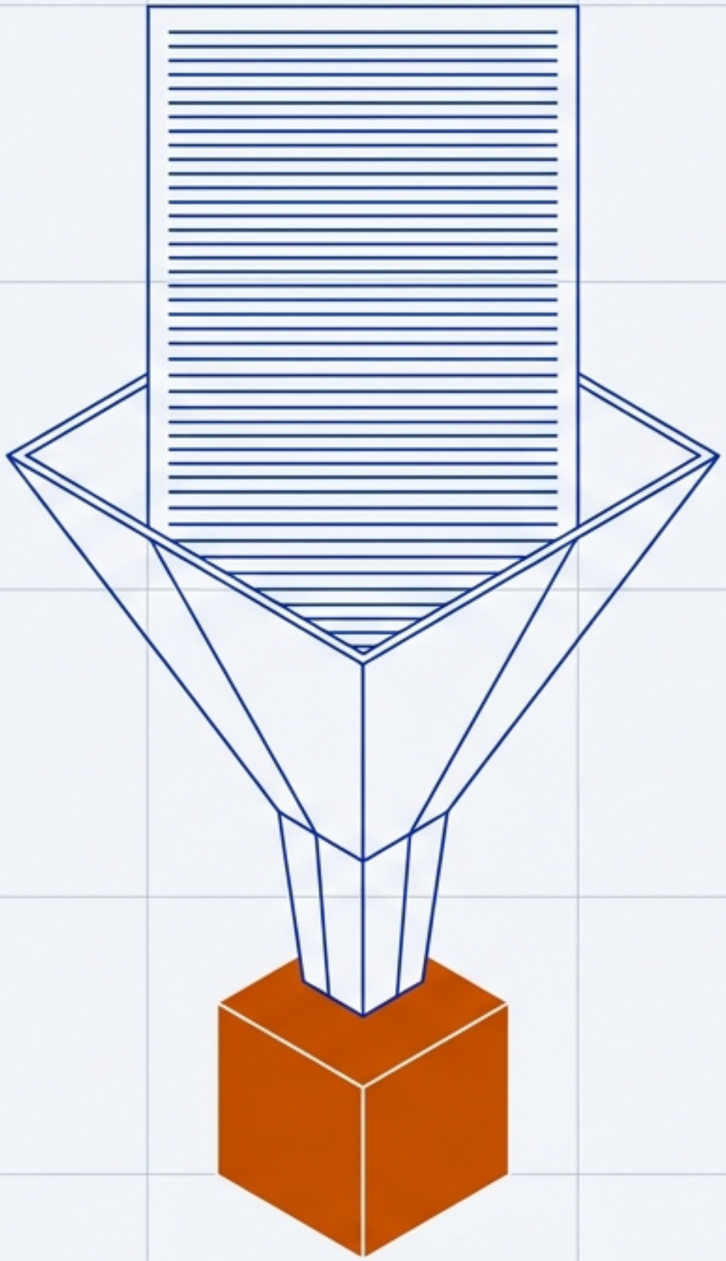
การปรับใช้ LLM เพื่อการย่อความบริบท

สถาปัตยกรรม AutoCompressor และการใช้งาน Summary Vectors เพื่อขยาย Context Window



นำเสนอข้อมูลเชิงเทคนิคสำหรับนักวิจัย AI และวิศวกร NLP

NotebookLM

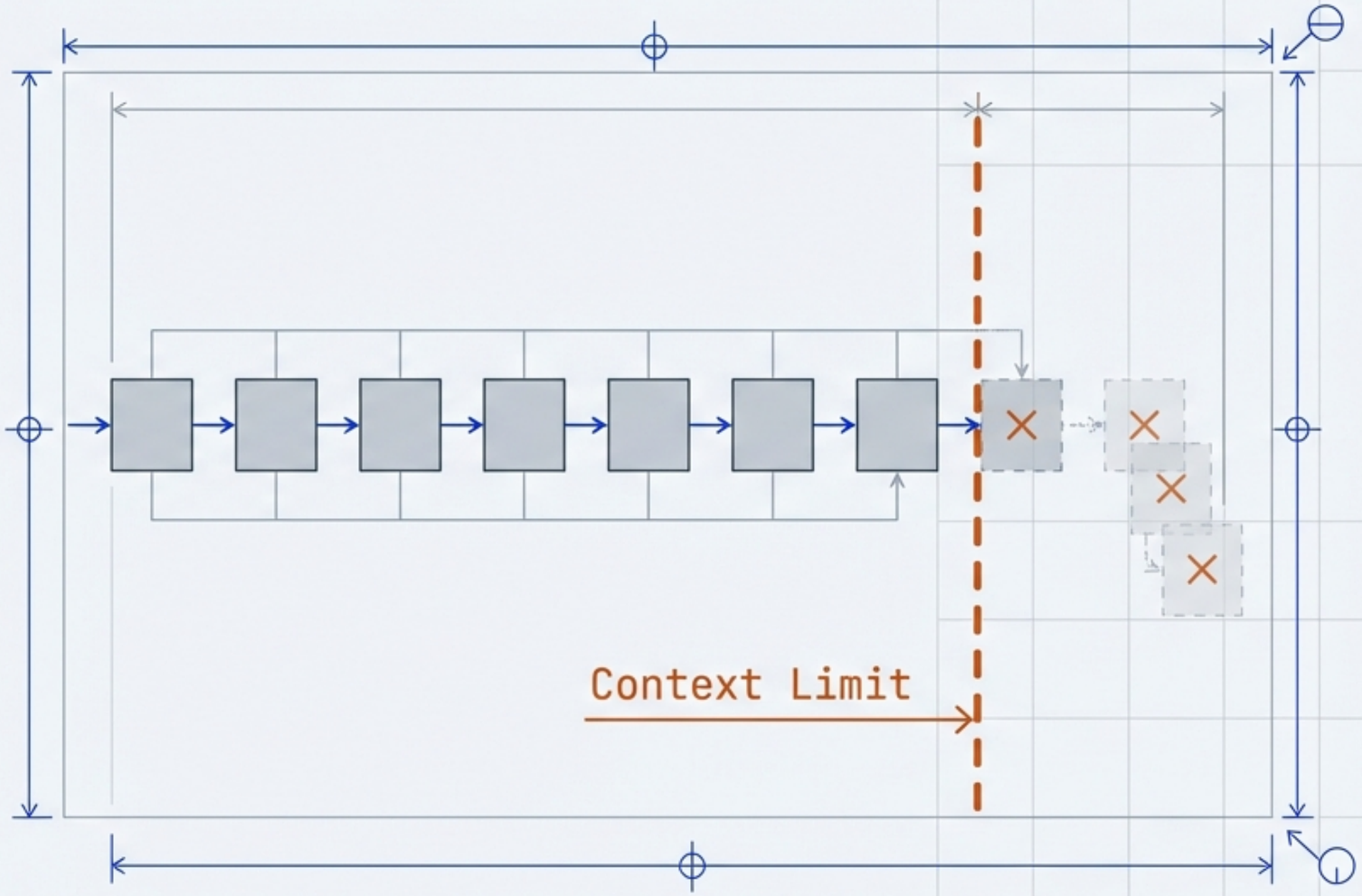
ข้อจำกัดของ Attention-based LLMs ในปัจจุบัน

ภาระการคำนวณที่สูง (Computational Burden)

การใช้ Self-attention กับเอกสารที่มีความยาวมาก (Long Text Document) ต้องใช้ทรัพยากรการคำนวณมหาศาล

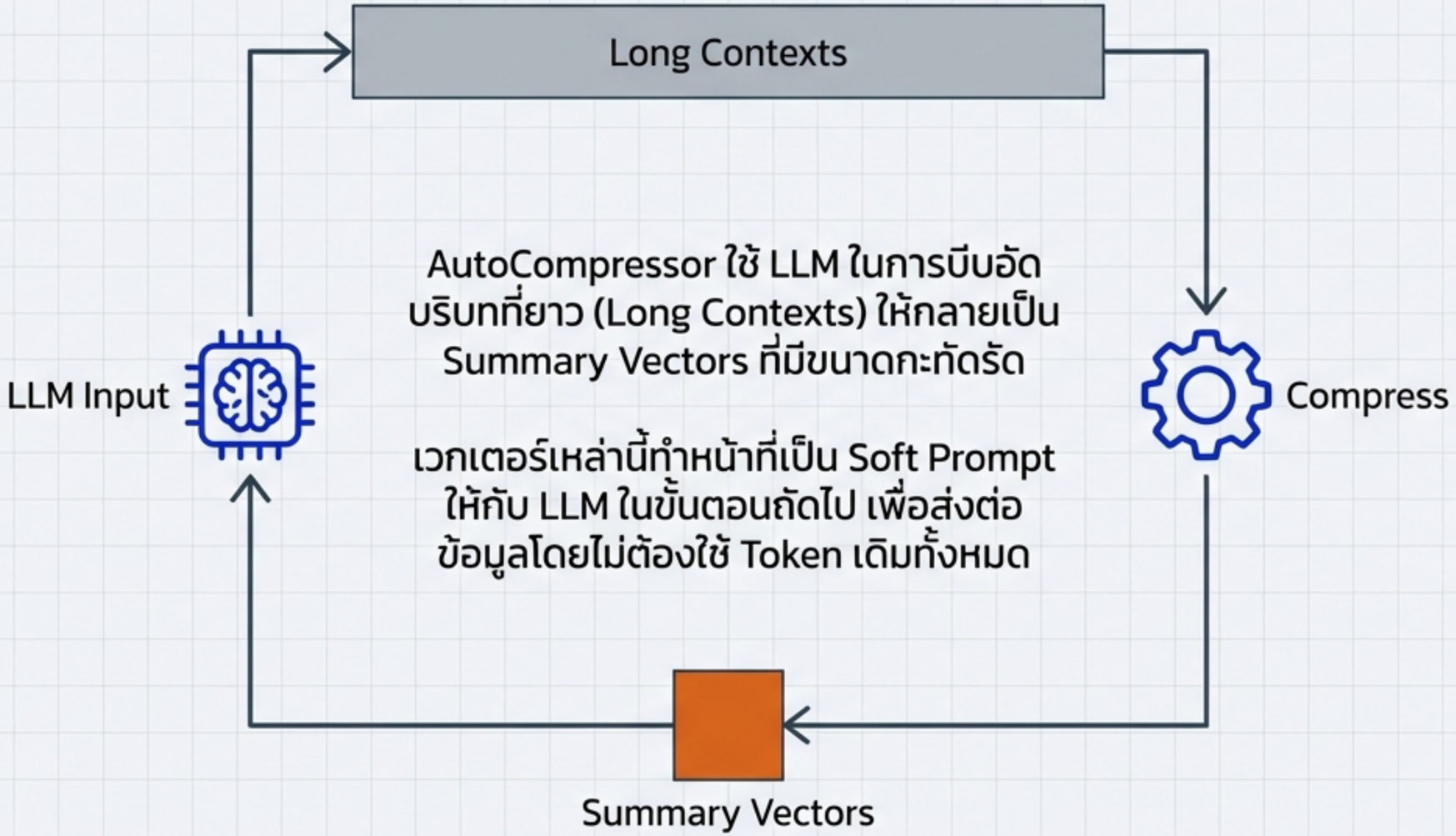
Context Window ที่จำกัด (Finite Context Window)

โมเดลไม่สามารถประมวลผลข้อมูลที่ยาวเกินกว่าหน้าต่างบริบทที่กำหนดไว้ได้ ทำให้ข้อมูลสำคัญบางส่วนสูญหาย



NotebookLM

แนวคิดหลัก: การบีบอัดบริบทด้วย AutoCompressor

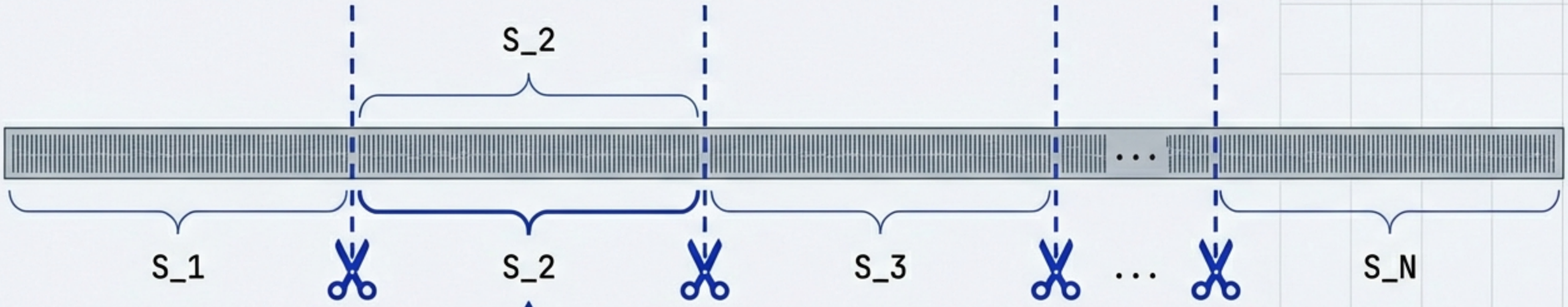


AutoCompressor ใช้ LLM ในการบีบอัดบริบทที่ยาว (Long Contexts) ให้กลายเป็น Summary Vectors ที่มีขนาดกะทัดรัด

เวกเตอร์เหล่านี้ทำหน้าที่เป็น Soft Prompt ให้กับ LLM ในขั้นตอนถัดไป เพื่อส่งต่อข้อมูลโดยไม่ต้องใช้ Token เดิมทั้งหมด

NotebookLM

ขั้นตอนที่ 1: การแบ่งส่วนข้อมูล (Segmentation)

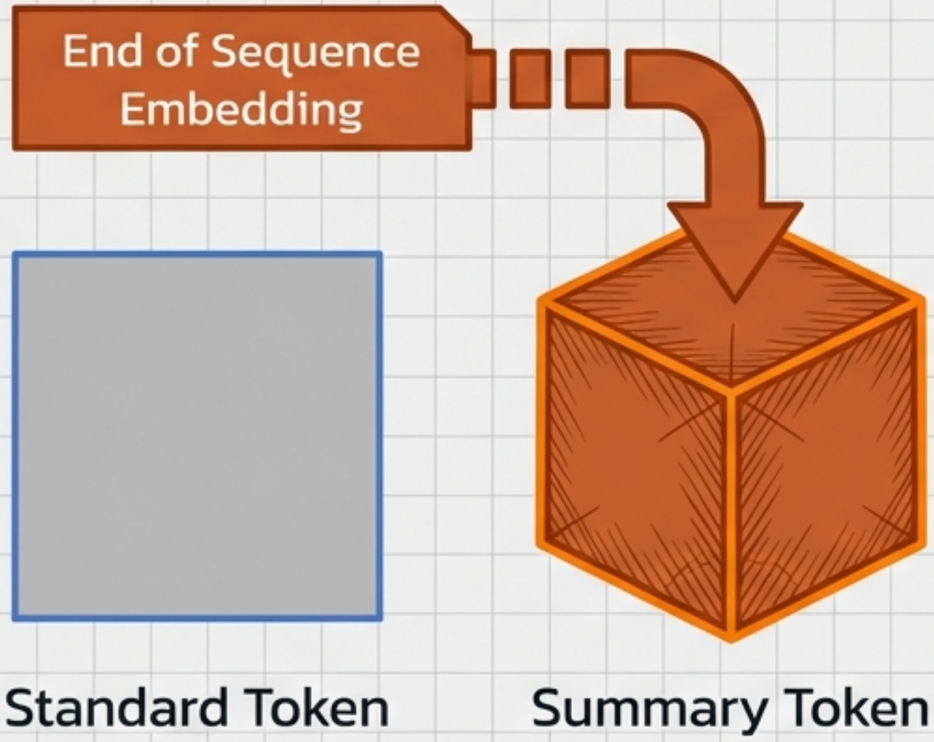


แต่ละส่วน (S_i) จะประกอบด้วยจำนวน Token ที่กำหนด (M_i tokens) ตั้งแต่ $X_{_1}$ ถึง X_{M_i}

เอกสารขนาดยาว (Long Document) จะถูกแบ่งออกเป็นส่วนย่อยๆ (Segments) ตั้งแต่ $S_{_1}$ จนถึง S_N

NotebookLM

องค์ประกอบสำคัญ: Summary Tokens



- Summary Tokens ที่ป้อนเข้าสู่ LLM คือ Key Special Tokens ที่ถูกกำหนดขึ้นมาโดยเฉพาะ
- โทเคนเหล่านี้เริ่มต้น (Initialized) ด้วย Pre-trained Embedding ของโทเคน 'End of Sequence' เพื่อให้โมเดลเข้าใจจุดสิ้นสุดและสรุปความ

NotebookLM

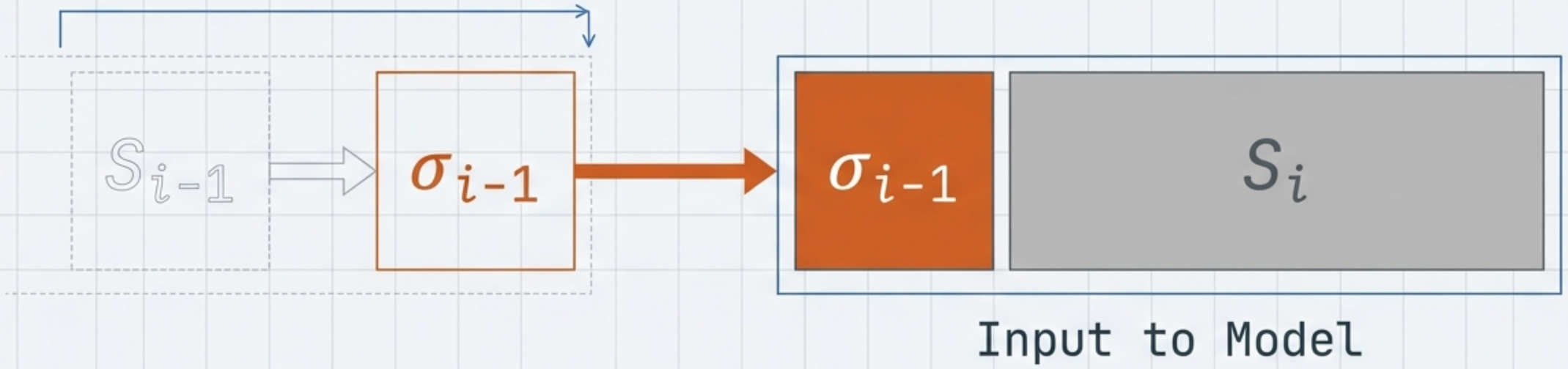
ขั้นตอนที่ 2: การสร้างเวกเตอร์สรุปความ (σ)

LLM ที่ผ่านการฝึกฝน (Pre-trained) จะทำการบีบอัดบริบทของแต่ละส่วน (S) ให้กลายเป็น Summary Vectors แบบสั้น (\σ)

การประมวลผลเป็นแบบลำดับ (Sequential) โดยเวกเตอร์ σ_{i-1} เกิดจากการบีบอัดข้อมูลจากส่วน S_{i-1}

NotebookLM

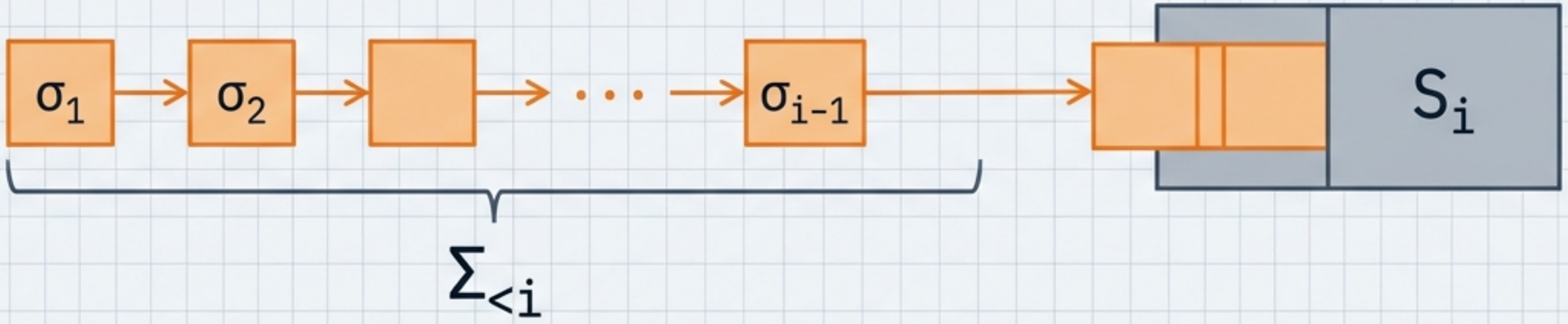
การส่งต่อข้อมูลแบบวนซ้ำ (Recurrent Processing)



- Summary Vector จากส่วนก่อนหน้า (σ_{i-1}) จะถูกนำไปวางไว้ข้างหน้า (Prepended) ข้อมูล Input (Embedded Input) ของส่วนปัจจุบัน (S_i)
- เพื่อให้โมเดลรับรู้บริบทที่เพิ่งเกิดขึ้นทันที ก่อนที่จะเริ่มประมวลผลข้อมูลชุดใหม่

NotebookLM

การสะสมบริบททั้งหมด (Accumulation $\Sigma_{<i}$)



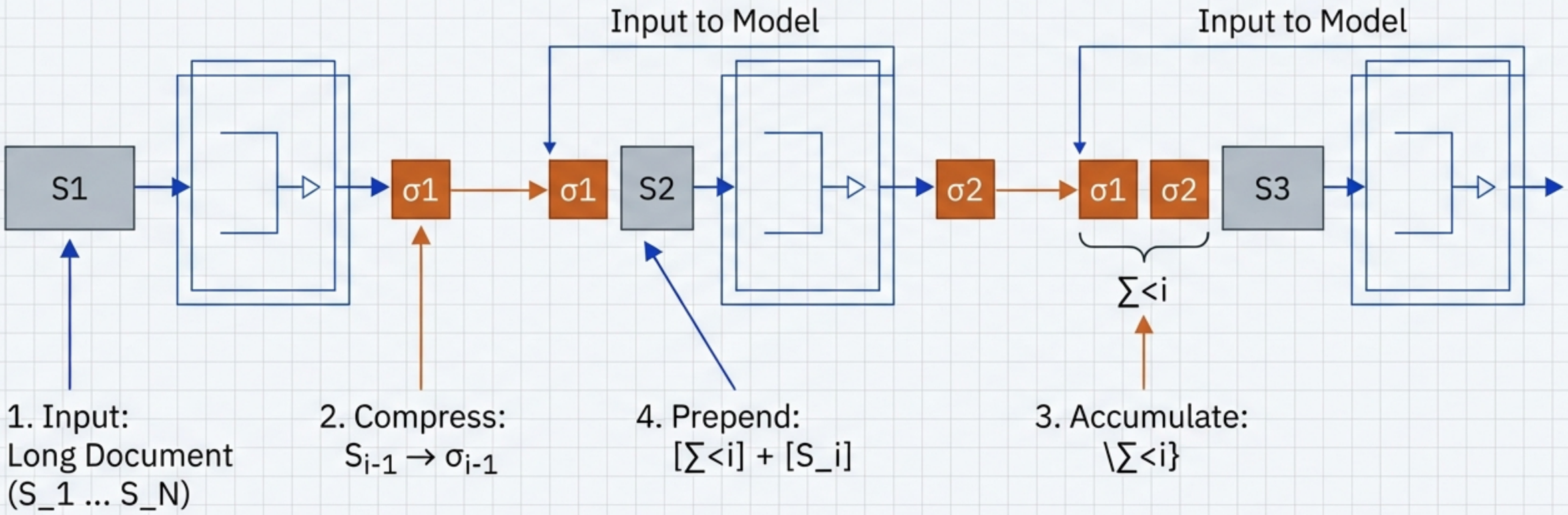
$$\Sigma_{<i} = [\sigma_1][\sigma_2]...[\sigma_{i-1}]$$

ไม่ใช่แค่ส่วนก่อนหน้าเท่านั้น แต่เป็นการรวมประวัติทั้งหมด

ข้อมูลสะสมนี้ ($\Sigma_{<i}$) จะถูกนำไปวางหน้า S_i เช่นกัน ทำให้โมเดล 'จำ' ข้อมูลตั้งแต่เริ่มต้นเอกสารได้

NotebookLM

สถาปัตยกรรมกระแสข้อมูล (Full Data Flow Architecture)



NotebookLM

ฟังก์ชันความสูญเสีย (Cross-Entropy Loss Objective)

แสดงสมการ Cross-Entropy Loss ที่คำนวณจากเอกสารทั้งหมด (N tokens)

$$L = -\sum_{t=1}^{N} \log P(x_t \mid x_{<t}, \Sigma_{<segment(t)})$$

- N : จำนวนโทเคนทั้งหมด (Total number of tokens)
- P : ความน่าจะเป็นของโทเคนถัดไป (Next token probability)

NotebookLM

แรงจูงใจในการเรียนรู้ของโมเดล (Incentivization)

- วัตถุประสงค์ของการฝึกฝนนี้ยังคงรักษาความสามารถเดิมของ Pre-trained LLM ในส่วนแรก (S_1) ไว้
- ระบบถูกกระตุ้น (Incentivized) ให้จัดเก็บ ‘ข้อมูลที่มีประโยชน์’ (Useful Information) ไว้ใน Summary Vectors
- ข้อมูลที่ถูกบีบอัดนี้จะถูกนำไปใช้ใน Segments ในอนาคต เพื่อให้การทำนาย Token แม่นยำยิ่งขึ้น

NotebookLM

บทสรุปและอ้างอิง

AutoCompressor ช่วยให้ LLM ประมวลผลเอกสารยาวได้อย่างมีประสิทธิภาพผ่านการเรียนรู้แบบ Unsupervised บนเอกสารทั่วไป

Reference: “Adapting LLMs to Compress Context’

Based on concepts from: AutoCompressor & Summary Vectors

Source Paper

NotebookLM