Function List and Descriptions

prepare_and_split_corpus(ds)
  - Concatenates 'train', 'validation', and 'test' dataset text into a single corpus, then splits it into training, validation, and test sets by 80%, 10% and 10%.

tokenize(text)
  - Performs a basic whitespace tokenization on the input text. Used to prepare text for custom BPE tokenizer training.

compute_coverage(tokenizer, dataset, sample_size=1000)
  - Calculates the percentage of tokens in the dataset that are covered by the tokenizer (i.e., not mapped to <UNK>). Useful for evaluating tokenizer quality.

encode_text_str(tokenizer, text)
  - Encodes raw text into a list of token IDs using the trained BPE tokenizer.

build_input_target_pairs(token_ids, block_size=32)
  - Generates (input, target) pairs using a sliding window approach for training next-token prediction models.

LanguageModelingDataset
  - A custom PyTorch Dataset class that wraps tokenized input-target sequences for use with DataLoader(shuffle the data in each epoch).

EmbeddingLayer
  - Combines token and positional embeddings into dense vectors with dropout. Forms the input layer of the GPT model.

SelfAttention
  - Implements multi-head self-attention using dot-product attention mechanism to capture inter-token dependencies.

FeedForward
  - A two-layer MLP with ReLU activation and dropout, applied independently to each token position.

TransformerBlock
  - A single transformer block composed of LayerNorm, multi-head attention, and feed-forward layers with residual connections.

GPT
  - The main transformer-based language model class. Stacks multiple TransformerBlocks with final LayerNorm and linear output head.

compute_loss(model, batch)
  - Computes cross-entropy loss between model outputs and target token

```
sequences.

train_one_epoch(model, dataloader, optimizer, loss_fn, device)
  - Trains the model for one epoch by running forward and backward
passes, and updating weights.

evaluate_loss(model, val_loader, loss_fn, device)
  - Evaluates average loss on the validation set.

evaluate_perplexity(model, dataloader)
  - Calculates perplexity from average validation loss. Perplexity is
an evaluation metric for language models.

compute_grad_norm(model)
  - Computes the total gradient norm across model parameters to
monitor training stability.

hyperparameter_search(...)
  - Performs a grid search over model hyperparameters such as learning
rate, number of layers/heads, embedding dimension and returns the best
configuration.
  - Use validation data to fine tune
  - Each config runs 10 epochs

retrain_with_best_parameters(...)
  - Perform a retrain on the tuned best parameters
  - Use validation data to fine tune
  - Use on test data to compute perplexity
  - Each config runs 20 epochs
```