



การค้นหาและการจัดกลุ่มบทความทางวิชาการ ด้วยการประมวลผลภาษาธรรมชาติ

โดย

นางสาว พิมพ์กา เดชประภัสสร

โครงการพิเศษนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร

วิทยาศาสตร์บัณฑิต

สาขาวิชาวิทยาการคอมพิวเตอร์

คณะวิทยาศาสตร์และเทคโนโลยี มหาวิทยาลัยธรรมศาสตร์

ปีการศึกษา 2567

ลิขสิทธิ์ของมหาวิทยาลัยธรรมศาสตร์

การค้นหาและการจัดกลุ่มบทความทางวิชาการ ด้วยการประมวลผลภาษาธรรมชาติ

โดย

นางสาว พิมพ์กา เดชประภัสสร

โครงการพิเศษนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร

วิทยาศาสตร์บัณฑิต

สาขาวิชาวิทยาการคอมพิวเตอร์

คณะวิทยาศาสตร์และเทคโนโลยี มหาวิทยาลัยธรรมศาสตร์

ปีการศึกษา 2567

ลิขสิทธิ์ของมหาวิทยาลัยธรรมศาสตร์

RETRIEVAL AND CLUSTERING OF ACADEMIC ABSTRACTS USING
NATURAL LANGUAGE PROCESSING TECHNIQUES

BY

Ms. PIMPIKA DEJPRAPATSORN

A FINAL-YEAR PROJECT REPORT SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF BACHELOR OF SCIENCE

COMPUTER SCIENCE

FACULTY OF SCIENCE AND TECHNOLOGY

THAMMASAT UNIVERSITY

ACADEMIC YEAR 2024

COPYRIGHT OF THAMMASAT UNIVERSITY

มหาวิทยาลัยธรรมศาสตร์
คณะวิทยาศาสตร์และเทคโนโลยี

รายงานโครงการพิเศษ

ของ

นางสาว พิมพ์กา เดชประภัสสร

เรื่อง


การค้นหาและการจัดกลุ่มบทความทางวิชาการ ด้วยการประมวลผลภาษาธรรมชาติ

ได้รับการตรวจสอบและอนุมัติ ให้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร

หลักสูตรวิทยาศาสตรบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์


เมื่อ วันที่ 30 พฤษภาคม พ.ศ. 2568

อาจารย์ที่ปรึกษา



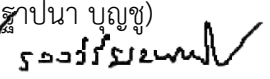
(ผศ.ดร. ปกป้อง ส่องเมือง)

กรรมการสอบโครงการพิเศษ



(ผศ.ดร. ฐาปนา บุญชู)

กรรมการสอบโครงการพิเศษ



(ผศ.ดร. ทรงศักดิ์ ร่องวิริยะพานิช)

มหาวิทยาลัยธรรมศาสตร์
คณะวิทยาศาสตร์และเทคโนโลยี

รายงานโครงการพิเศษ

ของ

นางสาว พิมพ์กา เดชประภัสสร

เรื่อง

การค้นหาและการจัดกลุ่มบทความทางวิชาการ ด้วยการประมวลผลภาษาธรรมชาติ

ได้รับการตรวจสอบและอนุมัติ ให้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร

หลักสูตรวิทยาศาสตรบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์

เมื่อ วันที่ 30 พฤษภาคม พ.ศ. 2568

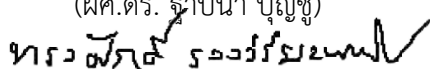
อาจารย์ที่ปรึกษา


(ผศ.ดร. ปกป้อง ส่องเมือง)

กรรมการสอบโครงการพิเศษ


(ผศ.ดร. รุาปนา บุญชู)

กรรมการสอบโครงการพิเศษ


ทรงศักดิ์ ร่องวิริยะพานิช

(ผศ.ดร. ทรงศักดิ์ ร่องวิริยะพานิช)

หัวข้อโครงการพิเศษ	การค้นหาและการจัดกลุ่มบทความทางวิชาการ ด้วยการประมวลผลภาษาธรรมชาติ
ชื่อผู้เขียน	นางสาว พิมพ์กา เดชประภัสสร
ชื่อปริญญา	วิทยาศาสตรบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์
สาขาวิชา/คณะ/มหาวิทยาลัย	สาขาวิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์และเทคโนโลยี มหาวิทยาลัยธรรมศาสตร์
อาจารย์ที่ปรึกษาโครงการพิเศษ	ผศ.ดร. ปกป้อง ส่องเมือง
ปีการศึกษา	2567

บทคัดย่อ

โครงการนี้มีวัตถุประสงค์ คือเสนอแนวทางเพื่อพัฒนาระบบการค้นหาและการจัดกลุ่มบทความ สำหรับผลงานวิจัยทางวิชาการที่มีอยู่ในปัจจุบันให้มีคุณสมบัติที่ดีขึ้น โดยใช้เครื่องมือที่เป็น Deep Learning เข้ามาช่วยในการจัดหมวดหมู่คำที่มีความหมายหรือความคล้ายคลึงกัน เพื่อให้ระบบสามารถค้นหาได้อย่างมีประสิทธิภาพโดยไม่จำเป็นต้องใช้คำศัพท์เฉพาะทาง หรือคำที่ปรากฏเฉพาะในบทความเท่านั้น ซึ่งช่วยอำนวยความสะดวกแก่ผู้ใช้งาน และลดระยะเวลาในการสืบค้นบทความได้อย่างมีประสิทธิภาพ

ในโครงการนี้ ผู้จัดทำได้ทำการทดลองการประเมินความแม่นยำในการค้นหาคำของเทคนิค NLP ทั้งหมด 4 เทคนิคด้วยกันคือ TF-IDF, Word2Vec, Doc2Vec และ BERT โดยเกณฑ์ในการวัดคือ ผู้จัดทำจะมี keyword สำหรับการค้นหา 100 บทความ ที่ถูกคัดเลือกมาแล้วว่าเมื่อนำคำเหล่านี้ไปค้นหาจะขึ้นผลลัพธ์ตรงกับบทความเหล่านั้น หลังจากทำ embedding แต่ละเทคนิคแล้ว ให้นำ keyword ที่เตรียมไว้ไปค้นหา เพื่อดูว่าเทคนิคไหนมีความถูกต้องในการค้นหามากที่สุด เพื่อนำเทคนิคนั้นไปทำการต่อยอดพัฒนาระบบการค้นหาบทความให้ดียิ่งขึ้น

คำสำคัญ: BERT, Deep Learning, การจัดหมวดหมู่คำ, Cosine Similarity, Word2Vec, Doc2Vec, TF-IDF, NLP

Thesis Title	RETRIEVAL AND CLUSTERING OF ACADEMIC ABSTRACTS USING NATURAL LANGUAGE PROCESSING TECHNIQUES
Author	Ms. Pimpika Dejprapatsorn
Degree	Bachelor of Science
Major Field/Faculty/University	Computer Science Faculty of Science and Technology Thammasat University
Project Advisor	Assistant Professor Dr.Pokpong Songmuang
Academic Years	2024

ABSTRACT

This project aims to propose an approach to enhance the efficiency of search and clustering systems for academic research abstracts. The development leverages deep learning tools to categorize words based on their semantic similarity, enabling the system to perform effective searches without requiring users to input highly specific or domain-specific terms that are exclusively present within the abstracts. This approach aims to facilitate user accessibility and significantly reduce the time spent searching for relevant abstracts.

In this project, the authors evaluated the search accuracy of four NLP techniques: TF-IDF, Word2Vec, Doc2Vec, and BERT. The evaluation was conducted by preparing a set of keywords specifically selected to correspond to a curated collection of 100 research abstracts. After generating embeddings for each technique, the keywords were used to perform searches against the dataset, and the accuracy of each technique was assessed based on the correctness of the retrieved abstracts. The technique demonstrating the highest accuracy will be selected for further development to enhance the performance and reliability of the abstract search system.

Keywords: BERT, Deep Learning, Clustering, Cosine Similarity, Word2Vec, Doc2Vec, TF-IDF, NLP Technique

กิตติกรรมประกาศ

โครงการพิเศษฉบับนี้สำเร็จลุล่วงได้อย่างสมบูรณ์ ด้วยความอนุเคราะห์และความกรุณาอย่างยิ่งจากอาจารย์ที่ปรึกษาโครงการ ผศ.ดร. ปกป้อง ส่องเมือง ซึ่งได้เสียสละเวลาอันมีค่าเพื่อให้คำปรึกษา แนะนำ ติดตามความก้าวหน้า และช่วยแก้ไขข้อบกพร่องต่าง ๆ ในการดำเนินโครงการฉบับนี้ จนทำให้โครงการมีความถูกต้องและสมบูรณ์ ผู้จัดทำขอกราบขอบพระคุณท่านอาจารย์ที่ปรึกษาเป็นอย่างสูงมา ณ โอกาสนี้

ขอขอบคุณที่ปรึกษาเอก ผศ. สุรสิทธิ์ อภัยวัฒนาวงศ์ ที่ให้ความอนุเคราะห์ร่วมกับ ผศ.ดร. ปกป้อง ส่องเมือง ที่คอยดูแล และให้คำแนะนำในการทำโครงการพิเศษฉบับนี้ขึ้นมา รวมทั้งการติดตามความก้าวหน้าของโครงการ จนทำให้โครงการนั้นมีความถูกต้องและสมบูรณ์

ขอขอบคุณคณะกรรมการสอบโครงการพิเศษที่กรุณาเป็นกรรมการสอบ ให้คำปรึกษา ข้อชี้แนะ คำแนะนำต่างๆในการปรับปรุงแก้ไขข้อบกพร่องในการทำโครงการพิเศษ โดยคณะกรรมการสอบประกอบด้วย ผศ.ดร. ฐาปนา บุญชู และ ผศ.ดร. ทรงศักดิ์ ร่องวิริยะพานิช

ขอขอบคุณคณาจารย์ทุกท่านภายในภาควิชาวิทยาการคอมพิวเตอร์ มหาวิทยาลัยธรรมศาสตร์ ที่มอบความรู้และประสบการณ์ที่ดีตลอดระยะเวลาที่ศึกษาภายในหลักสูตร

ขอขอบคุณครอบครัวสำหรับความห่วงใยและกำลังใจที่ดีมาตลอดตั้งแต่วันแรกที่เข้ารับการศึกษาจนถึงวันสุดท้ายของการศึกษา รวมถึงพี่น้อง และเพื่อนๆภายในภาควิชาวิทยาการคอมพิวเตอร์ทุกคนที่เป็นกำลังใจที่ดีในการทำโครงการพิเศษฉบับนี้ และคอยช่วยเหลือเกื้อกูลตลอดมา

นางสาวพิมพ์ิกา เดชประภัสสร

สารบัญ

	หน้า
บทคัดย่อ	1
ABSTRACT	2
กิตติกรรมประกาศ	3
สารบัญ	4
สารบัญตาราง	7
สารบัญภาพ	8
รายการสัญลักษณ์และคำย่อ	10
บทที่ 1 บทนำ	1
1.1 ความเป็นมาและความสำคัญของโครงการ	1
1.2 วัตถุประสงค์	2
1.3 ขอบเขตของโครงการ	2
1.4 ประโยชน์ของโครงการ	2
1.5 ข้อจำกัดของโครงการ	2
บทที่ 2 วรรณกรรมและงานวิจัยที่เกี่ยวข้อง	3
2.1 แนวคิดทฤษฎีที่เกี่ยวข้อง	3
2.1.1 NLP (Natural Language Processing)	3
2.1.2 Word Embedding	4
2.1.3 TF-IDF	5

2.1.4	Word2Vec	7
2.1.5	Doc2Vec	9
2.1.6	BERT (Bidirectional Encoder Representations from Transformers)	10
2.1.7	Cosine Similarity	17
2.1.8	K-Means	18
2.2	เครื่องมือ (Tools)	21
2.2.1	Google Colab (Google Colaboratory)	21
2.3	งานวิจัยที่เกี่ยวข้อง	22
2.3.1	Automatic Retrieval and Clustering of Similar Words	22
2.3.2	The application of NLP in information retrieval	23
2.3.3	Clustering articles based on semantic similarity	24
บทที่ 3	วิธีการวิจัย	26
3.1	ภาพรวมของโครงการ	26
3.1.1	ชุดข้อมูลที่ทำมาทำการทดลอง	26
3.1.2	Dataflow	27
3.1.3	กระบวนการทำ Data Preprocessing	29
3.1.4	การทำ Embedding และการวัดความคล้ายคลึงด้วย Cosine Similarity	31
3.2	การวิเคราะห์ขอบเขตและความต้องการของระบบ	34
3.3	ประเด็นที่น่าสนใจและสิ่งที่ท้าทาย	34
3.4	ผลลัพธ์ที่คาดหวัง	34
3.5	ระบบต้นแบบและผลลัพธ์เบื้องต้น	34
บทที่ 4	ผลการดำเนินงาน	36
4.1	ผลการทดสอบความแม่นยำในการค้นหาคำศัพท์ทางวิชาการของแต่ละเทคนิค	36

4.2 ผลการทดลองการ Clustering ด้วยเทคนิค K-means และ ใช้ Silhouette Score ใน การวัดความเหมาะสมของจำนวนกลุ่มกับข้อมูลในการทดลอง	37
บทที่ 5 สรุป	39
รายการอ้างอิง	41
ภาคผนวก	43

สารบัญตาราง

	หน้า
ตารางที่ 3-1 ผลการทดลองสำหรับการทำ Embedding และวัดความคล้ายคลึงของเอกสารด้วย Cosine Similarity จาก 4 เทคนิค	34
ตารางที่ 4-1 ผลการทดลองสำหรับการทำ Embedding และวัดความคล้ายคลึงของเอกสารด้วย Cosine Similarity จาก 4 เทคนิค	36
ตารางที่ 4-2 ผลการทดลองวัดความเหมาะสมของการจัดกลุ่ม โดยใช้ Silhouette Scores	38

สารบัญภาพ

	หน้า
ภาพที่ 2-1 ภาพแสดงการทำ Tokenization	3
ภาพที่ 2-2 ภาพแสดงการทำ POS Tagging	4
ภาพที่ 2-3 ภาพแสดงการทำ Word Embedding	5
ภาพที่ 2-4 ภาพแสดงการทำ Embedding Word Vector	6
ภาพที่ 2-5 สมการของ TF	6
ภาพที่ 2-6 สมการของ IDF	6
ภาพที่ 2-7 สมการของ TF-IDF	6
ภาพที่ 2-8 ภาพแสดงลักษณะของโครงสร้างของโมเดล CBOW เบื้องต้น	7
ภาพที่ 2-9 ภาพแสดงตัวอย่างการทำ CBOW	8
ภาพที่ 2-10 ภาพแสดงตัวอย่างการทำ Skip-Gram	8
ภาพที่ 2-11 ภาพแสดงตัวอย่างการทำงาน Doc2Vec ในรูปแบบ PV-DM	9
ภาพที่ 2-12 ภาพแสดงตัวอย่างการทำงานของ BERT	10
ภาพที่ 2-13 ภาพแสดงตัวอย่างการนำ BERT ไปใช้ในการค้นหา	11
ภาพที่ 2-14 ภาพแสดงการทดสอบ Mask Strategy	13
ภาพที่ 2-15 ภาพแสดงการทำ Next Sentence Prediction	14
ภาพที่ 2-16 ภาพแสดงการทดสอบ Mask Strategy	15
ภาพที่ 2-17 ภาพแสดงการทำ Fine-tuning เพื่อแก้ปัญหาที่อยู่ในระดับคำ	16
ภาพที่ 2-18 ภาพแสดงการทำ Fine-tuning เพื่อแก้ปัญหาที่อยู่ในระดับประโยค	16
ภาพที่ 2-19 สมการของ Cosine Similarity	17
ภาพที่ 2-20 ภาพแสดงตัวอย่างการหา Cosine Similarity	18
ภาพที่ 2-21 ภาพแสดงตัวอย่าง K-Means	19
ภาพที่ 2-22 ภาพสมการของ K-means	19
ภาพที่ 2-23 ภาพสมการของ K-means	19
ภาพที่ 2-24 ภาพสมการของ Euclidean Distance	21
ภาพที่ 2-25 ภาพแสดงสัญลักษณ์ของ Google Colab	22
ภาพที่ 3-1 ภาพแสดงข้อมูลเกี่ยวกับชุดข้อมูลที่นำมาทำการทดลอง	27
ภาพที่ 3-2 ภาพแสดง Data Flow ของโครงงาน	27
ภาพที่ 3-3 ภาพแสดงข้อมูลเกี่ยวกับการทำ Indexing	29

ภาพที่ 3-4 ภาพแสดงเกี่ยวกับการลบ Stop Words และตัวอย่างคำ Stop Words ที่มีอยู่ใน SpaCy	29
ภาพที่ 3-5 ภาพแสดงการทำ Tokenization และ Lemmatization	30
ภาพที่ 3-6 ภาพแสดงตัวอย่างข้อมูลที่ผ่านมาการทำ Data Preprocessing ครบทุกขั้นตอนแล้ว	30
ภาพที่ 3-7 ภาพแสดงตัวอย่างคำค้นที่เตรียมไว้สำหรับการทดลองการค้นหาคำตัดย่อ	32
ภาพที่ 3-8 ภาพแสดงการ Embedding และการทำ Cosine Similarity ของเทคนิค TF-IDF	32
ภาพที่ 3-9 ภาพแสดงการ Embedding และการทำ Cosine Similarity ของเทคนิค Doc2Vec	
โมเดล DM และโมเดล DBoW	32
ภาพที่ 3-10 ภาพแสดงการ Embedding และการทำ Cosine Similarity ของเทคนิค Word2Vec	
โมเดล CBOW และ Skip-gram	32
ภาพที่ 3-11 ภาพแสดงการ Embedding และการทำ Cosine Similarity ของเทคนิค BERT	32
ภาพที่ 3-12 ภาพแสดง Data Visualization การทดลองสำหรับการทำ Embedding และวัดความคล้ายคลึง ของเอกสารด้วย Cosine Similarity จาก 4 เทคนิค	35
ภาพที่ 4-1 ภาพแสดง Data Visualization การการ clustering ด้วยเทคนิค K-Means	37

รายการสัญลักษณ์และคำย่อ

สัญลักษณ์/คำย่อ

NLP

ML

TF-IDF

CBOW

คำเต็ม/คำจำกัดความ

Natural Language Processing

Machine Learning

Term Frequency Inverse Document
Frequency

Continuous Bag of words

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของโครงการ

ในปัจจุบันมีเว็บไซต์จำนวนมากที่ให้บริการค้นหาและรวบรวมข้อมูลเกี่ยวกับบทความที่เกี่ยวข้องกับผลงานวิจัยทางวิชาการ อย่างไรก็ตาม การสืบค้นข้อมูลในเว็บไซต์เหล่านี้ยังคงมีความยุ่งยากและซับซ้อนในบางกรณี เช่น เมื่อผู้ใช้งานไม่ทราบคำศัพท์เฉพาะที่เกี่ยวข้องและใช้คำศัพท์ทั่วไปในการค้นหา ซึ่งอาจส่งผลให้ไม่พบผลลัพธ์ที่ต้องการ ตัวอย่างเช่น ผู้ใช้งานพิมพ์คำว่า "Maid" ในการค้นหา แต่ระบบไม่สามารถแสดงผลลัพธ์ได้ ทั้งที่ในบทความที่เกี่ยวข้องอาจมีคำว่า "Housekeeper" หรือ "Servant" ซึ่งเป็นคำศัพท์เฉพาะที่สื่อความหมายเดียวกันอยู่ การพัฒนาระบบค้นหาที่สามารถเชื่อมโยงคำทั่วไปและคำศัพท์เฉพาะเข้าด้วยกันจึงมีความสำคัญ เพื่อเพิ่มความแม่นยำและประสิทธิภาพในการสืบค้นข้อมูลทางวิชาการ

นอกจากนี้ ระบบการค้นหาในปัจจุบันยังไม่สามารถรองรับการใช้คำที่มีความหมายใกล้เคียงหรือคำที่เกี่ยวข้องกันมาประมวลผลได้อย่างมีประสิทธิภาพ ตัวอย่างเช่น หากผู้ใช้งานพิมพ์คำว่า "Father" เพื่อค้นหา ข้อมูลที่คาดว่าจะควรปรากฏอาจรวมถึงคำที่มีความหมายเกี่ยวข้อง เช่น "Mother", "Parent", "Descendants", หรือ "Children" แต่ด้วยข้อจำกัดของระบบการค้นหาในปัจจุบัน ผลลัพธ์ดังกล่าวมักจะไม่ปรากฏตามที่คาดหวัง ซึ่งสะท้อนถึงความจำเป็นในการพัฒนาระบบที่สามารถเชื่อมโยงคำที่มีความหมายใกล้เคียงหรือคำที่สัมพันธ์กันเข้าด้วยกัน เพื่อให้การค้นหามีความสะดวกสบายและตอบสนองความต้องการของผู้ใช้งานได้ดียิ่งขึ้น

โครงการนี้จึงเสนอแนวทางในการพัฒนาระบบการค้นหาให้มีประสิทธิภาพที่ดียิ่งขึ้น โดยใช้อัลกอริทึม BERT เป็นเครื่องมือในการจัดกลุ่มคำที่มีความหมายใกล้เคียงกัน (Semantic Clustering) และยังมีคุณสมบัติที่สามารถเปลี่ยนแปลงความหมายของคำตามบริบทที่ปรากฏในประโยคได้ (Contextual Embedding) ตัวอย่างเช่น ในประโยค "Please turn on the light." และ "This shirt is too light for winter." คำว่า "Light" ในสองประโยคนี้นี้มีบริบทและหน้าที่ที่แตกต่างกัน โดยในประโยคแรก "Light" ทำหน้าที่เป็นกรรม(object) ขณะที่ในประโยคที่สอง "Light" ทำหน้าที่เป็นคำคุณศัพท์(Adjective) ในการอธิบายคุณลักษณะของเสื้อ โครงการนี้จึงนำคุณสมบัติในการเปลี่ยนแปลงความหมายของคำตามบริบทไปพัฒนาระบบค้นหาที่สามารถรองรับการค้นหาด้วยคำมากกว่าหนึ่งคำ เพื่อเพิ่มความเจาะจงและช่วยให้ระบบสามารถระบุความหมายเชิงลึกได้อย่างถูกต้องและแม่นยำยิ่งขึ้น

1.2 วัตถุประสงค์

โครงการนี้มีวัตถุประสงค์เพื่อเสนอแนวทางในการพัฒนาให้ระบบการค้นหาคำศัพท์ที่มีคุณสมบัติที่ดีขึ้น โดยเพื่อให้บรรลุเป้าหมายดังกล่าว จึงกำหนดวัตถุประสงค์โครงการดังต่อไปนี้

1. ศึกษาและทดลองแนวทางในการพัฒนาระบบการค้นหาคำศัพท์ที่สามารถค้นหาคำศัพท์จากความคล้ายคลึงทางความหมายของคำได้ เพื่อเป็นพื้นฐานสำหรับการต่อยอดพัฒนาในอนาคต
2. ศึกษาและทดลองการจัดกลุ่มคำศัพท์จากความหมายที่มีความคล้ายคลึงกันของคำ เพื่อประเมินศักยภาพของการจัดหมวดหมู่ทางความหมายจากเทคนิค NLP เพื่อใช้เป็นพื้นฐานในการพัฒนาระบบในอนาคต
3. ลดระยะเวลาในการค้นหา จากการจัดกลุ่มคำศัพท์ด้วยเทคนิค K-means เพราะคำศัพท์ถูกจัดกลุ่มตามความคล้ายคลึงกันในเชิงความหมาย จึงสามารถช่วยลดเวลาในการประมวลผลและเพิ่มความรวดเร็วในการค้นหาคำศัพท์

1.3 ขอบเขตของโครงการ

โครงการนี้จะต้องสามารถลดความซับซ้อนในการค้นหาให้กับผู้ใช้งานได้ โดยเสนอแนวทางในการพัฒนาจากการค้นหาด้วยคำศัพท์ทั่วไป ไม่จำเป็นต้องใช้คำศัพท์เฉพาะในการค้นหา และคาดหวังว่าเมื่อทำการค้นหาคำ 1 คำนั้น จะขึ้นผลลัพธ์เป็นคำที่มีความหมายเกี่ยวข้องกันด้วย เพื่อให้สะดวกสบายต่อผู้ใช้งาน โดยใช้อัลกอริทึม BERT ในการพัฒนา และใช้คุณสมบัติของอัลกอริทึมช่วยดูบริบทของคำนั้น ในกรณีนำคำมาค้นหา 2 คำขึ้นไป

1.4 ประโยชน์ของโครงการ

1. ลดระยะเวลาในการค้นหาคำศัพท์
2. สามารถค้นหาคำศัพท์ด้วยคำศัพท์ทั่วไปได้ ไม่จำเป็นต้องเป็นคำศัพท์เฉพาะ
3. อำนวยความสะดวกสบายต่อผู้ใช้งานโดยการแสดงผลคำศัพท์ที่มีความหมายเกี่ยวข้องกับคำที่นำมาค้นหา

1.5 ข้อจำกัดของโครงการ

1. ผู้ใช้งานต้องใช้งานระบบผ่านเว็บเบราว์เซอร์
2. ผู้ใช้งานสามารถป้อนคำที่ต้องการค้นหาได้เพียง 1 ภาษาเท่านั้น คือภาษาอังกฤษ
3. ข้อมูลที่นำมาทดลอง เป็นข้อมูลที่นำมาแค่ 10000 แถวจาก 1000000 แถว เพื่อให้ทดลองกับเครื่องมือในการทดลองได้

บทที่ 2

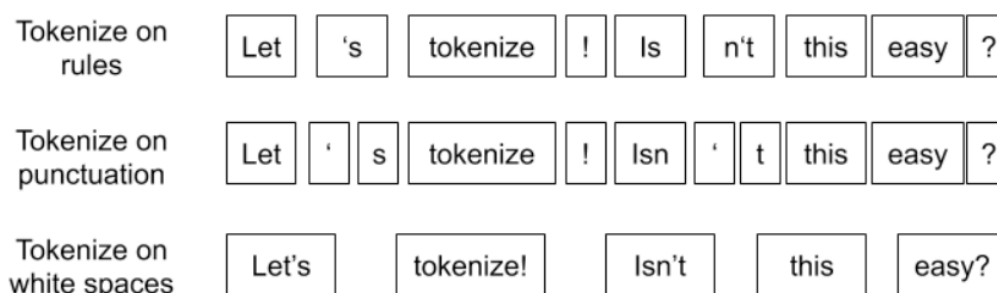
วรรณกรรมและงานวิจัยที่เกี่ยวข้อง

2.1 แนวคิดทฤษฎีที่เกี่ยวข้อง

2.1.1 NLP (Natural Language Processing)

คือการประมวลผลเพื่อให้คอมพิวเตอร์สามารถเข้าใจภาษามนุษย์ และสามารถสื่อสารกับมนุษย์ได้ โดยปัจจุบันมนุษย์มักใช้ NLP ในการประมวลผล วิเคราะห์ และจัดเก็บเอกสารขนาดใหญ่ วิเคราะห์ความเห็นของลูกค้า จัดประเภทและแยกข้อความกระบวนการทำงานของ NLP ได้แก่

1. การประมวลผลล่วงหน้า (Preprocessing) เป็นการจัดการข้อความให้อยู่ในรูปแบบที่คอมพิวเตอร์เข้าใจได้ง่ายขึ้น เช่น การทำ Tokenization ดังภาพ 2-1 แยกข้อความออกเป็นคำหรือประโยค [4], การทำ Stop word ลบคำที่ไม่มีความสำคัญ เช่น a, an, the, also, quite , การทำ Lemma แปลงคำให้อยู่ในรูปแบบรากศัพท์ เช่น วิ่งเล่น แปลงเป็น วิ่ง เป็นต้น , การทำ Normalization การปรับข้อความ เช่น เปลี่ยนตัวอักษรให้เป็นตัวพิมพ์เล็กทั้งหมด

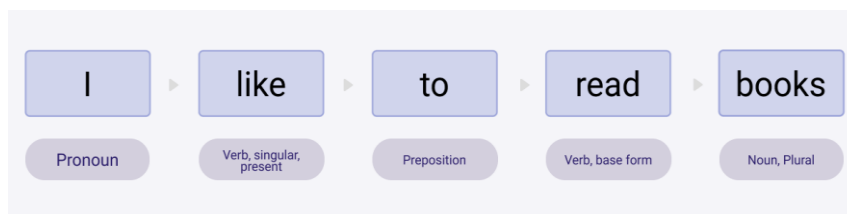


Let's tokenize! Isn't this easy?

ภาพที่ 2-1 ภาพแสดงการทำ Tokenization

2. การแปลงข้อความเป็นตัวแทนเชิงตัวเลข (Text Representation) ข้อความจะถูกแปลงให้อยู่ในรูปแบบตัวเลขที่คอมพิวเตอร์ สามารถเข้าใจและคำนวณได้ เช่น TF-IDF การชั่งน้ำหนักความสำคัญของคำในเอกสาร , การทำ Word Embedding การแปลงคำเป็นเวกเตอร์เชิงความหมาย เช่น Word2Vec , Doc2Vec

3. การวิเคราะห์ข้อมูล (Parsing and Analysis) การทำความเข้าใจโครงสร้างและความหมายของข้อความ เช่น Part of Speech Tagging (POS) ดังภาพที่ 2-2 การกำหนดชนิดคำ (noun, verb, adjective) [4] , Named Entity Recognitive (NER) การแยกข้อมูลเฉพาะเจาะจง เช่น ชื่อคน, สถานที่ , Dependency Parsing การวิเคราะห์โครงสร้างประโยค เช่น หาคำที่เป็นประธานและกริยา



ภาพที่ 2-2 ภาพแสดงการทำ POS Tagging

4. การเรียนรู้และวิเคราะห์ข้อมูลเชิงลึก (Deep Analysis) การใช้เทคนิค ML เพื่อทำความเข้าใจข้อความ เช่น การจัดหมวดหมู่ของข้อความ (Text Classification) , การสร้างข้อความใหม่ (Text Generation)

5. การนำผลลัพธ์ไปใช้งาน (Postprocessing) การนำผลลัพธ์ไปแสดงผลหรือประมวลผลต่อ เช่น การแนะนำคำค้นหา (Search Suggestion) , การสร้างข้อความตอบกลับ (Chatbots) , การวิเคราะห์ความรู้สึก (Sentiment Analysis)

2.1.2 Word Embedding

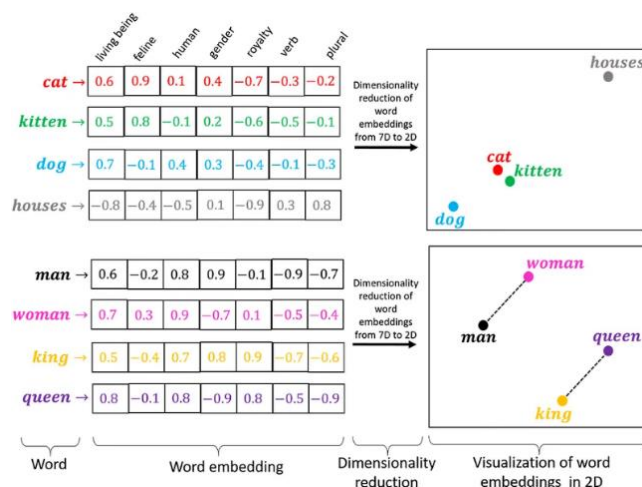
คือการแปลงคำเป็นตัวเลข โดยที่ผลลัพธ์ที่ได้จะออกมาในรูปแบบของเวกเตอร์ หมายความว่าคำหนึ่งคำสามารถแทนค่าได้ด้วยตัวเลขชุดนี้ จุดแตกต่างระหว่าง Word Embedding กับ TF-IDF คือ คำที่ถูกแปลงเป็นเวกเตอร์ของ TF-IDF จะเกิด sparse matrix (เมทริกซ์ที่ประกอบด้วยเลข 0) ได้ง่าย ตัวอย่างการทำ word embedding ดังภาพที่ 2-3 ในการนำคำว่า แอปเปิ้ล ส้ม สุนัข หมา ซึ่งหากเรามองจากลักษณะคำในความเป็นจริงแล้ว จะเห็นว่าคำที่สะกดแตกต่างกันนั้น อาจจะมี ความหมายที่คล้ายคลึงหรือเหมือนกัน เช่น คำว่า “หมา” และคำว่า “สุนัข” นอกจากนี้ ถึงแม้ว่าคำที่เรานำมาเปรียบเทียบกันอาจมีความหมายที่แตกต่างกัน แต่ระดับของความแตกต่างระหว่างคำคู่หนึ่งก็อาจมีค่าต่างจากความแตกต่างของคำอีกคู่หนึ่ง เช่นถ้าหากเราจะพิจารณาความหมายของคำว่า “ส้ม” กับคำว่า “แอปเปิ้ล” นั้น เราก็จะพบว่าคำทั้งสองมีความหมายที่ใกล้เคียงกันมากกว่าคำว่า “ส้ม” กับคำว่า “สุนัข” เป็นต้น [6]



ภาพที่ 2-3 ภาพแสดงการทำ word embedding

2.1.3 TF-IDF

คือการค้นหาความสำคัญของคำแต่ละคำภายในข้อความ และนำผลลัพธ์มาใช้เป็นตัวแทนของข้อความนั้นๆ ในรูปแบบของเวกเตอร์ หลักการของ TF-IDF คือพิจารณาข้อมูลโดยใช้ตัวคำภายในข้อความโดยตรง เช่น การเปรียบเทียบความคล้ายคลึงของข้อความจากคำที่ปรากฏภายในข้อความหรือการใช้การเปรียบเทียบสัดส่วนการมีอยู่ของจำนวนคำที่ไม่ซ้ำ (unique word) โดยให้ถือว่าคำคนละคำนั้นแตกต่างกันโดยสิ้นเชิง และไม่มีการนำความใกล้เคียงหรือเกี่ยวข้องกันของคำมาร่วมพิจารณาด้วยเลข ซึ่งในความเป็นจริงนั้นคำที่สะกดแตกต่างกัน อาจมีความหมายที่คล้ายคลึงกันหรือเหมือนกัน เช่น ตัวอย่างในภาพที่ 2-4 คำว่า “cat” และ “kitten” มีความหมายที่คล้ายคลึงกัน เพราะทั้งสองคำเกี่ยวข้องกับสัตว์ในตระกูลแมว แต่ถ้าเปรียบเทียบกับคำว่า “dog” จะพบว่าความสัมพันธ์ระหว่าง “cat” และ “kitten” มีความใกล้เคียงกันมากกว่า “cat” และ “dog” แม้ว่าทั้งหมดจะเกี่ยวข้องกับสัตว์ก็ตามเช่นเดียวกัน คำว่า “man” และ “woman” หรือ “king” และ “queen” ก็มีความสัมพันธ์ที่สะท้อนถึงมิติของเพศหรือสถานะในบริบทของมนุษย์ ซึ่งคำในแต่ละคู่มีความสัมพันธ์ใกล้เคียงกัน แต่ถ้าเปรียบเทียบ “man” กับ “queen” ความสัมพันธ์จะต่างไปและมีระยะห่างในเชิงความหมายมากกว่า การวัดความใกล้เคียงทางความหมายนี้ช่วยให้เราเข้าใจว่า คำที่มีความหมายคล้ายคลึงกัน เช่น “cat” และ “kitten” จะมีระยะห่างทางเวกเตอร์ที่สั้นกว่า เมื่อเทียบกับคำที่มีความแตกต่างในความหมายมาก เช่น “cat” และ “houses” หรือ “orange” และ “cat” ที่แทบไม่เกี่ยวข้องกันเลย [8]



ภาพที่ 2-4 ภาพแสดงการทำ Embedding Word Vector

แนวคิดของ TF-IDF คือถ้าหากคำคำไหนถูกพูดถึงอยู่บ่อยๆ ในเอกสารนั้นๆ มีความเป็นไปได้สูงว่าคำนั้นมีความเกี่ยวข้องกับใจความสำคัญของเอกสารนั้นๆ มาก เช่น ถ้าเลือกพิจารณาบทความเกี่ยวกับกาแฟ คำว่า “กาแฟ” และ “ชง” อาจจะได้เห็นได้หลายครั้งภายในบทความนั้น ค่าของ TF เป็นค่าที่บอกความถี่ของคำแต่ละคำที่ปรากฏในเอกสาร โดยคิดคำนวณจากการนำจำนวนครั้งที่คำนั้นๆ ปรากฏในเอกสารมาหารด้วยจำนวนคำทั้งหมดในเอกสาร

$$TF(\text{ของคำคำหนึ่ง}) = \frac{\text{จำนวนของคำนั้นๆ ในเอกสาร}}{\text{จำนวนของคำทั้งหมดในเอกสาร}}$$

ภาพที่ 2-5 สมการแสดง TF

Inverse Document Frequency (IDF) เป็นการคำนวณค่าน้ำหนัก (weight) ความสำคัญของแต่ละคำ โดยคำที่พบเจอได้บ่อยๆ จะมีค่า IDF ต่ำ บ่งบอกว่าคำเหล่านั้นไม่สามารถดึงเอาจุดเด่นของเอกสารที่คำเหล่านั้นปรากฏอยู่ออกมาได้ดี ค่า IDF สามารถคำนวณได้ด้วยสมการ

$$IDF(\text{ของคำคำหนึ่ง}) = \log \left(\frac{\text{จำนวนเอกสารทั้งหมดที่ใช้พิจารณา}}{\text{จำนวนเอกสารที่มีคำคำนั้นปรากฏอยู่}} \right)$$

ภาพที่ 2-6 สมการแสดง IDF

เมื่อนำค่าทั้งสองมารวมกัน จะได้ค่า TF-IDF โดยใช้สูตรคำนวณ

$$TFIDF = TF \times IDF$$

ภาพที่ 2-7 สมการแสดง TFIDF

TF-IDF จะทำงานภายใต้เงื่อนไขคือ คำที่ปรากฏบ่อยในเอกสารหนึ่ง แต่พบไม่บ่อยในเอกสารอื่น ๆ จะเป็นคำที่มีความสำคัญในการแยกความแตกต่างของเอกสารนั้น เทคนิคนี้จึงไม่คำนึงถึงลำดับหรือบริบทของคำ และถือว่าคำแต่ละคำมีความเป็นอิสระจากกัน ซึ่งเหมาะกับข้อมูลที่โฟกัสที่ความถี่ของคำในระดับเอกสารมากกว่าความหมายเชิงบริบท

2.1.4 Word2Vec

เป็นโมเดลที่ถูกพัฒนาโดย Tomas Mikolov และทีมงาน Google ในปี 2013 เพื่อใช้ในการแปลงคำเป็นเวกเตอร์ ที่สามารถนำไปใช้ในการประมวลผลภาษาธรรมชาติ(NLP) ได้อย่างมีประสิทธิภาพ โดยโมเดลนี้อาศัยการเรียนรู้จากบริบทของคำ เพื่อให้คำต่างๆ มีการแสดงผลเป็นเวกเตอร์ที่มีความหมายเหมือนกันในเชิงคณิตศาสตร์

วิธีการทำงานของ Word2Vec จะใช้หลักการเรียนรู้จากคำบริบทเพื่อสร้างเวกเตอร์ที่แทนคำที่มีลักษณะทางคณิตศาสตร์ โดยการฝึกโมเดลจะมีหลักการหลัก 2 แบบคือ Continuous Bag of Words (CBOW) และ Skip-Gram

1. Continuous Bag of Words (CBOW) วิธีนี้จะใช้คำที่อยู่รอบๆคำเป้าหมายมาเป็นข้อมูลในการทำนายค่ากลาง โดยจะคาดเดาค่ากลางจากคำที่อยู่ในบริบทใกล้เคียง ตัวอย่างในประโยค "the cat sits on the mat", ถ้าคำ "cat" เป็นค่ากลาง คำบริบท (context words) เช่น "the", "sits", "on", "mat" จะถูกใช้ในการทำนายค่ากลาง "cat" ผู้ใช้จะทำการสร้างเครื่องข่ายสมองแบบตื้น (shallow neural network) โดยจะใช้งาน word vector เป็น input layer ซึ่งเชื่อมต่อเข้ากับ hidden layer จำนวน 1 ชั้น (เราสามารถกำหนดจำนวน node ใน layer นี้ได้ตามต้องการ แต่โดยปกติจะมีจำนวนน้อยกว่าจำนวนมิติของเวกเตอร์ที่เป็น input ซึ่งก็คือเวกเตอร์ตัวแทนของคำ หรือ word vector นั้นเอง) และทำการต่อเข้าสู่ output layer ที่มีจำนวนมิติเท่ากับ Input layer จากนั้นจึงทำการฝึกฝนโมเดล (training) ในรูปแบบของการทำ classification

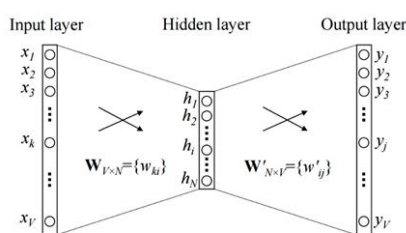


Figure 1: A simple CBOW model with only one word in the context

ภาพที่ 2-8 ภาพแสดงลักษณะของโครงสร้างของโมเดล CBOW เบื้องต้น

กล่าวคือสำหรับแต่ละคำในประโยคหรือข้อความที่จะวิเคราะห์นั้น ในขั้นตอนของการฝึกฝนโมเดลจะนำเอา ดังภาพที่ 2-8 word vectors ของคำที่อยู่รอบๆ คำดังกล่าวภายในระยะของบริบท (context size) ที่กำหนดมาใช้เป็น input ในการทำ classification และใช้ word vector ของคำที่กำลังพิจารณาซึ่งมีตำแหน่งอยู่ที่ตรงกลาง (center word) ของบริบทเป็นเป้าหมายในการทำนาย (output) [7]



ภาพที่ 2-9 ภาพแสดงตัวอย่างการทำ CBOW

ดังภาพที่ 2-9 เป็นภาพแสดงตัวอย่างการทำ CBOW คำว่า "อ่าน" (คำเป้าหมายหรือ Target Word) ถูกล้อมรอบด้วยคำบริบท (Context Words) ได้แก่ "ฉัน", "ชอบ", "หนังสือ", และ "มาก" ใน CBOW จะนำคำบริบท (เช่น "ฉัน", "ชอบ", "หนังสือ", "มาก") มาผสมผสาน (ผ่านกระบวนการ embedding และการหาค่าเฉลี่ย) จากนั้นโมเดลจะพยายามทำนายคำว่า "อ่าน" ซึ่งเป็นคำเป้าหมาย [6]

2. Skip-gram วิธีนี้ใช้คำกลางในการทำนายคำที่อยู่รอบๆ คำกลาง โดยวิธีนี้จะเน้นการทำนายคำบริบทจากคำกลาง ตัวอย่าง: ถ้าคำ "cat" เป็นคำกลาง โมเดลจะใช้คำนี้ในการทำนายคำที่อยู่ใกล้เคียงในประโยค เช่น "the", "sits", "on", "mat" โดยจะทำการสร้าง shallow neural network ในลักษณะเดียวกับที่ใช้ในงานโมเดล Continuous Bag of Words (CBOW) โดยมีขนาด input layer และ output layer เท่ากัน และมี hidden layer เป็นจำนวน 1 ชั้น เช่นเดียวกับที่แสดงในรูปที่ 1 แต่การฝึกฝนโมเดล Skip-gram นั้นจะต่างจากการโมเดล CBOW เพราะแทนที่จะใช้ word vectors ของคำต่างๆ ในบริบทของคำแต่ละคำมาเพื่อทำนายคำดังกล่าว โมเดล Skip-gram เลือกที่จะใช้คำหนึ่งๆ ในการทำนายคำทุกคำที่อยู่ในบริบทของคำนั้นแทน



ภาพที่ 2-10 ภาพแสดงตัวอย่างการทำ Skip-Gram

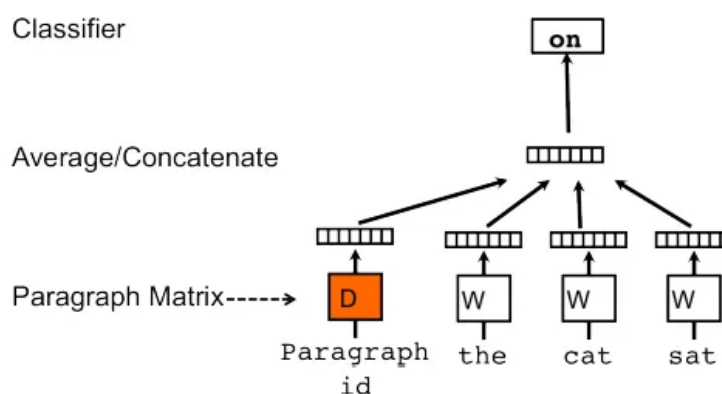
ดังภาพที่ 2-10 เป็นภาพแสดงตัวอย่างการทำ Skip-Gram โมเดลจะใช้คำตรงกลาง (Target Word) เพื่อทำนายคำบริบท (Context Words) รอบ ๆ คำว่า "อ่าน" (คำเป้าหมาย) ถูกใช้เป็นจุดศูนย์กลางในการทำนายคำรอบข้าง ได้แก่ "ฉัน", "ชอบ", "หนังสือ", และ "มาก" โมเดลก็จะทำการเรียนรู้ความสัมพันธ์ของคำว่า "อ่าน" กับคำที่เกี่ยวข้อง เช่น "อ่าน" มักเชื่อมโยงกับ "หนังสือ" "อ่าน" มักเกี่ยวข้องกับคำบ่งบอกความชอบ เช่น "ชอบ" [7]

Word2Vec อาศัยหลักการคือ คำที่มักปรากฏร่วมกันในบริบทเดียวกันมักมีความหมายที่คล้ายกัน กล่าวคือ หากคำสองคำมีบริบทแวดล้อมที่คล้ายกัน โมเดลจะเรียนรู้ให้เวกเตอร์ของคำนั้นมีทิศทางใกล้เคียงกัน ดังนั้น Word2Vec จึงเหมาะกับข้อมูลที่มีบริบทหลากหลาย และต้องการข้อมูลจำนวนมากในการฝึกฝนเพื่อให้ได้เวกเตอร์ที่แม่นยำ

2.1.5 Doc2Vec

เป็นเทคนิคที่พัฒนาโดย Quoc Le และ Tomas Mikolov ในปี 2014 ซึ่งเป็นส่วนขยายอย่างง่ายของ Word2Vec เพื่อขยายการเรียนรู้ของ Word Embedding จากระดับคำไปสู่ระดับประโยค และระดับเอกสาร ซึ่งเป็นการเปลี่ยนจากการเข้ารหัสคำ เป็นการเข้ารหัสข้อความหรือเอกสารทั้งหมด ให้อยู่ในรูปแบบเวกเตอร์

Doc2Vec ทำงานด้วยหลักการเดียวกับ Word2Vec ดังภาพที่ 2-11 ในการเข้ารหัสข้อมูลเพื่อสร้างเวกเตอร์ แต่ได้ทำการเพิ่ม Paragraph ID เข้ามาในกระบวนการสร้าง โดย Paragraph ID ถูกสร้างมาจากตัวเอกสาร และทำให้มีเลขเฉพาะตัวในการนำไปใช้วิเคราะห์ จากนั้นจะทำงานโดยการทำงานในลักษณะดังกล่าวเรียกว่า Distributed Memory Version of Paragraph Vector (PV-DM) [3]



ภาพที่ 2-11 ภาพแสดงตัวอย่างการทำงาน Doc2Vec ในรูปแบบ PV-DM

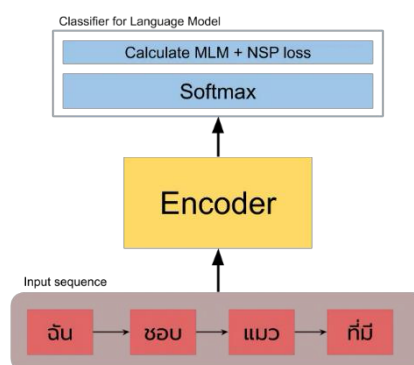
จุดเด่นของ Doc2Vec คือความสามารถในการแก้ปัญหาในด้านการจำกัดจำนวน Feature ที่ใช้สำหรับการเรียนรู้ของแบบจำลอง เช่น TF-IDF ที่จะต้องกำหนดจำนวนคำศัพท์ที่จะนำมาสร้างเวกเตอร์ โดย Doc2Vec มักใช้กับการหาความคล้ายคลึงของเอกสาร การจำแนกประเภทของเอกสาร และการประยุกต์เพื่อใช้งานด้านเครื่องจักรแปลภาษา

Doc2Vec ขยายแนวคิดจาก Word2Vec คือ เอกสารที่มีบริบทของคำที่คล้ายกัน จะมีความหมายที่คล้ายกัน และควรถูกแทนด้วยเวกเตอร์ในลักษณะเดียวกัน โมเดลจะเรียนรู้จากทั้งคำในเอกสารและรหัสเฉพาะของเอกสาร (Paragraph ID) ดังนั้น Doc2Vec จึงมีแนวคิดที่ว่า เอกสารที่มีคำหรือโครงสร้างประโยคคล้ายกันมักจะมีมีความหมายใกล้เคียงกัน ซึ่งเหมาะกับการจำแนกประเภทเอกสารหรือการวัดความคล้ายคลึงของบทความในระดับสูง

2.1.6 BERT (Bidirectional Encoder Representations from Transformers)

คืออัลกอริทึมของ Google ในปี 2018 เป็นเทคนิคการประมวลผลโครงสร้างเครือข่าย (NLP) ที่จะมียุทธศาสตร์ในการช่วยให้ Google เข้าใจและแสดงผลการค้นหาที่ตรงกับคำที่นำมาค้นหามากที่สุด

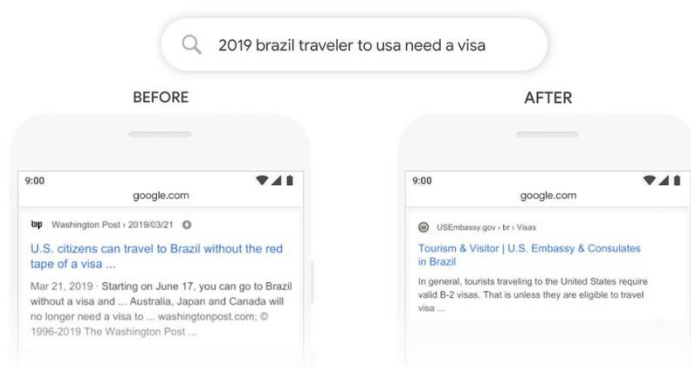
BERT ออกแบบให้เลือกเฉพาะส่วนที่เป็น encoder เพื่อให้สามารถเข้าใจข้อมูลในบริบทของข้อความ ซึ่งทำหน้าที่แปลงคำในประโยคให้เปลี่ยนเป็นเวกเตอร์ เพื่อให้ encoder สามารถทำหน้าที่เป็น Language Model (โมเดลที่พยายามคาดการณ์คำถัดไปจากคำก่อนหน้าที่มันเห็น เช่น "I am going to the" มันอาจจะคาดการณ์คำถัดไปว่า "store", "park", หรือ "office" ขึ้นอยู่กับบริบทที่มันเรียนรู้จากข้อมูลที่มี) ได้ BERT เพิ่มโมเดลอีก 1 ตัว ต่อจาก encoder ที่อยู่เดิม เพื่อทำหน้าที่ Classifier โดยนำเวกเตอร์ที่ได้จาก encoder ไปคำนวณต่อให้ได้คำตอบในรูปแบบคล้ายๆกับ Language Model ทั่วไป



ภาพที่ 2-12 ภาพแสดงตัวอย่างการทำงานของBERT

โดย BERT ขยายขนาดให้มีจำนวน Attention head (ช่วยให้โมเดลสามารถให้ความสนใจไปที่คำหรือส่วนต่าง ๆ ของข้อความที่เกี่ยวข้องกัน เพื่อให้สามารถเข้าใจบริบทได้ดียิ่งขึ้น) มากขึ้น มีจำนวน layer มากขึ้น และเพิ่มขนาด embedding vector เพื่อให้มั่นใจว่าโมเดลสามารถเรียนรู้คุณสมบัติทางภาษาให้ได้มากที่สุด

ดังรูปภาพที่ 2-12 ที่แสดงถึงการทำงานของ BERT โดย input layer มีคำว่า “ฉัน”, “ชอบ”, “แมว”, “ที่มี” คำเหล่านี้จะถูกแปลงเป็นเวกเตอร์ผ่าน Token Embeddings และใช้กลไก Self-Attention ในการทำ encoder เพื่อเรียนรู้บริบทของคำในประโยค ความพิเศษของ BERT คือ Bidirectional โดยจะเรียนรู้บริบทของคำจากทั้งซ้ายไปขวาและขวาไปซ้ายพร้อมกัน ทำให้เข้าใจความหมายเชิงลึกของคำในประโยค และใน Softmax layer ข้อมูลที่ทำการ encoder แล้วจะถูกมาทำนายความน่าจะเป็นของคำในประโยคหรือบริบทต่างๆ และทำการคำนวณ Loss function [6]



ภาพที่ 2-13 ภาพแสดงตัวอย่างการนำ BERT ไปใช้กับการค้นหา

จากภาพที่ 2-13 การคำค้นหาคือ "คนบราซิลจะไปอเมริกาต้องการ Visa สำหรับปี 2019" สิ่งที่คำค้นหานี้ต้องการคือข้อมูล หากเป็นเมื่อก่อนสิ่งที่ google จะแสดงคือข่าวเกี่ยวกับ 'คนอเมริกาที่ต้องการไปบราซิล' ซึ่งจะเห็นว่ามันไม่ตรงกับสิ่งที่ต้องการหาแม้หลายคีย์เวิร์ดจะเหมือนกัน แต่เมื่อมี BERT จะแสดงผลให้เห็นว่าผลลัพธ์คือเว็บของสถานทูตที่ให้ข้อมูลตรงการสิ่งที่มองหา พุดง่าย ๆ คือ BERT จะช่วยให้ Google วิเคราะห์และคิดเหมือนมนุษย์ยิ่งขึ้น ด้วยการศึกษานวนต่างๆ ของแต่ละคำหลัก [9]

ขั้นตอนการเทรน BERT แบ่งเป็น 2 ขั้นตอน คือ Pre-Training เป็นขั้นตอนการสอนเพื่อให้โมเดลเรียนรู้ โครงสร้างภาษาในภาพรวม ซึ่งจะสอนด้วยข้อมูลปริมาณมากๆ และตามด้วย Fine-tuning จะสอนโมเดลอีกครั้ง เพื่อให้โมเดลปรับตัวเองสำหรับปัญหาใดปัญหาที่ผู้พัฒนาต้องการ

2.1.6.1 Pre-Training

1. Masked Language Model หรือ MLM ถูกออกแบบมาเพื่อใช้แทนที่ Language Model แบบเดิม โดยในแต่ละรอบของการสอนโมเดล ประโยคจำนวนหนึ่งจะถูกป้อนเข้ามา โดยที่ประมาณ 15% ของคำทั้งหมดจะโดนลบออกไป (Masked words) และ สิ่งที่โมเดลต้องทำ คือ การเติมคำที่หายไปเหล่านั้นให้ถูกต้อง ซึ่งเพื่อเติมคำที่ถูกต้อง โมเดลต้องเข้าใจโครงสร้างของประโยคเช่นเดียวกับ Language Model แบบดั้งเดิม แต่ทั้งนี้โมเดลสามารถพิจารณาบริบทได้จากทุกคำที่อยู่รอบๆ

แต่ปัญหาหนึ่งที่เกิดขึ้น หลังจากโมเดลเรียนรู้ภายใต้ข้อมูลแบบ MLM ไปสักระยะหนึ่ง คือ โมเดลมักจะพยายามที่จะเดาคำที่หายไปเพียงอย่างเดียว จนไม่สนใจคำอื่นๆเลย (the model only tries to predict the [MASK] token) เวกเตอร์ของคำอื่นๆที่ได้จึงมีแนวโน้มว่า จะไม่มีข้อมูลที่มีประโยชน์ (might not be as rich as it could be) ตัวอย่างเช่น "The cat sat on the [MASK]." โมเดล MLM ถูกฝึกให้เดาคำที่หายไปในตำแหน่ง [MASK] โดยการคาดเดาว่า [MASK] น่าจะเป็นคำใดจากบริบทที่เหลืออยู่ เช่น "mat" หรือ "carpet". คำที่ไม่ถูกแทนที่ด้วย [MASK] (เช่น "The", "cat", "sat", "on", "the") จะไม่ได้รับความสนใจหรือการอัปเดตที่มากเท่าที่ควร เพราะโมเดลมักจะมุ่งไปที่การคาดเดา [MASK] เท่านั้น ทางออกที่ถูกนำเสนอขึ้นมา คือ Mixed Mask Strategy โดย ภายใน 15% ของคำทั้งหมดออกไป จะถูกแบ่งออกเป็น

- 80% ของคำทั้งหมด จะโดนแทนที่ด้วย [MASK] token
- 10% ของคำทั้งหมด จะโดนแทนที่ด้วยคำอื่นๆแบบสุ่ม
- 10% ของคำทั้งหมด จะไม่โดนเปลี่ยนแปลง

เนื่องจาก BERT จะคำนวณ loss จากเฉพาะในส่วนที่เป็น 15% ที่เลือกมาในรอบแรก และบางส่วนของคำถาม ไม่ใช่ [MASK] token อีกต่อไป โมเดลจึงต้องสนใจทุกๆ Token โดยการคงให้คำบางส่วนอยู่เหมือนเดิม จะช่วยให้โมเดลสามารถสร้างเวกเตอร์ที่ตรงตามคำที่เจอจริงๆได้ง่ายขึ้น (to bias the representation towards the actual observed word) และการแทนที่คำด้วยคำอื่นแบบสุ่มนี้ เป็นการเพิ่ม noise ให้กับโมเดล คล้ายๆกับการทำ regularization เพื่อไม่ให้โมเดลมั่นใจคำตอบตัวเองจนเกินไป อย่างไรก็ตาม การปรับสัดส่วนนี้ มีผลอย่างมากต่อประสิทธิภาพในภาพรวมของโมเดล [6] โดยเฉพาะ การแทนที่คำด้วยคำอื่น ซึ่งมีแนวโน้มจะทำให้โมเดลสับสน และ มีผลทำให้ประสิทธิภาพของโมเดลลดลง การเลือกสัดส่วนที่เหมาะสมจึงเป็นเรื่องสำคัญ ทั้งนี้ สัดส่วน 80-10-10 ที่เห็นนั้น เป็นสัดส่วนที่ดีที่สุดจากการทดสอบหลายๆครั้ง ซึ่งสามารถดูได้จากภาพที่ 2-14

Masking Rates			Dev Set Results		
MASK	SAME	RND	MNLI	NER	
			Fine-tune	Fine-tune	Feature-based
80%	10%	10%	84.2	95.4	94.9
100%	0%	0%	84.3	94.9	94.0
80%	0%	20%	84.1	95.2	94.6
80%	20%	0%	84.4	95.2	94.7
0%	20%	80%	83.7	94.8	94.6
0%	0%	100%	83.6	94.9	94.6

ภาพที่ 2-14 ภาพแสดงการทดสอบ Mask Strategy

2. Next Sentence Prediction หรือ NSP ถูกออกแบบมาเพื่อให้เรียนรู้ความสัมพันธ์ระหว่างประโยค โดยการป้อน 2 ประโยค จากนั้นให้โมเดลตัดสินใจว่าประโยคทั้ง 2 นี้เป็นประโยคที่อยู่ติดกันหรือไม่ โดยแบ่งข้อมูลครึ่งหนึ่ง เป็นคู่ของประโยคที่อยู่ติดกัน และอีกครึ่งหนึ่งเป็นประโยคที่โดนจับคู่กันแบบสุ่ม ตัวอย่างเช่น

ผลลัพธ์ที่เป็นประโยคคู่จริง (True pair)

- **ประโยคที่ 1:** "I went to the store to buy some groceries."
- **ประโยคที่ 2:** "I bought apples, bananas, and oranges."

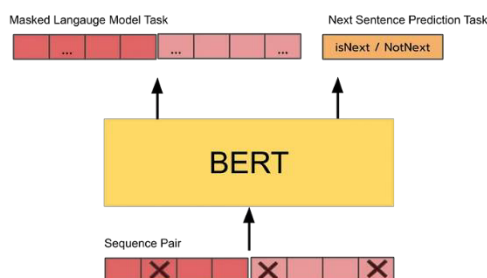
โมเดลจะทำนายว่า ประโยคที่สอง เป็นประโยคถัดไปจาก ประโยคแรก เพราะมันมีความเชื่อมโยงกันในบริบท (การไปที่ร้านและซื้อของ)

ผลลัพธ์ที่เป็นประโยคคู่เทียม (False pair)

- **ประโยคที่ 1:** "I went to the store to buy some groceries."
- **ประโยคที่ 2:** "The Eiffel Tower is located in Paris."

ในกรณีนี้ ประโยคที่สอง ไม่มีความเกี่ยวข้องกับ ประโยคแรก เลย ดังนั้นมันจะถูกพิจารณาว่าเป็น คู่เทียม (False pair)

วิธีการฝึกโมเดล ข้อมูลจะถูกเตรียมในลักษณะ คู่ประโยคที่เชื่อมโยง (True pairs) จะให้สองประโยคที่มีความเชื่อมโยงในเชิงบริบทหรือความหมาย คู่ประโยคที่ไม่เชื่อมโยง (False pairs) จะให้สองประโยคที่ไม่มีความเกี่ยวข้องกัน



ภาพที่ 2-15 ภาพแสดงการทำ Next sentence prediction

ดังภาพที่ 2-15 BERT แก้ไขปัญหานี้ด้วยการเพิ่มโจทย์ Next Sentence Prediction หรือ NSP [6] ไปพร้อมๆกับการทำ MLM เพื่อให้โมเดลเรียนรู้ความสัมพันธ์ระหว่างประโยค โดยการป้อน 2 ประโยค จากนั้นให้โมเดลตัดสินใจว่า ประโยคทั้ง 2 นี้เป็นประโยคที่อยู่ติดกันหรือไม่? โดยแบ่งข้อมูลครึ่งหนึ่ง เป็นคู่ของประโยคที่อยู่ติดกัน และอีกครึ่งหนึ่งเป็นประโยคที่โดนจับคู่กันแบบสุ่ม

ในแต่ละรอบของการสอน BERT จะรับคู่ของประโยค ซึ่งจะมีค่าบางส่วนหายไป จากนั้น BERT ก็พยายามทายคำในช่องว่าง พร้อมกับทายว่าประโยคทั้ง 2 อันที่ได้รับมา เป็นประโยคที่อยู่ติดกันหรือไม่ แล้วนำคำตอบที่ได้จากทั้ง 2 คำถามไปคำนวณความถูกต้อง และอัปเดตโมเดล สิ่งนี้ทำให้ BERT สามารถเรียนรู้ความสัมพันธ์ทั้งในระดับคำ และระดับประโยคไปพร้อมๆกัน

เพื่อเตรียมข้อมูลให้พร้อมสำหรับ MLM และ NSP tasks ประโยคทั้งหมดจะถูกจับคู่ จากนั้น ในแต่ละคู่ของประโยค จะโดนนำเข้าไปยังขั้นตอนการแปลงไปเป็นเวกเตอร์ โดยเริ่มต้นจากการเพิ่ม tokens พิเศษ 2 ตัว คือ

- [CLS] token (Classification token) เพื่อใช้แทนตำแหน่งเริ่มต้นของข้อมูล
- [SEP] token (Separator token) คั่นกลางระหว่างประโยคทั้ง 2 เพื่อกำหนดขอบเขตระหว่างประโยคที่ 1 และ ประโยคที่ 2

ตัวอย่าง เช่น

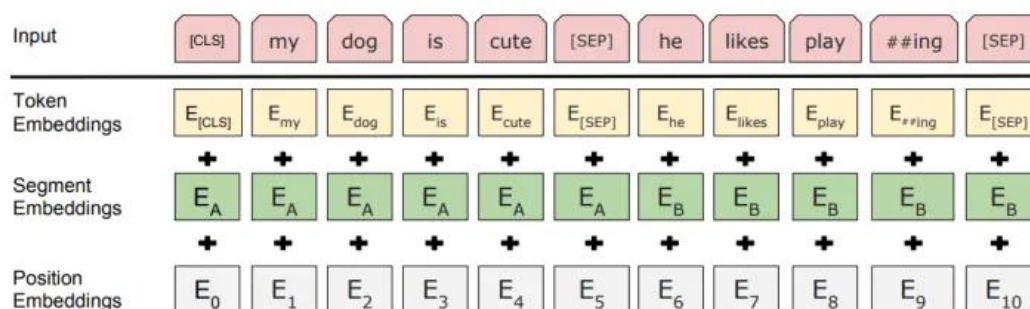
ประโยคที่ 1: "I went to the store."

ประโยคที่ 2: "I bought some milk."

เมื่อนำมารวมกันใน BERT จะเป็น "[CLS] I went to the store. [SEP] I bought some milk. [SEP]"

จากนั้น tokens ทั้งหมด จะถูกนำมารวมกันให้กลายเป็นก้อนของตัวเลข โดยผ่านการ embedding 3 ตัว ดังภาพที่ 2-16 คือ

1. Token embeddings เป็น Trainable Parameters ซึ่งระหว่างการเทรนโมเดล Embedding ชั้นนี้จะค่อยๆปรับตัวเอง เพื่อแทนข้อมูลในระดับคำของแต่ละ token
2. Positional embeddings เทคนิคเดียวกับที่ใช้ใน Transformer เพื่อเพิ่มข้อมูลเกี่ยวกับตำแหน่งของคำต่างๆ ให้กับโมเดล
3. Segment embeddings เป็น Embedding ใหม่ ที่เพิ่มเข้ามาใน BERT เพื่อช่วยให้โมเดลสามารถแยกแยะ ระหว่างประโยคที่ 1 และ ประโยคที่ 2 ที่อยู่ในข้อมูล [6]



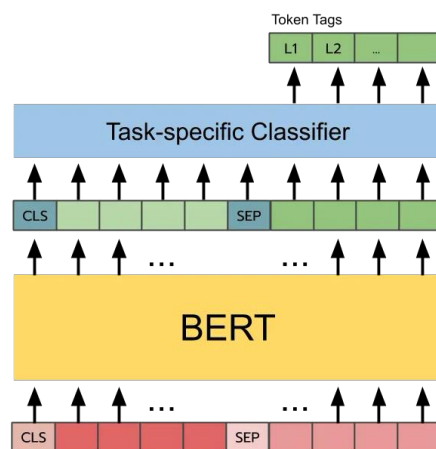
ภาพที่ 2-16 ภาพแสดงการทดสอบ Mask Strategy

2.1.6.2 Fine-Tuning

หลังจากที่ได้ Pre-trained BERT แล้วขั้นตอนสุดท้ายที่ขาดไม่ได้ คือ การ Fine-tuning หรือ การปรับรายละเอียดของโมเดลให้เหมาะสมกับงานที่ต้องการ ซึ่งในเชิงปฏิบัติ Fine-tuning คือ การสอนโมเดลอีกรอบด้วยข้อมูลที่ใช้สำหรับงานตามที่ต้องการ

กลุ่มปัญหาระดับคำ หรือ Token-level tasks ปัญหาในกลุ่มนี้ ส่วนใหญ่มักอยู่ในรูปแบบของ multiclass/binary classification เช่น

- Named Entity Recognition หรือ NER ซึ่งต้องการหาตำแหน่ง และจัดหมวดหมู่ของกลุ่มคำที่เป็น ชื่อบุคคล ชื่อบุคคล สถานที่ ตัวอย่างบททดสอบสำหรับปัญหานี้ คือ CoNLL-2003 NER
- POS tagging ซึ่งต้องการระบุประเภทของคำ เช่น Noun, Verb, Adjective เป็นต้น

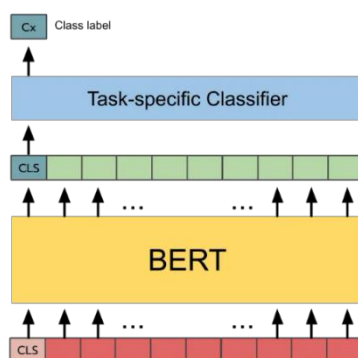


ภาพที่ 2-17 ภาพแสดงการทำ fine-tuning เพื่อแก้ปัญหาที่อยู่ในระดับคำ

BERT สามารถ Fine-tuning เพื่อแก้ปัญหาในกลุ่มนี้ คือปัญหาที่อยู่ในระดับคำ ดังภาพที่ 2-17 โดยการนำเวกเตอร์แต่ละคำที่ได้จาก BERT ส่งต่อไปให้ Classifier ประมวลผลต่อเพื่อทำนายว่า คำนี้อยู่ในกลุ่มอะไร จากนั้นนำคำตอบที่ได้ไปอัปเดตโมเดลต่อไปทั้งในส่วนที่เป็น Classifier และ BERT และทำซ้ำไปเรื่อยๆ จนกว่าจะได้ระดับความแม่นยำที่ต้องการ โดยการทำ Fine-tuning ในปัญหาประเภทนี้ จะเป็นการนำเวกเตอร์ที่เป็นผลลัพธ์ของ [CLS] token ไปใส่ Classifier เพื่อคำนวณคำตอบ [6]

ตัวอย่างของปัญหากลุ่มนี้ เช่น

- Semantic Textual Similarity: ต้องการให้คะแนนความเหมือนกันในเชิงความหมายระหว่างประโยค
- Natural Language Inference: ต้องการระบุว่าประโยคที่ได้รับมาเป็นเหตุเป็นผลกันหรือไม่ (เป็นเหตุเป็นผลกัน, ขัดแย้งกัน, ไม่เกี่ยวข้องกัน)
- Sentiment Analysis: ต้องการระบุว่าประโยคที่ได้รับมามีความหมายในแง่บวก หรือ ลบ



ภาพที่ 2-18 ภาพแสดงการทำ fine-tuning เพื่อแก้ปัญหาที่อยู่ในระดับประโยค

จากภาพที่ 2-18 ปัญหาในกลุ่มนี้ มักต้องการการประมวลผลภาพรวมของทั้งประโยค ดังนั้น BERT จึงเลือกใช้ [CLS] token (ซึ่งจะโดนใส่เป็น token แรกของทุกๆ ประโยค ทุกครั้งก่อนเริ่มประมวลผลประโยคนั้นๆ) เป็นตัวแทนของภาพรวมที่ว่า

โดยในการ Fine-tuning ในปัญหาประเภทนี้ จะเป็นการนำเวกเตอร์ที่เป็นผลลัพธ์ของ [CLS] token ไปใส่ Classifier เพื่อคำนวณคำตอบ [6]

BERT มีแนวคิดที่ว่าความหมายของคำขึ้นอยู่กับบริบทที่คำปรากฏอยู่ โดยโมเดลสามารถเข้าใจความหมายของคำจากทั้งด้านซ้ายและขวาในประโยคพร้อมกัน (Bidirectional Context) ซึ่งต่างจากเทคนิคอื่น ๆ ที่ไม่พิจารณาลำดับหรือบริบทอย่างลึกซึ้ง อย่างไรก็ตาม BERT ต้องการทรัพยากรในการฝึกฝนสูง และเหมาะกับข้อมูลที่ต้องการความเข้าใจระดับบริบทในระดับคำหรือประโยค

2.1.7 Cosine Similarity

คือการวัดความคล้ายคลึงกันระหว่างสองเวกเตอร์ในเชิงมุม โดยไม่คำนึงถึงขนาดของเวกเตอร์ มักใช้ในการคำนวณว่าเวกเตอร์ทั้งสองนั้นใกล้เคียงกันหรือไม่ใน vector space หรือในกรณีที่ใช้ในการวิเคราะห์ข้อความ Cosine Similarity ก็จะใช้วัดความคล้ายคลึงกันระหว่างข้อความที่เป็นตัวแทนด้วยเวกเตอร์

สูตรของ Cosine Similarity คำนวณจากการคูณจุด (dot product) ของเวกเตอร์สองตัว แล้วหารด้วยผลคูณของความยาวของทั้งสองเวกเตอร์ โดยสูตรมาจากกฎสามเหลี่ยม

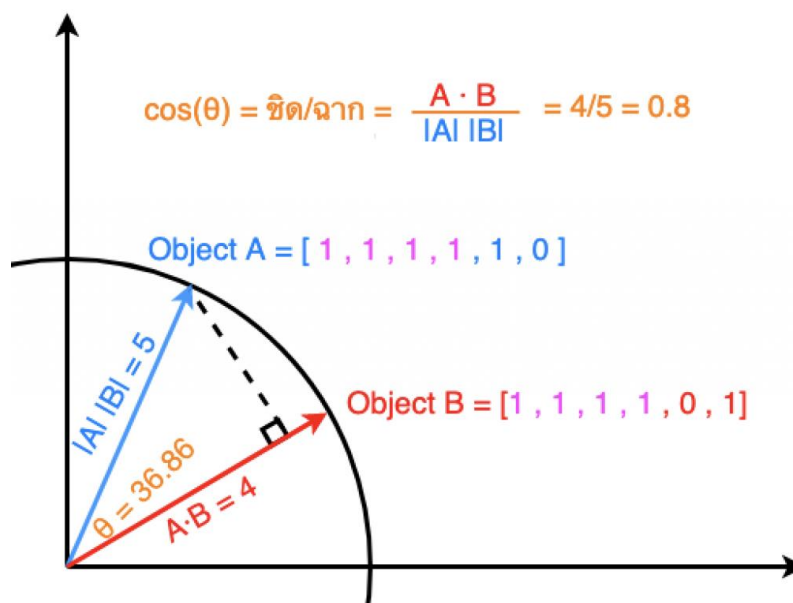
$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{||A|| ||B||} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}}$$

ภาพที่ 2-19 สมการของ cosine similarity

จากสมการอธิบายได้ดังนี้

- $A \cdot B$ คือผลคูณจุดของเวกเตอร์ A และ B
- $||A||$ และ $||B||$ คือความยาว (magnitude) ของเวกเตอร์ A และ B

ความหมายของผลลัพธ์ ถ้าค่าของ cosine similarity ที่ได้เป็น 1 แปลว่าเวกเตอร์ทั้งสองมีความคล้ายคลึงกันมาก แต่ถ้าค่าของ cosine similarity ที่ได้เป็น 0 แปลว่าเวกเตอร์ทั้งสองไม่มีความคล้ายคลึงกันเลย และถ้าค่าของ cosine similarity ที่ได้เป็น -1 แปลว่าเวกเตอร์ทั้งสองไม่มีความคล้ายคลึงกันและไม่มีความใกล้เคียงกันเลย



ภาพที่ 2-20 ภาพแสดงตัวอย่างการหา cosine similarity

จากรูปที่ 2-20 จะเป็นตัวอย่างของคำว่า “ยินดีที่ได้รู้จักครับ” และ “ยินดีที่ได้รู้จักค่ะ” โดยจะทำการตัดคำทั้งหมดก่อนให้เป็น [“ยินดี” , “ที่” , “ได้” , “รู้จัก” , “ครับ” , “ครับ”] และ [“ยินดี” , “ที่” , “ได้” , “รู้จัก” , “ครับ” , “ค่ะ”] เมื่อนำมาเขียนเป็น array จะได้ดังนี้

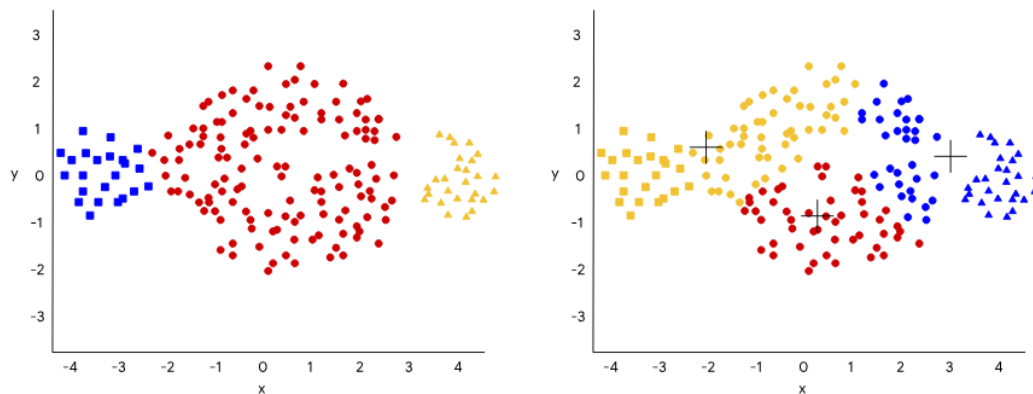
- Object A = “ยินดีที่ได้รู้จักครับ” = [1 , 1 , 1 , 1 , 1 , 0]
- Object B = “ยินดีที่ได้รู้จักค่ะ” = [1 , 1 , 1 , 1 , 0 , 1]
- $A \cdot B = A \cap B = 4$
- $|A||B| = \sqrt{(1^2+1^2+1^2+1^2+1^2+0)} * \sqrt{(1^2+1^2+1^2+1^2+0+1^2)} = 5$
- ค่า $\cos(\theta) = 0.8$ แปลว่า 2 เวกเตอร์นี้ มีความคล้ายคลึงกันอยู่

2.1.8 K-Means

คือวิธีการในการทำ Data Mining เป็นการเรียนรู้แบบไม่ต้องการ train และ test ก่อน หน้าที่หลักของ K-Means คือ การแบ่งกลุ่มแบบ clustering ซึ่งการแบ่งกลุ่มในลักษณะนี้จะใช้พื้นฐานทางสถิติ[2] วิธีการทำของ K-means คือ

1. กำหนดจำนวนกลุ่มขึ้นมาก่อนเช่น 2 กลุ่มหรือหมายความว่าค่า $K=2$ (กำหนดเป็น $C1$ และ $C2$) และสุ่มตำแหน่งแกน x,y ให้กับ $C1$ และ $C2$ จะได้ $C1(x1,y1)$ และ $C2(x2,y2)$
2. ดูตำแหน่งของสมาชิกแต่ละสมาชิกว่าอยู่ใกล้ใครมากกว่ากันก็ให้คนนั้นเป็นสมาชิกของ C นั้น จากตรงนี้เราจะรู้แล้วว่าสมาชิกแต่ละคนอยู่ในกลุ่มใดระหว่าง $C1$ และ $C2$

3. ปรับ x, y ของ C_1 และ C_2 ใหม่ให้อยู่ตรงกลางของกลุ่ม
4. ทำตามข้อ 2 และข้อ 3 อีกครั้งจนกว่า C_1 และ C_2 ตำแหน่งไม่เปลี่ยน



ภาพที่ 2-21 ภาพแสดงตัวอย่าง K-Means

$$\min_{C_1, \dots, C_K} J = \sum_{j=1}^K \sum_{x_i \in C_j} \|x_i - \mu_j\|^2$$

ภาพที่ 2-22 ภาพสมการของ K-means

จากภาพที่ 2-22 อธิบายสูตรได้ดังนี้

- C_j คือชุดของจุดข้อมูลที่อยู่ใน cluster ที่ j
- $\mu_j = \frac{1}{|C_j|} \sum_{x_i \in C_j} x_i$ คือจุดกึ่งกลางของคลัสเตอร์ j

ในทางปฏิบัติ ขั้นตอนของ K-means คือ

1. สุ่มเลือก จุดกึ่งกลางเริ่มต้น μ_1, \dots, μ_K
2. จัดกลุ่ม แต่ละจุด x_i ให้อยู่ใน cluster ที่มี μ_j ใกล้ที่สุด
(minimize $\|x_i - \mu_j\|^2$)
3. ปรับปรุง จุดกึ่งกลางของแต่ละ cluster ใหม่โดยคำนวณค่าเฉลี่ยของจุดที่ถูกจัดมาอยู่ cluster นั้น
4. ทำซ้ำขั้นตอน 2-3 จนไม่เกิดการเปลี่ยนแปลงการจัดกลุ่มหรือจุดกึ่งกลาง (convergence)

วัตถุประสงค์หลักของ K-Means คือ การลดความแปรปรวนภายใน cluster หรือกลุ่มที่แบ่งไว้ให้น้อยที่สุด ทำให้ข้อมูลที่อยู่ในแต่ละ cluster มีความคล้ายกันสูงและแตกต่างจาก cluster อื่นๆมากที่สุด [9]

2.1.8.1 Silhouette Score

เป็นเทคนิคที่ใช้วัด Instance นั้นมีความเหมือนกับ Cluster ที่จุดนั้นอยู่มากเพียงใด เมื่อเทียบกับ Cluster อื่นๆ ค่าของ Silhouette Score จะอยู่ในช่วง -1 ถึง 1 ยิ่งมีค่ามาก แสดงว่า Instance มีความคล้ายกับ Cluster ของมันมาก และมีความคล้ายกับ Cluster อื่นน้อย [13]

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \quad s(i) = \begin{cases} 1 - \frac{a(i)}{b(i)}, & \text{if } a(i) < b(i) \\ 0, & \text{if } a(i) = b(i) \\ \frac{b(i)}{a(i)} - 1, & \text{if } a(i) > b(i) \end{cases}$$

ภาพที่ 2-23 ภาพสมการของ K-means

จากภาพที่ 2-23 อธิบายสูตรได้ดังนี้

- $a(i)$ = ค่าเฉลี่ยของระยะทาง (distance) จากจุด i ไปยังสมาชิกทุกจุดใน cluster เดียวกัน (“intra-cluster distance”)
 - $b(i)$ = ค่าต่ำสุดของค่าเฉลี่ยระยะทางจากจุด i ไปยังสมาชิกใน cluster อื่นๆ (“nearest-cluster distance”)
 - กรณี $a(i) > b(i)$
ระยะภายใน cluster น้อยกว่า ระยะไป cluster ที่ใกล้ที่สุด \Rightarrow จุด i ถูกจัดกลุ่มได้ดี $\rightarrow s(i)$ เป็นบวก (สูงสุด +1)
 - กรณี $a(i) = b(i)$
ระยะในกับระยะออกเท่ากัน \Rightarrow จุด i อยู่ก้ำกึ่งระหว่างสอง cluster $\rightarrow s(i) = 0$
 - กรณี $a(i) < b(i)$
ระยะภายใน cluster มากกว่า ระยะไป cluster อื่น \Rightarrow จุด i น่าจะอยู่ผิด cluster $\rightarrow s(i)$ เป็นลบ (ต่ำสุด -1)

2.1.8.2 Euclidean Distance

คือระยะทางระหว่างจุด 2 จุดที่สั้นที่สุดในพื้นที่ n มิติ โดยใช้ทฤษฎีบทพีทาโกรัสในการวัดระยะทางระหว่างจุด 2 จุด โดยเปรียบเทียบได้กับการวัดระยะตรงจากจุดหนึ่งไปยังอีกจุดหนึ่งในงาน NLP วิธีนี้จะถูกใช้เพื่อวัดระยะทางระหว่างเวกเตอร์ของข้อความ กับจุดศูนย์กลางของแต่ละ cluster เพื่อกำหนดว่าข้อความนั้นควรอยู่ในกลุ่มใด [11]

$$d = \sqrt{\sum_{i=1}^n (q_i - p_i)^2}$$

ภาพที่ 2-24 ภาพสมการของ Euclidean Distance

จากภาพที่ 2-24 อธิบายสูตรได้ดังนี้

- d คือระยะทางระหว่างจุด 2 จุด
- n คือจำนวนมิติของเวกเตอร์
- p_i คือค่าของเวกเตอร์จากจุดที่ 1 ณ มิติที่ i
- q_i คือค่าของเวกเตอร์จากจุดที่ 2 ณ มิติที่ i
- $(q_i - p_i)^2$ คือความต่างระหว่างค่าของแต่ละมิติของเวกเตอร์ (ยกกำลังสองเพื่อให้ได้ค่าบวก)

2.2 เครื่องมือ (Tools)

2.2.1 Google Colab (Google Colaboratory)

Google Colab เป็นบริการ cloud ฟรีที่ให้ผู้ใช้งานสามารถเขียนและรันโค้ด python ผ่านเว็บเบราว์เซอร์ได้ โดยไม่ต้องติดตั้งซอฟต์แวร์ โดยใช้รูปแบบของ Jupyter Notebook (Software as a service-SaaS) [5]

คุณสมบัติของ Google Colab

1. ผู้ใช้งานสามารถเข้าถึง GPU และ TPU ได้ฟรี ซึ่งเป็นประโยชน์อย่างมากสำหรับการฝึกโมเดล Machine Learning และ Deep Learning ที่ต้องการพลังการประมวลผลสูง ปัจจุบันมี GPU ให้ผู้ใช้งานได้นำไปใช้งานดังนี้

- Nvidia Tesla K80
- Nvidia Tesla T4

- Nvidia Tesla P100
- 2. สามารถแชร์และทำงานร่วมกับผู้อื่นได้อย่างสะดวกสบาย
- 3. สามารถเชื่อมต่อและจัดเก็บไฟล์ใน Google Drive ได้
- 4. มีไลบรารียอดนิยมสำหรับ Data Science และ Machine Learning ติดตั้งไว้แล้ว เช่น TensorFlow, PyTorch, Scikit-learn เป็นต้น
- 5. สามารถนำเข้าข้อมูลจากหลายแหล่ง เช่น Google Drive, GitHub หรืออัปโหลดโดยตรงจากเครื่องคอมพิวเตอร์ของผู้ใช้งาน



ภาพที่ 2-24 ภาพแสดงสัญลักษณ์ของ Google Colab

2.3 งานวิจัยที่เกี่ยวข้อง

2.3.1 Automatic Retrieval and Clustering of Similar Words

งานวิจัย “Automatic Retrieval and Clustering of Similar Words” โดยคุณ Dekang Lin. (1998) เป็นการกล่าวถึงการเรียนรู้ความหมายของคำจากบริบท โดยการใช้เทคนิคการประมวลผลจากภาษาธรรมชาติ เพื่ออนุมานความหมายของคำที่ไม่รู้จักจากบริบทที่คำเหล่านั้นปรากฏอยู่ ตัวอย่างคำที่นำมา คือ (Nida, 1975, p.167): “ A bottle of tezgiiino is on the table. Everyone likes tezgiiino. Tezgiiino makes you drunk. We make tezgiiino out of corn.” คำว่า “tezgiiino” ไม่มีความหมาย แต่บริบทที่มีการใช้งานคำนี้ในประโยคช่วยให้สามารถอนุมานได้ว่า “tezgiiino” แปลว่าเครื่องดื่มแอลกอฮอล์ที่ทำจากข้าวโพด งานวิจัยนี้มีหลักการอยู่คือ

1. การอนุมานความหมายของคำจากบริบท เริ่มต้นด้วยตัวอย่างคำว่า “tezgiiino” ซึ่งไม่รู้จักในภาษาอังกฤษ แต่สามารถอนุมานได้จากบริบทในประโยคที่ปรากฏ เช่น “tezgiiino makes you drunk” และ “tezgiiino is made from corn” ซึ่งบ่งชี้ว่า tezgiiino น่าจะเป็น เครื่องดื่มแอลกอฮอล์ ที่ทำจากข้าวโพด การเรียนรู้ความหมายของคำจากบริบทในลักษณะนี้เป็นหัวใจสำคัญในการเรียนรู้ภาษา

2. ความคล้ายคลึงของคำ งานวิจัยนี้กล่าวถึงความสำคัญของ ความคล้ายคลึง (similarity) ในการเรียนรู้คำใหม่จากบริบทที่มันปรากฏในข้อความ การหาคำที่มีความคล้ายคลึงช่วยในการเข้าใจ

ความหมายของคำ โดยใช้คำที่มีลักษณะหรือการใช้งานคล้ายกัน เช่น การเชื่อมโยงคำว่า tezgiiino กับคำว่า "beer", "wine", "vodka" ในบริบทของเครื่องดื่มแอลกอฮอล์

3. การสร้าง thesaurus โดยใช้คำที่คล้ายคลึงจากข้อมูล (corpus) ที่มีอยู่ ซึ่งช่วยให้สามารถสร้างคำพ้องที่เหมาะสมกับประเภทของข้อมูลหรือเนื้อหาที่ใช้งาน เช่น การสร้าง thesaurus ที่เฉพาะเจาะจงในวงการหรือประเภทของเอกสาร (เช่น ข่าวสาร) เพื่อหลีกเลี่ยงคำที่ไม่ใช้งานบ่อยในบางประเภทของข้อมูล

4. การแก้ปัญหาข้อมูลขาดแคลน (Data Sparsity) งานวิจัยนี้ยังกล่าวถึงการใช้ similarity-based smoothing เพื่อแก้ปัญหาของข้อมูลที่มีคำที่พบบ่อยน้อยในข้อมูล ซึ่งอาจส่งผลให้ไม่สามารถคำนวณความน่าจะเป็นได้อย่างแม่นยำ เช่นในการ word sense disambiguation (การเลือกความหมายที่ถูกต้องของคำที่มีหลายความหมาย)

งานวิจัยนี้เสนอ วิธีการในการใช้ similarity measures เพื่อนำไปใช้ในการสร้าง thesaurus, แก้ปัญหาความขาดแคลนของข้อมูล (data sparsity), และ ปรับปรุงการตัดสินใจในงานประมวลผลภาษาธรรมชาติ เช่นการระบุความหมายของคำในบริบทต่าง ๆ หรือการทำ word sense disambiguation ซึ่งทำให้การทำงานกับข้อมูลที่มีคำที่พบบ่อยน้อยหรือคำที่ไม่รู้จักในข้อมูลมีความแม่นยำมากขึ้น

2.3.2 The application of NLP in information retrieval

งานวิจัย “The application of NLP in information retrieval” โดยคุณ Xurui Wang. เป็นการกล่าวถึงการพัฒนาและการประยุกต์ใช้ NLP ในการดึงข้อมูล (Information Retrieval) ในหลาย ๆ ด้าน ได้แก่ การค้นหาข้อมูลทางวิชาการ, การสกัดความรู้ทางการแพทย์, การค้นหาข้อมูลการเดินทาง, และการให้บริการด้านE-Commerce โดยเน้นการวิวัฒนาการของเทคนิค NLP จากการจับคู่คำที่ตรงไปตรงมา (keyword matching) ไปสู่การใช้ Deep Learning และ Machine Learning เพื่อเพิ่มประสิทธิภาพในการเข้าใจความหมายเชิงลึกและความตั้งใจของผู้ใช้

งานวิจัยกล่าวถึงการพัฒนาเปลี่ยนแปลงในประวัติศาสตร์ของ NLP ที่เริ่มจากการใช้ keyword matching ซึ่งทำให้สามารถค้นหาข้อมูลจากคำที่ตรงกับคำค้นหาไปจนถึงการใช้ deep learning และ semantic understanding ในการดึงข้อมูลที่สามารถจับความหมายของคำและบริบทได้ ทั้งนี้ การพัฒนาในแต่ละช่วงมีข้อดีและข้อจำกัดที่แตกต่างกันไป โดยสะท้อนให้เห็นถึงการพัฒนาของความต้องการของผู้ใช้ที่เปลี่ยนแปลงไป

การประยุกต์ใช้ NLP การค้นหาข้อมูลทางวิชาการ (Academic Information Retrieval): NLP ช่วยในการจัดหมวดหมู่เอกสารวิจัยและค้นหาข้อมูลที่เกี่ยวข้อง

การสกัดความรู้ทางการแพทย์ (Medical Knowledge Extraction): NLP ช่วยในการสร้างกราฟความรู้ทางการแพทย์และพัฒนาระบบถาม-ตอบที่ช่วยในการปรับปรุงผลลัพธ์ด้านสุขภาพ

การวางแผนการเดินทาง (Travel Planning): NLP ช่วยให้การค้นหาข้อมูลท่องเที่ยวที่ปรับให้เหมาะสมกับผู้ใช้เป็นไปอย่างสะดวก

บริการ E-Commerce: (NLP ช่วยในการแนะนำสินค้า, การวิเคราะห์ความคิดเห็นของผู้ใช้, และการบริการลูกค้าอัตโนมัติ)

NLP กำลังเปลี่ยนแปลงวิธีที่เราเข้าถึงและโต้ตอบกับข้อมูล โดยมุ่งเน้นที่การทำความเข้าใจและการปรับปรุงประสบการณ์การค้นหาข้อมูลให้ตรงกับความต้องการของผู้ใช้ในโลกดิจิทัลที่ขยายตัวอย่างรวดเร็ว งานวิจัยนี้แสดงให้เห็นถึงศักยภาพของ NLP ในการพัฒนาผลลัพธ์การค้นหาและบริการที่มีความแม่นยำและปรับให้เหมาะสมกับแต่ละบุคคล

2.3.3 Clustering articles based on semantic similarity

งานวิจัย “Clustering articles based on semantic similarity” โดยคุณ Shenghui Wang และ Rob Koopman. เป็นการกล่าวถึงการจัดกลุ่มบทความวิจัย โดยใช้ ความคล้ายคลึงทางความหมาย (semantic similarity) ซึ่งเป็นส่วนหนึ่งของกระบวนการผลิตความรู้ทางวิทยาศาสตร์ โดยมุ่งเน้นการใช้ ข้อมูลจากหลายส่วนของบทความ เช่น ผู้เขียน, วารสารที่ตีพิมพ์, หัวข้อเรื่อง, และการอ้างอิง เพื่อสร้าง การแทนที่ความหมาย (semantic representation) ของบทความและใช้มันในการจัดกลุ่ม (clustering) บทความที่มีความเกี่ยวข้องกัน โดยวิธีการทำงานวิจัยนี้มีขั้นตอน ดังนี้

1. การแยกประเภทและการจัดกลุ่ม:

- การจัดกลุ่ม (clustering) บทความวิจัยมีวิธีการต่าง ๆ ที่สามารถใช้เพื่อวัดความเกี่ยวข้องระหว่างบทความ โดยสามารถใช้สัญญาณหรือ เครื่องหมายการเชื่อมโยง เช่น:
 - การอ้างอิง (Citations): บทความที่อ้างอิงถึงกัน
 - การอ้างอิงร่วม (Cocitations): บทความที่ถูกอ้างอิงร่วมกันจากบทความอื่น
 - การเชื่อมโยงของบรรณานุกรม (Bibliographic Coupling): บทความที่มีการอ้างอิงร่วมกันในบรรณานุกรม
 - การเชื่อมโยงของคำ (Co-word Linkages): บทความที่ใช้คำเดียวกันในเนื้อหาของบทความ

การใช้ข้อมูลเหล่านี้สามารถสร้าง matrix ที่แสดงความคล้ายคลึงหรือความสัมพันธ์ระหว่างบทความ เพื่อใช้ในการ จัดกลุ่ม หรือ ระบุตัวหัวข้อ ของบทความ

2. วิธีการทางวิทยาการสารสนเทศ (Information Retrieval):

- วิธีการที่ใช้ในงานวิจัยนี้มีความคล้ายคลึงกับเทคนิคในการ ดึงข้อมูล แต่ต่างจากวิธีที่ใช้ word space ในการจัดกลุ่มบทความ โดยใช้ ข้อมูลจากหลาย ๆ ส่วนของบทความ (ไม่ใช่แค่เนื้อหาคำ) เช่น ชื่อผู้เขียน, ชื่อวารสาร, หัวข้อเรื่อง, การอ้างอิง, ฯลฯ
- การใช้ ข้อมูลจากหลายส่วนของบทความ ทำให้สามารถสร้าง การแทนที่ความหมายของบทความได้ดีขึ้น โดยอ้างอิงจาก โครงสร้างทางสังคม (เช่น ผู้เขียน), การสื่อสาร (เช่น วารสารที่ตีพิมพ์), และ การแลกเปลี่ยนความรู้ (เช่น การอ้างอิง)

3. คำถามการวิจัย:

- คำถามการวิจัยหลัก คือ (a) การสามารถสร้างการแทนที่ความหมายที่ถูกต้องสำหรับบทความจากองค์ประกอบต่าง ๆ ที่เกี่ยวข้องกับบทความนั้นได้หรือไม่ และ (b) สามารถระบุการจัดกลุ่มบทความที่เกี่ยวข้องโดยใช้วิธีการที่มีอยู่บนพื้นฐานของการแทนที่ความหมายนี้ได้หรือไม่

4. วิธีการจัดกลุ่ม (Clustering):

- บทความนี้กล่าวถึง สองวิธีการจัดกลุ่ม ที่เป็นมาตรฐาน ได้แก่:
 - K-Means clustering: วิธีการจัดกลุ่มที่ใช้การคำนวณค่าเฉลี่ยของจุดศูนย์กลาง (centroids) ในกลุ่ม
 - Louvain community detection algorithm: วิธีการจัดกลุ่มที่เน้นการค้นหาความสัมพันธ์ภายในกลุ่ม (community detection) โดยใช้วิธีการค้นหาที่เหมาะสมกับโครงสร้างของข้อมูล

บทที่ 3

วิธีการวิจัย

3.1 ภาพรวมของโครงการ

3.1.1 ชุดข้อมูลที่นำมาทำการทดลอง

เป็นชุดข้อมูลที่นำมาจากเว็บไซต์ Kaggle เป็นแพลตฟอร์มที่ไว้ค้นหาชุดข้อมูลเพื่อนำมาทำการวิเคราะห์ข้อมูล โดยข้อมูลที่นำมาใช้เป็นประเภท DataFrame มีทั้งหมด 1,000,000 แถว และมีทั้งหมด 8 คอลัมน์ ประกอบด้วย

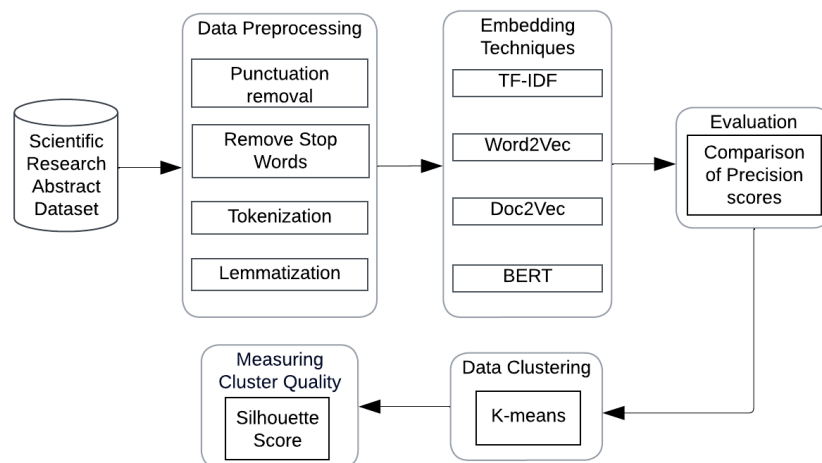
1. abstract เก็บข้อมูลเกี่ยวกับบทคัดย่อในงานวิจัย มีข้อมูลที่หายไป(null) จำนวน 172,467 ค่า ประเภทของข้อมูลเป็น object
2. authors เก็บชื่อของผู้เขียนงานวิจัย มี 2 ค่าที่เป็นค่าว่าง ประเภทของข้อมูลเป็น object
3. n_citation เก็บข้อมูลเกี่ยวกับจำนวนการอ้างอิงงานวิจัย ไม่มีข้อมูลที่หายไป(null) ประเภทของข้อมูลเป็น int64
4. references เก็บข้อมูลเกี่ยวกับรหัสที่อ้างอิงถึงเอกสารที่เกี่ยวข้อง มีข้อมูลที่หายไป(null) 124,417 ค่า ประเภทของข้อมูลเป็น object
5. title เก็บชื่อเรื่องของงานวิจัย ไม่มีข้อมูลที่หายไป(null) ประเภทของข้อมูลเป็น object
6. venue เก็บข้อมูลที่เกี่ยวข้องกับสถานที่ตีพิมพ์งานวิจัย มีข้อมูลที่หายไป(null) จำนวน 177,755 ค่า ประเภทของข้อมูลเป็น object
7. year เก็บปีที่งานวิจัยถูกเผยแพร่ ไม่มีข้อมูลที่หายไป(null) ประเภทของข้อมูลเป็น int64
8. id เก็บหมายเลข ID ของงานวิจัย ไม่มีข้อมูลที่หายไป(null) ประเภทของข้อมูลเป็น object

```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000000 entries, 0 to 999999
Data columns (total 8 columns):
#   Column      Non-Null Count  Dtype
---  -
0   abstract    827533 non-null  object
1   authors     999998 non-null  object
2   n_citation  1000000 non-null int64
3   references  875583 non-null  object
4   title       1000000 non-null  object
5   venue       822245 non-null  object
6   year        1000000 non-null int64
7   id          1000000 non-null  object
dtypes: int64(2), object(6)
memory usage: 61.0+ MB
```

ภาพที่ 3-1 ภาพแสดงข้อมูลเกี่ยวกับชุดข้อมูลที่นำมาทำการทดลอง

3.1.2 Dataflow



ภาพที่ 3-2 ภาพแสดง Data Flow ของโครงการ

แผนภาพนี้แสดงลำดับขั้นตอนการทำงานสำหรับการวิเคราะห์จัดกลุ่มบทความ โดย
ใช้เทคนิคการแปลงข้อความเป็นเวกเตอร์ (Embedding Techniques) และการวัดความคล้ายคลึงกัน
(Similarity Measures) โดยมีขั้นตอนดังนี้

1. ข้อมูลนำเข้า (input) : นำข้อมูลที่ยังไม่ผ่านการทำความสะอาดมา 1 ชุด ซึ่งเป็นชุดข้อมูล
ที่เกี่ยวกับบทความในงานวิจัยภาษาอังกฤษ

2. การเตรียมข้อมูล (Data Preprocessing) : เพื่อทำความสะอาดและจัดเตรียมข้อมูลให้อยู่
ในรูปแบบที่พร้อมสำหรับการแปลงเป็นเวกเตอร์ โดยมีขั้นตอนการทำ 4 ขั้นตอนด้วยกันคือ

1. การทำ Indexing
2. การ Remove Stop Words

3. การทำ Tokenize

4. การทำ Lemmatization

3. การแปลงข้อความเป็นเวกเตอร์ (Embedding Techniques) : นำข้อความมาแปลงให้อยู่ในรูปแบบเวกเตอร์เชิงตัวเลข เพื่อให้คอมพิวเตอร์สามารถเข้าใจและเกิดการประมวลผลได้ โดยมี 4 เทคนิคที่ใช้ด้วยกัน คือ

1. TF-IDF (Term Frequency-Inverse Document Frequency)

2. Word2Vec

- Skip-Gram
- CBOW (Continuous Bag of Words)

3. Doc2Vec

- PV-DM (Distributed Memory Model of Paragraph Vectors)
- PV-DBOW (Distributed Bag of Words Version of Paragraph Vector)

4. BERT (Bidirectional Encoder Representations from Transformers)

4. การวัดความคล้ายคลึง (Similarity Measures) : เปรียบเทียบเวกเตอร์ข้อความเพื่อให้เห็นถึงความคล้ายคลึงกันระหว่างข้อความหรือเอกสาร โดยเทคนิคที่นำมาใช้ในการวัดความคล้ายคลึง คือ Cosine Similarity

5. การจัดกลุ่ม (Clustering) : เพื่อจัดกลุ่มข้อความหรือเอกสารที่มีความคล้ายคลึงกันให้อยู่ในกลุ่มเดียวกันด้วยเทคนิค K-Means Clustering แล้วนำ Silhouette score เป็นตัวประเมินว่ากลุ่มที่จัด มีความเหมาะสมกับข้อมูลหรือไม่

3.1.3 กระบวนการทำ Data Preprocessing

Data preprocessing คือขั้นตอนในการเตรียมข้อมูลเพื่อที่จะนำไปประมวลผลขั้นตอนต่อไป ซึ่งจะทำให้ข้อมูลมีความถูกต้องและแม่นยำมากยิ่งขึ้น โดยขั้นตอนที่นำมาใช้ในโครงการมีดังนี้

1.การทำ Indexing คือการสร้างตัวอ้างอิงที่ช่วยให้การเข้าถึงและค้นหาข้อมูลมีประสิทธิภาพมากขึ้น คล้ายการทำสารบัญในหนังสือ ในโครงการนี้ได้นำคอลัมน์ title เป็นตัวกำหนดเพื่ออ้างอิงถึงคอลัมน์อื่นๆ

```
[ ] df = df.set_index('title')
```

```
df.head(10)
```

	title	abstract	authors	n_citation	references	venue	year	id
	A new approach of 3D watermarking based on image segmentation	In this paper, a robust 3D triangular mesh wat...	['S. Ben Jabra', 'Ezzeddine Zagrouba']	50	['09cb2d7d-47d1-4a85-bfe5-faa8221e644b', '10aa...	international symposium on computers and commu...	2008	4ab3735c-80f1-472d-b953-fa0557fed28b
	Attractor neural networks with activity-dependent synapses: The role of synaptic facilitation	We studied an autoassociative neural network w...	['Joaquin J. Torres', 'Jesus M. Cortés', 'Joaq...	50	['4017c9d2-9845-4ad2-ad5b-ba65523727c5', 'b118...	Neurocomputing	2007	4ab39729-af77-46f7-a662-16984fb9c1db
	A characterization of balanced episturmian sequences	It is well-known that Sturmian sequences are t...	['Geneviève Pequin', 'Laurent Vuillon']	50	['1c655ee2-067d-4bc4-b8cc-bc779e9a7f10', '2e4e...	Electronic Journal of Combinatorics	2007	4ab3a4cf-1d96-4ce5-ab6f-b3e19fc260de
	Exploring the space of a human action	One of the fundamental challenges of recognizi...	['Yaser Sheikh', 'Mumtaz Sheikh', 'Mubarak Shah']	221	['056116c1-9c7a-4f9b-a918-44eb199e67d6', '05ac...	international conference on computer vision	2005	4ab3a98c-3620-47ec-b578-884ec4a6206
	Generalized upper bounds on the minimum distance of PSK block codes	This paper generalizes previous optimal upper ...	['Efraim Laksman', 'Håkan Lennestad', 'Magnus...	0	['01a765b8-0cb3-495c-996f-29c36756b435', '5dbc...	Ima Journal of Mathematical Control and Inform...	2015	4ab3b585-82b4-4207-91dd-b6bce7e27c4e
	Applying BCMP multi-class queueing networks for the performance evaluation of hierarchical and modular software systems	Queueing networks with multiple classes of cus...	['Simonetta Balsamo', 'Gian-Luca Dei Rossi', '...']	6	['1c26e228-57d2-4b2c-b0c9-8d5851c17fac', '7539...	International Journal of Computer Aided Engine...	2015	4ab3e768-78c9-4497-8b8e-9e934cb5f2e4

ภาพที่ 3-3 ภาพแสดงข้อมูลเกี่ยวกับการทำ Indexing

2.การทำ Remove Stop Words คือการลบคำที่ไม่สำคัญออกจากข้อความ โดยคำใน Stop Words มักเป็นคำที่ไม่มีความหมายสำคัญ หรือไม่ส่งผลต่อความเข้าใจเนื้อหาของข้อความ เช่น คำเชื่อม, คำบุพบท หรือคำที่ใช้ทั่วไปในภาษา เช่น "is", "and", "the", "of" โดยโครงการนี้ได้ใช้ไลบรารีที่ชื่อว่า “SpaCy” ในการทำ

```
# กรอง stopwords และ punctuation
mytokens = [word for word in mytokens if word not in stopwords and word not in punctuations]
```

```
# แสดงตัวอย่างคำ stop words จำนวน 10 คำ
sample_stopwords = list(stopwords)[:10]
print(sample_stopwords)
```

```
['my', 're', 'himself', 'many', 'upon', 'if', 'it', 'you', 'up', 'may']
```

ภาพที่ 3-4 ภาพแสดงเกี่ยวกับการลบ Stop Words และตัวอย่างคำ Stop Words ที่มีอยู่ใน SpaCy

3.การทำ Tokenize คือการแยกข้อความออกเป็นหน่วยย่อย ที่เรียกว่า Token อาจจะเป็นคำ ประโยค หรือตัวอักษร เพื่อนำไปใช้ต่อไปในการแปลงข้อความเป็นตัวเลขในขั้นตอนของการ Embedding ในโครงงานนี้จะทำการ Tokenization ในระดับคำ โดยใช้ไลบรารีที่ชื่อว่า “SpaCy” ในการทำ

4.การทำ Lemmatization คือการแปลงคำให้อยู่ในรูปฐาน โดยพิจารณาจากความหมายและไวยากรณ์ของคำในบริบทของคำ เช่น

คำว่า "running" → Lemma: "run" (ในความหมายว่า "การวิ่ง")

คำว่า "better" → Lemma: "good" (ในบริบทนี้ better คือคำคุณศัพท์ขั้นกว่า)

ในโครงงานนี้จะทำการ Lemmatization โดยใช้ไลบรารีที่ชื่อว่า “SpaCy” ในการทำ

```
# ใช้ spacy tokenizer ในการแยกคำและทำ lemmatization
mytokens = parser(sentence)

# เปลี่ยนคำให้เป็น lowercase และ lemmatize (ไม่ใช่ pronouns)
mytokens = [word.lemma_.lower().strip() if word.lemma_ != "-PRON-" else word.lower_ for word in mytokens]
```

ภาพที่ 3-5 ภาพแสดงการทำ Tokenization และ Lemmatization

ข้อมูลที่ทำกร data preprocessing แล้วนั้นจะถูกเพิ่มเป็นอีก 1 คอลัมน์ ที่ชื่อว่า processed_text

title	abstract	authors	n_citation	references	venue	year	id	processed_text
A new approach of 3D watermarking based on image segmentation	In this paper, a robust 3D triangular mesh watermarking algorithm based on 3D segmentation is proposed. In this algorithm three classes of watermarking are combined. First, we segment the original image to many different regions. Then we mark every type of region with the corresponding algorithm based on their curvature value. The experiments show that our watermarking is robust against numerous attacks including RST transformations, smoothing, additive random noise, cropping, simplification and remeshing.	['S. Ben Jabra', 'Ezzeddine Zagrouba']	50	['09cb2d7d-47d1-4a85-bfe5-faa8221e644b', '10aa16da-3cc8-4af6-9d66-48037e915d76', '35cb45c3-9408-4096-ab30-bc2e4de3fb5d', '661a342e-a911-4420-b67d-51c75d3b14e9', '779553f3-e4c1-456e-bc01-5eb9d9567541', 'b24ba5c0-fee8-4a3e-9330-17f6564856cd', 'fd1c676d-1296-4f19-89b4-17c7ecd270f3']	international symposium on computers and communications	2008	4ab3735c-80f1-472d-b953-fa0557fed28b	paper robust 3d triangular mesh watermarking algorithm base 3d segmentation propose algorithm class watermarking combine segment original image different region mark type region corresponding algorithm base curvature value experiment watermarking robust numerous attack include rst transformation smoothing additive random noise cropping simplification remeshing

ภาพที่ 3-6 ภาพแสดงตัวอย่างข้อมูลที่ผ่านมาการทำ Data Preprocessing ครบทุกขั้นตอนแล้ว

3.1.4 การเตรียมคำสำคัญ (Keyword) สำหรับบทคัดย่อ

ในการทดลองนี้ ผู้จัดทำได้เตรียมชุดคำค้นจำนวน 100 คำ ซึ่งได้มาจากบทคัดย่อ 100 แถวแรกของชุดข้อมูล โดยทำการดึงคำสำคัญจากเนื้อหาของแต่ละบทคัดย่อด้วยตนเอง จากนั้น

จึงทำการปรับค่าเหล่านั้นให้เป็นค่าทั่วไปที่มีความหมายใกล้เคียง เพื่อใช้เป็นคำค้นในการทดลองค้นหาคำศัพท์ จำนวน 100 ครั้งในขั้นตอนถัดไป ในการแปลงคำสำคัญให้เป็นค่าทั่วไป มีวัตถุประสงค์เพื่อประเมินว่าแต่ละเทคนิคสามารถเชื่อมโยงคำที่มีความหมายใกล้เคียงกันได้หรือไม่ โดยไม่จำเป็นต้องใช้คำเฉพาะตรงตัวจากบทคัดย่อ

1-20
 "3D mesh",
 "neural networks",
 "sequence analysis",
 "action recognition",
 "error-correcting codes",
 "customer segmentation",
 "transistor circuits",
 "pattern recognition",
 "manipulation",
 "analog circuits",
 "coding theory",
 "record linkage",
 "model evaluation",
 "error propagation",
 "numerical approximation",
 "image resizing",
 "storage scaling",
 "probabilistic models",
 "cold adaptation",
 "distributed systems"

ภาพที่ 3-7 ภาพแสดงตัวอย่างคำค้นที่เตรียมไว้สำหรับการทดลองการค้นหาคำศัพท์

3.1.5 การทำ Embedding และการวัดความคล้ายคลึงด้วย Cosine Similarity

ในการทดลองนี้ ผู้จัดทำได้แปลงข้อความในบทคัดย่อและคำค้นทั้งหมดให้อยู่ในรูปแบบของเวกเตอร์ตัวเลข โดยใช้เทคนิคการฝังเวกเตอร์ (Embedding) ทั้งหมด 4 วิธี ได้แก่ TF-IDF, Word2Vec, Doc2Vec และ BERT

หลังจากได้เวกเตอร์ของบทคัดย่อและคำค้นแล้ว ระบบจะทำการวัดค่าความคล้ายคลึงระหว่างเวกเตอร์ทั้งสอง โดยใช้วิธีการวัด Cosine Similarity ซึ่งสามารถประเมินความใกล้เคียงของข้อความได้จากมุมระหว่างเวกเตอร์

จากผลการคำนวณ ระบบจะทำการเลือกบทความที่มีค่า Cosine Similarity สูงที่สุด เพียง 1 รายการ ต่อคำค้นหนึ่งคำ เพื่อใช้เป็นผลลัพธ์ของการค้นหาในแต่ละรอบของการทดลอง

1. TF-IDF (Term Frequency Inverse Document Frequency)

```
▶ tfidf_vectorizer = TfidfVectorizer()
  tfidf_matrix_subset = tfidf_vectorizer.fit_transform(df["processed_text"])

[ ] cosine_sim_matrix = cosine_similarity(tfidf_matrix_subset, tfidf_matrix_subset)
```

ภาพที่ 3-8 ภาพแสดงการ Embedding และการทำ Cosine Similarity ของเทคนิค TF-IDF

2. Doc2Vec

```
# infer vector จาก query
qv = model.infer_vector(query.split()).reshape(1, -1)
# คำนวณ cosine similarity
sims = cosine_similarity(qv, abstract_vectors)[0]
```

ภาพที่ 3-9 ภาพแสดงการ Embedding และการทำ Cosine Similarity ของเทคนิค Doc2Vec โมเดล DM และโมเดล DBOW

3. Word2Vec

```
# embedding
query_vec = get_sentence_vector(query, model).reshape(1, -1)
# คำนวณ cosine similarity
cosine_sim = cosine_similarity(query_vec, abstract_vectors)
```

ภาพที่ 3-10 ภาพแสดงการ Embedding และการทำ Cosine Similarity ของเทคนิค Word2Vec โมเดล CBOW และ Skip-gram

4. BERT

```
# สร้าง embedding ของ query
q_emb = model.encode([query], convert_to_tensor=False)[0]
# คำนวณ cosine similarity กับทุก document
sims = cosine_similarity([q_emb], abstract_embeddings)[0]
```

ภาพที่ 3-11 ภาพแสดงการ Embedding และการทำ Cosine Similarity ของเทคนิค BERT

3.1.6 การจัดกลุ่มบทความด้วย K-means และประเมินด้วย Silhouette Score

ในการทดลองนี้ ผู้จัดทำได้นำเวกเตอร์ที่ได้จากการแปลงบทความทั้งหมด ที่มาจากการ embedding ด้วย BERT มาทำการจัดกลุ่ม (Clustering) ด้วยอัลกอริทึม K-Means โดยมีวัตถุประสงค์เพื่อแยกบทความออกเป็นกลุ่มที่มีเนื้อหาใกล้เคียงกัน และค่า K ที่นำมาทดลอง มีตั้งแต่ K = 2 จนถึง K = 7 และใช้ค่า Silhouette Score เป็นตัวชี้วัดหลักในการประเมินคุณภาพของการจัดกลุ่มแต่ละค่า

```
import numpy as np
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from sklearn.decomposition import PCA

# 1) ลดมิติด้วย PCA (ลดเหลือ 2 มิติ)
pca = PCA(n_components=2, random_state=42)
embeddings_reduced = pca.fit_transform(abstract_embeddings)

# 2) วนลูปทำ clustering และคำนวณ Silhouette สำหรับ k = 2..7
for k in range(2, 8):
    # รัน KMeans บนข้อมูลที่ลดมิติแล้ว
    labels = KMeans(n_clusters=k, random_state=42).fit_predict(embeddings_reduced)

    # คำนวณ Silhouette score บนมิติที่ลดแล้ว
    score = silhouette_score(embeddings_reduced, labels)
    print(f"\n== K = {k} clusters (silhouette = {score:.4f}) ==")

    # ดูขนาดแต่ละคลัสเตอร์
    counts = np.bincount(labels)
    for cluster_id, size in enumerate(counts):
        print(f"Cluster {cluster_id}: size = {size}")

    # แสดงตัวอย่างข้อความจากแต่ละคลัสเตอร์ 3 รายการ
    for cluster_id in range(k):
        idxs = np.where(labels == cluster_id)[0][:3]
        for i, idx in enumerate(idxs):
            snippet = docs[idx][:80].replace("\n", " ")
            print(f"→ Example C{cluster_id}_{i+1} [Index {idx}]: {snippet}...")
```

ภาพที่ 3-12 ภาพแสดงการทำ clustering ด้วย K-Means และวัดประสิทธิภาพด้วย Silhouette Score

จากภาพที่ 3-12 อันดับแรกจะใช้ PCA (Principle Component Analysis) เพื่อลดขนาดมิติเหลือ 2 มิติ เพื่อให้ K-means สามารถทำงานได้เร็วขึ้น หลังจากนั้นทำการ Clustering และคำนวณ Silhouette Score ของแต่ละค่า K ตั้งแต่ K = 2 จนถึง K = 7 เมื่อทำการคำนวณหาค่า Silhouette Score แล้ว ให้แสดงจำนวนข้อมูลที่อยู่ในแต่ละ Cluster ด้วย และแสดงตัวอย่างบทความในแต่ละกลุ่ม กลุ่มละ 3 บทความ

3.2 การวิเคราะห์ขอบเขตและความต้องการของระบบ

ระบบสามารถทำการค้นหาบทความได้ถูกต้องตามที่ผู้ใช้งานต้องการ โดยไม่จำเป็นต้องระบุคำค้นหาเป็นคำศัพท์เฉพาะ และคาดหวังว่าบทความที่ขึ้นเป็นผลลัพธ์จะแสดงบทความที่มีคำศัพท์ที่เกี่ยวข้องกันทางความหมายกับคำที่นำมาค้นหาด้วย โดย dataset ต้องเป็นภาษาอังกฤษเท่านั้น

3.3 ประเด็นที่น่าสนใจและสิ่งที่ท้าทาย

3.3.1 ความถูกต้องและแม่นยำในการแสดงผลการค้นหาบทความ ที่อัลกอริทึมประมวลผลออกมา

3.3.2 ระยะเวลาที่อัลกอริทึมในประมวลผลในการหาผลลัพธ์ในการค้นหาบทความ

3.4 ผลลัพธ์ที่คาดหวัง

ระบบสามารถทำการค้นหาบทความได้ถูกต้องตามที่ผู้ใช้งานต้องการ โดยไม่จำเป็นต้องระบุคำค้นหาเป็นคำศัพท์เฉพาะ และคาดหวังว่าบทความที่ขึ้นเป็นผลลัพธ์จะแสดงบทความที่มีคำศัพท์ที่เกี่ยวข้องกันทางความหมายกับคำที่นำมาค้นหาด้วย

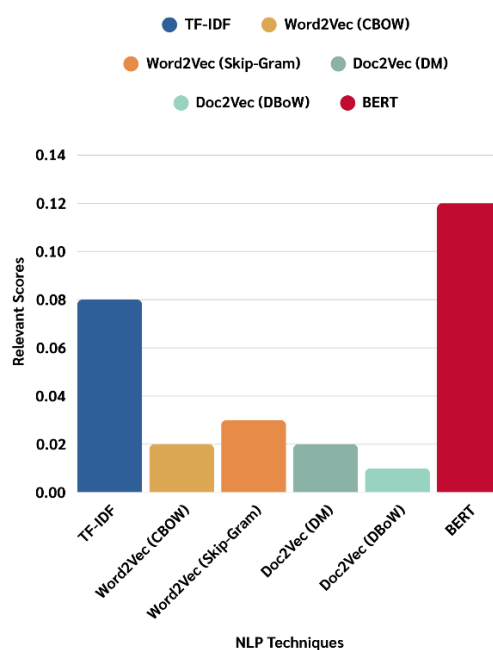
3.5 ระบบต้นแบบและผลลัพธ์เบื้องต้น

เมื่อนำการทดลองการทำ Embedding และวัดความคล้ายคลึงด้วย Cosine Similarity แล้วนั้น ผลการทดลองออกมาดังนี้

Technique	Relevant Scores
TF-IDF	0.08
Word2Vec (CBOW)	0.02
Word2Vec (Skip-Gram)	0.03
Doc2Vec (DM)	0.02
Doc2Vec (DBoW)	0.01
BERT	0.12

ตารางที่ 3-1 ผลการทดลองสำหรับการทำ Embedding และวัดความคล้ายคลึงของเอกสารด้วย Cosine Similarity จาก 4 เทคนิค

สรุปได้ว่า การใช้ BERT ในการทดลองค้นหาด้วย Keywords 1 คำ ต่อ 1 การค้นหา ทั้งหมด 100 คำ ได้ความแม่นยำมากที่สุดใน 4 เทคนิค เพราะว่า BERT ที่นำมาใช้ในการทดลองครั้งนี้ นำโมเดลที่ชื่อว่า SBERT หรือ Sentences BERT มาทำการทดลอง โมเดลนี้จะถูก fine-tune แบบ NLI คือการทำให้โมเดลสามารถเข้าใจความสัมพันธ์ในเชิงตรรกะ ของประโยคได้มากยิ่งขึ้น โดยโมเดล จะต้องตัดสินใจว่าประโยคที่ได้รับมาเป็นประโยคที่สอดคล้อง, ขัดแย้ง หรือ เป็นกลางกับประโยค ข้อเท็จจริงที่ตัวโมเดลมีอยู่ และการทำ STS หรือ Semantic Textual Similarity คือการให้โมเดลดู ความสอดคล้องและเชื่อมโยงของประโยค โดยจะต้องให้คะแนนความคล้ายกันทางความหมายเป็น ตัวเลข เพื่อให้เข้าใจความหมายจริงๆของประโยคนั้นๆ และนอกจากนั้นที่ BERT สามารถค้นหาได้ แม่นยำกว่าเทคนิคอื่นนั้น เพราะ BERT มีการสร้าง Embedding ให้แต่ละประโยคแตกต่างกันตาม context รอบๆคำ



ภาพที่ 3-11 ภาพแสดง Data Visualization การทดลองสำหรับการทำ Embedding และวัดความคล้ายคลึงของเอกสารด้วย Cosine Similarity จาก 4 เทคนิค

บทที่ 4

ผลการดำเนินงาน

จากที่กล่าวไปข้างต้นจะมีการทดสอบความแม่นยำในการค้นหาคัดย่อทางวิชาการของแต่ละเทคนิค เพื่อดูว่าเทคนิคไหนเหมาะสมต่อการนำไปพัฒนาและต่อยอดในระบบการค้นหามากที่สุด

4.1 ผลการทดสอบความแม่นยำในการค้นหาคัดย่อทางวิชาการของแต่ละเทคนิค

Technique	Relevant Scores
TF-IDF	0.08
Word2Vec (CBOW)	0.02
Word2Vec (Skip-Gram)	0.03
Doc2Vec (DM)	0.02
Doc2Vec (DBOW)	0.01
BERT (SBERT)	0.12

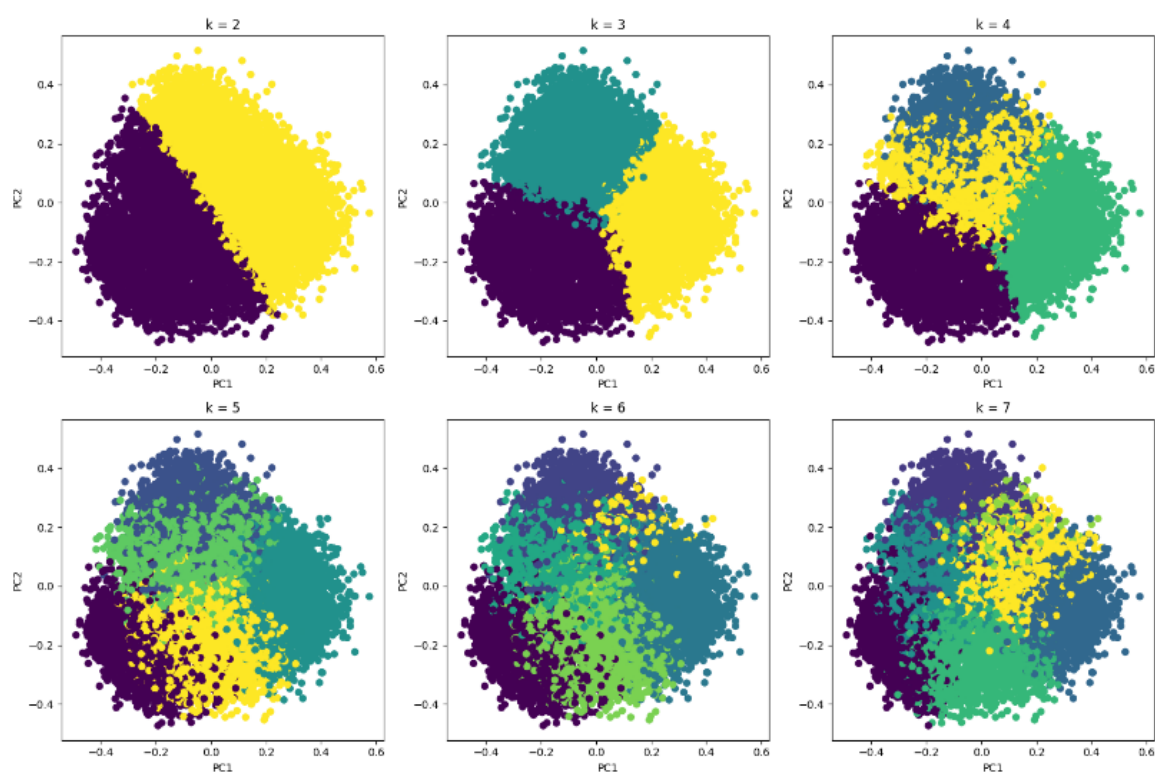
ตารางที่ 4-1 ผลการทดลองสำหรับการทำ Embedding และวัดความคล้ายคลึงของเอกสารด้วย Cosine Similarity จาก 4 เทคนิค

ตารางแสดงถึงความแม่นยำในการทดลองการค้นหาคำด้วย Keyword 100 คำต่อ 100 การค้นหา โดย 4 เทคนิคได้แก่ TF-IDF, Word2Vec (CBOW), Word2Vec (Skip-Gram), Doc2Vec (DM), Doc2Vec (DBOW), BERT โดยทำการ embedding มา 10,000 abstract แรก และสร้าง Keyword สำหรับ 100 abstract แรก เพื่อนำมาเป็นตัววัดความแม่นยำในการค้นหา พบว่า TF-IDF แม้จะไม่มี ความเข้าใจในเชิง semantic แต่การ match ตามคำศัพท์ตรงตัวช่วยให้สามารถดึง abstract ที่เป็น Top1 มาได้ค่อนข้างที่จะตรงในกรณีที่ keyword สั้นๆ นอกจากนี้ TF-IDF ยังใช้ทรัพยากรน้อย เพราะไม่ต้องผ่านกระบวนการ training ใดๆ ถัดมาในส่วนของ Word2Vec (CBOW/Skip-gram) ได้ความแม่นยำค่อนข้างต่ำ เกิดจากการนำเวกเตอร์คำมาเฉลี่ยโดยไม่ถ่วงน้ำหนัก ทำให้น้ำหนักของคำสำคัญจางลง ถัดมาเป็นส่วนของ Doc2Vec(DM,DBOW) ให้ผลลัพธ์ที่ใกล้เคียงกับ Word2Vec เพราะเหมาะใช้ในงานที่ต้องฝึก embedding แบบ document level แต่ไม่เหมาะกับการ keyword search เดี่ยว ๆ

ในทางตรงกันข้าม BERT (SBERT bi-encoder) ทำผลลัพธ์ได้สูงที่สุด เพราะเป็นโมเดล Transformer ที่เข้าใจบริบททั้งสองฝั่ง (bidirectional) ผ่าน pre-training ขนาดใหญ่และ fine-tuning บนงาน NLI/STS (ประเมินความสัมพันธ์ระหว่างประโยคทั้งสองประโยค และวัดระดับความคล้ายกันในเชิงความหมายระหว่างประโยค) จึงสามารถจับ semantic nuance (ความแตกต่างเล็กน้อยในความหมายของคำ เช่น big กับ large , house กับ home) ได้ดี แม้จะมีต้นทุนในการ encode abstracts ล่วงหน้าแล้ว และเรื่องการใช้ GPU/CPU ในการประมวลผล

สรุปได้ว่า BERT (SBERT bi-encoder) เหมาะกับการนำไปพัฒนาระบบการค้นหามากที่สุดสำหรับการเปรียบเทียบ 4 เทคนิคนี้ หากเน้นความเร็วในการค้นหา และโครงสร้างไม่ซับซ้อน TF-IDF จะทำได้ดีกว่า

4.2 ผลการทดลองการ Clustering ด้วยเทคนิค K-means และ ใช้ Silhouette Score ในการวัดความเหมาะสมของจำนวนกลุ่มกับข้อมูลในการทดลอง



ภาพที่ 4-1 ภาพแสดง Data Visualization การการ clustering ด้วยเทคนิค K-Means

จำนวนกลุ่ม (K)	Silhouette Scores
2	0.3350
3	0.4315
4	0.3693
5	0.3620
6	0.3566
7	0.3505

ตารางที่ 4-2 ผลการทดลองวัดความเหมาะสมของการจัดกลุ่ม โดยใช้ Silhouette Scores

จากการทดลองดังกล่าวคือการนำข้อมูลในส่วนที่ BERT ได้ทำการ embedding มาเรียบรื้อยจำนวน 10,000 abstract มาทำการลดมิติ (PCA) ลงเหลือ 2 มิติ และทำการใช้ K-means ในการแบ่งกลุ่มตั้งแต่ 2 กลุ่ม จนถึง 7 กลุ่ม และใช้ Silhouette Scores ในการวัดคุณภาพในการแบ่งกลุ่มพบว่า ค่า Silhouette Scores ที่มากที่สุดคือ 0.4315 ในจำนวนกลุ่ม 3 กลุ่ม เป็นการจัดกลุ่มที่กระชับและแยกตัวออกจากกันได้ดีที่สุด เมื่อเพิ่มค่า $K > 3$ จะเห็นได้ว่าค่า Silhouette Scores ลดลง เพราะการแบ่งกลุ่มมีเยอะเกินไปจากข้อมูล โดยข้อมูลทั้งหมดเป็น abstract ทางวิชาการที่เกี่ยวข้องกับวิทยาศาสตร์และเทคโนโลยี ทำให้การ cluster เริ่มมีข้อมูลที่ทับซ้อนกัน

ในทางตรงกันข้ามนั้น การเลือก $K < 3$ จะทำให้ cluster ใหญ่จนเกินไปทำให้การเกาะกลุ่มของข้อมูลต่ำ ทำให้ $K=3$ จึงเป็นจำนวนกลุ่มที่เหมาะสมที่สุดสำหรับข้อมูลชุดนี้

บทที่ 5

สรุป

5.1 สรุปผลการทดลอง

โครงการเรื่อง การค้นหาและจัดกลุ่มบทความทางวิชาการด้วยการประมวลผลภาษาธรรมชาติ จัดทำขึ้นเพื่อเสนอแนวทางในการปรับปรุงระบบการค้นหาบทความที่มีอยู่ในปัจจุบันให้สามารถค้นหาและจัดกลุ่มข้อมูลได้อย่างมีประสิทธิภาพยิ่งขึ้น โดยผู้จัดทำได้ทำการทดลองเพื่อประเมินความแม่นยำของระบบการค้นหาด้วยเทคนิคจำนวน 4 เทคนิค ได้แก่ TF-IDF, Word2Vec, Doc2Vec และ BERT

ข้อมูลที่ใช้ในการทดลองประกอบด้วยบทความจำนวน 10,000 รายการแรกจากชุดข้อมูลทั้งหมด โดยผู้จัดทำได้แปลงบทความเหล่านี้ให้เป็นเวกเตอร์ด้วยแต่ละเทคนิค หลังจากนั้นได้เตรียมคำค้น (Keyword) จำนวน 100 คำ ซึ่งได้จากการดึงคำสำคัญจากบทความจำนวน 100 แรก เพื่อใช้เป็นชุดคำถามในการประเมินระบบว่าคำค้นแต่ละคำสามารถนำไปค้นหาบทความต้นฉบับได้ถูกต้องเพียงใด และคำสำคัญนั้นจะต้องเป็นคำทั่วไปที่ไม่คำศัพท์เฉพาะ เพื่อดูว่าเทคนิคไหนที่สามารถจับความคล้ายคลึงกันของความหมายคำได้ดีที่สุด

การค้นหาดำเนินการโดยใช้วิธี Cosine Similarity เพื่อเปรียบเทียบความใกล้เคียงระหว่างเวกเตอร์ของคำค้นกับเวกเตอร์ของบทความที่ได้จากแต่ละเทคนิค ผลลัพธ์ที่ได้จากการค้นหาจะถูกนำเสนอในรูปแบบ Data Visualization เพื่อให้ผู้ใช้สามารถเปรียบเทียบประสิทธิภาพของแต่ละเทคนิคได้อย่างชัดเจน

เมื่อได้ผลการทดลองแล้ว เทคนิคที่ให้ผลลัพธ์ดีที่สุด จะถูกนำมาใช้ในการทดลองถัดไป คือการจัดกลุ่มข้อมูล (Clustering) ด้วยเทคนิค K-Means โดยใช้ข้อมูลบทความทั้งหมดจำนวน 10,000 รายการที่ถูกแปลงเวกเตอร์ไว้แล้ว การประเมินประสิทธิภาพของการจัดกลุ่มใช้ค่าชี้วัดที่เรียกว่า Silhouette Score ซึ่งช่วยวัดความแน่นของกลุ่ม (cohesion) และความแตกต่างจากกลุ่มอื่น (separation) โดยผลการทดลองแสดงผลในรูปแบบ Data Visualization เช่นกัน

จากผลการทดลอง พบว่าเทคนิคที่ให้ผลแม่นยำที่สุดในการค้นหาคือ BERT (SBERT) ซึ่งให้ค่า precision สูงสุดที่ 0.12 ความแม่นยำของ BERT นั้นเกิดจากโครงสร้างของโมเดลที่เป็นแบบ Transformer แบบสองทิศทาง (Bidirectional) และผ่านกระบวนการ Pre-training และ Fine-tuning ด้วยชุดข้อมูลขนาดใหญ่ เช่น NLI และ STS ทำให้สามารถเข้าใจบริบทของคำได้ลึกและละเอียด รวมถึงสามารถจับความแตกต่างเล็กน้อยทางความหมาย (Semantic Nuance) ได้ดีกว่าเทคนิคอื่น ๆ เช่น คำว่า “big” กับ “large” หรือ “house” กับ “home”

ในส่วนของการจัดกลุ่มด้วย K-Means จากการทดลองด้วยค่า K ตั้งแต่ 2 ถึง 7 พบว่า ค่า K ที่เหมาะสมที่สุดคือ 3 ซึ่งให้ค่า Silhouette Score เท่ากับ 0.4315 แสดงให้เห็นว่าการแบ่งเป็น 3 กลุ่มมีความเหมาะสมสูงสุด โดยข้อมูลแต่ละกลุ่มมีความคล้ายกันภายใน และแยกตัวออกจากกันได้อย่างชัดเจน ทั้งนี้ เนื่องจากชุดข้อมูลที่นำมาทดลองประกอบด้วยบทความด้านวิทยาศาสตร์และเทคโนโลยี ซึ่งมีเนื้อหาที่คล้ายคลึงกัน จึงเหมาะสมที่จะจัดกลุ่มในระดับกว้างเช่นนี้

5.2 ข้อเสนอแนะและแนวทางในการพัฒนาในอนาคต

การทดลองการค้นหาและการจัดกลุ่มข้อมูลในโครงงานนี้ ยังสามารถนำไปต่อยอดในการพัฒนาระบบ การค้นหาบทความเชิงความหมาย (Semantic Search) ที่สามารถค้นหาด้วยคำทั่วไป ไม่จำเป็นต้องใช้คำศัพท์เฉพาะ เพื่อให้ผู้ใช้ทั่วไปสามารถเข้าถึงข้อมูลทางวิชาการได้ง่ายขึ้น

โครงงานนี้ทำหน้าที่เป็นระบบต้นแบบในระดับเบื้องหลัง (backend prototype) ซึ่งสามารถนำไปต่อยอดและพัฒนาเพิ่มเติมให้กลายเป็นระบบการค้นหาบทความที่สมบูรณ์แบบไดโนอนาคต หากได้รับการพัฒนาอย่างเหมาะสม ระบบนี้จะมีศักยภาพในการสนับสนุนการค้นหาและวิจัยเชิงวิชาการได้อย่างมีประสิทธิภาพและตรงตามความต้องการของผู้ใช้งานมากยิ่งขึ้น

รายการอ้างอิง

- [1] Chakrit. (29 พฤษภาคม 2562). similarity-ความเหมือนที่แตกต่าง. สืบค้นจาก <https://www.softnix.co.th/2019/05/29/similarity-ความเหมือนที่แตกต่าง>
- [2] Chakrit. (06 กันยายน 2561). k-means-และการประยุกต์. สืบค้นจาก <https://www.softnix.co.th/2018/09/06/ว่าด้วย-k-means-และการประยุกต์/>
- [3] Gidi Shperber. (26 กรกฎาคม 2560). A gentle introduction to Doc2Vec. สืบค้นจาก <https://medium.com/wisio/a-gentle-introduction-to-doc2vec-db3e8c0cce5e>
- [4] Hanna Kleinings. (30 กันยายน 2567). How Natural Language Processing Works. สืบค้นจาก <https://levity.ai/blog/how-natural-language-processing-works>
- [5] John Burke. (25 กันยายน 2566). Why and how to use Google Colab. สืบค้นจาก <https://www.techtarget.com/searchenterpriseai/tutorial/Why-and-how-to-use-Google-Colab>
- [6] Pakawat Nakwijit. (4 กันยายน 2563). ทำความเข้าใจ BERT. สืบค้นจาก <https://medium.com/@chameleontk/ทำความเข้าใจ-bert-98589715545>
- [7] Patipan Prasertsom. (15 กรกฎาคม 2564). การค้นหาตัวแทนเชิงความหมายของข้อความ: Word2Vec Word Embedding, Part I. สืบค้นจาก <https://bdi.or.th/big-data-101/word2vec/>
- [8] Patipan Prasertsom. (1 ตุลาคม 2560). สกัดใจความสำคัญของข้อความด้วยเทคนิคการประมวลผลทางภาษาเบื้องต้น: TF-IDF, Part I. สืบค้นจาก <https://bdi.or.th/big-data-101/tf-idf-1/>
- [9] Pulkit Sharma. (1 พฤษภาคม 2568). K-Means Clustering Algorithm. สืบค้นจาก <https://www.analyticsvidhya.com/blog/2019/08/comprehensive-guide-k-means-clustering/>
- [10] Sasiwut Chaiyadecha. (5 พฤษภาคม 2564). รู้จักกับ Word embedding และ Word2Vec. สืบค้นจาก <https://lengyi.medium.com/word-embedding-word2vec-nlp-model-dbc4c892dfb9>

- [11] Sunny Kawinseksan (19 สิงหาคม 2563). Data-sci Diary : Euclidean Distance and Cosine Similarity. สืบค้นจาก <https://medium.com/@sunsun34naka/data-sci-diary-euclidean-distance-and-cosine-similarity-9a809fdb9ead>
- [12] Teeraphol A. (6 พฤศจิกายน 2562). Google Algorithm 'BERT' การวิเคราะห์ชั้นเลิศเพื่อชาว Search. สืบค้นจาก <https://www.linkedin.com/pulse/google-algorithm-bert-การวิเคราะห์ชั้นเลิศเพื่อชาว-search-teeraphol-ambhai/>
- [13] Weerasak Thachai. (15 พฤษภาคม 2560). การหาจำนวน k ที่เหมาะสมที่สุดด้วยวิธี Silhouette Score. สืบค้นจาก <https://medium.com/espressofox-notebook/การหาจำนวน-k-ที่เหมาะสมที่สุดด้วยวิธี-silhouette-b367fdae24d4>

ภาคผนวก

