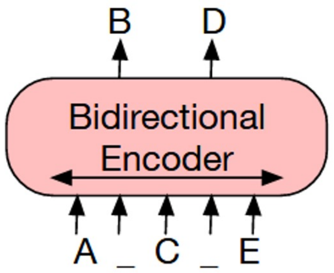
mBART

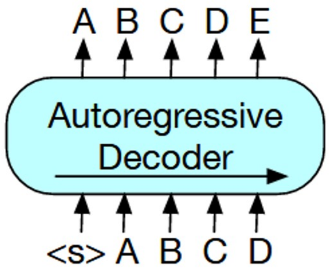
BART

https://arxiv.org/pdf/1910.13461.pdf

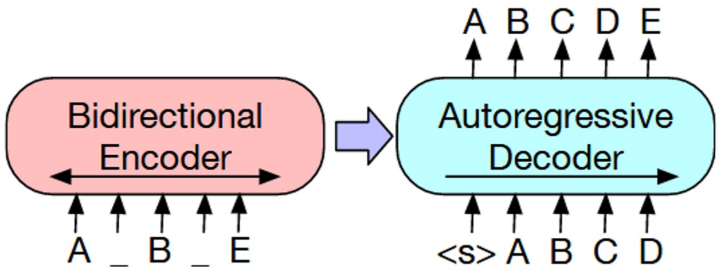
- Combines bidirectional and auto-regressive transformers



(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.

Pre-training

Pre-training has two stages:

1. Text is corrupted with a noising function
    a. Randomly shuffling the order of the original sentences
    b. Arbitrary length (including zero) spans of text are replaced with a single mask token
2. A seq2seq model is learned to reconstruct the original text

Architecture

6 (base) or 12 (large) layers in the encoder and decoder

The architecture is close to BERT’s one, but:

1. Each layer of the decoder additionally performs cross-attention over the final hidden layer of the encoder (as in the seq2seq model)
2. BART does not use an additional feed-forward network before word prediction

Noising

1. Token Masking
2. Token Deletion (In contrast to token masking, the model must decide which positions are missing inputs)
3. Text Infilling (some tokens ---> one [MASK] token)
4. Sentence Permutation (divide into sentences and shuffle)
5. Document Rotation (document is rotated so that is begins with chosen token)

Model	**SQuAD 1.1** F1	**MNLI** Acc	**ELI5** PPL	**XSum** PPL	**ConvAI2** PPL	**CNN/DM** PPL
BERT Base (Devlin et al., 2019)	88.5	**84.3**	-	-	-	-
Masked Language Model	90.0	83.5	24.77	7.87	12.59	7.06
Masked Seq2seq	87.0	82.1	23.40	6.80	11.43	6.19
Language Model	76.7	80.1	**21.40**	7.00	11.51	6.56
Permuted Language Model	89.1	83.7	24.03	7.69	12.23	6.96
Multitask Masked Language Model	89.2	82.4	23.73	7.50	12.39	6.74
BART Base						
w/ Token Masking	90.4	84.1	25.05	7.08	11.73	6.10
w/ Token Deletion	90.4	84.1	24.61	6.90	11.46	5.87
w/ Text Infilling	**90.8**	84.0	24.26	**6.61**	**11.05**	5.83
w/ Document Rotation	77.2	75.3	53.69	17.14	19.87	10.59
w/ Sentence Shuffling	85.4	81.5	41.87	10.93	16.67	7.89
w/ Text Infilling + Sentence Shuffling	**90.8**	83.8	24.17	6.62	11.12	**5.41**

Table 1: Comparison of pre-training objectives. All models are of comparable size and are trained for 1M steps on a combination of books and Wikipedia data. Entries in the bottom two blocks are trained on identical data using the same code-base, and fine-tuned with the same procedures. Entries in the second block are inspired by pre-training objectives proposed in previous work, but have been simplified to focus on evaluation objectives (see §4.1). Performance varies considerably across tasks, but the BART models with text infilling demonstrate the most consistently strong performance.

mBART

CC25 is a large-scale monolingual corpora across 25 languages extracted from the Common Crawl

The BART was applied to CC25. The input texts are noised, and the model is learned to recover the texts ===> mBART (multilingual BART)

Architecture

- 12 layers of the encoder and decoder
- Model dimension is 1024 on 16 heads (∼680M parameters)
- An additional layer-normalization layer was included on top of both the encoder and decoder

Text-to-Text Transfer Transformer (T5)

All tasks are formulated as text-to-text tasks

E.g. classification: predict “True” or “False”, any other string ---> penalty

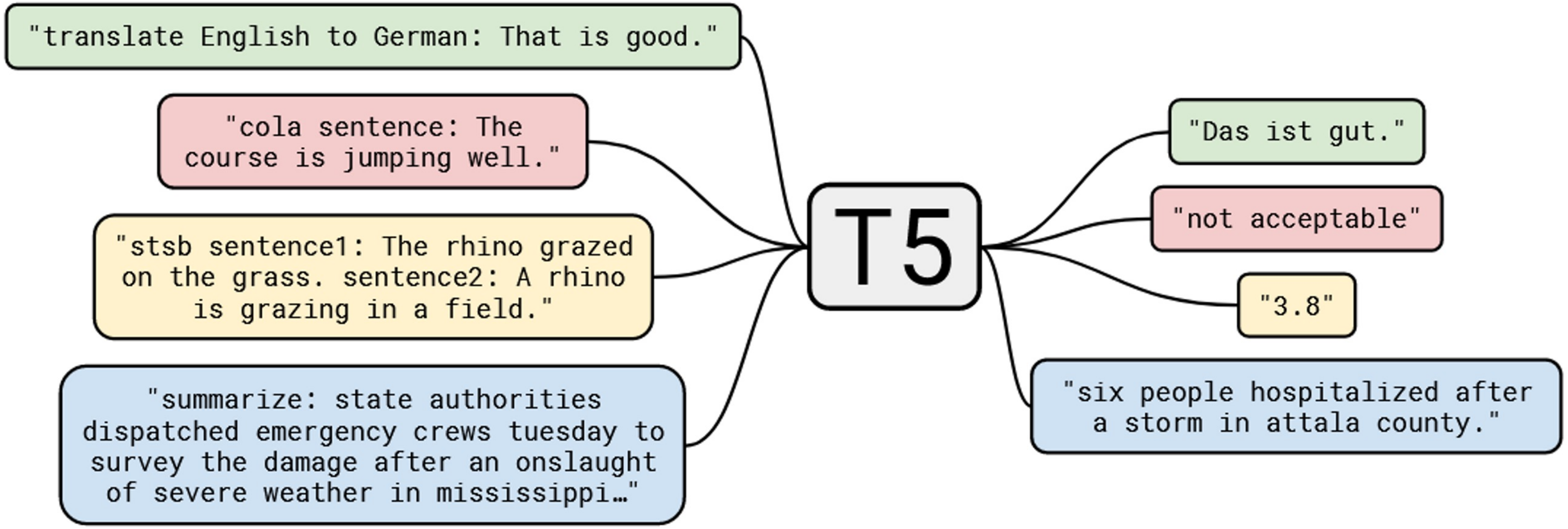

"translate English to German: That is good."
"cola sentence: The course is jumping well."
"stsb sentence1: The rhino grazed on the grass. sentence2: A rhino is grazing in a field."
"summarize: state authorities dispatched emergency crews tuesday to survey the damage after an onslaught of severe weather in mississippi…"
T5
"Das ist gut."
"not acceptable"
"3.8"
"six people hospitalized after a storm in attala county."

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".

Encoder

The encoder consists of a stack of “blocks”, each of which comprises two subcomponents:

1. Self-attention layer
2. Feed-forward network

Layer normalization is applied to the input of each subcomponent. After that, a residual skip connection adds each subcomponent’s input to its output.

Dropout is applied: within the feed-forward network; on the skip connection; on the attention weights; at the input and output of the entire stack.

Decoder

The decoder includes a standard attention mechanism after each self-attention layer that attends to the output of the encoder

The self-attention mechanism in the decoder also uses a form of autoregressive or causal self-attention, which only allows the model to attend to past outputs

The output of the final decoder block is fed into a dense layer with a softmax output, whose weights are shared with the input embedding matrix

Dataset

Colossal Clean Crawled Corpus (C4)

In the Common Crawl, there are a lot of “non-natural language” texts such as placeholders or error messages. For T5 pre-training, CC was cleaned with some heuristics.

“Note that 2^35 tokens only covers a fraction of the entire C4 data set, so we never repeat any data during pre-training”

Training

- AdaFactor for optimization
- 2^19 = 524,288 steps on C4 before fine-tuning
- fine-tuned for 2^18 = 262,144 steps on all tasks
- Maximum sequence length of 512 and a batch size of 128 sequence

“Whenever possible, we “pack” multiple sequences into each entry of the batch so that our batches contain roughly 2^16 = 65,536 tokens.”

In total, this batch size and number of steps corresponds to pre-training on 2^35 ≈ 34B tokens. This is considerably less than BERT (137B tokens), or RoBERTa (2.2T tokens)

- a reasonable computational budget
- still providing a sufficient amount of pre-training for acceptable performance