Laporan: Pembuatan Model Bahasa Berbasis Transformer Menggunakan PyTorch

I. Pendahuluan

Proyek ini bertujuan untuk membangun dan melatih model bahasa berbasis Transformer menggunakan PyTorch. Model ini mengandalkan konsep-konsep utama dalam arsitektur Transformer, seperti self-attention, multi-head attention, dan feed-forward layers, yang terbukti sangat efektif dalam berbagai tugas pemrosesan bahasa alami (NLP). Model ini dilatih pada teks Shakespeare untuk memprediksi urutan kata berikutnya, yang dihasilkan berdasarkan konteks input.

II. Tujuan

Tujuan dari proyek ini adalah untuk:

1. Membangun model bahasa berbasis Transformer menggunakan PyTorch.
2. Melatih model pada dataset teks Shakespeare.
3. Menggunakan model untuk menghasilkan teks berdasarkan konteks yang diberikan.

III. Metodologi

A. Pengaturan Hyperparameters

Pada bagian ini, beberapa hyperparameters yang digunakan dalam pelatihan model didefinisikan. Hyperparameters ini meliputi ukuran batch, panjang blok untuk prediksi, jumlah iterasi pelatihan, dan parameter lainnya yang mengontrol struktur dan pelatihan model.

```
python
Copy code
batch_size = 16
block_size = 32
max_iters = 5000
eval_interval = 100
learning_rate = 1e-3
device = 'cuda' if torch.cuda.is_available() else 'cpu'
eval_iters = 200
n_embd = 64
n_head = 4
n_layer = 4
dropout = 0.0
```

Penjelasan:

- `batch_size`: Jumlah sampel yang diproses secara paralel.
- `block_size`: Panjang konteks yang digunakan untuk prediksi.

- `max_iters`: Jumlah iterasi pelatihan.
- `eval_interval`: Interval evaluasi untuk menghitung loss pada data pelatihan dan validasi.
- `learning_rate`: Kecepatan pembelajaran untuk optimisasi.
- `n_embd`: Dimensi embedding untuk token dan posisi.
- `n_head`: Jumlah kepala dalam multi-head attention.
- `n_layer`: Jumlah lapisan Transformer yang digunakan.

B. Persiapan Data

Teks Shakespeare digunakan sebagai dataset untuk melatih model. Dataset ini pertama-tama diproses untuk mengonversi karakter-karakter unik menjadi angka, kemudian dibagi menjadi dua bagian: data pelatihan dan data validasi.

```
python
Copy code
with open('input.txt', 'r', encoding='utf-8') as f:
    text = f.read()
chars = sorted(list(set(text)))
vocab_size = len(chars)
```

Penjelasan:

- Data dikodekan menjadi urutan angka menggunakan kamus `stoi` dan `itos` yang menghubungkan karakter dengan indeks numerik.
- Teks kemudian dibagi menjadi data pelatihan (90%) dan data validasi (10%).

C. Struktur Model

Model ini menggunakan arsitektur Transformer yang terdiri dari beberapa bagian utama:

1. **Head**: Mewakili satu kepala self-attention.
2. **MultiHeadAttention**: Menggabungkan beberapa kepala self-attention.
3. **FeedForward**: Lapisan linier dengan non-linearitas yang diterapkan setelah perhatian.
4. **Block**: Kombinasi perhatian multi-head dan lapisan feed-forward.
5. **BigramLanguageModel**: Model utama yang menggabungkan embedding token dan posisi, serta beberapa blok Transformer.

```
python
Copy code
class Head(nn.Module):
    def __init__(self, head_size):
        super().__init__()
        self.key = nn.Linear(n_embd, head_size, bias=False)
        self.query = nn.Linear(n_embd, head_size, bias=False)
        self.value = nn.Linear(n_embd, head_size, bias=False)
        self.register_buffer('tril',
torch.tril(torch.ones(block_size, block_size)))
        self.dropout = nn.Dropout(dropout)
```

D. Training dan Optimisasi

Model dilatih menggunakan algoritma optimisasi AdamW dengan pembaruan parameter berdasarkan loss yang dihitung menggunakan fungsi `cross_entropy`:

```
python
Copy code
optimizer = torch.optim.AdamW(model.parameters(),
lr=learning_rate)
```

Model dilatih selama 5000 iterasi, dan pada setiap interval evaluasi, model dievaluasi pada data pelatihan dan validasi untuk memonitor kinerjanya.

```
python
Copy code
for iter in range(max_iters):
    if iter % eval_interval == 0 or iter == max_iters - 1:
        losses = estimate_loss()
        print(f"step {iter}: train loss {losses['train']:.4f},
val loss {losses['val']:.4f}")
```

E. Hasil dan Evaluasi

Setelah pelatihan selesai, model digunakan untuk menghasilkan teks berdasarkan konteks yang diberikan. Berikut adalah contoh output yang dihasilkan oleh model:

```
python
Copy code
context = torch.zeros((1, 1), dtype=torch.long, device=device)
print(decode(m.generate(context,
max_new_tokens=2000)[0].tolist()))
```

Model dapat menghasilkan teks yang menyerupai gaya Shakespeare dengan prediksi kata berikutnya berdasarkan konteks input.

IV. Hasil

Model berhasil dilatih dengan baik, dan loss pada data pelatihan dan validasi menurun secara signifikan seiring dengan bertambahnya iterasi. Setelah pelatihan, model mampu menghasilkan teks yang koheren dan menyerupai gaya Shakespeare.

V. Kesimpulan

Model bahasa berbasis Transformer yang dibangun menggunakan PyTorch berhasil dilatih untuk memprediksi urutan kata dalam teks Shakespeare. Dengan menggunakan konsep perhatian, model ini mampu memahami konteks teks dengan lebih baik dan menghasilkan teks yang koheren. Proyek ini menunjukkan bagaimana arsitektur Transformer dapat diterapkan untuk tugas pemrosesan bahasa alami dan menghasilkan model bahasa yang sangat kuat.