VIP Cheatsheet: ตัวแปลง & แบบจำลองภาษาขนาดใหญ่

อัฟชิน อามีดี และ เชอร์วิน อามิดี
ผู้แปล: บดินทร์ พรวิลาวัณย์ และชารินทร์ พลภาณุมาศ

27 เมษายน 2568

สรุปเนื้อหาฉบับ *VIP* นี้ ให้ภาพรวมของเนื้อหาในหนังสือ *"Super Study Guide: Transformers & Large Language Models"* ซึ่งบรรจุภาพประกอบกว่า *600* ภาพใน *250* หน้า และเจาะลึกแนวคิดต่างๆ ด้านล่าง ผู้สนใจสามารถดูรายละเอียดเพิ่มเติมได้ที่ *https://superstudy.guide*

1 พื้นฐาน

1.1 โทเคน

❐ **นิยาม** – โทเคน (token) คือหน่วยของข้อความ เช่นคำ (word) คำย่อย (subword) หรืออักขระ (character) ที่แยกย่อยลงอีกไม่ได้ และเป็นส่วนหนึ่งของรายการศัพท์ (vocabulary) ที่นิยามไว้ก่อนแล้ว

หมายเหตุ: โทเคนไม่รู้ *(unknown token) [UNK]* แทนชิ้นส่วนข้อความที่เราไม่รู้ค่า ส่วนโทเคนถมที่ *(padding token) [PAD]* ใช้เติมตำแหน่งที่ว่าง เพื่อให้ลำดับป้อนเข้า *(input sequence)* ยาวเท่ากันโดยตลอด

❐ **ตัวตัดคำ** – ตัวตัดคำ (tokenizer) T ทำหน้าที่แบ่งส่วนข้อความ ให้กลายเป็นโทเคนต่างๆตามระดับความละเอียดหนึ่งๆ

ตุ๊กตาหมีตัวนี้โคตรรรรน่ารัก → T → [ตุ๊กตาหมี] [ตัว] [นี้] [UNK] [น่ารัก] [PAD] ... [PAD]

ตัวตัดคำประเภทชนิดหลักๆมีดังนี้:

ชนิด	ข้อดี	ข้อเสีย	ภาพประกอบ
คำ	• ตีความได้ง่าย • ลำดับสั้น	• รายการศัพท์มีขนาดใหญ่ • ไม่ได้พิจารณารูปแปรต่างๆของคำ	[ตุ๊กตา] [หมี]
คำย่อย	• ใช้ประโยชน์จากรากคำ (word root) • การฝังตัว (embedding) มองเข้าใจทันที	• ลำดับยาวขึ้น • การตัดคำซับซ้อนขึ้น	[ตุ๊ก] [##ตา] [หมี]
ตัวอักขระ ไบต์	• ไม่มีปัญหาหลุดรายการศัพท์ (out-of-vocabulary) • รายการศัพท์มีขนาดเล็ก	• ลำดับยาวกว่ามาก • ตีความรูปแบบลำบาก เพราะอยู่ระดับต่ำเกินไป	[ต] [ุ] [๊] [ก] [ต] [า] [ห] [ม] [ี]

หมายเหตุ: ตัวตัดคำระดับคำย่อยที่ใช้บ่อยได้แก่ *BPE (Byte-Pair Encoding, การเข้ารหัสคู่ไบต์)* และ *Unigram*

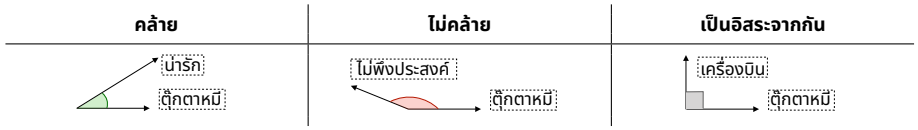
1.2 การฝังตัว

❐ **นิยาม** – การฝังตัว (embedding) คือตัวแทนเชิงตัวเลขของสมาชิกตัวหนึ่งๆ (เช่นโทเคนหรือประโยค) มีลักษณะเป็นเวกเตอร์ $x \in \mathbb{R}^n$

❐ **ความคล้าย** – ความคล้ายเชิงโคไซน์ (cosine similarity) ระหว่างโทเคน t_1, t_2 สองตัวมีค่าเท่ากับ:

$$\text{similarity}(t_1, t_2) = \frac{t_1 \cdot t_2}{||t_1||\ ||t_2||} = \cos(\theta) \in [-1, 1]$$

มุม θ แสดงความคล้ายระหว่างโทเคนทั้งสอง:

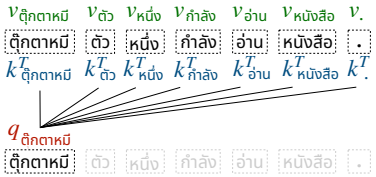


หมายเหตุ: *ANN (Approximate Nearest Neighbors, เพื่อนบ้านใกล้สุดโดยประมาณ)* และ *LSH (Locality Sensitive Hashing, การแฮชแบบใส่ใจถิ่นเดิม)* เป็นวิธีประมาณผลการคำนวณความคล้ายอย่างมีประสิทธิภาพในฐานข้อมูลขนาดใหญ่

2 ตัวแปลง

2.1 การเพ่ง

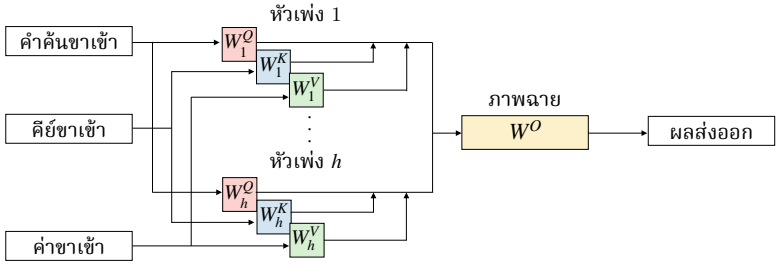
❐ **สูตร** – ให้คำค้น (query) q เราต้องการทราบว่าคำค้นนี้ควร "เพ่ง" คีย์ (key) k ใด เมื่อพิจารณาค่า (value) v ที่คู่กัน



เราคำนวณการเพ่งได้อย่างมีประสิทธิภาพด้วยเมทริกซ์ Q, K, V ซึ่งบรรจุคำค้น q, คีย์ k, ค่า v ตามลำดับ และบรรจุมิติ d_k ของคีย์เหล่านั้นไว้:

$$\text{attention} = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

❐ **MHA** – ชั้น MHA (***M**ulti-**H**ead **A**ttention*, ชั้นเพ่งหลายหัว) คำนวณการเพ่งพร้อมกันหลายหัว (head) ก่อนจะฉาย (project) ผลลัพธ์ที่ได้สู่ปริภูมิส่งออก (output space)

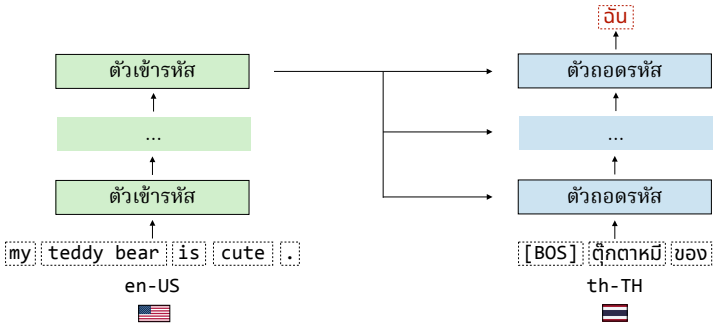


ชั้นนี้ประกอบด้วยหัวเพ่ง (attention head) จำนวน h หัว รวมถึงเมทริกซ์ W^Q, W^K, W^V ที่ฉายข้อมูลขาเข้าไปเป็นคำค้น Q, คีย์ K, และค่า V ต่างๆ การฉายนี้กระทำด้วยเมทริกซ์ W^O

หมายเหตุ: *GQA (Grouped-Query Attention,* เพ่งแบบรวบคำค้น*)* และ *MQA (Multi-Query Attention,* เพ่งหลายคำค้น*)* เป็น *MHA* อีกรูปที่ลดต้นทุนเชิงคำนวณลงด้วยการให้หัวเพ่งต่างๆใช้คีย์และค่าทั้งหลายร่วมกัน

2.2 สถาปัตยกรรม

❒ **ภาพรวม** – ตัวแปลง (transformer) เป็นแบบจำลองสำคัญที่อาศัยกลไกการเพ่งตน (self-attention) ประกอบด้วยตัวเข้ารหัส (encoder) และตัวถอดรหัส (decoder) ต่างๆ ตัวเข้ารหัสจะประมวลข้อมูลขาเข้าเป็นการฝังตัวที่มีความหมาย ซึ่งถัดจากนั้นตัวถอดรหัสจะใช้ทำนายโทเคนตัวถัดไปในลำดับ



หมายเหตุ: ตัวแปลงนั้น แม้เมื่อแรกจะเสนอขึ้นเป็นแบบจำลองสำหรับใช้กับงานการแปล แต่ปัจจุบันนี้ประยุกต์ใช้แพร่หลายในสาขาอื่นอีกมาก

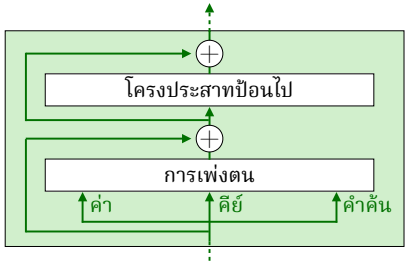
❒ **ส่วนประกอบ** – ตัวเข้ารหัสและตัวถอดรหัสเป็นส่วนประกอบพื้นฐานสองส่วนของตัวแปลง มีบทบาทต่างกันดังนี้:

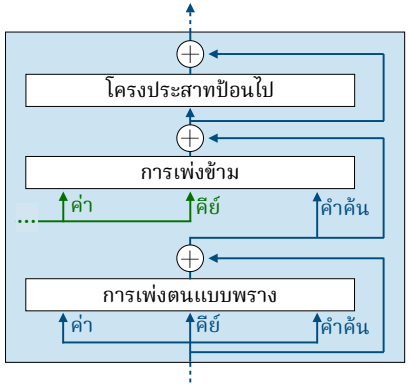
ตัวเข้ารหัส

การฝังเข้ารหัส (encoded embedding) บรรจุความหมายของข้อมูลขาเข้า

ตัวถอดรหัส

การฝังถอดรหัส (decoded embedding) บรรจุความหมายของทั้งข้อมูลขาเข้า และข้อมูลขาออกที่ทำนายมาแล้วถึงจุดนั้น



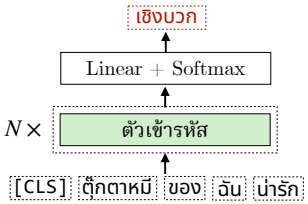


❒ **การฝังตำแหน่ง** – การฝังตำแหน่ง (position embedding) ให้ข้อมูลว่าโทเคนอยู่ตรงไหนในประโยค การฝังนี้มีมิติเท่ากับการฝังของโทเคน อาจนิยามตามความเหมาะสม หรือให้เรียนรู้จากข้อมูลก็ได้

หมายเหตุ: *RoPE (Rotary Position Embeddings,* การฝังตำแหน่งแบบเวียน*)* เป็นอีกรูปหนึ่งที่นิยมใช้และมีประสิทธิภาพ ทำงานด้วยการเวียนเวกเตอร์คำค้นและเวกเตอร์คีย์ เพื่อผนวกข้อมูลตำแหน่งสัมพัทธ์เข้าไว้ด้วย

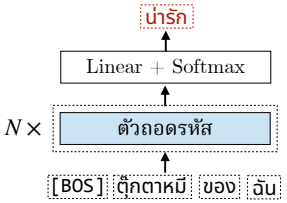
2.3 รูปแปรต่างๆ

❒ **ใช้แต่ตัวเข้ารหัส** – BERT (***B**idirectional **E**ncoder **R**epresentations from **T**ransformers*, ตัวแทนเข้ารหัสสองทางจากตัวแปลง) เป็นแบบจำลองอิงตัวแปลง ที่ประกอบด้วยตัวเข้ารหัสเป็นกองซ้อน รับข้อความบางอย่างเป็นข้อมูลขาเข้า แล้วคืนผลเป็นการฝังตัวที่มีความหมาย สามารถนำไปใช้ภายหลังในงานจำแนกที่ปลายน้ำลงไป



ตรงส่วนต้นของลำดับ จะเติมโทเคน `[CLS]` ไว้เพื่อเก็บความหมายของประโยค การฝังแบบเข้ารหัสของโทเคนนี้ มักนำไปใช้ในงานปลายน้ำต่างๆเช่นการสกัดความรู้สึก (sentiment extraction)

❒ **ใช้แต่ตัวถอดรหัส** – GPT (***G**enerative **P**re-trained **T**ransformer*, ตัวแปลงเชิงสร้างฝึกนำแล้ว) เป็นแบบจำลองอิงตัวแปลงแบบถดถอยในตัว (autoregressive) ประกอบด้วยตัวถอดรหัสเป็นกองซ้อน อนึ่ง GPT ต่างจาก BERT และอนุพันธ์ในกลุ่มนั้นตรงที่ GPT ถือว่าปัญหาทุกชนิดเป็นปัญหาการแปลงข้อความเป็นข้อความทั้งสิ้น



LLM ชั้นแนวหน้าในขณะนี้ ส่วนใหญ่ใช้สถาปัตยกรรมแบบใช้แต่ตัวถอดรหัส เช่น LLM ในตระกูล GPT, LLaMA, Mistral, Gemma, DeepSeek และอื่นๆ

หมายเหตุ: แบบจำลองที่ใช้ทั้งตัวเข้ารหัสและถอดรหัสเช่น *T5* ก็ถดถอยในตัวเช่นกัน และมีลักษณะเฉพาะหลายอย่างร่วมกับแบบจำลองที่ใช้แต่ตัวถอดรหัส

2.4 วิธีเพิ่มประสิทธิภาพ

❒ **การเพ่งแบบประมาณ** – การคำนวณการเพ่งต่างๆ อยู่ในระดับ $\mathcal{O}(n^2)$ ซึ่งเมื่อความยาว n ของลำดับยาวมากขึ้น เป็นไปได้ว่าจะกินทรัพยากรมาก มีวิธีคำนวณแบบประมาณอยู่หลักๆสองวิธี:

- ใช้ความโหรง (sparsity): การเพ่งตนไม่ได้เกิดทั่วทั้งลำดับ แต่เกิดเฉพาะระหว่างโทเคนที่มีส่วนเกี่ยวข้องมากกว่า

- ใช้แรงก์ต่ำ (low rank): ลดรูปสูตรคำนวณการเพ่งเป็นเมทริกซ์แรงก์ต่ำคูณกัน ทำให้ลดภาระการคำนวณได้

❐ **การเพ่งแฟลช** – การเพ่งแฟลช (flash attention) เป็นวิธีแม่นตรง (exact) ที่เพิ่มประสิทธิภาพการเพ่งด้วยการใช้ฮาร์ดแวร์ GPU อย่างชาญฉลาด โดยใช้ SRAM (***S**tatic **R**andom-**A**ccess **M**emory*, แรมสถิต) ซึ่งทำงานได้เร็ว คำนวณกับเมทริกซ์ต่างๆ ก่อนจะเขียนผลลัพธ์ไปยัง HBM (***H**igh **B**andwidth **M**emory*, หน่วยความจำแบนด์วิดท์สูง) ซึ่งทำงานช้ากว่า

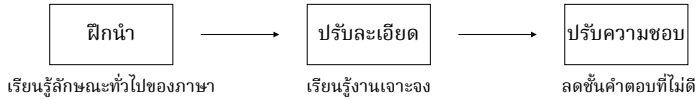
หมายเหตุ: ในทางปฏิบัติ วิธีนี้ช่วยให้ใช้หน่วยความจำน้อยลงและคำนวณได้เร็วยิ่งขึ้น

3 แบบจำลองภาษาขนาดใหญ่

3.1 ภาพรวม

❐ **นิยาม** – LLM (***L**arge **L**anguage **M**odel*, แบบจำลองภาษาขนาดใหญ่) คือแบบจำลองอิงตัวแปลงที่มีความสามารถสูงด้าน NLP (การประมวลผลภาษาธรรมชาติ) แบบจำลองนี้ "ใหญ่" ในแง่ว่าทั่วไปแล้วประกอบด้วยตัวแปรเสริม (parameter) หลักหลายพันล้านตัว

❐ **วงจรชีวิต** – LLM หนึ่งๆจะผ่านการฝึก 3 ขั้น: ฝึกนำ (pretraining), ปรับละเอียด (finetuning), และปรับความชอบ (preference tuning)

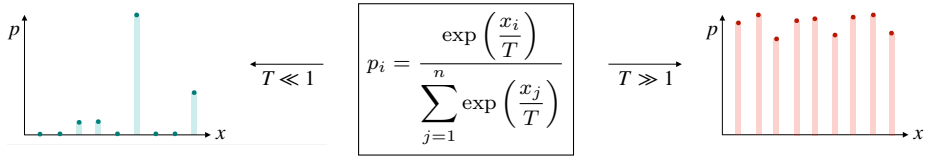


การปรับละเอียดและการปรับความชอบ เป็นวิธีหลังขั้นฝึก ที่มุ่งทำให้แบบจำลองเหมาะปฏิบัติงานเฉพาะอย่าง

3.2 การบอก

❐ **ความยาวบริบท** – ความยาวบริบท (context length) ของแบบจำลองหนึ่งๆ คือจำนวนโทเคนมากสุดที่ข้อมูลขาเข้าจุได้ ทั่วไปยาวตั้งแต่หลักหมื่นถึงหลักล้านโทเคน

❐ **การสุ่มตัวอย่างตอนถอดรหัส** – โทเคนที่ทำนาย สุ่มตัวอย่างมาจากการแจกแจงความน่าจะเป็นที่ทำนาย p_i ซึ่งควบคุมโดยตัวแปรเสริมสูง (hyperparameter) T ที่เรียกว่าอุณหภูมิ (temperature)



หมายเหตุ: อุณหภูมิสูงทำให้ผลลัพธ์ออกมาคิดสร้างสรรค์ *(creative)* มากกว่า ขณะที่อุณหภูมิต่ำทำให้ผลลัพธ์เป็นเชิงกำหนด *(deterministic)* มากกว่า

❐ **ห่วงโซ่ความคิด** – CoT (***C**hain-**o**f-**T**hought*, ห่วงโซ่ความคิด) เป็นกระบวนการให้เหตุผลแบบหนึ่ง แบบจำลองจะแตกปัญหาที่ซับซ้อนเป็นลำดับขั้นสายหนึ่ง ช่วยให้แบบจำลองสามารถสร้างคำตอบสุดท้ายที่ถูกต้องได้ ส่วน ToT (***T**ree **o**f **T**houghts*, ต้นไม้ความคิด) คือ CoT แบบหนึ่งที่ซับซ้อนขึ้น

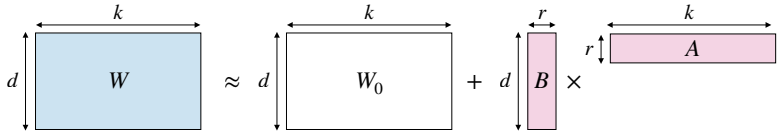
หมายเหตุ: ความสอดคล้องต้องกันในตัว *(self-consistency)* เป็นวิธีหนึ่งที่ใช้ประมวลคำตอบจากแนวให้เหตุผลแบบ *CoT (CoT reasoning path)* แนวต่างๆ

3.3 การปรับละเอียด

❐ **SFT** – SFT (***S**upervised **F**ine**T**uning*, ปรับละเอียดแบบสอน) เป็นแนวทางหลังขั้นฝึกแนวหนึ่ง ที่มุ่งทำให้แบบจำลองมีพฤติกรรมสอดคล้องกับงานเป้าหมายอย่างหนึ่ง วิธีนี้พึ่งพาคู่ข้อมูลขาเข้า-ขาออกที่มีคุณภาพสูง ซึ่งสอดคล้องกับงานนั้น

หมายเหตุ: ถ้าข้อมูล *SFT* เป็นเรื่องคำสั่ง จะเรียกขั้นนี้ว่า "การปรับให้รู้จักคำสั่ง *(instruction tuning)*"

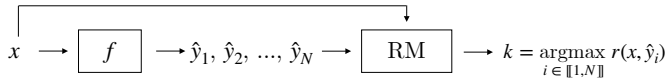
❐ **PEFT** – PEFT (***P**arameter-**E**fficient **F**ine**T**uning*, ปรับละเอียดแบบมีประสิทธิภาพด้านตัวแปรเสริม) เป็นวิธีทำ SFT อย่างมีประสิทธิภาพประเภทหนึ่ง โดยเฉพาะวิธี LoRA (***Lo**w-**R**ank **A**daptation*, ปรับแรงก์ต่ำ) ใช้วิธีประมาณน้ำหนัก W ต่างๆที่เรียนรู้ได้ ด้วยการตรึง W_0 ไว้ตายตัว แล้วเรียนรู้เมทริกซ์ A, B ที่มีแรงก์ต่ำแทน:



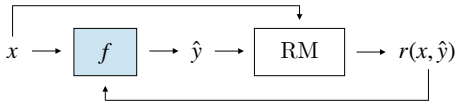
หมายเหตุ: ยังมีกลวิธีแนวทำ *PEFT* แบบอื่นๆเช่น การปรับส่วนเติมหน้า *(prefix tuning)* หรือการแทรกชั้นตัวปรับ *(adapter layer insertion)* เป็นต้น

3.4 การปรับความชอบ

❐ **แบบจำลองรางวัล** – RM (***R**eward **M**odel*, แบบจำลองรางวัล) เป็นแบบจำลองที่ทำนายว่า ข้อมูลขาออก $\hat{y}$ สอดคล้องกับพฤติกรรมที่ต้องการเพียงใดถ้าข้อมูลขาเข้าเป็น x การสุ่มตัวอย่างแบบ BoN (***B**est-**o**f-**N***, ดีสุดใน N ตัว) หรืออีกชื่อว่าสุ่มตัวอย่างแบบปฏิเสธ (rejection sampling) เป็นวิธีที่ใช้ RM เลือกผลตอบสนองที่ดีที่สุดในบรรดา N ตัวที่สร้างมา



❐ **การเรียนรู้แบบเสริมกำลัง** – RL (***R**einforcement **L**earning*, การเรียนรู้แบบเสริมกำลัง) เป็นวิธีที่ใช้ RM ปรับปรุงข้อมูลในตัวแบบจำลอง f อิงตามรางวัลที่ผลลัพธ์ที่สร้างได้รับ ถ้า RM อิงพื้นฐานความพอใจของมนุษย์ จะเรียกกระบวนการนี้ว่า RLHF (***R**einforcement **L**earning from **H**uman **F**eedback*, การเรียนรู้แบบเสริมกำลังจากมนุษย์ติชม)

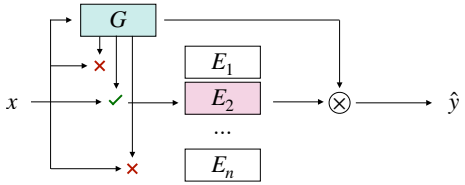


PPO (***P**roximal **P**olicy **O**ptimization*, การหานโยบายเหมาะสุดจากจุดใกล้ชิด) เป็นขั้นตอนวิธีทำ RL ที่นิยมใช้กันแบบหนึ่ง ใช้วิธีจูงใจให้มุ่งได้รางวัลสูงสุด โดยให้แบบจำลองยังคงอยู่ใกล้กับแบบจำลองที่เป็นฐาน เพื่อป้องกันการล่ารางวัลผิดวัตถุประสงค์ (reward hacking)

หมายเหตุ: มีแนวทางที่ทำแบบสอน *(supervised)* ด้วย เช่นวิธี *DPO (Direct Preference Optimization,* ปรับความชอบให้เหมาะสุดโดยตรง*)* ที่รวม *RM* กับ *RL* เข้าด้วยกันเป็นขั้นเดียวที่ทำแบบสอน

3.5 วิธีเพิ่มประสิทธิภาพ

❒ **เหล่าผู้เชี่ยวชาญ** – MoE (***M**ixture **o**f **E**xperts*, เหล่าผู้เชี่ยวชาญ) เป็นแบบจำลองที่จะเปิดใช้งานหน่วยประสาท (neuron) เพียงส่วนหนึ่งเวลาทำการอนุมาน (inference) แบบจำลองนี้ประกอบด้วยเกต G และผู้เชี่ยวชาญ $E_1, ..., E_n$



$$\hat{y} = \sum_{i=1}^{n} G(x)_i E_i(x)$$

LLM ต่างๆที่อิง MoE ใช้กลไกตั้งด่านเช่นนี้ในส่วน FFNN ของตัวเอง

หมายเหตุ: การฝึก *LLM* แบบที่อิง *MoE* เป็นเรื่องท้าทายยิ่ง ดังที่ผู้เขียนเปเปอร์ *LLaMA* ได้กล่าวไว้ในเนื้อหาเปเปอร์ว่า พวกเขาเลือกไม่ใช้สถาปัตยกรรมนี้ แม้จะมีประสิทธิภาพเวลาทำการอนุมานก็ตาม

❒ **การกลั่น** – การกลั่น (distillation) คือกระบวนการฝึกแบบจำลองศิษย์ S (ขนาดเล็ก) ด้วยผลลัพธ์ทำนายของแบบจำลองครู T (ขนาดใหญ่) โดยฝึกด้วยค่าสูญเสียความลู่ออก KL (KL divergence loss)

$$\text{KL}(\hat{y}_T||\hat{y}_S) = \sum_i \hat{y}_T^{(i)} \log\left(\frac{\hat{y}_T^{(i)}}{\hat{y}_S^{(i)}}\right)$$

หมายเหตุ: ฉลากฝึก *(training label)* ถือเป็นฉลาก "อ่อน *(soft)*" เพราะฉลากนี้เป็นตัวแทนความน่าจะเป็นที่จะอยู่ในหมวดต่างๆ

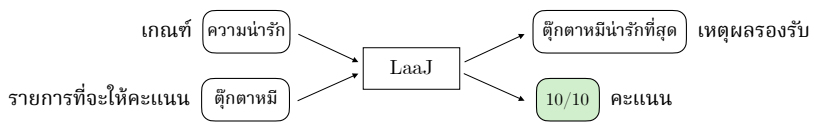
❒ **การแจงหน่วย** – การแจงหน่วยแบบจำลอง (model quantization) เป็นกลวิธีกลุ่มหนึ่งที่มุ่งลดความเที่ยง (precision) ของน้ำหนักในแบบจำลอง ในลักษณะที่จำกัดผลกระทบต่อสมรรถนะของแบบจำลองที่จะได้ ผลลัพธ์คือทำแล้วแบบจำลองจะใช้ทรัพยากรน้อยลง และทำการอนุมานได้เร็วขึ้น

หมายเหตุ: *QLoRA* เป็น *LoRA* แบบแจงหน่วยแล้วที่นิยมใช้กันแบบหนึ่ง

4 การประยุกต์ใช้

4.1 กรรมการ LLM

❒ **นิยาม** – LaaJ (***LLM-a**s-**a**-**J**udge*, กรรมการ LLM) เป็นวิธีใช้ LLM ให้คะแนนผลลัพธ์ต่างๆตามเกณฑ์บางอย่างที่กำหนดให้ จุดเด่นอย่างหนึ่งคือ วิธีนี้สามารถผลิตเหตุผลรองรับคะแนนที่ให้ได้ด้วย ซึ่งช่วยเรื่องความสามารถตีความผล (interpretability)



เทียบกับตัววัด (metric) ในยุคก่อน LLM เช่น ROUGE (***R**ecall-**O**riented **U**nderstudy for **G**isting **E**valuation*, นักแสดงสำรองเชิงระลึกเพื่อประเมินการสรุปความ) แล้ว LaaJ ไม่ต้องใช้ข้อความอ้างอิงใดๆ ทำให้ LaaJ ประเมินงานประเภทใดก็ได้โดยสะดวก โดยเฉพาะประเด็นว่า LaaJ มีสหสัมพันธ์กับคะแนนจากมนุษย์ในเกณฑ์ดีเมื่อพึ่งพาแบบจำลองที่ใหญ่และทรงพลัง (เช่น GPT-4) เนื่องจาก LaaJ ต้องมีความสามารถให้เหตุผลจึงจะทำงานได้ดี

หมายเหตุ: *LaaJ* มีประโยชน์เวลาจะประเมินด่วนหลายๆรอบ แต่สำคัญว่าต้องคอยเฝ้าสังเกตด้วยว่าข้อมูลขาออกจาก *LaaJ* สอดคล้องกับการประเมินของมนุษย์หรือไม่ เพื่อให้แน่ใจว่าทั้งสองอย่างไม่เบนแยกจากกัน

❒ **ความเอนเอียงที่พบบ่อย** – แบบจำลอง LaaJ อาจเอนเอียงในลักษณะต่อไปนี้:

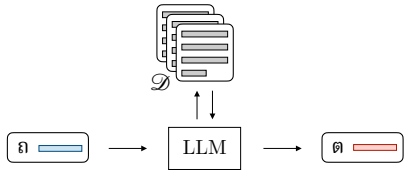
	เอียงตามตำแหน่ง	**เอียงไปทางเยิ่นเย้อ**	**เอียงให้ท้ายตนเอง**
ปัญหา	เอียงเข้าข้างตำแหน่งแรกเวลาเทียบเป็นคู่	เอียงเข้าข้างเนื้อหาที่ใช้คำฟุ่มเฟือยกว่า	เอียงเข้าข้างผลลัพธ์ที่ตัวเองสร้าง
ทางออก	สุ่มตำแหน่งแล้วเฉลี่ยค่าตัววัด	หักคะแนนตามความยาวของผลลัพธ์	ใช้กรรมการตัดสินที่สร้างจากแบบจำลองฐานอันอื่น

ทางแก้ประเด็นเหล่านี้ทางหนึ่งคือ นำ LaaJ มาปรับละเอียดในลักษณะสั่งทำเฉพาะ แต่วิธีนี้ต้องพยายามมาก

หมายเหตุ: ยังมีความเอนเอียงแบบอื่นๆนอกจากที่ระบุในบัญชีข้างต้น

4.2 RAG

❒ **นิยาม** – RAG (***R**etrieval-**A**ugmented **G**eneration*, การสร้างแบบเสริมค้นคืน) เป็นวิธีให้ LLM เข้าถึงข้อมูลภายนอกที่เกี่ยวข้องเพื่อตอบคำถามหนึ่งๆ วิธีนี้มีประโยชน์เป็นพิเศษในกรณีที่เราต้องการผนวกข้อมูลซึ่งมาทีหลังเวลาเส้นตายของความรู้ที่ใช้ฝึกนำ LLM



ให้ฐานความรู้ $\mathscr{D}$ กับคำถามหนึ่งๆ ตัวค้นคืน (**R**etriever) จะไปหยิบเอกสารที่ต่างๆ เกี่ยวข้องที่สุดมา แล้วเสริม (**A**ugment) ข้อมูลที่เกี่ยวข้องเข้ากับคำบอก (prompt) ก่อนจะสร้าง (**G**enerate) ผลลัพธ์

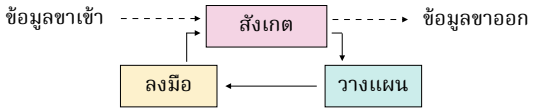
หมายเหตุ: ขั้นค้นคืนมักอาศัยการฝังตัวที่ได้จากแบบจำลองแบบที่ใช้แต่ตัวเข้ารหัส

❒ **ตัวแปรเสริมสูงต่างๆ** – เราเริ่มเตรียมฐานความรู้ $\mathscr{D}$ ด้วยการหั่นเอกสารต่างๆ เป็นก้อนๆ ขนาด n_c แล้วฝังก้อนเหล่านั้นเข้าไปเป็นเวกเตอร์ใน $\mathbb{R}^d$

4.3 ตัวแทน

❒ **นิยาม** – ตัวแทน (agent) คือระบบที่ทำตามเป้าหมาย และปฏิบัติงานให้ลุล่วงแทนผู้ใช้ ในลักษณะทำงานได้เอง (autonomous) ในการนั้นตัวแทนอาจเรียกใช้ LLM หลายๆ ห่วงโซ่ต่างกัน

❒ **ReAct** – ReAct (***R**eason* + ***Act***) เป็นกรอบแนวคิดที่เปิดให้สามารถเรียกใช้ LLM หลายห่วงโซ่เพื่อปฏิบัติงานที่ซับซ้อนให้ลุล่วง:



กรอบแนวคิดนี้ประกอบด้วยขั้นต่างๆดังนี้:

- สังเกต: ประมวลการกระทำต่างๆก่อนหน้า และระบุออกมาให้ชัดเจนว่าขณะนี้รู้อะไร
- วางแผน: แจงรายละเอียดว่าต้องบรรลุงานใดบ้าง และจะเรียกใช้เครื่องมือใดบ้าง
- ลงมือ: กระทำการผ่าน API หรือมองหาข้อมูลที่เกี่ยวข้องในฐานความรู้หนึ่งๆ

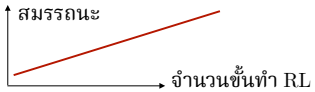
หมายเหตุ: การประเมินระบบเชิงตัวแทน *(agentic system)* เป็นงานท้าทาย แต่ก็มีวิธีทำอยู่ ทำได้ทั้งในระดับส่วนประกอบ ด้วยการใช้ข้อมูลขาเข้า-ขาออกเฉพาะที่ *(local inputs-outputs)* และในระดับระบบ ผ่านการเรียกเป็นห่วงโซ่

4.4 แบบจำลองให้เหตุผล

❒ **นิยาม** – แบบจำลองให้เหตุผล (reasoning model) เป็นแบบจำลองที่ใช้รอยให้เหตุผลตามแนว CoT (CoT-based reasoning trace) แก้ปัญหาในสาขาคณิตศาสตร์ การเขียนโค้ด และตรรกศาสตร์ที่ซับซ้อนขึ้น ตัวอย่างแบบจำลองให้เหตุผลเช่น แบบจำลองตระกูล o ของ OpenAI, DeepSeek-R1, และ Gemini Flash Thinking ของ Google

หมายเหตุ: *DeepSeek-R1* ระบุรอยให้เหตุผลอย่างชัดเจนในผลลัพธ์ อยู่ในแท็ก `<think>`

❒ **การย่อขยาย** – มีวิธีย่อขยายเพื่อเพิ่มความสามารถให้เหตุผลอยู่สองประเภท:

	คำบรรยาย	**ภาพประกอบ**
ย่อขยายตอนฝึก	ทำ RL นานขึ้น เพื่อให้แบบจำลองเรียนรู้วิธีผลิตรอยให้เหตุผลแนว CoT ก่อนจะตอบ	
ย่อขยายตอนใช้งาน	ให้แบบจำลองคิดให้นานขึ้นแล้วจึงค่อยตอบ ด้วยการใช้คำสำคัญที่บังคับงบ (budget forcing) เช่นคำว่า `"Wait"`