BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

นาย กิติพัฒน์ เรืองอมรวัฒน์ 63070501006
นาย สัณหณัฐ พรมจรรย์ 63070501069

Midterm Paper Study – Coding in AI 2023
King Mongkut's University of Technology Thonburi

Abstract

BERT ซึ่งย่อมาจาก Bidirectional Encoder Representations from Transformers ซึ่ง BERT เป็นแบบจำลองการแสดงภาษาที่ฝึกการแสดงแบบ deep bidirectional จากข้อความที่ไม่มี label โดยการปรับสภาพทั้งบริบทด้านซ้ายและขวาในทุกเลเยอร์ โมเดล BERT ที่ pre-trained สามารถทำ fine-tunned ได้ด้วยเอาต์พุตเลเยอร์เพิ่มเติมเพียงหนึ่งเลเยอร์เพื่อที่จะสร้าง model ที่มีประสิทธิภาพและล้ำสมัยสำหรับงานที่หลากหลายเกี่ยวกับการประมวลผล natural language ต่าง ๆ เช่น การตอบคำถามและการ inference ภาษาโดยไม่จำเป็นต้องปรับเปลี่ยน architecture เพื่อความเฉพาะของงาน โดย BERT มี concept ที่เรียบง่ายและมีประสิทธิภาพที่สามารถพิสูจน์ได้ผ่านการทดลอง โดยผ่านการประเมินประสิทธิภาพด้วยคะแนน GLUE ที่ 80.5% ความแม่นยำ MultiNLI ที่ 86.7% การตอบคำถาม SQuAD v1.1 Test F1 ที่ 93.2 และ SQuAD v2.0 Test F1 ที่ 83.1

1. Introduction

Language model pre-training ได้แสดงให้เห็นถึงประสิทธิภาพในการพัฒนางาน natural language processing ต่าง ๆ มากมาย [1, 2, 3, 4] รวมถึงงาน sentence-level เช่น การ inference [5, 6] และ paraphrasing [7] ของ natural language ซึ่งมีจุดมุ่งหมายเพื่อทำนายความสัมพันธ์ระหว่างประโยคโดยการวิเคราะห์แบบองค์รวม ตลอดจนงาน token-level เช่น named entity recognition(การกำหนดเอกลักษณ์หรือนิพจน์ของคำ) และการตอบคำถาม โดยโมเดลนั้นจำเป็นต้องสร้างผลลัพธ์ละเอียดในระดับ token-level [8, 9]

มีกลยุทธ์อยู่สองประการสำหรับการใช้การแสดง pre-trained language ในงาน downstream คือ feature-based และ fine-tuning วิธีการ feature-based เช่น ElMo [2] ใช้ architecture เฉพาะงานซึ่งรวมไปถึงการนำเสนอ pre-trained เป็นคุณสมบัติเพิ่มเติม และวิธีการ fine-tuning เช่น Generative Pre-trained Transformer (OpenAI GPT) [3] แนะนำพารามิเตอร์เฉพาะงานขั้นต่ำและได้รับการ train ในงาน downstream เพียงการ fine-tuning pre-trained พารามิเตอร์ทั้งหมดอย่างละเอียด ทั้งสองวิธีนี้ใช้ objective function แบบเดียวกันในระหว่าง pre-training โดยใช้ unidirectional language โมเดลในการเรียนรู้วิธีการนำเสนอ general language ข้อจำกัดของเทคนิคในปัจจุบันสำหรับการนำเสนอแบบ pre-trained โดยเฉพาะอย่างยิ่งสำหรับวิธีการ

fine-tuning ก็คือเทคนิคเหล่านี้ยังเป็นแบบ unidirectional ซึ่งทำให้เกิดข้อจำกัดในการเลือก architecture และเป็นอุปสรรคในการรวบรวมบริบทจากทั้งสองทิศทางในระดับ token-level เช่น การตอบคำถาม โดยใน paper นี้ได้พัฒนาวิธีการ fine-tuning based ด้วยการใช้ BERT ซึ่งช่วยสามารถลดข้อจำกัดที่ได้กล่าวไปก่อนหน้านี้นั่นก็คือ unidirectional โดยการใช้ masked language model (MLM) ซึ่งเป็น pre-training object จะทำการสุ่มในการ masks บาง token จาก input โดยมีวัตถุประสงค์เพื่อ predict id ดั้งเดิมของคำศัพท์ที่ masked ไว้โดยอ้างอิงตามบริบทเท่านั้น ไม่เหมือนกับโมเดลที่เป็น left-to-right language MLM นั้นจะเชื่อมบริบททั้งทิศทางซ้ายและขวาซึ่งก็ คือ left-to-right และ right-to-left ซึ่งช่วยทำให้สามารถ pre-train deep bidirectional transformer ได้ แล้วยังใช้ในการทำนายประโยคถัดไป

วิธีการเหล่านี้ได้ถูกนำไปใช้กับงานที่ละเอียดและมีความซับซ้อนกว่าอย่างเช่น sentence embedding และ paragraph embedding เพื่อการ train sentence ในก่อนหน้านี้ได้มีการใช้เพื่อจัดอันดับประโยคถัดไป, การสร้างคำถัดไปของประโยคแบบ left-to-right จากประโยคหรือคำก่อนหน้า

---

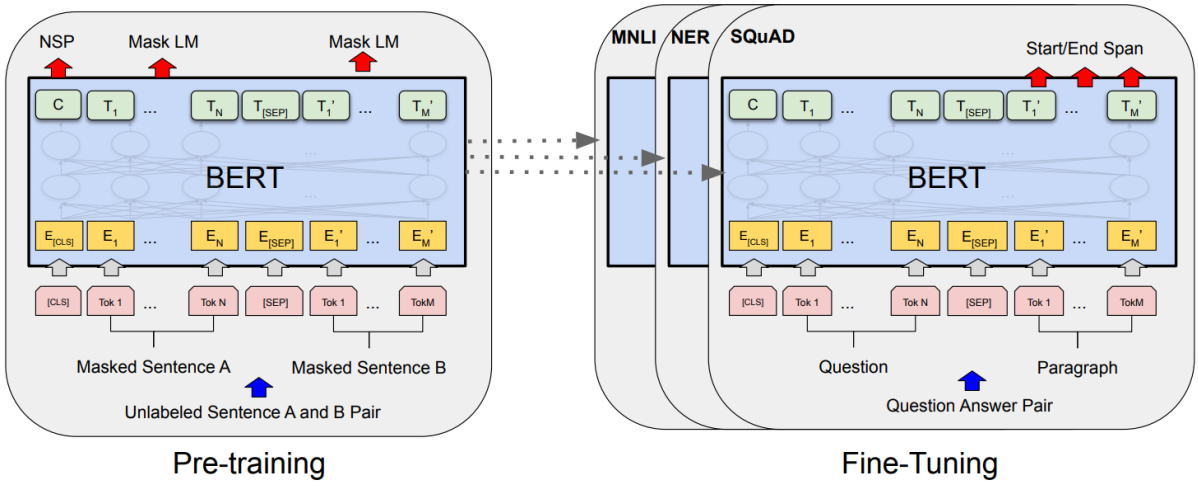
2. Theory

2.1. Pre-training general language

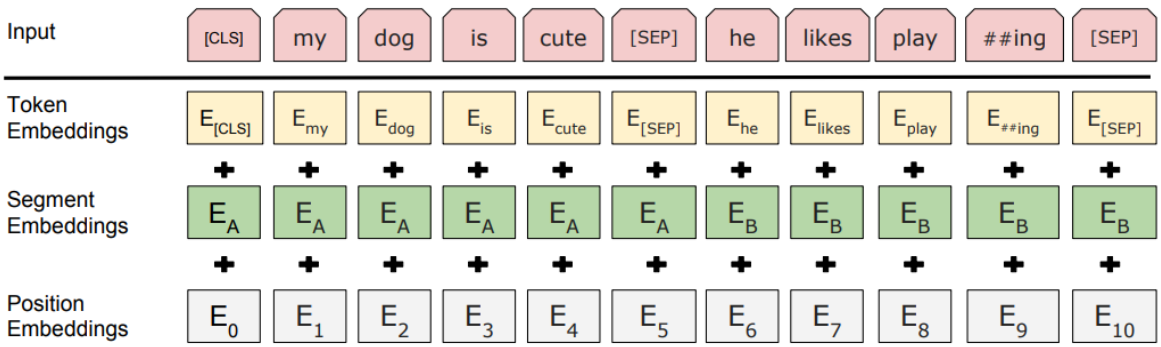
การแสดง Pre-training general language มีประวัติอันยาวนานและดำเนินการโดยใช้วิธีการ unsupervised feature-based และ unsupervised fine-tuning

1. **Unsupervised Feature-based** ได้รับการศึกษาและค้นคว้าอย่างกว้างขวางเพื่อการเรียนรู้การแสดงคำทั้งแบบวิธี non-neural [10, 11, 12] และ neural [13, 14] ด้วย pre-trained word embeddings ซึ่งเป็นส่วนสำคัญใน NLP system สมัยใหม่ แสดงให้เห็นถึงการพัฒนาที่สำคัญเหนือ scratch learned embeddings [15] ผ่านการสร้าง language model และการแยกแยะความถูกต้องจากคำที่ไม่ถูกต้องซึ่งยังถูกใช้กับงานที่มีรายละเอียดที่ซับซ้อนอย่างเช่น sentence embeddings [16, 17] และ paragraph embeddings [18] สำหรับการ train sentence ในอดีตนั้นทำเพื่อการจัดลำดับกลุ่มประโยคที่ถูกเลือกถัดไป [19, 20] ELMo เป็นวิธีการ feature-based ที่แยก features ที่ละเอียดอ่อนตามบริบทจาก language model ทั้ง left-to-right และ right-to-left และยังช่วยพัฒนาประสิทธิภาพในงาน NLP [2] เช่น การตอบคำถาม [9], การวิเคราะห์ความรู้สึก [21] และ ตั้งชื่อการรับรู้ entity [8] ด้วยการประยุกต์ contextual word embeddings รวมกับ architecture เฉพาะงาน
2. **Unsupervised Fine-tuning** ในตอนแรกนั้นมีแค่การ pre-trained พวก word embeddings พารามิเตอร์จาก unlabeled text [22] เช่นเดียวกันกับ feature-based และจากนั้นก็ได้นำไปใช้กับ sentence หรือ document encoders ซึ่งการสร้างโทเคนตามบริบทในการแสดงนั้นก็ถูก pre-trained จาก unlabeled text และได้รับการ fine-tuning ใน supervised downstream task [1, 3, 4] ข้อดีของแนวทางเหล่านี้คือจำเป็นต้องเรียนรู้พารามิเตอร์เพียงเล็กน้อยบางตัวจากการ scratch จากข้อได้เปรียบนี้ OpenAI GPT [3] จึงได้รับผลลัพธ์ที่ล้ำสมัยก่อนหน้านี้ในงานระดับ sentence จำนวนมากจากเกณฑ์มาตรฐาน GLUE [23]

3. **Transfer Learning from Supervised Data** ด้วยชุดข้อมูลขนาดใหญ่ที่มีประสิทธิภาพในการปรับปรุงประสิทธิภาพเป็น natural language inference [24] และ machine translation [25] การวิจัยด้าน Computer vision ได้แสดงให้เห็นถึงความสำคัญของการ transfer learning จาก pre-trained model ขนาดใหญ่ โดยที่สูตรที่มีประสิทธิภาพคือการ fine-tune models pre-trained ด้วย ImageNet [26, 27]



รูปที่ 1: ขั้นตอน pre-training และ fine-tuning โดยรวมสำหรับ BERT นอกเหนือจากเลเยอร์เอาต์พุตแล้ว architecture เดียวกันนี้ยังใช้ทั้งใน pre-training และ fine-tuning พารามิเตอร์โมเดลที่ได้รับ pre-trained เดียวกันนี้ใช้เพื่อเริ่มต้นโมเดลสำหรับ downstream tasks ที่แตกต่างกัน ในระหว่าง fine-tuning พารามิเตอร์ทั้งหมดจะได้รับ fine-tuning. [CLS] เป็นสัญลักษณ์พิเศษที่เพิ่มไว้หน้าตัวอย่างอินพุตทุกตัวอย่าง และ [SEP] เป็นโทเคนตัวแยกพิเศษ (เช่น การแยกคำถาม/คำตอบ)



รูปที่ 2: BERT input representation โดยอินพุต embedding คือผลรวมของ token embeddings, segmentation embeddings และ position embedding

2.2. BERT

BERT เป็น framework ที่ประกอบด้วย 2 ขั้นตอน คือการ pre-training และการ fine-tuning โดยที่โมเดลจะ pre-trained เกี่ยวกับข้อมูลที่ไม่มี label สำหรับการ fine-tuned โมเดล BERT จะถูกเตรียมใช้งานด้วยพารามิเตอร์ที่ pre-trained ในครั้งแรกจากนั้นพารามิเตอร์ทั้งหมดจะได้รับการ fine-tuned โดยใช้ข้อมูลที่ถูก label จาก downstream tasks ซึ่งแต่ละ task มีโมเดลที่ได้รับการ fine-tuned แยกกัน แม้ว่าจะเริ่มต้นด้วยพารามิเตอร์ที่ pre-trained เหมือนกันก็ตาม ตัวอย่างการตอบคำถามในรูปที่ 1 จะใช้เป็นตัวอย่างสำหรับส่วนนี้

คุณลักษณะที่โดดเด่นของ BERT คือมี unified architecture ใน task ต่าง ๆ โดยมีความแตกต่างน้อยที่สุดระหว่าง architecture ที่ pre-trained และ final downstream

Model Architecture คือ multi-layer bidirectional Transformer encoder based อิงตามการใช้งานดั้งเดิม [28] โดยงานนี้เราจะแสดงจำนวนเลเยอร์ (Transformer blocks) เป็น L, ขนาด hidden เป็น H และจำนวน self-attention head เป็น A ซึ่ง BERT มีสองขนาดรุ่นคือ 1. $\mathbf{BERT_{BASE}}$ (L=12, H=768, A=12, พารามิเตอร์ทั้งหมด=110M) และ 2. $\mathbf{BERT_{LARGE}}$ (L=24, H=1024, A=16, พารามิเตอร์ทั้งหมด=340M) โดย $\mathbf{BERT_{BASE}}$ ได้รับเลือกให้มีขนาดโมเดลเดียวกันกับ OpenAI GPT เพื่อวัตถุประสงค์ในการเปรียบเทียบ อย่างไรก็ตาม BERT Transformer ใช้ bidirectional self-attention ในขณะที่ GPT Transformer ใช้ constrained self-attention ซึ่งทุกโทเคนสามารถเข้าร่วมกับบริบททางด้านซ้ายเท่านั้น

Input/Output Representations เพื่อให้ BERT จัดการ downstream tasks ที่หลากหลาย การแสดงอินพุตจึงสามารถแสดงทั้งประโยคเดี่ยวหรือคู่ได้อย่างชัดเจน (เช่น คำถาม คำตอบ) ในลำดับโทเคนเดียว โดยนี้คำว่า “sentence(ประโยค)” หมายถึงเป็นช่วงของข้อความที่อยู่ติดกันตามใจชอบ แทนที่จะเป็นประโยคทางภาษาจริง และ “sequence(ลำดับ)” หมายถึงลำดับโทเคนอินพุตของ BERT ซึ่งอาจเป็นประโยคเดียวหรือสองประโยคที่รวมเข้าด้วยกัน

BERT ใช้ Word Piece embeddings [29] กับคำศัพท์โทเคน 30,000 รายการ และมีโทเคนการจำแนกประเภทพิเศษ ([CLS]) ที่จุดเริ่มต้นของทุกลำดับและ final hidden state ซึ่งสอดคล้องกับโทเคนนี้จะถูกใช้เป็นตัวแทนลำดับรวมสำหรับ classification task คู่ประโยคจะถูกรวมเข้าด้วยกันเป็นลำดับเดียว เราจะแบ่งการแยกประโยคออกเป็นสองวิธี วิธีแรกมีการแยกประโยคโดยใช้โทเคนพิเศษ ([SEP]) วิธีที่สองการแยกความแตกต่างระหว่างประโยคโดยใช้ learned embedding ให้กับทุกโทเคนเพื่อระบุว่ามันเป็นของประโยค A หรือประโยค B ดังแสดงในรูปที่ 1 เราแสดงว่าอินพุตที่ฝังเป็น E ซึ่งเป็น final hidden vector ของโทเคน [CLS] พิเศษเป็น $\mathbf{C} \in \mathbf{R^H}$ และ final hidden vector สำหรับโทเคนอินพุต $\mathbf{i^{th}}$ เป็น $\mathbf{T_i} \in \mathbf{R^H}$ สำหรับโทเคนที่กำหนด การแสดงอินพุตจะถูกสร้างขึ้นโดยการรวม Token, segment, และ position embeddings ที่สอดคล้องกัน การแสดงภาพความหมายนี้สามารถเห็นได้ในรูปที่ 2

1. **Pre-training BERT** โดยใช้สอง unsupervised task คือ Masked LM และ Next Sentence Prediction โดยที่อธิบายไว้ในส่วนนี้ ขั้นตอนนี้แสดงไว้ในส่วนด้านซ้ายของรูปที่ 1

   Task #1: Masked LM เกี่ยวข้องกับ mask แบบสุ่ม 15% ของโทเคน Word Piece ทั้งหมดในแต่ละ sequence และทำนายโทเคนเฉพาะคำที่ mask เหล่านั้นแทนที่จะสร้างอินพุตทั้งหมดใหม่ โดยใช้เอาต์พุต SoftMax การใช้ deep bidirectional model ใน BERT ช่วยให้สามารถแสดงข้อมูลได้มีประสิทธิภาพมาก

ขึ้นเมื่อเปรียบเทียบกับโมเดลแบบดั้งเดิม แต่ข้อเสียคือ เรากำลังสร้างความไม่ตรงกันระหว่าง pre-training และ fine-tuning เนื่องจากโทเคน [MASK] จะไม่ปรากฏขึ้นในระหว่าง fine-tuning เพื่อแก้ปัญหานี้ เราไม่ได้แทนที่คำที่ "masked" ด้วยโทเคน [MASK] จริงเสมอไป โดย Training data generator จะเลือก 15% ของตำแหน่งโทเคนโดยการสุ่มเพื่อการคาดการณ์โทเคน หากเลือกโทเคน *i-th* เราจะแทนที่โทเคน *i-th* ด้วยโทเคน [MASK] 80% ของเวลา, โทเคนแบบสุ่ม 10% ของเวลา และเก็บโทเคนไว้ไม่เปลี่ยนแปลง 10% ของเวลา จากนั้น $\mathbf{T_i}$ จะถูกนำมาใช้เพื่อทำนายโทเคนเดิมที่มี cross-entropy loss

Task #2: Next Sentence Prediction (NSP) เป็น binarized task ที่สามารถสร้างได้เพียงเล็กน้อยจากคลังข้อมูลภาษาเดียว โดยเมื่อเลือกประโยค A และ B สำหรับแต่ละตัวอย่าง pre-training 50% ของเวลา B คือประโยคถัดไปจริงที่ตามหลัง A (labeled as IsNext) และ 50% ของเวลาเป็นประโยคสุ่มจากคลังข้อมูล (labeled as NotNext) ดังที่เราแสดงในรูปที่ 1 มีการใช้ C สำหรับการทำนายประโยคถัดไป งาน NSP เกี่ยวข้องอย่างใกล้ชิดกับวัตถุประสงค์ representation-learning ที่ใช้ใน [17, 19]

Pre-training data ขั้นตอน pre-training ส่วนใหญ่เป็นไปตามการวิจัยที่มีอยู่เกี่ยวกับ language model pre-training สำหรับคลังข้อมูล pre-training ประกอบด้วย BooksCorpus (800 ล้านคำ) [30] และ English Wikipedia (2,500 ล้านคำ) สำหรับวิกิพีเดีย จะแยกเฉพาะข้อความและละเว้นรายการ ตาราง และส่วนหัว โดยจำเป็นอย่างยิ่งที่จะต้องใช้ข้อมูลระดับเอกสารมากกว่าคลังข้อมูลระดับประโยค

2. **Fine-tuning BERT** ทำได้ตรงไปตรงมา เนื่องจากใช้ self-attention mechanism ใน Transformer ช่วยให้ BERT สามารถสร้างโมเดล downstream tasks จำนวนมากได้ ไม่ว่าจะเกี่ยวข้องกับข้อความเดี่ยวหรือคู่ข้อความ โดยการสลับอินพุตและเอาต์พุตที่เหมาะสม ซึ่ง BERT จะใช้ self-attention mechanism เพื่อรวมสองขั้นตอนนี้เข้าด้วยกัน เนื่องจากการเข้ารหัสคู่ข้อความที่ต่อกันเข้ากับ self-attention นั้นรวม bidirectional cross-attention ระหว่างสองประโยคอย่างมีประสิทธิภาพ

---

3. Experimental Design and Results

3.1. GLUE

The General Language Understanding Evaluation (GLUE) [23] คือเกณฑ์มาตรฐานการประเมินความเข้าใจภาษาทั่วไปหลากหลายของงาน การทำความเข้าใจ natural language [31]

เพื่อที่จะ fine-tune บน GLUE จึงได้แสดงลำดับอินพุตที่ได้กล่าวไว้ในส่วนทฤษฎีของ BERT ในช่วง Input/Output Representations และใช้ hidden vector $\mathbf{C} \in \mathbf{R^H}$ ที่สอดคล้องกับ input token แรก ([CLS]) เป็นตัวสรุปข้อมูลของอินพุต พารามิเตอร์ใหม่เพียงตัวเดียวที่ถูกนำมาใช้ระหว่างการ fine-tunning คือ classification layer weights $\mathbf{W} \in \mathbf{R^{K \times H}}$ โดยที่ K คือจำนวนของ labels โดยคำนวณ standard classification loss ด้วย C และ W เช่น $\mathbf{log(SoftMax}(\boldsymbol{CW^T}))$

นักวิจัยใช้ batch size 32 และ fine-tune 3 epoch สำหรับงาน GLUE ทั้งหมด ในแต่ละงานนักวิจัยได้ใช้ learning rate ที่ดีที่สุด (ระหว่าง 5e-5, 4e-5, 3e-5 และ 2e-5) นอกจากนี้สำหรับ BERT (large) ยังพบว่าในการ fine-tuning บางครั้งยังไม่ได้เสถียรมากนักสำหรับชุดข้อมูลขนาดเล็กจึงแก้ไขด้วยการสุ่มรีสตาร์ทเพื่อที่จะเลือก

โมเดลที่ดีที่สุดในการสุ่มรีสตาร์ททีมนักวิจัยจะใช้ pre-trained checkpoint เดิมแต่จะสับเปลี่ยนข้อมูลในการ fine-tuning และ classifier layer initialization ที่แตกต่างกัน

System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	**Average** -
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**92.7**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**82.1**

ตารางที่ 1: ผลการทดสอบ GLUE ซึ่งให้คะแนนโดยเซิร์ฟเวอร์ประเมินผล (https://gluebenchmark.com/leaderboard) ตัวเลขด้านล่างแต่ละงานแสดงถึงจำนวนตัวอย่างการ training คอลัมน์ "ค่าเฉลี่ย" แตกต่างจากคะแนน GLUE อย่างเป็นทางการเล็กน้อย เนื่องจากเราไม่รวมชุด WNLI ที่เป็นปัญหา 8 BERT และ OpenAI GPT เป็นโมเดลเดียว งานเดียว มีการรายงานคะแนน F1 สำหรับ QQP และ MRPC มีการรายงานความสัมพันธ์ของ Spearman สำหรับ STS-B และคะแนนความแม่นยำสำหรับงานอื่นๆ เราไม่รวมรายการที่ใช้ BERT เป็นหนึ่งในองค์ประกอบ

จากผลลัพธ์ที่แสดงในตารางที่ 1 ทั้ง $\mathbf{BERT_{BASE}}$ และ $\mathbf{BERT_{LARGE}}$ มีประสิทธิภาพเหนือกว่าทุก systems ในทุกๆด้าน โดยได้รับการปรับปรุงความแม่นยำโดยเฉลี่ย 4.5% และ 7.0% ตามลำดับ เมื่อเทียบกับเทคโนโลยีรุ่นก่อน และจะเห็นได้ว่าทั้ง $\mathbf{BERT_{BASE}}$ และ $\mathbf{BERT_{LARGE}}$ ยังทำงานได้ดีต่อให้ training data จะน้อยก็ตาม

3.2. SQuAD v1.1

The Stanford Question Answering Dataset (SQuAD v1.1) คือชุดของคู่คำถาม/คำตอบที่มาจากฝูงชน 100,000 คู่ [9] เมื่อมีคำถามและข้อความจากวิกิพีเดียที่มีคำตอบ โดยการทดสอบคือจะต้องคาดเดาคำตอบในช่องว่าง

ดังรูปที่ 1 ในงานตอบคำถาม คำถามที่ป้อนเข้าและข้อความจะแสดงเป็นลำดับ single packed โดยคำถามจะเป็น A embedding และข้อความจะเป็น B embedding ตามลำดับ เวกเตอร์เริ่มต้นและสิ้นสุดถูกนำมาใช้ในระหว่างการ fine-tuning ความน่าจะเป็นที่คำที่เป็นจุดเริ่มต้นของช่วงคำตอบจะคำนวณโดยใช้ dot product ระหว่าง Ti และเวกเตอร์เริ่มต้น S ตามด้วย SoftMax เหนือคำทั้งหมดในย่อหน้า $P_i = \frac{e^{S \cdot T_i}}{\sum_j e^{S \cdot T_j}}$

The analogous formula ใช้เป็นจุดสิ้นสุดของช่วงคำตอบ คะแนนของ ช่วงที่ถูก candidate จากตำแหน่ง i ถึง j ถูกกำหนดเป็น $\mathbf{S \cdot T_i + E \cdot T_j}$ และช่วงที่มีคะแนนสูงสุดที่ $\mathbf{j \geq i}$ ถูกใช้เป็นการทำนาย วัตถุประสงค์ในการ training คือ ผลรวมของ log-likelihoods ของจุดเริ่มต้นและจุดสิ้นสุดที่ถูกต้อง โดยนักวิจัยได้ fine-tune ใน 3 epoch ด้วย learning rate 5e-5 และ batch size 32

System	Dev		Test	
	EM	F1	EM	F1
Top Leaderboard Systems (Dec 10th, 2018)				
Human	-	-	82.3	91.2
#1 Ensemble - nlnet	-	-	86.0	91.7
#2 Ensemble - QANet	-	-	84.5	90.5
Published				
BiDAF+ELMo (Single)	-	85.6	-	85.8
R.M. Reader (Ensemble)	81.2	87.9	82.3	88.5
Ours				
$\text{BERT}_{\text{BASE}}$ (Single)	80.8	88.5	-	-
$\text{BERT}_{\text{LARGE}}$ (Single)	84.1	90.9	-	-
$\text{BERT}_{\text{LARGE}}$ (Ensemble)	85.8	91.8	-	-
$\text{BERT}_{\text{LARGE}}$ (Sgl.+TriviaQA)	**84.2**	**91.1**	**85.1**	**91.8**
$\text{BERT}_{\text{LARGE}}$ (Ens.+TriviaQA)	**86.2**	**92.2**	**87.4**	**93.2**

ตารางที่ 2: ผลลัพธ์ของ SQuAD v1.1 ชุด BERT เป็นระบบ 7x ซึ่งใช้จุดตรวจ pre-training ที่แตกต่างกันและการ fine-tuning seeds

System	Dev		Test	
	EM	F1	EM	F1
Top Leaderboard Systems (Dec 10th, 2018)				
Human	86.3	89.0	86.9	89.5
#1 Single - MIR-MRC (F-Net)	-	-	74.8	78.0
#2 Single - nlnet	-	-	74.2	77.1
Published				
unet (Ensemble)	-	-	71.4	74.9
SLQA+ (Single)	-		71.4	74.4
Ours				
$\text{BERT}_{\text{LARGE}}$ (Single)	78.7	81.9	80.0	83.1

ตารางที่ 3: ผลลัพธ์ของ SQuAD v2.0 ไม่รวมรายการที่ใช้ BERT เป็นหนึ่งในองค์ประกอบ

System	Dev	Test
ESIM+GloVe	51.9	52.7
ESIM+ELMo	59.1	59.2
OpenAI GPT	-	78.0
$\text{BERT}_{\text{BASE}}$	81.6	-
$\text{BERT}_{\text{LARGE}}$	**86.6**	**86.3**
Human (expert)†	-	85.0
Human (5 annotations)†	-	88.0

ตารางที่ 4: ค่าความแม่นยำของ SWAG Dev and Test ประสิทธิภาพของมนุษย์วัดด้วยตัวอย่าง 100 ตัวอย่าง ตามที่รายงานในรายงาน SWAG

ในตารางที่ 2 ได้มีการแสดงลีดเดอร์บอร์ดจากการจัดอันดับระบบที่ถูกเผยแพร่แต่ผลลัพธ์อันดับต้น ๆ [2, 32, 33, 34] จาก SQuAD leaderboard ไม่มีรายละเอียดคำอธิบายของระบบสาธารณะที่ทันสมัยดังนั้นนักวิจัยจึงใช้การเพิ่มข้อมูลเล็กน้อยในระบบของพวกเขา

ระบบที่มีประสิทธิภาพดีที่สุดของนักวิจัยนี้ มีประสิทธิภาพเหนือกว่าระบบลีดเดอร์บอร์ดอันดับต้น ๆ โดย +1.5 F1 ในการประกอบ และ +1.3 F1 เป็นระบบเดียว ในความเป็นจริง โมเดล BERT เดี่ยวของเรามีประสิทธิภาพเหนือกว่าระบบอันดับต้นๆ ในแง่ของคะแนน F1

3.3. SQuAD v2.0

ในการทดสอบ SQuAD v2.0 จะเพิ่มการกำหนดปัญหาจาก SQuAD 1.1 โดยอนุญาตให้ไม่มีคำตอบสั้นๆ ในย่อหน้าที่ให้มา ทำให้สมจริงยิ่งขึ้น

นักวิจัยใช้วิธีง่าย ๆ ในการขยายโมเดล SQuADv1.1 BERT สำหรับการทดสอบนี้ นักวิจัยมองว่าคำถามที่ไม่มีคำตอบเหมือนกับช่วงคำตอบ โดยเริ่มต้นและสิ้นสุดที่ [CLS] token สำหรับการทำนาย จะเปรียบเทียบคะแนนของช่วงที่ไม่มีคำตอบ: $\mathbf{S_{NULL}} = \mathbf{S} \cdot \mathbf{C} + \mathbf{E} \cdot \mathbf{C}$ กับคะแนนของช่วงที่ไม่ว่างที่ดีที่สุด $\boldsymbol{S_{i,j}} = \boldsymbol{max_{j \geq i}}\, \mathbf{S} \cdot \boldsymbol{T_i} + \mathbf{E} \cdot \boldsymbol{T_j}$ นักวิจัยทำนายคำตอบที่ไม่ใช่ค่าว่างเมื่อ $\boldsymbol{S_{i,j}} > \mathbf{S_{NULL}} + \boldsymbol{\tau}$ โดยที่ thresh-old $\boldsymbol{\tau}$ ถูกเลือกบนชุด dev เพื่อเพิ่ม F1 ให้สูงสุด และไม่ได้ใช้ข้อมูล TriviaQA สำหรับแบบจำลองนี้ โดย fine-tuned ที่ 2 epoch earning rate 5e-5 และ batch size 48

ผลลัพธ์นั้นแสดงในตาราง 3 เปรียบเทียบกับโมเดลก่อน ๆ ไม่รวมระบบที่ใช้ BERT เป็นหนึ่งในองค์ประกอบจะเห็นได้ว่าค่า F1 เพิ่มขึ้น +5.1 จากระบบที่ดีที่สุดก่อนหน้านี้

3.4. SWAG

The Situations with Adversarial Generations (SWAG) เป็นชุดข้อมูลที่มีตัวอย่างการเติมประโยคคู่สมบูรณ์ 113,000 ตัวอย่างที่ประเมินการอนุมานสามัญสำนึกที่มีพื้นฐาน [35] เมื่อพิจารณาจากประโยคแล้ว การทดสอบคือเลือกประโยคต่อเนื่องที่เป็นไปได้มากที่สุดจากสี่ตัวเลือก

ในระหว่างการปรับแต่งชุดข้อมูล SWAG อย่างละเอียด ลำดับอินพุตสี่ลำดับจะถูกสร้างขึ้นโดยการรวมประโยคที่กำหนดเข้ากับความต่อเนื่องที่เป็นไปได้ และมีการใช้พารามิเตอร์เฉพาะงานเพื่อคำนวณคะแนนสำหรับแต่ละตัวเลือก

นักวิจัย fine-tune โมเดลใน 3 epoch ด้วย learning rate 2e-5 และ batch size 16 ผลลัพธ์ถูกแสดงที่ตารางที่ 4 $\mathbf{BERT_{LARGE}}$ มีประสิทธิภาพเหนือกว่าระบบ the authors' baseline ESIM+ELMo +27.1% และ OpenAI GPT 8.3 %

4. Ablation Studies

Tasks	Dev Set MNLI-m (Acc)	QNLI (Acc)	MRPC (Acc)	SST-2 (Acc)	SQuAD (F1)
$\text{BERT}_{\text{BASE}}$	84.4	88.4	86.7	92.7	88.5
No NSP	83.9	84.9	86.5	92.6	87.9
LTR & No NSP	82.1	84.3	77.5	92.1	77.8
+ BiLSTM	82.1	84.1	75.7	91.6	84.9

ตารางที่ 5: การตัดออกส่วนของงาน pre-training ใช้ $\textbf{BERT}_{\textbf{BASE}}$ architecture โมเดล “No NSP” ถูก train โดยไม่ใช้งาน next sentence prediction (NSP) 'LTR & No NSP' ถูก train เป็น left-to-right LM โดยไม่ได้ใช้ NSP เช่นเดียวกันกับ OpenAI GPT. + “BiLSTM” เพิ่ม BiLSTM ที่ถูกสุ่มเริ่มต้นไว้ด้านบนของโมเดล “LTR + NoNSP” ในขณะที่กำลัง fine-tuning"

Hyperparams				Dev Set Accuracy		
#L	#H	#A	LM (ppl)	MNLI-m	MRPC	SST-2
3	768	12	5.84	77.9	79.8	88.4
6	768	3	5.24	80.6	82.2	90.7
6	768	12	4.68	81.9	84.8	91.3
12	768	12	3.99	84.4	86.7	92.9
12	1024	16	3.54	85.7	86.9	93.3
24	1024	16	3.23	86.6	87.8	93.7

ตารางที่ 6: การตัดออกส่วนของขนาดโมเดล BERT #L = จำนวน layers; #H = hidden size; #A = จำนวน attention heads “LM (ppl)” คือ ความวุ่นวายของภาษาภายใน (perplexity) จากข้อมูลการฝึกที่ถูกปกป้อง

System	Dev F1	Test F1
ELMo (Peters et al., 2018a)	95.7	92.2
CVT (Clark et al., 2018)	-	92.6
CSE (Akbik et al., 2018)	-	**93.1**
Fine-tuning approach		
$\text{BERT}_{\text{LARGE}}$	96.6	92.8
$\text{BERT}_{\text{BASE}}$	96.4	92.4
Feature-based approach ($\text{BERT}_{\text{BASE}}$)		
Embeddings	91.0	-
Second-to-Last Hidden	95.6	-
Last Hidden	94.9	-
Weighted Sum Last Four Hidden	95.9	-
Concat Last Four Hidden	96.1	-
Weighted Sum All 12 Layers	95.5	-

ตารางที่ 7: ผลลัพธ์ Named Entity Recognition ในข้อมูล CoNLL-2003 พารามิเตอร์ที่ใช้ถูกเลือกด้วยชุดข้อมูล Dev ผลลัพธ์ที่รายงานใน Dev และ Test ถูกคำนวณเฉลี่ยจากการทดลอง 5 ครั้งที่สุ่มเริ่มใหม่โดยใช้พารามิเตอร์เหล่านั้น

4.1. Effect of Pre-training Tasks

นักวิจัยได้แสดงให้เห็นถึงความสำคัญของ deep bidirectionality of BERT โดยการประเมิน 2 pre-training objectives โดยใช้ pre-training data, fine-tuning schema, hyperparameters ที่เหมือนกันทุกประการเป็น $\mathbf{BERT_{BASE}}$

No NSP: bidirectional โมเดลซึ่ง train ด้วย MLM แต่ปราศจากงาน next sentence prediction (NSP)

LTR & No NSP: left-context-only โมเดลซึ่ง train ด้วย standard Left-to-Right(LTR) LM แทนที่จะเป็น MLM ซึ่งข้อจำกัดของ left-only ยังถูกนำไปใช้ใน fine-tuning เนื่องจากการถอดโมเดลออกทำให้เกิดความไม่ตรงกันระหว่างขั้นตอน pre-training และ fine-tuning ซึ่งทำให้ประสิทธิภาพในงาน downstream ลดลง

นอกจากนี้ โมเดล LTR และ No NSP ยังได้รับการ trained โดยไม่มีงาน NSP และเทียบเคียงได้โดยตรงกับโมเดล OpenAI GPT แต่มีชุดข้อมูลการฝึกที่ใหญ่กว่า การแสดงอินพุตที่แตกต่างกัน และรูปแบบการปรับแต่งที่แตกต่างกัน

ในตารางที่ 5 ได้แสดงให้เห็นว่าการลบ NSP จะทำให้ประสิทธิภาพลดลงอย่างมากใน QNLI, MNLI และ SQuAD 1.1 จากนั้นก็ได้ประเมินผลกระทบของการ training bidirectional โดยการเปรียบเทียบ "No NSP" กับ "LTR & No NSP" โมเดล LTR ทำงานได้แย่กว่าโมเดล MLM ในทุกงาน โดยลดลงอย่างมากใน MRPC และ SQuAD

สำหรับการทดสอบ SQuAD จะพบว่า LTR โมเดลนั้นทำงานได้ไม่ดีในการ predict token เพื่อการทำให้ระบบ LTR ดีขึ้นจึงมีการเพิ่ม BiLSTM ที่เริ่มต้นแบบสุ่มไว้ที่ด้านบนของโมเดล LTR จะปรับปรุงผลลัพธ์บน SQuAD แต่ยังคงทำงานได้แย่กว่าโมเดลสองทิศทางที่ได้รับการฝึกไว้ล่วงหน้า อย่างไรก็ตาม BiLSTM ส่งผลเสียต่อประสิทธิภาพในงาน GLUE

การฝึกโมเดล LTR และ RTL แยกกัน และการต่อการนำเสนอเข้าด้วยกันนั้นมีราคาแพงกว่าและมีประสิทธิภาพน้อยกว่าโมเดลสองทิศทางเชิงลึก และมันก็ไม่ง่ายนักสำหรับงานอย่าง QA ซึ่งโมเดล RTL ไม่สามารถกำหนดเงื่อนไขคำตอบของคำถามได้

4.2. Effect of Model size

ในส่วนนี้ได้มีการสำรวจผลกระทบของ model size ต่อความแม่นยำในการ fine-tuning โดยการ trained โมเดล BERT จำนวนหนึ่งด้วย number of layers, hidden units และ attention head ที่แตกต่างกัน ขณะเดียวกันก็ใช้ไฮเปอร์พารามิเตอร์และขั้นตอนการฝึกเดียวกันกับที่อธิบายไว้ก่อนหน้านี้

ผลลัพธ์ของแบบจำลอง BERT ในงาน GLUE ที่เลือก ซึ่งแสดงในตารางที่ 6 ตารางจะรายงานความแม่นยำ Dev Set โดยเฉลี่ยจากการรีสตาร์ทการ fine-tuning แบบสุ่ม 5 ครั้ง ซึ่งหมายความว่าโมเดลได้รับการ trained หลายครั้งด้วยการกำหนดค่าเริ่มต้นแบบสุ่มที่แตกต่างกันเพื่อให้แน่ใจว่าความแม่นยำที่รายงานนั้นเชื่อถือได้

ผลลัพธ์แสดงให้เห็นว่าโมเดลที่ใหญ่ขึ้นนำไปสู่การปรับปรุงความแม่นยำที่ถูกต้องในชุดข้อมูลทั้ง 4 ชุด แม้แต่สำหรับ MRPC ซึ่งมีตัวอย่างการ training ที่มีเพียง 3,600 labeled ตัวอย่าง และแตกต่างอย่างมากจากงาน pre-training

ซึ่งหมายความว่าโมเดล BERT สามารถสรุปงานและชุดข้อมูลใหม่ได้ดี แม้ว่าจะมีข้อมูลที่จำกัดก็ตาม น่าแปลกใจที่การปรับปรุงที่สำคัญดังกล่าวสามารถบรรลุผลได้นอกเหนือจากแบบจำลองที่ค่อนข้างใหญ่อยู่แล้วเมื่อเทียบกับ

งานวิจัยที่มีอยู่ ตัวอย่างเช่น Transformer ที่ใหญ่ที่สุดที่สำรวจใน [28] (L=6, H=1024, A=16) มีพารามิเตอร์ 100M สำหรับตัวเข้ารหัส ในขณะที่ Transformer ที่ใหญ่ที่สุดที่พบในวรรณกรรมมีพารามิเตอร์ 235M (L=64, H=512, A=2) [36] ในทางตรงกันข้าม $\mathbf{BERT_{BASE}}$ มีพารามิเตอร์ 110M และ $\mathbf{BERT_{LARGE}}$ มีพารามิเตอร์ 340M ซึ่งแสดงให้เห็นว่าโมเดล BERT สามารถบรรลุผลลัพธ์ที่ล้ำสมัยด้วยพารามิเตอร์ที่น้อยกว่าโมเดลอื่นๆ ในงานวิจัย

เป็นที่ทราบกันมานานแล้วว่าการเพิ่มขนาดโมเดลจะนำไปสู่การปรับปรุงงานขนาดใหญ่อย่างต่อเนื่อง เช่น machine translation และ language modeling ซึ่งแสดงให้เห็นโดยความงุนงงของ LM ของข้อมูลการ training ที่ถูกจัดขึ้น ดังแสดงในตารางที่ 6 อย่างไรก็ตามนักวิจัยเชื่อว่างานของพวกเขาเป็นงานแรกที่แสดงให้เห็นอย่างน่าเชื่อว่าการปรับขนาดโมเดลให้ถึงขีดสุดสามารถนำไปสู่การปรับปรุงครั้งใหญ่ในงานที่มีสเกลขนาดเล็กมากได้ โดยมีเงื่อนไขว่าโมเดลนั้นได้รับการ pre-training อย่างเพียงพอ

การศึกษาก่อนหน้าโดย [2] และ [37] แสดงให้เห็นผลลัพธ์ที่หลากหลายเกี่ยวกับผลกระทบของการเพิ่มขนาด bi-LM ที่ได้รับการ pre-training และขนาด hidden dimension ในงาน downstream ใน NLP โดยมีสมมติฐานว่าเมื่อโมเดลได้รับการ fine-tuning โดยตรงบนงาน downstream ด้วยพารามิเตอร์เพิ่มเติมจำนวนเล็กน้อย โมเดลเฉพาะงานจะได้รับประโยชน์จากการนำเสนอที่ได้รับการฝึกอบรมล่วงหน้าที่มีขนาดใหญ่กว่าและแสดงออกได้มากขึ้น แม้ว่าจะมีข้อมูลเฉพาะงานจำกัดก็ตามซึ่งชี้ให้เห็นว่าการ pre-training จะเป็นประโยชน์สำหรับงานขนาดเล็ก ตราบใดที่โมเดลได้รับ pre-trained อย่างเพียงพอ

4.3. Feature-based Approach with BERT

ผลลัพธ์ของ BERT ทั้งหมดที่นำเสนอจนถึงตอนนี้ได้ใช้วิธี fine-tuning ช่วยให้สามารถแยกคุณลักษณะคงที่ออกจากโมเดลที่ได้รับการ trained มาแล้ว โดยให้ข้อดี เช่น ความสามารถในการจัดการงานที่ไม่สามารถแสดงได้อย่างง่ายดายด้วย Transformer encoder architecture และประโยชน์ทางการคำนวณของการแสดงค่า pre-computing ที่สิ้นเปลืองของข้อมูลการฝึกอบรม

ในส่วนนี้นักวิจัยได้เปรียบเทียบสองวิธีในการใช้ BERT กับงาน CoNLL-2003 Named Entity Recognition (NER) [38] วิธีแรกใช้ BERT กับโมเดล Word Piece ที่รักษาตัวพิมพ์เล็กและตัวพิมพ์ใหญ่ และรวมบริบทของเอกสารสูงสุด

วิธีที่สอง คือวิธี feature-based ที่แยกการเปิดใช้งานจาก BERT หนึ่งเลเยอร์ขึ้นไปโดยไม่ต้อง fine-tuning พารามิเตอร์ใดๆ การ embeddings ตามบริบทเหล่านี้จะใช้เป็นอินพุตไปยัง BiLSTM two-layer 768-dimensional ที่เริ่มต้นแบบสุ่มก่อนเลเยอร์ classification layer

ผลลัพธ์แสดงไว้ในตารางที่ 7 BERTLARGE ทำงานอย่างมีความสามารถในการแข่งขันด้วยวิธีการอันล้ำสมัย วิธีที่มีประสิทธิภาพดีที่สุดจะเชื่อมโยงการแสดงโทเคนจาก hidden layer อยู่สี่อันดับแรกของ Transformer ที่ได้รับการ pre-trained ซึ่งอยู่เบื้องหลังการ fine-tuning ทั้งหมดเพียง 0.3 F1 สิ่งนี้แสดงให้เห็นว่า BERT มีประสิทธิภาพทั้งสำหรับการ fine-tuning และ feature-based

5. Conclusion

การพัฒนาที่เป็นที่ประจักษ์ล่าสุด เนื่องจากการ transfer learning ด้วย LM ได้แสดงให้เห็นถึงการ pre-training ที่สมบูรณ์และ unsupervised เป็นส่วนสำคัญของระบบการทำความเข้าใจภาษาต่างๆ โดยเฉพาะอย่างยิ่ง ผลลัพธ์เหล่านี้ช่วยให้งานที่ใช้ทรัพยากรต่ำได้รับประโยชน์จาก unidirectional architecture เชิงลึก ผลงานที่สำคัญของงานวิจัยนี้คือการสรุปการค้นพบเหล่านี้ให้เป็นแบบ bidirectional architecture เชิงลึกซึ่งช่วยให้โมเดลที่ได้รับการ pre-trained แบบเดียวกันสามารถจัดการกับงาน NLP ในวงกว้างได้สำเร็จ

References

[1] Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In Advances in neural information processing systems, pages 3079–3087.

[2] Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018a. Deep contextualized word representations. In NAACL.

[3] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding with unsupervised learning. Technical report, OpenAI.

[4] Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In ACL. Association for Computational Linguistics.

[5] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large, annotated corpus for learning natural language inference. In EMNLP. Association for Computational Linguistics.

[6] Adina Williams, Nikita Nangia, and Samuel R Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In NAACL.

[7] William B Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In Proceedings of the Third International Workshop on Paraphrasing (IWP2005).

[8] Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In CoNLL.

[9] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 2383–2392.

[10] Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. 1992. Class-based n-gram models of natural language. Computational linguistics, 18(4):467–479.

[11] Rie Kubota Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasksand unlabeled data. Journal of Machine Learning Research, 6(Nov):1817–1853.

[12] John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In Proceedings of the 2006 conference on empirical methods in natural language processing, pages 120–128. Association for Computational Linguistics.

[13] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In Advances in Neural Information Processing Systems 26, pages 3111–3119. Curran Associates, Inc.

[14] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In Empirical Methods in Natural Language Processing (EMNLP), pages 1532–1543.

[15] Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, ACL '10, pages 384–394.

[16] Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In Advances in neural information processing systems, pages 3294–3302.

[17] Lajanugen Logeswaran and Honglak Lee. 2018. An efficient framework for learning sentence representations. In International Conference on Learning Representations.

[18] Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In International Conference on Machine Learning, pages 1188–1196.

[19] Yacine Jernite, Samuel R. Bowman, and David Sontag. 2017. Discourse-based objectives for fast unsupervised sentence representation learning. CoRR, abs/1705.00557.

[20] Lajanugen Logeswaran and Honglak Lee. 2018. An efficient framework for learning sentence representations. In International Conference on Learning Representations.

[21] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In Proceedings of the 2013 conference on empirical methods in natural language processing, pages 1631–1642.

[22] Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In Proceedings of the 25th international conference on Machine learning, pages 160–167. ACM.

[23] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018a. Glue: A multi-task benchmark and analysis platform

[24] Alexis Conneau, Douwe Kiela, Holger Schwenk, Lo¨ıc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 670–680, Copenhagen, Denmark. Association for Computational Linguistics.

[25] Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In NIPS.

[26] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. FeiFei. 2009. ImageNet: A Large-Scale Hierarchical Image Database. In CVPR09.

[27] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How transferable are features in deep neural networks? In Advances in neural information processing systems, pages 3320–3328.

[28] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In Advances in Neural Information Processing Systems, pages 6000–6010.

[29] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. arXiv preprint arXiv:1609.08144.

[30] Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In Proceedings of the IEEE international conference on computer vision, pages 19–27.

[31] Detailed Descriptions for the GLUE Benchmark Experiments. Our GLUE results in Table 1 are obtained from https://gluebenchmark.com/ leaderboard and https://blog.openai.com/language-unsupervised. The GLUE benchmark includes the following datasets, the descriptions of which were originally summarized in Wang et al. (2018a):

[32] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In ICLR.

[33] Christopher Clark and Matt Gardner. 2018. Simple and effective multi-paragraph reading comprehension. In ACL.

[34] Minghao Hu, Yuxing Peng, Zhen Huang, Xipeng Qiu, Furu Wei, and Ming Zhou. 2018. Reinforced mnemonic reader for machine reading comprehension. In IJCAI.

[35] Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. 2018. Swag: A large-scale adversarial dataset for grounded commonsense inference. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP).

[36] Rami Al-Rfou, Dokook Choe, Noah Constant, Mandy Guo, and Llion Jones. 2018. Character-level language modeling with deeper self-attention. arXiv preprint arXiv:1808.04444.

[37] Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. context2vec: Learning generic context embedding with bidirectional LSTM. In CoNLL.

[38] Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In CoNLL.