GPT

Kode ini adalah eksplorasi mendalam tentang pembangunan model bahasa, yang berfokus pada arsitektur inti dari model *Transformer*. Tujuannya adalah untuk memahami prinsip-prinsip dasar yang mendasari model seperti GPT dengan membangunnya langkah demi langkah, mulai dari pengunduhan dan pemrosesan data, hingga implementasi dan pelatihan model itu sendiri. Kode ini tidak hanya berfokus pada implementasi teknis, tetapi juga memberikan pemahaman konseptual tentang teknik-teknik seperti *self-attention*, normalisasi lapisan, dan peran matriks dalam agregasi tertimbang.

Proses dimulai dengan mempersiapkan dataset teks melalui pengunduhan, tokenisasi, dan pengkodean ke dalam bentuk numerik yang dapat diproses oleh model. Kemudian, data ini dibagi menjadi set pelatihan dan validasi, dan dibentuk ke dalam *batch* yang siap untuk diumpankan ke model. Model *bigram* sederhana digunakan sebagai titik awal untuk melihat bagaimana model belajar dan menghasilkan teks. Selanjutnya, kode ini menunjukkan secara rinci bagaimana mekanisme *self-attention* bekerja, mulai dari penghitungan *query*, *key*, dan *value*, hingga penggunaan *masking* dan *softmax*. Pentingnya penskalaan pada *softmax* dan varian dari bobot juga turut dibahas dan dipraktekkan. Selain itu, kode ini juga membahas tentang *layer normalization* dan bagaimana cara kerjanya untuk menstabilkan proses pelatihan.

Di bagian akhir, semua konsep dan teknik yang telah dipelajari sebelumnya digabungkan untuk membangun model *Transformer* sederhana. Model ini kemudian dilatih dan digunakan untuk menghasilkan teks. Meskipun hasil *generate* teksnya masih belum koheren, proses ini berhasil memperlihatkan bagaimana setiap bagian dalam model *Transformer* bekerja, dan bagaimana interaksi antara setiap komponen dapat menghasilkan sebuah model yang powerful. Secara keseluruhan, kode ini memberikan pemahaman yang solid dan praktis tentang cara membangun model bahasa dengan arsitektur *Transformer*.

Biagram

kode ini secara keseluruhan dalam beberapa paragraf. Kode ini adalah implementasi dari model bahasa *bigram* sederhana menggunakan PyTorch, dengan tujuan untuk melatih model dan menghasilkan teks yang mirip dengan dataset Shakespeare kecil yang digunakan sebagai contoh.

Paragraf pertama akan menjelaskan *setup* dan data *preprocessing*. Kode ini dimulai dengan mengimpor library PyTorch dan mendefinisikan beberapa *hyperparameter* seperti ukuran batch, ukuran blok (panjang konteks), jumlah iterasi pelatihan, dan *learning rate*. Kode ini juga mengatur *seed* untuk *reproducibility*, dan menggunakan GPU jika tersedia. Selanjutnya, kode ini mengunduh dataset Shakespeare kecil dari URL yang diberikan, dan melakukan *preprocessing* sederhana yaitu membaca file, mengidentifikasi karakter unik, dan membuat *mapping* antara karakter dan integer. Dataset teks kemudian dienkode menjadi *tensor* bilangan bulat dan dibagi menjadi set pelatihan dan validasi.

Paragraf kedua berfokus pada fungsi *data loading* dan arsitektur model. Fungsi get_batch dibuat untuk mengambil *batch* data secara acak dari set pelatihan atau validasi, dengan memformat data menjadi bentuk yang sesuai (input dan target). Kemudian fungsi estimate_loss dibuat untuk mengevaluasi model setelah iterasi pelatihan. Selanjutnya, kode ini mendefinisikan class BigramLanguageModel, yang merupakan model bahasa sederhana yang terdiri dari *embedding table*. Fungsi *forward* pada model ini menerima input dan target dan kemudian menghitung *logits* dan *loss* jika *target* diberikan. Model ini juga memiliki fungsi generate untuk menghasilkan teks baru.

Paragraf ketiga menjelaskan tentang proses pelatihan model. Kode ini membuat instance dari model, memindahkannya ke GPU jika ada, dan kemudian membuat *optimizer* AdamW untuk memperbarui parameter model. *Training loop* dijalankan sejumlah max_iters dan melakukan *forward pass* pada setiap iterasi, menghitung *loss*, melakukan *backward pass*, dan memperbarui parameter model. Setiap beberapa iterasi tertentu, *loss* dievaluasi pada set pelatihan dan validasi untuk memantau kinerja model. Akhirnya, setelah pelatihan selesai, kode ini melakukan *generate* teks dengan menggunakan model yang sudah dilatih dan mencetaknya.

Secara keseluruhan, kode ini adalah contoh implementasi *end-to-end* dari model bahasa *bigram* yang komprehensif. Kode ini mencakup semua langkah penting, mulai dari *preprocessing* data, pembuatan model, pelatihan model, evaluasi model, dan juga generasi teks dari model. Meskipun model ini sederhana, kode ini dapat memberikan gambaran yang baik tentang cara model bahasa dilatih dan digunakan untuk menghasilkan teks.

Link YT : https://youtu.be/YBwTjk_fNko