IF5281 Deep Learning

Text Generation : LSTM vs Transformers

Ujian Akhir Semester Mata Kuliah Deep Learning

Oleh:

Randy Zakya Suchrady 23523027

PROGRAM STUDI MAGISTER INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG

2024

ABSTRAK

Studi ini bertujuan untuk membandingkan kinerja model berbasis LSTM dan transformers (GPT) dalam persoalan text generation khususnya pada konteks klaim paralelisasi yang dapat dilakukan oleh model transformers. Eksperimen dilakukan dengan dua babak: perbandingan arsitektur dan perbandingan panjang sekuens. Hasil utama menunjukkan bahwa model transformers memang benar memiliki kemampuan paralelisasi terlihat dari waktu pelatihannya pada jumlah parameter ataupun panjang sekuens yang bertambah secara berkala. Selain itu, diperoleh hasil lain berupa tinjauan terkait perbandingan kedalaman model, lebar model, loss, dan overfit ratio. Pada dataset yang digunakan, ditemukan bahwa model transformers cenderung memiliki loss yang lebih besar diakibatkan ukuran serta kualitas dataset serta teknik tokenisasi yang digunakan. Namun, kemampuan paralelisasi transformers membuka peluang skalabilitas untuk ukuran dataset ataupun skala model yang lebih besar. Hasil repositori kode studi dapat dilihat pada https://github.com/rdyzakya/MiniLMComparison.

Kata Kunci: LSTM, transformers, paralelisasi, waktu pelatihan

PENDAHULUAN

Arsitektur neural network berbasis attention yang dinamakan sebagai Transformers menjadi state-of-the-art dalam berbagai bidang khususnya pada bidang pemrosesan bahasa alami dalam representasi teks. Salah satu alasan kenapa transformers diusulkan yaitu dikarenakan waktu pelatihan neural network berbasis RNN seperti LSTM dan GRU cukup memakan waktu serta tidak bisa dilakukan paralelisasi yang kemudian dipecahkan permasalahan tersebut oleh arsitektur transformers. Pada laporan kali ini, akan dibuktikan apakah klaim yang diajukan oleh transformers benar, yaitu dengan membandingkannya dengan neural network berbasis RNN, yaitu LSTM. Pembuktian ini dilakukan untuk mengetahui mengenai kebenaran dari klaim arsitektur transformers. Pada tahapan eksperimen, model berbasis transformers, yaitu GPT, akan dibandingkan dengan model berbasis LSTM pada persoalan causal language modeling atau text generation. Kedua model tersebut akan dibandingkan pada dataset yang sama dengan variasi hyperparameter yang berbeda. Beberapa faktor (atau metrik) seperti jumlah parameter model, loss, dan waktu pelatihan akan menjadi tolak ukur perbandingan.

STUDI LITERATUR

Pada persoalan causal language modeling atau text generation, terdapat beberapa jenis model yang dapat digunakan untuk menyelesaikan persoalan tersebut. Mulai dari statistical language model (SLM), model neural network berbasis RNN seperti Basic RNN, GRU, dan LSTM, hingga model neural network berbasis attention seperti GPT (beserta turunannya).

Statistical Language Model (N-gram)

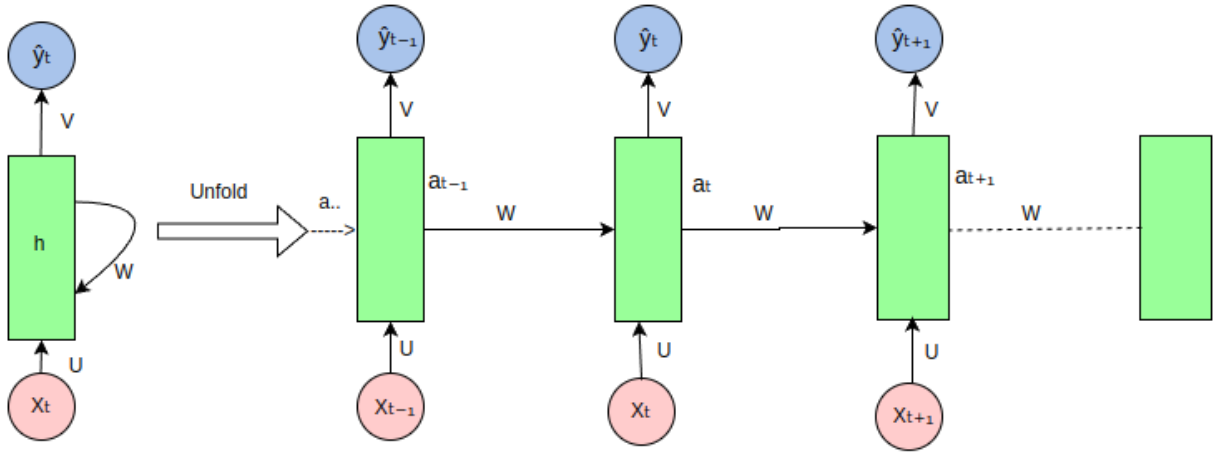
Menurut Jurafsky dan Martin (2024) pada buku Speech and Language Processing, statistical language model (SLM) merupakan model bahasa yang paling sederhana yang telah dikembangkan sejak tahun 1980. SLM bekerja dengan menggunakan perhitungan probabilitas dan statistik untuk melakukan estimasi distribusi probabilitas pada unit linguistik seperti kata, kalimat, atau dokumen. Salah satu model SLM yang paling dikenal adalah model N-gram. Model N-gram memiliki cara kerja yang sederhana, yaitu menghitung peluang kemungkinan sebuah kata atau token selanjutnya pada sebuah kalimat berdasarkan N-1 token terakhir. Contoh, pada kalimat “Aku sangat senang …”, model **bigram** (2-gram) akan menggunakan 1 token terakhir untuk menghitung probabilitas kemungkinan kata selanjutnya, sehingga hanya digunakan kata (atau token) “senang” untuk mengetahui kelanjutan kalimat tersebut. Terdapat beberapa kelebihan dan kekurangan pada penggunaan N-gram language model:

1. Kelebihan
    a. Model N-gram memiliki mekanisme yang sederhana dan beban komputasi yang dibutuhkan cukup kecil.
    b. Mudah dalam memahami (mudah diinterpretasi).
2. Kekurangan
    a. Terdapat keterbatasan konteks dikarenakan ukuran N-gram yang terbatas.
    b. Untuk menambahkan ukuran konteks (N), ukuran data latih menjadi banyak sehingga muncul permasalahan curse of dimensionality (data sparsity).

c. Tidak memiliki kemampuan generalisasi yang baik bila berhadapan dengan kasus yang tidak muncul pada data latih.

Basic RNN

Recurrent Neural Network (RNN) merupakan model neural network yang secara khusus didesain untuk menangani data sekuensial, dicetuskan oleh David Rumelhart pada tahun 1986. RNN bekerja dengan menerima input vektor untuk setiap time step, kemudian dihasilkan 2 keluaran: hidden state dan output.



Gambar 1. Arsitektur RNN (sumber : https://medium.com/@poudelsushmita878/recurrent-neural-network-rnn-architecture-explained-1d69560541ef)

Cara kerja dari RNN cukup sederhana, yaitu dengan mengolah input untuk setiap time step, kemudian model akan mengeluarkan hidden state serta output. Hidden state yang dikeluarkan sebelumnya akan digunakan untuk mengolah input pada time step berikutnya.

$$ht = g(Wx_t + Uh_{t-1}) \quad (1)$$

Pada persamaan 1, terdapat rincian persamaan dalam perhitungan update hidden state pada time step ke-t. Seperti mekanisme neural network pada umumnya, namun dengan tambahan

berupa adanya 2 input, yaitu hidden state pada time step ke-t-1 dan xt. Fungsi aktivasi g biasanya menggunakan jenis fungsi aktivasi yang bersifat halus seperti sigmoid.

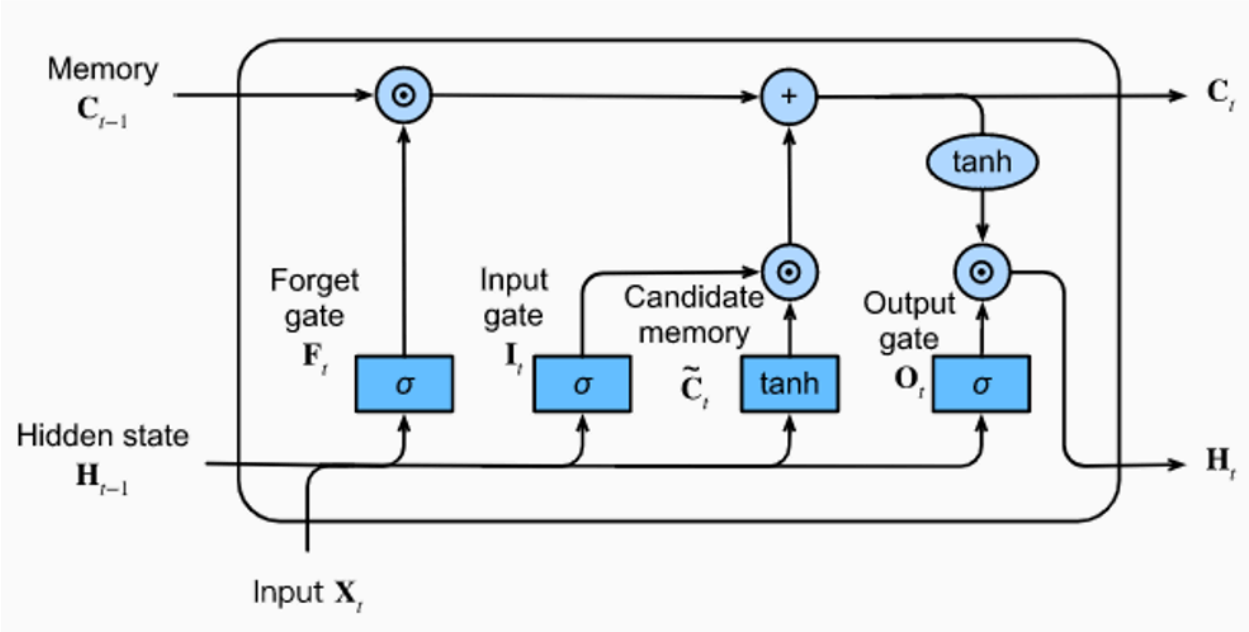
Terdapat beberapa kelebihan dan kekurangan yang dimiliki oleh RNN:

1. Kelebihan
    a. RNN dapat menangani jumlah time step yang beragam, dari time step yang sedikit sampai yang panjang.
    b. Terdapat konteks dari time step sebelumnya yang direpresentasikan oleh hidden state.
2. Kekurangan
    a. Terdapat permasalahan memory, model lupa terkait konteks yang sudah lampau sekali (terjadi pada jumlah time step yang besar).
    b. Vanishing gradient problem dan exploding gradient problem, terjadi akibat mekanisme backpropagation through time, gradien dapat menyusut nilainya hingga kecil sekali ataupun bisa saja menjadi besar sekali.
    c. Panjangnya time step mempengaruhi waktu pelatihan dan waktu inferensi.

LSTM

Sebelumnya, terjadi masalah krusial pada RNN biasa, yaitu terkait dengan persoalan long term memory, ketika sekuens terlalu panjang, konteks yang sudah lampau akan dilupakan oleh RNN. Dengan begitu, Hochreiter dan Schmidhuber (1997) mengajukan arsitektur neural network baru yang masih berbasis RNN namun bisa menyelesaikan permasalahan tersebut, yaitu Long Short-Term Memory (LSTM). Pada arsitektur LSTM, unit dari model disebut sebagai LSTM cell, terdiri dari beberapa gate: forget gate, input gate, output gate.

1. Forget gate : memilih informasi apa saja yang perlu dilupakan dan yang jangan dilupakan.
2. Input gate : mengolah input masukan dari LSTM cell.
3. Output gate : mengolah hidden state dan input untuk menghasilkan output.



Gambar 2. Arsitektur LSTM (sumber : https://medium.com/@ottaviocalzone/an-intuitive-explanation-of-lstm-a035eb6ab42c)

Terdapat cara perhitungan dari masing-masing gate yang ada pada LSTM cell. Selain itu, terdapat juga perhitungan cell state, yang menjadi informasi terkait status cell.

Tabel 1. Persamaan pada LSTM

Komponen	Persamaan
Forget Gate	$f_t^j = \sigma\left(W_f \mathbf{x}_t + U_f \mathbf{h}_{t-1} + V_f \mathbf{c}_{t-1}\right)^j$
Input Gate	$i_t^j = \sigma\left(W_i \mathbf{x}_t + U_i \mathbf{h}_{t-1} + V_i \mathbf{c}_{t-1}\right)^j$
Candidate Cell State	$\tilde{c}_t^j = \tanh\left(W_c \mathbf{x}_t + U_c \mathbf{h}_{t-1}\right)^j$
Cell State Update	$c_t^j = f_t^j c_{t-1}^j + i_t^j \tilde{c}_t^j$
Output Gate	$o_t^j = \sigma\left(W_o \mathbf{x}_t + U_o \mathbf{h}_{t-1} + V_o \mathbf{c}_t\right)^j$

Hidden State	$h_t^j = o_t^j \tanh\left(c_t^j\right)$

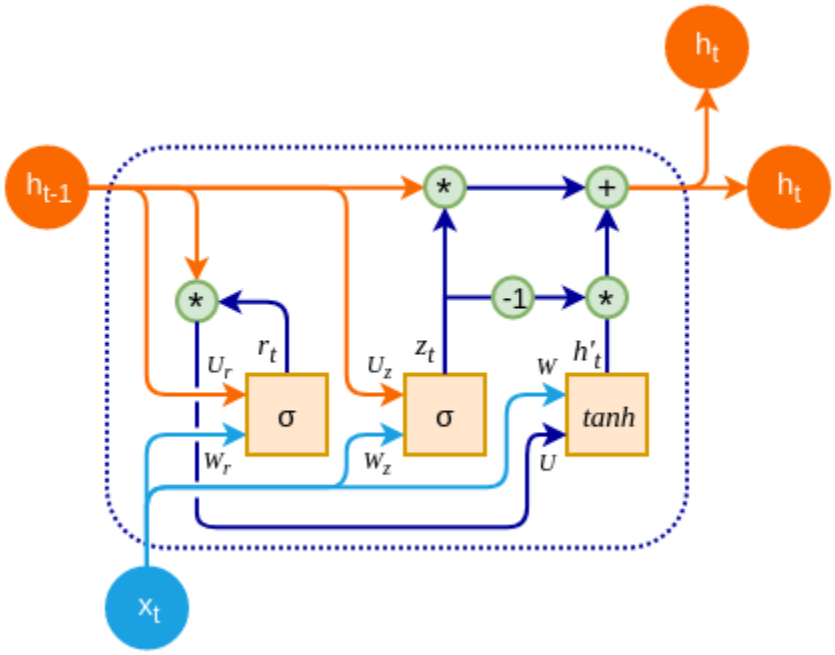
Terdapat beberapa kelebihan dan kekurangan yang dimiliki oleh LSTM:

1. Kelebihan
    a. Dapat menangani persoalan long-term memory.
    b. Dengan arsitektur ini, dapat terhindar dari terjadinya vanishing gradient problem dan exploding gradient problem.
2. Kekurangan
    a. Karena bersifat rekurens, sama seperti RNN, panjangnya time step mempengaruhi waktu pelatihan dan waktu inferensi.

GRU

Gated Recurrent Unit (GRU) diusulkan oleh Cho dkk. (2014) untuk membuat setiap unit rekuren secara adaptif menangkap dependency pada time scale yang berbeda. GRU memiliki mekanisme yang serupa dengan LSTM dengan memanfaatkan gating mechanism tanpa memisahkan cell state dan hidden state. Gating mechanism yang dimiliki oleh GRU terdiri dari update gate dan reset gate.

1. Update gate : menentukan informasi yang ada pada hidden state sebelumnya untuk mengolah hidden state sekarang.
2. Reset gate : menentukan informasi apa saja yang perlu dan tidak perlu dilupakan (sama seperti forget gate).



Gambar 3. Arsitektur GRU (sumber : https://www.oreilly.com/library/view/advanced-deep-learning/9781789956177/8ad9dc41-3237-483e-8f6b-7e5f653dc693.xhtml)

Terdapat cara perhitungan dari masing-masing gate yang ada pada GRU cell. Selain itu, terdapat juga perhitungan hidden state.

Tabel 2. Persamaan pada GRU

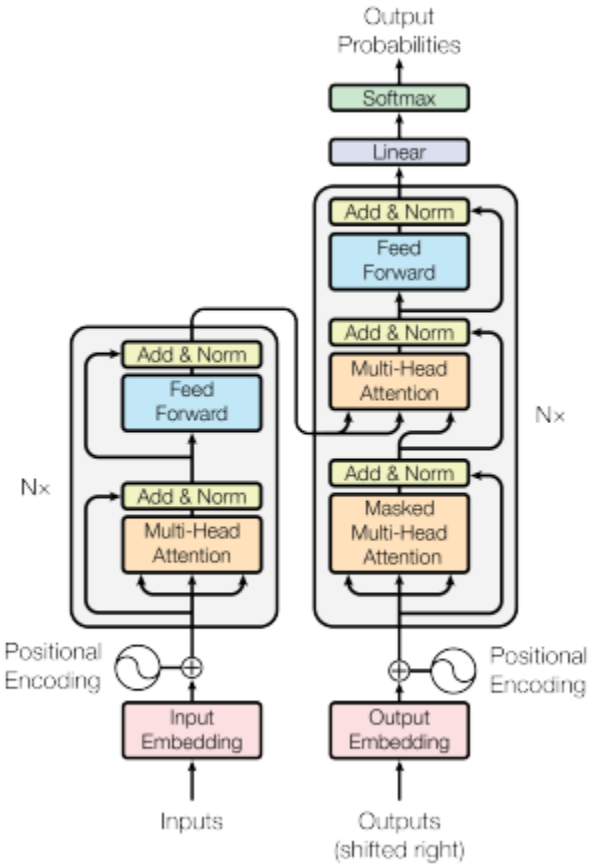
Komponen	Persamaan
Update Gate	$z_t^j = \sigma\left(W_z \mathbf{x}_t + U_z \mathbf{h}_{t-1}\right)^j$
Reset Gate	$r_t^j = \sigma\left(W_r \mathbf{x}_t + U_r \mathbf{h}_{t-1}\right)^j$
Candidate Hidden State	$\tilde{h}_t^j = \tanh\left(W \mathbf{x}_t + U\left(\mathbf{r}_t \odot \mathbf{h}_{t-1}\right)\right)^j$
Hidden State Update	$h_t^j = (1 - z_t^j) h_{t-1}^j + z_t^j \tilde{h}_t^j$

Terdapat beberapa kelebihan dan kekurangan yang dimiliki oleh GRU:

1. Kelebihan
    a. Dikarenakan hanya terdiri dari 2 gate, model menjadi lebih sederhana dan tidak sekompleks LSTM.
    b. Dikarenakan kompleksitas yang lebih sederhana, komputasi yang terjadi lebih efisien.
2. Kekurangan
    a. Karena bersifat rekurens, sama seperti RNN, panjangnya time step mempengaruhi waktu pelatihan dan waktu inferensi.
    b. Dikarenakan lebih sederhana daripada LSTM, terkadang terdapat informasi yang tidak dapat ditangkap oleh GRU yang dapat ditangkap oleh LSTM.

Transformers

Pada literatur “Attention is All You Need”, Vaswani dkk (2017) mengajukan arsitektur berbasis attention yang berbeda sekali dengan arsitektur berbasis RNN. Permasalahan yang ada pada arsitektur berbasis RNN adalah sifatnya yang rekurens membuatnya sulit untuk dilatih secara paralel. Arsitektur transformers yang diajukan terdiri dari dua komponen utama yaitu blok encoder dan blok decoder.



Gambar 4. Arsitektur Transformers (Vaswani dkk., 2017)

Terdapat beberapa mekanisme yang perlu diketahui yaitu sebagai berikut:

1. Mekanisme Attention (Multi-Head Attention + Masked Attention)

   Terinspirasi dari bagaimana sistem basis data bekerja, transformers memetakan input menjadi 3 representasi yang berbeda: query (Q), key (K), dan value (V). Setiap representasi merupakan hasil perkalian matriks dengan sebuah matriks bobot W (Wq, Wk, dan Wv). Hasil dari perkalian matriks tersebut kemudian dioperasikan melalui operasi attention yang dapat dilihat pada persamaan 2.

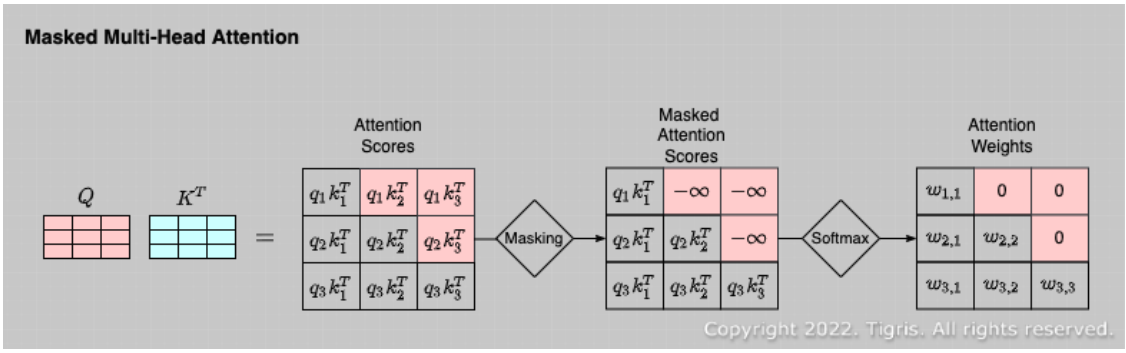
$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \quad (2)$$

   Pada multi-head attention, terdapat beberapa “head” yang masing-masing memiliki matriks bobotnya sendiri, serta operasi attention terjadi pada masing-masing

head secara paralel. Kemudian akan dilakukan agregasi berupa operasi konkatenasi pada head-head tersebut. Rinciannya dapat dilihat pada persamaan 3.

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_\text{h})W^O \tag{3}$$

Kemudian ada juga mekanisme attention lain yaitu masked attention (tepatnya masked multi-head attention) yang terjadi. Sama seperti attention pada umumnya, hanya saja pada masked multi-head attention, terjadi masking dengan mengganti hasil perkalian antara Q dan K dengan nilai dengan -∞. Pergantian nilai tersebut dilakukan untuk mencegah informasi sekuens yang ada di kanan, tidak mempengaruhi bagian sekuens di sebelah kirinya. Ilustrasi masked attention dapat dilihat pada gambar 5.



Gambar 5. Ilustrasi Masked Attention (sumber : https://tigris-data-science.tistory.com/entry/%EC%B0%A8%EA%B7%BC%EC%B0%A8%EA%B7%BC-%EC%9D%B4%ED%95%B4%ED%95%98%EB%8A%94-Transformer4-Masked-Multi-Head-Attention%EA%B3%BC-Decoder)

2. Positional Encoding

Dikarenakan tidak adanya proses rekurens seperti model berbasis RNN, diperlukan informasi posisi dari tiap unit sekuens. Diajukan sebuah encoding posisi dengan persamaan sebagai berikut:

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}}) \tag{4}$$

$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}}) \quad (5)$$

Pada persamaan 4 dan 5, masing-masing mendeskripsikan persamaan untuk menghitung encoding posisi pada posisi unit sekuens genap dan ganjil secara berurutan. Variabel i merupakan nilai indeks unit sekuens sedangkan dmodel merupakan ukuran hidden dimension pada model.

3. Position-wise Feed Forward Network

   Setelah melakukan mekanisme attention, terdapat mekanisme poistion-wise feed forward network. Yang dilakukan di dalamnya adalah dengan menggunakan lapisan convolution 1 dimensi dengan ukuran kernel 1, ukuran stride 1, dan nilai padding 0. Sehingga, untuk setiap unit sekuens, secara terpisah akan diproses dengan sebuah kernel feed forward network. Untuk rinciannya dapat dilihat pada persamaan 6.

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2 \quad (6)$$

Setelah terbit literatur “Attention is All You Need”, semakin banyak language model yang didasari pada konsep transformers. Muncul berbagai model seperti GPT (Radford dkk., 2018) yang terdiri dari blok decoder saja, BERT (Devlin dkk., 2018) yang terdiri dari blok encoder saja, dan T5 (Raffel dkk., 2020) yang merupakan model transformers encoder-decoder. Model-model tersebut mengadaptasi arsitektur transformers namun dengan beberapa modifikasi serta dilakukannya proses pre-training. Hal ini didasari dari kelebihan arsitektur transformers dibandingkan arsitektur berbasis RNN. Namun, terdapat juga beberapa kekurangan dari arsitektur transformers.

1. Kelebihan
   a. Paralelisasi

      Dikarenakan tidak adanya mekanisme rekurens, model dapat dilatih secara paralel sehingga waktu pelatihan lebih singkat.

   b. Skalabilitas

Dikarenakan waktu latih yang lebih singkat, ukuran model serta jumlah data latih dapat ditingkatkan untuk memperoleh kemampuan generalisasi yang lebih baik bagi model.

c. Tidak Ada Permasalahan Long-term Memory

Mekanisme attention memiliki operasi softmax yang bekerja sebagai selektor bagi informasi-informasi yang penting untuk diolah. Selain itu,

2. Kekurangan

a. Panjang Sekuens

Ukuran sekuens pada transformers menjadi terbatas pada kapabilitas hardware yang digunakan pada komputasi pelatihan. Hal ini disebabkan pada ukuran panjang sekuens berpengaruh pada besarnya hasil matriks pada mekanisme attention.

b. Haus Data

Kompleksitas arsitektur transformers memerlukan ukuran model dan ukuran data yang besar untuk mencapai generalisasi yang lebih baik.

DATASET

Pada eksperimen yang dilakukan, terdapat dua jenis perbandingan, yaitu perbandingan **arsitektur** serta perbandingan **panjang sekuens**.

Arsitektur

Pada babak perbandingan **arsitektur**, akan digunakan 3 dataset yang berasal dari kaggle (https://www.kaggle.com/datasets/paultimothymooney/poetry), yaitu dataset lirik lagu dari artis-artis ternama.

```
I walk these streets famished
You see, that was my wife who you decided to What She courted
me
And I'm drownin' in 'em
While you're asking for his blessin'
Wait
I hope that you burn
```

Gambar 6. Cuplikan Dataset Lirik Lagu

Ketiga dataset tersebut memiliki jumlah data yang bervariasi dari yang kecil - sedang - besar (terdapat banyak artis pada dataset utama, pengambilan kecil, sedang, dan besar didasari pada persentil jumlah data). Setiap dataset dibagi menjadi data latih dan data uji (data uji digunakan juga pada proses validasi) dengan rasio 4 : 1. Berikut adalah rincian dataset:

Tabel 3. Rincian Dataset Lirik Lagu

Ukuran Dataset	Nama Artis	Jumlah Data Latih	Jumlah Data Uji
Kecil	lin-manuel-miranda	437	110
Sedang	patti-smith	1744	436
Besar	prince	5868	1468

Panjang Sekuens

Pada babak perbandingan **panjang sekuens**, akan digunakan 1 dataset yang berasal dari kaggle (https://www.kaggle.com/datasets/rajneesh231/lex-fridman-podcast-transcript), yaitu dataset transkrip podcast milik Lex Fridman. Pada tahapan preprocessing, dataset akan dibagi menjadi baris-baris teks yang setiap barisnya memiliki panjang sekuens sebesar 512 karakter (yang termasuk karakter alfabet dan spasi saja, angka dan karakter lain tidak dihitung).

```
1  Traditional politics is a dead end. I don't see what I can…
2  w up before the internet. Maybe there is joy and deep…
3  gs. But I do it anyway. So the reason I bring that up is…
                                    …
```

Gambar 7. Cuplikan Dataset Podcast

Pemilihan 1024 karakter dikarenakan model yang akan dibandingkan akan dilatih pada level karakter, sehingga 1 karakter sama dengan 1 token. Angka 1024 digunakan sebagai panjang sekuens maksimum, pada teknis pelatihannya, akan dilakukan preprocessing tambahan dengan melakukan truncation yang bervariatif, yaitu 128, 256, 512, dan 1024. Untuk jumlah data yang digunakan pada data latih yaitu sebanyak 1600 baris dan data uji sebanyak 400 baris (rasio 4 : 1).

Preprocessing Tambahan

Teknis yang dilakukan pada saat proses pelatihan melibatkan beberapa tahapan preprocessing sebagai berikut:

1. **Case Folding** : Dilakukan dengan mengubah seluruh dataset menjadi huruf kecil.
2. **Remove Non Alphabet Character** : Karakter selain alfabet dan karakter spasi akan dihilangkan.
3. **Padding & EOS** : Penambahan token padding dan token end of sentence
4. **Truncation** : Pemotongan teks yang melewati panjang sekuens maksimal
5. **Tokenization** : Merubah data menjadi token-tokennya (level karakter)

METODE

Untuk membuktikan kebenaran klaim arsitektur transformers, dilakukan perbandingan model berbasis transformers dengan model berbasis LSTM sebagai model sekuensial yang paling banyak digunakan sebelum kehadiran model berbasis transformers. Persoalan yang digunakan sebagai materi perbandingan adalah persoalan causal language modeling pada level karakter. Untuk model berbasis transformers akan digunakan arsitektur decoder-only yang paling sederhana, yaitu GPT.

Akan dilakukan pelatihan beberapa kali untuk setiap model dengan kombinasi hyperparameter yang beragam terhadap dataset yang sama. Terdapat dua babak perbandingan yang sudah sempat disebut pada bab dataset, yaitu **arsitektur** dan **panjang sekuens**.

Arsitektur

Pada babak perbandingan ini, setiap model akan dilatih terhadap 3 dataset lirik lagu dengan ukuran kecil, sedang, dan besar. Dilakukan percobaan pelatihan terhadap setiap dataset dengan kombinasi hyperparameter dari setiap model yang telah ditentukan.

Tabel 4. Rincian Hyperparameter

Hyperparameter LSTM	**Hyperparameter GPT**
d_model (dimensi model) : [128, 256, 512, 1024]	d_model (dimensi model) : [128, 256, 512, 1024]
n_layer (jumlah layer) : [1, 2, 3, 4]	ff_dim (dimensi positional FFNN) : [128, 256, 512, 1024]
bidirectional : [true, false]	n_head (jumlah multi-head) : [1, 2, 3, 4]
	n_block (jumlah attention block) : [1, 2, 3, 4]
32 kombinasi / dataset	**256 kombinasi / dataset**

Selain kombinasi dari hyperparameter tersebut, terdapat hyperparameter serta setting lain yang bernilai tetap:

1. Seq_len (panjang sekuens) : 64 token (karakter)
2. gpu (gpu id) : 0
3. batch (ukuran batch) : 8
4. lr (learning rate) : 3e-4 (optimizer Adam)
5. epoch : 10

Pada pelatihan babak ini, akan ada beberapa nilai yang akan menjadi perhatian, yang kemudian akan menjadi bahan analisis:

1. Jumlah parameter
2. Kedalaman model
3. Lebar model
4. Training loss (Cross Entropy)
5. Validation loss (Cross Entropy)
6. Waktu pelatihan

Perbandingan utama akan dilihat pada jumlah parameter vs training loss vs waktu pelatihan. Kedalaman dan lebar model menjadi analisis tambahan terkait apakah pada masing-masing model, lebih baik untuk memiliki model yang memiliki kedalaman lebih tinggi (deep) atau memiliki model yang lebih lebar. Selanjutnya validation loss digunakan untuk membandingkan antara kedua model, model mana yang cenderung memiliki tingkat overfitting atau underfitting yang lebih tinggi.

Panjang Sekuens

Setelah memperoleh hasil babak perbandingan **arsitektur**, akan digunakan model dengan jumlah parameter dan waktu pelatihan yang mendekati. Dari kedua model tersebut, akan dilakukan pelatihan terhadap dataset transkrip podcast dengan variasi panjang sekuens yaitu 128, 256, 512, dan 1024. Kemudian akan diperhatikan apakah terdapat pengaruh terhadap waktu pelatihan masing-masing model.

EKSPERIMEN

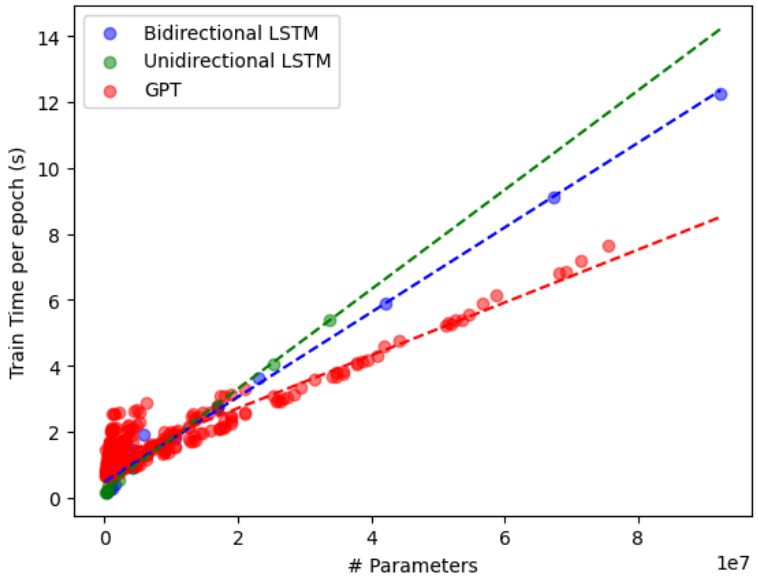
Lingkungan Eksperimen

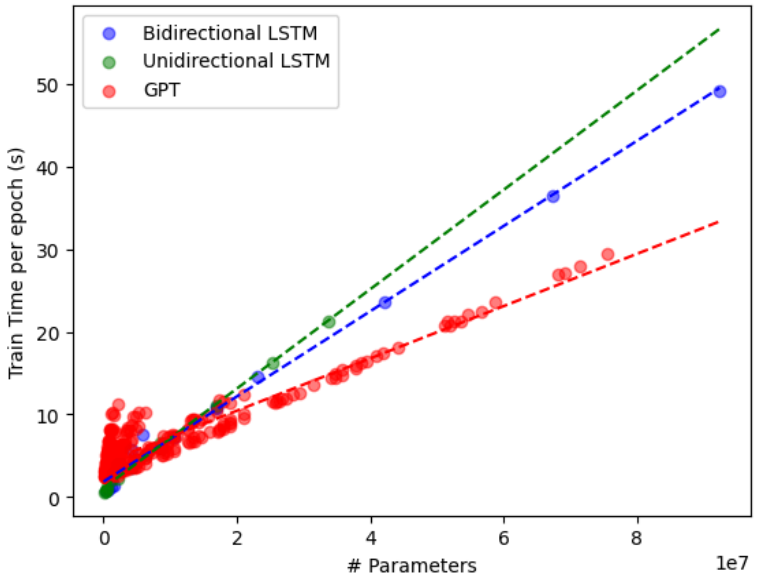
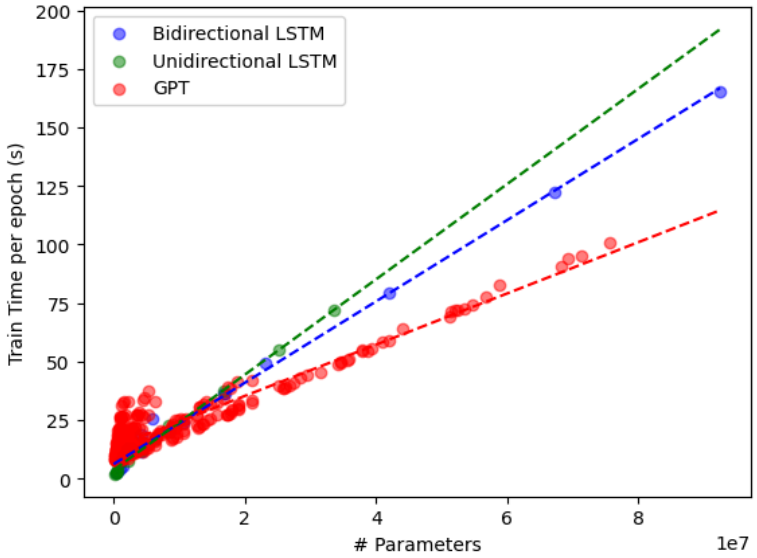
Eksperimen dilakukan menggunakan kakas Pytorch 2.0.1 dengan versi CUDA 11.7. Alat komputasi yang digunakan berupa 1 GPU NVIDIA GTX 1660 6144 MB. Kedua model ditulis menggunakan kakas Pytorch (bukan pretraining atau dengan menggunakan kakas lain seperti hugging face).

Hasil Eksperimen

Arsitektur

Jumlah Parameter vs Waktu Pelatihan

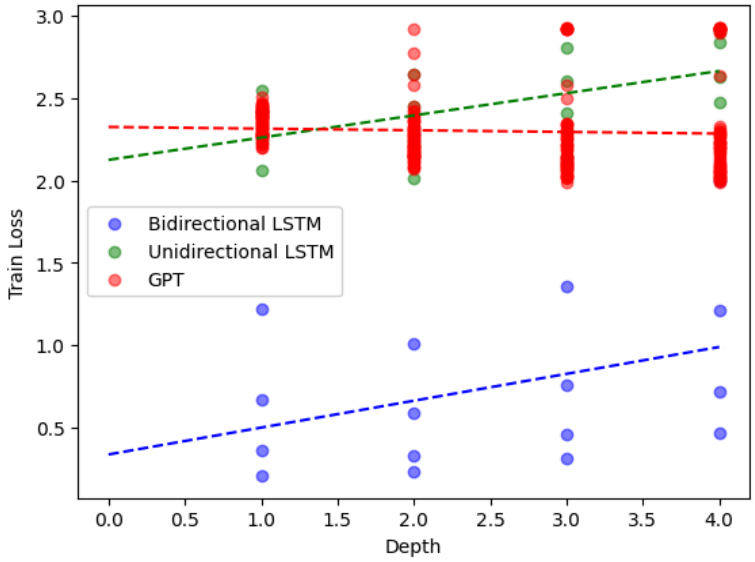
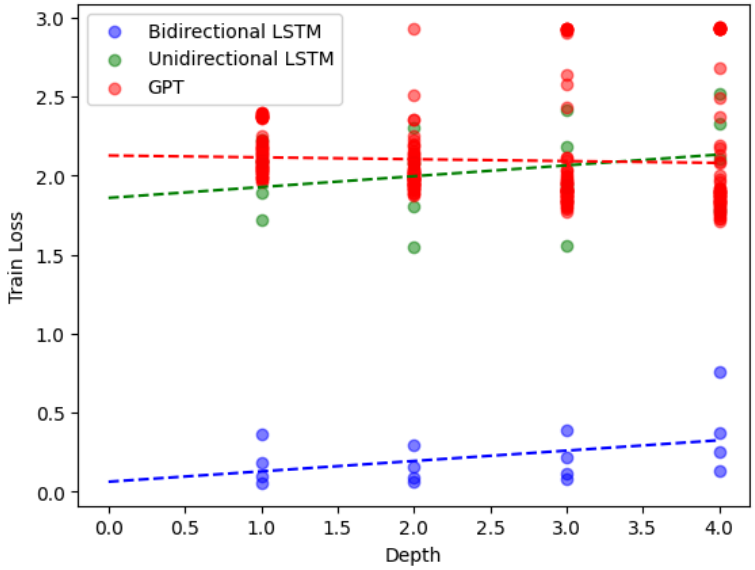
Dataset Kecil	

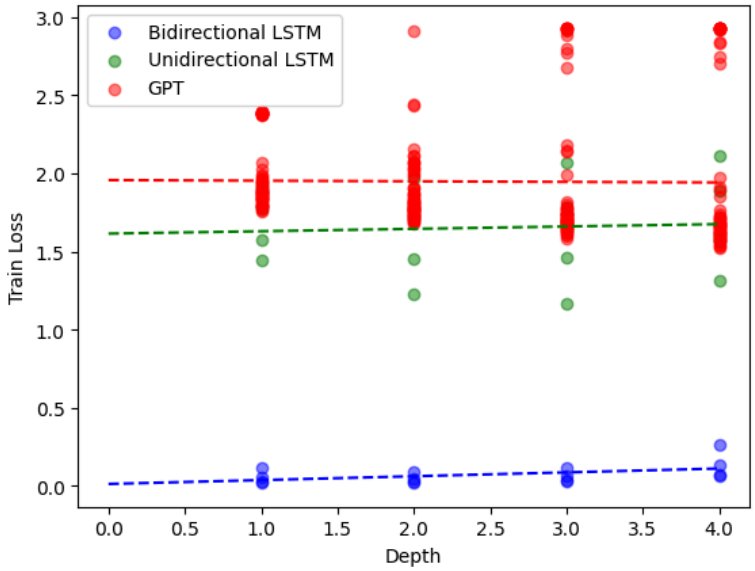
Dataset Sedang	
Dataset Besar	

Gambar 8. Grafik Jumlah Parameter vs Waktu Pelatihan

Dapat terlihat bahwa untuk ketiga ukuran dataset, waktu pelatihan tiap epoch akan naik seiring dengan naiknya jumlah parameter. Terbukti klaim transformers bahwa paralelisasi dapat dimaksimalkan untuk pelatihan, terlihat dari kenaikan waktu pelatihan yang tidak securam LSTM (unidirectional maupun bidirectional).

Depth vs Train Loss

Dataset Kecil	
Dataset Sedang	

Dataset Besar	

Gambar 9. Grafik Kedalaman vs Loss

Dapat terlihat bahwa untuk arsitektur LSTM, semakin dalam kedalaman modelnya, maka nilai loss nya juga akan semakin besar. Sedangkan, untuk GPT, dapat terlihat sedikit bahwa dengan semakin dalam model, maka semakin kecil nilai loss nya. Nilai loss yang relatif besar pada model GPT dapat terjadi dikarenakan berbagai faktor berdasarkan rancangan eksperimen yang dilakukan:

1. Ukuran Dataset

   Sesuai bab sebelumnya, model berbasis transformers dipercaya memiliki sifat yang haus data, sehingga untuk mencapai kinerja yang baik, diperlukan volume data dalam jumlah besar.

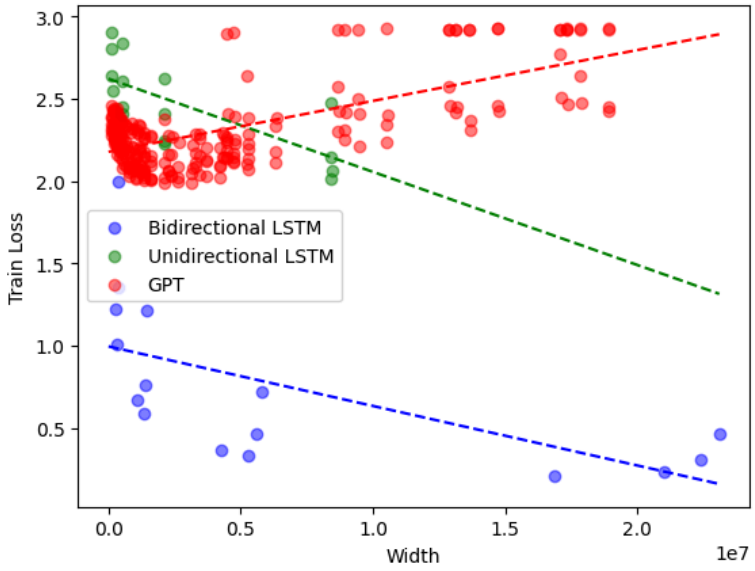
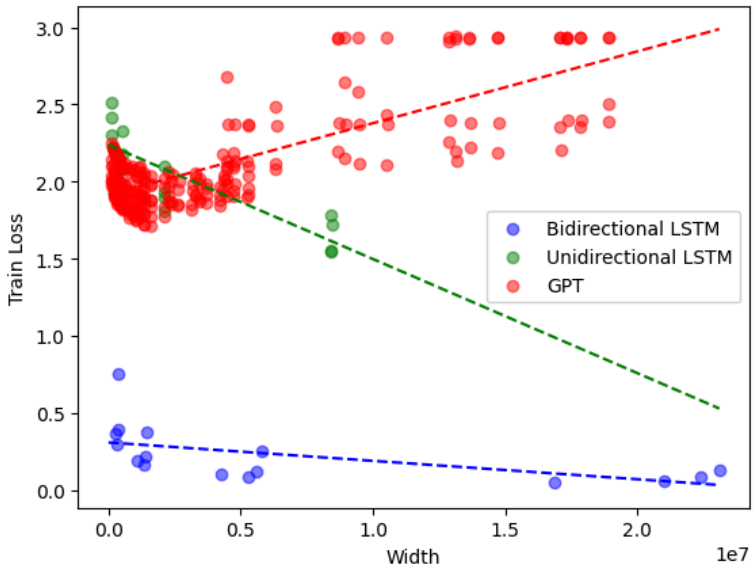
2. Kualitas Dataset

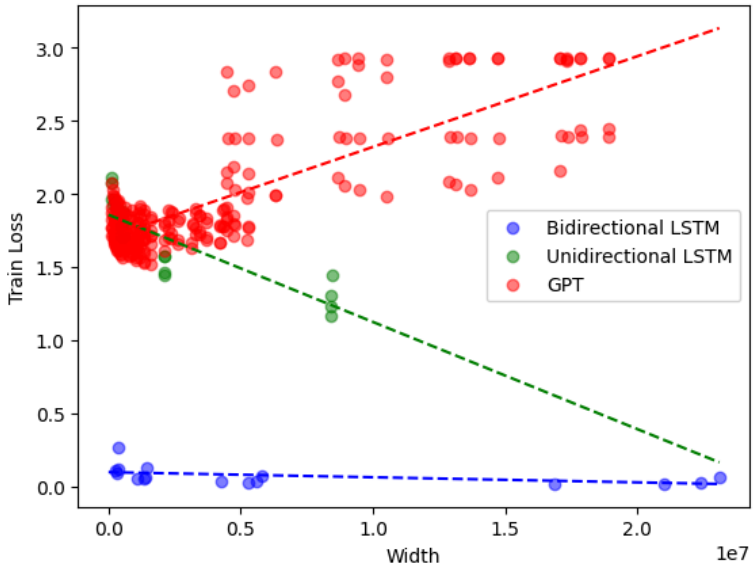
   Kualitas dataset yang ada bisa saja belum memenuhi standar yang baik serta distribusinya tidak baik.

3. Tokenisasi Level Karakter

   Penggunaan tokenisasi pada level karakter menyebabkan model hanya terbatas pada 29 token saja (26 huruf + spasi + token padding + token EOS). Keterbatasan tersebut menyebabkan model tidak mampu menangkap informasi yang lebih luas.

Width vs Train Loss

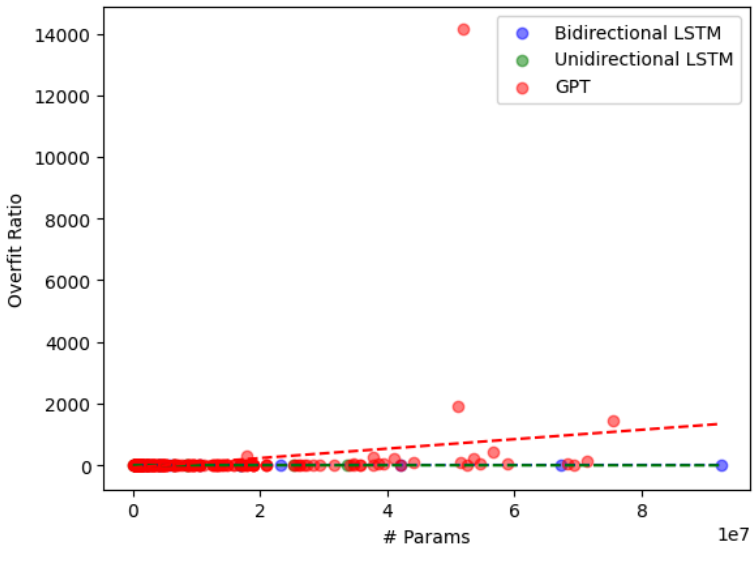
Dataset Kecil	
Dataset Sedang	

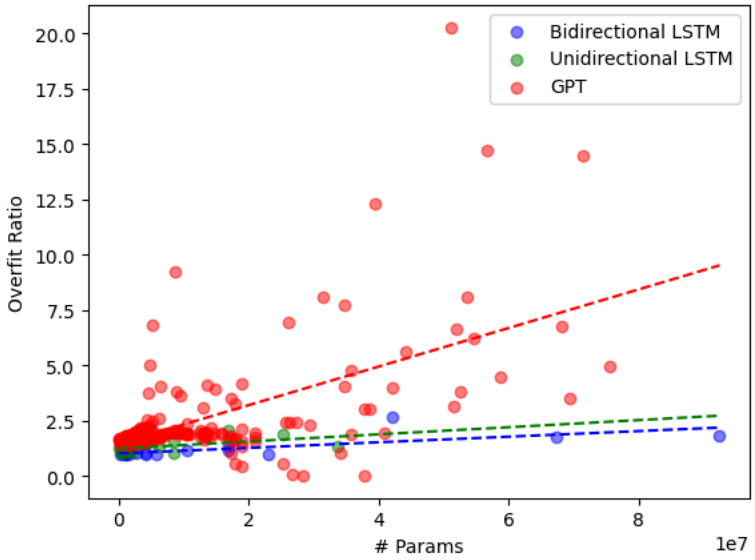
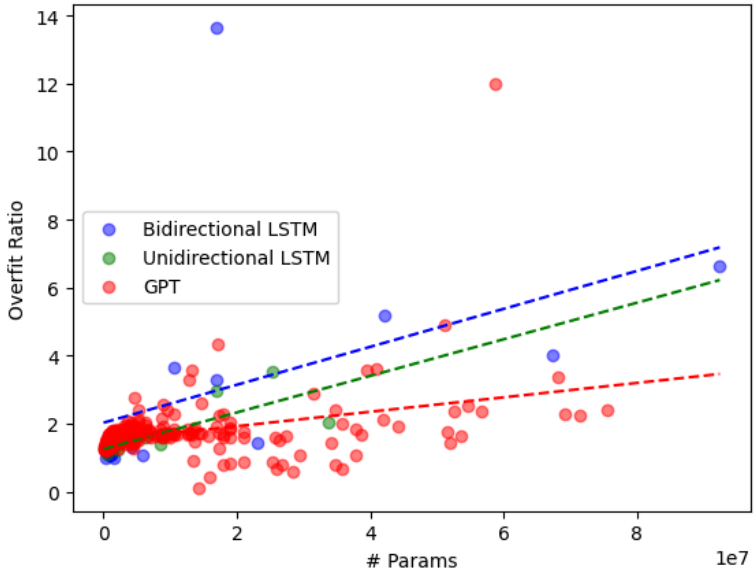
Dataset Besar	

Gambar 10. Grafik Lebar vs Loss

Dapat terlihat bahwa untuk arsitektur LSTM, semakin lebar modelnya, maka nilai loss nya akan semakin kecil. Sedangkan, untuk GPT, dapat terlihat bahwa semakin lebar model justru nilai loss nya akan semakin besar.

Jumlah Parameter vs Overfit Ratio

Dataset Kecil	

Dataset Sedang	
Dataset Besar	

Gambar 11. Grafik Jumlah Parameter vs Overfit Ratio

Overfit ratio merupakan perhitungan perbandingan antara nilai penurunan training loss dan validation loss. Berikut adalah persamaan yang mendeskripsikan overfit ratio.

$$OR = abs(\frac{avg\ train\ loss\ decrement}{avg\ val\ loss\ decrement}) \quad (7)$$

$$loss\ decrement = \frac{1}{(n-1)*max\ loss} \sum_{t=0}^{n-1} (loss_t - loss_{t+1}) \quad (8)$$

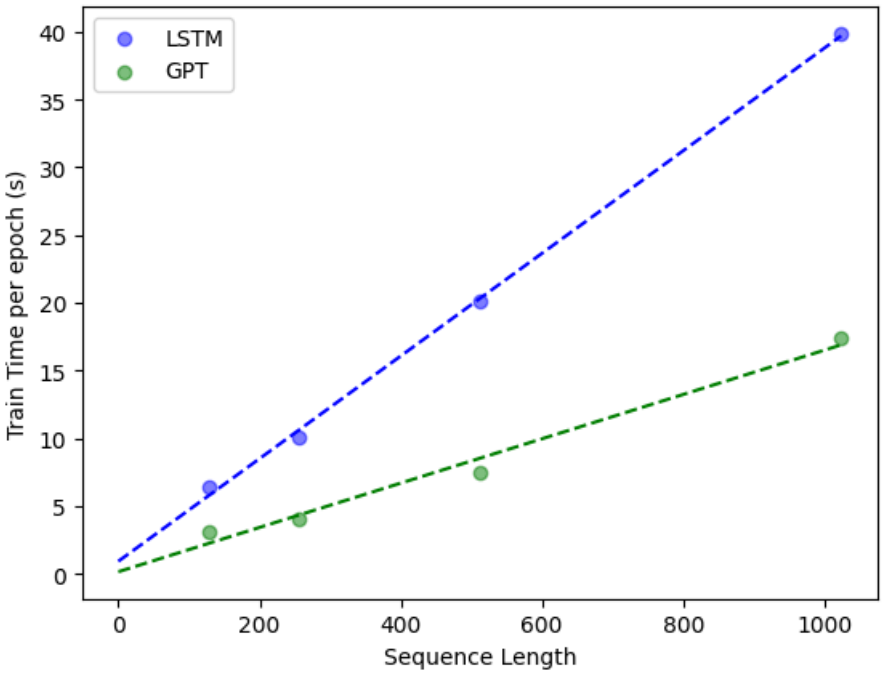
Semakin nilai overfit ratio mendekati 1, maka dapat dikatakan bahwa pelatihan berlangsung dengan baik (nilai avg train loss decrement sejalan dengan avg val loss decrement). Jika nilai overfit ratio lebih besar dari 1, maka dapat dikatakan bahwa model mengalami overfit (nilai avg train loss decrement lebih signifikan dibandingkan avg val loss decrement). Pada kasus overfit ratio mendekati 0, maka dapat dikatakan model mengalami underfit (avg val loss decrement lebih signifikan dibandingkan dengan avg train loss decrement). Dapat terlihat pada semua jenis dataset, bahwa GPT lebih mudah mengalami overfitting dibandingkan model LSTM (bidirectional maupun unidirectional).

Panjang Sekuens

Dari babak perbandingan arsitektur, diperoleh model LSTM dan model GPT yang memiliki jumlah parameter dan rata-rata waktu pelatihan yang mendekati (pada dataset berukuran kecil). Rincian jumlah parameter, waktu pelatihan pada babak perbandingan arsitektur, dan hyperparameter dapat dilihat pada tabel 5.

Tabel 5. Rincian Model Terdekat Pada Babak Perbandingan Arsitektur

	LSTM	**GPT**
Jumlah Parameter	1593885	1606144
Rata-Rata Waktu Pelatihan	0.758449	0.740287
Hyperparameter	d_model (dimensi model) : 256	d_model (dimensi model) : 512
	n_layer (jumlah layer) : 3	ff_dim (dimensi positional FFNN) : 512
	bidirectional : false	n_head (jumlah multi-head) : 1
		n_block (jumlah attention block) : 1



Gambar 12. Grafik Perbandingan Panjang Sekuens (X) vs Waktu Pelatihan (Y)

Dapat dilihat pada gambar 12, bahwa klaim terkait paralelisasi yang ada pada transformers memang betul. Kenaikan yang ada pada kasus transformers di kasus ini bisa saja berkaitan dengan kapasitas hardware yang dimiliki dan dipengaruhi oleh ukuran operasi attention, bukan berasal dari proses rekurens seperti pada LSTM.

KESIMPULAN

Pada persoalan text generation, dengan membandingkan model berbasis RNN, khususnya LSTM dan model berbasis transformers yaitu GPT, diperoleh bahwa klaim terkait kemampuan model transformers dalam melakukan pelatihan secara paralel adalah benar, dibuktikan dari waktu pelatihan yang naik secara signifikan untuk LSTM pada kasus jumlah parameter yang besar serta pada data sekuens yang panjang, sedangkan model GPT tidak terjadi kenaikan yang signifikan. Adapun kinerja terkait dengan loss, model GPT memerlukan ukuran, kualitas, serta teknik tokenisasi yang lebih besar lagi untuk mencapai kemampuan generalisasi yang lebih baik dibandingkan LSTM, hal tersebut wajar dan baik untuk skalabilitas penggunaan model pada penelitian mendatang.

REFERENSI

Cho, K., Van Merriënboer, B., Bahdanau, D., & Bengio, Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches. arXiv preprint arXiv:1409.1259.

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555.

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.

Jurafsky, D., & Martin, J. H. Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition.

Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding by generative pre-training.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., ... & Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. Journal of machine learning research, 21(140), 1-67.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. nature, 323(6088), 533-536.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. Advances in neural information processing systems, 30.