Q&A Chatbot

Transformer version

Latar Belakang

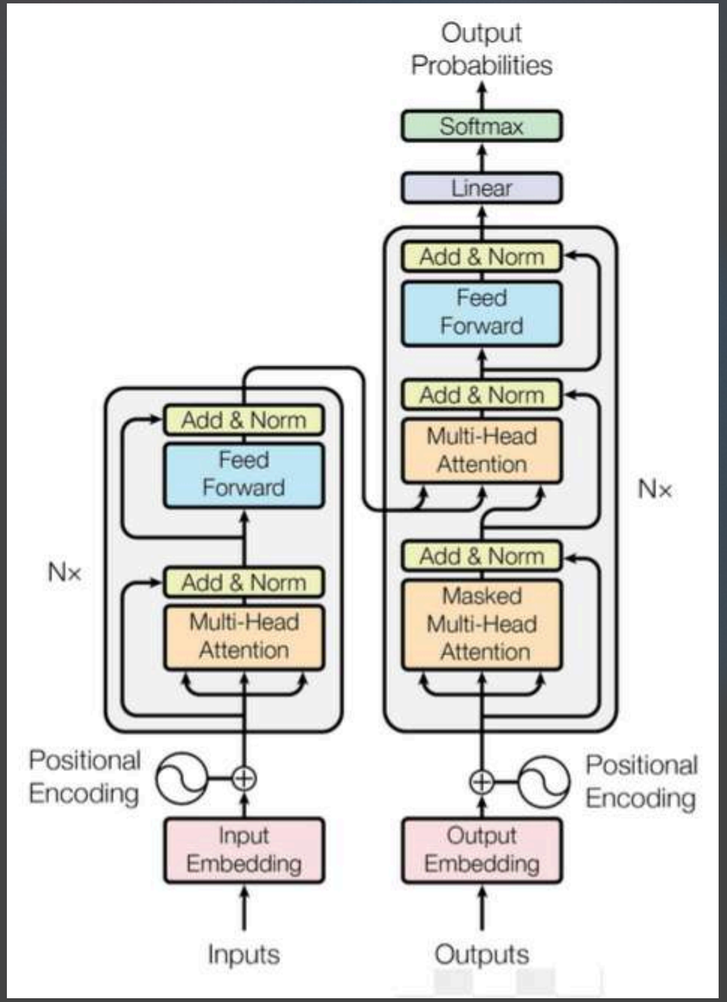
Model transformer yang kami buat dilatih menggunakan dataset yang diambil dari aplikasi Alodokter, dataset ini tersedia di kaggle.com. Dari dataset tersebut, kami mengambil data dari kolom question dan answer. Question mengandung string pertanyaan dari para pengguna aplikasi Alodokter yang dapat berisi keluhan, simptom penyakit yang dialami pengguna, maupun sekedar meminta saran. Sedangkan Answer mengandung jawaban, diagnosis, atau saran untuk Question tersebut yang ditulis oleh seorang dokter terpercaya

Link dataset: https://www.kaggle.com/datasets/gufranakasamudra/indonesia-qna-health-dataset

Deep Learning

Arsitektur Transformer

Model Transformer adalah dasar dari banyak model bahasa modern seperti GPT dan BERT. Arsitektur ini menggunakan mekanisme Self-Attention untuk memahami hubungan antar kata dalam kalimat. Tidak seperti RNN atau LSTM yang memproses data secara berurutan, Transformer memproses seluruh kata secara paralel, sehingga pelatihan lebih cepat dan pemahaman konteks lebih luas.



```python
# Layer Positional Encoding (Sinusoidal)
def positional_encoding(seq_length, d_model):
    position = np.arange(seq_length)[:, np.newaxis]
    div_term = np.exp(np.arange(0, d_model, 2) * -(np.log(10000.0) / d_model))

    pos_encoding = np.zeros((seq_length, d_model))
    pos_encoding[:, 0::2] = np.sin(position * div_term)
    pos_encoding[:, 1::2] = np.cos(position * div_term)

    return tf.cast(pos_encoding[np.newaxis, ...], dtype=tf.float32)
```

Komponen Utama Transformer

Layer Positional Encoding

- Menambahkan informasi urutan ke embedding menggunakan fungsi sinus dan cosinus.
- Tidak dipelajari (deterministik), hanya bergantung pada posisi dan dimensi model.
- Diformat sebagai (1, seq_len, d_model) lalu dijumlahkan ke embedding.

```python
# Layer untuk Word Embedding + Positional Encoding
class PositionalEmbedding(layers.Layer):
    def __init__(self, vocab_size, d_model, max_len=100, **kwargs):
        super().__init__(**kwargs)
        self.d_model = d_model
        self.embedding = layers.Embedding(vocab_size, d_model, mask_zero=True)
        self.pos_encoding = positional_encoding(max_len, d_model)

    def call(self, x):
        seq_len = tf.shape(x)[1]
        x = self.embedding(x)
        # Scaling embedding (sesuai paper Transformer)
        x *= tf.math.sqrt(tf.cast(self.d_model, tf.float32))
        # Tambahkan Positional Encoding
        x += self.pos_encoding[:, :seq_len, :]
        return x
```

Komponen Utama Transformer

Layer Positional Embedding

- Menggabungkan word embedding dan positional encoding.
- Embedding diskalakan dengan $\sqrt{d\_model}$ sebelum ditambahkan ke pos encoding untuk stabilitas numerik.
- Output siap untuk dimasukkan ke layer Transformer, dengan mask_zero=True agar padding otomatis ditangani.

```python
# Feed-Forward Network sederhana (digunakan di Encoder dan Decoder)
def feed_forward_network(d_model, dff):
    return keras.Sequential([
        layers.Dense(dff, activation='relu'),
        layers.Dense(d_model)
    ])
```

Komponen Utama Transformer

Feed-Forward Network (FFN)

- Dua layer Dense: Dense(dff, relu) → Dense(d_model).
- Diterapkan secara independen di tiap posisi (position-wise).
- Menambah non-linearitas dan kapasitas model.

Komponen Utama Transformer

Transformer Encoder Layer

- Komponen utama:
  a. Multi-Head Self-Attention
  b. Residual Connection + LayerNorm
  c. Feed Forward + Residual + LayerNorm + Dropout
- Menghasilkan representasi konteks dari input (encoder output).

```python
# Custom Layer: Transformer ENCODER Block
class TransformerEncoderLayer(layers.Layer):
    def __init__(self, d_model, num_heads, dff, dropout_rate=0.1, **kwargs):
        super().__init__(**kwargs)
        self.mha = layers.MultiHeadAttention(num_heads=num_heads, key_dim=d_model)
        self.ffn = feed_forward_network(d_model, dff)

        self.layernorm1 = layers.LayerNormalization(epsilon=1e-6)
        self.layernorm2 = layers.LayerNormalization(epsilon=1e-6)

        self.dropout1 = layers.Dropout(dropout_rate)
        self.dropout2 = layers.Dropout(dropout_rate)

    def call(self, x, training, mask=None):
        # 1. Multi-Head Self-Attention (Residual Connection + Layer Norm)
        attn_output = self.mha(x, x, attention_mask=mask)
        attn_output = self.dropout1(attn_output, training=training)
        out1 = self.layernorm1(x + attn_output)  # Add & Norm

        # 2. Feed Forward (Residual Connection + Layer Norm)
        ffn_output = self.ffn(out1)
        ffn_output = self.dropout2(ffn_output, training=training)
        out2 = self.layernorm2(out1 + ffn_output) # Add & Norm

        return out2
```

Komponen Utama Transformer

Transformer Decoder Layer

- Urutan proses:
  a. Masked Self-Attention (mencegah melihat token masa depan)
  b. Encoder-Decoder (Cross) Attention
  c. Feed Forward + Residual + LayerNorm
- Output diproyeksikan ke vocab logits untuk prediksi token.

```python
class TransformerDecoderLayer(layers.Layer):
    def __init__(self, d_model, num_heads, dff, dropout_rate=0.1, **kwargs):
        super().__init__(**kwargs)
        # Masked Self-Attention (Attention 1)
        self.mha1 = layers.MultiHeadAttention(num_heads=num_heads, key_dim=d_model)
        # Encoder-Decoder Attention (Attention 2)
        self.mha2 = layers.MultiHeadAttention(num_heads=num_heads, key_dim=d_model)

        self.ffn = feed_forward_network(d_model, dff)

        self.layernorm1 = layers.LayerNormalization(epsilon=1e-6)
        self.layernorm2 = layers.LayerNormalization(epsilon=1e-6)
        self.layernorm3 = layers.LayerNormalization(epsilon=1e-6)

        self.dropout1 = layers.Dropout(dropout_rate)
        self.dropout2 = layers.Dropout(dropout_rate)
        self.dropout3 = layers.Dropout(dropout_rate)

    def call(self, x, enc_output, training, look_ahead_mask=None, padding_mask=None):
        # 1. Masked Multi-Head Self-Attention (Input Decoder, dengan Causal Mask)
        # Note: Keras MHA mendukung `use_causal_mask=True` untuk look-ahead mask
        attn1 = self.mha1(query=x, value=x, key=x, use_causal_mask=True, attention_mask=look_ahead_mask)
        attn1 = self.dropout1(attn1, training=training)
        out1 = self.layernorm1(attn1 + x) # Add & Norm

        # 2. Encoder-Decoder Attention (Input Decoder Q, Encoder K/V)
        # Encoder output (enc_output) digunakan sebagai Key dan Value
        attn2 = self.mha2(query=out1, value=enc_output, key=enc_output, attention_mask=padding_mask)
        attn2 = self.dropout2(attn2, training=training)
        out2 = self.layernorm2(attn2 + out1) # Add & Norm

        # 3. Feed Forward (Residual Connection + Layer Norm)
        ffn_output = self.ffn(out2)
        ffn_output = self.dropout3(ffn_output, training=training)
        out3 = self.layernorm3(ffn_output + out2) # Add & Norm
```

Komponen Utama Transformer

Masking

- create_padding_mask: tandai token 0 (padding).
- create_look_ahead_mask: cegah akses ke token masa depan.
- create_masks: gabungkan semua mask untuk encoder dan decoder.
- Tujuan: memastikan attention hanya fokus pada token relevan.

```
# ============================================================
# 3. FUNGSI MASKING (WAJIB UNTUK TRAINING TRANSFORMER)
# ============================================================

# Fungsi untuk membuat padding mask
def create_padding_mask(seq):
    # Padding token di-set ke 0 (default Keras)
    mask = tf.cast(tf.math.equal(seq, 0), tf.float32)
    # Ubah mask menjadi bentuk yang bisa di-broadcast ke attentio
    return mask[:, tf.newaxis, tf.newaxis, :] # (batch_size, 1, 1

# Fungsi untuk membuat look ahead mask (Causal Mask) - TIDAK DIPE
# Namun, jika padding mask digabungkan, ini tetap diperlukan.
def create_look_ahead_mask(size):
    mask = 1 - tf.linalg.band_part(tf.ones((size, size)), -1, 0)
    return mask # (seq_len, seq_len)

# Menggabungkan padding mask dan look ahead mask untuk decoder
def create_masks(inp, tar):
    # Encoder padding mask (untuk Encoder MHA)
    enc_padding_mask = create_padding_mask(inp)

    # Decoder padding mask (untuk Decoder MHA 2/Cross-Attention)
    dec_padding_mask = create_padding_mask(inp)

    # Look ahead mask (untuk Decoder MHA 1/Self-Attention)
    look_ahead_mask = create_look_ahead_mask(tf.shape(tar)[1])

    # Gabungkan look ahead mask dan target padding mask
    tar_padding_mask = create_padding_mask(tar)
    combined_mask = tf.maximum(tar_padding_mask, look_ahead_mask)

    return enc_padding_mask, combined_mask, dec_padding_mask
```

Q&A Chatbot

Training Setup

- D_MODEL: Dimensi vektor yang merepresentasikan setiap token
- NUM_HEADS: Multi-head attention dibagi menjadi 8 heads yang bekerja paralel
- DFF: Dimensi hidden layer pada feed-forward network
- NUM_LAYERS: Jumlah blok encoder dan decoder yang ditumpuk
- DROPOUT_RATE: 30% neuron akan dimatikan secara random saat training untuk mencegah overfitting

```
# Hyperparameters model
D_MODEL = 256      # Dimensi embedding dan model
NUM_HEADS = 8
DFF = 512          # Dimensi Feed Forward
NUM_LAYERS = 4     # Jumlah layer Encoder/Decoder
DROPOUT_RATE = 0.3

# Bangun Model Transformer
transformer_model = build_transformer_model(
    input_vocab_size=NUM_ENCODER_TOKENS,
    target_vocab_size=NUM_DECODER_TOKENS,
    max_len=MAX_SEQ_LEN,
    d_model=D_MODEL,
    num_heads=NUM_HEADS,
    dff=DFF,
    num_layers=NUM_LAYERS,
    dropout_rate=DROPOUT_RATE
)
```

Q&A Chatbot

Training Setup

- Loss function menggunakan SparseCategoricalCrossentropy dengan masking untuk mengabaikan padding
- Accuracy juga menggunakan masking untuk hanya menghitung akurasi pada token yang valid
- Model dikompilasi menggunakan optimizer Adam dengan learning rate kecil (0.0001)
- Menggunakan custom loss function dan metric yang telah didefinisikan

```python
# Loss dan Optimizer (Menggunakan SparseCategoricalCrossentropy)
loss_object = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False, reduction='none')

def custom_loss_function(real, pred):
    # Masking loss: Abaikan token 0 (padding) saat menghitung loss
    loss = loss_object(real, pred)
    mask = tf.cast(tf.math.not_equal(real, 0), tf.float32)
    loss = tf.multiply(loss, mask)

    return tf.reduce_sum(loss) / tf.reduce_sum(mask)

# Custom Accuracy (Mengabaikan Padding)
def masked_accuracy(real, pred):
    accuracies = tf.equal(tf.cast(real, tf.int64), tf.argmax(pred, axis=2))

    mask = tf.math.logical_not(tf.math.equal(real, 0))
    accuracies = tf.math.logical_and(mask, accuracies)

    accuracies = tf.cast(accuracies, dtype=tf.float32)
    mask = tf.cast(mask, dtype=tf.float32)

    return tf.reduce_sum(accuracies) / tf.reduce_sum(mask)

# Compile model
transformer_model.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=1e-4),
    loss=custom_loss_function,
    metrics=[masked_accuracy]
)
```

Q&A Chatbot

Training Setup

- Early stopping untuk menghentikan training jika tidak ada peningkatan selama 5 epoch
- Batch size 128 untuk efisiensi memori dan komputasi
- Maximum 75 epochs dengan validasi
- Data training terdiri dari encoder input dan decoder input, dengan decoder target sebagai output yang diharapkan

```
# ============================================================
# Tambahkan EarlyStopping
# ============================================================

early_stop = EarlyStopping(
    monitor='val_loss',      # pantau validation loss
    patience=5,              # hentikan setelah 5 epoch tan
    restore_best_weights=True,
    verbose=1
)

# Training
BATCH_SIZE = 128
EPOCHS = 75

print(f"\n--- Memulai Training selama {EPOCHS} Epochs ---")
history = transformer_model.fit(
    [enc_train, dec_in_train],
    dec_tar_train,
    validation_data=([enc_val, dec_in_val], dec_tar_val),
    batch_size=BATCH_SIZE,
    epochs=EPOCHS,
    callbacks=[early_stop]
)
```

Komponen Utama Transformer

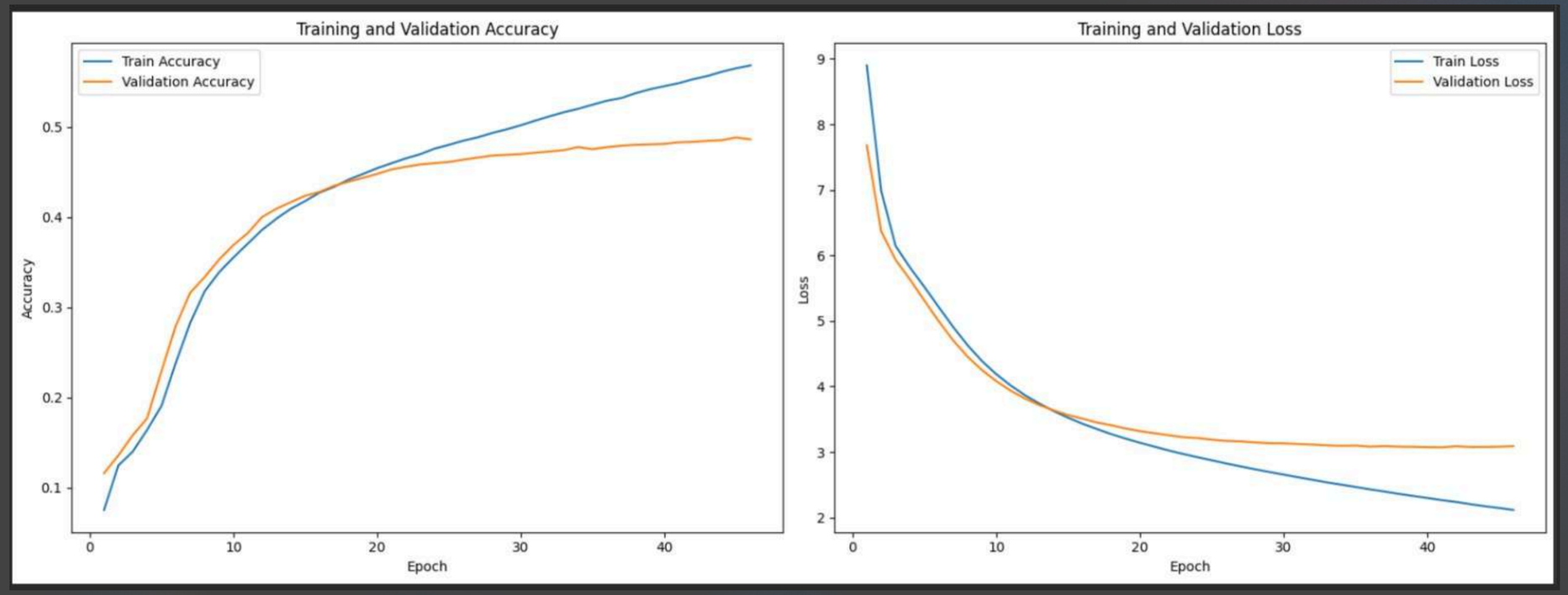
Inference dengan Beam Search

- Inference digunakan untuk menghasilkan teks (decoding) dari model Transformer.
- Beam search tidak langsung memilih kata terbaik satu per satu (seperti greedy decoding), tapi menyimpan beberapa kemungkinan jalur terbaik (beam) agar hasil akhirnya lebih baik dan lebih alami.

```
GENERIC_TOKEN_START_INDEX = 5
GENERIC_TOKEN_END_INDEX = 25
GENERIC_PENALTY_FACTOR = 0.5 # Penalti 50% untuk token generik ini
# --- END CONFIG ---

class Beam:
    """Class untuk menyimpan jalur decoding (untuk Beam Search)."""
    def __init__(self, sequence, score, state=None):
        self.sequence = sequence
        self.score = score
        self.state = state # Tidak digunakan dalam Transformer, tapi disimpan untuk konsistensi

    def __lt__(self, other):
        # Untuk heapq: kita ingin heapq mengurutkan berdasarkan skor tertinggi.
        # Karena kita menggunakan nlargest, kita bisa menggunakan skor positif.
        return self.score < other.score # Untuk heapq.nlargest

def decode_sequence_beam_search(input_seq, model=transformer_model, max_len=MAX_SEQ_LEN, beam_size=3):

    # 1. Inisialisasi Beam
    # Beam dimulai dengan token <start>
    initial_seq = [START_TOKEN]
    initial_score = 0.0 # Log probability

    # Priority queue untuk menyimpan beam
    beams = [Beam(initial_seq, initial_score)]

    final_beams = []

    # 2. Iterasi Autoregressive
    for i in range(1, max_len):
        candidates = []

        # Jika semua beam yang tersisa sudah berakhir, kita hentikan
        if not beams:
```

Demonstrasi

Hasil Train Model




```
============================================================
DEMO DECODING (Beam Search K=3)
============================================================

Q: kaki kram saat tidur kenapa?
A: hal yang bisa membuat anda alergi kompres hangat mata yang nyeri atau lebih rajin minum air putih makan makanan yang mengandung kaya vitamin c vitamin b kompleks vitamin

Q: cara menghilangkan kurap?
A: pada orang yang anda rasakan maka perlunya tindakan medis yang tepat untuk menjaga kebersihan organ intim dengan kompres hangat pada bagian kepala yang sakit selama 15 20 menit

Q: sakit perut
A: pada dokter yang dapat memberikan ibu hamil yang sesuai selain itu sebaiknya melakukan kontrol ke dokter yang merawat untuk melakukan pemeriksaan dokter kandungan sesuai anjuran dokter yang dapat
```

Evaluasi Metrik BLEU & ROUGE-L

ROUGE-L

> 0.5 = Baik

> 0.3 = Cukup

< 0.3 = Perlu diperbaiki

```
...
    ============================================================
    EVALUASI METRIK KINERJA MODEL
    ============================================================

    --- DEMO EVALUASI INDIVIDUAL ---

    ============================================================
    Sample 1:
    ============================================================
    Reference (Ground Truth):
    anda bisa melakukan treatment di klinik kecantikan yang memiliki treatment untuk mengatasi keluhan mata panda semoga dapat membantu...

    Hypothesis (Model Output):
    hingga mencapai ke depan keluhan anda masih belum membaik juga cobalah diperiksakan langsung ke ya semoga membantu ya...

    ROUGE-L Scores:
      - Precision: 0.1667
      - Recall:    0.1667
      - F1-Score:  0.1667

    BLEU Score: 0.0153

    ============================================================
    Sample 2:
    ============================================================
    Reference (Ground Truth):
    bercangkang minuman yang mengandung pemanis fruktosa buah kalengan jauhi alkohol dan perbanyak minum air putih semoga membantu ya...

    Hypothesis (Model Output):
    dan sebagainya jika langkah di atas belum bisa mengatasi keluhan anda cobalah diperiksakan langsung ke agar diberi penanganan...

    ROUGE-L Scores:
      - Precision: 0.0556
      - Recall:    0.0556
      - F1-Score:  0.0556

    BLEU Score: 0.0108

    ============================================================
    Sample 3:
```

BLEU

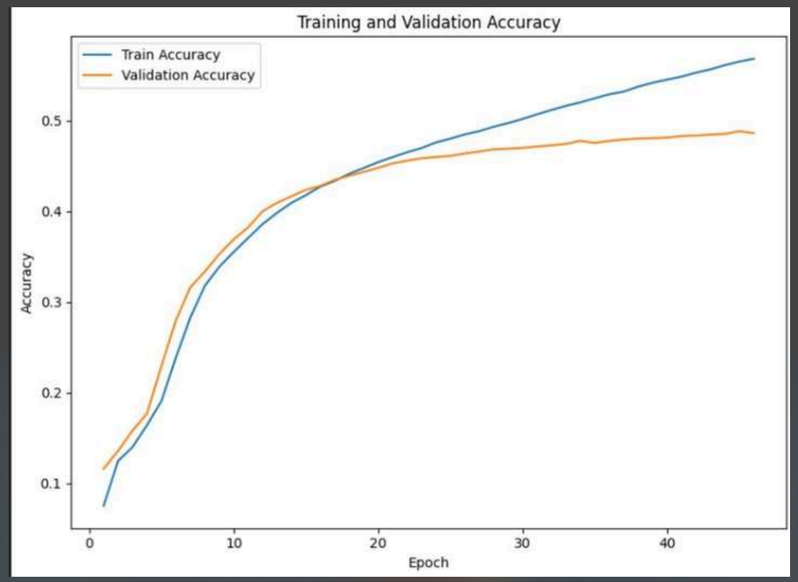
> 0.4 = Baik

> 0.2 = Cukup

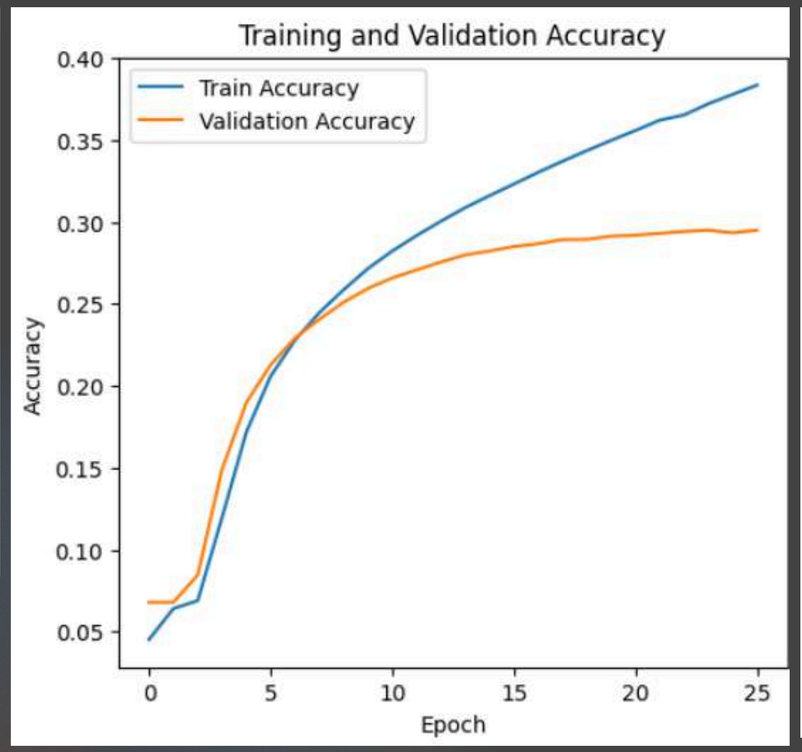
< 0.2 = Perlu diperbaiki

Perbandingan Akurasi Transformer dengan RNN & LSTM

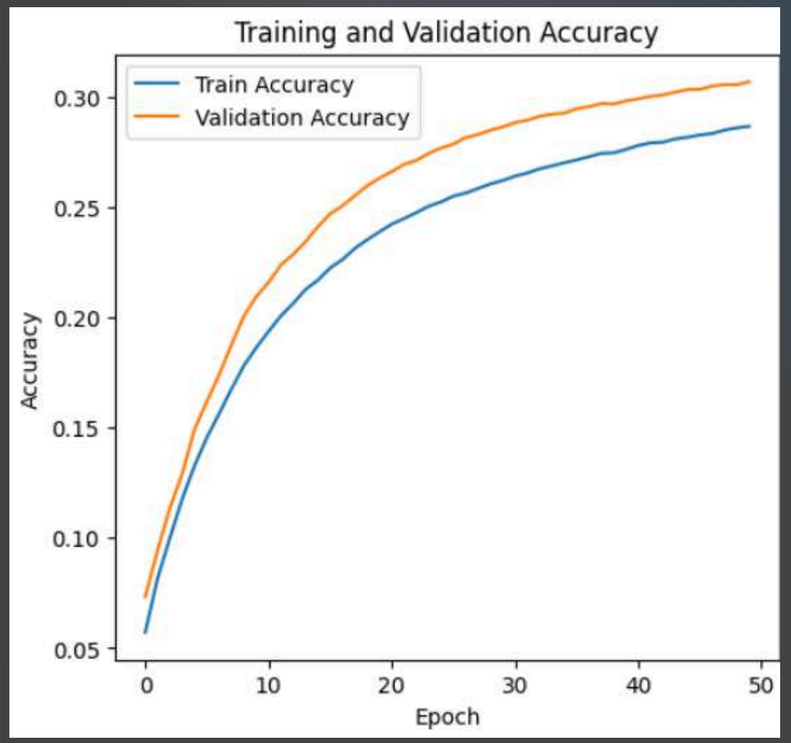
1. Transformer



2. RNN

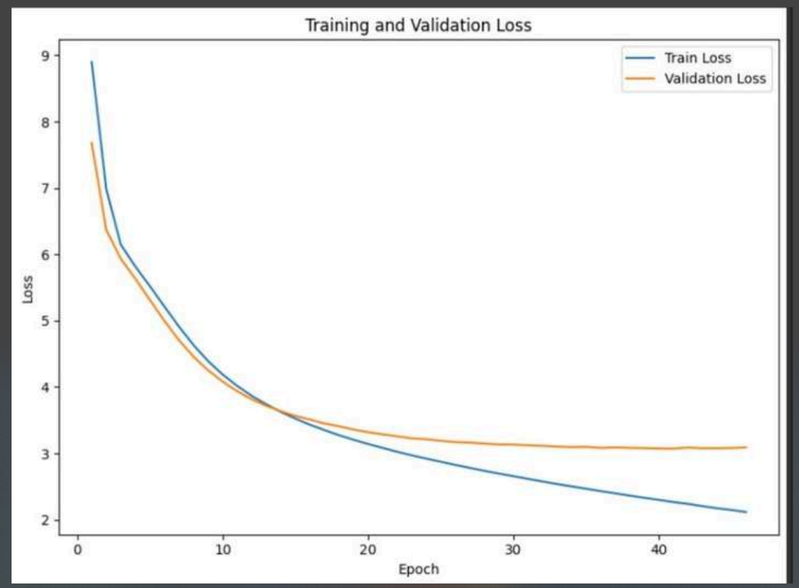


3. LSTM

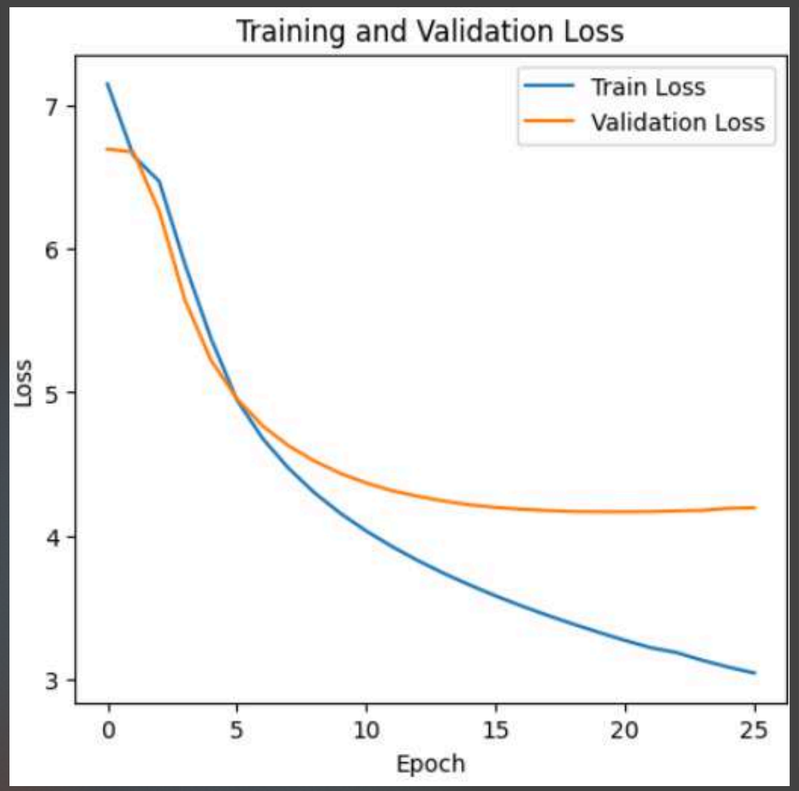


Perbandingan Loss Transformer dengan RNN & LSTM

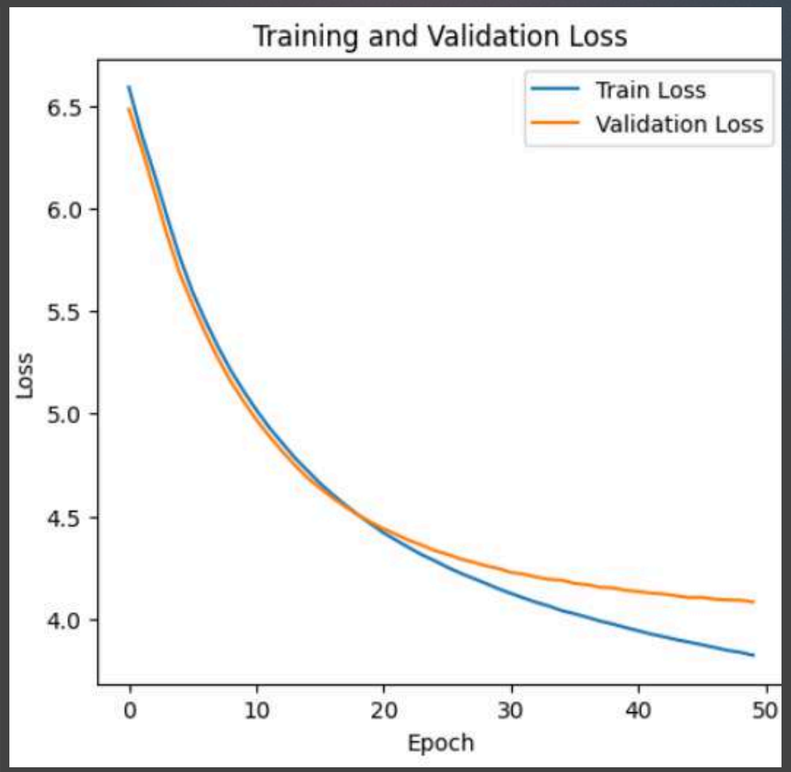
1. Transformer



2. RNN



3. LSTM



Kesimpulan

Meskipun model transformer Q&A yang dibuat oleh kelompok kami mencapai akurasi yang lebih tinggi dibanding model RNN dan LSTM yang telah kami buat sebelumnya, ROUGE-L / BLEU yang dihitung memberikan indikasi kualitas; skor rendah menunjukkan perlu lebih banyak data, regularisasi, atau arsitektur lebih besar untuk perbaikan.

Thank You

for your time
and attention