BAB II

TINJAUAN PUSTAKA DAN DASAR TEORI

2.1 Tinjauan Pustaka

Dunia pendidikan saat ini sudah mulai memanfaatkan *Artificial Intelligence* untuk mempermudah dalam proses belajar mengajar ke siswa. Dunia pendidikan dalam memanfaatkan AI untuk mempermudah guru dalam proses belajar mengajar salah satunya pembuatan umpan balik otomatis dari jawaban siswa yang merupakan salah satu aspek penting dalam dunia pendidikan. Perkembangan teknologi telah mendorong perhatian terhadap pengembangan sistem yang dapat memberikan umpan balik secara cepat dan konsisten, mengatasi keterbatasan waktu dan sumber daya manusia dalam memberikan umpan balik secara manual. Penelitian-penelitian terdahulu telah menginvestigasi berbagai metode dan teknik untuk meningkatkan kemampuan sistem dalam menghasilkan umpan balik yang relevan dan bermakna. Tabel 2.1 dan Tabel 2.1(lanjutan) menyajikan tinjauan dari penelitian-penelitian sebelumnya terkait pembuatan umpan balik otomatis dari *dataset essay*, laporan proyek siswa dan *short answer* dalam bahasa Inggris dengan menggunakan berbagai macam metode yang dapat menghasilkan umpan balik.

Tabel 2.1 Penelitian terdahulu

No	Penelitian	Bahasa	Dokumen	Metode	Evaluasi
1	Altoe dan Joyner, (2019)	Inggris	Essay	*Textrank, Automatic Concept Map Generation*	-
2	Nagata (2019)	Inggris dan Jepang	Essay	*Neural Retrieval-based Method, Sequence-to-sequence Model-based Method, Case frame-based Method*	Evaluasi Manual, *precision,Recall* dan F-*measure*
3	Süzen dkk. (2020)	Inggris	Uraian singkat	*K-Means Clustering*	MSE
4	Hanawa dkk. (2021)	Inggris	Essay	*simple generation, retrieve method dan retrieve-edit method*	*accuracy*

Tabel 2.1 Penelitian terdahulu (lanjutan)

No	Penelitian	Bahasa	Dokumen	Metode	Evaluasi
5	Lu dan Cutumisu (2021)	Inggris	Essay	*Constrained Metropolis-Hastings Sampling* (CGMH)	*Negative Likelihood (NLL), Human anotator*
6	Jia dkk. (2022)	Inggris	Laporan proyek Siswa	BART dan *diverse beam search*	*Human centered evaluation,* ROUGE, *Bertscore*
7	PHAN (2023)	Jepang	Essay	Sbert + GPT	*Human Evaluation*
8	Ueda dan Komachi (2023)	Inggris	Essay	BART dan T5	Bleu dan *manual evaluation*
9	Stahl dan Wachsmuth (2023)	Inggris	Essay	BART	Bleu, *Manual Evaluation*
10	Y. Liu dkk. (2024)	Inggris	Essay	Bi-Gru + *Attention Mechanism*	Bleu, ROUGE, DIST, *Manual Evaluation*

Süzen dkk.,(2020) melakukan penilaian otomatis soal jawaban singkat dengan fokus pada pengukuran kemiripan antara jawaban siswa dan jawaban model, pengelompokan respon siswa untuk umpan balik, dan prediksi nilai berdasarkan kemiripan. Pengelompokan respon siswa untuk umpan balik menggunakan K-means *clustering*. Hasil menunjukkan cluster dibagi 3 yakni kelompok *excellent, mixed* dan *weak*. Penelitian ini terdapat kekurangan yaitu untuk menghasilkan umpan balik masih secara *manual* berdasarkan jumlah *cluster* dan sensitivitasnya terhadap penempatan awal *centroid*, sehingga dapat menyebabkan hasil *clustering* yang kurang optimal.

Penelitian yang dilakukan oleh Altoe dan Joyner (2019), membuat rubrik penilaian esai otomatis dan menampilkan umpan balik. Peneliti menggunakan TextRank untuk peringkasan dan pemeringkatan esai untuk pembuatan rubrik otomatis. Langkah selanjutnya memanfaatkan *automatic concept map generation* / pembuatan peta konsep untuk presentasi umpan balik dengan cara analisis

kesamaan antara rubrik/acuan yang dihasilkan dan esai tes yang bukan merupakan bagian dari esai acuan/rubrik. Kekurangan dari metode penelitian ini adalah sangat bergantung pada korpus data pelatihan sehingga menyebabkan kreatifitas yang terbatas.

Nagata (2019), mengawali pembuatan *feedback comment generation essay* untuk pembelajaran menulis. Nagata (2019), membuat korpus dan korpus tersebut dapat digunakan untuk *feedback comment generation*. Nagata (2019) membandingkan tiga metode yaitu *neural retrieval based method*, *sequence to sequence model based method*, *case frame based method*. *Neural retrieval based method* merupakan metode yang memiliki kelayakan pemberian *feedback comment generation*. Metode *neural retrieval based method* terdapat kekurangan yaitu memiliki keterbatasan karena hanya dapat menghasilkan komentar umpan balik yang ada pada data pelatihan tertentu. Kekurangan dari penelitian yang dilakukan Nagata (2019) adalah berfokus pada preposisi saja padahal pemberian umpan balik bisa bersifat umum.

Hanawa dkk., (2021), mengembangkan penelitian yang dilakukan oleh Nagata (2019) yaitu menggunakan korpus yang telah dikembangkan (Nagata dkk., 2020) dengan fokus pada umpan balik preposisi dan penambahan umpan balik umum kemudian menguji *feedback comment generation* hasil korpus dengan tiga metode yaitu *simple generation, retrieve method dan retrieve* dan *edit method*. Akurasi hasil yang dilakukan oleh Hanawa dkk., (2021) dan Nagata (2019), masih belum sempurna, oleh karena itu (Ueda dan Komachi, 2023)memanfaatkan BART dan T5 untuk menghasilkan *feedback comment generation essay* dan hasilnya jauh lebik baik. Stahl dan Wachsmuth (2023) melakukan identifikasi jenis umpan balik untuk meningkatkan pembuatan komentar umpan balik.

Lu dan Cutumisu, (2021), melakukan penggabungan antara penilaian otomatis dan pemberian umpan balik otomatis yang bereksperimen dengan dataset ASAP, namun umpan baliknya dihasilkan dari skor keseluruhan, bukan dari skor sifat. Penelitan tersebut membuat sebuah rubrik pedoman untuk skor keseluruhan dan selanjutnya dirancang serta dilakukan pelatihan model untuk mendapatkan skor keseluruhan menggunakan berbagai jaringan saraf. Umpan balik akhir diperoleh

dengan menggunakan generator umpan balik yang dipelajari secara keseluruhan dari pedoman skor dan rubrik.

Phan, (2023), melakukan penelitian yaitu menggabungkan penilaian otomatis dan pemberian umpan balik otomatis pada soal *essay* yang ditulis oleh siswa Jepang. Penilaian otomatis menggunakan BERT. Hasil umpan balik dalam penelitiannya memiliki tujuan akhir yaitu untuk menghasilkan jawaban yang mirip manusia (umpan balik sugestif) dan melakukan perbandingan dengan efektivitas jawaban rekan (umpan balik korektif). SBERT digunakan untuk menghasilkan umpan balik jawaban rekan, dan ChatGPT API yang menggunakan model GPT-3 digunakan untuk hasil umpan balik yang mirip manusia.

Y. Liu dkk.(2024), memperkenalkan model jaringan saraf, GEEF, yang dirancang untuk menghasilkan umpan balik pada esai dengan memasukkan skor penilaian esai untuk meningkatkan kualitas umpan balik. Model ini menggunakan arsitektur *seq2seq* dengan *attention mechanism* dan mempertimbangkan aspek-aspek seperti koherensi, kekayaan, dan bakat sastra. Hasil eksperimen menunjukkan bahwa model ini melampaui metode dasar dalam menghasilkan masukan, menawarkan wawasan berharga bagi siswa untuk meningkatkan keterampilan menulis mereka dan membantu penilai dalam memberikan masukan yang lebih rinci.

Jia dkk.(2022), melakukan penelitian pemberian umpan balik pada laporan proyek siswa menggunakan BART dan *diverse beam search* untuk menghasilkan umpan balik. Hasilnya dengan evaluasi Bertscore dan *human centered evaluation*, Bart memiliki kemampuan untuk menghasilkan *text generation*. Penelitian – penelitian terdahulu terkait uraian singkat masih belum dilakukan penelitian lebih mendalam. . Penelitian seperti algoritma *deep learning* yang salah satunya BART memiliki kemampuan menghasilkan sebuah *text generation* dikarenakan BART merupakan transformers yang memanfaatkan arsitektur transformers yaitu *encoder* dan *decoder*. Penelitian sebelumnya mengenai pemberian umpan balik otomatis masih memiliki pemahaman kontekstual yang terbatas dan tidak bekerja dengan baik dalam mempertahankan urutan, hal tersebut penting dalam *text generation* umpan balik. BART sendiri memiliki kelebihan yaitu (I) dapat memproses teks

secara dua arah sehingga memiliki kemungkinan untuk menangkap konteks dan makna teks secara komprehensif. (II) BART dapat menghasilkan teks dari kiri ke kanan sehingga meningkatkan pemahaman konteks secara lebih menyeluruh. Akan tetapi terdapat kekurangan dari BART yaitu ketergantungan terhadap data latih dan *overfitting* jika tidak dilakukan *fine tune* yang tepat.

2.2 Dasar Teori

2.2.1 Umpan balik dalam pembelajaran

Umpan balik yang diberikan kepada siswa setelah menjawab pertanyaan penilaian merupakan elemen kunci dalam sistem penilaian. Guru memberikan umpan balik kepada siswa dalam sistem penilaian dapat memberi siswa informasi yang diperlukan untuk menutup kesenjangan antara kinerja mereka saat ini dan kinerja yang diinginkan (Demaidi dkk., 2018). Umpan balik hanya bisa efektif ketika pelajar memahami umpan balik tersebut dan bersedia serta mampu menindaklanjutinya. Pemberian umpan balik terdapat beberapa jenis yang dapat dilakukan oleh dosen / guru (Sychev dkk., 2020). Umpan balik pertama adalah Indikasi Umpan balik / *Knowledge of result* (KOR) berupa pemberian umpan balik jawaban benar atau salah. Umpan balik yang kedua adalah umpan balik koreksi /*Knowledge of correct response* (KCR) berupa pemberian umpan balik jawaban benar. Pemberian umpan balik ketiga adalah umpan balik penjelasan /*explanation feedback* (EF) berupa umpan balik mengenai alasan jawaban yang diberikan salah. Pemberian umpan balik keempat adalah umpan balik petunjuk / *Hint Feedback* (HF) berupa umpan balik yang memberikan solusi petunjuk untuk mendapatkan jawaban yang benar.

2.2.2 Evaluasi Pembelajaran

Evaluasi adalah kegiatan yang dilakukan berkenaan dengan proses untuk menentukan nilai dari suatu hal. Evaluasi merupakan hal yang tidak dapat dipisahkan dalam proses belajar mengajar, hal ini terjadi karena evaluasi merupakan proses untuk mengetahui tingkat pencapaian keberhasilan yang telah dicapai siswa atas bahan ajar atau materi yang disampaikan oleh guru. Guru biasanya memberikan evaluasi diakhir proses pembelajaran. Evaluasi yang bisa diberikan oleh guru berupa pilihan ganda dan uraian. Pilihan ganda merupakan jenis

soal yang menyediakan pilihan jawaban dan salah satu opsinya merupakan jawaban yang benar, sedangkan opsi lainnya berfungsi sebagai distraktor atau pengecoh. Soal tipe uraian adalah soal yang dibuat untuk mengetahui pemahaman siswa. Zainal (2017), jenis soal uraian yang diberikan oleh guru adalah *extended response* (tes uraian bentuk terbuka) adalah jawaban siswa bersifat terbuka, fleksibel, dan tidak berstruktur. Jenis tes uraian lain yang dapat diberikan oleh guru adalah *restricted response* (tes uraian terbatas) adalah jawaban siswa bersifat pasti dan terbatas.

2.2.3 *Data Augmentation*

Tantangan utama pada pengembangan model kecerdasan buatan adalah keterbatasan data yang berkualitas tinggi. *Data augmentation* adalah teknik yang digunakan untuk meningkatkan jumlah dan variasi data tanpa perlu pengumpulan tambahan, sehingga dapat meningkatkan generalisasi model (Zhou dkk, 2024).Penelitian dalam dunia NLP, metode konvensional seperti *back-translation*, *synonym replacement*, dan *random deletion* telah digunakan, tetapi pendekatan berbasis LLM seperti ChatGPT menawarkan solusi yang lebih fleksibel dan kontekstual (Ubani dkk., 2023). ChatGPT terbukti efektif dalam berbagai aplikasi, termasuk deteksi berita palsu, analisis sentimen, dan augmentasi teks medis. Penelitian terkait tugas untuk deteksi berita palsu, ChatGPT dapat digunakan untuk memperkaya dataset dengan variasi semantik yang serupa sehingga meningkatkan ketahanan model terhadap perbedaan gaya bahasa (Zhang dkk., 2024). Dalam analisis sentimen, teknik *zero-shot* dan *few-shot prompting* lebih unggul dibandingkan metode tradisional (Ubani dkk., 2023), dengan strategi *context-focused*, *aspect-focused*, dan *context–aspect augmentation* yang meningkatkan akurasi klasifikasi (Xu dkk., 2025).

Penelitian di bidang medis menunjukkan bahwa ChatGPT dapat digunakan untuk meningkatkan kualitas dan kuantitas data dalam analisis teks klinis (Latif dan Kim, 2024). Studi ini menggunakan ChatGPT untuk mereformulasi teks dalam dataset Clinical Health-Aware Reasoning across Dimensions (CHARDAT), menghasilkan variasi kalimat yang tetap mempertahankan makna asli namun lebih beragam. Dibandingkan dengan teknik augmentasi tradisional seperti *back-*

translation, metode berbasis ChatGPT lebih efektif dalam menjaga keselarasan konteks klinis dan mengurangi bias dalam model (Latif dan Kim, 2024). Eksperimen menunjukkan bahwa pendekatan berbasis ChatGPT dapat meningkatkan akurasi model NLP hingga 12.9% dibandingkan metode augmentasi tradisional (Yasser dkk., 2024). Selain itu, ChatGPT juga efektif dalam generasi teks panjang dengan mempertahankan struktur dan kohesi yang lebih baik dibandingkan model sebelumnya (Sawasaki dan Endo, 2024).

ChatGPT telah menjadi alat yang sangat efektif dalam augmentasi data NLP. Dengan teknik *prompting* yang tepat, model ini dapat meningkatkan performa model NLP dalam berbagai tugas, termasuk deteksi berita palsu, analisis sentimen, dan pemrosesan teks medis. Namun, optimalisasi lebih lanjut tetap diperlukan untuk memastikan kualitas dan efisiensi data yang dihasilkan (Zhou dkk., 2024).

2.2.3.1 Teknik Augmentasi pada data teks

Teknik augmentasi data untuk NLP perkembangan awalnya dimulai dari *rule based* atau yang sering dikenal EDA yang dikembangkan (Wei dan Zou, 2019). Teknik yang digunakan EDA untuk melakukan augmentasi data teks yaitu *synonym replacement, random insertion, random deletion, dan random swap.* Teknik ini memiliki kelebihan yaitu kemudahan untuk diterapkan (Shorten dkk., 2021). Akan tetapi teknik augmentasi menggunakan EDA ini memiliki kekurangan yaitu sering menghasilkan kalimat yang terdengar tidak natural atau bahkan merusak makna asli jika tidak dikontrol dengan baik.

Teknik augmentasi yang lain dengan cara *backtranslation* yang mulai memanfaatkan arsitektur *sequence-to-sequence* (seq2seq) pada *Neural Machine Translation* (NMT) yang dilakukan penelitian oleh (Sennrich dkk., 2016). Teknik ini menerjemahkan kalimat ke bahasa lain lalu menerjemahkannya kembali kemudian dihasilkan variasi struktur kalimat yang lebih natural dibanding metode berbasis aturan. Teknik *backtranslation* banyak digunakan karena terbukti efektif memperkaya variasi data, meskipun bergantung pada kualitas model terjemahan dua arah yang digunakan. Selanjutnya, perkembangan *pre-trained language* models (PLMs) seperti BERT mendorong munculnya contextual augmentation. Teknik ini memanfaatkan *Masked Language Model* untuk memprediksi kata atau frasa baru

berdasarkan konteks kalimat. Park dan Ahn (2019) mengusulkan pendekatan *Self-Supervised Contextual Data Augmentation* untuk NLP dengan memanfaatkan model *Masked Language Model* seperti BERT untuk menghasilkan variasi kalimat baru yang tetap relevan secara kontekstual. Teknik ini digunakan untuk memperkaya data latih pada tugas klasifikasi teks dan sentimen, sehingga membantu model memiliki generalisasi yang lebih baik. Teknik augmentasi ini dibandingkan dengan EDA, pendekatan ini menghasilkan variasi yang lebih kontekstual, tetapi memerlukan model besar dan komputasi tambahan.

Perkembangan selanjutnya di bidang NLP ditandai dengan hadirnya *Large Language Models* (LLMs) seperti GPT-3, ChatGPT, LLaMA, dan Gemini. LLMs membuka peluang baru untuk augmentasi data melalui strategi *prompt-based generation* yang tidak lagi bergantung pada skema *fine-tuning*. Hal ini sejalan dengan temuan (Liu dkk., 2023) yang menyebutkan bahwa paradigma 'pre-train, prompt, and predict' memungkinkan model pralatih digunakan secara langsung hanya dengan instruksi berbasis teks (*prompting*), tanpa pelatihan ulang yang mahal. Dengan pendekatan ini, pengguna dapat memanfaatkan *zero-shot prompting* (tanpa contoh) atau *few-shot prompting* (dengan beberapa contoh) kemudian akan menghasilkan sesuatu untuk augmentasi data dengan pendekatan pendekatan tersebut (Alsakran dan Alabduljabbar, 2024; Ubani dkk., 2023).

Pendekatan-pendekatan tersebut dapat menghasilkan variasi teks yang semantik lebih kaya dan natural melalui antarmuka percakapan. Penelitian ini berfokus pada *prompt based generation* yaitu *few shot prompting* yang memasukkan beberapa contoh kemudian mengeluarkan hasil yang telah dilakukan. Penyusunan atau pembuatan *prompting* merupakan suatu tantangan untuk melakukan augmentasi dikarenakan jika salah *prompting* maka hasil prompting akan gagal atau tidak memberikan dampak. Alsakran dan Alabduljabbar (2024) melakukan penelitian bahwa pembuatan *prompt* yang efektif pada LLM seperti ChatGPT harus memperhatikan prinsip *simplicity* dan *specificity*. Prinsip *simplicity* berarti *prompt* disusun dengan kalimat yang sederhana, jelas, ringkas, dan mudah dipahami oleh model sehingga meminimalkan ambiguitas dalam interpretasi instruksi. Sementara itu, prinsip *specificity* mengharuskan instruksi menjelaskan

secara rinci keluaran yang diharapkan, termasuk tujuan, format, dan batasan jika diperlukan. *Prompt* yang terlalu umum dapat membuat model menghasilkan jawaban yang melenceng dari konteks. Oleh karena itu, kombinasi instruksi yang sederhana namun spesifik dapat membantu LLM menghasilkan keluaran teks yang relevan, konsisten, dan sesuai kebutuhan pengguna.

Proses pembuatan prompting khususnya *few shot* adalah mengambil beberapa contoh dari *data training* kemudian meminta atau menuliskan prompt secara detail kemudian mengeluarkan hasil prompting (Dai dkk., 2025). Penelitian terbaru ini memanfaatkan ChatGPT untuk augmentasi data teks melalui teknik *few-shot prompting*. Inti prosesnya adalah peneliti mengambil beberapa contoh data pelatihan asli, kemudian memasukkannya ke dalam prompt agar ChatGPT menghasilkan kalimat baru yang tetap relevan secara makna tetapi bervariasi dalam gaya penulisan.

2.2.4 Pra-pemrosesan

Pra-pemrosesan adalah sebuah cara atau langkah untuk mengubah teks yang tidak terstruktur menjadi lebih terstruktur agar dapat mudah diolah (Kumar dkk., 2020). Pra- pemrosesan dalam *Natural Languange Processing* (NLP) terdapat beberapa proses yaitu *case folding, stopword removal, stemming,*tokenisasi. Pada penelitian ini, pra-pemrosesan data yang dilakukan adalah *case folding* dan tokenisasi. Pra pemrosesan seperti *stopword removal* dan *stemming* pada penelitian ini tidak digunakan untuk pra pemrosesan. Alasan tidak digunakan *stopword removal* karena tujuan dari *stopword removal* adalah menghapus kata kata yang tidak memiliki makna, jika penelitian ini melakukan *stopword removal* maka jawaban siswa menjadi hilang maknanya jika ada kata yang dihapus. Selain itu, *stemming* juga tidak digunakan karena *stemming* bertujuan untuk mengubah kata berimbuhan menjadi kata dasar hal ini dikhawatirkan akan menghilangkan makna teks. Pra Pemrosesan tokenisasi adalah proses memecah teks menjadi unit-unit lebih kecil yang disebut token (misalnya kata) sebagai representasi *input* yang dapat diproses oleh model (Singh, 2018). Tokenisasi dilakukan pada *input source* dan *target* dengan format sesuai yang dibutuhkan oleh *model*. Proses tokenisasi pada *input source* menghasilkan tensor input_ids dan attention_mask, sementara

tokenisasi target menghasilkan *output* berupa tensor label.

Pra-pemrosesan pada penelitian ini terdiri dari beberapa tahapan, yaitu *case folding* dan tokenisasi.

1. *Case Folding*

   Case Folding adalah suatu cara dalam *natural languange processing* (NLP) yang digunakan untuk melakukan standarisasi teks agar semua teks memiliki bentuk yang sama. Langkah yang dilakukan didalam normalisasi teks yaitu mengubah kata menjadi huruf kecil (*lowercase*), hal ini jika dilakukan akan mengurangi *noise* (Arora dan Kansal, 2019).

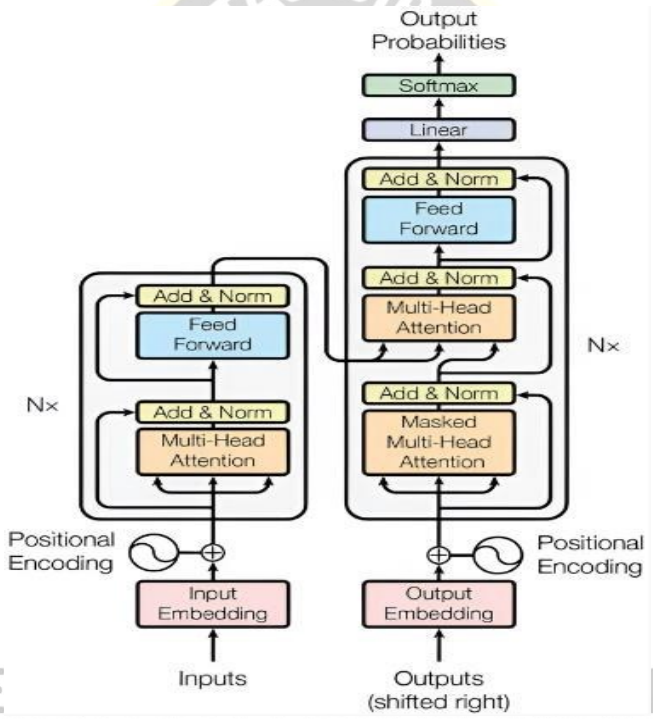
2. Tokenisasi

   Tokenisasi adalah proses memecah teks menjadi unit-unit lebih kecil. Tokenisasi dilakukan menggunakan *SentencePiece* yang dikembangkan (Kudo dan Richardson, 2018). *SentencePiece* merupakan perluasan dari dua algoritma segmentasi sub-kata, *byte pair coding* (BPE) dan *uni-gram language model* (ULM). *SentencePiece* memproses data teks mentah, memecahnya menjadi unit-unit subkata yang lebih kecil untuk menangani bahasa yang kompleks dan beragam secara lebih efektif. Proses ini dimulai dengan melatih model pada teks mentah, menggunakan algoritma *Byte Pair Encoding* (BPE) yang menggabungkan pasangan karakter yang paling sering muncul secara iteratif dan mengoptimalkan unit-unit subkata berdasarkan kemungkinan kemunculannya. Kelebihan dari tokenisasi menggunakan *SentencePiece* adalah tidak bergantung pada bentuk atau karakteristik bahasa karena melakukan pemisahan kata menurut frekuensi kemunculan tanpa pengetahuan sebelumnya tentang setiap bahasa (Choo dan Kim, 2023).

2.2.5 Arsitektur *Transformers*

Transformers merupakan arsitektur model *deep learning* yang dikembangkan oleh(Vaswani dkk., 2017). Model *transformers* memiliki tiga arsitektur dasar yaitu model *encoder,* model *decoder,* model *encoder-decoder*. Arsitektur dalam *transformers* menerapkan konsep *self-attention* yaitu model yang sebelumnya sudah dilatih dalam kumpulan teks berbahasa inggris standar tanpa adanya pengawasan sehingga dapat melakukan generalisasi dengan baik pada kumpulan

data yang ada sehingga menghasilkan hasil yang sesuai (Beseiso dkk., 2021). *Transformers* dapat memungkinkan menghindari hambatan representasi vektor antara *encoder* dan *decoder* sehingga model dapat mencari secara langsung *token* yang relevan dalam teks sumber ketika memprediksi *token* berikutnya untuk teks target (Ludwig dkk., 2021). *Encoder* terdiri dari N = 6 lapisan fungsional yang masing-masing terdiri dari dua sublapisan yakni *multi-head attention* dan *feed forward*. Sedangkan *decoder* memiliki N= 6 lapisan fungsional yang masing-masing terdiri dari tiga sublapisan yaitu *masked multi-head attention*, *multi-head attention*, dan *feed forward*. Gambar 2.1 merupakan arsitektur model *transformer* yang menunjukkan bagian kiri adalah *encoder* dan *decoder* untuk bagian kanan.



Sumber dari : Vaswani dkk.,2017

Gambar 2.1 Arsitektur Transformers (Vaswani dkk., 2017).

Encoder memiliki fungsi untuk membaca seluruh *input* teks secara bersamaan. Lapisan *encoder* ini berfungsi untuk menganalisis dan merepresentasikan urutan *input* sedemikian rupa sehingga model dapat dipahami. *Encoder* memproses urutan *input* dan menghasilkan representasi berkelanjutan atau penyematan (*embedding*) dari *input*. *Embedding* ini kemudian diteruskan ke *decoder* untuk menghasilkan urutan *output*. Kemudian *decoder* berfungsi

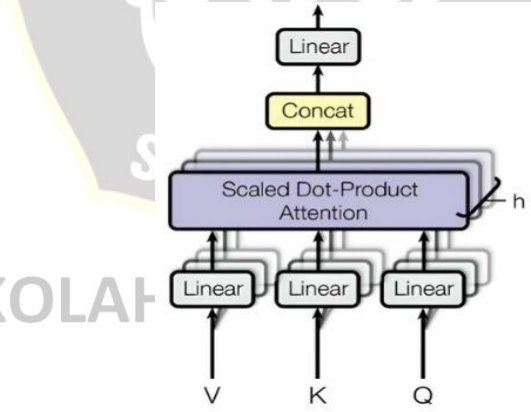
menghasilkan urutan keluaran berdasarkan keluaran *encoder* dan keluaran sebelumnya.

1. Lapisan Encoder

   Lapisan *encoder* ini terdiri dua lapisan yaitu *multi head attention* pada lapisan pertama kemudian *feed-forward* pada lapisan kedua.

   a. *Multi head attention*

      Arsitektur *transformer* terdapat *attention function* yang memiliki tugas untuk memetakan *vektor query, key, dan value* ke *output*. *Output* yang dihasilkan merupakan bobot penjumlah dari nilainya, bobot yang ditetapkan untuk setiap nilai dihitung oleh fungsi yang mencocokkan dari *query* dengan *key* yang tepat. *Multi head attention* dalam *transformers* digunakan untuk menghitung *attention* di waktu yang berbeda dengan bobot matriks yang berbeda dan menggabungkan hasilnya. Lapisan *multi-head attention* ini memungkinkan model untuk secara bersamaan memperhatikan informasi yang berasal dari sub-ruang representasi yang berbeda dengan posisi yang berbeda. Gambar 2.2 merupakan lapisan yang dimiliki *Multi-head attention*.



Sumber dari : Vaswani dkk.,2017

Gambar 2.2 Lapisan multi-head attention.

Penjelasan dari Gambar 2.2 yaitu huruf V yang merupakan matriks vektor *values,* dengan v_i menjadi *vektor valuer* tunggal yang terkait dengan satu kata *input.* Selanjutnya huruf K adala *keys,* dengan k_i menjadi *vektor key* tunggal terkait dengan satu kata *input.* Huruf Q merupakan matriks *vektor*

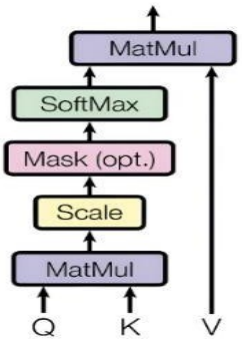
queries, dengan q_i menjadi *vektor query* tunggal yang terkait dengan satu kata *input*. Vektor-vektor ini menyimpan informasi kontekstual setiap kata. Langkah selanjtunya adalah menghitung skor kemiripan menggunakan vektor *query* dan *key*, sistem melakukan perhitungan jumlah tertimbang (*weighted sum*) dari vektor nilai tersebut. Bobot untuk masing-masing vektor nilai ditentukan oleh skor kesamaan, sehingga memastikan bahwa representasi kontekstual akhir lebih dipengaruhi oleh kata-kata yang relevan dalam kalimat. Perhitungan *multi-head attention* dapat direpresentasikan menggunakan rumus yang tercantum dalam Persamaan 2.1.

$$MultiHead(Q, K, V) = Concat(head_1, \dots, head_h)W^O$$

dengan $headi = Attention(QWi^Q, KWi^K, VWi^V)$ (2.1)

Multi-head attention, setiap bloknya menerapkan *scaled dot-product attention*. Input dari *scaled dot-product* berupa *query* dan *key* dari dimensi dk, serta *value* dari dimensi dv. *Scaled dot-product* ditunjukkan pada Gambar 2.3 *Scaled dot-product* menghitung *dot-product* antara *query* dan *key*. Hasil perhitungan tersebut akan diskalakan dengan $\sqrt{d_k}$, sehingga menghasilkan skor *attention*. Skor tersebut dimasukkan ke dalam fungsi *softmax* yang menghasilkan rangkaian *attention weight* (bobot perhatian) yang selanjutnya digunakan untuk menskalakan *value* melalui operasi perkalian berbobot. Secara keseluruhan, *scaled dot-product* menghitung *query*, *key*, dan *value* yang dikemas ke dalam matriks Q, K, dan V. Sistematis rumus *scaled dot-product* dapat dilihat pada Persamaan 2.2.

$$attention\ (Q, K, V) = softmax\ \left(\frac{QK^T}{\sqrt{d_k}}\right) V \quad (2.2)$$



Sumber dari : Vaswani dkk.,2017

Gambar 2.3 scaled dot-product attention.

b. Lapisan *feed forward network*

Lapisan *encoder* maupun *decoder* terdapat *feed forward network* yang diterapkan secara terpisah dan identik pada setiap posisi. *Feed forward network* ini menggunakan dua transformasi linear yang diantaranya terdapat ReLU (Vaswani dkk., 2017). Rumus persamaan *feed forward* dijelaskan pada persamaan 2.3.

$$FFN(x) = max(0, xW_1 + b_1)W_2 + b_2 \quad (2.3)$$

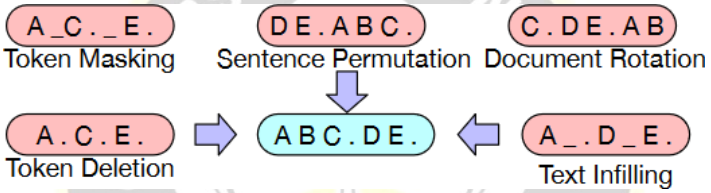
2. Lapisan *decoder*

Decoder disini nantinya adalah memiliki kemampuan untuk menghasilkan urutan *output* prediksi. Susunan N dari *decoder* sama seperti *encoder* yaitu berjumlah N=6. Setiap lapisan memiliki dua sub-lapisan yang dimiliki *encoder* juga dimiliki oleh *decoder* yakni *multi-head attention* dan *feed forward network*. Perbedaan lapisan *decoder* dengan lapisan *encoder* yaitu menambahkan satu lapisan baru yang bernama *masked multi head attention*. *Masked multi-head attention* untuk mencegah posisi agar tidak memperhatikan posisi berikutnya. Selanjutnya untuk mencegah sifat auto- regresif tersebut, diperlukan pencegahan aliran informasi ke arah kiri dalam *decoder*.

2.2.6 *Bidirectional and Auto Regressive from Transformers* (BART)

BART (*Bidirectional and Auto-Regressive from Transformer*) adalah model bahasa yang dilatih sebelumnya dengan menerapkan noise atau korupsi pada urutan input, dan selanjutnya model ditugaskan untuk merekonstruksi urutan masukan sebenarnya (Lewis dkk., 2019). Arsitektur model bahasa BART menggunakan

encoder pada BERT (Representasi Encoder Dua Arah dari Transformers) dan *decoder* pada GPT (*Generative Pre-Trained Transformer*) yang mampu melakukan tugas NLP di bentuk NLU dan NLG (Devlin dkk., 2019). BART untuk model dasar menggunakan 6 lapisan pada *encoder* dan *decoder* untuk model besar menggunakan masing-masing 12 lapisan(Lewis dkk., 2019). Model yang dilatih dengan arsitektur BART dikarenakan model bahasa yang dilatih dengan menerapkan *noise* atau korupsi pada urutan input. BART dalam proses pelatihan menggunakan beberapa teknik kebisingan dan beberapa teknik kebisingan baru untuk pra-pelatihan. Skema *noise* yang digunakan adalah *Token Masking, Token Deletion, Text Infilling, Sentence Permutation*, dan *Document* Rotation yang dijelaskan pada Gambar 2.4 (Lewis dkk., 2019). Penjelasannya adalah sebagai berikut:



Sumber dari : Lewis dkk., 2019

Gambar 2.4 Skema pelatihan BART.

1. *Token Masking*: Token acak dalam sebuah kalimat diganti dengan [MASK]. Model ini mempelajari cara memprediksi token tunggal berdasarkan rangkaian lainnya.
2. *Token Deletion*: Token acak dihapus. Model harus belajar memprediksi konten token dan menemukan posisi asal token tersebut dihapus.
3. *Text Infilling*: Sejumlah token yang berdekatan akan dihapus dan diganti dengan satu token [MASK]. Model harus mempelajari konten token yang hilang dan jumlah token.
4. *Sentence Permutation*: Kalimat (dipisahkan dengan titik) diurutkan secara acak. Hal ini membantu model mempelajari isi kalimat yang logis.
5. *Document Rotation*: Dokumen disusun ulang untuk memulai dengan token acak. Konten sebelum token ditambahkan di akhir dokumen. Hal ini memberikan wawasan tentang cara dokumen biasanya disusun dan bentuk atau akhir dokumen.

Model yang dilatih dengan arsitektur dapat disimpan menjadi *pre-trained model*. Model *pre-trained* adalah sebuah model yang dilatih dalam korpus besar yang dapat mempelajari representasi bahasa universal yang bermanfaat untuk Tugas *Natural Languange Processing* (NLP) dan dapat menghindari pelatihan model baru dari awal (Qiu dkk., 2020). *Pre-trained* sering digunakan untuk *transfer learning* dikarenakan *transfer learning* sering kali memanfaatkan model yang sudah dilatih tersebut untuk diterapkan pada kasus yang baru dan memiliki kemiripan dengan kasus yang sudah dilatih tersebut (Zhuang dkk., 2021). *Pre-trained* model dalam *transfer learning* penggunaannya Terdapat dua pendekatan yakni ekstrasi fitur dan *fine tuning. Fine tuning* dilakukan dengan cara menggunakan model *pre-trained* dengan menyesuaikan dan melatih ulang bagian-bagiannya (termasuk lapisan-lapisan internal) dengan data baru untuk meningkatkan kinerja pada tugas spesifik menggunakan model BART dari library Transformers yang disediakan oleh *HuggingFace*. Ekstraksi fitur yaitu menggunakan representasi fitur yang sudah ada dari model pre- trained dengan menambahkan lapisan di atasnya, tanpa mengubah model *pre-trained*.

2.2.7 *Text-To-Text Transfer Transformer* (T5)

Model T5 (Text-to-Text Transfer Transformer) adalah model bahasa berbasis arsitektur transformer yang dikembangkan oleh Google Research (Raffel dkk., 2020). T5 memiliki keunggulan yaitu terletak pada pendekatannya yang seragam untuk semua jenis tugas pemrosesan bahasa alami (Natural Language Processing/NLP), baik klasifikasi teks, ekstraksi informasi, terjemahan, rangkuman, penjawaban pertanyaan, maupun tugas-tugas generatif lainnya. Model T5 ini dapat diformulasikan ke dalam format yang sama yaitu konversi teks ke teks (text-to-text). Model akan menerima input berupa string teks dan menghasilkan output berupa string teks juga sehingga memungkinkan satu model tunggal menangani berbagai macam tugas NLP tanpa perlu modifikasi arsitektur atau fungsi khusus untuk setiap tugas. Misalnya, untuk tugas klasifikasi sentimen, input bisa berupa “sentiment analysis: Film ini sangat bagus”, dan outputnya adalah “positif”.

Model ini menggunakan arsitektur transformer encoder- decoder yang terdiri dari dua komponen utama yaitu encoder untuk memproses input teks dan menghasilkan representasi tersembunyi. Decoder untuk menghasilkan output teks berdasarkan representasi tersebut.

Proses pelatihan awal (pretraining) T5 dilakukan pada korpus data berskala besar disebut C4 (Colossal Clean Crawled Corpus) yang merupakan hasil pembersihan dan penyaringan dari data web publik. Pada tahap pretraining, T5 dilatih dengan menggunakan metode "span corruption" yaitu metode yang menyembunyikan (masking) rentang kata dalam input, lalu model diminta untuk memprediksi kata-kata yang disembunyikan. Pendekatan ini berbeda dari teknik pretraining pada model lain seperti BERT yang hanya melakukan masking pada token individual. Setelah tahap pretraining, T5 dapat disesuaikan (fine-tuned) untuk tugas-tugas spesifik dengan menggunakan dataset berlabel yang sesuai. Hal ini memungkinkan pemanfaatan kemampuan generalisasi yang diperoleh dari pretraining ke dalam berbagai aplikasi nyata. T5 juga tersedia dalam berbagai ukuran model mulai dari T5-Small hingga T5-XXL yang disesuaikan dengan kapasitas komputasi dan kompleksitas tugas yang dihadapi. Pendekatan text-to-text yang digunakan oleh T5 memberikan fleksibilitas yang tinggi karena format input dan output yang konsisten membuat integrasi dan pelatihan multitugas menjadi lebih efisien.

2.2.8 Proses Pelatihan BART dan T5

Pelatihan BART dan T5 dilakukan dengan menggunakan arsitektur encoder–decoder berbasis Transformer. Pada tahap awal, teks sumber (input sequence) dan teks target (output sequence) terlebih dahulu melalui proses tokenisasi; BART menggunakan Byte-Pair Encoding (BPE) sedangkan T5 menggunakan SentencePiece. Tokenisasi ini mengubah kalimat menjadi token-token terpisah yang kemudian dipetakan ke dalam ID sesuai dengan vocabulary yang dimiliki model. Selain token utama hasil pemetaan, BART dan T5 juga menambahkan token khusus seperti <s> (start), </s> (end), dan <pad> (padding) untuk menandai awal, akhir, dan pengisian urutan. Misalnya, kalimat "Saya belajar di kampus." setelah ditokenisasi dapat direpresentasikan menjadi [<s>, 3021, 785, 45, 2299, </s>].

Setelah tokenisasi, token-token tersebut diubah menjadi tensor agar dapat diproses oleh model. Tensor yang dihasilkan terdiri dari tiga bagian utama: input_ids, yaitu indeks numerik token sesuai vocabulary; attention_mask, yaitu penanda dengan nilai 1 untuk token yang diperhatikan model dan 0 untuk token padding yang diabaikan; serta labels, yaitu tensor target yang merepresentasikan urutan jawaban dalam bentuk indeks token. Label ini digunakan untuk menghitung loss selama proses pelatihan (Rojan dkk., 2020; Bogdanović dkk., 2024). Selanjutnya, setiap token ID diubah menjadi vektor kontinu melalui embedding layer. Setiap token ID mengambil baris tertentu dari matriks embedding yang ukurannya [vocab_size × hidden_dim]. Nilai-nilai dalam vektor ini, misalnya [0.42, -0.23, 0.11, 0.56, …], bukan merupakan makna literal per dimensi, tetapi representasi numerik yang mengandung informasi semantik token dalam konteks kalimat. Matriks embedding ini diperbarui selama pelatihan sehingga model belajar merepresentasikan hubungan semantik dan sintaksis antar-token. Agar model mengetahui posisi setiap token dalam urutan, BART dan T5 juga menambahkan absolute positional embedding, sehingga representasi akhir tensor memiliki dimensi [batch_size, sequence_length, hidden_dim] sebelum masuk ke encoder.

Encoder BART dan T5 terdiri atas beberapa lapisan Transformer yang setiap lapisan memiliki dua komponen inti: multi-head self-attention dan feed-forward network (FFN). Mekanisme self-attention memungkinkan model mempelajari hubungan antar-token dalam input secara dua arah, sehingga konteks kata dapat dipahami lebih baik. Secara matematis, perhatian ini dihitung dengan persamaan yang sudah dijelaskan 2.2. Setelah melalui self-attention, hasilnya diteruskan ke FFN, yaitu jaringan dua lapis linear dengan aktivasi non-linear GELU. Untuk menjaga kestabilan pelatihan, setiap sub-lapisan dilengkapi dengan residual connection serta layer normalization. Representasi akhir yang dihasilkan encoder merupakan representasi kontekstual dari keseluruhan input sequence. Proses berlanjut ke decoder yang juga terdiri atas beberapa lapisan Transformer, tetapi memiliki tiga blok utama. Pertama, masked multi-head self-attention digunakan untuk memproses token output secara autoregresif, kemudian causal mask memastikan bahwa prediksi suatu token hanya bergantung pada token sebelumnya,

bukan token di masa depan. Kedua, cross-attention menghubungkan representasi dari encoder dengan decoder, sehingga prediksi output mempertimbangkan informasi penuh dari input sequence. Ketiga, hasil dari cross-attention diteruskan ke FFN dan dinormalisasi menggunakan mekanisme yang sama dengan encoder.Keluaran dari lapisan terakhir decoder kemudian diproyeksikan ke dimensi vocabulary melalui linear layer atau language modeling head. Nilai logits yang dihasilkan diubah menjadi distribusi probabilitas menggunakan fungsi softmax. Model kemudian dibandingkan dengan target output menggunakan Cross-Entropy Loss, dengan token <pad> diabaikan melalui parameter ignore_index untuk memastikan loss hanya dihitung pada token bermakna.

Selama pelatihan, *backpropagation* digunakan untuk menghitung gradien dari fungsi loss terhadap parameter model. Gradien ini kemudian digunakan untuk memperbarui bobot melalui AdamW optimizer yang dilengkapi dengan weight decay untuk mengendalikan overfitting. Beberapa teknik tambahan juga diterapkan untuk meningkatkan stabilitas pelatihan, seperti learning rate scheduler dengan linear decay dan warmup steps, dropout untuk mencegah overfitting, serta gradient clipping untuk mencegah exploding gradient.

Tahap validasi, model menghasilkan output tanpa menggunakan *teacher forcing*. *Decoder* memprediksi token demi token secara autoregresif, dengan strategi decoding seperti greedy search, beam search, atau sampling. Hasil keluaran dibandingkan dengan teks target referensi, dan performa model diukur menggunakan metrik evaluasi otomatis, misalnya ROUGE untuk kesamaan n-gram dan BERTScore untuk kesamaan semantik.

2.2.9 Perbedaan BART dan T5

BART dan T5 sama-sama menggunakan arsitektur encoder–decoder berbasis Transformer, tetapi memiliki strategi pre-training yang berbeda. BART dilatih dengan *text denoising objective*, yaitu teks input sengaja diberi gangguan seperti penghapusan token, pengacakan urutan kata, atau masking, dan model dilatih untuk meregenerasi teks asli. Misalnya, kalimat “Saya pergi ke pasar untuk membeli sayuran segar” bisa diacak menjadi “pergi <mask> membeli segar sayuran ke pasar”, dan BART harus mengembalikannya ke bentuk asli. Pendekatan ini

memungkinkan encoder menangkap konteks dua arah, sementara decoder menghasilkan teks secara autoregresif, sehingga BART sangat efektif untuk tugas generatif murni seperti summarization atau text reconstruction. Sebaliknya, T5 menggunakan *span corruption objective*, yaitu beberapa span kata dihapus dan model diminta untuk memprediksi teks yang hilang. Misalnya, dari kalimat yang sama, T5 dapat memasking beberapa span menjadi "Saya pergi ke <mask> untuk membeli <mask>", dengan target output "pasar sayuran segar". Dengan cara ini, T5 belajar memahami konteks global dan dapat diaplikasikan ke berbagai tugas NLP dengan format *text-to-text*, termasuk *classification, translation*, dan *question answering*. Perbedaan utama antara keduanya terletak pada metode pre-training: BART fokus pada pemulihan teks dari input yang rusak, sedangkan T5 menekankan prediksi teks yang hilang dalam span untuk fleksibilitas multitasking.

2.2.10 *Hyperparameter*

Hyperparameter adalah Proses merancang arsitektur model ideal dengan konfigurasi (Yang dan Shami, 2020). Menyesuaikan *hyperparameter* dianggap sebagai komponen kunci dalam membangun model *Machine Learning* (ML) yang efektif, terutama untuk model ML dan *deep neural network* yang memiliki banyak *hyperparameter* (Yang dan Shami, 2020). Pemilihan nilai *hyperparameter* yang tepat dapat mempengaruhi performa model secara signifikan. Berikut beberapa *hyperparameter* yang digunakan pada penelitian ini:

1. *Learning Rate*

   Learning rate merupakan salah satu parameter *training* yang berfungsi untuk menghitung nilai laju pada saat proses pembealajaran (Astria dkk., 2022). Nilai *learning rate* berkisar dari nol hingga satu. Semakin besar nilai *learning rate* maka proses *training* akan berjalan semakin cepat, tetapi dapat mengalami kegagalan dalam memperoleh nilai *optimum*. Semakin kecil nilai *learning rate*, model akan mendapatkan *loss* yang kecil, tetapi waktu dalam proses *training* akan semakin lambat (Yuliany dkk., 2022).

2. *Batch Size*

   Batch size merupakan banyaknya data yang digunakan dalam setiap epoch untuk melatih jaringan. Semakin kecil nilai *batch size* yang digunakan, maka

proses pelatihan akan semakin cepat. *Batch size* semakin besar maka proses *training* akan memakan waktu yang cukup lama karena membutuhkan kapasitas penyimpanan yang lebih banyak (Kandel dan Castelli, 2020).

3. *Weight decay*

   Teknik regularisasi yang menambahkan penalti L2 pada fungsi kerugian untuk membatasi bobot model agar tidak terlalu besar. Tujuannya adalah mencegah overfitting dan meningkatkan generalisasi. Secara matematis, bobot besar diperkecil secara bertahap melalui penambahan suku penalti. Penelitian terkait penambahan *weight decay* membuktikan teknik ini efektif menjaga kapasitas model tetap sederhana (Hinton dan van Camp, 1993; Krogh dan Hertz, 1991). Pendekatan adaptif terbaru menunjukkan *weight decay* mampu meningkatkan akurasi di berbagai dataset (Nakamura dan Hong, 2019).

Penelitian-penelitian terdahulu seringkali memanfaatkan *hyperparameter settings* seperti *learning rate*, *batch size*, dan *weight decay* untuk merancang arsitektur model yang ideal. Dengan adanya pemilihan nilai hyperparameter yang tepat, performa model dapat meningkat secara signifikan. Sebagai contoh, Adnan dkk., (2024) melakukan fine-tuning BART untuk koreksi tata bahasa dalam berita olahraga Indonesia dengan *learning rate* 2e-5, *batch size* 16, dan 3 *epoch*, yang menghasilkan skor BLEU 0.8560 dan GLEU 0.9655, menunjukkan efisiensi pembelajaran dan generalisasi yang baik. (Ni dkk., (2025) menerapkan BART untuk penerjemahan Mon-Inggris dan menekankan pentingnya penyesuaian *learning rate* dan *batch size*, bahkan dalam kondisi *low-resource*. Sementara itu, Bharathi Mohan dkk, (2023) menggunakan BART untuk tugas *abstractive summarization* dengan *learning rate* 0.01 dan *batch size* 32, dan menunjukkan bahwa tuning hyperparameter berdampak langsung pada kualitas ringkasan yang dihasilkan.

2.2.11 *Optuna Hyperparameter Framework*

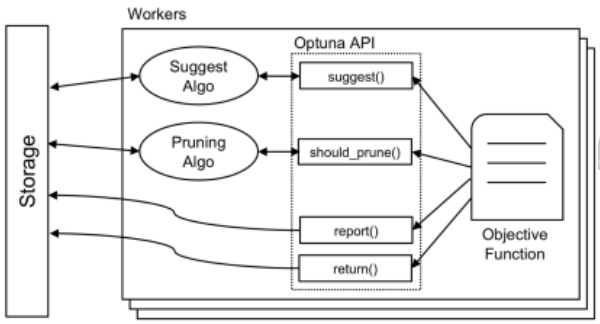
Optuna adalah *framework open-source* untuk optimasi hyperparameter yang lebih efisien dibandingkan metode tradisional seperti grid search dan random search (Akiba dkk., 2019). Optuna menggunakan algoritma *Tree-structured Parzen Estimator (TPE)*, sebuah pendekatan dalam *Bayesian Optimization*. Berbeda

dengan *Gaussian Process* (GP) yang mengasumsikan distribusi Gaussian, TPE membagi distribusi ke dalam dua model *Gaussian Mixture Model* (GMM): satu untuk nilai hyperparameter yang menghasilkan hasil terbaik, dan satu lagi untuk nilai lainnya (Shekhar dkk., 2021). Optuna memilih nilai yang memaksimalkan rasio antara kedua model ini untuk meningkatkan efisiensi pencarian. Optuna tidak hanya mempercepat proses pencarian *hyperparameter* tetapi juga meningkatkan peluang menemukan konfigurasi optimal dalam waktu yang lebih singkat. Dengan pendekatan *trial-based optimization*, Optuna secara otomatis mengeksplorasi dan memilih kombinasi hyperparameter optimal untuk meminimalkan loss function atau meningkatkan akurasi model. Keunggulannya terletak pada pencarian dinamis yang menyesuaikan strategi berdasarkan hasil trial sebelumnya sehingga lebih cepat dan adaptif.

Proses optimasi TPE dalam pencarian hyperparameter terbaik dimulai dengan fase eksplorasi awal melalui *random sampling* untuk mengumpulkan data dasar mengenai performa model. Setelah sejumlah trial awal, TPE mengurutkan hasil berdasarkan nilai *objective function* dan membagi data menjadi dua grup: trial dengan performa terbaik (25% teratas) dan trial dengan performa rendah (75% sisanya). TPE kemudian membangun dua distribusi probabilitas menggunakan *Parzen Estimator* untuk memodelkan karakteristik hyperparameter yang menghasilkan performa baik dan buruk. Langkah berikutnya menentukan hyperparameter kandidat selanjutnya, TPE menggunakan *Expected Improvement* (EI) yang dihitung sebagai rasio antara kedua distribusi probabilitas tersebut. TPE kemudian membangun dua distribusi probabilitas menggunakan *Parzen Estimator*,satu untuk karakteristik hyperparameter di grup "bagus" ($\ell(x)$) dan satu lagi untuk grup "kurang bagus" ($g(x)$). TPE kemudian mengevaluasi rasio antara kedua model probabilitas tersebut. *Hyperparameter* yang menunjukkan probabilitas tinggi pada distribusi performa bagus namun rendah pada distribusi performa kurang bagus dipilih sebagai kandidat optimal untuk eksperimen selanjutnya. Setelah eksperimen dijalankan, hasilnya ditambahkan ke dalam riwayat percobaan kemudian kedua model probabilitas diperbarui dan pemilihan ini diulang terus-menerus sampai tercapai jumlah trial maksimum atau tidak ada

peningkatan signifikan.

Optimisasi hyperparameter dengan Optuna dimulai dengan pembentukan sebuah study yang menjadi wadah seluruh percobaan (trials). Setiap trial didefinisikan dalam objective function yang menentukan ruang pencarian hyperparameter melalui pemanggilan trial.suggest_*(). Optuna mendukung berbagai algoritma sampling, mulai dari TPE untuk eksplorasi awal, Covariance Matrix Adaptation Evolution Strategy (CMA-ES) untuk mengeksploitasi wilayah parameter yang lebih menjanjikan, hingga Gaussian Process Bayesian Optimization (GP-BO) untuk evaluasi model yang mahal. Nilai hyperparameter yang dipilih digunakan untuk melatih model, sementara performa dilaporkan secara berkala melalui trial.report(). Informasi ini kemudian dievaluasi oleh algoritma pruning seperti Asynchronous Successive Halving Algorithm (ASHA), yang memutuskan apakah sebuah trial dilanjutkan atau dihentikan lebih awal dengan trial.should_prune(). Mekanisme ini mencegah pemborosan sumber daya pada trial yang berkinerja buruk, sehingga lebih banyak percobaan menjanjikan dapat dieksplorasi dalam waktu terbatas. Seluruh informasi, mulai dari hyperparameter, hasil sementara, hingga skor akhir, disimpan dalam storage terpusat (misalnya SQLite atau MySQL), yang memungkinkan pembaruan strategi pencarian berdasarkan riwayat trial sebelumnya.



Sumber dari : Akiba dkk., 2019

Gambar 2. 5 Optuna *System Design*

Gambar 2.5 merupakan Optuna *system design* yang terdapat beberapa komponen inti yang saling terhubung. Workers menjalankan trial secara paralel, Suggest Algorithm menghasilkan kombinasi hyperparameter baru, dan Objective

Function melaporkan hasil sementara melalui API report(). Informasi ini dievaluasi oleh Pruning Algorithm (misalnya ASHA) untuk menentukan kelanjutan trial, sementara semua data disimpan dalam storage agar dapat dimanfaatkan kembali oleh algoritma sampling maupun worker lain secara terdistribusi. Dengan kombinasi strategi sampling yang adaptif dan pruning yang agresif, Optuna tidak hanya mempercepat pencarian hyperparameter tetapi juga meningkatkan peluang menemukan konfigurasi optimal dalam waktu singkat. Hasil akhirnya dirangkum dalam study.best_params, yaitu kombinasi *hyperparameter* terbaik, serta study.best_value, yaitu performa tertinggi yang dicapai.

2.2.12 Inferensi

Inferensi merupakan suatu proses untuk mendapatkan prediksi data baru yang berasal dari proses pelatihan (X. Xu dkk., 2018). Proses pemilihan token *output* untuk menghasilkan teks dikenal sebagai *decoding*. Algoritma yang dapat digunakan dalam tahap inferensi untuk melakukan prediksi adalah *greedy search*, *beam search*. *Beam search* adalah algoritma yang sering digunakan. *Beam search* ini menghasilkan urutan token satu per satu dengan mempertahankan sejumlah kandidat aktif (*beam size*) pada setiap langkah (Bausch dkk., 2021). Pada penelitian ini, teknik *generation* yang digunakan adalah *beam search*. Berikut merupakan langkah *beam search* untuk bisa menghasilkan kalimat prediksi:

1. Mulai dari *node* akar, memilih *n* kata dengan probabilitas tertinggi dengan *n* adalah ukuran *beam size* yang telah ditentukan. Untuk setiap kata, buat *node decoder* baru.
2. Untuk setiap *n node decoder* yang dibuat, yang dipertimbangkan hanya *n* kemungkinan kata dengan probabilitas yang dipertimbangkan untuk *output* berikutnya
3. Selanjutnya adalah algoritma akan memfilter *n node* dengan fungsi nilai tertinggi hanya *n node* yang tersisa di dalam daftar *node*.
4. Ulangi langkah 2 dan 3 hingga mencapai maksimum kedalaman atau maksimum panjang kalimat. Apabila algoritma telah menemukan token *end-of-sentence* (EOS), maka *sequence* tersebut akan dimasukkan ke dalam daftar jawaban.

5. Dari daftar *node*, akan dipilih *sequence* dengan nilai fungsi objektif tertinggi sebagai jawaban akhir.

2.2.13 *ROUGE (Recall-Oriented Understudy for Gisting Evaluation)*

ROUGE (Recall-Oriented Understudy for Gisting Evalua-tion) adalah salah satu jenis evaluasi dalam *text generation* yaitu *content-overlap metric* (Lin, 2004). ROUGE sendiri secara umum yaitu memeriksa banyaknya kata atau n- gram dalam teks referensi yang muncul di teks yang dihasilkan. Biasanya ROUGE digunakan untuk mengevaluasi peringkasan teks otomatis dan terjemahan mesin. *ROUGE* bekerja dengan membandingkan ringkasan atau terjemahan yang dibuat secara otomatis dengan sekumpulan ringkasan referensi (biasanya buatan manusia). *ROUGE* dalam prosesnya yaitu melakukan perhitungan *Recall, Precission, F1-score* yang membandingkan referensi sistem dengan referensi manusia. ROUGE terdapat persamaan rumus sebagai berikut yang dijelaskan dalam persamaan 2.4:

$$ROUGE - N = \frac{banyaknya\ kata\ yang\ sama\ pada\ referensi\ dan\ kandidat}{total\ kata\ yang\ ada\ di\ model\ atau\ referensi} \quad (2.4)$$

Dengan N adalah panjang n-gram dan countmatch($ngram_n$) nilai maksimum dari ngram yang ditentukan yang dihitung berdasarkan referensi yang ditentukan. Nilai n ditentukan berdasarkan dua hal:

1. ROUGE - 1 adalah suatu unit yang dibandingkan antara sistem dan referensi berdasarkan 1-gram atau *unigram* (per kata).
2. ROUGE – 2 adalah suatu unit yang dibandingkan antara sistem dan referensi berdasarkan 2-gram atau *bigram* (per kata).

Rumus dari perhitungan *precission,recall* dan *f1-score* untuk ROUGE-1 dan ROUGE-2 pada persamaan 2.5 sampai 2.9 berikut.

$$ROUGE - 1\ Recall = \frac{banyak\ kata\ yang\ cocok\ (unigram)}{total\ kata\ di\ jawaban\ referensi} \quad (2.5)$$

$$ROUGE - 1\ Precision = \frac{banyak\ kata\ yang\ cocok\ (unigram)}{total\ kata\ di\ jawaban\ model} \quad (2.6)$$

$$ROUGE - 2\ Recall = \frac{banyak\ kata\ yang\ cocok\ (bigram)}{total\ kata\ di\ jawaban\ referensi} \quad (2.7)$$

$$ROUGE - 2\ Precision = \frac{banyak\ kata\ yang\ cocok\ (bigram)}{total\ kata\ di\ jawaban\ model} \quad (2.8)$$

$$ROUGE\ f1 - score = 2 \times \frac{precision \times recall}{precision + recall} \quad (2.9)$$

Perhitungan ROUGE selain ROUGE-N adapun perhitungan ROUGE yang lain salah satunya adalah ROUGE-L yang mengukur urutan pencocokan kata terpanjang menggunakan *Longest Common Subsequence* (LCS) yang secara otomatis dapat menyertakan *n-gram* dalam urutan terpanjang. Persamaan 2.10, 2.11 dan 2.12 berikut adalah rumusnya:

$$Rouge - L\ Recall = \frac{Lcs\ (model,referensi)}{total\ kata\ di\ jawaban\ referensi} \quad (2.10)$$

$$Rouge - L\ precission = \frac{Lcs\ (model,referensi)}{total\ kata\ di\ jawaban\ kandidat} \quad (2.11)$$

$$\mathrm{R}ouge - L\ F1 = \frac{(1+\beta^2)R_{lcs}P_{lcs}}{R_{lcs}+\beta^2\ P_{lcs}} \quad (2.12)$$

Keterangan :

LCS (referensi,kandidat) : Panjang *longest common subsequence* dari jawaban referensi dan jawaban kandidat.

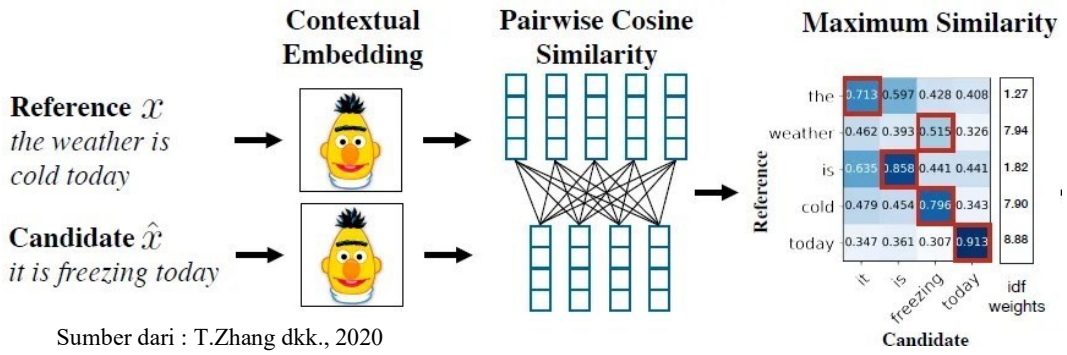
R_{lcs}: $Nilai\ recall\ dari$ longest common subsequence.

P_{lcs}: $Nilai\ precission\ dari$ longest common subsequence.

β: parameter yang mengontrol pengaruh *recall* dan *precison* pada skor akhir Nilai yang digunakan adalah $\beta = 1$

2.2.14 *BertScore*

BertScore merupakan jenis evaluasi *model based metric* yang digunakan untuk mengevaluasi hasil dari sebuah *text generation.* Bertscore menggunakan representasi kata dan kalimat yang dipelajari untuk menghitung kesamaan semantik antara teks yang dihasilkan dan teks referensi (Li dkk., 2021). BERTScore adalah metrik berbasis model yang diusulkan (T. Zhang dkk., 2020). Keunggulan dari Bertscore ini adalah memperhitungkan keragaman leksikal dan komposisi yang menjaga makna. Bertscore mengatasi dua kendala umum dalam metrik berbasis n-gram. Pertama, metode seperti itu sering kali gagal mencocokkan parafrase dengan tepat. Kedua, model n-gram gagal menangkap ketergantungan yang jauh dan menghukum perubahan pengurutan yang kritis secara semantik (T. Zhang dkk., 2020).



Sumber dari : T.Zhang dkk., 2020

Gambar 2.6 Proses Bertscore(Zhang dkk., 2020).

Gambar 2.6 merupakan proses atau cara perhitungan Bertscore berikut merupakan penjelasan dan langkah-langkah perhitungan Bertscore hingga mendapatkan nilai *precission,recall* dan *f1-score.*

1. Menyiapkan terlebih dahulu kalimat yang akan dibandingkan yaitu referensi dan model
2. Kedua kalimat hasil referensi dan model dilakukan tokenisasi
3. Setelah kedua kalimat telah dilakukan tokenisasi, selanjutnya yaitu melakukan perbandingan hubungan *similarity* setiap kata yang telah ditokenisasi. Perhitungannya menggunakan rumus *similarity* yang dijelaskan pada persamaan 2.13

$$similarity = \frac{x_i^T \hat{x}_j}{\|x_i\| \|\hat{x}_j\|} \quad (2.13)$$

Keterangan :

$\boldsymbol{x_i}$= vektor embedding token dalam referensi

$\hat{\boldsymbol{x}}_{\boldsymbol{j}}$= vektor embedding token dalam kandidat

$\boldsymbol{x_i^T \hat{x}_j}$=hasil perkalian dot product antara kedua vektor

$\|\boldsymbol{x_i}\|$ = Norma (panjang) dari *vektor* x_i

$\|\hat{\boldsymbol{x}}_{\boldsymbol{j}}\|$ = Norma (panjang) dari *vektor* $\hat{X}_J$

4. Setelah menghitung setiap hubungan kata yang sudah dilakukan *similarity* selanjutnya memilih *score* hubungan kata kata yang memiliki *similarity* yang tertinggi.
5. Setelah memilih dan mengumpulkan *similarity* yang tertinggi langkah selanjutnya yaitu melakukan perhitungan *precission,recall, f1-score.* rumus

persamaan *precission, recall, f1-score* sebagai berikut bisa dilihat persamaan 2.14, 2.15 dan 2.16.

$$Bertscore\ Recall = \frac{1}{|x|}\sum_{x_i \in x} max\ \boldsymbol{x_i^T x_j} \quad (2.14)$$

$$Bertscore\ \text{Precission} = \frac{1}{|\hat{x}|}\sum_{\hat{x}_j \in \hat{x}} max\ \boldsymbol{x_i^T x_j} \quad (2.15)$$

$$Bertscore\ \text{F1} = 2\frac{P_{bert}\ x\ R_{bert}}{P_{bert} + R_{bert}} \quad (2.16)$$

Keterangan :

$\boldsymbol{x}$: Sekuens token pada kalimat referensi.

$\hat{\boldsymbol{x}}$: Sekuens token pada kalimat kandidat.

$\boldsymbol{x_i}$: Token ke-i pada kalimat referensi

$\hat{\boldsymbol{x}}_{\boldsymbol{j}}$: Token ke-j pada kalimat kandidat.

$\boldsymbol{x_i^T \hat{x}_j}$: $Dot\ product\ antara\ x_i\ dan\ \hat{x}_j$

$max\ \boldsymbol{x_i^T \hat{x}_j}$: nilai tertinggi dari dot product

$|\boldsymbol{x}||\hat{\boldsymbol{x}}|$: Banyaknya token pada kandidat dan referensi.

2.2.15 *Human Centered Evaluation*

Human Centered Evaluation adalah meminta penilai manusia untuk menilai kualitas teks yang dihasilkan berdasarkan beberapa dimensi tertentu (misalnya, kejelasan dan ketepatan) dalam hasil feedback yang dihasilkan oleh sistem. Proses dalam melakukan *human centered evaluation*, biasanya dilakukan melalui pemberian kuesioner kepada dosen ataupun ahli untuk mengecek kualitas hasil *feedback* yang dihasilkan oleh sistem. Biasanya nantinya dibuatkan sebuah kuesioner dan diserahkan kepada orang atau ahli dibidangnya untuk memberikan nilai menggunakan skala likert 1-5 atau 1-7. Dalam *dataset* yang besar bisa dilakukan *random sampling* untuk melakukan pengambilan sebagian data dan untuk mendapatkan jumlah data bisa dilakukan dengan rumus slovin dalam menentukan jumlah data yang efektif atau optimal. Setelah menyusun data,membuat kuesioner dan ahli telah mengisi kuesioner untuk memproses score yang telah diberikan ahli bisa dilakukan uji reliabilitas yaitu *Inter Corellation Coeficient* (ICC).

2.2.15.1 Rumus Slovin

Pengambilan sampel dari populasi yang besar perlu dilakukan dengan

mempertimbangkan efisiensi dan keterbatasan sumber daya. Rumus Slovin digunakan sebagai metode praktis untuk menentukan jumlah sampel minimum yang diperlukan dari suatu populasi agar hasil penelitian tetap valid secara statistik. Rumus ini dinyatakan sebagai berikut yang bisa dilihat pada persamaan 2.17:

$$n = \frac{N}{1+N\ x\ e^2} \tag{2.17}$$

Keterangan :

n= ukuran sampel

N=Jumlah populasi

e= *margin of error* (10 %)

Slovin memberikan estimasi sederhana ketika variabilitas data atau standar deviasi populasi tidak diketahui, sehingga cocok digunakan pada tahap awal perencanaan penelitian.. Antoro (2024) menyatakan bahwa penerapan Slovin perlu disesuaikan dengan tujuan penelitian dan karakteristik data karena estimasi yang dihasilkan dapat kurang akurat bila populasi sangat bervariasi. Oleh karena itu, meskipun bersifat praktis, penggunaan rumus Slovin tetap membutuhkan pertimbangan metodologis yang tepat agar hasil penelitian tidak bias.

2.2.15.2 Inter corellation coeficient (ICC)

Intraclass Correlation Coefficient (ICC) merupakan ukuran statistik yang digunakan untuk menilai reliabilitas atau konsistensi antar penilai (inter-rater reliability) maupun antar pengukuran (test-retest) terhadap objek yang sama dalam konteks data kuantitatif berskala interval (Koo dan Li, 2016). ICC banyak digunakan dalam berbagai bidang seperti penelitian eksperimental, psikometri(Niggli dkk., 2021; Nyman dkk., 2023). ICC mengukur variasi skor yang berasal dari perbedaan antar subjek dibandingkan dengan variasi keseluruhan dan juga mencakup kesalahan pengukuran atau perbedaan antar penilai. Terdapat beberapa jenis ICC yang umum digunakan bergantung pada model pengukuran dan jenis kesepakatan yang dinilai antara lain: ICC(1,1) untuk one-way random effects dengan satu penilai, ICC(2,1) untuk two-way random effects dengan satu penilai dan penilaian absolute agreement, ICC(2,k) untuk rata- rata dari k penilai pada *model two-way random effects*, ICC(3,1) untuk *two-way mixed effects* dengan satu

penilai dan *absolute agreement*, serta ICC(3,k) untuk rata-rata dari k penilai pada model *two-way mixed effects* dan *absolute agreement*. Dalam penelitian ini digunakan ICC(3,k), yang berarti model *two-way mixed effects* dengan penilai tetap (fixed raters), menggunakan rata-rata dari k penilai, serta menilai kesesuaian absolut (*absolute agreement*) antar penilai. Rumus estimasi ICC(3,k) dinyatakan sebagai berikut yang dapat dilihat dari persamaan 2.18:

$$ICC\ (3,k)\ =\ \frac{BMS - EMS}{BMS} \tag{2.18}$$

Keterangan :

BMS : $nilai\ rerata\ kuadrat\ antar\ subjek$

EMS : $nilai\ rerata\ kuadrat\ kesalahan\ (residual)$

k : banyak penilai (rater)

Rumus ini menunjukkan bahwa semakin besar variabilitas antar subjek dibandingkan variabilitas dalam-subjek (kesalahan penilaian), maka nilai ICC akan mendekati 1, yang menandakan reliabilitas yang tinggi. Koo dan Li, (2016) memiliki aturan atau intreprestasi hasil nilai ICC yang dijelaskan pada Tabel 2.2. Penggunaan ICC dalam penelitian sangat penting untuk memastikan bahwa penilaian yang dilakukan oleh manusia memiliki konsistensi tinggi, sehingga hasil evaluasi terhadap sistem seperti sistem *feedback generation* otomatis dapat dipercaya. Dalam konteks ini, ICC berfungsi sebagai indikator statistik untuk menilai apakah hasil sistem mendekati penilaian manusia atau tidak.

Tabel 2.2 Intrepretation Rules ICC (Koo dan Li, 2016)

ICC	*Reliability*
< 0.5	*Poor*
0.5 – 0.75	*Moderate*
0.75 – 0.9	*Good*
>0.9	*Excellent*