

CS 489: Introduction to Natural Language Processing

Lecture 9: Masked LMs and Pretraining Objectives

Instructor: Freda Shi

October 8, 2026



UNIVERSITY OF
WATERLOO

Outline

1. The pre-training–fine-tuning paradigm
2. Three model families: decoder-only, encoder-only, encoder-decoder
3. BERT: pretraining objectives and fine-tuning
4. Beyond BERT: RoBERTa and the pretraining landscape

Recap: Where We Are

So far:

- **Lectures 1–5:** Foundations: tokenization, embeddings, supervised learning, statistical LMs.
- **Lecture 6:** Neural network foundations: MLPs, RNNs, LSTMs.
- **Lecture 7:** Transformers from scratch (attention, multi-head attention, encoder-decoder).
- **Lecture 8:** Neural language models: RNN LMs, GPT-style Transformer decoders, training with cross-entropy.

Today: Moving beyond left-to-right language models.

- What if we could use **both** left and right context?
- **Masked language models** (BERT); **encoder-decoder** overview.
- The **landscape of pretraining objectives**: which one should you use?

Motivation: The Limitation of Left-to-Right

Recall: autoregressive (GPT-style) LMs predict each token given **only the left context**:

$$P(x_t \mid x_1, \dots, x_{t-1})$$

Motivation: The Limitation of Left-to-Right

Recall: autoregressive (GPT-style) LMs predict each token given **only the left context**:

$$P(x_t \mid x_1, \dots, x_{t-1})$$

Problem: Many NLP tasks need **bidirectional** understanding.

- “The man went to the _____. He bought a gallon of milk.”
 - Left context alone: *park? store? bank?*
 - With right context: clearly **store**!
- Named entity recognition: is “Washington” a person, a state, or a city?
 - Depends on **surrounding** context, not just preceding words.
- Sentiment analysis: “The movie was not _____ at all.”
 - Need to see what follows “not” **and** what precedes it.

Motivation: The Limitation of Left-to-Right

Recall: autoregressive (GPT-style) LMs predict each token given **only the left context**:

$$P(x_t \mid x_1, \dots, x_{t-1})$$

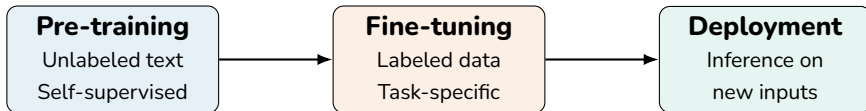
Problem: Many NLP tasks need **bidirectional** understanding.

- “The man went to the _____. He bought a gallon of milk.”
 - Left context alone: *park? store? bank?*
 - With right context: clearly **store**!
- Named entity recognition: is “Washington” a person, a state, or a city?
 - Depends on **surrounding** context, not just preceding words.
- Sentiment analysis: “The movie was not _____ at all.”
 - Need to see what follows “not” **and** what precedes it.

⇒ We want a model that can attend to **all** positions in the input.

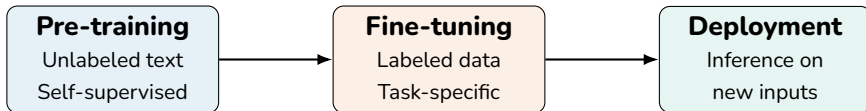
The Pre-Training–Fine-Tuning Paradigm

Core idea: Learn general-purpose representations from **unlabeled text**, then adapt to specific tasks.



The Pre-Training–Fine-Tuning Paradigm

Core idea: Learn general-purpose representations from **unlabeled text**, then adapt to specific tasks.

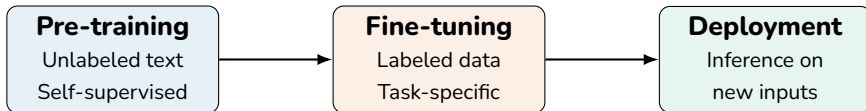


Why does this work?

- Predicting tokens from context forces the model to learn **syntax**, **semantics**, and **world knowledge**.
- This is **self-supervised**: the text itself provides the supervision signal.
- Labeled data is scarce and expensive; unlabeled text is virtually unlimited.

The Pre-Training–Fine-Tuning Paradigm

Core idea: Learn general-purpose representations from **unlabeled text**, then adapt to specific tasks.



Why does this work?

- Predicting tokens from context forces the model to learn **syntax**, **semantics**, and **world knowledge**.
- This is **self-supervised**: the text itself provides the supervision signal.
- Labeled data is scarce and expensive; unlabeled text is virtually unlimited.

Key question: What is the best **pretraining objective**?

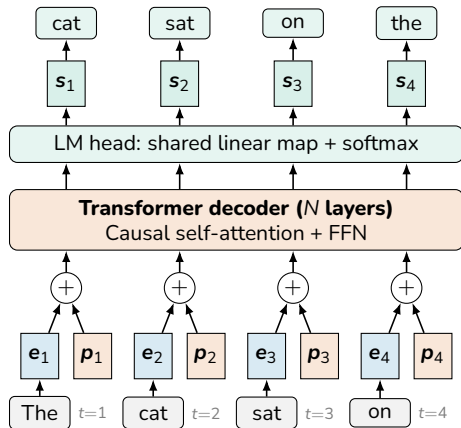
Decoder-Only Models: Autoregressive LMs

Predict the next token from left context.

$$\mathcal{L}_{\text{AR}} = - \sum_{t=1}^T \log P_{\Theta}(x_t \mid x_{<t})$$

- **Causal self-attention:** position t sees $x_{\leq t}$ and predicts x_{t+1} .
- No encoder or cross-attention.
- Natural for left-to-right generation.

Examples: GPT, LLaMA.



e_t : token embedding; p_t : position embedding.

$s_t \in \mathbb{R}^{|V|}$: next-token probability vector.

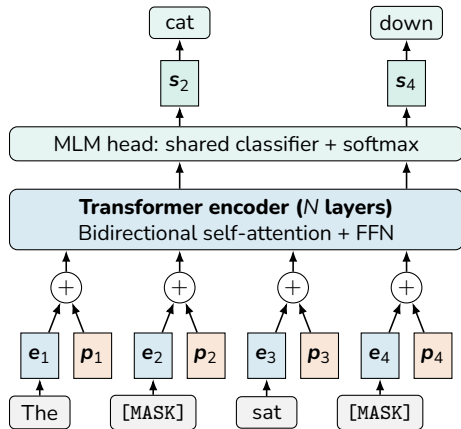
Encoder-Only Models: Masked LMs

Predict missing tokens from both sides.

$$P_{\Theta}(x_i | \tilde{\mathbf{x}})$$

- Mask selected input tokens; predict their original values.
- **Bidirectional self-attention**: use the full corrupted input $\tilde{\mathbf{x}}$.
- Loss only at selected positions.
- Useful for classification and tagging.

Examples: BERT, RoBERTa.



e_i : token embedding; p_i : position embedding.

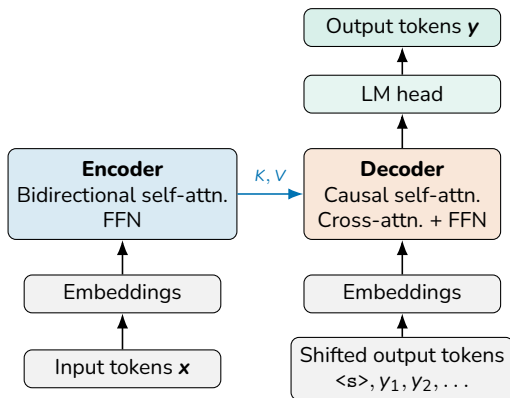
$s_i \in \mathbb{R}^{|V|}$: masked-token probability vector.

Encoder-Decoder Models

Generate an output given an input.

$$P_{\Theta}(\mathbf{y} \mid \mathbf{x}) = \prod_{t=1}^{|\mathbf{y}|} P_{\Theta}(y_t \mid y_{<t}, \mathbf{x})$$

- **Encoder:** Bidirectional input representations.
- **Decoder:** Causal self-attention plus **cross-attention** to the encoder.
- Useful for translation and summarization.



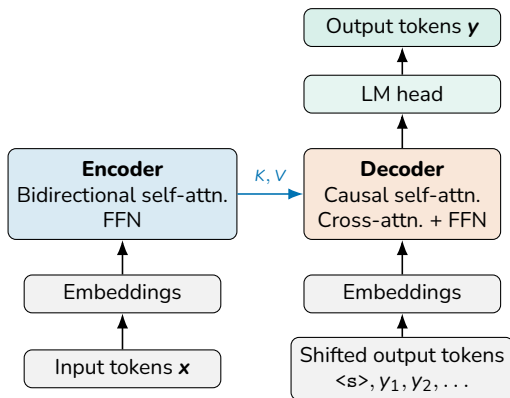
Encoder-Decoder Models

Generate an output given an input.

$$P_{\Theta}(\mathbf{y} \mid \mathbf{x}) = \prod_{t=1}^{|\mathbf{y}|} P_{\Theta}(y_t \mid \mathbf{y}_{<t}, \mathbf{x})$$

- **Encoder:** Bidirectional input representations.
- **Decoder:** Causal self-attention plus **cross-attention** to the encoder.
- Useful for translation and summarization.

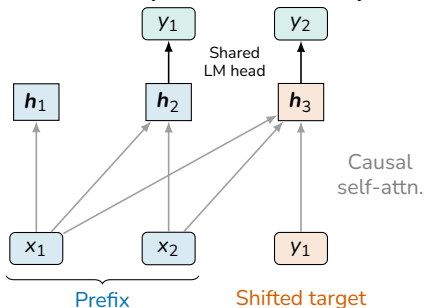
Example: T5 (Raffel et al., 2019) uses a text-to-text interface and pretrains by generating missing spans from corrupted input (**span corruption**).



Prefix-Based Prediction: Two Computation Graphs

Prefix: $\mathbf{x} = (x_1, x_2)$. **Target:** $\mathbf{y} = (y_1, y_2)$. Both model $P(y_t \mid \mathbf{x}, y_{<t})$.

Decoder-only: one causal sequence

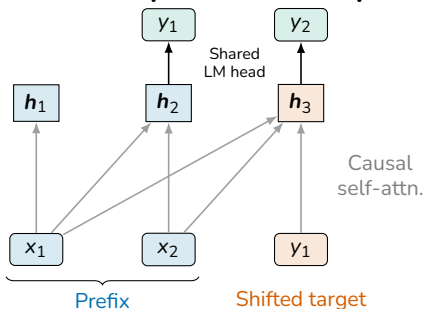


Prefix states are **causal**: h_1 cannot use x_2 . The last prefix position predicts y_1 .

Prefix-Based Prediction: Two Computation Graphs

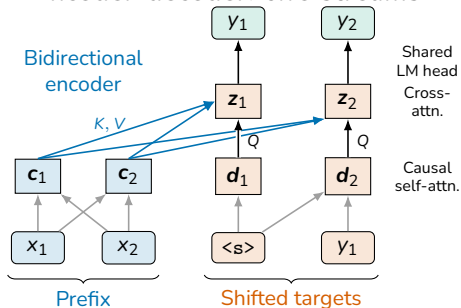
Prefix: $\mathbf{x} = (x_1, x_2)$. **Target:** $\mathbf{y} = (y_1, y_2)$. Both model $P(y_t | \mathbf{x}, y_{<t})$.

Decoder-only: one causal sequence



Prefix states are **causal**: h_1 cannot use x_2 . The last prefix position predicts y_1 .

Encoder-decoder: two streams



Bidirectional prefix states supply K, V to every decoder position via cross-attention.

Three Families of Pretrained Models

	Decoder-only	Encoder-only	Encoder-Decoder
Attention	Causal	Bidirectional	Enc.: bidir.; dec.: causal
Objective	Autoregressive LM	Masked LM	Span corruption
Output	Next token	Token representations	Sequence-to-sequence
Best for	Generation	Understanding	Conditional generation
Examples	GPT, LLaMA, Claude	BERT, RoBERTa	T5, BART

Three Families of Pretrained Models

	Decoder-only	Encoder-only	Encoder-Decoder
Attention	Causal	Bidirectional	Enc.: bidir.; dec.: causal
Objective	Autoregressive LM	Masked LM	Span corruption
Output	Next token	Token representations	Sequence-to-sequence
Best for	Generation	Understanding	Conditional generation
Examples	GPT, LLaMA, Claude	BERT, RoBERTa	T5, BART

Next: A closer look at **BERT**, its pretraining objectives, and fine-tuning.

Why Not Just Use a Bidirectional Autoregressive LM?

Naive idea: Train a left-to-right *and* a right-to-left LM, then combine.

- **ELMo** (Peters et al., 2018) did exactly this:
 - Forward LSTM: $P(x_t \mid x_1, \dots, x_{t-1})$
 - Backward LSTM: $P(x_t \mid x_{t+1}, \dots, x_T)$
 - Concatenate states: $\mathbf{h}_t = [\overrightarrow{\mathbf{h}}_t; \overleftarrow{\mathbf{h}}_t]$

Why Not Just Use a Bidirectional Autoregressive LM?

Naive idea: Train a left-to-right *and* a right-to-left LM, then combine.

- **ELMo** (Peters et al., 2018) did exactly this:
 - Forward LSTM: $P(x_t \mid x_1, \dots, x_{t-1})$
 - Backward LSTM: $P(x_t \mid x_{t+1}, \dots, x_T)$
 - Concatenate states: $\mathbf{h}_t = [\overrightarrow{\mathbf{h}}_t; \overleftarrow{\mathbf{h}}_t]$
- **Problem:** The directions never **interact** during training.
 - The forward model cannot use the backward model's information.
 - The representations are concatenated, not jointly conditioned.

Why Not Just Use a Bidirectional Autoregressive LM?

Naive idea: Train a left-to-right *and* a right-to-left LM, then combine.

- **ELMo** (Peters et al., 2018) did exactly this:
 - Forward LSTM: $P(x_t \mid x_1, \dots, x_{t-1})$
 - Backward LSTM: $P(x_t \mid x_{t+1}, \dots, x_T)$
 - Concatenate states: $\mathbf{h}_t = [\overrightarrow{\mathbf{h}}_t; \overleftarrow{\mathbf{h}}_t]$
- **Problem:** The directions never **interact** during training.
 - The forward model cannot use the backward model's information.
 - The representations are concatenated, not jointly conditioned.

BERT's insight: Use one Transformer with **full bidirectional attention**.

- How do we define a prediction task when every token can see every other token?

Why Not Just Use a Bidirectional Autoregressive LM?

Naive idea: Train a left-to-right *and* a right-to-left LM, then combine.

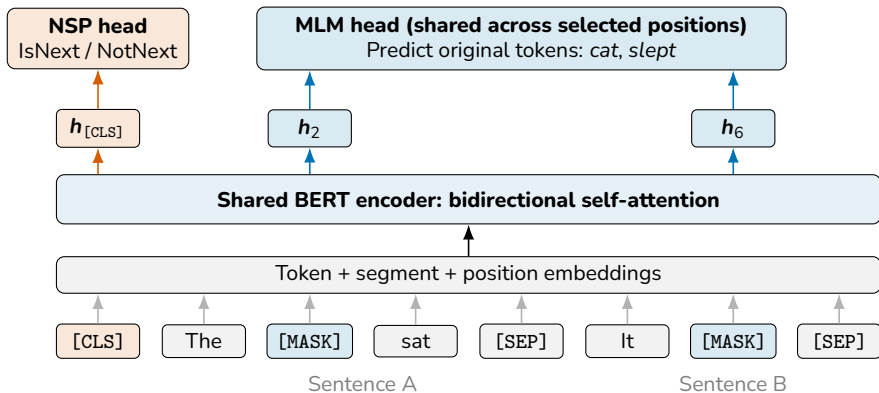
- **ELMo** (Peters et al., 2018) did exactly this:
 - Forward LSTM: $P(x_t \mid x_1, \dots, x_{t-1})$
 - Backward LSTM: $P(x_t \mid x_{t+1}, \dots, x_T)$
 - Concatenate states: $\mathbf{h}_t = [\overrightarrow{\mathbf{h}}_t; \overleftarrow{\mathbf{h}}_t]$
- **Problem:** The directions never **interact** during training.
 - The forward model cannot use the backward model's information.
 - The representations are concatenated, not jointly conditioned.

BERT's insight: Use one Transformer with **full bidirectional attention**.

- How do we define a prediction task when every token can see every other token?
- Mask some tokens and predict them: the MLM objective.

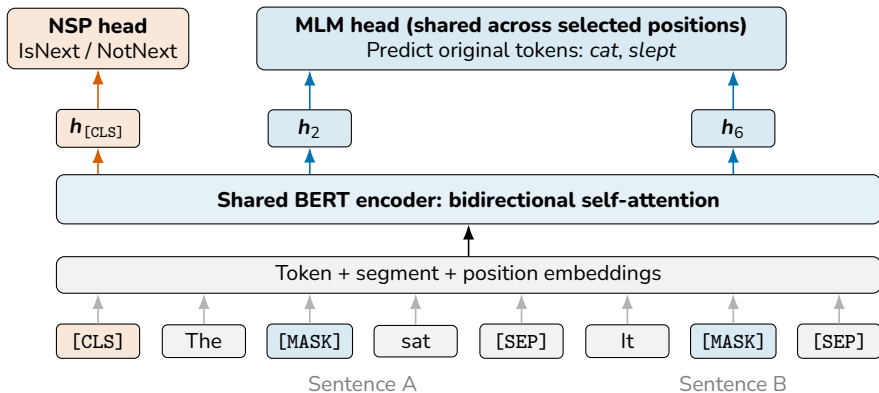
BERT: One Encoder, Two Pretraining Objectives

One shared encoder, with **token-level MLM** and **sentence-pair NSP**.



BERT: One Encoder, Two Pretraining Objectives

One shared encoder, with **token-level MLM** and **sentence-pair NSP**.



$$\mathcal{L}_{\text{BERT}} = \mathcal{L}_{\text{MLM}} + \mathcal{L}_{\text{NSP}}$$

BERT: Bidirectional Encoder Representations from Transformers

BERT (Devlin et al., 2019) introduced two pretraining objectives:

1. Masked Language Model (MLM):

- Randomly select 15% of input tokens.
- Of those selected:
 - 80%: replace with [MASK]
 - 10%: replace with a random token
 - 10%: keep unchanged
- Predict the original tokens at all selected positions.

BERT: Bidirectional Encoder Representations from Transformers

BERT (Devlin et al., 2019) introduced two pretraining objectives:

1. Masked Language Model (MLM):

- Randomly select 15% of input tokens.
- Of those selected:
 - 80%: replace with [MASK]
 - 10%: replace with a random token
 - 10%: keep unchanged
- Predict the original tokens at all selected positions.

2. Next Sentence Prediction (NSP):

- Input: two sentences A and B.
- 50% of the time, B is the actual next sentence.
- 50% of the time, B is a random sentence.
- Binary classification: is B the true continuation?
- Uses the [CLS] token representation.

BERT MLM: Why the 80/10/10 Split?

Recall: 15% of tokens are selected for prediction. Of those:

%	Action	Reason
80%	Replace with [MASK]	Primary signal: predict the masked token from context.
10%	Replace with a random token	Forces attention to context even when the token <i>looks</i> fine.
10%	Keep unchanged	Reconstruct the original token without an explicit mask signal.

BERT MLM: Why the 80/10/10 Split?

Recall: 15% of tokens are selected for prediction. Of those:

%	Action	Reason
80%	Replace with [MASK]	Primary signal: predict the masked token from context.
10%	Replace with a random token	Forces attention to context even when the token <i>looks</i> fine.
10%	Keep unchanged	Reconstruct the original token without an explicit mask signal.

The training and fine-tuning mismatch:

- Fine-tuning inputs contain **no** [MASK] tokens.
- If training used only [MASK], the model would not learn useful representations for real tokens.
- The random and unchanged cases encourage useful representations at **every** position.

BERT MLM: Formal Definition

Given a sequence $\mathbf{x} = (x_1, \dots, x_T)$, let $\mathcal{M} \subset \{1, \dots, T\}$ be the positions selected for prediction ($|\mathcal{M}| \approx 0.15T$).

Let $\tilde{\mathbf{x}}$ be the corrupted input: selected tokens are masked, randomly replaced, or left unchanged.

BERT MLM: Formal Definition

Given a sequence $\mathbf{x} = (x_1, \dots, x_T)$, let $\mathcal{M} \subset \{1, \dots, T\}$ be the positions selected for prediction ($|\mathcal{M}| \approx 0.15T$).

Let $\tilde{\mathbf{x}}$ be the corrupted input: selected tokens are masked, randomly replaced, or left unchanged.

MLM objective:

$$\mathcal{L}_{\text{MLM}} = - \sum_{i \in \mathcal{M}} \log P_{\Theta}(x_i \mid \tilde{\mathbf{x}})$$

Here $P_{\Theta}(x_i \mid \tilde{\mathbf{x}}) = \text{softmax}(\mathbf{W} \cdot \mathbf{h}_i + \mathbf{b})_{x_i}$ and $\mathbf{h}_i$ is the Transformer's output at position i .

BERT MLM: Formal Definition

Given a sequence $\mathbf{x} = (x_1, \dots, x_T)$, let $\mathcal{M} \subset \{1, \dots, T\}$ be the positions selected for prediction ($|\mathcal{M}| \approx 0.15T$).

Let $\tilde{\mathbf{x}}$ be the corrupted input: selected tokens are masked, randomly replaced, or left unchanged.

MLM objective:

$$\mathcal{L}_{\text{MLM}} = - \sum_{i \in \mathcal{M}} \log P_{\Theta}(x_i \mid \tilde{\mathbf{x}})$$

Here $P_{\Theta}(x_i \mid \tilde{\mathbf{x}}) = \text{softmax}(\mathbf{W} \cdot \mathbf{h}_i + \mathbf{b})_{x_i}$ and $\mathbf{h}_i$ is the Transformer's output at position i .

Key difference from autoregressive LM:

- **AR:** $P(x_t \mid x_1, \dots, x_{t-1})$; left context only; predicts all positions.
- **MLM:** $P(x_i \mid \tilde{\mathbf{x}})$; full bidirectional context; predicts only selected positions.

BERT MLM: Formal Definition

Given a sequence $\mathbf{x} = (x_1, \dots, x_T)$, let $\mathcal{M} \subset \{1, \dots, T\}$ be the positions selected for prediction ($|\mathcal{M}| \approx 0.15T$).

Let $\tilde{\mathbf{x}}$ be the corrupted input: selected tokens are masked, randomly replaced, or left unchanged.

MLM objective:

$$\mathcal{L}_{\text{MLM}} = - \sum_{i \in \mathcal{M}} \log P_{\Theta}(x_i | \tilde{\mathbf{x}})$$

Here $P_{\Theta}(x_i | \tilde{\mathbf{x}}) = \text{softmax}(\mathbf{W} \cdot \mathbf{h}_i + \mathbf{b})_{x_i}$ and $\mathbf{h}_i$ is the Transformer's output at position i .

Key difference from autoregressive LM:

- **AR:** $P(x_t | x_1, \dots, x_{t-1})$; left context only; predicts all positions.
- **MLM:** $P(x_i | \tilde{\mathbf{x}})$; full bidirectional context; predicts only selected positions.

Trade-off: MLM uses only $\sim 15\%$ of tokens for the loss, so it generally needs more data or epochs than an autoregressive LM, which uses every token.

Autoregressive vs. Masked: A Deeper Comparison

	Autoregressive LM (GPT)	Masked LM (BERT)
Probability	Valid $P(\mathbf{x})$	Separate conditionals; no $P(\mathbf{x})$
Training signal	All next tokens; left context	~15% selected positions; bidirectional
Generation	Natural	Iterative and awkward

Autoregressive vs. Masked: A Deeper Comparison

	Autoregressive LM (GPT)	Masked LM (BERT)
Probability	Valid $P(\mathbf{x})$	Separate conditionals; no $P(\mathbf{x})$
Training signal	All next tokens; left context	$\sim 15\%$ selected positions; bidirectional
Generation	Natural	Iterative and awkward

Why no joint probability for BERT? It predicts selected positions separately. The product $\prod_{i \in \mathcal{M}} P(x_i \mid \tilde{\mathbf{x}})$ is not a valid joint distribution when masked tokens are correlated, for example:

$$P(\text{New, York}) \neq P(\text{New}) \cdot P(\text{York}).$$

The Conditional Independence Assumption

BERT's implicit approximation:

$$P(\mathbf{x}_{\mathcal{M}} \mid \mathbf{x}_{\setminus \mathcal{M}}) \approx \prod_{i \in \mathcal{M}} P(x_i \mid \mathbf{x}_{\setminus \mathcal{M}})$$

Each selected token is predicted **independently** given the unmasked context.

The Conditional Independence Assumption

BERT's implicit approximation:

$$P(\mathbf{x}_{\mathcal{M}} \mid \mathbf{x}_{\setminus \mathcal{M}}) \approx \prod_{i \in \mathcal{M}} P(x_i \mid \mathbf{x}_{\setminus \mathcal{M}})$$

Each selected token is predicted **independently** given the unmasked context.

Why this can fail:

Input:	"I live in [MASK] [MASK]"
Plausible answers:	"I live in New York " or "I live in San Francisco "
Independent prediction:	"New" and "Francisco" → "New Francisco"

The Conditional Independence Assumption

BERT's implicit approximation:

$$P(\mathbf{x}_{\mathcal{M}} \mid \mathbf{x}_{\setminus \mathcal{M}}) \approx \prod_{i \in \mathcal{M}} P(x_i \mid \mathbf{x}_{\setminus \mathcal{M}})$$

Each selected token is predicted **independently** given the unmasked context.

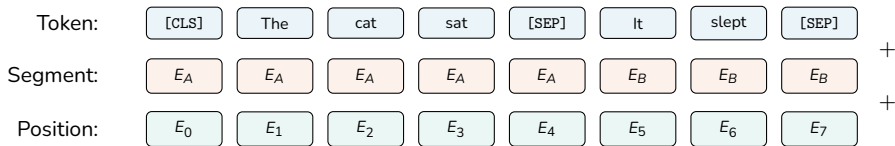
Why this can fail:

Input:	"I live in [MASK] [MASK]"
Plausible answers:	"I live in New York " or "I live in San Francisco "
Independent prediction:	"New" and "Francisco" → "New Francisco"

Practical implication: This is usually acceptable for classification fine-tuning, which does not generate text. For generation, autoregressive models are preferable.

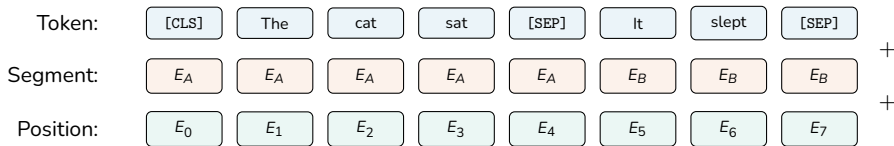
BERT: Input Representation

BERT's input is the sum of three embeddings:



BERT: Input Representation

BERT's input is the sum of three embeddings:



- **Token embeddings:** WordPiece tokenization ($\sim 30K$ vocabulary).
- **Segment embeddings:** E_A for sentence A, E_B for sentence B (for NSP).
- **Position embeddings:** Learned (not sinusoidal). Max sequence length: 512.
- **Special tokens:**
 - [CLS]: classification token (first position); its representation is used for sentence-level tasks.
 - [SEP]: separator between the two sentences.

BERT: Architecture Details

Model sizes

	BASE	LARGE
Layers	12	24
Hidden size	768	1024
Attention heads	12	16
Parameters	110M	340M

Training data

BooksCorpus: 800M words

English Wikipedia: 2.5B words

Total: ~ 3.3 B words

Training setup

WordPiece: ~ 30 K vocabulary

Maximum length: 512; batch size: 256

1M steps (~ 40 epochs)

Adam + L2 weight decay (0.01)

10K-step warmup + linear decay

4 BASE / 16 LARGE Cloud TPUs; 4 days

BERT: Architecture Details

Model sizes

	BASE	LARGE
Layers	12	24
Hidden size	768	1024
Attention heads	12	16
Parameters	110M	340M

Training data

BooksCorpus: 800M words

English Wikipedia: 2.5B words

Total: ~ 3.3 B words

Training setup

WordPiece: ~ 30 K vocabulary

Maximum length: 512; batch size: 256

1M steps (~ 40 epochs)

Adam + L2 weight decay (0.01)

10K-step warmup + linear decay

4 BASE / 16 LARGE Cloud TPUs; 4 days

Versus GPT-1

Same BASE size, but bidirectional; much more training data; exceeds GPT-1 on all benchmarks.

Fine-Tuning BERT for Downstream Tasks

Key idea: Add a small task-specific head to pretrained BERT, then fine-tune **all** parameters.

Sentence-level tasks (e.g., sentiment):

- Use the [CLS] representation.
- Add a linear classifier:
 $\hat{y} = \text{softmax}(\mathbf{W} \cdot \mathbf{h}_{[\text{CLS}]})$.

Sentence pairs (natural language inference, NLI):

- Input: [CLS] A [SEP] B [SEP].
- Classify with the [CLS] representation.

Fine-Tuning BERT for Downstream Tasks

Key idea: Add a small task-specific head to pretrained BERT, then fine-tune **all** parameters.

Sentence-level tasks (e.g., sentiment):

- Use the [CLS] representation.
- Add a linear classifier:
 $\hat{y} = \text{softmax}(\mathbf{W} \cdot \mathbf{h}_{[\text{CLS}]})$.

Sentence pairs (natural language inference, NLI):

- Input: [CLS] A [SEP] B [SEP].
- Classify with the [CLS] representation.

Token-level tasks (named entity recognition, NER):

- Use per-token representations $\mathbf{h}_1, \dots, \mathbf{h}_T$.
- $\hat{y}_t = \text{softmax}(\mathbf{W} \cdot \mathbf{h}_t)$.

Span extraction (extractive question answering):

- Predict start and end positions.
- $P_{\text{start}}(i) = \text{softmax}(\mathbf{w}_s^\top \mathbf{h}_i)$.
- $P_{\text{end}}(i) = \text{softmax}(\mathbf{w}_e^\top \mathbf{h}_i)$.

Fine-Tuning BERT for Downstream Tasks

Key idea: Add a small task-specific head to pretrained BERT, then fine-tune **all** parameters.

Sentence-level tasks (e.g., sentiment):

- Use the [CLS] representation.
- Add a linear classifier:
 $\hat{y} = \text{softmax}(\mathbf{W} \cdot \mathbf{h}_{[\text{CLS}]})$.

Sentence pairs (natural language inference, NLI):

- Input: [CLS] A [SEP] B [SEP].
- Classify with the [CLS] representation.

Fine-tuning is fast: Typically 2–4 epochs, $\text{LR} \sim 2 \times 10^{-5}$, and a few hours on one GPU.

Token-level tasks (named entity recognition, NER):

- Use per-token representations $\mathbf{h}_1, \dots, \mathbf{h}_T$.
- $\hat{y}_t = \text{softmax}(\mathbf{W} \cdot \mathbf{h}_t)$.

Span extraction (extractive question answering):

- Predict start and end positions.
- $P_{\text{start}}(i) = \text{softmax}(\mathbf{w}_s^\top \mathbf{h}_i)$.
- $P_{\text{end}}(i) = \text{softmax}(\mathbf{w}_e^\top \mathbf{h}_i)$.

BERT: Impact and Results

At publication (October 2018), BERT achieved SOTA on **11 NLP tasks**:

Benchmarks: GLUE = General Language Understanding Evaluation; SQuAD = Stanford Question Answering Dataset; SWAG = Situations With Adversarial Generations.

Benchmark	Previous SOTA	BERT _{LARGE}
GLUE (avg.)	72.8	80.5
SQuAD 1.1 (F1)	91.7	93.2
SQuAD 2.0 (F1)	78.0	83.1
SWAG (acc.)	78.0	86.3

Reported SWAG human accuracy was 88.0, measured on 100 examples.

BERT: Impact and Results

At publication (October 2018), BERT achieved SOTA on **11 NLP tasks**:

Benchmarks: GLUE = General Language Understanding Evaluation; SQuAD = Stanford Question Answering Dataset; SWAG = Situations With Adversarial Generations.

Benchmark	Previous SOTA	BERT _{LARGE}
GLUE (avg.)	72.8	80.5
SQuAD 1.1 (F1)	91.7	93.2
SQuAD 2.0 (F1)	78.0	83.1
SWAG (acc.)	78.0	86.3

Reported SWAG human accuracy was 88.0, measured on 100 examples.

Why was BERT so influential?

- Bidirectional pretraining greatly outperformed unidirectional GPT-1.
- It established pretraining followed by fine-tuning as the default approach.
- Simple task heads enabled many variants: RoBERTa, ALBERT, DistilBERT, ...

NSP: Does It Actually Help?

BERT's NSP objective: binary classification. Is sentence B the real next sentence?

NSP: Does It Actually Help?

BERT's NSP objective: binary classification. Is sentence B the real next sentence?

Subsequent work found that NSP is not very useful.

- **RoBERTa** (Liu et al., 2019) removed NSP and obtained **the same or better** results.
- **Why weak?** Random negative sentences are easy to distinguish by topic, so the task confounds topic prediction with coherence.
- **ALBERT** (Lan et al., 2019) used **Sentence Order Prediction (SOP)**: distinguish consecutive sentences in correct order from the same pair in reversed order.

NSP: Does It Actually Help?

BERT's NSP objective: binary classification. Is sentence B the real next sentence?

Subsequent work found that NSP is not very useful.

- **RoBERTa** (Liu et al., 2019) removed NSP and obtained **the same or better** results.
- **Why weak?** Random negative sentences are easy to distinguish by topic, so the task confounds topic prediction with coherence.
- **ALBERT** (Lan et al., 2019) used **Sentence Order Prediction (SOP)**: distinguish consecutive sentences in correct order from the same pair in reversed order.

Lesson: Not all pretraining objectives are equally useful. Details matter.

RoBERTa: A Robustly Optimized BERT

RoBERTa (Liu et al., 2019): Same architecture as BERT, better **training recipe**.

Design choice	BERT	RoBERTa
NSP	Used	Removed
Masking	Static preprocessing (10 masks/example)	Dynamic resampling for each input pass
Training data	16 GB: Books + Wiki	160 GB : adds CC-News, OpenWebText, Stories
Batch / updates	256 / 1M	8K / 500K
Input format	Sentence pairs	Full-sentence sequences

RoBERTa: A Robustly Optimized BERT

RoBERTa (Liu et al., 2019): Same architecture as BERT, better **training recipe**.

Design choice	BERT	RoBERTa
NSP	Used	Removed
Masking	Static preprocessing (10 masks/example)	Dynamic resampling for each input pass
Training data	16 GB: Books + Wiki	160 GB : adds CC-News, OpenWebText, Stories
Batch / updates	256 / 1M	8K / 500K
Input format	Sentence pairs	Full-sentence sequences

Result: With the same architecture, RoBERTa matched or exceeded BERT.

Key insight: BERT was significantly **undertrained**: data scale, training recipe, and removing NSP all matter.

Dynamic vs. Static Masking

BERT: static preprocessing

- Generate masks before training.
- Make 10 masked copies; each copy keeps its mask.
- Over 40 epochs, each pattern is reused four times.

Dynamic vs. Static Masking

BERT: static preprocessing

- Generate masks before training.
- Make 10 masked copies; each copy keeps its mask.
- Over 40 epochs, each pattern is reused four times.

RoBERTa: dynamic masking

- Resample the mask whenever a sequence is fed to the model.
- The same example can have a new corrupted form on every pass.
- More context diversity improves generalization.

Dynamic vs. Static Masking

BERT: static preprocessing

- Generate masks before training.
- Make 10 masked copies; each copy keeps its mask.
- Over 40 epochs, each pattern is reused four times.

RoBERTa: dynamic masking

- Resample the mask whenever a sequence is fed to the model.
- The same example can have a new corrupted form on every pass.
- More context diversity improves generalization.

Key distinction: Static means masks are precomputed. It does not mean one mask is reused for every epoch.

Dynamic example: “The [MASK] sat on the mat” → cat; “The cat [MASK] on the mat” → sat.

Other Notable Pretraining Approaches

- **XLNet** (Yang et al., 2019): Permutation language model.
 - Autoregressive, but over random permutations of the token order.
 - Gets bidirectional context without masking artifacts.
- **ELECTRA** (Clark et al., 2020): Replaced token detection.
 - Small “generator” creates corrupted text; large “discriminator” detects which tokens were replaced.
 - Much more sample-efficient than MLM (trains on **all** tokens, not just 15%).
- **BART** (Lewis et al., 2020): Denoising autoencoder.
 - Encoder-decoder like T5, but with richer corruption: token masking, deletion, infilling, sentence permutation, document rotation.
- **UniLM** (Dong et al., 2019): Unified LM.
 - Mixes bidirectional, unidirectional, and seq2seq attention masks within one model.

The Pretraining Objective Landscape

Unidirectional

Autoregressive

GPT, LLaMA

Bidirectional

Masked LM

BERT, RoBERTa

Enc-Dec

Span Corruption

T5

Permutation

XLNet

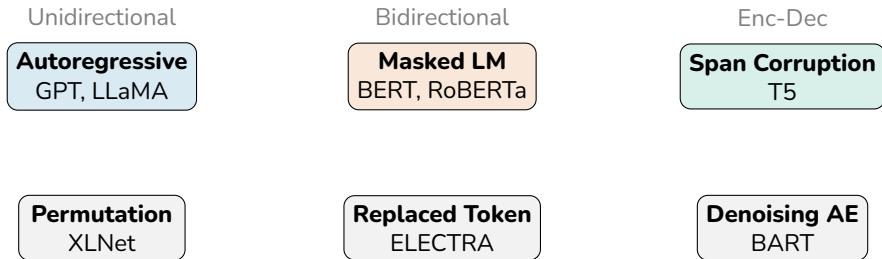
Replaced Token

ELECTRA

Denoising AE

BART

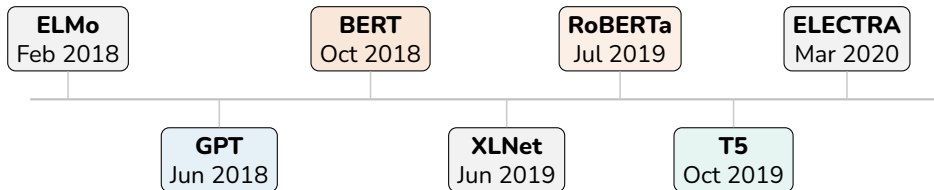
The Pretraining Objective Landscape



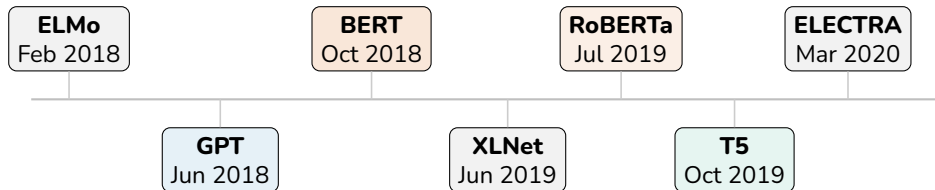
In 2024–2026: Decoder-only autoregressive LMs dominate.

- Scaling favors autoregressive models (simpler, more data-efficient at scale).
- But encoder models (BERT-family) remain dominant for classification, retrieval, and embeddings.
- **Understanding the full landscape** helps you choose the right model for each task.

Historical Context: The Timeline



Historical Context: The Timeline



Key milestones:

- **ELMo**: First pretrained contextual word representations (BiLSTM).
- **GPT**: First Transformer-based pretrained LM (decoder-only).
- **BERT**: Bidirectional pretraining with MLM → dominated NLU benchmarks.
- **RoBERTa**: Better training recipe for BERT → showed BERT was undertrained.
- **T5**: Encoder-decoder pretraining with a unified text-to-text interface.

Summary

- **Autoregressive LMs** (GPT): predict left-to-right. Natural for generation, but unidirectional context limits understanding.
- **Masked LMs** (BERT): mask 15% of tokens, predict from bidirectional context.
 - The 80/10/10 split reduces train-test mismatch.
 - RoBERTa removes NSP and improves the training recipe.
- **Encoder-decoder models**: combine bidirectional input encoding with autoregressive output generation.
- **Three model families**: decoder-only (generation), encoder-only (understanding), encoder-decoder (conditional generation). Choice depends on the task.
- The pre-train → fine-tune paradigm is the foundation of modern NLP.

Readings

Required:

- Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). [BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding](#). *NAACL*.

Optional / further reading:

- Raffel, C., et al. (2020; arXiv: 2019). [Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer](#). *JMLR*.
- Peters, M. et al. (2018). [Deep contextualized word representations \(ELMo\)](#). *NAACL*.
- Liu, Y., et al. (2019). [RoBERTa: A Robustly Optimized BERT Pretraining Approach](#).
- Clark, K., et al. (2020). [ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators](#). *ICLR*.
- Lewis, M., et al. (2020). [BART: Denoising Sequence-to-Sequence Pre-training](#). *ACL*.

Next Class: Midterm Examination

- **Date:** Tuesday, October 20, 2026
- **Time:** 11:30 AM–12:50 PM (80 minutes)
- **Location:** DC 1351

Please go directly to DC 1351 for the midterm exam, not our usual classroom.