ได้นำเสนอเทคนิคการคำนวณ Centroid Terms แบบใหม่ที่มีประสิทธิภาพสูง โดยใช้ Spreading Activation Algorithm ซึ่งเป็นเทคนิคที่ทำงานบนพื้นฐานของกราฟและหลักการทำงานแบบเฉพาะที่ (Local Working Principle) การศึกษานี้ชี้ให้เห็นว่า Centroids เป็นเครื่องมือที่สะดวกในการแสดงคำค้นหาและข้อความทั้งหมดด้วยคำศัพท์เชิงบรรยายเพียงคำเดียว ซึ่งสามารถนำไปใช้ในการกำหนดความคล้ายคลึงของเนื้อหาข้อความและการจัดกลุ่มเอกสารแบบลำดับชั้น อย่างไรก็ตาม การคำนวณตามคำจำกัดความแบบดั้งเดิมอาจใช้เวลามากและเป็นอุปสรรคต่อการนำไปใช้งานจริง ดังนั้น การพัฒนาอัลกอริทึมแบบใหม่ที่อิงตามกราฟ Co-occurrence จึงมีความสำคัญต่อการเพิ่มประสิทธิภาพการประมวลผล ซึ่งสอดคล้องกับแนวทางที่จะใช้ในโครงงานนี้สำหรับการวิเคราะห์การปรากฏร่วมและการสร้างกราฟเครือข่ายความรู้ที่มีประสิทธิภาพในการประมวลผลเอกสารขนาดใหญ่

2.4.3.3 **Enhancing Retrieval-Augmented Generation Systems by Text-Representing Centroid**

การ ศึกษา ของ Yanakorn Ruamsuk, Anirach Mingkhawn, Herwig Unger (2025) เรื่อง "Enhancing Retrieval-Augmented Generation Systems by Text-Representing Centroid" ได้นำเสนอแนวทางใหม่ในการปรับปรุงระบบ Retrieval-Augmented Generation (RAG) โดยการบูรณาการเทคนิค Text-Representing Centroid (TRC) เพื่อแก้ไขข้อจำกัดของฐานข้อมูลเวกเตอร์แบบดั้งเดิม วิธีการนี้สามารถรักษาความสัมพันธ์เชิงโครงสร้างและปรับตัวตามความซับซ้อนของเนื้อหา ส่งผลให้การค้นคืนข้อมูลมีประสิทธิภาพและความแม่นยำที่สูงขึ้น การมีส่วนสนับสนุนที่สำคัญได้แก่ การสร้างกราฟขั้นสูง อัลกอริทึมการให้คะแนนความเกี่ยวข้อง และการตรวจสอบความถูกต้องอย่างครอบคลุม พร้อมการอภิปรายเกี่ยวกับการประยุกต์ใช้ที่เป็นไปได้และการวิจัยในอนาคต หลักฐานเชิงประจักษ์แสดงให้เห็นว่าเทคนิค TRC สามารถบรรลุอัตราความสำเร็จ 75 เปอร์เซ็นต์จากคำถาม 100 ข้อ ซึ่งมีประสิทธิภาพเหนือกว่าวิธีการเวกเตอร์แบบดั้งเดิม การศึกษานี้มีความเกี่ยวข้องโดยตรงกับโครงงานที่เสนอ เนื่องจากแสดงให้เห็นถึงความเป็นไปได้ในการใช้เทคนิค Co-occurrence Analysis และ Centroid-based Methods ในการพัฒนาระบบวิศวกรรมความรู้ที่มีประสิทธิภาพสูง

2.4.4 เนื้อหา เหตุผล และทฤษฎีที่สำคัญ

โครงงาน Co-Occurrence Knowledge Engineering Platform นี้มีพื้นฐานทางทฤษฎีที่แข็งแกร่งและเหตุผลที่ชัดเจนในการพัฒนา โดยอาศัยหลักการทางวิศวกรรมความรู้และเทคนิคการวิเคราะห์ข้อมูลขั้นสูงหลายแนวทาง

เหตุผลในการพัฒนาโครงงาน

ในยุคสารสนเทศปัจจุบัน ปริมาณข้อมูลและเอกสารดิจิทัลเพิ่มขึ้นอย่างรวดเร็ว แต่การจัดการและการค้นหาความเชื่อมโยงระหว่างความรู้จากแหล่งต่าง ๆ ยังคงเป็นความท้าทายสำคัญ ผู้ใช้งานส่วนใหญ่ไม่สามารถมองเห็นภาพรวมของความสัมพันธ์ระหว่างแนวคิดที่ปรากฏในเอกสารหลายฉบับได้อย่างชัดเจน การวิเคราะห์การปรากฏร่วม (Co-occurrence Analysis) แบบดั้งเดิมใช้เวลามากและซับซ้อน ดังนั้น จึงจำเป็นต้องมีระบบที่สามารถแปลงเอกสารให้เป็นกราฟเครือข่ายความรู้ที่เข้าใจได้ง่าย และสามารถผสมผสานข้อมูลจากหลายแหล่งเพื่อสร้างความรู้ใหม่

ทฤษฎีพื้นฐานที่ใช้ในการพัฒนา

1. ทฤษฎีการปรากฏร่วม (Co-occurrence Theory)

หลักการพื้นฐานของการวิเคราะห์การปรากฏร่วมอิงตามสมมติฐานที่ว่า คำหรือแนวคิดที่ปรากฏใกล้กันในข้อความมักจะมีความสัมพันธ์หรือความเชื่อมโยงกัน ทฤษฎีนี้ได้รับการสนับสนุนจากงานวิจัยของ Mario M. Kubek et al. (2016, 2017) ที่แสดงให้เห็นว่าการวิเคราะห์ Centroid Terms และ Spreading Activation Algorithm สามารถช่วยในการระบุความสัมพันธ์เชิงความหมายได้อย่างมีประสิทธิภาพ

2. ทฤษฎีกราฟและเครือข่าย (Graph Theory and Network Theory)

การแสดงความรู้ในรูปแบบกราฟเครือข่ายอิงตามทฤษฎีกราฟ ซึ่งประกอบด้วยโหนด (Nodes) แทนแนวคิดหรือเอนทิตี และขอบเชื่อม (Edges) แทนความสัมพันธ์ ทฤษฎีนี้ช่วยให้สามารถคำนวณคุณสมบัติต่าง ๆ ของเครือข่าย เช่น ความหนาแน่น (Density) ความเป็นศูนย์กลาง (Centrality) และการจัดกลุ่ม (Clustering) ซึ่งมีความสำคัญต่อการวิเคราะห์และการค้นพบความรู้

3. ทฤษฎีการประมวลผลภาษาธรรมชาติ (Natural Language Processing Theory)

การสกัดข้อความและการวิเคราะห์เนื้อหาจากเอกสาร PDF อาศัยหลักการของ NLP รวมถึงเทคนิค Tokenization, Part-of-Speech Tagging, Named Entity Recognition และ Semantic Analysis เพื่อให้สามารถระบุและแยกแยะแนวคิดสำคัญได้อย่างแม่นยำ

4. ทฤษฎีการเรียนรู้ของเครื่อง (Machine Learning และ Deep Learning Theory)

การบูรณาการกับ Large Language Models (LLM) อาศัยหลักการของ Deep Learning และ Transformer Architecture เพื่อปรับปรุงความแม่นยำในการระบุความสัมพันธ์และการทำ Semantic Reasoning งานวิจัยของ Yanakorn Ruamsuk et al. (2025) แสดงให้เห็นว่าการใช้ Text-Representing Centroid ร่วมกับ LLM สามารถบรรลุอัตราความสำเร็จ 75 เปอร์เซ็นต์ในการตอบคำถาม

5. ทฤษฎีการจัดการฐานข้อมูล (Database Management Theory)

การจัดเก็บและจัดการข้อมูลกราฟอาศัยหลักการของ Graph Database และ NoSQL Database เพื่อรองรับการประมวลผลข้อมูลเชิงสัมพันธ์ที่ซับซ้อนและการ Query แบบ Real-time

2.5 วิธีดำเนินการจัดทำโครงงานพิเศษ

การพัฒนาแพลตฟอร์ม Co-Occurrence Knowledge Engineering Platform จะดำเนินการโดยใช้แนวทางการพัฒนาระบบแบบครบวงจร (Full-Stack Development) ร่วมกับเทคโนโลยีคลาวด์และเครื่องมือออกแบบสมัยใหม่ เพื่อให้ได้ระบบที่มีประสิทธิภาพ ปลอดภัย และใช้งานง่าย

วิธีการดำเนินการหลัก

2.5.1 **การพัฒนาระบบบนคลาวด์เซิร์ฟเวอร์**

- 2.5.1.1 ใช้คลาวด์เซิร์ฟเวอร์เป็นสภาพแวดล้อมหลักในการ Host โครงงาน
- 2.5.1.2 ติดตั้งและกำหนดค่าระบบปฏิบัติการ Linux (Server) สำหรับการประมวลผล
- 2.5.1.3 ปรับแต่งสภาพแวดล้อมสำหรับการพัฒนา TypeScript, Go และฐานข้อมูล

2.5.2 **การออกแบบ User Interface และ User Experience**

- 2.5.2.1 ใช้เครื่องมือ Figma ในการออกแบบ Wireframes และ Mockups ทั้งหมด
- 2.5.2.2 สร้าง Design System ที่สอดคล้องกับการใช้งานของ Knowledge Engineering Platform
- 2.5.2.3 ออกแบบ Interactive Prototypes สำหรับการแสดงผลกราฟเครือข่าย
- 2.5.2.4 ทดสอบ Usability และปรับปรุงการออกแบบตามผลการทดสอบ
- 2.5.2.5 สร้าง Responsive Design เพื่อรองรับการใช้งานบนอุปกรณ์ต่าง ๆ

2.5.3 **การพัฒนาระบบจำลองและการทดสอบ**

- 2.5.3.1 สร้างระบบจำลองการวิเคราะห์ Co-occurrence ด้วยข้อมูลตัวอย่าง
- 2.5.3.2 พัฒนา Proof of Concept สำหรับการสร้างกราฟเครือข่ายจากเอกสาร PDF
- 2.5.3.3 สร้างและทดสอบการบูรณาการกับ Large Language Models (LLM)

2.5.4 **การสร้างเอกสารและข้อกำหนดระบบ**

- 2.5.4.1 จัดทำเอกสาร System Requirements และ Functional Specifications
- 2.5.4.2 สร้าง API Documentation สำหรับการใช้งานระบบ
- 2.5.4.3 เขียน User Manual และ Administrator Guide
- 2.5.4.4 จัดทำเอกสาร Technical Architecture และ Database Schema
- 2.5.4.5 สร้าง Test Cases และ Test Plans สำหรับการทดสอบระบบ

2.5.5 **การสร้างสภาพแวดล้อมความปลอดภัย**

- 2.5.5.1 ติดตั้งและกำหนดค่า VPN Server สำหรับการเชื่อมต่อที่ปลอดภัย
- 2.5.5.2 สร้าง Private Network สำหรับการเข้าถึงฐานข้อมูลและทรัพยากรภายใน
- 2.5.5.3 ปรับแต่งระบบ Firewall และ Security Groups สำหรับการควบคุมการเข้าถึง
- 2.5.5.4 ใช้ SSL/TLS Certificates สำหรับการเข้ารหัสข้อมูล

2.5.6 **การพัฒนาและการทดสอบระบบ**

- 2.5.6.1 สร้าง CI/CD Pipeline สำหรับการ Deploy และ Testing อัตโนมัติ
- 2.5.6.2 ทดสอบระบบด้วย Unit Testing, Integration Testing และ End-to-End Testing
- 2.5.6.3 ประเมินประสิทธิภาพระบบด้วย Performance Testing และ Load Testing
- 2.5.6.4 ทดสอบความปลอดภัยด้วย Security Testing และ Penetration Testing

สถานที่ดำเนินการ

2.5.7 **สถานที่หลักในการพัฒนา**

2.5.7.1 ห้องปฏิบัติการคอมพิวเตอร์ ภาควิชาเทคโนโลยีสารสนเทศ คณะเทคโนโลยีและการจัดการอุตสาหกรรม สำหรับการพัฒนาและทดสอบระบบ

2.5.7.2 ห้องส่วนตัวที่มีการเชื่อมต่ออินเทอร์เน็ตความเร็วสูง สำหรับการพัฒนาและการออกแบบ

2.5.7.3 คลาวด์เซิร์ฟเวอร์ สำหรับการ Host และ Deploy ระบบ

2.5.8 **สภาพแวดล้อมการทำงาน**

2.5.8.1 ระบบพัฒนาบนเครื่อง Local Development Environment (macOS/Linux)

2.5.8.2 ระบบทดสอบบน Staging Environment ในคลาวด์เซิร์ฟเวอร์

2.5.8.3 ระบบจริงบน Production Environment ที่มีการรักษาความปลอดภัยสูง

2.5.8.4 ระบบการออกแบบบน Figma Cloud Platform

2.6 แผนกิจกรรมและตารางเวลาในการจัดทำ

2.6.1 แผนกิจกรรมหลักและระยะเวลา

ตารางที่ 1: แผนการดำเนินงานภาคการศึกษาที่ 1

ขั้นตอนการดำเนินการ	ก.ค.				ส.ค.				ก.ย.				ต.ค.			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1) วางแผนการพัฒนา																
2) เก็บ Requirement																
3) ออกแบบ UI และ UX																
4) ออกแบบสถาปัตยกรรมซอฟต์แวร์																
5) พัฒนาระบบส่วนแกนหลัก																

ตารางที่ 2: แผนการดำเนินงานภาคการศึกษาที่ 2

ขั้นตอนการดำเนินการ	ธ.ค.				ม.ค.				ก.พ.				มี.ค.			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
6) ทดสอบระบบส่วนแกนหลัก																
7) ทดสอบความปลอดภัยของระบบ																
8) ทดสอบสภาพแวดล้อมจริง																
9) จัดทำเอกสารแพลตฟอร์ม																
10) ส่งมอบแพลตฟอร์ม																

2.6.2 แผนภูมิขั้นตอนการจัดทำโครงงานพิเศษ โดยละเอียด

การพัฒนาแพลตฟอร์ม Co-Occurrence Knowledge Engineering Platform จะดำเนินการตามผังงานที่มีการแบ่งขั้นตอนและกิจกรรมอย่างชัดเจน พร้อมทั้งกำหนดอัตราส่วนของแต่ละกิจกรรมเป็นร้อยละเพื่อการติดตามความก้าวหน้าอย่างมีประสิทธิภาพ

ผังงานการดำเนินงาน (Development Flow Chart)

Phase 1: การวางแผนและวิเคราะห์ (Planning & Analysis Phase) - 15% ของโครงงานทั้งหมด

1.1 **การวางแผนการพัฒนา (5%)**

- 1.1.1 กำหนดขอบเขตโครงงานและวัตถุประสงค์
- 1.1.2 วิเคราะห์ความเป็นไปได้ทางเทคนิค (Technical Feasibility)
- 1.1.3 จัดทำ Project Charter และ Timeline
- 1.1.4 กำหนดทรัพยากรและงบประมาณ

1.2 **การเก็บความต้องการ (Requirements Gathering) (10%)**

- 1.2.1 วิเคราะห์ Functional Requirements
- 1.2.2 วิเคราะห์ Non-Functional Requirements
- 1.2.3 สร้าง Use Case Diagrams และ User Stories
- 1.2.4 กำหนด Acceptance Criteria สำหรับแต่ละฟีเจอร์
- 1.2.5 วิเคราะห์ข้อจำกัดและความเสี่ยง

Phase 2: การออกแบบระบบ (System Design Phase) - 25% ของโครงงานทั้งหมด

2.1 **การออกแบบ UI และ UX (12%)**

- 2.1.1 สร้าง User Persona และ User Journey Mapping
- 2.1.2 ออกแบบ Wireframes และ Mockups ด้วย Figma
- 2.1.3 สร้าง Interactive Prototypes
- 2.1.4 ทดสอบ Usability และปรับปรุงการออกแบบ
- 2.1.5 สร้าง Design System และ Component Library

2.2 **การออกแบบสถาปัตยกรรมซอฟต์แวร์ (13%)**

- 2.2.1 ออกแบบ System Architecture และ Component Diagram
- 2.2.2 ออกแบบ Database Schema และ Data Model
- 2.2.3 ออกแบบ API Architecture และ Endpoints
- 2.2.4 ออกแบบ Security Architecture และ Authentication
- 2.2.5 ออกแบบ Deployment Architecture และ Infrastructure
- 2.2.6 สร้าง Technical Specifications Document

Phase 3: การพัฒนาระบบ (Development Phase) - 40% ของโครงงานทั้งหมด

3.1 **การพัฒนาระบบส่วนแกนหลัก (Core System Development) (40%)**

3.1.1 **Backend Development (20%)**

3.1.1.1 สร้าง API สำหรับการอัปโหลดและประมวลผล PDF (4%)

3.1.1.2 พัฒนาระบบ Co-occurrence Analysis Engine (6%)

3.1.1.3 สร้างระบบ Graph Generation และ Management (5%)

3.1.1.4 พัฒนาระบบ Database และ Data Storage (3%)

3.1.1.5 บูรณาการกับ Large Language Models (LLM) (2%)

3.1.2 **Frontend Development (15%)**

3.1.2.1 สร้าง Dashboard และ Main Interface (4%)

3.1.2.2 พัฒนา File Upload และ Management System (3%)

3.1.2.3 สร้าง Interactive Graph Visualization (5%)

3.1.2.4 พัฒนา Search และ Filter Components (2%)

3.1.2.5 สร้าง Export และ Share Features (1%)

3.1.3 **Integration และ API Development (5%)**

3.1.3.1 Integration Testing ระหว่าง Frontend และ Backend (2%)

3.1.3.2 สร้าง API Documentation (1%)

3.1.3.3 พัฒนา Error Handling และ Logging System (1%)

3.1.3.4 Optimization และ Performance Tuning (1%)

Phase 4: การทดสอบระบบ (Testing Phase) - 15% ของโครงงานทั้งหมด

4.1 **การทดสอบระบบส่วนแกนหลัก (8%)**

4.1.1 Unit Testing สำหรับแต่ละ Component (2%)

4.1.2 Integration Testing สำหรับระบบทั้งหมด (3%)

4.1.3 Performance Testing และ Load Testing (2%)

4.1.4 Functional Testing และ User Acceptance Testing (1%)

4.2 **การทดสอบความปลอดภัยของระบบ (4%)**

4.2.1 Security Testing และ Vulnerability Assessment (2%)

4.2.2 Authentication และ Authorization Testing (1%)

4.2.3 Data Protection และ Privacy Testing (1%)

4.3 **การทดสอบสภาพแวดล้อมจริง (3%)**

4.3.1 Deployment Testing บน Production Environment (1%)

4.3.2 End-to-End Testing กับข้อมูลจริง (1%)

4.3.3 User Experience Testing และ Feedback Collection (1%)

Phase 5: การจัดทำเอกสารและส่งมอบ (Documentation & Delivery Phase) - 5% ของโครงงานทั้งหมด

5.1 **การจัดทำเอกสารแพลตฟอร์ม (3%)**

5.1.1 เขียน User Manual และ Administrator Guide (1%)

5.1.2 สร้าง API Documentation และ Technical Guide (1%)

5.1.3 จัดทำ Installation Guide และ Troubleshooting (0.5%)

5.1.4 สร้าง Video Tutorial และ Demo Materials (0.5%)

5.2 **การส่งมอบแพลตฟอร์ม (2%)**

5.2.1 Final System Testing และ Quality Assurance (0.5%)

5.2.2 Knowledge Transfer และ Training Session (0.5%)

5.2.3 Project Handover และ Final Presentation (0.5%)

5.2.4 Post-Implementation Support Planning (0.5%)

การติดตามความก้าวหน้า (Progress Monitoring)

การประเมินความก้าวหน้าจะดำเนินการโดยใช้ระบบ Milestone-based Tracking ซึ่งแบ่งการประเมินออกเป็น:

5.3 **Weekly Progress Review (รายสัปดาห์)**

5.3.1 ตรวจสอบความก้าวหน้าตาม Timeline ที่กำหนด

5.3.2 ประเมินคุณภาพของงานที่ส่งมอบในแต่ละ Phase

5.3.3 ระบุและแก้ไข Issues หรือ Blockers ที่เกิดขึ้น

5.3.4 ปรับปรุง Plan หากจำเป็น

5.4 **Phase Gate Reviews (รายเฟส)**

5.4.1 การประเมินผลลัพธ์ที่ได้จากแต่ละ Phase

5.4.2 การตรวจสอบคุณภาพตาม Acceptance Criteria

5.4.3 การอนุมัติให้ดำเนินการใน Phase ถัดไป

5.4.4 การจัดทำ Lessons Learned สำหรับปรับปรุงในอนาคต

สรุปการแบ่งสัดส่วนงาน

- การวางแผนและวิเคราะห์: 15%
- การออกแบบระบบ: 25%
- การพัฒนาระบบ: 40%
- การทดสอบระบบ: 15%
- การจัดทำเอกสารและส่งมอบ: 5%

2.7 ทรัพยากรที่ต้องใช้ในการจัดทำโครงงานพิเศษ

การพัฒนาแพลตฟอร์ม Co-Occurrence Knowledge Engineering Platform ต้องใช้ทรัพยากรหลากหลายประเภทเพื่อให้การดำเนินงานมีประสิทธิภาพและบรรลุวัตถุประสงค์ที่กำหนดไว้ โดยทรัพยากรเหล่านี้จะแบ่งออกเป็นหมวดหมู่ต่าง ๆ ตามลักษณะการใช้งานและความสำคัญต่อโครงงาน

ทรัพยากรหลัก (Primary Resources)

2.7.1 ทรัพยากรด้านฮาร์ดแวร์และโครงสร้างพื้นฐาน

2.7.1.1 เครื่องคอมพิวเตอร์สำหรับการพัฒนา

- 2.7.1.1.1 MacBook (CPU: Apple M2, RAM: 16GB ขึ้นไป, SSD: 256GB ขึ้นไป)
- 2.7.1.1.2 จอภาพเพิ่มเติมขนาด 24-27 นิ้ว สำหรับการแสดงผลและออกแบบ
- 2.7.1.1.3 อุปกรณ์รับเสียงและไมโครโฟนสำหรับการจัดทำวิดีโอการนำเสนอ

2.7.1.2 คลาวด์เซิร์ฟเวอร์และบริการ

- 2.7.1.2.1 Virtual Private Server (VPS) หรือ Cloud Instance สำหรับ Development และ Testing Environment
- 2.7.1.2.2 Production Server พร้อม Load Balancer สำหรับการใช้งานจริง
- 2.7.1.2.3 Content Delivery Network (CDN) สำหรับการกระจายข้อมูลและเพิ่มประสิทธิภาพ
- 2.7.1.2.4 SSL/TLS Certificates สำหรับการรักษาความปลอดภัย

2.7.1.3 ระบบฐานข้อมูลและการจัดเก็บข้อมูล

- 2.7.1.3.1 Graph Database (เช่น Neo4j, ArangoDB) สำหรับการจัดเก็บกราฟเครือข่าย
- 2.7.1.3.2 Document Database (MongoDB) สำหรับการจัดเก็บเอกสารและเมตาดาต้า
- 2.7.1.3.3 File Storage System (MinIO) สำหรับการจัดเก็บไฟล์ PDF และผลลัพธ์ที่ส่งออก
- 2.7.1.3.4 Cache Database (Redis) สำหรับการเพิ่มประสิทธิภาพการประมวลผล

2.7.2 ทรัพยากรด้านซอฟต์แวร์และเครื่องมือพัฒนา

2.7.2.1 เครื่องมือการพัฒนาและ IDE

- 2.7.2.1.1 Visual Studio Code สำหรับการเขียนโปรแกรม
- 2.7.2.1.2 Git และ GitHub สำหรับการจัดการเวอร์ชันโค้ด
- 2.7.2.1.3 Docker และ Docker Compose สำหรับการจัดการ Container
- 2.7.2.1.4 Kubernetes หรือ Docker Swarm สำหรับการ Orchestration

2.7.2.2 เครื่องมือการออกแบบและ UI/UX

- 2.7.2.2.1 Figma สำหรับการออกแบบ Interface, Flowchart และ Prototyping
- 2.7.2.2.2 Excalidraw สำหรับการสร้าง Diagram ทั่วไป

2.7.2.3 เครื่องมือการทดสอบและการจัดการคุณภาพ

- 2.7.2.3.1 Trivy สำหรับการวิเคราะห์ความปลอดภัยของ Container
- 2.7.2.3.2 Postman สำหรับการทดสอบ API
- 2.7.2.3.3 Performance Testing Tools (k6)

2.7.3 **ทรัพยากรด้านเทคโนโลยีและภาษาโปรแกรม**

2.7.3.1 **เทคโนโลยี Backend Development**

2.7.3.1.1 Go (Golang) สำหรับการพัฒนา High-Performance Backend Services

2.7.3.1.2 GraphQL สำหรับการสร้าง Flexible API

2.7.3.1.3 RESTful API Frameworks (Fiber for Go)

2.7.3.1.4 Langchain สำหรับการจัดการ LLM เช่น Context, Memory

2.7.3.2 **เทคโนโลยี Frontend Development**

2.7.3.2.1 React.js หรือ Next.js สำหรับการสร้าง Interactive Web Interface

2.7.3.2.2 TypeScript สำหรับการเพิ่มความปลอดภัยของโค้ด

2.7.3.2.3 D3.js หรือ Vis.js สำหรับการสร้าง Interactive Graph Visualization

2.7.3.2.4 CSS Frameworks (Tailwind CSS, Material-UI) สำหรับการออกแบบ

2.7.3.3 **เทคโนโลยี Machine Learning และ NLP**

2.7.3.3.1 Python Libraries (NumPy, Pandas, Scikit-learn) สำหรับการวิเคราะห์ข้อมูล

2.7.3.3.2 Natural Language Processing Libraries (spaCy, NLTK, Transformers)

2.7.3.3.3 PDF Processing Libraries (PyPDF2, pdfplumber, Tesseract OCR)

2.7.3.3.4 Large Language Model APIs (OpenAI GPT, Google PaLM, Claude)

2.7.3.4 **เทคโนโลยี DevOps และ Infrastructure**

2.7.3.4.1 CI/CD Tools (GitHub Actions, GitLab CI, Jenkins)

2.7.3.4.2 Infrastructure as Code (Terraform, CloudFormation)

2.7.3.4.3 Monitoring และ Logging (Prometheus, Grafana, ELK Stack)

2.7.3.4.4 Container Orchestration (Kubernetes, Docker Swarm)

2.7.4 **ทรัพยากรด้านข้อมูลและการวิจัย**

2.7.4.1 **ข้อมูลสำหรับการทดสอบและพัฒนา**

2.7.4.1.1 เอกสาร PDF ตัวอย่างจากแหล่งวิชาการต่าง ๆ สำหรับการทดสอบระบบ

2.7.4.1.2 Dataset ข้อมูลงานวิจัยและเอกสารทางวิชาการจากแหล่งเปิด

2.7.4.1.3 Sample Co-occurrence Networks สำหรับการเปรียบเทียบผลลัพธ์

2.7.4.2 **แหล่งข้อมูลและการอ้างอิง**

2.7.4.2.1 การเข้าถึงฐานข้อมูลวิชาการ (IEEE Xplore, ACM Digital Library, SpringerLink)

2.7.4.2.2 เครื่องมือการจัดการ References (Zotero, Mendeley, EndNote)

2.7.4.2.3 Online Learning Resources และ Technical Documentation

2.7.4.2.4 การสมัครสมาชิก Professional Communities และ Forums

2.7.5 **ทรัพยากรด้านบุคลากรและการสนับสนุน**

2.7.5.1 **ทีมพัฒนาและที่ปรึกษา**

2.7.5.1.1 อาจารย์ที่ปรึกษาโครงงาน

2.7.6 **ทรัพยากรด้านการเงินและงบประมาณ**

2.7.6.1 **ค่าใช้จ่ายด้านโครงสร้างพื้นฐาน**

2.7.6.1.1 ค่า Cloud Server อยู่ที่ 0 บาท (ใช้ Cloud Server ของตนเอง จึงไม่นับค่าใช้จ่าย)

2.7.6.1.2 ค่าจัดทำปริญญานิพนธ์ ประมาณ 1,000 บาท

2.8 ผลที่คาดว่าจะได้รับ

การพัฒนาแพลตฟอร์ม Co-Occurrence Knowledge Engineering Platform คาดว่าจะให้ผลลัพธ์ที่มีคุณค่าและส่งผลกระทบเชิงบวกต่อการจัดการความรู้และการวิจัยในหลายมิติ โดยผลที่คาดว่าจะได้รับสามารถแบ่งออกเป็นหมวดหมู่ตามประเภทของผู้ใช้งานและขอบเขตการประยุกต์ใช้

ผลทางเทคนิคและเทคโนโลยี (Technical and Technological Outcomes)

2.8.1 **ระบบแพลตฟอร์มที่สมบูรณ์และมีประสิทธิภาพ**

2.8.1.1 แพลตฟอร์มวิศวกรรมความรู้เชิงสัมพันธ์การปรากฏร่วมที่มีความสามารถในการประมวลผลเอกสาร PDF ขนาดใหญ่ได้อย่างรวดเร็วและแม่นยำ

2.8.1.2 กราฟเครือข่ายความรู้แบบโต้ตอบที่สามารถแสดงผลความสัมพันธ์เชิงซับซ้อนได้อย่างเข้าใจง่ายและสวยงาม

2.8.1.3 ระบบบูรณาการกับ Large Language Models (LLM) ที่ช่วยเพิ่มความแม่นยำในการค้นหาและสกัดความรู้

2.8.1.4 ส่วนติดต่อผู้ใช้ (User Interface) ที่ใช้งานง่าย เข้าใจง่าย และตอบสนองได้รวดเร็วบนอุปกรณ์ต่าง ๆ

2.8.2 **นวัตกรรมทางเทคนิคและอัลกอริทึม**

2.8.2.1 เทคนิคการสร้างกราฟเครือข่ายความรู้แบบไดนามิกที่สามารถปรับตัวตามข้อมูลและบริบทได้

2.8.2.2 ระบบการจัดการส่วนของกราฟ (Graph Management) ที่ช่วยให้สามารถผสมผสานและเปรียบเทียบความรู้จากแหล่งต่าง ๆ ได้

2.8.2.3 เทคนิคการบูรณาการ LLM กับกราฟเครือข่ายที่สามารถนำไปประยุกต์ใช้ในงานวิจัยด้านอื่น ๆ ได้

2.8.2.4 ระบบ Knowledge Discovery ที่สามารถค้นพบรูปแบบและความเชื่อมโยงใหม่ที่ไม่เคยมีมาก่อน

2.8.3 **เครื่องมือและ APIs สำหรับนักพัฒนา**

2.8.3.1 RESTful APIs ที่สมบูรณ์พร้อมเอกสารประกอบสำหรับการพัฒนาระบบต่อยอด

2.8.3.2 SDK และ Libraries สำหรับการเชื่อมต่อกับแพลตฟอร์มในภาษาโปรแกรมต่าง ๆ

2.8.3.3 เครื่องมือ Export และ Import ข้อมูลในรูปแบบมาตรฐาน (JSON, CSV, GraphML)

2.8.3.4 Template และ Examples สำหรับการใช้งานในกรณีต่าง ๆ

2.8.3.5 ระบบ Plugin Architecture ที่ช่วยให้สามารถขยายฟังก์ชันการทำงานได้ง่าย

ผลทางวิชาการและการวิจัย (Academic and Research Outcomes)

2.8.4 **การส่งเสริมการวิจัยและการเรียนรู้**

2.8.4.1 เครื่องมือที่ช่วยให้นักวิจัยสามารถวิเคราะห์ความเชื่อมโยงระหว่างงานวิจัยได้อย่างมีประสิทธิภาพ

2.8.4.2 แพลตฟอร์มที่ช่วยในการค้นพบ Research Gaps และโอกาสในการวิจัยใหม่ ๆ

2.8.4.3 ระบบที่ช่วยให้การทำ Literature Review มีประสิทธิภาพและครอบคลุมมากขึ้น

2.8.4.4 เครื่องมือสำหรับการวิเคราะห์ Trend และ Pattern ในสาขาวิชาต่าง ๆ

2.8.4.5 แพลตฟอร์ม ที่ สนับสนุน การ ทำงาน ร่วม กัน ระหว่าง นัก วิจัย จาก สาขา ต่าง ๆ (Interdisciplinary Research)

2.8.5 **การพัฒนาความรู้และองค์ความรู้ใหม่**

2.8.5.1 องค์ความรู้ใหม่เกี่ยวกับการประยุกต์ใช้ Co-occurrence Analysis ในการสร้างกราฟเครือข่าย

2.8.5.2 ความเข้าใจที่ลึกซึ้งเกี่ยวกับการบูรณาการ LLM กับระบบวิศวกรรมความรู้

2.8.5.3 รูปแบบและวิธีการใหม่ในการแสดงผลความสัมพันธ์เชิงซับซ้อนให้เข้าใจง่าย

2.8.5.4 แนวทางการพัฒนาระบบ Knowledge Management ที่มีประสิทธิภาพสูง

2.8.5.5 ต้นแบบสำหรับการพัฒนาระบบ AI-assisted Knowledge Discovery ในอนาคต

ผลทางสังคมและการประยุกต์ใช้ (Social and Application Outcomes)

2.8.6 **การเพิ่มประสิทธิภาพการทำงานและการตัดสินใจ**

2.8.7.1 การลดเวลาในการค้นหาและวิเคราะห์เอกสารสำหรับนักวิจัยและนักวิชาการลงได้ 50-70 เปอร์เซ็นต์

2.8.7.2 การเพิ่มความแม่นยำในการค้นพบความเชื่อมโยงระหว่างแนวคิดและงานวิจัย

2.8.7.3 การช่วยให้องค์กรและสถาบันการศึกษาจัดการความรู้ได้อย่างมีระบบมากขึ้น

2.8.7.4 การ สนับสนุน การ ตัดสิน ใจ เชิง นโยบาย ที่ อิง ตาม หลัก ฐาน และ ข้อมูล (Evidence-based Decision Making)

2.8.7.5 การเพิ่มโอกาสในการเกิดนวัตกรรมจากการเชื่อมโยงความรู้จากสาขาต่าง ๆ

2.8.7 **การสร้างชุมชนและเครือข่ายความร่วมมือ**

2.8.8.1 ชุมชนนักพัฒนาและผู้ใช้งานที่ร่วมกันพัฒนาและปรับปรุงแพลตฟอร์ม

2.8.8.2 เครือข่ายการแลกเปลี่ยนความรู้และประสบการณ์ระหว่างนักวิจัยจากสถาบันต่าง ๆ

2.8.8.3 การสร้างมาตรฐานและ Best Practices ในการจัดการกราฟเครือข่ายความรู้

2.8.8.4 การส่งเสริมการทำงานร่วมกันข้ามสาขาวิชาและข้ามสถาบัน

2.8.8.5 การสร้างแรงบันดาลใจให้เกิดโครงการพัฒนาเครื่องมือประเภทนี้เพิ่มเติม

2.8.8 **การประยุกต์ใช้ในหลากหลายสาขา**

2.8.9.1 การใช้งานในสาขาวิทยาศาสตร์และเทคโนโลยีสำหรับการวิเคราะห์ความก้าวหน้าทางเทคนิค

2.8.9.2 การประยุกต์ใช้ในสาขาสังคมศาสตร์สำหรับการวิเคราะห์แนวโน้มทางสังคม

2.8.9.3 การประยุกต์ใช้ในการศึกษาสำหรับการพัฒนาหลักสูตรและการเรียนการสอน

2.8.9.4 การใช้งานในหน่วยงานราชการสำหรับการวิเคราะห์นโยบายและการวางแผน

ผลกระทบระยะยาว (Long-term Impact)

2.8.12 **การพัฒนาต่อยอดและความยั่งยืน**

2.8.12.1 แพลตฟอร์มที่สามารถขยายขนาดและเพิ่มฟีเจอร์ได้อย่างต่อเนื่อง

2.8.12.2 ระบบที่มีความยืดหยุ่นสูงและสามารถปรับตัวตามเทคโนโลยีใหม่ ๆ ได้

2.8.12.3 ชุมชนผู้ใช้งานและนักพัฒนาที่แข็งแกร่งและมีการพัฒนาอย่างต่อเนื่อง

2.8.12.4 การเป็นพื้นฐานสำหรับการพัฒนาเทคโนโลยี Knowledge Engineering รุ่นถัดไป

2.8.12.5 การส่งผลกระทบเชิงบวกต่อการพัฒนาสังคมและเศรษฐกิจในระยะยาว

สรุปผลที่คาดว่าจะได้รับ

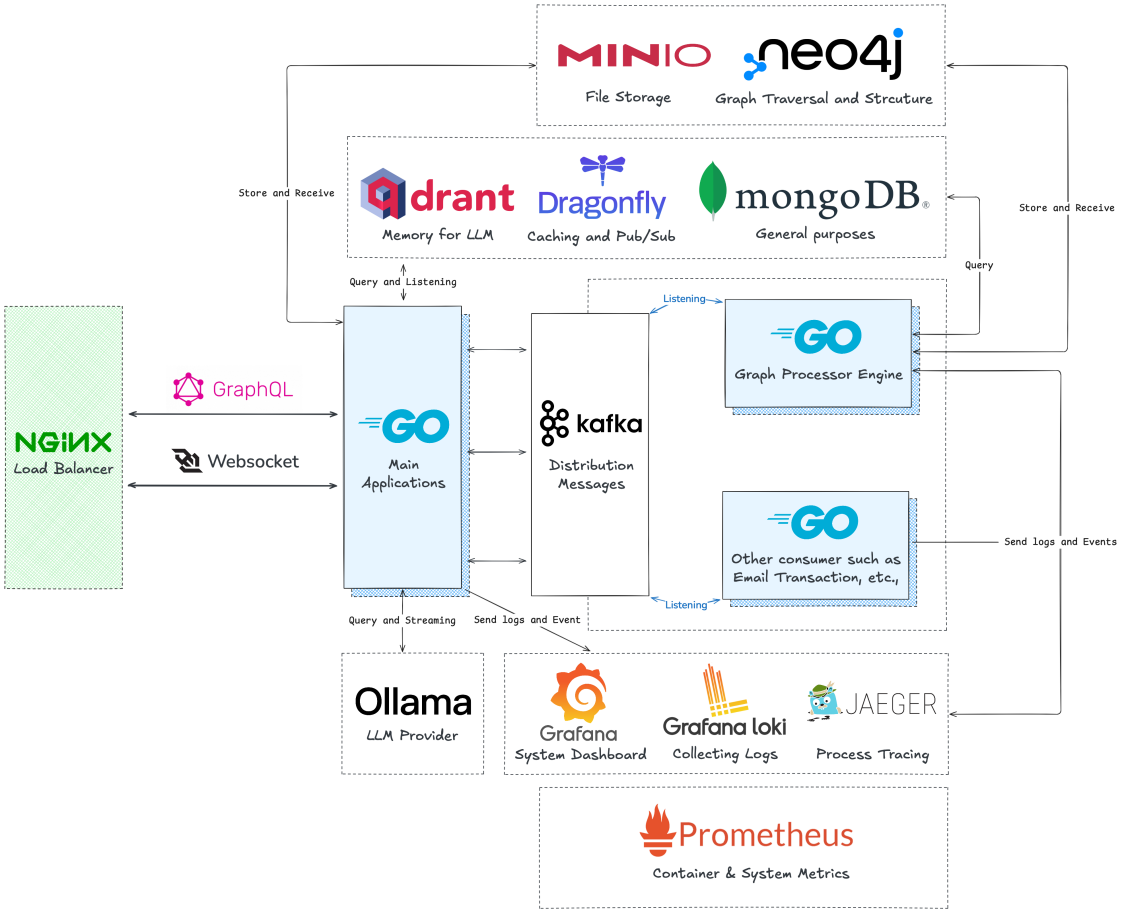
การพัฒนาแพลตฟอร์ม Co-Occurrence Knowledge Engineering Platform คาดว่าจะส่งผลกระทบเชิงบวกในหลากหลายมิติ ตั้งแต่การสร้างนวัตกรรมทางเทคนิค การส่งเสริมการวิจัยและการเรียนรู้ การเพิ่มประสิทธิภาพการทำงาน ผลลัพธ์เหล่านี้จะไม่เพียงแต่ตอบสนองความต้องการในปัจจุบันเท่านั้น แต่ยังเป็นการวางรากฐานสำหรับการพัฒนาเทคโนโลยีการจัดการความรู้ในอนาคตอีกด้วย

2.9 เอกสารอ้างอิง

1. Mario M. Kubek, Herwig Unger (2016)
   Centroid Terms as Text Representatives. [สืบค้นเมื่อ 16 กันยายน 2024]. [ออนไลน์]. https://doi.org/10.1145/2960811.2967150
2. Mario M. Kubek, Thomas Böhme, Herwig Unger (2017)
   Spreading activation: a fast calculation method for text centroids.
   [สืบค้นเมื่อ 25 ตุลาคม 2024]. [ออนไลน์]. https://doi.org/10.1145/3162957.3163014
3. Aidan Hogan, Eva Blomqvist, Michael Cochez, et al. (2021) **Knowledge Graphs.**
   [สืบค้นเมื่อ 9 สิงหาคม 2024]. [ออนไลน์]. https://doi.org/10.1145/3447772
4. Supaporn Simcharoen, Anirach Mingkhawn, Herwig Unger (2024)
   X-WiKi: Graph-Based Knowledge Aggregation from Wikipedia Pages.
   [สืบค้นเมื่อ 10 มิถุนายน 2025] [ออนไลน์]. http://dx.doi.org/10.1109/BigComp60711.2024.00100
5. Yanakorn Ruamsuk, Anirach Mingkhawn, Herwig Unger (2025)
   Enhancing Retrieval-Augmented Generation Systems by Text-Representing Centroid.
   [สืบค้นเมื่อ 18 มิถุนายน 2025]. [ออนไลน์]. https://doi.org/10.1145/3711542.3711558

2.10 ภาคผนวก

2.10.1 สถาปัตยกรรมของแพลตฟอร์มแบบภาพรวม (Architecture Overview)



รูปที่ 1: สถาปัตยกรรมของแพลตฟอร์มแบบภาพรวม

สถาปัตยกรรมของแพลตฟอร์ม Co-Occurrence Knowledge Engineering Platform ได้รับการออกแบบตามหลักการของ Microservices Architecture และ Event-Driven Architecture เพื่อให้สามารถรองรับการประมวลผลข้อมูลขนาดใหญ่และการใช้งานแบบเรียลไทม์ได้อย่างมีประสิทธิภาพ โดยแบ่งองค์ประกอบหลักออกเป็นชั้นต่าง ๆ ดังนี้

ชั้น Gateway และ Load Balancing

Nginx (Reverse Proxy & Load Balancer) ทำหน้าที่เป็นจุดเข้าถึงหลักของระบบ รับผิดชอบในการกระจายโหลดไปยัง Service ต่าง ๆ และจัดการการเชื่อมต่อจากผู้ใช้งาน โดยรองรับการสื่อสารผ่าน GraphQL API สำหรับการค้นหาและจัดการข้อมูลแบบยืดหยุ่น และ WebSocket สำหรับการสื่อสารแบบเรียลไทม์ เช่น การอัปเดตสถานะการประมวลผลกราฟ การแจ้งเตือน และการซิงโครไนซ์ข้อมูลระหว่างผู้ใช้หลายคน ระบบนี้ยังทำหน้าที่ในการจัดการ SSL/TLS Termination และการรักษาความปลอดภัยในระดับต้น

ชั้น Application Services

Main Application (Go) เป็นแกนหลักของระบบที่เขียนด้วยภาษา Go เพื่อความเร็วและประสิทธิภาพในการประมวลผล รับผิดชอบในการจัดการ Business Logic หลัก เช่น การรับและประมวลผลไฟล์ PDF การวิเคราะห์ Co-occurrence การจัดการผู้ใช้งาน การควบคุมการเข้าถึงข้อมูล และการประสานงานระหว่าง Service ต่าง ๆ ระบบนี้ยังทำหน้าที่ในการส่ง Message ไปยัง Message Queue สำหรับงานที่ต้องการการประมวลผลแบบ Asynchronous

Graph Processor Engine (Go) เป็น Service พิเศษที่เขียนด้วย Go เพื่อจัดการกับการสร้างและประมวลผลกราฟเครือข่ายขนาดใหญ่ ทำงานแบบ Consumer โดยรับ Message จาก Kafka และประมวลผลการสร้าง Node และ Edge ของกราฟแบบขนานกัน (Parallel Processing) เพื่อให้สามารถจัดการกับข้อมูลขนาดใหญ่ได้อย่างมีประสิทธิภาพ Service นี้มีความสามารถในการ Scale horizontally เมื่อมีปริมาณงานเพิ่มขึ้น

Email Transaction Service (Go) จัดการกับการส่งอีเมลต่าง ๆ เช่น การยืนยันบัญชีผู้ใช้ การแจ้งเตือนเมื่อการประมวลผลเสร็จสิ้น การส่งรายงานประจำ และการแจ้งเตือนเหตุการณ์สำคัญ โดยทำงานแบบ Asynchronous ผ่าน Message Queue เพื่อไม่ให้กระทบต่อประสิทธิภาพของ Main Application

ชั้น Message Queue และ Event Streaming

Apache Kafka ทำหน้าที่เป็นแกนกลางของ Event-Driven Architecture รับผิดชอบในการรับและกระจาย Message จำนวนมากจาก Main Application ไปยัง Consumer Services ต่าง ๆ โดยเฉพาะงานหนักอย่างการสร้าง Network Graph ขนาดใหญ่ที่จำเป็นต้องแบ่งการประมวลผลออกเป็นหลาย ๆ ส่วน Kafka ช่วยให้ระบบสามารถรองรับ High Throughput และมี Fault Tolerance สูง โดยการจัดเก็บ Message แบบ Persistent และการ Replication ข้ามหลาย Broker

ชั้นการจัดเก็บข้อมูล (Data Storage Layer)

Qdrant (Vector Database) ทำหน้าที่เป็น Memory และ Knowledge Base สำหรับ Large Language Models (LLM) จัดเก็บ Vector Embeddings ของข้อความและแนวคิดต่าง ๆ เพื่อให้ LLM สามารถทำ Semantic Search และ Context-aware Query ได้อย่างมีประสิทธิภาพ ช่วยเพิ่มความแม่นยำในการตอบคำถามและการค้นหาความเชื่อมโยงระหว่างแนวคิด

DragonflyDB (Redis Fork) เป็นระบบ In-Memory Cache ที่มีประสิทธิภาพสูง ทำหน้าที่ในการ Cache ข้อมูลที่ใช้บ่อย เช่น ผลการค้นหา Session ของผู้ใช้ และข้อมูล Metadata ต่าง ๆ นอกจากนี้ยังทำหน้าที่เป็น Pub/Sub Broker สำหรับการสื่อสารระหว่าง Service แบบเรียลไทม์ และการซิงโครไนซ์ข้อมูลระหว่าง Instance ต่าง ๆ

MongoDB (Document Database) จัดเก็บข้อมูลทั่วไปของระบบในรูปแบบ Document เช่น ข้อมูลบัญชีผู้ใช้ Metadata ของเอกสาร การตั้งค่าต่าง ๆ ประวัติการใช้งาน และข้อมูล Configuration ต่าง ๆ เหมาะสำหรับข้อมูลที่มีโครงสร้างแบบยืดหยุ่นและต้องการการเข้าถึงที่รวดเร็ว

Neo4j (Graph Database) เป็นหัวใจสำคัญของระบบที่จัดเก็บ Network Graph ที่เกิดจากการวิเคราะห์ Co-occurrence รวมถึงความสัมพันธ์ระหว่าง Node และ Edge ต่าง ๆ รองรับการ Query แบบ Graph-specific ด้วย Cypher Query Language ทำให้สามารถค้นหารูปแบบความสัมพันธ์ที่ซับซ้อนและการวิเคราะห์ Network Properties ได้อย่างมีประสิทธิภาพ

MinIO (S3-Compatible Object Storage) จัดเก็บไฟล์ต่าง ๆ ที่ผู้ใช้อัปโหลดเข้าสู่ระบบ เช่น ไฟล์ PDF เอกสารต้นฉบับ รูปภาพที่ส่งออกจากกราฟ และไฟล์ผลลัพธ์ต่าง ๆ รองรับการ Scale แบบ Distributed และมี API ที่เข้ากันได้กับ Amazon S3 ทำให้สามารถ Migrate หรือ Backup ได้ง่าย

ชั้น AI และ Machine Learning

Ollama (LLM Service) ให้บริการ Large Language Models เพื่อการประมวลผลภาษาธรรมชาติ การตีความข้อความ การสร้าง Summary และการตอบคำถามเกี่ยวกับเนื้อหาในกราฟเครือข่าย ทำงานร่วมกับ Qdrant ในการ Retrieve Context ที่เกี่ยวข้องและสร้างคำตอบที่มีความแม่นยำสูง รองรับการ Load Balance และ Scale ตามความต้องการใช้งาน

ชั้น Observability และ Monitoring

Grafana Dashboard เป็นหน้าต่างหลักสำหรับการมองภาพรวมของระบบทั้งหมด แสดงผล Metrics ต่าง ๆ เช่น การใช้งาน CPU, Memory, Network Traffic จำนวนผู้ใช้งาน สถานะของ Service ต่าง ๆ และ Performance Indicators ที่สำคัญ ช่วยให้ผู้ดูแลระบบสามารถติดตามและตัดสินใจได้อย่างรวดเร็ว

Grafana Loki (Log Aggregation) รวบรวมและจัดการ Log จาก Service ทุกตัวในระบบ ช่วยในการ Debug การวิเคราะห์ปัญหา และการติดตาม Event ต่าง ๆ ที่เกิดขึ้น รองรับการ Query Log แบบ Real-time และการสร้าง Alert เมื่อเกิด Error หรือ Anomaly

Jaeger Tracing ติดตาม Request Flow ทั้งหมด ที่ เกิด ขึ้น ใน ระบบ แสดง Latency และ Dependencies ระหว่าง Service ต่าง ๆ ช่วยในการ Optimize Performance และการระบุ Bottleneck ในระบบ เป็นเครื่องมือสำคัญสำหรับการ Debug ปัญหาที่เกิดขึ้นใน Microservices Environment

Prometheus (Metrics Collection) รวบรวมและจัดเก็บ Metrics จาก Service ทุกตัวในระบบ เช่น Response Time, Error Rate, Throughput และ Resource Usage รองรับการสร้าง Alert Rule และการ Integration กับ Grafana สำหรับการแสดงผลแบบ Real-time

การไหลของข้อมูลและการทำงานของระบบ

เมื่อผู้ใช้อัปโหลดเอกสาร PDF ผ่าน Web Interface ข้อมูลจะผ่าน Nginx เข้าสู่ Main Application ซึ่งจะประมวลผลข้อความและส่ง Message ไปยัง Kafka สำหรับการสร้างกราฟ Graph Processor Engine จะรับ Message และสร้าง Node/Edge ใน Neo4j พร้อมทั้งอัปเดต Vector Embeddings ใน Qdrant ผลลัพธ์จะถูก Cache ใน DragonflyDB และผู้ใช้จะได้รับการแจ้งเตือนผ่าน WebSocket เมื่อการประมวลผลเสร็จสิ้น

ระบบนี้ได้รับการออกแบบให้มีความ Scalable, Fault-tolerant และ Observable ช่วยให้สามารถรองรับการใช้งานระดับ Enterprise และการพัฒนาต่อยอดในอนาคตได้อย่างมีประสิทธิภาพ

2.10.2 **ความสัมพันธ์และแนวคิดข้อมูล (Data Relations and Concepts)**

การพัฒนาระบบ Co-Occurrence Knowledge Engineering Platform ต้องอาศัยการทำความเข้าใจเกี่ยวกับโครงสร้างและความสัมพันธ์ของข้อมูลอย่างลึกซึ้ง เนื่องจากระบบนี้มีเป้าหมายในการแปลงข้อความจากเอกสารให้กลายเป็นกราฟเครือข่ายความรู้ที่สามารถแสดงความเชื่อมโยงระหว่างแนวคิดต่าง ๆ ได้อย่างมีความหมาย ดังนั้นจึงจำเป็นต้องมีการออกแบบโครงสร้างข้อมูลและแนวคิดหลักที่รองรับการทำงานของระบบ

โครงสร้างกราฟเครือข่าย (Network Graph Structure)

ในการทำระบบ Knowledge Engineering จำเป็นต้องมี **Network Graph Structure** เพื่อการค้นหาความเชื่อมโยงระหว่างความรู้อย่างมีประสิทธิภาพ โครงสร้างกราฟเครือข่ายนี้ประกอบด้วยองค์ประกอบหลัก 3 ส่วน ได้แก่

2.10.2.1 **โหนด (Nodes)** แทนแนวคิด คำศัพท์ หรือเอนทิตีที่สกัดมาจากข้อความ

2.10.2.2 **ขอบเชื่อม (Edges)** แทนความสัมพันธ์และความแข็งแกร่งของการเชื่อมโยงระหว่างแนวคิด

2.10.2.3 **เซนทรอยด์ (Centroid)** เป็นจุดกึ่งกลางของกราฟทั้งหมด ซึ่งเป็น Concept หลักในการแสดงกราฟทั้งหมด

แนวคิดเซนทรอยด์ (Centroid Concept)

เซนทรอยด์ (Centroid) เป็นแนวคิดสำคัญที่ใช้ในการระบุจุดศูนย์กลางหรือแนวคิดหลักของกราฟเครือข่าย โดยเซนทรอยด์จะถูกคำนวณจากความถี่การปรากฏและระดับการเชื่อมโยงของแต่ละโหนดในกราฟ เซนทรอยด์ทำหน้าที่เป็น:

2.10.2.3.1 **ตัวแทนความหมายหลัก** ของเนื้อหาทั้งหมดในกราฟ

2.10.2.3.2 **จุดอ้างอิง** สำหรับการคำนวณความสัมพันธ์กับโหนดอื่น ๆ

2.10.2.3.3 **เครื่องมือสรุปเนื้อหา** ที่ช่วยให้เข้าใจแนวคิดหลักได้อย่างรวดเร็ว

2.10.2.3.4 **ดัชนีการค้นหา** ที่เพิ่มประสิทธิภาพในการ Query และ Retrieval

การคำนวณเซนทรอยด์ใช้อัลกอริทึม Spreading Activation ที่พิจารณาจากค่าน้ำหนักและความเชื่อมโยงของโหนดทั้งหมดในกราฟ

เงื่อนไขการสร้างกราฟเครือข่าย (Graph Creation Criteria)

ในการสร้าง Network Graph สำหรับ Co-occurrence Graph ขึ้นมาได้ จำเป็นต้องมีข้อกำหนดพื้นฐานสำหรับข้อมูลต้นทางและเงื่อนไขทางคณิตศาสตร์ที่ชัดเจน ดังนี้:

เงื่อนไขทางคณิตศาสตร์สำหรับการสร้าง Co-occurrence Graph:

1. เงื่อนไขความยาวประโยค:

สำหรับประโยคข้อความ S ที่มีความยาว n คำ จำเป็นต้องมี:

$$n > 0$$

โดยที่ n คือจำนวนคำในประโยคหลังจากการประมวลผล (Tokenization) และการกรอง Stop Words

2. เงื่อนไขการปรากฏร่วม (Co-occurrence Condition):

การปรากฏร่วมคือการปรากฏของคำสองคำในประโยคเดียวกัน โดยกำหนดให้:

สำหรับคำ w_i และ w_j ใน corpus C ที่มีการปรากฏร่วมกันด้วยความถี่ $f(w_i, w_j)$

$$f(w_i, w_j) \geq \sigma$$

โดยที่:

$f(w_i, w_j)$ = ความถี่การปรากฏร่วมของคำ w_i และ w_j ในประโยคเดียวกัน

σ = ค่าเกณฑ์ขั้นต่ำ (threshold) สำหรับการปรากฏร่วมที่มีนัยสำคัญ

กำหนดให้ $\sigma = 2$ สำหรับระบบนี้

3. เงื่อนไขการสร้างขอบเชื่อม (Edge Creation Condition):
ขอบเชื่อม e_{ij} ระหว่างโหนด w_i และ w_j จะถูกสร้างขึ้นก็ต่อเมื่อ:

$$e_{ij} = \begin{cases} 1, & \text{ถ้า } f(w_i, w_j) \geq \sigma \\ 0, & \text{ถ้า } f(w_i, w_j) < \sigma \end{cases}$$

4. น้ำหนักของขอบเชื่อม (Edge Weight):
น้ำหนักของขอบเชื่อม w_{ij} คำนวณจาก:

$$w_{ij} = \frac{f(w_i, w_j)}{\max_{k,l \in V} f(w_k, w_l)}$$

โดยที่ V คือเซตของโหนดทั้งหมดในกราฟ
5. เงื่อนไขการกรองข้อมูล (Data Filtering Condition):
เมื่อกำหนด $\sigma = 2$ แล้ว ข้อความใดที่มีความถี่การปรากฏร่วมต่ำกว่าค่านี้จะไม่ถูกนำมาสร้างกราฟ กล่าวคือ:

$$\forall (w_i, w_j) : f(w_i, w_j) < 2 \Rightarrow (w_i, w_j) \notin E$$

โดยที่ E คือเซตของขอบเชื่อมทั้งหมดในกราฟ
6. เงื่อนไขการคำนวณ Centroid:
สำหรับการคำนวณ Centroid ของกราฟ ใช้สูตร:

$$C = \sum_{w_j \in V} w_{ij} \cdot d_{ij}^{-1}$$

โดยที่:

- C = Centroid ของกราฟ
- d_{ij} = ระยะทางสั้นที่สุดระหว่างโหนด w_i และ w_j
- V = เซตของโหนดทั้งหมดที่ผ่านเงื่อนไขความถี่

7. เงื่อนไขความหนาแน่นของกราฟ (Graph Density Condition):
ความหนาแน่นของกราฟ D คำนวณจาก:

$$D = \frac{2|E|}{|V|(|V| - 1)}$$

โดยที่:

- $|E|$ = จำนวนขอบเชื่อมที่มีความถี่ $\geq \sigma$
- $|V|$ = จำนวนโหนดทั้งหมดในกราฟ

8. ตัวอย่างการประยุกต์ใช้:

ให้ประโยค: "ระบบวิศวกรรมความรู้ช่วยในการวิเคราะห์ข้อมูล"

หลังจาก Tokenization และ Stop Words Removal: ["ระบบ", "วิศวกรรม", "ความรู้", "วิเคราะห์", "ข้อมูล"]

Co-occurrence pairs ที่เป็นไปได้:

- (ระบบ, วิศวกรรม), (ระบบ, ความรู้), (ระบบ, วิเคราะห์), (ระบบ, ข้อมูล)
- (วิศวกรรม, ความรู้), (วิศวกรรม, วิเคราะห์), (วิศวกรรม, ข้อมูล)
- (ความรู้, วิเคราะห์), (ความรู้, ข้อมูล)
- (วิเคราะห์, ข้อมูล)

เฉพาะ pairs ที่มี $f(w_i, w_j) \geq 2$ เท่านั้นที่จะถูกนำมาสร้างเป็นขอบเชื่อมในกราฟ

กระบวนการสร้างกราฟ:

2.10.2.4.1 **การแบ่งคำ (Tokenization)** แยกประโยคออกเป็นคำแต่ละคำ

2.10.2.4.2 **การกรองคำ (Filtering)** กำจัด Stop Words และคำที่ไม่มีความหมาย

2.10.2.4.3 **การสร้างโหนด (Node Creation)** แปลงคำที่เหลือเป็นโหนดในกราฟ

2.10.2.4.4 **การคำนวณ Co-occurrence** วิเคราะห์การปรากฏร่วมของคำ

2.10.2.4.5 **การสร้างขอบเชื่อม (Edge Creation)** เชื่อมโยงโหนดตามค่า Co-occurrence

การจัดเก็บ Metadata และ Context

สำหรับทุกคำที่ถูกแปลงเป็นโหนดในกราฟ ระบบจะสร้างและจัดเก็บ **Metadata** ที่สำคัญดังนี้:

2.10.2.5.1 **ข้อความประโยคต้นฉบับ (Original Sentences)** เก็บประโยคเดิมที่คำนั้น ๆ เคยปรากฏอยู่

2.10.2.5.2 **ตำแหน่งในเอกสาร (Document Position)** บันทึกหน้า บรรทัด และตำแหน่งของคำ

2.10.2.5.3 **ความถี่การปรากฏ (Frequency)** นับจำนวนครั้งที่คำปรากฏในเอกสาร

2.10.2.5.4 **บริบทโดยรอบ (Surrounding Context)** เก็บคำ ๆ ข้างเคียงเพื่อสร้าง Context Window

2.10.2.5.5 **ข้อมูลการประมวลผล (Processing Metadata)** เก็บข้อมูลเกี่ยวกับวิธีการประมวลผลและ Algorithm ที่ใช้

Metadata เหล่านี้จะถูกนำไปใช้เป็น **Context** ให้กับ Large Language Models (LLM) เพื่อการ:

2.10.2.5.6 **Context-Aware Question Answering** ตอบคำถามโดยอิงจากบริบทที่ถูกต้อง

2.10.2.5.7 **Semantic Understanding** เข้าใจความหมายของคำในบริบทต่าง ๆ

2.10.2.5.8 **Knowledge Discovery** ค้นพบความรู้ใหม่จากการเชื่อมโยงบริบท

2.10.2.5.9 **Content Summarization** สร้างสรุปเนื้อหาที่มีความหมาย

ความสัมพันธ์ระหว่างข้อมูล (Data Relationships)

ระบบใช้หลายรูปแบบความสัมพันธ์เพื่อแสดงการเชื่อมโยงระหว่างข้อมูล:

2.10.2.6.1 **Co-occurrence Relationship** ความสัมพันธ์จากการปรากฏร่วมในข้อความ

2.10.2.6.2 **Semantic Relationship** ความสัมพันธ์เชิงความหมาย โดยใช้ Vector Embeddings

2.10.2.6.3 **Hierarchical Relationship** ความสัมพันธ์แบบลำดับชั้น จากการจัดหมวดหมู่

2.10.2.6.4 **Temporal Relationship** ความสัมพันธ์เชิงเวลา จากลำดับการปรากฏ

2.10.2.6.5 **Contextual Relationship** ความสัมพันธ์จากบริบทที่ใช้งาน

โมเดลข้อมูลสำหรับระบบ (Data Model)

ระบบใช้โมเดลข้อมูลแบบ Hybrid ที่ผสมผสานหลายแนวทาง:

Graph Data Model:

- Nodes: แทนแนวคิดและเอนทิตี
- Edges: แทนความสัมพันธ์และน้ำหนัก
- Properties: แทน Metadata และ Attributes

Vector Data Model:

- Embeddings: แทนความหมายเชิงลึกของข้อความ
- Similarity Scores: แทนความคล้ายคลึงทางความหมาย
- Clustering Information: แทนการจัดกลุ่มตามความหมาย

Document Data Model:

- Original Text: เก็บข้อความต้นฉบับ
- Processed Text: เก็บข้อความหลังประมวลผล
- Metadata: เก็บข้อมูลเสริมและการประมวลผล

การประยุกต์ใช้แนวคิดในระบบ

แนวคิดเหล่านี้ถูกนำไปประยุกต์ใช้ในระบบเพื่อสร้างประสบการณ์การใช้งานที่มีประสิทธิภาพ ได้แก่:

2.10.2.7.1 **Intelligent Search** การค้นหาที่เข้าใจบริบทและความหมาย

2.10.2.7.2 **Dynamic Visualization** การแสดงผลกราฟที่ปรับตัวตามข้อมูล

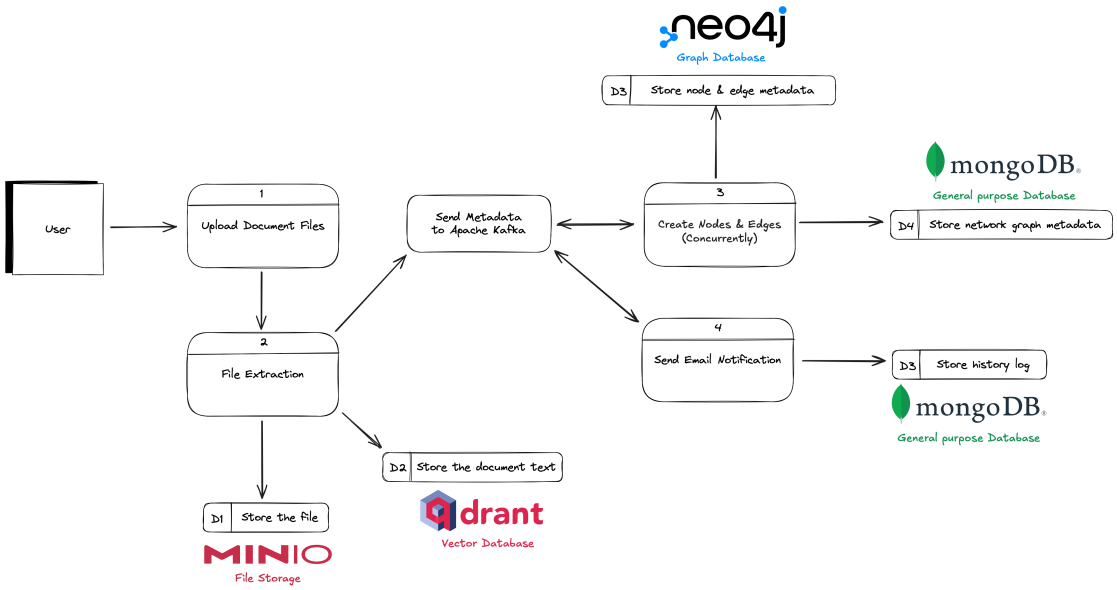
2.10.2.7.3 **Pattern Discovery** การค้นพบรูปแบบและแนวโน้มใหม่

2.10.2.7.4 **Knowledge Fusion** การผสมผสานความรู้จากหลายแหล่ง

2.10.2.7.5 **Automated Insights** การสร้าง Insights อัตโนมัติจากข้อมูล

ความสำคัญของการออกแบบโครงสร้างข้อมูลและแนวคิดเหล่านี้คือ ช่วยให้ระบบสามารถจัดการกับความซับซ้อนของความรู้และสร้างความเข้าใจที่ลึกซึ้งจากข้อมูลที่มีอยู่ ทำให้ผู้ใช้งานสามารถค้นพบความเชื่อมโยงและสร้างความรู้ใหม่ได้อย่างมีประสิทธิภาพ

2.10.3 เส้นทางข้อมูลการสร้างกราฟ (Graph Creation Data Flow)



รูปที่ 2: เส้นทางข้อมูลการสร้างกราฟของแพลตฟอร์ม

เส้นทางข้อมูลการสร้างกราฟของแพลตฟอร์ม Co-Occurrence Knowledge Engineering Platform ได้รับการออกแบบให้มีความสะดวก เป็นระบบ และมีประสิทธิภาพในการจัดการข้อมูลจำนวนมาก โดยใช้หลักการของ **Event-Driven Architecture** และ **Microservices** เพื่อให้สามารถประมวลผลข้อมูลแบบ Asynchronous และ Scale ได้ตามปริมาณงาน ขั้นตอนการทำงานของระบบมีดังนี้:

ขั้นตอนที่ 1: การอัปโหลดเอกสารจากผู้ใช้ (Document Upload)

ผู้ใช้งานเริ่มต้นกระบวนการโดยการอัปโหลดไฟล์เอกสาร โดยเฉพาะไฟล์ PDF ผ่านทาง Web Interface ของแพลตฟอร์ม ระบบจะทำการตรวจสอบความถูกต้องของไฟล์ เช่น ขนาดไฟล์ รูปแบบไฟล์ และความสมบูรณ์ของข้อมูล หลังจากการตรวจสอบเสร็จสิ้น ไฟล์จะถูกส่งผ่าน **Nginx Load Balancer** เข้าสู่ระบบ Backend เพื่อเริ่มขั้นตอนการประมวลผล การออกแบบนี้ช่วยให้ระบบสามารถรองรับการอัปโหลดไฟล์หลายไฟล์พร้อมกันได้อย่างมีประสิทธิภาพ

ขั้นตอนที่ 2: การสกัดข้อความจากเอกสาร (Text Extraction)

เมื่อไฟล์เอกสารเข้าสู่ระบบ Backend แล้ว **Main Application** จะทำการสกัดข้อความ (Text Extraction) จากไฟล์ PDF โดยใช้ **PDF Text Parsing** สำหรับเอกสารที่มีข้อความดิจิทัล ระบบจะทำการจัดการกับเลย์เอาต์ที่ซับซ้อน การแยกส่วนหัวข้อ เนื้อหา และส่วนอ้างอิง รวมถึงการจัดการกับภาษาไทยที่มีลักษณะการเว้นวรรคและการตัดคำที่แตกต่างจากภาษาอังกฤษ ผลลัพธ์ที่ได้จะเป็นข้อความที่สะอาดและพร้อมสำหรับการวิเคราะห์ขั้นต่อไป

ขั้นตอนที่ 3: การจัดเก็บข้อมูลในระบบหลายชั้น (Multi-tier Data Storage)

ข้อมูลที่ผ่านการประมวลผลแล้วจะถูกจัดเก็บในระบบที่มีหลายชั้น ได้แก่ **MinIO Object Storage** สำหรับเก็บไฟล์ต้นฉบับและไฟล์ที่เกี่ยวข้อง เช่น รูปภาพที่สกัดจากเอกสาร Metadata ต่าง ๆ และ **Qdrant Vector Database** สำหรับจัดเก็บ **Vector Embeddings** ของข้อความ ซึ่งจะใช้ในการทำ **Semantic**

Search และการให้บริการ **Large Language Models (LLM)** ในขั้นตอนต่อไป การแยกการจัดเก็บข้อมูลแบบนี้ช่วยให้ระบบสามารถเข้าถึงข้อมูลได้รวดเร็วและมีประสิทธิภาพตามการใช้งานที่แตกต่างกัน

ขั้นตอนที่ 4: การส่งข้อมูลผ่าน Message Queue (Batch Processing via Kafka)

เพื่อให้ระบบสามารถจัดการกับข้อมูลจำนวนมากได้อย่างมีประสิทธิภาพ ระบบจะส่งข้อมูลไปยัง **Apache Kafka Message Queue** แบบทีละ Batch ซึ่งช่วยในการจัดการกับข้อมูลที่มีปริมาณมากโดยไม่ทำให้ระบบล่มหรือตอบสนองช้า Kafka ทำหน้าที่เป็น **Event Streaming Platform** ที่รองรับการประมวลผลแบบ **Asynchronous** และมี **Fault Tolerance** สูง โดยสามารถกู้คืนข้อมูลได้ในกรณีที่เกิดปัญหา การออกแบบนี้ช่วยให้ระบบสามารถรองรับการใช้งานจำนวนมากพร้อมกันได้โดยไม่ส่งผลกระทบต่อประสิทธิภาพ

ขั้นตอนที่ 5: การสร้าง Node และ Edge Metadata พร้อมการแจ้งเตือน (Graph Construction & Notification)

Graph Processor Engine ที่เขียนด้วยภาษา Go จะทำหน้าที่เป็น **Consumer** รับข้อมูลจาก Kafka แล้วประมวลผลการสร้าง **Node และ Edge** ของกราฟเครือข่าย โดยใช้อัลกอริทึม **Co-occurrence Analysis** ในการคำนวณความสัมพันธ์ระหว่างคำและแนวคิดต่าง ๆ ข้อมูล Metadata ที่สร้างขึ้นจะถูกจัดเก็บใน **Neo4j Graph Database** เพื่อการค้นหาและวิเคราะห์ความสัมพันธ์แบบกราฟ ในขณะเดียวกันระบบจะส่ง **Email Notification** ผ่าน **Email Transaction Service** เพื่อแจ้งให้ผู้ใช้ทราบเมื่อการประมวลผลเสร็จสิ้น และจัดเก็บ **History Log** ลงใน **MongoDB** เพื่อการติดตามและการตรวจสอบประวัติการทำงานของระบบ

ขั้นตอนที่ 6: การจัดเก็บ Network Graph และการเตรียมพร้อมใช้งาน (Graph Storage & Optimization)

หลังจากการสร้างกราฟเครือข่ายเสร็จสิ้น ข้อมูล **Network Graph** ทั้งหมดจะถูกจัดเก็บใน **MongoDB** ในรูปแบบที่เหมาะสำหรับการเรียกใช้งานในครั้งถัดไป โดยการจัดเก็บจะรวมถึง **Graph Metadata, Node Properties, Edge Weights, และ Visualization Settings** ต่าง ๆ ระบบจะทำการ **Indexing** และ **Optimization** เพื่อให้การเข้าถึงข้อมูลมีความรวดเร็ว นอกจากนี้ ข้อมูลที่ใช้บ่อยจะถูก Cache ลงใน **DragonflyDB (Redis Fork)** เพื่อเพิ่มประสิทธิภาพในการตอบสนอง และระบบจะส่งสัญญาณผ่าน **WebSocket** เพื่อแจ้งให้ผู้ใช้ทราบว่าข้อมูลพร้อมใช้งานแล้ว

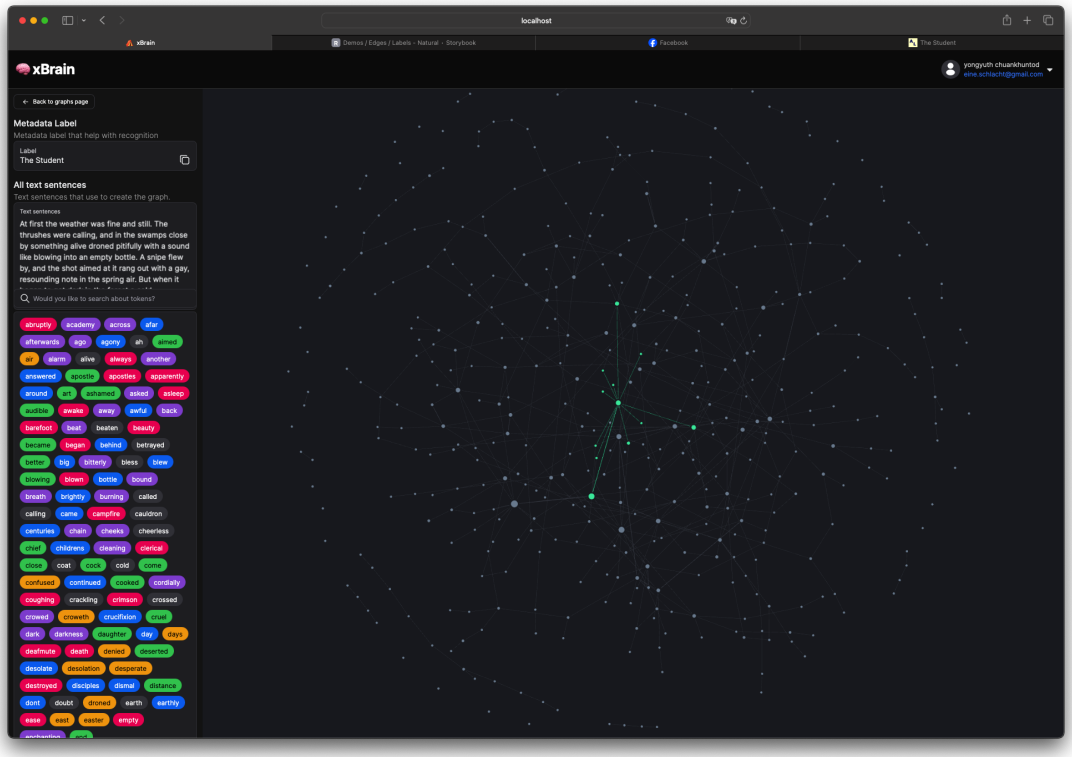
ข้อดีของการออกแบบเส้นทางข้อมูลแบบนี้

1. **ประสิทธิภาพสูง (High Performance):** การใช้ Event-Driven Architecture ช่วยให้ระบบสามารถประมวลผลข้อมูลแบบขนานกันได้ ลดเวลาในการรอ
2. **ความทนทานต่อข้อผิดพลาด (Fault Tolerance):** Kafka และ Message Queue ช่วยให้ระบบสามารถกู้คืนได้เมื่อเกิดปัญหา
3. **ความสามารถในการขยายตัว (Scalability):** สามารถเพิ่ม Consumer และ Worker ได้ตามปริมาณงานที่เพิ่มขึ้น
4. **การแยกความรับผิดชอบ (Separation of Concerns):** แต่ละ Service มีหน้าที่เฉพาะ ทำให้ง่ายต่อการพัฒนาและบำรุงรักษา
5. **การติดตามและควบคุม (Monitoring & Observability):** ระบบมี Logging, Metrics และ Tracing ครบถ้วนผ่าน Grafana, Loki, Jaeger และ Prometheus

การประยุกต์ใช้กับ Large Language Models (LLM)

ข้อมูลที่จัดเก็บใน Vector Database (Qdrant) และ Graph Database (Neo4j) จะถูกนำไปใช้กับ **Ollama LLM Service** เพื่อการทำ **Retrieval-Augmented Generation (RAG)** ทำให้ LLM สามารถตอบคำถามและให้ข้อมูลที่แม่นยำขึ้นโดยอิงจากบริบทจากกราฟเครือข่ายที่สร้างขึ้น การรวมกันนี้ช่วยให้ผู้ใช้งานสามารถค้นหาความรู้และทำความเข้าใจความเชื่อมโยงระหว่างแนวคิดต่าง ๆ ได้อย่างลึกซึ้งและมีประสิทธิภาพ

ภาพตัวอย่างของแพลตฟอร์ม Co-Occurrence Knowledge Engineering Platform ที่แสดงให้เห็นถึงการทำงานและความสามารถของระบบ โดยจะแบ่งออกเป็น 2 ภาพหลัก ได้แก่ กราฟขนาดใหญ่ที่แสดงความสัมพันธ์ระหว่างเอกสารและกราฟขนาดเล็กที่แสดงความสัมพันธ์ระหว่างแนวคิดต่าง ๆ ในเอกสาร



รูปที่ 3: กราฟขนาดใหญ่ที่แสดงความสัมพันธ์

ลงชื่อ **นายยงยุทธ ชวนขุนทด** ผู้เสนอโครงการ

(นาย ยงยุทธ ชวนขุนทด)

วันที่ ...../...../.....

ความเห็นอาจารย์ที่ปรึกษา .....

.....

.....

ลงชื่อ ..... อาจารย์ที่ปรึกษา

(.....)

วันที่ ...../...../.....

สาขาวิชา / ภาควิชาที่ได้รับแบบเสนอโครงงานวันนี้ .....

ผลการพิจารณา .....

.....

.....

ลงชื่อ ..... ประธาน

(.....)

วันที่ ...../...../.....

ลงชื่อ ..... กรรมการ

(.....)

วันที่ ...../...../.....

ลงชื่อ ..... กรรมการ

(.....)

วันที่ ...../...../.....

ลงชื่อ ..... กรรมการ

(.....)

วันที่ ...../...../.....