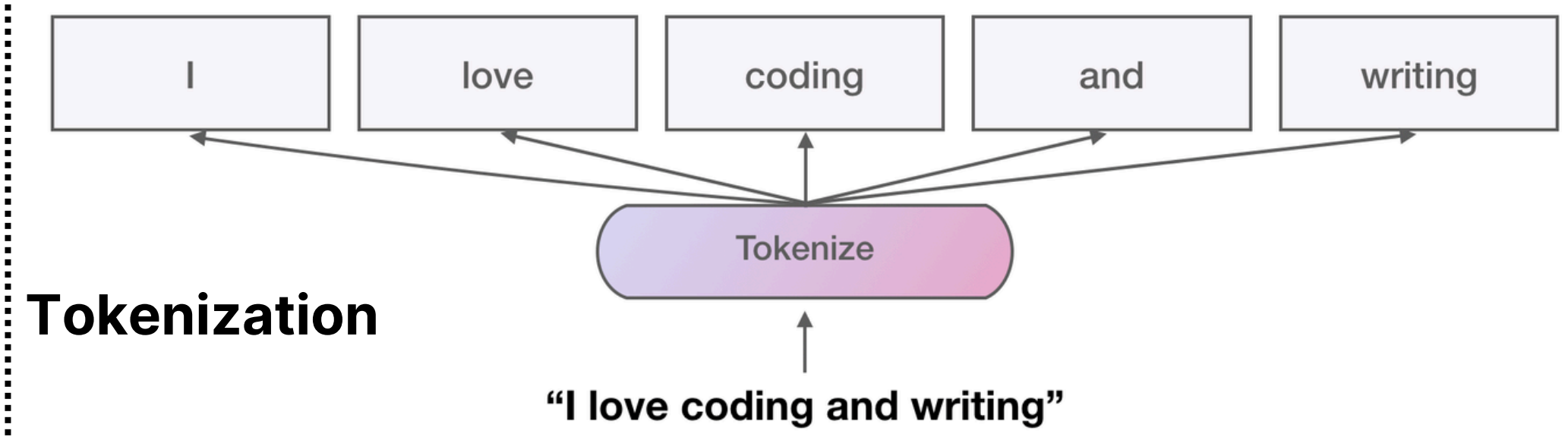
Analytics Vidhya

# Mastering LLMs

# Day 10: Implenting Tokenizers




## Tokenization

```python
from transformers import AutoTokenizer, AutoModel

tokenizer = AutoTokenizer.from_pretrained('bert-base-uncased')

sample_text = "Transformers are very powerful models for natural language processing."

inputs = tokenizer(sample_text, return_tensors="pt", padding=True, truncation=True)

print("Tokenized Output:", inputs)

model = AutoModel.from_pretrained('bert-base-uncased')

outputs = model(**inputs)

print("Last Hidden State Shape:", outputs.last_hidden_state.shape)
print("Pooler Output Shape:", outputs.pooler_output.shape)
```

# Line by Line implementation of Tokenizer

# 1. Importing Libraries

```
from transformers import AutoTokenizer, AutoModel
```

- **AutoTokenizer**: Automatically detects and loads the appropriate tokenizer for the specified model.
- **AutoModel**: Automatically detects and loads the appropriate pre-trained model based on the specified model name.

Using AutoTokenizer and AutoModel makes the code adaptable for various transformer models, such as BERT, RoBERTa, DistilBERT, and more.

# 2. Initializing the Tokenizer

```
tokenizer = AutoTokenizer.from_pretrained('bert-base-uncased')
```

- Loads the pre-trained BERT tokenizer for the bert-base-uncased model.
- The bert-base-uncased model processes text in lowercase form, meaning it converts all text to lowercase before tokenization.

**Key Benefits**

- Ensures that the input text is tokenized according to the model's vocabulary.
- Handles various tokenization details, such as WordPiece tokenization used by BERT.

# 3. Defining Sample Text

```
sample_text = "Transformers are very powerful models for natural language processing."
```

A sample sentence to demonstrate the tokenization and model inference process.

# 4. Tokenizing the Input Text

```
inputs = tokenizer(sample_text, return_tensors="pt", padding=True,
truncation=True)
```

- **return_tensors**="pt": Returns the tokenized output as PyTorch tensors (use "tf" for TensorFlow).

- **padding**=True: Ensures the text is padded to the required length.

- **truncation**=True: Truncates text that exceeds the model's maximum input length.

The output inputs dictionary contains:

- **input_ids**: Tokenized numerical representations of the input text.

- **attention_mask**: Specifies which tokens are real input vs. padding (1 for real tokens, 0 for padding).

- **token_type_ids** (optional): Distinguishes between different parts of the input if applicable (e.g., question-answer tasks).

## Example output

```
{
   'input_ids': tensor([[101,  ... , 102]]),
   'attention_mask': tensor([[1, 1, 1, ... , 1]])
}
```

Analytics Vidhya

# 5. Printing Tokenized Output

```
print("Tokenized Output:", inputs)
```

Displays the tokenized representation, useful for debugging and understanding how text is transformed before entering the model.

# 6. Initializing the Pre-Trained Model

```
model = AutoModel.from_pretrained('bert-base-uncased')
```

- Loads the pre-trained bert-base-uncased model.
- This model processes the tokenized input and generates contextual embeddings for each token.
- Using AutoModel ensures compatibility with various architectures, not just BERT.

# 7. Passing Tokenized Input to the Model

```
outputs = model(**inputs)
```

- The tokenized input is passed to the BERT model for processing.

The model output contains:

- last_hidden_state: A tensor of contextual embeddings for each token in the input sequence.
- pooler_output: A single vector representation of the entire input (useful for tasks like classification).

# 8. Printing Model Outputs

```
print("Last Hidden State Shape:", outputs.last_hidden_state.shape)
print("Pooler Output Shape:", outputs.pooler_output.shape)
```

**outputs.last_hidden_state**

- Provides embeddings of each token in the input sequence.
- Shape: [batch_size, sequence_length, hidden_size]
- Example for bert-base-uncased: [1, 16, 768] (batch of 1, 16 tokens, 768 hidden dimensions per token).

**outputs.pooler_output**

- Provides a single vector representation for the entire sequence.
- Shape: [batch_size, hidden_size], e.g., [1, 768].
- Often used for classification and other downstream tasks.

# Summary of Key Steps

- Import AutoTokenizer and AutoModel (recommended for flexibility).
- Initialize the tokenizer using bert-base-uncased.
- Prepare the input text.
- Tokenize the input with padding and truncation enabled.
- Print the tokenized output.
- Load the pre-trained BERT model using AutoModel.
- Pass the tokenized input to the model to obtain embeddings.
- Print and analyze the output embeddings.

Analytics Vidhya

Stay Tuned for **Day 11** of

# Mastering LLMs