BAB II
TINJAUAN PUSTAKA

2.1 Landasan Teori

2.1.1 Jurnal Ilmiah

Jurnal ilmiah adalah publikasi periodik yang berisi artikel-artikel yang ditulis oleh peneliti, akademisi, atau ahli di bidang tertentu. Artikel dalam jurnal ilmiah biasanya melalui proses peer-review, di mana para ahli di bidang yang sama menilai kualitas, validitas, dan orisinalitas penelitian sebelum diterbitkan. Jurnal ilmiah berfungsi sebagai media untuk menyebarluaskan hasil penelitian, memfasilitasi diskusi ilmiah, dan mendokumentasikan perkembangan ilmu pengetahuan (Agus Dian Mawardi, 2023).

2.1.2 Text Summarization

Text Summarization adalah proses yang melibatkan pemadatan teks dalam jumlah besar dan menjadi versi yang lebih pendek dengan tetap mempertahankan informasi dari makna yang penting. Tugas ini menjadi semakin penting karena pertumbuhan eksponensial konten tekstual di internet dan di berbagai arsip, seperti artikel berita, makalah ilmiah, dan dokumen hukum (El-Kassas et al., 2021). Terdapat tiga pendekatan dari *text summarization* yaitu ektraktif, abstraktif dan gabungan.

2.1.3 Abstractive Summarization

Abstractive Summarization adalah teknik dalam pemrosesan bahasa alami (NLP) yang bertujuan untuk menghasilkan ringkasan dari teks sumber dengan cara menulis ulang informasi penting dalam bentuk yang lebih ringkas dan mudah dibaca, sering kali menggunakan kosakata yang tidak ada dalam teks asli (Mahira Kirmani et al., 2024).

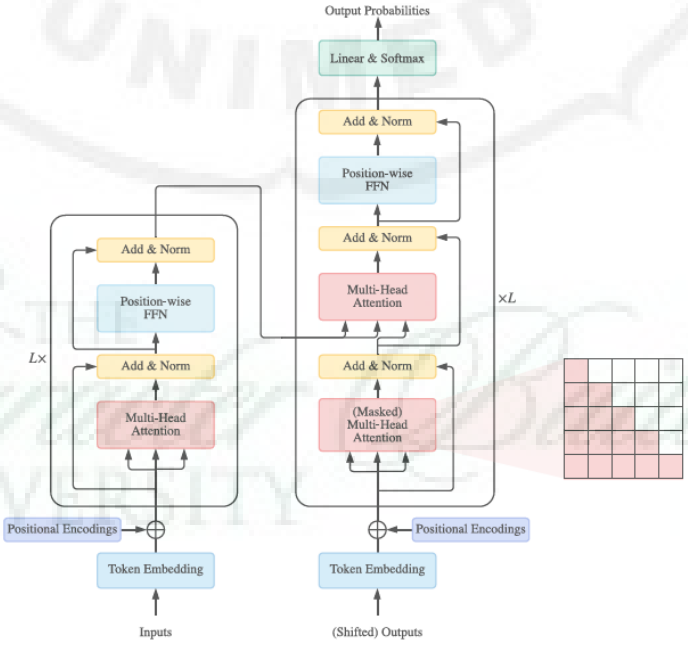
Tantangan yang dihadapi dalam melakukan *Abstractive Summarization* ialah menjaga konsistensi faktual, menangani struktur bahasa yang beragam dan

memastikan pentingnya infromasi. Meskipun matriks evaluasi seperti ROUGE umumnya sering di gunaka, tetap saja evaluasi manusia menjadi standar yang lebih bagus untuk menilai kualitas ringkasan (Virgile Rennard et al., 2022).

Abstractive Summarization telah mengalami kemajuan yang signifikan dengan pengembangan model *Deep Learning*, terutama yang didasarkan pada arsitektur *Sequence-to-Sequence* dan model *Transformer* (Mengli Zhang et al., 2022). Model-model ini mampu memahami dan menghasilkan teks yang mirip manusia, sehingga cocok untuk tugas-tugas *Abstractive Summarization* (Ayham Alomari et al., 2022).

2.1.4 Transformer

Transformer adalah arsitektur model *deep learning* yang telah diadopsi di berbagai bidang kecerdasan buatan seperti *Natural Language Processing, Computer Vision*, dan *Speech Processing*. Arsitektur Transformer terdiri dari encoder dan decoder, masing-masing terdiri dari tumpukan blok identik yang menggabungkan modul *self-attention multi-head* dan *feed-forward network* (Tianyang Lin et al., 2021).



Gambar 2.1 Arsitektur Transformer

Self-attention adalah mekanisme kunci dalam arsitektur Transformer yang memungkinkan model untuk mempertimbangkan hubungan antara setiap pasangan elemen dalam input sequence. Mekanisme ini menghitung bobot perhatian untuk setiap elemen berdasarkan kesamaan dengan elemen lainnya, memungkinkan model untuk menangkap dependensi jangka panjang dalam data (Tianyang Lin et al., 2021).

2.1.5 Text-to-Text Transfer Transformer

T5, atau Text-to-Text Transfer Transformer, adalah model berbasis transformer yang dirancang untuk mendukung pembelajaran multitugas dalam domain pemrosesan bahasa alami (NLP) (Antonio Mastropaolo et al., 2021). Model ini mengikuti arsitektur encoder-decoder dari transformer yang memungkinkan penanganan input dengan ukuran variabel menggunakan lapisan self-attention, bukan RNN atau CNN (Betul Ay et al., 2023). Setiap blok encoder terdiri dari dua sub-kompenen yaitu lapisan self-attention dan feed-forward sederhana (Antonio Mastropaolo et al., 2021). Decoder bekerja mirip dengan encoder, tetapi memiliki mekanisme perhatian tambahan yang memperhatikan output dari encoder (Betul Ay et al., 2023).

T5 lebih efisien daripada RNN Karena memungkinkan komputasi lapisan output secara paralel serta mampu mendeteksi ketergantungan tersembunyi dan jarak jauh antar token yang sangat relevan dalam tugas terkait ringkasan (Antonio Mastropaolo et al., 2021). T5 memiliki lima varian yang berbeda yaitu T5-Small, T5-Base, T5-Large, T5-3 Billion dan T5-11 Billion dimana T5-Small memiliki 60 juta parameter, T5-Base memiliki 220 juta parameter, T5-Large memiliki 770 juta parameter, T5-3 Billion memiliki 3 miliar paramater dan T5-11 Billion memiliki 11 miliar parameter (Qurrota A'yuna Itsnaini et al., 2023).

T5 digunakan untuk multitask learning, yang melibatkan dua fase: pre-training dan fine-tuning. Pre-training mendefinisikan basis pengetahuan bersama untuk berbagai tugas sequence-to-sequence, sementara fine-tuning mengkhususkan model untuk tugas tertentu (Antonio Mastropaolo et al., 2021).

Misalnya ada sebuah teks yaitu “Cuaca hari ini sangat cerah. Banyak orang yang memanfaatkan hari ini untuk berjalan-jalan ke taman atau berolahraga di luar ruangan.". Pemrosesan teks tersebut untuk *text summarization* menggunakan T5 adalah sebagai berikut:

1. Pre-Processing

Pada tahap ini, data teks dibersihkan dan disiapkan agar sesuai dengan kebutuhan model. Prosesnya mencakup penghapusan karakter khusus, normalisasi teks, serta penyesuaian format input. Khusus untuk model T5, teks juga diberi prefix "summarize:" untuk memberi tahu model bahwa tugas yang dilakukan adalah text summarization. Maka setelah dilakukan pre-processing teks tersebut menjadi "summarize: Cuaca hari ini sangat cerah. Banyak orang yang memanfaatkan hari ini untuk berjalan-jalan ke taman atau berolahraga di luar ruangan."

2. Tokenisasi

Setelah pre-processing, teks akan dipecah menjadi unit-unit kecil yang disebut token menggunakan tokenizer spesifik T5 yaitu SentencePiece. Token bisa berupa kata, potongan kata, atau bahkan karakter. Setelah ditokenisasi, kata yang didapat dari proses sebelumnya menjadi “_summarize”, “:”, “_Cuaca”, “_hari”, “_ini”, “_sangat”, “_cerah”, “.”, “_Banyak”, “_orang” “_yang”, “_memanfaatkan”, “_hari”, “_ini”, “_untuk”, “_berjalan”, “-“, “_jalan”, “_ke”, “_taman”, “_atau”, “_berolahraga”, “_di”, “_luar”, “_ruangan”.

3. Token ID, Embedding, dan Positional Encoding

Setelah proses tokenisasi, setiap token diubah menjadi Token ID, yaitu representasi numerik dari masing-masing token berdasarkan daftar kosakata (vocabulary) yang telah dipelajari oleh model. Selanjutnya, Token ID ini akan dipetakan ke dalam vektor berdimensi tetap melalui proses Embedding. Proses ini bertujuan untuk merepresentasikan makna setiap token dalam bentuk vektor numerik berdimensi tetap (contoh: 512 dimensi atau 768 dimensi, tergantung model). Setelah mendapatkan vektor embedding, diterapkan Positional Encoding. Positional Encoding adalah metode yang digunakan dalam Transformer untuk memberikan informasi posisi pada token dalam sebuah urutan, karena Transformer

sendiri tidak memiliki mekanisme bawaan untuk menangkap urutan atau posisi token (Tianyang Lin et al., 2021).

Untuk dimensi ganjil:

$$PE_{(pos,2i)} = sin\left(\frac{pos}{10000^{\frac{2i}{d}}}\right) \quad (2.1)$$

tuk dimensi genap:

$$PE_{(pos,2i+1)} = cos\left(\frac{pos}{10000^{\frac{2i}{d}}}\right) \quad (2.2)$$

mana:

$pos = posisi\ token$

$i = indeks\ dimensi\ dalam\ vektor$

$d = jumlah\ dimensi\ vektor$

Sebagai contoh token ketiga yaitu "_Cuaca", lalu diubah menjadi sebuah vektor yang telah melalui tahapan Token ID, Embedding dan Positional Encoding dengan dimensi 3. Pertama-tama ubah token tersebut menjadi Token ID, anggap Token ID dari token tersebut adalah 3 dengan vektor embedding $[0.5,\ 0.3, 0.2]$ sesuai dengan kosakata model. Selanjutnya menjumlahkannya dengan Positional Encoding.

Untuk dimensi 0 dengan $i = 0$

$$PE_{(2,0)} = sin\left(\frac{2}{10000^{\frac{2.0}{3}}}\right) = sin(2) \approx 0.9093$$

Untuk dimensi 1 dengan $i = 1$

$$PE_{(2,1)} = cos\left(\frac{2}{10000^{\frac{2.1}{3}}}\right) = cos(0.000928) \approx 0.91$$

Untuk dimesi 2 dengan $i = 2$

$$PE_{(2,2)} = sin\left(\frac{2}{10000^{\frac{2.2}{3}}}\right) = sin(0.000431) \approx 0.000431$$

Maka positional encoding untuk token itu ialah [0.9093, 0.91, 0.000431]. Selanjutnya, jumlahkan positional encoding tersebut dengan vektor embedding, sehingga vektor akhir dari token "_Cuaca" ialah [1.40493, 1.21, 0,200431].

4. Encoder

Tahap encoder pada T5 bertujuan untuk mengubah input teks yang telah diproses (token, embedding, positional encoding) menjadi representasi yang kaya makna melalui serangkaian transformasi. Setiap input token (dalam bentuk vektor) akan diproses melalui blok encoder yang terdiri dari beberapa komponen utama yaitu self-attention, add & normalization, feed-forward network, dan add & normalization (Tianyang Lin et al., 2021).

Pertama-tama, input vektor akan masuk ke self-attention layer. Di sini, untuk setiap token, dibuat tiga vektor baru yaitu Query (Q), Key (K), dan Value (V) melalui operasi linier menggunakan bobot terlatih W^Q, W^K, W^V.

Query (Q) adalah representasi dari elemen yang ingin mencari informasi atau "menanyakan" konteks dari elemen lain. Key (K) adalah representasi dari elemen-elemen yang menjadi kandidat untuk diperhatikan atau dicocokkan dengan Query. Value (V) adalah representasi informasi yang akan diambil berdasarkan kecocokan antara Query dan Key (Tianyang Lin et al., 2021).

$$Q = X \cdot W^Q \tag{2.3}$$

$$K = X \cdot W^K \tag{2.4}$$

$$V = X \cdot W^V \tag{2.5}$$

Dimana X adalah input vektor. Selanjutnya, nilai perhatian (attention scores) dihitung dengan rumus:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{D_k}}\right)V \tag{2.6}$$

Di sini, QK^Tadalah hasil perkalian Query dengan transpos dari Key, lalu dibagi dengan akar dari dimensi D_k(biasanya sama dengan dimensi per kepala attention) untuk menstabilkan nilai softmax. Softmax akan menghasilkan distribusi

perhatian (seberapa fokus satu token ke token lain), dan hasilnya digunakan untuk menggabungkan nilai V.

Setelah menghasilkan output dari self-attention, dilakukan proses Add & Norm, yaitu menambahkan hasil attention dengan input awal (menggunakan residual connection) dan kemudian dinormalisasi menggunakan Layer Normalization.

$$Output_{addnorm} = LayerNorm\big(X + Attention(Q, K, V)\big) \quad (2.7)$$

Output ini kemudian masuk ke Feed Forward Neural Network (FFN), yang merupakan dua lapis transformasi linier dengan aktivasi non-linear di tengahnya.

$$FFN(x) = W_2\big(ReLU(W_1X + b_1)\big) + b_2 \quad (2.8)$$

Dimana W_1, b_1, W_2, $b_2$adalah bobot dan bias terlatih. Fungsi ReLU di sini memperkenalkan non-linearitas sehingga model dapat mempelajari pola-pola kompleks. Output dari FFN kemudian juga melewati proses Add & Norm kedua, serupa seperti sebelumnya. Hasil akhirnya adalah representasi akhir (hidden states) dari input yang sudah memahami hubungan antar token dan konteks global kalimat.

Seperti contoh, untuk token "_Cuaca" telah didapatkan vektor nya yaitu $[1.40493,\ 1.21,\ 0.20043]$. Maka proses perhitungan pada tahapan encoder dimulai dengan melakukan *Self-Attention* dengan ketentuan bobot yang didapatkan dari pelatihan model sebagai berikut:

$$W_Q = \begin{bmatrix} 0.5 & 0.1 & 0.2 \\ 0.3 & 0.7 & 0.5 \\ 0.6 & 0.4 & 0.8 \end{bmatrix} W_K = \begin{bmatrix} 0.4 & 0.2 & 0.3 \\ 0.5 & 0.6 & 0.1 \\ 0.7 & 0.5 & 0.9 \end{bmatrix} W_V = \begin{bmatrix} 0.3 & 0.8 & 0.5 \\ 0.2 & 0.4 & 0.6 \\ 0.7 & 0.5 & 0.9 \end{bmatrix}$$

Maka nilai $Q, K\ dan\ V$ adalah:

$$Q = [1.40493, 1.21, 0.20043] \times \begin{bmatrix} 0.5 & 0.1 & 0.2 \\ 0.3 & 0.7 & 0.5 \\ 0.6 & 0.4 & 0.8 \end{bmatrix}$$

$$Q = [1.1857, 1.0677, 1.0463]$$

$$K = [1.40493, 1.21, 0.20043] \times \begin{bmatrix} 0.4 & 0.2 & 0.3 \\ 0.5 & 0.6 & 0.1 \\ 0.7 & 0.5 & 0.9 \end{bmatrix}$$

$$K = [1.3073, 1.1072, 0.7229]$$

$$V = [1.40493, 1.21, 0.20043] \times \begin{bmatrix} 0.3 & 0.8 & 0.5 \\ 0.2 & 0.4 & 0.6 \\ 0.7 & 0.5 & 0.9 \end{bmatrix}$$

$$V = [0.8038, 1.7082, 1.6089]$$

$$Attention\ Score = \frac{Q \times K^T}{\sqrt{D_k}} = \frac{3.4895}{\sqrt{3}} = \frac{3.4895}{1.732} = 2.0154$$

$$Attention\ Output = \ softmax(2.0154) \times [0.8038, 1.7082, 1.6089]$$

$$Attention\ Output = \ 1 \times [0.8038, 1.7082, 1.6089] = [0.8038, 1.7082, 1.6089]$$

Setelah mendapatkan vektor *Attention Output*, dilanjutkan dengan tahap Add & Norm (Residual Connection + Layer Normalization).

$$Residual\ Output = X + Output\ Attention$$

$$Residual\ Output = [1.40493, 1.21, 0.20043] + [0.8038, 1.7082, 1.6089]$$

$$Residual\ Output = [2.20873, 2.9182, 1.8132]$$

$$Layer\ Normalization = \frac{Residual\ Output - \mu}{\sigma}$$

$$\mu = \frac{2.20873 + 2.9182 + 1.8132}{3} = 2.1045$$

$$\sigma = \sqrt{\frac{(2.20873 - 2.1045)^2 + (2.9182 - 2.1045)^2 + (1.8132 - 2.1045)^2}{3}}$$

$$\sigma = \sqrt{\frac{(0.10423)^2 + (0.8137)^2 + (-0.2913)^2}{3}}$$

$$\sigma = \sqrt{\frac{0.01086 + 0.66210 + 0.00848}{3}}$$

$$\sigma = \sqrt{\frac{0.75781}{3}} = \sqrt{0.25260} = 0.50259$$

$$Layer\ Normalization = \frac{[2.2087, 2.9182, 1.8132] - 2.1045}{0.5025}$$

$$Layer\ Normalization = [0.2073, 1.6193, -0.5797]$$

Setelah mendapatkan vektor hasil Layer Normalization, selanjutnya menghitung *Feed Forward Network* dengan nilai bobot dan bias sebagai berikut:

$$W_1 = \begin{bmatrix} 0.2 & 0.5 & 0.3 \\ 0.1 & 0.4 & 0.2 \\ 0.6 & 0.7 & 0.1 \end{bmatrix} \; b_1 = [0.05 \quad 0.1 \quad 0.15]$$

$$W_2 = \begin{bmatrix} 0.7 & 0.5 & 0.3 \\ 0.2 & 0.1 & 0.4 \\ 0.3 & 0.6 & 0.8 \end{bmatrix} \; b_2 = [0.05 \quad 0.1 \quad 0.15]$$

$$FFN(X) = ReLU(X \cdot W_1 + b_1)W_2 + b_2$$

$$Y = ReLU(X \cdot W_1 + b_1)$$

$$Y = [0.2073 \quad 1.6193 \quad -0.5797] \cdot \begin{bmatrix} 0.2 & 0.5 & 0.3 \\ 0.1 & 0.4 & 0.2 \\ 0.6 & 0.7 & 0.1 \end{bmatrix} + [0.05 \quad 0.1 \quad 0.15]$$

$$Y = [-0.0944 \quad 0.4455 \quad 0.4780]$$

$$FFN(X) = [-0.0944 \quad 0.4455 \quad 0.4780] \cdot \begin{bmatrix} 0.7 & 0.5 & 0.3 \\ 0.2 & 0.1 & 0.4 \\ 0.3 & 0.6 & 0.8 \end{bmatrix} + [0.05 \quad 0.1 \quad 0.15]$$

$$FFN(X) = [0.2164 \quad 0.3842 \quad 0.06823]$$

Setelah mendapatkan nilai FFN (Fast Forward Network) dilakukan lagi proses Add & Norm sama seperti sebelumnya.

$$Residual\ Output = Input\ FFN + Output\ FFN$$

$$Residual\ Output = [0.2073, 1.6193, -0.5797] + [0.2164 \quad 0.3842 \quad 0.06823]$$

$$Residual\ Output = [0.4237 \quad 2.0035 \quad -0.5115]$$

$$Layer\ Normalization = \frac{Residual\ Output - \mu}{\sigma}$$

$$\mu = \frac{0.4237 + 2.0035 - 0.5115}{3} = 0.6386$$

$$\sigma = \sqrt{\frac{(0.4237 - 0.6386)^2 + (2.0035 - 0.6386)^2 + (-0.5115 - 0.6386)^2}{3}}$$

$$\sigma = \sqrt{\frac{0.0462 + 1.8630 + 1.3227}{3}}$$

$$\sigma = \sqrt{\frac{3.2319}{3}} = \sqrt{1.0773} = 1.0374$$

$$Layer\ Normalization = \frac{[0.4237 \quad 2.0035 \quad -0.5115] - 0.6386}{1.0374}$$

$$Layer\ Normalization = [-0.2072 \quad 1.3160 \quad -1.1088]$$

Maka untuk token "_Cuaca" vektor hasil encoder nya ialah

$$[-0.2072 \quad 1.3160 \quad -1.1088]$$

5. Decoder

Decoder di T5 bertugas menghasilkan output (ringkasan, dalam kasus text summarization) berdasarkan hasil dari encoder dan token-token sebelumnya yang sudah dihasilkan. Decoder mirip struktur dengan encoder, tetapi ada tambahan penting yaitu Cross Attention yang mana token decoder mencocokkan dirinya dengan output encoder (Tianyang Lin et al., 2021).

Misalnya token "_Cuaca" yang telah melewati tahap encoder menghasilkan vektor yaitu [-0.2072, 1.3160, -1.1088] maka dilakukan perhitungan Self-Attention sebagai berikut:

$$W_Q = \begin{bmatrix} 0.2 & 0.1 & 0.4 \\ 0.3 & 0.5 & 0.2 \\ 0.6 & 0.7 & 0.9 \end{bmatrix} W_K = \begin{bmatrix} 0.3 & 0.6 & 0.9 \\ 0.4 & 0.2 & 0.8 \\ 0.7 & 0.5 & 0.1 \end{bmatrix} W_V = \begin{bmatrix} 0.5 & 0.7 & 0.4 \\ 0.1 & 0.8 & 0.3 \\ 0.6 & 0.2 & 0.9 \end{bmatrix}$$

Maka nilai $Q, K\ dan\ V$ adalah:

$$Q = [-0.2072, 1.3160, -1.1088] \times \begin{bmatrix} 0.2 & 0.1 & 0.4 \\ 0.3 & 0.5 & 0.2 \\ 0.6 & 0.7 & 0.9 \end{bmatrix}$$

$$Q = [-0.31192, -0.13888, -0.8176]$$

$$K = [-0.2072, 1.3160, -1.1088] \times \begin{bmatrix} 0.3 & 0.6 & 0.9 \\ 0.4 & 0.2 & 0.8 \\ 0.7 & 0.5 & 0.1 \end{bmatrix}$$

$$K = [-0.31192, -0.41552, 0.75544]$$

$$V = [-0.2072, 1.3160, -1.1088] \times \begin{bmatrix} 0.5 & 0.7 & 0.4 \\ 0.1 & 0.8 & 0.3 \\ 0.6 & 0.2 & 0.9 \end{bmatrix}$$

$$V = [0.37408, 0.69944, -0.85896]$$

$$Attention\ Score = \frac{Q \times K^T}{\sqrt{D_k}} = \frac{-0.4626}{\sqrt{3}} = \frac{-0.4626}{1.732} = -0.267$$

$$Attention\ Output = \ softmax(-0.267) \times [0.37408, 0.69944, -0.85896]$$

$$Attention\ Output = [0.37408, 0.69944, -0.85896]$$

Setelah mendapatkan vektor Attention Output, dilanjutkan dengan tahap Add & Norm (Residual Connection + Layer Normalization).

$$Residual\ Output = Output\ Attention + Input\ Decoder$$

$$Residual\ Output = [0.37408, 0.69944, -0.85896] + [-0.2072, 1.3160, -1.1088]$$

$$Residual\ Output = [0.16688, 2.01544, -1.96776]$$

$$Layer\ Normalization = \frac{Residual\ Output - \mu}{\sigma}$$

$$\mu = \frac{0.16688 + 2.01544 + (-1.96776)}{3} = 0.07152$$

$$\sigma = \sqrt{\frac{(0.16688 - 0.07152)^2 + (2.01544 - 0.07152)^2 + (-1.96776 - 0.07152)^2}{3}}$$

$$\sigma = \sqrt{\frac{7.98044}{3}} = \sqrt{2.66015} \approx 0.50259$$

$$Layer\ Normalization = \frac{[0.16688, 2.01544, -1.96776] - 0.07152}{0.50259}$$

$$Layer\ Normalization = [0.0585, 1.192, -1.262]$$

Setelah mendapatkan vektor hasil Layer Normalization, selanjutnya menghitung *Encoder-Decoder Attention* dimana Q ialah output dari layer normalization, Key dan Value ialah vektor output dari encoder.

$$Attention\ Score = \frac{Q \times K^T}{\sqrt{D_k}} = \frac{2.9577554}{\sqrt{3}} = \frac{2.9577554}{1.732} \approx 1.707$$

$$Encoder\ Decoder\ Output = softmax(-0.267) \times [-0.2072, 1.3160, -1.1088]$$

$$Encoder\ Decoder\ Output = [-0.2072, 1.3160, -1.1088]$$

Setelah mendapatkan vektor hasil Encoder Decoder Attention, selanjutnya menghitung *Feed Forward Network* dengan nilai bobot dan bias sebagai berikut:

$$W_1 = \begin{bmatrix} 0.5 & -0.3 & 0.8 \\ -0.7 & 0.2 & 0.4 \\ 0.6 & -0.5 & 0.1 \end{bmatrix} \quad b_1 = [0.1 \quad -0.2 \quad 0.05]$$

$$W_2 = \begin{bmatrix} 0.3 & -0.6 & 0.2 \\ -0.5 & 0.1 & 0.7 \\ 0.4 & -0.2 & 0.5 \end{bmatrix} \quad b_2 = [0.05 \quad -0.1 \quad 0.2]$$

$$FFN(X) = ReLU(X \cdot W_1 + b_1)W_2 + b_2$$

$$h_1 = X \cdot W_1 + b_1$$

$$h_1 = [-0.2072 \quad 1.3160 \quad -1.1088] \cdot \begin{bmatrix} 0.5 & -0.3 & 0.8 \\ -0.7 & 0.2 & 0.4 \\ 0.6 & -0.5 & 0.1 \end{bmatrix} + [0.05 \quad 0.1 \quad 0.15]$$

$$h_1 = [-1.28544 \quad -0.23528 \quad -0.8432]$$

$$Y = ReLU(h_1)$$

$$Y = ReLU([-1.28544 \quad -0.23528 \quad -0.8432])$$

$$Y = [0 \quad 0 \quad 0]$$

$$FFN(X) = [0 \quad 0 \quad 0] \cdot \begin{bmatrix} 0.3 & -0.6 & 0.2 \\ -0.5 & 0.1 & 0.7 \\ 0.4 & -0.2 & 0.5 \end{bmatrix} + [0.05 \quad -0.1 \quad 0.2]$$

$$FFN(X) = [0.05 \quad -0.1 \quad 0.2]$$

Setelah mendapatkan nilai FFN (Fast Forward Network) dilakukan lagi proses Add & Norm sama seperti sebelumnya.

$$Residual\ Output = Input\ FFN + Output\ FFN$$

$$Residual\ Output = [-0.2072 \quad 1.3160 \quad -1.1088] + [0.05 \quad -0.1 \quad 0.2]$$

$$Residual\ Output = [-0.1572 \quad 1.2160 \quad -0.9088]$$

$$Layer\ Normalization = \frac{Residual\ Output - \mu}{\sigma}$$

$$\mu = \frac{-0.1572 + 1.2160 - 0.9088}{3} = 0.0500$$

$$\sigma = \sqrt{\frac{(-0.1572 - 0.0500)^2 + (1.2160 - 0.0500)^2 + (-0.9088 - 0.0500)^2}{3}}$$

$$\sigma = \sqrt{\frac{2.3218}{3}} = \sqrt{0.7739} = 0.8795$$

$$Layer\ Normalization = \frac{[-0.1572 \quad 1.2160 \quad -0.9088] - 0.0500}{0.8795}$$

$$Layer\ Normalization = [-0.2356 \quad 1.3259 \quad -1.0903]$$

Maka untuk token "_Cuaca" vektor hasil decoder nya ialah

$$[-0.2356 \quad 1.3259 \quad -1.0903]$$

6. Linear & Softmax

Setelah melewati seluruh lapisan decoder, setiap posisi dalam urutan keluaran menghasilkan representasi vektor berdimensi tinggi. Vektor ini mengandung informasi kontekstual baik dari input sequence (melalui mekanisme cross-attention) maupun dari token-token output yang telah dihasilkan sebelumnya (melalui masked self-attention). Namun, vektor ini belum berbentuk distribusi abilitas atas kosakata target, sehingga belum dapat langsung digunakan untuk tugas seperti penerjemahan atau pembuatan teks.Tahap akhir dalam arsitektur decoder Transformer terdiri atas dua proses berurutan yaitu transformasi linear dan fungsi aktivasi softmax dengan rumus sebagai berikut:

$$Logits = H_{dec}W_{out} + b_{out}$$

Dimana:

- H_{dec} adalah output decoder

- W_{out} adalah matriks bobot linear
- b_{out} adalah bias

$$p_i = Softmax(Z_i) = \frac{e^{Z_i}}{\sum_j e^{Z_i}} \quad (2.9)$$

Dimana:

- e^{Z_i} adalah nilai logit untuk token ke-i di posisi
- $\sum_j e^{Z_i}$ adalah jumlah seluruh nilai logit

Misalnya untuk token "_Cuaca" hasil decodernya adalah [-0.2356,1.3259,-1.0903]. Maka dilakukan perhitungan Logits dan Softmax sebagai berikut.

$$W_{out} = \begin{bmatrix} 0.4 & 0.3 & 0.2 \\ -0.7 & 0.6 & 0.8 \\ 0.6 & -0.1 & 0.5 \end{bmatrix} \; b_{out} = [0.1 \quad -0.2 \quad 0.05]$$

$$Logits = [-0.2356 \quad 1.3259 \quad -1.0903] \begin{bmatrix} 0.4 & 0.3 & 0.2 \\ -0.7 & 0.6 & 0.8 \\ 0.6 & -0.1 & 0.5 \end{bmatrix} + [0.1 \quad -0.2 \quad 0.05]$$

$$Logits = [-1.67655 \quad 0.83389 \quad 0.46845] + [0.1 \quad -0.2 \quad 0.05]$$

$$Logits = [-1.57655 \quad 0.63389 \quad 0.51845]$$

$$p_i = Softmax(Z_i) = \frac{e^{Z_i}}{\sum_j e^{Z_i}}$$

$$e^{-1.57655} \approx 0.2066$$

$$e^{0.63389} \approx 1.8845$$

$$e^{0.51845} \approx 1.6793$$

$$Sum = 0.2066 + 1.8845 + 1.6793 = 3.7704$$

$$p_1 = \frac{0.2066}{3.7704} = 0.0548$$

$$p_2 = \frac{1.8845}{3.7704} = 0.4998$$

$$p_i = \frac{1.6793}{3.7704} = 0.4454$$

$Softmax = [0.0548 \quad 0.4998 \quad 0.4454]$

Setelah mendapatkan nilai softmax dari masing masing token, tahap selanjutnya adalah menentukan output ringkasan dengan greedy search yaitu mengambil token dengan softmax tertinggi, misalnya diambil 7 token untuk jadikan ringkasan

Tabel 2.1 Hasil Softmax

Token	Softmax
"_Cuaca"	[0.8998, 0.05, 0.0502]
"_hari"	[0.05, 0.6, 0.35]
"_ini"	[0.15, 0.2, 0.65]
"_sangat"	[0.25, 0.7, 0.05]
"_cerah"	[0.85, 0.1, 0.05]
"."	[0.2, 0.7, 0.1]
"_Banyak"	[0.9, 0.05, 0.05]
"_orang"	[0.88, 0.07, 0.05]
"_yang"	[0.1, 0.8, 0.1]
"_memanfaatkan"	[0.2, 0.7, 0.1]
"_hari" (ulang)	[0.05, 0.7, 0.25]
"_ini" (ulang)	[0.1, 0.3, 0.6]
"_untuk"	[0.2, 0.6, 0.2]
"_berjalan"	[0.25, 0.5, 0.25]
"-"	[0.1, 0.8, 0.1]

"_jalan"	[0.2, 0.5, 0.3]
"_ke"	[0.1, 0.7, 0.2]
"_taman"	[0.3, 0.6, 0.1]
"_atau"	[0.3, 0.5, 0.2]
"_berolahraga"	[0.85, 0.1, 0.05]
"_di"	[0.85, 0.1, 0.05]
"_luar"	[0.9, 0.05, 0.05]
"_ruangan"	[0.2, 0.6, 0.2]

Berdasarkan tabel itu, token dengan abilitas tertinggi yaitu “_Cuaca”, “_cerah”, “_Banyak”, “_orang”, “_berolahraga”, “_di”, dan “_luar”. Sehingga hasil output ringkasannya adalah “Cuara cerah banyak orang berolahraga di luar”.

2.1.6 ROUGE

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) adalah metrik yang digunakan untuk mengevaluasi kualitas ringkasan teks dengan membandingkannya dengan ringkasan emas (gold summary) yang dihasilkan oleh manusia. Metrik ini mengukur tingkat kesamaan berdasarkan unit yang tumpang tindih, seperti pasangan kata, n-gram, dan urutan kata, antara ringkasan yang dihasilkan oleh model dan ringkasan referensi (Alshimaa et al., 2023).

ROUGE memiliki beberapa varian, di antaranya ROUGE-N dan ROUGE-L. ROUGE-N mengukur jumlah n-gram yang tumpang tindih antara ringkasan model dan referensi. ROUGE-L menggunakan Longest Common Subsequence (LCS) untuk mengevaluasi kesamaan berdasarkan urutan kata terpanjang yang muncul secara berurutan dalam kedua ringkasan (Alshimaa et al., 2023).

$$ROUGE - N = \frac{\sum S \in \{ReferenceSummaries\} \sum gram_n \in SCount_{match}(gram_n)}{\sum S \in \{ReferenceSummaries\} \sum gram_n \in SCount(gram_n)} \quad (2.10)$$

Dimana:

- S ∈ {ReferenceSummaries} → Menunjukkan bahwa diambil setiap ringkasan referensi dalam kumpulan ringkasan emas.
- $gram_n \in S$ → Menunjukkan bahwa diambil setiap n-gram dalam ringkasan referensi tersebut.
- $Countmatch(gram_n)$ → Jumlah n-gram dalam ringkasan model yang cocok dengan n-gram di ringkasan referensi.
- $Count(gram_n)$ → Jumlah total n-gram dalam ringkasan referensi.

ROUGE-L juga mempertimbangkan konsistensi struktur tingkat kalimat dan dihitung menggunakan formula yang melibatkan recall dan precision, dengan parameter β yang menyesuaikan signifikansi antara keduanya (Alshimaa et al., 2023).

$$ROUGE - L = \frac{(1 + \beta^2) * Recal * Precision}{Recall + \beta^2 * Precission} \quad (2.11)$$

$$Recall = \frac{TP}{TP + FN} \quad (2.12)$$

$$Precision = \frac{TP}{TP + FP} \quad (2.13)$$

Dimana:

- TP (True Positive): Kata dalam ringkasan model yang juga ada dalam referensi, dalam urutan yang benar sesuai dengan Longest Common Subsequence (LCS).
- FP (False Positive): Kata dalam ringkasan model yang tidak ada dalam referensi atau urutannya salah.
- FN (False Negative): Kata dalam ringkasan referensi yang tidak muncul dalam ringkasan model.

Misalnya ada sebuah teks yaitu “Cuaca hari ini sangat cerah. Banyak orang yang memanfaatkan hari ini untuk berjalan-jalan ke taman atau berolahraga di luar ruangan." dan diketahui hasil ringkasan dari model yaitu “Banyak orang berjalan-jalan atau berolahraga karena cuaca cerah.” Sedangkan referensi ringkasannya yaitu “Hari ini cuaca cerah, sehingga banyak orang beraktivitas di luar ruangan.”.

Pemrosesan evaluasi menggunakan ROUGE nya adalah sebagai berikut:

1. Tokenisasi Hasil Ringkasan dan Referensi Ringkasan

Didapat tokenisasi hasil ringkasan yaitu “banyak”, “orang”, “berjalan-jalan”, “atau”, “berolahraga”, “karena”, “cuaca”, “cerah”. Sedangkan referensi ringkasan yaitu “hari”, “ini”, “cuaca”, “cerah”, “sehingga, “banyak”, “orang”, “beraktivitas”, “di”, “luar”, “ruangan”.

2. Menghitung ROUGE-1

Unigram yang muncul dari hasil ringkasan dan referensi ringkasan yaitu “banyak”, “orang”, “cuaca”, “cerah”.

$$Precision = \frac{TP}{TP + FP} = \frac{4}{4 + 4} = \frac{4}{8} = 0.5$$

$$Recall = \frac{TP}{TP + FN} = \frac{4}{4 + 7} = \frac{4}{11} = 0.3636$$

$$F1 - Score = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} = \frac{2 \cdot 0.5 \cdot 0.3636}{0.5 + 0.3636} \approx \frac{0.3636}{0.8636} \approx 0.421$$

3. Menghitung ROUGE-2

Di dapat bigram pada hasil ringkasan yaitu 'banyak orang', 'orang berjalan-jalan', 'berjalan-jalan atau', 'atau berolahraga', 'berolahraga karena', 'karena cuaca', 'cuaca cerah' sedangkan bigram pada referensi ringkasan yaitu 'hari ini', 'ini cuaca', 'cuaca cerah', 'cerah sehingga', 'sehingga banyak', 'banyak orang', 'orang beraktivitas', 'beraktivitas di', 'di luar', 'luar ruangan' maka bigram (2-gram) yang muncul dari hasil ringkasan dan referensi ringkasan yang cocok adalah "banyak orang" dan "cuaca cerah" sedangkan jumlah bigram di hasil ringkasan ialah 7 dan jumlah bigram di referensi ringkasan ialah 10.

$$Precision = \frac{TP}{TP + FP} = \frac{2}{2 + 5} = \frac{2}{7} \approx 0.2857$$

$$Recall = \frac{TP}{TP + FN} = \frac{2}{2 + 8} = \frac{2}{10} = 0.2$$

$$F1 - Score = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} = \frac{2 \cdot 0.2857 \cdot 0.2}{0.2857 + 0.2} \approx \frac{0.1143}{0.2857} \approx 0.2354$$

4. Menghitung ROUGE-L

Longest Common Subsequence (LCS) atau urutan kata yang paling panjang yang muncul di kedua kalimat (hasil dan referensi ringkasan) dalam urutan yang sama dari adalah "cuaca", "cerah", "banyak", "orang".

$$Precision = \frac{TP}{TP + FP} = \frac{4}{4 + 4} = \frac{4}{8} = 0.5$$

$$Recall = \frac{TP}{TP + FN} = \frac{4}{4 + 7} = \frac{4}{11} = 0.3636$$

$$F1 - Score = \frac{(1 + \beta^2) * Recal * Precision}{Recall + \beta^2 * Precission}$$

$$F1 - Score = \frac{(1 + 1^2) \cdot 0.5 \cdot 0.3636}{0.5 + 1^2 \cdot 0.3636} \approx \frac{0.3636}{0.8636} \approx 0.421$$

Tabel 2.2 Hasil Evaluasi ROUGE

ROUGE	Precision	Recall	F1-Score
ROUGE-1	0.5	0.3636	0.421
ROUGE-2	0.2857	0.2	0.2354
ROUGE-L	0.5	0.3636	0.421

2.2 Penelitian Relevan

1. I Nyoman Purnama dan Ni Nengah Widya Utami (2023), pada penelitian yang berjudul “IMPLEMENTASI PERINGKAS DOKUMEN BERBAHASA INDONESIA MENGGUNAKAN METODE TEXT TO TEXT TRANSFER TRANSFORMER (T5)”. Penelitian ini berfokus pada implementasi sistem peringkas dokumen berita berbahasa Indonesia menggunakan model T5 (Text to Text Transfer Transformer). Tujuan dari penelitian ini adalah untuk menyajikan informasi penting dari berita digital dalam bentuk ringkasan singkat, mengingat banyaknya informasi yang tersedia yang dapat membuat pembaca kesulitan menemukan intisari. Penelitian ini menggunakan dataset INDOSUM, yang terdiri dari 19.000

berita dan ringkasan yang dibuat oleh manusia. Evaluasi dilakukan menggunakan metrik ROUGE untuk menilai kualitas ringkasan yang dihasilkan. Terdapat tiga skenario preprocessing yang diuji dalam penelitian ini yaitu skenario pertama menggunakan stemming dan penghapusan stopwords. Hasil dari skenario ini menunjukkan nilai F1-score rata-rata yang rendah sebesar 0.07612, dengan nilai maksimum 0.5497. Penghapusan stopwords secara signifikan mengurangi panjang kata, yang berdampak negatif pada kualitas peringkasan. Skenario kedua yaitu menggunakan stemming tanpa penghapusan stopwords. Skenario ini mencapai nilai F1-score rata-rata yang lebih baik sebesar 0.17866, dengan nilai maksimum 0.5297. Hasil ini menunjukkan peningkatan yang signifikan, menandakan bahwa penghapusan stopwords sangat mempengaruhi kinerja. Dan Skenario terakhir ialah tidak menggunakan stemming maupun penghapusan stopwords. Skenario ini menghasilkan nilai F1-score rata-rata sebesar 0.17568, dengan nilai maksimum 0.5397. Hasil ini sedikit lebih rendah dibandingkan skenario 2, menunjukkan bahwa stemming memiliki efek positif yang kecil terhadap kinerja. Kesimpulan dari penelitian ini adalah bahwa penghapusan stopwords memiliki dampak yang substansial terhadap kualitas peringkasan, sementara stemming memiliki efek yang lebih kecil. Meskipun stemming dapat meningkatkan kinerja, hal itu tidak secara signifikan mengubah kealamian ringkasan dibandingkan dengan skenario tanpa stemming. Penelitian ini menunjukkan potensi model T5 dalam peringkasan teks abstraktif dalam bahasa Indonesia.

2. Andika Bahari dan Kania Evita Dewi (2024), pada penelitian yang berjudul "PERINGKASAN TEKS OTOMATIS ABSTRAKTIF MENGGUNAKAN TRANSFORMER PADA TEKS BAHASA INDONESIA". Penelitian ini berfokus pada peringkasan teks abstraktif dalam bahasa Indonesia menggunakan arsitektur Transformer, khususnya model T5. Sebelumnya, model-model berbasis recurrent seperti RNN, LSTM, dan GRU sering digunakan untuk tugas ini, namun mereka memiliki kelemahan dalam hal kohesi dan tata bahasa. Transformer, dengan mekanisme attention-nya, menawarkan solusi yang lebih baik dengan mempertahankan informasi dari

seluruh urutan input. Penelitian ini menggunakan model T5 yang telah dilatih sebelumnya dengan dataset IndoSum, yang terdiri dari 19 ribu pasangan berita dan ringkasan. Evaluasi menunjukkan bahwa model ini mencapai skor ROUGE-1 sebesar 0.61 dan ROUGE-2 sebesar 0.51, yang menunjukkan performa yang baik dalam peringkasan teks abstraktif. Selain itu, penelitian ini juga membandingkan dua model: T5-default (menggunakan vocab non-Indonesia) dan T5-id (menggunakan vocab Indonesia). Hasilnya menunjukkan bahwa penggunaan vocab Indonesia mengurangi jumlah token dan mempercepat waktu pelatihan, meskipun menghasilkan validation loss yang lebih tinggi. Namun, model T5-id menunjukkan performa yang lebih baik dalam evaluasi ROUGE, dengan skor yang lebih tinggi dibandingkan T5-default. Meskipun T5-id memiliki skor yang lebih baik, T5-default menunjukkan struktur kalimat yang lebih konsisten. Penelitian ini menyoroti efektivitas model transformer dalam peringkasan teks, meskipun masih ada masalah seperti kesalahan ketik dan tata bahasa. Rekomendasi untuk penelitian di masa depan termasuk penggunaan vocab yang relevan, model pra-latih, dan teknik fine-tuning untuk meningkatkan performa dan kepatuhan terhadap standar bahasa Indonesia.