Machine Learning (CE 40717)

Fall 2024

Ali Sharifi-Zarchi

CE Department
Sharif University of Technology

December 21, 2024

1. Encoder Architecture
2. References

1. Encoder Architecture
   - Language Modeling
   - BERT Introduction
   - BERT Architecture
2. References

Introduction to Language Modeling

Language Modeling:

- Language modeling involves predicting the probability of a sequence of words.
- Given a sequence $x = \{x_1, x_2, \ldots, x_n\}$, the probability of the entire sequence can be decomposed into the product of conditional probabilities of each word, given the context.

Mathematical Representation:

$$P(x) = \prod_{i=1}^{n} P(x_i \mid x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$$

- $P(x)$: The probability of the entire sequence x.
- Each word x_i depends on all other words in the sequence, including its left and right context.
- This approach captures the dependencies between words, which is essential for understanding language semantics.

Encoder Language Model

Encoder language models, like BERT, use masked tokens to learn bidirectional representations of text.

- **Masked Language Modeling (MLM):** Predicts randomly masked tokens in a sequence.
- **Bidirectional Context:** Considers information from both directions for each token.
- **Applications:** Used for classification, NER(Named entity recognition), and other NLP tasks.

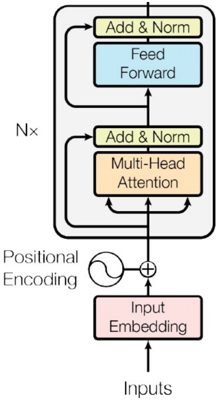
BERT: Key Contributions

- It is a model based on a deep Transformer Encoder that can be fine-tuned for specific tasks.
- The key: learn representations based on **bidirectional context**

  Why? Because both left and right contexts are important to understand the meaning of words.

  Example #1: we went to the river bank.
  Example #2: I need to go to bank to make a deposit.
- **Pre-training objectives:** masked language modeling + next sentence prediction
- State-of-the-art performance on a large set of **sentence-level** and **token-level** tasks

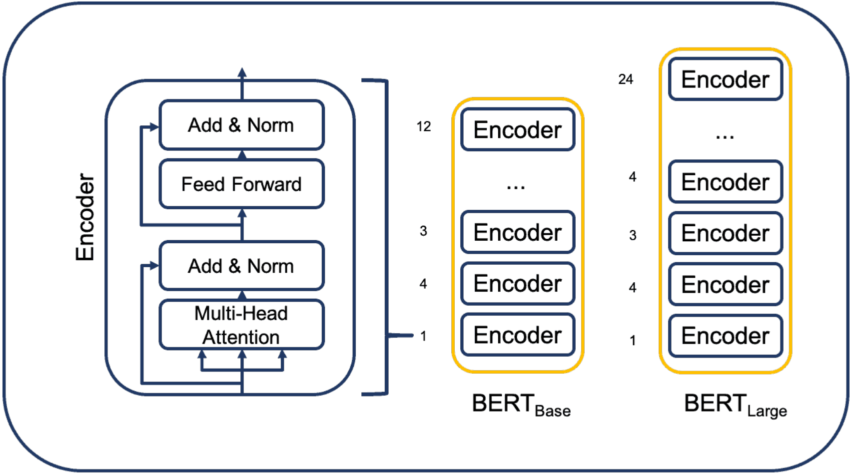
BERT Models



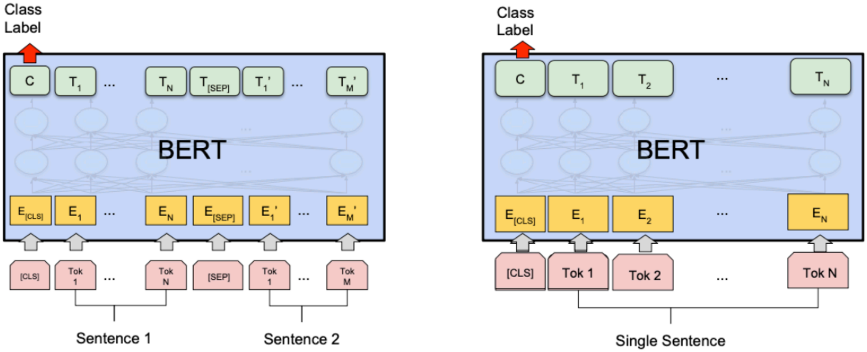
https://arxiv.org/abs/1706.03762

- **BERT-Base:** 12 layers, 768 hidden size, 12 attention heads, 110M parameters
- **BERT-Large:** 24 layers, 1024 hidden size, 16 attention heads, 340M parameters
- **Training corpus:** Wikipedia (2.5B words) + BooksCorpus (0.8B words)
- **Max sequence size:** 512 tokens (sub-word units). For tasks involving two input sequences, this typically includes 256 tokens for each sequence.
- **Training duration:** Trained for 1 million optimization steps (iterations), with a batch size of 128,000 tokens per step.

BERT Base vs BERT Large



Sentence-Level Tasks



(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

(b) Single Sentence Classification Tasks: SST-2, CoLA

https://arxiv.org/abs/1810.04805

Sentence-Level Tasks(cont.)

- Sentence pair classification tasks:

  MNLI

  - **Premise:** A soccer game with multiple males playing.
  - **Hypothesis:** Some men are playing a sport.
  - Result: {entailment, contradiction, neutral}

  QQP

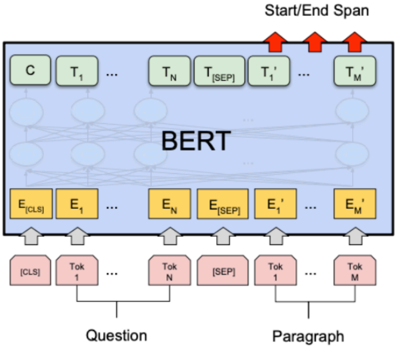
  - Q1: Where can I learn to invest in stocks?
  - Q2: How can I learn more about stocks?
  - Result: {duplicate, not duplicate}

- Single sentence classification tasks:

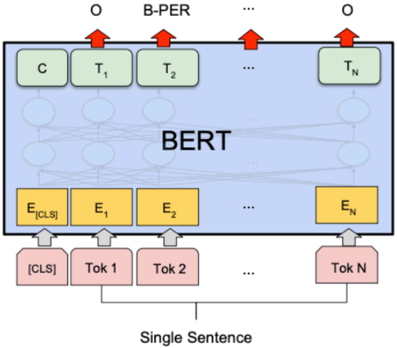
  SST2

  - Sentence: rich veins of funny stuff in this movie
  - Result: {positive, negative}

Token-Level Tasks



(c) Question Answering Tasks: SQuAD v1.1



(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

https://arxiv.org/abs/1810.04805

Token-Level Tasks: Extractive Question Answering

- Extractive question answering e.g., SQuAD (Rajpurkar et al., 2016)

SQuAD

Question: The New York Giants and the New York Jets play at which stadium in NYC ?

Context: The city is represented in the National Football League by the New York Giants and the New York Jets , although both teams play their home games at MetLife Stadium in nearby East Rutherford , New Jersey , which hosted Super Bowl XLVIII in 2014 .

(Training example 29,883)

Example Result: MetLife Stadium

Token-level tasks: Named Entity Recognition

Token-level tasks

- Named entity recognition (Tjong Kim Sang and De Meulder, 2003)

CoNLL 2003 NER

John	Smith	lives	in	New	York
B-PER	I-PER	O	O	B-LOC	I-LOC

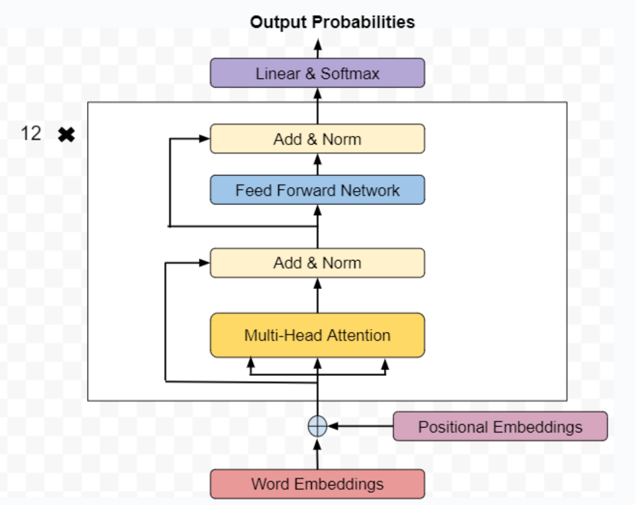
BERT Architecture Overview

- Transformer Encoder Stack
- Positional Encodings
- Special Tokens
- Pretraining Details
- Fine-Tuning Details
- Training BERT
- Optimizations and Variants

Transformer Encoder Stack

- BERT uses an encoder-only architecture.
- Consists of multiple identical layers.
- Each layer contains:
  1. Multi-Head Self-Attention
  2. Feed-Forward Network (FFN)
  3. Residual Connections and Layer Normalization

Transformer Encoder Stack

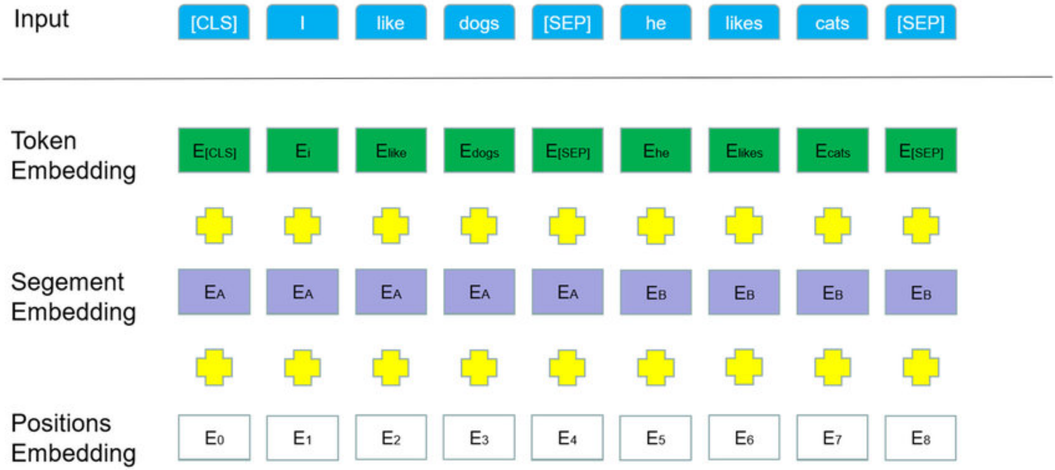


Input Embedding Layer

- Combines three types of embeddings:
  1. **Token Embeddings**: WordPiece embeddings for tokens.
  2. **Segment Embeddings**: Distinguishes sentence pairs (e.g., [0, 0, 0, 1, 1]).
  3. **Positional Embeddings**: Adds positional encodings for sequence order.
- Final input to each layer:

$$E = \text{TokenEmbedding} + \text{SegmentEmbedding} + \text{PositionalEmbedding}$$

Input Embedding Layer



Multi-Head Self-Attention

- Key innovation for contextual representation.
- Computes pairwise attention scores between all tokens.
- For each head:
  1. Learnable matrices: W_Q, W_K, W_V
  2. Project embeddings into queries (Q), keys (K), and values (V).
- Attention computation:

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

- Combine attention-weighted values:

$$\text{head}_i = A \cdot V$$

- Concatenate outputs from all heads:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h) \cdot W_O$$

Feed-Forward Network (FFN)

- Each layer includes a point-wise FFN applied to each token embedding.
- FFN formula:

$$\mathrm{FFN}(x) = \mathrm{ReLU}(xW_1 + b_1)W_2 + b_2$$

- Operates independently on each token.
- Shares parameters across all tokens.

Residual Connections and Layer Normalization

- Residual connections are added around:
  1. Self-Attention layer
  2. FFN layer
- Layer normalization is applied to ensure stable gradient flow.

Positional Encodings

- Encodes sequence order into embeddings.
- Sinusoidal positional encoding formulas:

$$\mathrm{PE}(pos, 2i) = \sin\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right)$$

$$\mathrm{PE}(pos, 2i+1) = \cos\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right)$$

Special Tokens

- **[CLS]**:
  - Special classification token prepended to every input.
  - Used as a global representation for tasks like classification.
- **[SEP]**:
  - Separator token used for segmenting sentences in NSP or marking sequence ends.

Pretraining Details

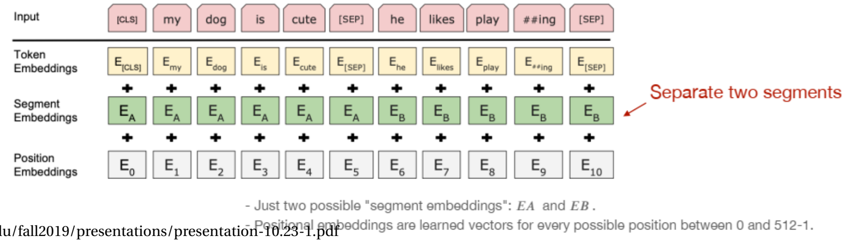
- Two main objectives:
  1. Masked Language Modeling (MLM)
  2. Next Sentence Prediction (NSP)

BERT Pre-training: Putting Together

- **Vocabulary size:** 30,000 wordpieces (common sub-word units) (Wu et al., 2016)

(Image: Stanford CS224N)

- **Input embeddings:**



https://cs330.stanford.edu/fall2019/presentations/presentation-10.23-1.pdf

BERT Pre-training: Putting Together

- MLM and NSP are trained together
- `[CLS]` is pre-trained for NSP
- Other token representations are trained for MLM

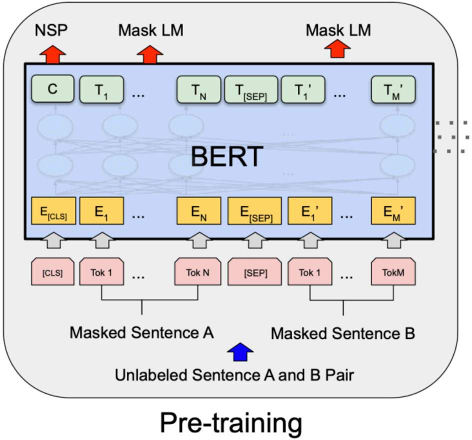


Figure 1: *

https://arxiv.org/abs/1810.04805

Masked Language Modeling (MLM)

- **Q:** Why we can't do language modeling with bidirectional models?

- **Solution:** Mask out a percentage k of the input words, and then predict the masked words.

$$\begin{array}{cccc} & \textbf{store} & & \textbf{gallon} \\ & \downarrow & & \downarrow \\ \text{the man went to} & [\mathit{MASK}] & \text{to buy a} & [\mathit{MASK}] \text{ of milk} \end{array}$$

MLM: Masking Rate and Strategy

- **Q: What is the value of k?**
  - They always use $k = 15\%$.
  - Too little masking: computationally expensive (we need to increase # of epochs)
  - Too much masking: not enough context
  - See (Wettig et al., 2022) for more discussion of masking rates:
    - Masking 40% outperforms 15% for BERT-large size models on GLUE and SQuAD
    - High masking rate of 80% can still preserve 95% fine-tuning performance

- **Q: How are masked tokens selected?**
  - 15% tokens are uniformly sampled
  - Is it optimal? See span masking (Joshi et al., 2020) and PMI masking (Levine et al., 2021)

Example: He **[MASK]** from Kuala **[MASK]**, Malaysia.

Masked Language Modeling (MLM)

- **Masking Strategy**:
  1. 15% of tokens are randomly selected for masking.
  2. 80% replaced with [MASK].
  3. 10% replaced with a random token.
  4. 10% unchanged.
- Prevents model from overfitting to [MASK].
- **Loss Function**:

$$L_{MLM} = -\sum_{t \in \text{masked}} \log P(t_{\text{true}} \mid \text{context})$$

Next Sentence Prediction (NSP)

- Motivation: many NLP downstream tasks require understanding the relationship between two sentences (natural language inference, paraphrase detection, QA).
- NSP is designed to reduce the gap between pre-training and fine-tuning.

[CLS]: a special token always at the beginning

[SEP]: a special token used to separate two segments

Input = `[CLS] the man went to [MASK] store [SEP]`

`he bought a gallon [MASK] milk [SEP]`

Label = `IsNext`

Input = `[CLS] the man [MASK] to the store [SEP]`

`penguin [MASK] are flight ##less birds [SEP]`

Label = `NotNext`

They sample two contiguous segments for 50% of the time and another random segment from the corpus for 50% of the time

Next Sentence Prediction (NSP)

- 50% of training pairs are consecutive sentences (labeled as **IsNext**).
- 50% are randomly paired sentences (labeled as **NotNext**).
- **NSP Objective**:
  - Binary classification loss applied to the [CLS] representation.

Fine-Tuning Details

- Requires task-specific modifications.
- Examples include:
  1. Text Classification
  2. Named Entity Recognition (NER)
  3. Question Answering (QA)
  4. Sentence Pair Classification

Fine-Tuning for Text Classification

- **Objective**: Classify input text into predefined categories (e.g., sentiment analysis, topic classification).
- **Approach**:
  - Utilize BERT's [CLS] token embedding from the last hidden layer as a summary representation of the input.
  - Add a linear (dense) classification layer on top of the [CLS] embedding.
- **Model Architecture**:

$$y = \text{softmax}(W \cdot h_{[\text{CLS}]} + b)$$

  - $h_{[\text{CLS}]}$: Hidden state of the [CLS] token.
  - W, b: Weights and bias of the classification layer.
  - y: Probability distribution over the target classes.
- **Loss Function**:

$$L_{TC} = -\sum_{i=1}^{C} y_i^{\text{true}} \log P(y_i \mid h_{[\text{CLS}]})$$

Fine-Tuning for Named Entity Recognition (NER)

- **Objective**: Identify and classify named entities (e.g., persons, organizations, locations) in text.
- **Approach**:
  - Utilize BERT's token embeddings from the last hidden layer.
  - Add a linear classification layer to predict entity labels for each token.
- **Model Architecture**:

$$y_i = \text{softmax}(W \cdot h_i + b)$$

  - h_i: Hidden state of the i-th token.
  - W, b: Weights and bias of the classification layer.
  - y_i: Probability distribution over entity labels for token i.
- **Training Details**:
  - **Label Encoding**: Use BIO (Begin, Inside, Outside) tagging scheme.
  - **Loss Function**: Cross-entropy loss computed over all tokens.

Fine-Tuning for Question Answering (QA)

- **Objective**: Predict the start and end positions of the answer span within a given context.
- **Approach**:
  - Use BERT's token embeddings from the last hidden layer.
  - Add two linear layers to predict start and end positions separately.
- **Model Architecture**:

$$\text{Start}_i = \text{softmax}(W_{start} \cdot h_i + b_{start})$$

$$\text{End}_i = \text{softmax}(W_{end} \cdot h_i + b_{end})$$

  - h_i: Hidden state of the i-th token.
  - $W_{start}, W_{end}, b_{start}, b_{end}$: Weights and biases for start and end prediction layers.
  - Start_i, End_i: Probability distributions for start and end positions.
- **Loss Function**:

$$L_{QA} = -\left(\log P(\text{start\_true} \mid \text{context}) + \log P(\text{end\_true} \mid \text{context})\right)$$

Fine-Tuning for Sentence Pair Classification

- **Objective**: Determine the relationship between two sentences (e.g., entailment, contradiction, or similarity).
- **Approach**:
  - Input consists of two sentences separated by the [SEP] token.
  - Use the [CLS] token's embedding for classification.
  - Add a linear classification layer on top of the [CLS] embedding.
- **Model Architecture**:

$$y = \text{softmax}(W \cdot h_{[\text{CLS}]} + b)$$

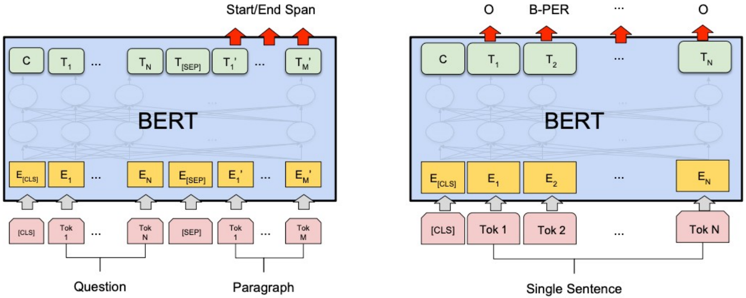
  - $h_{[\text{CLS}]}$: Hidden state of the [CLS] token.
  - W, b: Weights and bias of the classification layer.
  - y: Probability distribution over relationship classes.
- **Loss Function**:

$$L_{Pair} = -\sum_{i=1}^{C} y_i^{\text{true}} \log P(y_i \mid h_{[\text{CLS}]})$$

Fine-tuning BERT

"Pre-train once, finetune many times."

token-level tasks



(c) Question Answering Tasks: SQuAD v1.1

(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

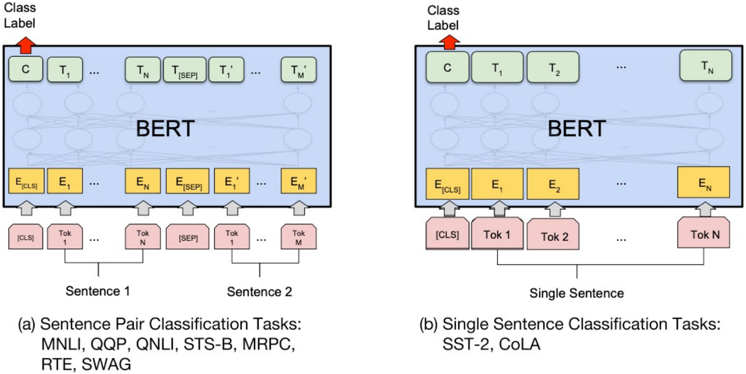
For token-level prediction tasks, add linear classifier on top of hidden representations

Q: How many new parameters?

Fine-tuning BERT

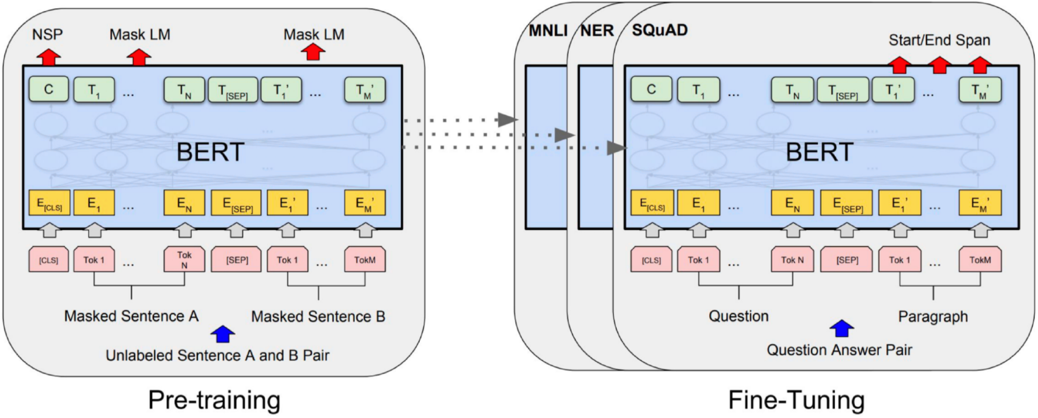
"Pre-train once, finetune many times."

sentence-level tasks



For sentence pair tasks, use [SEP] to separate the two segments with segment embeddings and add a linear classifier on top of [CLS] representation.

Finetuning Paradigm in NLP



BERT Training

Dataset: Let $\mathscr{D}$ be a set of examples $(x_{1:L}, c)$ constructed as follows:

- Let A be a sentence from the corpus.
- With probability 0.5, let B be the next sentence.
- With probability 0.5, let B be a random sentence from the corpus.
- Let $x_{1:L} = [\text{CLS}], A, [\text{SEP}], B$.
- Let c denote whether B is the next sentence or not.

Objective. Then the BERT objective is:

$$\mathcal{O}(\theta) = \sum_{(x_{1:L},c)\in D} \underbrace{\mathbb{E}_{I,\tilde{x}_{1:L}\sim A(\cdot|x_{1:L},I)}\left[\sum_{i\in I} -\log p_\theta(\tilde{x}_i \mid x_{1:L})\right]}_{\text{masked language modeling}} + \underbrace{-\log p(c \mid \phi(x_{1:L})_1)}_{\text{next sentence prediction}}.$$

Training BERT - Hyperparameters

- **Optimizer**: AdamW (Adam with weight decay).
- **Warmup Steps**: Gradual learning rate increase during early steps.
- **Learning Rate**:

$$[10^{-5}, 10^{-4}]$$

  for fine-tuning.
- **Batch Size**: 16–32 for fine-tuning.

Training BERT - Regularization

- **Dropout**: Applied to attention scores and FFN (typical rates: 0.1–0.3).
- **Weight Decay**: Helps generalization during pretraining.

Computational Complexity

- Attention mechanism scales quadratically:

$$\mathcal{O}(n^2 d)$$

where:

  - n = sequence length
  - d = hidden size

Optimizations and Variants

1. **DistilBERT**
   - Lighter version with fewer parameters.
   - Retains 97% of performance with 40% fewer parameters.
2. **ALBERT**
   - Reduces memory overhead by parameter sharing across layers.
   - Decomposes embeddings.
3. **RoBERTa**
   - Removes NSP.
   - Trains on larger datasets.
   - Uses dynamic masking.
4. **Longformer**
   - Modifies attention to handle long sequences efficiently.
   - Uses sparse attention mechanisms.

DistilBERT

- A lighter version of BERT.
- Fewer parameters, leading to faster training and inference.
- Maintains approximately 97% of BERT's performance.
- Reduces parameter count by 40%.

ALBERT

- Aimed at reducing memory footprint.
- **Parameter Sharing**:
  - Shares parameters across all layers.
- **Embedding Factorization**:
  - Decomposes the embedding matrix to reduce size.

RoBERTa

- An improved version of BERT.
- Key improvements:
  1. Removal of Next Sentence Prediction (NSP).
  2. Training on larger datasets.
  3. Implementation of dynamic masking during training.

Longformer

- Designed to handle long sequences efficiently.
- Modifies the attention mechanism to use sparse attention.
- Reduces computational complexity from $\mathcal{O}(n^2 d)$ to linear or near-linear.

These slides were prepared with contributions from Aren Golazizian

1. Encoder Architecture
2. References

References I

- Asgari, E. "Natural language processing." Sharif University of Technology.
- Soleymani, M. "Machine learning." Sharif University of Technology.
- Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I. (2019). "Language Models are Unsupervised Multitask Learners." OpenAI Blog. Retrieved from https://openai.com/research/language-unsupervised
- Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... Amodei, D. (2020). "Language Models are Few-Shot Learners." *arXiv preprint arXiv:2005.14165.*
- Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., ... Stoyanov, V. (2019). "RoBERTa: A Robustly Optimized BERT Pre-training Approach." *arXiv preprint arXiv:1907.11692.*
- Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., Soricut, R. (2020). "ALBERT: A Lite BERT for Self-supervised Learning of Language Representations." In *Proceedings of the International Conference on Learning Representations (ICLR).*
- Joshi, M., Chen, D., Liu, Y., Weld, D. S., Zettlemoyer, L., Levy, O. (2020). "SpanBERT: Improving Pre-training by Representing and Predicting Spans." *Transactions of the Association for Computational Linguistics*, vol. 8, pp. 64-77.

References II

- Wettig, A., Baykal, C., Ruder, S., Søgaard, A. (2022). "Should All Tokens be Masked? A Pilot Study of Masked Language Model Performance on Diagnostic Classifiers." *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 10, pp. 10993-11001.
- Tjong Kim Sang, E. F., De Meulder, F. (2003). "Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition." In *Proceedings of the 7th Conference on Natural Language Learning at HLT-NAACL 2003.*
- Radford, A., Narasimhan, K., Salimans, T., Sutskever, I. (2018). "Improving Language Understanding by Generative Pre-Training." Retrieved from https://www.cs.ubc.ca/ amuham01/LING530/papers/radford2018improving.pdf