BERT

BERT (Bidirectional Encoder Representations from Transformers)

What is BERT?

- **BERT (Bidirectional Encoder Representations from Transformers)** is a powerful deep learning model for natural language processing (NLP).
- **It reads every word in both directions at once, so it “understands” context better than older models**
- It is **pre-trained** on a large corpus of text (e.g., Wikipedia, books) and can be **fine-tuned** for specific tasks like classification, named entity recognition (NER), etc.
- Unlike older models, BERT reads text **bidirectionally** (left-to-right and right-to-left), which helps it understand context better.
- Think of it as looking at words from both directions, before and after, to understand their meaning better

> 💡 **BioBERT = BERT trained further on PubMed & clinical text.** Same architecture, but the vocabulary and weights understand biomedical words (e.g., “tachycardia”).

Key Concepts

- **Tokenization**: BERT breaks text into subwords (e.g., "unhappiness" → "un", "happiness").
- **Embeddings**: Words are converted into numerical vectors.

- **Transformer Architecture**: BERT uses attention mechanisms to weigh the importance of words in a sentence.
- **Pre-training & Fine-tuning**:
  - Pre-training: BERT learns general language patterns (Masked Language Modeling + Next Sentence Prediction).
  - Fine-tuning: Adapt BERT for specific tasks (e.g., classification, NER).

Building blocks of BERT

Block	What it is	Why it exists
WordPiece tokenizer	Breaks “hypertension” → “hyper ##tension”. 30 k sub-words.	Keeps vocabulary small yet handles any new word.
Token embeddings	One vector per token.	The raw “meaning.”
Position embeddings	Adds “I am token #7.”	Transformers otherwise have no sense of order.
12 × Encoder layers (base model)	Each layer has:• Multi-Head Self-Attention (12 heads)• Feed-Forward network	Attention lets every token talk to every other; FFN mixes information.
CLS token	Fake token at the start.	After the 12 layers, its vector is the “summary” of the whole sentence for classification.

Default parameters to remember

Parameter	Default in 🤗 Transformers	What happens if you change it
`hidden_size`	768	Larger → stronger but slower.
`num_attention_heads`	12	Must divide `hidden_size`.
`max_position_embeddings`	512	Limits sentence length.
`hidden_dropout_prob`	0.1	↑ dropout can help small data, ↓ may overfit.

Pre-training tasks (how BERT learned English/Biomedical)

1. **Masked-Language-Modeling (MLM)**

   Randomly mask 15 % tokens, ask BERT to guess them.

2. **Next-Sentence-Prediction (NSP)**

   Show two sentences; 50 % are consecutive, 50 % random; predict "IsNext?"

During **fine-tuning** these heads are thrown away and replaced with yours.

BioBERT (and friends)

Model	Corpus	Good for
`dmis-lab/biobert-v1.1`	PubMed abstracts + PMC full text	Generic biomedical NLP
`microsoft/BiomedNLP-PubMedBERT-base-uncased-abstract`	PubMed only, trained **from scratch**	Strong on unseen biomedical words
`bert-base-uncased` (vanilla)	Wikipedia + Books	Baseline when data is very general

Tip: always start with BioBERT; fall back to vanilla if BioBERT's tokenizer keeps splitting too many consumer-language words.

Trivia & interview gold-nuggets

- **Parameter count:** BERT-base has 110 M; good to mention.
- **CLS vs. pooled output:** `model.pooler_output` is a dense-tanh of CLS; for most tasks, logits already come from CLS.
- **Why BioBERT over SciBERT?** BioBERT's tokenizer keeps chemical names intact; SciBERT splits "acetaminophen" weirdly.
- **Common pitfall:** Forgetting to update `num_labels`—model silently assumes 2 and crashes if your label map ≠ 2.

Create VENV

```
conda create -n bert_env python=3.10 -y
```

Then activate the environment:

```
conda activate bert_env
```

Create `requirements.txt`

```
# ========================
# Core BERT Libraries
# ========================
transformers==4.40.1        # BERT, BioBERT, etc.
datasets==2.19.1            # Dataset handling from Hugging Face

# ========================
# Deep Learning Backends
# ========================

# TensorFlow (Keras)
tensorflow==2.15.0          # Optional: omit if using PyTorch only

# PyTorch (install this even if you use TensorFlow — many models require it)
torch==2.3.0                # Required for many Hugging Face models (even for to
kenizers)

# ========================
# NLP Preprocessing
# ========================
tokenizers==0.19.1          # Fast tokenizer backend
spacy==3.7.2                # For custom tokenization, NER, POS
nltk==3.8.1                 # Stopwords, stemming, etc.
tqdm==4.66.4                # Progress bars

# ========================
# Data Science Core
```

```
# ========================
pandas==2.2.2
numpy==1.26.4
scikit-learn==1.5.0        # Metrics like precision/recall/f1
matplotlib==3.8.4
seaborn==0.13.2

# ========================
# Demo / Deployment
# ========================
streamlit==1.35.0          # For fast web apps
gradio==4.26.0             # For interactive demos

ipykernel
```

```
pip install -r requirements.txt
```

Save models in venv

1. Go to → C:\Users\Jeevan\.cache\huggingface\hub
2. Create a file: config.json
3. Paste exactly:

```
{
  "cache_dir": "D:/Python_Env/BERT/hf_models"
}
```

📁 Create:

```
D:/Python_Env/BERT/hf_models
```

✅ To Check If It Works:

Run this Python code in your virtual environment:

```
from transformers import AutoModel, AutoTokenizer

model_name = "bert-base-uncased"
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModel.from_pretrained(model_name)
```

Then check:

Go to:

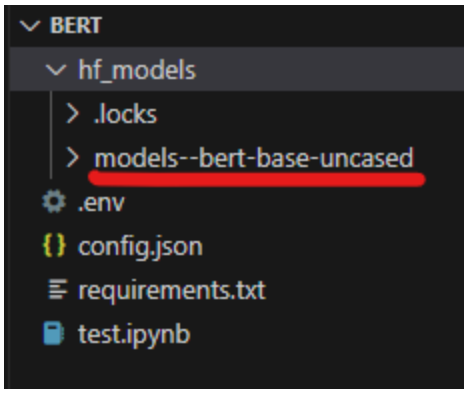
```
D:/Python_Env/BERT/hf_models/models--bert-base-uncased
```

If this folder exists, 🎯 it is working correctly.

> 💡 **This is not working**

Try:

```
model = AutoModel.from_pretrained(
    "bert-base-uncased",
    cache_dir="hf_models"  # Custom cache path
)
```



- This is working