Folha de Dicas

Pacote/Método	Descrição	Exemplo de código

NLTK	NLTK é uma biblioteca Python usada em processamento de linguagem natural (NLP) para tarefas como tokenização e processamento de texto. O exemplo de código mostra como você pode tokenizar texto usando o tokenizador baseado em palavras do NLTK.	`import nltk`
		`nltk.download("punkt")`
		`from nltk.tokenize import word_tokenize`
		`text = "Unicórnios são reais. Eu vi um unicórnio ontem. Não consegui vê-lo hoje."`
		`token = word_tokenize(text)`
		`print(token)`

spaCy	spaCy é uma biblioteca de código aberto usada em NLP. Ela fornece ferramentas para tarefas como tokenização e embeddings de palavras. O exemplo de código mostra como você pode tokenizar texto usando o tokenizador baseado em palavras do spaCy.	`import spacy`
		`text = "Unicórnios são reais. Eu vi um unicórnio ontem. Não consegui vê-lo hoje."`
		`nlp = spacy.load("en_core_web_sm")`
		`doc = nlp(text)`
		`token_list = [token.text for token in doc]`
		`print("Tokens:", token_list)`

BertTokenizer	BertTokenizer é um tokenizador baseado em subpalavras que usa o algoritmo WordPiece. O exemplo de código mostra como você pode tokenizar texto usando o BertTokenizer.	`from transformers import BertTokenizer`
		`tokenizer = BertTokenizer.from_pretrained("bert-base-uncased")`
		`tokenizer.tokenize("A IBM me ensinou sobre tokenização.")`

Pacote/Método	Descrição	Exemplo de código

XLNetTokenizer	XLNetTokenizer tokeniza texto usando os algoritmos Unigram e SentencePiece. O exemplo de código mostra como você pode tokenizar texto usando o XLNetTokenizer.	`from transformers import XLNetTokenizer`
		`tokenizer = XLNetTokenizer.from_pretrained("xlnet-base-cased")`
		`tokenizer.tokenize("A IBM me ensinou sobre tokenização.")`

torchtext	A biblioteca torchtext é parte do ecossistema PyTorch e fornece as ferramentas e funcionalidades necessárias para NLP. O exemplo de código mostra como você pode usar o torchtext para gerar tokens e convertê-los em índices.	`from torchtext.vocab import build_vocab_from_iterator`
		
		Define um conjunto de dados
		
		`dataset = [`
		`(1,"Introdução ao NLP"),`
		`(2,"Fundamentos do PyTorch"),`
		`(1,"Técnicas de NLP para Classificação de Texto"),`
		`(3,"Reconhecimento de Entidades Nomeadas com PyTorch"),`
		`(3,"Análise de Sentimentos usando PyTorch"),`
		`(3,"Tradução Automática com PyTorch"),`
		`(1,"NLP Entidade Nomeada, Análise de Sentimentos, Tradução Automática"),`
		`(1,"Tradução Automática com NLP"),`
		`(1,"Entidade Nomeada vs Análise de Sentimentos em NLP")]`
		
		Aplica o tokenizador ao texto para obter os tokens como uma lista
		
		`from torchtext.data.utils import get_tokenizer`
		`tokenizer = get_tokenizer("basic_english")`
		`tokenizer(dataset[0][1])`
		
		Recebe um iterador de dados como entrada, processa o texto do iterador, e gera a saída tokenizada individualmente
		
		`def yield_tokens(data_iter):`
		`for _,text in data_iter:`
		`yield tokenizer(text)`
		
		Cria um iterador
		
		`my_iterator = yield_tokens(dataset)`
		
		Busca o próximo conjunto de tokens do conjunto de dados
		
		`next(my_iterator)`
		
		Converte tokens em índices e define <unk> como o palavra padrão se uma palavra não for encontrada no vocabulário

Pacote/Método	Descrição	Exemplo de código
		vocab = build_vocab_from_iterator(yield_tokens(dataset), specials=[“<unk>”]) vocab.set_default_index(vocab[“<unk>”]) **Fornece um dicionário que mapeia palavras para seus índices numéricos corresponde** vocab.get_stoi()
vocab	O objeto vocab faz parte da biblioteca torchtext do PyTorch. Ele mapeia tokens para índices. O exemplo de código mostra como você pode aplicar o objeto vocab diretamente aos tokens.	# Recebe um iterador como entrada e extrai a próxima frase tokenizada. **Cria uma lista de índices de tokens usando o dicionário de vocabulário para cada** def get_tokenized_sentence_and_indices(iterator): tokenized_sentence = next(iterator) token_indices = [vocab[token] for token in tokenized_sentence] return tokenized_sentence, token_indices **Retorna as sentenças tokenizadas e os índices de token correspondentes.** **Repete o processo.** tokenized_sentence, token_indices = get_tokenized_sentence_and_indices(meu_iterador) next(meu_iterador) **Imprime a sentença tokenizada e seus índices de token correspondentes.** print(“Sentença Tokenizada:”, tokenized_sentence) print(“Índices de Token:”, token_indices)
Tokens especiais no PyTorch: <eos> e <bos>	Tokens especiais são tokens introduzidos nas sequências de entrada para transmitir informações específicas ou	# Adiciona <bos> no início e <eos> no final das sentenças tokenizadas **usando um loop que itera sobre as sentenças nos dados de entrada** tokenizer_en = get_tokenizer(‘spacy’, language=’en_core_web_sm’) tokens = [] max_length = 0

Pacote/Método	Descrição	Exemplo de código

	servir a um propósito particular durante o treinamento. O exemplo de código mostra o uso de <bos> e <eos> durante a tokenização. O token <bos> denota o início da sequência de entrada, e o token <eos> denota o fim.	`for line in lines:`
		`  tokenized_line = tokenizer_en(line)`
		`  tokenized_line = [‘<bos>’] + tokenized_line + [‘<eos>’]`
		`  tokens.append(tokenized_line)`
		`  max_length = max(max_length, len(tokenized_line))`

Tokens especiais no PyTorch: <pad>	O exemplo de código mostra o uso do token <pad> para garantir que todas as sentenças tenham o mesmo comprimento.	`# Preenche as linhas tokenizadas`
		`for i in range(len(tokens)):`
		`  tokens[i] = tokens[i] + ['<pad>'] * (max_length - len(tokens[i]))`

Classe Dataset no PyTorch	A classe Dataset permite acessar e recuperar amostras individuais de um conjunto de dados. O exemplo de código mostra como você pode criar um conjunto de dados personalizado e acessar amostras.	`# Importa a classe Dataset e define uma lista de sentenças`
		`from torch.utils.data import Dataset`
		`sentences = ["Se você quer saber como é um homem, observe como ele trata seus inferiores, não seus iguais.",`
		`"Fae é uma amiga inconstante, Harry."]`
		
		`# Faz o download e lê os dados`
		
		`class CustomDataset(Dataset):`
		`  def init(self, sentences):`
		`  self.sentences = sentences`
		`  # Retorna o comprimento dos dados`
		`  def len(self):`
		`  return len(self.sentences)`
		`  # Retorna um item no índice`
		`  def getitem(self, idx):`
		`  return self.sentences[idx]`
		
		`# Cria um objeto de conjunto de dados`
		
		`dataset=CustomDataset(sentences)`
		
		`# Acessos a amostras como em uma lista`
		
		`Por exemplo, dataset[0]`

Pacote/Método	Descrição	Exemplo de código

Classe DataLoader no PyTorch	A classe DataLoader permite o carregamento e a iteração eficientes sobre conjuntos de dados para treinar modelos de aprendizado profundo. O exemplo de código mostra como você pode usar a classe DataLoader para gerar lotes de sentenças para processamento adicional, como treinar um modelo de rede neural	`# Cria um objeto iterador`
		`data_iter = iter(dataloader)`
		
		Chama a próxima função para retornar novos lotes de amostras
		
		`next(data_iter)`
		
		Cria uma instância do conjunto de dados personalizado
		
		`from torch.utils.data import DataLoader`
		`custom_dataset = CustomDataset(sentences)`
		
		Especifica um tamanho de lote
		
		`batch_size = 2`
		
		Cria um carregador de dados
		
		`dataloader = DataLoader(custom_dataset, batch_size=batch_size, shuffle=True)`
		
		Imprime as sentenças em cada lote
		
		`for batch in dataloader:`
		`  print(batch)`

Função de colagem personalizada no PyTorch	A função de colagem personalizada é uma função definida pelo usuário que define como amostras individuais são coladas ou agrupadas. Você pode utilizar a função de colagem para	`# Define uma função de colagem personalizada`
		`def collate_fn(batch):`
		`  tensor_batch = []`
		
		Tokeniza cada amostra no lote
		
		`for sample in batch:`
		
		`  tokens = tokenizer(sample)`
		
		Mapeia tokens para números usando o vocab

Pacote/Método	Descrição	Exemplo de código

	tarefas como tokenização, conversão de índices tokenizados e transformação do resultado em um tensor. O exemplo de código mostra como você pode usar uma função de colagem personalizada em um carregador de dados.	`tensor_batch.append(torch.tensor([vocab[token] for token in tokens]))`
		
		Preenche as sequências dentro do lote para ter comprimentos iguais
		
		`padded_batch = pad_sequence(tensor_batch,batch_first=True)`
		
		`return padded_batch`
		
		Cria um carregador de dados usando a função de colagem e o conjunto de dados pers
		
		`dataloader = DataLoader(custom_dataset, batch_size=batch_size, shuffle=True, collate_fn=collate_fn)`

Skills Network