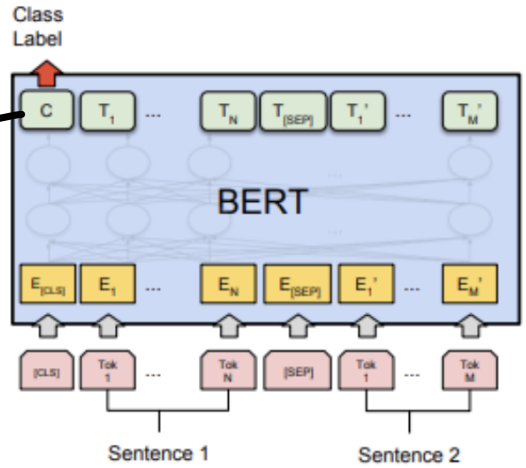
BERT: Bidirectional Encoder Representations from Transformers

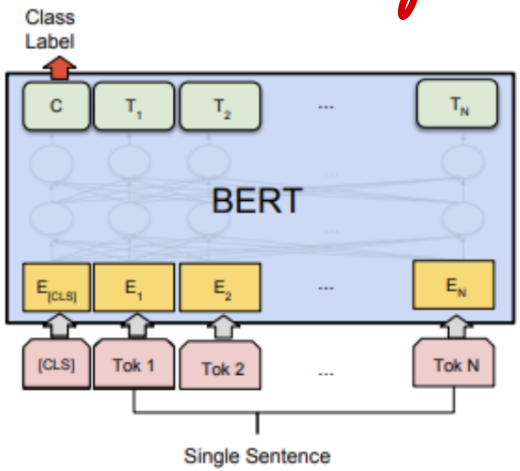
Details about BERT

- Two models were released:
  - BERT-base: 12 layers, 768-dim hidden states, 12 attention heads, 110 million params.
  - BERT-large: 24 layers, 1024-dim hidden states, 16 attention heads, 340 million params.
- Trained on:
  - BooksCorpus (800 million words)
  - English Wikipedia (2,500 million words)
- Pretraining is expensive and impractical on a single GPU.
  - BERT was pretrained with 64 TPU chips for a total of 4 days.
  - (TPUs are special tensor operation acceleration hardware)
- Finetuning is practical and common on a single GPU
  - "Pretrain once, finetune many times."

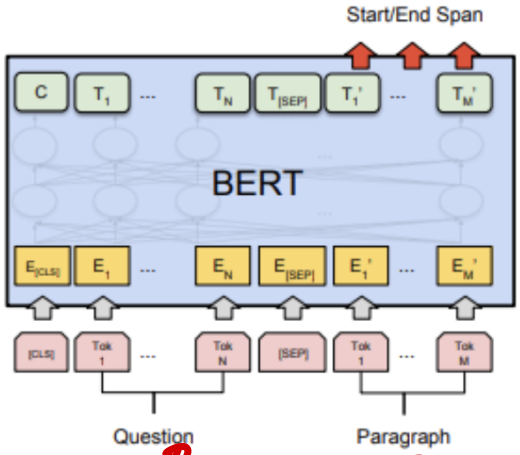
Using BERT for different tasks



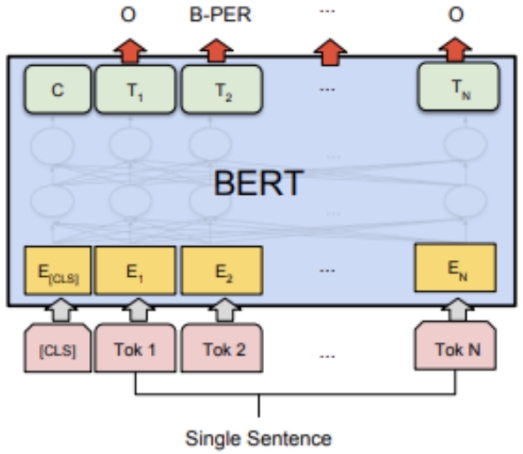
(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG



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



(c) Question Answering Tasks: SQuAD v1.1



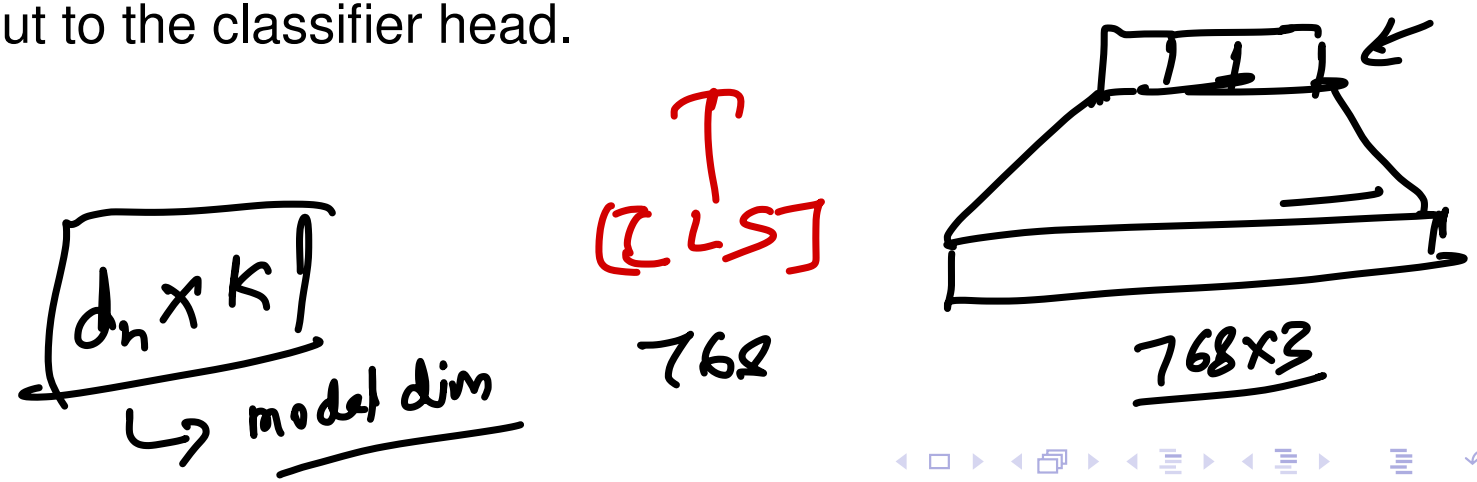
(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

Transfer Learning through Fine-tuning

- The power of pretrained language models lies in their ability to extract generalizations from large amounts of text
- To make practical use of these generalizations, we need to create interfaces from these models to downstream applications through a process called *fine-tuning*.
- Fine-tuning facilitates the creation of applications on top of pretrained models through the (possible) addition of a small set of application-specific parameters.
- The fine-tuning process consists of using labeled data from the application to train these additional application-specific parameters.
- Typically, this training will either freeze or make only minimal adjustments to the pretrained language model parameters.

Fine Tuning for Sequence Classification

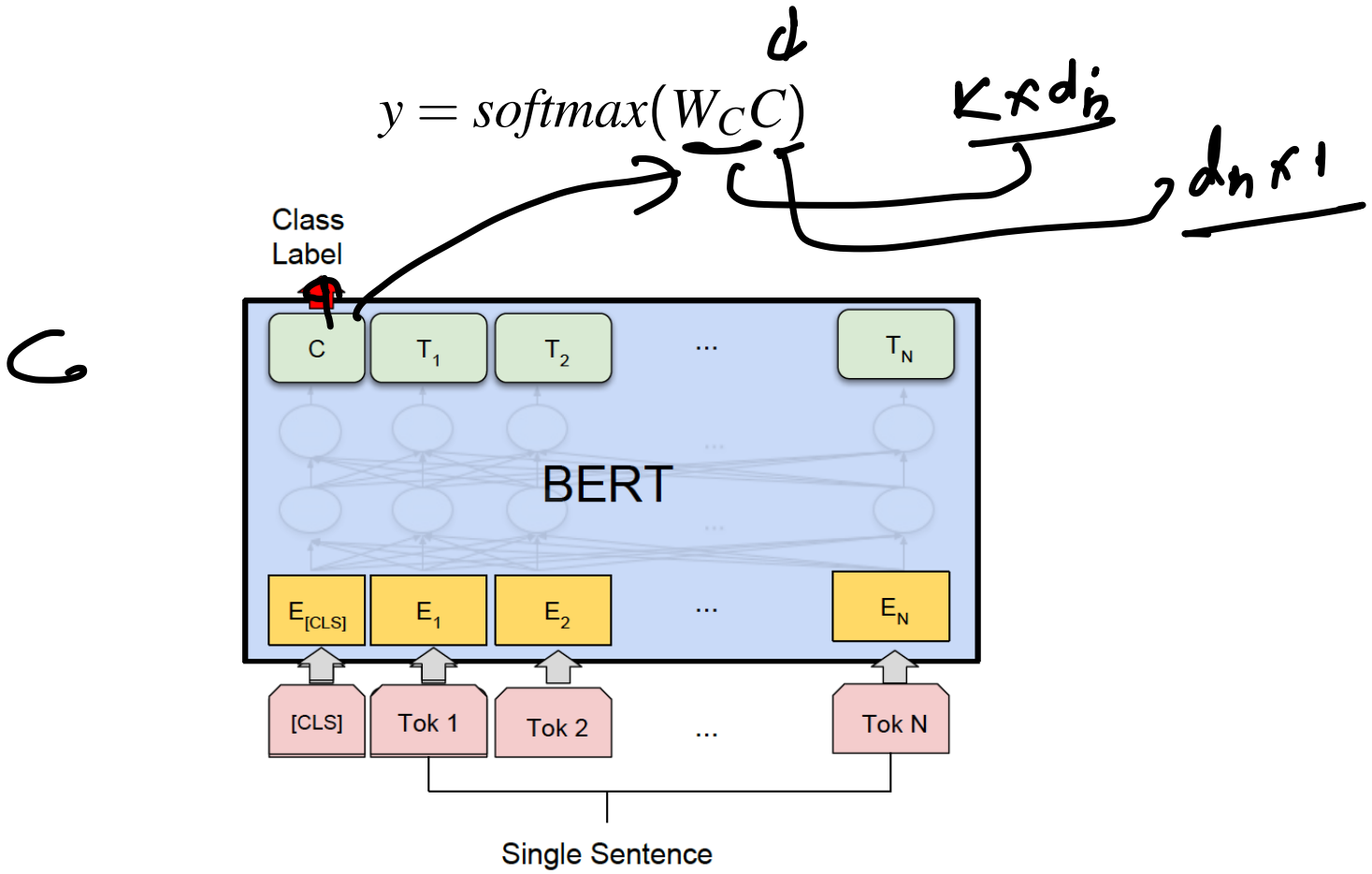
- With RNNs, we used the hidden layer associated with the final input element to stand for the entire sequence. In BERT, the [CLS] token plays the role of sentence embedding.
- This unique token is added to the vocabulary and is prepended to the start of all input sequences, both during pretraining and encoding.
- The output vector in the final layer of the model for the [CLS] input serves as the input to the classifier head.



Fine Tuning for Sequence Classification

- With RNNs, we used the hidden layer associated with the final input element to stand for the entire sequence. In BERT, the [CLS] token plays the role of sentence embedding.
- This unique token is added to the vocabulary and is prepended to the start of all input sequences.
- The output vector $C \in R^{d_h}$ in the final layer of the model for the [CLS] input serves as the input to classifier head ~~a classifier head~~.
- The only new parameters introduced during fine-tuning are classification layer weights $W_C \in R^{K \times d_h}$, where K is the number of labels.

Fine Tuning for Sequence Classification



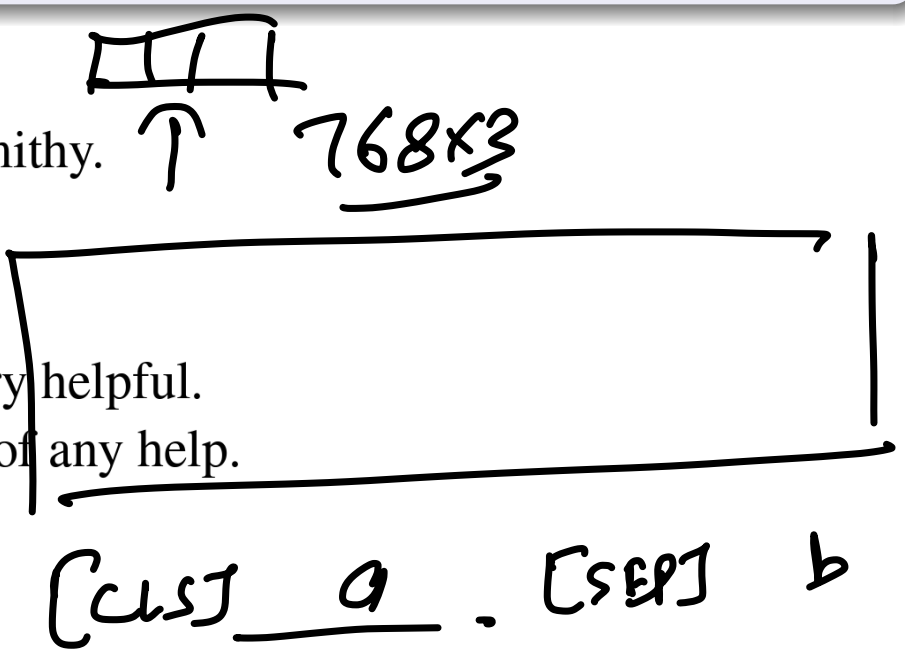
(b) Single Sentence Classification Tasks: SST-2, CoLA

Pair-wise Sequence Classification

Example: multiNLI

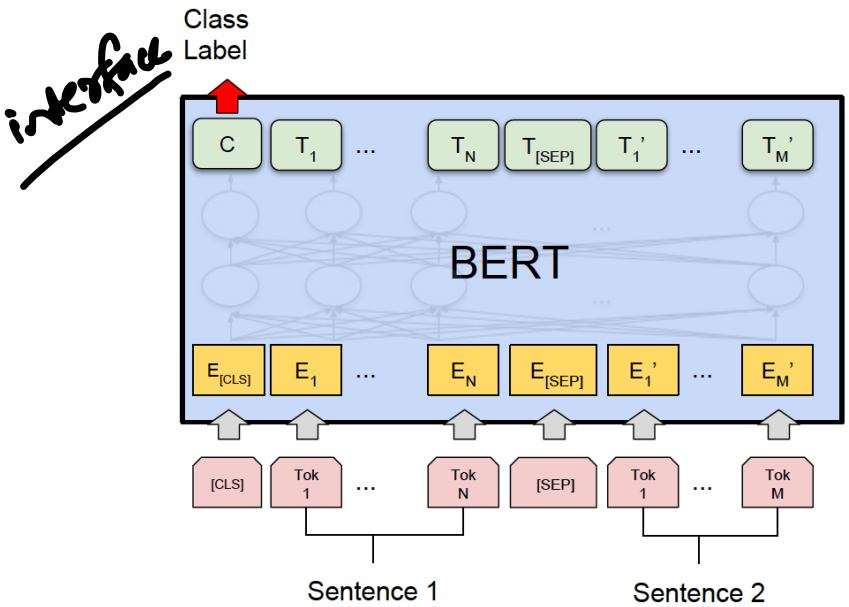
- Pairs of sentences are given one of 3 labels: entails, contradicts and neutral.
- These labels describe a relationship between the meaning of the first sentence (the premise) and the second sentence (the hypothesis).

- `Neutral`
  - a: Jon walked back to the town to the smithy.
  - b: Jon traveled back to his hometown.
- `Contradicts`
  - a: Tourist Information offices can be very helpful.
  - b: Tourist Information offices are never of any help.
- `Entails`
  - a: I'm confused.
  - b: Not all of it is very clear to me.



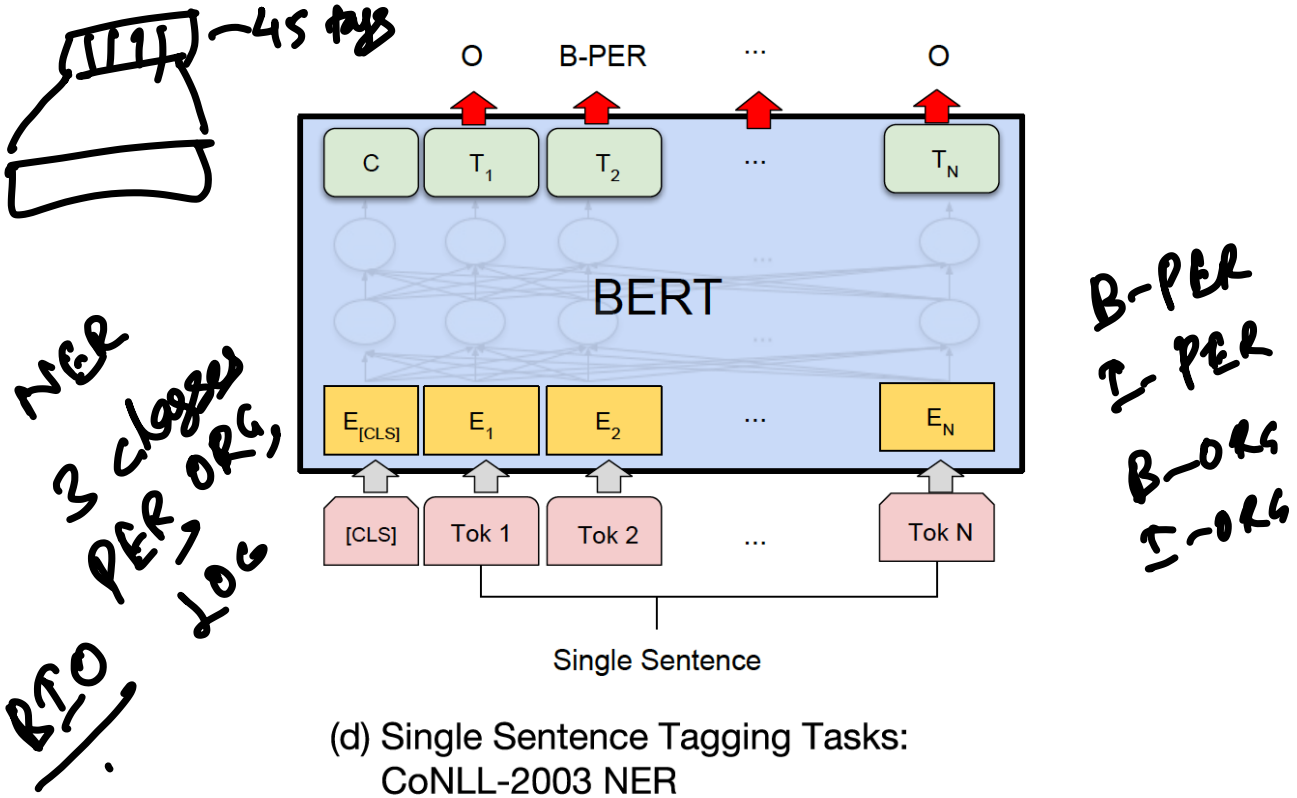
Pair-wise Sequence Classification

- As with NSP training, the two inputs are separated by a [SEP] token.
- As with sequence classification, the output vector associated with the prepended [CLS] token represents the model's view of the input pair.
- This vector C provides the input to a three-way classifier that can be trained on the MultiNLI training corpus.



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

Sequence Labeling



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

- Here, the final output vector corresponding to *each input token* is passed to a classifier that produces a softmax distribution over the possible set of tags.
- The set of weights to be learned for this additional layer is $W_K \in R^{k \times d_h}$, where k is the number of possible tags for the task.

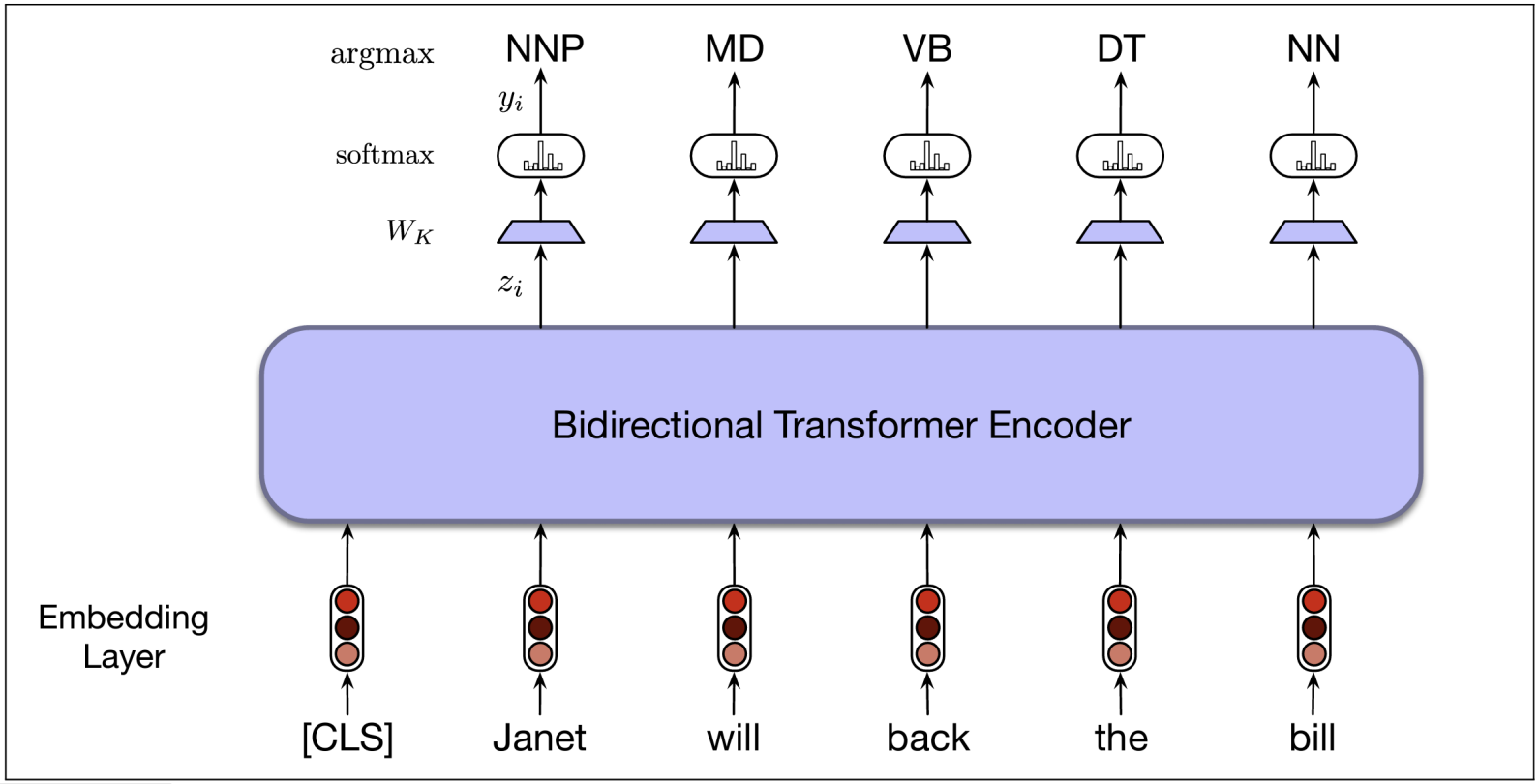


Figure 11.9 Sequence labeling for part-of-speech tagging with a bidirectional transformer encoder. The output vector for each input token is passed to a simple k-way classifier.

Named Entity Recognition and BIO Scheme

Supervised training data for tasks like named entity recognition (NER) is typically in the form of BIO tags associated with text segmented at the word level. For example:

[LOC **Mt. Sanitas**] is in [LOC **Sunshine Canyon**]

would have the following set of per-word BIO tags.

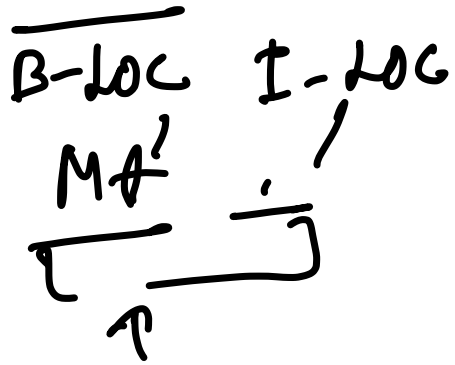
(11.14) *Mt.* *Sanitas* *is* *in* *Sunshine* *Canyon* .
B-LOC I-LOC O O B-LOC I-LOC O

BIO Scheme with subwords

[LOC **Mt. Sanitas**] is in [LOC **Sunshine Canyon**]

would have the following set of per-word BIO tags.

(11.14) *Mt. Sanitas is in Sunshine Canyon .*
B-LOC I-LOC O O B-LOC I-LOC O



After wordPiece Tokenization:

```
'Mt', '.', 'San', '##itas', 'is', 'in', 'Sunshine', 'Canyon' '.
```

The sequence does not align with the original tags.

Solution: Training and Decoding

- **Training:** we can just assign the gold-standard tag associated with each word to all of the subword tokens derived from it.
- **Decoding:** the simplest approach is to use the argmax BIO tag associated with the first subword token of a word.

Fine-tuning for span-based applications

In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under **gravity**. The main forms of precipitation include drizzle, rain, sleet, snow, **graupel** and hail... Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals **within a cloud**. Short, intense periods of rain in scattered locations are called "showers".

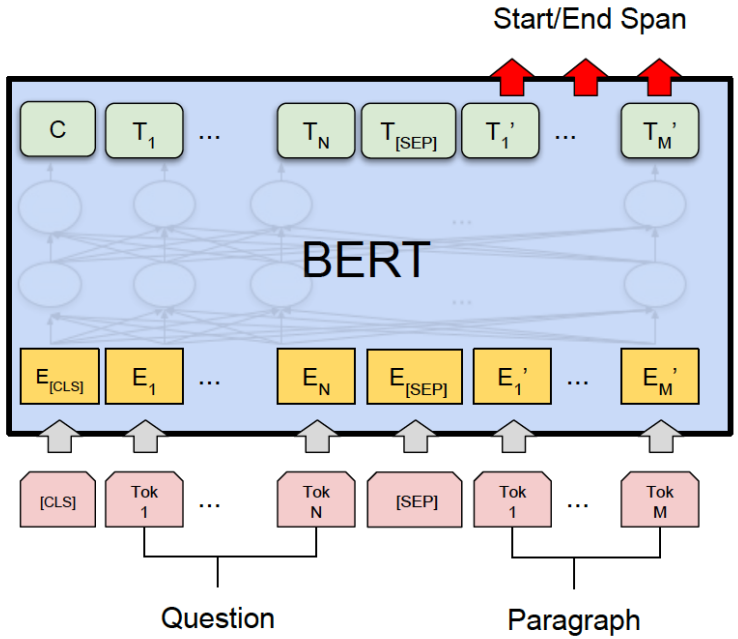
Para

What causes precipitation to fall?
gravity

What is another main form of precipitation besides drizzle, rain, snow, sleet and hail?
graupel

Where do water droplets collide with ice crystals to form precipitation?
within a cloud

Fine-tuning for span-based applications



(c) Question Answering Tasks: SQuAD v1.1

Fine-tuning for SQUAD

- We represent the input question and passage as a single packed sequence (with [SEP])
- We only introduce a start vector $S \in R^{d_h}$ and an end vector $E \in R^{d_h}$ during fine-tuning.
- The probability of word i being the start of the answer span is computed as a dot product between T_i and S followed by a softmax over all of the words in the paragraph.

$$P_i = \frac{e^{S \cdot T_i}}{\sum_j e^{S \cdot T_j}}$$

- The analogous formula is used for the end of the answer span.
- The score of a candidate span from position i to position j is defined as $S \cdot T_i + E \cdot T_j$, and the maximum scoring span where $j \geq i$ is used as a prediction.

Fine-tuning for span-based applications

- Formally, given an input sequence x consisting of T tokens, $(x_1, x_2, \ldots, x_T)$, a span is a contiguous sequence of tokens with start i and end j such that $1 \leq i \leq j \leq T$.
- This formulation results in a total set of spans equal to $\frac{T(T+1)}{2}$.
- For practical purposes, span-based models often impose an application-specific length limit L, so the legal spans are limited to those where $j - i < L$.
- We'll refer to the enumerated set of legal spans in input x as $S(x)$

$\leq d_n$

$\leq d_n$

Evaluation: GLUE Benchmark

GLUE?

General Language Understanding Evaluation

Corpus	|Train|	|Test|	Task	Metrics	Domain
			Single-Sentence Tasks		
CoLA	8.5k	**1k**	acceptability	Matthews corr.	misc.
SST-2	67k	1.8k	sentiment	acc.	movie reviews
			Similarity and Paraphrase Tasks		
MRPC	3.7k	1.7k	paraphrase	acc./F1	news
STS-B	7k	1.4k	sentence similarity	Pearson/Spearman corr.	misc.
QQP	364k	**391k**	paraphrase	acc./F1	social QA questions
			Inference Tasks		
MNLI	393k	**20k**	NLI	matched acc./mismatched acc.	misc.
QNLI	105k	5.4k	QA/NLI	acc.	Wikipedia
RTE	2.5k	3k	NLI	acc.	news, Wikipedia
WNLI	634	**146**	coreference/NLI	acc.	fiction books

Results on GLUE

- **QQP:** Quora Question Pairs (detect paraphrase questions)
- **QNLI**: natural language inference over question answering data
- **SST-2**: sentiment analysis
- **CoLA**: corpus of linguistic acceptability (detect whether sentences are grammatical.)
- **STS-B**: semantic textual similarity
- **MRPC**: microsoft paraphrase corpus
- **RTE**: a small natural language inference corpus

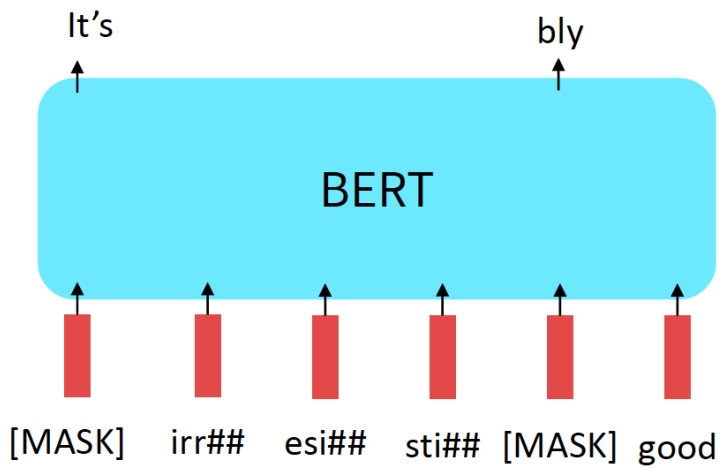
System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	**Average** -
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
$\text{BERT}_{\text{BASE}}$	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**

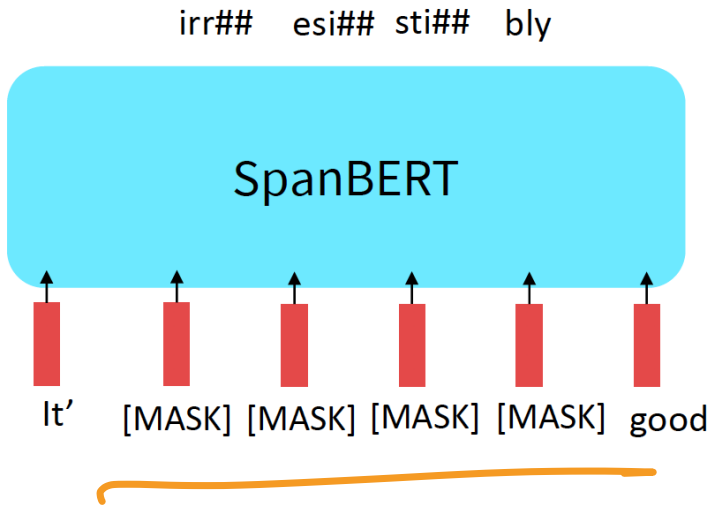
Extensions of BERT

You'll see a lot of BERT variants like RoBERTa, SpanBERT, +++

Some generally accepted improvements to the BERT pretraining formula:

- RoBERTa: mainly just train BERT for longer and remove next sentence prediction!
- SpanBERT: masking contiguous spans of words makes a harder, more useful pretraining task





Extensions of BERT

A takeaway from the RoBERTa paper

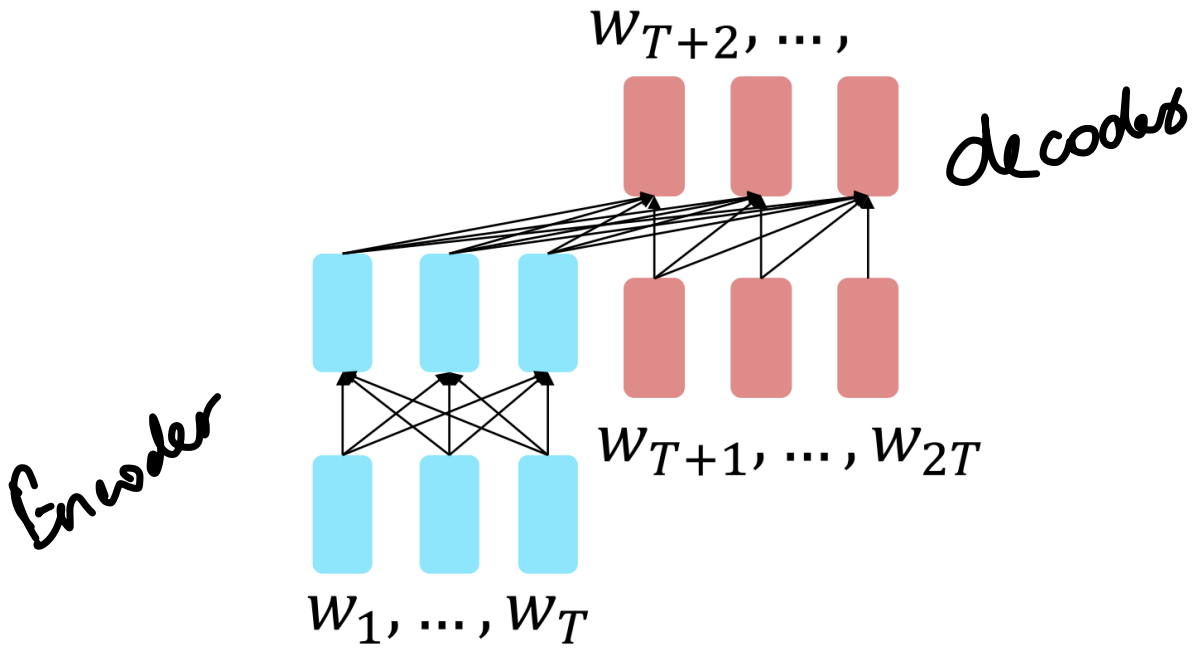
More compute, more data can improve pretraining even when not changing the underlying Transformer encoder.

Model	data	bsz	steps	SQuAD (v1.1/2.0)	MNLI-m	SST-2
RoBERTa						
with BOOKS + WIKI	16GB	8K	100K	93.6/87.3	89.0	95.3
+ additional data (§3.2)	160GB	8K	100K	94.0/87.7	89.3	95.6
+ pretrain longer	160GB	8K	300K	94.4/88.7	90.0	96.1
+ pretrain even longer	160GB	8K	500K	**94.6/89.4**	**90.2**	**96.4**
$\text{BERT}_{\text{LARGE}}$						
with BOOKS + WIKI	13GB	256	1M	90.9/81.8	86.6	93.7

Pretraining encoder-decoders

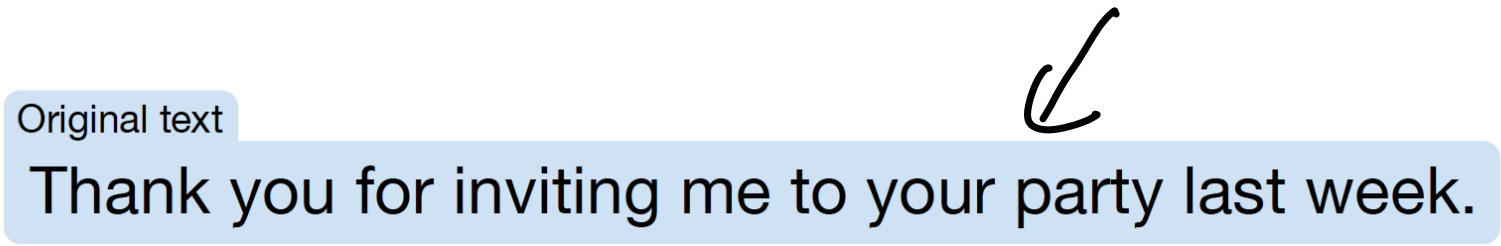
What pretraining objective to use?

For encoder-decoders, we could do something like language modeling, but where a prefix of every input is provided to the encoder and is not predicted.



The encoder portion benefits from bidirectional context; the decoder portion is used to train the whole model through language modeling.

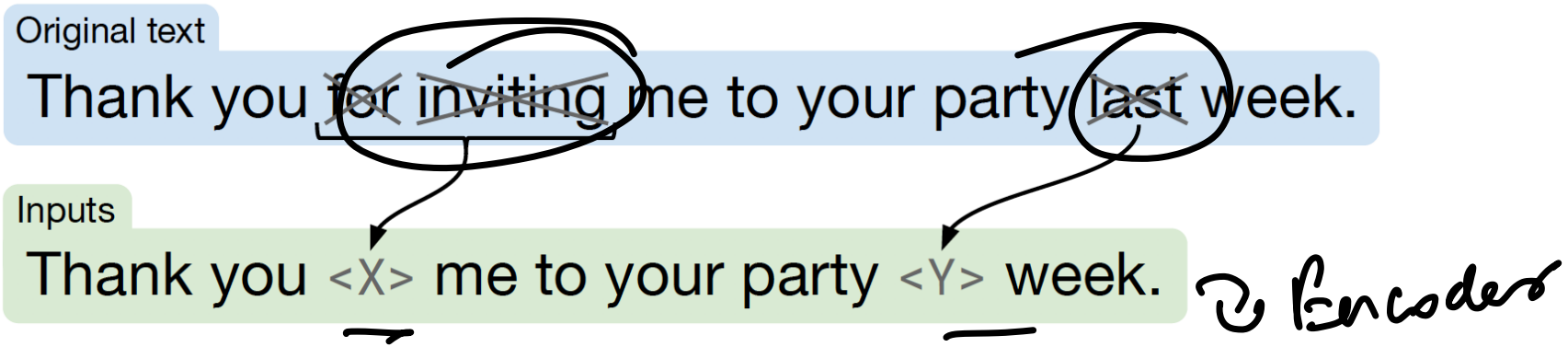
T5: A New Training Objective



T5: A New Training Objective

Span Corruption

Replace different-length spans from the input with unique placeholders;

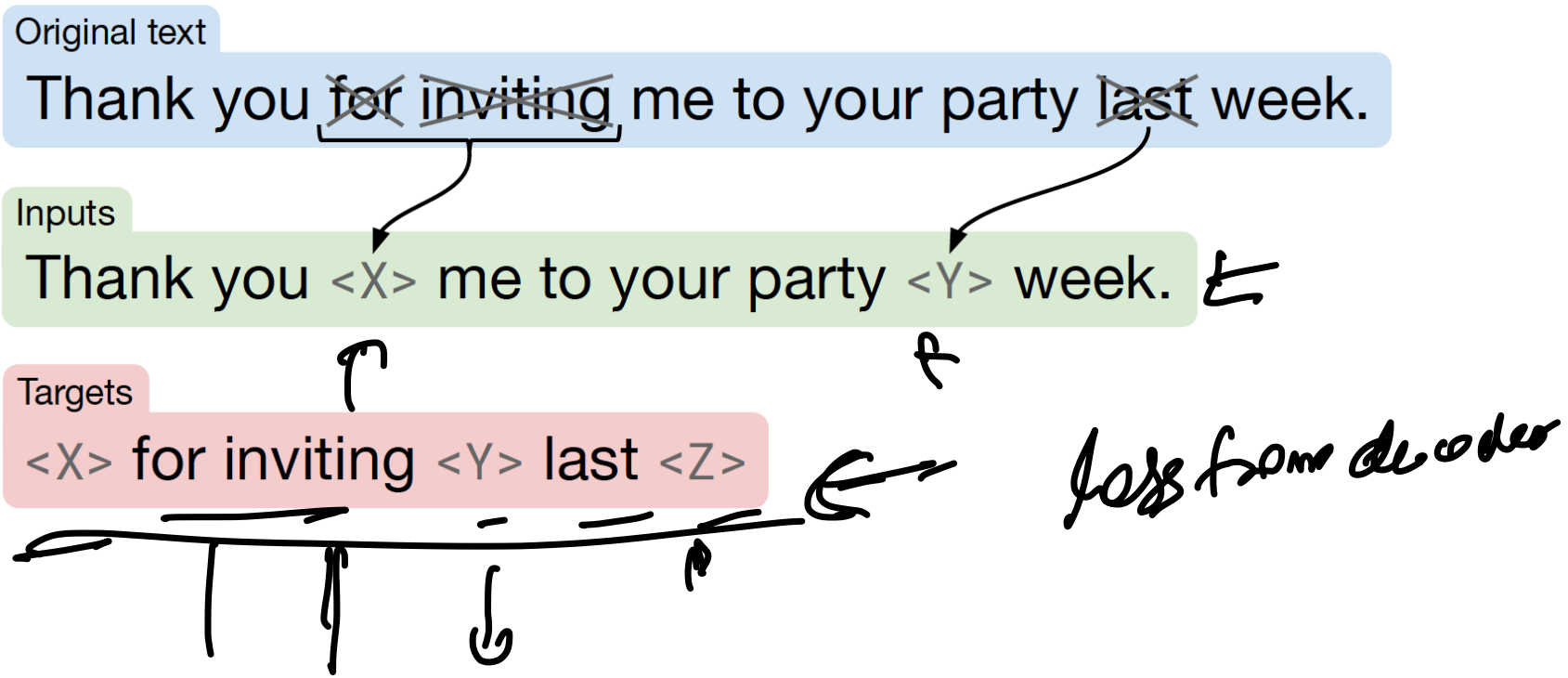


T5: A new Training Objective

mT5

Span Corruption

decode out the spans that were corrupted



T5 can be used for various tasks

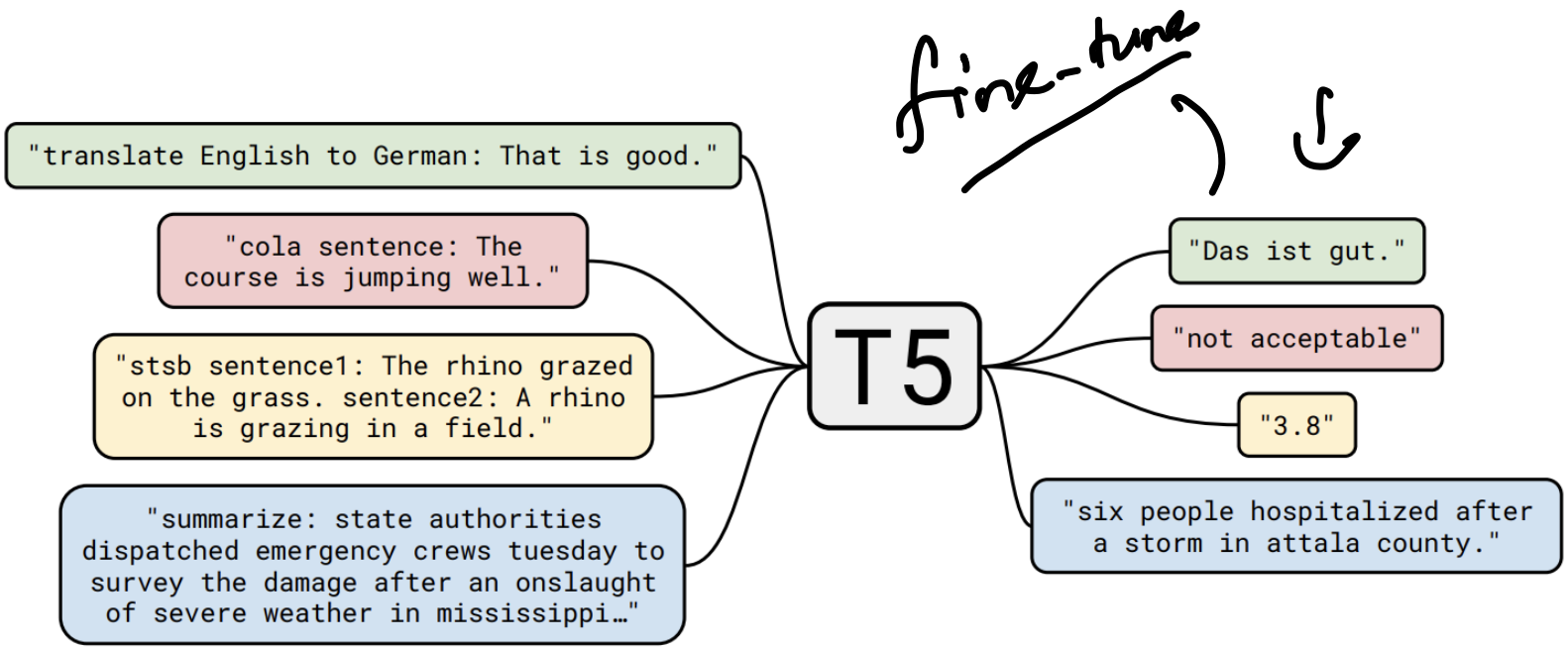
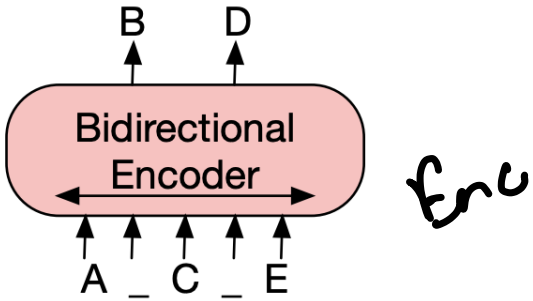
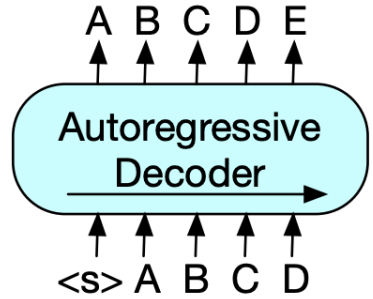


Figure 1: A diagram of our text-to-text framework. Every task we consider—including translation, question answering, and classification—is cast as feeding our model text as input and training it to generate some target text. This allows us to use the same model, loss function, hyperparameters, etc. across our diverse set of tasks. It also provides a standard testbed for the methods included in our empirical survey. "T5" refers to our model, which we dub the "**T**ext-**t**o-**T**ext **T**ransfer **T**ransformer".

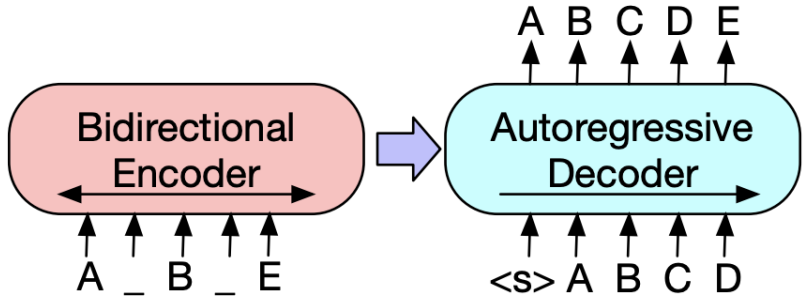
Bi-directional and Auto-Regressive Transformers (BART)



(a) BERT: Random tokens are replaced with masks, and the document is encoded bidirectionally. Missing tokens are predicted independently, so BERT cannot easily be used for generation.

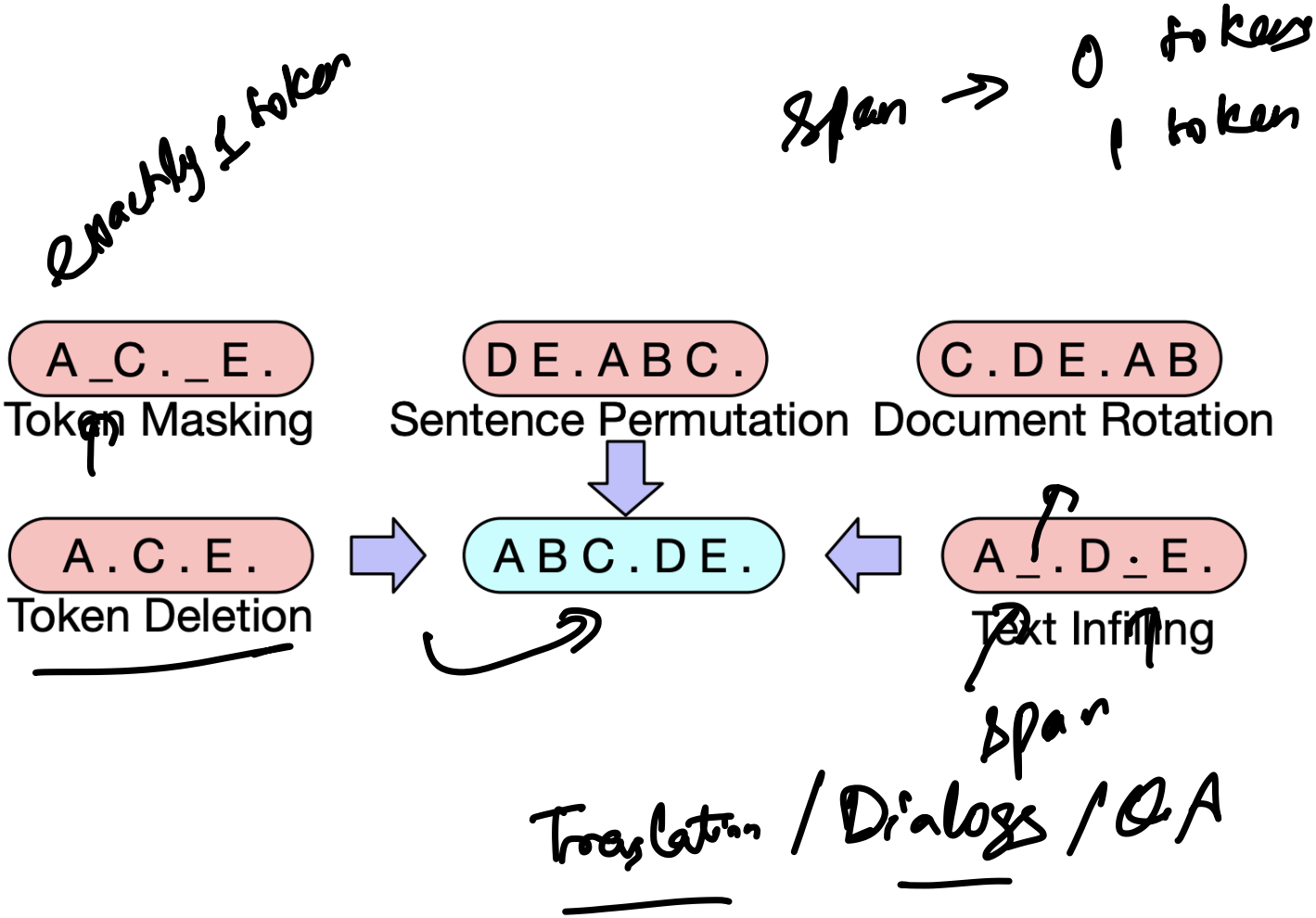


(b) GPT: Tokens are predicted auto-regressively, meaning GPT can be used for generation. However words can only condition on leftward context, so it cannot learn bidirectional interactions.



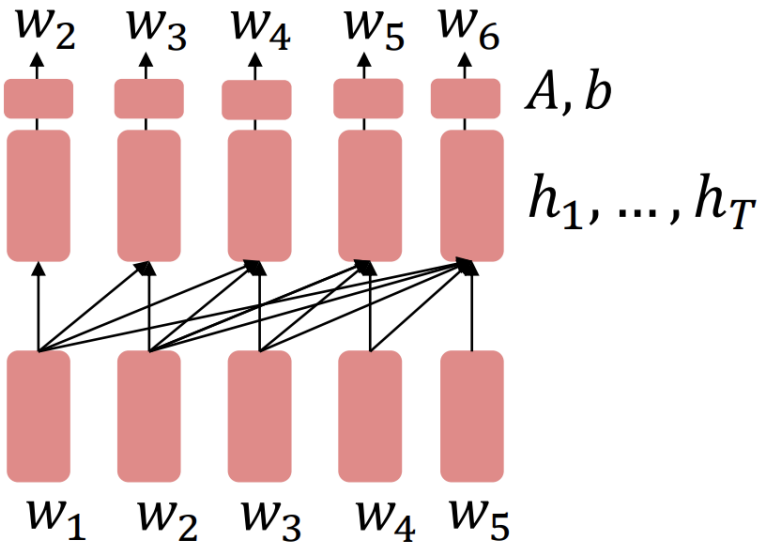
(c) BART: Inputs to the encoder need not be aligned with decoder outputs, allowing arbitary noise transformations. Here, a document has been corrupted by replacing spans of text with mask symbols. The corrupted document (left) is encoded with a bidirectional model, and then the likelihood of the original document (right) is calculated with an autoregressive decoder. For fine-tuning, an uncorrupted document is input to both the encoder and decoder, and we use representations from the final hidden state of the decoder.

BART: Transformation for noising the input



Pretraining Decoders

It's natural to pretrain decoders as language models



Generative Pretrained Transformer (GPT)

- Transformer decoder with 12 layers, 117M parameters.
- 768-dimensional hidden states, 3072-dimensional feed-forward hidden layers.
- Byte-pair encoding with 40,000 merges
- Trained on BooksCorpus: over 7000 unique books.
- Contains long spans of contiguous text, for learning long-distance dependencies.

Pretrained Decoders