Laporan Chapter 6 - NLP dengan Hugging Face

The Tokenizers Library

Nama Rey Rizqi Anugerah
Kelas TK45 01
NIM: 1103210146

Tujuan Chapter

1. Memahami fungsi utama library **Tokenizers** untuk tokenisasi teks.
2. Memanfaatkan fitur-fitur canggih seperti tokenisasi paralel dan manipulasi tokenizer.
3. Belajar membuat tokenizer kustom dari awal atau memodifikasi yang sudah ada.

1. Apa Itu The Tokenizers Library?

Library **Tokenizers** dirancang untuk proses tokenisasi yang cepat dan efisien. Dibangun dengan Rust untuk performa tinggi, library ini mendukung berbagai metode tokenisasi yang digunakan dalam model NLP.

Kelebihan Tokenizers

- **Cepat**: Dibangun di atas Rust, memungkinkan pemrosesan jutaan teks dalam hitungan detik.
- **Serbaguna**: Mendukung metode tokenisasi seperti WordPiece, BPE, dan Unigram.
- **Fleksibel**: Mendukung pembuatan tokenizer kustom.
- **Multi-threading**: Mendukung tokenisasi paralel.

2. Instalasi dan Import

Instalasi

Copy code

pip install tokenizers

Import

from tokenizers import Tokenizer

from tokenizers.models import WordPiece

from tokenizers.pre_tokenizers import Whitespace

from tokenizers.trainers import WordPieceTrainer

3. Membuat Tokenizer Kustom

Langkah-Langkah Membuat Tokenizer Kustom

1. **Inisialisasi Tokenizer**:

   tokenizer = Tokenizer(WordPiece(unk_token="[UNK]"))

2. **Menambahkan Pre-tokenizer**: Pre-tokenizer digunakan untuk memisahkan teks mentah menjadi token.

   tokenizer.pre_tokenizer = Whitespace()

3. **Pelatihan Tokenizer**: Menggunakan trainer untuk melatih tokenizer pada dataset tertentu.

   trainer = WordPieceTrainer(vocab_size=30000, special_tokens=["[UNK]", "[CLS]", "[SEP]", "[PAD]", "[MASK]"])

   files = ["data.txt"] # File teks untuk pelatihan

   tokenizer.train(files, trainer)

4. **Menyimpan dan Memuat Tokenizer**:

   tokenizer.save("custom_tokenizer.json")

   tokenizer = Tokenizer.from_file("custom_tokenizer.json")

4. Menggunakan Tokenizer

Tokenisasi Teks

Tokenisasi teks menjadi subword.

output = tokenizer.encode("I am learning NLP with Hugging Face.")

print(output.tokens) # Output token

print(output.ids) # Output ID

Dekode Token

Mengonversi ID token kembali ke teks.

decoded_text = tokenizer.decode(output.ids)

print(decoded_text)

5. Fitur-Fitur Tokenizers

Pre-tokenizer

Mengatur cara pemisahan teks mentah sebelum tokenisasi.

from tokenizers.pre_tokenizers import Whitespace

tokenizer.pre_tokenizer = Whitespace()

Normalizer

Digunakan untuk menormalisasi teks, seperti mengubah huruf kapital menjadi huruf kecil.

from tokenizers.normalizers import Lowercase, NFD, StripAccents

from tokenizers import normalizers

tokenizer.normalizer = normalizers.Sequence([NFD(), Lowercase(), StripAccents()])

Post-processing

Menambahkan token khusus seperti [CLS] dan [SEP] setelah tokenisasi.

from tokenizers.processors import TemplateProcessing

tokenizer.post_processor = TemplateProcessing(

single="[CLS] $A [SEP]",

pair="[CLS] $A [SEP] $B:1 [SEP]:1",

special_tokens=[("[CLS]", 1), ("[SEP]", 2)],

)

6. Penerapan Tokenizer

Tokenisasi Paralel

Memproses banyak teks sekaligus menggunakan multi-threading.

texts = ["I love NLP.", "Hugging Face is amazing!"]

encoded_batch = tokenizer.encode_batch(texts)

for encoded in encoded_batch:

print(encoded.tokens)

7. Studi Kasus: Membuat Tokenizer WordPiece

Langkah-Langkah

1. **Inisialisasi Tokenizer**:

   from tokenizers import Tokenizer

   from tokenizers.models import WordPiece

   tokenizer = Tokenizer(WordPiece(unk_token="[UNK]"))

2. **Menambahkan Pre-tokenizer**:

   from tokenizers.pre_tokenizers import Whitespace

   tokenizer.pre_tokenizer = Whitespace()

3. **Melatih Tokenizer**:

   from tokenizers.trainers import WordPieceTrainer

   trainer = WordPieceTrainer(vocab_size=30000, special_tokens=["[UNK]", "[CLS]", "[SEP]", "[PAD]", "[MASK]"])

   tokenizer.train(["data.txt"], trainer)

4. **Menyimpan dan Memuat Tokenizer**:

   tokenizer.save("wordpiece_tokenizer.json")

   tokenizer = Tokenizer.from_file("wordpiece_tokenizer.json")

5. **Menggunakan Tokenizer**:

   output = tokenizer.encode("Hugging Face is fantastic!")

   print(output.tokens)

   print(output.ids)

Kesimpulan

Chapter ini menekankan pentingnya memahami dan memanfaatkan **Tokenizers Library** untuk meningkatkan efisiensi dalam memproses data teks. Dengan fitur seperti tokenisasi paralel, pre-tokenizer, normalizer, dan post-processing, library ini menjadi alat yang sangat berguna untuk proyek NLP.