**BartForConditionalGeneration**

**class transformers.BartForConditionalGeneration**

[<source>](https://github.com/huggingface/transformers/blob/main/src/transformers/models/bart/modeling_bart.py#L1322)

( config: BartConfig )

Parameters

* **config** ([BartConfig](https://huggingface.co/docs/transformers/main/en/model_doc/bart#transformers.BartConfig)) — Model configuration class with all the parameters of the model. Initializing with a config file does not load the weights associated with the model, only the configuration. Check out the [from\_pretrained()](https://huggingface.co/docs/transformers/main/en/main_classes/model#transformers.PreTrainedModel.from_pretrained) method to load the model weights.

The BART Model with a language modeling head. Can be used for summarization.

This model inherits from [PreTrainedModel](https://huggingface.co/docs/transformers/main/en/main_classes/model#transformers.PreTrainedModel). Check the superclass documentation for the generic methods the library implements for all its model (such as downloading or saving, resizing the input embeddings, pruning heads etc.)

This model is also a PyTorch [torch.nn.Module](https://pytorch.org/docs/stable/nn.html#torch.nn.Module) subclass. Use it as a regular PyTorch Module and refer to the PyTorch documentation for all matter related to general usage and behavior.

**forward**

[<source>](https://github.com/huggingface/transformers/blob/main/src/transformers/models/bart/modeling_bart.py#L1368)

( input\_ids: typing.Optional[torch.LongTensor] = Noneattention\_mask: typing.Optional[torch.Tensor] = Nonedecoder\_input\_ids: typing.Optional[torch.LongTensor] = Nonedecoder\_attention\_mask: typing.Optional[torch.LongTensor] = Nonehead\_mask: typing.Optional[torch.Tensor] = Nonedecoder\_head\_mask: typing.Optional[torch.Tensor] = Nonecross\_attn\_head\_mask: typing.Optional[torch.Tensor] = Noneencoder\_outputs: typing.Optional[list[torch.FloatTensor]] = Nonepast\_key\_values: typing.Optional[transformers.cache\_utils.Cache] = Noneinputs\_embeds: typing.Optional[torch.FloatTensor] = Nonedecoder\_inputs\_embeds: typing.Optional[torch.FloatTensor] = Nonelabels: typing.Optional[torch.LongTensor] = Noneuse\_cache: typing.Optional[bool] = Noneoutput\_attentions: typing.Optional[bool] = Noneoutput\_hidden\_states: typing.Optional[bool] = Nonereturn\_dict: typing.Optional[bool] = Nonecache\_position: typing.Optional[torch.LongTensor] = None )→ [transformers.modeling\_outputs.Seq2SeqLMOutput](https://huggingface.co/docs/transformers/main/en/main_classes/output#transformers.modeling_outputs.Seq2SeqLMOutput) or tuple(torch.FloatTensor)

Parameters

* **input\_ids** (torch.LongTensor of shape (batch\_size, sequence\_length), *optional*) — Indices of input sequence tokens in the vocabulary. Padding will be ignored by default.

Indices can be obtained using [AutoTokenizer](https://huggingface.co/docs/transformers/main/en/model_doc/auto#transformers.AutoTokenizer). See [PreTrainedTokenizer.encode()](https://huggingface.co/docs/transformers/main/en/internal/tokenization_utils#transformers.PreTrainedTokenizerBase.encode) and [PreTrainedTokenizer.**call**()](https://huggingface.co/docs/transformers/main/en/internal/tokenization_utils#transformers.PreTrainedTokenizerBase.__call__) for details.

[What are input IDs?](https://huggingface.co/docs/transformers/main/glossary#input-ids)

* **attention\_mask** (torch.Tensor of shape (batch\_size, sequence\_length), *optional*) — Mask to avoid performing attention on padding token indices. Mask values selected in [0, 1]:
  + 1 for tokens that are **not masked**,
  + 0 for tokens that are **masked**.

[What are attention masks?](https://huggingface.co/docs/transformers/main/glossary#attention-mask)

* **decoder\_input\_ids** (torch.LongTensor of shape (batch\_size, target\_sequence\_length), *optional*) — Indices of decoder input sequence tokens in the vocabulary.

Indices can be obtained using [AutoTokenizer](https://huggingface.co/docs/transformers/main/en/model_doc/auto#transformers.AutoTokenizer). See [PreTrainedTokenizer.encode()](https://huggingface.co/docs/transformers/main/en/internal/tokenization_utils#transformers.PreTrainedTokenizerBase.encode) and [PreTrainedTokenizer.**call**()](https://huggingface.co/docs/transformers/main/en/internal/tokenization_utils#transformers.PreTrainedTokenizerBase.__call__) for details.

[What are decoder input IDs?](https://huggingface.co/docs/transformers/main/glossary#decoder-input-ids)

Bart uses the eos\_token\_id as the starting token for decoder\_input\_ids generation. If past\_key\_values is used, optionally only the last decoder\_input\_ids have to be input (see past\_key\_values).

For translation and summarization training, decoder\_input\_ids should be provided. If no decoder\_input\_ids is provided, the model will create this tensor by shifting the input\_ids to the right for denoising pre-training following the paper.

* **decoder\_attention\_mask** (torch.LongTensor of shape (batch\_size, target\_sequence\_length), *optional*) — Default behavior: generate a tensor that ignores pad tokens in decoder\_input\_ids. Causal mask will also be used by default.

If you want to change padding behavior, you should read modeling\_bart.\_prepare\_decoder\_attention\_mask and modify to your needs. See diagram 1 in [the paper](https://huggingface.co/papers/1910.13461) for more information on the default strategy.

* **head\_mask** (torch.Tensor of shape (num\_heads,) or (num\_layers, num\_heads), *optional*) — Mask to nullify selected heads of the self-attention modules. Mask values selected in [0, 1]:
  + 1 indicates the head is **not masked**,
  + 0 indicates the head is **masked**.
* **decoder\_head\_mask** (torch.Tensor of shape (decoder\_layers, decoder\_attention\_heads), *optional*) — Mask to nullify selected heads of the attention modules in the decoder. Mask values selected in [0, 1]:
  + 1 indicates the head is **not masked**,
  + 0 indicates the head is **masked**.
* **cross\_attn\_head\_mask** (torch.Tensor of shape (decoder\_layers, decoder\_attention\_heads), *optional*) — Mask to nullify selected heads of the cross-attention modules in the decoder. Mask values selected in [0, 1]:
  + 1 indicates the head is **not masked**,
  + 0 indicates the head is **masked**.
* **encoder\_outputs** (list[torch.FloatTensor], *optional*) — Tuple consists of (last\_hidden\_state, *optional*: hidden\_states, *optional*: attentions) last\_hidden\_state of shape (batch\_size, sequence\_length, hidden\_size), *optional*) is a sequence of hidden-states at the output of the last layer of the encoder. Used in the cross-attention of the decoder.
* **past\_key\_values** (~cache\_utils.Cache, *optional*) — Pre-computed hidden-states (key and values in the self-attention blocks and in the cross-attention blocks) that can be used to speed up sequential decoding. This typically consists in the past\_key\_values returned by the model at a previous stage of decoding, when use\_cache=True or config.use\_cache=True.

Only [Cache](https://huggingface.co/docs/transformers/main/en/internal/generation_utils#transformers.Cache) instance is allowed as input, see our [kv cache guide](https://huggingface.co/docs/transformers/en/kv_cache). If no past\_key\_values are passed, [DynamicCache](https://huggingface.co/docs/transformers/main/en/internal/generation_utils#transformers.DynamicCache) will be initialized by default.

The model will output the same cache format that is fed as input.

If past\_key\_values are used, the user is expected to input only unprocessed input\_ids (those that don’t have their past key value states given to this model) of shape (batch\_size, unprocessed\_length) instead of all input\_ids of shape (batch\_size, sequence\_length).

* **inputs\_embeds** (torch.FloatTensor of shape (batch\_size, sequence\_length, hidden\_size), *optional*) — Optionally, instead of passing input\_ids you can choose to directly pass an embedded representation. This is useful if you want more control over how to convert input\_ids indices into associated vectors than the model’s internal embedding lookup matrix.
* **decoder\_inputs\_embeds** (torch.FloatTensor of shape (batch\_size, target\_sequence\_length, hidden\_size), *optional*) — Optionally, instead of passing decoder\_input\_ids you can choose to directly pass an embedded representation. If past\_key\_values is used, optionally only the last decoder\_inputs\_embeds have to be input (see past\_key\_values). This is useful if you want more control over how to convert decoder\_input\_ids indices into associated vectors than the model’s internal embedding lookup matrix.

If decoder\_input\_ids and decoder\_inputs\_embeds are both unset, decoder\_inputs\_embeds takes the value of inputs\_embeds.

* **labels** (torch.LongTensor of shape (batch\_size, sequence\_length), *optional*) — Labels for computing the masked language modeling loss. Indices should either be in [0, ..., config.vocab\_size] or -100 (see input\_ids docstring). Tokens with indices set to -100 are ignored (masked), the loss is only computed for the tokens with labels in [0, ..., config.vocab\_size].
* **use\_cache** (bool, *optional*) — If set to True, past\_key\_values key value states are returned and can be used to speed up decoding (see past\_key\_values).
* **output\_attentions** (bool, *optional*) — Whether or not to return the attentions tensors of all attention layers. See attentions under returned tensors for more detail.
* **output\_hidden\_states** (bool, *optional*) — Whether or not to return the hidden states of all layers. See hidden\_statesunder returned tensors for more detail.
* **return\_dict** (bool, *optional*) — Whether or not to return a [ModelOutput](https://huggingface.co/docs/transformers/main/en/main_classes/output#transformers.utils.ModelOutput) instead of a plain tuple.
* **cache\_position** (torch.LongTensor of shape (sequence\_length), *optional*) — Indices depicting the position of the input sequence tokens in the sequence. Contrarily to position\_ids, this tensor is not affected by padding. It is used to update the cache in the correct position and to infer the complete sequence length.

Returns

[transformers.modeling\_outputs.Seq2SeqLMOutput](https://huggingface.co/docs/transformers/main/en/main_classes/output#transformers.modeling_outputs.Seq2SeqLMOutput) or tuple(torch.FloatTensor)

A [transformers.modeling\_outputs.Seq2SeqLMOutput](https://huggingface.co/docs/transformers/main/en/main_classes/output#transformers.modeling_outputs.Seq2SeqLMOutput) or a tuple of torch.FloatTensor (if return\_dict=False is passed or when config.return\_dict=False) comprising various elements depending on the configuration ([BartConfig](https://huggingface.co/docs/transformers/main/en/model_doc/bart#transformers.BartConfig)) and inputs.

* **loss** (torch.FloatTensor of shape (1,), *optional*, returned when labels is provided) — Language modeling loss.
* **logits** (torch.FloatTensor of shape (batch\_size, sequence\_length, config.vocab\_size)) — Prediction scores of the language modeling head (scores for each vocabulary token before SoftMax).
* **past\_key\_values** (EncoderDecoderCache, *optional*, returned when use\_cache=True is passed or when config.use\_cache=True) — It is a [EncoderDecoderCache](https://huggingface.co/docs/transformers/main/en/internal/generation_utils#transformers.EncoderDecoderCache) instance. For more details, see our [kv cache guide](https://huggingface.co/docs/transformers/en/kv_cache).

Contains pre-computed hidden-states (key and values in the self-attention blocks and in the cross-attention blocks) that can be used (see past\_key\_values input) to speed up sequential decoding.

* **decoder\_hidden\_states** (tuple(torch.FloatTensor), *optional*, returned when output\_hidden\_states=True is passed or when config.output\_hidden\_states=True) — Tuple of torch.FloatTensor (one for the output of the embeddings, if the model has an embedding layer, + one for the output of each layer) of shape (batch\_size, sequence\_length, hidden\_size).

Hidden-states of the decoder at the output of each layer plus the initial embedding outputs.

* **decoder\_attentions** (tuple(torch.FloatTensor), *optional*, returned when output\_attentions=True is passed or when config.output\_attentions=True) — Tuple of torch.FloatTensor (one for each layer) of shape (batch\_size, num\_heads, sequence\_length, sequence\_length).

Attentions weights of the decoder, after the attention softmax, used to compute the weighted average in the self-attention heads.

* **cross\_attentions** (tuple(torch.FloatTensor), *optional*, returned when output\_attentions=True is passed or when config.output\_attentions=True) — Tuple of torch.FloatTensor (one for each layer) of shape (batch\_size, num\_heads, sequence\_length, sequence\_length).

Attentions weights of the decoder’s cross-attention layer, after the attention softmax, used to compute the weighted average in the cross-attention heads.

* **encoder\_last\_hidden\_state** (torch.FloatTensor of shape (batch\_size, sequence\_length, hidden\_size), *optional*) — Sequence of hidden-states at the output of the last layer of the encoder of the model.
* **encoder\_hidden\_states** (tuple(torch.FloatTensor), *optional*, returned when output\_hidden\_states=True is passed or when config.output\_hidden\_states=True) — Tuple of torch.FloatTensor (one for the output of the embeddings, if the model has an embedding layer, + one for the output of each layer) of shape (batch\_size, sequence\_length, hidden\_size).

Hidden-states of the encoder at the output of each layer plus the initial embedding outputs.

* **encoder\_attentions** (tuple(torch.FloatTensor), *optional*, returned when output\_attentions=True is passed or when config.output\_attentions=True) — Tuple of torch.FloatTensor (one for each layer) of shape (batch\_size, num\_heads, sequence\_length, sequence\_length).

Attentions weights of the encoder, after the attention softmax, used to compute the weighted average in the self-attention heads.

The [BartForConditionalGeneration](https://huggingface.co/docs/transformers/main/en/model_doc/bart#transformers.BartForConditionalGeneration) forward method, overrides the \_\_call\_\_ special method.

Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the pre and post processing steps while the latter silently ignores them.

Example summarization:

Copied

from transformers import AutoTokenizer, BartForConditionalGeneration

model = BartForConditionalGeneration.from\_pretrained("facebook/bart-large-cnn")

tokenizer = AutoTokenizer.from\_pretrained("facebook/bart-large-cnn")

ARTICLE\_TO\_SUMMARIZE = (

"PG&E stated it scheduled the blackouts in response to forecasts for high winds "

"amid dry conditions. The aim is to reduce the risk of wildfires. Nearly 800 thousand customers were "

"scheduled to be affected by the shutoffs which were expected to last through at least midday tomorrow."

)

inputs = tokenizer([ARTICLE\_TO\_SUMMARIZE], max\_length=1024, return\_tensors="pt")

*# Generate Summary*

summary\_ids = model.generate(inputs["input\_ids"], num\_beams=2, min\_length=0, max\_length=20)

tokenizer.batch\_decode(summary\_ids, skip\_special\_tokens=True, clean\_up\_tokenization\_spaces=False)[0]

'PG&E scheduled the blackouts in response to forecasts for high winds amid dry conditions'

Mask filling example:

Copied

from transformers import AutoTokenizer, BartForConditionalGeneration

tokenizer = AutoTokenizer.from\_pretrained("facebook/bart-base")

model = BartForConditionalGeneration.from\_pretrained("facebook/bart-base")

TXT = "My friends are <mask> but they eat too many carbs."

input\_ids = tokenizer([TXT], return\_tensors="pt")["input\_ids"]

logits = model(input\_ids).logits

masked\_index = (input\_ids[0] == tokenizer.mask\_token\_id).nonzero().item()

probs = logits[0, masked\_index].softmax(dim=0)

values, predictions = probs.topk(5)

tokenizer.decode(predictions).split()

['not', 'good', 'healthy', 'great', 'very']