Defines the special tokens for the tokenizer:

* [CLS]: Classification token
* [MASK]: Mask token
* [PAD]: Padding token
* [SEP]: Separator token
* [UNK]: Unknown token[[4]](#fn4)

**tokenizer.json**

Describes the tokenizer configuration and vocabulary:

* **Truncation**: Right, max length 128, "LongestFirst" strategy
* **Padding**: Fixed to length 128, pad token [PAD] (ID 0)
* **Special Tokens**: [PAD] (0), [UNK] (100), [CLS] (101), [SEP] (102), [MASK] (103)
* **Normalizer**: BertNormalizer (lowercase, clean text, handle Chinese chars)
* **Pre-tokenizer**: BertPreTokenizer
* **Post-processor**: Adds [CLS] at start and [SEP] at end (single and pair sequences)
* **Decoder**: WordPiece with prefix ##
* **Vocabulary**: Large, includes [unused0] to [unused992], standard punctuation, English words/subwords, and multilingual tokens[[5]](#fn5)

**tokenizer\_config.json**

Provides further tokenizer configuration:

* **Special tokens** and their properties (IDs 0, 100, 101, 102, 103)
* **Lowercasing**: true
* **Model max length**: 512
* **Tokenizer class**: DistilBertTokenizer
* **Other options**: No cleanup of tokenization spaces, supports Chinese character tokenization[[6]](#fn6)

**trainer\_state.json**

Tracks model training state and history:

* **Epochs**: 3.0 (all planned epochs completed)
* **Global step**: 639 (training finished)
* **Logging**: Every 10 steps, with loss, grad norm, learning rate, etc.
* **Eval loss**: 0.127 (after epoch 1), 0.155 (after epoch 2), 0.196 (after epoch 3)
* **Batch size**: 16
* **Save steps**: 500
* **Total FLOPs**: 338,586,670,964,736
* **Callbacks**: TrainerControl flags for stopping, saving, logging, etc.[[7]](#fn7)

**Summary Table**

|  |  |  |
| --- | --- | --- |
| File Name | Status | Description/Key Contents |
| rng\_state.pth | Not readable | PyTorch RNG state (binary)[[1]](#fn1) |
| [scheduler.pt](http://scheduler.pt) | Not readable | PyTorch scheduler state (binary)[[2]](#fn2) |
| training\_args.bin | Not readable | Training arguments (binary)[[3]](#fn3) |
| special\_tokens\_map.json | Readable | Defines [CLS], [MASK], [PAD], [SEP], [UNK] tokens[[4]](#fn4) |
| tokenizer.json | Readable | Full tokenizer config, vocab, normalization, post-processing, special tokens[[5]](#fn5) |
| tokenizer\_config.json | Readable | Tokenizer settings, max length, lowercasing, special tokens, class[[6]](#fn6) |
| trainer\_state.json | Readable | Training progress, eval loss, step/loss history, batch size, callbacks[[7]](#fn7) |

**Conclusion:**  
All text-based configuration and training state files are readable and contain standard BERT/DistilBERT tokenizer and training information. The binary files (rng\_state.pth, scheduler.pt, training\_args.bin) could not be read as text, which is expected. To access their contents, you would need to load them in a Python environment using the appropriate PyTorch or Hugging Face tools. The rest of the files confirm a standard setup for a BERT-like tokenizer and a completed training run.