NLP

```
from collections import Counter
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
import string
import nltk

# download necessary NLTK data files
nltk.download('punkt_tab')
nltk.download('stopwords')

# Sample text corpus

text_corpus = """Natural Language Processing (NLP) is a fascinating fiels of Artifical Intelligence (AI) that focuses on the interactio
NLP techniques enable machines to understand and process human language."""

# Tokenize the text
tokens = word_tokenize(text_corpus.lower())

# Remove punctuation and stopwords
stop_words = set(stopwords.words('english'))
cleaned_tokens = [word for word in tokens if word.isalnum() and word not in stop_words]

# Compute word frequencies
word_frequencies = Counter(cleaned_tokens)

# Print the most common words
for word, frequency in word_frequencies.most_common(10):
    print(f"{word}: {frequency}")
```

```
language: 3
nlp: 2
human: 2
natural: 1
processing: 1
fascinating: 1
fiels: 1
artifical: 1
intelligence: 1
ai: 1
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Package punkt_tab is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Attention mechanism

```
import numpy as np

# Step 1: Define Query(Q), Key (K), and Value (V) matrices
Q = np.array([[1, 0, 1]]) # Query vector (1x3)
K = np.array([[1, 1, 0], [1, 1, 0], [0, 0, 1]]) # Key vectors (3x3)
V = np.array([[1, 2],
              [0, 3],
              [1, 1]]) # Value vectors (3x2)

# Step 2: Compute dot product between Q and K^T (similarity scores)
scores = np.dot(Q, K.T)

# Step 3: Scale the scores (optinional but common)
dk = Q.shape[1] # dimension of key vectors
scaled_scores = scores / np.sqrt(dk)

# Step 4: Compute softmax to get attention weights
attention_weights = np.exp(scaled_scores) / np.sum(np.exp(scaled_scores), axis=1, keepdims=True)

# Step 5: Multitly attention weights with values matrix to get output
output = np.dot(attention_weights, V)

# Display results
print("Query (Q):")
print(Q)
print("\nKey Vectors (K):")
print(K)
print("\nValue Vectors (V):")
print(V)
print("\nAttention Weights:")
print(attention_weights)
print("\nScaled Scores:")
print(scaled_scores)
print("\nOutput:")
print(output)
```

```
Query (Q):
[[1 0 1]]

Key Vectors (K):
[[1 1 0]
 [1 1 0]
 [0 0 1]]

Value Vectors (V):
[[1 2]
 [0 3]
 [1 1]]

Attention Weights:
[[0.33333333 0.33333333 0.33333333]]

Scaled Scores:
[[0.57735027 0.57735027 0.57735027]]

Output:
[[0.66666667 2.        ]]
```

BERT fine-tuning

```
!pip install datasets==2.18.0
```

```
Requirement already satisfied: datasets==2.18.0 in /usr/local/lib/python3.11/dist-packages (2.18.0)
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-packages (from datasets==2.18.0) (3.18.0)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.11/dist-packages (from datasets==2.18.0) (2.0.2)
Requirement already satisfied: pyarrow>=12.0.0 in /usr/local/lib/python3.11/dist-packages (from datasets==2.18.0) (18.1.0)
Requirement already satisfied: pyarrow-hotfix in /usr/local/lib/python3.11/dist-packages (from datasets==2.18.0) (0.7)
Requirement already satisfied: dill<0.3.9,>=0.3.0 in /usr/local/lib/python3.11/dist-packages (from datasets==2.18.0) (0.3.7)
Requirement already satisfied: pandas in /usr/local/lib/python3.11/dist-packages (from datasets==2.18.0) (2.2.2)
Requirement already satisfied: requests>=2.19.0 in /usr/local/lib/python3.11/dist-packages (from datasets==2.18.0) (2.32.3)
Requirement already satisfied: tqdm>=4.62.1 in /usr/local/lib/python3.11/dist-packages (from datasets==2.18.0) (4.67.1)
Requirement already satisfied: xxhash in /usr/local/lib/python3.11/dist-packages (from datasets==2.18.0) (3.5.0)
Requirement already satisfied: multiprocess in /usr/local/lib/python3.11/dist-packages (from datasets==2.18.0) (0.70.15)
Requirement already satisfied: fsspec<=2024.2.0,>=2023.1.0 in /usr/local/lib/python3.11/dist-packages (from fsspec[http]<=2024.2.0,
Requirement already satisfied: aiohttp in /usr/local/lib/python3.11/dist-packages (from datasets==2.18.0) (3.11.15)
Requirement already satisfied: huggingface-hub>=0.19.4 in /usr/local/lib/python3.11/dist-packages (from datasets==2.18.0) (0.33.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packages (from datasets==2.18.0) (24.2)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.11/dist-packages (from datasets==2.18.0) (6.0.2)
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in /usr/local/lib/python3.11/dist-packages (from aiohttp->datasets==2.18.0)
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.11/dist-packages (from aiohttp->datasets==2.18.0) (1.3.2)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.11/dist-packages (from aiohttp->datasets==2.18.0) (25.3.0)
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.11/dist-packages (from aiohttp->datasets==2.18.0) (1.7.0
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.11/dist-packages (from aiohttp->datasets==2.18.0) (6.5
Requirement already satisfied: propcache>=0.2.0 in /usr/local/lib/python3.11/dist-packages (from aiohttp->datasets==2.18.0) (0.3.2)
Requirement already satisfied: yarl<2.0,>=1.17.0 in /usr/local/lib/python3.11/dist-packages (from aiohttp->datasets==2.18.0) (1.20.
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.19.4-
Requirement already satisfied: hf-xet<2.0.0,>=1.1.2 in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.19.4->datase
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests>=2.19.0->datasets
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests>=2.19.0->datasets==2.18.0) (3
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests>=2.19.0->datasets==2.18
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests>=2.19.0->datasets==2.18
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from pandas->datasets==2.18.0) (2
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas->datasets==2.18.0) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas->datasets==2.18.0) (2025.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.8.2->pandas->datasets==
```

```
# Load the dataset

from datasets import load_dataset
dataset = load_dataset("imdb")
```

```
# Load pretrained BERT and Tokenizer

from transformers import BertTokenizer, BertForSequenceClassification
tokenizer = BertTokenizer.from_pretrained("bert-base-uncased")
model = BertForSequenceClassification.from_pretrained("bert-base-uncased", num_labels=2)
```

```
tokenizer_config.json:  100%                     48.0/48.0  [00:00<00:00,  1.56kB/s]

vocab.txt:  100%                                 232k/232k  [00:00<00:00,  2.88MB/s]

tokenizer.json:  100%                            466k/466k  [00:00<00:00,  5.85MB/s]

config.json:  100%                               570/570  [00:00<00:00,  35.1kB/s]

model.safetensors:  100%                         440M/440M  [00:06<00:00,  96.2MB/s]
Some weights of BertForSequenceClassification were not initialized from the model checkpoint at bert-base-uncased and are newly init
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
```

```
# Tokenize the Dataset
def tokenize_function(example):
  return tokenizer(example["text"], padding="max_length", truncation=True)

tokenized_datasets = dataset.map(tokenize_function, batched=True)
```

```
Map:  100%                     25000/25000  [01:50<00:00,  191.55  examples/s]

Map:  100%                     25000/25000  [02:09<00:00,  182.51  examples/s]

Map:  100%                     50000/50000  [04:33<00:00,  207.55  examples/s]
```

```
# Prepare for training

from transformers import TrainingArguments, Trainer

training_args = TrainingArguments(output_dir="./results",
                  eval_strategy="epoch",
                  per_device_train_batch_size=8,
                  per_device_eval_batch_size=8,
                  num_train_epochs=3,
                  weight_decay=0.01)

trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_datasets["train"].shuffle(seed=42).select(range(2000)),
    eval_dataset=tokenized_datasets["test"].shuffle(seed=42).select(range(500))
)

trainer.train()
```

```
# Evaluate the model
trainer.evaluate()
```

```
[63/63 10:51]
{'eval_loss': 0.39584383368492126,
 'eval_runtime': 662.6447,
 'eval_samples_per_second': 0.755,
 'eval_steps_per_second': 0.095,
 'epoch': 3.0}
```

```
# Make Predictions
text = "This movie was fantastic!"
inputs = tokenizer(text, return_tensors="pt", truncation=True, padding=True)
outputs = model(**inputs)
logits = outputs.logits
predicted_class = logits.argmax().item()
print(predicted_class)
```

```
1
```

Comece a programar ou gere código com IA.